Cloud Security for Beginners

Exploring the foundations

Sasa Kovacevic



www.bpbonline.com

First Edition 2025

Copyright © BPB Publications, India

ISBN: 978-93-65896-206

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

To View Complete
BPB Publications Catalogue
Scan the QR Code:

www.bpbonline.com

Dedicated to

*To my friends **Julijana** (aka Julie), **Goran** (aka Smile), and **Amir** (aka an absolute legend) for keeping me sane and for all the fun times we have. To my friend **Kreso** (and his extended family) for all the fun times we have had and all the life lessons we learned together. To my mother, **Mirjana**, for all the sacrifices she made so I can be the man I am today. To all my friends and colleagues at Microsoft, Amazon, Google, and elsewhere for keeping the cloud exciting. To everyone in Ireland for making me feel as much at home here as I was in Croatia.*

About the Author

Sasa Kovacevic is a digital technology strategist and a cloud architect who has worked with top cloud computing customers across all three hyperscale clouds to deliver solutions, especially security solutions, spanning the globe. It is highly likely you are using services, technologies, and applications every day of your life where he has had direct input or has added value in some way. Workloads of all sorts have passed through his fingers - from law enforcement, defense, banking, insurance, telecommunications, logistics, and more. The result of all this experience is also in this book, available for you and your organization to learn from. Sasa has been there for the origins of cloud computing, has been there for all the major security events since then, and has been helping the industry survive them.

About the Reviewer

Sheetal Agarwal has completed her B.Tech (CSE) and M.Tech (CSE) from Teerthanker Mahaveer University, Moradabad, India. She is an AWS Community Builder. With proficient experience in multi-cloud infrastructure and DevOps automation, she specializes in transforming business operations through cutting-edge cloud technologies and automation. She holds certifications including AWS Certified DevOps Engineer Professional, AWS Certified Solutions Architect Associate, AWS Certified Cloud Practitioner, Google Cloud Certified Professional Cloud Security Engineer, Google Cloud Certified Professional Cloud Architect, Google Cloud Certified Professional DevOps Engineer, Google Cloud Certified Associate Cloud Engineer, GitHub Foundations, and others. She has authored research papers in various Scopus-indexed Conference proceedings and peer-reviewed journals indexed in Web of Science. Her areas of interest are Cloud Computing, DevOps, Big Data, Cloud Security, Knowledge Graphs, and Machine Learning. She is a technical writer and has published various technical articles and blogs. She is currently working as a DevOps Engineer at Twimbit Research India Pvt Ltd.

Acknowledgement

I would like to express my sincere gratitude to all those who contributed to the completion of this book.

First and foremost, I extend my heartfelt appreciation to my family and friends for their unwavering support and encouragement throughout this journey. Their love and encouragement have been a constant source of motivation.

I am immensely grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. Their support and assistance were invaluable in navigating the complexities of the publishing process. If it wasn't for the wonderful folks always engaging with me, this book would never see the light of day.

I would also like to acknowledge the reviewers, technical experts, and editors who provided valuable feedback and contributed to the refinement of this manuscript. Their insights and suggestions have significantly enhanced the quality of the book.

Last but not least, I want to express my gratitude to the readers who have shown interest in our book. Your support and encouragement have been deeply appreciated.

Thank you to everyone who has played a part in making this book a reality.

Preface

Understanding the basics of cloud security is important in today's rapidly evolving world. This book covers the essential concepts that form the backbone of cloud security.

Comprising eighteen insightful chapters, this book covers a wide range of topics essential for understanding the intricacies of cloud security.

We start with the key concepts. We then guide the reader through their responsibilities, and the responsibilities of their peers in their organization. Securing the cloud infrastructure is next, then networking, apps and data. Then, we go into cloud security and governance. Authentication, authorization, data privacy, and compliance are next. Securing APIs is next. Then, we go into the specifics of securing virtual machines, containers, functions, networking, and storage. Automation and incident response are next. As is the forensics. We close off with DevSecOps, platform engineering, and site reliability engineering practices before we close with machine learning, artificial intelligence, and the future of cloud security.

Chapter 1: Cloud Security, Key Concepts - Cloud security involves protecting data, applications, and infrastructure hosted in the cloud from cyber threats and vulnerabilities. It combines technical controls, governance, and compliance measures to mitigate risks across public, private, and hybrid environments.

Chapter 2: Service Models and Deployment Models - Understanding service models like IaaS, PaaS, and SaaS is critical as each introduces unique security challenges and responsibilities. Deployment models—public, private, hybrid, and multi-cloud—affect risk exposure and dictate tailored security strategies.

Chapter 3: Shared Responsibility and Supply Chain - The shared responsibility model defines the split in security roles between cloud providers and customers, requiring clear boundaries and adherence to best practices. Securing the supply chain involves evaluating third-party vendors and ensuring the integrity of software dependencies.

Chapter 4: Securing Cloud Infrastructure and Identity and Access Management - Strong identity and access management (IAM) practices prevent unauthorized access to cloud resources, enforcing the principle of least privilege. Securing cloud infrastructure includes robust configuration management, monitoring, and leveraging proper tools.

Chapter 5: Network Security - Cloud network security involves securing virtual networks through segmentation, firewalls, and encryption to mitigate unauthorized access and data breaches. Threat detection systems such as intrusion detection and prevention systems (IDPS) bolster defense against attacks.

Chapter 6: Securing Applications and Data - Securing applications requires secure coding practices, regular vulnerability assessments, and patching against threats. Data protection involves encryption, access controls, and backups to ensure confidentiality, integrity, and availability.

Chapter 7: Cloud Security and Governance - Effective governance aligns security policies and procedures with organizational goals and compliance requirements. Regular audits and policy enforcement ensure adherence to frameworks like ISO 27001 and SOC 2.

Chapter 8: Authentication, Authorization, Data Privacy, and Compliance - Robust authentication mechanisms, such as multi-factor authentication (MFA), strengthen identity verification. Compliance with privacy regulations like GDPR and HIPAA involves securing data collection, storage, and sharing processes.

Chapter 9: Securing APIs, Observability, and Incident Response - API security ensures data integrity and prevents exploitation by attackers through validation, authentication, and rate limiting. Observability aids in identifying vulnerabilities, while incident response minimizes damage and speeds up recovery.

Chapter 10: Virtual Machines and Containers - Virtual machines (VMs) require patching and monitoring to prevent hypervisor-level attacks, while container security focuses on image scanning, runtime protection, and secure orchestration. Both need effective isolation to limit risk.

Chapter 11: Serverless - Serverless architecture introduces unique security concerns like securing event triggers and managing function permissions. Adopting a zero-trust approach minimizes potential exposure to attacks.

Chapter 12: Networks and Storage - Securing cloud networks involves encryption of data in transit and protecting endpoints against eavesdropping. Cloud storage security emphasizes access controls, encryption at rest, and consistent backup strategies.

Chapter 13: Protecting Workloads through Automation and Threat Intelligence - Automation streamlines workload protection through tools like CI/CD pipelines and automated patching. Integrating threat intelligence enables proactive defense against emerging cyber threats.

Chapter 14: Incident Response, Forensics, Security Assessment, and Penetration Testing - A robust incident response plan ensures swift containment and recovery from security events. Regular assessments and penetration testing identify vulnerabilities and improve overall resilience.

Chapter 15: Compliance and Auditing - Cloud compliance requires adherence to industry standards, regulatory requirements, and internal policies. Auditing processes validate compliance efforts and provide actionable insights for continuous improvement.

Chapter 16: DevSecOps, Platform Engineering, and Site Reliability Engineering - DevSecOps integrates security into development workflows, ensuring early detection and mitigation of vulnerabilities. Platform engineering and SRE prioritize reliability and scalability while embedding security measures throughout the lifecycle.

Chapter 17: Machine Learning and Artificial Intelligence - Machine learning enhances threat detection and response by identifying anomalous patterns in real-time. AI-driven tools enable predictive analytics, strengthening proactive defense strategies.

Chapter 18: Future of Cloud Security - The future of cloud security will focus on zero-trust architecture, quantum-resistant encryption, and AI-driven automation. Evolving threats will demand continuous innovation in securing cloud environments and adapting to new technologies.

Coloured Images

Please follow the link to download the
Coloured Images of the book:

https://rebrand.ly/a6326d

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

business@bpbonline.com for more details.

At **www.bpbonline.com**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Table of Contents

1. Cloud Security, Key Concepts 1
Introduction 1
Structure 1
Objectives 2
Cloud and cloud computing 2
Community definition 2
Other computing options 3
Cloud computing advantages 4
Reliability 5
Cost optimization 5
Operational excellence 5
Performance efficiency 6
Service models 6
Addition to the definition 7
Security 8
Cybersecurity 8
Confidentiality 10
Integrity 10
Availability 10
Azure, Amazon Web Services, Google Cloud Platform 10
Authentication 11
Authorization 11
Cloud security 12
Why cybersecurity now 16
Why is cybersecurity for you? 17
Conclusion 18
Questions 18
References 19

2. Service Models and Deployment Models 21
Introduction 21

Structure ... 21
Objectives ... 21
Service models ... 22
Self-service ... 22
Resource pooling ... 23
Elasticity ... 24
Network access ... 25
Pay-as-you-go ... 25
Virtualization ... 27
High availability ... 28
Privacy ... 30
Data transfer ... 32
Marketplace ... 33
IaaS vs. PaaS vs. SaaS briefly ... 34
Infrastructure as a Service ... 35
Platform as a Service ... 36
Software as a Service ... 38
Deployment models ... 40
Public cloud ... 40
Private cloud ... 40
Hybrid cloud ... 41
Multi cloud ... 42
Conclusion ... 43
Questions ... 43
References ... 44

3. Shared Responsibility and Supply Chain ... 45
Introduction ... 45
Structure ... 46
Objectives ... 46
Trust ... 46
Definition of trust ... 46
Why is trust good? ... 47
Why is trust bad? ... 47

Zero trust *48*
Positives and negatives of trust *49*
Shared responsibility *51*
Supply chain 54
Vendor selection *55*
Risk management *56*
Cybersecurity best practices *57*
Data protection and privacy *57*
Third-party due diligence *57*
Physical security *58*
Transparency *58*
Incident response and contingency planning *58*
Compliance *59*
Employee training *59*
Real-world examples *60*
Software development 61
Real-world examples *62*
Software deployment 63
Real-world examples *63*
Conclusion 64
Questions 64
References 65

4. Securing Cloud Infrastructure and Identity and Access Management 67
Introduction 67
Structure 68
Objectives 68
Securing cloud infrastructure 68
Business continuity 69
Encryption 70
Operational security 72
Data sanitization and anonymization 75
Data masking 76
Equipment disposal 76

Understanding IAM and RBAC 77
Principle of least privilege 78
Zero trust *79*
Google Cloud Platform IAM 79
Azure Privileged Identity Management 83
Conclusion 90
Questions 90
References 91

5. Network Security 93
Introduction 93
Structure 93
Objectives 94
History 94
Precursors *94*
Global reach *96*
The issues *97*
Open Systems Interconnection 98
OSI layers *99*
OSI layer security *100*
Azure services across OSI layers *102*
Cloud technologies that work across layers *103*
Secure Sockets Layer/Transport Layer Security *104*
Handshake *104*
Certificate exchange *104*
Key exchange *104*
Data encryption *105*
Data integrity *105*
Networking patterns 105
Direct connection *105*
DNS *106*
Bastion *108*
DDoS protection *108*
Private link *108*

Subnets *108*
Conclusion 111
Questions 112
References 112

6. Securing Applications and Data 113
Introduction 113
Structure 114
Objectives *114*
Scenario *114*
Architecture *115*
Access patterns *116*
Frontend 118
App service *118*
Backend *123*
Management and security services 128
Application/service itself 131
Conclusion *132*
Questions 132
References 133

7. Cloud Security and Governance 135
Introduction 135
Structure 136
Objectives 136
General principles of governance 136
Identity *136*
Patterns *136*
Common antipatterns *137*
Layers *138*
Common antipatterns *138*
Encryption *139*
Patterns *139*
Common antipatterns *139*
Automation *140*

Patterns *140*
Common antipatterns *140*
Monitoring *141*
Common antipatterns *141*
Response *142*
Patterns *142*
Common antipatterns *143*
Governance 143
Managing risk *144*
Threat modeling 146
What are we working on? *150*
What can go wrong? *151*
What are we going to do about it? *151*
Did we do a good enough job? *152*
Microsoft SDL practices 152
Conclusion 153
Questions 153
References 154

8. Authentication, Authorization, Data Privacy, and Compliance 157
Introduction 157
Structure 158
Objectives 158
Authentication 158
Patterns *158*
Anti-patterns 159
Authorization 160
Patterns *160*
Anti-patterns *161*
Data privacy 162
Patterns *162*
Anti-patterns *163*
Compliance 163
Patterns *164*

Anti-patterns *164*
General Data Protection Regulation 165
Patterns *166*
Anti-patterns *167*
System and Organization Control 2 168
Patterns *172*
Anti-patterns *172*
Conclusion 173
Questions 173
References 174

9. Securing APIs, Observability, and Incident Response 175
Introduction 175
Structure 175
Objectives 176
Securing APIs 176
Authentication *177*
Authorization *178*
Input validation *178*
Parameterized queries *178*
Secure communication *178*
Rate limiting *179*
Audit logs *179*
Token management *179*
Error handling *179*
Cross-Origin Resource Sharing *180*
Content Security Policy *181*
API versioning *181*
Security headers *181*
Data encryption *181*
Dependency scanning *182*
Security testing *182*
Denial-of-Service protection *182*
API gateway security *183*

Secure file uploads *183*
Education and training *183*
Purple teaming APIs *183*
Observability *184*
Observability *184*
Defining clear objectives *186*
Instrumentation *186*
Centralized logging *187*
Metrics collection *187*
Distributed tracing *187*
Real-time monitoring *187*
Alerting mechanisms *188*
Auto-scaling and load balancing *188*
Security monitoring *189*
Dependency mapping *189*
Cloud-native tools *189*
Cost monitoring *189*
Collaboration and communication *189*
Continuous improvement *190*
Documentation *190*
User experience monitoring *190*
Compliance monitoring *190*
Education and training *191*
Incident response plan *191*
Incident response 192
Understanding your assets *192*
Risk assessment *192*
Create an Incident Response Team *193*
Communication plan *193*
Training and drills *193*
Incident identification *194*
Incident categorization *194*
Escalation procedures *194*
Documentation

Legal and compliance considerations *194*
Incident response playbooks *195*
Incident reporting *195*
Evidence handling *196*
Forensic readiness *196*
Collaboration with external entities *197*
Continuous improvement *197*
Incident analysis *197*
Incident containment *198*
Public relations strategy *198*
Vendor management *199*
Conclusion 200
Questions 200
References 201

10. Virtual Machines and Containers 203
Introduction 203
Structure 203
Objectives 204
Similarities 204
Differences 205
Virtualization 206
Virtual machines 206
AWS VM *208*
Containers 211
AWS containers *213*
Orchestration *216*
Migration 217
On-premises to VMs *217*
On-premises to containers *218*
Conclusion 219
Questions 220
References 220

11. Serverless 221
Introduction 221
Structure 222
Objectives 222
Introduction to serverless 222
Serverless services 224
Cloud Functions 225
Security 233
Conclusion 237
Questions 237
References 238
12. Networks and Storage 239
Introduction 239
Structure 240
Objectives 240
Virtual network concepts 240
Address space 240
Subnets 241
Virtual network best practices 242
No overlapping 242
Plan ahead 242
Fewer is better 242
Subnet security 245
Artificial intelligence 245
Planning a network 245
Storage concepts 246
Key questions 246
Storage capabilities 247
Redundancy and availability 249
Short term storage options 250
Caching 253
Conclusion 255
Questions 255
References

13. Protecting Workloads through Automation and Threat Intelligence 257
Introduction 257
Structure 257
Objectives 258
Automation 258
CI/CD 258
Continuous integration patterns 259
Continuous development patterns 260
Azure DevOps 261
Threat intelligence 268
Threat modeling 269
Conclusion 271
Questions 272
References 272

14. Incident Response, Forensics, Security Assessment, and Penetration Testing 273
Introduction 273
Structure 274
Objectives 274
Incident response 274
Incident response lifecycle 275
Preparation 275
Identification 276
Containment 276
Eradication 277
Recovery 278
Lessons learned 278
Incident response challenges 279
Forensics 280
Cloud forensics techniques 280
Log analysis 280
Memory acquisition 280
Virtual machine introspection 281
Evidence collection and preservation 281

Collaboration with cloud providers *282*
Benefits *282*
Data volatility *283*
Security assessment 283
Cloud Controls Matrix *284*
Shared responsibility model *284*
Challenges of vulnerability management in the cloud *284*
Best practices *285*
Penetration testing 286
Importance of threat modeling 286
Penetration testing as an assessment tool *287*
Reporting *288*
Lessons learned from a penetration test *288*
Tools *289*
Conclusion *290*
Questions 290
References 291

15. Compliance and Auditing 293
Introduction 293
Structure 293
Objectives 294
Introduction to compliance 294
Legal and financial protection *294*
Enhanced security and risk management *295*
Reputation and trust *295*
Frameworks and standards *295*
Regulatory bodies *295*
Common standards and regulations *296*
Industry-specific compliance requirements *297*
Balancing compliance and security 297
How compliance enhances security *298*
The concept of more secure *298*
The concept of best-effort security *299*

Auditing events *300*
Scope *300*
Events *300*
Tools *301*
Patterns *302*
3rd party auditing 302
Purpose *303*
Types of audits *303*
Preparation *304*
Challenges *304*
Cloud providers compliance portals 305
Azure *305*
AWS *306*
GCP *306*
Conclusion 307
Questions 307
References 308

16. DevSecOps, Platform Engineering, and Site Reliability Engineering 309
Introduction 309
Structure 309
Objectives 309
Cloud security 310
DevOps *310*
DevSecOps *311*
Platform Engineering *311*
Site Reliability Engineering *312*
Similarities and differences *313*
Patterns and anti-patterns *313*
DevOps *313*
DevSecOps *315*
Platform Engineering *318*
Site Reliability Engineering *320*
Use cases 323

DevOps *323*
DevSecOps *323*
Platform Engineering *324*
Site Reliability Engineering *324*
Conclusion *324*
Questions 325
References 326

17. Machine Learning and Artificial Intelligence 327
Introduction 327
Structure 327
Objectives 328
Machine learning versus artificial intelligence 328
Machine learning *328*
Artificial intelligence *329*
Examples of ML that people confuse for AI *329*
Examples of AI people confuse for ML *330*
The recent prominence of AI 331
Pros of AI *331*
Cons of AI *333*
C-suite understanding of ML and AI 333
Chief Executive Officer *334*
Chief Financial Officer *334*
Chief Technical Office *334*
Chief Information Officer *335*
Cloud security 335
ML security issues *335*
AI security issues *336*
Managing ML and AI workloads 337
AIOps *339*
ML/AI security *340*
Conclusion 342
Questions 343
References 343

18. Future of Cloud Security 345
Introduction 345
Structure 345
Objectives 346
Patterns in the emerging technologies 346
Zero trust 346
Confidential computing 347
ML/AI for threat detection 348
Quantum-safe cryptography 348
Serverless, event-driven code 349
Distributed Ledger Technology for security 350
Future cloud security issues 351
Data breaches 351
Misconfigurations 352
Insider threats 352
Insecure APIs 353
Compromised accounts 353
Lack of visibility and control 353
Compliance challenges 354
Distributed denial of service attacks 354
Zero-day vulnerabilities 354
Be ready for anything 355
Large Language Models 356
Managing data influx 357
Tools sprawl 358
Top tips 359
Conclusion 360
Question 361
References 361

Index 363-371

Chapter 1
Cloud Security, Key Concepts

Introduction

This first chapter of the book will give you an introduction to the cloud, security, and security in general. We will delve into the definition and significance of cloud technology, the widespread presence of individuals in the cloud, and the inescapable nature of its existence.

The key takeaways from this chapter will be an understanding of the cloud, its benefits, and why cloud security and cybersecurity are important. The chapter will also introduce the hyper-scale cloud providers - Microsoft, Amazon, and Google and their respective platforms, Azure, AWS, and GCP.

We will also discuss why the reader is in the best place in time with this book and how the concepts in this book will reflect in their daily lives and their work.

Structure

This chapter will cover the following topics:

- Cloud and cloud computing
- Security
- Cybersecurity

- Azure, Amazon Web Services, Google Cloud Platform
- Cloud security

Objectives

By the end of this chapter, you will be confident about the baseline definition of some security related terms, and the application of what you will learn by reading this book to your organization's cloud provider of choice.

You will also learn why the current time is best suited for cloud security, and why you should start getting more involved with cloud security. Reading this book can set you up for a long and successful career in security, or a long and prosperous career in delivering cloud services to your customers while being aware of all security considerations and implications of your architectural decisions.

Cloud and cloud computing

You might already be familiar with what cloud computing is. However, let us spend some time discussing its definition so that we are on the same page.

So, what is the cloud? And what is cloud computing?

We will first look at the community-accepted answer and then the definition provided by this book as an addition to this definition.

Community definition

One common definition of cloud computing may be around the delivery of computing services over the internet. Instead of provisioning and managing your own local servers and/or personal devices to store data and run applications, cloud computing allows you to access and use a network of remote servers and services hosted on the internet and managed by a cloud provider of your choice (or the preferred management choice in your organization).

Cloud computing is renting underlying infrastructure for a little while - a second, a minute, an hour, a day, a week, a month, a quarter, a year, or a multi-year period. All these time periods are available for the users.

You can rent a server like you normally would from another hosting provider. You would normally rent servers for longer periods (not always though) – months, years, or more. This practice primarily applies to your baseline servers, the ones consistently employed to deliver your personal or organizational services.

The cloud computing paradigm allows you to change your mind at any time unless you have reserved the capacity well in advance and gotten a better price because of it. The following figure details the benefits of cloud computing:

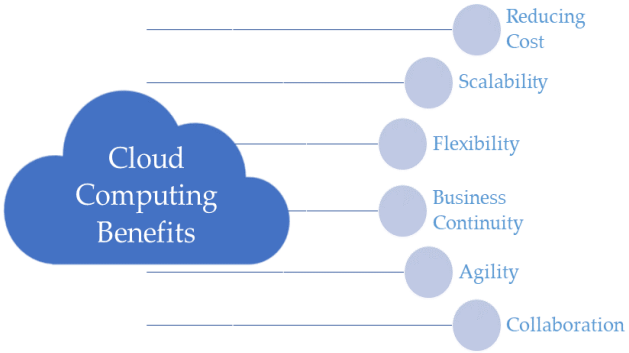


***Figure 1.1**: Cloud computing benefits:*

Other computing options

However, in the cloud, there are other computing options other than servers. There are containers, for example, that are a smaller unit of computing than servers, usually faster to start/stop and easier (if done right) to scale.

There are also functions (for example, Azure Functions, AWS Lambda, GCP Cloud Functions) that are even smaller units of computing than servers or containers, allowing you to run your business logic that requires computing on a per millisecond basis. A usual example of this is a service that creates a thumbnail every time a user uploads a large image. In this service, a function triggers an upload of the larger file, runs for a few milliseconds, and creates a thumbnail (a smaller image) of the original image. Since you only usually need to do this once per image and you need to do this only as many times as new images are uploaded, the function code runs, creates a thumbnail, and then exits. The wonderful thing about this function is that it can be called for each of the images independently, start, and conclude independently.

Here is an example in pseudo-code how this function would work:

```
[Trigger: OnImageUploadToStorage]
Function CreateThumbnail()
{
      // get the path and name for the original image from the trigger
      // create a thumbnail from the original image
      // save the thumbnail to storage
}.
```

The triggering mechanism is provided to the function in a specific way that the provider defines – based on the cloud platform. It is also possible to extend and define these triggers – again based on the cloud platform.

There are three general types of cloud computing providers (relevant for this book):

- Hyperscale cloud providers
- Local or niche cloud providers
- Partners / resellers

In the first category there are only three providers available: Azure (from Microsoft), **Amazon Web Services** (**AWS**) from Amazon, and **Google Cloud Platform** (**GCP**) from Google. These are the only global players in this space. Hyperscale providers are characterized by their ability to provide a huge array of cloud services anywhere in the world, and to organizations of all sizes.

In the second category are all other cloud providers. For example – IBM, Oracle, Alibaba, Huawei, and others.

And lastly, the third category is reserved for partners and resellers of categories one and two. They will depend on the cloud providers and services from categories one and two but will complement this service with a white glove approach for organizations of all sizes.

You should be able to deliver almost all your workloads with the help of any of the players in each category, so from that point of view they are similar. However, for truly global, cutting-edge, and hyperscale delivery, you will usually depend on category one (and the focus of this book, especially in the examples as these providers are known and ubiquitous).

Cloud computing advantages

Cloud computing offers several key advantages (according to Microsoft Azure – learn more here: **https://learn.microsoft.com/en-us/azure/well-architected/**):

- Reliability
- Security
- Cost optimization
- Operational excellence
- Performance efficiency

Note that these are from something called a well-architected framework, meaning while the cloud provider enables these benefits you still need to do work and architect your workloads well. The good news is that these benefits are achievable more easily than if you were just doing computing on your own in your own data center.

Of course, the focus of this book will be on security, but let us quickly explain reliability, cost optimization, operational excellence, performance efficiency, and their general impact on security. The following figure shows the well-architected framework of Azure:

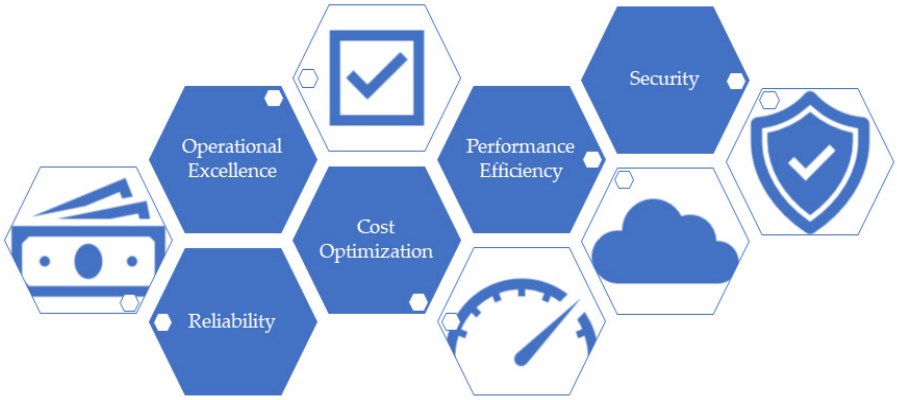


Figure 1.2: *Well-architected framework (Azure)*

Reliability

Reliability means that a workload is both resilient and available. A resilient workload is a workload that performs equally well when under heavy load and when underlying services may have issues, networking latency for example (network latency is a time delay that occurs due to data packet travel from one location in the world to another, which while usually consistent may suffer due to network issues). Available workload is a workload that is consistently available. This is usually achieved by deploying a service across physical and logical boundaries. For example, in two different data centers that are separated by a 400km distance, or by deploying a service into two different network topologies that are independently supplied with electricity and connectivity to the internet. The more reliability you have the more cost you should be prepared to absorb, and securing services across physical and logical boundaries is more costly as well.

Cost optimization

Cost optimization refers to delivering a service with minimal costs (sometimes referred to as marginal costs, in other words, the cost to deliver a service per customer or a service call). The optimization part kicks in when continually reviewing and improving the cost model for the organization's workflows. The utopia is delivering services for a cost of 0 (zero). Believe it or not, due to the free tiers available for most services it is possible to deliver some services at a cost of 0. Usually, however, security services are delivered at static costs (or as an added cost per service call) so there will always be a cost for security services on top, for example, protection from **Distributed Denial of Service** (**DDoS**) attacks.

Operational excellence

Operational excellence refers to operations and processes that keep the application or service running in production (live, real-world environment available to others that

depend on it – internally or externally). A large part of operational excellence is the ability to quickly deliver patches or fixes (again, truly relevant to security as you need to deliver security patches as quickly as possible and as widely as possible) as well as roll back (revert to a previous deployment because there is an issue with the latest one) any service in the event when problems are detected. Security needs to be baked in (in other words, considered together with) these processes and everyone throughout these processes must be enabled to act in the best interest of security and empowered to speak up when any security issue is discovered and not addressed.

Performance efficiency

Performance efficiency is the ability of a workload to scale and deliver the service in an efficient manner. For example, performing an internet search must return results quickly and in milliseconds rather than in seconds or minutes – for the simple reason that we are not willing to wait for a long time to get the results back. The security impact of this feature on cloud computing services is subtle. For example, if on every call to your service, we must also check the origin of the call, the credentials (username/password or an API key) of the call, and the maliciousness of the call itself, this may, in fact, impact performance. It usually does so only by a little (as measured in milliseconds), but cumulatively (counted all together), these calls matter to performance. This is not the reason to neglect security, but a call to action to add security and plan services in such a way that this impact is mitigated.

Service models

Cloud computing is utilized on top of several service models.

- **Infrastructure as a service:** Virtualized computing resources such as virtual machines, storage, and networking allow you and your organization to build and manage your own IT infrastructure.
- **Platform as a service:** A platform for developers to build, deploy, and manage applications and services without worrying (too much) about the underlying infrastructure.
- **Software as a service:** Delivers software applications and services over the internet, eliminating the need to install and manage them – usually accessing them through a web browser or thin clients.

Refer to the following figure for an illustrated understanding of these service models:

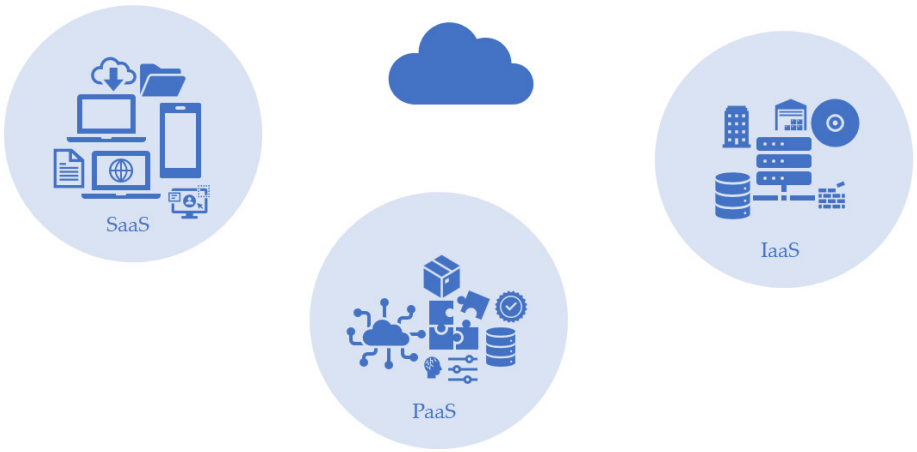


Figure 1.3: *Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS)*

We will discuss these models further in the next chapter.

In conclusion, cloud computing allows you to rent infrastructure quickly, easily, and on-demand without the burden of managing physical infrastructure yourself.

Addition to the definition

Addition to the above is something that is overlooked easily when discussing cloud computing and is especially relevant to security. Let us delve deeper into this.

Running services on rented infrastructure is all well and good but without the consideration of how those services are performing for the business, in other words, are they delivering the value to the business that the business expects, there can be no true adoption of the cloud model.

Therefore, another big reason for adopting the cloud model is agility. It is the most crucial reason for adopting the cloud model.

In today's ever-changing world of digital transformation and market disruption, delivering services quickly and with a focus on business goals is crucial. Nothing can stand in the way of that, or you and your organization risk being made obsolete by a competitor service.

Every organization wants to move quickly. Delivering security quickly is almost an oxymoron. Delivering security must be done in a considered, planned, and comprehensive way. It is therefore essential that security is not an after-thought or an add-on to existing services, but that security is a part of all processes, agility, software delivery, compliance, etc.

These are the core benefits of the cloud, all the other previously mentioned ones are great, but they are secondary and derived from desired business outcomes and agility in delivering them. The following figure illustrates agility in detail:



Figure 1.4: *Agility*

Security

This is a book about cloud security, but what is security?

The concept of security refers to the state or condition of being protected from harm and risks. It involves implementing measures to mitigate risk and manage security by introducing safeguards.

Security aims to prevent unauthorized access, breaches, attacks, and other forms of harm. It aims to deliver safety and protection.

Cybersecurity

Cybersecurity, also referred to as information security, is a branch of overall security that focuses on protecting digital information, systems, applications, and services from unauthorized access, disruption, theft, or damage (sometimes also shortened to - protection from hackers which isn't 100% true of course).

It further involves a set of practices, technologies, and processes designed to safeguard the infrastructure (networks, devices, appliances, applications, services, and data) from such threats as malware, ransomware, phishing, etc. The following figure details some aspects of cybersecurity:



Figure 1.5: *Cloud security*

Important concepts mentioned in *Figure 1.6* are (also referred to as the CIA triad):

- Confidentiality
- Integrity
- Availability

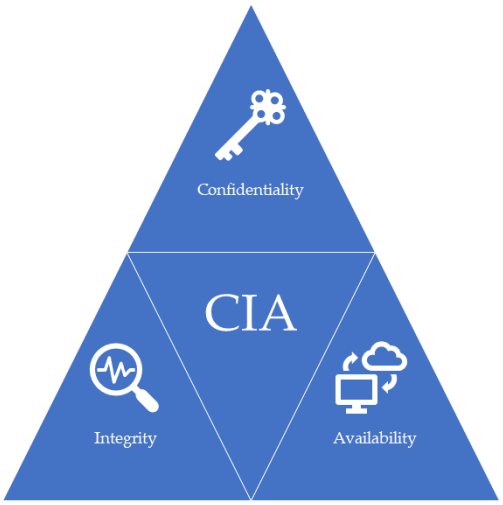


Figure 1.6*: CIA triad*

Confidentiality

Confidentiality refers to the notion that information is accessible only to those who are authorized to access it. Otherwise, if everyone has access or if some unauthorized individuals have access, there is no confidentiality, and the information is essentially public. Preventing unauthorized access, wiretapping or data disclosure is the concern of confidentiality. Countermeasures will include access controls, encryption, and data classification (data classification is important to decide what measures if any need to be employed to protect certain kinds of data).

Integrity

Integrity refers to the notion that information is not tampered with and that it remains accurate, complete, and unaltered. Preventing unauthorized modification, tampering, or data corruption is done to preserve integrity. Techniques include:

- **Data validation:** Ensuring data is readable and correct.
- **Checksums:** Ensuring data has not changed by generating a unique ID at different stages of data use and verifying it against the previous ones. Checksum functions are very sensitive to data change and a very minimal change results in a completely different checksum.
- **Digital signatures:** To verify people or processes that changed the data.
- **Version control:** To have an audit tree of all changes to data, including the changes themselves but also who changed it, when, from which location, through which process, who authorized the change, etc.

Availability

Availability is the notion that authorized people and processes do indeed have access whenever they need it (after all, data that cannot be accessed when needed is useless). Ensuring resources, services, and data are available is done through redundancy, fault tolerance (ensuring the continuing functioning of a system when faced with faults or failures), disaster recovery planning (preparing for those situations that require intervention to keep the availability at the expected levels), and above all – observability, which is the ability to see in real-time the key performance indicators and acting on them when they become unhealthy or better yet before they become unhealthy.

Azure, Amazon Web Services, Google Cloud Platform

Topics in this book can and must be applied to all workloads regardless of where they are located. As this book is focused on cloud security, we will focus on the three hyperscale cloud providers – Azure, AWS, and GCP.

Security concepts apply equally to all of them, but implementation will differ.

Let us learn this on the concepts of authentication and authorization.

Authentication and authorization are two essential aspects of managing access to resources in cloud computing environments. While the general concepts remain similar across Azure, AWS, and GCP, there are some differences in how these providers handle authentication and authorization best practices. Here is an overview of these key differences.

Authentication

Authentication verifies the identity of users or entities trying to access a system or resource. It ensures that the user is who they claim to be. In Azure, AWS, and GCP, authentication can be achieved through various methods:

- Azure provides **Azure Active Directory** (**Azure AD**) as its **identity and access management** (**IAM**) service. Azure AD supports various authentication methods, such as username and password, **multi-factor authentication** (**MFA**), Azure AD B2C for external user authentication, and integration with on-premises Active Directory.
- AWS offers AWS IAM as its authentication and access control service. IAM allows users to manage access to AWS services and resources through authentication methods like AWS access keys, IAM users and groups, federation with **External Identity Providers** (**IDPS**), and MFA.
- Google Cloud IAM is used for authentication and access control in GCP. GCP IAM supports several authentication options, including username and password, service accounts, OAuth 2.0, and third-party identity providers like Google Sign-In and OpenID Connect.

So, while the concepts of security around authentication are universal, there is a nuance on how to apply them to services from each cloud provider. In this book, we will focus on the universal concepts with an occasional specific example in a particular cloud.

Authentication concepts of IAM, usernames, passwords, MFA, external access, and integration, are security concepts you must consider. This is the focus of this book.

Let us continue with authorization.

Authorization

Authorization determines the permissions and privileges granted to authenticated users or entities and governs what actions they can perform on resources.

Here are the differences in authorization best practices between Azure, AWS, and GCP:

- Azure provides **Role-Based Access Control** (**RBAC**) for authorization. RBAC

to users, groups, or service principals, defining their permissions at various levels, such as subscription, resource group, or individual resources. Azure also supports custom RBAC roles for fine-grained access control.

- AWS IAM provides granular access control using policies. Policies are JSON documents that define permissions and are attached to IAM users, groups, or roles. AWS IAM policies use a resource-based authorization model and allow you to specify fine-grained permissions at the level of individual AWS services and resources.
- GCP IAM employs a similar approach to AWS IAM, where you can define permissions using IAM policies. IAM policies are JSON documents that specify who (identity) has what access (role) to which resources. GCP IAM allows you to assign predefined roles or create custom roles to manage access to GCP resources.

While the concepts of authentication and authorization remain consistent across Azure, AWS, and GCP, the specific services and terminology used may differ. It is important to consult the documentation and resources provided by each cloud service provider to understand and implement the recommended best practices for authentication and authorization within their respective environments.

How to achieve each of these within your cloud provider's range of services is beyond the scope of this book. Luckily though, once you understand the concepts, the documentation provided by each cloud provider should be an easy read for you.

Cloud security

After quite a bit of level-setting, we are finally here. Cloud security is our next topic.

What is cloud security?

Cloud security is the notion of protecting data, applications, services, and infrastructure within a cloud computing environment. As you or your organization adopt the cloud model and all the cloud services, ensuring the security of cloud-based assets becomes essential, of course.

Cloud security involves primarily:

- Implementing controls to protect the data privacy of your, your organization's users and external users.
- Prevent unauthorized access to infrastructure resources and data.
- Ensure compliance with regulations in regions where you are delivering services and in regions where your users are located, as well as any industry regulation you are doing business in (e.g., insurance).
- Mitigate cloud-specific risks in addition to general security risks, such as insider threats. You now need to focus on risks specific to the cloud, for example, a public-facing internet service.

This requires a cultural shift from the old way of delivering services to the new way of delivering cloud services with agility and at-scale.

Various aspects of cloud security now need to be managed:

- **Secure cloud architecture:** Architecting services with a cloud-first focus.
- **Identity and access management:** Authentication and authorization for internal access to be able to deliver and deploy service as well as service-to-service communication and user and third-party access.
- **Data encryption:** In transit and at rest that is, user-to-service and service-to-service communication – and when persisting data to storage.
- **Threat detection and prevention:** Modeling threats and addressing them to mitigate risks, preventing incidents, and reacting to actual incidents, small and large.
- **Security governance:** A framework of policies, processes, and procedures that an organization establishes to guide, oversee, and steer security-related activities and initiatives.

The wonderful news here is that of course no one can get everything done in one go but must go through multiple iterations. Cloud security is an ongoing process so it will continue for as long as you have workloads in the cloud.

Catch twenty-two is the term used when some security features and measures are more important than others.

For example, in data classification, it is important to know what data to focus to protect, and at what cost, but authentication and authorization are more important and should - if a choice must be made – be prioritized. Once unauthorized individuals access data it is gone forever. On the other hand, you can protect all data equally, and then once the classification processes conclude, you can relax some protections if necessary or tighten them as your threat models show or require. This is an overly simplistic case of course, but it shows that you must gain experience through careful consideration and execution on what and when to prioritize.

Cloud security aims to focus on and address the specific challenges and risks that arise from the adoption of cloud computing, in part or wholesale – that is whether the cloud is a part of your software delivery strategy or the entirety of it.

Cloud computing involves sharing physical infrastructure, such as virtual machines (servers), storage, and networking among multiple customers. The challenge is to ensure that each customer's data, applications, and services are isolated. This means that there are guarantees around the separation of shared resources, and each set of resources belonging to a customer is protected from unauthorized access by other customers (customers sharing the same resources are also referred as tenants).

To fulfill this requirement, virtualization technologies are used:

- Azure relies on Hyper-V,
- AWS relies on Xen,
- GCP relies on KVM.

Security-wise, the fulfillment of **tenant separation** is entirely the responsibility of the cloud provider. However, depending on your organization you may be running tenants of your own internally and/or externally, so knowing about tenant separation and technologies to achieve this is crucial. This is a major topic for architecture folks, but, of course, it also includes a security component. One of the most important security concerns is to keep the data separated from each tenant (remember: tenants are different customers), as not disclosing data from one tenant to another is a critical requirement.

Data residency and compliance are other important considerations of cloud security.

You may have customers all over the world and they may have legal and practical requirements around data residency. For example, you may have a Canadian customer who has a legal requirement to keep their data in Canada only; or a customer in Germany who has a legal requirement to keep their data in the European Union. Knowing these requirements is essential when architecting and securing workloads in the cloud.

Azure, AWS, and GCP all have data centers in many regions around the world.

For example, Azure has announced or opened new regions recently (at the time of writing this in July 2023) in Poland, Qatar, China, Finland, Belgium, Norway, the US (West-US-3), Malaysia, Indonesia, Georgia, Chile, Denmark, Taiwan, Austria, Greece, Italy, New Zealand, Spain, Mexico and Israel. Those are in addition to the 60+ regions already available. The number one primary reason for distributing computing capacity so far and wide is (outside of latency, which could be satisfied with a lot fewer regions) data residency requirements. Latency and edge computing are second, and the third is expanding capacity (for example, the US West region listed above). An increased number of countries and regions (for example, the European Union) have some requirements for data residency.

Cloud based services, such as serverless computing, databases, storage, container orchestration platforms, etc. introduce their own sets of security considerations. Here are some examples:

- Serverless computing involves running snippets of code in ephemeral environments, ensuring secure execution is crucial so **code integrity checks, input validation and input/output encoding** are again a huge vector open for abuse if not closed effectively, leading to a whole new (or old/existing) set of injection attacks.
- Database access control and data encryption are now more involved as (again, remember - multiple tenants may be using the same resources – including databases) databases may be shared and hence row-level access patterns or encryption may be needed.
- Storage requirements may extend beyond the business requirements and certain regulation such as the

Insurance Portability and Accounting Act (**HIPAA**) have their own requirements, such as immutability and right-to-be-forgotten. For example, what do you do with data that has been legally required to be wiped (forgotten), but you have that data in your backups?

- Container orchestration platforms are only as effective as the images (container packages/images are lightweight, standalone, executable packages and are sometimes referred to as simply - images) they spin-up (run) so image security is important in a shared environment so a compromise of one image does not escalate to others.

IAM in the cloud brings more cloud-centric controls that are required because you are now running in a shared environment. Multi-factor authentication may not have been needed on your local network due to it being isolated to your office only, but now your environment can be accessed from anywhere in the world (for example, working from home), hence MFA is an absolute requirement.

Encryption was easy in the on-premises world (if such a thing can ever be said). You kept the keys, you managed keys, and you used them for encryption and decryption. If you were an advanced organization, you used hardware devices to generate/manage keys. In the cloud, multiple services, users, and customers need encryption, and they need to communicate. So now you have options such as using the provider-generated keys, bringing your own into the cloud, and/or using the keys you keep in the cloud or keys your customers provide on-demand themselves (never to be stored on the cloud). And you are never in a one-size-fits-all scenario, so you will see all these scenarios as your cloud adoption matures. And so, you need to evolve your cloud security posture.

Only by addressing all these unique challenges and risks - you can stay on top of cloud security requirements and be a factor in securing your applications and services. The following figure sums up these features:



Figure 1.7: *Security*

Why cybersecurity now

Now you might be wondering, why we are focusing so much on security recently, and is this time special in any way that it brings to the forefront security, cybersecurity, and cloud security especially?

The good and the bad news is that: yes, this is a special time.

Yes, this is a special time, a time when cloud security is a critical skill. But at any other time in the future, cloud security is going to be an even more critical skill. Why?

Well, in Azure alone there are over two hundred different services, and they all have special security requirements on top of the general security requirements. And in the future, with things like quantum computing, machine learning, artificial intelligence, augmented and virtual realities, etc. (we will address more about the future in the last chapter) there will only be more services, not less.

Now couple that with the fact that Gartner forecasts *Worldwide Public Cloud End-User Spending to Reach Nearly $600 Billion in 2023* and that McKinsey Digital is *Projecting the global value of cloud: $3 trillion is up for grabs for companies that go beyond adoption*.

Extrapolating from these figures and the figures from previous years, the adoption of the cloud is not stopping. A global pandemic could not stem the tide of cloud adoption, the global supply chain crisis also is not stopping it.

Now is the best time to start your cloud security journey.

Cloud adoption is increasing, and organizations of all sizes are rapidly adopting cloud computing for its scalability, cost-efficiency, and flexibility as increasingly more critical systems and heavily regulated industries are making the move to the cloud. Hence, a need for robust security measures to follow the business' ambitions.

At the same time, we are living in an increasingly polarized and unstable world. More threat factors than ever before are banging on the gates of the cloud workloads trying to find any weakness they can exploit. A successful attack on you and your organization's cloud infrastructure can lead to severe consequences, such as data breaches, financial losses, fines, and worst of all reputational damage.

Just think of the consequences of a breach and gaining access to the health records of a nation, now increasingly often hosted in the cloud – consequences ranging from loss of privacy to compromised security of other services through the use of knowledge learned from this breach.

Or think of the consequences of unauthorized access to bank systems which are now increasingly often hosted in the cloud – consequences such as theft of assets on the customer side and brand reputational damage on the bank side, and other consequences.

Have I been pwned lists of some of the largest data breaches, and some of them are in the hundreds of millions of customers impacted? And as time goes on larger and larger breaches are happening.

Because of the cloud's shared responsibility model, which we will discuss later in the upcoming chapters, the responsibility for securing your workloads is shared between the cloud provider and yourselves. Providers are responsible for ensuring the security of the underlying infrastructure and you are responsible for securing your applications, services, and data. Failure to understand this model and any misunderstanding on who secures what, how, and when will result in a potential security breach. Understanding the delineation lines is important to be able to secure your workloads.

For example, while the cloud provider will work hard to attempt to prevent a DDoS attack that targets its infrastructure (including yours) they will do nothing to prevent one of your services DDoS-ing another. This can happen if one service is misconfigured (and not throttled) and it sends a million requests a second and not an expected one thousand. To you, this may be a huge issue and might bring down your multiple workloads, but to a cloud provider that is a blip they might not even notice let alone respond to.

Let us take a look at another example. While the cloud provider may prevent unauthorized access to your cloud account from a known threat actor, they might not prevent access if your account is compromised through a phishing attempt.

Hopefully these two examples showcase (for now, remember we will go more in-depth on this topic in the next chapter) the shared responsibility between you and your cloud provider.

Why is cybersecurity for you?

This is potentially the easiest part of this chapter. Do any of these reasons appeal to you?

- The demand for cloud security professionals is growing and accelerating as organizations increasingly adopt cloud and cloud technologies. There is a significant shortage of skilled professionals in this field, which creates excellent job prospects and opportunities for career advancement.
- If you like excitement (although not always the best to have too much of in a security career) and challenging yourself over and over as security threats evolve, this is the field for you.
- Found a great organization and want to stay? As a security professional, any organization will be happy to have you. Want to move around and try a lot of different things? You can do that as well. The shortage of security staff means you will be welcomed.
- Some of the strongest earners are security professionals, now neck and neck with the AI, ML, and data-related professions.
- Cloud security offers diverse career paths with opportunities to specialize in various areas, such as cloud architecture, threat intelligence, incident response, compliance, risk management, and many others (some not even named yet). You

- Organizations will happily pay for you to further your education. The more you know, the more they invest in you, the happier you will be, and the better protected they will be, so organizations will invest in your education.
- You will play a critical role in protecting your organization and will contribute directly to the overall security posture of the organization. You will also have direct metrics to show what you have done, from incidents investigated to data protection.
- Cloud security is in demand globally. You can live and go anywhere. Every continent, and every country, they need security professionals. And you can collaborate and network across the world, across borders, and across physical and logical boundaries.
- As we discussed above, your career is also future-proof. Cloud services are ever-expanding and cloud adoption is growing so you need not worry about your future, as long as you continuously learn.

Conclusion

In this chapter, you have learned a lot of the basic concepts around security and cloud security. You have learned what are the benefits of cloud computing, what are the new challenges when it comes to cloud computing and you have even learned about the basic differences between IaaS, PaaS, and SaaS.

The chapter also discussed why you are in the right place at the right time, to be starting your cloud security career or augmenting your existing cloud role with more focus on security.

In the next chapter, we will expand on the topics that you have learned here.

Questions

1. What is the Cloud and what are the benefits of the cloud computing model?
2. What is the difference between security, cybersecurity, and cloud security?
3. Which three hyperscale providers are leading the global world of cloud computing?
4. What is cloud security?
5. What is the CIA triad and what does each constituent mean?
6. What are the differences between virtual machines, containers, and functions?
7. What is a well-architected framework?
8. What security considerations are specific to cloud services?
9.

10. What is the primary security consideration in a multitenant environment?
11. What is authentication? What is authorization?

References

1. Microsoft Azure Well-Architected Framework:
   https://learn.microsoft.com/en-us/azure/well-architected/)
2. Gartner research:
   https://www.gartner.com/en/newsroom/press-releases/2022-10-31-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023
3. McKinsey Digital research:
   https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/projecting-the-global-value-of-cloud-3-trillion-is-up-for-grabs-for-companies-that-go-beyond-adoption
4. Have I been pwned?
   https://haveibeenpwned.com/

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 2
Service Models and Deployment Models

Introduction

In this chapter, we will discuss and learn about the two models – service models and deployment models.

Cloud service models refer to the diverse types of services that are provided by the cloud platform. Cloud deployment models refer to diverse ways in which cloud computing services are made available.

Cloud security is then layered on top of these models, on top of services and on top of your and your organization's applications and services.

Structure

The chapter covers the following topics:

- Service models
- Deployment models

Objectives

By the end of this chapter, you will be confident about the definitions and differences between types of service models and types of deployment models.

You will also learn what the impact on security is in each of the types and what the many fallacies and assumptions may impact your decision on when to use which model (as it may very well be the case that your and the needs of your organization will have uses for different models based on the different workloads you may be trying to bring to the cloud).

Service models

Cloud computing providers offer various services. They can broadly be classified into Infrastructure as a Service, Platform as a Service, and Software as a Service. Before we go into the details of each, let us first focus on what is the service model and what are the characteristics we can use to judge a service model for its suitability, especially in terms of security.

Self-service

The cloud service model is a self-service one, or more accurately, on-demand self-service. What this means is that it offers access to the services you and your organization need without going through a cumbersome process of human interaction or a laborious process. All you need to do is sign up. And you can do that now. You can go to any of these URLs (internet locations) and sign up right now for an account that will have access to almost all the services available:

- Azure:

  https://azure.microsoft.com/en-us/free

- AWS:

  https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all

- GCP:

  https://cloud.google.com/free

You even get some completely free services and even credits for services that do have a cost so you can try whatever you need or are curious about. The reason for all this is that the competition between these providers and the genuine usefulness of the cloud are great incentives for each of them to seamlessly onboard onto the platform and try everything they have to offer – with almost absolute confidence that you will use more and more as you go forth and eventually end up being a paying customer.

Outside of just signing up, self-service is important going forward as you start deploying workloads to the cloud because you can be certain that all these services will be there for you when you need them.

For a security-related example, you deploy your first website and start offering a few products for sale and you discover one day that your website is down, and the traffic is enormous. You do some research and discover that this is called a **Distributed Denial of Service** (**DDoS**) attack.

DDoS is a flood of traffic so large in volume and scope that it brings your website down because it cannot handle so much traffic at any one time. Now all you can do is scale your site so it can handle a lot more volume, but some of the DDoS attacks in the wild are so huge it would bankrupt you to handle them. One reason DDoS attacks are hugely problematic is because they are usually asymmetric, meaning a little bit of generated traffic from the source can amplify and result in huge computation times or responses so it is cheap to start the attack but expensive to prevent it.

So, what can you do? It is just you, and you do not have experience with something like this. Luckily, there is a service that can mitigate this attack for you (either automatically or by working with the support of the cloud provider), usually called something like **DDoS protection or mitigation**. You can turn this service on and gain the benefits at once. If this was not available or if it was not available as a self-service, you would be out of luck, and this could result in a long downtime as you try to address this yourself.

Another easier example, although not security related, is just making products that are so popular that actual users want to buy them, and they accidentally DDoS you, just by legitimate traffic trying to access your site wanting to buy the products. In this case, you may be able to scale the services underpinning your site (such as an app service, a database service, or a storage service) to allow you to process all this traffic and sell your products.

Now that is what self-service is all about. It is easy access to services whenever you need them. Contrasting this with an IT department in enterprise companies running their own infrastructure, this is a difference between getting access to a service in minutes versus days, weeks, or months. This works not just for you but for anyone in your organization – anyone can access these services. Security comes back in when you try and manage this proliferation of services obtained on demand by folks in your organization so the services obtained in this way stay secure.

Resource pooling

Cloud providers can supply all the infrastructure through the self-service mechanism because they have a lot of it. Some of the data centers can have around a hundred thousand virtual machines in just part of one location (usually called an availability zone, and each cloud region consists of multiple availability zones).

So how can they afford to have the infrastructure you need when you need it? One aspect of self-service is also - self-return. You can provision a thousand virtual machines for a year, but also for a week or even for an hour or a minute. And you only pay for the time you are using them.

One way this can be made to work is to supply a pool of resources that can then support a much larger pool of customers because not everyone will need the resources at the same time and at the same volume. The public cloud is full of services that are provided as a pool, meaning shared, which leads to efficient use and cost optimization. This also means that virtualization is used to subdivide larger infrastructural items like that is storage into smaller units of storage, which different customers can use. For example, if you have a disk of 1TB, you could subdivide it in two and allow two customers to use half of it each. Customers, of course, must not know of this, be able to access data from one another, or, in the case of a disk, be able to access the data previously stored on them by a previous customer.

Securing these aspects is luckily the responsibility of the cloud provider. You need to be aware of the cloud provider's security measures and responses, and - you must architect your workload to be aware of this. Because that disk may die and be replaced on the fly, or it may require an update to the underlying software that manages it, or the system may just decide to shuffle data around to accommodate another customer, so it could go away at any time. There are ephemeral services that are priced cheaply (or even free) because of it and there are higher tiers of services that give you assurance that in case anything happens to the disk, your data is still safe.

Resource pooling in the cloud applies to power, storage, compute power, memory, network bandwidth, etc. All of these are allocated dynamically – with no human intervention – to fulfill the self-service and scale requirements. No human can manage disks on hundreds of thousands of machines without automation.

Elasticity

Self-service and resource pooling have another consequence, which is the need for reallocation or resizing of the allocated resource. Because of these requirements events such as resizing a resource must be handled by the dynamic system. If you need more storage, get a larger disk or more disks. If you need more computing, get a larger virtual machine or get thousands more of them.

This is another great benefit of the cloud, which is elasticity. If you want to test a service that you want to offer to the market, elasticity allows you to start small (sometimes very small. For example, an AWS virtual machine called t2.nano comes with just 0.5GiB RAM and 1 vCPU).

- Learn more about the t2.nano here:

  https://aws.amazon.com/ec2/instance-types/t2/

Then once you realize the service is popular and you need to scale it, you can either do it through automation so it scales with demand or by manually scaling it as you see fit.

Outside of agility, which is the number one benefit of the cloud, elasticity is the second-best benefit. Never in history has anyone on the planet with an idea and internet access gone

From the security aspect, one aspect that is impacted by this is the attack surface. Scaling of course increases the attack surface. Scaling from one virtual machine to a thousand means you need to now secure not one but a thousand of them. Here is where automation comes in, building one machine and securing it, thoroughly testing it, and having auditing and alarming in place allows you to scale security also. One useful pattern for that in this case of virtual machines is the golden image, a single well-tested image that is then used through automation to create virtual machines with identical OS/data disks that are in a known good state. There is a lot more to this pattern but for now, it is important to understand the security implications of the fact that the cloud supplies one of the benefits - elasticity.

Network access

Broad network access through standard protocols is something on which all services rely in the cloud. Due to the nature of these services not being local to you or to your users, we must get to them somehow. We do this through the global network we call the internet. This also applies to services that need to communicate with each other, they also need networking because in a datacenter with hundreds of thousands of machines there might not be a guarantee that the disk is on the same one that does compute as well, so once again, networking is required.

Different devices, various locations, and a variety of services, all need to talk to each other, so networking is the best way we know how to do that.

From a security aspect networking, network access, control of the network access, and reliability of all networking services, all need to be secured. We need to ensure that access works for all our services, but also for all users accessing them, and all administrators administering them, while at the same time limiting such access to only what is strictly necessary and restricting it when not necessary or when it is outright forbidden.

For example, in GCP you configure rules as a firewall rule to restrict network access and you should avail of this service (and others) to secure your network. One good thing about cloud networking is that it is very similar to your local networking, the rules are quite similar, and the only difference is implementation. For example, in GCP a rule can have a priority of anything between 0 and 65535 while Azure allows for anything between 100 and 4096. This means you will need to adjust your automation scripts to account for this, but the rules themselves are the usual allow/deny rules as you would have to set in your own local network (to a point of course, integration between services may require some configuration that you would not necessarily see locally, such as Azure's service tags which are used to manage and keep automatically updated lists of IP addresses assigned to services).

Pay-as-you-go

One of the benefits of the cloud model is the pricing.

Pay-as-you-go allows you and your organization to pay only for the services you use. Contrasting this with the IT models of the past where you

and when it was not in use the cost would still be there. With pay-as-you-go, you can use the services you require when you require them and only pay for them when you are using them. For example, if you have a virtual machine running for 2 hours a day you only pay for those 2 hours, or when you use a serverless function you only pay for the milliseconds of execution time, etc.

The scalability of infrastructure works well with the pay-as-you-go model. If you need more services, you can start using them and start paying for them and when you do not need them you can return them and stop paying for them. For example, you might need more computing and storage just before a holiday season for your e-commerce site due to increased demand.

This model also enables you to not have long-term commitments. If you were buying your own infrastructure once you buy it, it is there - whether you need it or not. For example, if you have a workload digitalizing all the existing patient records you need infrastructure to support that, but once all the records are digitalized you may no longer need the infrastructure. In this cloud model, you can have that infrastructure for the time it takes you to perform the digitalization of the records and then return it and stop paying for it.

Pay-as-you-go also comes with built-in granular billing, so you will know exactly how much each service is costing you – for each workload, for each service type, for each resource.

From a security perspective, this is also vital to keep track of as unexpected spikes can also show that there is an intruder that has taken over some infrastructure and is using it for their purposes, or if the data egress costs spike (meaning someone is exfiltrating data, that is copying or moving data to the outside of your environment) that may indicate that data theft is happening within your environment.

By paying only for the specific resource and for the time you are using them you only pay for the actual usage. And since in this cloud model provisioning and de-provisioning resources is a scriptable event or a few clicks on the administration console (or a portal) this allows you to really ensure you are not paying for any unused resources.

To further enforce this most services allow you to set up an elastic scaling meaning resources can expand or contract automatically as they are needed (For example, during times of day when there are more customers). You still only pay for the resources used but you do not even have to directly manage them to cover the use cases when there is more demand.

In this model, you pay for value. What this means is that you align costs directly to value. Paying for resources when you need them and they are directly valuable allows you to track cost per value directly, rather than averaging over the time when the resource does and does not provide value (which is the usual case when you own the resources you bought).

And finally, there is the Opex versus Capex benefit. Opex is the operational expenditure cost and Capex is the capital expenditure cost. Opex costs may provide financial flexibility as Opex costs can be more easily adjusted to meet business needs. In this case Opex benefits are provided by the flexible pay-as-you-go services as opposed to the Capex that you would have if you bought the infrastructure yourself in your own data center.

Virtualization

Virtualization provides a lot of benefits when it comes to managing infrastructure.

The first and most important concept (both from the cloud model and also from a security standpoint) is resource pooling. Cloud virtualization allows for the pooling of physical resources – servers, storage, networking devices – into a shared pool of compute, storage, and networking services. This enables efficient utilization and the cost benefits and the scalability benefits, as resources can be dynamically allocated to meet demand.

From a security standpoint, this is important to understand because different workloads, and different tenants (clients, customers, organizations) using the same resources between them require an understanding of security best practices. Usually, the cloud provider handles this. For example, they wipe the disks between uses, ensure network traffic is segregated, etc., all the way to no disk leaving the data center without being destroyed and chopped into the sizes of a few millimeters at most. Thus, ensuring the security and confidentiality of data is preserved.

Abstractions that virtualization provides also make it easy to use multiple physical resources as larger pools for your and your organization's use. For example, if you need a virtual machine with a large memory capacity or a large storage capacity all represented to you and your application or a service as a single resource you can have it in a virtualized environment, while the cost of such a single physical machine could be cost prohibitive.

Elasticity and isolation are also provided by the virtualization of resources. If you have provisioned a VM with 16GB of memory and now you need that same VM with 16 times as much (total of 256GB) you can just adjust the VM type from a list and have that available to you in near real-time.

Migration and mobility are simplified in a virtualized environment as moving a virtual resource usually requires just a setting (or an endpoint) adjustment rather than moving the physical hardware.

Compatibility and interoperability of multiple vendor types or resource types is also a possibility in a virtualized environment. Virtualization allows for seamless migration of workloads between different environments. This increases the flexibility in infrastructure choices and the possibility to run legacy applications and services on modern hardware. You will appreciate this if your organization is a large enterprise, as they most often have such legacy applications.

Disaster recovery is simplified as well, allowing for easier solutions to some of the business continuity challenges. One term to remember here is snapshots. Snapshots are exactly what they sound like a point-in-time capture of a virtual machine, a database or a storage account, and they can be done at any time. Basically, a copy of the data as it exists at a point in time. This reduces downtime and improves **Recovery Time Objectives** (**RTOs**) and **Recovery Point Objectives** (**RPOs**).

From a security perspective, you need to be aware of the implications of virtualization to you and your organization. While these are handled by the provider, they may have implications on the security of your and your organizations' workloads.

There is a whole slew of attacks on the virtualization layer, such as virtual machine isolation (a way to isolate virtual machines from each other so issues, attacks, or breaches of one won't affect the other), virtual machine escape attacks (if one VM is compromised and the virtualization layer has a flaw it may be possible for an attacker to go from on VM to another), data leakage, insecure virtual machine images (ensuring the safety of the VM image is your responsibility), insufficient resource allocation (allowing for all sorts of resource exhaustion attacks), misconfiguration, and again generally shared infrastructure risks.

High availability

Achieving high availability in the cloud is done through redundancy of all critical components. All critical components such as virtual machines, networking, and storage and all (or almost) services have redundant counterparts. High availability is achieved in such a way that if one component fails another takes over seamlessly. This is again a shared responsibility between the cloud provider and you and your organization. You have to architect for high availability, as the services provided to you by the cloud provider will happily allow you to run just a single instance if you choose to do so.

Increased uptime, minimized downtime, and improved reliability are the benefits of this approach. Cost saving comes from not having to manage these failure events yourself, but to delegate the management to the services themselves.

This redundancy does, however, add complexity to the infrastructure, requiring security measures to ensure redundant components are all equally protected. This is best done through automation.

Systems that are designed to be fault tolerant (that is, they can withstand different failures without impacting the overall system) further enhance resilience reducing the impact of software or hardware failures. Securing all the fault-tolerant systems and ensuring all failure modes are covered by comprehensive security is very important. Intrusion detection systems and a comprehensive approach to observability help identify and respond to security events should they occur.

Load balancing is used to distribute incoming traffic evenly across the resources.

If there is one resource, 100% of the traffic goes to it.

This extremely basic example of high availability is shown in the following figure. First, take a look at the starting conditions – with just one instance of a resource:

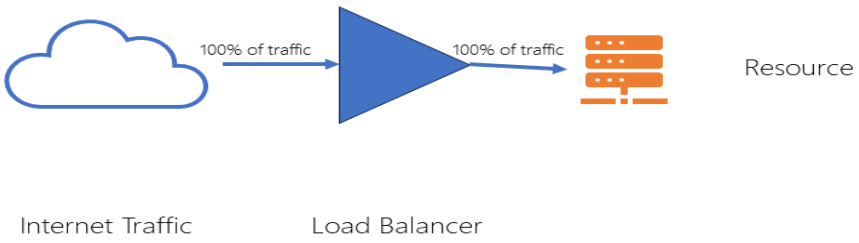


***Figure 2.1**: Load balancing with one resource*

Now consider a high availability type scenario with 3 identical resources behind a load balancer. If there are three resources, 1/3 of traffic gets directed to each one, and if one fails, then it goes from one-third of traffic going from each of the 3 resources to half of traffic going to each of the remaining two resources, as shown:

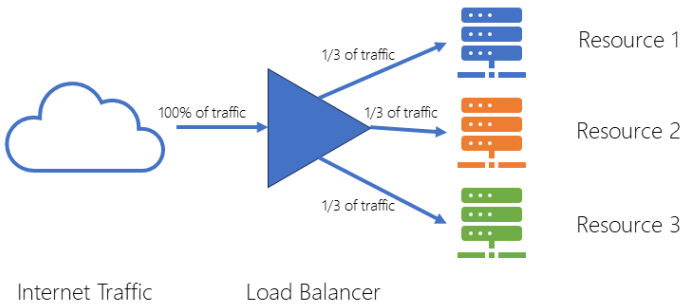


***Figure 2.2**: Load balancing with 3 resources (all healthy)*

Finally, consider how high availability ensures the continuous functioning of a service when one resource out of the three starts having issues and is taken offline. Refer to the following figure:

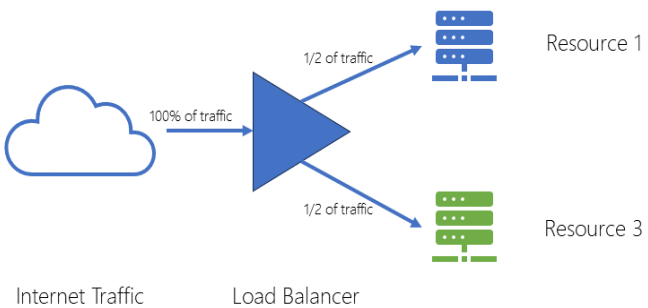


Figure 2.3

If a resource comes back (restarts) the traffic routing will return to the one-third split. All this presupposes that traffic is routed equally to each resource.

As you can see, all traffic is routed to one endpoint (whose high availability is managed by the provider – if it dies it will get spun up again) that is load balancing the traffic to not one, but more than one resource behind it, giving the high availability resiliency through the expectation that not all of the resources behind the load balancer will fail at the same time, so if one or two do fail there is plenty of others to handle the traffic. Note that all resources behind the load balancer are identical as they need to handle the identical traffic coming in.

Load balancers need to be properly configured and secured to prevent unauthorized access.

Another important approach to high availability architecture is automatic failover. In the examples above, suppose that the load balancer and the resources are all in a single region. If you want your workload to failover to another region you need to replicate that setup in one or more regions. In the case that an entire region is lost you can then send all traffic to another region. This is the basic approach to automatic failover.

Geographical redundancy comes into the high availability as well, as regions need to be sufficiently distant from one another. Usually, this is three hundred miles or rounded up to five hundred kilometers (300 miles = 483km) between each region. This is done to ensure that events such as floods or electricity outages have less likelihood to occur at both of these distant locations.

Data replication across these geographical regions needs to be secured in transit and when stored.

Resilient networking is an essential requirement for a highly available system. Latency implications (due to the distance traveled by the signal) and seamless communication between resources need to be solved by the architecture as well.

Network-based services are vulnerable to network-based attacks, such as DDoS attacks.

In order to do any of the high-availability architecture designs and then manage them when issues occur, automated health checks need to be implemented by all the various services and resources so that the network can automatically route around or failover as needed. This helps with proactive identification of potential issues before they occur, reduces manual interventions needed, and allows for automated (faster) responses.

Privacy

To ensure privacy, data encryption protects sensitive data. All cloud providers automatically encrypt the data when stored anyway, but that is at the service level across all tenants. This ensures that if a physical disk with data is stolen the data is encrypted and essentially useless.

To ensure that data remains confidential and complies with privacy regulations, encryption in transit and at rest must be used. There are many options for choosing keys for encryption:

- Server-side encryption (using service that is, cloud provider managed keys)
- Customer-managed keys (for example in a specific service that stores secrets)
- Customer-managed keys on customer-controlled hardware
- Client-managed keys (completely externally to the cloud provider)

Due to specific regulations such as GDPR in the EU, the model that is encouraged by cloud privacy advocates is to only store the necessary data for the minimum length of time required to store it. This decreases the attack surface as minimal data is collected as needed and disposed of when no longer needed. This approach is sometimes referred to as data minimization.

As with all data, access controls, access reviews, and other data compromise reduction schemes are required to prevent unauthorized access to data.

Segregating data at different levels also helps with this. You can store all data for all your customers in a single location or you can segregate it and structure the environment in such a way that a compromise of a single customer of yours does not also compromise data from your other customers.

Transparency when collecting, storing, and using data is the best policy. Acquiring consent from those participants you are collecting data from and informed consent will save a lot of privacy and compliance-related issues later.

Collected data is valuable and might be used for a variety of purposes (with consent), so you may be required to move data or even duplicate data so that it can be used for these different purposes more easily. Implementing secure data export and import mechanisms, encryption during data transfer, and validation of data integrity ensures data portability without compromising security.

Once data is no longer needed it can be discarded (that is, securely deleted) or anonymized or pseudonymized. Anonymization removes **Personally Identifiable Information** (**PII**) data, and pseudonymization on the other hand replaces identifiable data with pseudonyms.

Implementing proper techniques and processes to ensure the effectiveness of anonymization and pseudonymization methods, and preventing potential re-identification attacks is an important part of securing the data.

Should an incident occur, and data accessed through a breach, a well-defined incident response plan will help to deal with the consequences. This plan needs to be in place before the breach occurs as it will be difficult producing the plan during or immediately after.

Conducting regular drills while following the plan will identify gaps and through iteration the plan will get better and the confidence of you and everyone in your organization will grow through the shared knowledge of the plan. In certain jurisdictions having this plan in place is a legal requirement.

Data transfer

The cloud bandwidth model offers high-speed connectivity, ensuring fast and efficient data transfer between cloud resources and your users.

Usually ingesting data (that is, data that is coming into your cloud environment) is free, while sending data back to your users (that is, data that is leaving your cloud environment) is not free. This is not a benefit of the cloud, but rather a mechanism to keep you from switching to a different cloud provider. There is an expression that illustrates this: **data has gravity!** This means that once a large amount of data is in place, it is difficult to move it.

So, you need to plan for data movement. Between you and your clients and customers, and between your environments.

High-speed networking requires robust security measures designed such that traffic is always encrypted, that the network is segmented and that intrusion detection systems are fully operational.

Bandwidth provided scales with the use of different-sized services, so you need to make a note and a connection between services and their bandwidth requirements, and the bandwidth provided. Bandwidth is usually cheap, but because there might be a lot of data going back and forth it might add up to a significant cost. This cost makes the organizations then plan and use different mechanisms to minimize the costs, such as collocating data (that is, copying the data closer to consumers) or using **Content Delivery Networks** (**CDNs**), which replicate data to hundreds of locations worldwide, so they are as close to the end users as possible). This requires a security audit of these patterns and careful testing of access patterns.

Quality of Service (**QoS**) requirements might need to be implemented at service levels, as not all services and not all bandwidths are equally important. For example, it might make sense to answer queries from the live site immediately, but your background replication processes might not be important enough to have real-time replication. So, implementing QoS mechanisms to prioritize bandwidth allocation to important services might be required.

Proper configuration and monitoring of QoS policies help ensure that security-critical applications receive appropriate network resources without compromising security.

As with all other services, redundancy is key to high availability. Cloud providers already have the necessary redundancy in their network connectivity to and from the cloud, but you might need the same robust implementation at the edge as well (for example, your office, important customer locations, etc.).

Security needs to be evaluated for each provider you choose to connect to and failover needs to be tested as this is when security issues can creep in.

Network security is always important, but from a security standpoint, securing networks with massive amounts of data coming in and out needs careful planning. Network security

appliances may be expensive, and they scale with some difficulty depending on your and your organization's requirements. Deciding which traffic to manage in what way is important as costs usually don't align with the most secure solution in this case. You must not compromise security for the sake of efficiency here as unsecured traffic is an incident waiting to happen. There is also the flipside to that, having a lot more security and traffic inspections between your own environments may be an overkill, so you need to carefully decide what is appropriate. And what is appropriate in one case or workload may not be in another.

Marketplace

What is a cloud marketplace? It is a centralized (from an access standpoint, the architecture behind it is certainly not) platform where you and your colleagues from the organization can discover, procure, and deploy a wide variety of cloud-based services, applications, and resources from a wide variety of vendors. For example, network security appliances come in a lot of shapes and sizes, and the marketplace is usually the best place to discover, procure, and deploy them.

These marketplaces simplify service discovery (it is just like other e-commerce sites; you search for, evaluate, and find the solution that fits your needs) and streamline the procurement process. Usually, you can pay for them together with your cloud bill.

From a security standpoint, again, here you must be careful to evaluate these services for suitability and security. Not all marketplace offerings are secured by default, as they can be there for demos, trials, discovery, or any other purpose – other than for running them in production. Additionally, sometimes the best security is only found in more expensive tiers of the offerings. Sometimes, the vendor keeps access to these environments to assist you or verify your usage, or for other reasons, and this may then become a back door into your environment if not secured. So, due diligence is required when using resources from the marketplace.

Marketplace's greatest benefit is its largest security weakness – diverse offerings. You need to evaluate each and every one, even if they come from a provider you already know and love.

The good news is that the marketplace offerings come with simplified deployment, as these are the prebuilt solutions for a particular customer problem identified by the vendor. Hence why they provided the solution in the marketplace. These are template-able solutions that come with automation scripts and configuration defaults. You need to double-check and test all the assumptions in those templates, scripts, and configurations if you want to ensure the security of the workload.

One way you can streamline evaluation is by checking out the ratings and user reviews for solutions. This must not prevent you from still doing your due diligence, but these just might call out the security issues if present, which you can then address or find another, more appropriate solution. As with any marketplace, do not automatically believe any positive reviews.

Marketplace fosters that cultural requirement around agility as well by giving you quick and easy templates and scripts and an easy path to production for these marketplace services. Sometimes this can increase your agility with regards to security as well as you can get your security setup quicker and easier than you thought possible. Sometimes, you might discover a security solution you need and haven't even considered by just browsing the marketplace. This is especially useful for a beginner as there are already a lot of ready-made services there for you to evaluate.

It is important to also not make the decision once to deploy something from the marketplace but to regularly come back and re-evaluate the decision. Maybe there are newer and better solutions, maybe there are solutions that were not available before, and maybe there is an improvement to the existing solution. So, check back often.

If you are part of industries, communities, or other organizations that care about security, some of the marketplace solutions may be already curated for you, so you can start there, as opposed to the entirety of the marketplace.

Azure, AWS, and GCP all have their respective marketplaces and are a great place to start by checking them out before planning and building your own service. Refer to the following links:

- Azure Marketplace:
  https://azuremarketplace.microsoft.com/en-us/
- AWS Marketplace:
  https://aws.amazon.com/marketplace
- GCP Marketplace:
  https://cloud.google.com/marketplace

IaaS vs. PaaS vs. SaaS briefly

You might have seen the following figure already (on-premises included in this one as well) as it is commonly used to show a difference between the models:

On-Premises	IaaS	PaaS	SaaS
Data	Data	Data	Data *
Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime
Middleware	Middleware	Middleware	Middleware
OS	OS	OS	OS
Virtualization	Virtualization	Virtualization	Virtualization
Servers	Servers	Servers	Servers
Storage	Storage	Storage	Storage
Networking	Networking	Networking	Networking
Data Center	Data Center	Data Center	Data Center

Data * This is a shared model between data stored in SaaS and your own data

***Figure 2.4**: Differences between what you manage (dark shade) and what the cloud provider manages (light shade)*

Let us now go into more detail on each of the models with their characteristics and security implications.

Infrastructure as a Service

Let us consider what **Infrastructure as a Service** (**IaaS**) is, and in the following sections, we can contrast it with the other models:

- Provides you with virtualized infrastructure components such as VMs, storage, and networking.
- You have the most control and the most responsibility for securing applications, services, and data.
- High scalability and flexibility, allow you to provision and scale resources based on demand.
- Provides you with an environment where you can install and configure your preferred operating systems, applications, and services.
- You are responsible for maintaining and patching the operating system, applications, services, and runtime environments.
- Allows you to rapidly provision resources while you maintain the responsibility for application deployment and configuration.
- Pay-as-you-go model with costs based on resource consumption.
- Offers the most flexibility and easier migration to different providers due to lower levels of dependency on provider-specific tools and services.
- Compliance requirements are on you. You need to implement security controls to meet those requirements relevant to the infrastructure and services used.
- Requires you to have skills such as managing and configuring virtualized environments and strong knowledge of operating systems, security hardening, and network configuration.

IaaS provides virtualized computing resources over the Internet, and each major cloud provider offers a range of IaaS services. Here are a few examples from Azure, AWS, and GCP:

- Azure:
    - **Azure Virtual Machines**: This service offers scalable, on-demand virtual machines for running applications.
    - **Azure Blob Storage**: A scalable, object storage solution for storing large amounts of unstructured data.
    - **Azure Virtual Network**: Provides an isolated, private environment in the cloud to securely run VMs and applications.

    - **Azure Disk Storage**: Offers high-performance, durable block storage for Azure Virtual Machines.

- AWS:
    - **Amazon Elastic Compute Cloud (EC2)**: Provides resizable compute capacity in the cloud, allowing users to run and manage server instances.
    - **Amazon Simple Storage Service (S3)**: An object storage service that offers industry-leading scalability, data availability, security, and performance.
    - **Amazon Virtual Private Cloud (VPC)**: Enables the provision of a logically isolated section of the AWS cloud where you can launch AWS resources in a virtual network.
    - **Amazon Elastic Block Store (EBS)**: Offers block-level storage volumes for use with EC2 instances, providing high-availability and durability.
- GCP:
    - **Google Compute Engine**: Provides scalable and flexible virtual machine instances for running applications.
    - **Google Cloud Storage**: A unified object storage solution that leverages Google's infrastructure for storing and retrieving any amount of data.
    - **Google Cloud VPC (Virtual Private Cloud)**: A comprehensive set of Google-managed networking capabilities to connect VM instances, Kubernetes clusters, and other resources.
    - **Google Persistent Disk**: Provides block storage for Google Compute Engine and is designed for high durability and availability.

These services are fundamental offerings in each cloud provider's portfolio and form the backbone of many cloud-based infrastructure solutions.

Platform as a Service

Let us consider what **Platform as a Service** (**PaaS**) is, and note the differences between IaaS already:

- Offers a development platform and runtime environment for building and deploying applications and services.
- You have less control over the underlying infrastructure but retain responsibility for securing applications and the data.
- Application development and deployment is provided in a scalable and flexible way, with built-in capabilities for autoscaling and resource allocation.

- Offers various development frameworks, tools, and runtime environments that simplify applications and services development and deployment with a more limited set of customization options.
- Cloud provider patches the underlying platform services, with you in charge of patching and supporting just your applications and services built upon them.
- Provides preconfigured development environments that reduce time to market by accelerating your application and services development in these environments.
- A pay-as-you-go model based on resource usage and features and tiers provided by individual cloud services.
- Some level of vendor lock-in due to specific services and configuration options available.
- You still need to ensure that the developed applications and services comply with industry standards and security and privacy regulations.
- You need to be proficient in application development, programming languages, and deployment tools specific to the PaaS services. Understanding of platform APIs is required as well. Must be familiar with architecture patterns, scalability patterns, and performance optimization patterns.

PaaS provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. Here are a few examples of PaaS services from Azure, AWS, and GCP:

- Azure:
    - **Azure App Service**: A fully managed platform for building, deploying, and scaling web apps.
    - **Azure SQL Database**: A fully managed relational database service based on SQL Server.
    - **Azure Kubernetes Service (AKS)**: A managed Kubernetes service for running and scaling containerized applications.
    - **Azure Functions**: A serverless compute service for running event-driven functions without managing infrastructure.
    - **Azure Logic Apps**: A cloud service that helps you automate and orchestrate tasks, business processes, and workflows when you need to integrate apps, data, systems, and services across enterprises or organizations.
- AWS:
    - **AWS Elastic Beanstalk**: An easy-to-use service for deploying and scaling web applications and services.

  - **Amazon Relational Database Service (RDS)**: A managed relational database service that supports MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB.
  - **AWS Lambda**: A serverless compute service that lets you run code without provisioning or managing servers.
  - **Amazon Simple Storage Service (S3)**: Although primarily an IaaS offering, it can be used as a PaaS for hosting static websites and storing application data.
  - **Amazon Elastic Container Service (ECS)**: A highly scalable, high-performance container orchestration service that supports Docker containers.
- GCP:
  - **Google App Engine**: A fully managed platform for building and deploying applications across a range of languages.
  - **Google Cloud SQL**: A fully managed relational database service that supports PostgreSQL, MySQL, and SQL Server.
  - **Google Cloud Functions**: A serverless execution environment for building and connecting cloud services.
  - **Google Kubernetes Engine (GKE)**: A managed, production-ready environment for running containerized applications.
  - **Google Cloud Run**: A managed platform that enables you to run stateless containers that are invocable via HTTP requests.

These PaaS offerings provide a mix of development tools, database management systems, application hosting capabilities, and more, simplifying the process of deploying and managing applications in the cloud.

Software as a Service

Let us consider what **Software as a Service** (**SaaS**) is and note the differences between IaaS and PaaS models:

- Delivers a complete software application for you to use.
- You have the least control (if any) over the infrastructure and services used. The provider manages most aspects of security. You may need to secure the user's access to the application and integrate for a single sign-on capability.
- Limited flexibility as you cannot modify the software application other than configure your own piece of it.
- You get ready-to-use applications, a service, or a suite of them with some minimal customization options – both per tenant and per user.

- The provider is responsible for all security of the underlying hardware, architecture, services, data, and access patterns.
- You do have instant access to the fully functional software with no development or deployment requirements from your side.
- Subscription-based model, usually tiered in some way is based on user licenses or usage tiers.
- You have selected a vendor/provider and switching may be difficult as another vendor/provider may have built software in a unique way and with different services built in.
- Security and privacy are the responsibility of the provider, usually with a small level of customization to account for global customers in different jurisdictions.
- Familiarity with the software is the only skill requirement, other than the general requirements of knowing how to use the computer, mouse, keyboard, and internet browser.

Software as a Service (**SaaS**) is a software distribution model in which applications are hosted by a third-party provider and made available to customers over the Internet. Here are a few examples of SaaS services offered by Azure, AWS, and GCP.

- Azure:
    - **Microsoft 365 (formerly Office 365)**: A suite of cloud-based productivity tools including Microsoft Word, Excel, PowerPoint, Outlook, OneNote, and Teams.
    - **Dynamics 365**: A collection of enterprise resource planning (ERP) and customer relationship management (CRM) software applications.
- AWS:
    - **Amazon WorkMail**: A secure, managed business email and calendar service with support for existing desktop and mobile email clients.
    - **Amazon Chime**: A communication service that lets you meet, chat, and place business calls inside and outside your organization.
- GCP:
    - **Google Workspace (formerly G Suite)**: A suite of cloud computing, productivity, and collaboration tools, software, and products developed by Google, including Gmail, Docs, Drive, Calendar, and Meet.
    - **Apigee**: A platform for developing and managing API proxies that allow enterprises to securely share data and services across myriad devices and channels.

These services represent just a small selection of the SaaS offerings available from each provider, as many other services are also available either directly from these companies or from third-party vendors within their ecosystems.

Deployment models

The cloud deployment model refers to a few different ways in which cloud computing resources are made available to customers.

Public cloud

Microsoft Azure, Amazon Web Services, and Google Cloud Platform are the current dominant players in the world. They all provide a hyperscale cloud model that is interchangeable; that is to say, you can almost perform all actions in each of them, but slightly differently.

Here are the characteristics of the public cloud model:

- The infrastructure is owned and operated by a provider, and customers have limited control over the underlying architecture and infrastructure of the platform itself.
- This infrastructure is hosted in the provider's data centers, which are quite geographically dispersed to cater to most world locations.
- Security measures are in place to protect customers' data, but the customers are responsible for securing the applications and data they host in the cloud.
- Limitless (practically) scalability and performance capabilities allow customers to scale dynamically to meet changes in demand.
- The pay-as-you-go model is the norm. Pay only for the resources you consume.
- Possible to become dependent on cloud providers' APIs and services.
- Internet connectivity is required between the provider, all the customer locations, and all user locations.
- Robust disaster recovery options are available, including data replication and geo-redundancy.
- Organizations require a high level of expertise in managing and securing the workloads deployed.

Private cloud

Let us take a look at the characteristics of a private cloud. They are described as follows:

- Owned and operated by a single organization, providing greater control and options in defining the data center.

- Hosted on-premises within the organization's own data centers or a private cloud provider.
- A lot more control and a lot more responsibility over data security and compliance.
- Limited scalability when compared to the public cloud model.
- Higher upfront costs are a characteristic of the private cloud, due to the investments needed to maintain them. It does offer long-term cost advantages for steady workloads over time.
- More control over deployments but may still rely on specific hardware and/or software vendors.
- Usually offers direct network connectivity to your organization's infrastructure, either from a third party or your own data centers.
- Have to design and implement a lot more disaster recovery and business continuity strategies.
- Require skills in building, maintaining, and securing the infrastructure at a low level.

Hybrid cloud

The hybrid cloud is a mixture of both public and private cloud capabilities.

Public cloud environments require more trust in the cloud provider's security measures, while private cloud and hybrid cloud deployments offer greater control over security configurations and policies, which usually puts a lot more onus on the efforts of you and your organization for which you may or may not have skills. Is your organization as good at securing infrastructure and does it have the same capital to fund those efforts when compared with the multi-billion-dollar investments by the hyper-scale cloud providers? It has the following characteristics:

- Physical locations affect data sovereignty, regulations, and compliance, so a public cloud provides greater access to global markets, while the private cloud offers easier access to the data center if required.
- Public cloud requires you to follow the models it prescribes for the best security measures to protect your workloads. A private cloud provides more control over security and compliance with a lot more work and skills on your side.
- Scaling workloads in the cloud, you must consider and apply consistent security controls and automation options to ensure consistency.
- Cost optimization measures should not compromise security in either model.
- Vendor dependency and lock-in can impact your organizations' ability to switch technologies, providers, or platforms easily in both public and private clouds. If you need to plan for transitions in the future, it is best to plan for it now in order

- Hybrid cloud requires connectivity between your public and private clouds, and there may be limited options due to geography and available providers of such connectivity. You also need to consider interception, unauthorized access, and DDoS attacks. Encryption and access controls are your best friends here.
- You can leverage the public cloud for disaster recovery or scaling beyond the public cloud capacity, but this too needs a security plan to implement correctly.

Multi cloud

Multi cloud approach simply means deploying your workloads to more than one cloud provider. This can mean different workloads in different clouds or the same workloads across the clouds. There are implications for both.

Here are just some reasons to consider a multi-cloud approach:

- You want to mitigate the vendor lock-in.
- You want to improve resiliency and redundancy further.
- You want to use best-of-breed services from each provider.
- You need geo locations not available with just one provider.
- You want to cost-optimize the best solution for each workload.
- You want to have more agility in moving workloads from one provider to another.
- You want to avoid a single point of failure in case one provider suffers a catastrophic failure across their locations.
- You need to manage different compliance requirements in different geographies.
- You want to have access to ground-breaking services as soon as they are available.
- You want to have negotiating power over providers to negotiate better terms.

Here are some more reasons for a multi cloud approach:

- You want to have increased complexity and management overhead managing multiple cloud providers.
- You want to have skills in multiple cloud provider frameworks and services, and you want to hire and maintain more staff to support this.
- You want to move applications, services, and data across the providers, and you want to struggle with completely different services and similar but not quite services.
- You want to double the number of integrations you need to create and maintain.
- You want to increase the cost and complexity of networking.
- You want to increase the work you need to do to keep security and compliance risks under control.

- You want to fragment your support options, maintainability, and accountability between different providers.
- You want to manage multiple vendors and contracts and contract negotiations.

Hopefully, with this list, you see that the choice to go multi cloud or a single cloud vendor should be a carefully considered one.

Finally, here are a few real-life examples of why multi cloud is a promising idea or a necessity:

- You are part of a large 1000+ developer organization and are working in multiple industries or on multiple products that don't need to interact.
- You are a part of an organization that is acquiring a lot of other organizations and they all have their own tech stacks, products, and cloud dependencies.
- You are modernizing and are temporarily in a multi cloud mode until you complete the modernization and settle in one cloud.
- You are a part of an organization that is a public sector organization, and you have a requirement from the government to be multi-cloud.

