Multi-Cloud Administration Guide

Manage and optimize cloud resources across Azure, AWS, GCP, and Alibaba Cloud

Jeroen Mulder

bpb

Multi-Cloud Administration Guide

Manage and optimize cloud resources across Azure, AWS, GCP, and Alibaba Cloud

Jeroen Mulder

www.bpbonline.com

Copyright © 2024 BPB Online

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor BPB Online or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

BPB Online has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, BPB Online cannot guarantee the accuracy of this information.

First published: 2024

Published by BPB Online

WeWork

119 Marylebone Road

London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55515-551

www.bpbonline.com

Dedicated to

My wonderful wife Judith
and
My girls, Rosalie and Noa

About the Author

After his study in Journalism, Jeroen Mulder (born in 1970) started his career as editor for the economic pages of Dutch agricultural newspapers. From 1998, he got involved in internet projects for Reed Business Information, in creating websites and digital platforms. Highly attracted by the possibilities of the new digital era and the booming business of internet, Jeroen decided to pursue a career in digital technologies. In 2000, he joined the IT company Origin, as communication specialist in a group designing and developing cross media platforms. Origin evolved to AtoS where he fulfilled many roles, lastly as principal architect.

In 2017, he joined the Japanese IT services company Fujitsu as Senior Lead Architect, with focus on cloud and cloud native technology. From May 2020, he held the position of Head of Applications and Multi-Cloud Services for Fujitsu Netherlands, leading a group of frontrunners in digital transformation. Then Philips knocked on his door. In 2021 and 2022, he was assigned Principal Cloud Architect at Philips Precision Diagnosis. Yet, Jeroen returned to Fujitsu in the summer of 2022 as Principal Consultant for the Global Technology Solutions Business Group, focusing on hybrid IT.

About the Reviewer

Fouad Mulla is a seasoned Lead Consultant, Digital Leader, and Cloud Security Architect with 15 years of professional experience in the digital and software industry at global corporations. Fouad excels in designing and implementing comprehensive cloud solutions across multi-cloud platforms. He has assisted numerous businesses in effectively governing and safeguarding their information, proactively identifying cybersecurity risks, and enabling them to make informed and strategic business decisions. Fouad is CISSP, CISM, CASP+ certified.

Acknowledgements

First of all, I have to express my deepest gratitude to my beloved and wonderful wife, my girls and my entire family and dearest friends. This has been a difficult year with health issues whilst planning for the move to a new house. Thank you for standing next to me, even when I have not been the best version of me. I am truly sorry.

Next, I also have to thank my employer Fujitsu for granting all the time I needed to get well again. And thank you all at BPB for all the patience.

And of course: a big thank goes to you, dear followers and readers. You're making all the effort worthwhile.

Preface

Cloud first is not a strategy, but a statement at best. For starters, cloud first does not say anything about your business strategy. It does not say anything about the goals you want to achieve by using cloud technology. Yet, almost every company on earth is using the cloud in some form. And then the trouble starts: how do we manage our workloads in the cloud? A lot of companies find out the hard way that managing cloud is something different from managing the more traditional IT. That is what this book is about: managing workloads in multi-cloud.

In this book, the reader will get guidance and hands-on instruction on operating multi-cloud environments. We will discuss all the different aspects that come with mulit-cloud such as interoperability between different cloud environments, networking configuration, data integration, and of course security requirements. There is no way of talking about cloud without addressing security and compliance.

This book provides new adopters of the multi-cloud approach with numerous frameworks and ideas for an efficient and sound multi-cloud infrastructure. Throughout the book, you will hopefully find solutions, techniques, designs, and administrative guidance for various types of multi-cloud environments, using AWS, Azure, GCP and Alibaba Cloud. Why these? Because these are the most used clouds.

This book will hopefully help you in understanding the necessary steps in multi-cloud management. Let us now go over the chapters in the book.

Chapter 1: Using the Cloud Adoption Frameworks - provides an overview of the various Cloud Adoption Frameworks (CAF) that help in setting up and manage environments in AWS, Azure, GCP and Alibaba Cloud. CAFs contain pillars such as security, identity and access, cost and governance. We will discuss the CAFs of the major providers and show readers how to use them to get maximum benefit.

Chapter 2: Virtualizing and Managing Connectivity - covers all aspects about connectivity in cloud. Networking in cloud is software based. In this chapter, you will learn what cloud networking is, with guidance to software building blocks, deployment models and operating networks in cloud. Concepts such as SD-WAN, SD-LAN and edge will be discussed. We will also introduce micro-services and how network virtualization can help with this.

Chapter 3: Virtualizing and Managing Storage - explains one of the major benefits of cloud: the almost limitless amount of storage that is available to us. But cloud is a shared resource model, so we need to understand how storage works in cloud, how to make sure that we get the right amount of storage and the right type, with the right performance. Cloud architects and admins should know about I/O, pooling and the different storage types. All of that is covered in this chapter.

Chapter 4: Virtualizing and Managing Compute - covers the basics of compute, starting with virtual machines, but also discussing serverless and containers as cloud-native technologies. Public cloud providers offer a wide variety in compute power with different deployment models. We

need to understand how compute in cloud works regarding for instance CPU and memory. In this chapter, we will also look at on premises offerings by major cloud providers.

Chapter 5: Creating Interoperability - explains why interoperability is one of the biggest challenges in multi-cloud. In this chapter, you will learn how to overcome these challenges between public clouds and between public and private clouds. The chapter discuss various solutions and frameworks such as Open Compute.

Chapter 6: Managing Data in Multi-Cloud - starts with defining a data strategy in multi-cloud. We need that strategy to determine where data is stored, who and what may access data, what the usage of data is as well as how to prevent events such as data-leaks and data loss. You will learn about data quality, security and integrity and data gravity.

Chapter 7: Build and Operate Cloud Native - further explores cloud-native technologies such as serverless and containers. Cloud native development offers solutions to create scalable applications using, for instance, micro-services, container and serverless concepts, and deploying declarative code. This chapter also contains an in-depth explanation of setting up micro-services architectures and how to manage these.

Chapter 8: Building Agnostic with Containers - covers all aspects of developing and managing containers in cloud platforms. Since Kubernetes has evolved to become the industry-standard, we will study setting up and managing Kubernetes clusters with Docker containers in more detail. Next, we will explore the requirements to monitor our containerized workloads.

Chapter 9: Building and Managing Serverless - helps in understanding the concept of serverless, providing guidance in developing and provisioning serverless functions. In this chapter, we will learn how to define our environments as functions, that we can deploy and manage as serverless environments.

Chapter 10: Managing Access Management - introduces the cornerstone of security in multi-cloud: access management. We do not want just anyone to be able to access data and applications in the cloud; we want to have control and hence, we need access management. The chapter provides an overview of various tools that we can use in multi-cloud, but also addresses concepts as privileged access management and Identity as a Service.

Chapter 11: Managing Security - discusses the principe of the single pane of glass view to monitor and manage security. With distributed environments, we must use frameworks that cover the various cloud technologies and tools that can handle various clouds.

Chapter 12: Automating Compliancy - starts with explaining why compliance in cloud might be more of a challenge than in traditional IT, where we have workloads mostly on premise. Governmental bodies, certification authorities and auditors are setting compliancy guardrails to allow usage of major cloud providers. In this chapter, we will learn how to use automation and even AI to ensure that our workloads in the cloud remain compliant.

Code Bundle and Coloured Images

Please follow the link to download the

Code Bundle and the Coloured Images of the book:

https://rebrand.ly/x7mjh27

The code bundle for the book is also hosted on GitHub at In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at 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 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 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

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. Using the Cloud Adoption Frameworks

Introduction

Structure

Exploring the business challenge of multi-cloud

Introducing CAF: how to use them

Strategy

Plan

Prepare

Adopt

Govern

Operate

Deep dive in the CAF of Azure and AWS

Frameworks by GCP and Alibaba cloud

Lead

Learn

Scale

Secure

Similarities and differences

Monitoring multi-cloud and keeping track of value propositions

Measuring business metrics

Introducing best practices in monitoring

Conclusion

Points to remember

Questions

Answers

2. Virtualizing and Managing Connectivity

Introduction

Structure

Building blocks in networks

Virtual switching

Routing

Virtual firewalls

Load balancing

Cloud network operations

Deployment concepts SD-WAN, Edge, and SASE

Managing networks in multi-cloud

Setting up VPN

Setting up direct connections

Conclusion

Points to remember

Questions

Answers

3. Virtualizing and Managing Storage

Introduction

Structure

Types of storage virtualization

Storage concepts in public clouds

Managing storage assets in multi-cloud

Protecting data from data loss

Using third-party storage products

Managing storage lifecycles and tiering

Automating tiering

Automating scaling of storage

Managing data access

Conclusion

Points to remember

Questions

Answers

4. Virtualizing and Managing Compute

Introduction

Structure

Comparing compute models in public clouds

Key considerations for choosing compute

Rightsizing compute in deployment and autoscaling

Rightsizing using autoscaling

Exploring on premises propositions of public cloud

Azure Stack and Azure Arc

AWS outposts

Google Anthos

Extending VMWare

Deploy and manage compute assets

Automating infrastructure management

Conclusion

Points to remember

Questions

Answers

5. Creating Interoperability

Introduction

Structure

Defining interoperability

Requirements for interoperability

Explaining the difference between portability and interoperability

Solutions to create interoperability in public and hybrid clouds

Working with Open Compute Project

Conclusion

Points to remember

Questions

Answers

6. Managing Data in Multi-Cloud

Introduction

Structure

Defining a data strategy

Planning data migration

Managing data governance

Improving data quality

Data quality in AWS

Data quality in Azure

Data quality in GCP

Data quality in Alibaba Cloud

Securing data

Conclusion

Points to remember

Questions

Answers

7. Build and Operate Cloud Native

Introduction

Structure

Understanding cloud-native concepts

Organizing cloud-native with DevOps

Explaining micro-services

Managing releases in micro-services

Conclusion

Points to remember

Questions

Answers

8. Building Agnostic with Containers

Introduction

Structure

Understanding container technology

Pitfalls and risks of container technology

Container services from major cloud providers

Developing and provisioning containers using industry standards

Guided plan to develop and deploy containers

Exploring container management using a single pane of glass view

Managing security in container platforms

Deep dive into container monitoring and log management

Collecting and analyzing logs

Conclusion

Points to remember

Questions

Answers

9. Building and Managing Serverless

Introduction

Structure

Understanding the serverless concept

Developing and provisioning serverless functions from architecture

Using CI/CD for serverless deployments

Managing multi-cloud environments with serverless frameworks

Following best practices in serverless

Deep dive into monitoring serverless

Conclusion

Points to remember

Questions

Answers

10. Managing Access Management

Introduction

Structure

Exploring the basics of access management

Understanding managed identities

Challenges in access management

Using cloud tools for access management

Understanding and working with PAM

Privileged session and privileged behavior analytics

Creating and storing secrets

Managing secrets and keys in AWS and Azure

Managing secrets and keys in GCP and Alibaba Cloud

Avoiding pitfalls in managing secrets and keys

Defining, implementing, and managing Role Based Access Control

Monitoring access control

Conclusion

Points to remember

Questions

Answers

11. Managing Security

Introduction

Structure

Working with cloud security frameworks

Example of implementing CIS guidelines for Azure

Choosing the security tools

Managing security through one lens

Introducing Cloud Security Posture Management

Keeping track and up to date with security trends

Conclusion

Points to remember

Questions

Answers

12. Automating Compliancy

Introduction

Structure

Understanding compliance in multi-cloud

Automating governance

Using RPA for automating compliance

The next step: using AI for compliance checking

Conclusion

Points to remember

Questions

Answers

Index

CHAPTER 1

Using the Cloud Adoption Frameworks

Introduction

Welcome to the cloud. Or better said: welcome to the multi-cloud. The major public cloud providers, such as Azure and AWS, offer Cloud Adoption Frameworks to help customers set up and manage environments in their clouds. Their usage should be secure and efficient. CAFs are good guidance for architects and engineers. These frameworks contain pillars such as security, identity and access, cost, and governance.

In this chapter, we will first discuss what multi-cloud is and next study the CAFs of the major providers, showing how to use them to get maximum benefit. We will also discuss monitoring, including keeping track of (business) Key Performance Indicators At the end of the day, we should be creating value from our cloud, and value needs to be measured.

Structure

In this chapter, we will discuss the following topics:

Exploring the business challenges of multi-cloud

Introducing CAF: how to use them

Deep dive in the CAF of Azure and AWS

Frameworks by GCP and Alibaba cloud

Similarities and differences

Monitoring multi-cloud and keeping track of value propositions

Exploring the business challenge of multi-cloud

Before we dive into the challenges of multi-cloud, we must define what multi-cloud is. Multi-cloud refers to the practice of using multiple cloud service providers to distribute an organization's computing resources and applications. By leveraging the strengths of different cloud platforms, such as Amazon Web Services Microsoft Azure, Google Cloud Platform and others, businesses can optimize their IT infrastructure for performance, cost, security, and scalability.

The rise of multi-cloud strategies has become an important topic in today's IT landscape for several reasons:

Flexibility and avoiding vendor Utilizing multiple cloud providers allows organizations to prevent reliance on a single vendor, offering them the flexibility to choose the best services and pricing structures for their specific needs.

Optimal resource Each cloud provider has unique strengths and weaknesses. A multi-cloud approach enables organizations to allocate resources based on the specific capabilities of each platform, ensuring optimal performance and cost-effectiveness.

Enhanced security and Distributing data and applications across multiple cloud environments can help organizations reduce the risk of data breaches, meet regulatory requirements, and adhere to industry standards.

Increased resilience and A multi-cloud strategy can improve business continuity by providing redundancy in case of outages or failures in a single cloud environment. This ensures that critical applications and data remain available and operational.

Innovation and competitive Leveraging multiple cloud platforms allows organizations to access cutting-edge technologies and tools, fostering innovation and providing a competitive edge in the market.

Most companies are multi-cloud, even when they have a single cloud strategy. The staff will work with Office365 of Microsoft, store customer contacts in Salesforce, the book travels through SAP Concur, and have meetings through Zoom. At the same time, the backend systems of companies might be hosted on a public cloud such as AWS or Azure or on servers in a privately owned data center. Thus, companies use Software as a Service Platform as a Service and Infrastructure as a Service and all these different environments must be managed. This is the IT challenge of multi-cloud.

Multi-cloud strategies are motivated by the desire to optimize IT infrastructure using different cloud platforms' unique strengths. Benefits include flexibility, optimal resource allocation, enhanced security and compliance, increased resilience and redundancy, and access to innovative technologies. These advantages make multi-cloud strategies relevant and valuable in today's competitive digital landscape.

But what is the business challenge of multi-cloud? Amongst others, we can think of the following:

Cloud sprawl: We discuss cloud sprawl when a company lacks visibility into and control over the spread of its environments in various clouds, including instances, services, or providers across the company.

Lock-in, including data gravity: A mistake that companies often make is assuming that multi-cloud decreases the risk of lock-in. That risk still exists, but now it is spread over multiple clouds. This risk is directly related to portability. It is not as easy as it seems to migrate native services across different clouds. Next, the issue of data gravity plays an important role. Applications often need to be close to the data. Having data sitting in a different cloud than the application may lead to issues such as latency. Moreover, rules for compliancy can cause issues. Think of laws that prohibit companies from having data outside country borders, limiting the choice of clouds.

Lack of multilingual knowledge: If a company uses various clouds, it also means that it has to know how to use these clouds. Although the principles of public cloud are largely the same, clouds such as AWS and Azure still do differ in terms of operating workloads on these platforms. The company will need resources, engineers, and architects to cover the different technologies used.

Dynamics of changing cloud features: Cloud is evolving fast. During the yearly large conferences such as Ignite for Azure and re: Invent for AWS, these providers launch hundreds of new services. Over the year, even more, new features and services are added to the portfolio. Not everything might be of use to a company, but it needs to keep track of features and releases of new services to be able to improve its own cloud environments. This is not trivial, and certainly not when a company is

operating multi-cloud. In most cases, cloud providers will help their customers in getting the best out of the cloud by adopting the right technologies.

Integration: Using environments on different platforms might lead to integration issues simply because workloads cannot communicate with each other. This can be due to network issues such as bad routing and because technologies are not compatible.

Of course, there are many more challenges to overcome. Think of network performance and latency, security and compliance, governance, and policy management, not to mention controlling costs and the cloud vendor relationship as part of the governance. All these items are captured in the cloud adoption frameworks. During the course of this book, we will discuss these items in more detail.

Following best practices and guidelines from CAFs can help to at least address these issues and design solutions to overcome them.

Introducing CAF: how to use them

What is a cloud adoption framework, and how should we use it? Maybe a better first question would be: why would we use a CAF? The answer to that question is: because the CAF will help as long as we follow the guidelines and guardrails as defined in the CAF, it will be a lot easier to get support from the cloud providers when we encounter issues. It is fair to say that the CAF provides a universal language between the cloud provider and the customer. The CAF is basically a set of documentation with guidelines and best practices on how we can best design and operate our cloud.

Before we dive into the details of the CAF for Azure and AWS, which are the leading public clouds, we will study the generic pillars of the CAF. The six pillars of the CAF are as follows.

Strategy

Moving to the cloud just because you can, is not a strategy. Cloud first, for that matter, is not a strategy. Using cloud technology should be valuable to a business. This means that there must be a business justification. We will discuss this in the section about similarities and differences between the various CAFs. A business will have an ambition laid out in business goals. The next step is to define how the business can achieve those goals and, in the end, fulfill the ambition. The architecture will lay out what the ambition will look like (sometimes referred to as the North Star architecture), but more importantly, how to reach the goals. What steps must a business take, and in what order? That defines the business strategy.

Plan

Despite what a lot of people think, the cloud is not solely about the technology. Of course, technology is an important part of the CAF and the forthcoming architecture, but cloud adoption is even more so about aligning business processes, people, and technology. In adopting the cloud, we will likely move workloads such as applications to a cloud platform. Ask these questions in drafting the plan:

What do we use?

Why do we use it?

Who uses it?

When do we use it?

The answers will help in defining the strategy to migrate workloads and applications to the designated cloud platform. One essential question is: does it bring added value to move a workload to the cloud? Followed by the question: how will it bring that value? This is where the following five R's are important:

This is lift and shift. Workloads such as applications are not modified but migrated as they are to the cloud platform.

This is lift and shift too, but this time some modifications are done. For instance, a business chooses to keep the application as it is, but some parts are shifted to managed services by the cloud provider. Think of having the databases managed through a managed service such as Relational Database Service by AWS.

By refactoring an application, the application is modified. Services are replaced by cloud-native services, for instance, using container technology or serverless functions. This often means a redesign of the application, for instance, from a monolith architecture to micro-services.

An outcome of the strategy or planning phase might be that an application is obsolete and can be retired.

Retain: There might also be workloads and applications that cannot be migrated to the cloud for various reasons. The application must be close to the data source or the machine that it operates, which is typically the case in operational technology Think of manufacturing or healthcare. There might be restrictions to using public clouds because of legal compliance, or an application is critical to the business but simply too old to move to the cloud. These might all be reasons to retain an application, meaning that they are not touched at all.

Prepare

The next step is to prepare the cloud platform that will host the workloads and applications. Typically, this starts with setting up the landing zone in the designated cloud. The landing zone is the foundation. If we are building a house, we need to know what the house looks like before we can lay out the foundation. It is the same for the cloud. We have to know what sort of workloads we will be migrating to the cloud to define and design the landing zone. During the course of this book, we will discuss the landing zone extensively.

Adopt

This is the phase where the workloads are migrated to the cloud according to the plan and the migration strategy that we have defined. We can either lift and shift workloads as-is or transform the workloads and adopt cloud-native services.

Govern

We need an organization that is able to manage the cloud and the workloads in the cloud. These are necessarily the same thing. In the govern phase, organizations might want to form a Cloud Centre of Excellence with a specific platform team, which manages the cloud, and application teams that manage the specific applications in the cloud.

Operate

This is the phase where organizations will monitor the workloads and make sure that these are performing in the most optimal way, following the best practices of the cloud provider and fulfilling the business requirements.

Most CAFs have added two more pillars to these six: security and sustainability. These might be debatable since both security and sustainability should be intrinsic and taken into account for every workload that is migrated to a cloud platform. In other words, security and sustainability are part of all six stages in the CAF. Yet, both AWS and Azure have security as separate pillars in the CAF, as we will learn in the next section.

Deep dive in the CAF of Azure and AWS

First, we take a look at the CAF of We will recognize the generic pillars of the CAF, but AWS calls these the foundational capabilities:

The business perspective helps to set the strategy for digital transformation. The AWS CAF takes the need for digital transformation as the starting point. In other words: it is not the question of whether a business must digitize but how. The business perspective helps define how cloud investments can accelerate this transformation.

The people perspective is mainly about transforming the culture of a business. Digital businesses need people with a growth mindset and people who are willing to learn continuously and change accordingly. One remarkable aspect of the people perspective is cloud People need to understand the cloud, in this case, AWS. It might require a workforce transformation.

The governance perspective is all about project and program management, guiding organizations in their journey to AWS, and making optimal use of AWS services. This includes risk management and cloud financial management or FinOps.

This is, obviously, about the cloud platform itself and how we build it in AWS. There is one golden rule that applies here: AWS is responsible for the cloud, the customer of what is in the cloud. AWS provides its customers with a toolkit to build a virtual private cloud on their platform. It is up to the customer to use these tools and build a scalable, resilient environment to host

applications and data. The CAF will help with best practices for platform, data, and application architecture, including Continuous Integration and Continuous Delivery through (automated) pipelines that integrate with AWS.

Security: As said in the previous section, AWS, and Azure have separate pillars for implementing and managing security in the cloud. It includes Identity and Access Management threat detection, protection of infrastructure, data, and applications, and the management of the security postures in the cloud.

Operations: From the business, requirements will be set concerning performance and reliability. This must be monitored and managed. Typically, IT operators manage environments using IT service management frameworks such as ITIL, including incident, change, configuration, and problem management. Observability is key, next to fast detection and response. The AWS CAF specifically mentions AIOps, predictive management through artificial intelligence

These capabilities are required to go through cloud transformation value chains. The value chains lead to the following business outcomes:

Reduction of business risks

Improved Environmental, Social, and Governance values

Growth of revenue

Increasing operational efficiency

To reach goals in business outcomes, businesses must go through a transformation. AWS specified four transformation domains:

Technology

Process

Organization

Product

All these domains will continuously change and transform. But by using cloud technology, these transformations can become more agile: adaptable and scalable. If we put this all together, we get the CAF of AWS, as shown in the following Figure

Figure 1.1: High-level representation of AWS Cloud Adoption Framework

A whitepaper about AWS CAF can be found at:

As we will see in Azure as well, the CAF is not a one-time exercise but more of a lifecycle. That makes sense if we realize that the business, and the cloud itself, constantly changes with updates and new features. AWS presents this as the cycle from envisioning to aligning, launching, and scaling. The business envisions how the cloud can help in achieving business goals, aligns this with the foundation capabilities, launches the new services and products as Minimal Viable Products or a pilot, and lastly, expands it to production. From there, the cycle starts over again.

Microsoft Azure presents the CAF as a cloud adoption lifecycle, too, starting with the definition of a strategy. The strategy is all about defining the desired business outcomes and the accurate justification to start the cloud journey. The Azure CAF is represented in the following Figure

Figure 1.2: High-level representation of Azure Cloud Adoption Framework

To get started with the Azure CAF, Microsoft recommends working from scenarios. These scenarios have been chosen from various business standpoints. Perhaps one remarkable scenario is the hybrid and multi-cloud scenario. It is remarkable since this scenario focuses on businesses that will

have more than one cloud and even cloud combined with on-premises environments. Using the CAF, businesses can establish unified and centralized operations across these different clouds and their on-premises data center. The CCoE is an important element in this scenario, combining knowledge of various cloud solutions and integrating these into one unified set of processes and best practices for architecture.

One other special scenario is desktop virtualization, allowing customers to migrate workplaces to Azure Virtual Desktop Using the CAF guidelines, businesses can implement AVD instances in Azure and integrate this with Windows and Office365, the latter being a SaaS proposition.

These scenarios all follow the same approach that is set out in the CAF: strategy, plan, migrate, manage (operate), and govern. The business will formulate the ambition and the goals that are worked out in a plan. Next, the workloads – for instance, the virtual desktops – are migrated. An organization centralized in the governing CCoE will manage the workloads compliant with the business requirements.

The Azure CAF pays extra attention to so-called antipatterns. There is a list of antipatterns to be found on but there are two in particular that we like to mention here:

IT as cloud This is the antipattern where the business treats its IT organization as the cloud provider. It is not the cloud provider; they are using the technologies in the cloud. Keep in mind that the cloud provider is responsible for the cloud, the customer of what is in the cloud. For example, the failure of a region in Azure or any other cloud is not the responsibility of the IT organization. Monitoring and managing the resiliency of specific workloads, where these failover to another region, is the responsibility of IT. That,

however, starts with business requirements and the forthcoming architecture to design the resiliency of that workload.

Inaccurate out-of-the-box security Again, cloud providers offer a massive number of tools that will help organizations to secure workloads in the cloud. Public clouds are likely the best-secured platforms in the world, but that doesn't mean that workloads are secured by default. That depends on how the customer applies security guardrails, guidelines, and usage of tools to protect applications and data in the cloud. The assumption that the cloud provider automatically takes care of that is wrong.

The appropriate use of the CAF will help avoid these pitfalls and antipatterns. In the next section, we will study the CAFs of Google Cloud and Alibaba Cloud, which are a bit different from AWS and Azure.

Frameworks by GCP and Alibaba cloud

Google Cloud Platform and Alibaba Cloud also have versions of a CAF. We will study these in this section. GCP defines its CAF in four themes and, with that, takes a completely different approach to cloud adoption.

Lead

This is about leadership from sponsors in the business, which supports the migration to the cloud. It also includes the teams themselves and how they collaborate and motivate one another in a successful transition and transformation to a cloud platform.

Learn

Cloud adoption is so much about technology but more about adopting a new way of working. Companies will have to learn how public clouds work. In other CAFs, this is typically gathered under people or as part of the operating model, including a center of excellence. Moreover, the staff needs to be trained and upskilled. This goes beyond technical skills.

A company and its employees also must learn to understand how, for instance, financing works in the cloud. What financial models are applicable in the cloud? Typically, organizations start with pay-as-you-go in the cloud, but there might be situations where reserved capacity might be a much better choice. Reserved capacity often means that a company still needs to invest or at least confirm and commit that it will use resources in the cloud for a longer period.

Migrating to the cloud is a learning process in many aspects. Not only is the technology different from traditional IT, but applications and data are managed differently in the cloud. Migrating to the cloud is a huge change and requires transformation and change management. Governance, security, development, operations, and financial management: these are all part of the transformation. In this book, we will mainly focus on the technical management of cloud environments, but it is good to keep in mind that cloud adoption involves more than just technology.

Scale

One of the most important and obvious reasons for companies is that the cloud offers scalability. GCP focuses on limiting manual processes as much as possible. Hence, automation is a major topic in the adoption framework. Workloads and services in the cloud must scale automatically but are always triggered by business processes. This is referred to as event-driven. For example, an event can be a customer that places an order on a website. That will trigger the process of payment and delivery process of the product. When a company launches a new product, this might lead to a peak in orders. Using automation, the cloud services will automatically scale to facilitate the peak and make sure that the websites and associated applications keep performant. As soon as traffic decreases again, automation will also take care of scaling down, avoiding unnecessary costs.

Secure

Performance and cost control are important, but there is one more item that is at least as equally important or perhaps even more important. The fourth pillar in the CAF of GCP is, therefore, security. Security starts with identity and access management but also includes several tactics and techniques to protect workloads and services in the cloud.

Next, the framework addresses three levels of adoption: tactical, strategic, and transformational. Simply put, tactical concerns the individual workloads in the cloud, but there is no plan to leverage cloud-native services, enhancing automation and scalability. It is a simple lift and shift of workloads to the cloud, causing no disruption to the company. Basically, the cloud is used as a traditional data center.

On a strategic level, there is a plan to automate individual workloads and start decreasing the manual efforts to manage these workloads. On the transformational level, organizations use the cloud to innovate, using automated development and deployment pipelines to enable regular releases of new features to products or new products as a whole. The cloud now has become essential in shortening time to market, decreasing the cost of sales, and, with that, increasing revenue. The cloud is adding value to the business and, with that, has become part of the digital transformation of the business. We will talk about this in the final section of this chapter.

Putting the four pillars and the three stages together results in the cloud maturity scale that GCP uses. It can be seen in the following Figure

Figure 1.3: High-level representation of Google Cloud Adoption Framework

We can use this cloud maturity scale to define where our organization stands and what the ambition should be. To make this a bit clearer, we will use an example:

Scale on a tactical level means that environments are hardly scalable. There is a lot of manual work involved in managing the workload.

On a strategic level, an organization might already use automated templates to deploy workloads.

On the transformational level, all workloads scale automatically, using blueprints from CI/CD pipelines, including automated deployment and scaling scripts. Manual work is very limited.

We can do the same for the three other pillars. This will help architects define what is needed to get the most out of the cloud and help businesses by adding value. A whitepaper with Google's approach to the CAF can be found at

Like Azure, Alibaba Cloud presents the CAF as a journey and cloud lifecycle. The first step is setting the strategy. Essential in setting the strategy is answering the question of why the organization should move to the cloud; Alibaba calls this the Cloud Adoption Motivation. Quite obviously, this starts with business requirements. Next, Alibaba Cloud provides examples of motivations such as:

Speeding up global delivery of applications.

Reduction of costs: Remarkably, Alibaba Cloud argues that most costs in the cloud are Operating Expenditure where upfront investments are not needed in contrast to Capital Expenditure A characteristic of CAPEX is the need for upfront investments. This might be true for a lot of cloud services, but there are situations where upfront investments will be required, for instance, when reserving resources for a longer period in the cloud. The shift from CAPEX to OPEX in a cloud computing context has significant business impacts:

Financial OPEX models allow organizations to pay for services as they use them, providing greater financial flexibility and reducing upfront investments typically associated with CAPEX.