As you see, nothing is black and white and there are pros and cons for everything. So, think carefully and choose the solution that you can support, especially considering the security requirements of supporting your decision. Usually, that can guide you in the right direction.

Conclusion

In this chapter, you have learned about all the different cloud and deployment models and the nuances between them as well as the security implications of each. You should be able to pick and choose models that make sense for you and your organization and your organizations' workloads.

Following this, we will discuss more about cloud security and the shared responsibility model essentially important to understanding the benefits of cloud security and the differences between cloud security and on-premises security responsibilities.

Questions

1. Do you have a free account on Azure, AWS, and/or GCP? If not, what are you waiting for?
2. Have you browsed the Azure, AWS, and GCP marketplaces?
3. Have you browsed the Azure, AWS, and GCP marketplaces in the security category?
4. Define 10 differences between IaaS, PaaS, and SaaS.

5. If you are tasked with securing IaaS, PaaS, and a SaaS environment, what specifically do you need to pay attention to with each model?
6. What are the differences between the Public, Private, and Hybrid models?
7. When should you consider a multi-cloud model approach?
8. What are the security implications of deploying workloads into a shared environment?
9. What should you be mindful of when you are designing your network in the cloud?
10. What are the advantages and what are the disadvantages of the pay-as-you-go model?
11. What are the security implications of virtualization?
12. What are the security implications of high availability?
13. What is a load balancer and how does it work?
14. What are the benefits of marketplace solutions?
15. What are the security implications of deploying a marketplace solution?
16. Pick a workload you must manage. What is the best model for it? IaaS or PaaS?
17. When is a multi-cloud approach necessary?

References

- Azure free account:

  https://azure.microsoft.com/en-us/free
- AWS free account:

  https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all
- GCP free account:

  https://cloud.google.com/free
- Learn more about the t2.nano AWS instance here:

  https://aws.amazon.com/ec2/instance-types/t2/
- Azure Marketplace:

  https://azuremarketplace.microsoft.com/en-us/
- AWS Marketplace:

  https://aws.amazon.com/marketplace
- GCP Marketplace:

  https://cloud.google.com/marketplace

Chapter 3
Shared Responsibility and Supply Chain

Introduction

In this chapter, we will discuss and learn about one of the most important security issues in the world of computing today and even outside of the world of computing as well.

Everything you and your organization do in the cloud, in your software development and deployment process is not done in isolation.

Unless you and your organization are that very special snowflake that produces all software code internally (that is, without contractors and without outsources), uses only in-house developed libraries (that is, without using common libraries for cloud services or other ancillary functionality), and runs your own data centers (that is, is not in the public cloud or even in a hosting solution provided by others), you are subject to supply chain considerations.

Therefore, shared responsibility concepts come into play because now you must place trust in others, software library authors, cloud providers, open-source modules for your pipelines, etc. And absolutely nothing is as perilous for security as trust. We will go into this cynical view that is at the same time pragmatic as well in this chapter.

Structure

The chapter covers the following topics:

- Trust
- Supply chain
- Software development
- Software deployment

Objectives

By the end of this chapter, you will understand trust, why it is important, and how to protect against trust-based security issues.

You will learn what the impact on security is and what the many fallacies and assumptions that may impact your decision on when to use which model, as it may very well be the case that you and the needs of your organization will have uses for different models based on the different workloads you may be trying to bring to the cloud.

Trust

Consider the following thought experiment. In 2001, *Warren Edward Buffett* (an American business magnate, investor, and philanthropist) gave a talk at the University of Georgia.

He asked this question (paraphrased) first:

- If you could invest (that is, bet toward their success) in your friend's future and get a certain percentage of their income for life, which of your friends would you pick?

And then he asked (paraphrased):

- If you could short (that is, bet against their success) a friend's earnings, which one of your friends would you pick?

These two questions are less about money than they appear. They are more about inviting you to think about what qualities you should and should not look for in your friends and surround yourself with those who do not have the characteristics that would make you consider shorting them.

Now, you could also argue that trust is a crucial element as well. So, now let us define trust.

Definition of trust

Trust, in general terms, without thinking about cloud computing, supply chain, and so on, is a fundamental belief and/or confidence that someone or something is reliable, honest, and will almost always act in a predictable manner.

It also necessarily involves a willingness to rely on or place faith in a person, an organization, or a process, all based on perceived experiences, reputation, and integrity.

Why is trust good?

Trust underpins cooperation, collaboration, and even the functioning of societies and institutions in general. In real-life (as opposed to cloud computing), it would be impossible to function without trust, as it is simply impractical to verify everything we need to live in a society.

For example, you place trust in some or all of these:

- Capability of institutions to keep the food you buy, the water you drink, and the air you breathe healthy, clean, and aligned with recognized standards. Otherwise, you would have to check these yourself every time you consume food, water, or air.
- The value of money, specifically your local currencies. Otherwise, you would not trust buying or selling anything.
- The rule of law locally and internationally. Otherwise, you would not be able to do business.

Trust is good because your life would be a huge pain (or even impossible) if you had no trust. And yet all too often trust is violated. Still, as a society, we have a trust that while sometimes that trust is violated, there are processes in place for those errors to be corrected and for trust to be established again.

In the case of life, trust is good and trust is needed in a high degree in order for you to be able to enjoy life with your fellow humans.

Why is trust bad?

When trust is violated, you feel personally hurt, and all those ideas of cooperation and collaboration fail. As discussed, in general, a degree of trust is required for the functioning of societies, due to the fact that it is impossible for you to verify each interaction.

The good news in a section of why trust is bad is that in cloud computing, trust is another variable to be considered and addressed, and due to a high degree of automation and enforcement it is possible to enforce trust.

There is a saying of **trust, but verify**, that was heavily used in US-Russian discussions and relations during the Cold War.

Learn more about this proverb and its significance on Wikipedia: **https://en.wikipedia.org/wiki/Trust,_but_verify**

Again, in real life, it is impossible to accept and act on this proverb in all situations, but in some real-life situations, you can do this. For example,

the process itself is based on **trust, but verify** as you interact with the agent, the bank, and other parties – allowing you to find and document your application. Everyone in the mortgage process accepts assertions from others and then verifies them through the documentation provided, which is trusted because this is not a one-off process for them. They have gone through it many times and they trust each other, but they still verify all the documentation.

In cloud computing, you can establish a similar process. There are two difficult things in computing, namely cache invalidation and naming things. So, we have named this process of **trust but verify** as – **zero trust**.

Zero trust

Zero trust concept is a concept and an approach that assumes that no entity – users, devices, or network should be automatically trusted. This goes for all entities, regardless of whether they are first party (yours) or third party (someone else's).

In this model, every single request is every single time verified and authenticated. Access is only ever granted if it makes sense for it to be granted based on a need-to-know and just-in-time framework.

The good news again is that in cloud computing, this is indeed possible to achieve.

The crucial aspects of the zero trust approach are:

- Verify the identity of users and devices (for example, username and password, or an API access key).
- Verify the originating location of these requests (that is, if the location is not the usual one, require further verification or deny access).
- Verify internal and external entities equally strictly (for example, one of your own services accessing another, or a customer requesting access to a file).
- Grant the minimum level of access needed to perform the task (for example, if the request is to read a file, do not automatically grant a write permission as well).
- Limit access only to the specific resources required to perform the task (for example, if access to a file is required, grant access to that file only and not to the parent container with all the files in it).
- Segment networks, applications, services, and resources into smaller and self-isolated zones (for example, in a multi-tenant environment separate individual files from different customers into different containers).
- Monitor (continuously) all access patterns. That is, know when anomalous access was granted and see if additional verification is required in the future, or know when anomalous access was attempted but was not granted and see if you can further strengthen and restrict access).

- Respond at once (ideally automatically) to any issues (for example, when you detect a mass egress of data from a customer, prevent it by stopping it and locking the account – and notifying the customer).
- Adapt access controls to consider all the numerous factors in granting access (for example, location, user behavior, device health, service health, time of access, number of requests, etc.).

Zero trust is a huge shift away from the traditional perimeter-based security, where the perimeter was protected, but all resources, activities, and processes within the perimeter were trusted by default.

The reason for this is not that cloud computing is inherently less secure than on-premises setups so you need this extra protection, it just forces you and your organization to do the things you should have been doing on-premises as well but – did not. If you manage on-premises environments in addition to the cloud environments, zero trust must apply to both.

The zero trust model enables your organization to enhance its security posture, protect sensitive data, and better mitigate the risks associated with common threats in today's modern world – including insider attacks.

In addition to the zero trust approach enhancing the security posture, it also enhances protections against certain other issues such as:

- Neglect and laziness (for example, forgetting to update service or enhance security, zero trust should detect this).
- Misconfiguration (for example, misconfiguring a service – zero trust should prevent access).
- Human errors and automation errors (for example, targeting the wrong service or sending too many requests – zero trust should alert of these).

Positives and negatives of trust

The best value of trust in cloud computing comes from:

- **Collaboration:** Ease of seamless collaboration among teams and other stakeholders due to trust in cloud services. Securely accessing shared resources such as data, services, and applications from anywhere at any time, and collaborating in real-time, result in increased productivity.
- **Cost:** Saving significant costs by trusting reputable cloud providers due to economies of scale. Avoiding heavy upfront costs to verify the trust as well as upfront costs for reserving resources instead availing of all the benefits of pay-as-you-go models which allow for better financial planning and better utilization of paid resources, as well as agility due to being able to pivot and change from

- **Scalability:** Easily scale resources in a flexible manner, efficiently allocating resources and paying only when using them.
- **Resilience:** Due to robust recovery solutions and globally distributed data centers, everything is available at a moment's notice. Trusting these features and their availability when needed offers a huge piece of mind and a significant reduction of resources required to achieve the same.
- **Global reach:** With every business being on a level playing field with every other, it is possible to trust that you can target and provide access to your resources (all underpinned by zero trust) to any customer anywhere in the world.

Careful consideration then needs to be given to address the negatives of trust:

- **Security:** Trusting a cloud provider with *the keys to your kingdom* is a huge ask. That is one further reason there are relatively few such providers. We can trust Google, Amazon, and Microsoft, that they will sufficiently protect these keys, because their reputation is at stake with each (even the smallest) customer, and any issues will be reported high and wide. Still, data breaches, insider threats, and vulnerabilities across the provider's stacks are issues to be addressed.
- **Control:** Sometimes with certain services or with certain aspects of cloud providers day to day operations, we might not have all the control over data and infrastructure that we had on-premises where you could if the worst comes to worst – **yank** the cables or do something equally desperate. It might seem crazy, but at least you had that option.
- **Reliability:** Downtime is guaranteed in any cloud. You can architect to protect against it, but even while you remain available, a certain number of services or resources will be down for you. That is the basic premise of the cloud, built on shared resources and resources available at a massive scale. Consider that even with 0.01% of yearly failure rates of disks, in a pool of 25 million disks, that means 250 thousand will fail. And 0.01% is not the yearly failure of disks, it is much worse. To prevent these failure rates without data, we trust (but verify) that the cloud providers make half a dozen or more copies to isolate us from this, but we also plan ahead for when we need to recover some lost data.
- **Lock-in:** Vendor lock-in is always happening. No matter which library, service, application, or provider you use there will be some vendor lock-in. This makes it a challenge at best to switch in case you need to. So we trust the cloud providers to keep prices aligned and affordable, services available and data durable, but we also plan for when that is not the case.
- **Privacy:** Concerns about data privacy control and management and jurisdictional differences between the US, EU, and other markets are valid concerns and need to be addressed as well. Data privacy is handled with the procedures outlined in each provider' documentation, and we trust and accept it, but the jurisdictional

> differences are a concern all on their own as how and where data is kept and who has access to it from the view of the provider, the governments and you and your organization as well as your customers.

BearingPoint has developed the "Cloud Trust Pyramid" […] as a framework to [analyze] the criteria required to build and manage trust as a prerequisite for transferring services to the cloud.

~ BearingPoint

Figure 3.1 illustrates the cloud trust pyramid:

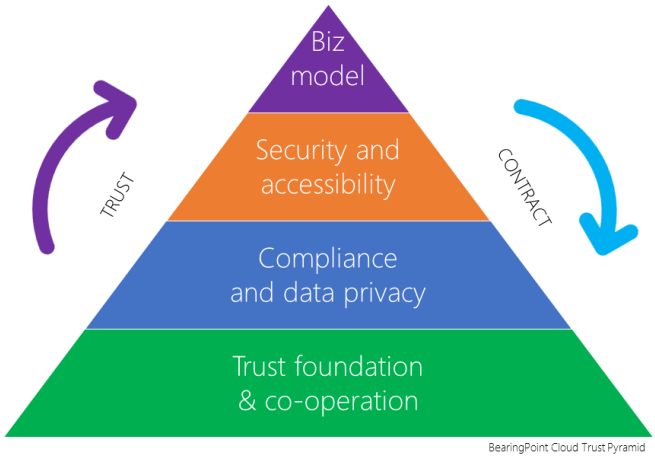


Figure 3.1: *Cloud trust pyramid*

All of the previous examples are also where all cloud providers are upfront with you and specifically asking you and your organization to not place too much trust in them.

They are specifically telling you to manage and double-check security; manage and retain control over your resources; and even warning you that every resource in the cloud can and will fail and you need to be ready and architect for it.

This is where shared responsibility comes in.

Shared responsibility

You might have seen the following figure already in the previous chapter, it was presented to showcase the differences between the IaaS, PaaS, and SaaS models, and here it is presented to show you the concerns of trust and the shared responsibility of each of the layers:

On-Premises	IaaS	PaaS	SaaS
Data	Data	Data	Data *
Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime
Middleware	Middleware	Middleware	Middleware
OS	OS	OS	OS
Virtualization	Virtualization	Virtualization	Virtualization
Servers	Servers	Servers	Servers
Storage	Storage	Storage	Storage
Networking	Networking	Networking	Networking
Data Center	Data Center	Data Center	Data Center

Data * This is a shared model between data stored in SaaS and your own data

***Figure 3.2**: Differences between what you manage (dark shade) and what the cloud provider manages (light shade)*

Shared responsibilities in cloud computing refer to the distribution of security and management responsibilities between the cloud providers and you and your organization. In this model, collaboration is needed to ensure that the environments and workloads remain secure. Each party takes on specific responsibilities, roles, and tasks.

So, referring to the above figure, the cloud provider is always responsible for:

- Securing the underlying cloud infrastructure – servers, electricity, networking cables, and data centers themselves (including physical security). This is not something you and your organization can secure, so you must place sufficient trust that the cloud provider is doing this correctly (and verify through the documentation they provide and the independent contractor's verification processes).
- Securing the network, storage, and servers as shared services between providers' customers so that they remain isolated between them. Again, this is not something you or your organization can do, as you should not have access to any other customers' environments.
- Securing the virtualization layer, including things such as operating systems, runtime, and middleware components as well. This is where the IaaS model allows you and there is a requirement on you and your organization, that is, there is a shared responsibility between the provider and your organization.
- Securing the applications and services in the case of the SaaS model. In this case, you are using the applications and services as provided, trusting the mechanisms deployed for your protection (for example, MFA, username and password, API keys). In the case of a PaaS model where you are building the applications and services, you and your organization have a shared responsibility to secure these.

What you and your organization may be (and usually are) responsible for in this shared responsibility mode is described as follows:

- Securing your environments and workloads, where the cloud provider handles securing the underlying infrastructure, you need to ensure sufficient security of your environments and workloads. Cloud providers may and will, in some circumstances alert you of the practices you are violating and that you should consider improving, but they may also happily allow you to deploy your workloads in an insecure way if you so choose (for example, the provider cannot know if your website is intended to be accessible for your employees only or you intended to publish it to the entire world).
- Securing your or your organizations' or your customers' data and identities is always your responsibility, as any on-premises resources as well, and the cloud components you control depending on the services you use (for example, the provider cannot know an employee has left the company and therefore their account should be locked/deleted).

In the traditional on-premises model, many security responsibilities should have been fulfilled by you and your organization, but they may not have been due to limited resources at your disposal. In the cloud approach, however, a lot of the responsibility shifts to the cloud provider or is shared between the provider and yourself, allowing you to do more with fewer resources.

You can and will make mistakes, and the provider will alert you for a lot of them, but if you do not act on those prompts (for example, you left your storage account publicly accessible), then you and your organization have failed in your part of the shared responsibility. Quite a few complaints aired publicly on blogs and social media are indeed a failure of the organization to address their part of the shared responsibilities around security.

For example, if someone steals your data from a cloud provider - it is likely their fault, but if someone steals the data of your customers that you host in the cloud – it is likely your and your organizations' fault.

It is well worth stressing this, you must address your shared responsibility with regard to security, and avoid thinking that you are secured by default in any model. Even in the SaaS, you are still responsible for your own password! So, always plan for and address your security responsibilities.

Now, as we move on to the next sections, also remember that all the terms, definitions and examples above only deal with 2 parties:

1. Your chosen cloud provider, and
2. You and your organization

Now we need to broaden this scope to all the additional parties that participate in this relationship. We introduce you to - the supply chain.

Supply chain

The supply chain is a complex and above all dynamic system in which different entities play distinct roles in ensuring the flow of goods and services to efficiently meet demand. It refers to the interconnectedness of organizations, resources, activities, and processes in the production, distribution, and delivery of goods or services to end consumers.

In the specific case of security and cloud security, software development and deployment security refers to the complex and dynamic systems of various networks, hardware, software, applications, and services involved in the software delivery process, everything from engineering through distribution and maintenance, to monitoring and optimizing the systems themselves. Every entity and every process that contributes to the end product is a part of it.

It includes hardware components, everything from the underlying infrastructure to IoT devices and end-user devices. It includes the development of software, be it creation, testing, or distribution, as well as operating systems and any firmware on the devices themselves. It also includes all third-party components, either from cloud providers themselves, from vendors, or from other sources such as open source. It includes distribution and logistics, production (local and global), transportation (physical or digital), and delivery of all the components to end-users. Cloud providers, library providers, communities of developers, managed service providers, and IT support companies – are all a part of it. Vendors are also a part of it, as it is very unlikely that you and your organization will produce everything needed for the end product yourselves. A huge part of managing the supply chain is risk management at every level mentioned – identifying and mitigating security risks throughout the chain is essential (for example, malicious or neglectful tampering, counterfeiting at all levels – parts of wholesale, vulnerabilities included by suppliers). The supply chain also encompasses all the practices that ensure security at every stage – everything from secure coding through vulnerability testing to secure configuration management and, eventually, any patches and updates.

The basic tenants we are trying to protect through supply chain management are:

- Security
- Availability
- Integrity
- Confidentiality
- Privacy

In this way, it closely aligns with SOC 2 certification. Developed by the **American Institute of CPAs** (**AICPA**, Certified Public Accountants), SOC 2 defines criteria for managing customer data based on five trust service principles — security, availability, processing integrity, confidentiality, and privacy. We will not go deep into SOC 2 in this book, but just remember that it exists.

Vendor selection

In your and your organization's supply chain, there will be many vendors. We all need products and services that we just do not produce. Therefore, we need vendors to supply them to us.

Do we want to use just any vendor? Of course not.

Sometimes governments and international organizations will mandate the use of a particular vendor or vendors or discourage or forbid the use of others – these are for geopolitical reasons and other legal reasons, and we will not go deep into this, other than to say that once you start working with vendors, it is important to seek out legal advice on the vendor selection process.

Assuming that goes well, we are now confronted with a few vendors, almost never with just a single one. Now we need to figure out which is the best, the most cost-effective, and which vendor is secure enough for our requirements.

We are now in the domain of vendor assessment and eventual selection. The importance of conducting a thorough assessment of vendors (also suppliers and third parties) cannot be stressed enough.

The vendor selected must align with your and your organizations' security practices as well as any standards you want or need to adhere to.

Here are some (you will need to tailor the criteria for your organization) criteria for vendor selection:

- Reputation and record of accomplishment
- Quality of produced products and services
- Cost and pricing structures
- Financial stability
- Technical expertise
- Support options and capabilities
- Security
- Data privacy
- Scalability and flexibility
- Communications
- Customer service
- Integrations and compatibilities
- Short or long-term relationship
- Partnership potential

Not only must you get the information to be able to assess these, but you also need to verify it for yourself (and/or through other means, for example, through membership in similar or complementary communities and/or organizations).

Risk management

Exploring strategies for identifying, assessing, and managing risks is important on every level and throughout the supply chain.

Here are some (again, tailor them to your needs) strategies for effective supply chain risk management:

- Map an end-to-end supply chain for your organization.
- Conduct risk assessment and identify and prioritize risks based on severity and likelihood.
- Diversify suppliers and vendors to mitigate the risk of any single supplier or vendor becoming a single point of failure.
- Consolidate suppliers and vendors if there are too many to manage.
- Plan for every conceivable contingency, prioritize based on severity and likelihood, and implement those plans.
- Plan for business continuity in every eventuality (for example, a global pandemic), but make those plans as general as possible rather than target specific scenarios. A broader business continuity plan that can be implemented for more scenarios is better than focusing on a plan that only mitigates one specific scenario.
- Observability is everything in cloud computing and in supply chain management, so invest in observing and tracking events with your vendors and the areas they do business in. This will give you actionable insight into the above risk management plans.
- Establish contractual agreements that define these risks and allocation of funds to address them between the contractual parties. This may seem like a legal issue only, but as it pertains to technology and the supply chain, legal may not be addressing your specific issues so you must collaborate.
- Obtain insurance (for example, cybersecurity) that will help you with some of the impacts but will also force you to adhere to their best practices in order to even be eligible for insurance.
- Exercise different scenarios and perform simulations on all levels to play with (imaginary) supply chain disruptions and address them as real. You will discover many issues in your plans this way.

Cybersecurity best practices

Sometimes, there is already a readily available solution (a pattern), to help you address issues. In cybersecurity, there are many best practices you can readily adopt (mind how they integrate into and affect your organization) without reinventing the wheel.

There are a lot of these practices we have gone through and will go into in this book.

Some of the more obvious ones everyone should be adhering to are:

- Multi-factor authentication.
- Encryption of data in transit and at rest.
- Secure coding practices in your languages of choice.

And some more involved, but nevertheless still very obvious and readily available:

- Regular security audits
- Red/blue/purple teams and practices
- Site reliability engineering practices

And never forget the individual cybersecurity best practices to keep your people and your organization safe such as the use of password managers and phishing awareness training.

Data protection and privacy

We all have data that we have to protect. It could be your organizations' data, data collected from your users, or any other data. It must be protected. This data must be protected throughout the supply chain, and you need to be involved at every stage. Follow these given practices:

- Encrypt all data
- Enforce access controls
- Comply with data protection regulations
- Delete data as soon as it is no longer needed

Third-party due diligence

Examine all the processes around due diligence on third parties to ensure you are constantly on top of any issues – which means due diligence is performed continuously, not just at the beginning of the relationship.

Also, ensuring everything in this section that applies to you and your organization, also applies to third parties. If you require data encryption at transit at rest, require it from your vendors. If you delete data when no longer needed, ensure there is a requirement to do so for your vendors as well. And verify that they are doing as agreed.

Physical security

The security measures we have been discussing in this book are almost all about the digital world but remember – there is a physical world out there as well. And no amount of digital security can help if the attacker has physical access to your infrastructure.

In the shared responsibilities we discussed above, your cloud provider will ensure the physical security principles are being adhered to on their end, but you must do your very best on your end as well.

Protecting important data, confidential information, networks, software, equipment, facilities, company's assets, and personnel is what physical security is all about.

Physical penetration testing can help you identify and rectify any gaps you may have there. The clean desk and conference rooms policy is a huge help as well.

If you have on-premises data centers this of course all applies to those as well.

Transparency

There is no security by obscurity but consider the pros and cons of disclosing information. You and your organization need to be very careful what you disclose but at the same time do not go too far in the opposite direction and become too opaque. There is something to be said for transparency.

The benefits of transparency are across the supply chain, allowing organizations to trace products and services, components, and materials, to their origin. Sometimes this is a legal requirement as well. Here are some of the benefits:

- Supplier disclosure and information sharing are the best paths to transparency.
- Tracing and tracking via barcodes, RFID tags, or blockchain is a great way to achieve the goals of transparency as well.
- Certifications and audits will keep you and your vendors honest.
- Public reporting and transparency to your customers a great differentiators among your competitors.
- A code of conduct for yourselves and vendors across the supply chain ensures transparency and improved security if implemented and audited.
- Collaboration with advocacy groups is another way to keep you honest and engaged with security and privacy initiatives.

Incident response and contingency planning

Incidents will happen. Hopefully, they are all minor events rather than large worldwide brand-perception-changing events for you.

The patterns are known, so here they are for you:

- Establish a dedicated incident response team.
- Establish a detailed incident response plan.
- Conduct regular training and drills.
- As always, observability is key, so, have in place real-time monitoring and alerting.
- Classify and prioritize incidents.
- Learn and execute effective communication and reporting.
- Back up all data and test recovery regularly.
- Contain incidents and preserve forensics for law enforcement.
- Conduct post-mortems and analyze what went wrong, what went right, where you got lucky, which lessons you have learned, and what behavior or process will improve so that this type of incident will not happen again.
- Collaborate with vendors across the supply chain and with external partners such as law enforcement, cybersecurity, and industry peers.

Compliance

Regulatory compliance is closely related to supply chain management because it does not apply only to your organization, it applies across your organization's supply chain as well.

If your organization must or chooses to comply with an ISO or SOC framework, usually this is a recursive requirement for all your supply chain vendors and partners to comply as well. Cloud providers are usually compliant as well, to ensure that your organization can use them – due to this recursive requirement.

There are many other voluntary compliance frameworks that have similar requirements. Then there are many legally required compliance frameworks, such as GDPR in the EU. If you are in the healthcare industry, you might need to comply with HIPAA (in the US), and if you are in the e-commerce industry, you might need to comply with the **Payment Card Industry Data Security Standard** (**PCI DSS**).

There are requirements from all of these that directly impact which cloud providers and which cloud services you can use, as well as which software features you must have to support the requirements.

Assessing these and acting to enforce and comply with them is a part of managing your security and privacy posture.

Employee training

Educating employees and stakeholders about supply chain security risks, best practices,

The key reasons are:

- **Risk awareness and mitigation:** By educating everyone on supply chain attacks, everyone becomes more aware of potential threats.
- **Social engineering defense practices:** Preventing disclosure of information or bypassing role-based and individual-based access controls is the main goal.
- **Promoting a security culture:** Every employee and everyone else in the supply chain must be aware that any breach of processes may result in a supply chain occurring and/or propagating through the supply chain.
- **Compliance and regulations:** Adhering to compliance and regulation can only be achieved if, at every level in the organization, requirements are gathered and adhered to.
- **Preventing insider threats:** Be it from deliberate maliciousness or accidental neglect – the result is usually the same, so preventing both with a set of security measures is equally important. It also gives peace of mind to employees who are more protected from damage resulting from accidents and mistakes.
- **Improving the supply chain resilience:** With an aware staff and security mindset you improve resiliency across the supply chain.
- **Competitive advantage:** Having a more resilient supply chain which results in fewer disruptions so business as usual can continue is a sure competitive advantage. Even a single quarter dealing with an incident can put the business results back by years or defer new innovative solutions so they are late to market.

Real-world examples

Some real-world examples of cybersecurity attacks are discussed as follows:

- SolarWinds supply chain attack was a major cybersecurity incident that affected numerous organizations, including government agencies. The attackers compromised the software supply chain by injecting malware into SolarWinds' Orion software updates. This allowed them to gain access to the systems of SolarWinds' customers, leading to a data breach.
- NotPetya ransomware attack targeted a Ukrainian accounting software MeDoc. The attackers inserted the ransomware into a software update, and when organizations downloaded the update, it spread through their networks. This incident highlights the importance of supply chain risk management to prevent such attacks from propagating through interconnected systems.
- Capital One experienced a data breach in which an attacker exploited a misconfigured **Web Application Firewall** (**WAF**). The breach exposed the

implementing cybersecurity best practices like proper configuration management and regular security audits.

- The Cambridge Analytica scandal involved the unauthorized harvesting of Facebook user data by a third-party app. This incident raised concerns about data protection and privacy breaches within the supply chain of data-sharing partnerships.
- Macy's experienced a data breach due to a third-party vendor's compromise. The attackers gained unauthorized access to customer data through the vendor's systems, highlighting the importance of thorough due diligence on third-party partners.
- Thailand's floods disrupted the supply chains of various industries, including automotive and electronics. Many manufacturing facilities were submerged, causing considerable damage and delays in production. Proper physical security measures and location assessments can help mitigate the impact of natural disasters on the supply chain.
- Nestlé, in its commitment to supply chain transparency, has implemented a blockchain-based system to trace the journey of its products, including milk, from farms to consumers. This transparency allows consumers to verify the authenticity and sustainability of their products.
- Colonial Pipeline faced a ransomware attack that led to a temporary shutdown of its operations. The incident highlighted the importance of having a robust incident response and contingency plans to recover from cyberattacks and ensure the continuity of critical services.
- The European Union's **General Data Protection Regulation** (**GDPR**) imposes strict regulations on data handling and privacy. Organizations dealing with EU citizens' data must follow GDPR requirements, impacting supply chain processes involving personal data.
- Target experienced a data breach in 2013 due to a phishing attack on one of its third-party vendors. The attack stemmed from an employee clicking on a malicious link in a phishing email. This incident emphasizes the importance of ongoing employee training and awareness to prevent similar attacks in the future.

As you can see, once an incident happens, it can lead to worldwide visibility and a reputation loss as well as business disruptions as the business deals with the incident. Learning from the experience of others is the least costly path to your experience.

Software development

Software development is also impacted by supply chain issues, in – maybe (arguably)

- **Dependency on third-party libraries:** Reliance on third-party libraries, frameworks, and modules accelerates development, but at what cost? Each and every one must be vetted and approved. From a security standpoint, it must be secure and robust and in agile development. This is because vulnerabilities in these dependencies can affect all the services that use them. In some organizations, the number of services can be in hundreds and thousands.
- **Sourcing and licensing:** As above, ensuring reputable and trustworthy vendors are the ones providing these libraries is not a trivial pursuit.
- **Collaboration and source control:** Ensuring code repositories and version control systems are accessible and secure is paramount for teams that are now increasingly globally dispersed.
- **Hardware and toolchain supply issues:** Relying on the supply chain for hardware and toolchains (for example, compilers, and debuggers) is necessary when working with hardware components (for example, in embedded software), and ensuring the integrity of these is paramount.
- **CI/CD:** Automated development, testing, and eventually deployment all rely on the fact that the artifacts produced are produced and kept secure.

Real-world examples

Some real-world examples of cybersecurity attacks related to software development are as follows:

- The Equifax data breach was caused by a vulnerability in the Apache Struts framework, a third-party library used in Equifax's web applications. The failure to promptly update the library with a security patch allowed attackers to exploit the vulnerability and access sensitive data.
- Oracle vs. Google lawsuit revolved around Google's use of Java APIs in Android without proper licensing. The dispute highlighted the importance of correctly sourcing and licensing software components to avoid legal repercussions.
- GitHub, a popular code collaboration platform, experienced a security incident where attackers gained unauthorized access to a number of repositories, emphasizing the need for secure code collaboration and version control practices.
- The supply chain for hardware components used in IoT devices can impact software development. The Mirai botnet infected thousands of IoT devices with malware by exploiting vulnerabilities in the hardware supply chain.
- A security incident at Codecov led to the compromise of the CI/CD pipelines of multiple organizations. Attackers modified the tool's Bash Uploader script to exfiltrate sensitive data during the software build process.

Software deployment

Software deployment is also affected by supply chain issues. They are:

- **Deployment environments:** Dependencies on deployment environments, cloud services, data centers, physical locations, software-defined networks, etc. must be managed, and ensuring these are all secure and reliable is crucial.
- **Patch and update management:** While the shared responsibilities model comes heavily into play here as the cloud provider does these for most services depending on your model for deploying applications and services, it nevertheless is still up to you and your organization to patch and update your own applications and services. Doing this securely at scale and across geographies is a logistics issue. Ensuring these patches do not introduce new vulnerabilities and can quickly deploy patches when they do is a full-time job in and of itself.
- **Distributing and installing software:** While software may be installed at times and on devices you have no control over, it is still up to you to ensure security and reliability. If you think cloud services are difficult, services not connected to you are even more difficult to debug and require elaborate setups to test well (for example, networking software on commercial ships which usually do not have the best connectivity across all seas).

Real-world examples

Some real-world examples of cybersecurity attacks related to software deployment are as follows:

- Amazon Web Services experienced a major outage due to a typo in a command during a routine server update. The incident highlighted the importance of managing deployment environment dependencies carefully.
- The Supermicro hardware supply chain was affected by the insertion of malicious chips. The incident raised concerns about hardware security and supply chain integrity.
- The WannaCry ransomware outbreak exploited a vulnerability in Microsoft's Windows operating system that had an available patch. Many organizations failed to apply the patch promptly, leading to widespread infections.
- Attackers compromised the download links of the popular HandBrake video converter software, distributing a version containing malware. Users unknowingly installed the infected software, highlighting the importance of secure software distribution.

Conclusion

In this chapter, you have learned all about trust, why it is essential for society and for cloud computing, and why and how it can be broken. Then we discussed how to address trust issues, the zero-trust model, and how it allows you to **bake-in** security at every level.

The shared security model is something you should now understand and how it helps you narrow down your responsibilities and how it allows you and your organization to focus on adding value in security terms to your own part of the shared responsibility model.

Managing the supply chain and mitigating risks is the next thing we discussed. You should have learned all the different ways that a supply chain attack can impact your organization. And finally, we discussed software development and deployment practices with regard to supply chain attacks.

In the next chapter, we will start discussing different ways of securing particular types of services, specifically, we will start with Securing Cloud Infrastructure and Identity and Access Management.

Questions

1. What is trust in cloud computing?
2. What is the best thing about trust in cloud computing?
3. What is zero trust?
4. How does zero trust enhance the security posture of an organization?
5. How does zero trust protect against neglect?
6. How does zero trust protect against human error?
7. Which considerations you must consider when placing your trust in the global hyperscale cloud providers?
8. What is the shared responsibility concept?
9. How does shared responsibility explicitly affect security in the cloud?
10. What is a supply chain?
11. What are the security implications of managing the supply chain?
12. What are the most important requirements for vendor management?
13. How to manage risk effectively? Hint: identify and prioritize!
14. What cybersecurity best practices should you personally follow?
15.

16. When should data be deleted?
17. What is your organization's plan for incident response? Can you find it in less than 5 minutes? Have you performed an exercise in the last quarter?
18. Explain how to prevent a supply chain attack from reaching your organization.

References

- Warren Buffett's talk at the University of Georgia:
  https://twitter.com/george__mack/status/1685293755916591104
- SolarWinds supply chain attack story:
  https://www.wired.com/story/the-untold-story-of-solarwinds-the-boldest-supply-chain-hack-ever/
- Codecov breach:
  https://cycode.com/blog/the-codecov-breach-development-infrastructure-is-the-weakest-link-its-now-rapidly-being-exploited/

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 4
Securing Cloud Infrastructure and Identity and Access Management

Introduction

In this chapter, we will discuss and learn about securing the cloud infrastructure.

We will delve under the surface of asset protection and resilience, specifically around the areas of physical location, legal jurisdiction, data centers, data protection at rest and in transit, data sanitization and anonymization, equipment disposal, and physical resilience and high availability - both as a shared responsibility and as a direct responsibility of the reader.

We will also discuss operational security with all its implications - an often overlooked part of cloud security.

We will take it a step further than most with **Just-In-Time (JIT**) access and how that alleviates all sorts of escalation of privilege issues, but as with everything, layered security is key.

Personas will be discussed as well as why it does not really make sense for everyone in the organization to be treated equally (in the case of access to resources only of course).

Diversity, equity, and inclusion are an absolute necessity when it comes to building a well-rounded, healthy, and happy organization. Still, diversity, equity, and inclusion are antipatterns regarding access controls.

Structure

The chapter covers the following topics:

- Securing cloud infrastructure
- Business continuity
- Encryption
- Operational security
- Data sanitization and anonymization
- Data masking
- Equipment disposal
- Understanding IAM and RBAC
- Google Cloud Platform IAM
- Azure Privileged Identity Management

Objectives

By the end of this chapter, you will understand the concepts around securing cloud infrastructure. You will learn what are the patterns to achieve security through managing identities and access to resources based on individual roles. You will know about the principles of least privilege, zero trust, role-based, and JIT access and how to use them, and why they are essential for cloud security.

You will learn how to start doing operational security and how to apply the lessons learned to your and your organization's workloads in the cloud.

Securing cloud infrastructure

Before getting into the principles and patterns that will help you and your organization keep your workloads secure, we need to discuss physical security briefly.

Your organization may have a diverse set of physical locations – offices, data centers, retail outlets, and others, so does the cloud provider. What matters to us in this section is the physical security of data centers. The good news is that you should not worry too much about any of that, but – you do need to know what considerations around physical security are in place so that you can explain why everything you do from software-based security is layered on top of the existing layer of physical security.

The cloud provider carefully selects data center locations to mitigate global and local environmental risks such as flooding, extreme weather conditions, seismic activity, and

more. Then, they are designed in such a way to provide redundancy in case of any failure events that do occur, by, for example, moving traffic away from affected areas.

Business continuity

You and your organization should have a business continuity plan. The cloud provider has it as well. These are plans that evaluate and plan for mitigating or lessening the impacts of disruptions.

As always, in business continuity, the pattern is a general plan. This is a plan that works for a wide variety of scenarios and is better than a specific plan that addresses one specific scenario. This plan is continually updated and improved upon. For example, the pandemic response has been included in the plan by many organizations.

Employees and third parties have restricted access, usually to pre-approved employees, after applying and supplying a valid reason or a business justification. Access is granted for a particular time, to a specific individual, and to a specific area. This access, along with the actions performed, is monitored and audited. Monitoring is also done 24/7 for general security as well.

Surveillance is done via CCTV cameras, various sensors, and on-site personnel. Entry and exit points are covered by staff and detection systems, both to prevent unauthorized entry and bringing unauthorized equipment in, as well as taking unauthorized equipment out. Intrusion detection systems react to changes in the environment and raise appropriate alarms when necessary.

Managing assets is done through automated tools. Assets in this case are everything in the data center, including computers, networking, and storage, but all other assets as well, for example, diesel engines that serve as backup power in an emergency.

Speaking of power, redundant power is supplied to data centers. Usually from multiple providers. Humidity and temperature controls are in place and sensors that react to changes in either are deployed throughout the data center. Automatic and manual fire suppression systems are in place to prevent a fire outbreak and containment in case it does break out. Water leakage detection is similarly detected and dealt with.

Equipment and the environment in data centers are managed and maintained. Preventative maintenance is performed when needed and whenever possible, before an issue occurs.

Third parties are accredited for validating security and business continuation plans are in place to continuously verify compliance.

This was an abridged version of what AWS claims are (and supplies evidence for) the controls in place for their own data centers:

- AWS Cloud Security – Our Control
  https://aws.amazon.com/compliance/data-center/controls/

Everything mentioned here is something that the cloud providers do for you, but you also need to consider your own workloads in the cloud. And this is what we will discuss more about further in this chapter.

Everything done by the cloud provider for the physical security of data centers, you must do the same for your cloud environment or environments, such as:

- You should have a business continuity plan.
- Select the physical location for your workloads (data center location or preferably locations).
- Plan for redundancy of your workloads (when and as much as necessary).
- Plan for the capacity you will need and the availability of services.
- Restrict access for your employees as well as third parties (in your case not physical but rather digital and logical).
- Monitor your workload and act on alerts (automation is key here).
- You need to have intrusion detection deployed and react to its findings in real-time.
- Manage all your assets (knowing all of them and how they are managed, owned, utilized, paid for, and when and how they are decommissioned).
- Plan for failure. As we have discussed in the previous chapters, the cloud is built on commodity hardware and shared hardware, and any and all of its services or components can and will fail eventually.
- Plan for disaster recovery.
- Finally, you likely (if you have external customers, if you are in a regulated industry, if you process payments or personally identifiable data, or health data, etc.) as well need some third-party attestation for your workflows and your organization (ISO, SOC, etc.) and you need to plan for it, evidence your compliance and continuously improve.

As you can see, your digital and logical workloads need the same protection as a data center needs on a physical level. We will discuss how to achieve this in this chapter and the following ones.

Encryption

We will discuss more on how to do encryption correctly in chapters about networking and storage, in-transit, and at-rest. For now, let us make sure we agree on the basics.

Encryption is a process (nowadays dependent strictly on mathematics and mathematical algorithms) that we use to protect data. We protect

compromised, or even just read. This process involves modifying data in such a way as to scramble it, and only those in possession of an encryption/decryption key can unscramble it back to its readable form.

Two well-defined keywords to remember around encryption are **plaintext** and **ciphertext**. Plaintext is the usual text, such as the one in this book, and ciphertext is the text after encryption has been performed on it.

Encryption had quite an early start – back in the ancient times of Egypt in 1900BC and other civilizations of the era. It is a fascinating subject that we will not go into here.

- To dip your toes into the history, start at the encryption page of Wikipedia to get some references you can then research further:

  https://en.wikipedia.org/wiki/Encryption

Encryption is the process of generating ciphertext from plaintext and decryption is the reverse process, generating plaintext from ciphertext.

Generally (without complicating it further), there are two types of encryption algorithms: symmetric and asymmetric.

- Symmetric encryption algorithms use the same key to encrypt and decrypt. In practical terms, this means that if you want to encrypt a message and someone else needs to decrypt it – you need to share the encryption key. In cloud computing, this gets complex very quickly when you consider that you need to distribute this key to potentially millions of users. It is not easy to share keys without compromising them, there are too many ways to acquire a copy of the key easily.
- Asymmetric encryption algorithms, also known as public-key cryptography algorithms use (at least) two keys. The first one is public, which is used to encrypt a message for a particular recipient, who can then use the second one. The private key is to decrypt any messages sent to them. This makes the vulnerability of single key algorithms moot, because with asymmetric encryption it is desirable to share the public key with anyone who wants it, while the reader has their own key – the private key to decrypt any messages.

Encryption relies on mathematical algorithms that are easy to do in one direction (for example - multiplication) and very difficult in the other direction (for example - factorization).

Encryption is important because without it none of our daily functions could be performed safely and reliably, especially remotely – through the internet.

Our contexts for encryption are based on the need to encrypt data in-transit, that is when traveling through computer networks) and at-rest, that is, stored in place. Due to regulations, encryption is also a regulatory and/or lawful requirement.

To recognize the value of encryption, consider these functions of encryption:

- Confidentiality ensures that the data is in fact confidential and/or secured and/or secret. You may hear any of these terms. This means that the encryption algorithm used is secure and only those with the key or keys perform the decryption function.
- Integrity ensures that the origin of the data is known or verifiable. By using the specific private key to decrypt data, you can also know who is sending the message, if there is only one other who has the corresponding public key. There are also ways to sign each piece of data by the sender so that you as the recipient know who sent it.
- Authentication ensures that data has not been altered. Due to how certain algorithms for encryption and decryption work, changing a single character of a message results in a message that cannot be decrypted anymore.
- Nonrepudiation ensures that the sender cannot deny sending the data. Again, there are many different mechanisms to achieve this.

As with a lot of cybersecurity and cloud security concepts we will discuss, encryption can also be abused. Consider ransomware - pieces of malware that encrypt data and then demand ransom to provide the key for decryption. In order to be resistant and resilient to ransomware, Operational Security is important.

Operational security

Operational Security (**OpSec**) is a security and risk management process that works by classifying all data and ensuring its protection to the level of classification. Let us take an example. Public data is data that can be published, and anyone accessing it is allowed to do so. As a counter-example, personal health information should be kept between the user and those health providers that the user considers trustworthy enough to grant access.

As technology becomes more difficult to breach, attention turns to users of said technology. Adversaries will try and compromise users first rather than risk raising alarms attempting to compromise a computer system.

There is a term for this type of attack, called social engineering attack. Social engineering relies on tricking users into doing the hard work themselves. For example, it is more difficult (not always – choose your passwords carefully) to discover someone's password through a technological breach, than it is to call a user on the phone and pretend you need it, or their expenses will not be processed on time. There is a reason (other than regulatory) for all those security training courses you and your organization mandate for all your employees.

Being aware of social engineering attacks is the number one defense against them. Social engineering is primarily a psychological attack where the user is tricked into doing an action or giving away information – that will then come back to harm them or their organization. These attacks usually involve creating a sense of urgency and coupled with a solution that seems benign but bypasses or ignores security procedures. This leads to

receiving sensitive information that is then used in other similar attacks or attacks moving into the domain of technology again.

If at any point you think someone is trying to trick you or pressure you, do these two things at once:

- STOP communicating.
- VERIFY the request independently. From the example above, on receiving a call requesting your password to process your expenses, notify your HR department, your LEGAL department, or your MANAGER, and ask them to confirm the request.

Having learned about social engineering and having absorbed the information in hopes to protect yourself and your organization let us now focus on operational security in the context of cloud computing environments.

The reason we discussed social engineering first is that no amount of cloud operational security can protect from a successful social engineering attack, so consider yourself warned.

There are ways to further protect your cloud environments and we will go into some of them in this very chapter.

Before that, let us establish some ground rules for operational security in the cloud.

These are the general steps to minimize the risk of threats:

- **Identify critical information**: your organization's data must be known to you in order to protect it, so it is essential to have a complete catalog of it. What you do not know exists, you cannot protect. Once cataloged, it needs to be classified correctly into neat, distinct categories. Different categories will require different protection plans. You need to define the security parameters for each category.
- **Analyze threats and vulnerabilities**: Assessing potential threats and threat vectors (for example – state-level attack is a threat actor you need to protect against, and a vector for their attack might be an attack against your identity system). Continuously looking for (also known as threat hunting) threats is essential. This work is never done until the cloud workload is terminated. Vulnerabilities mitigation is another never-ending duty. As long as your workload is in the cloud, as long as there is code running, software running, libraries are used, and data is transiting or stored – vulnerabilities will keep rearing their head. Staying on top of them and patching them is essential.
- Assess the risk factors such as what is the likelihood of a risk occurring and what is the damaging impact (this is usually expressed in direct currency value). Once you have these two pieces of information you can assess what effort would be required to recover from each. It goes without saying that higher priority should be given to

high-risk and/or high-damage attacks. We are saying it anyway, and you should be saying it in any risk management meetings you and your organization conduct as this fact often gets ignored because high risk and/or high damage also might require a lot of effort to plan for.

- Last but not least of these is: do in fact apply what you have learned above in a series of countermeasures that address the risks. Do not just talk about it, do not prolong work unnecessarily, and always – be mindful of the business impact.

The MITRE ATT&CK® framework for cloud is a subset of the broader MITRE ATT&CK framework, focusing on tactics and techniques specific to cloud environments. It outlines how adversaries might target cloud platforms, services, and infrastructure. The framework is organized into pillars, each representing a category of attack techniques. Here is a brief definition of each of the MITRE ATT&CK for cloud pillars, along with a pictorial representation:



Figure 4.1: *Tactics and techniques representing the MITRE ATT&CK® Matrix for Enterprise covering cloud-based techniques*

- **Initial access:** This pillar covers tactics and techniques that attackers use to gain their initial foothold in a cloud environment. It includes methods such as exploiting misconfigured resources, compromising identities, and leveraging trusted relationships.
- **Execution:** The execution pillar focuses on techniques adversaries use to run malicious code or commands within cloud environments. This can involve abusing legitimate cloud management tools or executing malicious scripts.
- **Persistence:** Persistence tactics and techniques involve maintaining unauthorized access to cloud resources over an extended period. Attackers may create backdoors, manipulate permissions, or set up rogue service accounts to achieve persistence.
- **Privilege escalation**: In this pillar, techniques that adversaries employ to elevate

exploiting vulnerabilities, abusing misconfigurations, or gaining excessive permissions.

- **Defense evasion**: The defense evasion pillar addresses techniques that attackers use to avoid detection by security tools and mechanisms. They might modify logs, disable security controls, or employ encryption to conceal their activities.
- **Credential access:** This pillar encompasses tactics that involve stealing or obtaining credentials. Attackers can target users, applications, and services to gain access to valid credentials for unauthorized use.
- **Discovery:** Discovery techniques focus on attackers' efforts to gather information about the cloud environment, including its architecture, resources, and configurations. This knowledge helps them plan the next stages of the attack.
- **Lateral movement:** Lateral movement involves techniques that attackers use to move laterally (lateral meaning sideways). For example, from having access to a single VM to getting access to a user account because they discover it exists in the VM logs, enabling them to find out more and move laterally again within a cloud environment once they have gained an initial foothold. They aim to expand their access and reach valuable assets.
- **Collection:** The collection pillar covers tactics related to gathering data and information from cloud resources. This could include exfiltrating sensitive data, capturing configurations, or collecting credentials.
- **Exfiltration:** Exfiltration techniques involve stealing and transferring data from a cloud environment to an external location controlled by the attacker. Methods can include data copying, data sharing mechanisms, or leveraging legitimate cloud APIs.
- **Impact**: In this pillar, techniques that adversaries use to cause damage or disruption to cloud services, data, or infrastructure are discussed. This might involve deleting resources, altering configurations, or interfering with service availability.
- Learn more about tactics and techniques:

  https://attack.mitre.org/matrices/enterprise/cloud/

Data sanitization and anonymization

Encryption is a two-way process. We can encrypt data and we can decrypt data. Sometimes, however, we want that process to be one way, that is we want to change data in a certain way that it cannot be reversed.

To dispose of (some) data in the cloud environments, depending on the goal, we can use the data sanitization and anonymization processes.

If we get a GDPR removal request from a right-to-be-forgotten form (in the EU, individuals have the right to request their data be removed from the system) we can sanitize the data by completely removing this individual's personal data from our systems. We need to ensure all data is removed, and this is important – it cannot be recovered. So, this requires planning ahead of time on how to achieve this, especially in a multi-tenant, multi-user environment. And then, what do we do with backups we have already made that contain that data? If at any point we recover the data, we will also recover the data that was requested to be removed.

Wholesale removal of data should not be that difficult, it might require deletion from a few places, but all data can be removed and then it is – gone. If, on the other hand, we want to keep some data for analytics purposes or for AI training, we need to remove pieces of it.

This is where anonymization comes into play. We can use methods such as generalization, suppression, and perturbation to prepare the data for analytics. These ensure that personally identifiable information is removed so that data is still useful but is not any more personally identifiable. Be very careful here, as data can be easily de-anonymized in many cases that are considered permanently anonymized. So the anonymization process also needs to ensure this is not the case. Maintaining data utility while removing personal information is not a trivial process.

Data masking

In cases where we want to retain data but limit who can read it, we can use data masking techniques. For example, administrators of the cloud platform may, in some cases, be required to access databases containing personal data to perform maintenance on the data and on the database or to act on user requests to fix corrupted entries. Data masking allows us to set up rules on who can see data so we can have a situation where administrators have full access to the databases, and they see the columns in the database table, but the individual data in some of these columns is not displayed – for example - they can see the ID of an order, but they cannot see the full address where the order is to be dispatched.

Data masking is useful in a lot of access scenarios. Later in this chapter, we will go into identity and access management and role-based access controls, so in the context of each of the roles you should also look into data masking requirements for each role.

Data masking is performed differently depending on where the data is. Some databases have data masking features built in, and some mediums of storage – do not. In the latter cases, you may need to do additional work to ensure data masking – maybe at an API level to mask the data depending on the access requirements.

Equipment disposal

Disposing of physical equipment processes is important at the end of the lifecycle of any equipment. Cloud providers generally do not allow data

data centers intact. Disks are usually pulverized, shredded, and destroyed in various ways so that they guarantee data destruction – including magnets, presses, shredders, chemical baths, and various other ways of ensuring data is not accessible. Usually, disks only leave data centers in millimeter-sized chunks.

Disks are cheap, and ensuring data destruction is more important than recycling disks.

Understanding IAM and RBAC

Different cloud providers have different ways of managing identity and access. In this chapter, we will show how to perform these activities in **Google Cloud Platform** (**GCP**). The good news is that all the concepts are translatable between providers.

The reason we will vary cloud providers in our specific examples is to show that the concepts in this book and in cloud security are interchangeable. The bad news is that in order to do the equivalent actions in other clouds you will need to consult their respective documentation.

Before we go into the example, it is important to understand the differences between the topics of **Identity and Access Management** (**IAM**) and **Role-Based Access Control** (**RBAC**). These are sometimes used interchangeably, but there are distinctions between them.

IAM is a broad framework that encompasses policies, processes, and technologies for managing and controlling user identities and their access to the system, apps, and data. User authentication and verification is used to ensure the user is who they claim they are by using mechanisms like username/passphrase combination, biometrics, multifactor authentication, etc. IAM manages the lifecycle of identity – everything from creation to deletion from the system, including regular audits and management activities. Single sign on is usually a part of an IAM solution as well, enabling users to use their own identities created in their organization to access resources in the cloud environments. Individual management of passphrases and generally self-service around IAM is a must due to the ever-changing landscape of users and their requests.

RBAC on the other hand is a specific access control model within IAM that focuses on managing access to specific resources within the cloud environments and more broadly within an organization's footprint. Roles are defined based on job functions, teams, tasks, or types of resources. For example:

- Job function: Cloud architect,
- Teams: Cloud platform team,
- Tasks: Migration of on-premises resources to the cloud,
- Types of resources: Network administrators.

Permissions are sets of actions allowed for each role and assigning each user (or user group) to a specific set of permissions is crucial.

Permissions can also be granted to bots, meaning specific functions are performed automatically when certain conditions are met. One usual example of this is the automated backup requirement, where all business-critical resources are backed up regularly. Because we want to ensure this happens every time without fail, we usually automate these activities, as relying on users to do this would inevitably lead to mistakes by either backing up the wrong resources in the incorrect ways or forgetting to perform the function regularly.

This leads us to discuss the **principle of least privilege** and **zero trust** concepts.

Principle of least privilege

The principle of least privilege is a pattern in security and cloud security that states that granting users, applications or processes access should be done in a way that minimizes the level of access so that the user, application, or process can perform their authorized tasks but cannot perform any other activity that is not related to that task.

The idea here is to limit the damage that a rogue user, application, or process can do.

Minimizing the attack surface ensures this. In our example above, when backing up a database on a regular schedule, the process that performs this activity should have permission to back up the database but may not necessarily need to have read access to the database records or also have access to networking functions.

Remember lateral movement from the MITRE ATT&CK® Matrix? This approach of least privilege also mitigates against that because access is granted only to the specific resources and functions and not a broader set of resources and functions.

Well-intentioned employees can also do a lot more accidental harm if their role allows them broad access to cloud environments. For example, a cloud architect may require read-only access to ensure architectural decisions are respected, but do they also need write access to the entirety of the cloud environment? Or does a networking expert troubleshooting need access to the network segment they are working on, but do they also need access to backup and restore functions?

Compromised accounts' impact is also lessened by following the principle of least privilege. If an account of a user with administrator privileges is compromised the attacker can do a lot worse than if a user with backup-only privileges is compromised.

This pattern ensures that demonstrating to auditors which people, applications or processes have access to is possible and trivial and will keep you and your organization compliant based on regulatory standards.

Monitoring and detection will have fewer false positive alerts on lateral movement, so this

Zero trust

Let us now contrast the principle of least privilege with zero trust. Zero trust is an approach to cloud security that ensures that users, devices, and locations are not automatically trusted. All users, devices, and locations are verified and authenticated. Networks are divided into smaller micro-segments with limited access to each – reducing lateral movement. The least privilege pattern is used for every access request. Policies are evaluated dynamically and in real-time. For example, a user may have access from Ireland because that is where they normally work from, but if the request is coming from Croatia, it is scrutinized more (perhaps an MFA is triggered again and/or the IT team is alerted so they can verify) or even outright banned.

Zero trust also hinges on encryption being in place for every resource and automation utilized to enforce access control, respond to threats, and adapt to the ever-changing conditions (such as the fact that employees will travel for work, so they may require access from various locations).

JIT access adds an additional layer of protection. No access to resources is always granted, rather – access to resources must be requested, approved, and then granted for a defined period. Because it requires an MFA and an approval process, this approach further limits damage from compromised accounts as, by default, no access is granted. So, even if the account is compromised, there is no access to resources.

In the last parts of this chapter, we will take a look at how this would work on two examples in two different cloud provider environments.

Google Cloud Platform IAM

Now let us see how IAM would work in practice, using GCP's Identity and Access Management features.

For this exercise you will need:

- A GCP account:
  https://console.cloud.google.com/getting-started
- Access to the GCP Console:
  http+s://console.cloud.google.com/
- An empty/new project created

Let us browse the users, roles, and permissions and set some up:

1. Start by accessing the GCP console at **https://console.cloud.google.com/**.

2. Select the menu item IAM **&** Admin, as shown in the following screenshot:

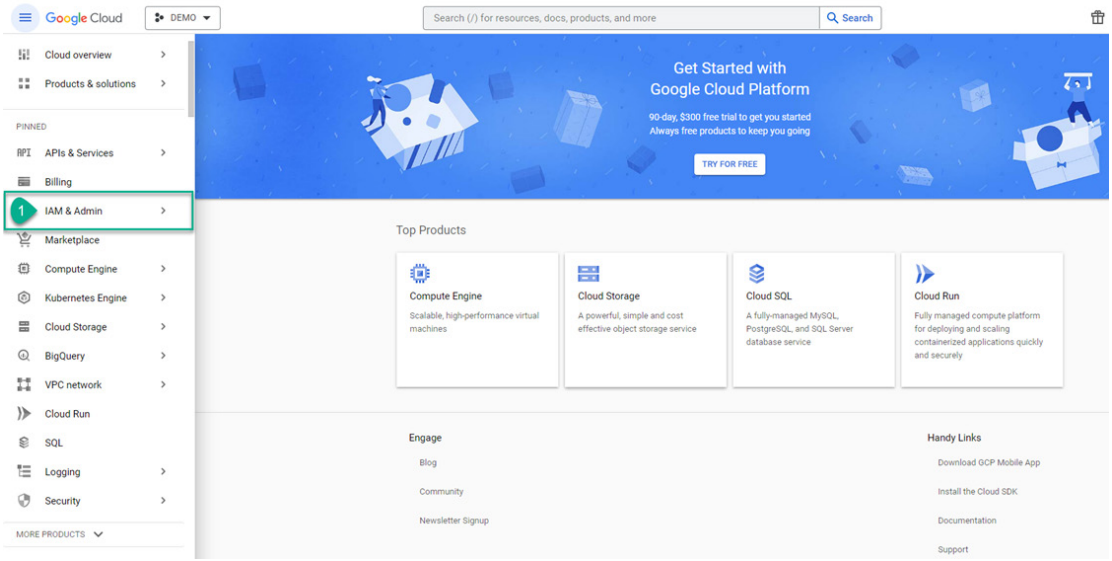


Figure 4.2: *GCP Console*

3. Check and see the list of principals that have access. If you are doing this in an empty project, it is likely you will only see one, yours. In an organization-type-setting you would also see others here as well, as shown:

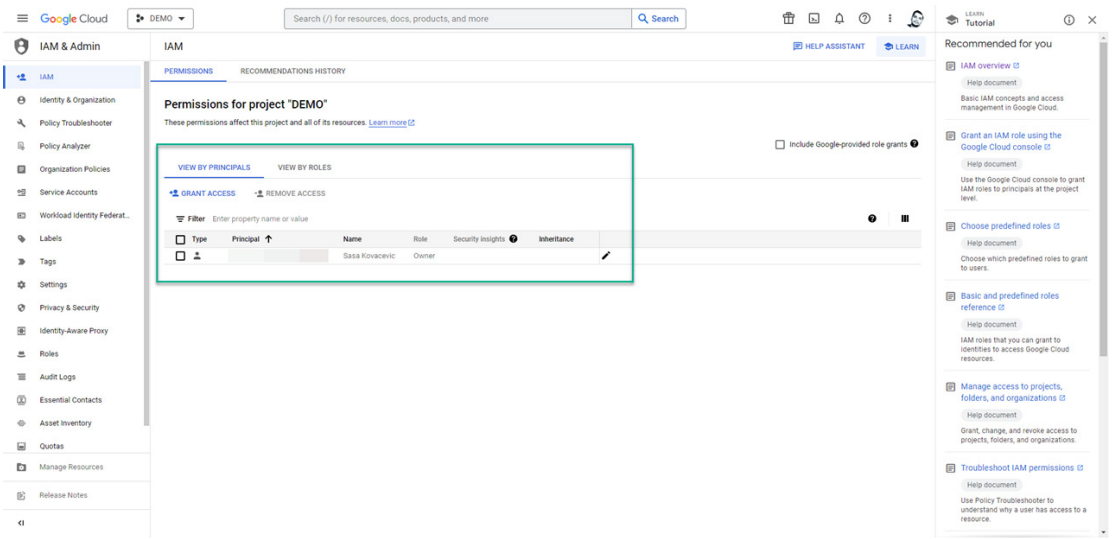


Figure 4.3: *GCP Console – IAM permissions*

4. Check and see the list of roles that have access. If you are doing this in an empty project, it is likely you will only see one, yours, and you will be listed as an owner. In an organization-type-setting you would also see others here as well:

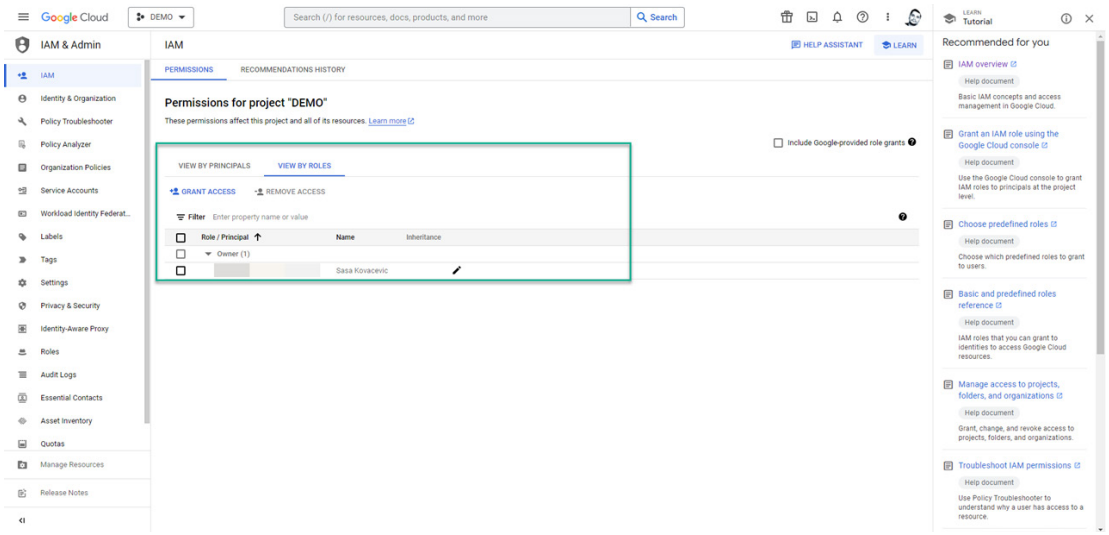


Figure 4.4: *GCP Console – IAM roles*

5. Select the Roles menu and browse all the possible roles from the list. From here you can create a role, as illustrated in the following figure:

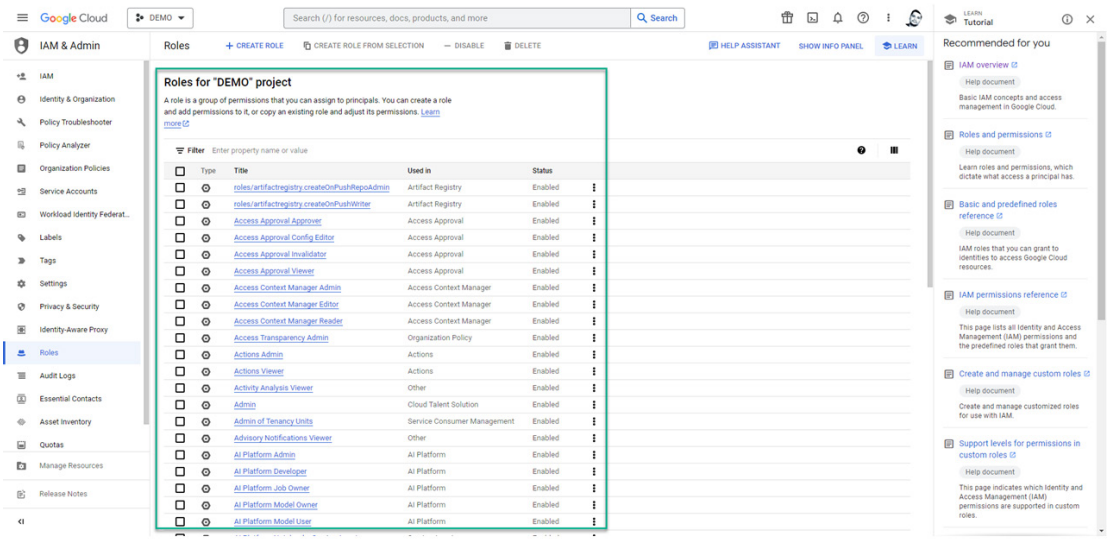


Figure 4.5: *GCP Console – IAM roles list*

6. Pick a role you like for this example and click on the three-dot menu.

7. Select the option from the popup named Create a role from this role:

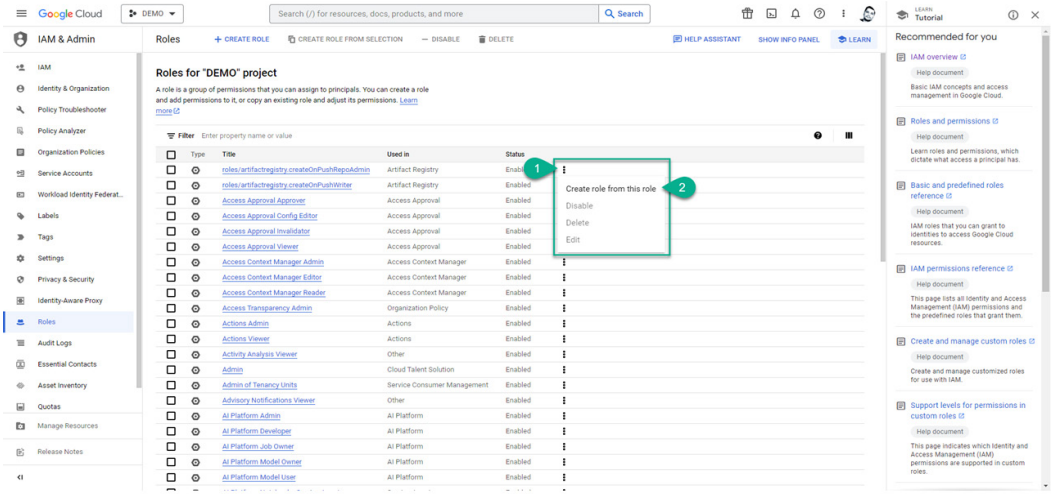


Figure 4.6: *GCP Console – create a role*

8. Complete the form by providing the required information in steps a, b, and c. You do not need to complete the role creation as the role you picked in the previous step was a random one for you, but this is the way you would go about creating one in GCP through the GCP console.

    a. Title.

    b. Description.

    c. Pick the permissions you would like your role to have.

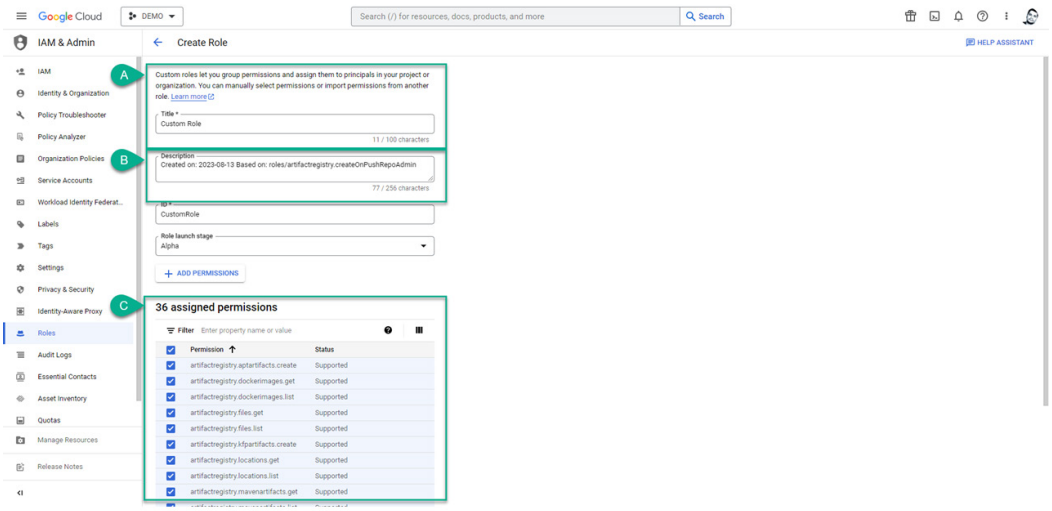


Figure 4.7: *GCP Console – Assign permissions*

Now you know how to access the GCP console, how to browse for the roles, and how to create one.

Azure Privileged Identity Management

Now let us see how JIT access would work in practice, using Microsoft Azure's **Privileged Identity Management (PIM)** features.

For this exercise you will need:

- An Azure account:
  https://azure.microsoft.com/en-us/free
- Access to the Azure Portal:
  https://portal.azure.com/
- PIM license (you can get it through the Azure Portal as a 30-day trial):
  https://learn.microsoft.com/en-us/azure/active-directory/privileged-identity-management/subscription-requirements
- Grant yourself the role of PIM administrator.

Let us set up PIM and activate a role:

1. Start by accessing the Azure Portal at **https://portal.azure.com/**.
2. Select Azure AD Privileged Identity Management as shown in *Figure 4.8*:
   a. From the quick access toolbar, or

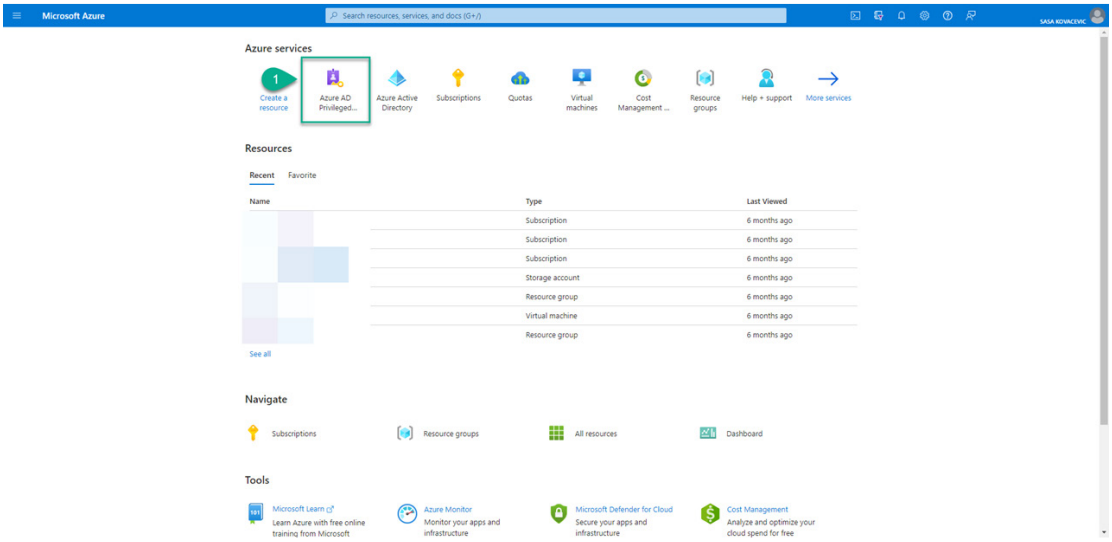


Figure 4.8:

b. By searching for PIM, as shown:

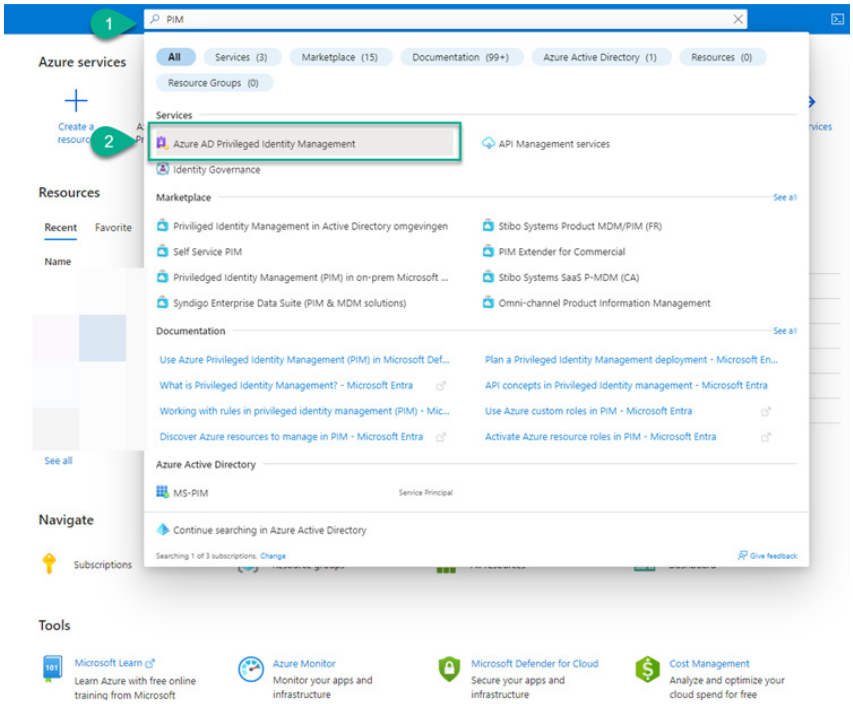


Figure 4.9: *Azure Portal – search - PIM*

3. Select Azure resources from the menu, as shown in the following screenshot:

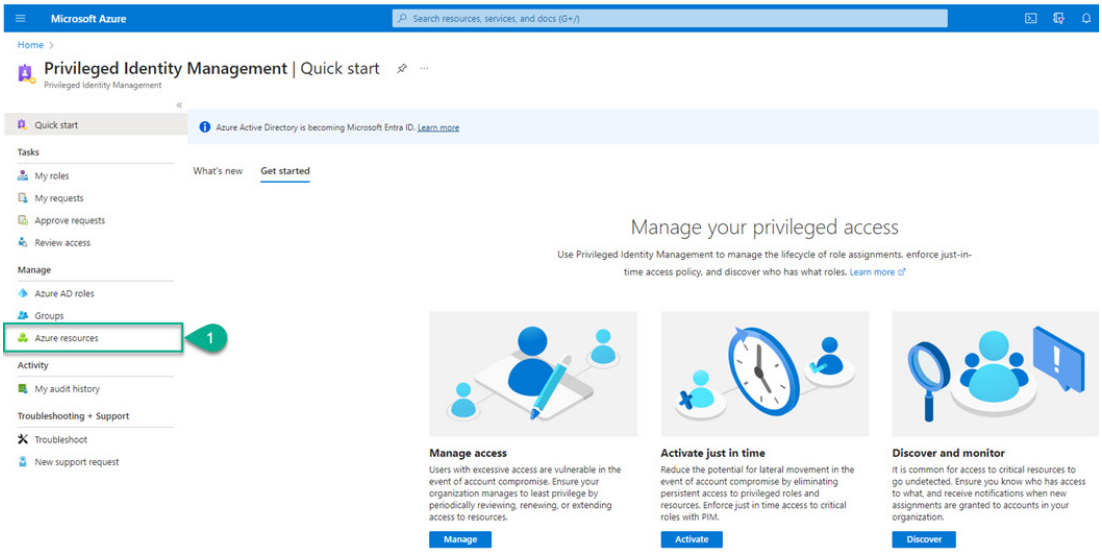


Figure 4.10: *Azure Portal - PIM*

4. Select Discover resources and choose a resource to manage (a subscription in this example).

5. After the resource shows up in the resources list, select it:

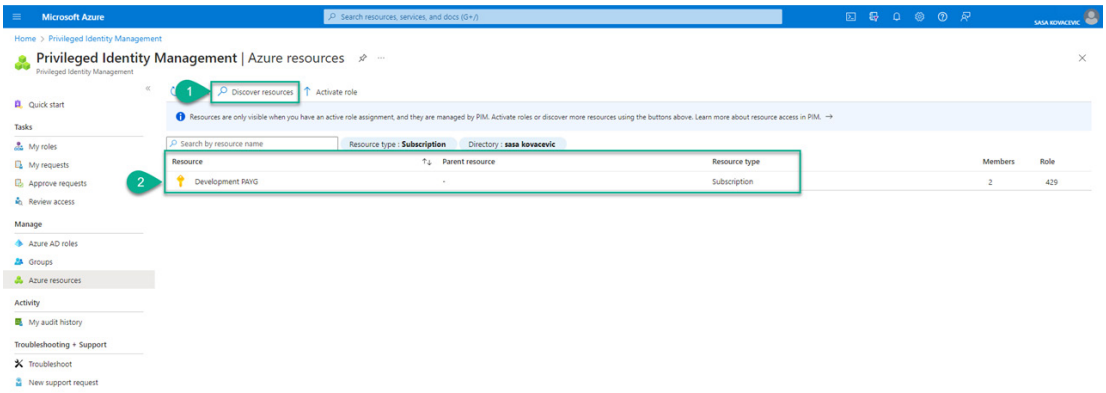


Figure 4.11: *Azure Portal – PIM – Azure resources*

6. Select the menu item **Add assignments** as shown in the screenshot:

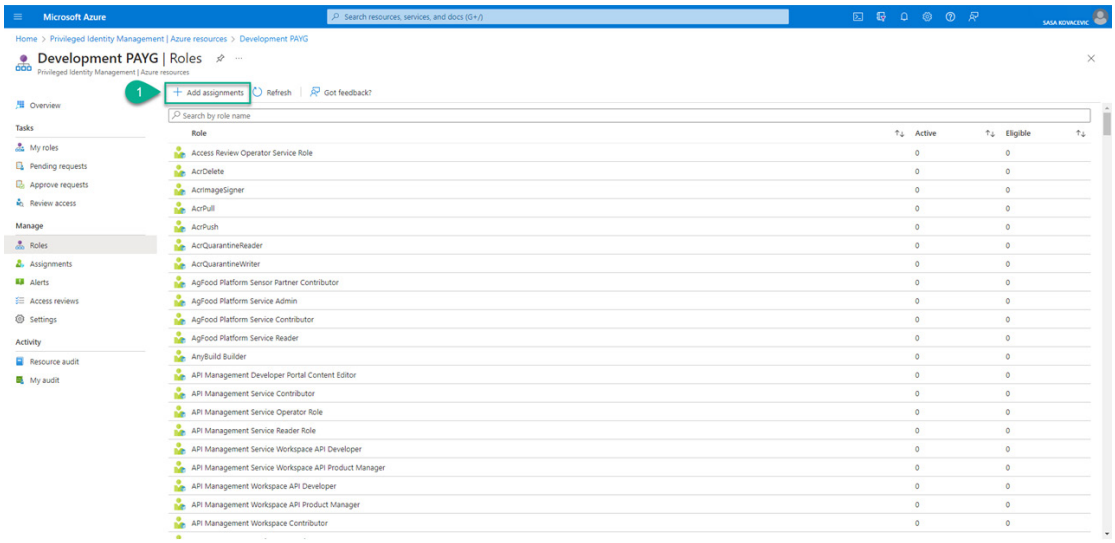


Figure 4.12: *Azure Portal – Add assignments*

7. Fill out the form by choosing a role and selecting a member to assign the role to. In this case, it is likely there is just you in your active directory, so select yourself, as shown:

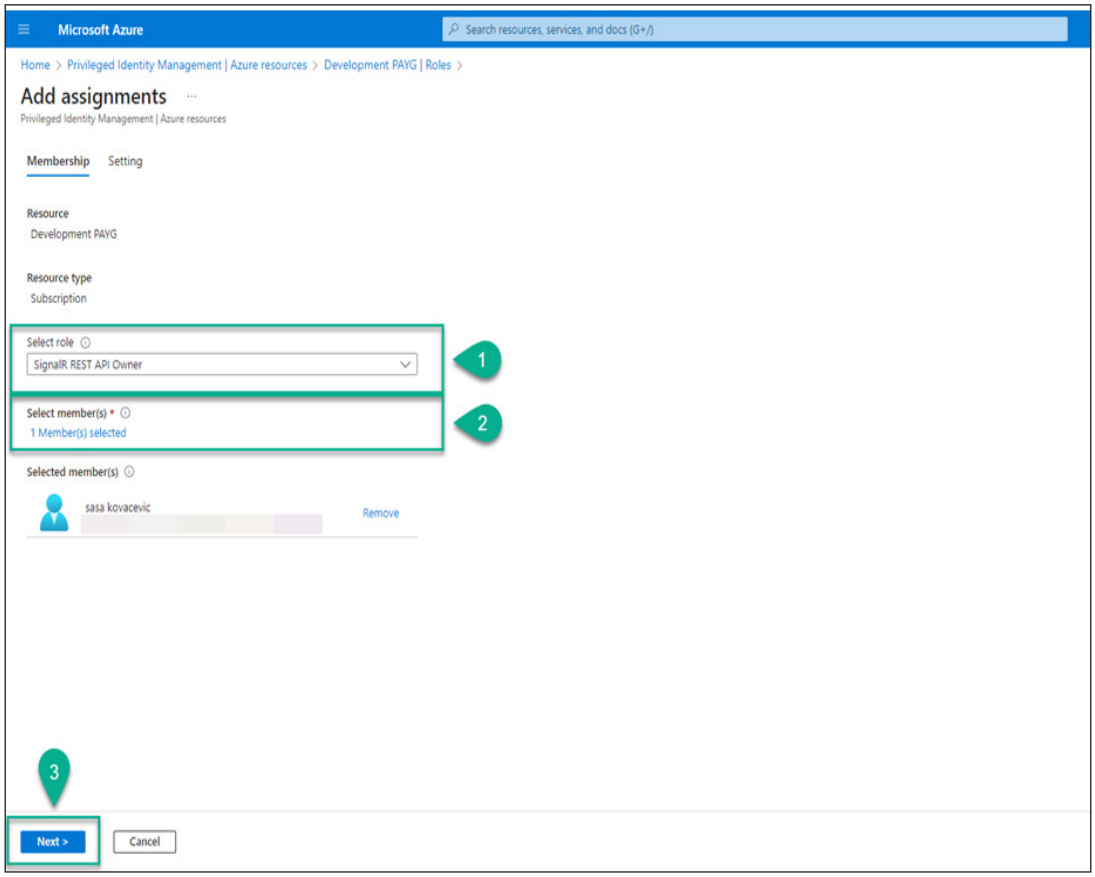


Figure 4.13: *Azure Portal – Add assignments - Membership*

8. Adjust the setting for assignment type to ELIGIBLE, which makes the role eligible for activation. The other option makes the role permanently active which is not what we want – remember, we want to protect our assets from a compromised account so we do not want them available all the time, we only want them available after ACTIVATION.

9. Adjust the start and end dates. In an organizational setting, this would be a quarter (for the PRODUCTION environment) or a year (for the development environment). This makes the assignment expire eventually, which means auditing and regular reviews are necessary.

10. Select the **Assign** button, as shown:

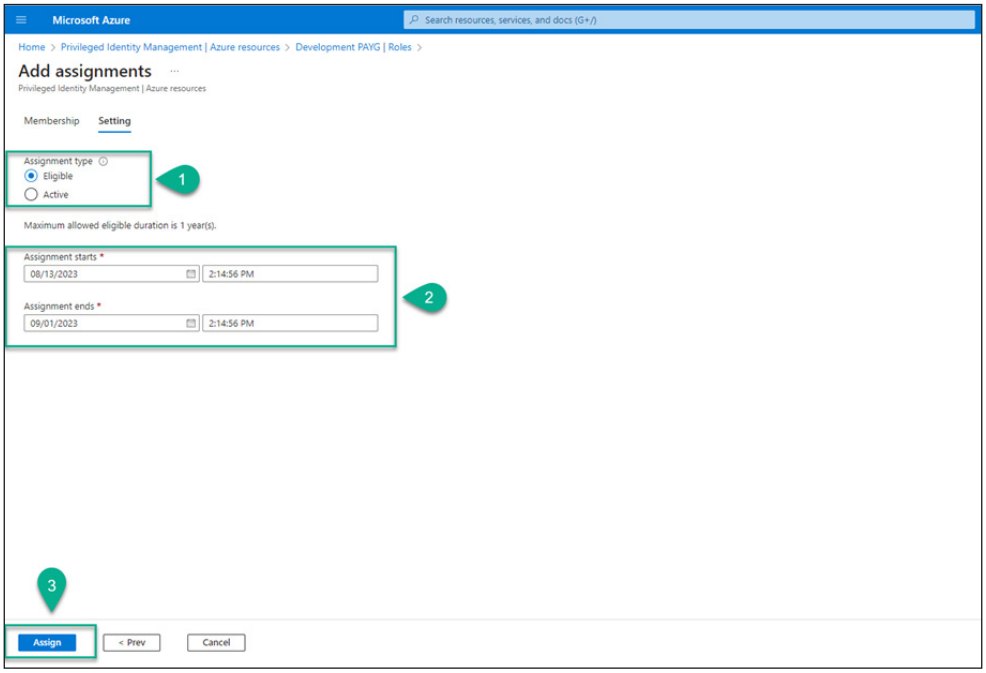


Figure 4.14: *Azure Portal – Add assignments - Setting*

This concludes the PIM setup for this role. Now we will look at this from the perspective of a user trying to activate the role:

1. Navigate to PIM again and select the My Roles menu, as displayed in the following figure:

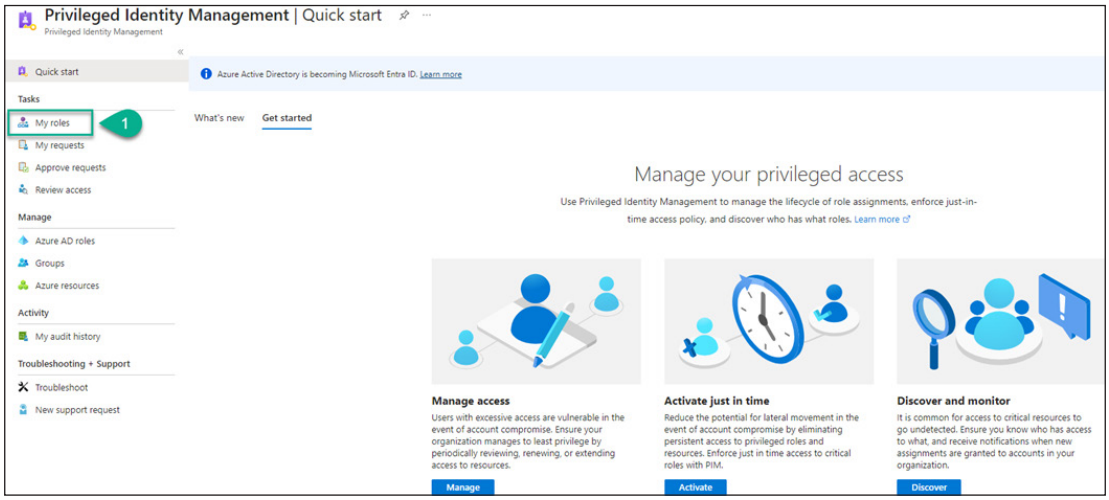


Figure 4.15:

2. Then select the Azure resources menu item. This shows you the role we created. This view would show ALL the roles if you have more than this one.

3. Select the **Activate** action for the role you want to activate, as shown:

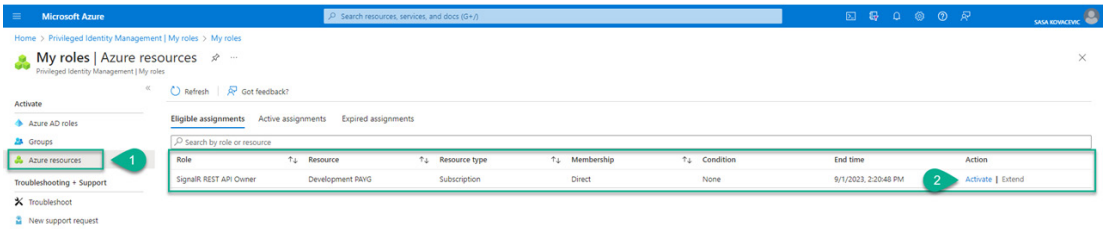


Figure 4.16: *Azure Portal – PIM - Azure resources*

4. A popup is displayed from which you need to select the duration of the activation of the role. This role is eligible for a maximum duration of 8 hours, but – to keep the window of opportunity small for the attacker we select here a duration of 2 hours as that is how long we estimate our tasks will require. If the role expires you can activate it again.

5. Another requirement is to fill in a reason for the activation. This helps auditors and you to remember what the reason for this activation was.

6. Select **Activate** to proceed:

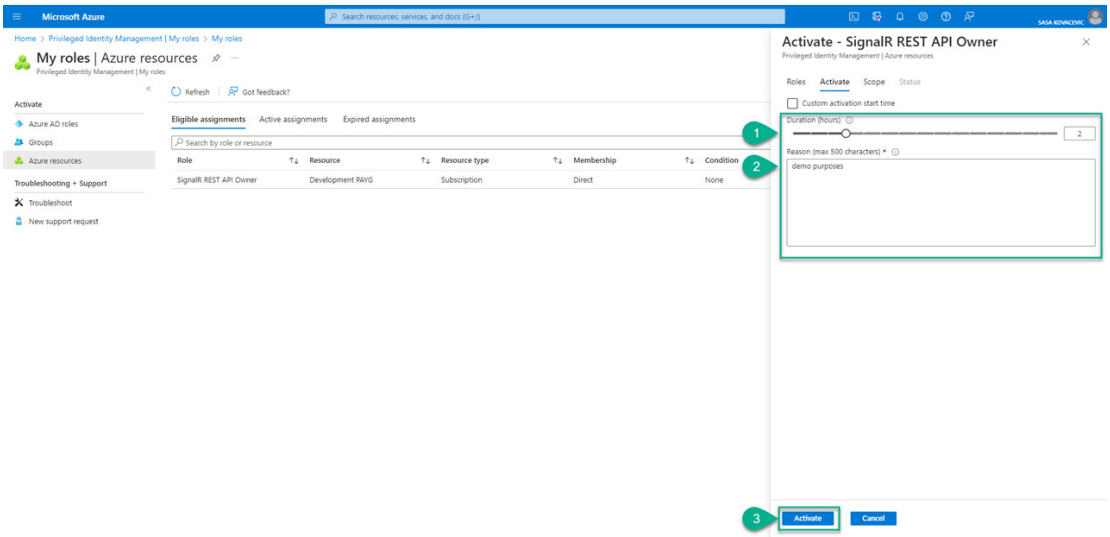


Figure 4.17: *Azure Portal – PIM - Activation*

7. The activation starts. There are three steps that the activation process will go through processing, validation of the activation, and the activation itself. The following screenshot shows these steps:

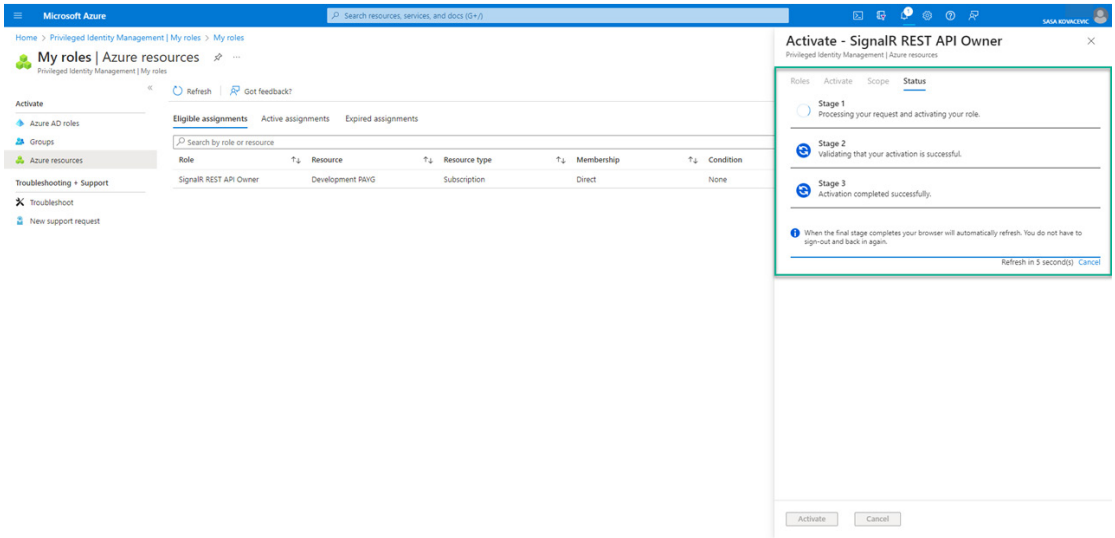


Figure 4.18: *Azure Portal – PIM - Activation - Processing*

8. Once all three steps are completed, the browser window will refresh, and you will have activated the role.

9. Now you can go on and access the resources you were just granted access to, as displayed:

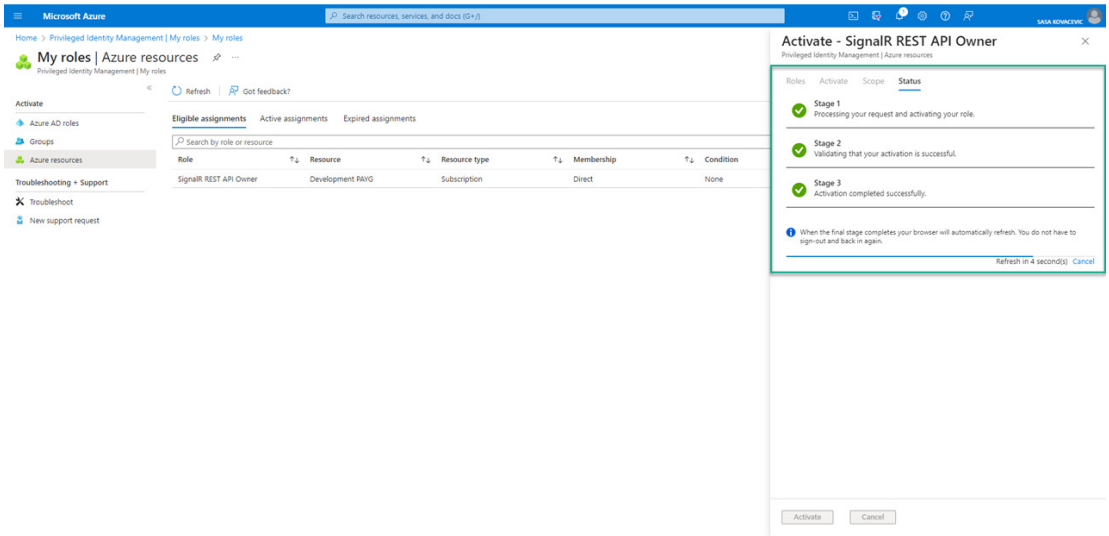


Figure 4.19: *Azure Portal – PIM - Activation - Completed*

Conclusion

We have discussed what controls are in place from a physical security standpoint in cloud provider data centers and how they relate to what you and your organization should do similarly for your workloads.

We also discussed business continuity processes and scenarios related to it.

Encryption is an important defense against data leaks, and we have gone into different types of encryptions, patterns around encryption itself, and the concepts of confidentiality, integrity, authentication, and nonrepudiation in this chapter.

Operational security was next, and we delved into what it means on a personal level and what it means on the cloud security level.

Anonymizing data, sanitizing it, and masking it – depending on use cases was discussed next and we discussed patterns for each.

Next, IAM and RBAC concepts were introduced, together with the least privilege principle, zero trust principle, and JIT access controls.

Lastly, you have gone through the typical setup of IAM/RBAC/JIT, in two different cloud providers' environments. You should now be able to transfer these learnings and these hands-on skills to any cloud provider.

In the next chapter, we will be looking into network security.

Questions

1. What types of controls are in place for a typical cloud provider data center?
2. What is threat hunting?
3. What is operational security, and what would be an example operational security plan against particular threats against your organization?
4. What is business continuity?
5. What is encryption?
6. What is the private key used for in public key cryptography?
7. What is the concept of nonrepudiation?
8. What is privilege escalation and what is typically exploited to achieve it?
9. How does an attacker discover information about target cloud environments?
10. What is lateral movement?
11.

12. What is AIM?
13. What is RBAC?
14. What is JIT access? How would you approach setting it up?

References

- AWS Cloud Security – Our Control:
  https://aws.amazon.com/compliance/data-center/controls/
- Encryption page of Wikipedia:
  https://en.wikipedia.org/wiki/Encryption
- Tactics and techniques representing the MITRE ATT&CK® Matrix for Enterprise covering cloud-based techniques:
  https://attack.mitre.org/matrices/enterprise/cloud/
- A GCP account:
  https://console.cloud.google.com/getting-started
- Access to the GCP Console:
  https://console.cloud.google.com/
- An Azure account:
  https://azure.microsoft.com/en-us/free
- Access to the Azure Portal:
  https://portal.azure.com/
- PIM license (you can get it through the Azure Portal as a 30 day trial):
  https://learn.microsoft.com/en-us/azure/active-directory/privileged-identity-management/subscription-requirements

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 5
Network Security

Introduction

In this chapter we will go from day one of the internet all the way to the present day and explain how networking has evolved and with it the security controls to manage networks for your organization in a globally connected world.

We will explain about the physical layer, data link layer, network layer, transport layer, session layer, presentation layer, and application layer - from the **Open Systems Interconnection** (**OSI**) model.

And then we will discuss how to secure all that, which services to use, how and why.

Structure

The chapter covers the following topics:

- History
- Open Systems Interconnection
- Secure Sockets Layer/Transport Layer Security
- Networking patterns

Objectives

The objectives of this chapter are straightforward. We want to bring you up to speed on network security, from its humble origins in the 1960s al through today while calling out relevant landmarks and technologies we will then talk about more in the later parts of this chapter.

You will learn about networking, packet switching, routing, TCP/IP, DNS, and OSI.

You will learn about all the different ways to protect the network you create in the cloud.

History

In order to understand how networking started, how it evolved, where it is not, and where it is going – we have to start from the very early beginnings of humanity.

Some might argue that fire is the most important invention in human history because it provided warmth, protection from predators, a way to cook food, and a way to extend activities into the night, enabling first social conventions, structures, and communications.

Some might argue that it was the wheel that was the most important, revolutionizing transportation and enabling the movement of goods and people everywhere.

Some might argue that it is the printing press because it provided an easy way to record, spread, and transform knowledge.

Some might argue that it was the vaccines that led to the eradication or significant reduction of the spread of illnesses, saving countless lives and improving public health on a global scale.

And some might argue that the invention of the internet is the most important invention in human history. The internet has transformed our societies, and the way we communicate, access information, conduct business and interact with each other. The internet has connected people across the globe, facilitated a rapid exchange of ideas and information, enabled the development of new technologies, and fostered collaboration and innovation on an unprecedented scale. And all that is underpinned by one technology – the humble - network.

Precursors

The origins of the internet can be traced to the 1960s. In the 1960s, the world was starting to focus more on anti-war movements, civil rights movements, and music and artistic expression. The 1960s inspired us with moon landings, counterculture, and social movements.

The 1960s also brought significant technological advancements, including the development of the first computer mouse and graphical user interface. These innovations laid the groundwork for the personal computing revolution in the decades that followed.

In the 1960s Advanced Research Projects Agency Network of the United States Department of Defense's Advanced Research Projects Agency came about. One of the goals was to create a decentralized network, one that could withstand disruptions – nuclear attack was top of mind back then.

It turns out that a global and decentralized network that can withstand disruption is also useful in other areas of our civilization.

The world welcomes networking!

In order to facilitate that lofty goal, packet switching was introduced. This meant that instead of sending information as a continuous stream, packet switching was designed to break data into chunks – smaller packets that can take different routes through the network, avoid any disruption, and be reassembled at the destination. This made communication over a network comprised of many different types of devices more robust and more efficient.

The world welcomes – packet switching!

Then, in the 1970s the **Transmission Control Protocol** (**TCP**) and **Internet Protocol** (**IP**) were developed. These protocols formed the foundation of the modern internet. TCP ensures reliable data delivery, while IP is responsible for routing packets across interconnected networks.

The world welcomes routing!

Let us look at a diagram of packet switching and routing in *Figure 5.1*. The sender in the top left is trying to send a message to the receiver in the bottom right through the network of connected devices on the internet. As per packet switching, the message is decomposed into 3 smaller packets (in our example) and sent over the network.

Packets 1 and 2 seem to take almost (but not quite) the same path through the network, while packet 3 goes off on its own route, before eventually rejoining the other two at the destination.

The receiver then reconstructs the original message from all the packets it has received.

It is quite possible and usual that, in some cases, packets will get lost and retransmitted. In others, they will just be lost, but the message is just fine without it, that packages may arrive out of sequence, and so on.

These are all the cases that can and do happen, there is nothing unusual about them. Take a look at the following figure for reference:

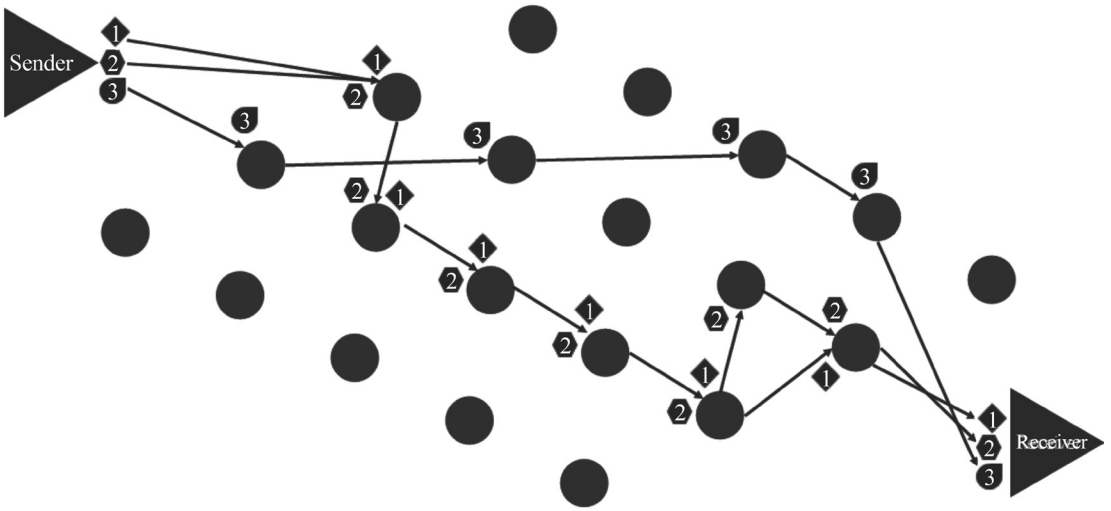


***Figure 5.1**: Packet switching and routing*

Ethernet introduced a method for connecting computers within a local area network using cables. This enables computers to communicate with each other in a shared network. As different institutions started creating their own computer networks, the need to interconnect them became evident. This led to the concept of early network gateways, or routers that could connect different networks together. And once that was possible, there was no stopping the network. Eventually, we gave this network of networks a name.

The world welcomes – the internet!

As the internet became more ubiquitous and integrated into every facet of our society, it came to be seen as less of a unique entity and more of a general concept or utility. The internet has become a common part of our technologies and our language.

In order to navigate and address all the vast arrays of network locations, we can navigate to the **Domain Name System** (**DNS**). It was introduced to provide a simple, hierarchical, and human-readable naming system. Instead of using numerical addresses that could and would change and were not as easy to remember, we got a variety of domain names.

The world welcomes – DNS!

Global reach

In the 1980s and 1990s, the internet transitioned from a (small but growing) research network to a commercial platform. The introduction of the World Wide Web marked the next significant milestone.

> *Creating the web was really an act of desperation, because the situation without it was very difficult when I was working at CERN later. Most of the technology involved in the web, like the hypertext, like the Internet, multifont text objects, had all been designed already. I just had to put them together. It was a step of generalising, going to a higher level of abstraction, thinking about all the documentation systems out there as being possibly part of a larger imaginary documentation system.*
>
> *Tim Berners-Lee*

The WWW provided a user-friendly interface for accessing information on the internet through web browsers.

The world welcomes the World Wide Web!

In the 1990s, we saw rapid growth in internet usage. Everyone everywhere wanted to get on the internet. Educational organizations, commercial organizations, governmental organizations, and individuals – they all wanted it. We all wanted it.

> *There are two equalizers in life: the Internet and education.*
>
> *John T. Chambers*

The issues

IP version 4 addresses (for example, the ones that look like: **192.168.11.222**) are numerous, but not as numerous as you would think. There are exactly 4,294,967,296 (2^{32}) unique addresses. That number is read as 4 billion, 294 million, 967 thousand, and 296. The issue is that there are now 8.1 billion people (in 2023) in the world, and these people have increasingly more devices connected to the internet – everything from aquarium pumps to sneakers and fridges are connected to the internet, including of course, all the computers, tablets and phones. There are now 15 billion mobile devices in the world (in 2023). While IPv4 has some clever tricks to account for all these and manage them, we are desperately running out of IPv4 addresses.

The world welcomes – IPv6!

The world is still moving towards IPv6 globally. IPv6 addresses (for example, the ones that look like **0000:0000:0000:0000:0000:ffff:c0a8:0bde** or compressed to**::ffff:c0a8:bde**) are more numerous than IPv4. How much more numerous? To the tune of about 340 undecillion (3.4 followed by 38 zeroes). This is so large that every atom on Earth could be assigned one, and we would still have plenty left over.

Security is also something of a concern on the internet. The early internet technologies were envisioned to facilitate communication and information sharing between relatively few researchers and academics. It operated within this small and trusting community. It was not anticipated that it would be used for global communication on the scale we

see today. E-commerce, online banking, and other sensitive transactions are now the norm. Malicious threats, vulnerabilities, worms, viruses, and other forms of malware are now prolific. Security flaws that were considered minor originally are now seen through a different lens now that the potential for widespread harm increased. Privacy and confidentiality requirements of today were not recognized as issues early on.

All of this leaves the internet in a precautious state today. While clever people and techniques can mitigate these issues, network design still requires a lot of thought, planning, research, and experience to do well.

One consequence of cloud computing and the shared responsibility principles is that managing networks is easier now than ever when you share the responsibility with multi-billion-dollar organizations that can invest heavily into security on their and your behalf. However, due to the permissive original design of IPs, it is still possible to inadvertently leave the front door wide open.

The world welcomes – cloud computing!

Balancing security and accessibility is one of your and your organization's tasks in today's world. The challenges lie in adapting the internet's architecture and protocols to meet modern security needs without compromising its fundamental principles.

And finally – the world welcomes software-defined networking, shared responsibility, zero trust, the Internet of Things, and so much more every day! It is your and your organization's job to continuously navigate this world - with the knowledge that one mistake could bring down your organization's brand, reputation, business results, or even the organization itself to its knees. The good news is that security is the one area where there is a lot of collaboration, including between direct competitors. The bad news is that it is only because of the enormous complexity of implementing security correctly with the global threat landscape we are enjoying today.

Open Systems Interconnection

The OSI model is a conceptual model from the **International Organization for Standardization** (**ISO**) that *provides a common basis for the coordination of standards development for the purpose of systems interconnection*.

OSI model recognizes seven layers, from the lowest (1) to the highest, as shown in the following figure:

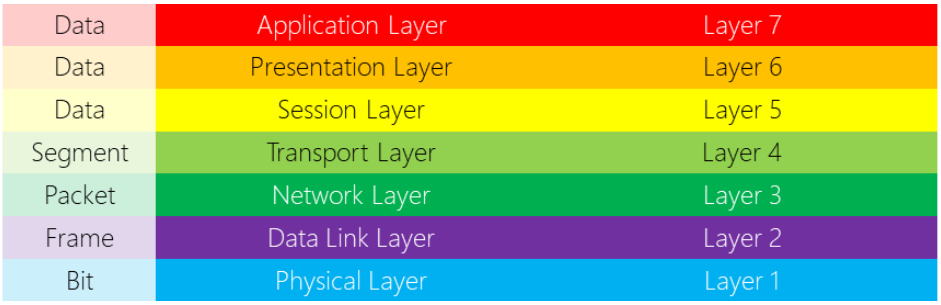


Figure 5.2: *OSI protocol data model per layer*

Let us now briefly describe each model and security measures for each of the layers and then let us show specific cloud services from Microsoft Azure (as an example) on each of the layers.

OSI layers

Each OSI layer describes a particular purpose, so let us go through them.

- Application Layer is responsible for application-layer functions. Examples of these are file sharing, message handling, database access, and through HTTP (web), FTP (files), SMTP (email), and other protocols. It is quite usual for applications to have access to multiple application layer protocols. For example, HTTP and database access at the same time – one to communicate with the user's browser and the other to communicate with the database.
- Presentation Layer is responsible for data formats and data translation from various formats if and when required. Data encryption, decryption, compression, and decompression is handled here. If you are familiar with software development and development environments, this layer is also responsible for serialization and deserialization.
- Session Layer is responsible for creating and destroying connections between two or more devices. One of the reasons (other than security) you might occasionally get disconnected from services is the number of sessions it supports at any one time and the need to manage the overwhelming number of them for some popular services. DNS operates at this layer as well. This layer is usually required if layers 6 or 7 require remote connections or remote procedure calls.
- Transport Layer is responsible for quality of service functions, meaning it is responsible for the transfer of data while ensuring an agreed level of service. It works as both connectionless and connection-oriented. It breaks large protocol units into smaller ones to ensure they fit into the maximum packet size. Important functions such as flow and error control, and any acknowledgments and sequencing requirements are handled here. As some protocols are state and

connection-oriented, this layer also ensures a successful handshake before the transmission of data and keeps track of segments sent, and the need to retransmit some packets as well as retransmission itself. The speed of communication is dependent on the protocol selected. For example, UDP is a protocol that allows for packet loss, duplication, reordering, etc. For example, video transmission. By the time the packets arrive, a few video frames might be skipped or duplicated or erroneous, but they might not be obvious to the viewer, so the protocol is designed for speed, not for absolute correctness in transmission.

- Network Layer is responsible for transferring packets from one network to another. Nodes have their own addresses and the network layer is responsible for navigating between them in a way to attempts to ensure (but does not guarantee) delivery of the packets. It is at this layer where the message gets split into packets if the message is too large to be transmitted all at once.
- Data Link Layer is responsible for the communication between the nodes that are communicating. It is responsible for defining the protocol, and it starts and terminates connections on each node.
- The Physical Layer is responsible for the transmission and reception of raw data between devices. It is literally the physical layer as it is responsible for translating between the digital bits and the physical–electrical or optical signals. This is how electricity or light is translated to digital bits and vice versa. For example, the **wires** only understand electricity terms such as voltage, while the computer only understands bits such as 0 or 1, and layer 1 translates between the two.

The messages sent and received go through all the layers but in different orders. Very simplistically, the sender message goes through layers 7 to 1, ensuring that whatever data the application wants to send is decomposed and sent through **the wire**, while on the receiver's end message goes through layers 1 through 7 in order to receive the message and construct it again and pass it on to the application.

From a security standpoint, there are also cross-layer functions – meaning services that affect more than one layer. This is especially common in security software and security functions as security usually needs to be performed on more than one layer by a particular application.

OSI layer security

Here is an overview of security measures for each of the seven layers that your organization or your cloud provider can deploy:

- Application Layer has security controls and security measures such as authentication and authorization by implementing sufficiently strong mechanisms and fine-grained authorization control. The evolving sophistication of attackers and exponential growth in computing capabilities force the evolution of ever-

stronger security mechanisms. Input validation and secure coding practices should (must) be deployed at this layer as well.

- The Presentation Layer has security controls and security measures such as securing data transformation in a secure way to prevent attacks such as SQL injection and **cross site scripting** (**XSS**). Data compression security ensures that any compression does not introduce vulnerabilities or allow attackers to exploit decompression algorithms.
- Session Layer has security controls and security measures such as session tokens and session encryption in order to ensure only authorized parties to the communication have access and protect against eavesdropping and general data interception.
- Transport Layer has security controls and security measures such as encryption by using protocols such as TLS/SSL to encrypt data in transit. Session management to prevent session hijacking, as well as port filtering, can be done at this layer as well.
- Network Layer has security controls and security measures such as firewall and access control lists to filter incoming and outgoing traffic (ingress and egress) based on IP addresses, ports, and protocols. Intrusion detection and prevention systems provide security on this layer, as well as **Network Address Translation** (**NAT**) in order to hide internal network topography from the external networks.
- Data Link Layer has security controls and security measures such as **Media Access Control** (**MAC**) address filtering (is a unique identifier assigned to a **Network Interface Controller** (**NIC**) of a device, such as a computer, smartphone, or network hardware like a router or switch), port security to restrict the number of MACs allowed on each, **Virtual Local Area Network** (**VLAN**) segmentation to logically segregate network traffic and prevent unauthorized access.
- The Physical Layer has security controls and security measures such as access controls, surveillance, restricted entry to prevent unauthorized entry by persons or persons to the physical infrastructure, preventing physical tempering of cables and cable infrastructure, equipment lockdown of routers, switches, or servers in locked cabinets or racks.

Security measures on each level and across levels must be brought to bear in harmony with each other in order to provide sufficient protection for the data transmitted over **the wires**. A layered security approach is the only way to ensure the strengths and weaknesses of each layer are considered and result in a comprehensive security posture and defense strategy.

Similarly, attackers will consider attacking each layer and see if they can exploit either individual layers or disharmony between them in order to gain access.

Azure services across OSI layers

Now let us see how the principle of shared responsibility in securing your Azure environment applies to each of the OSI layers. It may be redundant to say this but still – you should strongly consider using each and every service listed here to secure your environments in Azure.

- Application Layer security in Azure is addressed with services such as Azure Active Directory, Azure App Service, Azure API Management, and Azure Web Application Firewall:
    - Azure Active Directory provides identity and access management services, crucial for authentication and authorization at the application layer.
    - Azure App Service offers a platform for building, deploying, and managing web and mobile applications at the application layer.
    - Azure API Management helps with the creation, publishing, and management of APIs, primarily at the application layer.
    - Azure Web Application Firewall offers protection against web-based attacks and is positioned at the application layer.
- Presentation Layer security in Azure does not have services specifically tied to it, as this layer is often handled by application-specific code.
- Session Layer security in Azure does not have services specifically tied to it, as modern applications often handle session management at higher layers.
- Transport Layer in Azure is addressed with services such as Azure Load Balancer which balances traffic across multiple instances of an application or services and can work on both transport and network layers. In addition, Azure Application Gateway acts as a load balancer and performs application layer functions.
- Network Layer security in Azure is addressed with services such as Azure Virtual Network providing routing, IP address management, and network isolation features. In addition, Azure Firewall acts as a network layer firewall to control and inspect traffic between Azure and external networks.
- Data Link Layer security in Azure is addressed with services such as Azure Virtual Network and Azure Network Security Groups – both there to ensure network segmentation, subnets, and network isolation with access controls based on MAC addresses, IP addresses, and port numbers.
- Physical Layer Security in Azure is completely in the hands of the cloud provider (Microsoft and its global contractors), as it entails securing the data center, cables, and physical hardware.

It is important to note that many Azure services work across multiple layers of the OSI model, offering a comprehensive approach to building and securing cloud-based applications and infrastructure.

Cloud technologies that work across layers

The lines between layers can blur at times, especially with the advent of modern software-defined networking and cloud technologies that work across layers. The mapping of Azure services to specific layers may not always be straightforward and can vary based on the context and use case, that is, depending heavily on your and your organization's workflows.

There are many applications and services that operate across multiple layers of the OSI model:

- Firewalls and security appliances are used to inspect traffic at the network, transport, and application layers.
- Load balancers and application delivery controllers distribute network traffic across multiple instances of an application or service, operating at the transport and application layers, managing traffic flow, and attempting to improve performance.
- Intrusion detection and prevention systems monitor network traffic for suspicious activity. They can be used to detect anomalies in traffic patterns in the network layer and inspect traffic/packet content in transport and application layers.
- Proxy servers operate at various layers as well, in order to provide features such as caching on the presentation layer, content filtering on the application layer, and further load balancing on the transport layer.
- **Virtual private networks** (**VPNs**) create secure connections across public networks. They can operate on data links, networks, and transport layers to ensure confidentiality, and integrity of data.
- Web application firewalls protect web applications from attacks such as SQL injection, cross site scripting, and others. They also inspect http requests and responses for malicious patterns (for example, data exfiltration).
- Application delivery and acceleration solutions optimize performance by reducing latency and optimizing data flow. Compression, caching, and protocol optimization are the usual techniques operating at the transport and application layers.
- Software-defined networking solutions dynamically adjust network configuration and optimize traffic flow.
- Unified threat management appliances and network monitoring and logging tools also operate across layers by virtue of unifying multiple functions into one appliance and capturing and analyzing network traffic patterns to troubleshoot any problems occurring and mitigate against various threats.

Secure Sockets Layer/Transport Layer Security

Secure Sockets Layer (SSL) and **Transport Layer Security** (**TLS**) are cryptographic protocols designed to provide secure communication over a computer network.

They are used to establish encrypted connections between a client and a server to ensure that data exchanged between them remains confidential and protected from unauthorized access (for example, between a client such as a browser and a server such as a web server).

SSL was the original protocol, but due to vulnerabilities and advancements in cryptography, it was succeeded by TLS.

TLS is an improved and more secure version of SSL. However, the terms SSL and TLS are often used interchangeably, and the protocol is generally referred to as SSL/TLS. Although nowadays it almost exclusively means TLS in practice.

SSL/TLS is commonly used to secure various types of online communication, including web browsing, email, file transfers, and more. When you see **https://** in a website's URL, it indicates that the website is using SSL/TLS to encrypt the data being exchanged between your browser and the web server. This is particularly important for protecting sensitive information like passwords, credit card details, and personal data from being intercepted by malicious actors.

SSL/TLS performs functions such as handshake, certificate exchange, key exchange and - data encryption and integrity check.

Handshake

When a client wants to establish a secure connection with a server, they initiate a handshake process. During this process, the client and server exchange information, including the supported encryption algorithms and certificates.

Certificate exchange

The server presents a digital certificate to the client. This certificate is issued by a trusted third party called a **Certificate Authority** (**CA**). The certificate contains the server's public key and other identifying information. The client can use this certificate to verify the server's authenticity.

Key exchange

Using asymmetric cryptography, the client and server establish a shared secret called a session key. This session key is used for symmetric encryption, which is more efficient for data transmission.

Data encryption

Once the session key is established, all data exchanged between the client and server is encrypted using symmetric encryption. This ensures that even if someone intercepts the data, they cannot decipher it without the session key.

Data integrity

In addition to encryption, SSL/TLS also provides data integrity through the use of hash functions. Hashes are used to ensure that the data has not been tampered with during transmission.

Networking patterns

There are certain networking patterns used by organizations everywhere, enabled by cloud computing providers. In this section of the chapter, we will go through some of them – particularly related to securing the networks for the cloud workflows.

Direct connection

In order to connect your organization to the cloud securely there are other options other than connecting over the internet. These options are usually considered if the traffic between the organization and the cloud is considerable – and it usually is for enterprise-sized companies.

Azure has Express Route, AWS has Direct Connect, and GCP has Cloud Interconnect. There are differences between them, but we will concentrate on the common pattern of connecting your organization's network to the cloud provider.

These work on the basis that a third-party provider will provide connectivity between your organization's on-premises network and your network setup in the cloud. Your cloud provider and your communication provider will work with you to establish this connection, as shown in the following figure:

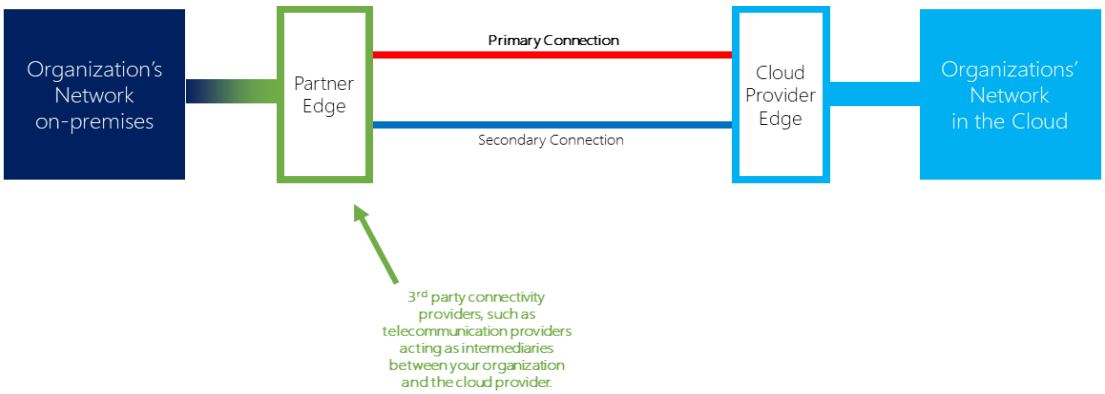


Figure 5.3

An alternative to the private connection may be a VPN. You can choose to connect site-to-site, point-to-site, or virtual-network-to-virtual-network. These depend on your security and connectivity needs and the amount of traffic expected.

DNS

Managing your own domain names can be managed by using a DNS service. Azure provides it as a DNS Zone service. An example of setting up a DNS zone is explained as follows:

1. In the Azure Portal, search for DNS. You will see a few options for setting up DNS.
2. For this example, we are using the **DNS Zone** service so select that option, as shown:

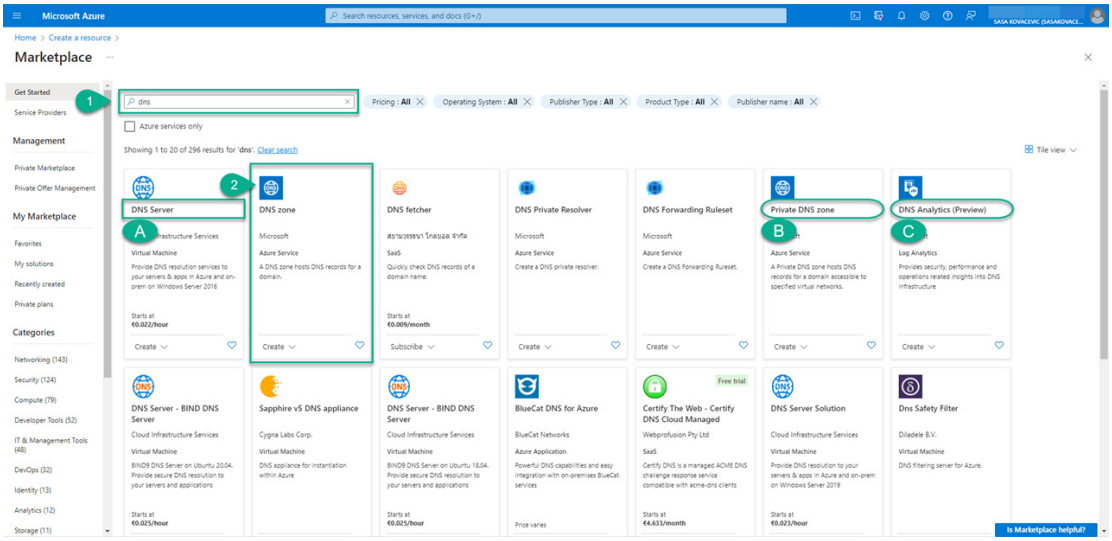


Figure 5.4: *DNS Zone creation*

3. Fill in the basic information, as shown in *Figure 5.5*:
    a. **Subscription**
    b. **Resource group**
    c. **Name**
    d. **Resource group location**

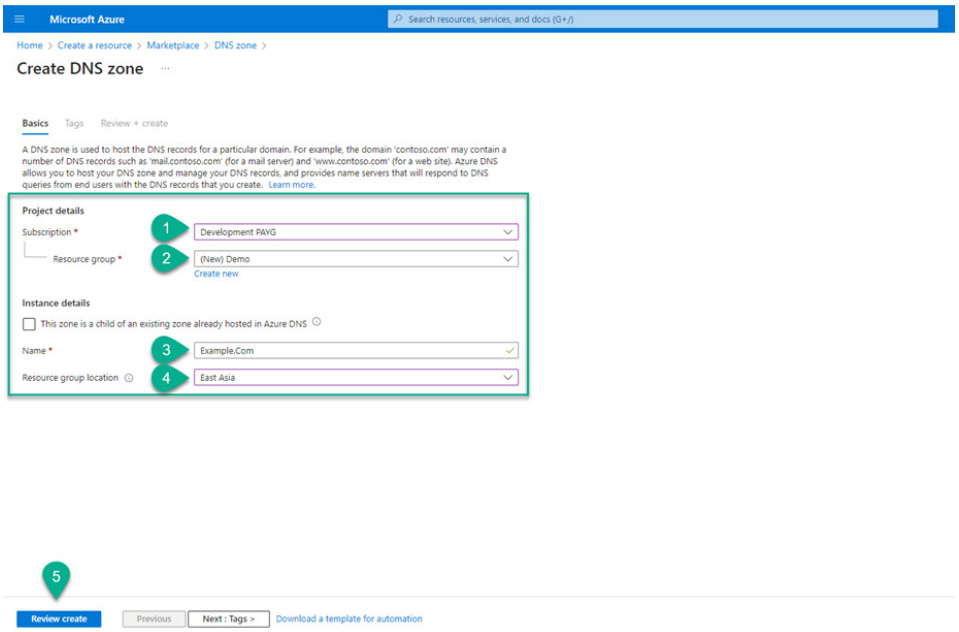


Figure 5.5: *DNS zone basic information*

Once you create the DNS zone you can see the default entries and the name servers. These name servers are the ones you would add in the control panel of your domain registrar to point your actual domain to resources in Azure. Adding record sets is how you define the DNS to resource mappings. In the following figure you can see how to view and add a DNS record:

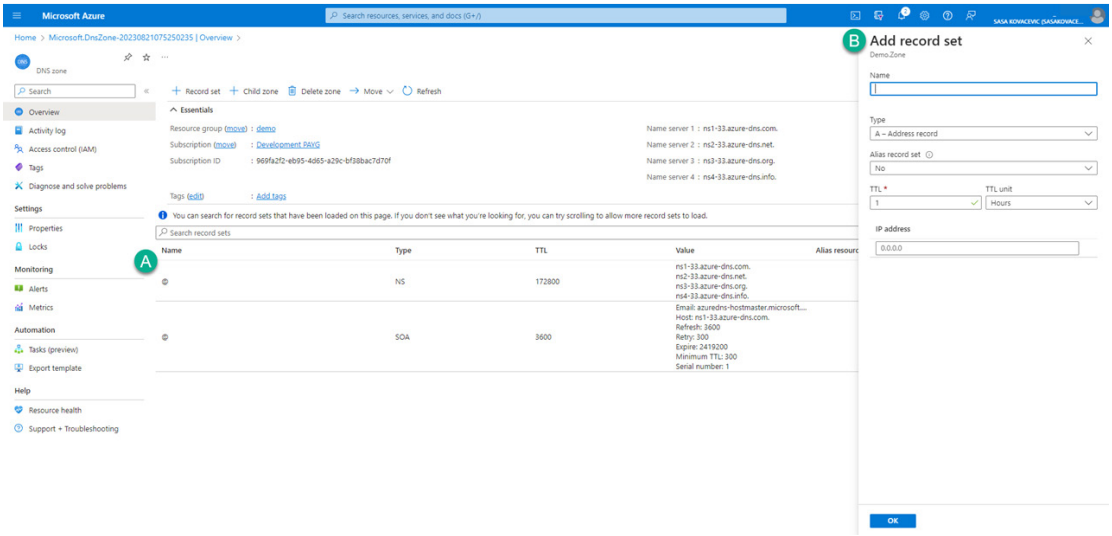


Figure 5.6: *DNS zone →*

Bastion

In order to securely access resources on the network, one pattern is extremely useful. Using a Bastion server or a service that is deployed into the same network and has permission to access these other resources, you can access resources securely and from a point of access that can be managed, audited, and even turned on and off as needed.

DDoS protection

If you or your organization have public-facing resources, it is likely that you will need to avail of the DDoS protection service at some point. This service provides DDoS mitigation capabilities for your applications, services, and resources deployed in virtual networks.

DDoS protection provides 24/7 traffic monitoring, adaptive real-time tuning, and access to rapid response in the case of an ongoing DDoS attack.

To tie it back to the OSI layers, when deployed with a **Web Application Firewall** (**WAF**), Azure DDoS Protection protects both at the network layer (Layer 3 and 4, offered by Azure DDoS Protection) and at the application layer (Layer 7, offered by a WAF). WAF offerings include Azure Application Gateway WAF SKU and third-party WAF offerings available in the Azure Marketplace.

- Azure DDoS protection overview:

  https://learn.microsoft.com/en-us/azure/ddos-protection/ddos-protection-overview

Private link

Some PaaS services come with an internet connection by default – for example, Cosmos DB. In order to remove the public endpoint these services can be integrated through a private endpoint into your organization's network.

This removes the public endpoint from the internet, and the only way to access these resources is through your own network.

Commonly in n-tier application architecture, not all tiers need to be exposed to the public. While the UI might need to be, the backend is certainly better to be removed from the internet and only accessible from the network you set up and control. If these resources need to be accessed by the public, it is always better to provide this service behind an API or another service (proxy, reverse proxy) that you control rather than just exposing the endpoint to everyone on the internet.

Subnets

When designing the network topology, it is important to segregate workflows as much as possible in order to allow access only from/to resources that require it. One common pattern revolves around segregating subnets.

Before we consider a scenario, during the creation of a virtual network, you get a security screen/step in the Azure portal that highlights some of the patterns we have just been discussing.

Azure Bastion, Azure Firewall, and Azure DDoS Network Protection are all the options that you are advised to consider for this virtual network you are creating. You should be able to confidently defend your decision not to use these services when creating a virtual network, and there may be legitimate reasons why you do not need some or all of these services. But that decision needs to be made after fully considering the security implications of each. In the following figure you can see how to create a virtual network:

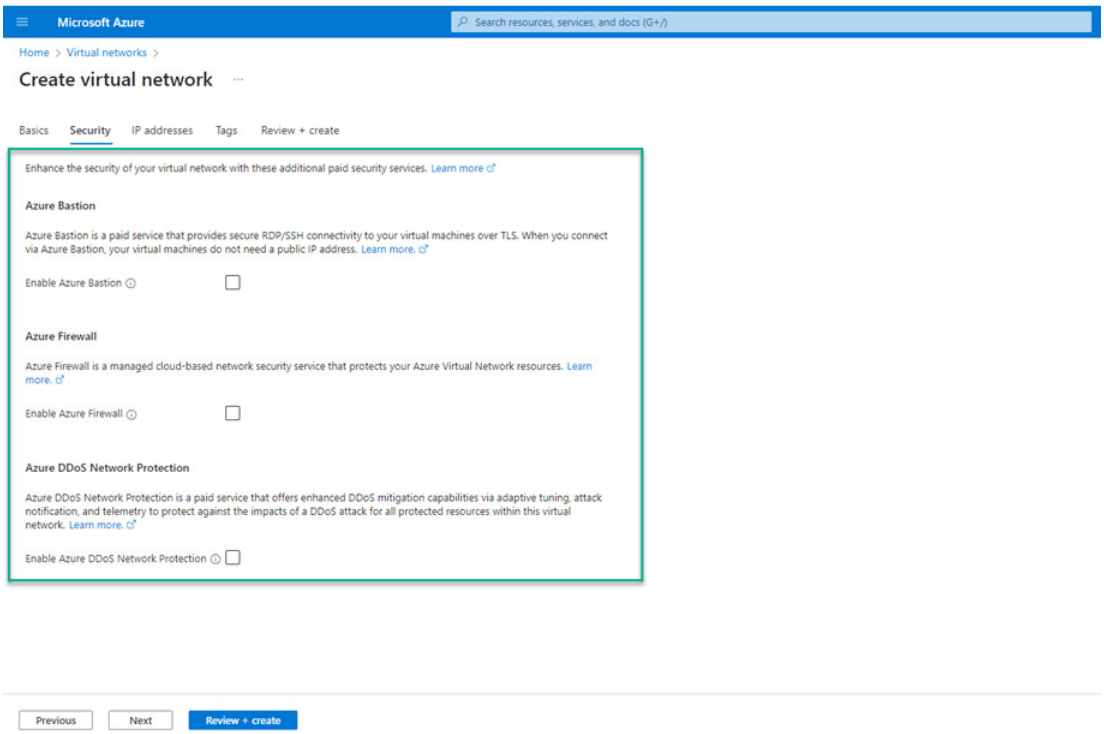


***Figure 5.7**: Create a virtual network → Security*

How you choose to set up subnets in this virtual network is up to you. Here is an example of how this might look:

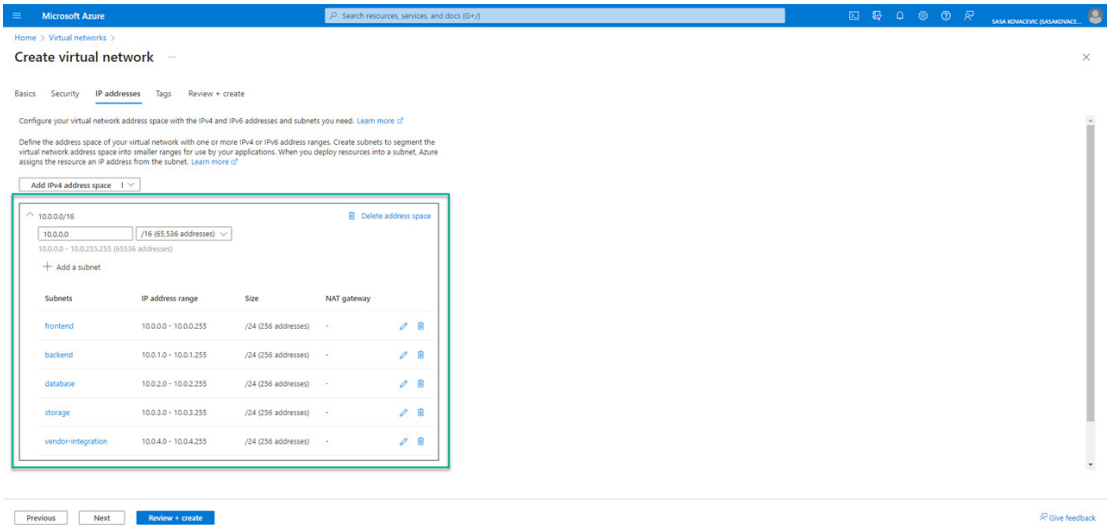


***Figure 5.8**: IP addresses*

In the virtual network 10.0.0.0/16 we have chosen to create 5 subnets:

- Frontend – 10.0.0.0/24
- Backend – 10.0.0.1/24
- Database – 10.0.0.2/24
- Storage – 10.0.0.3/24
- Vendor - Integration – 10.0.0.4/24

All these subnets can now be secured using **Network Security Groups** (**NSGs**), as shown:

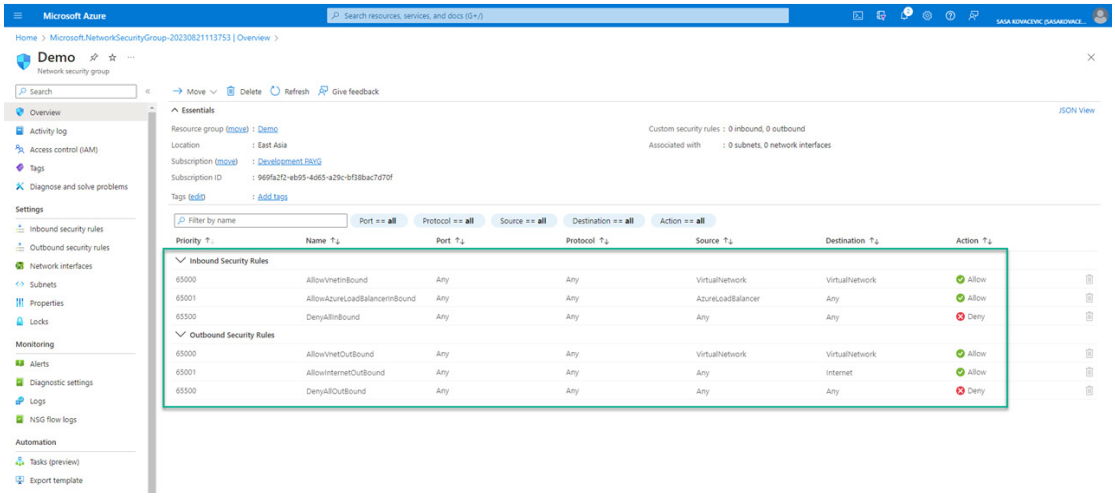


***Figure 5.9:** Network Security Group*

By default, NSGs come with a few, well-named rules.

Notice the important columns:

- **Priority:** Priority with a lower number gets evaluated first, hence why for example DenyAllInBound is priority number 65500, meaning the evaluating engine will check all the rules with lower priority numbers and evaluate them, and then deny all other inbound traffic if no rule matched prior to it.
- **Port and protocol:** Defaulted to any here, but when you are adding rules, you can specify each.
- **Source and destination:** Self-explanatory, shows what the targets are in a virtual network, internet, or any other network.

Shown above is one NSG, just created. For our purpose, we would need to create a few of them to correctly secure our network. How would we in our subnet example restrict traffic flows? One such way is described as follows:

- **Frontend:** This should allow access from and to the internet and to the backend.
- **Backend:** Should allow access from the frontend (therefore not from the internet) and to database, storage, and vendor integration subnets.
- **Database:** This should allow access from the backend subnet.
- **Storage:** Should allow access from the backend and from the internet (maybe we want to allow users to upload data – this should of course be done in a controlled way via the use of shared access signatures. A shared access signature provides secure delegated access to resources in the storage account with granular control over how a client can access data).
- **Vendor-Integration:** This should allow access from the backend and to the internet.

You can apply NSGs to individual resources, but that may be an antipattern and make access management difficult to understand or verify.

Conclusion

As you have seen, there are so many different techniques and technologies to protect the networks in the cloud. There are so many because you have to cater to all the OSI layers, to the different workloads you will be hosting in the cloud, and to the different requirements of services used in those workloads.

In this chapter, we went through all the distinct kinds of services you can use to protect your network, everything from protecting ingress and egress traffic, to protecting your resources in different subnets. Careful architectural and security decisions are needed at each step of securing the cloud networks you deploy.

You have learned a general way to protect the networks you manage, in the concrete example of Azure. However, you should be able to use the same knowledge, techniques,

and patterns to secure the network in AWS and GCP and, to a lesser extent (due to not being able to rely on all the readily available cloud services) on-premises as well.

Networks have been around since the very creation of the internet, and they have secured then and now in the same ways. Today, in the cloud, you have a lot more support from the cloud provider to secure your networks than you needed to do in the past by yourself.

In the next chapter, we will discuss securing applications and data.

Questions

1. How did networking, packet switching, routing, and the internet come about?
2. What benefits are brought to the forefront with packet switching?
3. What benefits are brought to the forefront with routing?
4. What benefits are brought to the forefront with TCP - and what about IP?
5. What is DNS?
6. Why does the world need IPv6?
7. What is OSI?
8. Which Azure services map to which OSI layer?
9. Can you find out which services from AWS and GCP are provided for each OSI layer as we did with Azure services in this chapter?
10. Which security measures can be brought to bear on each of the OSI layers?
11. Which functions does the SSL/TLS perform?
12. What are all the different ways you can protect the network in the cloud?
13. What cloud resource would you use to inspect incoming traffic for SQL injection?

References

- Tim Berners-Lee:
  https://en.wikipedia.org/wiki/Tim_Berners-Lee
- IPv6 information on Wikipedia:
  https://en.wikipedia.org/wiki/IPv6
- OSI Model:
  https://en.wikipedia.org/wiki/OSI_model
- Azure DDoS protection overview:
  https://learn.microsoft.com/en-us/azure/ddos-protection/ddos-protection-overview

Chapter 6
Securing Applications and Data

Introduction

Everything we have talked about until now has been leading us to one of the craziest ideas ever - to deliver a useful service to our customers.

How can we do that on a scale? What does multi-tenancy mean? Are all applications cut from the same cloth? Can we and how can we trust our customers? The answer to the last question is – we cannot.

Data security in transit and at rest is the next topic we will discuss in this chapter. How to do just that - secure data in transit and at rest - and what are the common pitfalls when protecting data?

Data has gravity. We will explain what that means and how that influences cloud security.

Data wants to be shared, free, reused, and transformed, but data also needs to be protected from itself and from others that would abuse it. We will show how to do that - protect it - while at the same time enabling your organization to use the data it has to the maximum capabilities allowed by law and by morality.

And remember, not all data is equal. There are different types, different qualities, and different formats. And there are challenges coming from those facts as well.

Encryption is another can of worms we will just open, have a glance at, and then quickly close. Different types of encryptions are suited for different types of data and different requirements for the usage of said data.

Structure

The chapter covers the following topics:

- Scenario
- Frontend
- Backend
- Management and security services
- Application/service itself

Objectives

In this chapter, we will learn about securing applications and data - from the initial request all the way through processing the request in different ways and through different code paths in different workflows, and finally to the end result, which is either saving some data and storing it for a little (or a long while) and returning a response.

There are many ways to code applications, architecture and application, transmit and store data, and many ways that any of that can go wrong.

In this chapter, we will investigate some of the (arguably - easiest) ways to architect and deploy a cloud application that accepts and handles data and does something useful with it. We will also investigate how to secure the application and the data along the way.

We will also step into the **Information Security Officer's** (**ISO**) role and plan how we could address the security challenges throughout the application and data lifecycle.

We will do this on an actual application example, which we will continue to use in this chapter and subsequent chapters to go in-depth into every aspect of securing the defined cloud architecture.

Scenario

Here is the (simplified) scenario for our (simplified) application.

We are working for a micro-mobility-type organization. Our organization operates electric cars, bikes, and scooters that can be rented by users for mostly short-term use but sometimes for longer-term use as well.

The high-level requirements for this are as follows:

- We are tasked with architecting and securing an application and its services to ingest the data from all the cars, bikes, and scooters and train an AI model based on it, which enables scheduling and executing just-in-time repairs or exchanges of consumable parts, such as batteries.
- We also want to expose the statistics to individual service users – for both fun and payment purposes.

The architecture team has considered the business case and has produced the architecture in Azure (it would work just as well in AWS and GCP).

We need to review it for security practices. In this chapter, we will focus on application and data security. In the subsequent chapters, we will add and expand on the architecture.

Architecture

First, let us consider the high-level application architecture, as shown:

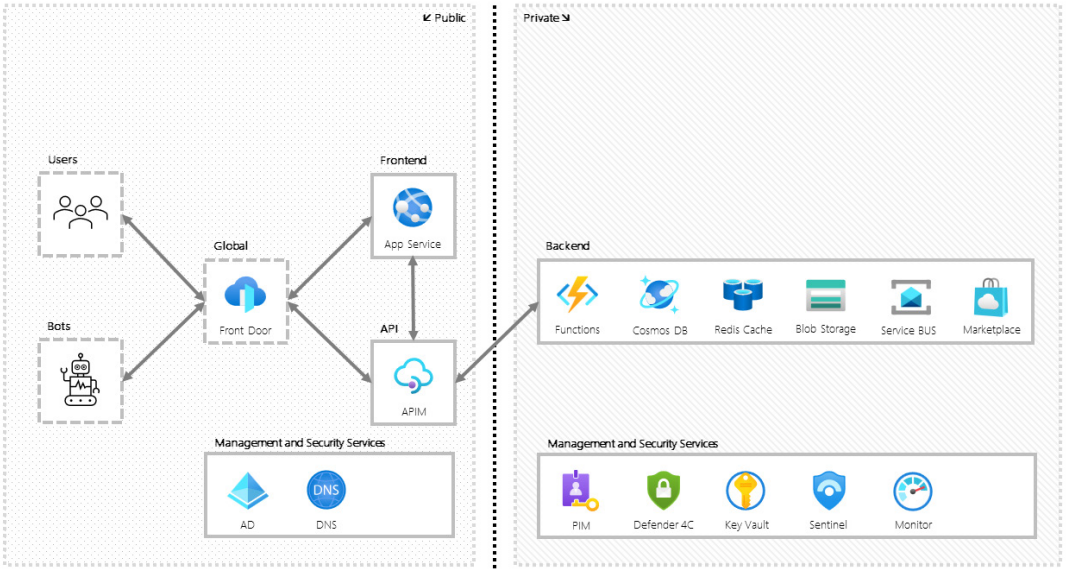


Figure 6.1: *Application Architecture (partial, simplified)*

This part of the architecture is quite straightforward and is in reference to the second requirement above. We will discuss more on the first requirement in the following chapters, and even our second requirement based architecture will change in the next chapters.

Now, we will consider the diagram as depicted.

We are showing here a general serverless, microservices-type architecture.

All the services we will be using to achieve this architecture are divided into public services (that is, accessible from the Internet) and private services (that is, not accessible from the Internet).

Public Azure services are as follows:

- Front Door
- **Active Directory** (**AD**)
- **Domain Name System** (**DNS**)
- App service
- **API Management** (**APIM**)

Private Azure services are as follows:

- Functions
- Cosmos DB
- Redis Cache
- Blob Storage
- Service BUS
- Marketplace (a catch-all for all marketplace and 3rd party services not depicted)
- **Privilege Identity Management** (**PIM**)
- Defender for Cloud (Defender 4C)
- Key vault
- Sentinel
- Monitor

In this chapter, we will consider the security implications of application and data services. We will discuss the rest of the services in the following chapters.

Access patterns

To provide users access to their usage patterns and billing, we want to facilitate these access patterns:

- Users using a mobile application or a website through a browser can sign in to their account and see graphs and maps of their usage patterns as well as download their invoices. The following figure shows the access pattern:

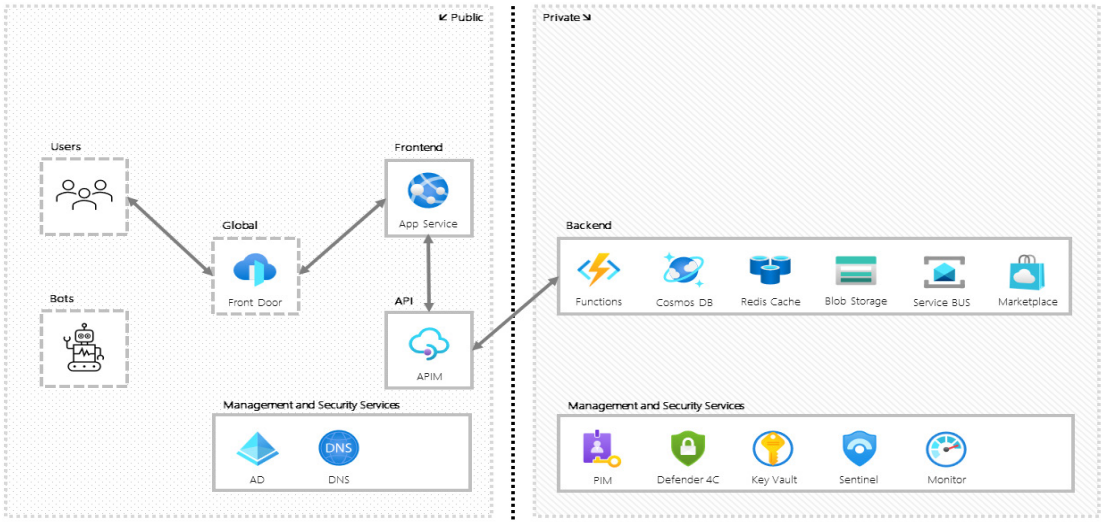


Figure 6.2: *Access pattern – users through a mobile app or a web browser*

- We also allow integration through our APIs so access to usage data, graphs, maps, and invoices can be achieved from other applications (for example, from other applications, prices comparison sites, accountant software, etc.). This means service-to-service communications can only occur through the APIM service and the API layer, meaning different services can only communicate through APIs, and no direct communication is allowed. Refer to the following figure:

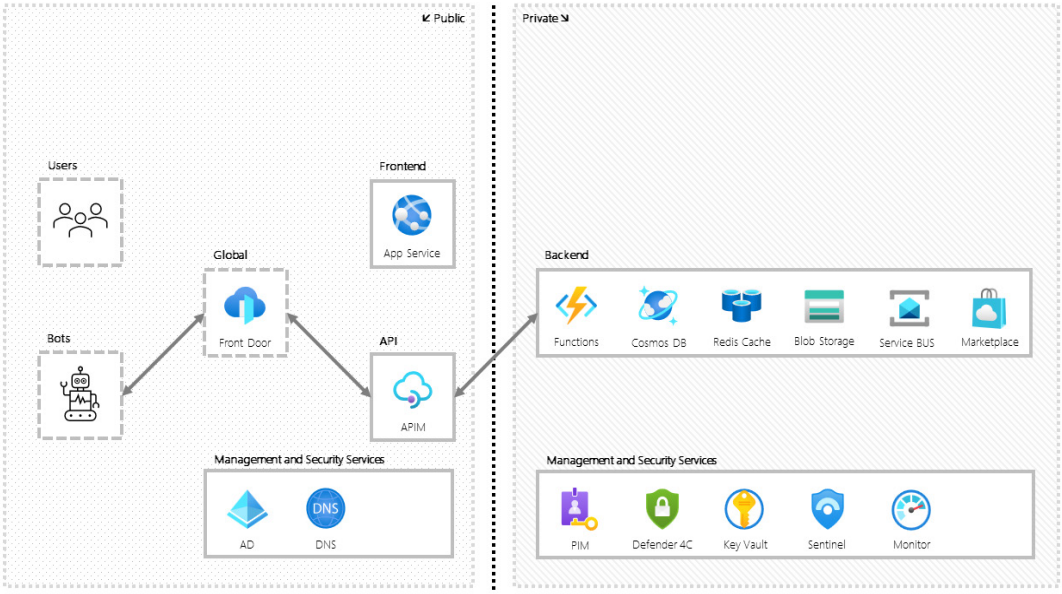


Figure 6.3: *Access pattern – bots (3rd party access) through the APIs*

In both of these cases, it is important to note that all access to private services is done through APIM. No other service has permission to access private services other than APIM. Users authenticate through AD and bots authenticate through the usage of API keys. These are the only two ways in.

That means that both third-party access and our own first-party access (from our mobile app and our website) must use API keys in order to access the data. We will discuss how to protect the APIs themselves in *Chapter 9, Securing APIs, Observability, and Incident Response*.

In order to access the APIM, a request must go through our global services (Front Door, DNS). We will discuss more about securing our network in the next chapter.

Frontend

In front of our application, we have networking and identity services. For now, let us consider the practical ways to protect the app service itself.

App service

App service is a publicly available service (that can be accessed through the internet) and serves our website application.

So, remembering the shared responsibility principles, the responsibility of the cloud provider is to:

- Ensure application resources are isolated from other customers.
- Update and upgrade the underlying hardware and software, which is an Azure PaaS service.
- Ensure that secrets (connection strings, etc.) are communicated within Azure and do not cross any network boundaries, the app service is hosted in our own environment.
- Ensure all communication over the app service connectivity features is encrypted.
- External connections from command line interfaces, consoles, REST APIs, etc., are encrypted.
- Monitor the service 24/7 and protect against malware, distributed denial of service, and similar threats.

Now, you can and should secure the app service in other ways as well.

Nowadays, there is no reason to serve a site from an http connection. Https must be enabled and enforced, and http must be disabled completely. By default, the app service is also available on the https protocol as well on the default domain name provided by Azure.

Obtaining and introducing your own domain and https certificate is your task. Buying domains and certificates for them is something you need to do outside of Azure, so we will not get into it here.

To create an app service in this example, consider just the basics screen:

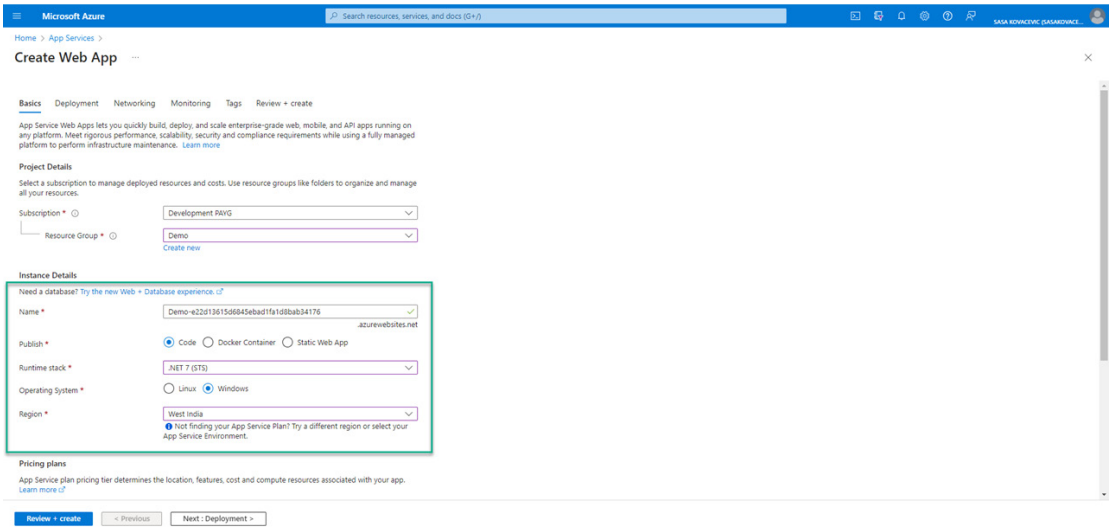


Figure 6.4: *App service creation*

- **Name**: Must be unique as it is also an address (a sub domain) within the default domain Azure provides.
- **Publish**: An option between code (dynamic), a docker image (container), or a static web app.
- **Runtime stack**: Technology stack for the code execution environment.
- **Operating system**: A choice between Linux and Windows (Windows includes a license).
- **Region**: Where the resource should be provisioned (that is, in which of the 60+ Azure regions).

After the app service is created, let us focus on security.

To access the security settings for an app service, follow these steps:

1. Go to the created app service.
2. From the menu of the app service, select **Configuration**.
3. Select **General settings** from the configuration blade menu, as shown in the following screenshot:

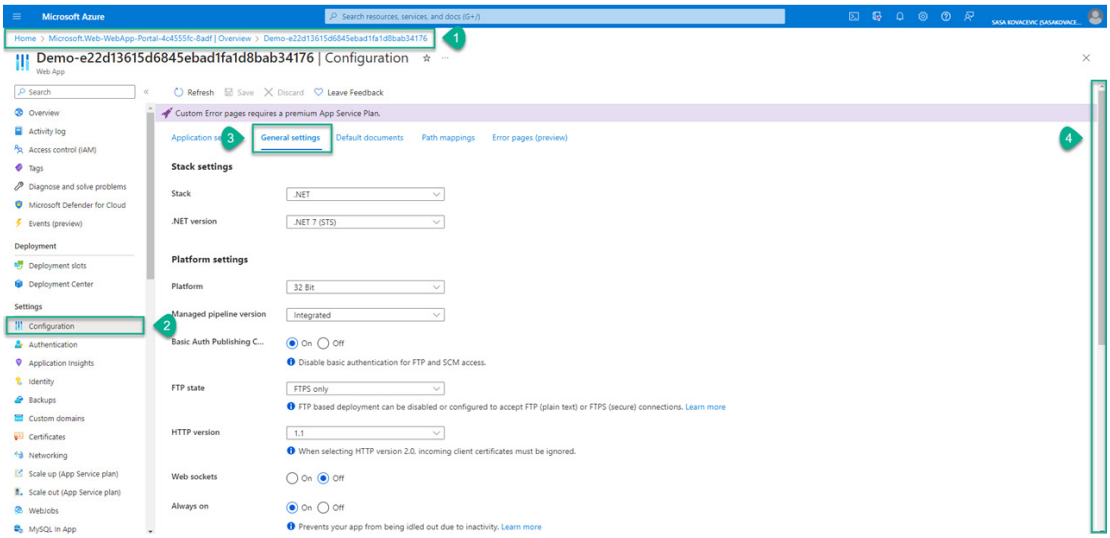


Figure 6.5: *General Settings of the app service*

4. The last part is to scroll down a bit to get to the important security settings.

As you can see, the default options will allow you to provision an app service with less-than-ideal default values. The reason is, as always, backward compatibility.

If you are starting anew, most of these should be changed to more strict settings. If you are working with an existing service, you will need to carefully evaluate and test each of these options to be confident it will not break your application or the expectation of ways of accessing it by your users.

The security settings we want to consider are marked in the following figure:

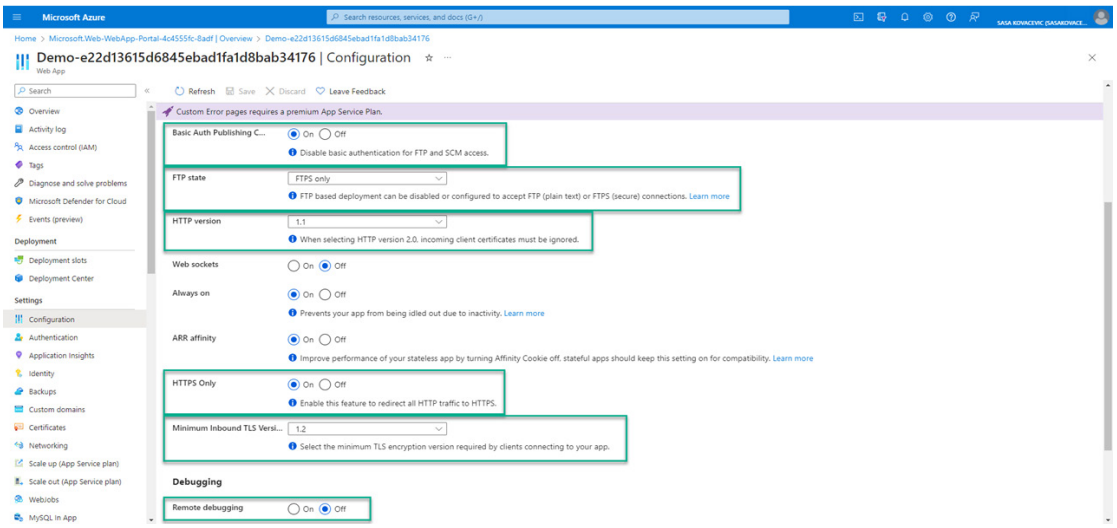


Figure 6.6:

- We want to disable the basic authentication for FTP and SCM access.
- If we need to use FTP to move the files into the service, we should at least enforce FTPS, but it is even better if we can disable FTP access altogether. Firstly, the FTP service is very permissive and quite old, so it should be considered insecure by default. Secondly, if you are not using a particular protocol, service, or feature, it should be disabled to minimize the attack surface that needs to be defended.
- The http version should be set to 2.0.
- Https only should be set to on, meaning no http access will be allowed.
- Minimum inbound TLD should be set to 1.2 (or latest – in the future).
- Remember to also turn off remote debugging as this setting can and will feed a lot of information useful to attackers – in error messages and the underlying security settings and permissions.

In this case, we do not want to restrict IP addresses as we want to provide access to all our customers worldwide, but if you were creating this resource for a smaller group of customers or for internal use, you could restrict access by IP. In a development environment, this might be something you do to enforce access only from your organization's IP ranges.

If you do need to do this, you can follow the given steps:

1. Search for IP in the menu to reduce the number of menu items you need to scroll through.
2. Select **Networking** from the menu.
3. Then select **Access restrictions**, as shown:

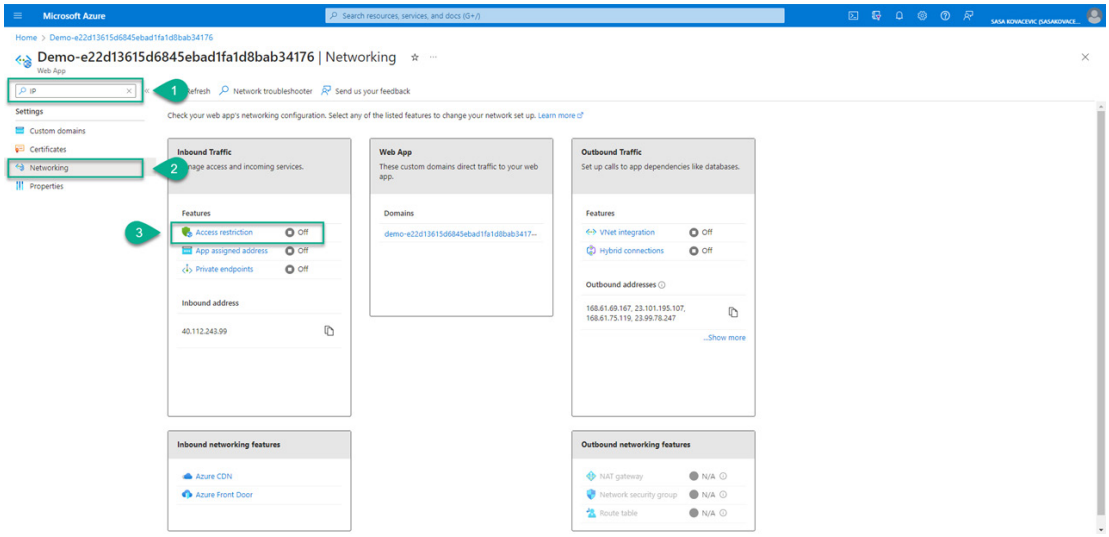


Figure 6.7: *IP restrictions*

This action will lead you to the Access Restrictions screen. Here, you can restrict access as much or as little as you like. We went through similar settings when we were discussing network security groups so we will not repeat it here, other than to note that you can add allow and deny rules. They will be evaluated according to their priority. Testing this is as important as testing networking access to ensure there were no mistakes in setting IP restrictions. To set the IP restrictions, you can add the rules as shown:

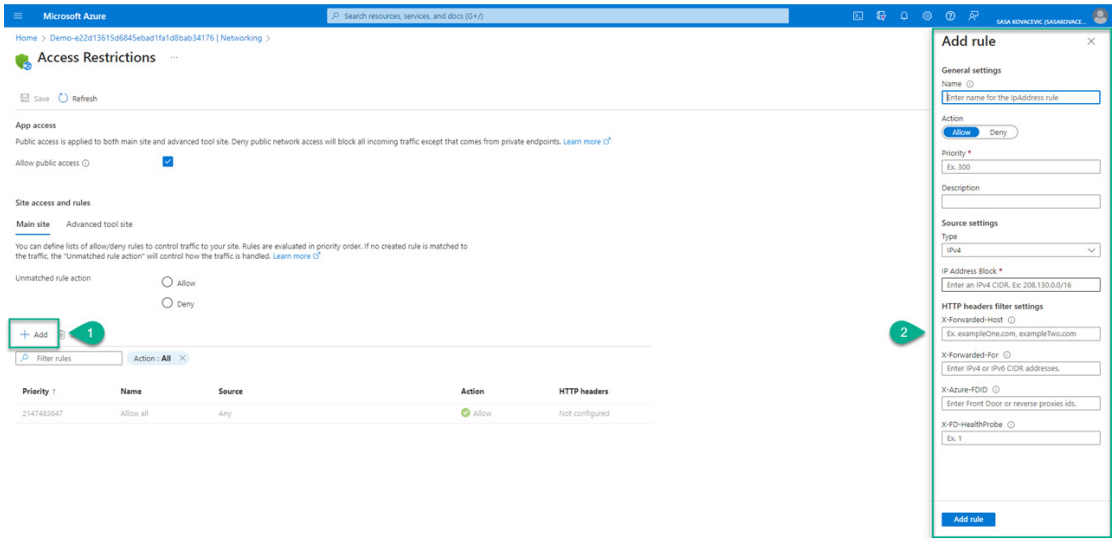


Figure 6.8: *IP restrictions: Adding a rule*

A lot of other cloud services have similar security capabilities, so remember them going forward and apply the same restrictions to other services we will discuss. This way, as you add more varied cloud services into your architecture, you will amass a lot of patterns that are useful across services.

Some of these general rules are:

- Turning off services and features that are not used.
- Enforcing the latest versions of protocols, services, or features.
- Restricting access by IP addresses.
- Ensuring that the default settings are reviewed, evaluated for suitability, and hardened.

Because cloud services are developed in an agile way and go through changes from their introduction to their removal, it is important to regularly check for these patterns and not just at the start of the project or when introducing the service. Changes to cloud services are constant, and security requirements will change. What is today a great idea security-wise might be a terrible one tomorrow.

Backend

In one of the coming chapters, we will focus on serverless services in detail and on storage, so in this chapter, we will focus on the backend services related to data, specifically databases.

In our example case we have chosen Cosmos DB as our database. The reason is that Cosmos DB is a database specifically designed for the cloud. It is a fully managed database service offering single-digit millisecond response time and automatic (instant) scalability, with guarantees on performance at any scale. It is a revolutionary database service ready-made for the cloud.

Guaranteed speed at any scale is especially crucial if the service you and your organization are architecting will need to scale globally.

The database itself is tailor-made to simplify application development with multiple APIs from a database you might have used before – APIs for NoSQL, MongoDB, Apache Cassandra, Apache Gremlin, and PostgreSQL. This means that if you have used any of these technologies before, you can use the same API with Cosmos DB.

In addition, there is the NoSQL API if you have not or do not want to use the APIs related to the above technologies. Schema-less service automatically indexes all data, data that can be supplied in **JavaScript Object Notation (JSON)** – a lightweight data-interchange format, and the database itself will **dynamically adjust the schema** with any JSON fields introduced. This speeds up development as well, and only the JSON payload in the application service needs to change – the database is immediately ready to accept it.

The database comes with the change feed, which is made in order to easily track the feed of changes to the database, giving an easy way to connect other event-based functionality to the database.

From a security standpoint, Cosmos DB, of course, has encryption-at-rest and, in this case, also accepts self-managed keys. The data itself is subject to role-based access control rules ensuring secure access.

As data security is a shared responsibility between the cloud provider and you and your organization, a lot of Cosmos DB security is managed by the cloud provider as it is a PaaS service.

As with any data workload, there are security practices that should be considered before the Cosmos DB database security is even reached. Network security, user authentication, and access controls should already be in place. To mitigate disaster recovery, global (or regional) replication and failover should be in place as well. Local replication as well, in order to mitigate underlying hardware issues. HTTPS should be used when sending JSON payloads, thus also enforcing the encryption-in-transit.

Let us create a Cosmos DB instance:

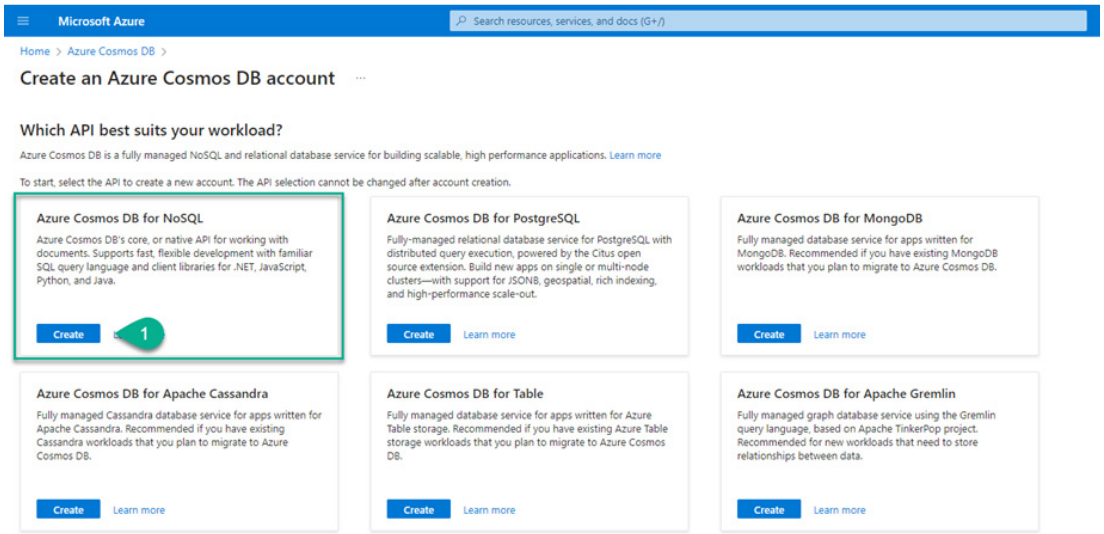


Figure 6.9: *Cosmos DB creation*

- In this case, we will go for the NoSQL API:

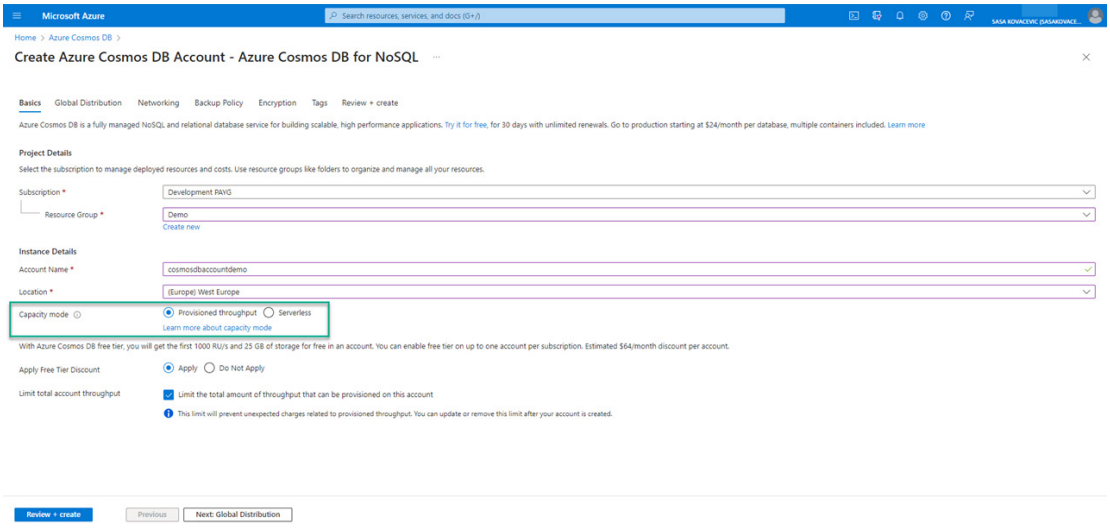


Figure 6.10: *Basic information*

- Notice that it is possible to create Cosmos DB in serverless capacity mode as well. In Cosmos DB, there are differences between the two capacity modes, with the provisioned throughput having more options, so we will pick that one here for this demo, as shown:

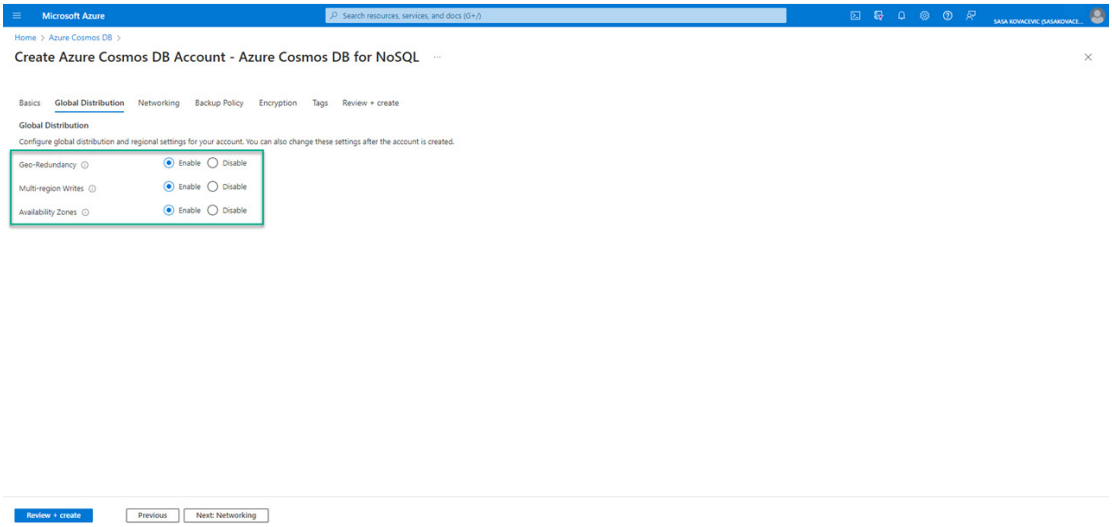


Figure 6.11: *Global distribution*

- Simply by turning any of the global distribution options, we can have this Cosmos DB instance:
    - Be geo-redundant.
    - Have multi-region writes (as opposed to having one write location where all write events go and multiple read replicas).
    - Across multiple availability zones (availability zones are essentially multiple data centers in the same region with separate networking and electricity/ cooling in order to mitigate hardware failures within a region).

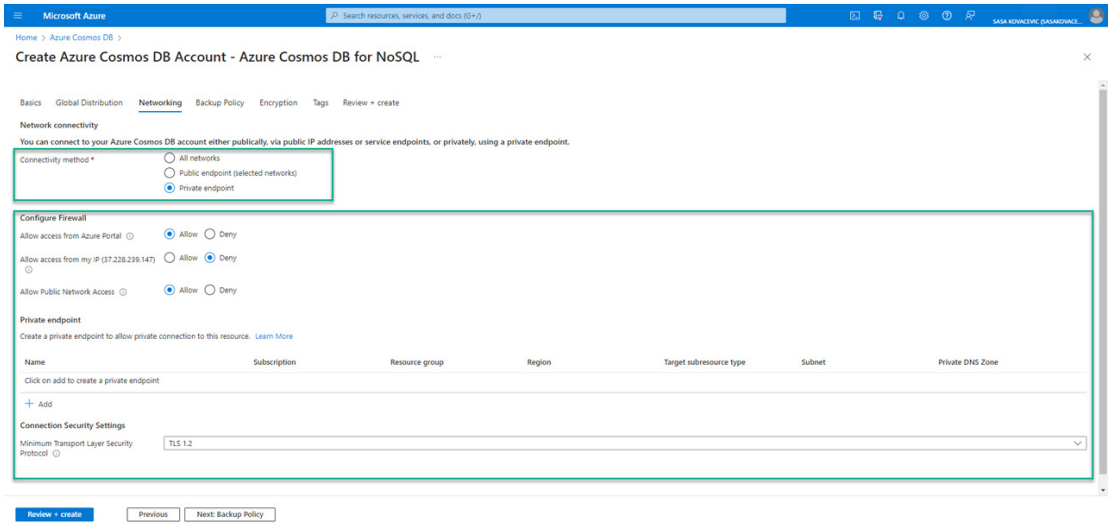


Figure 6.12: *Networking*

- As this is the backend service for our purpose, we want to isolate the database from the internet and have it only be available on a private endpoint within our virtual network.
- We also need to configure the firewall rules on the Cosmos DB instance itself.
- We also ensure, as we did for the app service, that TLS 1.2 is required, as shown in the following figure:

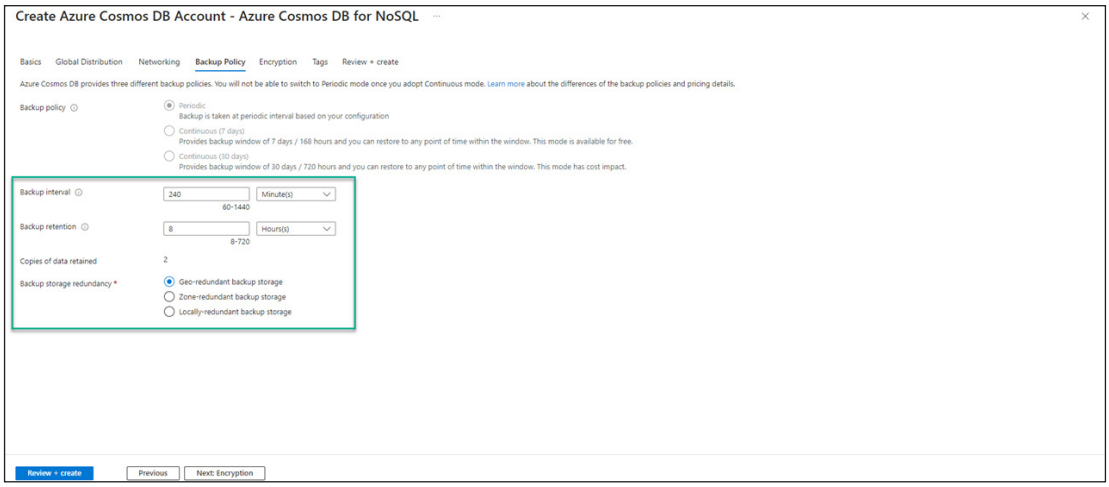


Figure 6.13: *Backup*

- For the purposes of this demo, you can skip the backup, however, this is an important setting for any production workload:

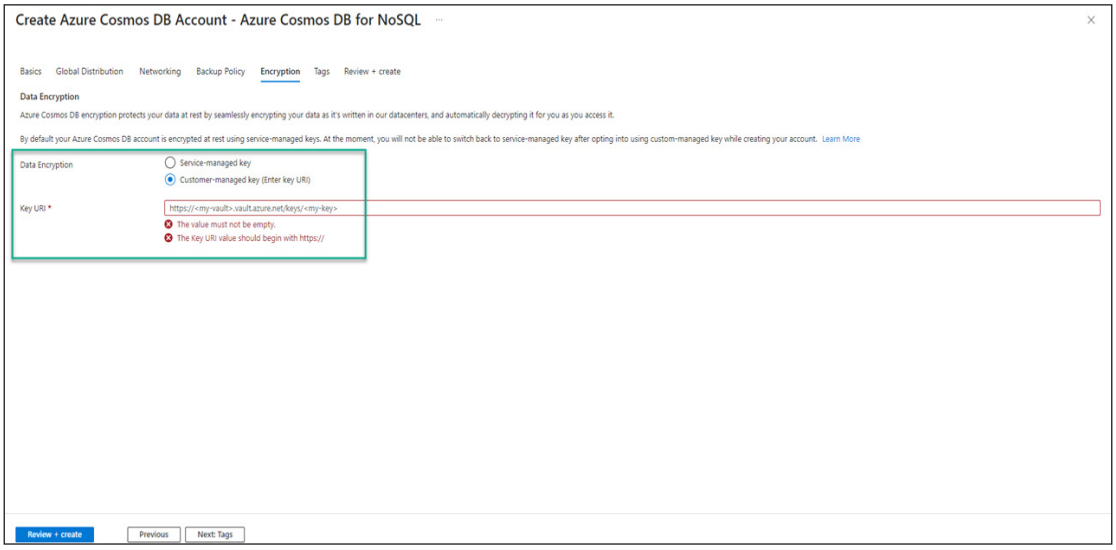


Figure 6.14:

- Securing Cosmos DB even further, we can provide here a link to our own Key used for encryption of data. This is not always necessary as the level of encryption is the same with the managed and user-owned key, but in some instances (regulatory, compliance, elevated risk), the level of security of key management matters, so if that is important, you can set it up here.