OPEX models enable businesses to scale resources up or down based on demand, improving cost efficiency and reducing the risk of overprovisioning or underutilization.

Faster Lower upfront investments and the ability to quickly deploy resources reduce the time-to-market for new products or services, offering a competitive advantage.

Focus on core By moving to an OPEX model, organizations can allocate resources towards their core business functions, while cloud providers handle infrastructure management and maintenance.

Improved In this case also, there is a trade-off. Public clouds are likely the best-secured platforms in the world since they serve millions of customers. Cloud providers offer extensive toolsets to protect workloads and data in their clouds. However, it is still the responsibility of the customer to use these tools.

Evaluating all these aspects is part of setting the strategy. The following step in Cloud Adoption Motivation is setting up the organization. Alibaba Cloud recommends having a Cloud Centre of Excellence with cloud technologists, application owners, and security specialists. The application team works closely together with the business, responding to the business requirements. The cloud team is mainly responsible for the continuity of services. Lastly, the security team takes care of defining and controlling the security guardrails that must be followed in the cloud.

The next phase is cloud adoption preparation and management framework building. The first and main task in this phase is setting up the landing zone in the cloud. In the CAF, the landing zone refers to the foundation of cloud management, including:

Financial management

Security management

Compliance and auditing

Automation

Network planning

Resource planning

Once the landing zone has been defined, our cloud is ready to onboard applications, which is the following phase. The phase includes the migration of applications and data and development innovations to enhance applications and usage of data. In the case of the latter, we can think of adding Artificial Intelligence to analyze data.

Now, we have a CCoE helping the business to start the digital transformation and setting up the landing zone to start adopting applications and data. This new cloud environment must be operated and managed from a cost and security perspective, identifying and quantifying risks that may impact the business in a timely manner and ensuring business continuity. This is done in the final phase of organization and governance. Reacting to events will lead to new insights and new motivations, taking us back to the first phase in the CAF, adjusting the cloud adoption strategy.

The complete CAF of Alibaba Cloud can be downloaded from:

Similarities and differences

If we compare the different CAFs with the generic principles, we will notice similarities. All CAFs start with defining the business strategy, which is absolutely a crucial step. There is no point in moving to the cloud just for the sake of being in the cloud: there has to be a business justification. A lift and shift of environments – applications – to a cloud platform will barely bring any benefit. The rationales for businesses to migrate and embrace cloud technology include business agility and scaling.

Business The ability of businesses to respond and adapt to changing customer demand or market circumstances swiftly. Since in the cloud, virtually everything is coded using Infrastructure as Code and Configuration as Code, it allows for applying fast changes and rapid development. One of the key elements in achieving business agility is scaling.

Remember that scaling can be both up and down, preferably automated. This way, a business will have the ability to use exactly what is required and also pays for the resources that are used. This is addressed by the CAFs from various angles: automation, financial management, and resource management.

Typically, we see a couple of common use cases for businesses to migrate to the cloud. Outsourcing is the number one reason why the existing data

center is decommissioned. This is often financially driven, where data centers require a lot of upfront investments that must be depreciated over the years. These are CAPEX costs. By migrating to the cloud, companies can shift to OPEX.

Another reason is the lack of efficiency in the existing set-up of IT. There might be technical debt slowing down development and increasing operational efforts. A lack of skilled resources to manage the legacy IT or skilled resources that cannot be utilized for innovations because of too many operational tasks will also be arguments to migrate to the cloud, including the adoption of automated, cloud-native services.

All CAFs emphasize the need for training and upskilling people. This includes technical staff but also other personnel, including financial specialists and senior management. Companies shifting to the cloud as part of digital transformation programs will encounter cultural changes. That requires sponsorship and teamwork. Management must endorse the adoption of the cloud; teams will need to learn to work with the cloud.

One other major similarity is the drive to set up specific governance. IT and business must be aligned, but in the cloud, this becomes even more important. The reason for this is that with the cloud, IT shifts closer to the business, meaning that the business gets more control over the development and deployment of products, whereas, in more traditional IT, the business is completely dependent on how fast IT can implement technology. In the cloud, this has become a much easier process. Every asset is turned into code, including the infrastructure. There is no need to define a long list of requirements for equipment; using the cloud, the infrastructure is a string of code that can be easily adapted to the exact needs of the application.

The cloud enables business agility and makes businesses really scalable, but the cloud must be managed. That is where the CAFs focus on governance and management, stressing the importance of:

Cost management

Security management

Development and deployment of resources (starting with MVPs)

The outcome of the CAF should be business value. That is the central theme in all CAFs. To achieve this, organizations must agree on the strategy. All stakeholders must be involved and convinced that cloud migrations will add business value. The following step is to train the appropriate capabilities in the organization. People, processes, and technology must be aligned in a new way of working. Cloud adoption will bring significant changes to an organization, and this needs careful planning in a lot of domains. The CAFs will show you where the organization stands and where and how it can grow those capabilities. This is where all discussed CAFs have the same focus.

Still, there are differences too. These differences can often be explained by the origin of a specific cloud provider. AWS is completely cloud-born and focuses primarily on cloud-native services. With that, AWS is set up in a more modular way, offering a lot of different, customizable building blocks, whereas Azure offers more out-of-box, complete services. Next, Azure also focuses more on the hybrid cloud. Microsoft was already an established name in computing, with an operation system that could run

on a variety of machines, from personal computers to servers. Microsoft is used to working in hybrid environments.

The Azure portfolio contains several services that enable the hybrid strategy. Think of Azure Stack, which is basically an Azure extension on-premises, but also Azure Arc, which allows to bring non-Azure workloads under the control of the Azure console.

Note that customers could have AWS on-premises, too, via VMWare on AWS, allowing on-premises VMWare stacks to be stretched into the AWS cloud. This was later followed by native AWS on-premises, with AWS Outposts.

The bottom line is that a company that starts its journey in digital transformation and adopting cloud services must first set out its business strategy. From the business strategy, a solution and the right fit of cloud technology will follow. The CAFs are a great aid in defining the business strategy and how cloud technology can help in achieving the goals and ambition of that strategy.

Monitoring multi-cloud and keeping track of value propositions

So far, we discussed the various CAFs that will guide us in setting up environments in the cloud. If we have workloads running in Azure and AWS, just as an example, then we have to manage a multi-cloud environment. However, the same is already true when a company uses SaaS, such as Office365 and Salesforce, next to having backend IT hosted in a hyper-scale cloud (major cloud providers with a global presence, such as the providers that we discuss in this book: AWS, Azure, GCP and Alibaba Cloud).

Alternatively, an organization might have workloads in public clouds and on-premises in privately owned data centers – this is typically referred to as a hybrid. To cut a long story short, the introduction of cloud technology has not made things easier or less complex. That is the reason why companies need to think about their cloud strategy before they start migrating workloads to any cloud. The CAFs are good guidance in defining that strategy, as we have learned in the previous section.

But that is not all. Through the course of this book, we will learn that managing multi-cloud is not easy. There is a good reason why all CAFs mention the human factor in setting up and managing cloud environments. We need professionals: architects, engineers, and developers that have a thorough knowledge of the cloud.

Those are the hard skills, but soft skills are just as important. Cloud architects must be able to understand the business and translate business challenges into cloud solutions. We cannot stress it enough: going to the cloud, just because you can will not bring any value to the business. At the end of the day, it is about value propositions. How can cloud technology help businesses in digital transformation and move the business forward? Cloud architects need to understand business metrics.

Measuring business metrics

Measuring the performance of the business is done through business metrics that quantify that performance. Quantifying parameters can be defined in finance, marketing, operations, production, human resources, and IT. We can think of the following:

Sales revenue

Cost of sales

Conversion rate

Web traffic

Goods sold

Time to delivery

Payment cycle, including days payables outstanding

Cloud technology can help in increasing revenue, lowering overall operating costs, reaching more customers, increasing the speed of innovation, releasing new products, and decreasing time to market. Is this part of the CAFs? The answer is yes. The CAFs are not as much about the

technology itself. That is where concepts such as well-architected frameworks and reference architectures play a much bigger role. The CAF is really about business adoption and implementing value propositions. So, how can a business increase sales using the cloud? For instance, by providing webshops around the globe that automatically processes orders, thus decreasing the cost of sales.

That is the key to digital transformation. Companies adopting digital technologies to change the existing business or, even better, create new business. The cloud can help in this, but it likely requires a new operating model wherein digital expertise plays a much bigger role. First of all, the transformation must be data-driven. Only through accurate, timely, and continuous analysis of data, a business will be able to make the right decisions. What does the customer want? What is my competition doing? What is the best time to launch a new product? This is where the cloud brings real added value. In the cloud, we can bring data together, analyze it and use it to refine the strategy.

It has a major impact on a lot of aspects of a company. Does the company have the right skills? Has it chosen the right partners? Is the leadership supporting the changes? Does the company have a budget? Does it have access to the right data sources? Who should have access to data and resources? Before we get into the application, data, and technical architectures, these questions must be answered. It defines the level of adoption of the change that comes with digital transformation and the use of cloud technology.

Introducing best practices in monitoring

Once we have solutions deployed in the cloud, we must start managing them. That is the core topic of this book. Managing starts with seeing what is going on; it starts with monitoring. Now, admins are likely not very fond of having to use multiple tools and consoles to monitor the various environments. Monitoring multi-cloud is best served with a single pane of glass view. Best practices in monitoring are as follows:

Use monitoring tools that are capable of monitoring heterogeneous multi-cloud environments.

Have a Configuration Management Database that is truly responsive to the cloud. If an engineer spins up a VM in the cloud, it must be reflected as a new asset in the CMDB.

Following the CMDB and the responsiveness: monitoring must support automation.

Preferably, monitoring must be intelligent, supporting anomaly detection, self-healing, and automated remediation.

The next level is to have monitoring that can determine the impact of events on the business metrics. We need full-stack visualization for this: linking business processes to technical artifacts. This is the domain of AIOps, tooling that can determine the impact of events on business

processes across multi-cloud. This is part of the operations phase in the CAFs. Operations typically contain:

Operations fulfilling business commitments.

Security ensuring business resilience.

Governance limiting business risks.

Monitoring with full-stack visibility is a must.

This concludes this first introductory chapter. In the next chapters, we will learn how to manage environments in multi-cloud.

Conclusion

Before we learn how to manage environments in the cloud, we must first get to the cloud. This starts with understanding why a company migrates its IT landscape to the cloud. There has to be a business justification to start a cloud migration. Moving to the cloud will have a major impact on the company: on the business, the operations, and the people. In this book, we will mainly focus on the major hyper scale's AWS, Azure, GCP, and Alibaba Cloud. In this chapter, we studied the Cloud Adoption Frameworks of these providers. All CAFs start with the business justification and next provide best practices on how to transform a business by migrating to their respective cloud platforms. The CAFs focus on setting up a cloud management organization, obtaining cost control, and ensuring that environments in the cloud are protected and secured.

Migrating to the cloud, therefore, starts with studying the CAF and applying the best practices. In the following chapters, we will learn how to manage the environments once they have been migrated to the cloud, starting with understanding the principles of virtualized compute resources.

Points to remember

Cloud Adoption Frameworks are a good guide in setting a strategy to migrate workloads to the cloud.

Every cloud provider – Azure, AWS, Google Cloud, and Alibaba Cloud – offers a CAF.

It is very important to remember that every move to the cloud must have a business justification. Why is a business moving to the cloud, and what are the benefits to the business?

CAFs cover important topics such as setting up governance, controlling costs, and operating security according to the best practices of the respective provider.

Questions

In what migration strategy do we simply lift and shift a workload to the cloud without making changes to it?

What are the four stages in Google's CAF?

An important aspect of monitoring is having a CMDB. What is a CMDB?

Answers

Rehost.

Learn, lead, scale, and secure.

Configuration Management Database – a database holding all assets that IT has to manage.

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

Virtualizing and Managing Connectivity

Introduction

To understand connectivity in the cloud, we must understand the virtualization of networks since networking in the cloud is also software based. In this chapter, we will learn what cloud networking is, with guidance on software building blocks, deployment models, and operating networks in the cloud. Concepts such as SD-WAN, SD-LAN, and edge will be discussed. We will also introduce micro-services and how network virtualization can help with this.

Structure

In this chapter, we will discuss the following topics:

Building blocks in networks

Cloud network operations

Deployment concepts SD-WAN, Edge, and SASE

Managing networks in multi-cloud

Building blocks in networks

In the cloud, everything is coded, and with that, almost everything is virtualized. Public clouds use the concept of shared resources on which capacity can be reserved for one tenant. This includes computing, storage, and networks. In the following chapters, we will study computing and storage, but before we can start building servers and developing applications on top of these servers, we must create connectivity. In this section, we will learn what the building blocks are for networks and how these are implemented in the cloud, discussing the various options in the major public clouds.

The three main building blocks for networks and connectivity are switches, routers, and firewalls. That applies to the cloud too. All of them take care of the following:

Switch

Router

Firewall

Load balancing

Virtual Network Function (VNF)

Network Function Virtualization (NFV)

In the cloud, we will have virtualized, software-defined versions of these building blocks. Let us start with the virtual switch or vSwitch. In all its essence, this is software that allows a virtual machine to communicate with another machine to set up connections between machines. Virtual switches do the same thing as physical switches, but in the cloud, everything is transferred into code. So, a virtual switch is just software. That makes it easier to manage, as administrators will only have to look at some code and do not have to worry about hardware and network cabling. The software virtualizes the hardware and the cabling, so to speak.

Virtual switching

We use virtual switches in the cloud to connect to networks. Like physical switches, they use network interface cards with virtual ports, which allow for traffic to the switch and to the network that it is attached to. The switch will detect data packets and allow it to travel to the packet's destination, as shown in Figure

Figure 2.1: Working of a virtual switch

Cloud providers offer various solutions for virtual switching. In Azure, we will use vNets, virtual networks. vNets allow for traffic between virtual machines and other resources in Azure, and they look very much like the virtual local area networks that we had in the traditional data center, allowing for segmentation in the data center network. Typically, every resource in a vNet will be able to communicate with each other. If we want to enable

communication between resources in different vNets, then we need to set up vNet peering in Azure. The basic setup of a vNet is shown in Figure

Figure 2.2: High-level architecture for vNet in Azure

Network security groups in Azure are used as Access Control Lists to determine what traffic is allowed or denied to resources in the network subnets. The principle of vNet peering is shown in Figure

Figure 2.3: High-level architecture for vNet peering in Azure

The principle of virtual networks and switching works the same in other clouds. In AWS, we use Virtual Private Networks Just like in Azure, we can also define subnets in a VPC, allowing for more segmentation of our networks. If we want to allow resources in one VPC to communicate with resources in another VPC, we use VPC peering connections. But what if we have multiple VPCs that we want to connect to each other and route traffic? AWS offers Transit Gateway as a solution, acting as a central hub, also to the outside world over the internet. The basic principle of the Transit Gateway is shown in Figure

Figure 2.4: High-level architecture for Transit Gateway in AWS

In Google Cloud, we will also find VPCs. As you may have concluded already, vNets and VPCs are not exactly the same as switches. They act on layer 2 of the Open Systems Interconnection model: the data link layer, which is the layer that only takes care of the transport of data.

Routing

Making sure that data reaches its destination requires more than just switching or, in the cloud, setting up vNets or VPCs. We must direct the data in the right direction. That is what we use routing for. Routing in the cloud is not quite like the traditional router that we had in the traditional data center. Switching can be a task of a router, which supports the following functions: connecting different networks, receiving, and sending data packets, and segmentizing network zones. Routers typically act on layer 3 of the OSI model: the network layer, connecting networks physically to each other.

Routing in cloud environments works differently. In that case, routing still takes care of connections between networks, either in the cloud or with on-premises networks, but it adapts dynamically to changing network conditions using the Border Gateway Protocol Routing selects the optimal path for traffic in a network or between networks, and hence it is dynamic. All clouds use BGP as a routing protocol, allowing routers to quickly adapt to send packets over a different connection if an internet route goes down, taking dynamic decisions based on paths, rules, or network policies.

Each BGP router maintains a standard routing table that is used to route packets to their destination. This table is used in conjunction with a separate routing table, the Routing Information Base which is a data table stored on a server on the BGP router. With that, we are at the level of routing tables in the cloud. In all clouds, we must define these routing

tables or user-defined routing as they are referred to in Azure. The route table holds a set of rules that determines where network traffic will be directed, with preferred and secondary routes.

In addition to BGP, other routing protocols that could be used in cloud environments include Open Shortest Path First Enhanced Interior Gateway Routing Protocol Routing Information Protocol Intermediate System to Intermediate System and static routing. While BGP is the most common choice for public cloud providers due to its scalability and flexibility, private and hybrid cloud environments may use different protocols depending on network requirements and infrastructure.

Virtual firewalls

Routing is essential for the next artifact in networking: firewalls. Firewalls are essential in allowing or rejecting network traffic coming in or going out of cloud environments. Since it is very hard to install physical firewalls (not impossible, though), the firewalls are virtualized, meaning that we have software fulfilling the function of a firewall. That function will be the same as in a physical data center. Firewalls help with intrusion prevention, filtering of internet addresses such as uniform resource locator IP, and blocking specific files or file types.

The big advantage of virtual firewalls is that they are scalable: we can apply them on various locations in our network and have them protect specific resources, allowing for fine-grained protection of resources. For instance, we can have web application firewalls that protect a specific application or set of applications, filtering and monitoring HTTP traffic between a web application and the internet. Figure 2.5 shows a simple setup of a WAF:

Figure 2.5: Example of architecture for Web Application Firewall

Notice that the diagram also shows a load balancer. We will discuss load balancing in a moment.

Cloud providers offer native firewalls, such as Azure Firewall, AWS Network Firewall, and Control Traffic, in Google Cloud and Alibaba Cloud Firewall. Yet, companies might want to use firewalls of companies that are specialized in designing these devices and that they use in their datacenters. Cloud providers also have marketplaces where customers can find products and services of third-party companies, enabled for use in the cloud. Figure 2.6 shows the screen in AWS Marketplace, offering various types of firewalls that are supported by AWS. The following screenshot shows Palo Alto and Fortinet, but there are many other options:

Figure 2.6: AWS Marketplace showing firewall propositions

Note the left-hand side of the screen, where the various deployment methods are shown. We can deploy a firewall as an Amazon Machine Image a CloudFormation template, or a container image, amongst others. The AMI will deploy the firewall software as an image to a virtual machine. CloudFormation is the Infrastructure as Code in AWS. One other possibility is to get the firewall as Software as a Service completely managed by the vendor.

Native cloud provider firewalls and third-party firewalls or Software as a Service have some key differences that may affect which one is the best option for your organization:

Native cloud provider firewalls are tightly integrated with the cloud platform they are provided by, such as AWS, Azure, or GCP. This can make them easier to set up and manage, as they are designed to work seamlessly with the cloud infrastructure. Third-party firewalls, on the other hand, may require more configuration to work with cloud platforms, although many of them now offer integrations with major cloud providers.

Third-party firewalls and SaaS offerings may provide a wider range of features and greater customization options than native cloud provider firewalls. This can be particularly important for organizations with complex security needs, as third-party firewalls may offer more advanced threat detection and mitigation capabilities.

Native cloud provider firewalls are often included in the pricing of the cloud platform, which can make them a more cost-effective option for organizations with simpler security requirements. Third-party firewalls and SaaS offerings may require additional licensing fees or subscription costs.

Native cloud provider firewalls are typically supported by the cloud platform provider, which can provide a more seamless support experience. Third-party firewalls and SaaS offerings may have their own support teams, which can provide more specialized support but may require additional coordination with the cloud platform provider.

The decision between using native cloud provider firewalls and third-party firewalls or SaaS offerings will depend on the specific needs and requirements that we have defined.

Load balancing

Load balancers do exactly what the name suggests: they balance the load across cloud resources, distributing the network traffic across resources so that resources do not get overloaded. Incoming traffic is routed across multiple resources, such as network segments and virtual machines in those segments. It is even possible to distribute network traffic load across different cloud regions.

There are different types of load balancing available, each with its own strengths and weaknesses:

Network-level load balancing operates at the transport layer (Layer 4) of the OSI model and distributes incoming traffic based on factors such as IP address, port number, and protocol. Network-level load balancing is ideal for environments with high traffic volumes and low processing requirements, such as simple web applications.

Application-level load balancing operates at the application layer (Layer 7) of the OSI model and distributes traffic based on more granular factors such as HTTP headers, cookies, and URL paths. Application-level load balancing is ideal for complex web applications with multiple components, such as microservices or API gateways.

DNS load balancing uses DNS servers to distribute incoming traffic across multiple servers or instances based on geographic location or availability.

DNS load balancing is ideal for global applications with distributed user bases and can help to improve performance and reduce latency.

Each type of load balancing has its own advantages and disadvantages, and the choice of load balancing strategy will depend on the specific requirements and characteristics of the workload. In general, application-level load balancing is best suited for complex web applications with multiple components, while network-level load balancing is more suitable for simpler, low-resource applications. DNS load balancing, meanwhile, is ideal for global applications with distributed user bases.

Cloud providers offer a variety of load-balancing options that are designed to meet different needs and workloads. It's important for organizations to carefully evaluate these options and select the right load-balancing strategy for their specific use case to ensure optimal performance, availability, and scalability of their cloud-based applications.

In Azure, we will use Azure Load Balancer, which operates on OSI layer 4, routing traffic between VMs. Azure also provides layer seven load balancers, the layer for applications: both Azure Traffic Manager and Azure Application Gateway operate on that layer. Lastly, we have to mention Azure Front Door, used for microservices. Azure Front Door acts as a global load balancer and Application Delivery Network that can route traffic to the closest available backend, providing low-latency, high-performance access to web applications. We will talk about microservices in the final part of this chapter.

In AWS, we use Elastic Load Balancing to distribute and route traffic to resources such as EC2 instances. Elastic Load Balancing can operate on various layers, both on the network and on the application layer. Google

Cloud offers Cloud Load Balancing for internal and external load balancing. This is the same service that Google itself uses for its own services.

The most common load-balancing service in Alibaba Cloud is Server Load Balancing which is fully managed by Alibaba Cloud. We started this section by stating that public clouds use the concept of shared resources by reserving capacity for a specific customer, typically referred to as a tenant. Hence, we need means of separating networks in the cloud so that one tenant cannot reach the other or even resources are separated from each other in different network segments. That is an important task of network operations: a task that is heavily changed due to the concept of Software-Defined Networks In the next section, we will explore the concept of SDN in more detail and discover how we can set up network operations in the cloud.

Cloud network operations

As we have learned in the first section of this chapter, in the cloud, we have virtualized networks, too, by translating network functionality into code. The concept is called software defined. But what exactly does that mean, and, more important, what does it mean to operate networks?

Software-defined is all about the abstraction of layers. The term was launched by VMWare, which introduced the concept of software-defined data center at the beginning of the millennium. With that, they shared a vision of how to build data centers with software and thus limit the use of hardware by virtualizing it as much as possible. In the next couple of chapters, we will discuss the virtualization of storage and computing in more detail, but in essence, VMware introduced technology that could tell a server or any other device that it was more than just one server. That concept was later also applied to network appliances: creating multiple network segments on top of one stack of network appliances in the data center, instead of having to use dedicated network equipment for every network segment.

In fact, it is the technological cornerstone of any cloud. With virtualization and software defined concepts, we can create multi-tenancy in the cloud. Look at the cloud this way: it is one big building where the owner of the building makes sure that everyone gets electricity, water, and heating. But every tenant still has their own apartment. The tenant is free to do with the apartment as it pleases if the shared services of the building are not compromised. The electricity, water, and heating are the software that serves multiple tenants – instead of each tenant having its own water supply and heating system.

This is what software-defined does. Now the focus is on software-providing solutions instead of constantly having to bring in new hardware. Hence, we are abstracting the functionality of networking from the actual devices, as shown in Figure

Figure 2.7: Abstracting layers in software-defined networking

In a software-defined architecture, we decouple the network controls from the underlying infrastructure: we are abstracting physical resources, making our networks programmable and very scalable. Applications can now treat the network as a logical or virtual unit. To abstract the control layer from the infrastructure, we use OpenFlow, a communications protocol that enables a server to communicate with a network device, instructing where traffic should go. The applications communicate with network services using Application Programming Interface

Obviously, this does have an impact on how we operate networks. Although this book is completely about the cloud, we must be conscious of the fact that

a lot of enterprises and organizations will have legacy networks that they have to maintain as well, simply because only very few organizations will completely replace the existing infrastructure. Therefore, network operators will have to manage both the legacy infrastructure and the software-defined networks. It is good advice and best practice to explore the existing infrastructure – routers, switches, firewalls, proxies – to determine if these support SDN as well. For a start, it is strongly recommended to have tools that can monitor both stacks and preferably be used to manage the existing and the software-defined networks. Operators can use the existing tools and view the status of network components through dashboards that they are already familiar with.

Still, to understand networking operations in software-defined networks, the administrator will have to learn different skills. Be aware: there will still be hardware involved, but this hardware is now managed by the cloud provider. The network administrator – or cloud engineer – does not have to worry about the devices themselves anymore. The concept of SDN and networks in cloud computing does make life easier: there is no need to understand the operation details of hundreds of various network protocols. Still, understanding the concept of software-defined networking is essential. The concept contains three layers:

Applications that handle resource requests and perform actions in the network.

Controllers that route requests and actions through the network.

Networking devices that receive the requests and process the actions.

Figure 2.8 shows how a software-defined network is configured:

Figure 2.8: Layers in Software-Defined Networking

So, we need to define and implement policies to instruct the network controllers. The controller will eventually program the underlying infrastructure. For controllers, we can use NOX, Floodlight, or Meastro. NOX is the original OpenFlow controller. Floodlight and Maestro are open-source JAVA OpenFlow controllers and good, widely used alternatives.

This concept is applied in every cloud. In every cloud, we will first have to define our tenant by creating a virtual private cloud or virtual network. Next, we must define who may access this tenant: the policies. Typically, this is done by ACL. In Azure, NSG functions as ACL. An ACL simply tells what traffic and from what sender the traffic is allowed. If we want to control outgoing traffic and connections outside the tenant, then the use of a virtual gateway is recommended, such as, for instance, if we want our tenant in the cloud to be connected to on-premises environments.

The Cloud Adoption Frameworks and Well-architected Frameworks will help us with best practices in setting up our networks in the cloud. The benefits will be significant:

Enhanced network Preferably through a single pane of glass view. SDN allows for centralized management and monitoring.

Since it is all software, networking through SDN is extremely scalable. No need to install new hardware if we need to extend the network.

Enhanced Since SDN is centrally managed, security policies can be propagated easily throughout the entire network, including all components and all devices.

Open In theory, SDN uses open-source protocols and, with that, is agnostic to hardware. SDN abstracts the network functionality from the physical devices so that SDN is vendor neutral as long as the devices support the appliance of SDN.

SDN is less expensive to operate than physical networks, where we also need engineers that are trained in specific equipment. Keep in mind that occasionally we will still need these engineers since, somewhere, SDN sits on physical machines.

Now that we understand the concept of software defined networking, we can take the next step and start deploying and managing software defined wide area networks local area networks and stretching networks to the Edge with edge computing. In the next section, we will also discuss Secure Access Service Edge

Deployment concepts SD-WAN, Edge, and SASE

Enterprises will have a WAN that connects the offices with data centers and the cloud. It includes multiple locations of the enterprise, including the location it has in the cloud. A common technology to set up a WAN was by using Multi-Protocol Label Switching as a private network. MPLS takes care of connectivity on OSI layers 2 and 3, the data, and the network layer. Before the cloud, MPLS was used in most WAN setups. The problem with MPLS, however, is that it is not very flexible. Typically, MPLS networks have fixed connections, providing secure tunnels to allow traffic between locations. With that, it is perceived as very reliable and secure. The problem is that MPLS is also complex to manage, costly to upgrade, and hard to scale.

In the cloud, we want flexibility and agility. The network must be able to support this, and that is where SD-WAN comes in. As explained in the previous section, it abstracts the controls from the physical layer, offering much more flexibility. To put it simply, we now only have software that represents our network. Managing the network is done by managing the software that can be configured and deployed to, in theory, any hardware stack.

SD-WAN allows for multiple types of connections and provides possibilities to route traffic in the most optimal way, choosing the best network path. This can be defined in the policies that we discussed earlier. With that, SD-WAN will offer better performance for applications and data transfer, regardless of where applications and data sit. It makes SD-WAN very suitable for creating networks that include cloud environments.

There is something that must be carefully considered, though, and that is security. MPLS is a private network, whereas SD-WAN is not. When we create an SD-WAN, we must apply security policies and implement tools since security is not natively integrated. We will take a look at SASE in a moment.

Before we get into SASE, we must understand the concept of edge computing. The idea of the cloud is that we have centralized environments with shared services, preventing enterprises from having to build their own data centers. But the cloud is a data center too, and they have locations all over the globe. Let us look at Azure: Microsoft has Azure data centers in a lot of regions and countries. If we look for locations in Europe, we will find the data centers in Ireland (Dublin) for North Europe and The Netherlands (Amsterdam) for West Europe. There are also sites in France, Germany, Switzerland, Greece, Norway, and Sweden. Several other sites, for instance, in Finland, Denmark, and Spain, have been announced. European customers can host applications and store data in one of the Azure data centers.

There might be applications and data that have to be really close to the user to avoid latency. If the company is in Portugal with critical workloads that require very low latency, hosting applications and data in a data center in France might not be the best solution. Even with the fastest network connections, latency might still be too high. For this, edge computing might be a solution. In most cases, we then have edge computers close to the customer, although managed from the cloud, even with the cloud services of a specific cloud provider. Examples are Azure Stack and AWS Outposts. Hence, we are bringing a piece of the cloud closer to the user.

One remark: not all applications require low latency, and edge computing might not be suitable for every use case.