Once the database instance is created, you can go into the settings and enable the global data replication if that is something your service needs, as shown:

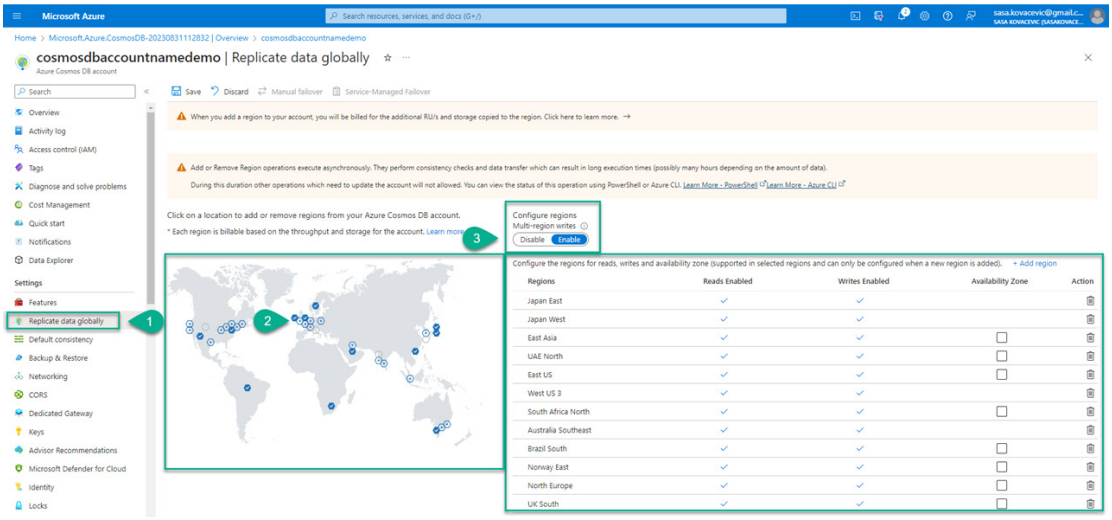


Figure 6.15: *Global data replication*

- Open the Cosmos DB instance that has been created and select the **Replicate data globally** menu item.
- Select on the world map regions you need the data to replicate. Ensure this is both required (due to additional cost for each region) and allowed (from a compliance point of view related to data residency).

Similar exercises should be done to ensure security regardless of which database (or databases if you choose multiple different database technologies, which has validity if you require different ways to store and query data) you choose to ensure that data and the database instance are fully secure.

The most important security functionality is the access granularity. Always (and this includes service-to-service communication) use the lowest needed access permission needed to perform the operation – for example, when writing data to the database give only the write permission to the service principal or the user. When reading, only give read permission, and only give these out when necessary. Too often (as seen in real-world examples), an admin user with a full set of credentials is used for all operations on the database.

Management and security services

You are already familiar with the PIM service, so we will enable it for these workloads as well.

What we also want to enable for this workload (considering that this is not the only workflow we are managing and enabling these services for all of the workloads to have a comprehensive security analysis at our fingertips) is:

- **Defender for cloud:** It is a cloud-native application protection platform. This is essentially a set of capabilities that will check, enforce, and alert on security practices that apply to our workload. This tool and its findings are something your organizations' security team should be utilizing to stay on top of the ever-changing cloud security landscape. This service will also catch any changes in the underlying cloud services that may be security-related.
- **Azure Monitor:** It is a central service for collecting all the telemetry events from cloud services and monitoring and alerting them. It can also be used to collect events from other clouds or on-premises to give you a single pane of glass overview of all your telemetry events.
- **Sentinel:** It is a Microsoft threat intelligence tool that provides **Security Information and Event Management** (**SIEM**) and **Security Orchestration, Automation, and Response** (**SOAR**). This enables you to proactively monitor and manage threats to your workloads and cloud environments. You can create reports, share and analyze them with the teams, and act on the findings. The number one thing it is useful for is correlating alerts across a spectrum of sources into specific end-to-end incidents. It is made of crowd (Microsoft + the community) sourced workbooks, playbooks, and queries that you can use to jump-start your and your organizations' activities. Refer to the following figure:

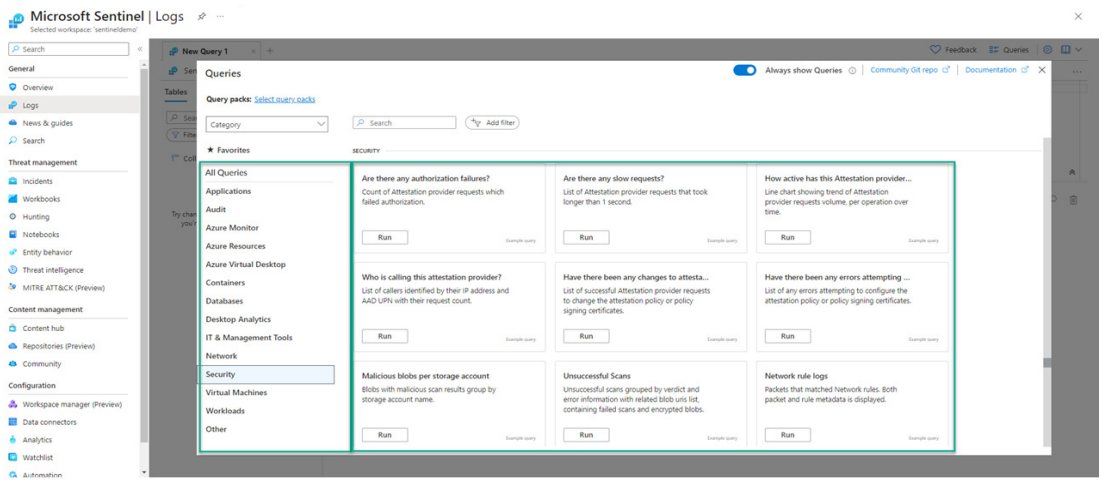


Figure 6.16: *Sentinel queries*

- Under the security section of the queries browser, you can find a lot of ready-made queries to jump-start you onto your cloud security work, as shown:

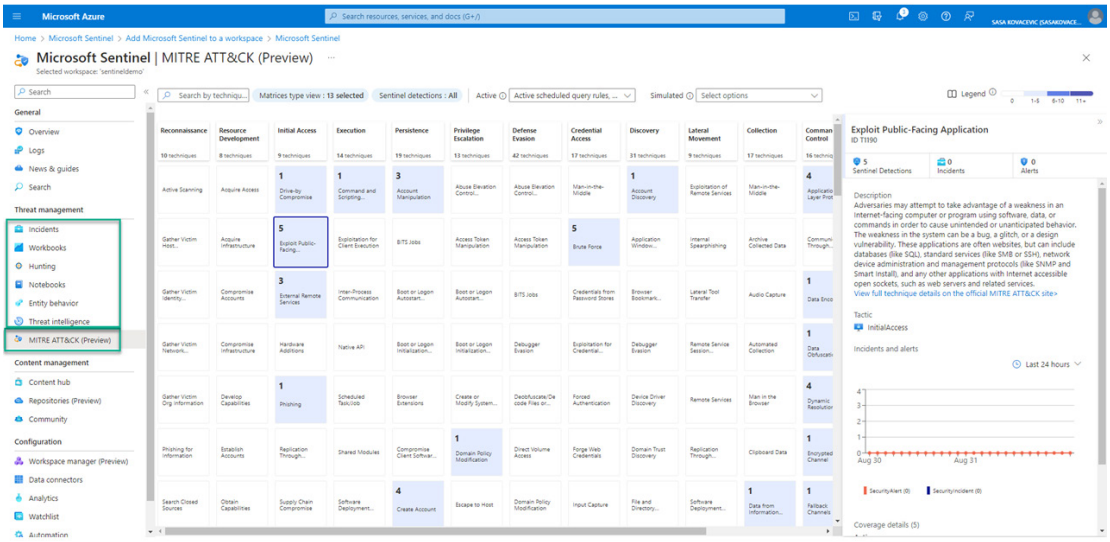


***Figure 6.17**: Sentinel MITRE ATT&CK*

- MITRE ATT&CK® is a globally accessible knowledge base of adversary tactics and techniques based on real-world observations.
- Sentinel provides links to its findings in order to help everyone already familiar with MITRE ATT&CK quickly and easily narrow down the findings.

And finally, let us discuss an especially important topic – where to store the secrets to enable all this secure communication, encryption, and more.

Azure Key Vault service enables secure storage of secrets that can then be accessed from other services and application components. Secrets are stored as name/value pairs. The recommendation for a key vault is to use one per application or service and environment in order to prevent secret sharing between environments (such as production keys with the development keys) and reduce the threat in case of a breach. In our example, this key vault will only be used by this service.

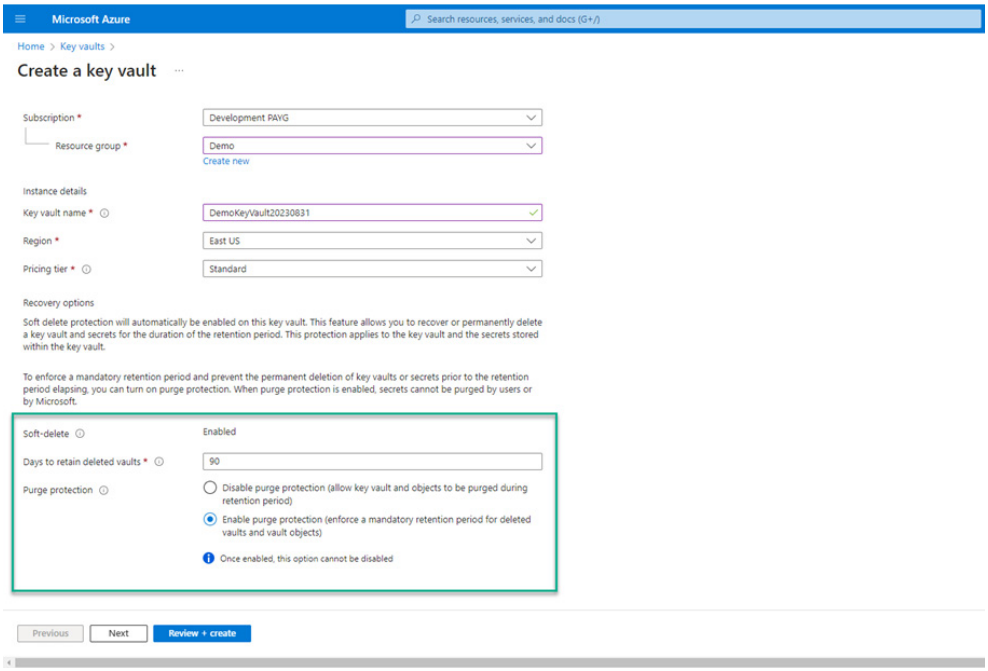


Figure 6.18: *Key vault – soft delete options*

- When creating a key vault, pay particular care to select the correct options on soft-delete options, meaning it is possible to recover keys if they are accidentally deleted – for a while. This will help with any mishaps in production environments, allowing for an easier recovery, as shown:

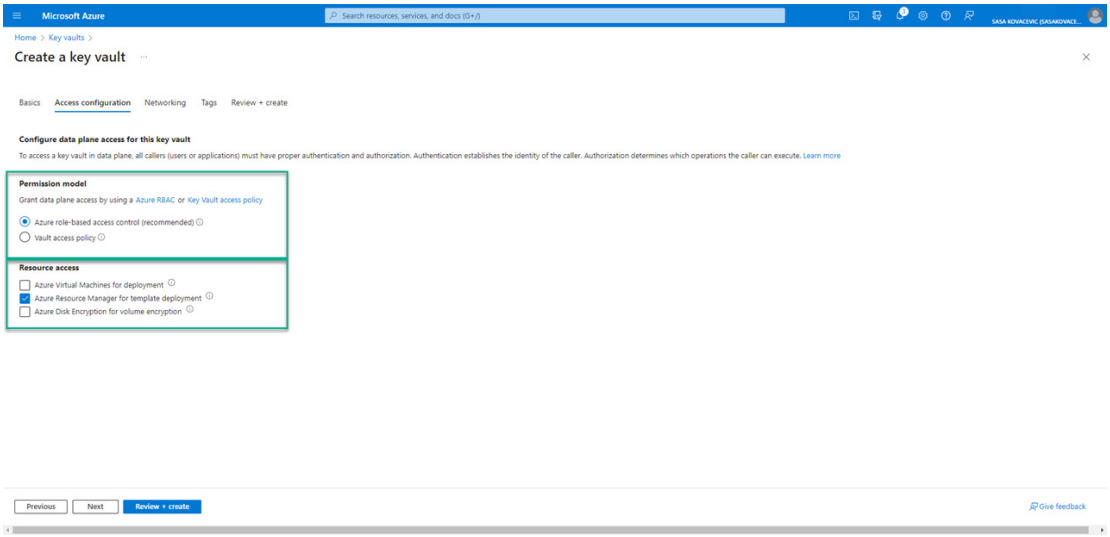


Figure 6.19

- For the permission model, we want to go with role-based access controls again. This allows for more granularity of access to the vault.
- In order to facilitate the usage, some resources might need access, and here is where you can grant it to:
    - Azure **Virtual Machines** (**VMs**)
    - Azure ARM (templates for deploying environments)
    - Azure Disk Encryption (when used for volume encryption)
- In this demo case, you do not need any of these options, but remember that ARM access is likely going to be needed in actual development and production environments for your services, as shown:

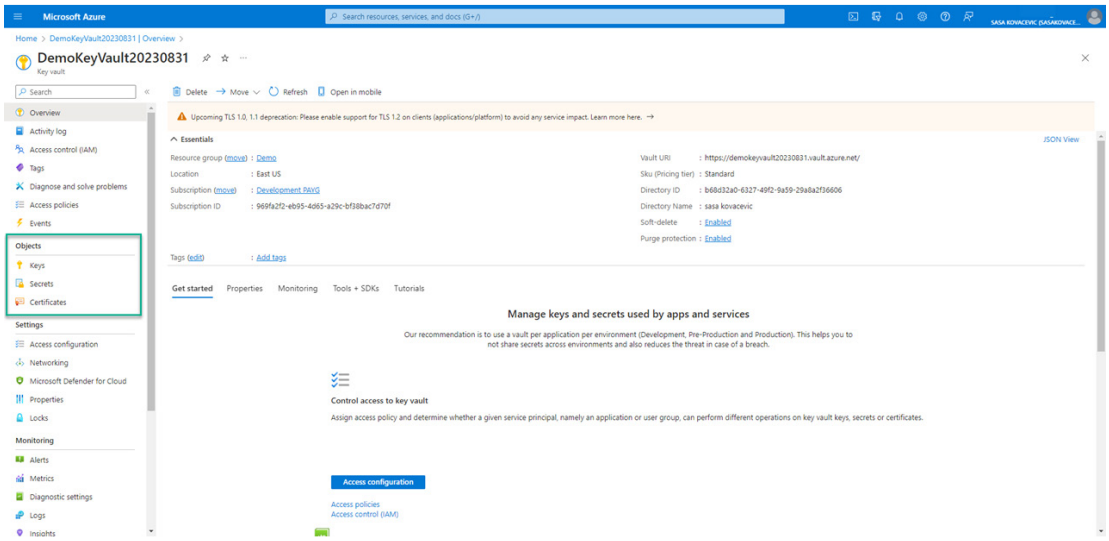


Figure 6.20: *Key vault – objects*

- There are three types of objects you can store in the key vault:
    - **Keys**: Store and manage cryptographic keys. Vaults support software-protected and **Hardware Security Module** (**HSM**) protected keys.
    - **Secrets**: Secure storage of generic secrets, such as passwords and database connection strings.
    - **Certifications**: For management of your X.509 certificates.

Application/service itself

In closing this chapter, let us shift the security focus to the application itself and not on the underlying services that enable it.

In order to ensure application and data security, we must also consider the application and the application code itself to ensure that it is secure enough.

One way to check for this is penetration testing. This can be done internally or you and your organization can find a partner to help you with this. One reason penetration testing is important is to verify that your own application and/or service are not vulnerable, because if they are, that might be an easy way 'in' into your workload, potentially (depending on how exactly your application is vulnerable) enabling privilege escalation and making your security hardening of cloud services – moot.

This is exactly why zero trust is a necessary practice, as you want to ensure that client communication, service-to-service communication, and each individual service from you and your organization or from the cloud provider is independently secure, and there is no implicit trust between any of them. It will go a long way to protect your overall security, even if some local (constrained to one service) security vulnerability occurs.

Conclusion

In this chapter, we started discussing the security implications of a real-world-type application. We focused on applications and data, and we will expand on this in the following chapters.

In the example of a typical Azure architecture, we went through a thought process an individual and an organization should and would go through when evaluating security for a particular service, taking into account the requirements and deciding which services to use, which to make public, which to protect and make private, and – how to evaluate the configuration of each service so it fits into your perfect security definition.

We started following the architecture design that we will modify and expand in the chapters that are coming up to do the same type of security analysis we started in this chapter.

The next chapter is all about cloud security and governance, meaning how to ensure security governance in an organization throughout the journey to the cloud.

Questions

1. What is an Azure app service?
2. Can you find out what are the AWS and GCP alternatives to an Azure app service?
3. When setting up an app service, which security settings are important to consider?
4. Why is blocking the http protocol important?
5. Why is blocking remote debugging important?
6.

7. Why should you use IP restrictions at an app service level?
8. When should you and when should you not set IP restrictions on an app service?
9. What are all the ways to secure a database instance for your backend services?
10. What is SIEM?
11. What is SOAR?
12. How to store and protect secrets in the cloud?

References

- Azure Front Door:
  https://azure.microsoft.com/en-us/products/frontdoor
- Active Directory:
  https://azure.microsoft.com/en-us/products/active-directory
- DNS:
  https://azure.microsoft.com/en-us/products/dns
- App service:
  https://azure.microsoft.com/en-us/products/app-service
- API Management:
  https://azure.microsoft.com/en-us/products/api-management
- Functions:
  https://azure.microsoft.com/en-us/products/functions
- Cosmos DB:
  https://azure.microsoft.com/en-us/products/cosmos-db
- Redis Cache:
  https://azure.microsoft.com/en-us/products/cache/
- Blob Storage:
  https://azure.microsoft.com/en-us/products/storage/blobs/
- Service BUS:
  https://azure.microsoft.com/en-us/products/service-bus/
- Marketplace:
  https://azure.microsoft.com/en-us/partners/marketplace/

- Privilege Identity Management:

  https://learn.microsoft.com/en-us/azure/active-directory/privileged-identity-management/

- Defender for Cloud:

  https://azure.microsoft.com/en-us/products/defender-for-cloud

- Key vault:

  https://azure.microsoft.com/en-us/products/key-vault/

- Sentinel:

  https://www.microsoft.com/en-ie/security/business/siem-and-xdr/microsoft-sentinel

- Monitor:

  https://azure.microsoft.com/en-us/products/monitor/

Chapter 7
Cloud Security and Governance

Introduction

Cloud security and governance is a task to which everyone in the organization must contribute. Managing security across the organization is a part of an organization's governance processes. What they are, how they should work, and how they should not work will be the topic of this chapter.

In the previous chapter, we made a start to ensure the security of our applications, services, and data. How can we prove it? How can we stay on top of it going forward and ensure it stays secure? How can we ensure that any new development and infrastructure activities do not compromise all the great work we have done, and how can we continuously improve our security going forward?

We must now start thinking about security throughout the lifecycle of a service, starting from the time we innovate and think about the new services we want to bring in, through design, implementation, deployment, and operation, and – finally, either through continuous life of the service or until the service retirement and possible future replacement.

We also need to consider how we do development and operations. Are we doing it securely? Have we built-in security at every step of our journey? Are we asking the right questions from different stakeholders and from ourselves?

Structure

The chapter covers the following topics:

- General principles of governance
- Governance
- Threat modeling
- Microsoft SDL practices

Objectives

This chapter has one objective. To ensure you are asking the right questions at the right time and providing guidance and governance along the way to your stakeholders, including engineering, quality assurance, architecture, information security, operations, and other teams, as well as your C-level executives.

Throughout, we will also try to understand the trade-offs that will inevitably be made along the way, so we want to make sure they are the right trade-offs. We also need to be clear on **red lines** – lines we do not compromise on regarding security.

General principles of governance

There are some general principles of governance and some high-level questions we must answer about any cloud workflow.

Identity

To start our governance efforts, we must plan our identity strategy. So, let us start with some patterns we should and should not consider.

Patterns

Here are some common patterns when it comes to identity:

- Implement the principle of least privilege throughout the cloud workload, including the application and control planes, thus preventing data exfiltration.
- Enforce the clear and effective separation of duties, thus ensuring the audit log is accurate and actions are traceable.
- Plan and set up an appropriate authorization for interaction with the cloud environment, thus ensuring operations teams can do their actions at any time with sufficient privileges.

- Centralize our identity management solution, thus ensuring that onboarding and offboarding, as well as managing users through their lifecycle from either the workload or the organization, is a single action that propagates throughout the cloud and organization environment. Speaking of centralized identity, we should also not forget to plan any identity federation scenarios – for example, how will third parties such as contractors access our environments? Same with social media sign-ins, partner access, auditor access, and other users not directly belonging to our organization.
- Eliminate any static credentials, thus ensuring any compromised credentials are quickly replaced and are not in easily accessible places for the attacker to get at them.

Common antipatterns

Here are some common antipatterns when it comes to identity:

- Roll out our own identity solution instead of using a solution specific to the cloud provider and the environment we are deploying, thus ensuring we are now not managing a custom identity solution that will require decades of operational experience and hardening before we can be confident the solution works as expected and is secure enough for production use. Unless the organization builds identity solutions, we should not be rolling out our own.
- Have a few overly broad roles for all users, thus ensuring that almost everyone can do almost everything, which is a terrible idea because if any account is compromised, we are essentially giving the **keys to the kingdom** to the first attacker that manages to do so. This is also a bad idea for all the employees as they can inadvertently either do something wrong with the security consequences or be suspected of doing something they should not be doing in the first place. So, to protect our employees, we should not have overly broad permissions.
- Not having a **Multi-Factor Authentication (MFA**) for all accounts. With the possible exception of the break-glass accounts, all accounts need to go through the MFA flow every time they sign in, activate a roll (remember just-in-time access), or do a potentially destructive action (such as deletion) or security-related action (such as changing a password).
- Inconsistently applying security policies across different providers, accounts, environments, workloads, applications, and services, thus leaving gaps and/or confusion in the way security is maintained. This can also lead to security issues due to assumptions by the staff if one set of services has a security setting a certain way and others do not. People are bad at remembering exceptions, but they are very good at remembering a reinforcing security pattern that is applied uniformly.

Layers

Layers are key to security. We need to have as many layers as possible to ensure that the separation of concerns is fully in effect.

For example, usually, engineers and designers do not need access to the organization's financial records, and the CEO does not need access to DevOps pipelines. So, we must manage these different environments, applications, and/or file shares differently.

On the other hand, we also want to have security layers across our environments, including edge, networking, load balancing, compute services, operating systems, code, and others. Managing them as specific layers and assigning access if and only if needed is key to layering security across the stack.

Hardening resources per type of resource and workload is also something that we must consider. Understanding how each layer is protected is important, and this is something that is easily omitted when reviewing the security of a particular workload, as people tend to focus on one security approach rather than on layers of security. This leads to forgetting important principles such as the Zero Trust principle.

Common antipatterns

Here are some common antipatterns when it comes to layering security:

- Security cannot be an afterthought. Adding security after the fact is hard and sometimes even impossible. Security should be integrated from the beginning of the design and planning phases to ensure it is an integral part of the infrastructure.
- As mentioned, we need as many layers as required – but no more and no less. Overly complex layers lead to the same result as overly lax layers – insecurity, chaos, and assumptions. Creating an overly complex security layering strategy can be counterproductive. Implementing too many security tools and layers can lead to confusion, increased management overhead, and potential conflicts between security measures.
- Inadequate integration of layered security leads to gaps and vulnerabilities sneaking in. Some examples are:
    - If the firewall rules do not align with the **Intrusion Detection System** (**IDS**) alerts, important security threats may go unnoticed.
    - If the IDS operates independently from the **Security Information and Event Management** (**SIEM**) system, we may miss important alerts and fail to correlate events effectively.
    - If one system relies on MFA while another uses only passwords, it creates inconsistencies that attackers can exploit.

  - If access control policies are not synchronized or are inconsistently applied across cloud services and applications, it can result in unauthorized access or data leaks.
  - Deploying security tools that do not communicate or share threat intelligence can result in fragmented security efforts.
  - Keeping data siloed and not integrating data protection measures across all storage and processing components can lead to data breaches. Encryption, data masking, and access controls should be consistently applied across the entire data lifecycle.
  - Failing to integrate cloud service provider APIs into the security monitoring and management tools can result in a lack of visibility into cloud-specific security events and configurations.

- Failing to conduct regular penetration testing, vulnerability assessments, and security audits can lead to a false sense of security. It is crucial to continuously test and validate the effectiveness of security layers to identify weaknesses and address them.
- Creating rigid security policies can hinder our ability to adapt to evolving threats and business needs. Security policies should be dynamic and able to adjust to emerging threats and changing requirements.

Encryption

All data must be classified, and we must protect it using encryption, tokenization, anonymization, and access controls.

Patterns

Here are some common patterns when it comes to encryption:

- We must classify all data, regardless of the source, and then enforce security solutions and patterns for that classification level. If feasible, treat all data as if it is the highest classification.
- Plan for managing the lifecycle of data, including when to completely purge it.
- Encrypt all data at all times.

Common antipatterns

Here are some common antipatterns when it comes to encryption:

- Leaving data around essentially forever. Especially in nooks and crannies that are then omitted from regular security reviews.

- Not having a specific owner of each set of data within our organization. This leads to no one feeling responsible for it, and thus, the data will not be properly secured. If an owner exists and is known and responsible, data is usually better protected.
- Having an organizational culture that is not conducive to challenging the status quo or decisions around data. At all levels of the organization, people must feel they are responsible for data security and, at the same time, must feel safe and free to challenge any decision they disagree with on the basis that it might lead to insufficient protection.
- Rolling out our own encryption schemes. Encryption produces a **garbled, unreadable mess**, even bad encryption. So, it is very difficult to evaluate it for security issues. Whenever possible (that is, always), we must use established libraries (industry standards) and follow the encryption and decryption standards set by it in order to ensure encryption is done properly.

Automation

Automation is the key to everything in cloud computing. So, it is also the key to everything in cloud security. Only through automation can we achieve the security levels we must maintain at scale.

Patterns

Here are some common patterns when it comes to automation:

- Automation of software-based security mechanisms improves our ability to apply security to workflows at the cloud scale. It is also cost-effective, sure, but without automation, there is no security. In any non-trivial size organization, there will be so many workloads that without automation, there will need to be so many manual steps, reviews, and committees to satisfy that nothing will ever get done or worse – security requirements will not be met. It is a catastrophe waiting to happen.
- Security architectures, patternable and ready to go for the organizations' typical workflows, need to be created ahead of time. This ensures that they can then be applied without delay when a new workflow comes along.

Common antipatterns

Here are some common antipatterns when it comes to encryption:

- Forgetting to automate security from the first idea, through the first code commit, to when the production workload is live and available.
- Failing to maintain a human oversight role in automation can be problematic or

to have human experts who can intervene in exceptional situations, troubleshoot issues, and make critical decisions. This is, for example, where your **Site Reliability Engineering** (**SRE**) team comes in regarding site reliability – which includes security.

- Computers will do what we tell them to do, and they will do so efficiently. Implementing automation without thorough testing can lead to unintended consequences. Lack of testing can result in errors, security vulnerabilities, and service disruptions. We must always test automation scripts and workflows in a controlled environment before deploying them in production.
- While automation is great, it is only great at the intersection of usefulness and flexibility. Creating automation scripts or workflows that lack flexibility can lead to challenges when the requirements change (and they always change). Ensuring automation solutions are adaptable and can accommodate evolving business needs and technologies.
- Automation is the first priority. The second is – documentation. Neglecting proper documentation and ongoing maintenance for automation scripts and processes can lead to **automation debt**. Without documentation, it becomes challenging to troubleshoot issues, onboard new team members, and ensure the long-term sustainability of automation solutions.

Monitoring

The importance of monitoring cannot be overstated primarily because it is impossible to act or react to a piece of information if you do not have it.

We must monitor all changes in the environment in real-time or near real-time to act on them. As mentioned, automation should be in place for all the reoccurring events that can be automated away, but people (and more recently artificial intelligence as well, to assist people) should be in place and react if they need to.

Logging and telemetry should be collected from all services, all environments, all applications, and services and centralized so that the SRE and security teams can review and browse the events and set up alerts when they need to.

Auditing actions done in the environment is important from the regulatory perspective, but also in order to have a complete traceability of every action taken by automation and manual action.

Common antipatterns

Here are some common antipatterns when it comes to encryption. These are extremely common and yet crucial to get right:

- Lack of clarity in key performance indicators. For example, if we see in telemetry that the CPU is at 80%, or memory is at 99% or a queue length is 25 thousand - what exactly does that mean; is the service running well or not? Knowing beforehand what the values should be and automating actions (for example, scaling up or out) will lead to better outcomes than determining this while the service is live. Or worse yet, guessing.
- Not having a constant health check on each service that is smart enough to check all the components and return an actual health check for the services – having considered everything from DNS and frontend to backend and databases all the way to third-party health. Usually, this health check returns an evaluation such as ok, not ok, performance degradation, security alert, or unknown. These are more useful items to act on, as opposed to a memory percentage.
- Making cost considerations the primary factor in deciding what telemetry events to ingest. This will lead to premature optimization of ingested telemetry. Make the default sensible; for example – collect all telemetry for two years, then evaluate this quarterly and see if it makes sense to maybe take a sample of telemetry to save on some costs. Be very careful not to sample security events, though, as you might miss some important information that might be useful in threat hunting or after-the-fact investigations. Data should be retained for long enough so that we can compare and/or trace events through the system and have all the data available for any forensic investigation. Sometimes forensic investigations happen quite a while away from the current time, so having that data available is a must.
- Relying too much on alerts, meaning reacting to events rather than proactively monitoring the environment, is another common antipattern that is present in organizations with not enough operational maturity.
- If there are too many alerts, or they occur too often, or they are alerts that do not require action or action cannot be taken, it leads to alert fatigue. It is the equivalent to **crying wolf**.

Response

Preparing for a security event and the careful, planned response to events is something that often is not considered.

Make no mistake, security events, small and large in impact, will happen. How we prepare for them and how we react in these instances will mean a difference between mitigating an issue and making it (sometimes) unimaginably worse.

Patterns

Here are some common patterns when it comes to responding to a security event:

- Running regular, frequent, and varied security exercises and simulations are the only way to prepare for actual events. No amount of thought and planning is enough – without simulations.
- Running simulations in isolation, without legal, ISO, engineering, SRE, architecture, IT, marketing, PR, and other teams, as well as C-level executives – is a terrible idea. Simulations must be as close to real response as possible to be useful. As we will be running many simulations, it is acceptable that you cannot get everyone every time, but again – simulations should be as close as possible whenever possible.

Common antipatterns

Here are some common antipatterns when it comes to a security event:

- Lack of an incident response plan. A well-documented, tested response plan is the only thing that stands between a proper response and chaos.
- Not defining roles and responsibilities beforehand will lead to confusion, chaos, and delays in response. Coordination across the team can only be achieved if every member of the response team knows their roles and responsibilities and has practiced them often.
- Inadequate knowledge of organizations' own services and cloud services will prevent proper response and can only be to the detriment of the workflow and to the detriment of security. So, learning everything about the services that are running is crucial. Or better yet, the team involved in building and deploying the services is also a part of the security response as well.
- Neglecting to consider regulatory compliance requirements and not reporting incidents promptly to relevant authorities can lead to legal and financial repercussions. Compliance with data breach reporting requirements is crucial to avoid potential penalties.

Governance

What is security governance? Security governance is the process of supporting business objectives through defining policies and control objectives to do one thing – manage risk.

The controls we implement are there to manage risk. Risks can be on a scale of likelihood, between likely and unlikely, and on a scale of value (in monetary terms) between low and high. Based on these factors, we can prioritize risks and work to report on them and mitigate them.

Governance helps correlate business priorities and regulatory requirements with technological decisions and actions taken.

Hope is not a strategy; it is a traditional SRE saying.

This saying implies that we better have a strategy; otherwise, we only have hope. And - we can implement and act on a strategy, we cannot implement and act on hope.

The good news, of course, is that any strategy is better than no strategy, and with continuous improvement, we can hone that strategy to be better. So even small improvements can add up and slowly but surely, we can do more and do better each time. Initially, this continuous improvement will be with leaps and bounds, but eventually, the strategy will crystallize itself and be validated in our environments, our policies, and our actions, and we will reach a state where we need to update it occasionally when we discover better approach ourselves of when we learn from others' mistakes.

The cloud computing model approach gives us live visibility into our cloud environments, and we can act on prompts by cloud security services and telemetry. This also drives the digital transformation of our organizations into faster and faster turnaround times – some might call this agility.

It is key to remember that security governance is a process, a process of continuous improvement, a process that does not end as long as we are running workloads in the cloud.

Managing risk

In order to manage risk within the governance process, it is important to:

- **Identify risk**: In this phase, we must identify potential risks and/or threats that could affect the organization's objectives, projects, or assets. We must brainstorm and gather input from stakeholders to create a comprehensive list of potential risks. The same principle applies, we can get there iteratively, identifying more and more risks as we go along. We need to document the identified risks, their sources, and their potential impact. Then we can categorize risks into categories that make sense for the workloads and for the organization. This will help with defining who and how the analysis of each risk is done. We can bring in industry trends, regulatory requirements and changes, market conditions and other factors that can help with defining and explaining the risk in more detail.

- **Assess risk:** In this phase, we must evaluate the risks and determine the likelihood value and other crucial parameters as they apply to the organization. One very important metric is how quickly the risk materializes and its velocity. Then, we can prioritize risks, initially just by considering them as they are and in the context of the industry, regulation, etc., but then also by running simulations for a more thorough assessment. This will help us understand each risk in detail and might guide us on how to approach mitigating it. Each organization has a risk tolerance they can accept in general, and now we must assess the risk tolerance in each workload we are considering, as it can vary wildly on the basis of different workflow categories.

- **Control risk:** In this phase, we must implement control measures, for example – security policies, technical safeguards, and governance processes if they need updating. The goal is to reduce the likelihood or the value or both of each risk we have identified. We can also purchase insurance, for example – cybersecurity insurance – to further mitigate the risk. That would be an example of a risk transfer. We can also do risk avoidance by changing plans, timelines, business strategies, technologies, staffing levels, or anything else that might help avoid the risk. Sometimes, we must accept the risk; sometimes, there is nothing we can do or nothing that we want to do to help mitigate the risk, so we must accept it. As long as we are aware of it and of our decision to accept it, we can move on. We can, of course, have contingency plans for all risks if they change in likelihood or value.
- **Review controls:** In this phase, we must continuously monitor the risks, the measures we put in place, and any changes to the identified risks or new risks making themselves known. It is not unusual for risks to change, so we must be ready for it. Conducting regular audits of these risks will help us validate that controls are still valid and that no further action is needed. In security, this is typically rarely the case, so we might need to go through these four stages of security risk management a few times before the risks stabilize. Continuously improving our risk identification, assessment, control, and review is the only way to stay on top of the ever-changing landscape of cloud security.

Here is a very simple diagram showing the process of risk management. It is important to compare it with the next diagram. They are very similar:



Figure 7.1: *Risk management*

Specifically, in cloud security there is a technique very well suited to discovering security risks. It is called threat modeling.

Threat modeling

Aside from general risks that ought to be planned during any project, security risks are as numerous and as complex as one can imagine. To help simplify and properly assess security risks, we use the threat modeling technique.

Threat modeling works to identify, communicate, and understand threats and mitigations within the context of protecting something of value.

A threat model is a structured representation of all the information that affects the security of an application. In essence, it is a view of the application and its environment through the lens of security.

Threat modeling can be applied to a wide range of things, including software, applications, systems, networks, distributed systems, Internet of Things (IoT) devices, and business processes.

- Victoria Drake

Here is a very simple figure showing the process of threat modeling. It is important to compare it with the previous figure:



***Figure 7.2**: Threat modeling*

As you can see, threat modeling aligns to risk management in that there are four steps we need to go through or four questions we can ask ourselves:

- What are we working on?
- What can go wrong?
- What are we going to do about it?
- Did we do a good enough job?

These four questions align quite well with:

- Identify
- Assess
- Control
- Review

Everyone in the organization that is concerned with privacy, safety, and security needs to start recognizing details of our security model, how our services work, and so we can start to recognize everything that can go wrong with our workflow from a security standpoint.

We threat model early and often, as early and as often as makes sense for a particular workflow. This helps us identify issues very early on, and this helps mitigate them earlier and more cost-effective.

The threat model we produce delivers threats as it outputs, which then informs our decisions as to how to mitigate them.

A threat categorization such as STRIDE is useful in the identification of threats by classifying attacker goals such as:

- Spoofing
- Tampering
- Repudiation
- Information disclosure
- Denial of Service
- Elevation of privilege

There is one key value on the threat modeling manifesto site that it makes sense to call out:

Doing threat modeling over talking about it. *-Threat Modeling Manifesto*

Here are a few tips for doing threat modeling, ordered by (somewhat depends on your proclivities and your and your organizations' maturity) increasing difficulty of implementation:

- **Start with Data Flow Diagrams (DFDs):** Begin with simple DFDs that illustrate the flow of data through your system. Identify the main components and data stores involved.
- **Identify assets:** Determine what needs to be protected. This could be sensitive data, user accounts, infrastructure components, or any other critical assets.
- **Identify threats:** List potential threats and vulnerabilities to your assets. Start with common issues like unauthorized access, data breaches, or denial-of-service attacks.

- **Assess attack vectors:** For each threat, identify potential attack vectors or paths that adversaries might use to exploit vulnerabilities. Consider both technical and human-based attack vectors.
- **Prioritize threats:** Prioritize threats based on their potential impact and likelihood. Focus on the most critical threats that pose the greatest risk to your assets.
- **Mitigation strategies:** Develop mitigation strategies for each high-priority threat. This could involve security controls, best practices, or architectural changes to reduce the risk.
- **Iterate and refine:** Continuously refine your threat model as your system evolves. New threats may emerge, or existing ones may change in significance.
- **Consider advanced threats**: Once you have mastered the basics, delve into more advanced threat modeling techniques, like threat trees or attack graphs, to model complex attack scenarios.
- **Include security experts:** Involve security experts in the threat modeling process. They can provide valuable insights and help identify threats that might be overlooked.
- **Automated tools:** Consider using automated threat modeling tools and platforms to assist with complex systems. These tools can help analyze threats and generate reports.

There is no substitute for doing threat modeling, and it is easy to start as it is easy to (attempt to) answer the four questions we posed. To demonstrate this principle, let us consider the service we started securing in the previous chapter, as shown in the following figure:

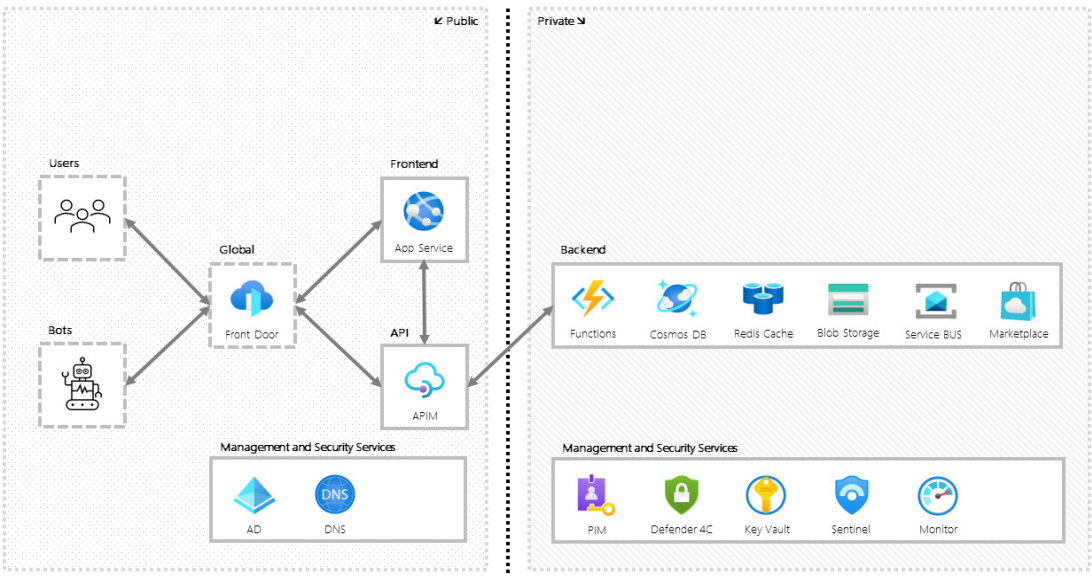


Figure 7.3: *Architecture*

That is our architecture, so let us now apply a (simplified) threat modeling process to it.

We have recreated the previous diagram in slightly more but still quite a high-level detail, as shown:

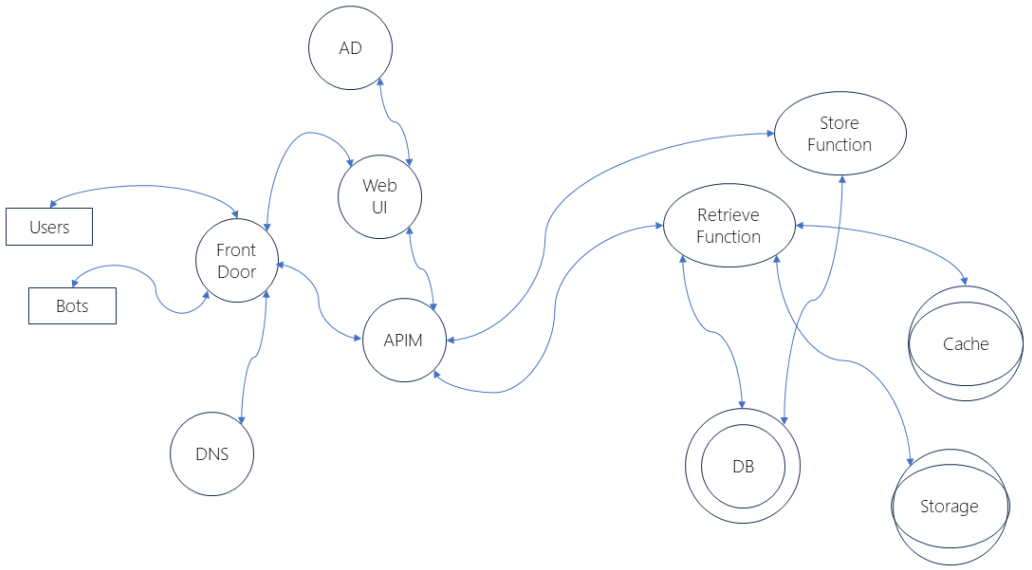


Figure 7.4: *Threat model – step 1 (simplified)*

- We still have all our components and services, as shown:

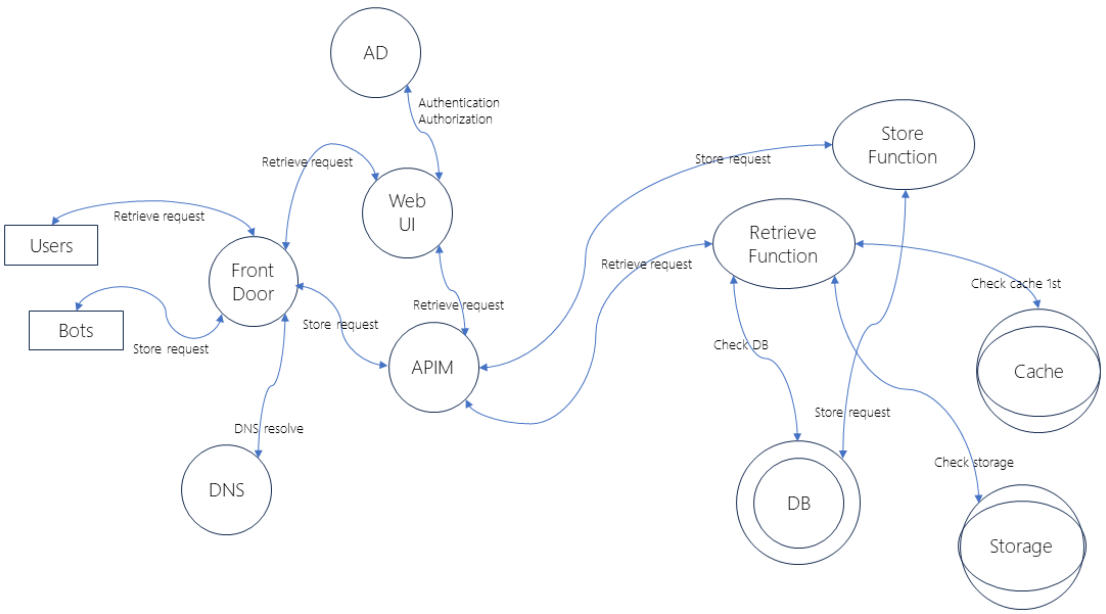


Figure 7.5: *Threat model – step 2 (simplified)*

- Then, we name requests we need to cater to:
    - Retrieve data
    - Store data
    - (others are now shown here, but can be discussed in a similar, separate diagram):

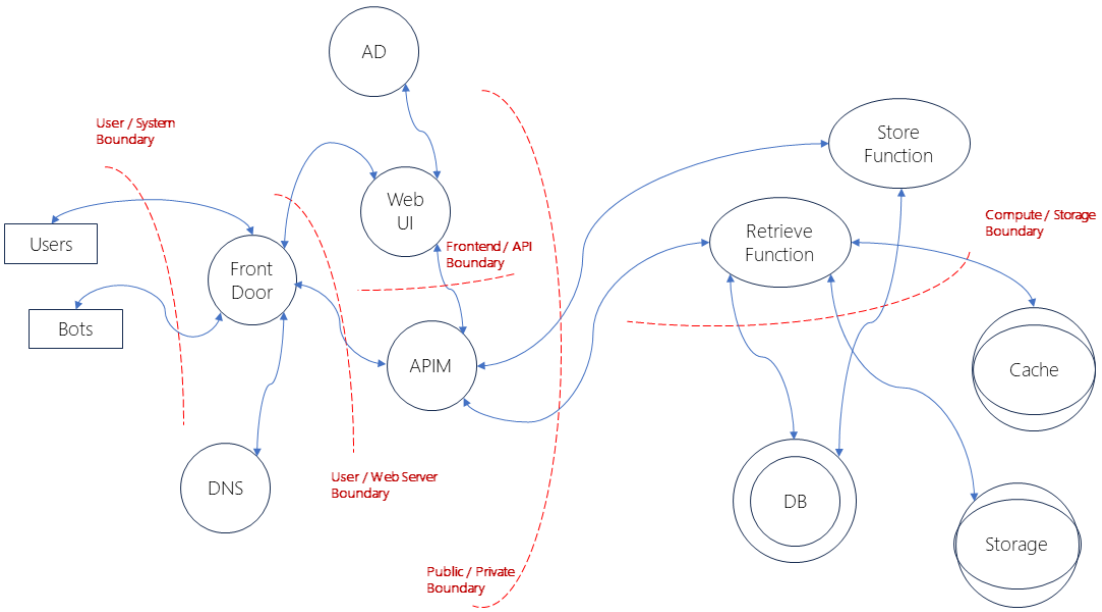


Figure 7.6: *Threat model – step 3 (simplified)*

- Then, we have added all the boundaries we perceive:
    - User / System boundary at the entrance to our system
    - User / Web Server boundary between the global DNS services and our servers
    - Public / Private boundary to remind us of private and public services
    - Compute / Storage boundary as we are doing fundamentally different things with these services – computing results and doing some business logic processing versus storing data

Now, we can proceed with the threat modeling based on the data shown.

Let us answer the four questions.

What are we working on?

On a high level, we are working on a service that accepts and stores data and retrieves it on demand. Here, we would describe the service in detail, but since we have done that

previously, we will not duplicate that here. This is also a normal pattern in threat modeling – if we already have proper documentation, we can refer to it.

What can go wrong?

Here we would discuss and document all the risks we identify, classifying them with STRIDE (remember: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege).

We will not go into a comprehensive list of all findings here, but let us look into one – Denial of Service:

- Denial of Service for us would happen on the user/system boundary and would propagate to the UI and API layers.
- We identified this risk and have concluded it is likely to happen, and since it could disrupt legitimate users, the value is also high (an exact value in currency is preferred here, so just for the sake of this exercise, let us say it is $1 million).

What are we going to do about it?

Now that we know about this risk, what are we going to do about it? We do not want to accept it as it might render our system unavailable to all legitimate users, so we want to mitigate.

Well, luckily, we can mitigate it, Azure Front Door service and the Azure DDoS protection service exist just for this reason. Follow the given steps:

1. We will turn on Azure DDoS protection. We will also contact our Microsoft support and account teams, and they will help us verify our setup is sufficient for our purposes. Another set of eyes on this is great since this is a high-value risk for us.
2. On the Front Door service, we will investigate (or we get a tip from the Microsoft account team) how we can limit access further:
    a. First, we limit access to our origins only, so only traffic that goes through Front Door is accepted. This ensures that what we do next always applies.
    b. Then, we enable WAF so we can inspect traffic, protecting our system from various and somewhat typical attacks. This ensures that even without custom WAF rules, we are already protected from things generally known as attack patterns.
    c. And finally (for this exercise), we will turn on geo-filtering so that we only accept traffic from the regions where the customers are. We have two deployment stamps in Europe and Asia, so we will restrict each one to accept only traffic to the Front Door from these regions/countries. This

3. Of course, none of the mitigate steps are 100% proof, so we will also decide to monitor the traffic in real-time, and we will document and train our team to recognize legitimate traffic and put procedures in place for them to follow when they identify issues:

    a. We will educate the team on the Front Door and WAF, and they will know how to tweak Front Door settings and WAF settings and even when to reach out to Azure support if things get overwhelming for them. We will define what overwhelming means and the exact procedure for contacting support.

    b. We will develop playbooks for various scenarios, including for persistent DDoS attacks, which may include scaling out, because, as per our previous agreement with Microsoft, we can be refunded for all scaled-out services used to combat DDoS attacks.

Did we do a good enough job?

Initially, it seems the answer is yes, but as we have not yet seen the worst that the internet can throw at us, we also decided to review this risk and the decisions to mitigate it every three weeks and act on any (even small scale) disruption we encounter along the way. Later we may decide to reschedule the review policy to once per quarter as we are more confident that this risk is mitigated, for now.

We will also continue to track changes to Front Door, WAF, and DDoS services to learn of any new advances we can use for our benefit in mitigating this cloud security risk.

Microsoft SDL practices

Finally, before we end this chapter on cloud security and governance, let us discuss a set of practices useful for anyone doing governance.

Microsoft Security Development Lifecycle practices, which are useful for everyone doing governance, in any cloud really.

According to the SDL site, *The **Security Development Lifecycle** (**SDL**) consists of a set of practices that support security assurance and compliance requirements. The SDL helps developers build more secure software by reducing the number and severity of vulnerabilities in software while reducing development costs.*

These practices are divided into 12 sections:

1. Provide training
2. Define security requirements
3. Define metrics and compliance reporting

4. Perform threat modeling
5. Establish design requirements
6. Define and use cryptographic standards
7. Manage the security risk of using third-party components
8. Use approved tools
9. Perform static analysis security testing
10. Perform dynamic analysis security testing
11. Perform penetration testing
12. Establish a standard incident response process

As you might have noticed, we have been going through a lot of these processes and practices in a beginner-friendly and practical way through this book and will continue to, but SDL practices are a good reference point if you want to learn more on specific Microsoft and Azure services that can help deliver these practices.

These practices are generally useful for anyone and in every cloud, but the specific implementation is tied to Azure here, so it might take some effort to achieve the same results in AWS or GCP.

Conclusion

In this chapter, we have discussed cloud security and governance, and while it is difficult to be too specific on how you and your organization should set up these processes as every organization is different, what you should have taken away from this chapter is a slew of questions, patterns, antipatterns, practices and tidbits of information you should consider. How you and your organization choose to document, implement, and operate them is up to you.

In the next chapter we will go back to our service and see how we can secure it in various but still practical configurations and with some different services in the mix.

Questions

Only for this chapter, we will not ask questions here on material learned in this chapter; what we will do instead is ask you to pick a cloud workflow you are working on or considering and ask you a few questions on how you would govern the security of the workflow.

You can go through this question-and-answer exercise a few times for a few different workflows:

1. How would you go about performing a threat modeling of your workload?
2. What considerations for compliance and governance must you consider?
3. What is your formal approach to developing software?
4. Are the workloads developed in a secure way, or are you adding security after the fact?
5. Are the workloads configured securely? How can you verify it?
6. Are you monitoring security-related events? How?
7. What networking security is in place? Is it sufficient?
8. What security controls are in place for accessing the workflow in the cloud environment?
9. How are you managing the identities of staff and users?
10. How are you managing encryption? Where are the encryption keys stored? When are they rotated?
11. Are you using IoT devices as part of your workflow? How are they connecting to the workflow? How are these devices secured physically, and how are they secured logically?
12. How will you respond to security incidents?
13. What is your strategy to respond to your users on questions of general security of the workload?
14. What is your strategy for responding to your users questions about a security incident?
15. Are you doing ML and AI (machine learning, artificial intelligence) as part of your workflow? How are you securing ML/AI components? Did you do a threat model on the ML/AI services?
16. How soon would you know your workflow security is compromised?

References

- OWASP threat modeling article by Victoria Drake:
  https://owasp.org/www-community/Threat_Modeling
- OWASP threat modeling process:
  https://owasp.org/www-community/Threat_Modeling_Process

- Threat modeling manifesto
  https://www.threatmodelingmanifesto.org/
- STRIDE on Wikipedia:
  https://en.wikipedia.org/wiki/STRIDE_%28security%29
- Microsoft SDL (security development lifecycle) practices:
  https://www.microsoft.com/en-us/securityengineering/sdl/practices
- Microsoft threat modeling:
  https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling
- AWS approach to threat modeling article:
  https://aws.amazon.com/blogs/security/how-to-approach-threat-modeling/
- GCP threat modeling podcast:
  https://cloud.withgoogle.com/cloudsecurity/podcast/threat-models-and-cloud-security/

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 8
Authentication, Authorization, Data Privacy, and Compliance

Introduction

We have already discussed Identity Management, and now we will translate that into specific patterns and antipatterns for authentication and authorization. Authentication and authorization are easy to implement. Still, a good one has a very specific set of patterns that have been proven to work, so reinventing the wheel is not what we should be doing here (unless your organization is delivering authentication and authorization services to others as a product in its own right).

Your data should not be everywhere. It must only be in specific places, with specific guidance on access, transfer, and usage. Here, we will discuss those patterns and antipatterns.

General Data Protection Regulation (**GDPR**) in the European Union is a prime example. GDPR is nothing else but a set of requirements for managing data, a best practice, or a pattern. A pattern we should have all been following anyway, but only some of us did. So now there are fines. So now we must pay attention to this reason as well.

While we will not focus here on GDPR or any other regulation in its narrow definition, we will show the patterns and antipatterns when it comes to securing data, processing data, and transferring data and general patterns that will occur and reoccur in different frameworks.

Structure

The chapter covers the following topics:

- Authentication
- Authorization
- Data privacy
- Compliance
- General Data Protection Regulation
- System and Organization Control 2

Objectives

This chapter is all about patterns and anti-patterns. The objective of this chapter is for you to learn and understand which patterns are common for a reason – as in – they are battle-tested, they were proven to work, and they should always be considered. On the other hand, there are also some patterns that were widely tried (for one reason or another) and proven to be, at best, useless and, at worst, harmful.

It is important that we learn from the mistakes of others and from the experiences of others rather than going through a try-fail process from scratch.

Authentication

Authentications determine if users are who they say they are or who they are claiming to be. User requirements are to prove this by supplying a password, PIN, and even things like biometrics – voice, images of face, iris, or a fingerprint. Sometimes, it also includes things like knowledge or possession (for example, security questions and MFA, respectively).

Authentication naturally happens before authorization.

Users can generally see the authentication process and are usually required to interact with it - unless something like a **Single Sign On (SSO**) is implemented. Users can also influence the process. They can change the way they authenticate, their passwords, PINs, and others.

The IT or networking staff are usually responsible for authentication implementation, while users are responsible for providing information when requesting access.

Patterns

So, what are some common patterns that should be considered and implemented with regard to authentication? Here are some specific ones for your applications:

- **Multi-Factor Authentication (MFA):** It requires users to provide multiple forms of authentication, such as a password and a one-time code sent to their mobile device, to enhance security. Remember that SMS (text messages) based MFA is one of the least secure ones, authenticator apps being the most secure ones.
- **Single Sign-On (SSO):** Implement SSO to allow users to log in once and access multiple services without the need to enter credentials repeatedly. This pattern simplifies user management. It also simplifies the lives of your users because they can switch between services/applications seamlessly.
- **Federated authentication:** Use identity providers like Google, Facebook, or enterprise identity providers to authenticate users, reducing the burden of managing user credentials. Do this only if it makes sense for you and your organization. This can also be an anti-pattern, so consider carefully if this is something that would be of benefit to your users and plan accordingly.
- **Token-based authentication:** Issue and validate tokens (for example, JWTs or OAuth tokens) for secure and stateless authentication, promoting scalability and security. This also aligns with how many cloud services expect the process to happen, making integration easier.
- **Passwordless authentication:** Eliminate traditional passwords in favor of more secure methods like biometrics, one-time codes, or hardware tokens.
- **Device trustworthiness:** Assess the trustworthiness of user devices (for example, mobile phones or laptops) before granting access to sensitive resources, adding an additional layer of security.

Anti-patterns

Here are some anti-patterns that you should avoid and that are pretty commonly seen and abused:

- **Storing plain text passwords:** Storing user passwords in plain text or using weak hashing methods exposes sensitive information to potential attackers. In this day and age, this should not be something that is ever done, but you will still run into code examples that do this.
- **Weak password policies:** Allowing weak or easily guessable passwords can expose the authentication process to brute force attacks and unauthorized access.
- **Inadequate session management:** Failing to manage user sessions securely, leading to session fixation or session hijacking vulnerabilities. This is especially true if your audience is using shared computers – traveling, in medical facilities, and in other places where users are likely to use shared computers. This can also be done remotely, so either way, managing user sessions is important. At the very least, provide facilities such as showing users their last locations and/or IPs and devices currently logged in.

- **Hardcoding credentials:** Embedding authentication credentials directly in code or configuration files makes them susceptible to exposure and compromise. An extension to this is committing these credentials to repositories, especially public ones.
- **Lack of brute-force protection:** Failing to implement mechanisms to detect and mitigate brute-force attacks against authentication endpoints.
- **Ignoring session timeout:** Not enforcing session timeouts can leave authenticated sessions open for extended periods, increasing the risk of unauthorized access. Again, especially on shared computers.

Authorization

Authorization determines the level (or granularity) of access that should be given or denied to a user. A prerequisite for authorization is that the user is given or denied access to specific resources ahead of time, either by specifically granting or revoking access or by virtue of a user belonging to a role or a group.

Authorization naturally happens after authentication.

Users are generally not able to see or interact with the authorization processes. They happen in the background and are transparent to the user. Users cannot influence the process unless they are in a role or a group that can grant and/or revoke permissions from others. Usually, users are also unable to grant and/or revoke their own access.

Leadership is responsible for setting security strategies. Different departments and groups define access criteria. Sometimes, this can also be transferred to users – especially in the cloud, multi-tenant, and/or SaaS environments, when in the onboarding process, a tenant admin is onboarded from the customers' organization, and then they are responsible for administering the rest of the users. IT and networking staff are responsible for implementing authorization.

Patterns

So, what are some common patterns that should be considered and implemented with regard to authorization? Here are some specific ones for your applications:

- **Role-Based Access Control (RBAC):** Assign specific roles to users or groups and then define permissions and access levels associated with each role. This pattern simplifies access management by grouping users with similar access needs. Some would advocate for always adding roles to groups, even when there will be just one user initially.
- **Attribute-Based Access Control (ABAC):** Evaluate access decisions based on attributes such as user attributes, resource attributes, and environmental conditions.

ABAC allows for fine-grained access control and dynamic policy enforcement. This is something that another service in the cloud would usually provide, so use it and have an extra layer of protection.

- **OAuth 2.0:** Use OAuth 2.0 for delegated authorization, enabling secure access to resources on behalf of a user. This pattern is especially useful for authorizing third-party applications to access user data. You have seen this often, for example – applications that want to post to LinkedIn, Facebook, or X (previously Twitter).
- **JSON Web Tokens (JWT) claims-based authorization:** Embed authorization information within JWTs, allowing for stateless and scalable authorization checks. JWTs can carry user roles or custom claims for fine-grained control.
- **Policy-based authorization:** Define and manage access policies separately from the application code. Policies can be expressed in a standardized format like **eXtensible Access Control Markup Language** (**XACML**).
- **Time-based authorization:** Incorporate time-based restrictions into your authorization system, allowing you to control access based on specific timeframes or schedules. This is useful for limiting access during maintenance windows or business hours.
- **Role permission inheritance:** Establish a hierarchy of roles with varying levels of permissions. Roles lower in the hierarchy inherit permissions from higher-level roles, reducing the complexity of access control management.
- **Hierarchical resource access:** Implement a resource hierarchy where permissions cascade downward. Users granted access to a parent resource also gain access to child resources, simplifying access control management.

Anti-patterns

Here are some anti-patterns that you should avoid and that are pretty commonly seen and abused:

- **Hardcoding authorization rules:** Embedding authorization logic directly into application code can lead to inflexible and error-prone systems that are difficult to maintain and scale.
- **Overly permissive access:** Granting excessive permissions to users or roles, increasing the risk of data breaches or unauthorized access.
- **Ignoring the least privilege principle:** Failing to grant users the minimum level of access required for their tasks can lead to security vulnerabilities.
- **Lack of audit trails:** Neglecting to log and monitor authorization decisions, making it difficult to detect and investigate security incidents.

- **Inadequate access revocation:** Failing to promptly revoke access when it is no longer needed, potentially leaving sensitive data exposed.
- **Complex access rules:** Creating overly complex access control rules that are difficult to manage and understand can lead to misconfigurations and security holes.

Data privacy

Data privacy refers to the protection of an individual's personal information and their right to control how their data is collected, used, shared, and stored. It encompasses the principles and practices that ensure the confidentiality, integrity, and availability of personal data while respecting the privacy rights of individuals.

Patterns

So, what are some common patterns that should be considered and implemented with regard to data privacy? Here are some specific ones for your applications:

- **Data classification:** Implement a data classification scheme to categorize data based on sensitivity levels (for example, public, confidential, sensitive, and highly confidential). This helps in applying appropriate privacy controls. Remember that if you are unable to classify data correctly at a record level (because, for example – you do not know what users will store in a freeform text input), consider the implications for you and your organization and your users' data and apply the strictest classification.
- **Data encryption:** Use strong encryption techniques to protect data both at rest and in transit. Encryption ensures that even if data is accessed by unauthorized parties, it remains unreadable.
- **Privacy Impact Assessments (PIA):** Conduct PIAs or **Data Protection Impact Assessments** (**DPIAs**) to evaluate the potential risks and privacy implications of data processing activities. This helps in identifying and mitigating privacy risks.
- **Data masking/anonymization**: Implement techniques such as data masking or anonymization to replace or obfuscate sensitive data with dummy or pseudonymous values when not required for processing, reducing the exposure of sensitive information.
- **Data access auditing and monitoring**: Set up robust auditing and monitoring mechanisms to track who accesses sensitive data, when, and for what purpose. This enables quick detection of unauthorized access or privacy breaches.

Anti-patterns

Here are some anti-patterns that you should avoid and that are pretty commonly seen and abused:

- **Data hoarding:** Collecting and retaining excessive data that is not necessary for business purposes increases the risk of data breaches and non-compliance with data privacy regulations. Note that this does not apply to telemetry, logging, and auditing – if they are free from users' data, you can and should keep that data if you can use it for analysis by data models or AI.
- **Lack of data minimization**: Failing to apply the principle of data minimization by storing more data than necessary or retaining data indefinitely without a valid purpose. This might be very similar to the previous one, but both mentalities need to be avoided – you should not hoard data, and for the data you do keep, you should think about why you are collecting it and could you minimize the data collected by excluding certain information.
- **Data silos:** Storing personal data in isolated silos across the organization without centralized visibility and control can result in fragmented data governance and increased privacy risks.
- **Inadequate consent management:** Not obtaining proper consent from individuals for data processing or not allowing them to easily withdraw their consent can result in privacy violations.
- **Failure to delete obsolete data:** Neglecting to delete outdated or unnecessary data can lead to compliance issues and increased data exposure.
- **Weak access controls:** Implementing weak or inconsistent access controls, which allow unauthorized users or applications to access sensitive data, can result in data breaches and privacy violations.
- **Ignoring cross-border data transfers:** Failing to address the legal and regulatory requirements for cross-border data transfers, potentially leading to non-compliance with data protection laws.
- **Insufficient vendor assessment:** Failing to assess and ensure the data privacy practices and compliance of third-party vendors and cloud service providers can lead to shared liability and privacy risks.

Compliance

Compliance, in the context of data privacy and security, refers to the adherence to laws, regulations, standards, and internal policies that govern the collection, processing, storage, and management of data. Compliance efforts are aimed at ensuring that organizations operate in accordance with legal and regulatory requirements, industry standards, and their own internal policies and best practices.

Patterns

So, what are some common patterns that should be considered and implemented with regard to compliance? Here are some specific ones for your applications:

- **Regulatory alignment**: Identify and align your cloud infrastructure and practices with relevant industry-specific regulations and standards, such as GDPR, HIPAA, or PCI DSS, depending on your business domain and/or location.
- **Policy as Code (PaC):** Implement compliance policies as code using tools like **Infrastructure as Code** (**IaC**) and **Configuration Management** (**CM**) to ensure that your cloud resources are provisioned and configured in accordance with compliance requirements.
- **Automated compliance scanning:** Use automated tools and services to continuously scan and assess your cloud environment for compliance violations, vulnerabilities, and misconfigurations.
- **Role-Based Access Control (RBAC)**: Enforce RBAC to restrict access to cloud resources based on job roles and responsibilities, ensuring that only authorized personnel can make changes that affect compliance.
- **Regular compliance auditing**: Conduct regular internal and external compliance audits and assessments to verify that your cloud infrastructure and practices adhere to regulatory requirements and industry standards.

Anti-patterns

Here are some anti-patterns that you should avoid and that are pretty commonly seen and abused.

- **Lack of awareness:** Failing to understand the specific compliance requirements that apply to your organization's cloud environment can lead to unintentional violations.
- **Manual compliance checks:** Relying solely on manual checks and audits without automation can result in missed violations and introduce human error.
- **Inconsistent documentation:** Inadequate documentation of compliance processes, configurations, and changes can hinder audit trails and make it challenging to demonstrate compliance.
- **Ignoring change management:** Failing to implement robust change management processes can result in uncontrolled changes that may lead to compliance violations.
- **Vendor dependency:** Assuming that a **cloud service provider** (**CSP**) automatically ensures compliance without understanding shared responsibility models or the need for additional customer-side configurations and controls.

General Data Protection Regulation

Let us consider an example of GDPR.

GDPR is a comprehensive data protection and privacy regulation enacted by the **European Union** (**EU**) in May 2018. GDPR is designed to provide individuals within the EU greater control over their personal data and to harmonize data protection laws across EU member states. It also has implications for organizations worldwide that process the personal data of EU residents.

Here are the GDPR pillars:



Figure 8.1: *Pillars of GDPR*

Here are the key aspects of GDPR:

- **Scope:** GDPR applies to the processing of personal data of individuals residing in the EU, regardless of where the organization processing the data is located. This means that organizations outside the EU that handle EU residents' data must comply with GDPR.
- **Data subjects:** GDPR grants data subjects (individuals whose data is being processed) several rights, including the right to access their data, rectify inaccuracies, erase data (the *right to be forgotten*), and restrict or object to processing in certain situations.
- **Data controllers and data processors:** GDPR distinguishes between data controllers (entities that determine the purposes and means of processing personal

data) and data processors (entities that process data on behalf of controllers). Both controllers and processors have specific obligations under GDPR.

- **Consent:** Organizations must obtain explicit and informed consent from individuals before processing their personal data. Consent should be freely given, specific, and easily withdrawable.
- **Data Protection Impact Assessments (DPIAs):** For high-risk processing activities, organizations are required to conduct DPIAs to assess and mitigate potential risks to individuals' data privacy.
- **Data breach notification:** Organizations must report data breaches to the relevant supervisory authority within 72 hours of becoming aware of the breach unless the breach is unlikely to result in a risk to individuals' rights and freedoms.
- **Data Protection Officers (DPOs):** Some organizations are required to appoint a Data Protection Officer, an individual responsible for ensuring GDPR compliance within the organization.
- **International data transfers:** GDPR restricts the transfer of personal data outside the EU unless certain safeguards or mechanisms (such as standard contractual clauses or binding corporate rules) are in place.
- **Accountability and governance:** Organizations are expected to implement appropriate technical and organizational measures to demonstrate compliance with GDPR. This includes data protection policies, employee training, and maintaining records of processing activities.
- **Fines and penalties:** GDPR introduces substantial fines for non-compliance, with penalties of up to €20 million or 4% of the global annual turnover, whichever is higher.

GDPR has already had a significant impact on how organizations handle personal data, emphasizing transparency, accountability, and data protection as core principles. Organizations have had to adapt their data processing practices, privacy policies, and security measures to comply with the regulations and protect individuals' privacy rights.

Non-compliance with GDPR can result in severe financial penalties, so it is crucial for organizations to understand and adhere to its requirements when handling personal data.

Some organizations have opted to default to GDPR for all their data processing needs, meaning they have the same processes for EU and non-EU citizens.

Patterns

What are some common patterns that should be considered and implemented with regard to GDPR? Here are some specific ones.

- **Data minimization:** Only collect and process the data that is necessary for the intended purpose. Minimizing data reduces the risk of exposure and ensures compliance with the GDPR's data minimization principle.
- **Privacy by design and default**: Incorporate privacy protections into the design and default settings of your cloud-based services. Implement features like data encryption, access controls, and anonymization by default.
- **Data encryption:** Implement encryption for data at rest and in transit. Use only strong encryption algorithms and key management practices to safeguard personal data.
- **Data access controls:** Use RBAC and fine-grained access controls to restrict access to personal data to only those who need it for their tasks.
- **Data subject rights handling:** Establish processes and mechanisms to facilitate data subject rights, including the right to access, change/correct, delete, or export their data. Ensure timely responses to data subject requests. In GDPR, there is always a deadline. For example, the data breach notification requirement is 72 hours from becoming aware, and requests to change data, amend data, erase data, and so on are on a deadline of 1 month from the request. Check these deadlines and plan for your processes to fit within them.

Anti-patterns

Here are some anti-patterns that you should avoid and that are pretty commonly seen and abused:

- **Ignoring data mapping:** Failing to understand and document where personal data is stored and processed within your cloud infrastructure can lead to non-compliance.
- **Incomplete data records:** Neglecting to maintain accurate records of data processing activities, data flows, and consent records can hinder GDPR compliance efforts.
- **Inadequate vendor assessment:** Not thoroughly assessing the GDPR compliance of cloud service providers or relying solely on their assurances without contractual safeguards can result in shared liability.
- **Insufficient data security measures:** Relying on basic security measures or failing to update security controls can lead to data breaches and GDPR violations.
- **Data retention issues:** Holding onto personal data longer than necessary without a valid legal basis or consent can result in non-compliance with GDPR's data storage limitations.

System and Organization Control 2

System and Organization Controls 2 (**SOC 2**) is a framework for assessing and reporting on the security, availability, processing integrity, confidentiality, and privacy of customer data by service organizations. SOC 2 reports are often used by organizations that provide services to other businesses (service providers) to demonstrate their commitment to security and data protection. These reports help build trust with customers and partners by providing assurance that the service organization has effective controls in place to protect sensitive data.

Here are the SOC 2 pillars:

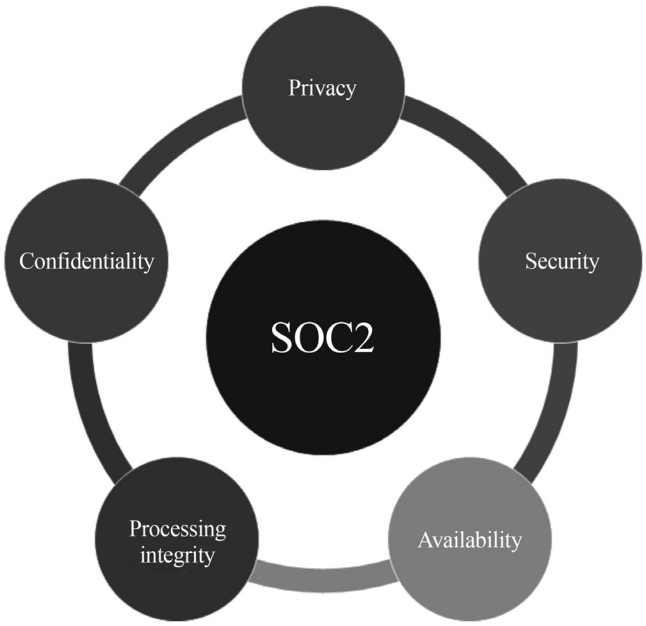


Figure 8.2: *Pillars of SOC 2*

SOC 2 is based on the Trust Service Criteria, which include the following five principles:

- **Security:** The system is protected against unauthorized access, use, or disclosure, and data is safeguarded against security breaches.
- **Availability**: The system is available for operation and use as agreed upon or required.
- **Processing integrity:** System processing is complete, accurate, timely, and authorized.
- **Confidentiality:** Information designated as confidential is protected as required.
- **Privacy:** Personal information is collected, used, retained, disclosed, and disposed of in conformity with the privacy notice and criteria set forth in **the American**

Institute of Certified Public Accountants (**AICPA**) **Generally Accepted Privacy Principles** (**GAPP**).

SOC 2 reports come in two types:

- **SOC 2 Type 1:** This report evaluates the suitability of the service organization's controls at a specific point in time.
- **SOC 2 Type 2:** This report assesses the effectiveness of the controls over a specified period (usually six months to a year) and provides a more comprehensive view of control implementation and operation.

The key aspects of SOC 2 are:

- **Scope and boundary:** Service organizations define the scope and boundary of their SOC 2 assessment, outlining the systems and services covered by the report. This helps customers understand the specific areas of assessment.
- **Third-party auditors:** SOC 2 assessments are typically conducted by independent third-party auditors who evaluate the organization's controls and processes against the Trust Service Criteria.
- **Report distribution:** Service organizations provide SOC 2 reports to customers, partners, and other stakeholders to demonstrate their commitment to security and data protection. These reports are valuable for due diligence and risk assessment.
- **Continuous improvement:** SOC 2 is not a one-time compliance check; it promotes continuous improvement in security and data protection practices. Organizations often update and enhance their controls based on audit findings and changing risks.
- **Regulatory alignment:** SOC 2 can be aligned with various regulatory requirements and standards, such as GDPR, HIPAA, or NIST Cybersecurity Framework, depending on the organization's industry and customer needs.

SOC 2 reports help organizations assess and demonstrate the effectiveness of their internal controls and processes, giving their customers confidence that they are taking data security and privacy seriously. These reports are particularly important in industries where data protection and security are critical, such as cloud service providers, data centers, and SaaS companies.

Before we go into the patterns and anti-patterns, let us quickly look at how cloud providers can assist you in understanding and implementing policies that will handle some requirements that can be automated.

We will do this in Azure, as shown by the following steps:

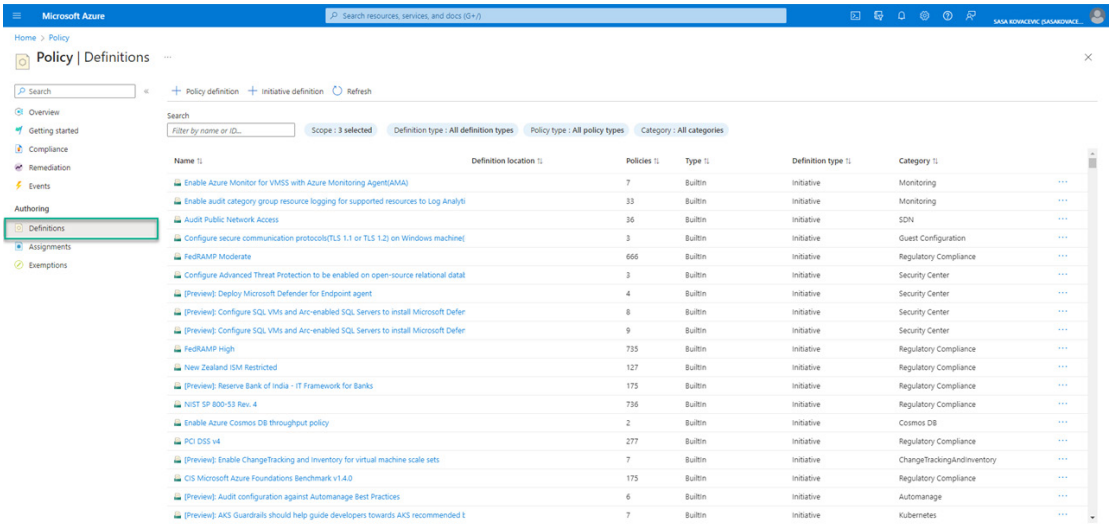


Figure 8.3: *Pillars of SOC 2*

1. First, go to the Azure portal and search for **Policy**.
2. Then, you will be able to see the definitions by selecting the **Definitions** option from the menu, as shown in the following figure:

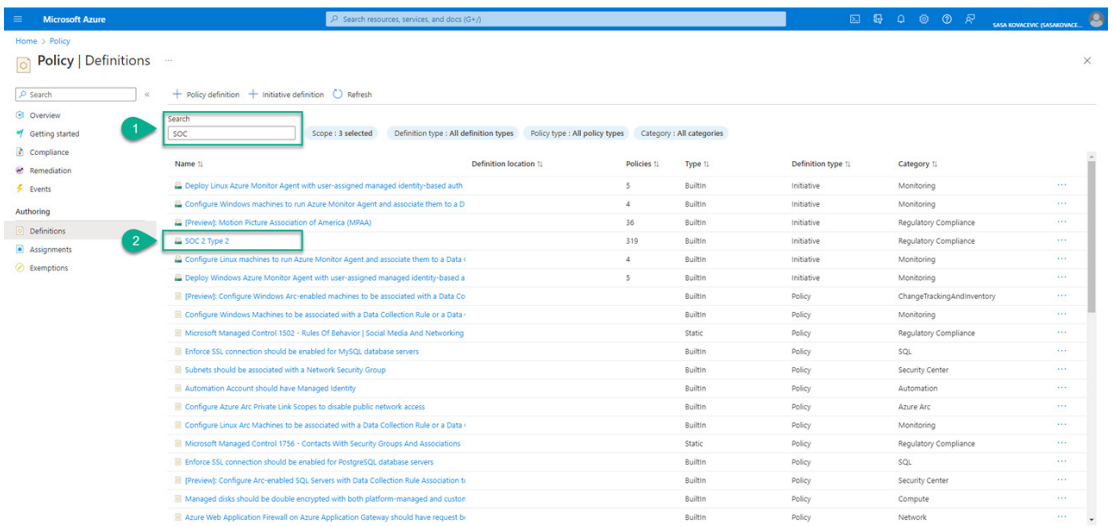


Figure 8.4: *Pillars of SOC 2*

3. You can see and browse through all the policies available.
4.

5. Then select the SOC 2 Type 2 from the list:

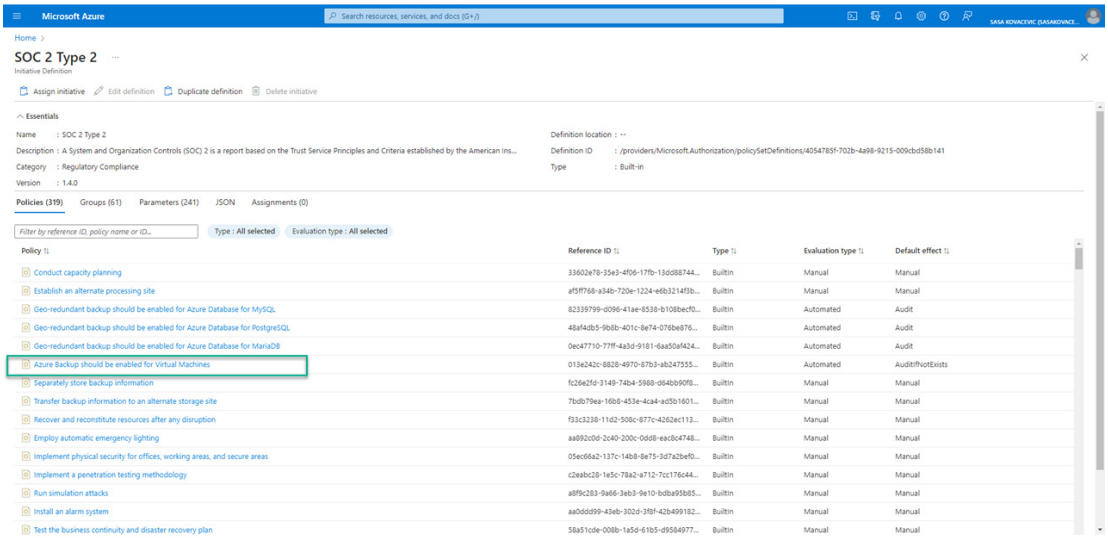


Figure 8.5: *Pillars of SOC 2*

6. You can now see all the policies under the SOC 2 Type 2 initiative.

7. You can see that the Evaluation Type column is telling you if the evaluation needs to be done manually or can be automated. There are a lot of manual evaluations for SOC 2 Type 2.

8. Select Azure Backup should be enabled for Virtual Machines in order to see the actual policy, as shown in *Figure 8.6*. Policies are JSON documents in Azure.

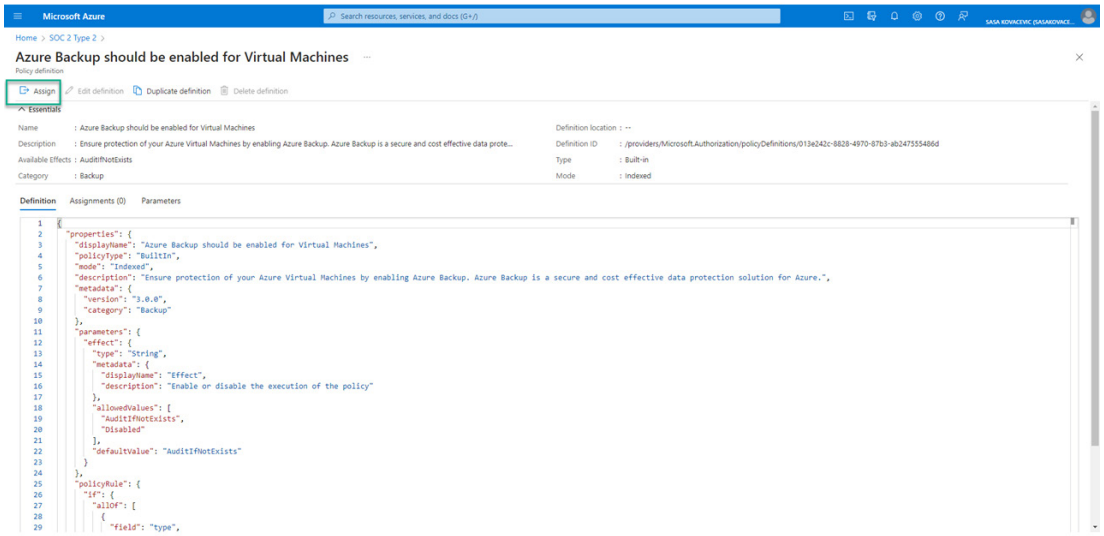


Figure 8.6: *Pillars of SOC 2*

9. Here, you can see the name, description, and the actual JSON of the policy.
10. Of specific importance, here are the available effects. In this case, the AuditIfNotExist effect enables auditing of resources related to the resource that matches the if condition but does not have the properties specified in the details of the then condition.
11. In order to apply this particular one, you can select the **Assign** option from the menu.
12. As you have seen, policies are grouped into initiatives, and you can apply the entire initiative as well from the previous screen.

Similarly, this effect can be achieved in AWS and GCP as well.

Patterns

So, what are some common patterns that should be considered and implemented with regard to SOC 2? Here are some specific ones:

- **Access controls:** Implement robust access controls and authentication mechanisms to ensure that only authorized individuals or systems can access your cloud resources and data.
- **Security Information and Event Management (SIEM):** Set up an SIEM system to centrally collect and analyze logs and events from your cloud infrastructure, enabling real-time threat detection and response.
- **Incident response plan:** develop and maintain a well-defined incident response plan that outlines how your organization will handle security incidents in your cloud environment, including reporting and communication procedures.
- **Continuous monitoring:** Implement continuous monitoring of your cloud environment for security vulnerabilities and compliance deviations, allowing for timely remediation.
- **Third-party assessments:** Engage third-party auditors to conduct regular SOC 2 assessments to validate your compliance with Trust Service Criteria and provide assurance to your customers.

Anti-patterns

Here are some anti-patterns that you should avoid and that are pretty commonly seen and abused.

- **Lack of clear scope definition:** Failing to define the scope of your SOC 2 assessment accurately can lead to ambiguities and incomplete coverage.

- **Insufficient documentation:** Neglecting to maintain comprehensive documentation of security policies, procedures, and controls can hinder the audit process.
- **Inadequate training:** Not providing adequate training to employees and personnel involved in the cloud environment can result in non-compliance due to human error.
- **Overreliance on Cloud Service Providers (CSPs):** Assuming that your CSPs handle all SOC 2 requirements without considering shared responsibility can lead to gaps in compliance.
- **Static controls:** Implementing controls and procedures as a one-time effort and failing to adapt to changing threats and risks can result in outdated security practices.

Conclusion

In this chapter, we discussed a lot of patterns and anti-patterns on topics ranging from authentication and authorization to privacy and compliance.

You should now be better equipped to avoid, recognize, and handle these security concerns.

We also used GDPR and SOC 2 as examples of how to go through and use patterns and avoid anti-patterns when it comes to specific compliance regulations.

You should now be able to at least somewhat transfer those skills into managing other compliance processes.

Questions

1. What happens first, authentication or authorization? Why?
2. What is MFA and what requirements does it make of users?
3. What is SSO, and how does it simplify user management?
4. What is OAuth 2.0, and when should you use it?
5. What is the principle of least privilege, and how does failing to apply it lead to security vulnerabilities?
6. Why is data classification important?
7. What is data masking? When should it be used?
8. What is data minimization, and how can you fail to apply it?
9. What is RBAC, and how does it support compliance?
10. What is GDPR?

11. What is SOC 2?
12. Can you find a compliance framework you and your organization need to conform to and apply a similar pattern/anti-pattern exercise?

References

- Attribute-based access control:
  https://en.wikipedia.org/wiki/Attribute-based_access_control
- eXtensible Access Control Markup Language:
  https://en.wikipedia.org/wiki/XACML
- GDPR Info:
  https://gdpr-info.eu/
- SOC 2:
  https://us.aicpa.org/interestareas/frc/assuranceadvisoryservices/aicpaSOC2report
- Azure compliance:
  https://learn.microsoft.com/en-us/azure/compliance/
- AWS compliance:
  https://aws.amazon.com/compliance/
- GCP compliance:
  https://cloud.google.com/security/compliance
- Azure policy effects:
  https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effects

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 9

Securing APIs, Observability, and Incident Response

Introduction

In this chapter, we will be discussing three important topics: APIs, Observability, and Incident Response.

Securing APIs is essential for safeguarding sensitive data, preventing unauthorized access, ensuring business continuity, meeting regulatory compliance, and mitigating security threats. A comprehensive API security strategy is vital for the overall security and reliability of modern software ecosystems.

Cloud observability is essential for maintaining a reliable, performant, and secure cloud infrastructure. It empowers organizations to respond quickly to issues, optimize resource usage, and meet compliance standards, ultimately contributing to the overall success of cloud-based applications and services.

Incident response planning is not just about reacting to security incidents; it is a proactive strategy for minimizing the impact of incidents, protecting critical assets, and demonstrating a commitment to security and compliance.

Structure

The chapter covers the following topics:

- Securing APIs
- Observability
- Incident response

Objectives

By the end of this chapter, you should be able to understand why protecting APIs is important, how to start securing your APIs, and how to continuously improve your security posture.

You should also understand the importance of observability (sometimes shortened to o11y), how to implement observability correctly across your portfolio of services, and how to continuously improve your observability.

Finally, you should understand why incident response cannot be an improvised activity and why it needs to be deliberate, planned, and effective. You should also understand how to set up an incident response and how it should work for you and your organization.

Securing APIs

First, let us define an **Application Programming Interface** (**API**). An API is a way for two or more services to communicate with each other. At its core, this is what an API is. It sounds simple, and it is a simple concept.

However, they are complex and not uniform. APIs have a lot of moving parts, and in a production environment:

- There may be many of them.
- They may be secured in different ways and with different services.
- They may be created and maintained by different teams with different incentives.
- They may be stale and old or exciting and new.
- They may be public or private.
- They may be first-party or third-party.
- They may use different technologies under the hood.
- They may have been tested well or not at all.
- They may have defined structures, or they may return different structures based on different calls.
- They may be centralized or not.
-

APIs are conceptually very simple—communication between services—but in production, they may and usually are very complex unless your organization is extremely disciplined about building, deploying, and monitoring APIs.

APIs are used to move data around, so if data is the new oil, then APIs are the new pipelines. APIs are everywhere; they are ubiquitous, so you and your organization nowadays cannot get away from APIs, even though sometime in the distant past, this was possible.

Securing APIs is in some ways similar to the security of any other infrastructure in the cloud, but in some ways, it is very specific with APIs in mind.

Securing APIs is crucial for several reasons. They serve as the gateway for data exchange between different software systems.

APIs handle sensitive information, including user data, authentication tokens, and proprietary business data. Securing APIs is essential to protect this sensitive information from unauthorized access and ensure its confidentiality and integrity.

Unauthorized access to APIs can lead to data breaches, data manipulation, and other malicious activities. Securing APIs with strong authentication and authorization mechanisms helps prevent unauthorized users from accessing sensitive resources, reducing the risk of data breaches and unauthorized actions.

Many applications and services rely on APIs to function properly. If an API is compromised or experiences a security breach, it can have a cascading effect on the connected systems, potentially leading to service disruptions, downtime, and financial losses. Securing APIs is essential for maintaining business continuity and the overall reliability of interconnected systems.

Various industries have specific compliance and regulatory requirements regarding the protection of user data and privacy. Securing APIs is often a critical component of meeting these requirements. Failure to secure APIs adequately may result in legal consequences, fines, and damage to the organization's reputation.

APIs are susceptible to various security threats, including injection attacks, **cross-site scripting** (**XSS**), and **denial-of-service** (**DoS**) attacks. By implementing security best practices, such as input validation, encryption, and proper error handling, organizations can significantly reduce the risk of these threats and create a more robust defense against potential security vulnerabilities.

Authentication

We implement strong authentication mechanisms, such as OAuth 2.0 or API keys, to ensure that only authorized users can access your API.

Authentication is the process by which a system verifies the identity of a user or entity attempting to access its resources or services. It serves as the first line of defense against

entry. Authentication methods range from traditional passwords and PINs to more advanced biometric scans and cryptographic keys. As digital ecosystems expand and cyber threats evolve, the importance of robust authentication mechanisms cannot be overstated. Effective authentication not only safeguards sensitive data and resources but also instills trust and confidence among users, fostering a secure and seamless digital experience.

Authorization

Enforcing proper access controls to ensure that authenticated users have the proper permissions to access specific API resources is as important as ever. Nowadays this is done through the use of API keys, which may be exposed on a self-service portal so they can be independently rotated by users themselves. There may even be an API exposed that allows for key rotation.

In Azure, for example, these capabilities can be provided by the API Management service.

Input validation

Validating and sanitizing all input data to prevent common security issues such as injection attacks (for example, SQL injection, XSS) is crucial. APIs must only accept valid and sanitized input. For example, if the required input is the date of birth, only the number and characters required to represent the data are allowed, and the maximum length is defined such that all dates are accepted, but anything longer than that is rejected.

Parameterized queries

Using parameterized queries when interacting with databases to prevent SQL injection attacks goes hand in hand with the previous section. This is an important coding practice, so your developers should follow the best practices and guidelines based on the language of their choice.

This is, by far, still one of the most common ways of breaching API security.

Secure communication

Use HTTPS/TLS to encrypt data in transit and protect it from eavesdropping and man-in-the-middle attacks.

In the vast digital landscape of the modern world, communication over networks has become an integral part of daily life. Whether you are browsing the web, accessing online services, or transmitting sensitive information, the need for secure communication has never been more critical. One of the fundamental technologies ensuring this security is **Hyper Text Transfer Protocol Secure** (**HTTPS**).

As discussed in previous chapters, this is hopefully self-evident to you by now.

Rate limiting

Implement rate limiting to prevent abuse, unauthorized access, and DoS attacks.

This is a simple and effective way to prevent both accidental abuse and deliberate attacks. Rate limiting means defining and only allowing a specific number of attempts or calls to the specific API in a given time period.

This is very important for all security-related APIs, such as signing in, changing passwords, and others. It is also important to prevent or detect data exfiltration.

In Azure, for example, these capabilities, too, can be provided by the API Management service.

Audit logs

Keep detailed logs of API activities, including authentication and authorization events, to monitor and respond to security incidents.

As with all advice, we have learned in this book, keeping an audit of system events is a must, and APIs are no exception to this.

Token management

If using tokens (for example, JWT), ensure they are securely generated, transmitted, and validated to prevent token-based attacks.

This topic can be quite complex if the APIs have not been standardized on a library to handle tokens, and even then, there are different implementations to consider, so again – refer your engineers to best practices for their chosen technologies.

Error handling

Provide customized error messages to users but avoid exposing sensitive information that could be exploited by attackers.

It is really important from a business perspective to be helpful and provide correct and useful error messages, but – as always – it is also important not to leak security information.

This tip might not be immediately obvious, so here is an example (or two).

Let us say an application encounters an unexpected error, and the error message includes a detailed stack trace. The stack trace might reveal the internal structure of the application, including file paths, class names, and potentially sensitive information. If this detailed error information is returned to the client in a production environment, it could aid attackers in identifying vulnerabilities and crafting targeted attacks.

```
"error": {
  "message": "Internal Server Error",
  "code": 500,
  "stackTrace": "at com.example.controller.UserController.
getUser(UserController.java:42)\n..."
}
```

In the above example, the **stackTrace** field provides detailed information about the **UserController** class and the specific method that caused the error. An attacker could use this information to understand the application's internal structure and potentially exploit vulnerabilities in the code.

As a real-life example of such vulnerability, let us consider the Heartbleed vulnerability.

While not specifically an API error message, the Heartbleed vulnerability serves as a real-life example of information leakage. Heartbleed was a security bug in the OpenSSL cryptography library, and it allowed attackers to read sensitive data from the memory of millions of web servers. The vulnerability leaked information, including private keys, passwords, and session cookies.

In the context of APIs, similar vulnerabilities could arise if error messages reveal too much information about the underlying infrastructure or if they expose sensitive data. It underscores the importance of crafting informative yet generic error messages in a production environment to prevent unintentional disclosure of sensitive information.

A simple rule of thumb is to not expose any information in an error message that you did not type yourself. If generic data is required (for example, an exact API request field that is missing or malformed), this data should also be validated and sanitized.

Cross-Origin Resource Sharing

Configure **Cross-Origin Resource Sharing** (**CORS**) headers properly to control which domains can access your API and mitigate cross-origin attacks.

CORS is a critical mechanism in web security that governs how web browsers permit web applications running in one domain to request resources from another domain. CORS provides a way for web servers to specify which origins are allowed to access their resources, thus mitigating the risks associated with cross-origin requests, such as XSS attacks. By defining a set of HTTP headers that allow or deny cross-origin requests based on specified criteria, CORS helps ensure the integrity and security of web applications while enabling seamless communication between different origins.

This simple definition may help prevent access from unauthorized sources.

Content Security Policy

Implement **Content Security Policy** (**CSP**) headers to mitigate the risk of XSS attacks by controlling which sources are allowed to load content.

CSP is a powerful security standard that web developers utilize to mitigate the risks of XSS, data injection attacks, and other malicious activities on their websites. CSP allows website administrators to define a set of rules and directives that instruct the browser on which resources can be loaded and executed on a web page. By specifying trusted sources for scripts, stylesheets, images, and other content, CSP helps prevent unauthorized code execution and protects against various types of attacks aimed at exploiting vulnerabilities in web applications. This proactive approach to web security enhances user privacy, reduces the likelihood of data breaches, and fortifies the overall integrity of online platforms.

Similar to the previous section, this may help mitigate a whole range of attacks.

API versioning

Include version information in your API URLs to manage changes without breaking existing clients and to help prevent security issues due to outdated API versions.

By delivering a new version of an API and deprecating previous ones, you prevent legacy versions from lingering for too long. By definition, being legacy versions, these versions will unlikely receive the same treatment for all the new security vulnerabilities.

Limiting the scope and surface area of your APIs will make it easier to secure your current API versions.

Security headers

Utilize security headers such as **HTTP Strict-Transport-Security** (**HSTS**) and X-Content-Type-Options to enhance the overall security posture.

Consider the best practices depending on the technologies used in your organization.

Data encryption

Encrypt sensitive data at rest using strong encryption algorithms to protect it from unauthorized access.

Remember how, in previous chapters, we mentioned that you should encrypt everything at rest and in transit? APIs are no different. With especially sensitive data, such as health data, you may want to consider encrypting that data with keys specific to the requester so that they and only they may decrypt the data, so in case of a successful attacker that somehow got a hold of API keys of your customers they would still be unable to read the encrypted health information stored within as this information is encrypted/decrypted

with keys that have been exchanged once and are therefore less likely to be intercepted within the parameters of the usual request/response communication.

Dependency scanning

Regularly scan and update dependencies, libraries, and frameworks to address vulnerabilities and ensure a secure development environment.

Dependency scanning is a crucial aspect of modern software development that involves analyzing an application's dependencies, libraries, and components for known vulnerabilities and security issues. By automatically scanning the codebase and its dependencies, developers can identify potential risks and vulnerabilities early in the development lifecycle, allowing for timely mitigation and remediation.

Dependency scanning tools leverage databases of known vulnerabilities to detect insecure versions of libraries and components, providing actionable insights to developers to address security concerns proactively. This proactive approach to security helps organizations maintain the integrity and safety of their software applications, reducing the likelihood of security breaches and ensuring the delivery of secure and robust software products to end-users.

Security testing

Conduct regular security assessments, including penetration testing and code reviews, to identify and address potential vulnerabilities.

Regularly means as often as it makes sense.

If you use third parties for this, usually, there is a yearly test planned or quarterly – due to the costs involved. This is not sufficient, as technology and technological vulnerability capabilities are expanded more often than yearly.

The best approach is to have your own purple team that deals with this continuously. Adding occasional third-party testing is then just a bonus.

Denial-of-Service protection

Implement mechanisms to protect against DoS attacks, such as rate limiting, traffic shaping, and cloud-based DDoS protection services.

All cloud providers provide various levels of DDoS mitigation, so work with them to find the best and most cost-effective solution for your APIs.

API gateway security

If using an API gateway, ensure it is configured securely and serves as an added layer of protection for your API.

With the increasing complexity and volume of API interactions, robust security measures are essential to protect against unauthorized access, data breaches, and other malicious activities. API gateways enforce security policies, such as rate limiting, authentication protocols, and encryption, to ensure that only authenticated and authorized users or applications can access the APIs while safeguarding sensitive data and resources.

By providing a unified interface for managing security across diverse API endpoints, API gateways play a crucial role in maintaining the confidentiality, integrity, and availability of digital assets in distributed architectures.

Secure file uploads

If your API allows file uploads, validate file types, limit file sizes, and consider storing files in a secure location with proper access controls.

This is similar to input validation but for files. Do not forget to do this.

Education and training

Provide ongoing security training for developers and other stakeholders involved in API development to ensure they are aware of security best practices and can identify potential risks.

Treat management should be a part of your continuous process, so education on threat management should be considered as well.

Purple teaming APIs

Purple teams come from merging the red and blue teams (attackers are red and defenders are blue, so purple is a mixture of both, or swapping team members between these two teams or tasks).

APIs are tested in the following different ways:

- **Black box:** Simulating an attacker with very limited knowledge.
- **Grey box:** Simulating an attacker with limited knowledge but with some information on the target.
- **White box:** Testing with full knowledge, up to and including source code and all the target information.

In order to defend your APIs, you and your organization need to consider:

- The full inventory of your APIs
- Who, when, and how often is calling your APIs?
- Data received and sent.
- Where did the call originate from, and how trusted is the originator?
- The normal behavior of your APIs.
- How do you define an anomalous behavior, and how do you detect and alert on it?
- The current state of security of your APIs.
- How often do you test the APIs, and how comprehensively?
- How do you feed the testing information back to the development and operation teams, and how do they act on the information provided in order to improve?

APIs consist of:

- Hostnames (for example, example.com),
- API paths (for example, /api/v2023-11-23/user/add),
- Methods (for example, POST, GET, PATCH, DELETE, etc.).

Since APIs are usually very well defined, public, and even have tutorials and documentation published as you and your organization usually want them used, it is exceedingly difficult to figure out what is valid use, accidental abuse, and deliberate attack. This makes APIs difficult to defend, as attackers can have perfect information on the API definitions, behaviors, and security.

Observability

How will you know if your APIs are secure, responsive, performant, and helping users and machines alike achieve their goals? You need proper observability.

This ties perfectly into our plan for this chapter, so let us move from the topic of API security to observability.

Observability

Cloud observability is crucial as it provides insights into the performance, reliability, and security of applications and systems in a cloud environment.

Cloud observability allows organizations to detect and diagnose issues in real-time. By

problems, leading to faster resolution times. This is crucial for maintaining a high level of service availability and minimizing downtime.

Observability helps in understanding how different components of a system interact and perform. By analyzing metrics and logs, organizations can identify bottlenecks, optimize resource utilization, and ensure that applications run efficiently. This leads to better performance, improved user experience, and cost optimization through efficient resource allocation.

Cloud observability supports the dynamic nature of cloud environments (especially in the context of microservices and containerized applications). By monitoring workloads and resource usage, teams can implement auto-scaling and dynamic load-balancing strategies, ensuring that the infrastructure scales seamlessly to handle varying workloads. This contributes to increased agility and the ability to respond quickly to changing business demands.

Observability is crucial for monitoring and responding to security threats. Organizations can detect abnormal patterns or suspicious activities by collecting and analyzing logs and traces, allowing for early identification of security breaches. This proactive approach enhances the overall security posture of cloud-based systems.

Many industries and organizations are subject to regulatory compliance standards. Cloud observability helps in meeting these requirements by providing the necessary monitoring and logging capabilities. This ensures that organizations can demonstrate adherence to compliance standards, which is particularly important in industries such as healthcare, finance, and government.

One real-life example of observability not done correctly in a cloud environment is the AWS outage that occurred on March 2, 2017. This incident was a result of an error made during routine maintenance of the Amazon **Simple Storage Service** (**S3**).

"The four-hour AWS outage caused S&P 500 companies to lose $150 million, Cyence, a startup that models the economic impact of cyber risk, a Cyence spokeswoman said via email. US financial services companies lost $160 million, the firm estimated." **https://www.datacenterknowledge.com/archives/2017/03/02/aws-outage-that-broke-the-internet-caused-by-mistyped-command**

During the maintenance activity, an Amazon employee executed a command to remove a small number of servers from the S3 subsystem that was used by the AWS S3 billing process. The intention was to take a few servers offline, but the employee inadvertently entered a larger number of servers than intended.

The root cause of this incident highlighted a lack of proper observability of the dependencies and potential impact of the S3 subsystem. Amazon S3 is a critical component of many cloud-based applications, and the error had a cascading effect, impacting a larger number of servers than anticipated.

There was insufficient visibility into the dependencies between different subsystems and services within the AWS infrastructure. The impact of the maintenance command on other services was poorly understood, leading to unintended consequences.

The incident revealed a gap in real-time monitoring capabilities. The impact of the error was not immediately apparent, and it took time for Amazon to identify and acknowledge the problem. During this period, customers experienced disruptions in various AWS services.

The lack of proper observability resulted in widespread service disruptions for several hours, affecting a significant number of high-profile websites and services that relied on AWS infrastructure. The incident highlighted the importance of comprehensive observability in understanding the interdependencies of cloud services and the need for real-time monitoring to detect and respond to issues promptly.

Following this incident, AWS took steps to improve its observability practices. This included enhancing real-time monitoring, improving visibility into dependencies, and implementing additional safeguards to prevent similar issues in the future. The incident underscores the critical role of observability in maintaining the reliability and resilience of cloud environments.

In order to achieve proper observability, this section discusses some of the most important patterns to consider.

Defining clear objectives

Clearly define your observability goals. Know what you want to monitor and why.

This is a complex requirement and not as easy as you might think. Observability means different things to different people, roles, domains, and departments, and it changes over time.

There is a simple tip to help you in the cloud, and nothing less will do – your observability systems must be able to answer any question at any time about your system without the need to change the system to be able to do so. If you can confidently confirm this for your entire estate, you are on a good path in defining your objectives for each component.

Instrumentation

Implement effective instrumentation to collect relevant data from your applications and infrastructure.

Cloud vendors will have various services to help you do this for infrastructure and even for your own code in your own services, so avail yourself of them whenever possible. There will also be APIs for you to call and for you to send custom telemetry.

Do not forget to include the business logic metrics as well, which cloud providers and third-party logging services will not understand—but you and your dashboards will.

Centralized logging

Aggregate logs from all your services into a centralized logging system for easy analysis and troubleshooting.

Remember that tip from a few sections ago? You need to be able to query your system for information on any data point you need, and you need to be able to do this across the board. If you can do this from a centralized location – it helps a lot.

Metrics collection

Collect and analyze key performance metrics such as response times, error rates, and resource utilization.

Do not forget to include the business logic metrics as well, which cloud providers and third-party logging services will not understand—but you and your dashboards will.

Distributed tracing

Use distributed tracing to visualize and understand the flow of requests across your microservices architecture.

Distributed tracing is a powerful technique used in software engineering to monitor and troubleshoot complex, distributed systems. As modern applications increasingly rely on microservices architectures and cloud-native environments, understanding the flow of requests across various services becomes challenging.

Distributed tracing addresses this challenge by instrumenting applications to generate trace data, which provides insights into the path and performance of individual requests as they traverse multiple services. By correlating and visualizing these traces, developers can identify bottlenecks, latency issues, and errors within the system, facilitating effective performance optimization and debugging.

Distributed tracing enables teams to gain comprehensive visibility into the behavior of distributed systems, empowering them to deliver reliable and scalable applications with confidence.

Real-time monitoring

Implement real-time monitoring to detect issues as they occur and respond promptly.

Automation is key in the cloud, and real-time monitoring should be included in your automation scripts. You need to automate any reoccurring events that need systemic and

typical responses in order to allow your teams to focus on exceptional events when they occur and not be overwhelmed by the day-to-day operation of repetitive tasks.

Alerting mechanisms

Set up proactive alerting to notify you of potential issues before they impact the user experience. Here, moderation is key. You only want to send alerts on events that need human interaction otherwise you are creating noise that will result in missing a critical alert in a sea of alerts. A rule of thumb is – if it does not require an immediate response from a human – do not send an alert. Automate a response instead or dashboard a cumulative number of these alerts and only alert once the number of these events is so high that it requires a response from humans.

The usual cases where this is abused or incorrectly set are alerts from events that are known to sometimes require retries. For example, a backup attempt has failed, but a retry has succeeded. There is no need to be alarmed. If it continuously fails and it requires a response from a human, then alert.

Auto-scaling and load balancing

Leverage auto-scaling and load balancing to ensure optimal performance during varying workloads. Even if these events are rare, automate them.

Auto-scaling is a fundamental concept in cloud computing that enables applications to dynamically adjust their computing resources based on demand. With auto-scaling, cloud services can automatically add or remove resources such as virtual machines, containers, or serverless functions to match changing workloads.

By monitoring metrics such as CPU utilization, memory usage, or request rates, auto-scaling ensures that applications maintain optimal performance and availability while minimizing costs by scaling resources up during periods of high demand and down during periods of low demand. This flexibility allows organizations to efficiently manage their infrastructure resources and deliver a seamless experience to users regardless of fluctuations in traffic or workload.

Load balancing is a component of distributed computing architectures that distributes incoming network traffic across multiple servers or resources to ensure optimal utilization and performance. By evenly distributing requests, load balancers help prevent individual servers from becoming overwhelmed, thus improving the reliability, responsiveness, and scalability of web applications. Load balancers can operate at various layers of the network stack, including the application layer (HTTP/HTTPS), transport layer (TCP/UDP), or even the network layer (IP). Modern load balancers often provide advanced features such as health checks, session persistence, and SSL termination, further enhancing the resilience and efficiency of distributed systems.

Security monitoring

Integrate security monitoring tools to identify and respond to potential security threats. Observability means observability in your entire system. This includes your tools and processes; observability should be built into those as well.

Dependency mapping

Create dependency maps to understand how different components of your system interact. Keep them up to date and automate checks to verify these assumptions. Alert if these assumptions have changed.

For example, if a service never needs public internet access, alert once this occurs. If a service pings another every hour at least, alert when this does not happen. Or, if a service like CPU/Memory never needs scaling, alert when it does. Or, if the last 365 deployments occurred over the weekdays and never during a weekend, alert on a weekend deployment.

Cloud-native tools

Use cloud-native monitoring tools provided by your cloud service provider for better integration. There are third-party tools out there, but your best bet is on the native services of your cloud provider. This is cost effective and minimizes latency between events occurring and observability showing you this.

Cost monitoring

Monitor resource usage and costs to optimize your cloud spending.

Budgets (as in budgets specified in the consoles/portals of your cloud provider) are your friend. Changes to these will alert you to where you and your organization need to pay attention, as either there are unexpected changes or there is heavy development – in both cases, your development and operations teams should be paying attention.

Collaboration and communication

Foster collaboration between development, operations, and security teams for effective issue resolution. Effective collaboration between development, operations, and security teams is paramount for timely and efficient issue resolution in today's complex software landscape. By fostering a culture of collaboration and communication, organizations can leverage the unique expertise and perspectives of each team to address issues comprehensively and proactively.

- Development teams bring deep knowledge of the application codebase and architecture, allowing them to pinpoint root causes and implement fixes efficiently.

- Operations teams contribute insights into infrastructure and deployment processes, enabling them to identify system-level issues and optimize performance.
- Security teams offer expertise in identifying and mitigating security vulnerabilities and ensuring compliance with regulatory standards. By breaking down silos and promoting cross-functional collaboration, organizations can streamline issue resolution processes, accelerate time-to-resolution, and ultimately deliver more reliable and secure software to end-users.

Continuous improvement

Review and update your observability strategy regularly based on changing requirements and technological advancements. Continuous improvement is key. Establish a feedback loop to continuously improve observability based on insights gained from monitoring and incident responses.

Documentation

Document your observability processes and configurations to facilitate knowledge sharing and troubleshooting, as well as enable new joiners to be as effective as your existing team members.

User experience monitoring

Monitor the end-user experience to understand and address performance issues from the user's perspective.

User experience monitoring (**UEM**) is a critical practice in modern software development aimed at understanding and optimizing the interactions between users and digital products or services. By continuously collecting and analyzing data on user behavior, performance metrics, and application usage patterns, UEM enables organizations to gain insights into the quality of the user experience and identify areas for improvement. Through techniques such as **real-user monitoring** (**RUM**), synthetic monitoring, and session replay, UEM provides visibility into how users interact with web and mobile applications, helping teams diagnose issues, prioritize enhancements, and drive iterative improvements to enhance user satisfaction, engagement, and retention. With user experience becoming a key differentiator in today's competitive landscape, UEM empowers organizations to deliver superior digital experiences that meet the evolving needs and expectations of their users.

Compliance monitoring

Implement monitoring solutions that help you meet regulatory compliance requirements.

Compliance monitoring is a fundamental practice in modern governance and regulatory frameworks, ensuring that organizations adhere to relevant laws, regulations, and industry standards. By systematically assessing and verifying adherence to compliance requirements, organizations can mitigate legal, financial, and reputational risks while maintaining trust with stakeholders.

Compliance monitoring involves ongoing audits, assessments, and reviews of policies, procedures, and controls to ensure they align with applicable regulations and standards. Through continuous monitoring and reporting mechanisms, organizations can promptly identify gaps, deviations, or non-compliance issues, allowing for timely corrective actions and improvements. With the regulatory landscape constantly evolving, compliance monitoring serves as a proactive approach to managing compliance risks and promoting a culture of accountability and integrity within organizations.

Education and training

Ensure your team has the necessary skills and training to make the most of your observability tools.

Ensuring that your team possesses the requisite skills and training to leverage observability tools effectively is paramount in today's complex digital landscape. Observability tools provide invaluable insights into the behavior and performance of distributed systems, enabling teams to identify and resolve issues swiftly, optimize system performance, and deliver exceptional user experiences. However, the efficacy of these tools hinges on the proficiency of the individuals tasked with utilizing them. Therefore, investing in comprehensive training programs and skill development initiatives is essential to maximize the value derived from observability solutions.

First and foremost, providing your team with in-depth training on the functionalities and capabilities of observability tools equips them with the knowledge and proficiency needed to navigate and harness these tools effectively. Training sessions can cover various aspects, including data collection mechanisms, query languages, visualization techniques, and troubleshooting methodologies. Additionally, hands-on workshops and practical exercises allow team members to gain practical experience in using observability tools in real-world scenarios, honing their problem-solving skills and familiarity with the tool's features. By empowering your team with comprehensive training, you enable them to harness the full potential of observability tools, enabling proactive monitoring, rapid issue resolution, and continuous optimization of your systems and applications.

Incident response plan

Develop a comprehensive incident response plan to address issues promptly when they occur.

This ties perfectly into our plan for this chapter, so let us move from the topic of observability to incident response.

Incident response

An effective incident response plan helps in quickly identifying and containing security incidents. This swift response minimizes the potential damage caused by the incident and reduces the overall downtime of critical systems and services.

Incidents often involve the compromise of sensitive information. A well-prepared incident response plan ensures a rapid and organized response to data breaches, protecting sensitive data from unauthorized access or disclosure.

Public perception is a significant factor in the aftermath of a security incident. A prompt and transparent incident response demonstrates a commitment to security and can help preserve the organization's reputation. Maintaining customer trust is vital for long-term success.

Many industries are subject to regulations and compliance standards that mandate a robust incident response capability. Failure to comply with these regulations can result in legal consequences and financial penalties. An incident response plan helps organizations meet these regulatory requirements.

Incident response planning is not a one-time effort; it is an ongoing process. Regular testing, reviews, and post-incident analyses contribute to continuous improvement. Organizations can learn from each incident, refine their processes, and enhance their overall security posture over time.

In order to create an incident response plan, consider the features discussed next in this section.

Understanding your assets

Identify and classify critical assets and data so you can plan for different modalities of your incident response. Finding your customers' data on the dark web and a compromised account of one of your staff might require different responses.

Still, a general response plan is better than a specific one. One that covers any scenario or as many scenarios as possible is better than a hundred specific plans that can only address a very niche requirement.

Risk assessment

Perform a thorough risk assessment to prioritize potential threats, as you and your organizational resources are finite. This also allows for focus to be applied to prioritized

Create an Incident Response Team

Establish a dedicated **Incident Response Team** (**IRT**) with defined roles and responsibilities. Isolate your team from the CEO, customers, and other distractions while working on an issue. Communication is key.

Communication plan

Develop a clear communication plan for internal and external stakeholders. Only one person can make the final decision on communications going out. Only one person can know the status of the work; there can be no interruptions from addressing the incident to reporting the same information over and over again to the CEO, marketing, social media team, individual customers, and so on. Some folks work on an issue, and some report and write communication documents. You can switch who does what on different incidents.

Training and drills

Regular training of your IRT is essential to ensure they remain prepared and effective in responding to incidents promptly and efficiently. Conducting simulated exercises on a regular basis provides an invaluable opportunity for team members to practice their incident response procedures in a controlled environment, helping them refine their skills and identify areas for improvement.

These exercises can simulate various types of incidents, ranging from minor disruptions to critical system failures, allowing the team to test their communication protocols, decision-making processes, and technical proficiency in a realistic setting. By exposing team members to different scenarios and challenges through regular simulations, you not only enhance their individual capabilities but also foster a cohesive and coordinated response strategy within the team.

The frequency of training and simulated exercises should align with the dynamic nature of cybersecurity threats and the evolving technology landscape. Incidents can occur at any time, and the effectiveness of your response depends on the readiness of your team.

Therefore, "regularly" implies a continuous commitment to training and readiness rather than a sporadic or infrequent occurrence. Implementing a structured training program that includes recurring exercises, workshops, and knowledge-sharing sessions ensures that the IRT remains agile, adaptive, and well-prepared to handle emerging threats and challenges. By ingraining a culture of continuous learning and improvement within the team, you can strengthen your organization's resilience to cybersecurity threats and minimize the impact of potential incidents on your operations and reputation.

Incident identification

Implement tools and processes for real-time incident detection. Remember observability? This is where that insight needs to come in.

Your incident identification and response can only be as good as your observability. You cannot manage that which you cannot identify and see and monitor and track and act upon.

Incident categorization

Classify incidents based on severity and impact. Report each to stakeholders, likely different ones based on both severity and impact. Prepare a summary for the C-levels and for the customers.

Escalation procedures

Define clear escalation paths for different types of incidents and have deputies for all people involved – as life does not stop because you and your organization have an incident. People leave, have accidents - or worse. Plan for it.

Documentation

Maintaining comprehensive documentation of incident response procedures is essential for ensuring a swift and coordinated response to security incidents. This documentation should encompass detailed protocols, escalation paths, contact information, and step-by-step instructions tailored to different types of incidents.

Regularly revisiting and updating this documentation is crucial to reflect changes in technology, processes, and organizational structure. Sharing this documentation widely within the organization, including with stakeholders and customers, fosters transparency and alignment on incident response protocols. Soliciting feedback from stakeholders, especially customers, provides valuable insights into their expectations and concerns, enabling organizations to refine and improve their incident response procedures iteratively. By prioritizing the documentation and dissemination of incident response procedures, organizations can enhance their preparedness, accountability, and trustworthiness in the face of cybersecurity threats.

Legal and compliance considerations

Understand and comply with legal and regulatory requirements.

Understanding and complying with legal and regulatory requirements is paramount for ensuring the security and integrity of an organization's operations. In today's highly

is not only a legal obligation but also a critical aspect of maintaining trust with customers, partners, and stakeholders.

Legal and regulatory requirements vary depending on the industry, geographic location, and nature of your business, encompassing areas such as data privacy, cybersecurity, consumer protection, and financial regulations. Therefore, it is very important for organizations to conduct thorough assessments to identify applicable requirements and establish robust compliance programs tailored to their specific needs.

By staying on top of changes in legislation and proactively addressing compliance obligations, organizations can mitigate legal risks, avoid costly penalties, and uphold their reputation as responsible corporate citizens. Moreover, a strong commitment to compliance demonstrates organizational integrity and accountability, fostering trust and confidence among customers and stakeholders.

Incident response playbooks

Develop detailed playbooks for common incident scenarios - up to a point. If the same incidents are reoccurring, you might be legally liable, or you just need more automation or product fixes. Carefully evaluate playbooks with these in mind.

Incident reporting

Establish a standardized process for reporting incidents. Centralization is key here. Everyone must know where to go to trigger an incident. Yes, you can automate some of these as well.

Automating incident reporting is a strategic approach that streamlines the process of documenting and disseminating critical information about security incidents within an organization. By leveraging automation tools and workflows, organizations can capture incident details in real-time, ensuring accuracy and consistency while minimizing manual intervention.

Automated incident reporting systems can be configured to collect relevant data such as incident type, severity, affected systems or assets, and timestamps automatically from various sources, including monitoring tools, security alerts, and user reports. Additionally, these systems can facilitate the creation of standardized incident reports, which can be customized based on organizational requirements and regulatory obligations.

By automating incident reporting, your organization can expedite the response process, enable faster decision-making, and improve overall incident management efficiency in order to minimize the impact of security incidents on business operations and brand reputation.

Evidence handling

Define procedures for collecting, preserving, and analyzing digital evidence. This is also a legal requirement, so tread really carefully and solicit support from your friendly local law enforcement agency as soon as your incident response identifies the need. Do not delay.

If you remember one thing – do not turn off/delete/disable/decommission/archive your infrastructure. Prevent what is happening but preserve evidence.

In the cloud, also contact your provider and explain the need to preserve evidence and have support assist you. In the cloud, you are on shared infrastructure and unreliable (by design) infrastructure, so assistance from support may be required for you to preserve evidence.

Forensic readiness

Ensure systems are forensically sound to facilitate investigations. You will need to get help to address this requirement from an authorized party, depending on your jurisdiction.

- **Implement robust logging mechanisms:** Ensure that systems generate comprehensive logs capturing relevant activities, including user authentication, access attempts, system changes, and network traffic. These logs should contain sufficient detail to reconstruct events accurately and facilitate forensic analysis during investigations.
- **Enable file integrity monitoring:** Implement **file integrity monitoring** (**FIM**) solutions to detect unauthorized changes to critical system files, configurations, and directories. FIM tools can alert security teams to suspicious modifications, providing valuable insights into potential security breaches and unauthorized activities.
- **Preserve evidence integrity:** Establish procedures for preserving the integrity of digital evidence to maintain its admissibility and reliability in legal proceedings. This includes securing and protecting evidence from tampering, ensuring proper chain of custody, and adhering to forensic best practices throughout the investigation process.
- **Conduct regular audits and assessments:** Regularly audit and assess system configurations, security controls, and access permissions to identify vulnerabilities and ensure compliance with security policies and regulations. These audits help maintain the integrity of systems and preemptively address potential security gaps that could compromise forensic investigations.
- **Train personnel in forensic procedures:** Provide training and resources to security personnel and incident responders on forensic investigation techniques, tools, and methodologies. Equipping personnel with the necessary skills and knowledge enables them to conduct thorough and effective

evidence and uncovering insights crucial for incident response and remediation efforts.

Collaboration with external entities

Establish contacts with external organizations for information sharing. Your peers are your greatest strength. Be careful with what and when you can share, though.

Continuous improvement

Regularly review and update your incident response plan:

- **Conduct comprehensive tabletop exercises:** Regularly schedule tabletop exercises to simulate various security incidents and test the effectiveness of your incident response plan. These exercises involve key stakeholders from across the organization and provide an opportunity to identify weaknesses, gaps, and areas for improvement in the plan. By simulating realistic scenarios, teams can validate response procedures, refine communication protocols, and enhance coordination to ensure readiness for actual incidents.
- **Incorporate lessons learned from past incidents:** Review and analyze the outcomes of previous security incidents to identify trends, patterns, and recurring issues. Use this information to update the incident response plan, incorporating lessons learned and best practices to strengthen future response efforts. By leveraging insights from past incidents, organizations can refine response procedures, implement preventive measures, and mitigate the risk of similar incidents occurring in the future.
- **Stay abreast of emerging threats and technologies:** Continuously monitor the threat landscape and technological advancements to ensure that your incident response plan remains relevant and effective. Regularly review industry reports, threat intelligence feeds, and security advisories to identify emerging threats, vulnerabilities, and attack vectors. Incorporate this knowledge into the incident response plan by updating detection mechanisms, response procedures, and mitigation strategies to address evolving threats effectively. By staying proactive and adaptive, organizations can enhance the resilience of their incident response capabilities and better protect against emerging security challenges.

Incident analysis

Conduct post-incident analysis to identify lessons learned. Publish these at least internally, preferably to your customers as well. This shows you are capable of acting when incidents occur and holds you to tasks to learn from your mistakes.

Incident containment

Develop strategies for isolating and containing incidents.

Developing strategies for isolating and containing incidents is essential for minimizing the impact of security breaches and preventing further spread within an organization's network. Implementing network segmentation and access controls can help isolate affected systems or compromised areas, limiting the lateral movement of attackers and containing the incident to a specific segment of the network.

Deploying endpoint detection and response solutions enables organizations to quickly identify and isolate compromised endpoints, preventing malicious actors from accessing sensitive data or spreading malware to other devices.

Utilizing threat-hunting techniques and behavioral analytics can aid in the early detection of anomalous activities, allowing security teams to proactively contain threats before they escalate. Implementing incident response playbooks and automated response mechanisms can streamline containment efforts by providing predefined steps and actions for isolating affected systems and mitigating the impact of security incidents. By developing comprehensive strategies for isolating and containing incidents, organizations can minimize the duration and scope of security breaches, reducing operational disruption and safeguarding critical assets and data.

Public relations strategy

Have a plan for managing public relations during and after incidents.

A well-crafted **public relations** (**PR**) strategy is essential when it comes to incident response, playing a crucial role in managing the public perception of an organization during and after a security incident.

Transparency is paramount. Organizations need to communicate the details of the incident, including the nature of the breach, the steps being taken to address it, and any potential impact promptly and honestly on customers or stakeholders. A transparent approach helps build trust by demonstrating accountability and a commitment to addressing the situation responsibly.

A proactive communication plan should be established. This involves not only addressing the incident itself but also keeping stakeholders informed throughout the entire incident response process. Regular updates should be provided detailing the progress made in resolving the issue, steps taken to prevent future occurrences, and any additional measures implemented to safeguard data and systems. By actively engaging with the public and stakeholders, organizations can shape the narrative surrounding the incident, showcase their dedication to resolving the issue, and reassure those affected that their concerns are being taken seriously.

Organizations should be prepared to handle media inquiries effectively. Designating a spokesperson and providing them with the necessary training to communicate with the media can ensure a consistent and controlled message. The organization's narrative should emphasize lessons learned, improvements made to prevent similar incidents in the future, and the ongoing commitment to security. A well-managed PR strategy not only helps mitigate potential reputational damage but also positions the organization as proactive, responsible, and dedicated to maintaining the trust of its customers and the public.

Vendor management

Establish relationships with external vendors for support during incidents.

Effective vendor management is a critical component of incident response, as many organizations rely on external vendors for various services and support. Establishing strong relationships with vendors and incorporating them into your incident response plan is essential for a comprehensive and coordinated approach.

It is crucial to conduct thorough due diligence when selecting vendors. Assess their security practices, incident response capabilities, and overall resilience to ensure they align with your organization's security standards. A careful evaluation during the vendor selection process can help prevent potential vulnerabilities and ensure that your partners are equipped to respond effectively in case of a security incident.

Once vendors are onboarded, it is essential to clearly define roles and responsibilities in the incident response plan. Clearly articulate the expectations from each vendor, including the reporting and communication protocols during an incident. This collaborative approach ensures that all parties are aligned and can work together seamlessly to address and mitigate the impact of an incident.

Regular communication and information sharing with vendors are also crucial elements of effective vendor management in incident response. Establishing open lines of communication facilitates the exchange of threat intelligence, ensuring that both parties are aware of potential risks and can collectively implement proactive measures to enhance security. This collaboration extends beyond incident response planning and contributes to a broader, mutually beneficial security posture.

In addition to collaboration, organizations should include contractual obligations related to incident response in their agreements with vendors. Clearly outline the expectations regarding notification timelines, reporting requirements, and other incident-related responsibilities. Having these contractual provisions in place provides a legal framework for holding vendors accountable and reinforces the importance of security cooperation.

Continuous monitoring and assessment of vendor security postures are essential. Regularly review and update your vendor management processes, conduct security audits, and ensure that your vendors adhere to the agreed-upon security standards. A proactive and ongoing approach to vendor management not only enhances incident response capabilities

Conclusion

In this chapter, we discussed some of the many facets of security around APIs, observability, and incident response.

Seamless integration of robust API security measures is crucial in fortifying an organization's digital ecosystem against potential threats. By establishing a comprehensive API security framework, organizations can mitigate the risk of unauthorized access, data breaches, and other malicious activities. This foundation not only safeguards sensitive information but also fosters a secure environment for the seamless exchange of data between interconnected systems.

The synergy between API Security and Observability is pivotal in maintaining a proactive stance towards potential security incidents. Observability provides real-time insights into the API infrastructure, enabling organizations to detect anomalies, monitor performance metrics, and identify potential vulnerabilities promptly. This symbiotic relationship enhances the overall resilience of the system, ensuring that any deviations from the norm are promptly addressed, thereby minimizing the risk of security incidents.

As organizations strive to uphold the highest standards of cybersecurity, integrating API security and observability becomes a linchpin in a well-rounded Incident Response Plan. In the event of a security incident, the insights gained through Observability serve as a critical component in the Incident Response Plan, aiding in the swift identification, containment, and resolution of the issue. The interconnected nature of API security, Observability, and Incident Response solidifies a holistic approach to cybersecurity, creating a dynamic and adaptive defense mechanism that evolves with the ever-changing threat landscape.

Now, you are ready to start planning this journey for your organization as we talk about virtual machines and containers.

Observability is the key lesson from this chapter, which you should continue understanding and exploring in depth as we venture into the world of virtual machines and containers and beyond in the chapters that follow.

Questions

1. What are APIs?
2. Why is it important to secure APIs?
3. Why is input validation important, and what are some of the best ways to achieve it?
4. How can error messages leak security information?
5. What is rate limiting?

6. Why is maintaining legacy API versions not desirable in the long term?
7. What is a purple team?
8. What is observability?
9. How do you know if you have good observability?
10. How can you go wrong with alerting?
11. What is dependency mapping, and how does it tie into observability?
12. What is an incident response plan?
13. Why is forensic readiness important?
14. What are the good parts of your organization's incident response, and what needs to be improved?

References

- Wikipedia API:
  https://en.wikipedia.org/wiki/API
- OWASP API Security Top 10 - 2023:
  https://owasp.org/API-Security/editions/2023/en/0x00-header/
- Azure API Management:
  https://azure.microsoft.com/en-us/products/api-management
- Heartbleed bug:
  https://heartbleed.com/
- AWS outage that broke the Internet:
  https://www.datacenterknowledge.com/archives/2017/03/02/aws-outage-that-broke-the-internet-caused-by-mistyped-command

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 10
Virtual Machines and Containers

Introduction

This chapter will focus primarily on two technologies for delivering compute in the cloud: virtual machines and containers. Of course, there are many others, and we will get to those in the next chapters.

We will first discuss virtualization, what it means, and why it matters. Then, we will discuss securing **virtual machines** (**VMs**) in the cloud, followed by securing containers in the cloud. When discussing containers, we cannot skip an important concept: orchestration.

Since VMs and/or containers are usually the first services used when dipping one's toes into the cloud, we will also discuss migration—migration from on-premises to the cloud—and what to consider in those circumstances.

Structure

The chapter covers the following topics:

- Similarities
- Differences
- Virtualization

- Virtual machines
- Containers
- Migration

Objectives

In this chapter, you will learn about securing VMs and containers. While securing other cloud infrastructure services is very similar to securing VMs and containers, there are some specifics that we will cover as well.

By the end of this chapter, you will have a general understanding of what you need to consider and keep in mind when planning your VM and container deployments.

If you or your organization are also considering migrating from on-premises to the cloud, you will understand what to plan for in that scenario.

Similarities

There are quite a few similarities between VMs and containers. Let us take a look at them:

- **Isolation:** Both VMs and containers provide a level of isolation for applications. They encapsulate applications and their dependencies to avoid conflicts.
- **Portability:** VMs and containers are portable, allowing applications to run consistently across different environments, such as development, testing, and production. Even other clouds.
- **Resource efficiency**: Both technologies aim to optimize resource utilization by allowing multiple applications to run on the same physical hardware without interference.
- **Snapshot and rollback:** Both VMs and containers support the creation of snapshots and rollback features, enabling easy backup and recovery of application states.
- **Networking:** VMs and containers can be configured to communicate with each other and external networks, allowing for the development of complex distributed systems.
- **Automation:** Virtualization technologies enable automated deployment, scaling, and management of applications, contributing to efficient DevOps practices.
- **Platform independence:** VMs and containers abstract away underlying hardware and infrastructure details, making applications independent of the specific host environment.

- **Isolation of dependencies:** Both technologies allow for the encapsulation of application dependencies, reducing the likelihood of conflicts and simplifying dependency management.
- **Resource isolation:** VMs and containers offer mechanisms to control and allocate resources such as CPU, memory, and storage to ensure fair sharing among multiple applications.
- **Lifecycle management:** VMs and containers provide tools and APIs for managing the entire lifecycle of applications, from creation and deployment to scaling and decommissioning.

Differences

There are also quite a few differences between VMs and containers:

- **Hypervisor vs. container engine:** VMs run on a hypervisor (for example, VMware, Hyper-V), while containers run on a container engine (for example, Docker with orchestration on Kubernetes).
- **Overhead:** VMs typically have more overhead as they include a full operating system, whereas containers share the host OS kernel, resulting in lower overhead.
- **Startup time:** Containers generally have faster startup times compared to VMs due to their lightweight nature.
- **Resource utilization:** Containers are more efficient in resource utilization as they share the host OS, whereas VMs have more significant resource overhead.
- **Isolation level:** VMs provide stronger isolation since they have separate OS instances, while containers share the host OS kernel, offering a lighter form of isolation.
- **Image size:** Container images are smaller compared to VM images, making them quicker to deploy and transfer.
- **Scaling:** Containers are often more suitable for microservices architectures and dynamic scaling, while VMs are better suited for monolithic applications.
- **Security model:** VMs have a more mature security model due to stronger isolation, while containers need additional measures for secure deployment.
- **Use cases:** VMs are traditionally used for running diverse workloads, while containers are popular for microservices, **continuous integration, and continuous delivery** (**CI/CD**) pipelines.
- **Management complexity:** VMs may require more management overhead, especially in large-scale deployments, whereas containers are designed for easier management and scaling.

Virtualization

Virtualization means and involves slightly different concepts depending on whether we are talking about VMs or containers. They are discussed as follows:

- **VMs:** For VMs, virtualization involves creating a virtual version of a physical machine using a hypervisor. It allows multiple VMs to run on a single physical server, each with its own operating system, providing strong isolation.
- **Containers:** For containers, virtualization in the context of containers involves encapsulating an application and its dependencies into a lightweight container. Containers share the host OS kernel, offering efficient resource utilization and portability. Container orchestration tools like Kubernetes help manage containerized applications at scale.

Virtual machines

VM security maturity is based on established security practices translated from computer security on dedicated hardware. Let us discuss these established practices:

- **Software updates:** Update the VM's operating system, hypervisor, and any installed software regularly. Cloud providers offer automated patch management services to simplify this process, and third parties also provide this service in multi-cloud environments.
- **Authentication:** Enforce strong authentication methods such as **multi-factor authentication** (**MFA**) for accessing VMs to prevent unauthorized access, especially considering that VMs in the cloud are often accessible over the internet due to the fact that they often host monolithic applications without much separation between the frontend and the backend, for example.
- **Network security**: Network security rules allow you to control inbound and outbound traffic to VMs based on rules. Define rules to allow only necessary traffic, restricting access to specific ports and IP ranges.
- **Firewalls:** Configure host-based firewalls on VMs to control traffic at the operating system level, blocking unnecessary ports and services.
- **Encrypt data in transit:** Ensure that communication to and from VMs is encrypted using protocols like SSL/TLS. This is crucial for securing data as it travels over networks.
- **Encrypt data at rest:** Enable disk encryption for VMs to protect data stored on their disks. This adds an extra layer of security, especially in the event of data breaches or unauthorized access.

- **Regular backups:** Schedule regular backups of your VMs and store them in a secure, separate location. This is essential for recovering data in case of system failures or security incidents.
- **Least privilege principle:** Follow the principle of least privilege, granting VM users and applications only the minimum permissions necessary to perform their tasks.
- **Role-Based Access Control (RBAC):** Implement RBAC to control and manage user access to VMs based on their roles within the organization.
- **Security groups:** Group VMs based on their functions and assign security policies accordingly. For example, web servers and databases may have different security requirements.
- **Audit logging:** Enable and regularly review audit logs on VMs to detect and respond to suspicious activities. This can help in identifying potential security incidents.
- **Monitoring and alerts:** Implement monitoring solutions to track VM performance and security. Configure alerts to notify you of unusual activities that may indicate a security threat.
- **Vulnerability scanning:** Regularly scan VMs for vulnerabilities using automated tools. Address identified vulnerabilities promptly to reduce the risk of exploitation.
- **Penetration testing:** Conduct penetration testing on VMs to proactively identify and address security weaknesses and vulnerabilities.
- **Unused services:** Disable unnecessary services and features on VMs to reduce the attack surface. Only enable services that are essential for the VM's function.
- **Use bastion hosts:** Implement bastion hosts to control and monitor remote access to VMs. This adds an additional layer of security for accessing VMs.
- **Secure APIs:** If VMs interact with APIs, secure the API endpoints with proper authentication mechanisms and encryption to prevent unauthorized access and data breaches.
- **Resource tagging**: Use resource tagging to categorize VMs and apply security policies based on their tags. This helps in organizing and managing security controls effectively.
- **Security patch management:** Establish a patch management process to promptly apply security updates to VMs, minimizing the window of vulnerability.
- **Incident response plan:** Develop and regularly update an incident response plan specific to VMs. This plan should outline steps to be taken in the event of a security incident involving VMs.

- **Data Loss Prevention (DLP):** Implement DLP solutions to monitor and prevent the unauthorized transmission of sensitive data from VMs.
- **Isolation of environments:** Keep development, testing, and production VM environments isolated to prevent security risks associated with shared resources.
- **Container security:** If using containers within VMs, secure container orchestration platforms and follow best practices for container security. This includes securing container images and ensuring proper isolation.
- **External storage security:** If VMs use external storage services, ensure that access controls and encryption are applied to the stored data, protecting it from unauthorized access.
- **Regular security training:** Provide regular security training for the team responsible for managing VMs. Ensure they are aware of the latest security threats and best practices for securing VMs in the cloud environment.

AWS VM

Here is how you can quickly launch a VM instance in AWS. EC2 is the name of the service in AWS that provides VMs for you to use.

1. **Amazon Elastic Compute Cloud** (**Amazon EC2**) is a web service that provides secure, resizable computing capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction.

   https://aws.amazon.com/ec2/ec2-get-started/

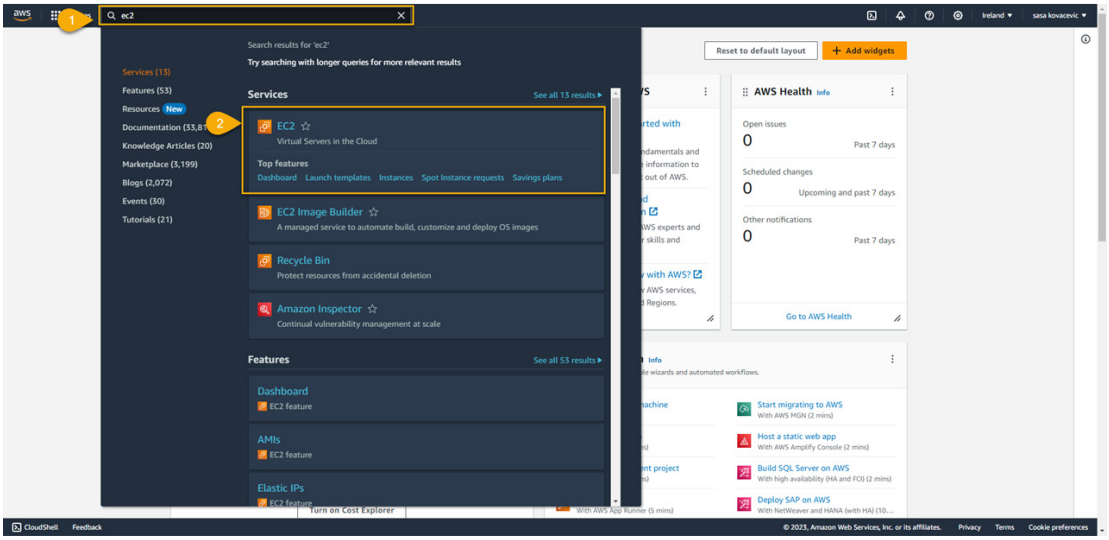


Figure 10.1: *AWS Console – EC2*

2. Go to your AWS console. For example, my home is at:

   https://eu-west-1.console.aws.amazon.com/ec2/home?region=eu-west-1#Home:

3. Type 'ec2' in the search bar.

4. Select EC2 from the search results, as shown in the following figure:

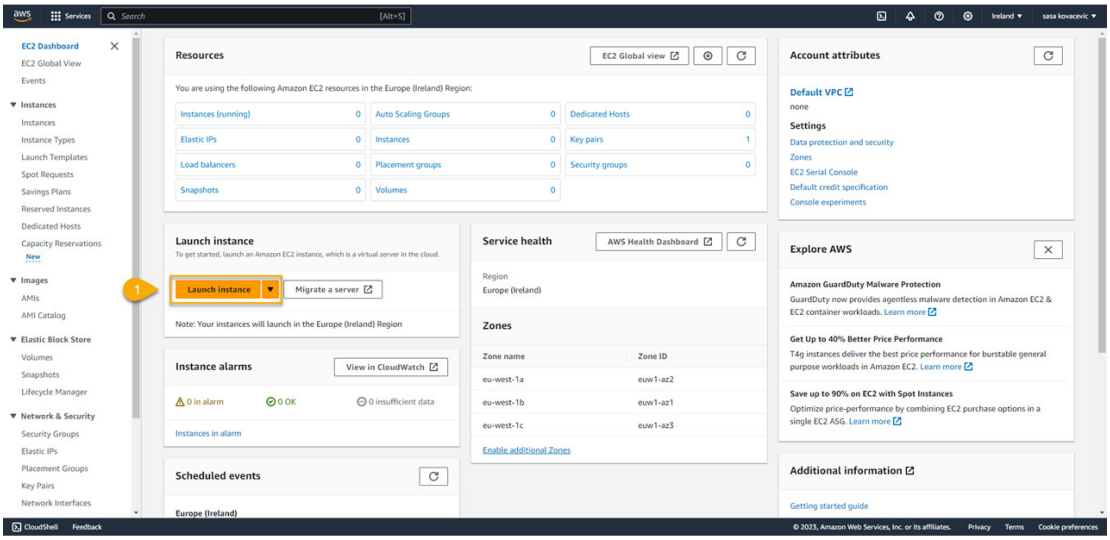


Figure 10.2: *Launch an instance*

5. To launch an EC2 instance, select the **Launch Instance** button, as shown:

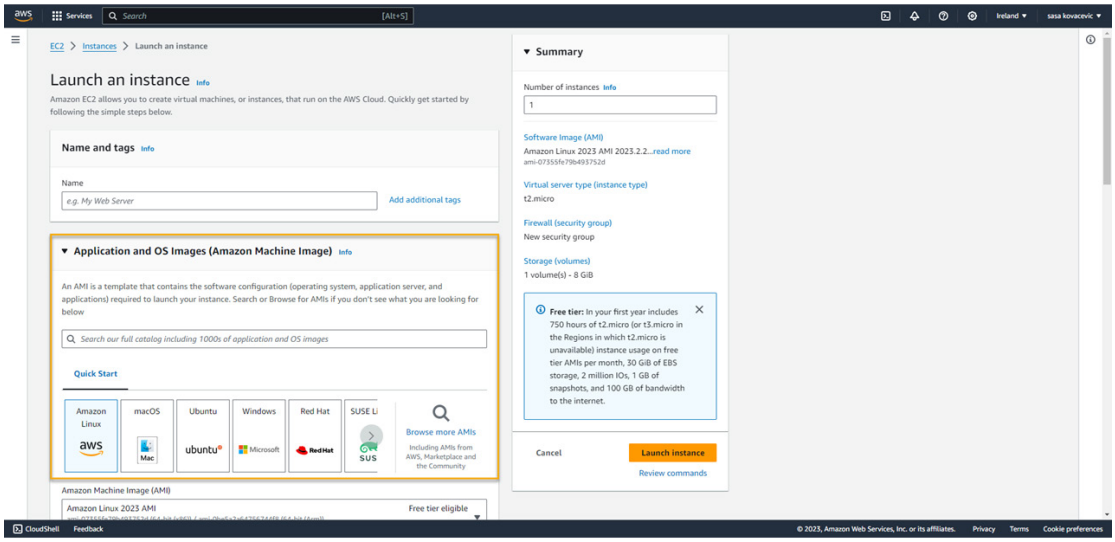


Figure 10.3: *Launch an instance – OS image*

6. You are now on the 'Launch an instance' page.
7. Under the section 'Application and OS images,' you can quickly pick the most common types of instances or search the full catalog, including 1000s of application and OS images. Refer to the following figure:

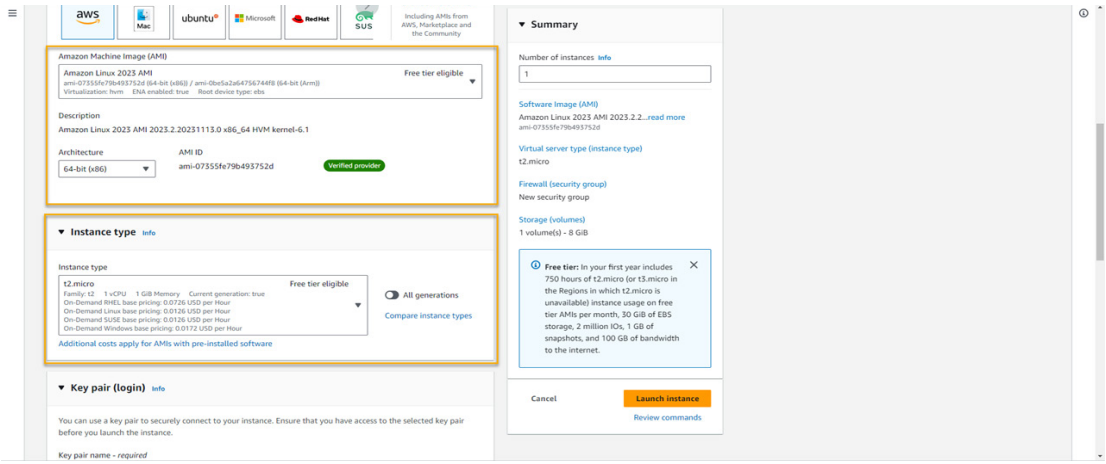


Figure 10.4: *Launch an instance – Amazon Machine Image (AMI) and instance type*

8. In the AMI selection you can further specify exactly which instance you want.
9. AWS maintains its own Linux instance image/s, and if you are new to AWS, these are usually a good default to start with.
10. One great security feature is the 'VERIFIED PROVIDER' callout for some AMIs in the gallery that have been specifically vetted and guaranteed to be coming from a verified provider. You are still responsible for securing the AMI, but at least you start with the knowledge that it has not been tampered with. Refer to the following figure:

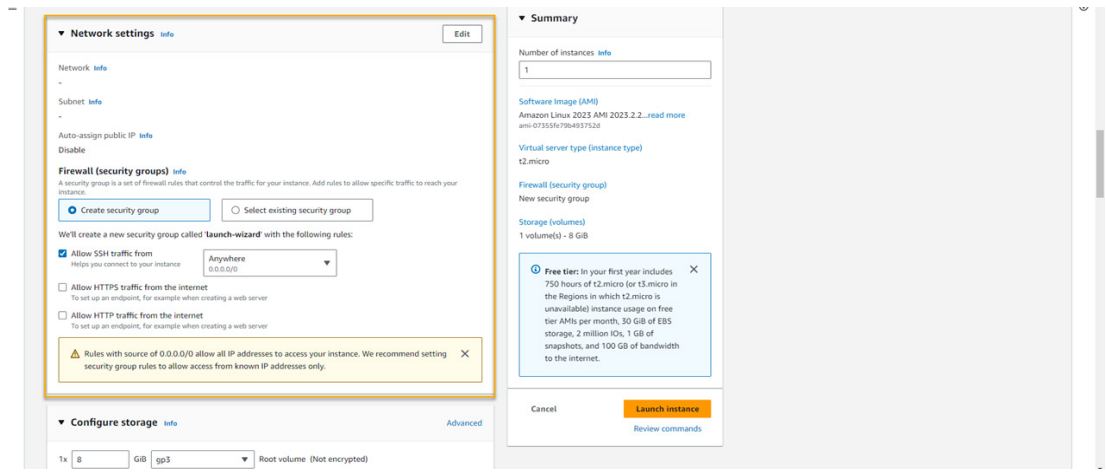


Figure 10.5:

11. Follow the discussed security best practices when setting up networking for your VM.

12. You can allow or restrict traffic to/from your VM, allow only https traffic, and so on, as shown in the following figure:

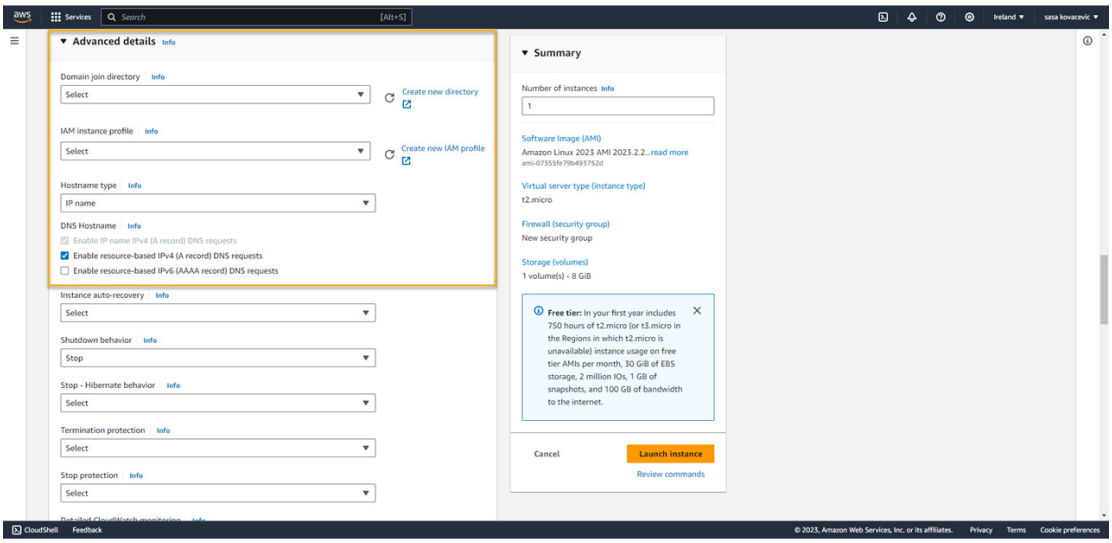


Figure 10.6: *Launch an instance – Advanced details*

13. Under 'Advanced details' you can join your VM to a domain and control other aspects of the VM.

Containers

Securing containers is crucial for maintaining the integrity and security of your applications. Remember that VM security is the basis for container security as well, so consider all the VM security for containers - as well as additional container-based security baselines.

Specifically, when discussing containers, we need to consider:

- **Container image security:** Regularly scan and validate container images for vulnerabilities before deploying them. Use trusted base images and keep images updated with the latest patches.
- **Use lightweight base images:** Choose minimal and lightweight base images to reduce the attack surface and minimize the number of potentially vulnerable components.
- **Container orchestration security:** Secure your container orchestration platform (e.g., Kubernetes) with strong authentication, RBAC, and network policies to control access and prevent unauthorized control of containers.

- **Network segmentation:** Implement network segmentation for containers to isolate services and prevent lateral movement in case of a security breach.
- **API security:** Secure container APIs to prevent unauthorized access. Use authentication and authorization mechanisms to control API access.
- **Immutable infrastructure:** Embrace the concept of immutable infrastructure, where containers are treated as disposable, and any changes result in the deployment of a new container rather than modifying an existing one.
- **Container runtime security:** Utilize container runtimes with built-in security features. There are a few out there, and which ones you should use should be a decision based on your and your teams' skills.
- **Container escape prevention:** Harden the host OS to prevent container escapes. Limit kernel capabilities and use tools from third parties that can assist with security posture management.
- **Container signing:** Implement digital signatures for container images to verify their integrity and authenticity before deployment.
- **Runtime monitoring:** Use container security tools for runtime monitoring to detect abnormal behavior and potential security threats.
- **Resource constraints:** Apply resource constraints to containers to prevent resource abuse and denial-of-service attacks.
- **Secrets management:** Implement secure secrets management practices to protect sensitive information, such as API keys and database credentials, used by containers.
- **File system protection:** Utilize read-only file systems and limit write access inside containers to reduce the impact of container compromise.
- **Audit logging:** Enable and regularly review audit logs for containers to identify and respond to security incidents promptly.
- **Container Identity and Access Management:** Manage container identities and access carefully. Use service accounts and limit permissions to the minimum necessary for each container.
- **Image registry security:** Secure your container image registry with proper authentication and access controls. Use HTTPS for image transfers.

- **Runtime isolation:** Utilize features like namespaces and groups to provide process and resource isolation for containers.
- **Container scanning in CI/CD:** Integrate container image scanning into your CI/CD pipeline to catch vulnerabilities early in the development process.
- **Patch management:** Regularly update and patch the underlying host system and container runtime to address security vulnerabilities.
- **Dynamic threat analysis:** Use tools that perform dynamic analysis of container behavior to identify and respond to evolving threats.
- **Incident response plan:** Develop an incident response plan specifically for container security incidents. Outline procedures for detecting, isolating, and mitigating container-related threats.
- **Container security training:** Train your development and operations teams on container security best practices to ensure a shared understanding of security responsibilities.
- **Container health checks:** Implement health checks within containers to detect and automatically respond to issues, improving resilience and security.
- **Serverless function security:** If using serverless functions within containers, apply security practices specific to serverless computing, such as monitoring and least privilege access.
- **Continuous security monitoring:** Implement continuous security monitoring for containers, including runtime activity, network traffic, and system logs, to quickly identify and respond to security incidents.

AWS containers

Here is how you can quickly launch a container in AWS:

Amazon Elastic Container Service (**ECS**) is the name of the service in AWS that provides container orchestration for you to use.

1. **Amazon Elastic Container Service** (**Amazon ECS**) provides a fully managed container service solution that is easy to use, scalable, secure, and reliable.

   https://aws.amazon.com/ecs/

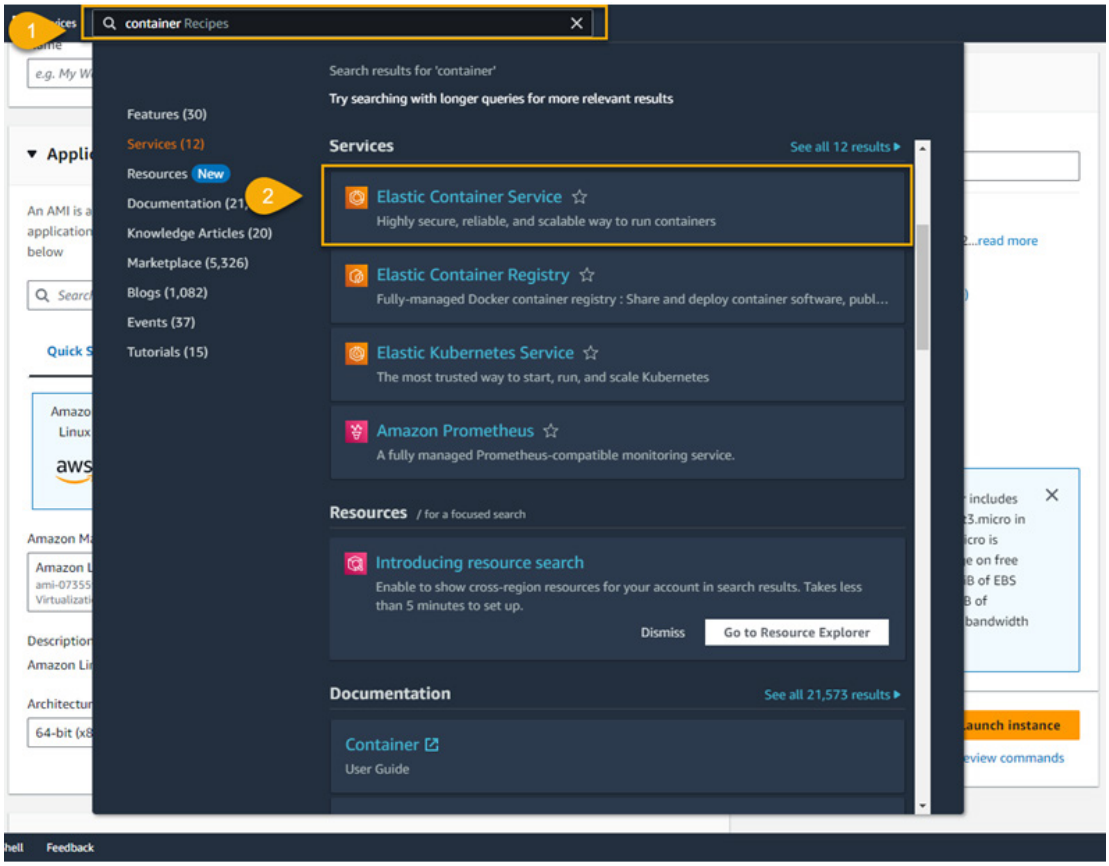


Figure 10.7: *AWS Console – ECS*

2. Go to your AWS console. For example, my home is at:

   https://eu-west-1.console.aws.amazon.com/ec2/home?region=eu-west-1#Home:

3. Type **container** in the search bar.
4. Select **Elastic Container Service** from the search results.

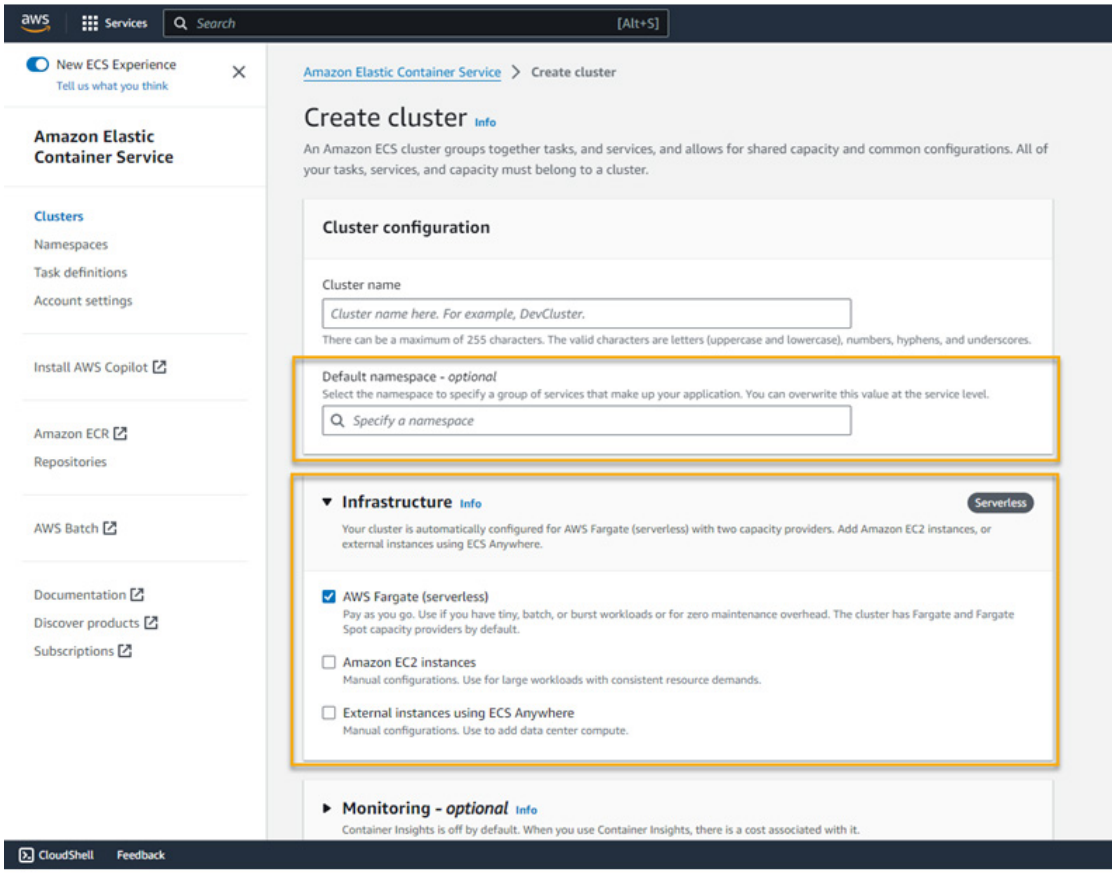


Figure 10.8: *AWS Console – Create cluster*

5. One security setting is exposed immediately. This is the namespace. While optional, it is a good practice to select it as it provides further isolation between containers in different namespaces.

6. You also have the option here to create a cluster using any of the several AWS services that support container deployments.

7. You can use EC2 (VMs from the earlier sections) for high demand and high utilization, or you can use AWS Faregate which is an option to host containers for burstable loads.

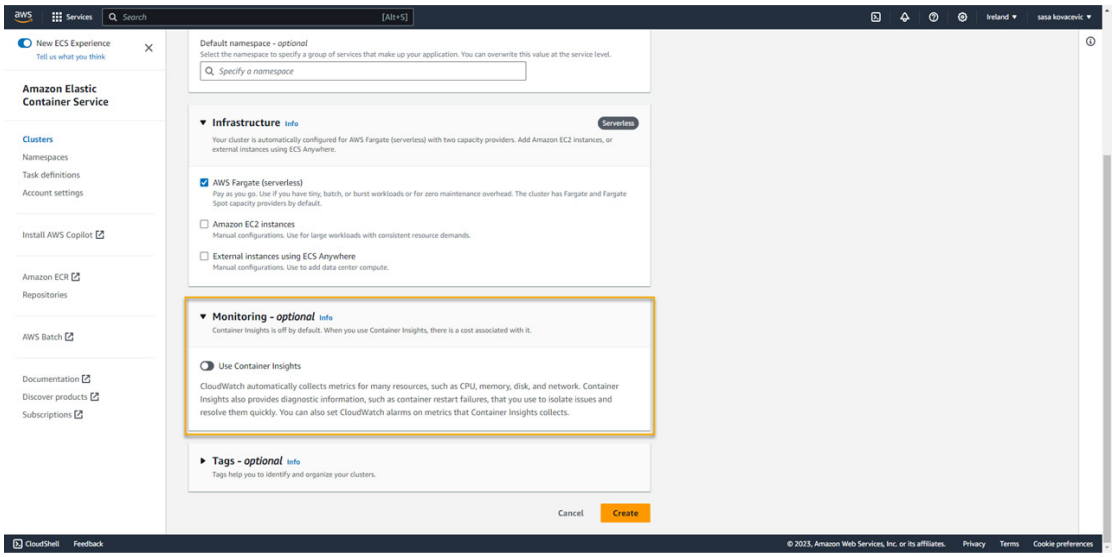


Figure 10.9: *AWS Console – Monitoring*

8. Under 'monitoring,' you can turn on the collection of container insights, which uses an AWS service called Cloud Watch (Container) that automatically collects, aggregates, and summarizes metrics and logs. If this book has taught you anything, then you should always turn these on.

Orchestration

Container orchestration is the automated process of managing, deploying, scaling, and operating containerized applications. Containers are lightweight, portable, and consistent environments that package applications and their dependencies and container orchestration tools help streamline the deployment and management of containerized applications in large-scale, dynamic environments.

Key features of orchestration engines (such as Kubernetes – the *de facto* standard) are:

- Deployment automation
- Scaling
- Load balancing
- Service discovery
- Rolling updates and rollbacks

- Health monitoring
- Resource management
- Declarative configuration
- Security

When should you consider orchestration a key component of your container strategy? The answer is that in almost all but the most trivial cases, you need orchestration. If you are experimenting or just starting out or running only a few containers, orchestration might not be that important to you, but once you start using containers *en masse*, you and your organization will need an orchestration strategy, including sections for all the key features listed in this section.

As **Kubernetes** (**k8s**) is the standard today, you would be best position to look into it if your organization wants to go down the container route.

Migration

Let us consider these migration scenarios:

- Migration from on-premises to VMs
- Migration from on-premises to containers

On-premises to VMs

Migrating from on-premises infrastructure to VMs in the cloud is a significant undertaking. To ensure a successful migration, consider the following key pieces of information:

- Understand the existing on-premises workloads, their dependencies, resource requirements, and performance characteristics. This assessment helps in planning for the appropriate cloud resources.
- Develop a comprehensive data migration strategy. Determine how data will be transferred to the cloud, considering factors such as volume, speed, and the impact on ongoing operations. Choose between strategies like bulk transfer, incremental sync, or streaming.
- Identify and understand security and compliance requirements for your workloads. Ensure that the cloud provider's security measures align with your organization's standards. Implement encryption, access controls, and other security best practices.
- Assess the network connectivity between your on-premises environment and the cloud. Consider factors such as latency, bandwidth, and security. Plan for VPN connections, Direct Connect (in the case of AWS), or ExpressRoute (in the case of Azure) for secure and efficient communication.

- Review software licensing agreements and understand how licensing models change in the cloud. Consider the cost implications of cloud services, including computing, storage, data transfer, and additional services. Explore cost-saving options like reserved instances or spot instances.
- Evaluate the performance requirements of your workloads and determine the appropriate cloud VM sizes and configurations. Consider scalability aspects, such as auto-scaling capabilities, to handle varying workloads efficiently.
- Plan for high availability and disaster recovery in the cloud. Utilize features such as availability zones, redundant storage, and automated backup solutions. Consider implementing a multi-region or multi-cloud strategy for enhanced resilience.
- Ensure seamless integration with existing on-premises systems, applications, and databases. Plan for the necessary adjustments in configurations, network settings, and authentication mechanisms to maintain business continuity.
- Establish a robust monitoring and management strategy. Leverage cloud-native monitoring tools to track the performance, availability, and health of your VMs. Implement logging and alerting to quickly respond to issues.
- Invest in training for your IT staff to familiarize them with cloud technologies. Cloud platforms often have unique features and management interfaces, so ensuring your team is well-equipped is crucial for successful operations in the cloud.

Thorough planning and collaboration with stakeholders are key to a smooth and successful migration to VMs in the cloud.

On-premises to containers

Migrating from on-premises infrastructure to containers in the cloud involves specific considerations to ensure a smooth transition.

Consider the following points:

- Evaluate existing on-premises applications to determine their suitability for containerization. Identify dependencies and assess the effort required to containerize each application.
- Choose a container orchestration platform that aligns with your needs. Kubernetes is a popular choice, but other options might be more suitable depending on your requirements and your existing infrastructure.
- Consider transitioning towards a microservices architecture, breaking down monolithic applications into smaller, more manageable services. Containers are well-suited for microservices, allowing for greater scalability and flexibility.

- Plan for data management and storage in a containerized environment. Decide whether to store data within containers, use external storage solutions, or leverage cloud-native services. Ensure data persistence and consider the statelessness of containers.
- Adapt your networking infrastructure for containerized applications. Implement container networking to allow communication between containers and external services. Consider service discovery mechanisms to dynamically locate and connect to containers.
- Implement robust security practices for containerized applications. This includes securing container registries, implementing network policies, using container security tools, and ensuring proper access controls.
- Establish a CI/CD pipeline for automated container builds, testing, and deployment. Automate the process of pushing container images to a container registry and deploying them to the cloud environment.
- Implement monitoring and logging solutions specifically designed for containerized environments. Utilize container orchestration tools for monitoring container health, resource usage, and application performance. Set up centralized logging for better visibility.
- Leverage the dynamic scaling capabilities of containers to efficiently utilize cloud resources. Set up auto-scaling based on application demand and optimize resource allocation to minimize costs.
- Invest in training for your development and operations teams to ensure they are proficient in container technologies and cloud-native practices. Familiarity with container orchestration tools and cloud services is essential for successful containerization.

Remember that migrating to containers involves a shift in both technology and mindset. The key is to plan comprehensively, test thoroughly, and gradually transition your workloads to ensure a successful migration and maximize the benefits of containerization in the cloud.

Conclusion

In this chapter, we discussed VM and container technologies and what you should consider from a cloud security perspective.

You have learned not just about running VMs and containers in the cloud but also about thinking ahead and preparing for the needs of your ever-expanding VM and/or container workload. We discussed orchestration to start understanding the needs of managing hundreds and thousands of containers. We also discussed migration paths from on-premises to VMs and from on-premises to containers.

In the next chapters, we will showcase some alternatives to both the VM and container strategies. While VMs and containers are universally available and interchangeable (to a point), some of the more advanced technologies may not be so universal and/or interchangeable, but they will provide you and your organization with many other options for computing in the cloud.

Questions

1. What are the key differences between VMs and containers?
2. What are the considerations when migrating on-premises services into the cloud? What specifically is important to consider when migrating into VMs, and what when migrating into containers?
3. Can you create a Windows VM in the cloud through the portal/console?
4. Can you create a Linux VM in the cloud through the portal/console?
5. Can you create a Kubernetes cluster in the cloud?
6. Why is regular training on technologies used by your and your organizations' teams in the cloud a key requirement to cloud adoption?
7. Which service would you use to be able to connect to your VMs in the cloud?

References

- Get Started with AWS EC2:
  https://aws.amazon.com/ec2/ec2-get-started/
- Get Started with AWS ECS
  https://aws.amazon.com/ecs/

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 11
Serverless

Introduction

The best way to perform 'computing' in the cloud is serverless. There is still a huge number of legacy workloads, which is where VMs and containers come in. While containers saddle the border between legacy and cloud-native, there is no doubt that serverless is cloud-native in its purest form.

What is serverless? Serverless is a way to run (small snippets of) code without managing the underlying infrastructure in the cloud [for example, VMs or containers (you can do serverless containers as well)].

This means you can write functions as small as a few lines of code, deploy them, and run them. How are a few lines of code useful? Event-based triggering and processing are extremely powerful as they can be chained, and functions can be developed independently. They can always be changed or adapted to do more or less as required. It also serves as a basis for no-code solutions, which enables citizen developers (meaning non-engineering folk) to be (almost) as productive as engineers.

Since each unit may be so extremely tiny that it appears insignificant, proper security procedures are often skipped or assumed - both terrible ideas.

Structure

The chapter covers the following topics:

- Introduction to serverless
- Cloud Functions
- Events
- Triggers
- Security

Objectives

In this chapter, we will discuss the serverless model and functions, and using Google Cloud Platform as an example, we will show how to deploy, code, and test these functions rapidly.

We will also discuss the security implications of serverless and how to protect your serverless workloads. By the end of this chapter, you should be well-versed in understanding the model of serverless functions and how they work.

Finally, you should also be able to confidently think of the security model around serverless functions and how to start protecting these at scale. One main characteristic of serverless functions is that they are usually quite small, and so are relatively easy and quick to develop and deploy, which leads to proliferation; once you and your organization start on the path of serverless functions, you will quickly need to manage dozens, hundreds or thousands of them.

Introduction to serverless

Serverless can be any cloud service where you do not need to manage servers. Many services have such capabilities. Even containers can be deployed on a serverless service, meaning the cloud provider will host the containers on your behalf on the underlying architecture. You have to only provide some parameters around scaling and sizing of the container, that is, how many instances you want and how much memory and CPU should be allocated to each.

This model allows legacy applications and services to be containerized and deployed alongside modern decoupled workloads.

In this chapter, we focus on that modern and decoupled model by looking into serverless functions, small, independent pieces of code running in the cloud that perform any and all functions your application or service requires.

Before we go into that, here are some specific differences between serverless and non-serverless (traditional) services in cloud computing for comparison:

- **Infrastructure management:**
  - **Serverless**: Developers do not need to manage or provision servers. The cloud provider automatically handles server management, scaling, and maintenance.
  - **Non-serverless**: Developers are responsible for provisioning, configuring, and managing servers or virtual machines.

- **Scaling:**
  - **Serverless**: Automatically scales based on demand. It can handle varying workloads without manual intervention.
  - **Non-serverless**: Scaling typically requires manual configuration or the use of auto-scaling groups, and it may not be as instantaneous as in a serverless environment.

- **Billing model:**
  - **Serverless**: Billing is based on actual usage, typically measured in function execution time and resources consumed.
  - **Non-serverless**: Billing is often based on reserved or provisioned capacity, regardless of whether the resources are actively used.

- **Execution model:**
  - **Serverless**: Code is executed in short-lived, stateless functions. Each function is triggered by an event (for example, HTTP request or database change).
  - **Non-serverless**: Applications are typically deployed as long-running processes or services that handle multiple requests over an extended period.

- **Operational complexity:**
  - **Serverless**: Reduces operational overhead as the cloud provider manages infrastructure, updates, and security patches.
  - **Non-serverless**: Requires more operational management, including patching, updating, and ensuring the availability and performance of servers.

- **Cold starts:**
  - **Serverless**: Applications may experience latency, known as *cold starts*, when a function is invoked for the first time or after a period of inactivity.
  - **Non-serverless**: Applications may have a more consistent performance profile since servers are typically pre-warmed or always running.

- **Use cases:**
  - **Serverless**: Well-suited for event-driven, short-duration, and stateless functions. Ideal for microservices architecture.
  - **Non-serverless**: Better for long-running applications or services that require persistent connections or specialized configurations.

Serverless services

Across the three hyperscale clouds, there are a lot of serverless services, so let us look at some of them:

- **AWS:**
  - In AWS, you have AWS Lambda and AWS Fargate.
    - AWS Lambda is *an event-driven, pay-as-you-go computi service that lets you run code without provisioning or managing servers*.
    - AWS Fargate is *a serverless compute engine that works with Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS)*.
  - Then, there are various other application integration services and data services such as AWS Step Functions, Amazon Dynamo DB, and others.
    - AWS Step Functions is *a visual workflow orchestrator that makes it easy to sequence multiple AWS services into business-critical applications*.
    - Amazon Dynamo DB is *a key-value and document database service that delivers single-digit millisecond performance at any scale*.
- **Azure:**
  - In Azure, you have serverless functions and Serverless Kubernetes.
    - Serverless functions execute code written in the language of your choice with Azure Functions, an event-driven computing experience. Scale on demand and pay only for the time your code is executed. Available as a managed service in Azure and Azure Stack, the open source functions runtime also works on multiple destinations, including Kubernetes, Azure IoT Edge, on-premises, and even in other clouds.
    - Serverless Kubernetes elastically provision pods inside container instances that start in seconds without the need to manage additional compute resources. Create serverless, Kubernetes-based applications using the orchestration capabilities of **Azure Kubernetes Service** (**AKS**) and AKS virtual nodes, which are built on the open-source Virtual Kubelet project. Get the best of an event-driven approach by adding KEDA event-driven autoscaling to your AKS cluster. KEDA is an open-source component that enables containers to process events directly from event sources, and it provides the ability to scale to zero.

  - Then, there are various other storage, integration services, and data services such as Azure Serverless Storage, Logic Apps, and Azure Cosmos DB.
    - **Azure Storage**: Build static web applications on Azure Blob storage or use it as massively scalable storage for unstructured data. Leverage storage events to respond to operations on blobs with multiple serverless architectures. Blob events are pushed through Event Grid to subscribers using functions, Logic Apps, or even your own custom HTTP listener.
    - **Azure Logic Apps**: Azure Logic Apps is a leading **integration Platform as a Service** (**iPaaS**) built on a containerized runtime. Deploy and run Logic Apps anywhere to increase scale and portability while automating business-critical workflows anywhere.
    - **Azure Cosmos DB**: Use Azure Cosmos DB, a globally distributed, massively scalable, multi-model database service, to create database triggers, input bindings, and output bindings.

- **GCP:**
  - GCP has Cloud Run and Cloud Functions.
    - **Cloud Run**: Build applications in your favorite language, dependencies, and tools and deploy them in seconds. Cloud Run abstracts away all infrastructure management by automatically scaling up and down from zero almost instantaneously - depending on traffic. Cloud Run only charges you for the exact resources you use.
    - **Cloud Functions**: Develop faster by writing and running small code snippets that respond to events. Use Cloud Functions to connect with Google Cloud or third-party cloud services via triggers to streamline challenging orchestration problems.

Now, let us deploy a Cloud Function in GCP. The most similar services in AWS and Azure are AWS Lambda and Azure Functions, so once you learn the GCP way, you can easily try it in AWS and Azure, with remarkably similar steps and setup options.

Cloud Functions

Setting up a Cloud Function requires some setup steps ahead of actual function deployment, and because most of the work is done in the console and very little is done in the actual function, there are a few more steps than when deploying some of the more traditional services, but as you will see, each step is very simple.

The complexity comes from managing the functions themselves and the underlying security model.

So, let us begin with signing into the GCP Console. You can, as with other cloud providers, create a free 'demo' account at any time. You will need to provide your billing details, even though nothing we will do here should cost you.