Edge networking allows to bring network segments closer to the user in a distributed network but is centrally managed. Cloud providers offer solutions for edge networking. Azure has private Multi-access Edge Compute and Private 5G Core. Indeed, 5G is typically used to set up edge connectivity for obvious reasons: 5G offers high speed, low latency, and immense capacity.

AWS offers Wavelength that connects AWS services to the edge using 5G. In Google Cloud, we use the Edge Point of Presence that connects edge devices to the Google network. For this, Google uses internet exchanges from Internet Service Providers across the globe. If we want to connect various clouds with each other, we can use these internet exchange services. The well-known ones are Equinix IX and they enable peering between different networks of network providers, ISPs, and cloud providers. The technology used for this is Multi-Lateral Peering Exchange it uses BGP sessions to securely exchange routing information with connected networks.

All these connections must be secured, and for that, we use SASE. But what is SASE? It is an architecture concept combining SD-WAN, native cloud security, and zero trust principles. A zero-trust network verifies user identities and establishes device trust before granting access to authorized applications, preventing unauthorized access and limiting security breaches. To set up a zero-trust network using SD-WAN, we must also apply native cloud technologies:

Cloud Access Security Broker Controlling traffic between cloud applications and services.

Secure Web Prevents unsecured traffic from accessing the organization's internal network, also in the cloud.

Firewall as a Also referred to as Next Generation Firewall protects applications on layer 7, the application layer. Service includes URL filtering with black- and whitelisting, threat protection, and intrusion prevention.

Content Delivery This is a geographically distributed network of servers delivering content close to where the users are. The service is a solution to solve issues with latency but also enhances security since the CDN servers can function as a shield to the servers where the original content is stored.

Domain Name Services Authentication of DNS requests before communication with the systems of an organization is established.

Cloud providers offer native solutions to establish this, but as previously mentioned, we will find a variety of third-party solutions in the respective marketplaces, including SaaS offerings that are completely managed by the vendor.

We have explored solutions, concepts, and components to create networks. Now, let us discuss how we can manage the connectivity between clouds. That is the topic of the final section of this chapter.

Managing networks in multi-cloud

There are multiple ways to connect to the cloud and to connect clouds with each other. In this section, we will discuss the various technologies to create and manage connectivity, starting with the one that is still commonly used, the VPN. Next, we will explore the different direct connections that cloud providers offer. Connecting clouds with each other is typically done through brokerage, using a connectivity broker. That will be discussed in the final part of the section.

Setting up VPN

A VPN uses the internet as a carrier: what we do is that we define a tunnel over the internet by only allowing traffic from a specific IP range to a destination IP range or address, typically the address of a gateway in a public cloud. Hence, the only thing that we should do is assign IP addresses or the IP range from which communication is allowed and to what destination.

Assuming that the destination is in the cloud, we have to define a point where the connection terminates. We do not want traffic to be able to reach every single resource in our cloud, so we are terminating the VPN on one location in the cloud. From there, we define routing inside the cloud. This implies that we have a zone where it is safe to have a VPN connecting before other cloud resources are accessed. This zone is referred to as a Demilitarized Zone or a transit zone.

A DMZ is a network segment that sits between a public network, such as the Internet, and a private network, such as a company's internal network. The DMZ acts as a buffer zone that provides an additional layer of security for the private network by isolating it from direct contact with the public network.

In a cloud-based DMZ architecture, virtual servers or instances are typically used to provide the necessary network segmentation and security. Traffic from the internet enters the DMZ and is directed to these servers, which act as proxies or gateways, filtering and forwarding traffic to the appropriate destination in the private network. This allows organizations to expose certain services or applications to the internet while keeping sensitive data and resources protected behind the DMZ.

A basic architecture for VPN connectivity is shown in Figure

Figure 2.9: VPN connectivity to DMZ or Transit Gateway

Cloud providers offer solutions to securely set up VPN connectivity to their clouds using VPN gateways:

Azure VPN Gateway

AWS Virtual Private Gateway and Transit Gateway

Google Cloud VPN

VPN Gateway in Alibaba Cloud

These solutions commonly support the following standards:

OpenVPN, based on Secure Socket Layer and Transport Layer Security

IKEv2, based on IPsec

Border Gateway Protocol Used to establish routing between autonomous systems on the internet, such as cloud platforms and cloud services.

First, we must define our private cloud in Azure, AWS, GCP, or Alibaba, and we must define a segment where we can terminate the connection securely. Now, we can place a gateway in that segment. This is a software-defined device that usually provides several features, such as authentication, authorization, encryption, firewalling, and routing. As with any service, you can use a native service from the public cloud providers or a third-party vendor that is supported in that cloud. Common brands are Juniper, Cisco, Palo Alto, Fortinet, and so on. We must define which traffic is allowed to communicate with the gateway and the resources in our cloud. From the gateway, the traffic will then be routed to the destination resources.

As said, VPNs use the public internet as carriers. An alternative is setting up direct connectivity between the enterprise and the designated public cloud. We will explore the solutions in the following section.

Setting up direct connections

Secured, encrypted tunnels are safe, yet a lot of enterprises will prefer an even more secure solution that is also more stable and really dedicated just to the enterprise. Direct connections provide that type of connectivity. Basically, with a private, dedicated connection, we extend the corporate network into the cloud, something that is enabled by software defined networking.

Direct connections have benefits over VPNs:

Faster connection Direct connections often provide faster connection speeds compared to VPNs, as there's no overhead associated with encryption and decryption.

Simpler Direct connections are often easier to set up and maintain, as they don't require additional software or configurations.

No encryption Direct connections do not incur the performance overhead associated with encrypting and decrypting traffic.

A drawback of direct connections might be limited network access, as some organizations restrict access to their internal network from outside their physical location. On the other hand, VPNs have their downsides too. Typically, VPNs have slower connection speeds due to the overhead associated with encryption and decryption. VPNs might also be more complex to set up since often this requires additional software and configurations.

Now let's have a look at the various offerings that public clouds have for direct connections. In Azure, we can use ExpressRoute, deployed as point-to-point ethernet, Any-to-any IPVPN, or using a cloud exchange colocation. The latter will be discussed a bit further in this section.

The point-to-point ethernet provides connectivity on layer 2, the network layer, or layer 3, the data layer. Another option is any-to-any IPVPN, where we can even connect our Azure environment with an MPLS WAN network, which we discussed in the first section of this chapter. Connectivity will always be established over layer 3.

In AWS, we use allowing enterprises to connect routers or firewalls that are on-premises directly to a VPG in an AWS VPC. DirectConnect also allows for direct connectivity to AWS services, such as storage in S3. AWS uses private virtual interfaces to set up these connections. AWS uses colocations to establish these connections.

Google Dedicated Interconnect uses the same principles as AWS. In a colocation, a direct link from a customer is connected to the peering edge of GCP. From this peering zone, the connection is forwarded to a gateway in GCP.

The service for direct connectivity in Alibaba Cloud is offered through Express Connect.

Now, how are these connections managed? Most enterprises will use a connectivity broker or a telecom provider to establish and manage connectivity to clouds. In colocation data centers, all cloud providers have PoP, which is, simply put, the front door allowing entrance into the cloud. Network or telecom providers will also have these POPs, making it easy to

connect to the cloud POPs. In the case of dedicated connections, this is literally a network cable coming from a telecom or network provider going into the router of the cloud provider. That is what makes this a completely different solution than using VPNs over the public internet.

How does this work? Telecom and network providers have meet-me rooms or zones where they can connect customers to the peering zone of the preferred cloud provider. As a consequence and a benefit to the customer, the connection will be completely managed by the telecom or network provider. One other major advantage is that enterprises that are multi-cloud can get multiple connections to the respective cloud platforms yet managed by one connectivity provider, including secure access, monitoring, and logging. Companies such as Equinix and Digital Realty have global coverage and offer connection services to all major public clouds. The principle is shown in the following Figure

Figure Direct connectivity via telecom or network provider

One alternative way of establishing multi-cloud connectivity is by using Network as a Service A company such as Megaport offers this service.

Customers will need a port as a steppingstone in a datacenter of their choice and an account in the Megaport service. Next, we define the B-port or the destination cloud. Megaport will set up the connection and manage the bandwidth according to the data limits that have been set. Megaport does this for all public cloud providers but also for SaaS services and connecting on-premises datacenters. The concept of NaaS is shown in Figure

Figure 2.11: Concept of Network as a Service

Obviously, all of this would not be possible without the possibilities of software-defined networking.

We have virtualized our networks. The next steps are virtualizing our resources. In the next chapter, we will start with the storage and explore the storage solutions in clouds and how we can use these in multi-cloud concepts.

Conclusion

Moving workloads into the cloud starts with establishing connectivity. In this chapter, we discussed how we can set up connectivity to the cloud. Before we explored the various services that clouds offer, we studied the principles of SDN. In the cloud, we will not be managing physical networks but define networks as software. This is crucial since public clouds are, by default, multi-tenant, meaning that customers share services in the cloud. To define our own domain in the cloud, we must understand how to set up VPC that holds our workloads. To reach and access these VPCs, we must create secure connectivity. We can do that by defining tunnels over the internet or by setting up direct connections to the clouds we use. Next, we learned how the routing of traffic and protection of workloads can be configured in clouds using gateways, web application firewalls, and load balancers.

We extensively looked at software defined concepts such as SD-WAN, Edge, and SASE and learned how to extend and manage our corporate networks into the cloud. In the final section, the concept of NaaS was discussed, where a NaaS provider takes care of connectivity to various clouds and cloud services.

The next chapter will be about virtualizing and managing storage, exploring the different storage solutions that public cloud providers offer.

Points to remember

Software-defined networking is the cornerstone of networking in clouds; it allows for abstracting the functionality of networking from physical devices.

All network devices can be virtualized, including switches, routers, firewalls, and load balancers. In the cloud, these devices are offered as services.

SDN makes it possible to deploy corporate SD-WAN and extend the network to the cloud and the edge of the cloud, close to the user.

Corporate networks often use MPLS as a private network and are secure. When we create an SD-WAN, we must apply security policies and implement tools since security is not natively integrated. We do that through SASE.

We can establish connections to the cloud with private tunnels over the public internet (VPN) or by using direct connections that cloud providers offer.

Questions

Corporate networks still often use MPLS. This is considered to be secure. Why?

To isolate workloads in Azure, we must set up private network segments. What are these called in Azure?

SDN architecture consists of three layers. Name these layers.

What is the name of the direct connection service that GCP offers?

Answers

MPLS is considered to be a private network.

vNet.

Infrastructure, control, applications.

Direct Interconnect.

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 3

Virtualizing and Managing Storage

Introduction

One of the benefits of the cloud is the almost limitless amount of storage that is available to users. However, the cloud is a shared resource model, so we need to understand how storage works in the cloud. How do we make sure that we get the right amount of storage and the right type with the right performance? Cloud architects and admins should know about the different storage types, tiering, and automated scaling of storage. It is covered in this chapter, starting with explaining the virtualization of storage as the basic principle.

Structure

In this chapter, we will discuss the following topics:

Types of storage virtualization

Storage concepts in public clouds

Managing storage assets in multi-cloud

Managing storage lifecycles and tiering

Managing data access

Types of storage virtualization

Before we dive into the various types of storage virtualization, we must understand the concept of virtualization. The concept is the same as with networks that we discussed in the previous chapter. We are abstracting the functionality of providing storage capacity from the actual storage devices. Let us try to understand this in a very simple way. In the traditional on-premises datacenter, we would need to attach storage cabinets to our servers. If an application on a server needed more storage, we needed to extend the physical storage that was attached to that machine, for instance, by adding discs. With storage virtualization, we can create pools of physical storage devices, acting as if it was one storage device. The software will now allocate storage from that storage pool, using capacity that is available on any of the physical devices that are part of the pool. These physical storage devices are not visible to the user, and the virtual storage appears as one physical machine, drive, or a Logical Unit Number

To enable this, we must create A Storage Area Network first since we must connect physical storage devices with each other and to the network where the servers are with the hosted applications. A SAN is basically just a network of storage devices, where these devices are separated from the servers in their own dedicated network. This is the main difference with Direct Attached Storage where the storage device – typically discs – is directly connected to the server. The problem with DAS is that when the server fails, the storage fails with it, and that might cause data corruption. SAN is much more resilient.

To connect a SAN to servers, we have to look at the various layers of the SAN technology. Firstly, there is the host layer in the server that allows the server to connect and communicate with the SAN. In the traditional datacenter setup, this is typically done through Fiber Channel, a cable that connects the server host to a switch to which the storage devices are connected. This layer with switches, cables, and routers in the SAN is called the fabric layer.

The final layer is the storage layer. To enable traffic to communicate between the devices in the SAN, the Internet Small Computer Systems Interface protocol is used. iSCS operates over TCP/IP, linking data storage and providing block-level access. We will come to the differences between block and other storage types later in this section. Figure 3.1 shows the basic architecture of a SAN:

Figure 3.1: Basic architecture of a SAN

A virtual SAN does the same thing in principle. However, we need to get back to DAS first. vSAN was invented by VMWare as a solution to aggregate local and DAS and create a single storage cluster that would allow virtual machines to use one pool of storage devices within the virtualization cluster. In different words, by virtualizing the storage that was attached to the servers, it creates a virtual SAN using that storage, but now as a logical layer instead of physical machines in one network attached to server farms using a fiber channel. Figure 3.2 shows the setup of a vSAN:

Figure 3.2: Basic architecture of a vSAN (Host-based)

If we want to connect servers to the vSAN cluster that are outside of the cluster, we need to set up an iSCSI connection between these servers and the vSAN. In that way, we can also present the vSAN to servers that are not part of the cluster, enabling these servers to use the storage in the cluster.

Everything that we discussed so far was related to block storage. Now, let us explore the different types of storage. There are five types of storage virtualization:

The most common type of storage virtualization, where the virtualization host presents storage to the guest operating system. A good example is the virtual disks that are attached to a virtual machine. Figure 3.2 shows the host-based principle.

With this type of storage virtualization, we can create storage tiering in the hardware that we use to virtualize the storage. For example, Solid State Disks are used to provide high performant, fast storage, and hard disk for slower storage.

Some operating systems allow to implement array-based storage as a function of the operating system itself. An example of this is Windows Storage Spaces.

File This typically refers to Microsoft's Distributed File System The technology divides data into files across various file servers or Server Message Blocks but to the user, it appears that the data is still on only one file server.

Fiber This might be a confusing term since it typically refers to the virtualization of the SAN switches, enabling VMs to connect to a SAN or a storage array.

It all comes down to providing access to data that is still stored on a physical device but using virtualization, and we can use these devices in an optimized way by dynamically allocating storage space to that data. The virtualization software does that by creating maps of paths to where data is stored, typically through metadata. As with any form of virtualization, it abstracts the functionality of storage from physical devices. Administrators will not have to worry about the physical infrastructure anymore, making it easier to execute tasks such as backup and archiving data.

Understanding this principle is essential to grasp the different storage concepts. Next, we must understand the difference in storage types: file, block, and object.

With file storage, the data is not broken into blocks but written to storage as a complete file in a hierarchical folder structure. These folders can be placed in other folders. By using a directory path, the folder and the specific file are located. File storage is typically used by Network Attached Storage as centralized, dedicated file storage to multiple clients such as servers or PCs. Applications and users will be able to retrieve documents, images, and other files, based on name and location.

Block storage divides data into equal blocks and then writes it to the storage device. It is a more efficient way of using storage capacity since it optimizes the allocation of the capacity. With block storage, a disk is

divided into maximum volumes and then attached to an operating system. The blocks do not contain any metadata, but each has a unique address. This address matches the stored data. The moment the data is accessed, the storage layer checks whether the addresses are correct. Do the addresses match? Then the data will be retrieved and presented. Block storage is the technology that is typically used in the cloud.

The last storage option that we must discuss is object storage. In this case, data is divided into separate units – objects – and spread across storage devices. However, the objects are stored in a single repository. One of the major differences between block and object storage is the fact that object storage does contain metadata. This metadata and the unique identifier of the object allow storage systems to retrieve the data quickly. Object storage is the best solution if we want to store large amounts of unstructured data because of the search capabilities – using the metadata – and the scaling that object storage provides. This type of storage is designed for storing large amounts of unstructured data, such as images, videos, and audio files. It is highly scalable and can handle a high volume of read and write requests. For that reason, object storage is the most popular form of storage in clouds.

In the next section, we will look at the various storage solutions that public clouds offer.

Storage concepts in public clouds

As you may have understood from the previous section, it very much depends on what we want to do or achieve in terms of using storage in the cloud. Cloud providers offer a variety of solutions, and every one of them serves a specific use case. As an example, on https://azure.microsoft.com/en-us/products/category/storage/ we find a comprehensive overview of all offerings that Azure provides. For AWS, we refer to

AWS and Azure offer a range of storage options to suit different needs, including file storage, object storage, and block storage, the various storage types that we discussed in the previous section.

One of the main storage options in both AWS and Azure is object storage. In AWS, this type of storage is provided by Amazon S3 Storage while in Azure, it is provided by Azure Blob Storage.

Another option in both AWS and Azure is file storage. This type of storage is used for storing files that can be accessed via the Network File System protocol. It is suitable for storing files that need to be accessed and edited by multiple users, such as documents and spreadsheets. In AWS, this type of storage is provided by Amazon Elastic File System while in Azure, it is provided by Azure Files.

In addition to object storage and file storage, both AWS and Azure also offer block storage options. In AWS, this type of storage is provided by

Amazon Elastic Block Store In Azure, it is provided by Azure Disk Storage.

Both AWS and Azure also offer options for storing data in a database. AWS offers Amazon Relational Database Service for storing data in a variety of relational database management systems, such as MySQL, PostgreSQL, and Oracle. Azure offers Azure SQL Database for storing data in a SQL-based database, as well as Azure Cosmos DB for storing data in a NoSQL database.

AWS and Azure also offer other types of storage solutions, such as archival storage and in-memory storage. Archival storage is used for storing data that is not accessed frequently and is typically less expensive than other types of storage. In-memory storage is used for storing data in memory for fast access, making it suitable for applications that require low latency.

Next, let us look at the storage options in GCP and Alibaba Cloud. GCP offers a range of storage options to suit different needs and use cases. Some of the main storage options provided by GCP are:

Cloud Object storage service designed for storing large amounts of unstructured data, such as images, videos, and audio files. It is highly scalable and can handle a high volume of read and write requests.

Cloud File storage service that allows us to store and access files using the network file system protocol. It is suitable for storing files that need to be accessed and edited by multiple users, such as documents and spreadsheets.

Cloud Bigtable: A NoSQL database service that is designed for storing large amounts of structured data. It is suitable for applications that require fast read and write performance, such as real-time analytics and Internet of Things applications.

Cloud SQL: A fully-managed SQL database service that is based on the MySQL and PostgreSQL database engines. It is suitable for storing structured data and running SQL queries.

Cloud Datastore: A NoSQL document database service that is designed for storing and querying structured data. It is suitable for applications that require flexible, scalable data storage.

GCP also offers other types of storage solutions, such as Cloud Memorystore for in-memory storage and Cloud Spanner for globally-distributed database storage.

Alibaba Cloud offers the following storage options:

Object Storage Service Object storage service designed for storing large amounts of unstructured data, such as images, videos, and audio files. It is highly scalable and can handle a high volume of read and write requests.

Elastic File System File storage service that allows for storing and accessing files using the NFS protocol. It is suitable for storing files that

need to be accessed and edited by multiple users, such as documents and spreadsheets.

Block Storage Service Block storage service that is used to store data in fixed-size blocks, like how a hard drive stores data. It is suitable for storing data that needs to be accessed quickly, such as the operating system and application files.

A fully-managed database service that is based on the MySQL, SQL Server, and PostgreSQL database engines. It is suitable for storing structured data and running SQL queries.

Table Store: A NoSQL database service that is designed for storing and querying structured data. It is suitable for applications that require flexible, scalable data storage.

Alibaba Cloud provides other types of storage solutions, such as MaxCompute for big data processing and Data Lake Analytics for data analytics.

We are talking multi-cloud, so we might want to run storage in various cloud platforms. How do we manage these storage assets in the various clouds? That is the topic for the next section.

Managing storage assets in multi-cloud

We talk about multi-cloud storage when we are using multiple cloud computing and storage services from different providers in a single environment. This may offer many benefits, such as increased flexibility, cost optimization, and improved data protection, but it also introduces new challenges in terms of storage management. In this section, we will discuss some best practices for managing storage in a multi-cloud environment.

One of the main considerations when managing storage in a multi-cloud environment is making sure that the data is stored in the most suitable location: the right cloud with the right solution that suits our needs. This may involve placing data in the cloud that is closest to the users who will be accessing it or choosing a cloud provider that offers the best price for a particular type of storage.

Next, we must consider the specific requirements of different types of data, such as whether it needs to be stored in a compliant manner or if it requires a certain level of performance. Compliance and performance are crucial requirements in picking the most appropriate type of storage solutions. Only when we have validated that requirements are met; can we start migrating data to that storage.

Migration can be a complex process, especially if we are moving large amounts of data between different cloud providers. We must carefully plan

the migration process and test the migration before moving all the data to ensure that everything goes smoothly. There are several tools and services available that can help with data migration, such as CloudEndure Migration, which allows for migrating data to the cloud with minimal downtime. CloudEndure Migration is an AWS solution – AWS acquired CloudEndure in 2019 - that helps in migrating workloads and data to AWS by means of agents that support lift and shift to the cloud. Basically, the solution continuously replicates the data between the source and the target environment, ensuring that there is no downtime and data loss during the migration.

Of course, there are many more solutions that will help in migrating data. Azure Data Box, Google Cloud Transfer Service, and AWS DataSync are good examples:

Azure Data Box is a physical device used to securely transfer large amounts of data to and from Azure. Customers can order a Data Box device and then use it to transfer data by loading it with their data and shipping it to an Azure datacenter, where the data is uploaded to their Azure account. Azure Data Box is useful for transferring large amounts of data more quickly and securely than over the internet.

Google Cloud Transfer Service is a managed service that enables customers to transfer data to and from the Google Cloud Platform. Customers can use Transfer Service to move data from on-premises storage, another cloud provider, or an internet-connected device to Google Cloud. Transfer Service supports a range of transfer protocols and provides scheduling, notification, and reporting capabilities.

AWS AWS DataSync is a managed data transfer service that simplifies moving large amounts of data to and from AWS. DataSync can be used to transfer data between on-premises storage, AWS storage services, and other cloud storage services. DataSync uses a high-speed, parallel data transfer mechanism to optimize transfer performance and supports automated scheduling, bandwidth throttling, and checksum verification to ensure data integrity.

In the next section, we will discuss one of the main risks in migrating and managing data, which is the loss of data.

Protecting data from data loss

Data loss is likely one of our main concerns in managing storage in the cloud. Once the data is in the cloud, it is important to ensure that it is protected against data loss and unauthorized access. This can be achieved using backup and recovery solutions, such as snapshotting and replication, as well as security measures, such as encryption and access controls. It is also important to consider the risks associated with storing data in a single location and to implement strategies to mitigate these risks, such as storing data in multiple locations or using a cloud provider with a strong track record of data protection.

In a multi-zone architecture, data is replicated across multiple availability zones within the same region. This ensures that if one availability zone goes down or experiences an outage, data is still available from another availability zone. Multi-zone architectures provide high availability and can help prevent data loss due to infrastructure failures or disasters within a single region. In a multi-region architecture, data is replicated across multiple regions, which are in different geographic locations. This ensures that if an entire region experiences an outage or disaster, data is still available from another region.

By using both multi-zone and multi-region architectures together, we can ensure maximum availability and data durability. Data can be replicated across multiple availability zones within the same region for high availability and then also replicated to multiple regions for disaster recovery.

When we talk about data, data migration, and protecting data, we also have to talk about the costs that come with managing the required storage. There is one huge misconception about storage costs in the cloud. If we enter a console of a cloud provider, we will find that storage can be really cheap per gigabyte. However, if we do not actively manage our storage, the costs can become very high. We need to regularly monitor and optimize the storage usage to ensure that we are not paying more than necessary. There are several ways to optimize storage costs, such as by reducing the amount of data that we store and by choosing the right type of storage to fulfill our requirements. Choosing the right type of storage is related to something we call tiering. We will discuss this extensively in the next section.

One of the key benefits of a multi-cloud storage strategy is the ability to take advantage of the strengths of different cloud providers. For example, we can choose to use one provider for data storage and another for computing resources or choose to use a combination of different types of storage - such as object storage and block storage - from different providers. By following these best practices, we can effectively manage storage in a multi-cloud environment and take advantage of the benefits that this approach offers.

Using third-party storage products

Something that we have not touched upon yet is the use of third-party storage products in the cloud. So far, we discussed the native storage offerings in the different clouds, but companies will likely still have data in on-premises storage that needs to be connected to the cloud. It might be a good decision to extend the technology that is used on-premises to the cloud. A reason to do so could be commercial benefits for using that same technology in both on-premises and cloud or for compliance reasons.

Products that are of interest here are, for instance, NetApp or file systems built with Lustre. Let us start with NetApp Cloud Volumes ONTAP, a popular solution to manage storage in various environments. ONTAP uses a scale-out architecture, which means that it can scale up or down in capacity and performance as needed by adding or removing storage nodes. The solution supports the most commonly used storage protocols, including NFS, SMB, and iSCSI, and storage use cases such as file, block, and object storage. NetApp is typically associated with so-called filers and physical storage systems, but the Cloud Volumes ONTAP solution can also be used in the cloud. For example,

AWS: Cloud Volumes ONTAP for managing storage on AWS instances, Cloud Sync for syncing data between on-premises storage and AWS, and Cloud Volumes Service for migrating and managing data in the cloud.

Azure: Cloud Volumes ONTAP for managing storage on Azure virtual machines, Azure NetApp Files for high-performance file storage, and Cloud Volumes Service for migrating and managing data in the cloud.

GCP: Cloud Volumes ONTAP for managing storage on GCP instances, Cloud Sync for syncing data between on-premises storage and GCP, and Cloud Volumes Service for migrating and managing data.

A basic architecture for NetApp ONTAP - in this example, AWS has been chosen - as shown in Figure NetApp ONTAP in AWS uses FSx, an AWS-managed service that takes care of the file shares in the storage environment:

Figure 3.3: Basic architecture of NetApp ONTAP in AWS

As Figure 3.3 shows, we can attach NetApp and non-NetApp storage on-premises or other clouds to the storage in AWS. The NetApp storage is synchronized with Snap Mirror and the other storage with, for example, Robocopy or Cloud Sync. NetApp has its own solution for this, called NetApp XCP, for data migration any-to-NetApp or NetApp-to-NetApp.

This is just an example of a third-party product that can be used to connect external storage to the cloud. Other examples include Veeam and Acronis.

The open-source file system Lustre is commonly used for applications that require fast and extremely scalable storage in the cloud, particularly in high-performance environments where data-intensive workloads are common.

In AWS, Lustre can be deployed on EC2 instances, and storage can be provided using Amazon Elastic Block Store or S3. Lustre can be used to store and manage data for a wide range of applications, including big data analytics, scientific simulations, and media processing. Think of use cases in financial modeling or in the development of new medicines using genomics.

In Azure, Lustre can be deployed on Azure VMs, and storage can be provided using Azure Premium SSDs or Azure Files. There are many more solutions to provision storage in the cloud. It is not a one size fits, especially when we are looking for solutions that connect on-premises systems to the cloud. A variety of hybrid cloud storage solutions might be explored, such as AWS Storage Gateway, Azure StorSimple, and Google Anthos:

AWS Storage Gateway is a hybrid cloud storage service that provides a way to connect on-premises applications to AWS cloud storage. It enables on-premises applications to access AWS storage services, such as S3, Glacier, and EBS, through standard storage protocols such as NFS, SMB, and iSCSI. AWS Storage Gateway can be deployed as a virtual appliance or as a hardware appliance.

Azure StorSimple is a hybrid cloud storage solution that helps organizations manage and store data across on-premises and cloud environments. It provides cloud-integrated storage, automated data tiering, and data protection capabilities. StorSimple uses a hybrid storage array that combines solid-state drives, hard disk drives, and cloud storage to provide high-performance and cost-effective storage.

Google Anthos is a hybrid and multi-cloud application platform that enables organizations to build, deploy, and manage applications across on-premises, multi-cloud, and hybrid cloud environments. Anthos allows organizations to modernize existing applications and build new cloud-native applications with Kubernetes, an open-source container orchestration system. Anthos also provides tools for application lifecycle management, service mesh, and policy management across multi-cloud environments.

Once we have deployed our storage solution, we need to design and continuously optimize our storage solution using tiering and lifecycle management to prevent storage costs from going out of control. This is the topic for the next section.

Managing storage lifecycles and tiering

In the previous section, we discussed the need to choose the appropriate storage solution to match our requirements, optimize the solution, and manage costs. Storage Lifecycle Management and tiering can be of great help in this. It involves the management of data storage from the time it is initially created, to the time it is no longer needed, including the transition of data between different storage tiers through lifecycles. First, we will need to understand the principles of tiering.

Data storage is typically classified into different tiers based on its accessibility and usage. The most expensive tier is usually the one that provides the highest level of accessibility and performance, such as the cloud provider's native object storage or a cloud-based file system. Examples would be Azure Blob and AWS EFS. The least expensive tier is typically the one that provides the lowest level of accessibility and performance, such as cold storage or archival storage.

A good example of typical archive storage would be AWS Glacier, which is used for low-cost backup and storage of data. But even when we use Glacier, we still have options that we must evaluate. If we require fast retrieval of archived data, we should choose S3 Glacier Instant Retrieval. If we do not need that fast retrieval but still need to keep the data stored for a long time, then S3 Glacier Deep Archive is a better choice. In that case, we must be aware that data retrieval can take up to 12 hours.