Once you are signed in to the console, continue with the following steps to p-provision a serverless function:

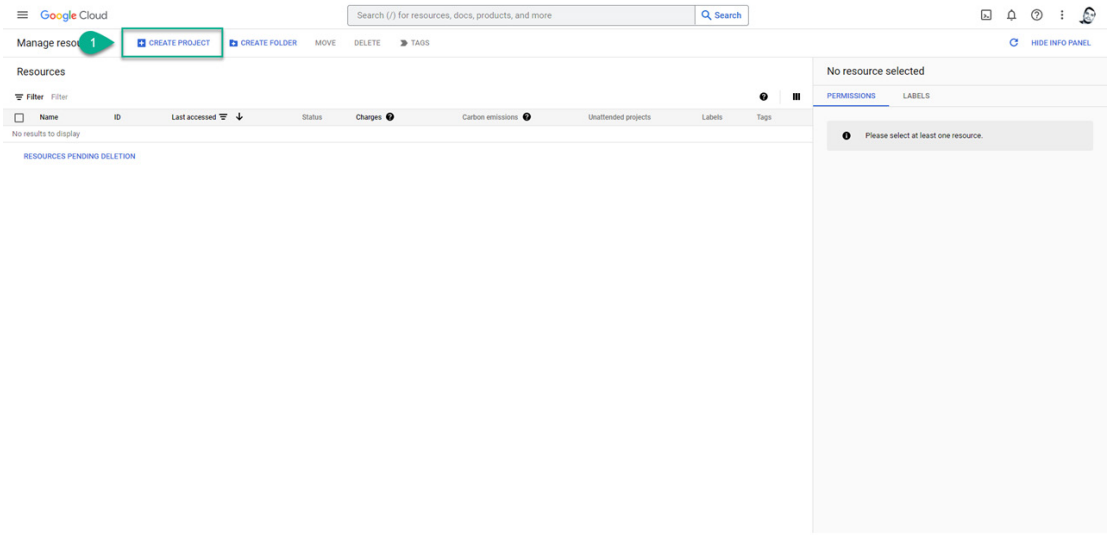


Figure 11.1: *GCP Console – Create a project*

1. First, we will create a project, so tap the **Create Project** menu item. You can use any existing projects, but clean-up and deletion will be easier if you perform this exercise in a separate project. To start, create a project:

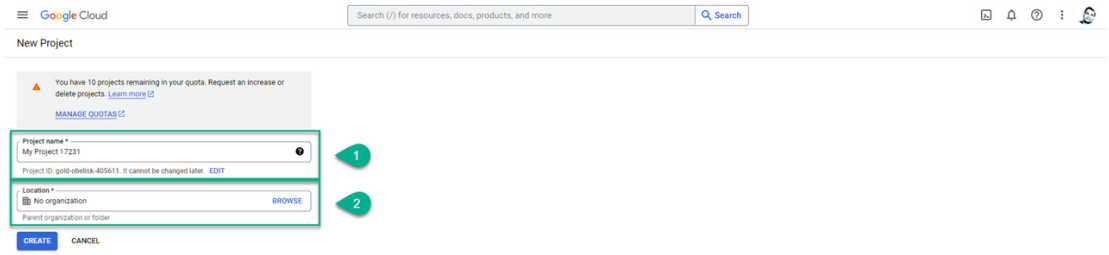


Figure 11.2*: GCP Console – Create Project details*

2. Type in the project name and the organization. You can also leave them as defaults, as shown. Usually, you would follow your organization's structure and naming conventions, but for this example, it does not matter. You should just remember the project's name so you can select it in the next steps. First, search for the function:

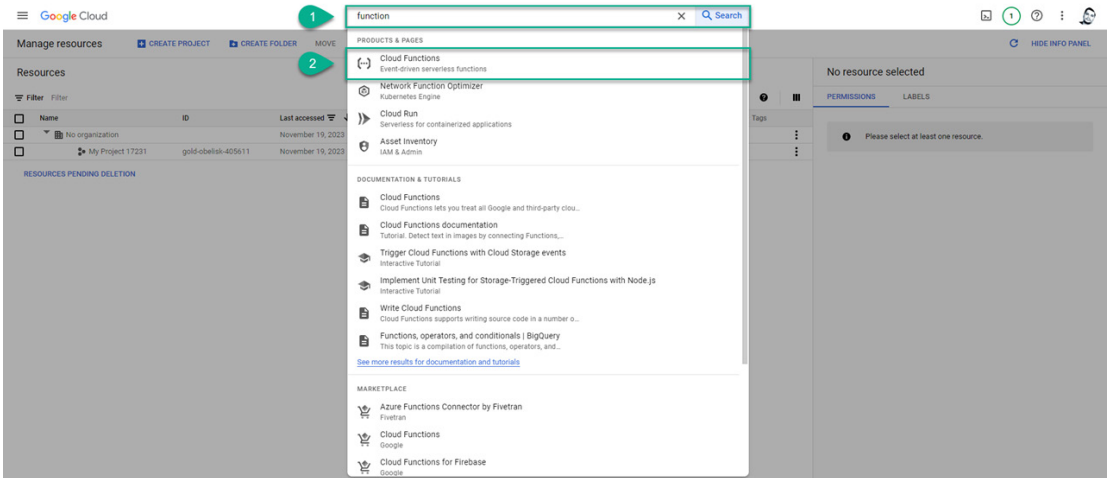


Figure 11.3: *GCP Console – Search for Cloud Functions*

3. In the search field, type **function**.
4. Select the **Cloud Function** option from the search results:

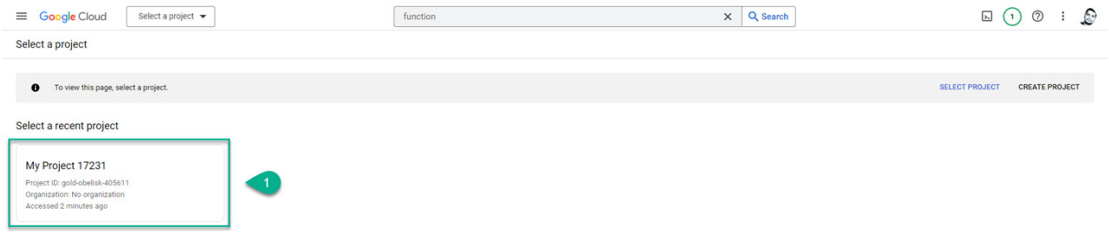


Figure 11.4*: GCP console – Project selection*

5. Select the project you created previously. This is the project in which we will create a function:

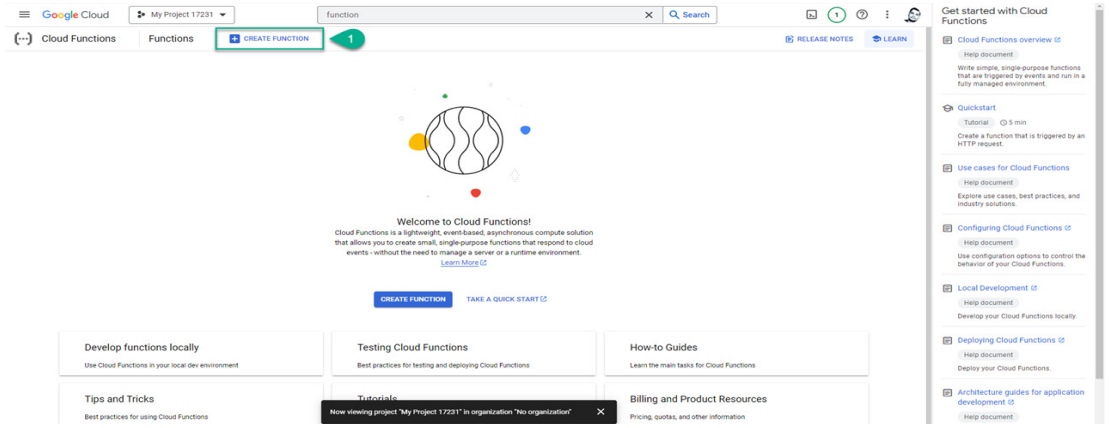


Figure 11.5

6. Select the **Create Function** option from the Cloud Functions menu:

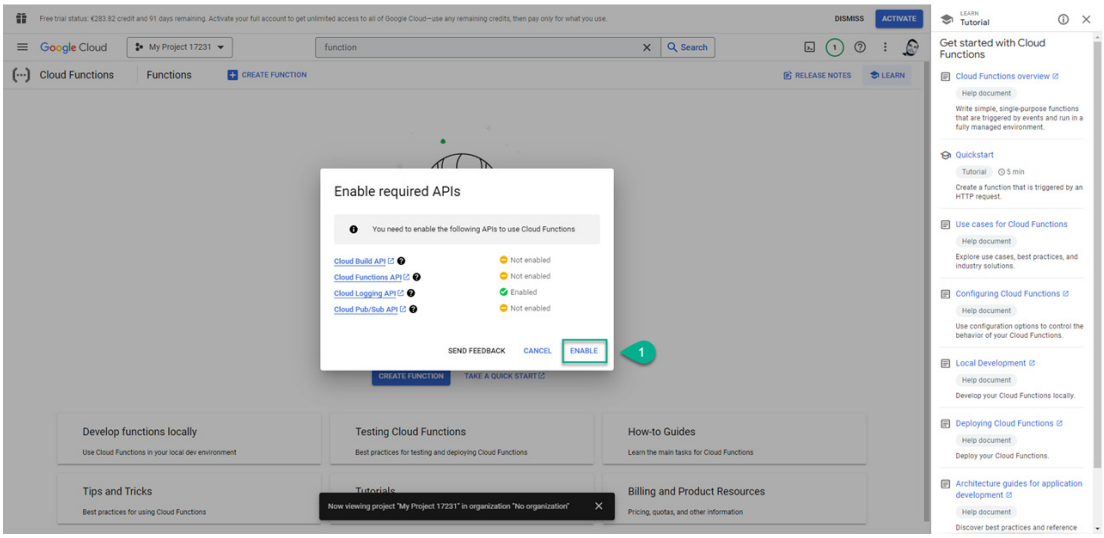


Figure 11.6: *GCP console – Enable required APIs (1)*

7. For GCP to protect you and themselves, not all required APIs are enabled for every project. As good security practices are observed here, these APIs were not enabled by default to restrict the security perimeter.

8. However, since we need these APIs now to continue deploying our function, we need to enable them.

9. So, note the APIs we are enabling and tap **Enable** to continue:

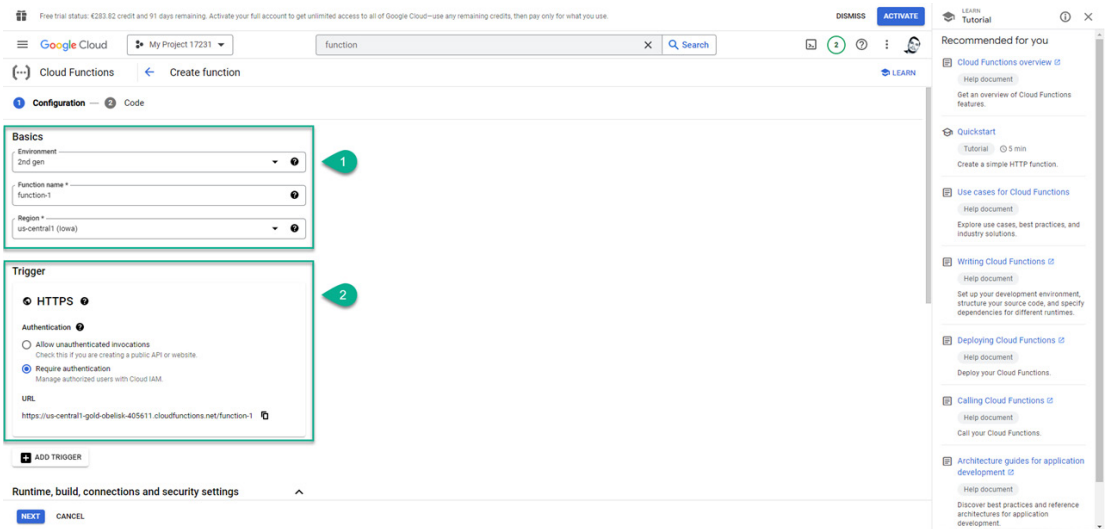


Figure 11.7

10. Fill in the basic information, such as the environment and the name and region.
11. Select **2nd gen** for the environment option.
12. For the trigger, note that the default options are **HTTPS** and **require authentication**. These are great security defaults. For this function, we will need to select **Allow unauthenticated invocations**, as that makes sense for a public API and makes these example steps easier:

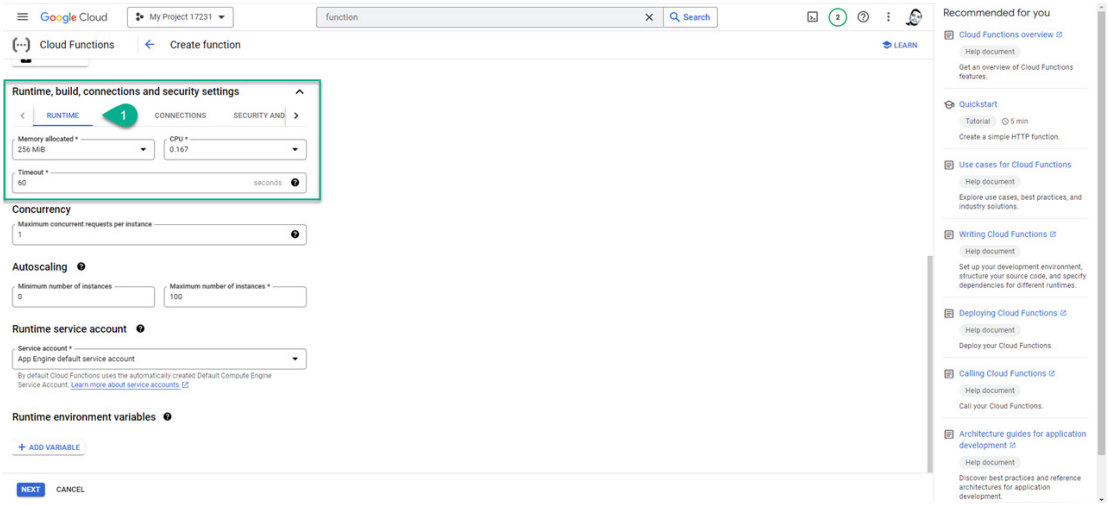


***Figure 11.8**: GCP console – Create a new function (continued)*

13. Under the **runtime** option, we specify the memory and CPU allocations, as we discussed previously in this chapter. As you can see, functions can be tiny and require very little memory and CPU:

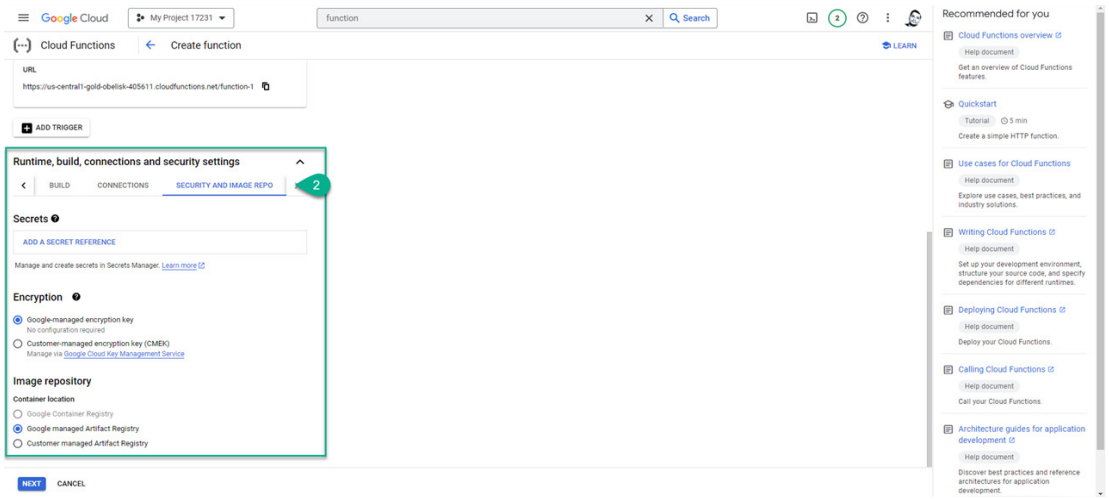


Figure 11.9:

14. Under **Security and Image** Repo, you can check out the security and repository options, both of which concern securing the function we are about to create.

15. Browse and understand all the options, but do not change any defaults for this example:

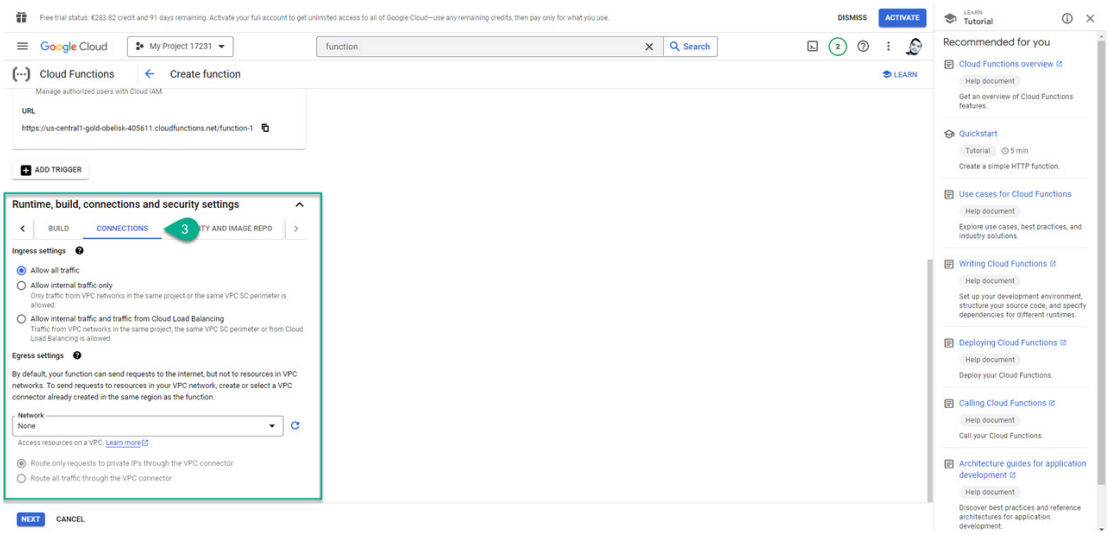


***Figure 11.10**: GCP console – Create a new function (continued)*

16. Under the **Connections** option, you can allow traffic from the internet, internally from your services, and other GCP services. Here, we want to allow all traffic to minimize troubleshooting of the next steps:

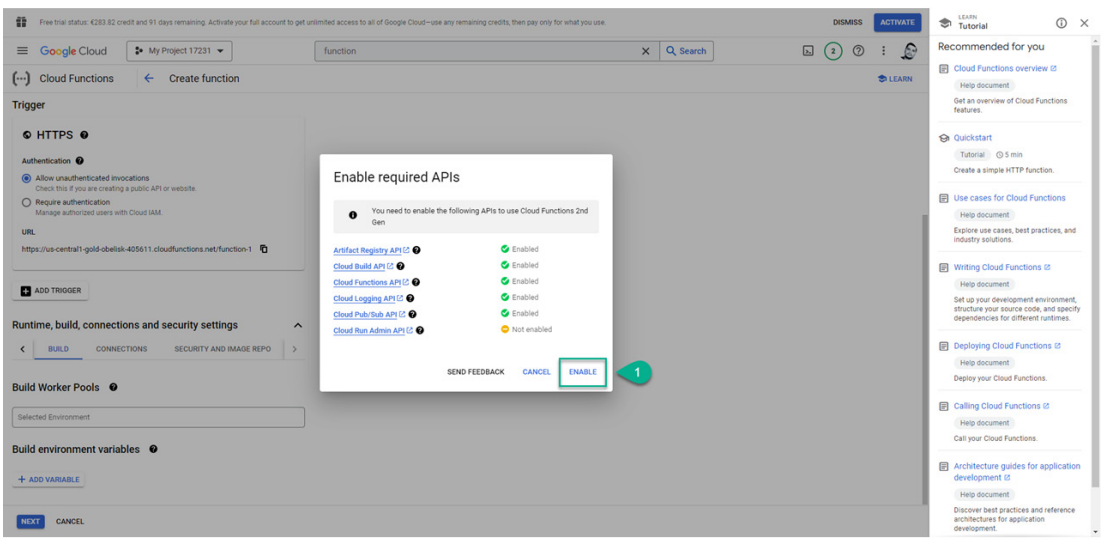


***Figure 11.11**: Enable required APIs (2)*

17. This is where another round of APIs that need to be enabled. Again, GCP here follows the least-privilege security principle we have talked about in previous chapters and only requires enabling specific APIs when they are needed for what we are trying to accomplish. Go ahead and enable them:

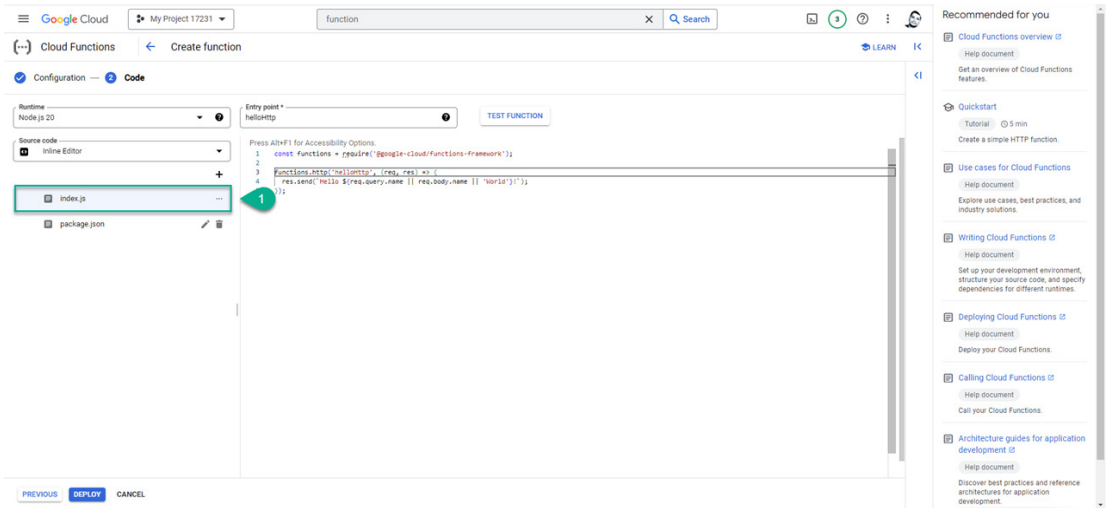


***Figure 11.12**: GCP console – index.js*

18. By default, we get an example function with very minimal code, but it is good enough for this demonstration.

19. You can switch between the two files required for this function to run.

20. The index.js file contains the function's content, the actual code that will run when the function is called:

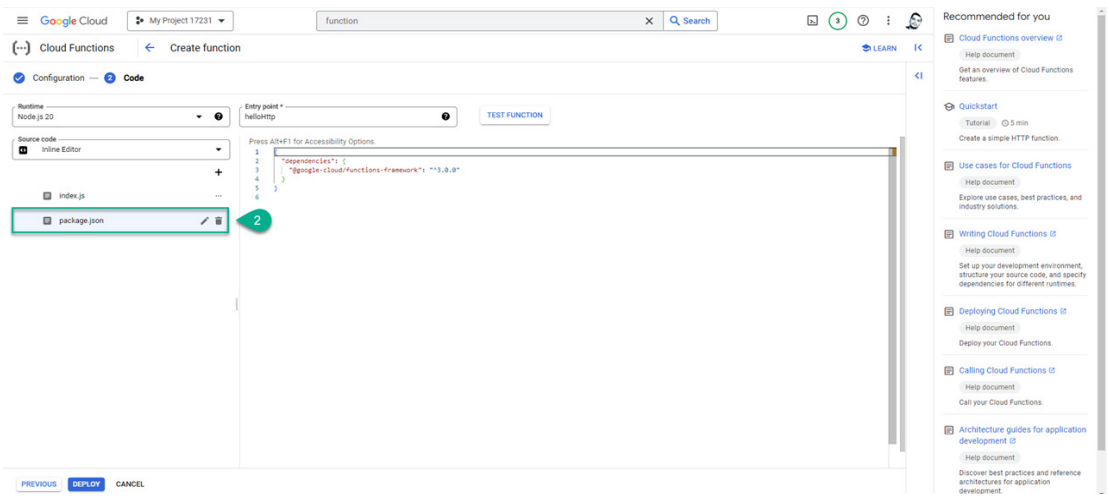


Figure 11.13:

21. In the **package.json** file, you can see the dependencies needed for the function to run:

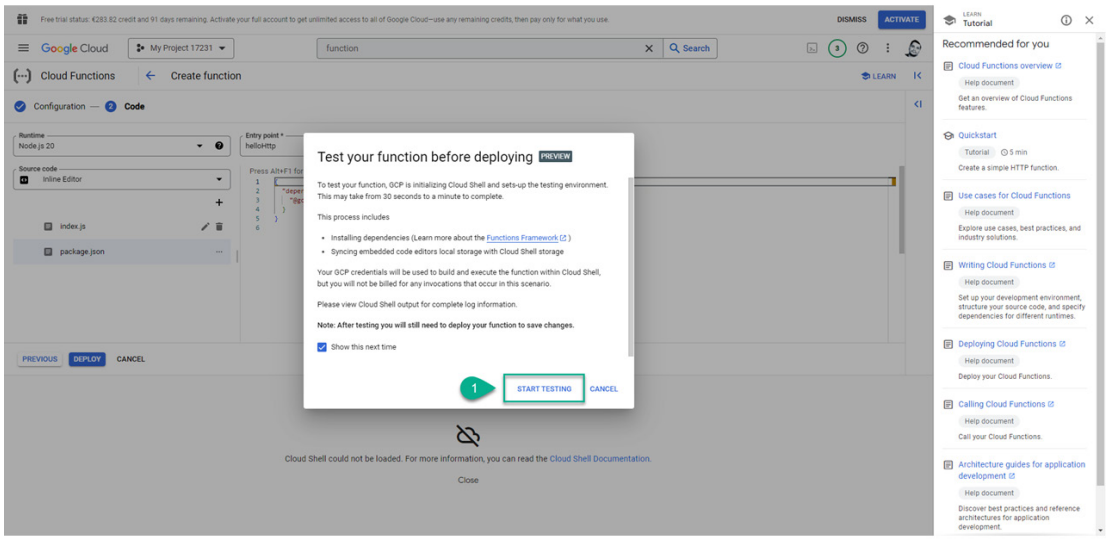


Figure 11.14: *GCP console – Test*

22. GCP has in-place function testing, which is amazing as per our requirements. So, select **Start Testing** and proceed.

23. The environment may need a few minutes to get provisioned for the test, so wait a few minutes while this completes (you will see progress indicators), and then proceed to the next step:

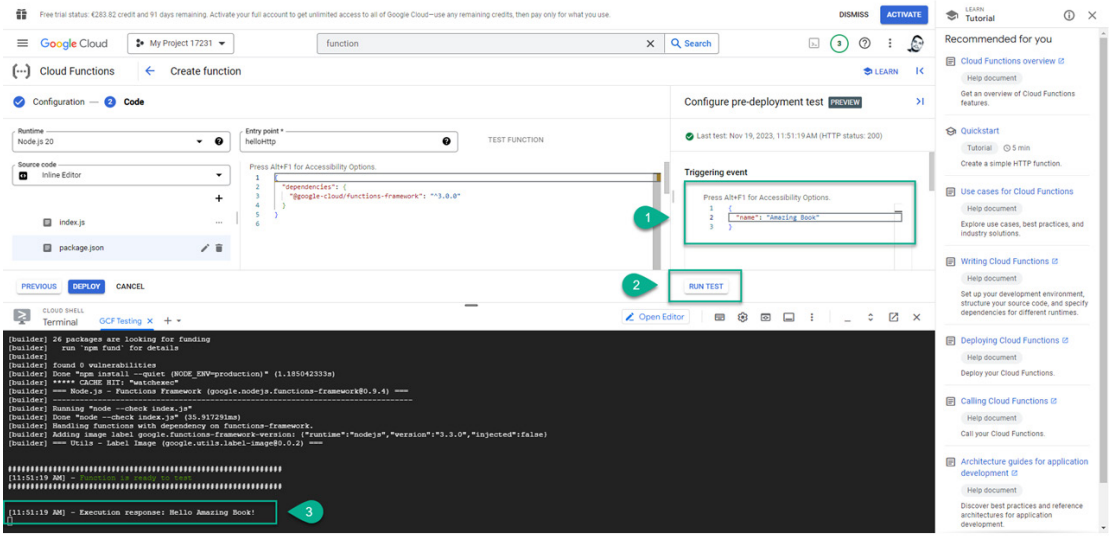


Figure 11.15

24. We have expanded the bottom **Terminal** panel, so you should do the same to see more of its content if it is too small for you.
25. In the triggering event box, we can change the payload event (the request). Change it to your name.
26. Let us describe what the function does when invoked. It reads the name parameter and inserts it into the output. In the example above, it takes the *Amazing Book* text and directly inserts it into the output.
27. In the **Terminal**, you can see the output after you change the payload and tap the **Run Test** button.

By following these steps, you have deployed your first function into GCP.

Hopefully, you have also noticed a security violation; we pass the input parameter text directly into the output without validating or sanitizing it. As the function is written in JavaScript, if you are familiar with it, you can change the code of the function in the **index.js** file to include validation and sanitization, perhaps by limiting the length and the characters allowed to be accepted as input. If you do change it, you can tap the **Run Test** button to retest the function.

Do not forget to delete the project you created in GCP, which contains this and other functions you may have tried, to prevent an unsecured and unmonitored public function endpoint in your GCP environment.

Security

Let us discuss the security implications of serverless functions through a series of examples and mitigation strategies:

- **Authentication and authorization:**
    - **Example**: Lack of proper authentication allows an unauthorized user to invoke a sensitive serverless function, leading to unauthorized access to confidential data or services.
    - **Mitigation**: Implement robust authentication mechanisms, such as OAuth or API keys, and use **role-based access control** (**RBAC**) to enforce proper authorization. Regularly audit and review access permissions.
- **Data encryption:**
    - **Example**: Failing to implement encryption for data in transit enables a **Man-in-the-Middle** (**MitM**) attacker to eavesdrop on the communication between serverless functions and compromise sensitive information.
    - **Mitigation**: Use HTTPS for data in transit and employ encryption services or features provided by the serverless platform for data at rest. Leverage managed services for key management.

- **Insecure dependencies:**
  - **Example**: A popular third-party library used in a serverless function has a known vulnerability, and an attacker exploits this vulnerability to execute arbitrary code or compromise the serverless environment.
  - **Mitigation**: Regularly update dependencies to the latest secure versions. Use dependency scanning tools to identify and address vulnerabilities in third-party libraries.
- **Cold start attacks:**
  - **Example**: During the initialization of a cold start, insufficient security measures allow an attacker to inject malicious code or compromise the integrity of the serverless function.
  - **Mitigation**: Harden serverless functions during initialization. Employ code integrity checks and validate inputs to prevent injection attacks during cold starts.
- **Denial of Service (DoS) attacks:**
  - **Example**: An attacker floods a serverless application with a high volume of requests, causing it to scale up unnecessarily and resulting in increased costs or service degradation for legitimate users.
  - **Mitigation**: Implement rate limiting, use traffic analysis tools to detect abnormal patterns, and leverage cloud provider services that offer DDoS protection.
- **Event data injection:**
  - **Example**: An attacker manipulates input data in an event trigger, injecting malicious code executed when the serverless function processes the event.
  - **Mitigation**: Validate and sanitize input data from event triggers. Use parameterized queries to prevent injection attacks. Implement input validation and filtering.
- **Secrets management:**
  - **Example**: Hardcoding API keys in serverless function code exposes sensitive credentials, allowing a malicious actor to gain unauthorized access to external services or resources.
  - **Mitigation**: Use secure storage solutions for secrets. Avoid hardcoding credentials in code and utilize environment variables or dedicated secrets management tools.

- **Logging and monitoring:**
  - **Example**: Insufficient logging and monitoring allow an attacker to perform actions without detection, and security incidents go unnoticed until significant damage has occurred.
  - **Mitigation**: Implement comprehensive logging and monitoring. Use tools to set up alerts for suspicious activities. Review logs regularly for unusual patterns.
- **Permissions and least privilege:**
  - **Example**: Overly permissive permissions grant a serverless function more access than necessary, enabling an attacker to perform unauthorized actions on other resources.
  - **Mitigation**: Follow the principle of least privilege. Assign the minimum permissions necessary for each function and regularly audit and review access controls.
- **Network security:**
  - **Example**: Inadequate network security configurations allow an attacker to intercept or manipulate the communication between serverless functions and other services.
  - **Mitigation**: Secure network connections using Virtual Network service endpoints. Implement firewalls and security groups to control incoming and outgoing traffic.
- **Code injection:**
  - **Example**: poor input validation in a serverless function allows an attacker to inject malicious code, leading to code execution vulnerabilities.
  - **Mitigation**: validate and sanitize input data. Use parameterized queries for database interactions. Implement security controls like **Web Application Firewalls** (**WAF**) for web-based serverless applications.
- **Cross-Site Scripting (XSS):**
  - **Example**: Lack of input validation in a web-based serverless application allows an attacker to inject malicious scripts, compromising the security of users interacting with the application.
  - **Mitigation**: Implement input validation and output encoding. Use **Content Security Policy** (**CSP**) headers to restrict the sources of executable scripts.
- **Integrity of serverless code:**
  - **Example**: An attacker gains unauthorized access to the serverless environment and modifies the code of a deployed function, introducing malicious logic.

- **Mitigation**: Implement code signing and verification mechanisms. Regularly monitor and audit changes to the serverless codebase. Use version control systems.

- **Resource enumeration:**
  - **Example**: Inadequate access controls enable an attacker to enumerate available serverless functions, identifying potential targets for exploitation.
  - **Mitigation**: Implement proper access controls to prevent unauthorized enumeration. Use tools and services that detect and block enumeration attempts.

- **Data storage security:**
  - **Example**: Insufficient access controls on serverless function storage allow an unauthorized user to access or modify stored data.
  - **Mitigation**: Use secure storage solutions with proper access controls. Encrypt sensitive data at rest. Regularly audit and review access permissions to storage resources.

- **Ephemeral storage risks:**
  - **Example**: Data left in ephemeral storage after the execution of a serverless function contains sensitive information that can be accessed by an attacker.
  - **Mitigation**: Avoid storing sensitive data in ephemeral storage. Use secure and persistent storage solutions for data that needs to persist beyond the function execution.

- **Compliance and regulations:**
  - **Example**: A serverless application that processes personal data fails to comply with data protection regulations, leading to legal consequences and financial penalties.
  - **Mitigation**: Stay informed about relevant regulations. Implement security controls and practices that align with compliance requirements. Regularly conduct compliance audits.

- **Error handling:**
  - **Example**: Poorly implemented error handling in a serverless application exposes detailed error messages that reveal sensitive information, aiding attackers in their exploits.
  - **Mitigation**: Implement generic error messages and avoid exposing sensitive information. Log detailed errors internally for troubleshooting while presenting user-friendly messages to clients.

- **Dependency confusion:**
  - **Example**: An attacker publishes a malicious package with the same name as a widely used dependency, tricking the serverless environment into installing and using the malicious code.
  - **Mitigation**: Use package managers that support dependency pinning. Implement package signing and verification. Regularly monitor for security updates and vulnerabilities in dependencies.
- **Lifecycle management:**
  - **Example**: Outdated and unpatched serverless functions become vulnerable to known exploits, and an attacker leverages these vulnerabilities to compromise the application. Regular updates and patching could have prevented the attack.
  - **Mitigation**: Implement a robust lifecycle management process. Regularly update and patch serverless functions. Retire and decommission unused functions. Conduct regular security audits and vulnerability assessments.

Conclusion

This chapter is instrumental in helping you understand serverless functions and the security requirements for developing, deploying, and managing them.

In this chapter, you have learned to create a function in GCP, test it, and even identify security issues and fix them, such as the one with sanitizing input coming from the request payload.

You should now have a firm grasp on when to use serverless workloads, specifically serverless functions and how to secure them. The next step to securing functions is to go from securing one to securing them *en masse* and automate this.

In the next chapter, we will start focusing on networking and storage and how to secure them in the cloud. We will especially focus on the common patterns and practices as networking and storage services are widely used together in all kinds of different workloads.

Questions

1. What are serverless functions?
2. What are the benefits of a serverless approach as opposed to the traditional workloads?
3. Why is memory and CPU allocation important for functions?
4. What are the benefits of independent functions delivering services?
5. How would you secure dependencies within functions?

6. What is a cold start?
7. How would you manage secrets required for functions at runtime?
8. What is code injection?
9. How would you mitigate resource enumeration?

References

1. AWS Serverless services: **https://aws.amazon.com/serverless/**
2. Azure Serverless services: **https://azure.microsoft.com/en-us/solutions/serverless**
3. GCP Serverless services: **https://cloud.google.com/serverless?hl=en**

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 12
Networks and Storage

Introduction

Everything in the cloud is on a network, explicit or implicit. Specifically created by you or your organization, or implicitly created for you when you provision a service.

You can plan and secure them yourself or you can inherit the (usually secure) defaults. Maintaining network security is one of the priorities of running workloads in the cloud.

In the cloud, we use software-defined networking. In essence, we cannot manage physical networks, so software-defined ones are the only alternative.

On the contrary, storage accounts are also very common, and are used to store data. Accidentally opened storage accounts are very common and happen across all levels of organization and across all sizes of organizations. But it is such a well-known issue that you must be aware of this.

In the cloud, not all storage is created the same, so we will also discuss the nuances of selecting the correct resting space.

When one mentions storage, usually the mind immediately goes to files, but there are all kinds of other places to store data such as databases, persisted cache, queues, etcetera. We will go into these as well along with how to manage their security and when to pick which storage over another.

Structure

The chapter covers the following topics:

- Virtual network concepts
- Virtual network best practices
- Planning a network
- Storage concepts
- Redundancy and availability
- Short term storage options
- Caching

Objectives

In this chapter, we will learn about securing cloud networks and cloud storage. At the end of this chapter, you should be able to pick the correct networking and storage solution for your workflows and secure them.

Virtual network concepts

A **virtual network** (**VNET**) refers to a network that is built using software and hardware that emulate the functions of a traditional physical network. Instead of relying solely on physical infrastructure like routers and switches, a VNET utilizes software-based components to create, manage, and operate the network.

This is what we call software-defined networking. The primary reason it exists as a service in the cloud offering portfolio is to ensure different cloud customers are not sharing the network in a way that might expose traffic or data from one customer to the other. The secondary reason is that there must be a way for customers to organize their networks as it suits them without interfering with other customers or being interfered with by other customers. These network services also apportion specific portions of wire traffic to each customer so that no customer crowds or impacts another.

So, software-defined networking is an approach to networking that uses software-based controllers or **application programming interfaces** (**APIs**) to direct traffic on the network and communicate with the underlying hardware infrastructure.

VNETs are highly flexible and scalable, allowing for efficient resource utilization, easier network management, and the creation of complex network architectures.

Address space

In networking, the term *address space* refers to the range of unique addresses that

communication within the network and across different networks. The **Internet Protocol** (**IP**) is a fundamental protocol for network communication, and an IP address represents a unique numerical label assigned to each device connected to a computer network that uses the IP for communication.

IP addresses are hierarchical and can be categorized into different classes, such as IPv4 (32-bit address space) or IPv6 (128-bit address space). The address space is divided into network and host portions. The network portion identifies the network to which a device belongs, and the host portion identifies the specific device within that network.

Subnets

A subnet, short for *subnetwork*, is a logical subdivision of an IP network. Subnetting involves dividing a larger network into smaller, more manageable subnetworks. This process helps improve network performance, security, and management by grouping devices based on their functions, locations, or other criteria that you choose.

The primary purpose of subnetting is to reduce network congestion, enhance security through isolation, and facilitate efficient use of IP addresses. Each subnet has its own unique range of IP addresses, allowing for better organization and management of network resources.

When a network is subnetted, it introduces the concept of a subnet mask. The subnet mask is a 32-bit number that, when combined with an IP address, defines the network and host portions. Subnetting is particularly important in IPv4 networks, where the available address space can be efficiently used by dividing it into smaller, more manageable subnets.

Subnet masks and the total number of IP addresses a subnet can accommodate are defined as follows:

Mask	IP Addresses	Netmask
/31	2	255.255.255.254
/30	4	255.255.255.252
/29	8	255.255.255.248
/28	16	255.255.255.240
/27	32	255.255.255.224
/26	64	255.255.255.192
/25	128	255.255.255.128
/24	256	255.255.255.0
/23	512	255.255.254.0
/22	1024	255.255.252.0
/21	2048	255.255.248.0
/20	4096	255.255.240.0
/19	8192	255.255.224.0
/18	16384	255.255.192.0
/17	32768	255.255.128.0
/16	65536	255.255.0.0

Figure 12.1

This shows the ways you can subdivide an IPv4-based network. Depending on your use case, you will need to pick and choose sizes of the network subnets for your workloads.

Virtual network best practices

The best practices you will need to follow when designing your networks are discussed in this section.

No overlapping

The addresses should not overlap. Your organization can have a few or many networks. If these networks are required to interoperate now or in the future, the addresses should not overlap.

An example of this is how an existing on-premises network and a new network designed in the cloud should not overlap.

The usual examples of reserved ranges are addresses 192.168.0.0/16, 10.1.0.0/16, 10.2.0.0/16, and so on. Historically, these ranges are/were reserved for just such use.

Plan ahead

Reserving address spaces ahead of time is preferable to recreating the network. Plan your networks and subnet spaces with care and only assign the subnet sizes as needed for the workloads. The networks can and usually are huge, and there is no need for a few of your smaller workloads to take over unnecessarily huge ranges.

Fewer is better

Several larger networks are better than thousands of small ones as they are easier to manage, easier to understand, and easier to manage if they need to communicate with each other. Even then, you have options in the cloud, and you should never run out of addressable spaces, even if that means you might need to create new networks to accommodate.

A few larger networks are better than many smaller ones. This is important to bear in mind, as creating service in the cloud through the portals or consoles usually defaults to creating networks as well.

The following are Azure Portal examples of the same.

Here is how to create a **virtual machine** (**VM**) and set up the network:

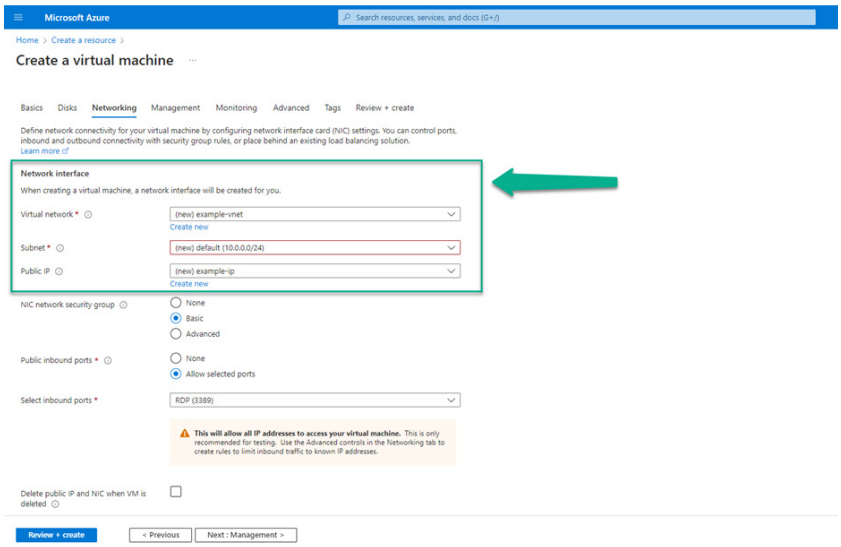


***Figure 12.2**: Create a virtual machine*

- When creating a VM, the default choice is to create a new VNET if there is no pre-existing network (as defined by RBAC controls). For example, if different team members can access their own resource groups, they will default to creating separate networks.
- This will lead to the creation of many unnecessary networks.

Here is how to create a web app and the VNET:

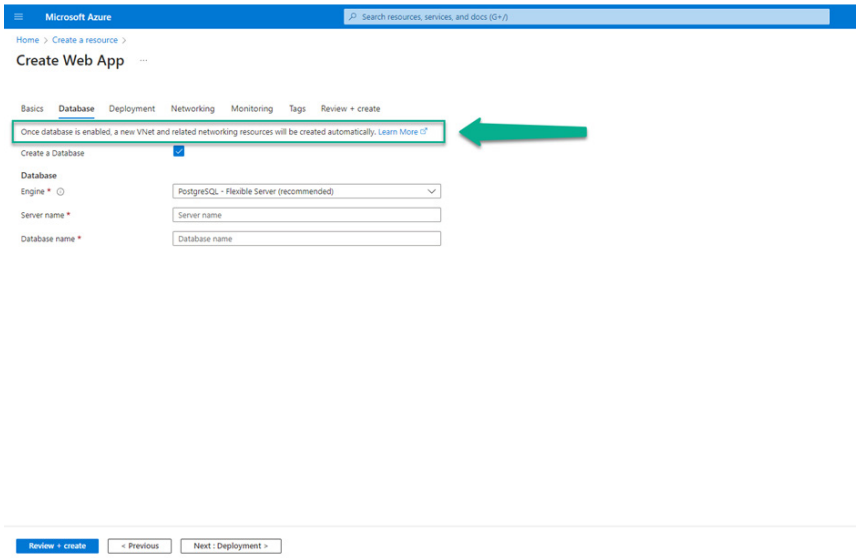


***Figure 12.3**: Create a web app*

- When creating a web app, if you choose to create a DB, the default choice is to create a new network as well.
- A better option for long-term manageability is to create a network first, then a DB, and then a web app, and connect them on the network rather than going with the default choice, which is good for sandbox environments, for example:

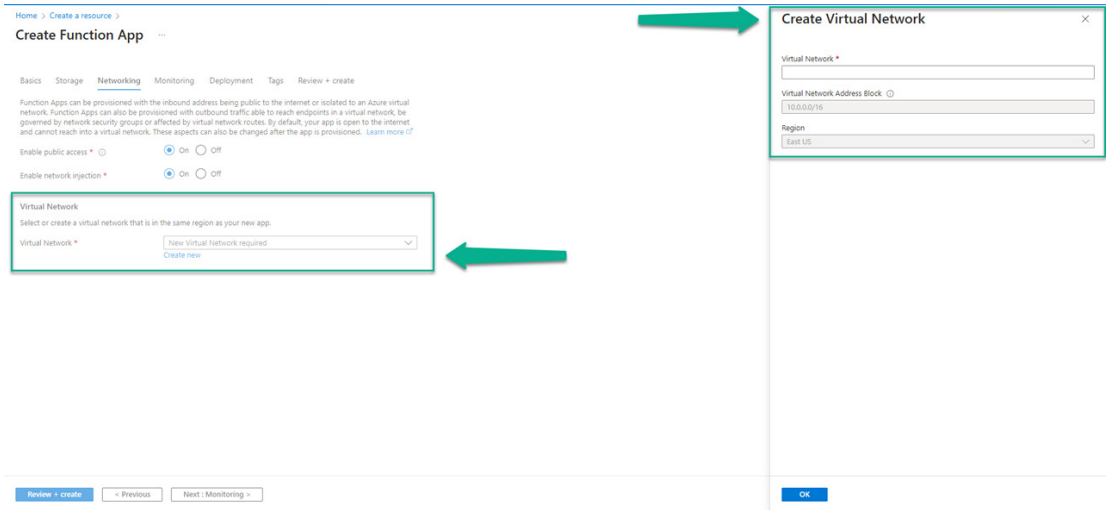


***Figure 12.4**: Create a function app*

- When creating a function app, the default choice is to create a new VNET if no network already exists (as defined by RBAC controls).
- This, again, leads to unnecessary proliferation of networks.

Here is an example on creating a VNET:

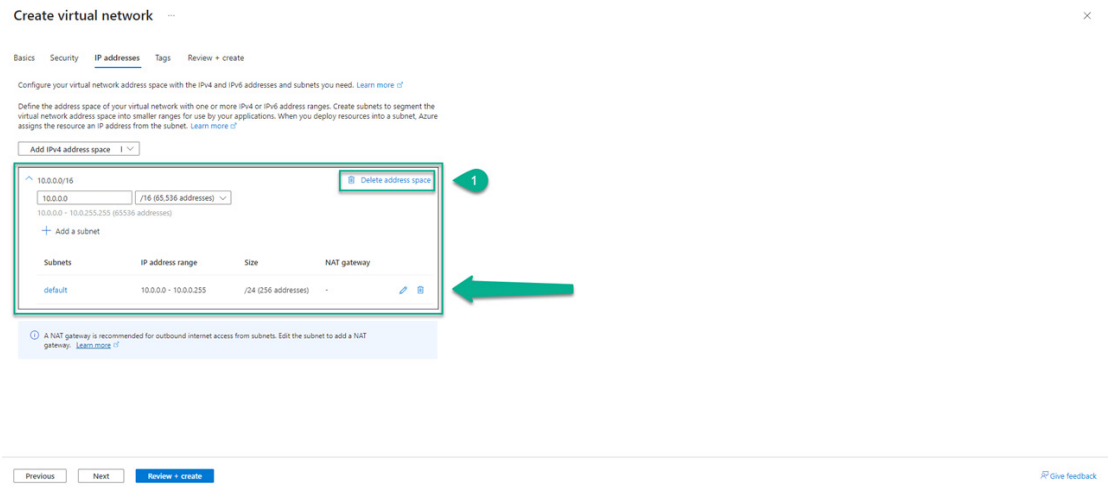


Figure 12.5

- A better, secure, and manageable option is to first create a VNET, following the best practices approach.
- Even with this option, you must be careful, as default choices might not be what you want. For example, the default choice for the first subnet is a /24 (256 addresses) size, which might not be appropriate.
- You can always select the option **Delete address space** and then create the network and the subnets that make sense for your workload.

Ensuring the creation of VNETs is a rare and consistent event and a restricted one at that - is the best approach towards the best practices as described.

Subnet security

Securing traffic from/to subnets is the most manageable way to handle network security. By choosing subnets carefully with a target workload in mind, you can manage network security on the subnet rather than on individual resources. The result is a final, specified number of subnets, as opposed to hundreds or thousands or even hundreds of thousands of different resources on the network with bespoke security access settings.

Artificial intelligence

By following these best practices, you will also take advantage of the artificial intelligence support services within each cloud provider, which will have an easier way of observing your network traffic for you. Through this process, they can alert you if network security patterns change in ways it has not before. For example, if your database subnet never communicated with the internet directly, once it does, you will be alerted, and you can evaluate the reasonableness of the change and whether it was deliberate or coincidental.

Planning a network

To successfully plan your network, consider different scenarios and really understand your workloads and your cloud security needs:

- **Scenarios:** Consider these different common scenarios when attempting to secure your network:
  - Is your workload public-facing?
  - Is it in the same cloud as the bulk of your services, in another cloud, or on-premises?
  - Do all, some, or none of the resources have access to each other and the internet?
- **Understanding:** To really understand your network needs, consider the following:
  -

  - What infrastructure are you supporting with your network, and which storage and computing service will you be using?
  - What are the access patterns with respect to both internally and externally?
  - Do you require connectivity services, or is the workload isolated from other services and locations?

- **Implementation:** To successfully implement your network, you need to build a security containment strategy and define all the tests you will perform to validate it. You will, of course, need to automate and run them consistently and continuously, to nensure not onlycurrent adherence but also future adherence as well.
  - To implement all the requirements, you will need to rely on other teams in your organization, so ensure you have a clear understanding of who does what and who is responsible for testing each assumption you made.
  - As always, ensure network segmentation between workflows and your ingress and egress traffic management.
- **Added services:** If you cannot answer some of these questions, that might indicate that you are not ready to plan a successful networking strategy just yet. However, with your answers to the above questions, we can suggest additional services for you that will help secure the network and network traffic. These are as follows:
  - **Firewall**: For all your traffic management needs, especially on public facing interfaces.
  - **Web application firewall**: This protects your web application from things like SQL injection and other security vulnerabilities.
  - **DDoS protection:** For all attacks, deliberate or accidental, that target your public endpoints with volume-based attacks.
  - **Flow logs**: For monitoring and visualizing your network traffic in real time.

Storage concepts

Let us now focus on the storage. First, let us define some concepts.

For our purposes, storage is not just a 'storage account', which is a way to store files, but is a service that exists with each cloud provider. For our purposes, storage is any way of storing data.

Key questions

Let us consider different ways to store data, based on requirements, through some basic questions:

- Does your workload require a form of storage that provides an interface to read, write, and edit files?
- Do you have a need for easy upload and download of files? How large are these files?
- Are you attempting to store archival data?
- Do you need log storage and/or analytics data?
- Are you doing backups that will need to be restored? Or are you just storing data for rare events when something might need to be read or confirming the existence of a piece of data or origin or metadata information?
- Is your workload only cloud-based? Is it a single cloud or multi-cloud, or does it involve on-premises requirements or capabilities?
- Is this data local to where it is stored, or will it need to be replicated?
- Is data required to be queried? Is it relational data? Does it need to be replicated and/or scaled due to access patterns?
- What control do you need in place for access? Is it operating system-based, database engine-based, or neither?
- Which API do you want to use for access, since many of these services offer either one way of API access or dozens of different APIs are supported?
- Will you need powerful search engine-type capabilities?
- Is the data stored for a really short time (possibly milliseconds) or for a long time (possibly decades)?

Storage capabilities

Based on your answers above, there might be different services you can avail. The services are as follows:

- **Blob storage**: Storing data in blobs for use by other applications. An example is the Excel files, image files or video files that will be downloaded (or uploaded) with regularity.
- **File storage**: When you need to support legacy applications that expect a certain structure of data, location of data, and ways of access, which only file-based storage can support. For example, on VMs, you can extend your disk space with a file storage service. This allows an application local to the VM to access files transparently as if they were on a local disk, but this gives you almost limitless size as opposed to disk sizes.

- **SSD disks:** When you have file access needs that have large volumes or are accessed frequently, and you have performance limits that should not be exceeded. For example, this could store data that is used frequently or that would cause users to be dissatisfied with the access speeds if they had to wait longer than expected.
- **Data lake**: A way for you to store large volumes of data, for example, your organization's business data, that will be used and reused, manipulated, transformed, and queried for use in **machine learning or artificial intelligence** (**ML/AI**) models. You might need to store multiple copies of data in different formats or segment data for access by different people/services for different purposes.
- **Relational DB**: Storing data that is related to each other, which is split into fields and is quarriable by different services in different ways. It could also be gated behind an API so that the data is not accessed directly but still allows various querying patterns.
- **Non-relational DBs**: Storing data that may or may not be relational but in a way that emphasizes and improves scale and performance of ingestion of data, with pre-planned access patterns that accommodate various scenarios. Unlike relational data, which is supposed to be clean and sticks to a preplanned structure, non-relational (better known as no-SQL databases) databases allow ingestion of ad-hoc data or data without a preplanned or known structure.
- **Log analytics services:** Storing analytics and log data so that accessing the queries across billions of records is possible in milliseconds, while usually data is not accessed that often. You can store billions of records per second in these services and query them quickly.

Here is an example of using the Azure Data Explorer to query collected data:

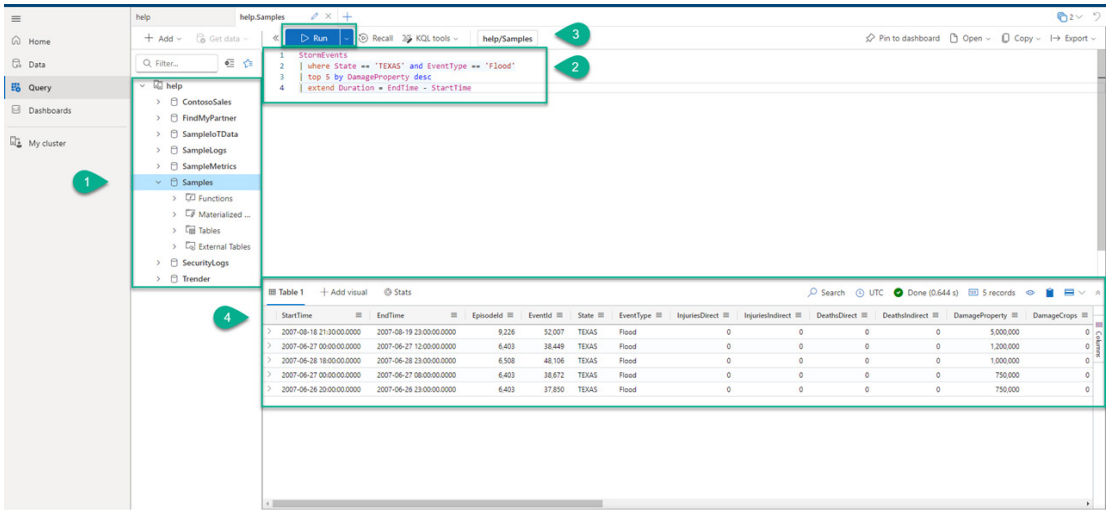


Figure 12.6

- The example above is from a sample existing in the Azure Data Explorer service, which is a service that helps you explore data from any source quickly and easily.
- This process is typical since the following steps are performed:
    1. Data is ingested (not pictured how) and is made available as a 'table.'
    2. You can then formulate your query.
    3. After formulation, you can execute it.
    4. This gives you your results in milliseconds, and you are free to go back and refine your query and execute it again to explore the data set at your disposal.

This is an approach that you should learn well, as many different services have a similar approach. Logs, analytics, data explorers, security events, and many other data sets can be explored by ingesting them into such services.

In the previous example, we were using a **Kusto Query Language** (**KQL**). KQL is used to write queries in Azure Data Explorer, Azure Monitor Log Analytics, Azure Sentinel, and other services in Azure.

Here is the query used in the example:

```
StormEvents
| where State == 'TEXAS' and EventType == 'Flood'
| top 5 by DamageProperty desc
| extend Duration = EndTime - StartTime
```

We address the 'table' **StormEvents** and formulate a query that returns the top 5 records where the input for the state is **Texas,** and the event type is **Flood**. In the last line of the query we also calculate a new field named **Duration** to show that KQL is more powerful than what is shown in the example.

Redundancy and availability

With each of these options to store data from a cloud security perspective, consider different redundancy and availability options.

We can store data locally, within a region, or replicate data across regions if we need to.

In order to protect this data, we need to:

- Secure the storage through role-based access controls.
- Secure data in transit and at rest by encrypting data.
- Delegate access to services that need access to data.
- Analyze access patterns, authentication, authorization and more in order to ensure that these are in line with our workload patterns.

Short term storage options

Sometimes your data storage needs are not long term but very short term (possibly milliseconds). This usually happens when storing events.

This is just like any other data, but it is only required for a little while for communicating events between services or events between customers and services.

An example of service-to-service communication is an inventory service that publishes events on the availability of items to the search service. In this way, the search service does not need to query the inventory service when customers search for the items to buy.

For an example of customer-to-service communication, look at a document generation service. Maybe the customer wants to download all their data from an organization. This takes time, and it may not execute immediately, so a customer makes a request, and an export service responds later once it is done collecting all the data.

There are usually two modalities for messaging events, it is either a producer-consumer pattern or a publisher-subscriber pattern. Their name may differ based on the service, but the general difference is as shown in the following two figures:

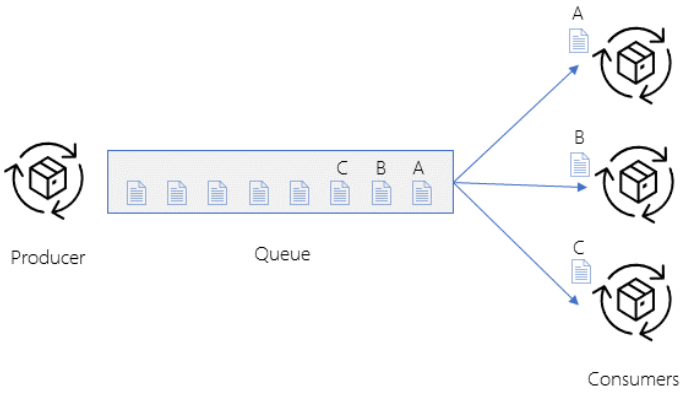


***Figure 12.7**: Producer-Consumer pattern*

- Note that there can be many producers and many consumers. The figure above is a simplified view to show you the difference.
- The producer publishes the events. In this case, the focus is on the A, B, and C events, as shown in *Figure 12.7*.
- For the events to be processed quickly (possible milliseconds), there must be many consumers. They will each grab one message from the queue and process it. Note that we refer to the message location as a queue as they are queued for processing.
- When we have one producer and three consumers depicted, this means that three events can be processed parallelly. This allows us to gather as many consumers

as we want to process events as quickly as possible. It is not uncommon to have hundreds of thousands of consumers if necessary.

- An example would be an ordering system that produces orders, which are then consumed and processed. The more consumers we have, the faster we can process the events. In this case, no consumer sees all the messages because it does not need to see all of them.

Now, let us contrast that with a different pattern:

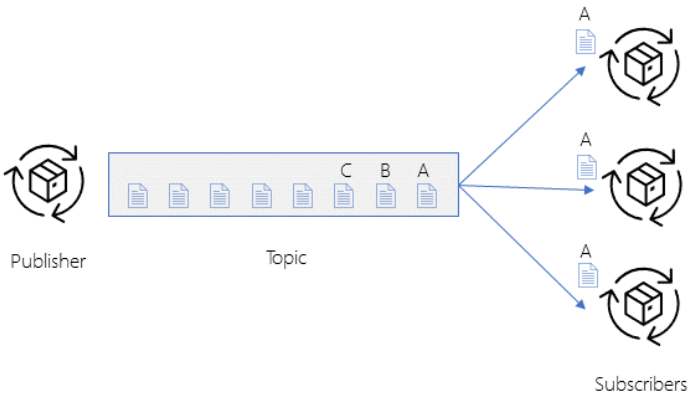


***Figure 12.8**: Example of a table with records that you can query*

- Note that there can be many publishers and many subscribers. The figure above is a simplified view to show you the difference.
- The publisher publishes the events. In this case, the focus is on the A, B, and C events, as shown in *Figure 12.8*.
- Here, the focus is not on the rapid processing of events but on delivering a single event to multiple subscribers. This is why we refer to the event storage as a topic, as we might have multiple topics, and based on these topics, subscribers will decide which topic they want subscriptions to.
- When we have one publisher and three subscribers depicted, this means each event can be delivered in parallel to multiple subscribers. This allows for different services to subscribe to the same event and process it in a way that makes sense for them.
- An example would be an inventory service publishing an event about the return of an item from a customer. The three subscribers would then be a service for monitoring quality control of the item, a refund service that returns the money to the customers, and a warehousing service that notifies a robot to pick up the item in the warehouse location to be repackaged and returned to the seller, or resold. In this case, all three services need to receive the same event to work.

Both the patterns are useful depending on the use case, but they also require slightly different security models.

Other than the general security requirements such as zero trust and network isolation, etcetera, we also need specific security around each of the patterns.

In the producer-consumer pattern, we need security around creating consumers. They need to be created and destroyed depending on the number of messages, and they all need the same security access to the queue that holds the events. If it is a known consumer, no other types of consumers should have access.

The publisher-subscriber is where we are publishing events on topics, and we need a mechanism to handle all existing and future services that n access to the topic.

In our example, suppose there is a new service your organization rolled out that sends a survey to each customer returning the item to get feedback on what the reason for the return was and how happy they were with the return process. The new service would not necessarily have access to the events that happened in the past (unless we saved them elsewhere), but it could come online at any time and react to new events that will come up. In this case, it makes sense to just start from the point of time when it comes online, as we do not want to survey people from years ago about their return process experience.

So, to secure these events, we must ask ourselves the following question:

- How reliable do the events need to be? If we miss some, is it a big deal or not?
- Do we need to store the events elsewhere and secure them, or will the events just live on the queue/topic for a while and then disappear once they are processed?
- Do we want the pull or the push model for the event delivery? Does the consumer/subscriber constantly pull the events, or do they get notified to an endpoint once there is an event for them to process?
- Is the message ordering important or not?
- Is a single message complete, or are several messages related and need to be saved and processed together?
- What happens with malformed events that we do not know how to process? Do we delete them? Do we stop processing any events? Do we save them elsewhere and continue processing others?
- Do we allow ad-hoc topics, or are they defined ahead of time? How do we secure access in these two completely different security requirements?
- What is the throughput? Is 10 million events per second what we normally expect? What do we do when we have many messages? Do we purge the queue/topic? Do we stop processing?

A security concern to be noticed here is related to the cost of the Redis Cache service. As this service is (relatively) inexpensive, it makes sense to create not one instance but as many as necessary and isolate it within each service that needs it. This achieves security segregation. The pattern of creating multiple separate instances which are accessible only to their related service, we can ensure no data is shared/accessible between unrelated services.

Caching

Sometimes, usually for performance reasons, we need to store data that is often requested which rarely changes, in a performance service (usually storing data in memory).

For performance reasons, we store this data in key/value form, meaning we can request a piece of data by referencing just one field, the key. And we get back the value.

Now, we also need to decide whether it is okay if this data is lost. Can we recreate it? Or do we need to store it on the disk as well? These decisions impact our security strategy.

Here is how to ensure security when creating an Azure Redis Cache service, firstly, we create the Redis service:

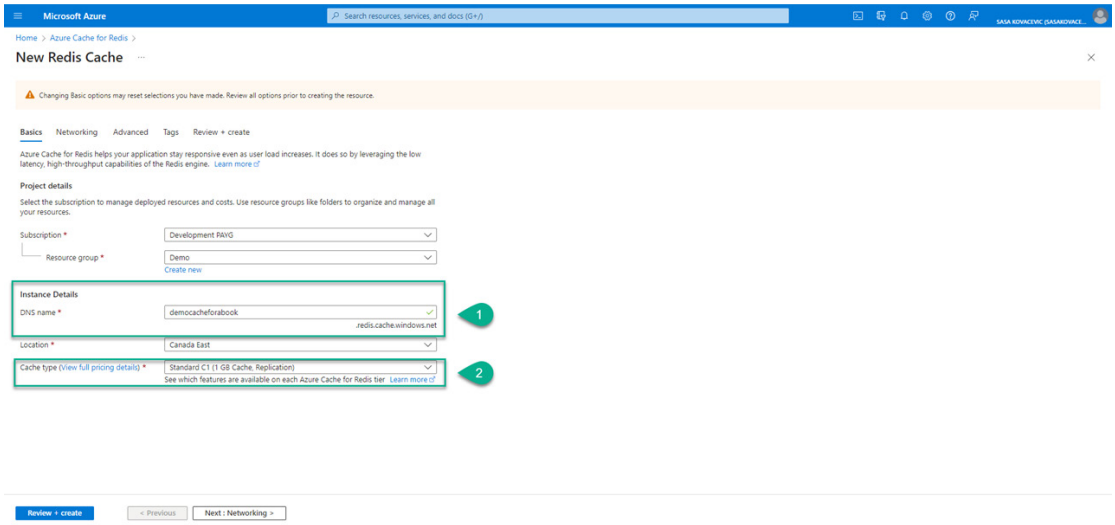


***Figure 12.9**: Redis Cache – Basics*

- Just like in many other services, the DNS name of this service must be globally unique; hence, it requires a DNS name.
- The cache type is important because different types have different security capabilities. Here, we have chosen one of the 'Standard' types. Other standard types differ based on the amount of storage/memory allocated.

Next, we setup the networking:

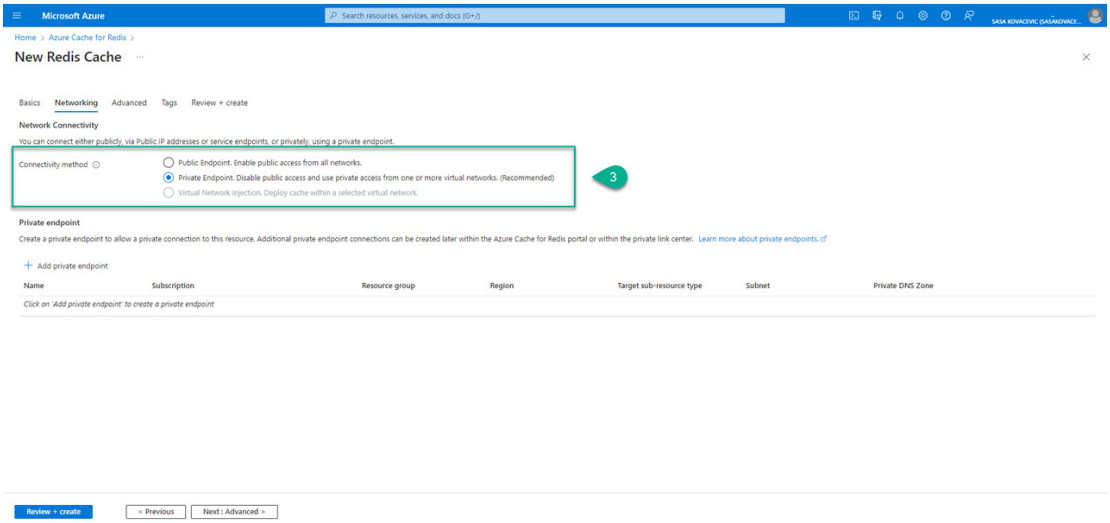


***Figure 12.10**: Redis Cache – Networking*

- Now, for this service, it is very unlikely that it is accessed directly from the internet, so, we choose the private endpoint deployment, which means that this service will only be accessible from within our network.
- As we have created the network previously, we want to use the previous one and set up an endpoint within it rather than creating another VNET just for this resource.

  Use the advanced options to further secure the instance:

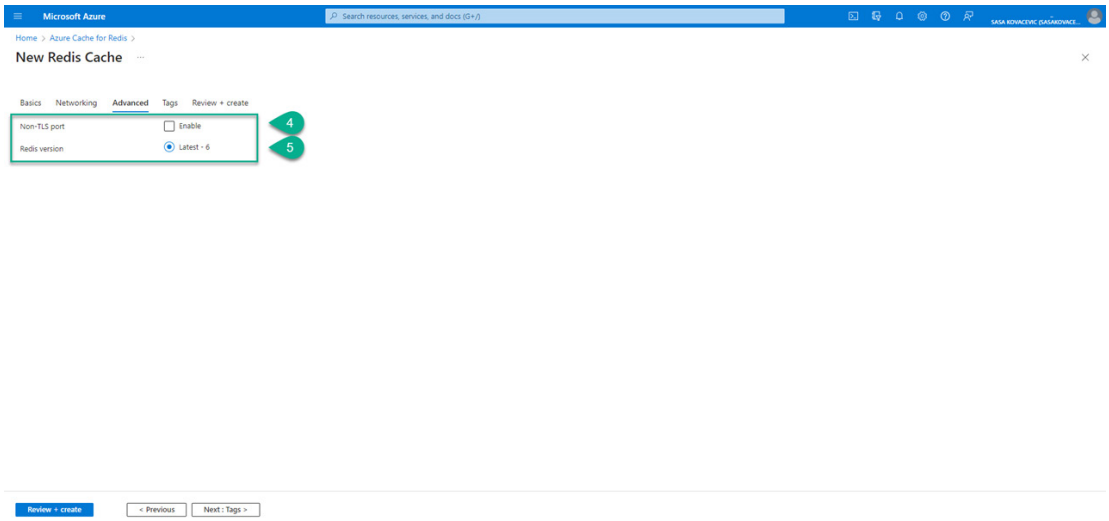


Figure 12.11

- When a sensible default choice is offered, and the **Non-TLS port** option is disabled. You should consider always leaving this option disabled, as we always need the encryption during the transfer.
- We should always choose the latest, stable, and long-term supported version of the service whenever possible. In the example taken, we only have one option, as the version used has been in this state for a while, and the older versions have been phased out.

Conclusion

In this chapter, we learned about networking, specifically software-defined networking, how it works, why it is important, and how to secure your organizations' networks.

We also learned about data, how to store data the right way, and how to tie your data strategy with your networking strategy. You now know how to catalog the data you have, will have, where to store it, and how to secure it.

Event processing and caching were also discussed, and you now know when these services are useful and how to secure them.

Questions

1. What is address space?
2. What does the subnet mask /28 mean?
3. Why should there be no overlapping of address spaces?
4. How would you ensure no overlapping?
5. How do you manage 3000 small networks?
6. How do you start planning your network?
7. What is DDoS protection, and when is it useful?
8. What is the best way to store data in the cloud?
9. What are the different options and capabilities you need to consider when storing data in the cloud?
10. How would you store log data coming from all the infrastructure services in the cloud?
11. How would you secure your log data?
12. What is blob storage?
13.

References

- Azure network best practices:

  https://learn.microsoft.com/en-us/azure/security/fundamentals/network-best-practices

- KQL:

  https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/tutorials/learn-common-operators

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 13

Protecting Workloads through Automation and Threat Intelligence

Introduction

In today's modern cloud-based computing and cloud security, there is something to be said for laziness. Very few people like doing the same thing manually over and over again. This people characteristic works well for us in the cloud and especially in cloud security.

In this chapter, we will be discussing what automation is, why it is beneficial, and some of the processes we need to put in place to ensure automation adoption across the board in our cloud infrastructures.

Automation in workload management is a transformative tool that enhances efficiency and productivity across various industries. One of the key advantages of workload automation is its ability to streamline all the wonderfully useful but repetitive and time-consuming tasks.

Structure

The chapter covers the following topics:

- Automation
- CI/CD

- Threat intelligence
- Threat modeling

Objectives

By the end of this chapter, you will learn how automation not only helps but is essential to cloud operations and cloud security. Without proper **continuous integration and continuous development/deployment** (**CI/CD**), managing the cloud is nearly impossible and ensuring security even less so.

So, you will learn about various tools and patterns when it comes to easily (and without unnecessary complications) planning the security of your and your organizations' services.

Automation

By automating routine processes such as data entry, file transfers, and job scheduling, organizations can significantly reduce the risk of human errors and free up valuable human resources for more strategic and creative endeavors.

Workload automation not only accelerates the pace of operations but also ensures a more consistent and reliable workflow, leading to improved overall performance. Moreover, the adaptability of automation tools allows for the seamless integration of all the wonderful modern technologies and the scaling of operations, making it an essential component for businesses aiming to stay competitive in today's dynamic and rapidly evolving digital landscape.

We live in an era dominated by digital transformation, which means that we are all moving fast and breaking things, as we should, to reach the agility that all businesses need to stay competitive. So, to counteract the impetus of agility, we need to introduce ways to navigate the intricate landscape of cyber threats.

At the heart of modern cloud security lies the transformative power of automation. Automated processes not only enhance the efficiency of security protocols but also enable organizations to proactively respond to emerging threats.

Routine tasks and complex incident response scenarios require an understanding of the foundation of automation.

CI/CD

Continuous integration and continuous delivery, better known as CI/CD, is a way for organizations to tackle the automation requirements in the cloud.

Let us look at specifics and go through some of the patterns that are universally understood to be useful when automating workloads in the cloud and in cloud security.

Continuous integration patterns

Some patterns are so universal that it makes sense to follow them always. Such as the following:

- **Automated builds:** Set up automated build processes to compile, package, and test the application whenever changes are pushed to the version control system, usually defaulted to git nowadays.
- **Version control integration:** Ensure seamless integration with a version control system to trigger CI pipelines on code changes. Automation works on hooks, triggers, or events; one such event is committing code, and on this event, usually, a lot of automation starts.
- **Test automation:** Implement a comprehensive suite of automated tests, including unit tests, integration tests, and end-to-end tests, to validate the integrity of the codebase. This is a great place to also test for known and/or previous vulnerabilities; you should make sure that each vulnerability, once discovered, becomes a test.
- **Code quality analysis:** Integrate static code analysis tools to identify code smells, bugs, and security vulnerabilities early in the development process. The earlier you discover these, the less costly they will be.
- **Parallel execution:** Optimize CI pipelines for parallel execution of tasks to reduce build times and provide faster feedback to developers. It was not uncommon for build tasks to last hours, but today, with the power of the cloud, anything longer than seconds is unacceptable. The cost of running 1 process for 100 hours or running 100 parallel processes for 36 seconds is (nearly) the same. But you must ensure all tests are standalone and can be executed as such.
- **Artifact management:** Use a repository manager to store and manage build artifacts, dependencies, and libraries.
- **Continuous inspection:** Employ tools for continuous inspection of code style, formatting, and adherence to coding standards.
- **Branching strategy:** Define and enforce a branching strategy to manage code changes effectively, ensuring stability in the main branch. The goal is for the main branch to always (at a moment's notice) be ready to go to production.
- **Notification system:** Implement a notification system to alert developers about the status of CI builds, including success, failure, or any issues that need attention.
- **Environment consistency:** Ensure consistency between development, testing, and production environments to minimize deployment-related issues.
- **Containerization:** Consider using containerization to package applications and their dependencies, ensuring consistency across different environments. Or consider the serverless approach we discussed earlier with functions.

- **Infrastructure as Code (IaC):** Use IaC tools (for example, Terraform and/or Ansible) to automate the provisioning and configuration of infrastructure for testing and deployment.
- **Code review automation:** Integrate tools that facilitate automated code reviews to maintain code quality and adherence to best practices.
- **Incremental deployment:** Implement strategies for incremental deployment to test environments, allowing continuous testing of new features.
- **Continuous feedback:** Collect and analyze metrics from CI processes to provide continuous feedback on build performance, code quality, and test coverage.

Continuous development patterns

Some patterns are so universal that it makes sense to follow them always. Such as the following:

- **Deployment automation:** Automate the deployment process to move validated and approved changes from the CI environment to production with minimal manual intervention. To ensure consistency, do not rebuild the artifacts at every stage independently; rather, progress the existing artifact through the stages.
- **Release orchestration:** Implement a release orchestration process to manage and coordinate the deployment of multiple components and services as part of a release. Better yet, ensure that that services are independently deployable.
- **Feature switches:** Use feature switches or feature flags to enable/disable features dynamically, allowing for easy rollback or phased feature releases.
- **Rollback strategy:** Define and implement a rollback strategy to quickly revert to a previous version in case of deployment issues or unexpected production behavior. In the cloud, how to do this very much depends on the service in question, so documentation of each is your best friend there.
- **Environment promotion:** Establish a systematic approach for promoting code through different environments (development, testing, staging, production) while maintaining consistency.
- **Blue-green deployments:** Consider implementing blue-green deployments to reduce downtime and enable quick rollback by switching between two identical production environments.
- **Canary releases:** Gradually release new features to a subset of users (canary group) to monitor for issues before a full release.
- **Configuration management:** Manage configuration settings separately from code to enable easy changes and rollback of configurations without code changes.

- **Continuous monitoring:** Implement robust monitoring solutions to detect and respond to issues in real-time during and after deployment.
- **Security scanning:** Integrate security scanning tools to identify vulnerabilities in the application and dependencies during the CD pipeline.
- **Documentation updates:** Ensure that documentation is automatically updated as part of the CD process to reflect changes in the application or its infrastructure.
- **Database migrations:** Handle database schema changes and migrations as part of the CD process to maintain consistency between application and database versions.
- **Performance testing:** Include performance testing in the CD pipeline to identify and address performance-related issues before reaching production.
- **User Acceptance Testing (UAT):** Automate UAT processes to ensure that the application meets business requirements before deployment to production.
- **Compliance and auditing:** Implement processes to ensure compliance with regulatory requirements and maintain audit trails for changes made during the CD pipeline.

Azure DevOps

Azure DevOps is (arguably) an industry-leading CI/CD solution, allowing you and your organization to *Plan smarter, collaborate better, and ship faster with a set of modern dev services*.

Let us quickly go through setting up automation at all stages of CI/CD. The steps are as follows:

1. First, to start the process of setting up Azure DevOps, search for DevOps and select the Azure DevOps service:

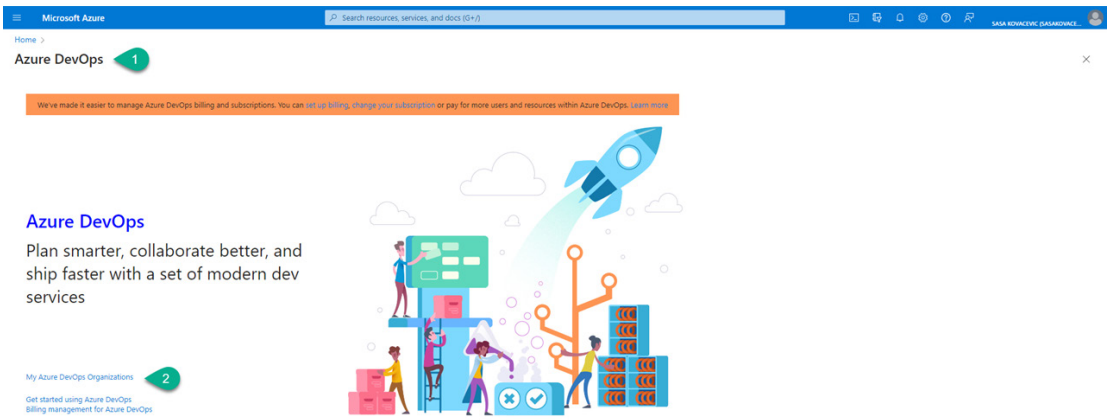


***Figure 13.1**: Azure DevOps*

2. From there, select **My Azure DevOps Organizations**, of which you can be a part of many, but at least one is required for this exercise:

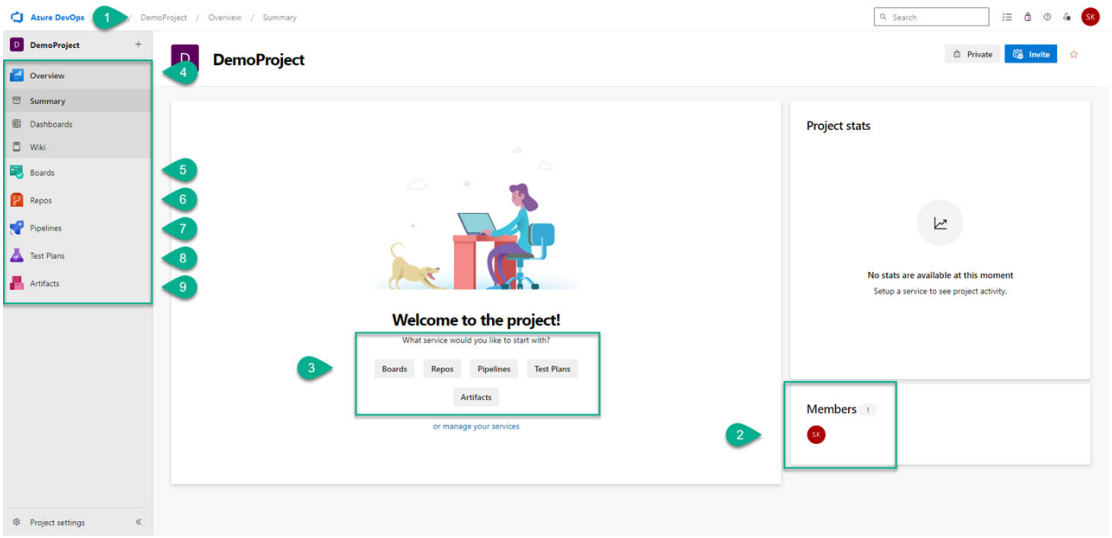


***Figure 13.2**: Azure DevOps – Overview*

3. Once you complete the setup initially (not shown here), you will need to create a project (not shown here). You can complete these steps by following the Azure DevOps process on the screen.

4. Once you have created a project and selected it, you will be presented with the home screen, as shown in *Figure 13.3*.

   i. On it, you can see:

   - Your project is in the hierarchy of your organization.
   - Members of your project, as many as you like or as you need to complete the work. Some projects may have hundreds of members, some very few.
   - Quick menu of services available:
     - **Boards**: Kanban boards to help you plan the work.
     - **Repos**: Repository for your code/files.
     - **Pipelines**: Your CI/CD pipelines help you automate your integration and deployment processes.
     - **Test plans**: Your test plans will ensure your tests are comprehensive, are passing, and are showing continuous improvement.
     - **Artifacts**: Your artifacts, the files ready to be deployed in your various environments.

- Access to this overview page.
- Access to the boards in the project.
- Access to repositories in the project.
- Access to pipelines in the project – both CI and CD.
- Access to test plans in the project.
- Access to artifacts in the project means you can manage the retention of various artifacts in order not to accumulate gigabytes of them.

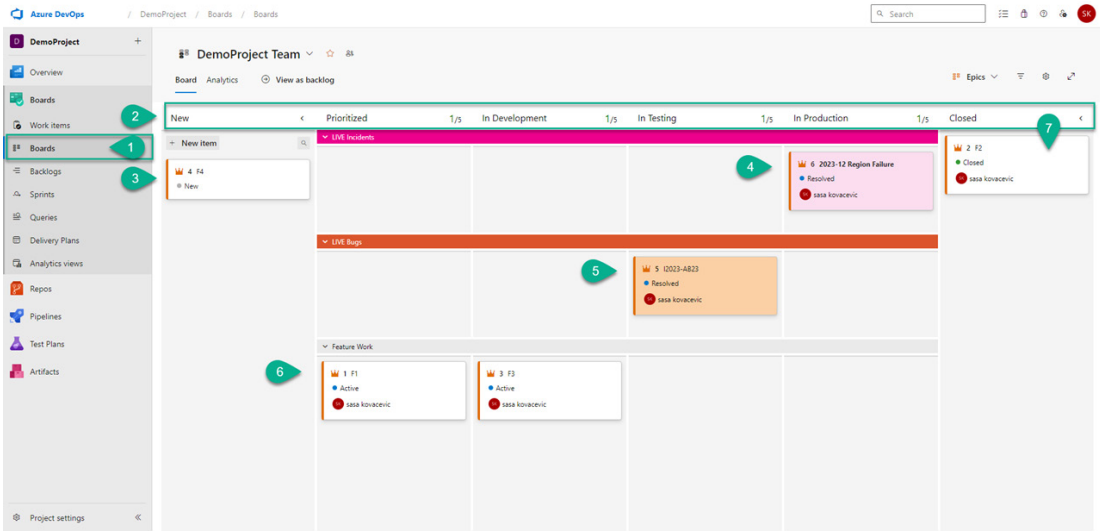


***Figure 13.3**: Azure DevOps – Boards*

5. By accessing the project Kanban boards, you will be able to create, track, and close your work items, as shown in *Figure 13.4*.

   i. You can customize this view to your and your team's liking.

      - Here, we have shown a typical (simplified) setup. The columns have been extended so we have **New**, **Prioritized**, **In Development**, **In Testing**, **In Production**, and **Closed** items. All names are self-explanatory, except maybe – **In Production**.
      - In this team, we have revised the definition of done, that is, when an item of work is completed, to mean that we have completed development and testing, we have deployed the code to production, and we are receiving telemetry on performance and usage and are happy with it. Only then can we claim the work is done, and can we close the item.

- We have also split the lanes (called swimlanes) into three, from the bottom up: the general feature work, LIVE Bugs, and LIVE Incidents. The top swimlanes have priority as they pertain to LIVE PRODUCTION environments, as you can imagine, live site incidents have priority ahead of live site bugs and feature work.
- We also have a closed column that shows, at a glance, what the team has been working on, what was completed, and what can be demoed to the broader organization or customers.

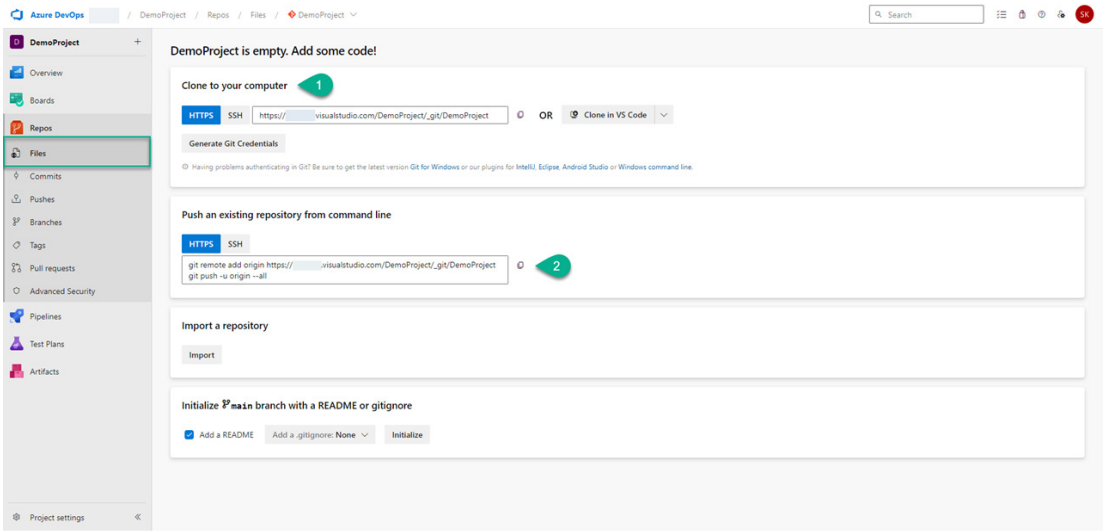


***Figure 13.4**: Azure DevOps – Repos*

6. Repositories allow us to store files.
   i. There are a few ways to initialize the repository, which, if you are familiar with git, just means cloning and/or pushing.
   ii. Here, we also track all changes to files in granular detail so we can review the changes whenever new code is pushed to the repo.