Still, we are not done yet. Storage pricing for Glacier starts at 0,0036 cents per Gb when we host in US-East. That is the pure cost of the storage. The retrieval costs come on top of that, and that comes in three offerings: expedited, standard, and bulk. Something that is often forgotten is the fact that data transfer is charged as well: data transfer into Glacier is free of charge, and data transfer out of Glacier varies between 0,05 per Gb up to 0,09 per Gb per month. And then it also depends on where the data is transferred to: data transferred to an EC2 instance in the same region, for instance, is free of charge.

The other cloud providers have similar propositions, so it does make sense to have SLM in place. SLM helps organizations to optimize their storage costs by automatically transitioning data from the most expensive tier to the least expensive tier as it ages and becomes less frequently accessed. This process is known as For example, an organization may choose to store its most frequently accessed data in the cloud provider's native object storage and transition the data to cold storage or archival storage as it becomes less frequently accessed. Figure 3.4 shows an example of storage tiering:

Figure 3.4: Representation of storage tiering

Archive storage will usually have the least performance but will also use the most capacity. Data that requires fast access should be hosted on high-performant storage, but likely only in a limited capacity. Storage will typically be tiered from frequently accessed to archive and not the other way around, but in theory, this is, of course, possible.

In summary, SLM helps organizations to reduce their storage costs by automatically transitioning data to the least expensive tier as it ages. This can be a requirement for organizations that generate a large amount of data and need to store it for an extended period of time. We can apply SLM too to enhance the performance of our storage systems through the same principle: by storing frequently accessed data in the most expensive tier and less

frequently accessed data in the least expensive tier. This will be the use case for applications that require fast access to data, such as databases and big data analytics platforms.

Lastly, SLM helps organizations to comply with regulatory and legal requirements by automatically transitioning data to the appropriate storage tier based on its retention period. This can be particularly useful and even a hard legal requirement for organizations that are required to store data for a specific period of time, such as financial institutions and healthcare organizations.

There are several factors to consider when implementing SLM in the cloud. First, it is important to define the storage tiers that will be used and the criteria for transitioning data between tiers. This may involve setting up rules based on the age of the data, the frequency of access, or the retention period. Next, we must consider the tools and technologies that will be used to implement SLM. This may involve using the cloud provider's native storage management tools, such as Azure Blob Lifecycle Management and AWS S3 lifecycle, or third-party tools that are specifically designed for SLM, such as Spectra StorCycle.

Be aware that implementing SLM and storage tiering must be tested intensively, especially in terms of the performance of the storage tiers. A lot of this can and should be automated. We can automate tiering, scaling storage, and even performance testing. Before we implement SLM, we must understand the various storage performance metrics. Understanding these metrics is crucial in determining the appropriate storage tiers and optimizing storage performance, also because cloud providers use these metrics to categorize the different tiers. Examples of these metrics are:

IOPS (Input/Output Operations Per This metric measures the number of read and write operations that can be performed per second. IOPS is important for measuring the speed of random-access workloads, such as databases.

This metric measures the amount of data that can be transferred per unit of time, usually in MB/s or GB/s. Throughput is important for measuring the speed of sequential access workloads, such as large file transfers.

This metric measures the time it takes for a storage system to respond to a request, usually in milliseconds. Latency is important for measuring the speed of workloads that require low response times, such as online transaction processing.

Queue This metric measures the number of pending I/O requests in a storage system. Higher queue depths can improve performance for workloads with high concurrency but can also increase latency and reduce throughput.

This metric measures the amount of data that can be transferred per unit of time, usually in bits per second. Bandwidth is important for measuring the maximum network speed that a storage system can achieve.

CPU This metric measures the percentage of CPU resources used by a storage system. High CPU usage can indicate that the system is processing I/O requests at its maximum capacity.

SLM is a best practice for managing storage tiers and related costs. But, we will have to consider other cost optimization strategies, too, such as data deduplication, compression, and thin provisioning can help us to manage storage costs in multi-cloud environments.

Automating tiering

Automating the tiering of storage involves the use of tools and technologies to automatically transition data from the most expensive tier to the least expensive tier as it ages and becomes less frequently accessed. We will discuss automating tiering for Azure and AWS.

In Azure, the Azure Storage Account can be configured to automatically tier data to the appropriate storage tier based on the age of the data or the frequency of access. This can be done using the Azure Management Portal or the Azure Storage REST API. Azure also provides several tools and technologies for automating the tiering of storage, including Azure Blob Storage Lifecycle Management, Azure Data Lake Storage Gen2, and Azure Archive Blob Storage. This type of action is executed by defining a rule stating that if data is not accessed at least once per month, it can be automatically moved to the archive.

For reference, an example for Azure in PowerShell format is provided as follows:

```
# List the files in the storage account

$files = Get-AzStorageFile -StorageAccountName -ShareName

# Filter the list of files based on last access time
```

```
$oldFiles = $files | Where-Object { $_.LastModified -lt (Get-Date).AddMonths(-1) }

# Copy the old files to the archive storage



foreach ($file in $oldFiles) {

  Start-AzStorageFileCopy -SourceUri $file.Uri -DestinationShare

-DestinationPath $file.Name

}
```

This is what it will do: the script will list the files in our storage account: The Get-AzStorageFile cmdlet is used to retrieve a list of all the files in that account. We will need to specify the name of the storage account and the file share or directory that we want to list.

Next, the list of files is filtered using the Where-Object cmdlet. It filters the list of files based on the last time they were accessed. We can use the Last-Modified property of the file object to determine when the file was last accessed.

Last, we move the filtered files to archive storage using the Start-AzStorageFileCopy cmdlet that copies the filtered files to a separate storage account or file share that we are using as an archive. Optional, we can now delete the original copies of the files. Be aware: the script

demonstrates how to list files, filter them based on their last access time, and copy them to archive storage.

In AWS, the Amazon S3 storage service can be configured to automatically tier data to the appropriate storage tier using the S3 Lifecycle policies. This can be done using the AWS Management Console or the AWS SDKs. Like Azure, AWS provides several tools and technologies for automating the tiering of storage, including S3 Intelligent-Tiering, S3 Standard-Infrequent Access, and S3 Glacier.

Automating scaling of storage

Automating the scaling of storage involves the use of tools and technologies to automatically increase or decrease the capacity of the storage system based on the demand for storage.

In Azure, the Azure Storage Account can be configured to automatically scale up or down based on the demand for storage using the Azure Management Portal or the Azure Storage REST API. Various tools and technologies are available for automating the scaling of storage, including Azure Storage Scalability and Performance Targets and Azure Storage Auto-Scale.

Obviously, S3 can be scaled automatically as well through the AWS Management Console or SDKs. Tools and technologies for automating the scaling of storage include S3 Auto Scaling and S3 Transfer Acceleration. S3 does not have a native auto-scaling mechanism for storage capacity, as it automatically scales based on the amount of data stored. We can enable S3 Transfer Acceleration for faster data transfer, but scaling the storage capacity itself is not required.

In the following YAML script, we define an S3 bucket and set up scaling policies to increase and decrease the number of replicas of the bucket based on the size of the bucket. This could be used as a template in CloudFormation. The script can be used in the scenario where we have an S3 bucket whose replicas are managed by an Auto Scaling Group. In this specific use case, the template is designed to increase or decrease the number of replicas based on the S3 bucket size. In most scenarios, S3

automatically scales to accommodate the amount of data stored without requiring manual intervention or scaling policies.

Note: Before we get to the increase and decrease statements, we will need to specify the resources and define the roles that these resources have in AWS. This is not included in the example. The script defines a CloudWatch alert that triggers the resizing of storage.

```
  IncreaseReplicaCountPolicy:

    Type: AWS::AutoScaling::ScalingPolicy

    Properties:

      AdjustmentType: ChangeInCapacity

      AutoScalingGroupName: !Ref MyBucket

      Cooldown: 300

      ScalingAdjustment: 1

  DecreaseReplicaCountPolicy:

    Type: AWS::AutoScaling::ScalingPolicy

    Properties:
```

```
    AdjustmentType: ChangeInCapacity
    AutoScalingGroupName: !Ref MyBucket
    Cooldown: 300
    ScalingAdjustment: -1
BucketSizeAlarmHigh:
  Type: AWS::CloudWatch::Alarm
  Properties:
    AlarmDescription: "Alarm when bucket size exceeds 10GB"
    MetricName: BucketSize
    Namespace: AWS/S3
    Statistic: Maximum
    Period: 300
    EvaluationPeriods: 1
```

```
    Threshold: 10000

    ComparisonOperator: GreaterThanThreshold

    AlarmActions:

      - !Ref IncreaseReplicaCountPolicy

  BucketSizeAlarmLow:

    Type: AWS::CloudWatch::Alarm

    Properties:

      AlarmDescription: "Alarm when bucket size falls below 5GB"

      MetricName: BucketSize

      Namespace: AWS/S3


      Statistic: Maximum

      Period: 300
```

Let us review the various parameters in the script in more detail:

This is a CloudFormation resource of type It defines a policy to increase the number of replicas in an Auto Scaling Group (referred by !Ref by 1 when triggered.

Similar to the previous resource, it's also of type It defines a policy to decrease the number of replicas in the Auto Scaling Group by 1 when triggered.

This resource is a CloudWatch alarm that triggers when the bucket size exceeds 10 GB. If this condition is met, the alarm will trigger the IncreaseReplicaCountPolicy scaling policy.

This is another CloudWatch alarm that triggers when the bucket size falls below 5GB. In this case, the alarm doesn't have any associated action, so it won't trigger any scaling policy.

We have studied examples in Azure and AWS for reference. Obviously, GCP and Alibaba Cloud offer services for autoscaling. GCP offers Cloud Storage Transfer Service and Cloud Storage Nearline for tiering and automated scaling of storage. In Alibaba Cloud, we use OSS Cold Storage and OSS Archive Storage.

By using the tools and technologies provided by the cloud providers, organizations can automate the process of transitioning data between storage tiers and scaling the storage capacity based on the demand, even real time. This can help organizations to optimize and scale their storage capacity to meet the changing needs of their applications and workloads.

Managing data access

Storage is just a technology that enables us to store data in a secure way. But we do not want anyone to be able to access the data. Hence, managing storage is mainly about managing data access. We will discuss in this final section.

Managing data access in a multi-cloud environment can be a complex task, but it is an essential part of ensuring the security, compliance, and performance of our data. First, we need to set policies that define who, when and why data may be accessed: we must specify exactly which users, groups, or applications are authorized to access which data. These policies should be based on the principle of least privilege, which means that users should only have the minimum level of access required to perform their job duties.

To achieve this, we have to use the Identity and Access Management tools in our clouds. For access management in data, we should consider multi-cloud management platforms like Flexera, Scalr, and Morpheus. These platforms offer centralized management of storage, compute, and networking resources across multiple cloud environments and can help simplify the process of managing storage assets in multi-cloud scenarios.

Most enterprises use Active Directory as main identity source. Alternatives for access management are Okta, ForgeRock, CyberArk and Ping. These tools can help to manage and enforce your data access

policies across multiple cloud platforms. They allow us to create users, groups, and roles with specific permissions, as well as to track and monitor access to data. But this is just a first step. Best practices are as follows:

Data encryption: Encrypting data is an effective way to protect it from unauthorized access, even if it falls into the wrong hands. Many cloud platforms offer encryption services, but we can also use third-party tools such as GnuPG or Cloud KMS to encrypt data before it is uploaded to the cloud. It is good to know that per January of 2023 all data that is entered in S3 buckets is encrypted at 256 bits per default.

Apply data access Access controls such as ACLs and object-level permissions can help in restricting access to specific data sets or objects. For example, we can use ACLs to allow only certain users or groups to read or write to a particular object in an S3 bucket or use object-level permissions to grant or revoke access to specific objects in a Google Cloud Storage bucket.

Monitor and audit data Regularly monitoring and auditing data access is essential for detecting and preventing unauthorized access. Many cloud platforms offer tools for monitoring and auditing data access, such as AWS CloudTrail and Azure Activity Logs. We can also use third-party tools such as Splunk or Sumo Logic to centralize and analyze log data from multiple cloud platforms.

None of this will be effective if we do not regularly review the data access policies. Policies and controls must be updated frequently to ensure that they are effective and aligned with the compliance standards and requirements of the organization.

This concludes the chapter about storage. We now have a good understanding of networks and storage in clouds. Next, we must have machines to run our workloads. Although containers and serverless are gaining popularity, a vast amount of workloads still run on virtual servers in the cloud. We will study setting up compute infrastructure in the next chapter.

Conclusion

In this chapter, we discussed the many various options that cloud providers offer in terms of storage. First, we learned the differences between storage types. It is important to be able to make a distinction between file, block and object storage in order to pick the right type of storage in the cloud.

Next, we learned that managing storage is mainly about optimizing the storage, for instance by using SLM and tiering. Data that is not frequently accessed could be moved to a cheaper type of archive storage, as an example, saving a lot of costs. We also learned that we should automate tiering as much as possible. In the last section, we studied the best practices of managing data access by implementing data access policies and controls.

We also discussed data protection. The importance of data redundancy and disaster recovery strategies in multi-cloud storage must be emphasized. In this chapter we learned about concepts of data replication across multiple cloud providers or regions, as well as disaster recovery.

Points to remember

Before we start defining and designing storage solutions in cloud, we must understand the differences in storage types, for instance block, file or object-based storage.

Cloud providers offer a wide variety of storage solutions. The appropriate solution must be chosen based on the requirements for our environment. Requirements can be set for performance, durability and compliance.

It is strongly recommended to apply storage tiering and SLM. Tiering and SLM can reduce storage costs by automatically transitioning data to the least expensive tier as it ages.

Storage holds data and that data must be protected at all times. Data redundancy and disaster recovery are crucial topics to consider in storage architecture and management. Monitoring and auditing of data access is essential to ensure security and compliance in multi-cloud storage management.

Cost management is an important aspect of storage management. The price per Gb of storage can be very low, but there might be additional costs for data transfer, for instance.

Storage is about data. This means that we have to consider strict principles and policies for accessing data. We need IAM to ensure that only authorized users can access data, but we have more controls at our disposal. For example, we can and should encrypt data.

Questions

What is the most common way of storage virtualization?

Name the three storage types.

What is the most used storage solution in Azure?

If we want to archive data in AWS, what solution can we use?

Answers

Host based

File, block, object

Blob

Glacier

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

Virtualizing and Managing Compute

Introduction

We discussed networks and storage, but there is one more component that we need, to run workloads in cloud: compute power. This chapter is about the basics of compute, starting with virtual machines which are still often used in clouds, despite the rise of serverless and containers (we will discuss this in later chapters). Public cloud providers offer a wide variety in compute power with different deployment models. We need to understand how compute in cloud works regarding, for instance, CPU and memory. Next, we will discuss the various deployment models, from spot to reserved. In this chapter, we will also look at the on-premises propositions of cloud providers, such as Azure Stack and AWS Outpost.

Structure

In this chapter, we will discuss following topics:

Comparing compute models in public clouds

Key considerations for choosing compute

Rightsizing compute in deployment and autoscaling

Exploring on premises propositions of public cloud

Deploy and manage compute assets from consoles

Automating infrastructure management

Comparing compute models in public clouds

We discussed setting up networks and connectivity in the cloud, and explored how we can store data. But we also need compute power to run applications that can process this data and present the outcomes to users. Azure, AWS, GCP, and Alibaba Cloud offer a variety of solutions to host applications and services. We typically refer to applications and services as workloads. Each of the mentioned providers offer different compute models to meet the needs of different types of workloads. Virtual machines, containers, and serverless are all different compute models that can be used to run applications and services in the cloud.

Before we start comparing, it is good to have an understanding of virtualization of servers, just like we did in the previous chapter about virtualizing storage. In the traditional data center, we had physical servers. Each server would host one application. The server was sized to the expected usage of that application, meaning that it was expected to use a certain amount of processing power and memory in the machine. However, it could happen that the application was not using all the resources that were available in the machine, or that the application was too heavy for the machine, maxing out the CPU and memory and with that causing severe performance issues or even crashes.

Virtualization of servers was invented to make more efficient usage of compute power by dynamically distributing the resources of servers across workloads. Server virtualization allows for running multiple virtual servers on a single physical server. The technology that enables it, is the hypervisor, which sits between the physical hardware and the virtual servers. The

hypervisor dynamically allocates the physical resources of the server, such as CPU, memory, and storage, to the different virtual servers. It also isolates the virtual servers from each other, so that they cannot interfere with each other's operation.

When a virtual server is created, the hypervisor creates a VM, that acts as a software representation of a physical server. The VM includes virtual versions of the physical resources, such as a virtual CPU, virtual memory, and virtual storage. The virtual server runs on top of the VM, and it is able to access the virtual resources as if they were real physical resources.

One of the main benefits of server virtualization is that it allows for greater resource utilization. Instead of having a single operating system and application running on a physical server, multiple virtual servers can run on the same physical server, each with its own operating system and applications. This allows for more efficient use of the physical resources, as well as more flexibility in terms of scaling and deploying new servers. With that, it is fair to say that virtualization is the corner stone of public clouds.

VMs are likely still the most popular compute model. VMs provide on-demand, scalable computing resources. They emulate a physical computer and can run a variety of operating systems and configurations. They are typically used to run traditional, monolithic applications that require a full operating system and a specific set of dependencies. VMs are well-suited for applications that require a high degree of customization and control over the underlying infrastructure.

Containers are becoming increasingly popular since they are lighter than VMs. Containers allow applications to be packaged in a portable and lightweight format. Containers are built on top of the host operating system, and they share the same kernel. This allows them to be more lightweight and

efficient than VMs. Containers are typically used to run microservices-based applications, which are composed of many small, independent services. They are well-suited for applications that require a high degree of scalability and portability. The difference between VMs and contains is shown in Figure

Figure 4.1: Difference between VMs and containers

Serverless is the last compute model that we will discuss. The term serverless might cause confusion since there is still server technology involved. However, now, we can run code without the need to provision or manage servers. In a serverless model, the cloud provider is responsible for managing and scaling the underlying infrastructure, and users only pay for the compute resources used by their code: think of processing power of the CPU and memory. Serverless is typically used to run event-driven, and code-based applications, which are composed of small, independent functions. They are well-suited for applications that require a high degree of scalability and cost-efficiency.

Now, let us explore the various propositions that Azure, AWS, GCP and Alibaba Cloud offer. In Azure, we can choose the following services:

Virtual machines

Azure Container Instances to run Docker containers in serverless mode.

Azure Functions as a serverless compute option.

AWS offers several compute models, including:

Elastic Compute Cloud to run VMs.

Elastic Container Service to run and manage Docker containers.

Lambda as a serverless compute option.

Google Cloud Platform offers:

Compute Engine for VMs

Kubernetes Engine to run containers on Kubernetes clusters

Cloud Functions as a serverless compute option

And lastly, Alibaba Cloud:

Elastic Compute Service for VMs

Container Service

Function Compute as a serverless compute option

In summary, Azure, AWS, Google Cloud, and Alibaba Cloud all offer a range of compute deployments to meet the needs of different types of workloads. These include virtual machines, container orchestration and serverless computing. The choice for a certain option will depend on the specific use case.

Key considerations for choosing compute

To cut it short, the key considerations in choosing compute instance, are cost, scalability and performance. Let us start with cost, since one of the misperceptions of using public cloud is that it is always cheaper. This is not the case. Cloud will bring a lot of benefits, but only if done right and all options have been explored in terms of 'best fit' to a specific business case.

Let us delve into the pros and cons of the various compute models that we discussed in the previous section.

When choosing between virtual machines, containers, or serverless technologies, there are several considerations to take into account. Each option has its own set of pros and cons, and the decision depends on factors such as the specific use case, scalability requirements, resource utilization, development workflow, and operational preferences.

Pros of using virtual machines include:

VMs provide strong isolation between applications, allowing you to run different operating systems and software stacks.

VMs can run legacy or specialized software that may not be compatible with other technologies.

VMs offer more configuration options, allowing you to fine-tune resources such as CPU, memory, and storage.

VMs provide a higher level of security through isolated environments.

Cons of using virtual machines are:

Resource VMs require a dedicated operating system and consume more resources, leading to higher costs and slower startup times.

VMs are slower to scale since they need to be provisioned and booted individually.

Management complexity: VMs require more manual management, including patching, updates, and infrastructure provisioning.

The pros of using containers are:

Containers share the host operating system kernel, resulting in faster startup times, lower resource consumption, and higher density.

Containers can be rapidly scaled up or down, allowing for efficient resource utilization and responsiveness to demand.

Containers can be deployed consistently across different environments, from development to production, reducing compatibility issues.

DevOps Containers integrate well with modern DevOps practices, enabling reproducible builds and streamlined deployment pipelines.

Cons of containers might include:

Weaker Containers share the host OS, so there is a possibility of security vulnerabilities or conflicts between containerized applications.

Limited Containers may not support applications with specific hardware or kernel-level dependencies.

Orchestration Managing containerized applications at scale requires additional tools (for example, Kubernetes) that add complexity to the infrastructure.

Lastly, using serverless options might have the following pros:

No infrastructure Serverless abstracts away the underlying infrastructure, allowing developers to focus solely on writing code.

Serverless platforms automatically scale resources based on demand, providing high scalability and cost-efficiency.

With serverless, you only pay for the actual execution time of your code, which can be cost-effective for sporadic workloads.

Easy Serverless platforms often offer seamless integration with other cloud services and event-driven architectures.

Serverless will also have cons. Although debatable, these might include:

Cold start Serverless functions experience a slight delay during the initial invocation due to container initialization (cold start).

Limited execution Serverless platforms impose limits on execution time, which may not be suitable for long-running or resource-intensive tasks.

Vendor Adopting serverless can make it challenging to migrate to a different provider or switch to a different architectural approach.

Ultimately, the choice between virtual machines, containers, or serverless depends on our specific requirements, the level of control and isolation needed, scalability demands, resource utilization goals, and the expertise of your development and operations teams. In some cases, a combination of these technologies may be the most suitable solution.

Each public cloud provider offers a variety of pricing options, including on-demand, reserved, and spot instances. On-demand instances are the most flexible option, as you pay for what you use without any upfront costs. Reserved instances, on the other hand, require an upfront payment in exchange for a lower hourly rate. This is an important topic: cloud is often chosen by companies since it is mainly Operational Expenditure meaning that the company only pays for what it is using at that time. With Capital Expenditure a company typically has to do upfront investments that it has to depreciate over time, regardless of if the machines that have been invested in are used or not. Now think about reserved instances once

more: in that case, we are investing upfront so that capacity is reserved. Indeed, this is considered CAPEX in the financial world and might lead to debates with controllers in companies.

Last type of instances that we have to mention are spot instances, which are available on AWS and GCP. These allow us to bid on unused capacity at a discounted rate but come with the risk of being terminated if the bid is too low. Hence, there is a high risk on the availability of these instances and therefore not recommended to run production workloads.

Another important factor to consider is scalability. All three public cloud providers offer scalable compute options, but each has a slightly different approach. AWS offers the ability to automatically scale instances up or down, based on demand, while Azure offers a similar option called GCP offers a more manual approach, where you can manually add or remove instances, based on demand. So, there are differences in approach of scaling between the providers.

The last point to consider is that of performance. Many factors can impact performance in cloud, such as:

Resource The amount and type of resources allocated to workloads can significantly affect performance. Factors such as CPU, memory, storage, and network bandwidth must be appropriately provisioned to meet the workload's demands.

The ability to scale resources up or down based on workload requirements is crucial for achieving optimal performance. Elastic scaling allows us to

handle sudden spikes in traffic or adjust resources during periods of low demand.

Network The speed and reliability of network connections between the cloud infrastructure and end-users or other services can impact performance. Higher latency will lead to slower response times and decreased user experience.

Geographic The physical location of our cloud infrastructure can affect latency and response times. Choosing data centers that are closer to our target audience or integrating Content Delivery Networks can help reduce latency and improve performance.

Storage The type and configuration of storage used can impact the performance of workloads. For example, Solid-State Drives generally offer faster read/write speeds compared to traditional Hard Disk Drives Refer to Chapter Visualizing and Managing Storage to read more about storage options in cloud.

Instance Cloud service providers offer different instance types with varying levels of computing power, memory, and networking capabilities. Choosing the right instance type for a workload's requirements is important to ensure optimal performance.

Software Optimizing software and application code can greatly enhance performance. Techniques such as caching, efficient algorithms, and parallel processing can help minimize resource usage and improve response times.

Load Distributing incoming traffic across multiple instances or servers can help distribute the workload and prevent any single resource from becoming a performance bottleneck. Load balancers ensure that requests are evenly distributed and help achieve higher availability and scalability.

Monitoring and performance Regular monitoring of the workload's performance metrics help to identify bottlenecks, analyze resource utilization, and make informed decisions for optimization. Utilize monitoring tools to track KPIs and identify areas for improvement.

Third-party Integrating third-party services, such as databases, content delivery networks, or external APIs, can introduce dependencies and impact performance. We must carefully assess the performance characteristics and reliability of these services before incorporating them into workloads.

All this needs to be considered in choosing the right compute option for our workloads. Additionally, staying up to date with cloud provider offerings and best practices can help leverage new technologies and optimizations for improved performance.

Different compute models may offer different levels of performance, depending on the type of workload. For example, if you need High Performance Computing for scientific or engineering applications, GCP offers Preemptible VMs, which are a cost-effective option for batch workloads and other fault-tolerant workloads.

The most important thing to consider is the specific needs of our workload. Each public cloud provider offers a variety of options,

including VMs, containers, and serverless computing. VMs are the most traditional option and offer the most control over the underlying infrastructure. Containers, on the other hand, offer a more lightweight and portable option, and are well suited for microservices and other cloud-native applications. Serverless computing, available on AWS, Azure, and GCP, allows you to run code without provisioning or managing servers, and is well suited for event-driven, ephemeral workloads. We will extensively talk about the different deployment options, in a further section of this chapter.

It is important to remember that we must evaluate and compare the different options to make an informed decision that aligns with the specific needs of our business.

Rightsizing compute in deployment and autoscaling

When it comes to cloud computing, rightsizing compute models can save both time and money. We must consider different topics when we define the size of our compute power. There is actually a big advantage here, in comparison to physical machines. When we had to order equipment in the data center, the architect also had to calculate the required capacity that a server should have. Ordering a bigger server would mean a higher investment. If that machine would sit idle for most of its capacity, it would be a disinvestment. Thus, the role of the architect was to define the right capacity for the workloads of the company.

It works the same in cloud, but the cloud is more flexible and has more options to really tailor the capacity to the requirements. Still, we must consider these parameters:

The type of workload that we are running. Different types of workloads have different resource requirements.

The number of instances that we plan to run. The more instances we have, the more resources we will need.

The size of the instances. Larger instances tend to have more resources than smaller instances.

The number of hours that instances are running. The more hours the instances are running, the more resources we will need to take into

account.

To define the capacity of compute models in AWS, we use the AWS Cost Explorer tool, which helps in identifying underutilized instances and resources. Additionally, we can use the AWS Auto Scaling feature to automatically adjust the number of instances based on the workloads. In Azure, we have the Azure Cost Management tool at our disposal. We can also use the Azure Autoscale feature to automatically adjust the number of instances. The alternatives in GCP are the Cost Management tool and the Autoscale feature.

Thus, when deploying an instance type, it is important to consider the right size in order to ensure optimal performance and cost efficiency. The size of the compute system refers to the number of resources, such as CPU and memory, which are available in the system.

One of the key considerations when determining the right size of compute in deployment is the workload that the system will be handling. If the workload is expected to be high, then a larger compute system may be necessary to ensure that the system can handle the increased load. On the other hand, if the workload is expected to be low, a smaller compute system may be sufficient. It might be tempting to select a smaller instance for cost reasons. However, selecting an instance type that is too small for the workload can lead to poor performance and decreased user satisfaction. On the other hand, selecting a system that is too large can lead to wasted resources and increased costs. Hence, there is a delicate balance that must be weighted in choosing the right instances or other forms of compute power.

Be aware that in this section, we mainly talk about instances or VMs. In serverless propositions this works different, since in that case, the automation will ensure that only the required CPU and memory is really allocated to the workload. Having said that, not all workloads – applications – are suitable to run serverless.

We already touched upon the cost of sizing. As the size of the compute system increases, so does the cost. Therefore, it is important to consider the cost-benefit trade-off when selecting the size of the compute system. To minimize costs, it is often best to start with a smaller compute system and then scale up as necessary.

Rightsizing using autoscaling

Once the right size of compute in deployment has been determined, it is important to be able to change the size as needed, preferably dynamically. One way to do this is by using autoscaling. Autoscaling is the process of automatically adjusting the number of resources in an instance, in response to changes in the workload. This allows the system to automatically scale up or down as necessary, ensuring that the right number of resources are available at all times.

AWS, Azure, GCP, and Alibaba Cloud all offer autoscaling capabilities for their computer systems. AWS offers autoscaling for its EC2 and ECS services, allowing users to automatically scale their compute resources in response to changes in demand. Azure offers autoscaling for its virtual machines, and GCP offers autoscaling for its Compute Engine instances. Alibaba Cloud also offers autoscaling for its ECS instances:

In AWS, Autoscaling can be achieved by using Amazon Auto Scaling, which allows us to automatically increase or decrease the number of Amazon EC2 instances or Amazon ECS tasks in a group based on CloudWatch Alarms.

In Azure, we can use Virtual Machine Scale Sets, which enables us to create and manage a group of identical, load balanced VMs.

In GCP, Autoscaling is done through Autoscaler, a tool that automatically adjusts the number of instances in a managed instance group based on demand.

In Alibaba Cloud, Auto Scaling is provided to automatically adjust the number of ECS instances based on demand.

By using autoscaling, the size of the compute system can be changed as necessary, ensuring that the right number of resources is always available. AWS, Azure, GCP, and Alibaba Cloud all offer autoscaling capabilities for their compute instances, thus making it easy to implement this feature in your deployment. There are a few things that we must be aware of in terms of using autoscaling. For one, autoscaling as it can lead to unexpected costs, if not configured properly. Automatic scaling can have impact on performance. This can be mitigated by monitoring the performance of instances and adjusting resources as needed. In other words, monitoring of resource utilization is crucial.

One topic that we have not yet addressed, is security. But it is crucial in selecting compute options and implementing automation such as autoscaling. It is important to note that while VMs offer stronger isolation compared to containers and serverless, they also come with increased overhead in terms of resource utilization and management complexity. Containers and serverless platforms provide more lightweight and scalable options, but they may require additional security measures and careful configuration to ensure isolation and protect sensitive data.

Security is a critical consideration when choosing compute models in the cloud. The choice between VMs, containers, and serverless can have

implications for isolation and data security. Let us now compare the three models that we discussed.

Virtual machines:

VMs provide strong isolation by running applications on separate virtualized operating systems. Each VM operates independently, reducing the risk of one application compromising the security of another.

Data VMs allow us to implement security measures at both the host and guest OS levels. We can apply security patches, firewall rules, and access controls specific to each VM. Data within VMs can be encrypted, and VM snapshots can help protect against data loss or corruption.

Containers:

Containers share the host operating system's kernel, which means that they have a lower level of isolation compared to VMs. However, containerization technologies employ mechanisms like namespaces and control groups to provide isolation between containers, preventing direct access to other containers' resources.

Data Container images should be carefully constructed to include only necessary components, reducing the attack surface. Isolating sensitive data within containers can be achieved using encryption and access control mechanisms. Regular image scanning and updates are crucial to address security vulnerabilities.

Serverless:

Serverless functions run in a managed environment provided by the cloud provider. Each function executes in its own isolated container or runtime environment. While serverless platforms handle isolation, the underlying implementation details and level of isolation can vary between providers.

Data With serverless, we rely on the cloud provider's security measures for the underlying infrastructure. Data security within serverless functions involves applying appropriate security controls to the code and using encryption techniques for sensitive data. The responsibility for securing the application and implementing access controls lies with the developer.

Regardless of the chosen compute model, best practices for securing cloud workloads apply, such as regular updates and patches, secure coding practices, least privilege access controls, encryption of data in transit and at rest, and proper monitoring and auditing of the environment. Evaluating the security features, compliance certifications, and documentation provided by the cloud provider is also essential in making an informed decision about the compute model that aligns with your security requirements.

Isolation and data security will also greatly drive the choice to keep workloads on premises. That is the topic of the next section, where we will explore other infrastructure propositions that cloud providers offer, extending the public cloud to on premises.

Exploring on premises propositions of public cloud

Many organizations must run applications and keep data on premises for various reasons. In a lot of cases, compliancy is an important driver to implement hybrid solutions. Some applications and sensitive data are kept on premises, whilst organizations can still benefit from the advantages of cloud. The major cloud providers provide solutions for these hybrid set-ups. In this section, we concentrate on the main offerings: Azure Stack, AWS Outposts and Google Anthos. Lastly, we will look at a solution that extends VMWare environments to public clouds.

Azure Stack, AWS Outposts, and Google Anthos are all hybrid cloud solutions that allow organizations to run their applications and services both on-premises and in the public cloud. Each of these solutions offers its own set of features and benefits, making them suitable for different use cases.

Azure Stack and Azure Arc

Azure Stack is a hybrid cloud platform from Microsoft that allows organizations to run Azure services on-premises. This solution is particularly useful for organizations that have strict data sovereignty and compliance requirements, as it allows them to keep sensitive data within their own facilities while still being able to take advantage of the scalability and flexibility of the Azure public cloud. Azure Stack also provides a consistent development and deployment experience, allowing organizations to use the same tools and processes for both on-premises and public cloud deployments.

The two major offerings of Azure Stack are Azure Stack HCI and Stack Hub. Stack HCI is built on top of the Windows Server Software-Defined program, which is a partnership between Microsoft and its hardware partners. In essence, it allows to run workloads on hardware of partners, such as Dell and HP, and to manage these workloads as if they were hosted in Azure.

Azure Stack Hub, on the other hand, is a hybrid cloud platform that allows organizations to run Azure services on-premises. It includes a subset of Azure services, such as virtual machines, storage, and networking, and allows organizations to develop and deploy applications using the same tools and APIs, as they would in the public Azure cloud. The main difference between Azure Stack HCI and Azure Stack Hub is that the former is focused on virtualized workloads, while the latter is focused on providing a hybrid version of Azure services.

One service that needs mentioning here is Azure Arc, which also allows to manage non-Azure workloads from the Azure cloud. The key feature of Azure Arc is its ability to extend Azure management capabilities to resources, regardless of where they are running. This can be in other clouds or on premises, and it includes Windows and Linux servers, Kubernetes clusters, and applications. In Arc, we use ARM templates to deploy and manage resources, Azure Policy for compliance and governance, and Azure Monitor for monitoring and logging.

AWS outposts

AWS Outposts is a hybrid cloud solution from AWS, providing an on-premises solution to run workloads, whilst managing these from the AWS cloud. Like Azure Stack, this solution is particularly useful for organizations that have strict data sovereignty and compliance requirements. AWS Outposts also provides a consistent development and deployment experience, allowing organizations to use the same tools and processes for both on-premises and public cloud deployments. One important thing to remember is that Outposts is a fully managed service, providing a solution for compute and storage. Outposts come in preconfigured computer racks, with customizable capacity.

With AWS Outposts, organizations can run a wide range of AWS services on-premises, including EC2 and ECS. Additionally, organizations can use the same management and operational tools and services on-premises, as they do in the cloud, such as AWS Management Console, AWS IAM, and AWS Security Group.

There is a second offering on AWS Outposts that includes the implementation of VMWare. This too is a fully managed service that allows organizations to run VMware Cloud on AWS on-premises. In this case, we can use the same VMware control plane and APIs on-premises, as we would in the cloud. We will talk about VMWare, later in this section.

Google Anthos

The hybrid solution that Google offers is Anthos. Like Azure Stack and AWS Outposts, it allows us to run applications on-premises, in the public cloud, or in other clouds such as AWS and Azure. However, Anthos is different than the solutions that we previously discussed. Of course, it is an on-premises solution, managed from GCP. So, firstly, we need to have our environment set up in GCP. Next, to use Google Anthos, we will need to have a Kubernetes cluster running on-premises. We can use the Anthos GKE on-premise, to set up a cluster, or connect an existing cluster to GCP. That is the big difference with the other propositions: Anthos is relying on Kubernetes.

Once our on-premises cluster is connected to GCP, we can use the Kubernetes Engine, Cloud Build, and other GCP services to deploy and manage applications.

Extending VMWare

Other hybrid options are VMWare on public cloud, extending on premises VMWare environments to respective public clouds. The basic architecture for this is shown in Figure

Figure Extending VMWare from on premises to various public clouds

The solution uses VMWare Aria, previously known as vRealize, that enables management of resources built with VMWare products across on premises infrastructures and various public clouds. It is beyond the scope of this book to discuss the different components of Aria. We refer to https://www.vmware.com/nl/products/aria.html for more information.

This is a popular solution, since many organizations and especially larger enterprises, have IT environments that are operated with VMWare virtualization products, such as vSphere for compute, vSAN for storage and NSX to virtualize networks. These organizations still want to benefit from the scalability and flexibility in cloud but might not want to change their technology stack completely. Extending VMWare to public clouds can be a good solution, providing a consistent way of managing assets on premises and in the cloud.

Deploy and manage compute assets

Deploying compute assets in cloud environments such as Azure, AWS, GCP, and Alibaba can be a daunting task. However, with the right set of best practices in place, it is possible to make the process more efficient and less error-prone. Let us now discuss some of the best practices:

First is the most obvious option that cannot be stressed enough: work with a plan. Before we begin deploying any compute assets such as instances, it is essential to have a clear plan in place. This should include details on the type of infrastructure we need, including instances and the number of instances that we require. The plan should also include the configurations that will be used. This plan will serve as a roadmap for our deployment and will help to stay on track throughout the process.

Next, we have to explore the different compute options that are available to us. Each of the cloud providers offer a variety of compute options, including VMs, containers, and serverless. It is important to understand the pros and cons of each option and choose the one that best fits our use case. For example, VMs are great for applications that require a specific operating system or software stack, while containers are more suitable for microservices-based applications. Serverless options are mainly used in event-based functionality.

In cloud, we want to automate as much as we can, which is easier than in the traditional data center. The reason for this is the fact that in cloud, we

work with code. Basically, everything is code which makes it easier to blueprint, template, script and with that, automate. We should use automation and configuration management tools such as Terraform, Ansible, and Chef. These tools can help streamline the deployment process and ensure consistency across different environments. These tools can automate the provisioning of compute assets, as well as configure and manage them once they are deployed.

Lastly, leverage cloud-native services to deploy and manage compute assets. For example, AWS offers ECS for container orchestration, Azure offers Azure Container Instances for serverless container deployment, GCP provides Google Kubernetes Engine for container orchestration, and Alibaba Cloud offers Alibaba Container Service for Kubernetes for container orchestration.

Once we have deployed assets, we need to manage them. That starts with observability: we have to monitor our assets to see how they perform, and to identify issues in a timely manner. All providers offer their own monitoring and logging tools: CloudWatch and CloudTrail for AWS, Log Analytics and Azure Monitor for Azure, Google Cloud Monitoring (previously Stackdriver) for GCP, and CloudMonitor for Alibaba Cloud.

Part of monitoring is making sure that our assets are secured. Security is a critical aspect of any deployment, and it is important to have security measures in place, to protect compute assets. Each of the cloud providers offers a variety of security options, such as network segmentation, firewalls, and security groups. It is important to understand the security options available and implement them as needed.

Now that we have learned the overarching guiding best practices, we can have a closer look at the various deployment methods that the cloud providers offer. Let us first explore the various methods in AWS:

Manual This is the simplest method, where we manually create and configure instances using the AWS Management Console or command-line tools. This method is suitable for small-scale deployments or for testing and development environments.

Auto This allows for automated increase or decrease the number of instances. In this case, we set up a group of instances, known as an Auto Scaling group, and configure policies to automatically add or remove instances from the group based on certain conditions.

Elastic This is a fully managed service that makes it easy to deploy, run, and scale web applications and services. It handles the provisioning of instances, load balancing, and automatic scaling for you.

EC2 This allows for provisioning a fleet of instances with a mix of instance types, purchase options, and Amazon Machine Images

This is the infrastructure-as-code service of AWS that enables to model infrastructure and provision it in an automated and repeatable way. We can use CloudFormation templates to define the configuration of your instances and other AWS resources, and then use the CloudFormation service to deploy and manage them.

This is a serverless compute service that we can use to run code without provisioning or managing servers. We can run Lambda functions in

response to certain events, such as changes to data in an S3 bucket.

ECS and These are container orchestration services that makes it easy to deploy, run, and scale containerized applications.

App This is a fully managed service that makes it easy to build and deploy containerized applications quickly. It automates the creation, configuration, and scaling of the underlying infrastructure and provides an easy-to-use console to define an application.

This is a deployment service that automates the deployment of applications to a variety of compute services, including EC2 instances, Lambda functions, and ECS tasks.

In Azure, we have the following options:

Virtual Machines This is the most basic method of deploying compute instances in Azure. We can manually create and configure VMs using the Azure Portal, Azure CLI, or Azure PowerShell. This method is suitable for small-scale deployments or for testing and development environments.

If we want to automatically increase or decrease the number of VMs, then we use Auto-scaling by setting up a group of VMs, known as an Auto-Scale set. Next, we configure policies to automatically add or remove VMs from the group based on certain conditions.

Azure Kubernetes Service This is a fully managed Kubernetes service in Azure. AKS handles the provisioning and management of the underlying infrastructure for containers that run on Kubernetes.

Azure Container Instances With this serverless compute service, we are able to run containers without provisioning or managing servers.

Azure App This is a fully managed platform-as-a-service that makes it easy to build and deploy web, mobile, and API applications. Azure handles the provisioning and management of the underlying infrastructure.

Azure The serverless compute service that enables to execute code without provisioning or managing servers. It is the equivalent of Lambda in AWS.

Azure This is a special one and can be best described as the Swiss Army Knife to build and deploy applications through CI/CD pipelines. Azure DevOps is a set of services that provides a collaborative environment for software development. We can use Azure DevOps to plan, build, test, and deploy code to Azure.

Azure Resource Manager These are JSON templates that we can use to define the configuration of Azure resources and deploy them in an automated and repeatable way. ARM templates are used to deploy and manage all types of Azure resources, including VMs, AKS clusters, and other Azure services. ARM is likely the service that administrators will use most of the time in managing and automating resources in Azure. We will look at an example in the final section of this chapter.

Let us now discuss the deployment options for compute in GCP:

Manual We can manually create and configure compute instances through the GCP Console or the command-line interface

Automatic By defining autoscaling groups, we can have GCP automatically create and terminate compute instances.

Preemptible This is a very specific GCP service. The service terminates instances that are not being actively used, to free up resources for other customers. It is somehow comparable to the technology that Azure and AWS use to free up capacity and offer spot instances.

Kubernetes GKE is the service to deploy and manage containerized applications using Kubernetes. Automated scaling is done through Cloud Run.

Cloud Deployment This allows us to use templates in YAML or JSON format to create and manage deployments. It also helps to automate the process of deploying resources, as we will show in the last section of this chapter with two examples.

Cloud This is the GCP equivalent of Azure Functions or AWS Lambda. With functions, we can run code without provisioning or managing servers.

By now, you should have noticed that all public clouds offer similar methods to deploy compute instances to their respective cloud platforms. To conclude this section, let us list the options that Alibaba Cloud offers:

Elastic Compute Service This is probably the service that is most used to create and manage compute instances in Alibaba Cloud. We can use it to deploy instances with both Windows and Linux operating systems in a wide variety of distributions.

Auto scaling: This service automatically increases or decreases the number of compute instances.

Container service: Deploying and managing containerized applications using Kubernetes is enabled with this service.

Serverless computing: The equivalent for the serverless options that AWS, Azure and GCP offer, allowing to run code without provisioning or managing servers. However, Alibaba Cloud also offers Function Compute that is specifically developed to build and run event-driven applications and services in the cloud.

It is clear that we have a variety of choices in terms of provisioning and managing compute manually or to automate this using the advantages of Infrastructure as Code Preferably, we want to automate as much as we can in cloud. In the final section of this chapter, we will look at that and give some examples on how we can automate IaC by using common YAML scripts.

Automating infrastructure management

Deploying compute assets in cloud environments such as Azure, AWS, GCP, and Alibaba can be a complex process, but with the right set of best practices in place, it is possible to make the process more efficient and less error-prone. By creating a clear plan, understanding the different compute options available, leveraging automation and configuration management tools, utilizing cloud-native services, utilizing monitoring, and logging tools, considering security, and testing and iterating, organizations can ensure that their compute assets are deployed successfully and are running smoothly.

However, we do not want to manage these assets manually, especially not in large environments. Hence, we need to automate our infrastructure management. This is what IaC enables. Most importantly, IaC allows for the use of version control systems such as Git, allowing for easy tracking of changes and rollbacks. However, there are more benefits from working with IaC. We can use reusable templates and modules, reducing the amount of code that needs to be written. Obviously, it enables the automation of the infrastructure provisioning process, reducing the risk of human error and increasing the speed of deployment.

In DevOps especially, the concept of IaC is very relevant. IaC empowers DevOps teams to treat infrastructure as code, bringing automation, consistency, scalability, traceability, and collaboration to the infrastructure provisioning and management processes. It aligns with the core principles

of DevOps, enabling faster and more reliable software delivery and promoting efficient collaboration between development and operations teams:

IaC allows infrastructure provisioning and configuration to be treated as code. By using declarative or imperative scripts, infrastructure can be automatically provisioned, configured, and managed, reducing manual effort and human error. This automation enables faster and more reliable deployments.

With IaC, infrastructure configurations are defined in a code repository, ensuring consistent and repeatable deployments across various environments, such as development, testing, staging, and production. It eliminates the inconsistencies that arise from manual configurations, reducing the risk of deployment issues and increasing reliability.

IaC allows for easy scaling of infrastructure resources by defining desired configurations and specifying the desired number of instances, load balancers, storage, and other resources. This scalability ensures that the infrastructure can handle varying workloads efficiently, improving performance and responsiveness.

IaC treats infrastructure configurations as code, enabling the use of version control systems like Git. This brings traceability to infrastructure changes, allowing teams to track and review modifications, roll back changes if needed, and collaborate effectively. Version control also helps in auditing and compliance by providing a clear history of infrastructure changes.

IaC encourages collaboration between developers, operations teams, and other stakeholders by providing a common, version-controlled codebase for infrastructure configurations. Infrastructure changes become more transparent, and teams can work together to improve processes, review code, and share knowledge. It also ensures that infrastructure can be easily reproduced across different environments or by different teams.

IaC integrates seamlessly with CI/CD pipelines, enabling the automation of infrastructure deployments along with application deployments. Infrastructure changes can be automatically tested, validated, and deployed in a controlled manner, reducing manual intervention, and accelerating the delivery of software updates.

IaC allows for automated testing of infrastructure configurations. Tests can be written as code to validate that the infrastructure is provisioned and configured correctly. These tests can check for security compliance, performance, availability, and other criteria, ensuring that the infrastructure meets desired standards.

IaC promotes agility by enabling infrastructure changes to be implemented quickly and reliably. Infrastructure can be easily modified or replaced as needed, providing flexibility to adapt to changing business requirements and technology advancements.

If we have all our assets as code, then we can have scripts to manage the code. Let us assume that we want to regularly deploy new instances of exactly the same type, with the same policies and configured in a consistent way. We can use scripting tools such as PowerShell, Bicep or in AWS CloudFormation templates to do so.

Automating deployment of instances in Azure could be done through scripting an ARM template. In that case, we will first need to define the variables: VM name, VM size, admin username, admin password, storage account name, storage account type, image and location. We can then put the script together in YAML, and it would be something like this:

```
# Resources

- name: myVM

  type: Microsoft.Compute/virtualMachines

  apiVersion: 2018-04-01

  location: "[parameters(’location’)]"

  properties:

    hardwareProfile:



      vmSize: "[parameters(’vmSize’)]"

    storageProfile:

      imageReference:
```

publisher: "[parameters(’imagePublisher’)]"

offer: "[parameters(’imageOffer’)]"

sku: "[parameters(’imageSku’)]"

version: latest

osDisk:

name: myOsDisk

caching: ReadWrite

createOption: FromImage

diskSizeGB: 30

vhd:

uri: "
[concat(’http://’,parameters(’storageAccountName’),’.blob.core.windows.
net/vhds/’,parameters(’vmName’),’-osdisk.vhd’)]"

osProfile:

adminUsername: "[parameters(’adminUsername’)]"

```
      adminPassword: "[parameters('adminPassword')]"


      computerName: "[parameters('vmName')]"

    networkProfile:

      networkInterfaces:

        - name: myNic

          properties:

            primary: true

            ipConfigurations:

              - name: myIpConfig

                properties:

                  privateIPAllocationMethod: Dynamic

                  subnet:

                    id: "[variables('subnetId')]"

- name: myStorageAccount
```

```
  type: Microsoft.Storage/storageAccounts

  apiVersion: 2019-04-01

  location: "[parameters(’location’)]"

  properties:

    accountType: "[parameters(’storageAccountType’)]"


- name: myNic

  type: Microsoft.Network/networkInterfaces

  apiVersion: 2018-04-01

  location: "[parameters(’location’)]"

  properties:

    ipConfigurations:

      - name: myIpConfig

        properties:
```

```
        privateIPAllocationMethod: Dynamic

        subnet:

          id: "[variables('subnetId')]"
```

In AWS, we would use CloudFormation templates for this. In the following example, we are deploying a t2.micro EC2 instance, again using YAML:

```
---

AWSTemplateFormatVersion: '2010-09-09'

Resources:

  MyEC2Instance:

    Type: 'AWS::EC2::Instance'



    Properties:

      InstanceType: t2.micro

      ImageId: ami-0ff8a91507f77f867
```

```
    KeyName: MyKeyPair

    SecurityGroupIds:

      - sg-01234567890abcdef0

    SubnetId: subnet-01234567890abcdef0
```

This example creates a single EC2 instance of type t2.micro using an AMI with ID using the key pair MyKeyPair for SSH access, within the specified security group and subnet . Obviously, we can replace the properties with other values, but it should be clear that the main principle is the same as in Azure: we define variables that can be inserted into a template that can be reused. It is the basic first step to automation using IaC.

In GCP, we can also use YAML to create templates. Like in Azure and AWS, we will need to specify some parameters to the instance, such as name, type and location. The template can then be deployed from the gcloud CLI, which is the easiest way to work with GCP. In the example, we have defined a VM with the name The only thing we need to next is triggering the command through

```
gcloud deployment-manager deployments create my-instance --config my-instance.yaml
```

Updating the deployment is also very easy:

```
gcloud deployment-manager deployments update my-instance --config
my-instance.yaml
```

In the past three chapters, we discussed virtualization of networks, storage and compute as foundation to understand how public clouds work. However, this book is about multi-cloud. How do we get environments in different clouds to communicate with each other? That requires interoperability. It is the topic for the next chapter.

Conclusion

In this chapter, we learned about the various compute models that we can deploy in clouds, from virtual machines to serverless options. We explored the many offerings that Azure, AWS, GCP and Alibaba Cloud provide, to run workloads from their platforms. It is not only about the public cloud. Azure, AWS and GCP have solutions for on premises too: physical machines that can be managed from the respective clouds to create a hybrid platform. We learned that in some cases, organizations need to have compute and data on premises, for instance, because of latency or compliance reasons. Next, we discussed how we can deploy and manage compute assets, starting with manual deployments and expanding to fully automated deployments including auto-scaling.

We also discussed the concept of IaC and the benefits of having everything as code. IaC plays a vital role in DevOps by bringing automation, consistency, scalability, and traceability to the infrastructure provisioning and management processes. In the final section, we provided examples to automate with IaC, using YAML.

Points to remember

Before we start deploying compute assets in cloud, we must consider the use case. For some applications, we might need virtual machines, but in more cloud-native environments, we can also use containers or even serverless options.

Organizations might have a need to keep environments on premises. There are solutions that cater for hybrid set-ups. These solutions include offerings such as Azure Stack, AWS Outposts, Google Anthos and VMWare, extended into public clouds using VMWare Aria.

The main concept of compute in cloud is built on the principles of IaC, making it easy to configure templates and blueprints and with that, create repeatable, scalable solutions to provision infrastructure in cloud.

Infrastructure is also about cost management. It means that we have to carefully consider the required capacity and right-size our assets in cloud. Autoscaling and automation can be of great help in right-sizing environments.

Questions

What do we need to enable virtualization of servers?

Rate false or true: right-sizing is important in terms of cost control.

What is the name of the on-premises proposition of AWS?

Name the three main compute models in cloud.

Answers

Hypervisor

True

Outposts

Virtual machines, containers, serverless

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

Creating Interoperability

Introduction

One of the biggest challenges in multi-cloud is interoperability. Systems must be able to communicate with each other between public clouds, between clouds, and on-premises stacks. Interoperability must be defined on various levels, and in this chapter, we will explain how we ensure interoperability between environments. We will learn why this is so important, as well as how we use solutions such as containers and service mesh to create interoperability.

Structure

In this chapter, we will discuss the following topics:

Defining interoperability

Requirements for interoperability

Explaining the difference between portability and interoperability

Solutions to create interoperability in public and hybrid clouds

Working with Open Compute Project

Defining interoperability

Ask any company why interoperability in the cloud is important, and the most likely answer is probably to avoid vendor lock-in. That is the rationale behind the decision to adopt a multi-cloud strategy and not to have all workloads in one cloud. Multi-cloud is a strategy that promises to have the best of various worlds, with opportunities to reduce costs whilst improving flexibility and scalability.

The simple definition of interoperability is the ability of different systems, applications, or components to communicate and exchange data with each other in a seamless manner. In the context of multi-cloud, interoperability refers to the ability of different cloud platforms to work together seamlessly and exchange data, services, and applications. With the increasing popularity of multi-cloud, it is becoming more critical for organizations to ensure that cloud platforms are indeed interoperable. However, the use of multiple cloud platforms also presents new challenges, particularly in terms of interoperability.

Some of these challenges are discussed as follows:

Although this might seem contradictory in itself, one of the biggest challenges with multi-cloud is the risk of vendor lock-in. You might wonder how that is possible. Each cloud provider may use different technologies, tools, and services, which can make it difficult for organizations to transfer their data and applications from one cloud to another. That is why we need technologies to achieve interoperability and the ability to move data and

applications between environments, thus helping to avoid vendor lock-in without encountering technical or compatibility issues.

Data gravity is often forgotten about when discussing interoperability and portability, but it can be a real issue. We talk about data gravity when data attracts more data and applications to it as it grows in size and value. If data can be easily transferred from one location to another without any loss of quality or function, then we use the term data portability. The problem is that data gravity can make it difficult for organizations to move their data to other platforms or services, as the data becomes more difficult to extract and transfer as its size and complexity increase. Data portability can help organizations overcome the challenges posed by data gravity.

Interoperability helps reduce costs. In multi-cloud concepts, we can take advantage of the best features and pricing offered by different cloud providers. For example, we can use one cloud provider for storage and another for computing, taking advantage of the lower prices and better performance offered by the different clouds. However, this will impose a real risk. Managing these various solutions might become a challenge: it means that we must develop skills in each of the solutions that we use. It is essential to include this in the entire business case. The costs for upskilling and the need to have staff trained in multiple solutions might outweigh the reduction of the actual cloud spent. Introducing, adopting, and managing the cloud means more than just the technology.

The possibility of switching between various clouds will make our environments flexible and scalable. For example, if one cloud suffers from an outage, we can fail over to another provider, even without interrupting services. Business continuity and disaster recovery are one of the most common use cases for adopting multi-cloud or hybrid cloud with on-premises systems and a public cloud. However, this also has consequences for the

management of these environments. They might become very complex in implementation and management. For one, we will need to keep both environments in sync, depending on the requirements that we define for Recovery Time Objective and Recovery Point Objective

We mentioned one use case, business continuity, and disaster recovery. Other use cases are cloud bursts from the data center to the cloud to increase capacity fast or move applications and data closer to the user to overcome latency issues. The latter use case is often used in content delivery. Not every cloud provider has the same global presence, making it necessary to work with various providers to enable the delivery of services to users in a specific area. This can also be for compliance reasons, for instance, delivery of services in the China region using a native Chinese provider such as Alibaba Cloud or Tencent, where we might use AWS in other parts of the world. It should be clear that interoperability is crucial in multi-cloud for multiple reasons.

There are several approaches that we can choose to achieve interoperability in multi-cloud, including the following:

Standardization is the first option that we must consider. We should adopt standards and protocols that are supported by multiple cloud platforms. Examples are the OpenStack open-source cloud platform or the Cloud Data Management Interface standard. The Open Compute Project is another example that we will study in detail in the final section of this chapter.

Use of Application Programming Interfaces APIs allow different cloud platforms to communicate and exchange data and services by providing a standardized interface for accessing data and applications.

Implement a hybrid cloud as a combination of public and private cloud services. By using a hybrid cloud, organizations can take advantage of the best features of both public and private clouds and move data and applications between them as needed. In the previous chapter, we discussed the on-premises offerings of public clouds, such as Azure Stack and AWS Outposts, that enable us to create a hybrid environment. A widely adopted solution is VMware, which we can use on-premises and in various clouds, enabling shifting workloads between the different platforms.

Cloud brokers can help us by providing tools and services to manage and integrate cloud deployments, including data migration, security, and monitoring. The setup of a cloud broker is demonstrated in Figure

Figure 5.1: Basic Architecture for cloud broker

The same applies to cloud gateways that provide a unified access point for cloud services, enabling access and management of multiple cloud platforms from a single interface. Examples of such gateways are the Internet Business Exchange of Equinix or the services that Megaport provides.

When we discuss cloud brokers and gateways, we should also elaborate on the benefits and limitations of these solutions. Generally spoken, brokers have the following benefits:

A cloud broker provides flexibility by offering multiple cloud service options from different providers. It allows users to select the most suitable cloud services based on their specific requirements, such as cost, performance, and geographic location.

Cloud brokers can analyze the pricing models and offerings of different cloud providers to help users optimize their costs. They can identify cost-effective options and provide recommendations to minimize expenses.

A cloud broker simplifies the management of cloud services by providing a single interface or platform to manage multiple cloud environments. It eliminates the need to interact with different cloud providers individually.

Cloud brokers can enforce security and compliance policies across multiple cloud platforms. They provide centralized control and visibility, making it easier to implement security measures and ensure compliance with industry regulations.

Some limitations would be the dependency on the cloud broker, possible vendor lock-in, and even potential latency. Since a cloud broker act as an intermediary between the user and the cloud providers, there may be additional network latency introduced in the data transmission process. The benefits of a gateway would be somewhat comparable to the usage of a broker:

A cloud gateway provides a secure and simplified connection between an organization's on-premises infrastructure and cloud services. It allows seamless integration and data transfer between the two environments.

Cloud gateways typically offer robust security features, such as encryption, authentication, and access control. They help protect data during transit and ensure secure communication between on-premises systems and cloud services.

Cloud gateways can optimize network performance by caching data, reducing latency, and optimizing bandwidth usage. They can accelerate data transfer and improve application performance.

Cloud gateways often support protocol translation, allowing different systems or applications with incompatible protocols to communicate and exchange data effectively.

Like brokers, gateways have downsides that must be considered carefully in the architecture:

A cloud gateway may have limitations in terms of the cloud services it can connect to. Some gateways are designed to work with specific cloud providers or have compatibility constraints, restricting the choice of available services.

Organizations need to deploy and manage the cloud gateway infrastructure, which may require additional resources, expertise, and maintenance.

While cloud gateways aim to improve network performance, there can be cases where the additional overhead introduced by the gateway negatively impacts network latency or bandwidth utilization.

If a cloud gateway becomes unavailable or experiences a failure, it can disrupt the connectivity between on-premises systems and cloud services, leading to downtime or service interruptions.