The following figure shows the repository screen:

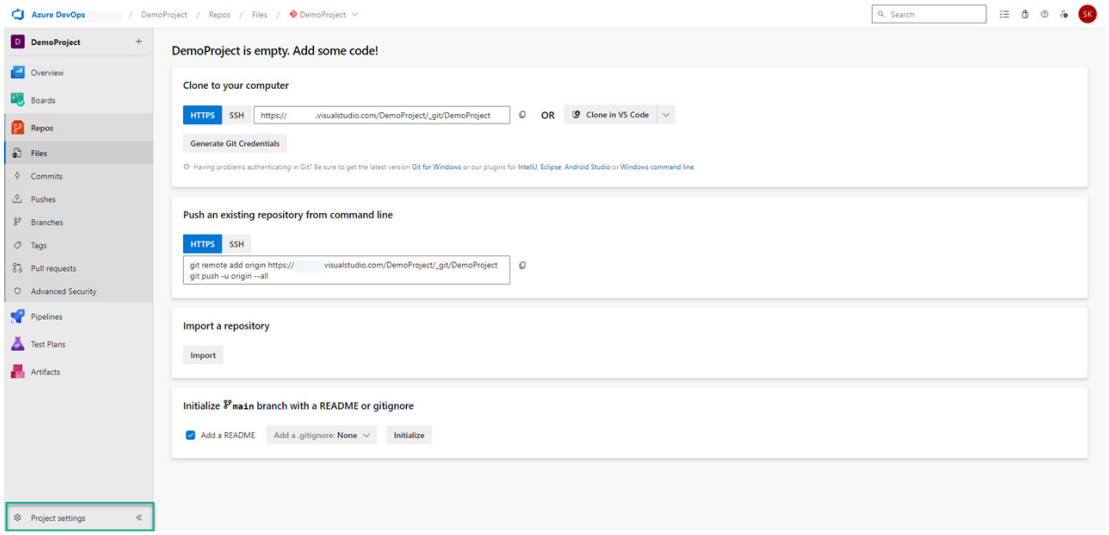


***Figure 13.5**: Azure DevOps – Project Setting*

iii. Now, let us change track a bit and select **Project Settings** from the bottom of the page:

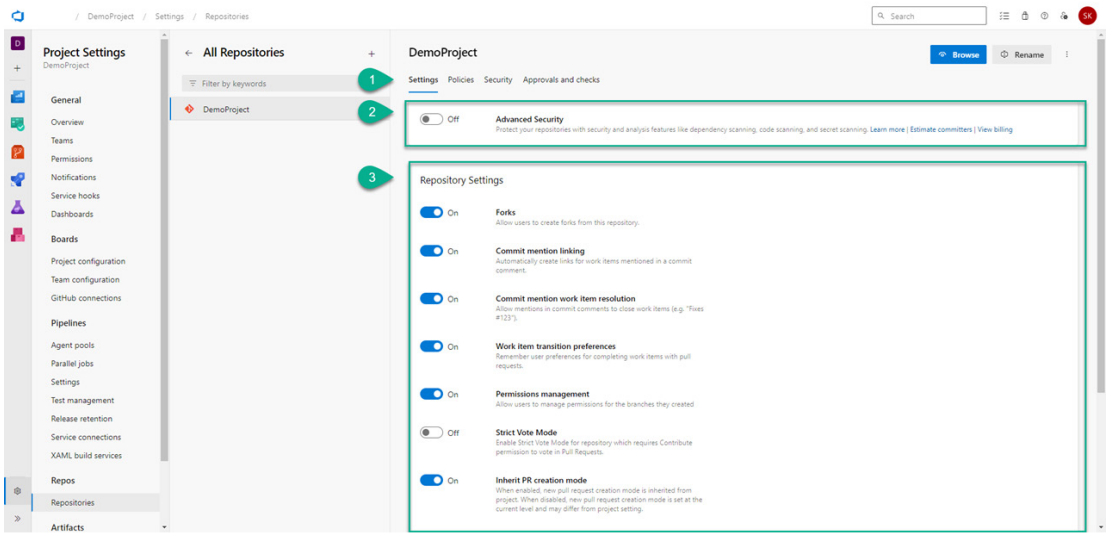


***Figure 13.6**: Azure DevOps –Repository Settings*

7. Under the **Repository Settings,** we can adjust many of the code repository settings, and you should adjust these to suit your team(s) needs:

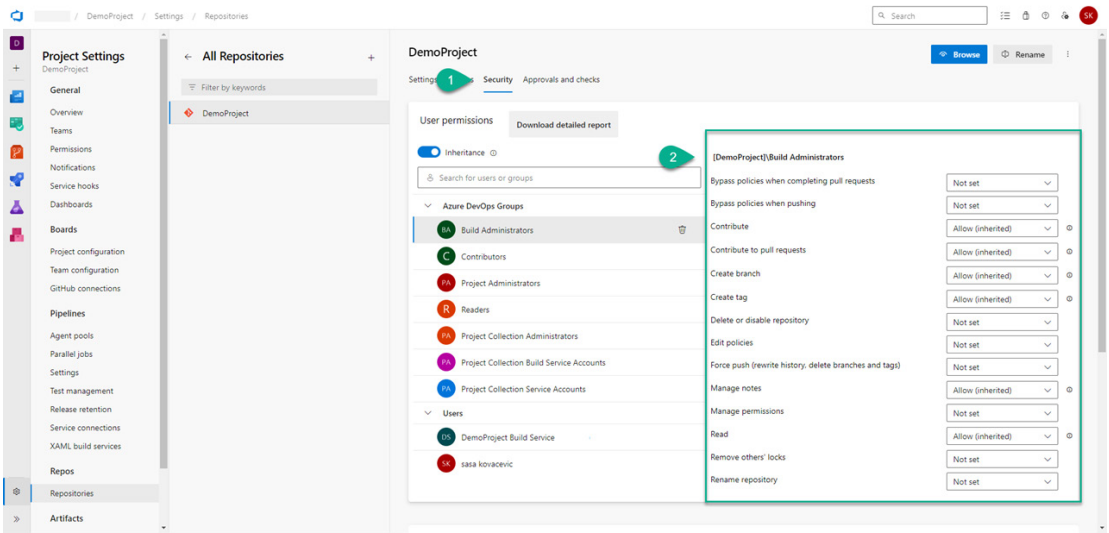


***Figure 13.7**: Azure DevOps – Security Settings*

8. Under the security settings, we need to set all the permissions for team roles, everything from who has access to who can manage access for others – all in very fine granularity.

9. Security-wise, it is important to understand these options and prevent overly broad permissions, as with all other cloud services:

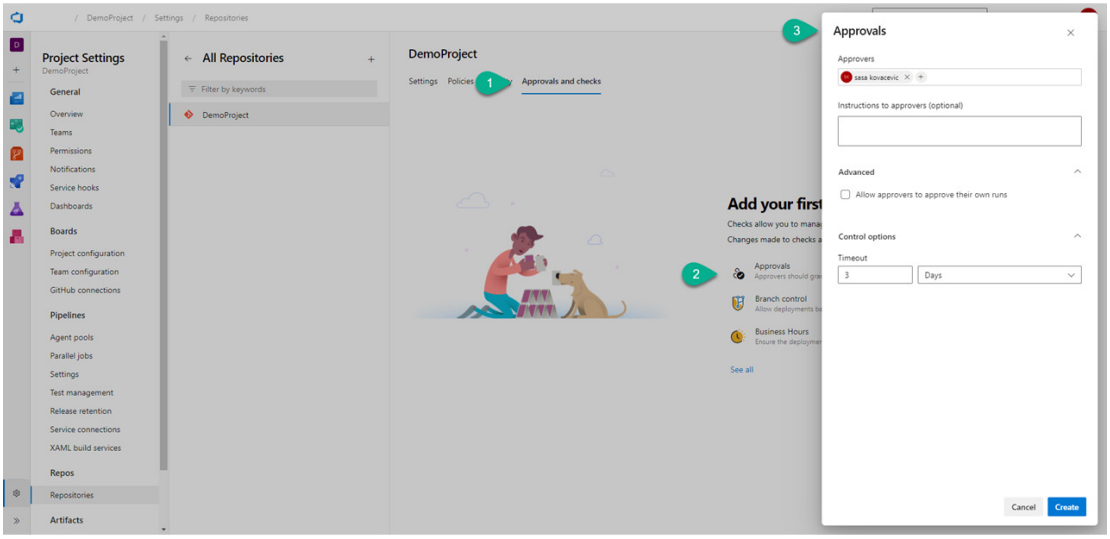


Figure 13.8:

10. In the next step, we can set up all the approvals we need by selecting **Approvals and checks**. Usually, we want team members to approve each other's work, which leads to improved security and better learning about each other's coding practices.
11. Also, we can define who can run the pipelines and when and after which approvals.
12. Some teams like strict controls; some like everyone in the team to be empowered, so security needs to align with these requirements and vice versa:

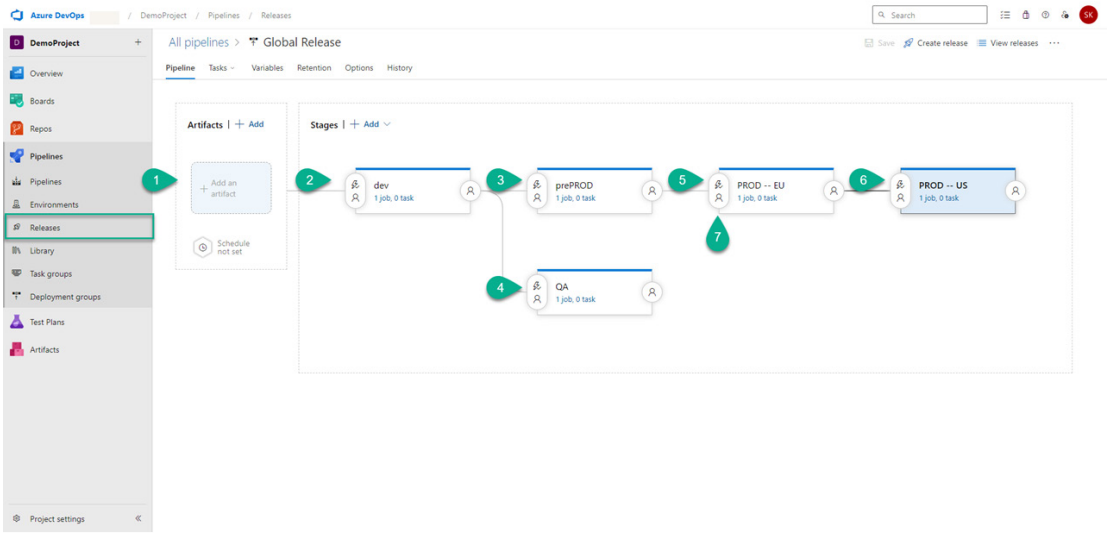


***Figure 13.9**: Azure DevOps – Pipelines*

1. Under the **Pipelines** and **Releases** options, we can set up our deployment pipelines.
    i. Here is a typical setup:

- On the left, the pipeline is triggered by the existence of a specific artifact. Setting this up allows the pipeline to automatically start whenever a new artifact is detected, which is usually what we want.
- Artifact is built in the CI process, and then we take it through our environments.
- First, we go to the development environment so that any debugging can be fast and easy and access to it, automated.
- Then, we take the same artifact forward to our pre-production environment to check integrations with other services while at the same time deploying it to a **quality assurance** (**QA**) environment for a comprehensive suite of tests, which may also include manual tests for some very important exploratory testing.

- Once the dev, prePROD, and QA environments have all shown good performance and test results, we may be ready to progress with this code to production.
- As we mentioned, canary releases are our friends, so we deploy to an environment—in this case, customers in Europe. We enable the feature switches for one customer, then more, then all customers. Telemetry is checked at every stage. If we are happy, we can then continue the rollout to the rest of our global environments.
- From a cloud security perspective, these stages also involve comprehensive security checks, penetration testing scenarios, and chaos testing, which aim to find issues as early as possible and fix them before they are released globally.

Threat intelligence

Automation plays a crucial role in bolstering cybersecurity efforts through threat intelligence. In cybersecurity, timely and accurate information is paramount for identifying, mitigating, and responding to all existing, real, and potential threats.

Automation in threat intelligence enables the rapid collection, analysis, and dissemination of relevant data, empowering security teams to stay ahead of cyber adversaries. Automated tools can continuously monitor vast amounts of data from diverse sources, identify patterns indicative of malicious activities, and provide real-time alerts. This proactive approach not only reduces response times but also allows organizations to fortify their defenses by automating the implementation of security measures in response to emerging threats.

Integration of automation in threat intelligence not only strengthens the overall security posture but also enables a more agile and adaptive defense strategy in the face of an ever-evolving threat landscape.

In the ever-evolving dance between defenders and adversaries, knowledge is power. Intricacies of gathering, analyzing, and using information must be understood to bolster proactive defense mechanisms. The significance of understanding the threat landscape - the types of intelligence available and how organizations can harness this knowledge to stay ahead in the perpetual game of cat and mouse with cyber threats, cannot be overstated.

Rapid identification and mitigation of threats form the backbone of effective workload security. From anomaly detection to automated incident response, you and your organization must navigate the landscape of cutting-edge technologies that enable real-time threat mitigation and minimize the potential impact of security incidents.

When selecting the right tool or tool set for threat intelligence, it is best to consult authoritative sources, such as Gartner and others. Analyzing these tools is not easy, and it takes a while to define requirements and find the best tool for you and your organization,

Threat modeling

We already discussed threat modeling in *Chapter 7*, *Cloud Security and Governance*, so let us try some software to assist in threat modeling. You can apply it regularly alongside your CI/CD processes as you do with documentation.

Check out the Microsoft thread modeling tool as depicted in the following figure:

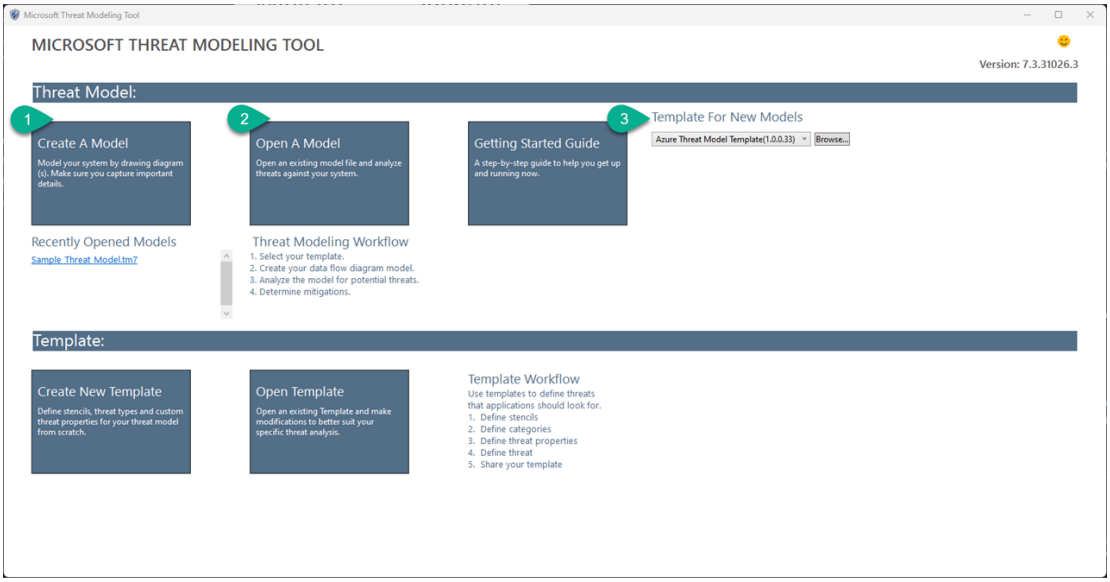


***Figure 13.10**: Microsoft Threat Modeling Tool*

On the home page of the Microsoft threat modeling tool (the link to the download is in the references at the end of this chapter), you can:

- Create a model,
- Open a model you have been working on or a model someone else created,
- See which templates exist for you to use – these are quick starts for you and your organization.

So, who will be creating these models? The best-case scenario is that each team will do so themselves. A somewhat worse case is that the security team, with collaboration and input from each team, will create the models. The worst-case scenario is that no one does so.

The importance of a good enough model cannot be overstated. Any model is better than no model. Here, perfect is the enemy of good. Teams should start with any model that will improve over time rather than skip this step altogether. The value is not in the model created but in the insight gleaned by the team and the wider organization during the creation and revision steps. This means that the process is important as it uncovers the

issues and promotes discussion; the models themselves are good as permanent documents but will rarely make sense to anyone outside of the discussion, so if new folks are required to understand them, a new discussion might be necessary.

Check out the example diagram in the following figure:

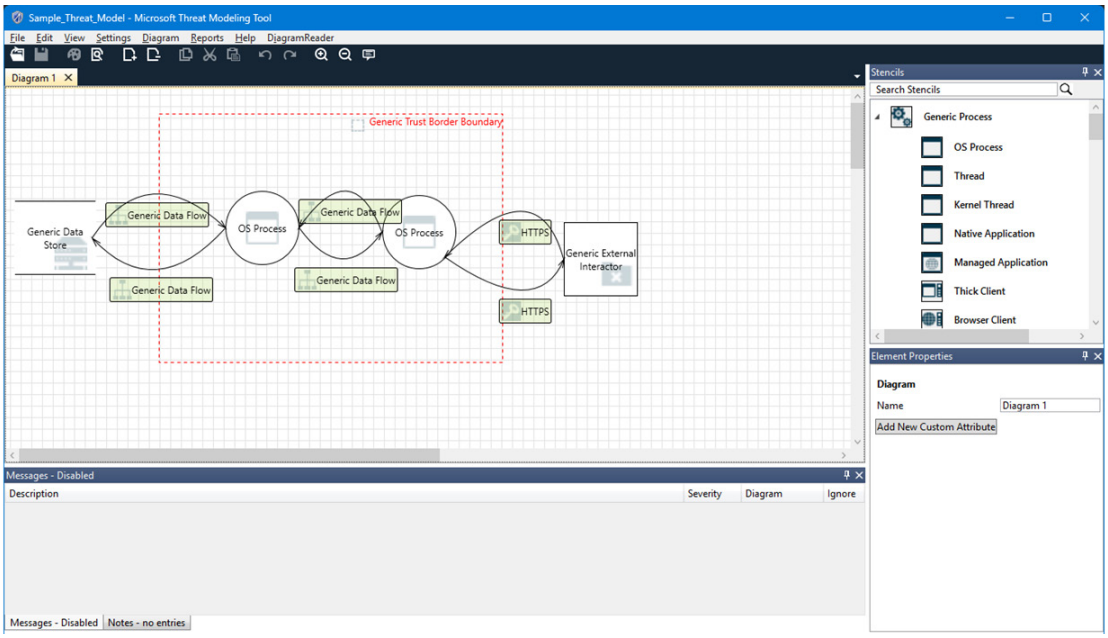


Figure 13.11: *Microsoft Threat Modeling Tool – Example diagram*

Here, you can see an example of a very basic threat model, with components dragged and dropped on the canvas. You can see trust boundaries and data flows, including which protocols are being used.

You can expand the template capabilities yourself and create reusable components in your own template.

Look at the following figure for a template example and adjust the properties as they make sense to you and your organization:

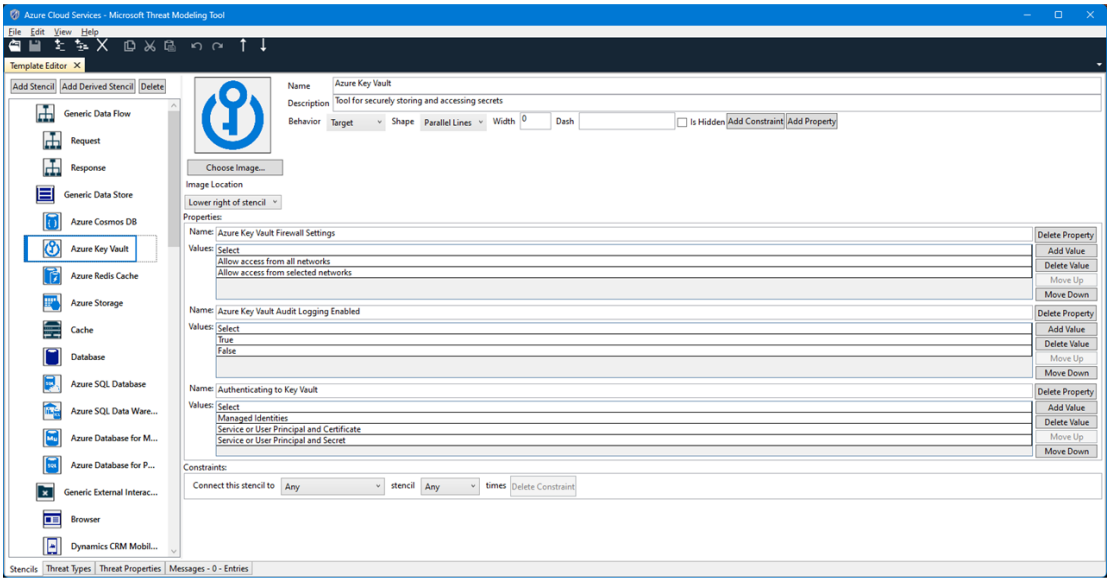


***Figure 13.12**: Microsoft Threat Modeling Tool – Example template*

If you open the Azure Cloud Services model template, you can see how these components are generally set up.

In the example of the key vault component, you can see different options and settings. Once you either standardize this template or build it, users throughout your organization can use it, and together, you can iteratively improve the template.

Conclusion

In this chapter, the overarching theme revolves around fortifying digital workloads within cloud environments, addressing the unique challenges presented by their dynamic and distributed nature. The chapter explores the pivotal role of automation in cloud security and hopefully teaches you all about the best patterns and practices.

Automated processes not only enhance the efficiency of routine security tasks but also provide a scalable and adaptive approach to the ever-evolving threat landscape within the cloud. By implementing automated security measures, organizations can ensure continuous monitoring, detection, and response to potential threats in real-time, minimizing the risk of vulnerabilities.

From automated security testing to vulnerability scanning, the incorporation of security into CI/CD pipelines enables organizations to identify and address issues early in the development cycle, reducing the attack surface, enhancing the overall security posture of cloud workloads, and minimizing cost.

By leveraging threat intelligence, organizations can stay ahead of potential threats by proactively gathering, analyzing, and applying relevant insights to their security measures. Simultaneously, the inclusion of threat modeling ensures a proactive approach to identifying and mitigating potential risks within the cloud architecture.

In the next chapter, we will go in-depth on the topics of incident response and forensics, as well as security assessment and penetration testing. Discussion of all these will benefit from this chapter's threat modeling work.

Questions

Answer the following questions:

1. What is automation, and why is it important for cloud workloads?
2. What does the CI stand for in CI/CD?
3. What does the CD stand for in CI/CD?
4. Why is parallel execution of tests important from a developer's perspective?
5. Why is environmental consistency important?
6. What is infrastructure as a code?
7. How and when would you use feature switches?
8. What is a canary release?
9. How do you track work in progress in Azure DevOps?
10. How would you create a pipeline to release a new global service, and how would it change for an existing one?
11. What is threat intelligence?
12. What is threat modeling?

References

1. Git: **https://en.wikipedia.org/wiki/Git**
2. Azure DevOps: **https://azure.microsoft.com/en-us/products/devops**
3. Threat Modeling Tool: **https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-getting-started**
4. Threat Intelligence: **https://csrc.nist.gov/glossary/term/threat_intelligence**
5. Threat Intelligence Tools: **https://www.gartner.com/reviews/market/security-threat-intelligence-services**

Chapter 14

Incident Response, Forensics, Security Assessment, and Penetration Testing

Introduction

No matter how much work you put into threat modeling and threat intelligence, there will always be security incidents. To prepare for that, we will discuss patterns and practices in this chapter that will help you respond to any incident.

Cloud's main benefit is not cost, is not a vast amount of resources and it is not even security. The main benefit is agility. Doing a lot more in a faster manner. Well, when you do a lot, and you do it quickly, and even if you do it as securely as possible, there will be incidents, so accept it and plan for it. Plan for a breach, plan for your reaction to it, and plan for everything from talking to your CEO before and after the breach to how you will enable law enforcement to assist you.

We have all watched CSI shows on TV or streaming services. While some of the procedures in the TV shows are laughable, actual forensics matters—and your actions post-breach could jeopardize collection.

The first reaction to a breach is almost universally shutting the software/system down. And that can be the very worst option. How will you know what to do? You will read about it in this chapter.

Even if you are good at your work, you are still human, and so, sometimes, you may overlook things. That is why you need a team, an internal one and an external one, to help you.

You are not alone; the world also needs these services. ISO, SOC 2, and other frameworks are here to help, as are other organizations that will do everything in their power to completely break your security.

Some organizations like that are bad actors, and you want to prevent them from succeeding. Well, how else to do that then, by employing good actors to do the same for you? Simultaneously, they tell you what they did and what they discovered, and you fix all the issues, and then you tell them to do it again. And again.

Running a service in production is serious business, and doing it without 24/7 attackers is lunacy. If you do not have your own 24/7 attackers, others will soon come along! Remember those bad actors? Well, they can automate as well as you, and automation never sleeps.

Structure

The chapter covers the following topics:

- Incident response
- Forensics
- Security assessment
- Penetration testing

Objectives

In this chapter, we will discuss the nuances of incident response in cloud environments, which will lead us to a discussion of forensics. This will give you an idea and help you understand how to apply this knowledge in your own and your organizations' environments.

We will also address security assessment, followed by penetration testing. These topics will clue you in and give you some ideas on how to execute the assessment and the testing properly.

Incident response

In the ever-evolving landscape of cybersecurity, incident response is a critical process for any organization leveraging the cloud. It is a structured approach to identifying, containing, eradicating, and recovering from security incidents that threaten your cloud-based data and systems.

Unlike traditional on-premises environments, cloud security incidents pose unique challenges. The shared responsibility model between cloud providers and organizations means that understanding where each party's responsibility lies is crucial. Additionally,

the dynamic nature of cloud deployments and the potential for rapid lateral movement within the cloud infrastructure need a well-defined and adaptable incident response strategy.

To effectively manage security incidents in the cloud, organizations must move beyond reactive measures and adopt a proactive stance. This involves setting up a comprehensive incident response plan that outlines the steps to take upon encountering a security event. This plan should be tailored to the specific cloud platform and services used, clearly defining roles and responsibilities for different teams.

Regular training and testing of the incident response plan are essential to ensure its effectiveness and the team's preparedness. By implementing a robust incident response strategy, organizations can minimize the impact of security incidents, protect their sensitive data, and ensure business continuity in the cloud.

Incident response lifecycle

Let us now focus on the different phases in the incident response process:

- Preparation
- Identification
- Containment
- Eradication
- Recovery

Preparation

The preparation phase forms the foundation of a successful incident response in the cloud. Here, the focus is on being proactive and building the necessary capabilities before an incident strikes.

This involves activities like conducting thorough risk assessments to identify potential vulnerabilities within your cloud deployments. These assessments should consider the specific cloud services used and how they align with your overall security posture.

Next comes the creation of a well-defined incident response plan. This plan outlines the steps to be taken upon encountering a security event, including the roles and responsibilities of different teams (security, IT, legal, etc.), communication protocols, and escalation procedures. It is crucial to tailor the plan to the specific cloud platform and services you use.

Additionally, the preparation phase involves assembling and training an incident response team on the plan and relevant forensic techniques. This team should also be familiar with the cloud environment and the tools used for investigation.

The tools mainly depend on cloud providers, services used, and workloads running in your environment. Take guidance from your cloud provider on these, as they will usually be at least partly covered by the supporting services offered. Of course, also consider the tools loved by your primary asset, your employees.

Due to their inherent complexities, the importance of a robust preparation phase is amplified in cloud environments. The shared responsibility model means both the cloud provider and the organization share security responsibility.

A well-prepared team can quickly identify and contain an incident, minimizing damage and downtime. The clarity provided by the plan ensures a coordinated response, reducing confusion and improving overall incident resolution efficiency.

Identification

The identification phase of incident response involves recognizing the occurrence of a security incident in your cloud environment.

This phase involves continuous monitoring of security logs, network activity, and system health to detect anomalies or suspicious behavior. We will discuss this topic in depth in the next chapter.

Security tools and frameworks like **Security Information and Event Management** (**SIEM**) can be invaluable in finding potential incidents by correlating events and identifying patterns that deviate from normal activity. Reports of suspicious activity or system issues should be investigated and thoroughly documented.

Once a potential incident is identified, the focus shifts towards verification and classification. This involves gathering more information to determine the nature and severity of the incident. Security analysts will analyze logs, examine affected systems, and potentially run forensic tools to understand the scope of the breach and the potential impact on data and operations. This verification process helps prioritize the response and decide on the appropriate course of action.

The identification phase is particularly crucial in cloud environments due to the potential for rapid lateral movement of threats. A fast and accurate identification allows for quicker containment measures, minimizing the damage caused by the incident.

Cloud environments generate vast amounts of data, making it essential to have effective log management and SIEM solutions in place to identify anomalies efficiently. Early identification allows for a more controlled response, minimizing disruption to cloud services and protecting sensitive data.

Containment

The containment phase of incident response in the cloud is all about stopping the bleeding (meaning that you have been wounded by an incident and now the incident is in progress, so it might be a good idea to prevent continuous bleeding).

Once a security incident has been identified and verified, decisive action is crucial to prevent further damage and limit the spread of the threat.

Containment strategies in the cloud environment involve isolating compromised systems or resources. This might involve taking infected virtual machines offline, suspending user accounts suspected of involvement, or restricting network access to specific cloud resources. Actions must be taken to contain the attacker's access by changing passwords, revoking access tokens, or implementing network segmentation to limit lateral movement within the cloud infrastructure.

Containment also involves preserving evidence for forensic investigation. This includes collecting system logs and memory snapshots and potentially taking forensic images of compromised systems. This evidence is crucial for understanding the root cause of the incident, naming attackers (to a degree), and implementing appropriate remediation steps.

The importance of a well-defined containment strategy in the cloud cannot be overstated. Cloud environments are dynamic, allowing threats to spread quickly. Prompt containment measures can minimize the scope of the incident, preventing sensitive data from being accessed or exfiltrated (removed from your organization’s environment).

By containing the attacker's access, organizations can get valuable time to investigate the incident, develop a remediation plan, and restore affected systems.

Eradication

The eradication phase of incident response in the cloud focuses on eliminating the root cause of the security incident and ensuring the threat is completely neutralized.

This involves an approach that tackles both the technical aspects and the underlying vulnerabilities exploited by the attacker. Security teams will strive to remove any malicious software or scripts from compromised systems. This might involve deploying anti-malware tools, disinfecting affected systems, or rebuilding them entirely from clean backups.

Additionally, the focus shifts towards patching vulnerabilities that were exploited in the attack to prevent similar incidents from occurring in the future. This may involve updating software versions, applying security patches to cloud resources, and reviewing security configurations to identify and address any weaknesses.

Eradication also encompasses hardening cloud security posture to prevent future attacks. This involves implementing further security controls, such as multi-factor authentication, network segmentation, and stricter access controls. Security teams will also review and update the incident response plan based on the lessons learned from the incident to improve future preparedness.

Eradication is crucial in cloud environments due to the interconnectedness and potential persistence of threats. A thorough eradication process ensures that the attacker is

completely removed from the system, preventing them from regaining access or launching further attacks.

By addressing vulnerabilities and hardening your security posture, your organization can significantly reduce the risk of similar incidents occurring in the future. This proactive approach strengthens the overall security posture of the cloud environment and minimizes the potential for future breaches.

Recovery

The recovery phase of incident response in the cloud revolves around restoring affected systems and data to a functional state after the threat has been eradicated. This phase prioritizes business continuity and minimizing downtime.

Recovery strategies in the cloud leverage the inherent benefits of cloud technology, such as scalability and elasticity. Organizations can restore compromised systems from backups stored in the cloud. Backup solutions should be regularly tested to ensure their integrity and speedy restore capabilities. Additionally, cloud providers often offer disaster recovery services that can facilitate rapid restoration of entire cloud environments in case of a major incident.

Following restoration, the focus shifts towards validating the recovery process. This involves testing recovered systems and data to ensure they function properly and are not compromised by residual malware. Security teams will also verify that security controls are functioning as intended and no vulnerabilities remain unaddressed.

A well-defined and tested recovery plan is paramount in cloud environments. The ability to quickly restore affected systems and data minimizes downtime and ensures business continuity. Cloud backups offer a significant advantage over traditional on-premises methods, allowing for faster and more efficient recovery.

By validating the recovery process, organizations can ensure they have not simply restored the compromised state along with the data, minimizing the risk of re-infection.

Lessons learned

And finally, let us focus on lessons learned and how we can improve the incident response further.

The lessons learned phase of incident response is a very crucial step for continuous improvement. It involves conducting a post-mortem analysis of the incident to identify areas for improvement in the overall response process. This analysis should involve the incident response team, stakeholders, and potentially external experts. The goal is to understand what went well, what could have been done better, and what needs to be changed to improve future incident response efforts.

The following are some ways to improve upon the lessons learned and do better next time:

- Schedule dedicated lessons-learned meetings. Do not rely on memory; hold dedicated meetings shortly after the incident while details are fresh.
- Encourage open discussion, create a safe space for honest feedback, and avoid assigning blame. Instead, focus on improving the processes that led to the undesired outcomes.
- Document everything. Maintain detailed notes on the incident, response actions, and found issues.
- Identify root causes. Do not just focus on symptoms; look at the underlying vulnerabilities that enabled the incident.
- Categorize findings and group lessons learned into themes (for example, technical controls, communication, and planning) for better organization.
- Translate findings into specific actions with deadlines and ownership assigned.
- Update the incident response plan with the lessons learned to improve future response effectiveness.
- Regularly test your incident response plan and revise it based on lessons learned and ongoing security assessments.
- Share knowledge broadly within your organization and externally in your organization's industry and in the cloud security communities. Disseminate lessons learned across the organization to raise security awareness and improve overall preparedness.
- Measure progress by tracking the implementation of action items and measure the impact of lessons learned on future incidents.

Incident response challenges

Responding to incidents in the cloud presents unique challenges due to the nature of distributed systems, the already discussed shared responsibility model, and concepts such as data residency.

Because of these, cloud forensics differ from traditional forensics, including the need for API access to services (both cloud provider and organizational ones) and the potential limitation on data collection (as some data is intercepted and/or managed by the cloud provider protecting their services).

Incident response plans are crucial for cloud environments. They must be tailored to the specific cloud provider, your environments, your industry and business challenges, and the specific services used, either as cloud services or service libraries.

Collaboration and communication are crucial, especially between different teams like

consistent communication style are very important during the incident response to quickly and easily convey progress without impeding the actual progress of the incident response.

As we have discussed in previous chapters, automation is key. Explore the advantages of automation in cloud incident response for faster detection, containment, and recovery.

Address the importance of considering relevant regulations (for example, GDPR, HIPAA) when handling incidents involving cloud data. We will cover this in more detail in the next chapter.

Forensics

Cloud forensics involves investigating and collecting digital evidence from cloud-based systems and infrastructure. Unlike traditional forensics, it relies on techniques that work within the constraints of the cloud environment and often involve collaboration with cloud providers.

Cloud forensics techniques

The following section provides a breakdown of the three most common cloud forensics techniques (there are many more specific to cloud provider environment, services used, etc., but this is a great start for you to understand them).

Log analysis

Log analysis logs are detailed records of events and activities within a cloud environment. Log analysis involves examining these logs to identify suspicious activity or potential security incidents.

Cloud platforms and services generate various logs, including system logs, application logs, and network access logs. Security analysts use specialized tools to collect and analyze these logs, searching for anomalies, unauthorized access attempts, or other indicators of compromise.

Log analysis offers a non-intrusive approach to identifying potential incidents early on. It can provide valuable insights into user activity, system changes, and resource access patterns.

Memory acquisition

Memory acquisition involves capturing the volatile memory of a compromised cloud instance or virtual machine. This memory snapshot can contain valuable forensic artifacts, such as running processes, loaded malware, and recently accessed data.

Specialized cloud forensics tools can be used to acquire memory from virtual machines in the cloud. Unlike physical RAM, acquiring memory in the cloud often requires cooperation

Memory analysis can reveal evidence that might not be present on persistent storage, such as malware code, attacker footprints, or stolen data.

Virtual machine introspection

Virtual machine introspection is a technique that allows for inspecting the internal state of a virtual machine while it is running. This can be helpful in identifying ongoing malicious activity or hidden processes.

Tools can be deployed within the cloud environment to monitor running VMs. These tools can provide real-time insights into system calls, network activity, and loaded modules, allowing for the detection of ongoing attacks or suspicious behavior.

Virtual machine introspection offers a powerful way to gather forensic data from live VMs without disrupting their operation. It is particularly useful for detecting and investigating advanced threats that operate in memory or hide their presence on the disk.

Evidence collection and preservation

The importance of collecting and, above all, preserving all evidence in the cloud according to legal and chain-of-custody requirements cannot be overstated. Pay very special attention to this and have a legal/law-enforcement expert on hand, as you can very easily run afoul of the lawful requirements on this topic.

Cloud forensics introduces a unique set of legal considerations compared to traditional on-premises forensics. **Electronic discovery** (**eDiscovery**) refers to the process of identifying, collecting, preserving, and producing electronically stored information in response to a legal matter, investigation, or lawsuit.

The distributed nature of cloud environments and the potential involvement of multiple jurisdictions complicate eDiscovery in the cloud. Data might be stored across different regions and even countries, making it challenging to determine which laws and regulations apply.

You should be aware of eDiscovery regulations like the **General Data Protection Regulation** (**GDPR**) in the EU. These regulations dictate how the information must be handled during legal proceedings.

Understanding the cloud provider's role in eDiscovery is crucial. Some providers offer tools and services to assist with eDiscovery, while others might require specific legal procedures to access relevant data.

For you, this means that you should be aware of these requirements and seek advice when these specific services are required.

Furthermore, maintaining a clear chain of custody is essential for evidence to be admissible in court. This involves documenting every step taken when handling evidence, from

collection to analysis. The cloud environment introduces further complexities, as evidence might traverse multiple systems and potentially involve the cloud provider.

The location of data can significantly impact admissibility. Data stored in certain jurisdictions might be subject to local laws that restrict access or limit its use as evidence. Following proper legal procedures for collecting evidence in the cloud is critical. This might involve obtaining warrants or subpoenas to compel the cloud provider to disclose relevant data. Cloud provider terms of service might have clauses regarding data access and legalities. Understanding these terms is crucial to ensure proper handling of evidence.

Proactively prepare for the need for these skills and either outsource them or hire in-house counsel (depending on the size of your organization) before you need them. You can also rely on law enforcement agencies to jump in and help investigate, but they must be contacted promptly and without delay, so even that needs to be planned in advance by you and your organization.

Collaboration with cloud providers

Collaboration with cloud providers is crucial for successful cloud forensics investigations due to several key factors. The key factors are as follows:

- **Data location and control:** Cloud providers have control over the physical infrastructure where your cloud resources reside. They manage the storage systems, network infrastructure, and potentially even backups of your data. Access to this data and logs often requires cooperation from the cloud provider.
- **Specialized tools and APIs:** Many cloud providers offer specialized tools and APIs specifically designed for cloud forensics investigations. These tools can facilitate memory acquisition, log analysis, and virtual machine introspection within the cloud environment, potentially providing deeper insights not readily available through traditional forensic methods.
- **Preserving evidence:** Cloud providers have established procedures for data retention and deletion. Without timely intervention, evidence related to an incident might be overwritten or automatically deleted based on pre-configured policies. Collaboration with the provider ensures evidence preservation while following legal guidelines.
- **Understanding the cloud environment:** Cloud providers possess a deep understanding of their own platform and infrastructure. Their expertise can be invaluable in interpreting logs, identifying anomalies within the cloud environment, and pinpointing the root cause of an incident.

Benefits

The following are some specific ways that cooperation with cloud providers benefits cloud forensics:

- Cloud providers can provide direct access to logs and memory snapshots, saving valuable time during an investigation.
- Leveraging specialized cloud forensics tools offered by providers can lead to a more comprehensive and in-depth analysis of evidence.
- Collaboration with the provider ensures proper handling and documentation of evidence, maintaining the chain of custody for legal purposes.
- Combining the expertise of your internal security team with the knowledge of the cloud provider can lead to a more efficient and effective investigation process.

Data volatility

Data volatility refers to the impermanence of certain types of data in a computer system. In the context of cloud forensics, data volatility presents a significant challenge because it can significantly limit the ability to collect and analyze evidence during an investigation.

The types of volatile data are as follows:

- In memory data
- Temporary files, buffers, and caches
- Logs and telemetry

Data volatility makes it challenging to collect a complete picture of the incident. Forensic investigators might miss crucial evidence that existed only in memory or temporary files at the time of the incident. The ability to collect evidence is often time-sensitive. If a cloud instance is powered down or restarted before forensic measures are taken, volatile data is lost forever. The focus of investigations might shift towards analyzing persistent data sources like storage volumes, which may not provide a comprehensive view of the attacker's activity.

Time is of the essence in cloud forensics. The faster an investigation begins, the higher the chance of capturing volatile data before it disappears. Specialized tools can be used to capture RAM from cloud instances, preserving volatile data for later analysis. Cloud providers might offer tools or APIs for memory acquisition and forensic analysis within their platforms. Implementing robust log management practices with proper retention policies can ensure relevant logs are available for forensic analysis.

Security assessment

A comprehensive cloud security assessment goes beyond basic vulnerability scanning. In this section, we will discuss the two very prominent methodologies used to evaluate the security posture of cloud deployments.

Cloud Controls Matrix

The Cloud Controls Matrix, developed by the **Cloud Security Alliance** (**CSA**), is a comprehensive framework that catalogs a wide range of cloud security controls. It categorizes these controls across different domains, such as security and risk management, incident response, data security, and access control.

A cloud controls matrix assessment involves mapping the security controls implemented by the cloud provider (shared responsibility) and the organization (your responsibility) against the relevant controls outlined in the cloud controls matrix. This mapping helps identify any gaps or inconsistencies in the overall security posture.

A cloud controls matrix assessment provides a standardized approach to evaluate cloud security. It facilitates a clear understanding of control ownership (cloud provider versus organization) and highlights areas where additional security measures are needed.

Shared responsibility model

The shared responsibility model is a fundamental concept in cloud security. It defines the division of security responsibility between the cloud provider and the organization utilizing the cloud services.

A shared responsibility model assessment evaluates how well both parties are adhering to their respective security responsibilities. This involves examining the security controls implemented by the cloud provider and the organization's security practices within the cloud environment.

This assessment ensures that both the cloud provider and the organization are fulfilling their security obligations. It helps identify potential weaknesses arising from a mismatch in security approaches and promotes a collaborative security posture.

By employing various methodologies, your organization can gain a comprehensive understanding of the cloud security posture, identify potential weaknesses, and implement appropriate controls to mitigate risks and ensure a secure cloud environment.

Challenges of vulnerability management in the cloud

While cloud platforms offer inherent security benefits, vulnerability management in the cloud presents unique challenges compared to traditional on-premises environments, such as:

- **Dynamic environments:** Cloud deployments are constantly evolving with frequent scaling, provisioning, and configuration changes. This makes it difficult to maintain a complete and up-to-date inventory of cloud assets, which is crucial for vulnerability scanning and patching.

- **Shared responsibility model:** The shared responsibility model in cloud security means both the cloud provider and the organization share responsibility for patching vulnerabilities. Organizations need to understand which vulnerabilities lie within their responsibility and actively manage them.
- **Limited visibility:** Traditional vulnerability scanners might not have full visibility into cloud environments due to firewalls, access restrictions, or limitations imposed by the cloud provider's APIs. This limited visibility can lead to missed vulnerabilities.
- **Third-party software and libraries:** Cloud deployments often rely on third-party software and services. Patching vulnerabilities in these components becomes complex as the organization might not have direct control over the patching process and may rely on the vendor's update schedule.
- **Compliance requirements:** Organizations must adhere to relevant security regulations and compliance standards. This necessitates maintaining a robust vulnerability management program that can identify and patch vulnerabilities within established timeframes.

Best practices

Let us discuss some best practices to address some of the challenges:

- Maintain a comprehensive and up-to-date inventory of all cloud assets, including virtual machines, containers, storage buckets, and network resources.
- Utilize vulnerability scanners designed specifically for cloud environments. These tools can leverage cloud provider APIs for deeper visibility and overcome the limitations of traditional scanners.
- Implement robust configuration management practices to ensure consistent security configurations across all cloud resources. This reduces the attack surface and minimizes the potential impact of vulnerabilities.
- Automate vulnerability patching processes whenever possible. This reduces the risk of human error and ensures timely patching of critical vulnerabilities.
- Prioritize vulnerabilities based on their severity, potential impact, and exploitability. Focus on patching critical vulnerabilities first to minimize the window of opportunity for attackers.
- Implement continuous vulnerability scanning to identify new vulnerabilities as they emerge. This proactive approach allows for early detection and patching before attackers can exploit them.
- Maintain open communication and collaboration with your cloud provider regarding vulnerability management. Understand their patching processes and leverage their expertise when needed/available.

By addressing these challenges and implementing these best practices, you and your organization can establish a robust cloud vulnerability management program. This program ensures they you can effectively identify, prioritize, and patch vulnerabilities in your cloud deployments, ultimately strengthening your overall cloud security posture.

Penetration testing

Traditional penetration testing methodologies need to be adapted when dealing with cloud environments due to their unique characteristics.

Let us now discuss some prominent frameworks specifically designed for cloud penetration testing.

- The Cloud Penetration Testing Execution Standard, developed by the Cloud Security Alliance, offers a comprehensive methodology for conducting penetration testing in cloud environments. It outlines specific phases tailored to the cloud context, such as planning, reconnaissance, exploitation, and reporting.
- Open Web Application Security Project Cloud Testing Guide builds upon the established OWASP methodology for web application testing and provides additional considerations and techniques for testing cloud-based applications. It emphasizes aspects like shared responsibility, API security, and infrastructure vulnerabilities.
- **National Institute of Standards and Technology** (**NIST**) Cloud Computing Security Risk Assessment Framework provides a high-level approach to assessing security risks in cloud environments. It can be used as a foundation for penetration testing by identifying critical assets, potential threats, and vulnerabilities to target during the testing process.

Importance of threat modeling

Threat modeling is a crucial step before conducting a cloud penetration testing exercise. The reason is as follows:

- Threat modeling helps identify potential attack vectors, vulnerabilities, and weaknesses specific to your cloud environment. This allows you to prioritize testing efforts and focus on the areas that pose the biggest security risks.
- By understanding the potential threats beforehand, penetration testers can optimize their testing strategy and resources, leading to a more efficient and effective testing process.
- Threat modeling helps define realistic attack scenarios that consider the attack surface, attacker motivations, and potential impact. This ensures the penetration testing exercise simulates real-world threats and identifies exploitable weaknesses.

- The threat modeling process fosters communication between security professionals, developers, and other stakeholders. This collaboration leads to a better understanding of the security posture and helps prioritize remediation efforts based on the penetration testing findings.

By incorporating threat modeling before cloud penetration testing, you and your organization can significantly enhance the effectiveness and value of the testing efforts. This ensures a targeted approach, improves resource utilization, and ultimately leads to a more secure cloud environment.

Penetration testing as an assessment tool

Penetration testing can be used to assess both the cloud provider's and the organization's security responsibilities in a cloud environment.

So, let’s go through some of the more important aspects of penetration testing and what you and your organization can do to make it a success.

- Test for vulnerabilities within the cloud provider's infrastructure that your organization relies on. This might involve attempts to exploit weaknesses in storage mechanisms, virtual machine escape techniques, or misconfigurations within the cloud platform itself (focus on shared responsibility).
- Simulate attempts to bypass or escalate IAM controls. This could involve testing for weaknesses in password policies, multi-factor authentication implementations, or access control lists (shared responsibility).
- Focus on testing the security of APIs used to access and manage cloud resources. This might involve identifying vulnerabilities in authentication mechanisms, authorization controls, or data validation within the APIs (shared responsibility).
- Test the strength of data encryption at rest and in transit within the cloud environment. This can involve attempting to steal data from storage buckets, databases, or during network transfers (shared responsibility).
- If your organization utilizes serverless functions within the cloud platform, penetration testing can identify vulnerabilities in these functions or their access controls (the organization's responsibility).
- Cloud environments are complex and prone to misconfigurations. Penetration testing can identify these misconfigurations in security groups, firewalls, or resource access controls that might expose vulnerabilities (shared responsibility).
- Simulate attacks launched by malicious insiders with authorized access. This can involve testing for privilege escalation attempts, unauthorized data exfiltration, or manipulation of cloud resources (the organization's responsibility).
- If your organization integrates third-party applications within the cloud

integration points and potential vulnerabilities arising from these integrations (the organization's responsibility).

- Simulate social engineering attempts to trick employees into granting unauthorized access or revealing sensitive information. This helps assess your organization's security awareness and preparedness against such attacks (organization's responsibility).
- Test the cloud provider's and your organization's defenses against DoS attacks that might target cloud resources or applications. This helps assess the overall resilience of the cloud environment and identify potential bottlenecks (shared responsibility).

Reporting

Following a penetration test, a well-crafted report and a commitment to remediation are crucial for maximizing its value. The report should include the following:

- A comprehensive report should document the entire testing process, including methodology, tools used, identified vulnerabilities, and successful exploits. It should prioritize vulnerabilities based on severity and potential impact, providing clear recommendations for remediation.
- The report should translate technical findings into actionable steps for the security team and developers. This includes timelines for fixing vulnerabilities, assigning ownership for remediation tasks, and outlining the patching or mitigation strategies.
- The report serves as a communication tool, facilitating discussions between security professionals, developers, and management. It fosters collaboration in prioritizing remediation efforts and allocating resources effectively.

Penetration testing reports serve as a historical record, allowing you to track progress over time and measure the effectiveness of your security posture. They can inform future security strategies and guide continuous improvement efforts.

Lessons learned from a penetration test

Penetration tests offer valuable insights into your cloud environment's security posture. The following are potential lessons you might learn:

- Even minor misconfigurations in security groups, firewalls, or access controls can create significant vulnerabilities.
- APIs are a common attack vector. Ensure strong authentication, authorization, and data validation mechanisms are in place.

- Unpatched vulnerabilities are low-hanging fruit for attackers. Prioritize timely patching of critical vulnerabilities.
- Penetration tests can highlight weaknesses in your defenses against insider threats. Invest in security awareness training for employees.
- The test might reveal gaps in data encryption or access controls. Strengthen data security measures to protect sensitive information.
- The test might expose vulnerabilities arising from integrated third-party applications. Evaluate the security posture of these applications.
- A successful attack might emphasize the importance of having robust backup and disaster recovery plans in place.
- Social engineering simulations can highlight the need for ongoing security awareness training for employees.
- The penetration testing process itself improves your organization's incident response preparedness by simulating real-world attacks.
- No environment is perfect. Penetration testing is an ongoing process that helps identify and address security weaknesses over time.

Tools

There are a lot of tools that help with penetration testing. The following are some examples across the providers, vendors, and scopes (security professionals are known to use funny names for their tools, so bear that in mind and be careful when searching for these as they are also often targeted by specific malware):

- **CloudSploit**: This tool focuses on identifying misconfigurations and security weaknesses within cloud platforms like AWS, Azure, GCP, and others. It is ideal for testing shared responsibility aspects and ensuring your cloud environment adheres to security best practices.
- **AWS Inspector**: This cloud-based service offered by Amazon Web Services helps identify potential security vulnerabilities and configuration risks within your AWS infrastructure. Use it for automated vulnerability scanning specifically within the AWS environment.
- **Microsoft Defender for Cloud**: Similar to AWS Inspector, Microsoft Defender for Cloud is a native offering from Microsoft Azure that provides security assessments and vulnerability scanning for Azure resources. Use it for continuous monitoring and vulnerability management within Azure.
- **Pacu**: This open-source tool is designed for penetration testing of AWS environments. It allows for the exploitation of vulnerabilities and the testing of

security controls. Use Pacu for simulating real-world attacks and assessing the effectiveness of your AWS security posture.

- **Nmap**: This versatile open-source network scanning tool can be used in cloud environments to identify live hosts, open ports, and services running on cloud instances. Use Nmap for initial reconnaissance and understanding the network topology of your cloud environment. You may have seen this legendary tool used in the movies Matrix, Bourne Ultimatum, Die Hard 4, and many others, which shows how popular these tools are becoming as they are featured in popular movies and movie franchises.
- **Burp Suite**: This industry-standard web application security testing platform can test the security of cloud-based web applications and APIs. It can identify vulnerabilities like SQL injection, XSS, and insecure APIs within your cloud applications.
- **CloudGoat**: This is a deliberately vulnerable cloud environment designed for practicing and learning cloud penetration testing skills. Use CloudGoat for penetration testing training and controlled experimentation with vulnerabilities in a safe environment.
- **Scout Suite**: This open-source multi-cloud auditing tool allows you to scan and identify security misconfigurations across multiple cloud platforms like AWS, Azure, GCP, Alibaba Cloud, and others. Use Scout Suite for consolidated security assessments across your multi-cloud deployments.

Conclusion

You should now be ready to tackle the important topics in cloud security. Incident response and forensics are important, and now you know why and how to prepare for these important tasks. Security assessment and penetration testing will help you minimize the number of incidents and forensics procedures you will have to undergo by improving your security posture and positioning you to proactively manage security in the cloud.

In the next chapter, we will discuss compliance and auditing topics closely related to this chapter's themes. Compliance will further help you understand your responsibilities in the security lifecycle, and auditing will give you important evidence of events happening in your organization and in your organization’s cloud platform.

Questions

1. Can you describe the phases involved in the cloud incident response lifecycle?
2. What are some of the challenges when responding to security incidents in the cloud environment?

3. What is the difference between forensics in the cloud versus traditional on-premises forensics?
4. Can you succinctly explain data volatility in the cloud and its impact on forensics investigations?
5. Why is collaboration with cloud providers crucial during an investigation?
6. What are the techniques used for cloud forensics analysis?
7. What are some of the key considerations when performing a security assessment of a cloud deployment under the shared responsibility model?
8. How does vulnerability management differ in a cloud environment compared to a traditional on-premises setup?
9. What is penetration testing?
10. How can penetration testing help assess the security of APIs used to access cloud resources?

References

- NIST Incident Response: **https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf** (PDF file)
- Cloud Security Alliance Cloud Incident Response Framework: **https://cloudsecurityalliance.org/artifacts/cloud-incident-response-framework**
- CloudFlare penetration testing introduction: **https://www.cloudflare.com/learning/security/glossary/what-is-penetration-testing/**
- Cloud Security Aaliance Cloud Controls Matrix: **https://cloudsecurityalliance.org/research/cloud-controls-matrix**
- Cloud Security Alliance Cloud Penetration Testing Playbook: **https://cloudsecurityalliance.org/artifacts/cloud-penetration-testing-playbook**
- OWASP Web Security Testing Guide: **https://owasp.org/www-project-web-security-testing-guide/**
- NIST Managing Risk in the Cloud: **https://csrc.nist.gov/pubs/book-section/2016/09/managing-risk-in-the-cloud/final**
- CloudSploit: **https://github.com/aquasecurity/cloudsploit**
- AWS Inspector: **https://aws.amazon.com/inspector/**
- Microsoft Defender for Cloud: **https://learn.microsoft.com/en-us/azure/defender-for-cloud/**

- Pacu: **https://github.com/RhinoSecurityLabs/pacu**
- NMAP: **https://nmap.org/**
- BurpSuite: **https://portswigger.net/burp/communitydownload**
- CloudGoat: **https://github.com/RhinoSecurityLabs/cloudgoat**
- ScoutSuite: **https://github.com/nccgroup/ScoutSuite**

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 15
Compliance and Auditing

Introduction

The wonderful world of regulated industries like banking or insurance, etc., is an additional security burden on you and your organization, but in a good way. Additional scrutiny makes you more secure if you flow with regulations and not against them. Notice we used the words more secure, not absolutely secure. You must do both: follow the regulations and do your personal best for security.

In this chapter, we will discuss two definitions of auditing – auditing as a function of recording all events (security and otherwise) that happen on/in your systems – and auditing as a function of third parties verifying your compliance.

Both are exhaustive topics, so we will give you some universal patterns to handle them.

Structure

The chapter covers the following topics:

- Introduction to compliance
- Frameworks and standards
- Balancing compliance and security
- 3rd party auditing

- Cloud providers compliance portals

Objectives

The primary objective of this chapter is to showcase the concept of auditing from two critical perspectives: the internal recording of all events within your systems and the external verification of compliance by third parties. Understanding these definitions is crucial as they form the backbone of robust security frameworks in regulated industries. By looking at the processes of meticulous event recording and comprehensive third-party audits, this chapter seeks to equip you and your organization with the knowledge to integrate regulatory compliance into your security strategies seamlessly.

Throughout this chapter, we will present universal patterns and best practices to manage both aspects of auditing efficiently. The goal is to provide practical insights that enable you and your organization to not only meet regulatory requirements but also enhance your security capabilities.

By adopting these approaches, you can achieve a higher level of security that, while not absolute (never an absolute), significantly reduces vulnerability exposure. Emphasizing the synergy between regulatory compliance and internal security measures, this chapter aims to foster a proactive and resilient security culture within regulated industries.

Introduction to compliance

Compliance refers to the process by which organizations adhere to established laws, regulations, guidelines, and specifications relevant to their business operations.

In the context of regulated industries, compliance involves meeting the requirements set forth by governmental bodies, industry standards, and internal policies. These requirements are designed to ensure that organizations operate in a legal, ethical, and secure manner. Compliance encompasses a wide range of activities, including data protection, financial reporting, operational procedures, and customer communications, all aimed at maintaining integrity and trust within the industry.

Legal and financial protection

Compliance is crucial for avoiding legal penalties, fines, and sanctions that can arise from non-compliance with laws and regulations. Regulatory bodies enforce penalties on organizations that fail to meet compliance standards, which can lead to substantial financial losses and legal battles. Adhering to compliance requirements protects organizations from these risks, ensuring their operations remain lawful and financially sound.

In recent times, the penalties have been applied more forcefully than in the past, as nation-states are taking more of an active interest in topics like data protection, cloud computing, artificial intelligence, anti-monopolistic regulation, local market protectionism, global and regional market standards, and so on.

Organizations are now under more scrutiny than ever before, and as penalties are applied to more and more organizations more often, the risk to you and your organization becomes greater.

Enhanced security and risk management

Compliance frameworks include robust security measures to protect sensitive data and mitigate risks.

Adhering to these standards, organizations can enhance their security posture, reducing the likelihood of data breaches, cyber-attacks, and other security incidents.

Compliance ensures that organizations implement best security practices, safeguarding their assets and maintaining the trust of customers and stakeholders.

Risk management has always been part of organizations' toolsets, and it is now more crucial than ever, especially in regulated industries. Running afoul of regulation can lead to exclusion from competing in certain markets altogether—EU, US, China, India, and so on. The size of these markets makes this an existential threat to organizations doing business within these markets.

Reputation and trust

Maintaining compliance is essential for building and sustaining a positive reputation in the industry. Customers, partners, and investors are more likely to trust and engage with organizations that demonstrate a commitment to regulatory compliance and ethical practices.

A strong compliance record enhances an organization's credibility and can be a key differentiator in competitive markets. Conversely, non-compliance can result in reputational damage that may take years or decades to repair.

For a current example, just look at the aviation industry and the consequences for some of the largest players after damaging the reputation and trust of the markets, customers, and regulators. It can happen to your organization if you do not navigate this threat carefully and with consideration of any misstep's impact on you and your organization's reputation.

Frameworks and standards

In this section, we will examine some examples of regulatory bodies, common standards and regulations, and industry-specific compliance requirements.

Regulatory bodies

When it comes to cloud computing workloads, some of the common regulatory bodies (in certain markets, you should check if any of these apply to you or if any of these are useful to you) are as follows:

- **International Organization for Standardization** (**ISO**):
    - Provides various standards, including those specific to cloud security (for example, ISO/IEC 27017 for cloud security and ISO/IEC 27018 for privacy).
- **National Institute of Standards and Technology** (**NIST**):
    - Develops standards, guidelines, and best practices, such as the NIST Cloud Computing Standards Roadmap and NIST SP 800-53, for security and privacy controls. NIST SP 800-53 provides a list of controls that support the development of secure and resilient federal information systems. These controls are the operational, technical, and management standards and guidelines information systems use to maintain confidentiality, integrity, and availability.
- **European Data Protection Board** (**EDPB**):
    - Governs data protection laws in Europe, ensuring compliance with the **General Data Protection Regulation** (**GDPR**). The GDPR's goals are to enhance individuals' control and rights over their personal information and to simplify the regulations for international business.
- **Cloud Security Alliance** (**CSA**):
    - Offers the CSA **Security, Trust & Assurance Registry** (**STAR**) program and the **Cloud Controls Matrix** (**CCM**).

As you can imagine, there are many more of these types of organizations, so you should research which applies to you and your organization in your markets, industries, nation-states, and supranational organizations (such as the EU). If you and your organization are doing business globally, successfully navigating these organizations and their requirements, assessments, and assistance will be crucial.

Common standards and regulations

Let us now look forward to the common standards and regulations you might encounter:

- **ISO/IEC 27001**: Standard for **information security management systems** (**ISMS**).
- **ISO/IEC 27017**: Guidelines for information security controls applicable to cloud services.
- **ISO/IEC 27018:** Guidelines for protecting personal data in the cloud.
- **NIST SP 800-53**: Security and privacy controls for federal information systems and organizations.
- **NIST SP 500-291**: NIST Cloud Computing Standards Roadmap.
- **CSA CCM**: Control framework for cloud security, mapped to multiple other frameworks and standards.

- **SOC 1, SOC 2, SOC 3**: Service Organization Control reports for the different types of audits and assurance on service organizations' controls.
- **Payment Card Industry Data Security Standard** (**PCI DSS**): Standard for organizations that handle branded credit cards from the major card schemes.

Understanding and researching these standards in depth (and others that apply to your and your organization's markets and industries) will be necessary for you and your organization to comply, prove compliance, and publish reports on how exactly you are complying.

Industry-specific compliance requirements

Now then, going into specific industries is always an unthankful job, as there are quite a few well-regulated industries out there. Hence, a comprehensive list is not possible within the confines of this book and this chapter. However, some industries are as follows:

- Healthcare (HIPAA/HITECH): **Health Insurance Portability and Accountability Act** (**HIPAA**) and **Health Information Technology for Economic and Clinical Health Act** (**HITECH**) regulate the handling of **protected health information** (**PHI**) in the U.S.
- **Retail and e-commerce (PCI DSS):** PCI DSS ensures security for credit card transactions.
- **Government (FedRAMP, ITAR):** FedRAMP for cloud services used by U.S. federal agencies. **International Traffic in Arms Regulations** (**ITAR**) controls the export and import of defense-related articles and services.
- **Finance (GDPR)**: GDPR impacts financial services handling the personal data of EU citizens.
- **Retail and e-commerce (GDPR):** Ensures data protection and privacy for consumers.

Compliance for cloud computing workloads involves adhering to various regulatory bodies, standards, and industry-specific requirements.

Organizations must ensure they are meeting these regulations and guidelines to protect data, maintain privacy, and secure cloud environments.

Balancing compliance and security

Compliance is crucial in enhancing security within organizations, particularly in regulated industries.

Adhering to compliance requirements ensures that organizations implement a comprehensive set of security measures designed to protect sensitive data and systems.

There are some key methods through which compliance enhances security. Then, there is the concept of being more secure (but not absolutely secure). Finally, there is a fine line between compliance and best-effort security. So, let us discuss them.

How compliance enhances security

Compliance enhances security in many ways, and the following are some of the key concepts:

- **Structured frameworks**: Compliance regulations provide a structured framework for reimplementing security controls. This ensures that organizations have a systematic approach to protecting their information assets.
- **Risk management**: Compliance standards typically include risk assessment and management protocols, which help organizations identify, evaluate, and mitigate potential security threats.
- **Data protection:** Regulations like GDPR and HIPAA mandate strict data protection measures, ensuring that personal and sensitive information is handled securely and breaches are promptly reported and managed.
- **Monitoring and auditing:** Compliance requires ongoing monitoring and auditing of security practices, which helps in the early detection and remediation of vulnerabilities and threats.
- **Training and awareness:** Compliance frameworks often require regular employee training and awareness programs, which ensure that staff are knowledgeable about security policies and best practices.
- **3rd party vetting:** External audits and certifications validate an organization's compliance with security standards, providing additional assurance that security measures are effective.

Then, there is the concept of being more secure (but not absolutely secure).

The concept of more secure

The concept of more secure, not absolutely secure, acknowledges that while compliance and security measures can significantly reduce risks, they cannot eliminate them entirely. This concept is important for setting realistic expectations and fostering a proactive approach to security through the following methods:

- **Continuous improvement:** Security is an ongoing process. Organizations must continually update and improve security measures to adapt to threats and vulnerabilities.
- **Layered security:** Implementing multiple layers of security controls helps mitigate risks. Even if one layer is compromised, other layers can provide protection.

- **Balanced approach:** While achieving absolute security is impossible, striving to be more secure involves balancing compliance requirements with proactive security measures tailored to the organization's specific risk profile.
- **Incident response and recovery:** Organizations must be prepared for potential security incidents. Robust incident response and recovery plans ensure that they can quickly contain and recover from breaches.

And finally, there is a fine line between compliance and best-effort security. So, let us discuss these.

The concept of best-effort security

Balancing compliance and best-effort security involves integrating regulatory requirements with additional security practices to achieve a robust security posture.

The key word is additional, as it does not matter what you and your organization implement in terms of compliance; these frameworks are just the starting point, not the end goal. These frameworks are attempting to guide you and your organization, but they cannot cover every eventuality and nuance of your business.

Also, in this chapter, we want to help you by providing some strategies for balancing compliance and security, as follows:

- **Beyond minimum requirements**: Compliance should be viewed as the baseline. Organizations should go beyond compliance by implementing additional security measures based on their unique risk assessments.
- **Customization and flexibility:** Tailor security measures to the organization's specific needs and risks. This ensures that security efforts are effective and efficient.
- **Integrate compliance into the security strategy:** Compliance should be an integral part of the overall security strategy rather than a separate or secondary consideration.
- **Regular review and updates:** Regularly review and update compliance and security practices to address new threats and changes in regulatory requirements.
- **Leverage technology:** Use advanced technologies like AI and machine learning to detect and respond to threats and streamline compliance processes.
- **Foster a security culture:** Promote a culture of security within the organization where compliance and security are seen as shared responsibilities among all employees.

By understanding how compliance enhances security, recognizing the limitations of being more secure, not absolutely secure, and effectively balancing compliance with proactive security measures, you and your organization can create a resilient security framework that protects against current and emerging threats. Furthermore, this process never ends.

Auditing events

Internal auditing is an independent, objective assurance and consulting activity designed to add value and improve an organization's operations.

It helps an organization accomplish its objectives by bringing a systematic, disciplined approach to evaluate and improve the effectiveness of risk management, control, and governance processes.

Let us now look at the scopes, events, tools, and patterns your organization and you can plan for, capture, use, and deploy.

Scope

The scope of internal auditing encompasses the examination and evaluation of the adequacy and effectiveness of the organization's governance, risk management, and control processes.

This includes a few important types of audits, listed as follows:

- **Financial audits:** Assessing the accuracy and reliability of financial records and statements.
- **Compliance audits:** Ensuring adherence to laws, regulations, and internal policies.
- **Operational audits:** Evaluating the efficiency and effectiveness of operational processes.
- **Information systems audits:** Reviewing the controls over IT infrastructure and data integrity.
- **Performance audits:** Assessing the performance of specific departments, programs, or initiatives against objectives and benchmarks.
- **Special investigations:** Conduct investigations into specific incidents, such as fraud or misconduct.

Hopefully, the overall scope is clear, so let us talk about events.

Events

Security events are incidents or activities that may indicate a breach of security policies, procedures, or controls.

These include some of the most important types of data that you can capture:

- **Unauthorized access:** Attempts to access systems, applications, or data without proper authorization.
- **Data breaches:** Unauthorized disclosure, access, or theft of sensitive information.
- **Malware and cyber attacks:** Incidents involving viruses, ransomware, phishing, or other malicious activities.

- **Policy violations:** Actions that contravene established security policies and procedures.
- **Security system alerts:** Notifications from intrusion detection/prevention systems, firewalls, and antivirus software.

Operational events are activities or incidents that impact the efficiency and effectiveness of business operations. These include some of the most important types of data that you can capture. Some events are as follows:

- **System downtime:** Periods when systems or applications are unavailable due to failures or maintenance.
- **Process failures:** Breakdowns in business processes that affect service delivery or product quality.
- **Resource utilization:** Monitoring the usage of system resources such as CPU, memory, and storage.
- **Transaction logs:** Recording all transactions processed by systems to ensure accuracy and accountability.
- **Performance metrics:** Capturing **key performance indicators** (**KPIs**) to assess operational efficiency and effectiveness.

Hopefully, the events are clear, so let us talk about some tools you and your organization should and can use

Tools

Some of the most important tools are also some of the most widely used and categorized tools, so follow along to the list (and identify tools you and your organization already use, and also identify any potential gaps):

- **Security Information and Event Management systems:** Collect, analyze, and correlate security events from various sources to provide real-time analysis and alerts.
- **Log management solutions:** Centralize and manage log data from multiple sources for analysis and reporting.
- **Intrusion Detection/Prevention Systems (IDS/IPS):** Monitor network and system activities for malicious activities or policy violations and provide alerts or take preventive actions.
- **Endpoint Detection and Response (EDR) solutions:** Monitor endpoint activities to detect and respond to security threats.
- **Automated Compliance Tools:** Ensure compliance with regulatory standards by automating the collection, analysis, and reporting of compliance-related data.

Hopefully, some of these tools are/will be useful to you and your organization, so let us

Patterns

Some of the best practices for effective internal auditing are as follows:

- **Establish clear objectives and scope:** Define the purpose, objectives, and scope of the audit to ensure it aligns with organizational goals and priorities.
- **Develop a risk-based audit plan:** Prioritize audit activities based on the organization's risk assessment to focus on areas with the highest risk and impact.
- **Ensure independence and objectivity:** Maintain the independence of the internal audit function to provide unbiased and objective assessments.
- **Use a structured methodology:** Follow a consistent and systematic approach to auditing, including planning, fieldwork, reporting, and follow-up.
- **Leverage known technology and tools:** Utilize advanced auditing tools and technologies to enhance the efficiency and effectiveness of audit processes.
- **Engage stakeholders:** Collaborate with stakeholders throughout the audit process to ensure their concerns and insights are considered.
- **Provide clear and actionable recommendations:** Deliver audit findings and recommendations that are clear, practical, and actionable to drive improvements.
- **Monitor and follow up:** Continuously monitor the implementation of audit recommendations and follow up on corrective actions to ensure they are effectively addressed.
- **Continuous improvement:** Review and update audit processes and practices regularly to incorporate lessons learned and adapt to changing risks and regulatory requirements.

Using these patterns should help you understand the concept of enhanced security we discussed earlier in this chapter.

By adhering to these best practices and leveraging appropriate tools and technologies, you and your organization can conduct effective internal audits that enhance security, ensure compliance, and improve overall operational efficiency.

3rd party auditing

External auditing is an independent examination of an organization's financial statements, operations, or compliance with regulatory standards conducted by an external, impartial party.

External auditors who are not affiliated with the organization provide an objective evaluation, ensuring that the organization's records and processes are accurate, reliable, and in compliance with applicable laws and regulations.

Purpose

The primary purposes of external auditing include the following:

- **Verification of financial integrity:** Ensure that financial statements present a true and fair view of the organization's financial position and performance.
- **Regulatory compliance:** Verify that the organization adheres to relevant laws, regulations, and industry standards.
- **Risk management:** Identify and assess risks associated with financial misstatements, operational inefficiencies, and non-compliance.
- **Enhancing stakeholder confidence:** Provide assurance to stakeholders, including investors, creditors, regulators, and customers, about the organization's integrity and reliability.
- **Operational improvement:** Offer recommendations for improving processes, controls, and overall operational efficiency.

Let us now look at the types of audits.

Types of audits

Types of audits that are important for you and your organization are listed as follows:

- **Financial audits:**
    - **Objective**: Verify the accuracy and completeness of an organization’s financial statements.
    - **Scope**: Includes examining financial records, transactions, accounting policies, and internal controls.
    - **Outcome**: An audit report expressing an opinion on the financial statements' fairness and compliance with accounting standards.
- **IT audits:**
    - **Objective**: Assess the effectiveness and security of an organization’s information systems and IT infrastructure.
    - **Scope**: Includes evaluating IT governance, data integrity, security controls, and disaster recovery plans.
    - **Outcome**: Recommendations for improving IT security, efficiency, and compliance with IT-related regulations.
- **Operational audits:**
    - **Objective**: Evaluate the efficiency and effectiveness of business operations and processes.

  - **Scope**: Includes assessing operational workflows, resource utilization, and internal controls.
  - **Outcome**: Insights and recommendations for enhancing operational performance and achieving strategic objectives.
- **Compliance audits**
  - **Objective**: Ensure adherence to applicable laws, regulations, and internal policies.
  - **Scope**: Includes reviewing procedures, documentation, and practices to verify compliance.
  - **Outcome**: Identification of non-compliance issues and recommendations for corrective actions.

And now, let us talk about how to prepare for an audit.

Preparation

For an audit to go smoothly, the points to keep in mind are as follows:

- **Understand the audit scope and requirements:** Clarify the scope, objectives, and requirements of the audit with the external auditors.
- **Assemble an audit team:** Designate a team responsible for coordinating with auditors and preparing necessary documentation.
- **Gather and organize documentation:** Collect relevant financial records, policies, procedures, and other documentation required for the audit.
- **Review internal controls:** Evaluate and, if necessary, strengthen internal controls to ensure they are robust and effective.
- **Conduct a pre-audit assessment:** Perform an internal review to identify and address potential issues before the external audit.
- **Communicate with stakeholders:** Inform relevant stakeholders about the audit process and their roles in providing information and support.

Finally, let us look at some audit challenges and how to overcome them.

Challenges

Let us discuss some audit challenges and how to overcome them.

The challenges are as follows:

- **Incomplete or disorganized documentation:** Difficulty in providing required documentation can delay the audit process and lead to adverse findings.

- **Lack of preparation:** Inadequate preparation can result in overlooked issues and non-compliance findings.
- **Communication gaps:** Poor communication between the organization and auditors can lead to misunderstandings and incomplete information.
- **Resource constraints:** Limited availability of staff and resources can hinder the audit process and responsiveness to auditor requests.
- **Resistance to change:** Organizational resistance to addressing audit findings and implementing recommendations can undermine audit effectiveness.

Now, the following points highlight how to overcome some of these challenges:

- **Maintain organized records:** Regularly update and organize financial records, policies, and other documentation to ensure they are readily accessible.
- **Early and thorough preparation:** Begin preparing for the audit well in advance, conducting internal reviews, and addressing potential issues proactively.
- **Foster open communication:** Establish clear and consistent communication channels with auditors and internal stakeholders to facilitate information exchange.
- **Allocate sufficient resources:** Ensure adequate staff and resources are allocated to support the audit process, including temporary hires if necessary.
- **Embrace a culture of continuous improvement:** Promote a positive attitude towards audit findings and recommendations, viewing them as opportunities for improvement rather than criticisms.

After looking at these theoretical sections, let us move ahead to look at some ways cloud providers can help you with compliance.

Cloud providers compliance portals

Compliance portals are your greatest asset. Microsoft Azure, Amazon Web Services, and Google Cloud Platform portals are all great places to start – depending on the cloud provider you and your organization favor. They all take slightly different approaches to compliance, so let's discuss these differences now.

Azure

The Microsoft Azure portal lists all the compliance offerings by region (such as global, US, etc.) and by industry (financial, healthcare, etc.).

The portal also offers information on privacy and GDPR, built-in regulatory compliance offerings (such as CIS, FedRamp, HIPAA, ISO 27001, etc.), white papers, and analyst reports.

Azure prominently gives importance to the following topics, due to Microsoft's extensive enterprise focus:

- Data residency
- Breach notification
- Data subject requests
- Security controls
- Shared responsibilities
- Risk compliance guides
- Practical guides.

AWS

Amazon Web Services compliance portal lists the benefits of using AWS in addressing certification, regulations, and frameworks, as well as proving their compliance record through 3rd party validation.

As with other topics in this book, compliance is a shared responsibility, and AWS calls this out and then addresses how you and your organization can meet the compliance goals on AWS.

AWS gives prominences to the following topics, focusing on use cases across the spectrum of software delivery:

- Security services
- Use cases
- Compliance
- Data protection
- FAQs.

GCP

Google Cloud Platform showcases all the compliance offerings with ready-made reports and trust papers. GCP breaks down compliance by region (USA, EMEA, etc.) and by industry (financial services, government, public sector, healthcare, etc.).

GCP gives importance to these topics, trying to showcase how these considerations play in the real world:

- Regulatory considerations
- Trust
- Government requests for cloud customer data
- Data portability

- Data interoperability
- Laws and regulations.

Conclusion

It is only by identifying your organization's requirements from standpoints of global/regional market access, industry requirements, and further voluntary choices of frameworks to enable compliance that you and your organization can be compliant, prove that you are compliant, stay compliant, and improve compliance and security going forward.

In this chapter, you have learned how Azure, AWS, and GCP try and help you in this important process through their dedicated portals on compliance. Depending on which cloud provider/s you use, you need to familiarize yourself with these resources and use them in your day-to-day management of the cloud platform.

You have also learned about some of the patterns and practices for internal and external (third-party) audits.

In the next chapter we will discuss the topics of DevSecOps, Platform Engineering, and Site Reliability Engineering. These topics and the practices they represent are closely aligned with the compliance topics we discussed in this chapter. In the context of compliance in cloud security, DevSecOps, Platform Engineering, and **Site Reliability Engineering** (**SRE**) each bring unique contributions that can support regulatory adherence, data protection, and risk mitigation.

Questions

1. Can you think of any more ways that compliance enhances security?
2. Can you think of any ways that compliance may not enhance security but, in fact, do the opposite?
3. Is it true that if your organization is compliant with its industry security frameworks, it is absolutely secure?
4. Why is reputation and trust important to your organization?
5. Which common standards and frameworks apply to your organization's industry/industries?
6. What is NIST?
7. What is SOC 2?
8. What is GDPR?
9. How do you balance compliance and security?
10. How does compliance enhance security?

11. Why is training your staff in compliance important?
12. What is best-effort security?
13. Which tools does your organization use to manage compliance?

References

- ISO - Global standards for trusted goods and services: **https://www.iso.org/home.html**
- NIST - Working with industry and academia to enhance economic security and improve our quality of life: **https://www.nist.gov/**
- EDPB - **The European Data Protection Board** (**EDPB**) is an independent European body with legal personality. It ensures that the General Data Protection Regulation and the Law Enforcement Directive are applied consistently and ensures cooperation, including on enforcement: **https://www.edpb.europa.eu/edpb_en**
- CSA - Empowering Cybersecurity Through Research: **https://cloudsecurityalliance.org/**
- Azure compliance documentation: **https://learn.microsoft.com/en-us/azure/compliance/**
- AWS compliance: **https://aws.amazon.com/compliance/**
- GCP compliance resource center: **https://cloud.google.com/compliance**

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 16

DevSecOps, Platform Engineering, and Site Reliability Engineering

Introduction

In this chapter, we will discuss software engineering, how to engineer software securely, and how to secure software once it is running in the cloud. We will also discuss how to manage software engineering and running the software in the cloud.

We will look at all of these topics through the lens of security.

Structure

The chapter covers the following topics:

- Cloud security
- Patterns and anti-patterns
- Use cases

Objectives

Each of the disciplines we will discuss contributes uniquely to cloud security. In this chapter, you will become familiar with them.

DevOps enhances security through automation, consistent configuration, and real-time monitoring and integrates security into every phase of the development lifecycle, promoting early detection and mitigation of vulnerabilities.

Platform Engineering ensures that platforms are secure by design and integrates security tools and architectures seamlessly and **Site Reliability Engineering** (**SRE**) focuses on reliability and resilience, incorporating security as a key component to ensure systems remain secure and operational.

By working together, these disciplines create a robust and secure cloud environment, addressing security from multiple angles and ensuring comprehensive protection against threats.

In this chapter we will go through all of these concepts in enough detail for you to get a grasp of the basics and learn about the advanced areas within each topic.

Cloud security

Cloud security is a crucial aspect of modern IT operations, and various disciplines, such as DevOps, DevSecOps, Platform Engineering, and SRE, play significant roles in enhancing it. In this section, we will look at how each of these relates to cloud security.

DevOps

DevOps is a set of practices that combines **software development** (**Dev**) and **IT operations** (**Ops**). The goal of DevOps is to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.

By fostering a culture of collaboration and shared responsibility, DevOps aims to improve the efficiency, speed, and quality of software development and deployment through automation, continuous integration, and continuous delivery.

DevOps relates to cloud security in the following manner:

- **With automation and continuous integration/continuous deployment (CI/CD)**: DevOps practices emphasize automation and CI/CD, which can help enforce security policies and integrate security testing into the software development lifecycle.
- **By templating infrastructure as code**: This allows for the automation of infrastructure provisioning, ensuring consistent security configurations and reducing the risk of human error.
- **By ensuring monitoring and logging is done well**: DevOps encourages the use of comprehensive monitoring and logging tools to detect and respond to security incidents in real-time.

DevOps enhances security through automation, consistent configuration, and real-time monitoring.

DevSecOps

DevSecOps is an extension of DevOps that integrates security practices into the DevOps process. The primary aim is to embed security measures throughout the software development lifecycle rather than treating security as an afterthought or a separate process.

By incorporating automated security checks and ensuring that developers are aware of security best practices, DevSecOps helps organizations identify and address vulnerabilities early, thereby reducing risk and enhancing the security posture of their applications and infrastructure.

DevSecOps relates to cloud security in the following manner:

- **By ensuring security integration end to end**: DevSecOps is an extension of DevOps with a focus on integrating security throughout the software development life cycle. This includes embedding security controls, practices, and tools into the CI/CD pipeline.
- **By shifting-left security**: DevSecOps promotes the idea of 'shifting left,' which means addressing security issues early in the development process, reducing vulnerabilities, and mitigating risks before they reach production.
- **By enforcing automated security testing**: Here, tools and processes are integrated to automate security testing, such as static code analysis, dynamic application security testing, and dependency checking.

DevSecOps integrates security into every phase of the development lifecycle, promoting early detection and mitigation of vulnerabilities.

Platform Engineering

Platform Engineering focuses on creating and maintaining the underlying infrastructure and platforms that support the development, deployment, and operation of software applications. This discipline involves designing and implementing scalable, reliable, and efficient systems that provide developers with the tools and environments they need to build and deploy applications.

Platform engineers work on everything from cloud infrastructure and container orchestration to monitoring and logging systems, ensuring that the platform is robust and easy to use.

Platform Engineering relates to cloud security in the following manner:

- **By practicing secure platform design**: Platform engineers design and build platforms with security as a foundational component, ensuring that the underlying infrastructure and services are secure by default rather than an afterthought.
- **Implementing concepts such as zero trust architectures**: Platform Engineering often involves implementing advanced security models like service meshes and zero trust architectures, enhancing security through strict access controls and micro-segmentation.
- **By ensuring security practices are thought out end to end: E**nsuring that security tools (for example., firewalls, intrusion detection systems) are seamlessly integrated into the platform and that they work harmoniously with other components.

Platform Engineering ensures that platforms are secure by design and integrates security tools and architectures seamlessly.

Site Reliability Engineering

SRE is a discipline that applies software engineering principles to IT operations with the goal of creating highly reliable and scalable software systems. SREs are responsible for ensuring the reliability, performance, and availability of services, often by automating repetitive tasks and implementing rigorous monitoring and incident response strategies.

SRE engineers work to balance the need for new features with the need for stability, using metrics and service-level objectives to guide their efforts.

SRE relates to cloud security as follows:

- **By ensuring reliability and security are considered and planned for**: SRE focuses on ensuring the reliability of systems, which inherently includes aspects of security. Ensuring uptime and resilience often involves implementing robust security measures to protect against attacks.
- **By having a robust incident response and management**: SRE teams are skilled in incident response, which is crucial for quickly addressing security breaches and mitigating damage. They develop playbooks and practice incident response scenarios.
- **By ensuring monitoring and metrics are collected and used**: SREs rely heavily on monitoring and metrics, which can be extended to include security monitoring. This involves setting up alerts for suspicious activities and ensuring that security logs are properly analyzed.

SRE focuses on reliability and resilience, incorporating security as a key component to ensure systems remain secure and operational.

Similarities and differences

While DevOps, DevSecOps, Platform Engineering, and SRE all aim to improve the software development and delivery process, they focus on different aspects, as highlighted through the following points:

- DevOps emphasizes collaboration between development and operations teams to streamline workflows and automate processes, aiming to deliver software more quickly and efficiently.
- DevSecOps extends DevOps by integrating security practices into the development and operations lifecycle, ensuring that security is a core component of the entire process.
- Platform Engineering focuses on creating and maintaining the underlying infrastructure and platforms that support development and operations, providing developers with the necessary tools and environments.
- SRE applies software engineering principles to operations, focusing on ensuring the reliability, performance, and availability of services through automation, monitoring, and incident management.

In essence, DevOps and DevSecOps are more about cultural and procedural changes within the development and operations teams. At the same time, Platform Engineering and SRE are more technically focused, dealing with the infrastructure and reliability aspects of software systems.

Patterns and anti-patterns

Although all these concepts have benefits, they also have some issues regarding cloud security that we must be very mindful of.

Let us check in with each and see which patterns are good and which are not.

Note how some of these points bridge the gap between these practices, usually with a slightly different view of them. Also, note how this ties into the concepts we have discussed in the previous chapter.

Each of these patterns and anti-patterns will contain an example of it as well, so look for those and try to identify an example in your own organization and with your own practices.

DevOps

Let us begin by defining DevOps together from different viewpoints and see what the benefits are:

- DevOps practices integrate security into the CI/CD pipelines, ensuring security checks are automated and consistent; for example, automated vulnerability scanning during the build process catches potential issues early.
- DevOps practices promote faster detection and response to security incidents due to automated monitoring and logging; for example, real-time alerts and automated incident response mechanisms enable quick mitigation of threats.
- DevOps tools and practices enable scalable and flexible security measures that can adapt to changing cloud environments; for example, infrastructure as code allows for the automated and consistent deployment of security configurations across environments.
- DevOps encourages proactive security measures through practices like automated testing, continuous monitoring, and regular updates; for example, regularly updated security patches and configurations are applied automatically to prevent vulnerabilities.
- The integration of numerous tools in a DevOps pipeline can create complexity, potentially leading to misconfigurations and security gaps, for example, misconfigured access controls due to improperly integrated CI/CD tools.
- Development and operations teams may lack specialized security knowledge, leading to inadequately addressed security concerns. For example, insecure coding practices may result from a lack of training in secure coding principles.

If not properly secured, automated processes can become attack vectors for cyber threats. For example, compromised deployment scripts can lead to unauthorized access or data breaches.

The emphasis on speed and rapid deployment in DevOps can sometimes lead to security being deprioritized or overlooked. For example, pushing untested code to production to meet tight deadlines can potentially introduce vulnerabilities.

Inadequate monitoring and logging can result in delayed detection of security incidents, increasing the potential damage. For example, failure to log critical security events makes it difficult to trace and respond to security breaches effectively.

Here are some patterns for you to consider in this context:

- Integrate security early in the development process by incorporating security testing and practices in the CI/CD pipeline; for example, by implementing static application security testing during the coding phase to detect vulnerabilities early.
- Define and manage infrastructure using code, ensuring security configurations are version-controlled and consistently applied. For example, use tools like Terraform or AWS CloudFormation to enforce security policies and automatically deploy secure infrastructure.

- Implement continuous monitoring of applications and infrastructure to detect and respond to security incidents in real-time. For example, use **security information and event management** (**SIEM**) systems to aggregate and analyze logs for suspicious activity.
- Automate compliance checks and governance policies to ensure adherence to security standards and regulations. For example, use tools like AWS Config or Azure Policy to automatically check for and enforce compliance with security best practices.
- Deploy immutable infrastructure where servers are replaced rather than updated, reducing the risk of configuration drift and security vulnerabilities, for example, by using containerization and orchestration to deploy applications in a consistent, repeatable manner (think Docker, think Kubernetes).

Now, consider some anti-patterns in this context:

- Addressing security only at the end of the development cycle leads to rushed or incomplete security measures, such as conducting security testing only right before deployment, which results in missed vulnerabilities.
- Relying on manual processes for managing and configuring infrastructure increases the risk of human error and inconsistent security practices, such as manually configuring firewalls and access controls without automated validation, leading to potential misconfigurations.
- Keeping security responsibilities isolated within a separate team rather than integrating security expertise throughout the DevOps process, for example, security issues being addressed only by the security team, leads to delays and a lack of security awareness among developers and operations teams.
- Failing to integrate security checks and tests into CI/CD pipelines, allowing vulnerabilities to go undetected until later stages, for example, skipping automated security tests in the CI/CD pipeline to speed up deployment, resulting in insecure code reaching production.
- Inadequate logging and monitoring of applications and infrastructure make it difficult to detect and respond to security incidents. For example, not implementing centralized logging or monitoring solutions leads to missed alerts and delayed incident response.

DevSecOps

By integrating security practices early in the development lifecycle, vulnerabilities are detected and addressed sooner. For example, automated security testing tools catch vulnerabilities during the coding phase, reducing the risk of them reaching production.

DevSecOps promotes the use of automated security tools and scripts, ensuring consistent application of security measures across all environments. For example, automated configuration management tools ensure that all deployed environments adhere to security policies.

Continuous monitoring and automated incident response enable faster remediation of security issues. For example, real-time alerts and automated patching system systems quickly address detected vulnerabilities, minimizing exposure.

DevSecOps fosters collaboration among development, security, and operations teams, creating a culture of shared responsibility for security. For example, regular meetings and integrated workflows between teams ensure that security is a priority throughout the development process.

Continuous integration of security practices leads to a proactive rather than reactive approach to security. For example, continuous security assessments and threat modeling help identify and mitigate potential threats before they can be exploited.

Automated security tools and processes can scale with the application, ensuring that security measures are not compromised as the application grows. For example, cloud-native security tools that scale with the infrastructure ensure comprehensive security coverage.

Integrating a wide range of security tools into the DevSecOps pipeline can be complex and may lead to configuration issues. For example, misconfigured security tools can create gaps in the security posture, leading to potential vulnerabilities.

Development and operations teams may lack the necessary security expertise, leading to insufficient security measures. For example, developers might not have sufficient training in secure coding practices, resulting in code that is vulnerable to attacks.

Continuous security checks and automated testing can add overhead and impact the performance of the CI/CD pipeline. Extensive security scans might slow down the build process, causing delays in deployment.

The emphasis on rapid deployment can sometimes lead to security being deprioritized or overlooked. For example, in the rush to meet deployment deadlines, security tests might be skipped or not performed thoroughly.

Ensuring consistent security across multiple environments (development, testing, production) can be challenging. For example, different configurations in various environments might lead to inconsistent security practices and potential vulnerabilities.

Without robust monitoring and incident response strategies, security incidents may go undetected or unmanaged. For example, a lack of centralized logging and monitoring makes it difficult to detect and respond to security breaches in a timely manner.