Interoperability is perceived as a must-do in a multi-cloud architecture. However, it requires a well-thought architecture that takes all aspects into account, including skills that we need to manage the solutions. Interoperability, as such, can be a requirement for adopting the cloud:

It must be possible to adapt the cloud service to requirements that are specific to a particular business. Think of specific business rules and associated values. This needs to be considered with every service of the provider because many cloud provider processes can and will be set up quite standardly.

Organizations do not want to be too dependent on a specific supplier. Changes in a specific service should not have a direct impact on the user organization and the application landscape. This means that a supplier should not simply remove functionality and allow customers of a specific service to migrate to a new version of this service at their own speed. Interfaces should be backward compatible because they are intertwined with the application landscape of the organization that uses the service. Suppliers should also notify their customers of significant changes in a timely manner.

Sourcing choices are not forever, meaning that organizations must be able to easily choose other products. They must therefore require suppliers to have access to the data managed by the service so that it can be migrated to a new

product. Legal agreements are also necessary, for example, to cover the risk if the supplier goes bankrupt. Think of exit strategies and associated exit plans. Consider, for example, an escrow agreement that ensures that customers have access to the (intellectual) property if a supplier goes bankrupt. Having said that, it is very unlikely for the larger providers that we discuss in this book to go bankrupt, but services might be stopped, which effectively leads to the same situations.

In the next section, we will discuss the definition of requirements in more detail.

Requirements for interoperability

We learned in the previous section that interoperability is critical for multi-cloud computing to be successful. By using standardized interfaces, data portability, security, and flexibility, we can ensure that our multi-cloud environments are able to meet the specific needs that we discussed previously. In this section, we will explore the requirements for achieving interoperability in multi-cloud concepts, as well as look at some use cases for both on-premises to cloud and between various public clouds.

Requirements that we must consider achieving interoperability in multi-cloud would include:

Standardized To achieve interoperability, the various cloud platforms must have standardized interfaces that allow for seamless communication between them. This includes APIs, protocols, and data formats that are common across all platforms.

This is very obvious but still very important to mention. Interoperability must not compromise security. Cloud platforms must implement robust security measures to ensure that data is protected as it is transferred between platforms. Having said that, cloud platforms such as Azure, AWS, GCP, and Alibaba Cloud are probably the best-protected platforms in the world. They have no other option than to implement maximum security for the thousands of customers that they host in their clouds.

However, these clouds offer customers an extensive toolbox with security controls. It is always up to the customer to apply these controls.

Interoperability must be flexible enough to accommodate different use cases and requirements. This includes the ability to easily add or remove cloud platforms from a multi-cloud environment, as well as the ability to scale resources as needed.

Data As we discussed, with data gravity being something that we must avoid, data must be able to move easily between different cloud platforms. Data portability is likely the most important requirement. This requires that the data be stored in a format that is compatible across platforms, such as Open Data Formats.

Data portability has become increasingly important. Organizations may want to migrate their data from one cloud provider to another due to factors such as cost, performance, or service offerings. Data portability allows for a seamless transfer of data, minimizing disruption and enabling a smooth transition. However, there are many more use cases, for example, when we have to upgrade software. When organizations upgrade their software systems or switch to different applications, they often need to transfer their existing data to the new system. Data portability ensures that important data is preserved and can be effectively migrated to the new platform. Other examples include:

Data backup and disaster Having data portability enables organizations to create backup copies of their data and store it in different locations or systems. In the event of a data loss or disaster, data portability allows for the recovery of critical data from backups, ensuring business continuity.

Cross-platform Data portability facilitates collaboration between different platforms or applications. It allows users to share and transfer data seamlessly between systems, enabling teams to work together, integrate workflows, and leverage data from various sources.

Personal data Individuals may want to have control over their personal data and be able to move it between different online services or social media platforms. Data portability empowers individuals to manage their data, switch platforms, or delete their data from a particular service.

Compliance with data Data portability is often a requirement under data protection regulations, such as the General Data Protection Regulation It ensures that individuals have the right to receive their personal data in a structured, commonly used, and machine-readable format, allowing them to transfer it to another organization if desired.

These are just a few examples of how data portability can be utilized in various scenarios. The ability to transfer data seamlessly between systems and platforms enhances flexibility, data control, and interoperability in digital environments.

If we follow these practices, we can create interoperability from on-premises to the cloud and between various clouds. Both use cases might be equally important for enterprises. Think of migrations of workloads: many organizations are looking to move some or all of their on-premises workloads to the cloud. Interoperability is essential for this migration, as it allows for the seamless transfer of data and workloads between the on-premises environment and the cloud.

In multi-cloud, we will use different multiple public cloud platforms for different purposes. For example, AWS for computing and storage and GCP for big data and analytics. Interoperability between these platforms is essential to ensure that data can be easily transferred between them.

One aspect of interoperability that must be addressed is unified monitoring. Interoperability implies that we will have multiple platforms where we host applications and store data. With that, the need for unified monitoring rises. Unified monitoring provides a single, centralized view of the performance and health of resources across multi-cloud environments. This allows us, as admins, to monitor and manage all of our cloud resources, regardless of the underlying cloud infrastructure or platform.

Without unified monitoring, we may have to use different tools and interfaces to monitor each cloud environment, making it difficult to get a comprehensive view of cloud resources and quickly identify and resolve any issues. It can also help us to optimize our cloud usage and spending.

You might be wondering why all of this is important. Let us illustrate that with an example in a specific industry: healthcare. We picked healthcare since interoperability in this industry can literally be lifesaving, as discussed:

In healthcare, patient data needs to be accessible to multiple stakeholders, including physicians, hospitals, and insurance companies. Interoperability ensures that this data can be easily shared and accessed regardless of where it is stored.

By having interoperable systems, healthcare providers can access and share relevant patient data in real time, leading to improved patient outcomes. This is especially important in emergency situations where quick access to patient data can mean the difference between life and death.

Healthcare organizations are subject to strict regulations regarding patient data privacy and security. Interoperability ensures that these regulations are adhered to as data is shared between different systems.

Interoperable systems can help healthcare organizations reduce costs by avoiding the need for redundant data storage and management systems.

Of course, this is just one example, but it hopefully makes clear why we should architect for interoperability. In the next section, we will study how we can manage interoperable systems with the use of containers and microservices. First, we need to understand the difference between interoperability and portability.

Explaining the difference between portability and interoperability

We mentioned data portability already a few times in the previous sections. The terms portability and interoperability tend to get mixed up. They are certainly related but are also quite different things, and it is important to understand the differences. We use the term interoperability to define the capability of easily moving workloads and data from one cloud provider to another or between private and public clouds. This also implies the portability of applications and especially data. So the question is: how are interoperability and portability different, and how are they related?

Interoperability describes the capability of systems, applications, or platforms to work together and exchange data without changing the data or the application. We speak about interoperable systems when these can communicate with each other and exchange information in a way that is seamless and transparent to the user.

Portability, on the other hand, refers to the ability of software or applications to run on different platforms or environments without any modifications or changes to the code. This means that applications can be easily moved from one platform to another, preserving their functionality and performance. Portability is important because it allows organizations to take advantage of new platforms and technologies as they emerge without having to worry about the compatibility of their existing applications.

It is important for organizations that are moving environments to the cloud as part of the digital transformation because portability helps in avoiding vendor lock-in since, in theory, organizations can easily switch to a different platform if needed without having to rewrite their applications from scratch. We state, in that in practice, this might be sometimes difficult, especially when native cloud solutions from a specific provider have been used.

Interoperability and portability both contribute to the overall flexibility and accessibility of data and applications. Interoperability helps ensure that different systems can communicate and exchange data, while portability ensures that applications can run on different platforms without altering the code.

In the context of cloud computing, interoperability, and portability are especially important. By ensuring that different cloud platforms and services are interoperable and that applications are portable, organizations can ensure that their cloud deployments are flexible and scalable and that they can easily adapt to changing business needs. This is the true goal of going multi-cloud.

Solutions to create interoperability in public and hybrid clouds

Why do companies hesitate to move their IT systems into the cloud? There are a couple of reasons for that, and all will be recognized. It can be because of compliance with local laws and regulations or because of technical constraints. But most companies hesitate because of the lack of cloud interoperability and the fear of

Achieving interoperability in systems, and particularly cloud, starts with architecting for interoperability. But first, we need to identify the areas where we need this interoperability:

Data Enabling data to be freely moved between platforms.

Application Enabling the use of application components such as PaaS on various platforms.

Platform This is mainly about the processing layer on top of the infrastructure. For example, can we move operating system images between platforms?

Application Enabling application or application components to communicate with each other, even when these are on different platforms.

That is what. Now, we have to define how since that will form the solution to create interoperability. Let us explore the

Adopt open Utilize widely adopted standards, such as HTTP, JSON, XML, and so on, to ensure that different systems can communicate with each other.

Use Design and implement APIs that allow different systems to communicate with each other. This needs more explanation since APIs are often seen as a sort of magic wand. First, we must understand the multiple forms of APIs and API architecture types.

Representational State Transfer This is the most used API type on the web. It can use JSON, XML, HTML, or plain text. REST APIs are very lightweight and flexible.

This is a query language for APIs and uses a schema to describe data. GraphQL is typically used in applications that require low bandwidth to fetch specific data in a single endpoint.

Simple Object Access Protocol This is used in secured environments. It is protocol independent, but it uses a strictly defined messaging framework.

Remote Procedure Call This is used for distributed systems as an action-based protocol, utilizing only HTTP GET and POST commands.

Apache This one is becoming more and more popular. It is used for event streaming and can process, publish, and store data as it occurs. It captures and delivers data in real time. Kafka utilizes TCP.

Implement security: Security is intrinsic. We have to make sure that secure communication between systems is established by using secure protocols such as HTTPS and by properly authenticating and authorizing access to

APIs. Remember data gravity. When we move applications or application components between platforms, we are likely also moving data with it. For a lot of companies, data is absolutely crucial and, therefore, must be extremely well-protected.

Consider Containerize applications to ensure that they can run in any environment, making them more portable and interoperable. We will discuss this in more detail in this section.

Utilize service Use service brokers that allow different cloud platforms to communicate with each other, such as the Azure Service Broker for AWS or the Google Cloud Interconnect for GCP.

Before we continue, we should not oversimplify the use of the services. APIs, for instance, can be very tricky. There are quite some challenges that we have to address in implementing and managing APIs across multiple cloud platforms:

API Different cloud platforms may have their own proprietary APIs, which can vary in terms of functionality, structure, and syntax. Ensuring compatibility between APIs from different providers can be complex and require additional development efforts to translate or adapt API calls.

Feature Cloud platforms may offer different sets of features and services. When using APIs across platforms, it can be challenging to find equivalent functionality or services in each platform. This can lead to limitations or gaps in the capabilities available through the APIs.

Authentication and Each cloud platform typically has its own authentication and authorization mechanisms. Managing user credentials and access control

across different platforms can be cumbersome, requiring additional effort to handle authentication and authorization logic consistently.

Error handling and Error handling and logging mechanisms can differ among cloud platforms. Consistently handling errors and logging information across different platforms and API responses may require additional code or integration logic to ensure uniformity.

Performance and APIs rely on network connectivity, and utilizing APIs across different cloud platforms introduces potential latency due to varying network infrastructure and data transmission protocols. Optimizing performance and minimizing latency in API interactions across platforms can be challenging.

Documentation and The quality and comprehensiveness of API documentation may vary across different cloud platforms. Inconsistent or inadequate documentation can make it more difficult to understand and effectively use APIs, leading to longer development cycles and increased troubleshooting efforts.

Versioning and Cloud platforms often release new versions of their APIs, introducing changes and deprecating older versions. Ensuring compatibility and managing version dependencies across multiple platforms can be demanding, requiring ongoing monitoring and maintenance.

Governance and When using APIs across different cloud platforms, it can be more complex to maintain governance and compliance standards consistently. Managing data privacy, security, and compliance measures across multiple platforms may require additional controls and monitoring.

Addressing these challenges require careful planning, architectural considerations, and robust integration strategies. API management tools and practices can also help mitigate some of the complexities and provide a unified approach to using APIs across different cloud platforms.

Now, let us look at the various services that cloud providers offer. For each of the clouds, we must consider the solutions that ensure interoperability. Let us have a look at some concrete examples in the various clouds, starting with Azure:

Azure Active Directory can be used for authentication and authorization across different systems and cloud platforms.

Azure API Management can be used to create, publish, and manage APIs to ensure secure and scalable communication between different systems.

Some examples in AWS are:

AWS Direct Connect can be used to establish a dedicated network connection between AWS and on-premises systems.

AWS App Mesh can be used to manage intercommunication between microservices, ensuring that they can communicate with each other securely and efficiently. We will talk about service mesh in more detail later in this section.

In Alibaba Cloud, we can mention:

Alibaba Cloud OpenAPI is a collection of APIs that allow developers to interact with Alibaba Cloud services programmatically. Think of creating and managing virtual machines and integrating Alibaba Cloud services with other cloud platforms and applications.

Alibaba Cloud Interconnect is a service that enables organizations to connect their on-premises data centers to Alibaba Cloud, creating a hybrid cloud environment.

Finally, here are some examples in GCP:

GCP Cloud Interconnect can be used to connect GCP to other cloud platforms or on-premises systems.

GCP endpoints can be used to manage APIs and ensure secure communication between different systems.

We need interoperability on network levels, data, and applications. Systems must be able to connect to each other in the first place, and then we have to make sure that data can be exchanged in readable and usable formats. Following that, applications must be able to process that data and present outcomes to users. Containers are a great technology to achieve this, but we will also explore micro-services and service mesh and how that can help in achieving interoperability.

We already mentioned that we should consider containers since they can help in achieving interoperability by enabling applications to run in any environment without dependencies on the underlying infrastructure. This abstraction from the infrastructure makes containers portable, meaning that they can run on any operating system and in any cloud environment. To

deploy and manage containers, we need an orchestration layer to operate these containers. Kubernetes has become the industry standard for container deployment and management. There are definitively more technologies available, but in this book, we will use Kubernetes.

First, let us understand what Kubernetes is. Simply put, it is an open-source orchestration system to deploy and operate containers. It features automated deployment, scaling, and management of containerized applications. How does that help in achieving interoperability? By using Kubernetes, organizations can run the same containerized application on different platforms, ensuring that the application can run on any infrastructure. That is the theory. Indeed, Kubernetes can be deployed on multiple cloud platforms such as AWS, GCP, and Azure. However, it does require specific Kubernetes runtime environments in these clouds. We will need Azure Kubernetes Services in Azure, Elastic Kubernetes Services in AWS, and Google Kubernetes Engine in GCP. However, the good news is that the deployment principles are the same, as shown in Figure

Figure 5.2: Basic architecture for Kubernetes

As shown in the figure, to operate Kubernetes in a cloud, we need to configure the specific cloud API that connects the control plane with the cloud services. If we have done that, then we can manage the clusters from the cloud service, such as AKS, EKS, and GKE. The control plane holds the scheduler for scaling the cluster nodes and the controller manager. The Kubelet is the node agent that connects to the control plane; the Kube-proxy or k-proxy takes care of the traffic to the node.

We might want to deploy and run containers on-premises, too, and even in edge or Internet of Things deployments, to enable applications to connect from edge devices to our clouds. For that, we have a lightweight version of Kubernetes called K3S. This was designed to run containers on edge and IoT devices, enabling applications to run on any environment, regardless of the underlying infrastructure, making it easier to ensure interoperability across different environments.

Thus, by using containers and orchestration engines such as Kubernetes and K3S, we can ensure that applications can run on any platform, making these applications more portable and interoperable. Figure 5.3 shows the principle architecture for K3S. It resembles the Kubernetes architecture. The main difference is that the control node runs a different server with the agent nodes, the Kubelets in the Kubernetes architecture, on separate devices:

Figure 5.3: Basic architecture for K3S

The agent nodes can run on devices with minimal resources. Containers can help us in achieving portability and interoperability. We also use containers to deploy services independently from each other in a micro-services architecture.

Let us recap once more what interoperability means: it is the ability of different systems, applications, or components to work together seamlessly. In the context of microservices, interoperability refers to the ability of individual services to communicate and exchange data with each other effectively. This can be a significant challenge in a microservices architecture, particularly as the number of services and their complexity increase. A basic architecture for micro-services is shown in Figure

Figure 5.4: High-level architecture for micro-services

Thus, interoperability does come with challenges that we must overcome, especially when we implement a micro-services architecture and work with containers. The advantage of micro-services is that we can use the services for multiple applications. When we have to upgrade, we do not need to bring down the entire application but only the services that are concerned with the upgrade, thus making development and operations more agile and flexible. The major challenge in a micro-services architecture, however, is to establish and manage the communication between the services. With the decentralization of functionality into individual services, the communication between services becomes more complex and difficult to manage. This can result in issues such as increased latency, decreased reliability, and increased latency.

Our micro-services will use data, and it might be the case that services use their own data stores. That might become a challenge in making sure that data is consistent across the various services, meaning that if data in a specific service is changed, these changes are reflected in all other associated services.

Keep in mind that micro-services, per default, operate independently from each other.

Lastly, we need to manage the configuration of the services. As every service is an independent function, it will require its own configuration. In complex environments where the number of services is vast, updates and changes of services can become challenging. It calls for some form of centralized management to keep the overview. That is where service mesh comes in as a solution.

Service mesh is a solution to these challenges of interoperability in microservices. It is, however, not easy to explain how service mesh works. In essence, it is a configurable infrastructure layer for microservices that enables the communication between the services by a common set of rules and functions that are used to manage the services and help to keep data consistent when it is transferred between the services. These common sets of rules and functions are attached to the service or app using sidecars. A typical service mesh architecture using sidecars can look like Figure

Figure 5.5: High-level architecture for service mesh using sidecars

To summarize it really shortly: service mesh helps reduce the complexity of managing the communication between services, and it improves the reliability of the overall system. It provides a centralized configuration mechanism that makes it easier to manage the configuration of services, particularly as the number of services grows. Services will be configured consistently whilst the risk of configuration errors is reduced.

Service mesh also provides improved visibility between services by providing detailed metrics and logging information about the communication between services. This information can be used to identify and resolve issues, improve performance and make informed decisions about the architecture and configuration of the system. A service mesh can be perceived as a platform that integrates security services into the micro-services architecture. Security policies are stored in so-called sidecars that can be connected to the various services, ensuring that each service is compliant with that specific policy.

Service mesh is a dedicated infrastructure layer that handles service-to-service communication in a microservices architecture. It provides a set of capabilities to manage and control the interactions between services. It takes care of the following functionality:

Service Service mesh typically employs a service registry to handle service discovery. The registry maintains a centralized catalog of available services and their network locations. When a service needs to communicate with another service, it queries the service registry to obtain the necessary network endpoint information.

Load Service mesh incorporates load balancing techniques to evenly distribute incoming requests across multiple instances of a service. It ensures optimal utilization of resources and prevents individual services from being overwhelmed. Load balancing algorithms can be configured within the service mesh to suit specific requirements, such as round-robin, least connections, or weighted load balancing.

Failure Service mesh helps manage failure recovery by implementing various mechanisms such as retries, timeouts, and circuit breakers. If a service encounters an error or becomes unavailable, the service mesh can automatically retry the request, apply timeouts to avoid waiting indefinitely or open a circuit breaker to stop sending requests to a failing service temporarily.

Service mesh collects metrics related to network traffic, latency, error rates, and other relevant data points. It gathers these metrics from the communication flow between services. These metrics provide insights into the health and performance of services and can be used for monitoring, troubleshooting, and performance optimization purposes.

Service mesh offers built-in monitoring capabilities to track and visualize the behavior of services within the mesh. It provides observability into communication patterns, latency, error rates, and other metrics. Monitoring tools and dashboards within the service mesh help operators and developers gain real-time visibility into the system, detect issues, and troubleshoot performance bottlenecks.

By abstracting these functionalities into the service mesh layer, developers can focus on building services without worrying about the intricacies of network communication, load balancing, or failure handling. Service mesh simplifies and standardizes these essential capabilities, making it easier to manage and scale microservices architectures effectively.

Obviously, we need technology to take care of all of this, preferably solutions that are agnostic to the different cloud platforms. Service mesh platforms such as Istio, Linkerd, or Consul often provide features and integrations to handle the discussed functionalities. They incorporate sidecar proxies (such as Envoy) alongside each service instance, enabling fine-grained control and management of service-to-service communication.

Some of the popular service mesh solutions that are available for different clouds include:

Istio is probably the most used solution for service mesh. It is an open-source solution that is supported by many cloud providers, including GCP, AWS, and Azure. It is also compatible with Alibaba Cloud Service Mesh.

Linkerd is also open-source and supported by many cloud providers. Linkerd provides a lightweight and efficient service mesh solution that is easy to

deploy and manage.

Next to these two popular cloud-independent solutions, the cloud providers offer service mesh solutions themselves as follows:

AWS App Mesh is a service mesh solution by AWS and, as such, integrated with the AWS ecosystem. It provides a native service mesh solution for micro-services that are deployed in AWS.

Google Cloud Traffic Director is the GCP service mesh solution. Google Cloud Traffic Director provides a fully managed service mesh solution for microservices deployed in GCP.

Microsoft Azure Service Fabric Mesh is the fully managed service mesh solution for micro-services that are deployed in Azure.

Lastly, we have to mention Alibaba Cloud Service again a fully managed service for micro-services in Alibaba Cloud.

To realize interoperability, it is important that cloud providers support open standards. The Open Compute Project is such a standard. We will briefly look at OCP in the closing section of this chapter.

Working with Open Compute Project

Interoperability is realized when technology providers agree on open standards. Let us use an example of the past: home video. In the 80s, there were three technologies introduced for home video: Video 2000, Betamax, and VHS. A Video 2000 tape could not be used in a VHS machine or vice versa. The consumer had to choose one system but was then constrained to the use of specific tapes for that system. The entertainment industry did learn from this: when the CD player was released, the manufacturers agreed upon one format. Be aware that in the gaming industry, we still must deal with different consoles that are not truly interoperable.

The key message is this: interoperability issues can be avoided if vendors support open standard initiatives. The OCP is such a standard. Better yet, it is a community of hardware and software developers dedicated to sharing designs for efficient data centers and cloud computing infrastructure. OCP was founded by Facebook in 2011 but has grown to include several big tech firms, including Microsoft, Intel, and Google.

In the multi-cloud world, the importance of the OCP has only increased. Interoperability and portability between different cloud providers is a major challenge for many organizations. The OCP helps to address these challenges by providing a set of standards and guidelines for data centers and cloud infrastructure, which can be used by any organization, regardless of which cloud provider they use. This helps to ensure that data centers and clouds can communicate with each other, share data and work

together, even if they are built on different hardware or software platforms.

OCP integrates with leading cloud providers, including AWS, Azure, and GCP. For example, AWS has adopted OCP hardware and standards for its data centers, and GCP has integrated OCP into its cloud infrastructure. Azure has also been a strong supporter of the OCP and is using its standards to help build their own data centers and cloud infrastructure.

It helps customers of these providers to easily move workloads between different clouds without having to worry about compatibility issues, avoiding vendor lock-in. The underlying hardware and software infrastructure is consistent, regardless of the cloud provider. This reduces the risk of technical issues and improves the overall stability of the infrastructure.

There is criticism, of course. OCP and using infrastructure standards are only a part of the interoperability solution. Standards, guidelines, and guardrails for exchanging data and coding applications are equally important.

Talking about data: the next chapter is about managing data in multi-cloud.

Conclusion

Companies try to avoid vendor lock-in, and that is the main reason behind implementing a multi-cloud architecture. Multi-cloud, however, comes with challenges. Interoperability and portability are the two biggest challenges. How do we make sure that applications and data can be freely moved from one platform to another, and how do we establish communication between systems that are on different platforms? In this chapter, we explained the differences between portability and interoperability. Next, we discussed various methods to create interoperability.

We extensively looked at containers, micro-services, and service mesh that will enhance the level of interoperability, but we also noticed that these cloud-native architectures can become complex, especially when our environments grow. Nonetheless, in multi-cloud, it is essential to have a good understanding of these emerging technologies. In the final section, we studied the Open Compute Project as a concept that aims at full interoperability between systems on-premises and in the cloud.

Points to remember

Interoperability and portability are related, but they are different things. Interoperability is about the capability to communicate between applications and exchange data, even when these are in different systems on different platforms. Portability refers to the capability to freely move applications and data from one platform to another.

Both interoperability and portability are essential in a multi-cloud architecture.

Cloud-native technologies such as containers and micro-services can improve both portability and interoperability. These architectures can become complex when environments grow with a lot of containers or micro-services that must be managed. Centralized orchestration and management plans such as Kubernetes for container management and service mesh concepts for micro-services can lift the heavy weight for administrators.

Questions

What has become the industry standard for container management?

What do we use when we attach configuration policies from a centralized vault to micro-services and apps?

What is K3S?

If an application or data attracts (more) data, we speak of?

Answers

Kubernetes

Sidecar

Lightweight Kubernetes, for example, IoT.

Data gravity

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 6

Managing Data in Multi-Cloud

Introduction

With multi-cloud, data can be everywhere in theory. It is one of the major challenges in multi-cloud: where is the data, where should it be, who may access it, who can access it, and how do we make sure that data is available at the right time and the right place? Hence, we need a clear data strategy on migration, where data is stored and who and what may access data, what the usage of data is, and how to prevent events such as data leaks and loss. Migrating and managing data in multi-cloud is all about data governance. In this chapter, we will cover the most important aspects of data management in the cloud.

Structure

In this chapter, we will discuss the following topics:

Defining a data strategy

Planning data migration

Managing data governance

Improving data quality

Securing data

Defining a data strategy

In today's digital era, data is increasingly becoming the true new gold for organizations. Almost every company will make an enormous effort to collect the right data, analyze it, and on the basis of the outcomes will make decisions for the future of the business. Organizations that effectively harness their data and turn it into actionable insights will make significant gains in terms of efficiency, productivity, and profitability.

However, with the explosion of data comes the need for advanced tools, infrastructure, and expertise to manage it effectively. This is where cloud computing comes in. Cloud computing has revolutionized the way businesses handle their data, providing them with an agile and scalable infrastructure that can store, process, and analyze vast amounts of data quickly and efficiently. However, it does not happen overnight. Companies will need to develop a data strategy.

In this section, we will discuss the required activities to define such a strategy. In terms of cloud computing, we will focus on the three major providers: AWS, Azure, and GCP. Let us go through it step-by-step. We define seven critical steps in defining a data strategy:

Define the business objectives: The first step in developing a data strategy for cloud computing is to define our business objectives. What are the key business problems our company is trying to solve with data? There will be various use cases. We might be looking to gain a competitive advantage through data insights or have a need to optimize business processes and

reduce costs. Once we have a clear understanding of the business objectives, we can then determine the type of data we must collect and analyze to achieve those objectives. Typically, this is done on the enterprise level of the company. The enterprise architect will closely work together with the business to determine the strategy. Based on the strategy, we can move on to the next step.

Determine the data requirements: This step is about the type of data that we must collect and analyze. This also determines what sources we will be using. What are our requirements regarding data quality and security measures? This is also a good time to identify any data governance and compliance requirements that we must adhere to. In the following section, we will extensively discuss data governance.

Develop the data architecture: The next step is to develop your data architecture. This includes designing the data models, data pipelines, and data workflows. We will need to determine how data flows through our architecture, how it is transformed and analyzed, and how it is stored and secured. This architecture is developed by the enterprise architect and the data architect. Make no mistake here: this is on the data layer that doesn't necessarily already involves technology. The technology will follow the data models.

Choose a cloud and a cloud provider: After defining the data architecture, we can further drill down to the technology layer. The first thing we must do on this layer is choose the right cloud and the associated cloud provider. Each cloud provider has its own strengths and weaknesses, so it is important to choose the one that best meets our specific needs. Azure is known for its integration with Microsoft products, AWS has a vast array of services, and GCP is known for its cutting-edge technology and innovation. Consultancy firms typically use battle cards to map the requirements of the enterprise to

the capabilities of a specific cloud, including the type of technology that can be used to meet the 6.1 shows an example of such a battle card:

Figure 6.1: Battle card for cloud comparison

The cloud infrastructure architecture: Once we have selected the cloud and cloud provider, we need to architect our cloud infrastructure requirements. What type of storage and computing resources do we need? What level of availability and redundancy does the company require? Cloud providers will help in providing best practices and blueprints for the so-called landing zones. That is the reason why Azure, AWS, GCP, and Alibaba Cloud have developed cloud adoption and well-architected frameworks that contain guidelines for setting up cloud infrastructure that must support specific use cases. It is strongly recommended to validate our architecture for a landing zone with specialists from the cloud provider that we work with.

Choose the data services: The next step is to choose the data services, for example, databases, warehousing, data lakes, and analytics services. However, we should also think of our disaster recovery and backup strategy.

With critical data, we might want to develop a strategy where data is backed up to a different platform, to ensure that we can always retrieve data. In designing such a strategy, it is also elementary to think about how we can keep data in sync. Each cloud provider offers a range of services, as we already have seen in the chapter about storage, which is closely related to data services. If we need a highly scalable and flexible database, we might choose AWS's Aurora or GCP's Bigtable. If you need a powerful data analytics platform, we may want to use Azure's Synapse Analytics or GCP's BigQuery.

Implement security measures: The final but crucial step is securing our data. We must include security measures to protect our data from unauthorized access, breaches, and other security threats. This includes implementing access controls, encryption, and monitoring tools to detect and respond to security incidents.

Developing a data strategy for cloud computing can be a complex exercise. We should consider topics such as data integrity and separation of concerns between the technology and data layer carefully. You might be wondering what we mean by that.

Data integrity refers to the accuracy, consistency, and reliability of data throughout its lifecycle. It ensures that data remains intact and consistent without any unauthorized modifications or corruptions. Maintaining data integrity is crucial for any data architecture because it guarantees the quality and trustworthiness of the information being stored and processed. Data integrity is crucial, or better said: reliable and accurate data is essential for making informed decisions. If data integrity is compromised, it can lead to incorrect analysis and flawed decision-making, which can have significant consequences for businesses. Data integrity builds trust and credibility among users and stakeholders. When data is consistent and accurate, it increases

confidence in the systems and processes relying on that data, fostering trust in the organization.

Next, data integrity is also important for compliance. Many industries have strict compliance requirements, such as healthcare (HIPAA) or finance (Sarbanes-Oxley Act). Data integrity plays a vital role in meeting these regulations, ensuring data privacy, and maintaining audit trails.

When we talk about data integrity, we must also address normalization. This is a technique used in database design to eliminate redundancy and improve data consistency. It involves organizing data into multiple related tables, each containing atomic and non-redundant information. Normalization helps eliminate data anomalies and redundancies by structuring data in a way that prevents duplicate or conflicting information. This ensures that changes made to data are propagated consistently throughout the system.

By reducing redundancy, normalization optimizes storage space and improves query performance. It allows for more efficient data retrieval and minimizes the risk of data inconsistencies or conflicts.

We discussed the fact that we distinguish a separate data layer and a technology layer in data architecture. It is important to have a good understanding of this separation of concerns. The data and technology layer are two very distinct artifacts, as discussed:

The data layer focuses on the organization, storage, and management of data. It includes activities such as data modeling, data integration, data quality assurance, and data governance. The data layer ensures that data is structured, standardized, and stored in a consistent manner, enabling efficient access and retrieval.

The technology layer encompasses the systems, tools, and infrastructure that facilitate data processing, analysis, and delivery. It includes hardware, software, networks, and platforms used to handle data. The technology layer enables data transformation, data analytics, and the presentation of data to end-users.

By separating data concerns from technology concerns, changes can be made in one layer without necessarily impacting the other. This modularity allows for flexibility and adaptability, making it easier to upgrade or replace technology components without affecting the underlying data structure.

Separating concerns allows individuals or teams to specialize in specific areas. Data professionals can focus on designing efficient data models, ensuring data quality, and establishing data governance practices. Technology professionals can specialize in selecting and implementing appropriate tools and technologies for data processing and analysis. This also ensures scalability and performance optimization: we can design the technology layer to scale horizontally or vertically to handle increasing data volumes and user demands without affecting the underlying data architecture.

There are many best practices that can help organizations achieve success. Here are a few:

Involve all business The success of any data strategy depends on its alignment with business goals and objectives. A strategy must be focused on solving the key business problems and delivering the expected outcomes. For that, we need the input of all business stakeholders.

Start small and Define a specific use case, pilot it, and next, refine it. If a use case is successful, we can expand the strategy. It is important to define our strategy and the solution with scalability as a requirement.

Design for Cloud provides organizations with an agile and scalable infrastructure that can grow and adapt to changing business needs.

Focus on data governance and Data governance and compliance are critical aspects of any data strategy, and it is important to establish robust policies and controls to ensure data quality, security, and compliance with regulations such as General Data Protection Regulation in the European Union and more specific regulations such as the Californian Consumer Privacy Act that were established in 2020. Both acts force companies to show exactly what data they collect, how they store and protect it.

Implement data We have to make sure that unauthorized access, breaches, and other security threats are detected on time and preferably avoided. If breaches materialize, we need to have plans for mitigating actions. This includes implementing access controls, encryption, and monitoring tools to detect and respond to security incidents. When discussing data security, we should emphasize security by design as the best practice. By integrating security into the data architecture from the outset, potential vulnerabilities and risks can be identified and addressed early on. This proactive approach minimizes the likelihood of security breaches, data leaks, and unauthorized access to sensitive information. Security by design ensures that appropriate measures are in place to protect the confidentiality, integrity, and availability of data. It involves implementing encryption, access controls, and authentication mechanisms to safeguard sensitive data against unauthorized access or modifications.

In the next sections, we will discuss these phases in more detail.

Planning data migration

Planning a data migration in a multi-cloud environment requires careful consideration of several factors to ensure a smooth and successful migration. In this section, we will discuss the best practices for performing data migration, targeting data storage in AWS, Azure, GCP, and Alibaba Cloud. Besides these best practices, we will also learn about some pitfalls that we should avoid.

First and rather obvious, we must define clear migration goals. This includes the data to be migrated, the timelines, and the expected outcome. Understanding the goals can help us identify the right approach, cloud provider, and tools for your migration.

Selecting the right cloud provider is not an easy task and should be very carefully considered. Each cloud provider has its own set of capabilities and features, which can impact your migration approach and timeline. For instance, AWS provides an extensive suite of migration tools, including AWS Database Migration Service, AWS Snowball, and AWS Transfer for SFTP. Similarly, Azure offers several migration tools, such as Azure Migrate, Azure Site Recovery, and Azure Database Migration Service.

Choosing the migration feature depends on a lot of aspects, for instance, the quantity of the data, but also how critical the data is. We will talk about data security in a later section but let us first discuss the quantity. The size of data files and the total of data that must be migrated will set the migration strategy. This can be online using an internet connection, but in a number of cases, this

will simply take too long. A common strategy is to freeze the data source and then copy the data to a device that can contain bulk data. AWS Snowball is such a device. The data is copied to the Snowball device, which is then transferred to an AWS datacenter. The concept is shown in Figure

Figure 6.2: The AWS Snowball concept

The data is now loaded to the AWS storage of choice, typically S3. As soon as the data is inside the AWS cloud, the data can be synced with the original data source up until the point where both the datastores contain the exact same data. From that point onwards, the original data source may be decommissioned since the data is now in the AWS cloud. Other clouds do offer similar strategies.

Of course, AWS is not the only provider that offers these services. Azure offers Azure Data Box, and in GCP, we can use Google's Transfer Appliance. Both offer physical data transfer solutions that help overcome challenges associated with moving large datasets to the respective cloud platforms. The Azure Data Box comes in various formats, such as:

Azure Data Box A portable SSD storage device available in capacities of 8 TB or 16 TB. It is suitable for transferring smaller amounts of data (up to 40 TB) and is typically used for offline data transfer.

Azure Data A ruggedized, tamper-resistant storage appliance with storage capacities of up to 100 TB or 560 TB. It is used for transferring larger volumes of data (from a few terabytes to multiple petabytes). Customers can connect the device to their network, copy the data onto it, and then ship it to Microsoft Azure for ingestion into Azure Blob Storage.

Azure Data Box A larger, rack-mountable variant of Azure Data Box with a capacity of 1 PB. It is suitable for massive data transfers and is typically used for offline data migration at scale.

Google's Transfer Appliance comes in two models: the Online model, with a capacity of 100 TB, and the Rack model, with a capacity of 480 TB. These devices allow for efficient and secure data migration in scenarios where transferring over the internet is not practical or feasible.

Before we can select the cloud and the migration tools, we must conduct an analysis of the data and the associated applications. A thorough analysis of the data and applications is absolutely required to determine the complexity, dependencies, and security requirements. This analysis will help us identify potential migration challenges and develop a migration plan accordingly.

Of course, we need to take data security into consideration from the start. We have to make sure that adequate security measures are in place to protect the data during migration. For instance, we must encrypt the data in transit and at rest to prevent unauthorized access. AWS offers several security services,

such as AWS Identity and Access Management and AWS Key Management Service to secure data during migration. As we have learned in Chapter Virtualizing and Managing when we studied the various storage concepts, all cloud providers encrypt data by default upon entry into the cloud. This is typically about data at rest. But since we are moving data, we also have to focus on securing data in transit. We can think of the following technologies to ensure secure data in transit:

Transport Layer Security (TLS) TLS is a protocol that encrypts data sent over the internet to ensure that it remains confidential and secure. It is important to ensure that TLS encryption is enabled for all data transfers between on-premises and cloud environments or between different cloud environments. All major public cloud providers, AWS, Azure, GCP, and Alibaba Cloud, support TLS encryption for data in transit.

Virtual Private Network VPNs provide a secure, encrypted tunnel for data to travel between on-premises and cloud environments. VPNs can help to protect data in transit by ensuring that it is not intercepted by unauthorized parties.

Direct In data transfer scenarios, direct connections might be a good option. AWS Direct Connect or Azure ExpressRoute can provide more secure and reliable connectivity for data migration.

Secure File Transfer Protocol SFTP is a secure protocol for transferring files over the internet. It uses encryption to protect data in transit, making it an ideal choice for migrating large amounts of data between on-premises and cloud environments. All cloud providers support SFTP for data transfers.

Multi-factor authentication MFA adds an additional layer of security to data transfers by requiring users to provide more than one form of authentication

to access cloud resources. This can help to prevent unauthorized access to data in transit.

A final step in planning the data migration includes test and validation. We must test and validate our migration plan before the actual migration. Tests will help spot potential issues and address them before the migration. It is crucial to design and execute comprehensive test cases that cover a wide range of data scenarios, including different data types, formats, and use cases relevant to your specific migration. Automated testing frameworks and tools can help streamline the testing process and ensure consistent and repeatable results. Tests must aim to validate the integrity, accuracy, and completeness of the data. Tests must include the following:

Data consistency and integrity:

Perform data integrity checks: Compare the source data with the migrated data in order to ensure that it remains consistent and accurate.

Verify data relationships: Validate the relationships between different data elements, such as foreign key constraints or referential integrity.

Check data types and formats: Ensure that the data types and formats are correctly preserved during the migration process.

Data completeness:

Conduct record counts: Compare the total number of records in the source data with the migrated data to ensure all records have been successfully transferred.

Validate data subsets: Verify the presence of specific data subsets or categories to ensure that critical data has not been omitted during the migration.

Data quality and validation:

Perform data quality checks: Evaluate the quality of the migrated data by comparing it against predefined quality standards and business rules.

Validate data transformations: If any data transformations or conversions were performed during the migration, validate that the transformations were accurately applied and did not introduce any errors.

Performance and scalability testing:

Assess data retrieval and processing performance: Test the performance of data retrieval and processing operations to ensure that the migrated data can be accessed efficiently in the cloud environment.

Evaluate scalability: Verify that the cloud infrastructure can handle the increased workload and data volume, particularly if the migration involves significant data growth.

Integration and interoperability:

Test data integration: If the migrated data needs to integrate with other systems or applications, validate the integration points and ensure data flows seamlessly between different components.

Validate data access controls: Test the access controls and permissions to ensure that only authorized users can access and modify the migrated data.

Error handling and exception scenarios:

Conduct error and exception testing: Test various error and exception scenarios to ensure that the system handles them appropriately and provides meaningful error messages or notifications.

Backup and recovery testing:

Test data recovery procedures: Validate backup and recovery mechanisms to ensure that in the event of data loss or corruption, data can be restored successfully from the backup.

All these steps are crucial in defining a data migration strategy. Skipping steps may lead to severe risks of data breaches. As an example, failure to consider dependencies can lead to data loss, data corruption, or application downtime. Therefore, it is crucial to analyze dependencies between data and applications and prioritize the order of migration accordingly.

Poor planning will definitively lead to delays, cost overruns, and failed migrations. A data migration plan must address all aspects of the migration, including timelines, budgets, and resource allocation. This includes the setup of the data governance, a topic that will be discussed in detail in the next section. Inaccurate data governance policies can lead to data breaches, compliance violations, and, again, loss of data. Data governance should include governance policies and tools in place to ensure data security, privacy, and compliance.

We already stressed the importance of testing. With proper testing, we can validate the capabilities and features of our designated cloud. This is a vital step in the data migration plan and the execution of that plan: we must understand the capabilities and features of the cloud provider and select the right migration tools and approaches accordingly. Lack of testing can lead to migration failures, data loss, and downtime.

It is strongly advised to have data governance in place before we start executing the data migration. It is the topic of the following section, where we will study the best practices for managing data governance.

Managing data governance

Implementing and managing data governance in multi-cloud concepts is essential to ensure that data is managed efficiently, effectively, and securely. Data governance involves a set of policies, procedures, and processes for managing data throughout its lifecycle. In a multi-cloud environment, where data is stored in multiple cloud environments, data governance becomes even more complex. Implementing proper data governance is a complex task in multi-cloud environments. In this section, we will explore some guidelines for implementing data governance, but first, we will study the concept of data governance in more detail.

Data governance involves the process of managing the availability, usability, integrity, and security of data used by an organization. It includes the creation of policies, procedures, and standards for managing data throughout its entire lifecycle, from creation to disposal. The primary goal is to ensure that data is managed in a way that meets business needs and regulatory requirements while also mitigating risks associated with data misuse, loss, or theft.

Data governance demands collaboration among different departments and stakeholders within an organization, including IT, legal, compliance, risk management, and business units. A data governance framework must be drafted to define the roles and responsibilities, policies, and procedures. Lastly, we need to define and decide upon the technology solutions that will support the secure management of data. The framework will contain principles on various aspects of data:

Data quality

Data security

Data privacy

Data lifecycle management

Data ownership

Data architecture

By implementing a robust data governance framework, we will ensure that data is accurate, secure, and compliant with legal and regulatory requirements. But what steps do we need to take, to compose this framework?

As a first step, we must define data governance policies that align with the goals, compliance requirements, and regulatory standards of the organization. This includes policies for data classification, data privacy, security, and retention. We must specify how critical data is, who is allowed to read, use or even change data, and how long the organization needs to keep the data, either directly available or in the archive. Who is responsible for managing these policies? The policies themselves are defined on an enterprise level, but we also need owners that make sure that policies are followed on specific data domains. In the industry, we refer to these owners as data They are responsible for managing specific

data domains or datasets. Data stewards are responsible for ensuring that data is accurate, complete, and available when needed.

Having stewards on specific data domains does not mean that data management should be distributed. It is recommended to have data management centralized. This will ensure that data is managed consistently across all cloud environments. This includes establishing a centralized data governance team responsible for managing data policies, processes, and procedures. Policies include adherence to data quality standards and ensuring that data is accurate, complete, and consistent across all cloud environments. Of course, we must monitor this. Just monitoring the infrastructure is not sufficient; we will also have to monitor the data usage to identify potential data breaches. We can do that by defining monitoring metrics such as usage patterns and data access protocols. Any deviation or abnormal usage will be flagged by the monitoring.

Before we can start monitoring, we first need to know what we monitor and where we monitor against. Hence, we must specify data security principles. How do we protect data from unauthorized access, modification, and deletion? This will lead to designing solutions for encryption, access controls, and data backup and recovery procedures.

All of this is about policies and technology, but we should not forget that we also need people to operate it all: the data administrators: Train data administrators to ensure that they have the necessary skills and knowledge to manage data in a multi-cloud environment. The role of the data administrator is critical in implementing and managing data governance in multi-cloud concepts. The administrator will have to have knowledge

about the various components of the cloud technologies that are used. The data administrator is responsible for managing data across all cloud environments, ensuring that data is accurate, complete, and secure and that data is being used appropriately.

Everything that we discussed above starts with the data administrator. He will define data quality standards and be responsible for setting up monitoring of data quality metrics. He is also responsible for defining data governance policies and procedures that align with organizational goals, compliance requirements, and regulatory standards. But even more important: he will need to work closely together with the data owners, the data stewards. These data stewards must be trained so that they have the necessary skills and knowledge to manage data in a multi-cloud environment.

To put it in simple words: the data administrator plays a critical role in implementing and managing data governance in a multi-cloud environment. However, it starts with having the right data in an organization. That data must have a specified quality to ensure that it is useful to the organization. In the next section, we will talk about data quality and how we can improve that quality using cloud technology.

Improving data quality

We discussed the governance of data management and the role of the data administrator. However, data has a purpose. Typically, organizations use data to make informed decisions, gain insights, and improve business processes. This can only be achieved if data meets quality requirements. However, maintaining data quality can be challenging when dealing with large volumes of data that are stored across different platforms and systems.

High-quality data is essential for accurate and informed decision-making. Business leaders rely on data to analyze trends, identify opportunities, mitigate risks, and make strategic decisions. If the data is of poor quality, decision-makers may draw incorrect conclusions, leading to flawed strategies and inefficient resource allocation. As an example, a retail company analyzes sales data to identify customer preferences and optimize inventory management. If the data contains errors, such as duplicate records or incorrect product codes, the analysis may lead to inaccurate insights, resulting in inadequate stock levels or wrong product promotions.

However, data quality also involves compliance and regulations. Compliance with data protection and privacy regulations is crucial for businesses across various industries. Think of the healthcare provider maintaining patient records. Inaccurate or incomplete patient data can lead to medical errors, compromised patient safety, and violations of privacy regulations, impacting the organization’s compliance and reputation.

Lastly, organizations increasingly rely on data-driven insights and innovation to gain a competitive edge. High-quality data serves as a foundation for advanced analytics, machine learning, and artificial intelligence applications, enabling businesses to uncover valuable insights and drive innovation.

First, let us explore the parameters that set out the quality of data. Data quality refers to the degree to which data meets the needs and requirements of its intended use. In other words, data quality is a measure of how well data satisfies its intended purpose. There are various dimensions that define data quality, including:

Accuracy: Data must be free from errors, omissions, or inconsistencies. Accurate data is essential for making informed decisions and avoiding costly mistakes.

Completeness: Data must contain all the necessary information for its intended use. Incomplete data can lead to incorrect conclusions and inaccurate insights.

Consistency: Data must be uniform and consistent across different systems, platforms, and time periods. Inconsistent data can lead to confusion and errors in decision-making.

Timeliness: Data must be up-to-date and relevant to its intended use. Timely data is essential for making informed decisions and taking timely action.

Relevance: Data must be useful for its intended purpose. Relevant data is essential for making informed decisions and gaining valuable insights.

Validity: Data must conform to the rules and requirements of its intended use. Valid data is essential for ensuring the accuracy and reliability of insights and decisions.

Improving data quality is important for almost every organization. A lot of companies today operate data-driven or might even use artificial intelligence To get the best decisions out of data-driven concepts and AI and continuously improve these decisions, we need data from which we can train these concepts. A good and very actual example of how this principle works is ChatGPT. Basically, ChatGPT formulates answers with the data that is available to ChatGPT. It cannot work with data it does not have, and we might miss part of the answer. We can tell ChatGPT that we think it misses data and provide that data, and it will then incorporate that data the next time. ChatGPT is a wonderful example of how AI works with data and how it trains itself with that data.

Cloud providers offer various tools and technologies that can help organizations improve their data quality by ensuring that data is accurate, consistent, and complete. Let us study some of these tools in the different clouds.

Data quality in AWS

AWS offers a suite for designing and managing data quality, as shown in Figure It shows the data quality cycle:

Figure 6.3: The AWS Data Lifecycle

To work according to this quality cycle, AWS offers a variety of tools, including:

AWS Glue DataBrew: This is a data preparation tool that can help us improve the quality of the data. It automatically identifies and fixes errors, such as missing values or inconsistent data types, in large volumes of data.

Amazon SageMaker Data Wrangler: Like Glue DataBrew, this is a tool for data preparation that cleans and transforms data. It automatically identifies and removes duplicates, outliers, and other data errors.

Amazon Redshift: This is probably already a well-known tool. It is a cloud-based data warehouse that automatically analyses data for errors and inconsistencies and provides recommendations for improving data quality.

Amazon This is a business intelligence tool that can help to visualize and analyze data. Like Redshift, it provides recommendations for improving data quality.

There is a very good blog post about data quality in AWS. It can be found at That post also discusses tools like Athena that help in analyzing data in S3 using standard SQL queries.

Data quality in Azure

Azure provides tools and services under the umbrella of Azure Data Quality Services The process within DQS works with two main stages, build and use. In the build, we explore the data and match policies against that data. In use, we correct, standardize and clean the data and then deduplicate. After cleansing and deduplication, the data flows back into the build for exploration and matching. This way, data quality is continuously improved. It is shown in Figure

Figure 6.4: The Data Quality Service Concept of Microsoft

In Azure, we can use a variety of tools:

Azure Data Factory: This is a cloud-based data integration service that automatically identifies missing values or inconsistent data types in large volumes of data. Data Factory is used to train data models by cleansing and deduplication of data.

Azure Data Lake Storage and Synapse Analytics: With these tools, we can analyze data for errors and inconsistencies and provide recommendations for improving data quality.

Azure Databricks: This is a very popular tool to clean and transform data. It can automatically identify and remove duplicates, outliers, and other data errors.

However, DQS offers an integrated toolset with Data Quality Server and Data Quality Client. These are implemented within SQL Server, and that sets it apart from the other tools that we have mentioned. DQS is designed for SQL Server to provide tools to clean and analyze data in SQL Server by using a knowledge base. More information can be found at

Data quality in GCP

If we look for data quality programs in GCP, we will find a solution called Dataplex. The principle is similar to the one we discussed under Azure, with a phase where we explore the data and one in which we clean and prepare the data. With Auto data quality, GCP offers a tool that provides insights about our data. It automates and simplifies quality definition with recommendations and workflows. In the next stage, we will use Dataplex data quality task that enables the management of rules to which data is analyzed, cleaned, and deduplicated.

Next to Dataplex, GCP offers more tools to improve data quality:

Cloud Dataflow: This tool automatically identifies errors, such as missing values or inconsistent data types.

Cloud Dataprep: As the name suggests, it helps in data preparation with cleansing and transformation. It also identifies and removes duplicates.

BigQuery: We discussed this tool before in our chapter about storage. BigQuery is GCP's data warehouse that also automatically analyses data for errors and inconsistencies.

Cloud data loss prevention: This is a GCP service that can help organizations protect their sensitive data. It automatically identifies and classifies sensitive data and provides recommendations. Of course, AWS

and Azure offer tools for Data Loss Prevention as well. In Azure, we can use Azure Information Protection. AWS does not offer a native solution for DLP, but with Macie and GuardDuty for S3, we have a solution for Data Loss Detection There are third-party solutions that we can use in AWS, such as stackArmor.

Data quality in Alibaba Cloud

Alibaba Cloud offers an integrated suite for managing data quality: DataWorks. Like tools we discussed in the other clouds, DataWorks helps in automatically identifying errors, such as missing values in large volumes of data. In DataWorks, we specify the rules by which data is analyzed and processed, including scheduling tasks.

DataWorks allows for integrating data from various sources, including structured and unstructured data, and supports batch and real-time data processing. Next, it provides data quality checks, profiling, and cleansing to ensure the accuracy and consistency of data. It can use machine learning algorithms to get better insights from data.

Of course, this overview of tools in various clouds is not complete. Azure, AWS, GCP, and Alibaba Cloud continuously improve their services to gain more and better insights from data, including the integration of AI and ML.

Securing data

In a multi-cloud environment, securing data can be a complex task. As we discussed in a previous section about governance, data administrators are responsible for managing data access, ensuring data integrity, and monitoring data usage. They must implement a data security plan that holds policies and procedures to safeguard sensitive data from unauthorized access, data breaches, and other security threats. To ensure data security in multi-cloud, data administrators must follow several guardrails when setting up data security policies. Let us have a look at these guardrails.

Securing data, first of all, means that we have to understand what sort of data we must protect and to what extent. Not all data is equally critical. Protecting all data at the highest level would be a very costly endeavor and, above all, very hard to manage. Hence, securing data starts with proper classification. That is the first task of the data administrator: it must classify data based on its sensitivity level and define access controls accordingly. Sensitive data, such as financial and personal data, must be encrypted and given restricted access to prevent unauthorized access.

Encryption must be implemented for all data in transit and at rest to prevent data breaches. Data administrators should use strong encryption algorithms and encryption keys to secure data. All cloud providers encrypt data, but there are differences in how this is done.

Amazon S3, EBS, and Glacier use Server-Side Encryption with the choice to use keys provided by AWS (SSE-S3 or SSE-KMS) or customer-managed keys (SSE-C). Amazon S3 also supports Client-Side Encryption where users can encrypt their data before uploading it to S3. Data in transit is encrypted using SSL/TLS.

Azure and GCP have a similar offerings. Azure supports SSE with customer-managed keys (SSE-CMK) or Microsoft-managed keys (SSE). Azure Disk Storage and Azure File Storage also support the encryption of data at rest using SSE. Azure also provides options for encrypting data in transit using SSL/TLS.

GCP provides multiple encryption options for data at rest and in transit. Google Cloud Storage supports SSE with customer-managed keys (SSE-C) or Google-managed keys (SSE-S3 or SSE-KMS). Google Compute Engine and Google Kubernetes Engine support the encryption of data at rest using Customer-Managed Keys GCP also provides options for encrypting data in transit using SSL/TLS.

Object Storage Service in Alibaba Cloud supports SSE with customer-managed keys or Alibaba Cloud-managed keys As with all other clouds, SSL/TLS is used to encrypt data in transit.

Another important aspect of securing data involves Identity and Access Management IAM policies must be implemented to ensure that only authorized users have access to data. Multi-factor authentication must be used for accessing sensitive data to provide an additional layer of security.

If we have implemented proper rules and guardrails, we also must audit these on a regular base. Regular security audits must be conducted to detect and prevent security breaches. Data administrators play an important role in implementing automated security monitoring tools and performing periodic security audits to identify vulnerabilities and ensure compliance with data security policies.

Lastly, for critical data, we should consider disaster recovery and business continuity planning. This means that we have a disaster recovery plan in place to ensure business continuity in the event of a security breach or a disaster.

We talked about moving infrastructure and data to the cloud, but will that bring us the real benefits of using cloud technology? The answer is likely to be no. We should look at transforming our environments and using cloud-native services to achieve the agility and scalability that businesses are looking for. The next chapter is all about working with cloud-native concepts.

Conclusion

In this chapter, we discussed one of the biggest challenges in multi-cloud: managing and securing data. We learned how to define a data and data migration strategy to set guidelines and guardrails on where data must be stored, who is allowed to access it, and how we make sure that the data is available at the right time and in the right place. We discussed the quality attributes of data and how we can use cloud technologies to improve data quality through automatic cleansing and deduplicating.

In the final sections, we looked intensively at securing data. Securing data in a multi-cloud environment requires a comprehensive data security policy and procedures. Data administrators must classify data based on its sensitivity, implement strong encryption, define access controls, perform regular security audits, and have a disaster recovery plan in place to ensure data security. By following these guardrails, organizations can ensure that their data is safe and secure in a multi-cloud environment.

Points to remember

Cloud provides us with an agile and scalable infrastructure in which we can store, process, and analyze vast amounts of data quickly and efficiently. But we need to define a data strategy to make sure that data is stored in a secure way and that it is accessible and available when required.

Data governance and compliance are critical aspects of any data strategy, and it is important to establish robust policies and controls to ensure data quality, security, and compliance with laws and industry regulations.

Unauthorized access, breaches, and other security threats must be detected on time and preferably avoided. If breaches materialize, we need to have plans for mitigating actions. This includes implementing access controls, encryption, and monitoring tools to detect and respond to security incidents. This is all part of the data security plan.

Questions

Name three data quality attributes.

If we need to migrate large volumes in bulk to AWS, we can use a specific service. What is the name of that service?

What are the tasks of a data steward?

Answers

Accuracy, completeness, consistency, timeliness, relevance, validity.

Snowball.

Data stewards are responsible for ensuring that data is accurate, complete, and available when needed.

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 7

Build and Operate Cloud Native

Introduction

Although lift and shift are still a popular way to get started in the cloud, the real transformation starts with building and operating cloud-native – making use of native services that the major providers offer. Cloud-native development offers solutions to create scalable applications using, for instance, micro-services, container, and serverless concepts, and deploying declarative code. In the following chapters, we will dive deeper into this, but this chapter is the introduction to cloud native.

Structure

In this chapter, we will discuss the following topics:

Understanding cloud-native concepts

Organizing cloud-native with DevOps

Explaining micro-services

Management of releases in cloud-native

Understanding cloud-native concepts

Cloud-native is a software development approach that leverages cloud infrastructure to build and deploy applications that are scalable, reliable, and flexible. Cloud-native applications are designed to take full advantage of cloud-native services, such as containerization, microservices architecture, and serverless computing. We will address these concepts in more detail in this and the following chapters, but here is a short description of these concepts:

Containers are a lightweight way of packaging an application and all of its dependencies into a single unit that can be easily moved between different environments. Chapter Building Agnostic with Containers is all about containers.

To avoid misunderstanding, with serverless, there are still servers involved. The difference is that with serverless, the cloud provider manages and allocates computing resources such as server hardware, networks, and storage automatically, without requiring the user to manage any infrastructure. Users only provide the code for their application or service, which is executed by the cloud provider in a container or runtime environment. Chapter Building and managing serverless is all about serverless.

Microservices are a software architecture pattern in which an application is broken down into smaller, independent services that communicate with

each other over a network. In a multi-cloud environment, these microservices can be deployed and operated across different cloud providers. This chapter will talk mainly about micro-services.

Cloud-native concepts can be used to improve cloud infrastructure in several ways. First, they allow developers to build and deploy applications faster and more efficiently, which can reduce operational costs and increase agility. Second, cloud-native services provide a more reliable and scalable infrastructure that can handle high volumes of traffic and support the rapid growth of applications. Finally, cloud-native services allow for more granular control over resources, which can improve security and reduce waste.

Let us have a look at the various propositions that cloud providers offer in terms of cloud-native concepts (all providers offer concepts for containerization, serverless, and automation):

AWS offers a wide range of cloud-native services. For containerization and container orchestration, this includes Amazon Elastic Container Service and Amazon Elastic Kubernetes Service The serverless computing proposition of AWS is AWS Lambda. Amazon RDS is used for managed databases. A serverless option for container orchestration is Fargate. AWS also offers a variety of tools and services for monitoring and managing cloud infrastructure, such as Amazon CloudWatch and AWS CloudFormation.

Microsoft Azure provides a similar set of cloud-native services, including Azure Kubernetes Service Azure Functions for serverless computing, and Azure Cosmos DB for managed databases. Azure also offers a range of

tools for managing and monitoring cloud infrastructure, such as Azure Monitor.

Kubernetes was originally invented by GCP, so it is no surprise that it offers several cloud-native services. For container orchestration, we use the Google Kubernetes Engine Cloud Functions is the option for serverless computing, and Cloud Spanner is for managed databases. GCP also offers a range of tools for managing and monitoring cloud infrastructure, such as Google Cloud Operations, formerly known as Stackdriver, and Google Cloud Deployment Manager.