The following are some patterns for you to consider:

- Integrate security policies and controls into code repositories and automate their deployment using infrastructure-as-code tools. For example, store firewall rules, access controls, and other security configurations in version-controlled repositories and use tools like Terraform or AWS CloudFormation to deploy them automatically.
- Implement continuous compliance checks within the CI/CD pipeline to ensure that all code and configurations adhere to security standards and regulatory requirements. For example, use automated tools to scan for compliance violations during the build and deployment process and generate reports to track compliance status.
- Use blue-green deployment strategies to ensure that new code releases are thoroughly tested for security vulnerabilities before going live. For example, deploy new versions to a staging environment (blue) that runs parallel to the production environment (green) and perform comprehensive security testing on the blue environment before switching traffic to it.
- Integrate automated incident response mechanisms into the DevOps pipeline to quickly detect, contain, and remediate security incidents. For example, set up automated alerts and responses for security events such as unauthorized access attempts or malware detections, using tools like AWS Lambda for custom remediation scripts.

Additionally, the following list highlights some of the important anti-patterns to avoid:

- Relying solely on manual security testing processes, which are time-consuming and prone to human error, leads to inconsistent and incomplete security coverage. For example, only conducting periodic manual penetration tests leaves potential vulnerabilities unaddressed between tests.
- Treating security as a separate concern from development and operations rather than integrating it into the DevOps culture and workflows, for example, security teams working in silos and not being involved in the early stages of the development lifecycle.
- Storing sensitive information such as API keys, passwords, and certificates directly in code or configuration files can be easily exposed. For example, embedding database passwords in application source code can lead to potential leaks if the code repository is compromised.
- Failing to assess and secure third-party tools and services integrated into the DevOps pipeline can introduce security vulnerabilities, for example, using third-party CI/CD tools without evaluating their security practices or implementing proper access controls.
- Neglecting comprehensive logging and monitoring of applications and

example, not collecting or analyzing logs from cloud services, making it difficult to trace and investigate security breaches.

- Adopting a reactive approach to security means addressing vulnerabilities and incidents only after they occur rather than proactively preventing them. For example, implementing security patches and updates only after a vulnerability has been exploited in the production environment.

Platform Engineering

Platform Engineering allows for centralized management of security policies, making it easier to enforce and update security controls; for example, a centralized platform team can manage and deploy consistent firewall rules and access controls across all cloud environments.

Automation tools and scripts ensure continuous compliance with security standards and regulations; for example, automated compliance checks and audits can be built into the platform to ensure adherence to frameworks like GDPR or HIPAA.

Centralized logging and monitoring provide comprehensive visibility into the security posture of cloud environments, for example, implementing a centralized SIEM system to collect and analyze logs from all cloud resources.

Platform Engineering promotes standardized security practices across all projects and teams, reducing the risk of misconfigurations. For example, it uses pre-approved templates and blueprints for deploying secure cloud resources.

Centralized incident response mechanisms enable faster and more coordinated responses to security incidents; for example, a dedicated incident response team within the Platform Engineering group can quickly address and mitigate security breaches.

Integrating security into CI/CD pipelines ensures that all code and configurations are tested for vulnerabilities before deployment; for example, automated security testing, such as static and dynamic analysis, is included in the CI/CD process.

Centralized management of access controls and identity ensures that only authorized users can access sensitive resources, for example, by implementing **single sign-on** (**SSO**) and **multi-factor authentication** (**MFA**) across all cloud services.

Platform Engineering can enforce resource isolation and network segmentation to limit the impact of security breaches. For example, it can use **Virtual Private Clouds** (**VPCs**) and subnets to isolate different parts of the application infrastructure.

Regular training and awareness programs for platform engineers keep them updated on the latest security threats and best practices, for example, conducting regular security workshops and simulations for the Platform Engineering team.

Platform Engineering provides scalable security solutions that grow with the organization's needs. For example, it implements auto-scaling security groups and policies that adapt to changes in cloud resource usage.

Centralized security management can become complex, especially in large organizations with diverse cloud environments; for example, managing and synchronizing security policies across multiple cloud providers and regions can be challenging.

Over-reliance on automated security tools can lead to complacency and overlooked vulnerabilities; for example, automated security scans might miss context-specific issues that require manual review.

Platform engineers may lack the specialized security skills needed to address sophisticated threats. For example, without continuous training, they might not be equipped to handle advanced security incidents.

If monitoring and response systems are not adequately set up, security incidents may go undetected or unmanaged. For example, a lack of real-time alerts and insufficient incident response protocols can delay the mitigation of security breaches.

Over time, discrepancies between the intended security configurations and the actual state of cloud resources can occur. For example, manual changes to cloud resources can lead to inconsistencies and vulnerabilities.

As the organization grows, scaling security practices and tools can become increasingly difficult. For example, ensuring that security policies and controls scale efficiently with the growth in cloud resource usage and the number of users can be challenging.

Striking the right balance between maintaining high-security standards and enabling agile development can be challenging; for example, strict security controls may slow down development processes, leading to friction between security and development teams.

Consider these important patterns:

- Define and manage infrastructure using code to enforce consistent security configurations across all environments, for example, by using tools like Terraform or AWS CloudFormation to automate the deployment of security groups, network ACLs, and encryption settings.
- Implement centralized logging and monitoring systems to provide comprehensive visibility and facilitate quick response to security incidents, for example, by using platforms like **Elasticsearch, Logstash, Kibana** (**ELK**) Stack, or AWS CloudWatch to aggregate logs from all cloud resources and set up alerts for suspicious activity.
- Integrate security testing tools into CI/CD pipelines to ensure that code is automatically tested for vulnerabilities before deployment. For example, incorporate tools like Snyk or OWASP ZAP into the CI/CD process to perform static and dynamic security analysis on every build.

- Implement RBAC and enforce the principle of least privilege to restrict access to cloud resources based on users' roles and responsibilities. For example, use **Identity and Access Management** (**IAM**) policies in AWS or Azure RBAC to grant users the minimum permissions necessary for their tasks.

Check out some obvious and not-so-obvious anti-patterns:

- Relying on manual processes to configure and manage cloud resources leads to inconsistent security practices and potential vulnerabilities, for example, manually setting up security groups and access controls, which can result in human errors and misconfigurations.
- Applying different security policies and controls in development, testing, and production environments can lead to security gaps. For example, more lenient security settings in development environments can be exploited to gain access to production systems.
- Failing to incorporate security checks and validations in infrastructure automation scripts and tools, for example, automating infrastructure deployment without verifying security configurations, such as missing encryption settings or open ports.
- Using overly complex security tools and processes that are difficult to manage and understand can lead to misconfigurations and reduced effectiveness. For example, deploying a complex mix of security tools that require extensive configuration and maintenance can result in potential security gaps.
- Not conducting regular security audits and reviews to assess the effectiveness of security controls and identify vulnerabilities, for example, failing to perform periodic reviews of IAM policies, security group rules, and compliance with security standards.
- Storing sensitive information such as API keys, passwords, and certificates directly in code or configuration files can be easily exposed. For example, embedding database passwords in application source code can lead to potential leaks if the code repository is compromised.

Site Reliability Engineering

SREs implement robust monitoring systems that proactively detect security anomalies and potential threats. For example, they use tools like Prometheus and Grafana to monitor unusual patterns in network traffic or system behavior.

SRE practices emphasize automated incident response, enabling quicker detection and mitigation of security incidents, for example, automated scripts that isolate compromised instances and notify the security team immediately upon detection of a breach.

SRE ensures high reliability and availability. It also involves maintaining a secure environment to prevent downtime due to security issues, such as implementing redundant security controls and failover mechanisms to ensure continuous protection.

SREs use infrastructure as code to manage infrastructure, allowing for consistent and repeatable security configurations. For example, they can use Terraform or Ansible to enforce security policies across all cloud environments automatically.

SREs adopt a culture of continuous improvement, learning from past incidents to strengthen security measures. For example, they conduct post-incident reviews to identify security weaknesses and implement improvements based on the lessons learned.

SREs often work closely with security teams, integrating security into the overall reliability framework. For example, they may conduct regular security reviews and threat modeling sessions with security experts to identify and address potential vulnerabilities.

SREs perform capacity planning and risk management, which includes anticipating and mitigating security risks. For example, they analyze system load and usage patterns to predict and prevent potential security bottlenecks and failures.

SREs use automation to consistently enforce compliance with security policies and regulatory requirements. For example, they set up automated compliance checks integrated into CI/CD pipelines to ensure all deployments meet security standards.

SREs focus on building resilient systems that can withstand and recover from security incidents, for example, implementing chaos engineering practices to test the system's response to security failures and improve resilience.

SREs enhance security observability by implementing comprehensive logging and tracing across systems, for example, using distributed tracing tools like Jaeger to track the flow of requests and detect anomalies indicating potential security issues.

Ensuring security measures do not negatively impact system performance and reliability can be challenging; for example, implementing extensive security checks may introduce latency or slow down system performance.

SREs may lack specialized security expertise, leading to gaps in the implementation of effective security measures, that is inadequate knowledge of advanced security threats and mitigation techniques among SREs.

Automating security processes can add complexity and may introduce new security vulnerabilities if not managed properly, for example, misconfigured automation scripts that inadvertently expose sensitive data or create security loopholes.

An overreliance on security tools without adequate manual oversight can result in missed threats or false security. For example, automated tools might fail to detect sophisticated attacks that require human intuition and analysis.

Integrating security into existing SRE practices and tools can be challenging, leading to potential gaps in coverage. For example, integrating security monitoring with existing performance monitoring tools can be difficult, resulting in fragmented visibility.

Now, let us check out some patterns to follow:

- Implement automated incident response mechanisms to quickly detect and mitigate security incidents, for example, automated scripts that trigger in response to security alerts to isolate affected resources and notify security teams.
- Use chaos engineering principles to proactively test security resilience and response capabilities. For example, introduce controlled security failures in production environments to assess system responses and improve defenses.
- Implement continuous monitoring of cloud infrastructure and applications to detect security threats in real-time, for example, Using SIEM tools to monitor logs and network traffic for suspicious activity.
- Define security configurations and policies as code to ensure consistent and repeatable deployment across environments, for example, using Terraform or CloudFormation templates to automate the provisioning of secure infrastructure resources.
- Use immutable infrastructure principles to minimize security risks by ensuring that deployed components are never modified after deployment. For example, use containerization (e.g., Docker) and orchestration tools (e.g., Kubernetes) to deploy and manage applications in a consistent, repeatable manner.
- Integrate security practices into the DevOps pipeline to ensure that security is considered at every stage of development and deployment. For example, incorporate automated security tests (such as static code analysis and vulnerability scanning) into CI/CD pipelines to catch issues early.
- Adopt a zero-trust architecture to verify and authenticate every request, regardless of whether it originates from inside or outside the network perimeter, for example, by implementing micro-segmentation and strict access controls based on identity and context rather than network location.

After considering the patterns, let us also consider some anti-patterns:

- Relying solely on manual processes for incident response can lead to delayed detection and response times during security incidents, for example, depending on human intervention to manually analyze logs and respond to security alerts without automated processes.
- Conducting chaos engineering experiments without considering security implications, potentially exposing vulnerabilities, for example, performing network or system stress tests without considering the impact on security controls or exposing sensitive data.

- Monitoring only selects parts of the infrastructure or applications, leaving blind spots that attackers could exploit, for example, monitoring server logs but neglecting to monitor database access logs or application layer traffic.
- Using static security configurations that are not updated or reviewed regularly increases the risk of outdated or insecure settings, for example, setting up firewall rules and access controls during initial deployment and never revisiting or updating them as security threats evolve.
- Depending too heavily on automated security tools without human oversight leads to false positives or missed security incidents, for example, trusting automated vulnerability scanners to detect all security issues without validating findings through manual analysis.
- Failing to include security configurations and checks in Infrastructure as Code templates leads to insecure deployments, for example, by deploying cloud resources using infrastructure as code scripts that do not enforce encryption, access controls, or secure networking configurations.
- Not providing SREs with adequate training on security best practices and emerging threats leads to gaps in security defenses, i.e., SREs lack awareness of common attack vectors or fail to understand the implications of new security vulnerabilities.

Use cases

It is crucial for you to understand and know when to use each practice in your daily management of your cloud workloads. Here is a summary and a definitive identification guideline.

DevOps

DevOps practices are essential for seamlessly integrating development and operations in cloud environments. Teams use DevOps methodologies to accelerate software delivery, but they should also embed security practices throughout the development lifecycle. DevOps is particularly valuable when organizations seek to automate application provisioning, deployment, and scaling on cloud platforms.

By incorporating security early in the pipeline (DevSecOps), DevOps teams can ensure that cloud deployments are not only fast and efficient but also resilient to security threats, with automated testing, monitoring, and compliance checks integrated into CI/CD workflows.

DevSecOps

DevSecOps extends DevOps principles by integrating security into every phase of the software development lifecycle, specifically focusing on cloud environments. This

deployments. DevSecOps practices ensure that security is not an afterthought but a core part of development, operations, and deployment processes.

Teams use automated security testing, continuous monitoring, and secure configurations to protect cloud resources and data. By embedding security controls early and throughout the development pipeline, DevSecOps enables organizations to mitigate risks proactively and maintain regulatory compliance in a cloud-agnostic manner.

Platform Engineering

Platform Engineering plays a pivotal role in architecting and managing the underlying infrastructure and services that support applications in a cloud environment. This discipline is vital when organizations need to design scalable, reliable, and secure cloud-native architectures.

Platform engineers implement best practices for network security, identity management, and data protection across diverse cloud platforms. They ensure that cloud deployments are resilient to cyber threats and adhere to industry standards. Platform Engineering is particularly valuable in multi-cloud or hybrid cloud environments, where consistency in security controls and infrastructure management is essential for operational efficiency and risk mitigation.

Site Reliability Engineering

SRE practices focus on ensuring the reliability, availability, and performance of cloud-based applications and services. SRE teams leverage automation, monitoring, and incident response strategies to maintain service uptime and operational excellence in cloud environments. This discipline is critical when organizations prioritize operational resilience and quick recovery from security incidents or disruptions.

SREs implement robust monitoring and logging solutions to detect and mitigate security threats promptly. They conduct regular resilience testing and performance optimization to enhance the overall security posture of cloud deployments. SRE practices are instrumental in maintaining **service-level objectives** (**SLOs**) and delivering a secure and reliable user experience in cloud-agnostic environments.

Conclusion

This chapter helped you get well-equipped in terms of DevOps, DevSecOps, Platform Engineering, and SRE to know what they stand for when to use them, and what security implications they bring with them. You also learnt how to apply some patterns and the anti-patterns you must avoid.

This was the last chapter, looking at current patterns and practices. In the next chapters, we will focus more on upcoming technologies and the future, so if you want to know

more about how everything you have learned so far in this book will be extended and used in the future (near and far), read on. Specifically, in the next chapter, we will focus on Machine Learning and Artificial Intelligence.

Questions

1. Explain how you integrate security practices into CI/CD pipelines in a cloud environment.
2. What strategies do you use to ensure secure configuration management across different cloud providers?
3. How do you manage secrets and sensitive information securely in a distributed cloud environment?
4. Describe how you approach vulnerability management and patching in a cloud-native application deployment.
5. How do you monitor and detect potential security incidents in a cloud-based infrastructure?
6. How do you integrate security testing (for example, static analysis, dynamic analysis) into the CI/CD pipeline for cloud applications?
7. Explain your approach to automating compliance checks and ensuring regulatory requirements are met in a multi-cloud environment.
8. What tools and techniques do you use to implement **infrastructure as code** (**IaaC**) securely in a cloud setup?
9. Describe how you handle security audits and ensure continuous security improvements in a DevSecOps culture.
10. How do you foster collaboration between development, operations, and security teams to promote a security-first mindset in cloud deployments?
11. How do you design and implement secure service mesh architecture in a cloud-agnostic environment?
12. Explain your approach to implementing secure network segmentation and isolation in a multi-cloud infrastructure.
13. What strategies do you use to ensure secure container orchestration and management across different cloud platforms?
14. Describe your methods for securing data storage and ensuring encryption at rest and in transit in a hybrid cloud setup.
15. How do you manage identity and access management (IAM) securely across various cloud providers and services?

16. How do you ensure high availability and disaster recovery while maintaining security in a cloud-native environment?
17. Explain your approach to monitoring and logging in a multi-cloud setup to detect and respond to security incidents promptly.
18. What measures do you take to ensure secure automation and orchestration of cloud infrastructure operations?
19. Describe how you implement resilience testing and chaos engineering to validate security controls in cloud deployments.
20. How do you conduct post-incident reviews and incorporate security improvements into SRE practices in a cloud-agnostic environment?

References

- Docker: **https://www.docker.com/**
- Kubernetes: **https://kubernetes.io/**
- Terraform: **https://www.terraform.io/**
- Azure Policy: **https://learn.microsoft.com/en-us/azure/governance/policy/overview**
- Snyk: **https://snyk.io/**
- ZAP: **https://www.zaproxy.org/**
- My talk at PlatformCon 2024 on "Platform team daily, weekly, monthly, and quarterly schedule!": **https://platformcon.com/talks/platform-team-daily-weekly-monthly-and-quarterly-schedule**

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 17

Machine Learning and Artificial Intelligence

Introduction

ML stands for machine learning, and AI stands for artificial intelligence. These two concepts are sometimes used interchangeably. We will try and define them here so that you can understand them within the context of cloud security.

AI is apparently everywhere these days. Is that hype justified? We are not going to try to answer that question here.

We are going to learn how to manage cloud security around these concepts. Securing ML and AI operations and running their related workloads in the public cloud has special challenges aside from the common cloud-related ones we have already discussed in all the previous chapters of this book.

So, let us look at how to manage, delegate, and oversee ML and AI workloads.

Structure

The chapter covers the following topics:

- Machine learning versus artificial intelligence
- The recent prominence of AI

- C-suite understanding of ML and AI
- Cloud security
- Managing ML and AI workloads

Objectives

The objective of this chapter is to give you an overview of ML, AI, and practical ways to manage these workloads in the public cloud.

By the end of this chapter, you should be able to differentiate between ML and AI and explain your workloads in ML and AI to anyone within your organization's hierarchy, as well as what they need to do to securely use, deploy, manage, and innovate on top of it.

Machine learning versus artificial intelligence

In this section, we will focus on defining both ML and AI in terms of how anyone can understand them, irrespective of their predefined understanding of the topic.

Machine learning

Let us start with three definitions of ML: one for a 5-year-old, one for an average person, and one appropriate for a scientist. Some basic points about ML are as follows:

- Machine learning is like teaching a computer to learn new things by showing it lots of examples, just like how you learn by looking at pictures and practicing. If you show a computer lots of pictures of cats and dogs, it can learn to tell the difference between them.
- Machine learning is a technology where computers learn from data and improve their performance over time without being explicitly programmed. They find patterns and make decisions based on the data they are given. When you use a music streaming service that suggests new songs you might like based on the songs you have listened to before, that is machine learning at work.
- Machine learning is a subset of artificial intelligence that focuses on the development of algorithms and statistical models enabling computers to improve their performance on a specific task through iterative learning from data without being explicitly programmed for each instance. In a supervised learning scenario, a neural network is trained on a labeled dataset to classify images into predefined categories, achieving state-of-the-art accuracy in image recognition tasks.

Now let us look at a few more examples of ML:

- Imagine teaching a robot to recognize your favorite toys by showing it many pictures of them until it learns to tell which toy is which.
- Your email service uses machine learning to filter out spam emails. It learns from thousands of examples to identify which ones are junk and which ones are important.
- A **convolutional neural network** (**CNN**) is used for image classification tasks. The network learns hierarchical features from pixel values through multiple layers of convolutional filters, resulting in high accuracy in object recognition challenges.

Artificial intelligence

Let us do the same as before by providing three definitions of AI: one for a 5-year-old, one for an average person, and one appropriate for a scientist.

AI occurs when computers or robots try to think and act like people. They can learn things, make decisions, and solve problems. For example, when you talk to a smart assistant like Alexa or Siri, and it answers your questions, that's artificial intelligence.

Artificial intelligence is a branch of computer science that aims to create machines capable of performing tasks that normally require human intelligence. These tasks include understanding natural language, recognizing patterns, solving problems, and making decisions. Self-driving cars use AI to navigate roads, avoid obstacles, and make decisions on where to go.

Now let us look at a few more examples of AI:

- A robot dog that can play fetch with you, understand when you call its name, and avoid bumping into furniture is using artificial intelligence to do all those things.
- AI is used in virtual personal assistants like Google Assistant, which can understand your voice commands, answer questions, set reminders, and control smart home devices.
- Implementing a **generative adversarial network** (**GAN**) to generate realistic images, where two neural networks, a generator and a discriminator, are trained simultaneously to create and evaluate images, leading to high-quality synthetic image generation.

Examples of ML that people confuse for AI

Let us now try to understand some common examples of ML that people often confuse with AI and why they do so:

- Spam filters use ML algorithms to classify emails as spam or not spam based on patterns learned from historical data. People often see spam filtering as a form of

intelligent behavior and thus attribute it to AI, even though it primarily involves ML techniques.

- Platforms like Netflix or Amazon use ML to recommend products or content based on user preferences and past behaviors. The personalized recommendations seem smart and intuitive, leading people to think of them as AI.
- ML models power features like autocomplete and predictive text in smartphones and search engines. The ability to predict what you are going to type next feels like a form of artificial intelligence.
- ML algorithms, especially CNNs, identify objects and people in images. Recognizing objects in images appears intelligent, so people often consider it AI, though it is a subset of ML.
- Businesses use ML to segment customers based on purchasing behavior and demographics. The segmentation process seems like an intelligent grouping of customers, leading to confusion with AI.

Examples of AI people confuse for ML

Let us now try to understand some common examples of AI that people often confuse with ML and why they do so:

- AI systems like chatbots and virtual assistants (for example, Siri and Alexa) understand and generate human language. While ML techniques are used in NLP, the broader capability to understand and respond to human language is a feature of AI.
- Self-driving cars use a combination of sensors, ML, and AI to navigate and make driving decisions. The complex decision-making and situational awareness required for driving are aspects of AI, not just ML.
- AI systems that can solve a variety of problems across different domains, like AlphaGo, which defeated human champions in the game of Go. These systems often use ML models, but their ability to handle a range of tasks and adapt strategies is characteristic of AI.
- AI systems that generate original artwork, music, or literature. The creative output of these systems seems like a straightforward application of ML, but true creative generation involves broader AI techniques.
- **Robotic process automation** (**RPA**) systems with cognitive AI can understand and process natural language documents, interact with humans, and make decisions based on context. RPA involves rule-based automation, but when enhanced with cognitive AI, it moves beyond the scope of ML.

The recent prominence of AI

Recently, AI has been featured prominently in the tech world and beyond. So why has AI recently had such intensive publicity or promotion? There are two answers on a basic level of human perception:

- Nowadays, everyone can use AI. AI is available to everyone with a smartphone or a web browser. The current applications of manipulating images or generating video seem like magic to people, and everyone wants to try it, and they actually can. We live in a time where AI is only a few seconds removed from anyone. You can try it now, reach for your phone, or reach out to your voice assistant and see how long it takes you before AI answers.
- AI is big business nowadays; companies – especially cloud providers – are promoting it, developing it, and – selling it or trying to sell it as the only way forward to the future. The claim is that the future of insurance, health, art, productivity, etc., is AI, and if you and your organization are not using AI, then your competitors are. It is understandable that cloud providers are hyping it so much because developing and running AI is expensive, and a pay-per-use model only works at enormous scales, so everyone must start using this AI service, or else they are not profitable.

Now, let us discuss some obvious and not-so-obvious pros and cons of AI.

Pros of AI

Here are some of the AI benefits that are immediately obvious across a broad range of fields:

- AI has the potential to transform industries by automating routine tasks, optimizing complex processes, and enabling new capabilities. For example, AI-driven medical diagnostics can improve patient outcomes and efficiency in healthcare.
- AI can analyze vast amounts of data to uncover insights that would be impossible for humans to detect. This leads to more informed and accurate decision-making in fields like finance, marketing, and logistics.
- AI technologies enable highly personalized experiences, from targeted advertising to customized product recommendations, which can significantly enhance customer satisfaction and loyalty. Achieving this with white-glove service that depends on people is not as easily scalable.
- By automating repetitive and time-consuming tasks, AI can significantly reduce operational costs and increase productivity. This is evident in applications like automated customer service and RPA.
- AI is driving innovation and creating new business models, such as autonomous vehicles, smart cities, and advanced robotics. These advancements are opening new economic opportunities and markets.

In addition to these cloud providers themselves are availing AI to improve their cloud business.

- Cloud providers can offer AI-powered services such as natural language processing, image recognition, and predictive analytics as part of their platform, attracting more customers. For instance, Google Cloud's AI Platform enables businesses to build and deploy machine learning models at scale, enhancing its appeal to data-driven enterprises.
- AI can optimize data center management by predicting hardware failures and optimizing energy usage. **Amazon Web Services** (**AWS**) uses AI to optimize its cooling systems in data centers, significantly reducing energy consumption and operational costs.
- AI enhances security by detecting and mitigating threats in real-time. Microsoft Azure uses AI-driven threat detection to identify and respond to cybersecurity threats faster and more accurately, providing a more secure environment for its customers.
- AI chatbots and virtual assistants can provide 24/7 customer support, handling common queries and issues. IBM Cloud employs AI-powered virtual agents to assist customers with technical support and account management, improving customer satisfaction and reducing the workload on human support staff.
- AI can predict resource demand and optimize the allocation of computing resources, ensuring scalability and cost-effectiveness. Google Cloud uses AI to manage and allocate resources dynamically, ensuring that applications have the resources they need without overprovisioning, which helps maintain performance and reduce costs.

Let us dig a bit deeper into one of these, the Microsoft Co-Pilot concept, specifically for writing code. Co-Pilot integrates AI to assist developers by providing intelligent code suggestions, automating repetitive tasks, and debugging code. This AI-driven tool significantly boosts developer productivity and reduces the time required to develop and deploy applications.

Co-Pilot leverages AI to understand the context of the code being written and offers real-time suggestions, like how a human pair programmer would. It helps with things like:

- Code completion by providing suggestions for completing code snippets, functions, and even entire code blocks based on the developer's current work context.
- Error detection and correction by identifying potential errors in the code and suggesting corrections, reducing the time spent on debugging.
- Learning and onboarding by helping new developers quickly get up to speed by using Co-Pilot to understand best practices and coding standards within a project.

A developer working on an Azure-based web application can use Co-Pilot to automatically generate boilerplate code for API endpoints, suggest optimal ways to handle data storage with Azure Cosmos DB, and even automate the setup of **continuous integration/continuous deployment** (**CI/CD**) pipelines. This not only speeds up the development process but also ensures higher code quality and consistency.

Co-Pilot's integration of AI into the development workflow exemplifies how cloud providers can leverage AI to enhance their platform's usability, making it more attractive to developers and enterprises seeking efficient and reliable cloud solutions.

Cons of AI

On the other hand, AI has some obvious disadvantages as well, depending on your perspective and how it is being used.

There is often an overestimation of what AI can currently achieve. Many AI systems are limited to narrow tasks and cannot match human intelligence in broader, more complex contexts.

AI systems can perpetuate and even amplify existing biases in data, leading to unfair or discriminatory outcomes. Ethical issues, such as privacy invasion and surveillance, also raise significant concerns.

Deploying AI effectively requires substantial investments in technology, infrastructure, and talent. Many organizations struggle with integrating AI into their existing systems and processes, leading to failed or suboptimal implementations.

Many AI models, especially deep learning algorithms, operate as *black boxes*, making it difficult to understand how they make decisions. This lack of transparency can hinder trust and accountability.

The rapid advancement of AI has outpaced the development of regulatory frameworks. This can lead to legal uncertainties and potential misuse of AI technologies. Additionally, AI systems are vulnerable to security threats, such as adversarial attacks and data breaches.

While AI has demonstrated significant potential and achieved remarkable advancements, it is important to recognize its current limitations and the challenges that need to be addressed. Balancing optimism with realistic expectations and a focus on ethical and responsible AI development will be crucial for realizing its full potential.

C-suite understanding of ML and AI

It is also crucial to understand how to discuss ML/AI with the C-suit and their respective part of the organization. So, let us quickly go through the differences in perspective from each of the C-suites.

Chief Executive Officer

The **Chief Executive Officer** (**CEO)** views ML and AI in cloud security as strategic tools that can enhance the company's competitive edge by ensuring robust asset protection, improving customer trust, and safeguarding the organization's reputation.

The focus is on the following:

- **Strategic risk management:** ML and AI in cloud security should align with the company's overall risk management strategy, protecting against potential cyber threats that could disrupt business operations.
- **Customer trust and reputation:** Leverage advanced security measures to build and maintain customer trust, showcasing a commitment to safeguarding their data.
- **Regulatory compliance:** Oversee the adoption of ML and AI technologies to ensure compliance with relevant data protection regulations and avoid legal issues.

Chief Financial Officer

The **Chief Financial Officer** (**CFO**) focuses on the financial implications of implementing ML and AI for cloud security. They are interested in the cost-effectiveness and **return on investment** (**ROI**) of these technologies.

The focus is on the following:

- **Cost management:** Evaluate the costs associated with deploying ML and AI for cloud security, ensuring that the investment is justified and aligns with budget constraints.
- **Financial risk reduction:** Use ML and AI to prevent financial losses due to cyber attacks, fraud, and data breaches, thereby protecting the company's financial health.
- **ROI analysis:** Assess the financial benefits of implementing ML and AI in cloud security, such as reduced incidence of security breaches and lower remediation costs.

Chief Technical Office

The **Chief Technical Office** (**CTO**) is focused on the technical implementation and effectiveness of ML and AI in enhancing cloud security. They are concerned with selecting the right technologies and integrating them into the existing infrastructure.

CTO focuses on the following:

- **Technology selection and integration:** Choose the most effective ML and AI technologies for cloud security and ensure seamless integration with existing systems and infrastructure.
- **Scalability and performance:** Ensure that the security solutions scale effectively with the company's growth and do not hinder performance.
- **Innovation and continuous improvement:** Drive continuous improvement in security measures by staying up-to-date with the latest advancements in ML and AI technologies.

Chief Information Officer

When it comes to using ML and AI for cloud security, the **Chief Information Officer** (**CIO**) focuses on data protection, operational efficiency, and regulatory compliance. They are concerned with implementing robust security protocols and ensuring smooth IT operations. Their focus is on the following:

- **Data protection and privacy:** Implement ML and AI to enhance data protection measures, ensuring sensitive information is secure and privacy is maintained.
- **Operational efficiency:** Use AI and ML to automate security processes, reduce manual intervention, and improve the efficiency of security operations.
- **Compliance and governance:** Ensure that security protocols comply with regulatory requirements and internal governance standards, using ML and AI to monitor and enforce these rules.

Cloud security

ML and AI have similar but distinct security areas that require your and your organization's focus to mitigate. These security issues highlight the complexities and challenges associated with deploying and managing ML and AI systems in the cloud. Proper security measures, continuous monitoring, and compliance with regulatory standards are essential to mitigate these risks.

ML security issues

Some of the most common ML-related issues you will run into when deploying cloud workloads are as follows:

- **Data privacy and confidentiality**: ML models often require vast amounts of data, which can include sensitive personal information. Ensuring this data remains confidential is crucial.

- **Data privacy and confidentiality**: ML models often require vast amounts of data, which can include sensitive personal information. Ensuring this data remains confidential is crucial.
- **Model theft**: Attackers can steal ML models deployed in the cloud, which can lead to intellectual property theft and unauthorized use of the model.
- **Model inversion attacks**: Adversaries can use model outputs to infer sensitive information about the training data, compromising privacy.
- **Adversarial attacks**: Carefully crafted inputs can be used to fool ML models, causing them to make incorrect predictions or classifications.
- **Insider threats**: Cloud service providers and employees with access to the ML models or data might misuse their access, leading to data breaches or model theft.
- **Lack of transparency**: Many ML models operate as black boxes, making it difficult to understand their decision-making processes and identify potential biases or vulnerabilities.
- **Model updates and patch management**: Ensuring that ML models are regularly updated and patched for security vulnerabilities can be challenging in a cloud environment.
- **Compliance and regulatory issues**: ML models and data in the cloud must comply with various regulatory requirements (for example, GDPR, HIPAA), and non-compliance can lead to legal issues.
- **Distributed Denial of Service (DDoS) attacks**: Attackers can target ML services in the cloud with DDoS attacks, disrupting the availability of these services.

AI security issues

Some of the most common AI-related issues you will run into when deploying cloud workloads are as follows:

- **Algorithmic bias**: AI systems can perpetuate and even amplify biases present in the training data, leading to unfair or discriminatory outcomes.
- **Autonomy and decision-making**: AI systems making autonomous decisions can have unintended consequences if they are not properly monitored and controlled.
- **Malicious use of AI**: AI can be used to develop sophisticated attacks, such as deepfakes, automated phishing, and more.
- **AI model exploits**: Attackers can find and exploit vulnerabilities in AI models to manipulate their behavior for malicious purposes.

- **Trust and reliability**: Ensuring the trustworthiness and reliability of AI systems is difficult, particularly when they operate in dynamic and unpredictable environments.
- **Explainability and accountability**: AI models, especially deep learning models, often lack explainability, making it hard to understand their decision-making process and hold them accountable.
- **Security of AI development pipelines**: Ensuring the security of the entire AI development pipeline, from data collection to model deployment, is challenging in a cloud environment.
- **Human-AI interaction risks**: Miscommunications or misunderstandings between humans and AI systems can lead to security risks, especially in critical applications.
- **Ethical and legal issues**: The use of AI raises numerous ethical and legal issues, such as surveillance, privacy, and the potential for AI to be used in harmful ways.
- **AI governance and control**: Implementing effective governance and control mechanisms for AI systems to ensure they operate within acceptable parameters is challenging, especially in a decentralized cloud environment.

With AI, there are additional considerations: data provenance and lineage, which means tracking data from the source to the end result and documenting the transformation; third-party vendor security, which means vetting third-party vendors that work with you on AI models; and transparency and explainability, which means proving your model is transparent, and results are explainable and free of bias.

Managing ML and AI workloads

Managing ML and AI workloads is important because of many aspects, some that are similar to other workloads and some that are unique to AI/ML. The importance is as follows:

- **Resource scaling and optimization:** Ensure that cloud resources can scale dynamically based on the workload requirements. Efficiently manage compute, storage, and network resources to handle varying demands without overspending. Use auto-scaling features in cloud services to adjust the number of virtual machines or containers based on the current load.
- **Cost management and budgeting:** Monitor and control the costs associated with running ML and AI workloads, including compute power, storage, and data transfer. Implement cost-management tools and practices to avoid unexpected expenses. Leverage cloud cost management tools to track and optimize spending on AI services and resources.

- **Data security and privacy:** Strong encryption methods protect sensitive data used for training and inference, both at rest and in transit. This ensures compliance with data protection regulations such as GDPR or CCPA. Encryption services and secure access controls safeguard data stored in cloud-based databases or storage services.
- **Model deployment and management:** Implement a robust deployment strategy for ML and AI models, including version control, testing, and rollback capabilities. Manage model lifecycle effectively to ensure consistent performance and reliability. Use model management tools to track different versions of models and facilitate easy deployment and rollback if needed.
- **Performance monitoring and optimization:** Continuously monitor the performance of ML and AI models to ensure they meet expected accuracy and efficiency. Optimize models and infrastructure to address performance bottlenecks. Employ monitoring tools to track model performance metrics and system resource usage and optimize algorithms or infrastructure based on the findings.
- **Data management and integration:** Effectively manage and integrate large datasets required for ML and AI workloads. Ensure data quality, consistency, and availability while integrating data from multiple sources. Use data lakes or cloud-based **Extract, Transform, Load** (**ETL**) services to manage and preprocess data before feeding it into ML models.
- **Compliance and regulatory requirements:** Ensure that ML and AI workloads adhere to relevant industry regulations and standards. Implement compliance checks and audit trails to meet legal and regulatory obligations. Incorporate compliance monitoring tools and practices to ensure adherence to regulations like HIPAA for healthcare data or PCI-DSS for payment information.
- **Model explainability and transparency:** Provide transparency into how ML and AI models make decisions to support accountability and trust. Implement tools and practices to explain model outputs and behavior to stakeholders. Use explainability frameworks or tools like **Shapley Additive Explanations** (**SHAP**) to provide insights into model predictions and feature importance.
- **Security and access control:** Implement strict security measures and access controls to protect ML and AI resources from unauthorized access or malicious activities. Regularly review and update security policies. Use **identity and access management** (**IAM**) policies to restrict access to ML models and datasets based on user roles and responsibilities.
- **Disaster recovery and business continuity:** Develop and test disaster recovery plans to ensure business continuity in case of failures or disruptions. Implement backup and recovery solutions for critical ML and AI resources. Set up automated backups for datasets and model checkpoints and establish recovery procedures to restore operations quickly in case of an outage.

- **Collaboration and communication:** Facilitate effective collaboration among data scientists, engineers, and other stakeholders involved in ML and AI projects. Use collaboration tools and practices to ensure smooth communication and coordination. Utilize cloud-based platforms that support collaborative development environments, such as Jupyter Notebooks or shared repositories, to enhance teamwork and project management.

AIOps

AIOps, or AI Operations, are patterns and practices you should consider and adopt in your daily ML/AI operations.

You must carefully consider ML/AI scenarios and the advantages and disadvantages of each approach to AI operations. When planning an approach, remember to consider service limits and data volumes, integrations, performance, types of models (simple or complex), costs, and your team's structure and experience.

To keep your ML/AI models running safely, securely, and accurately, you should observe some or all of the following patterns and practices:

- **Data integration and quality:** Integrate diverse data sources (logs, metrics, events) and ensure data quality before feeding it into AIOps platforms. Accurate and comprehensive data is crucial for effective anomaly detection and root cause analysis. Use data normalization and cleansing techniques to standardize log entries from different systems before analysis.
- **Define clear objectives and KPIs:** Set clear objectives and **key performance indicators** (**KPIs**) for what you aim to achieve with AIOps. This helps measure the effectiveness and ROI of AIOps implementations. Establish KPIs like reducing the **mean time to resolution** (**MTTR**) or the percentage of issues detected before they impact users.
- **Continuous model training and improvement:** Update and retrain machine learning models used in AIOps regularly to adapt to changing environments and patterns. This ensures that models remain accurate and effective over time. Implement automated retraining pipelines to periodically update models with the latest data.
- **Implement effective anomaly detection:** Use advanced algorithms and techniques to identify issues before they escalate. Employ supervised, unsupervised, or hybrid models depending on the context and data availability. Apply time-series analysis to detect unusual spikes in server metrics or network traffic.
- **Automate incident response:** Automate common incident response actions based on the insights provided by AIOps platforms. This reduces manual intervention and speeds up resolution times. Automatically restart services or adjust resource allocations in response to detected anomalies.

- **Ensure transparency and explainability:** Use AIOps tools that provide explanations and insights into their decision-making processes. This helps in understanding model predictions and fosters trust among IT teams. Implement explainable AI techniques such as SHAP or **Local Interpretable Model-Agnostic Explanations** (**LIME**) to make the reasoning behind anomaly detections more transparent.
- **Foster collaboration between teams:** Encourage collaboration between IT operations teams, data scientists, and developers to ensure that AIOps insights are actionable and aligned with operational goals. Create cross-functional teams that include members from IT operations, data science, and DevOps to jointly analyze and act on AIOps findings.

To avoid some of the pitfalls, be careful of some of the bad practices that will hinder your adoption of ML/AI Ops. The bad practices are as follows:

- **Ignoring data quality and cleanliness:** Using incomplete, inaccurate, or unclean data for AIOps can lead to unreliable insights and faulty decisions. Feeding raw, unfiltered logs from various sources without preprocessing can result in misleading anomaly detections.
- **Over-reliance on automation without oversight:** Relying too heavily on automated responses without adequate human oversight can lead to unintended consequences or missed issues. Automatically shutting down services in response to detected anomalies without validating the root cause can disrupt critical operations.
- **Lack of clear objectives or KPIs:** Implementing AIOps without clear objectives or KPIs can result in vague or unmeasurable outcomes, making it difficult to assess the value of the system. Deploying an AIOps platform without defined goals, such as reducing incident resolution time or improving system availability.
- **Neglecting model monitoring and maintenance:** Failing to monitor and maintain ML models used in AIOps can lead to model drift and decreased accuracy over time. Do not update or retrain models when there are significant changes in the IT environment or system behavior.
- **Overcomplicating systems and processes:** Making AIOps systems overly complex can hinder their effectiveness and usability. Complexity can lead to longer implementation times and increased maintenance efforts. Implementing overly complex algorithms or integrating too many data sources without a clear strategy can create confusion and reduce system performance.

ML/AI security

Focusing further on the security of ML/AI and some adversarial techniques that you should be aware of, here are just some of the recent issues widely reported:

- **Adversarial attacks on image recognition systems:** Adversarial attacks involve manipulating input data in a way that causes machine learning models to make incorrect predictions. These manipulations are often imperceptible to humans but can drastically alter the output of an AI model. Researchers have demonstrated that adding subtle noise to an image can cause image recognition systems to misclassify objects. For instance, a slight perturbation in a stop sign image could make an autonomous vehicle's AI system misinterpret it as a yield sign, posing significant safety risks.
- **Data poisoning attacks:** Data poisoning attacks occur when an attacker injects malicious data into the training dataset, leading to compromised model performance. This can result in models that make incorrect or biased predictions. In a real-world scenario, an attacker could poison a spam filter's training data with carefully crafted spam emails labeled as *not spam*. This would degrade the spam filter's accuracy, allowing more spam emails to bypass the filter.
- **Model inversion attacks:** Model inversion attacks enable attackers to reconstruct input data (such as sensitive information) from the model's outputs. This exploit leverages that some models inadvertently leak information about their training data. By querying a facial recognition system with various inputs, an attacker could reconstruct images of individuals from the model's training data, potentially revealing private information like identities and facial features.
- **Membership inference attacks:** Membership inference attacks allow adversaries to determine whether a particular data point was part of the model's training set. This can be a serious privacy concern, especially with sensitive datasets. If a hospital's predictive model for disease diagnosis is exposed to membership inference attacks, an attacker could infer whether an individual's medical record was used to train the model, thereby breaching patient confidentiality.
- **Model extraction attacks:** Model extraction attacks involve replicating a machine learning model by querying it and using the responses to build a copy. This can lead to intellectual property theft and financial losses. An attacker could repeatedly query a proprietary model offered as an online service, such as a financial prediction model, and use the outputs to train their own model, effectively stealing the original model's capabilities.
- **Backdoor attacks:** In backdoor attacks, malicious actors implant a backdoor into the model during the training phase. The backdoored model behaves normally under typical conditions but produces specific, incorrect outputs when triggered by specific inputs. For example, a backdoor attack on a voice recognition system could allow an attacker to control the system by saying a specific phrase, while regular users would not notice any abnormal behavior during typical use.
- **Trojan attacks:** Similar to backdoor attacks, trojan attacks involve embedding a trigger within the model that activates malicious behavior when a certain condition

is met. For example, a trojan attack on a self-driving car's AI system could cause the vehicle to malfunction when it encounters a particular visual pattern on the road, such as a specific graffiti tag.

- **Model skewing attacks:** Model skewing attacks involve manipulating input data in production environments to cause the model to perform poorly, leading to incorrect or biased results. An attacker could skew the input data to a credit scoring model to ensure that certain applications are either unfairly approved or denied, potentially causing financial losses or unfair treatment of individuals.
- **Algorithmic bias exploitation:** Exploiting preexisting biases in AI algorithms to achieve specific outcomes. Attackers can manipulate inputs to exploit these biases, resulting in skewed or harmful outputs. For example, an attacker could exploit bias in a hiring algorithm to systematically reject candidates from certain demographics by emphasizing specific characteristics in the input data.
- **Deepfake attacks:** Deepfake technology uses AI to create highly realistic but fake images, videos, or audio. These can be used for malicious purposes, such as spreading misinformation, impersonating individuals, or manipulating public opinion. Deepfake videos have been used to create fake news segments or impersonate public figures, leading to misinformation and potential political or social unrest.

The attacks discussed (and similar attacks) have recently caused the following:

- An airline to pay damages due to its chatbot's lies
- A magazine to publish an AI written articles
- A company held liable for its AI model rejecting CVs based on candidate age
- AI hallucinations of nonexistent court cases
- A real estate management company writing down millions and slashing the workforce due to an error in their AI management software
- Chatbot trained on public data spewing racist tweets

There are many avenues for ML/AI models to go wrong, and when they do, they do so at scale and with the tenacity of machines. Staying informed and in the loop about all ML/AI models is essential to stop/prevent these types of issues.

Conclusion

In this chapter, you have learned a lot about ML/AI, specifically various aspects of AI and ML, including their definitions for different audiences, security issues, views from C-suite executives, and best practices for cloud security. We highlighted good and bad practices in AIOps, recent AI exploits, and considerations for managing ML/AI workloads in the cloud.

Examples included adversarial attacks, data poisoning, and model extraction. Lastly, we outlined the strategic importance and challenges of deploying AI/ML technologies securely and effectively.

In the next chapter, we will address the future direction of security, especially cloud security, and how to use everything you have learned to prepare for any predictable or unpredictable events in the future.

Questions

1. How would you explain what ML and AI are to each of your C-suits?
2. What are the pros and cons of ML/AI cloud workloads?
3. What must you think about differently security-wise from other cloud workloads in terms of ML/AI models running in the cloud?
4. What security measures would you put in place to protect data during transit and storage for use in ML/AI?
5. How does your organization ensure the privacy and confidentiality of data used for training ML/AI models?
6. What protocols should you put in place to detect and mitigate potential security threats to ML/AI models?
7. How do you manage access control to the ML/AI models and the data they use?
8. What steps can you take to ensure the integrity of ML/AI models throughout their lifecycle?
9. How do you handle security patches and updates for the ML/AI infrastructure?
10. What measures can you take to protect against adversarial attacks on ML/AI models?
11. What incident response plan is in place for breaches or security incidents involving your ML/AI models?
12. How do you assess and mitigate bias and fairness issues in ML/AI models to ensure ethical deployment?
13. How do you ensure that ML/AI models are transparent and explainable to stakeholders?

References

- Google Cloud AI services: **https://cloud.google.com/products/ai**

- AWS AI services: **https://aws.amazon.com/ai/services/**
- Azure AI services: **https://azure.microsoft.com/en-us/solutions/ai**
- Microsoft responsible AI practices: **https://www.microsoft.com/en-us/ai/responsible-ai**
- SHAP documentation: **https://shap.readthedocs.io/en/latest/**
- LIME information **https://c3.ai/glossary/data-science/lime-local-interpretable-model-agnostic-explanations/**
- Jupyter: **https://jupyter.org/**
- NIST on AI: **https://www.nist.gov/artificial-intelligence**
- Famous AI disasters: **https://www.cio.com/article/190888/5-famous-analytics-and-ai-disasters.html**

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Chapter 18
Future of Cloud Security

Introduction

In this final chapter of the book, we will discuss the future of cloud security. This observation will always be correct: security will be an increasing concern at any time in the future.

So, in this chapter, let us discuss all the potential technologies, patterns and issues that we can anticipate even now. Of course, there will always be surprises; that is what makes cloud security such a fun field to be a part of.

Structure

The chapter covers the following topics:

- Patterns in the emerging technologies
- Future cloud security issues
- Be ready for anything
- Large Language Models
- Managing data influx
- Tools sprawl
- Top tips

Objectives

This chapter's objective is to give you hints (small and large) into the future direction of cloud security and what you will need to pay attention to as you continue deploying your organizations' workloads into the cloud. So, let us look at some topics and make sure you remember them, as they will come up repeatedly in the future.

Patterns in the emerging technologies

In this section, we will discuss some of the most important patterns in the emerging technologies of the future.

Zero trust

As cloud environments grow more complex and distributed, traditional perimeter-based security models become less effective. Zero trust security assumes no trust by default, even within the network, requiring continuous verification of user and device identities. This model is crucial for protecting cloud assets against sophisticated attacks and insider threats.

An explanation of the concept is as follows:

- Zero trust is a security model that assumes no implicit trust, even from within the network perimeter. It necessitates continuous verification of user identity and access privileges at each request, regardless of location or device. This model leverages principles such as least privilege access, micro-segmentation, and strong authentication mechanisms to minimize the risk of lateral movement and unauthorized access within a network.

For example, in 2023, security researchers discovered a vulnerability in Microsoft Azure AD that allowed attackers to forge authentication tokens. These tokens could be used to impersonate users and gain unauthorized access to various services and data within the Azure ecosystem.

Attackers identified a flaw in the way Azure AD handled token signing. By exploiting this flaw, they could forge tokens that appeared legitimate to Azure AD and other services relying on its authentication. Using the forged tokens, attackers could impersonate any user within the targeted Azure AD tenant, including highly privileged accounts. This allowed attackers to access sensitive data, applications, and services without proper authorization, effectively bypassing any zero trust mechanisms that were in place.

The forged tokens enabled unauthorized access to sensitive information, leading to potential data breaches. With control over privileged accounts, attackers could disrupt services, modify configurations, and cause significant operational issues. The incident eroded trust in Azure AD's security model, highlighting the challenges in implementing a true zero trust architecture.

The incident underscored the importance of continuous verification and monitoring of all access attempts, even from trusted sources. Ensuring the robustness of token handling and signing mechanisms is critical in a zero trust model to prevent similar vulnerabilities. A strong incident response plan, including rapid detection and remediation of unauthorized access, is essential to mitigate the impact of such violations.

This example highlights that even with a zero trust approach, vulnerabilities can still be exploited if the underlying systems are not rigorously secured. Continuous improvement, vigilant monitoring, and a proactive security stance are vital to maintaining the integrity of a zero trust architecture.

Confidential computing

Confidential computing involves processing encrypted data in memory without exposing it to the rest of the system. This is vital for protecting sensitive data in use and addressing concerns about data breaches and unauthorized access in multi-tenant cloud environments.

An explanation of the concept is as follows:

- Confidential computing protects data in use by performing computation within a hardware-based trusted execution environment. This ensures that data remains encrypted and isolated even during processing, providing robust protection against insider threats and advanced persistent threats that could potentially exploit vulnerabilities in the cloud infrastructure.

For example, in late 2022 and early 2023, researchers discovered a series of vulnerabilities in Intel's **Software Guard Extensions** (**SGX**), a technology designed to protect data during processing by creating secure enclaves. These vulnerabilities were exploited to extract sensitive data from within the supposedly secure enclaves.

Attackers used side-channel techniques to exploit the vulnerabilities. These attacks do not directly break the encryption but instead gather information from indirect sources, such as electromagnetic emissions, power consumption patterns, or timing information, to infer the data being processed. This type of side-channel attack allowed attackers to leak data from the SGX enclaves by exploiting flaws in the speculative execution process of Intel CPUs.

By carefully crafting their attacks, the researchers demonstrated that sensitive information could be extracted from the secure enclaves, undermining the core promise of confidential computing. The incident underscored the need for comprehensive threat modeling that includes potential side-channel attacks when designing confidential computing systems. The Intel SGX enclave vulnerabilities remind us that even advanced security technologies like confidential computing are vulnerable to sophisticated attacks.

ML/AI for threat detection

Artificial intelligence (**AI**) and **machine learning** (**ML**) can analyze vast amounts of data to detect anomalies and potential threats in real-time. These technologies are essential for identifying and responding to sophisticated cyberattacks that traditional security measures might miss, ensuring robust cloud security.

An explanation of the concept is as follows:

- ML and AI for threat detection involve leveraging advanced algorithms to analyze vast datasets for patterns indicative of malicious activities. By continuously learning from both benign and malicious behaviors, these systems can detect and respond to anomalies in real-time, significantly enhancing the efficacy of security operations centers in identifying and mitigating threats.

A recent example of using ML and AI for threat detection is the discovery of the SolarWinds supply chain attack. Although this attack was identified in 2020, its discovery heavily relied on advanced ML and AI-based threat detection techniques.

In 2020/2021, a sophisticated cyberattack was discovered in which hackers inserted malicious code into the software updates of SolarWinds' Orion IT management software. This allowed them to access the networks of numerous public and private sector organizations, including Fortune 500 companies and U.S. government agencies.

AI and ML algorithms were used to detect anomalies in network traffic and system behaviors. The attack involved sophisticated techniques to blend in with normal network activity, making traditional detection methods less effective. ML models trained on typical network behavior were able to identify subtle deviations indicative of malicious activity.

ML models analyzed user and entity behaviors to identify unusual patterns. For example, the attackers created new credentials and escalated privileges in ways that were atypical for legitimate users. AI-based behavioral analytics were crucial in flagging these activities as suspicious. AI systems integrated threat intelligence feeds and cross-referenced them with observed behaviors. This enabled a quicker correlation of the unusual activities associated with the malware with known indicators of compromise.

The SolarWinds attack serves as a prime example of how AI and ML can be effectively utilized for threat detection. These technologies enable the identification of sophisticated, stealthy threats through advanced anomaly detection, behavioral analysis, and integration of threat intelligence, thereby significantly enhancing the overall cybersecurity posture.

Quantum-safe cryptography

With the advent of quantum computing, current encryption methods could become obsolete. Quantum-safe cryptography aims to develop algorithms resistant to quantum attacks, ensuring data remains secure in the cloud as quantum computing becomes more prevalent.

Quantum computing harnesses the principles of quantum mechanics, particularly superposition and entanglement, to process information in fundamentally new ways. Unlike classical bits, which represent data as 0s or 1s, quantum bits (qubits) can exist simultaneously in multiple states (superposition). Additionally, entangled qubits are intrinsically linked, such that the state of one qubit instantaneously influences the state of another, regardless of distance. Quantum algorithms exploit these properties to solve certain classes of problems exponentially faster than classical algorithms. This holds transformative potential for fields such as cryptography, materials science, and complex system simulations.

An explanation of the concept is as follows:

- Quantum-safe computing involves developing cryptographic algorithms resilient to attacks from quantum computers, which can potentially break traditional cryptographic schemes. These quantum-resistant algorithms ensure the long-term security of data against future quantum adversaries by relying on mathematical problems that remain hard to solve even with quantum computational power.

IBM integrated quantum-safe algorithms, such as lattice-based cryptography, into its cloud services. These algorithms are designed to resist attacks from both classical and quantum computers.

IBM launched pilot programs for its clients to test quantum-safe cryptographic algorithms in their existing systems. This helped organizations understand the implications and benefits of transitioning to quantum-safe cryptography.

By integrating quantum-safe cryptographic algorithms, IBM's clients can enhance their security posture against future quantum threats, ensuring their data remains secure even as quantum computing advances. Early adoption of quantum-safe cryptographic algorithms is essential to ensure a smooth transition and to stay ahead of potential quantum threats.

Serverless, event-driven code

As serverless computing gains popularity, securing these environments becomes essential. Traditional security approaches may not apply, necessitating new strategies to protect serverless functions from vulnerabilities and ensure secure execution of workloads.

An explanation of the concept is as follows:

- Serverless event-driven computing abstracts the underlying infrastructure, allowing developers to deploy functions that automatically execute in response to specific events. This model provides scalable, cost-efficient code execution triggered by events such as HTTP requests, database changes, or messaging queues, optimizing resource utilization and reducing operational overhead.

For example, Amazon GuardDuty is a threat detection service that continuously monitors malicious activity and unauthorized behavior to protect AWS accounts and workloads.

By integrating AWS Lambda, a serverless computing service, organizations can create automated, event-driven responses to security findings detected by GuardDuty.

Amazon GuardDuty continuously analyzes streams of metadata for malicious or unauthorized activity using machine learning, anomaly detection, and integrated threat intelligence. When GuardDuty detects a potential security threat, it generates an alert. This alert can trigger an AWS Lambda function to automatically respond to the threat in real-time.

Security findings from GuardDuty are set up to trigger AWS Lambda functions. Upon receiving an alert, the Lambda function executes predefined actions such as isolating the threat, logging and auditing, and notifying and executing remediation actions.

The use of serverless event-driven code enables immediate response to security incidents, reducing the time window in which an attacker can cause damage. Automating incident response through serverless event-driven architecture significantly enhances the ability to manage and mitigate security threats swiftly. Seamless integration between detection tools like GuardDuty and response mechanisms like Lambda is essential for creating an effective security automation pipeline. Regularly updating and refining Lambda functions based on new threat intelligence and incident analysis is vital to keep the automated responses effective and relevant.

Distributed Ledger Technology for security

Distributed ledger technology (**DLT**), including blockchain, offers tamper-evident logging and decentralized trust mechanisms. In cloud security, it can provide immutable audit trails and secure data sharing, reducing the risk of tampering and ensuring data integrity.

An explanation of the concept is as follows:

- Distributed ledger technology enables decentralized and immutable recording of transactions across multiple nodes in a network. Each participant maintains a synchronized copy of the ledger, ensuring transparency, tamper-evidence, and consensus-driven updates. Blockchain, a type of DLT, underpins applications requiring secure, transparent, and auditable records, such as in financial services, supply chain management, and digital identity verification.

For example, in recent years, organizations have increasingly looked at blockchain technology to address the challenges of ensuring data integrity and secure access control in cloud environments. A notable implementation was carried out by a consortium of cloud service providers and cybersecurity firms, which utilized blockchain to create an immutable and transparent logging system.

Blockchain was used to create an immutable ledger of all access and modification logs. Every action taken on the data stored in the cloud, whether it is an access request, a data

ledger is used to provide a transparent and verifiable record of all actions. This made it possible to audit access and modifications without the risk of tampering, as each log entry was cryptographically linked to the previous one. The distributed nature of the blockchain ensured that no single entity had control over the entire log, thereby enhancing trust and security. This decentralized approach prevented any malicious insider from altering the logs without detection.

Smart contracts on the blockchain were used to enforce access control policies. These smart contracts ensured that only authorized users could access or modify specific data, and any attempt to breach these policies was immediately logged and flagged.

While blockchain provides security benefits, it can also introduce performance overhead. Careful design was needed to balance security and performance, particularly for real-time logging and monitoring.

Future cloud security issues

Let us now look at some of the cloud security issues already known to us that will prevail in the future and potentially become even more prevalent than today.

Data breaches

Unauthorized access to sensitive data stored in the cloud can result in significant financial and reputational damage. As data volumes grow, so do the attack surfaces, making data breaches a persistent and escalating threat. The impact is often a loss of sensitive information, regulatory fines, and a loss of customer trust.

For example, in 2023, Progress Software disclosed a critical vulnerability in its MOVEit Transfer file transfer software. This vulnerability was exploited by cybercriminals, leading to a widespread data breach affecting numerous organizations that relied on MOVEit Transfer for secure file transfers.

Attackers exploited a zero-day vulnerability in the MOVEit Transfer software, allowing them to gain unauthorized access to sensitive data stored within the affected systems. The vulnerability involved an SQL injection flaw, which allowed attackers to manipulate database queries and gain access to the underlying data. This type of attack enables malicious actors to exfiltrate data, execute commands, and potentially take over affected systems. Once inside the system, the attackers exfiltrated large volumes of sensitive data, including personal information, financial records, and confidential business documents.

The breach impacted a wide range of organizations, including government agencies, financial institutions, healthcare providers, and multinational corporations.

Affected organizations faced significant financial costs related to incident response, legal fees, and potential regulatory fines. The breach also led to class-action lawsuits and increased scrutiny from regulatory bodies.

The breach underscored the importance of robust third-party risk management practices. Organizations must regularly assess and monitor the security posture of third-party vendors and software providers.

Misconfigurations

Cloud environments are complex, and incorrect configurations can lead to vulnerabilities. Misconfigured storage buckets, inadequate access controls, and unpatched systems are common issues that can be exploited by attackers, resulting in data exposure, unauthorized access, and an increased risk of further attacks.

In 2023, it was discovered that a misconfiguration in Microsoft's Azure Blob Storage exposed sensitive information belonging to thousands of customers. This breach was particularly concerning because it affected a substantial amount of data stored in Microsoft's cloud service.

The root cause of the data exposure was a misconfigured Azure Blob Storage container that was inadvertently set to allow public access. This allowed anyone with the appropriate URL to access the data stored within the container without requiring authentication.

The misconfiguration was primarily due to human error, as most misconfigurations are during the setup or maintenance of the storage container. This highlights the risk of manual configurations in complex cloud environments.

A wide range of Microsoft Azure customers, from small businesses to large enterprises, were impacted by the breach. The breach affected customers across various sectors, including healthcare, finance, and technology.

The breach underscored the importance of stringent configuration management practices. Automated tools and regular audits can help detect and correct misconfigurations before they lead to data breaches.

Insider threats

Employees or contractors with access to cloud systems can intentionally or unintentionally cause security breaches. The rise of remote work increases the risk of insider threats due to less direct oversight.

This can result in data theft, sabotage, and/or accidental data loss.

In 2020, Tesla (an electric vehicle manufacturer) thwarted an insider threat when it discovered that a disgruntled employee had intentionally sabotaged part of the company's operations.

Tesla's internal monitoring systems detected malicious activity, which identified unusual network activity and unauthorized changes to critical systems. The company's security team quickly launched an investigation.

The incident could have led to substantial financial losses due to potential production halts and damage to Tesla's reputation. Prompt action helped mitigate these risks.

The incident highlighted the importance of continuous monitoring of employee activities, especially those with access to critical systems. Implementing robust monitoring tools can help detect unusual behavior early.

Insecure APIs

Application Programming Interfaces (**APIs**) are essential for cloud services but can be a weak link if not properly secured. Insecure APIs can be exploited to gain unauthorized access or manipulate data.

Service disruptions and unauthorized access resulting in data theft are the most common issues arising from insecure API design.

In 2021, a security researcher discovered a vulnerability in Peloton's API that exposed sensitive user data. Peloton is a popular fitness company known for its internet-connected exercise bikes and subscription-based workout programs.

The issue stemmed from insecure API endpoints that did not require proper authentication. This allowed anyone to access user account information by manipulating the API requests. The exposed API endpoints could be queried to retrieve the personal data of Peloton users, including user IDs, instructor IDs, group membership information, workout details, age, gender, weight, and more. Even if users had set their profiles to private, the data was still accessible due to the insecure API.

The exposed data included sensitive personal information, violating users' privacy. This raised concerns among users about how their personal information was being handled and protected. The incident attracted attention from regulatory bodies concerned with data protection and privacy, potentially leading to investigations and fines.

This incident underscored the importance of following API security best practices, including proper authentication and authorization mechanisms to ensure that only authorized users can access sensitive endpoints.

Compromised accounts

Attackers can gain control over cloud accounts through phishing, credential theft, or other means. Once compromised, they can access sensitive data, disrupt services, or conduct further attacks. This can result in a loss of control over your and your organization's cloud resources, data breaches, and financial loss (depending on the compromised accounts).

Lack of visibility and control

Organizations often struggle to maintain visibility and control over their cloud assets.

incident response. This can increase the risk of undetected breaches, compliance issues, and inefficient security management, making recovery even more difficult.

Compliance challenges

It is challenging to keep up with evolving regulations and ensure compliance in dynamic cloud environments. Non-compliance can result in significant penalties and legal consequences. Fines, legal action, and loss of certification are the likely results of non-compliance.

Distributed denial of service attacks

DDoS attacks aim to overwhelm cloud services, causing downtime and disruption. As services become more critical and interconnected, the impact of such attacks can be significant. Service outages, financial loss, and reputational damage usually follow DDoS attacks.

For example, in 2023, several major financial institutions in Denmark, including banks and insurance companies, experienced a large-scale DDoS attack. The attack led to significant disruptions in online banking services, affecting millions of customers.

The attackers launched a high-volume DDoS attack by flooding the targeted financial institutions' servers with excessive traffic. This overwhelming influx of data caused the servers to become unresponsive, denying legitimate users access to the services.

The attack utilized botnets, which are networks of compromised devices controlled by the attackers. These botnets sent massive amounts of traffic to the financial institutions' servers simultaneously.

The attack caused widespread service outages, leaving customers unable to access online banking services for several hours. These affected millions of individuals and businesses who rely on digital banking for their financial transactions. The inability to access banking services led to significant frustration and inconvenience for customers. Many took to social media to express their dissatisfaction and concern.

The incident highlighted the importance of having robust DDoS mitigation measures in place. Institutions need to deploy advanced technologies and strategies to detect and mitigate DDoS attacks quickly.

Zero-day vulnerabilities

As technology evolves, so do the threats. New and sophisticated attacks targeting cloud environments and previously unknown vulnerabilities (zero-day) can bypass existing security measures.

Compromised systems, data breaches, delayed response, and mitigation are the results of major zero-day vulnerabilities.

In 2023, Cisco disclosed a zero-day vulnerability in its IOS XR software, which is widely used in carrier-grade routers and network devices. The vulnerability was actively exploited in the wild, posing a significant threat to critical infrastructure.

The zero-day vulnerability resided in the Cisco IOS XR software's implementation of a specific network feature. It allowed attackers to execute arbitrary code or cause a **denial-of-service** (**DoS**) condition on the affected devices.

Exploiting this vulnerability, attackers could gain unauthorized access to network devices, potentially compromising the entire network infrastructure.

As a zero-day vulnerability, it was previously unknown to Cisco and had no available patch or workaround at the time of discovery.

The vulnerability affected a range of Cisco IOS XR devices used by **Internet Service Providers** (**ISPs**), data centers, and large enterprises. These devices are critical for managing network traffic and ensuring connectivity. Cisco promptly issued a security advisory detailing the vulnerability, its impact, and mitigation steps. This included temporary measures to reduce the risk of exploitation while a permanent fix was being developed. Within weeks of the disclosure, Cisco released software updates to patch the vulnerability. Customers were urged to apply these updates immediately to protect their networks from potential attacks.

Be ready for anything

Approaching a new security issue in cloud computing, especially in the face of uncertainty, requires a strategic and proactive mindset.

Ensure your organization has robust security policies and frameworks in place. This includes incident response plans, security best practices, and compliance with relevant standards. Continuously train and educate your staff on the latest security threats and best practices. Security is as much about people as it is about technology.

Subscribe to threat intelligence feeds and security bulletins to stay updated on the latest threats and vulnerabilities. Organizations like OWASP and various others provide valuable information. Implement advanced monitoring tools that use AI and machine learning to detect anomalies and potential threats in real-time.

Conduct regular security assessments, including vulnerability scans, penetration testing, and risk assessments, to identify and mitigate potential security gaps. Engage in red teaming (attack simulation) and blue teaming (defensive measures) exercises to test your security defenses and improve your response strategies.

Use automated security tools for tasks such as configuration management, compliance checking, and threat detection. Automation helps quickly identify and address security issues. Implement a Zero-Trust security model that continuously verifies and enforces security policies, regardless of where the request originates.

Form a dedicated incident response team equipped with the skills and tools to handle security incidents promptly. Regularly conduct incident response simulations and drills to ensure your team is prepared to act swiftly and effectively in the event of a security breach.

After resolving a security incident, conduct a thorough post-incident analysis to understand what happened, how it was addressed, and what can be improved. Create a feedback loop where lessons learned from incidents and new threats are integrated into your security policies, training programs, and technologies.

Participate in industry groups, forums, and cybersecurity communities to share knowledge and learn from others. Collaboration can lead to faster identification and mitigation of new threats. Maintain a strong partnership with your cloud service providers. They can provide insights, tools, and support to help address security issues effectively.

Invest in research and development to stay ahead of emerging threats. This includes exploring new security technologies, methodologies, and approaches. Run pilot programs to test new security solutions in a controlled environment before deploying them organization-wide.

Facing new security issues in cloud computing requires a blend of preparedness, adaptability, and continuous learning. By establishing a solid security foundation, staying informed, leveraging advanced technologies, and fostering a culture of continuous improvement, organizations can effectively tackle future security challenges.

Preparedness is not just about having the right tools but also about having the right strategies, teams, and mindset to respond to and mitigate the impact of security threats in an ever-evolving digital landscape.

Large Language Models

As the number one emerging technology discussed today, **Large Language Models** (**LLMs**)are already ubiquitous. LLMs, such as GPT-X, have become integral components in various cloud workloads, providing advanced capabilities in natural language processing, data analysis, and more.

However, their integration into cloud environments raises significant security concerns that must be addressed to ensure the protection of sensitive data and maintain operational integrity.

One primary security aspect involves securing the data used for training and inference. Ensuring that the data fed into LLMs is sanitized and devoid of sensitive information is crucial to prevent accidental data leaks. This can be achieved through stringent data governance policies and techniques such as differential privacy, which can help in obfuscating sensitive data while retaining its utility for training purposes.

Another critical security concern is protecting the LLM itself from unauthorized access

might interact with the LLM, robust access controls are essential. Implementing strong authentication mechanisms, such as multi-factor authentication and fine-grained access controls, can restrict access to the LLM, ensuring that only authorized personnel and applications can interact with it.

Additionally, continuous monitoring and logging of all interactions with the LLM can help detect and respond to suspicious activities promptly. This includes monitoring for unusual access patterns or attempts to inject malicious data that could compromise the model's integrity or functionality.

Lastly, addressing the potential misuse of LLMs is vital for maintaining security in cloud workloads. LLMs can generate human-like text, which could be exploited for malicious activities such as phishing, misinformation campaigns, or automated cyberattacks. Implementing usage policies that define acceptable use cases and deploying automated monitoring tools to detect and mitigate misuse are crucial steps in mitigating these risks.

Moreover, regular security assessments and updates to the LLM's underlying infrastructure can help identify and address vulnerabilities that attackers could exploit. By adopting a comprehensive security strategy that encompasses data protection, access control, and misuse prevention, organizations can leverage the power of LLMs in their cloud workloads while maintaining a robust security posture.

Managing data influx

The number one issue faced by organizations today, in general, is the overwhelming amount of data we all generate, collect, and act upon. Managing the influx of data in cloud workloads presents unique security challenges that require meticulous planning and robust strategies. As organizations increasingly move to cloud environments, they encounter a vast and continuous flow of data from various sources, including transactional systems, IoT devices, and user interactions.

This massive data influx can strain security systems, making it crucial to implement scalable solutions that ensure data integrity and confidentiality.

One effective approach is to use advanced data encryption techniques, both in transit and at rest, to protect data as it moves through and is stored within the cloud. Encryption helps safeguard sensitive information from unauthorized access and breaches, ensuring that even if data is intercepted or compromised, it remains unreadable without the appropriate decryption keys.

In addition to encryption, implementing strong access controls and data governance policies is essential for managing the security of large volumes of data. Role-based access controls and attribute-based access controls can restrict data access to authorized personnel based on their roles and attributes. This minimizes the risk of insider threats and accidental data exposure. Data governance frameworks should be established to define how data is classified, handled, and protected, ensuring compliance with regulations and standards.

Automated tools that enforce these policies and regularly audit data access and usage can help maintain security while handling the growing data volume efficiently.

Leveraging advanced analytics and machine learning tools can enhance security management by identifying and responding to potential threats in real-time. These tools can analyze vast amounts of data to detect anomalies, unusual patterns, or signs of potential breaches, allowing for swift intervention.

Implementing scalable data processing solutions, such as big data platforms with built-in security features, can further streamline the management of data influx. By combining encryption, access controls, data governance, and advanced analytics, organizations can effectively manage and secure large volumes of data within their cloud environments, ensuring both operational efficiency and robust security.

Tools sprawl

As a #1 issue teams managing the cloud environments are raising is the tool sprawl. The second one is an ever-increasing need for more and further education.

Let's focus on the tool sprawl.

Tool sprawl in cloud environments refers to the proliferation of security and management tools that organizations often adopt to address various aspects of their cloud infrastructure.

As cloud environments become more complex and diverse, teams may deploy a multitude of specialized tools to manage and secure different elements, such as network monitoring, data protection, identity management, and compliance.

This proliferation can lead to significant security challenges. Tool sprawl often results in fragmented visibility and inconsistent security policies, making it difficult to maintain a cohesive security posture. Each tool may have its own configuration, update cycle, and set of alerts, leading to potential gaps in coverage and increased risk of oversight.

Tool sprawl can create inefficiencies and complicate incident response efforts. When security incidents occur, having multiple tools with disparate data sources can delay the identification and remediation of threats.

Security teams may struggle to correlate information from various tools, resulting in slower response times and increased potential for damage. To mitigate this, organizations need to implement centralized management solutions and integration strategies that allow for better coordination and correlation of data across tools. Unified security platforms or Security Information and Event Management systems can aggregate data from different tools, providing a holistic view of the cloud environment and streamlining incident response processes.

Addressing tool sprawl also involves evaluating and optimizing the toolset regularly to ensure that each tool adds value and aligns with the organization's security strategy.

This requires conducting periodic reviews of the tool landscape, assessing their effectiveness, and retiring or consolidating redundant or underperforming tools.

Additionally, fostering a culture of standardization and ensuring that new tools are vetted for compatibility and integration with existing systems can help prevent further sprawl. By focusing on tool optimization and integration, organizations can enhance their security posture, reduce complexity, and ensure that their cloud security measures are both effective and manageable.

Top tips

Let us conclude with the top tips you can implement to protect your cloud workloads, be it today or in the future:

- Implement MFA to add an extra layer of security to user accounts and access to cloud resources.
- Ensure all data, both at rest and in transit, is encrypted using strong encryption algorithms.
- Keep all software, including cloud services and applications, up to date with the latest security patches.
- Grant users the minimum level of access necessary to perform their job functions.
- Ensure that APIs are securely configured and include proper authentication and authorization mechanisms.
- Continuously monitor and log all activities in your cloud environment to detect and respond to suspicious behavior.
- Conduct regular security audits and vulnerability assessments to identify and remediate potential security weaknesses.
- Use network segmentation to isolate sensitive workloads and limit the spread of potential breaches.
- Regularly backup data to prevent data loss from ransomware attacks or other incidents.
- Implement cloud security posture management tools to continuously monitor and improve your cloud security posture.
- Adopt a zero trust model in which no entity is trusted by default, even if it is inside your network.
- Conduct regular training sessions to educate employees about cloud security best practices and potential threats.
- Ensure all endpoints accessing cloud resources are secured with anti-malware, firewalls, and encryption.

- Use **Identity and Access Management** (**IAM**) tools to control and audit access to cloud resources.
- Use web application firewall to protect web applications from common attacks such as SQL injection and cross-site scripting.
- If you and your organization are using containers, implement container security solutions to scan for vulnerabilities and ensure secure configurations.
- Use automation tools to handle routine security tasks, such as patch management and compliance checks.
- Use data loss prevention solutions to monitor and protect sensitive data from unauthorized access and exfiltration.
- Continuously review and update your security policies to adapt to new threats and changes in your cloud environment.
- Develop and regularly update an incident response plan to quickly and effectively address security incidents when they occur.

Conclusion

As we move forward into an era defined by digital transformation, cloud security stands as a pivotal element in safeguarding the integrity, availability, and confidentiality of information.

The evolving landscape of cyber threats requires organizations to adopt a proactive stance, continuously evolving their security practices to address new vulnerabilities and attack vectors. The principles and technologies discussed throughout this book provide a robust framework for building resilient cloud infrastructures capable of withstanding the sophisticated threats of today and tomorrow.

Securing cloud environments is a continuous process that demands vigilance and, above all, adaptability. It is not merely about deploying the latest security tools but about fostering a culture of security awareness and resilience across all levels of an organization.

Collaboration between cloud service providers, businesses, and security professionals is essential to create a unified defense mechanism.

By effectively leveraging the shared responsibility model, organizations can ensure that technological and human elements work in harmony to fortify their cloud security posture.

The future of cloud security hinges on our ability (and yours and your organizations') to innovate, educate, and collaborate.

As new technologies emerge and the digital ecosystem expands, the strategies and insights provided in this book will serve as a guide for navigating the complexities of cloud security. By embracing these principles, organizations can not only protect their assets

but also unlock the full potential of cloud computing, driving growth and innovation in a secure and confident manner.

The path ahead is challenging, but with the right tools and mindset, it promises significant rewards and transformative possibilities. It also promises a long and fruitful career and, above all, a very interesting future for us all.

Question

1. If a new threat emerged tomorrow, one not discussed in this book, and one that impacts one or more of your and your organizations' workloads, how would you react? What would your course of action be? Who in your organization would help you, and which processes would hinder you?

References

Cloud Security Alliance (**CSA**) can be your first stop in dealing with the problems of the future in cloud security.

- "The CSA is the world's leading organization dedicated to defining and raising awareness of best practices to help ensure a secure cloud computing environment. CSA harnesses the subject matter expertise of industry practitioners, associations, governments, and its corporate and individual members to offer cloud security-specific research, education, certification, events, and products. CSA's activities, knowledge, and extensive network benefit the entire community impacted by the cloud — from providers and customers to governments, entrepreneurs, and the assurance industry — and provide a forum through which diverse parties can work together to create and maintain a trusted cloud ecosystem." **https://cloudsecurityalliance.org/**

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com



Index

Symbols

3rd party auditing 302
  challenges 304, 305
  preparation 304
  purpose 303
  types 303, 304

A

AI/ML workloads
  managing 337-339
AIOps 339, 340
AI security issues 336, 337
Amazon Chime 39
Amazon Elastic Block Store (EBS) 36
Amazon Elastic Compute Cloud (Amazon EC2) 36, 208
Amazon Elastic Container Service (ECS) 38, 213
Amazon GuardDuty 350
Amazon Relational Database Service (RDS) 38
Amazon Simple Storage Service (S3) 36, 38, 185
Amazon Virtual Private Cloud (VPC) 36
Amazon Web Services (AWS) 4
Amazon WorkMail 39
anonymization 76
Apigee 39
API security 176, 177
  alerting mechanisms 188
  API gateway 183
  API versioning 181
  audit logs 179
  authentication 177, 178
  authorization 178
  auto-scaling 188
  centralized logging 187

clear objectives, defining 186
cloud-native tools 189
cloud observability 184-186
collaboration and communication 189, 190
compliance monitoring 190, 191
continuous improvement 190
CORS 180
cost monitoring 189
CSP 181
data encryption 181
Denial-of-Service protection 182
dependency mapping 189
dependency scanning 182
distributed tracing 187
documentation 190
education and training 183, 191
error handling 179, 180
incident response plan 191
input validation 178
instrumentation 186, 187
load balancing 188
metrics collection 187
observability 184
parameterized queries 178
purple teaming APIs 183, 184
rate limiting 179
real-time monitoring 187
secure communication 178
secure file uploads 183
security headers 181
security monitoring 189
security testing 182
token management 179
user experience monitoring 190
application
backend 123-127
frontend 118
management and security services 128-131
Application Programming Interface (API) 176
application security
access patterns 116-118
high-level architecture 115, 116
scenario 114, 115
app service 118-122
artificial intelligence (AI) 245, 329
benefits 331-333
disadvantages 333
examples 330
recent prominence 331
Attribute-Based Access Control (ABAC) 160
authentication 11, 158
anti-patterns 159, 160
patterns 158, 159
authorization 11, 12, 160
anti-patterns 161, 162
patterns 160
automation 140, 258
common antipatterns 140, 141
common patterns 140
setting up 261
AWS containers 213-216
AWS Elastic Beanstalk 37
AWS IAM 11
AWS Inspector 289
AWS Lambda 38
AWS VM
instance, launching 208-211
Azure Active Directory (Azure AD) 11
Azure App Service 37
Azure Bastion 109

Azure Cosmos DB 225
Azure DDoS Network Protection 109
Azure DevOps 261
  automation, setting up 261-268
Azure Disk Storage 36
Azure Firewall 109
Azure Functions 37
Azure Kubernetes Service (AKS) 37, 224
Azure Logic Apps 37, 225
Azure Privileged Identity Management 83
  setting up 83-89
Azure SQL Database 37
Azure Storage 225
Azure Virtual Machines 35
Azure Virtual Network 35

B

Burp Suite 290
business continuity plan 69, 70

C

caching 253-255
Capex 27
Certificate Authority (CA) 104
Chief Executive Officer (CEO) 334
Chief Financial Officer (CFO) 334
Chief Information Officer (CIO) 335
Chief Technical Office (CTO) 334
CI/CD 258
cloud computing 2
  addition to definition 7, 8
  advantages 4
  community definition 2
  computing options 3, 4
  cost optimization 5
  operational excellence 5, 6
  performance efficiency 6
  reliability 5
  service models 6, 7
Cloud Controls Matrix 284
cloud forensics 280
  log analysis 280
  memory acquisition 280
  virtual machine introspection 281
Cloud Functions 225
  managing 226-233
CloudGoat 290
cloud infrastructure security 68
Cloud Interconnect 105
Cloud Run 225
cloud security 12-15, 310, 335
  AI security issues 336, 337
  DevOps 310
  DevSecOps 311
  ML security issues 335, 336
  patterns and anti-patterns 313-317
  Platform Engineering 311, 312
  Site Reliability Engineering 312
Cloud Security Alliance (CSA) 284
cloud service provider (CSP) 164
CloudSploit 289
compliance 163, 294
  anti-patterns 164
  common standards and regulations 296, 297
  enhanced security and risk management 295
  frameworks and standards 295
  industry-specific compliance requirements 297
  legal and financial protection 294
  patterns 164
  regulatory bodies 295, 296
  reputation and trust 295
compliance and security, balancing 297, 298

concept of best-effort security 299
concept of more secure 298, 299
events, auditing 300
internal auditing, best practices 302
scope of internal auditing 300
security events 300, 301
tools 301
compliance portals 305
AWS 306
Azure 305
GCP 306
containers 211-213
AWS containers 213-216
orchestration 216
Content Delivery Networks (CDNs) 32
Content Security Policy (CSP) 181
continuous development patterns 260, 261
continuous integration patterns 259, 260
convolutional neural network (CNN) 329
Cross-Origin Resource Sharing (CORS) 180
cross site scripting (XSS) 101
C-suites 333
Chief Executive Officer (CEO) 334
Chief Financial Officer (CFO) 334
Chief Information Officer (CIO) 335
Chief Technical Office (CTO) 334, 335
cybersecurity 8, 9
availability 10
confidentiality 10
importance 16-18
integrity 10
cybersecurity best practices 57
compliance 59
data protection and privacy 57
employee training 59, 60
incident response and contingency planning 58, 59
physical security 58
real-world examples 60, 61
third-party due diligence 57
transparency 58

D

data influx
managing 357, 358
data masking 76
data privacy 162
anti-patterns 163
patterns 162
Data Protection Impact Assessments (DPIAs) 162
data residency and compliance 14
data sanitization 75, 76
deployment models 40
hybrid cloud 41
multi cloud 42, 43
private cloud 40
public cloud 40
DevOps 310
DevSecOps 311
Direct Connect 105
Distributed Denial of Service (DDoS) attack 5, 23
Distributed ledger technology (DLT) 350
DNS Zone service 106
Domain Name System (DNS) 96
Dynamics 365 39

E

electronic discovery (eDiscovery) 281
emerging technologies, patterns

DLT 350, 351
ML / AI for threat detection 348
quantum-safe cryptography 348, 349
serverless event-driven code 349, 350
zero trust 346, 347
encryption 70-72, 139
common antipatterns 139, 140
patterns 139
equipment disposal 76, 77
Express Route 105
eXtensible Access Control Markup Language (XACML) 161
external auditing 302
compliance audits 304
financial audits 303
IT audits 303
operational audits 303
External Identity Providers (IDPS) 11

F

file integrity monitoring (FIM) 196
forensics 280
cloud forensics 280
collaboration, with cloud providers 282, 283
data volatility 283
evidence collection and preservation 281, 282
future cloud security issues 355, 356
compliance challenges 354
compromised accounts 353
data breaches 351
DDoS attacks 354
insider APIs 353
insider threats 352, 353
lack of visibility and control 353
misconfigurations 352
zero-day vulnerabilities 354, 355

G

General Data Protection Regulation (GDPR) 14, 157, 165, 281
anti-patterns 167
aspects 165, 166
patterns 166, 167
Google App Engine 38
Google Cloud Functions 38
Google Cloud IAM 11
Google Cloud Platform (GCP) 4, 77
Google Cloud Platform IAM 79-82
Google Cloud Run 38
Google Cloud SQL 38
Google Cloud Storage 36
Google Cloud VPC 36
Google Compute Engine 36
Google Kubernetes Engine (GKE) 38
Google Persistent Disk 36
Google Workspace 39
governance 143
governance, principles
automation 140
encryption 139
identity 136
layers 138
monitoring 141

H

Health Insurance Portability and Accounting Act (HIPAA) 14
HTTP Strict-Transport-Security (HSTS) 181
hybrid cloud
characteristics 41, 42
Hyper Text Transfer Protocol Secure (HTTPS) 178

I

identity

common antipatterns 137
patterns 136, 137
Identity and Access Management (IAM) 11, 77
incident response 274, 275
challenges 279, 280
incident response lifecycle 275
containment phase 276, 277
eradication phase 277, 278
identification phase 276
preparation phase 275, 276
recovery phase 278
incident response plan 192
assets, identifying 192
collaboration, with external entities 197
communication plan 193
continuous improvement 197
documentation 194
escalation procedures 194
evidence handling 196
forensic readiness 196
incident analysis 197
incident categorization 194
incident containment 198
incident identification 194
incident reporting 195
incident response playbooks 195
IRT team, creating 193
legal and compliance considerations 194, 195
public relations strategy 198, 199
risk assessment 192
training and drills 193
vendor management 199
Incident Response Team (IRT) 193
Infrastructure as a Service (IaaS) 35
integration Platform as a Service (iPaaS) 225
Intel's Software Guard Extensions (SGX) 347
International Organization for Standardization (ISO) 98
Internet Protocol (IP) 95, 241
Intrusion Detection System (IDS) 138

J

JavaScript Object Notation (JSON) 123
Just-In-Time (JIT) 67

K

Kubernetes (k8s) 217
Kusto Query Language (KQL) 249

L

Large Language Models (LLMs) 356, 357
layers 138
common antipatterns 138, 139

M

machine learning (ML) 328, 329
common examples 329, 330
Media Access Control (MAC) 101
Microsoft 365 39
Microsoft Defender for Cloud 289
Microsoft SDL practices 152, 153
migration 217
on-premises to containers 218, 219
on-premises to VMs 217, 218
MITRE ATT&CK® framework 74
ML/AI security 340-342
ML security issues 335, 336
monitoring 141
common antipatterns 141, 142
multi cloud 42, 43

N

Network Address Translation (NAT) 101
networking, history 94
  global reach 96, 97
  issues 97, 98
  precursors 94-96
networking patterns 105
  Bastion 108
  DDoS protection 108
  direct connection 105, 106
  DNS 106, 107
  private link 108
  subnets 108-111
Network Interface Controller (NIC) 101
Nmap 290
Non-TLS port option 255

O

OAuth 2.0 161
Operational Security (OpSec) 72-75
Opex 27
orchestration 216
orchestration engines
  features 216, 217
OSI model 98, 99
  Azure services across layers 102
  certificate exchange 104
  cloud technologies 103
  data encryption 105
  data integrity 105
  handshake 104
  key exchange 104
  layers 99, 100
  layer security 100, 101
  SSL/TLS 104

P

Pacu 289
pay-as-you-go 25
Payment Card Industry Data Security Standard (PCI DSS) 59
penetration testing 286
  as assessment tool 287, 288
  tools 289, 290
Personally Identifiable Information (PII) 31
Platform as a Service (PaaS) 36
Platform Engineering 311, 318-320
principle of least privilege 78
private cloud
  characteristics 40, 41
public cloud model
  characteristics 40
public relations (PR) strategy 198

Q

Quality of Service (QoS) 32
quantum-safe cryptography 348

R

real-user monitoring (RUM) 190
Recovery Point Objectives (RPOs) 28
Recovery Time Objectives (RTOs) 28
redundancy and availability 249
reporting 288
response 142
  common antipatterns 143
  common patterns 142, 143
robotic process automation (RPA) 330
Role-Based Access Control (RBAC) 11, 77, 160

S

Scout Suite 290
security 8

cybersecurity 8, 9
security assessment 283
best practices 285
Cloud Controls Matrix 284
shared responsibility model 284
vulnerability management challenges 284, 285
Security Development Lifecycle (SDL) 152
security governance 143, 144
risk management 144, 145
Security Information and Event Management (SIEM) 138, 276
serverless 221, 222
security 233-236
versus, non-serverless 223, 224
serverless services 224
AWS 224
Azure 224, 225
GCP 225
service models 22
data transfer 32
elasticity 24, 25
high availability 28-30
IaaS vs. PaaS vs. SaaS 34
Infrastructure as a Service (IaaS) 35, 36
marketplace 33, 34
network access 25
pay-as-you-go 25, 26
Platform as a Service (PaaS) 36, 37
privacy 30, 31
resource pooling 23, 24
self-service 22, 23
virtualization 27, 28
shared responsibility 51,-53
shared responsibility model 284
short term storage options 250-253
Single Sign On (SSO) 158
Site Reliability Engineering 312, 320-323
Site Reliability Engineering (SRE) team 141
Software as a Service (SaaS) 38, 39
software deployment 63
real-world examples 63
software development 61, 62
real-world examples 62
stackTrace field 180
storage 246, 247
capabilities 247-249
subnets
security 245
supply chain 54
risk management 56
vendor selection 55
System and Organization Controls 2 (SOC 2) 168
anti-patterns 172, 173
key aspects 169
patterns 172
pillars 170-172
principles 168
report types 169

T

tenant separation 14
threat intelligence 268
threat modeling 146-152, 269-271
importance 286, 287
Tool sprawl 358
Transmission Control Protocol (TCP) 95
trust 46
definition 46-48
positives and negatives 49-51
zero trust 48, 49

U
use cases
  DevOps 323
  DevSecOps 323
  Platform Engineering 324
  Site Reliability Engineering 324
UserController class 180
user experience monitoring (UEM) 190

V
virtualization 206
Virtual Local Area Network (VLAN) 101
virtual machines 206
  security practices 206-208
virtual network (VNET) 240
  address space 240, 241
  best practices 242-245
  planning 245, 246
  subnet 241
virtual private networks (VPNs) 103
VMs and containers
  differences 205
  similarities 204, 205

W
Web Application Firewall (WAF) 60

Z
zero trust 48, 49, 79