Alibaba Cloud provides Elastic Container Instance and Kubernetes Container Service for container orchestration. For serverless computing, we can use Function Compute. ApsaraDB is the solution for managed databases. Tools for managing and monitoring cloud infrastructure are CloudMonitor and Resource Orchestration Service

There is one thing that we should be aware of while using these solutions and offerings: operating cloud native is quite different than the more traditional workload concepts such as virtual machines. We will discuss this next.

Cloud-native applications are typically designed using containerization technologies such as Docker, which allows for a more lightweight and portable deployment process. This is in contrast to traditional VM deployments, which require the entire operating system to be installed and configured on each virtual machine instance. The fact that these concepts are more lightweight makes them very suitable for microservices architecture, allowing for greater flexibility and scalability. In contrast, traditional monolithic applications are designed as a single, large

application, which is more difficult to modify and scale. We will talk extensively about micro-services in the following sections of this chapter.

Next, cloud-native applications can take advantage of serverless computing, which allows for automatic scaling and pay-per-use pricing. This eliminates the need for manual scaling and reduces operational costs. They are commonly used in a DevOps approach for development and deployment, which emphasizes collaboration between developers and operations teams to ensure continuous integration and delivery. This approach allows for faster iteration and deployment cycles and can reduce the time to market for new features.

Organizations that have adopted DevOps will be able to build and deploy applications more quickly and efficiently, with greater reliability and scalability. With that, we are entering the domain of DevOps, which is the central topic of the next section.

Organizing cloud-native with DevOps

Cloud-native applications are designed to be modular, decoupled, and horizontally scalable, allowing them to run on multiple cloud environments. To achieve that, we need a set of principles and practices that utilize the benefits of the cloud infrastructure and services in order to build and deploy these applications. DevOps is a set of practices that emphasizes collaboration and communication between development and operations teams, with the goal of delivering software more rapidly and reliably. Combining cloud-native concepts with DevOps will help us in the development processes whilst reducing deployment times and improving the scalability and reliability of applications that are born in the

Applications that are born in the cloud typically consist of microservices, which are small, independent components that work together to provide the application's functionality. Each micro-service can be developed, deployed, and scaled independently of the others, which allows teams to work more efficiently and with greater agility.

DevOps is one it brings development and IT operations together and enables us to automate many aspects of the software development process, including testing, deployment, and monitoring. Using DevOps practices, we can rapidly iterate on and deploy micro-services.

DevOps capabilities for the development and deployment of applications, amongst others, include:

Containerization: The use of containers to develop and deploy an application.

Cloud-based development: Building the application in the cloud using native cloud technologies and tools.

API Management (APIM): Cloud services will be connected with each other using APIs. This requires API Management with the use of tools and services to create, publish, monitor, and secure APIs, as well as to manage their lifecycle.

Apps and database migration: We can build applications in the cloud, but in a lot of cases, we will also have to migrate existing applications and databases to the cloud, transforming them with cloud-native services.

Full Stack and Grand Stack: Full stack refers to an application that typically uses multiple technologies to build the various layers of the application, such as using a front-end framework such as React or Angular for the user interface, a back-end framework like Node.js or Ruby on Rails for the server-side logic, and a database technology like MySQL or MongoDB for data storage. Grand Stack is a different approach; in this case, an application is built using a combination of four technologies: GraphQL for the API layer, React for the user interface, Apollo for the data layer, and Neo4j for the database layer. The term GRAND stands for GraphQL, React, Apollo, and Neo4j

DevSecOps: Security is intrinsic; it needs to be a bolt in the entire development and deployment process. In DevSecOps, we include security in every step of development and operations, including Static Application Security Testing and Dynamic Application Security Testing as gateways in our development and deployment pipelines.

Document As Code: Documentation is treated as code and is managed using the same version control systems and tools used for software development. This means that documentation is written using markup languages such as Markdown, AsciiDoc, or reStructuredText, and stored as plain text files in a repository such as Git.

PaaS deployment: Using PaaS technologies from cloud providers to build and deploy applications. Think of database services such as Azure SQL and RDS in AWS.

Event-driven architecture: In an event-driven architecture, an application performs an action as a reaction to events triggered by changes in the system or through external inputs. Application components are designed to be loosely coupled and communicate through asynchronous messaging. This means that components can publish events when they perform a task or have new information, and other components can subscribe to those events and take actions based on them.

SaaS integration: Integrating third-party SaaS services into the development and deployment of applications. It includes the connection of different SaaS applications and services through APIs, webhooks, or other integration methods, to enable data sharing, automation, and collaboration across different systems.

On remark has to be made concerning DevSecOps. Of course, DevSecOps goes deeper than implementing SAST and DAST. Incorporating key risk strategies to manage and mitigate security risks effectively, it’s a holistic approach to software development that integrates security practices into the entire software development lifecycle (SDLC). This approach aims to ensure that security measures are considered and implemented throughout the

development process rather than being added as an afterthought. It involves a collaborative effort between development teams, security teams, and operations teams. Teams work together to integrate security practices into the development process seamlessly.

Key risk strategies that we should address in DevSecOps are:

Secure coding Developers are trained in secure coding techniques to prevent common vulnerabilities, such as input validation errors, buffer overflows, and injection attacks.

Continuous security Automated security testing tools are used throughout the development process to identify vulnerabilities and weaknesses in the code. This includes SAST, DAST, and penetration testing.

Secure configuration Secure configuration practices are implemented for all software components, including servers, databases, and network devices. This involves applying secure settings, removing unnecessary features, and regularly updating and patching software.

Access control and privilege Proper access controls are enforced to ensure that only authorized individuals have the necessary permissions to access and modify the software and its underlying infrastructure. Privilege escalation is carefully managed to prevent unauthorized access.

Incident response and A well-defined incident response plan is established to handle security incidents effectively. This includes processes for identifying, containing, eradicating, and recovering from security breaches. Regular drills and exercises help validate the effectiveness of the plan.

To enable these capabilities, we need a supporting platform. For that reason, DevOps is now often rebranded as platform services. Capabilities of the platform include:

Open platform: AWS, Azure, GCP, VMWare, OpenStack, Alibaba Cloud, and Tencent, amongst others, all allow for connecting third-party services to enable DevOps capabilities.

Kubernetes capabilities across clouds: Think of Azure Kubernetes Services Elastic Kubernetes Services in AWS Pivotal Kubernetes Services, or Tanzu of VMWare and OpenShift.

Infrastructure automation: Think of ARM and BICEP in Azure, CloudFormation in AWS, Ansible, and Terraform.

The cultural shift to DevOps: The organization must be ready to adopt the DevOps practices, for instance, by assembling teams with the right skills.

Platform and apps monitoring: We can choose Open-Source monitoring or cloud-based solutions.

Service mesh deployment: The platform must enable service mesh as a layer of infrastructure for managing and controlling communication between micro-services. This service mesh layer takes care of service-to-service communication, allowing developers to focus on application development rather than infrastructure concerns.

Additionally, DevOps enables teams to implement continuous integration and continuous delivery pipelines, which automate the process of testing,

building, and deploying applications. This reduces the time it takes to deliver new features and bug fixes, allowing teams to respond to customer needs more quickly. Before we explore cloud-native further, it's good to have a common understanding of CI/CD.

Continuous Integration focuses on merging code changes from multiple developers into a shared repository regularly. The main goal is to detect and resolve integration issues early and frequently. In a CI workflow, developers frequently commit their code changes to a version control system, triggering an automated build process. The CI server compiles the code, runs unit tests, and performs static code analysis. If any issues are identified, developers are notified immediately, allowing them to fix the problems quickly. CI ensures that code changes are continuously integrated and tested, reducing the risk of integration conflicts and improving code quality.

Continuous Delivery extends the CI process by automating the deployment of applications to various environments, such as development, staging, and production. The CD pipeline allows for the rapid and reliable release of software by automating the steps needed to deliver the application to end users. The CD pipeline includes activities such as packaging the application, configuring the deployment environment, running additional tests (for example, integration tests and acceptance tests), and deploying the application to the target environment. CD ensures that the application is in a deployable state at any given time, making it easier to release new features, bug fixes, and updates to users.

Essential steps in the pipeline comprise:

Code Developers commit their code changes to the version control system.

Build and The CI server automatically builds the code, runs tests (unit tests, integration tests, etc.), and performs code analysis.

Artifact If the code passes the tests, the CI server generates an artifact (for example, a compiled application, a Docker image) that represents a deployable version of the software.

The CD pipeline takes the artifact and deploys it to the desired environment, such as staging or production. This involves configuring the deployment environment, ensuring dependencies are met, and deploying the artifact.

Testing in the deployment Additional tests, such as integration tests or acceptance tests, are performed in the deployment environment to verify the behavior and functionality of the application in a realistic setup.

Approval and Once the application passes all tests and meets the desired quality criteria, it can be approved for release. Release management processes, such as versioning, change management, and user notifications, are typically included in this stage.

By implementing a CI/CD pipeline, development teams can automate and accelerate the software delivery process while maintaining a high level of quality and reducing the risk of errors.

A simple representation of a pipeline is presented in Figure

Figure 7.1: Simple representation of CI/CD pipeline

Let us go over the different steps in more detail:

Version Developers work on their code changes and commit them to a shared version control system, such as Git.

Continuous Integration (CI) includes:

Code Whenever code changes are committed, a CI server (for example, Jenkins, CircleCI) automatically triggers a build process.

The CI server compiles the code, ensuring it can be executed.

Automated Unit tests, integration tests, and other automated tests are executed to verify the code's functionality and quality.

Code Static code analysis tools check the code for coding standards, security vulnerabilities, and other potential issues.

Artifact generation includes:

If the code passes all tests and analysis, the CI server creates an artifact, such as a compiled application or a Docker image.

The artifact is typically tagged with a unique version number or identifier.

Continuous Delivery (CD) includes:

The artifact is deployed to a staging environment, which closely resembles the production environment.

Additional Integration tests, acceptance tests, and other tests are performed in the staging environment to validate the application's behavior and compatibility.

User Acceptance Testing If applicable, stakeholders or users can test the application in the staging environment to provide feedback.

Approval and Once the application passes all tests and receives approval, it can be released to the production environment.

Continuous Deployment includes:

Fully automated In some cases, organizations may choose to automate the release process directly from the staging environment to the production environment without manual approval.

Note that the CI/CD pipeline can vary depending on the specific tools and practices used by different organizations. The representation provided here gives a general idea of the flow and stages involved in a typical CI/CD pipeline. Figure 7.3 provides a more detailed example of a CI/CD pipeline.

How do we start working with cloud-native services and implement the DevOps practice? The DevOps approach is depicted in Figure

Figure 7.2: The DevOps approach

The first step in building cloud-native concepts using DevOps is to adopt a container-based architecture. Containers can be deployed and scaled quickly, and they help to isolate applications from one another. DevOps teams can use tools such as Docker to build, package, and deploy containers, while Kubernetes can be used to manage container orchestration and scaling. Kubernetes provides a unified API for managing different types of workloads, including stateless and stateful applications, batch jobs, and machine-learning workflows.

Using Kubernetes, DevOps teams can deploy applications to the cloud using a variety of different deployment strategies, including:

Rolling updates: With rolling updates, systems get updated in a gradual, controlled manner by updating a small batch of nodes or servers at a time rather than all at once. Once a batch has been updated and verified to be working correctly, the next batch is updated, and so on, until the entire system has been updated. Rolling updates are often used in conjunction with other deployment strategies, such as canary releases or blue/green deployments, to

ensure that the update process is reliable and safe and does not cause any downtime or disruption to users.

Blue/green deployments: In blue/green deployment, we will have two identical environments, one representing the current production environment (blue) and the other representing the new environment with the new feature enabled (green). This is a popular methodology to release new software or features. We will discuss this in more detail in the final section of this chapter.

Canary releases: New features or software releases are tested with a small subset of users, the so-called canary group before the feature or release is rolled out to the entire population.

These are just examples of strategies that allow for rapid deployment of new features and updates while minimizing downtime and reducing the risk of application failures.

Implementing DevOps for cloud-native architecture requires a number of key practices, including:

Automation: We must automate as many aspects of the development and deployment process as possible. This includes testing, building, and deploying applications. Automation reduces the risk of human error and speeds up the deployment process.

Containerization: We use containers to package and deploy applications.

Orchestration: If we build apps using containers, then we need an orchestration tool to manage our containers. Kubernetes has developed to

become the industry standard for automating the deployment, scaling, and management of containers.

CI/CD: For effective and efficient DevOps, we must implement a continuous integration and continuous delivery pipeline to automate the process of testing, building, and deploying applications. This ensures that applications are thoroughly tested and that new features and bug fixes can be quickly deployed. We already looked at the concept of CI/CD and a very simple representation of a pipeline, but a more detailed example is provided in Figure

Code
Create pull request
Policies
Development
Build
Unit Tests
Coverage
SCA
Compliance checks
Gate
Mainline merge
CI pipeline (Product / Platform)
Build (Full)
Unit test
Coverage
Security checks
License compliance
Publish
Deploy
Tests
Gate
Integration pipeline
Platform Integration
Product code
Platform code
Integration build
Publish
Deploy
Tests
Gate
Staging environment

Figure 7.3: Example of CI/CD Pipeline

One topic deserves special attention, and that is observability. We need proper monitoring that provides a view of the entire stack, from the infrastructure to the application and the containers that we use to deliver the application. Once applications have been deployed to the cloud using cloud-native concepts and DevOps practices, it is important to ensure that they are running smoothly and efficiently. This requires a range of monitoring and observability tools to help DevOps teams track performance metrics, detect and diagnose issues, and optimize application performance.

One important tool for monitoring cloud-native applications is Prometheus, which is an open-source monitoring system that collects metrics from targets by scraping metrics from HTTP endpoints on those targets. It stores the collected data in a time-series database and provides a powerful query language for analyzing and visualizing the data. Prometheus can be used to monitor a wide range of metrics, including CPU and memory usage, network traffic, and application-specific metrics such as request latency and error rates.

Another important tool for observability is tracing. Tracing as a methodology allows us to visualize the flow of requests through complex distributed systems and identify performance bottlenecks or other issues. One popular tracing tool for cloud-native applications is Jaeger. Jaeger provides a distributed tracing system with an easy-to-use user interface.

We have been talking about micro-services already, but it is time to do a proper deep dive into this concept. It is the main topic for the next sections

since building and operating micro-services is something completely different from managing traditional applications.

Explaining micro-services

In the previous sections, we concluded that containers and serverless are typically part of a micro-services architecture. In Chapter Creating we introduced micro-services in Figure In this section, we will explain how we can re-architect our applications to micro-services and how to operate them.

When we start working with micro-services, we must consider the following architectural artifacts:

Legacy migration: Unless we are building every application from scratch, we will certainly have to deal with legacy, existing applications, and even infrastructure. In a lot of cases, these will be monolithic. Some of this legacy might be suitable to re-architect as micro-services, but in that case, the business case must be proof that such a migration is providing benefits to an organization.

Application modernization: We talk about application modernization when we are updating, refactoring, or replacing existing applications with newer technologies, architectures, and development methodologies. The goal is to enhance the functionality, performance, and scalability of legacy applications while also reducing their maintenance costs and improving their overall user experience.

API development and integration: Micro-services are loosely coupled and communicate with each other through APIs. They might and likely will also use PaaS and SaaS services from the cloud provider or third-party vendors. API development and integration are, therefore, crucial elements in developing and deploying micro-services.

When you are operating microservices in a multi-cloud environment, it is important to consider several factors, such as security, network connectivity, and data synchronization. One approach to operating microservices is to use a service mesh such as Istio or Linkerd. In Chapter Creating we show the concept of service mesh in Figure

Service mesh provides a dedicated infrastructure layer for managing service-to-service communication, load balancing, and other networking features that are essential to microservices. This can help simplify the complexity of operating microservices in a multi-cloud environment.

Another approach is to use a multi-cloud management platform like Cloudify or Terraform. These platforms enable the deployment and management of micro-services across different cloud providers, as well as automate tasks such as scaling, monitoring, and failover.

At the beginning of this section, we mentioned legacy migration and application modernization. Transforming a monolithic application to a micro-services architecture in the cloud involves breaking down the application into smaller, independent services that can be deployed and operated separately. This is not an easy process, and a lot of aspects must be considered. Let us break it down into six basic, overarching steps:

The first step is to identify the different components of the monolithic application. This can be done by analyzing the application code, database schema, and other relevant documentation.

Once the components have been identified, the next step is to group them into logical services. Each service should have a clear and well-defined purpose that can be easily understood and managed.

Now that the services have been identified, we can proceed to the next step: defining the interfaces between them. These interfaces should be standardized to ensure interoperability and consistency across the different services.

The following step is very likely containerization. This involves packaging each service into a container that can be easily deployed and managed using container orchestration tools like Kubernetes.

We can start deploying containers to our designated cloud. This can be done using cloud-native tools such as AWS Elastic Beanstalk, Azure Kubernetes Service, or Google Kubernetes Engine.

The final step is to manage our new environment with micro-services. This involves monitoring the services for performance and availability, as well as automating tasks such as scaling and failover.

Transforming a monolithic application to a micro-services architecture in the cloud is a complex process that requires careful planning and execution. It is also important to ensure that the new architecture is properly secured and that data is properly managed across the different

services. The final section of this chapter is all about managing micro-services.

Managing releases in micro-services

Micro-services are, in essence, distributed systems. The great benefit of this is that we update specific services without having to touch the entire system. But it also comes with challenges. As soon as we perform an update to service in that system, we must make sure that the system as a whole will still be functioning as designed. This is where release management plays a crucial role: it is likely the most important task in managing micro-services architectures and systems.

Release management involves managing the deployment and release of individual services as part of a larger system. In order to manage releases effectively, it is important to have a process in place that includes feature management and feature toggling. With feature management, we control which features are available to users based on their roles, preferences, or other criteria. This can be done using feature flags, which are toggles that can be used to turn features on or off without the need for a full release.

Using feature toggling, we can turn features on or off based on specific criteria, such as user preferences, geographic location, or other factors. This can be used to gradually roll out new features or to test them in a controlled environment before releasing them to a wider audience. Obviously, we need tools to execute feature management and toggling. Some of the most popular and multi-cloud-ready tools are LaunchDarkly, CloudBees Rollout, Split.io, and the open-source platform Unleash.

All of these tools provide capabilities in feature flag management that enable us to manage features across different environments, including the gradual rollout and testing in controlled environments. Most tools are able to perform A/B testing, in which we compare two versions of a web page, mobile app, or other digital asset to determine which one performs better.

In an A/B test, two different versions of the asset are created and randomly shown to users, with each user seeing only one version. The performance of each version is measured based on a specific goal, such as conversion rate, click-through rate, or user engagement. The version that performs better is then identified and used as the final version.

But how should we implement feature management? First, we must define the feature requirements: this includes identifying the target audience for the feature, as well as any business rules or other criteria that should be used to determine when the feature should be enabled. Then we can design the feature flags that will be used to enable or disable the feature. This may lead to the need to create new code or modify existing code to support the feature flagging functionality.

Now we are ready to implement the feature flags in our micro-services landscape. We start with updating the code to check for the presence of the feature flag and enable or disable the feature accordingly. As already mentioned in the introduction of some tools, testing is of the essence. We have to make sure that the new features are working as expected. The best practice is to run A/B tests or other types of tests to evaluate the impact of the feature on user behavior and system performance.

As with anything in IT, we also must consider monitoring. Monitoring of features includes metrics of usage and other performance indicators to ensure

that the feature is meeting its objectives, as well as adjusting the feature flags as needed to optimize performance and user satisfaction. This is not a one-time exercise but on-going. Environments will be constantly updated with new features; that is what we have CI/CD pipelines for. Development teams will be continuously looking for ways to enhance applications. To control this continuous development, we can use a specific release methodology called blue/green deployment.

Blue/green deployment provides a way to release new features in a controlled and low-risk manner. Blue/green deployment involves deploying two identical environments, one representing the current production environment (blue) and the other representing the new environment with the new feature enabled (green). Figure 7.3 shows the concept of blue/green deployment:

Figure 7.4: Blue/Green deployment

Once the green environment is fully deployed and tested, traffic is routed to the green environment while the blue environment is left idle. This allows for easy rollback in case of any issues with the new feature. Once the new feature is fully tested and stable, the blue environment can be decommissioned, and the green environment becomes the new production environment.

In the context of feature management, blue/green deployment can be used to release new features gradually to a subset of users or to test them in a controlled environment before releasing them to a wider audience. By using blue/green deployment in conjunction with feature flags, teams can gradually roll out new features to a subset of users while minimizing the impact of any issues that may arise.

Now that we have a good understanding of working with cloud-native concepts, we can discuss containerization and serverless in more detail using examples in AWS, Azure, GCP, and Alibaba Cloud. We will start with containers in the next chapter.

Conclusion

In this chapter, we explored the use of cloud-native concepts such as containers and serverless. These concepts offer solutions to create scalable and flexible applications in a micro-services architecture, a software architecture pattern in which an application is broken down into smaller, independent services that communicate with each other over a network. With the use of micro-services, applications become more scalable and resilient since the impact of a single service failure can be reduced so that it does not affect the entire system.

Building and managing micro-services and cloud-native environments can become complex. We extensively explored the use of CI/CD pipelines in a DevOps setting, building and deploying applications with container platforms and serverless options such as Lamda in AWS and Azure Functions. Micro-services using containers and serverless options are decoupled, making the deployment faster and easier to release new services quickly with minimal disruption to the existing application. For this, we must consider methodologies and technologies for feature management and feature toggling. In the last section, we discussed the use of the blue/green deployment methodology to safely test and manage releases.

Points to remember

Cloud-native concepts are containers, container platforms and orchestration, serverless, and micro-services. All major clouds support these concepts. Most clouds have adopted Kubernetes as the industry standard for container orchestration.

Organizing teams in a DevOps setting is the best practice for the development and deployment of applications with cloud-native offerings. For effective and efficient DevOps, we must implement a continuous integration and continuous delivery pipeline to automate the process of testing, building, and deploying applications. This ensures that applications are thoroughly tested and that new features and bug fixes can be quickly deployed.

Transforming legacy applications to micro-services to run them in the cloud can become very complex. One of the key topics to remind in architecting, building, and managing micro-services architectures is APIM, API Management. Since microservices are loosely coupled services, they communicate with each other through APIs, including PaaS and SaaS services from the cloud provider or third-party vendors. API development and integration are, therefore, crucial elements in developing and deploying micro-services.

Questions

What does Grand Stack stand for?

What do we call a release where only a small group of users get the newest version of the software before it is rolled out to the entire population of users?

What is the purpose of a CI/CD pipeline?

Answers

GraphQL, React, Apollo, Neo4j Database.

Canary release.

Build, test, and deploy new software, streamlining the release of software.

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

Building Agnostic with Containers

Introduction

Container technology has become increasingly popular in software development and deployment on cloud infrastructure. It allows for very agile development and controlled distribution in various clouds, promising a form of agnostic cloud deployment. In this chapter, we will look at the concepts of containers, developing and provisioning, and management of container environments in multi-cloud settings.

Structure

In this chapter, we will discuss the following topics:

Understanding container technology

Developing and provisioning containers using industry standards

Exploring container management using a single pane of glass view

Deep dive into container monitoring and log management

Understanding container technology

Container technology has rapidly gained popularity over the past decade to package and deploy applications in a portable and efficient manner. But what are containers? Containers provide an isolated runtime environment for applications, allowing them to run consistently across different computing environments. This consistency is a key benefit of container technology, making it an essential tool for modern software development and deployment.

At its core, a container is a lightweight and portable package that includes all the necessary components to run a specific application. One remark must be made here: Containers are designed to be lightweight, meaning they consume fewer resources than traditional virtual machines. However, it's still important to allocate resources carefully to ensure that containers have enough CPU, memory, and disk space to run efficiently. Failure to allocate resources correctly can lead to performance issues or even container crashes.

The portable package includes the application code, dependencies, libraries, and configurations. Containers also come with an isolated runtime environment that allows them to run consistently across different computing environments, such as different operating systems, cloud providers, or local machines. Figure 8.1 shows the basic architecture of a container and all the components that we need to manage containers:

Figure 8.1: Basic architecture of container management

As we can see from the figure, architecting container technology requires a different approach than traditional application deployment. Containers have unique properties that require careful consideration during the application design and deployment process. The following are some key considerations for architecting for container technology:

Micro-services Containers are an essential tool for implementing micro-services architecture. Micro-services are a way to break down large monolithic applications into smaller, independent services that can be developed, deployed, and scaled independently. Containers provide a lightweight and portable runtime environment for each microservice,

allowing them to run independently of each other. In Chapter Build and Operate Cloud we extensively discussed micro-services architecture.

Resource Containers are designed to be lightweight, meaning they consume fewer resources than traditional virtual machines. However, it is still important to allocate resources carefully to ensure that containers have enough CPU, memory, and disk space to run efficiently. Failure to allocate resources correctly can lead to performance issues or even container crashes.

Containers provide an isolated runtime environment for applications, but this isolation is not enough to secure container ecosystems themselves. It is essential to take measures to secure containers, such as using secure container images, applying security patches, and limiting container privileges. Failure to secure containers can lead to security breaches or data loss.

Let us have a look at the best practices to deploy and run containers. First, we need an orchestration platform, such as Kubernetes or Docker Swarm. These platforms provide a way to manage and scale containers across different computing environments. Indeed, Kubernetes and Docker Swarm are cloud platform agnostic, meaning they can operate on different computing platforms such as VMWare, Azure, AWS, GCP, and Alibaba Cloud.

Orchestration platforms automate many of the tasks required to manage containers, such as scheduling, scaling, and load balancing. Using a container orchestration platform simplifies the management of containers and ensures that they run consistently across different environments.

Next, it is recommended to use standardized container images to ensure that containers are consistent. This will also help in managing containers. Standardized images are created from a base image that includes a minimal operating system and the necessary libraries and dependencies. This base

image can then be customized with the application code and configurations. Using standardized images ensures that containers are consistent across different environments, making it easier to manage and scale them. An example of a coded base image for a container is provided in the section under Guided Plan to develop and deploy containers.

Containers are designed to be lightweight, but they can still consume resources if not optimized correctly. It is important to optimize container size by removing unnecessary files and dependencies. This optimization reduces container size and improves performance, making it easier to manage and scale containers.

Pitfalls and risks of container technology

So far, we looked at the benefits of using containers, but there are pitfalls too. It might seem contradictory, but vendor lock-in is of the pitfalls. Container technology is not immune to vendor lock-in. Many container orchestration systems and cloud providers offer their own container solutions, which can make it difficult to switch to a different solution. It is important to consider the long-term implications of choosing a specific container solution and plan accordingly.

We already mentioned security, and there are certainly security risks to containers. Containers are vulnerable to security risks, such as unauthorized access, malware, and data loss. It is important to take measures to secure containers, such as using secure container images, applying security patches, and limiting container privileges.

Complexity and cost are also key here. Especially when containers are used with microservices, containers can become really hard to manage. For some companies, it's the reason to leave the microservices architecture and return to a more traditional architecture for applications. An example of that and cause for a real debate is the way AWS rearchitected the service for video monitoring at Prime. The service monitors thousands of video streams that Prime delivers to customers. Originally this task was done by a set of distributed components orchestrated by AWS Step Functions, a serverless orchestration service. But the responsible team found out that a monolithic approach to this task did a better job.

Container services from major cloud providers

Major cloud providers such as Azure, AWS, GCP, and Alibaba Cloud offer containerization services to their customers, allowing them to easily deploy and manage containerized applications. In this response, we will describe the best setup for a containerized application using the following examples from these cloud providers:

Azure offers several containerization services, including Azure Kubernetes Service Azure Container Instances and Azure Container Registry The best setup for a containerized application on Azure depends on the specific use case and requirements.

For example, if the application requires a high level of scalability and reliability, AKS is a good choice. AKS provides a managed Kubernetes environment that allows customers to deploy and manage containerized applications at scale. AKS also integrates with other Azure services, such as Azure Active Directory and Azure Monitor, to provide a complete containerization solution.

If the application requires a more lightweight containerization solution, ACI is a good choice. ACI provides a serverless environment for running containers, allowing customers to easily deploy and scale containerized applications without worrying about infrastructure management. ACI also integrates with other Azure services, such as Azure Virtual Networks and Azure Blob Storage, to provide a complete containerization solution.

AWS offers Amazon Elastic Container Service Amazon Elastic Kubernetes Service and Amazon Elastic Container Registry

For an application that requires a high level of scalability and reliability, EKS is a good choice. EKS is a managed Kubernetes environment that integrates with other AWS services, such as AWS Identity and Access Management and AWS CloudTrail.

ECS is a more lightweight containerization solution. ECS also integrates with other AWS services, such as Amazon Virtual Private Cloud and Amazon S3.

The container solutions of GCP include Google Kubernetes Engine Google Cloud Run, and Google Container Registry. If the application requires a high level of scalability and reliability, GKE is the right solution. GKE integrates with other GCP services, such as Google Cloud IAM and Google Cloud Logging, to provide a complete solution.

For a lightweight solution, Cloud Run is a good choice. Cloud Run is a serverless environment for running containers and integrates with other GCP services, such as Google Cloud SQL and Google Cloud Storage, making deployment and managing containers easy.

Finally, we look at Alibaba Cloud. It offers Container Service for Kubernetes Elastic Container Instance and Container Registry. ACK would be the best fit for applications that require a high level of scalability and reliability, where ECI is the lightweight variant. As with the other clouds, both services integrate with all other services of Alibaba Cloud.

Developing and provisioning containers using industry standards

We made a promise in the previous section: containers can run anywhere and on any platform. They are cloud platform agnostic, although we also mentioned that there's still a risk of vendor lock-in when we use a specific deployment platform for our containers. To ensure the interoperability and portability of containerized applications across different cloud platforms, the container technology industry has developed several standards. These standards cover various aspects of container technology, including container formats, container orchestration, and container registry. We will discuss these in this section.

We start with the format that containers have. A good example of such a format is the Open Container Initiative which is an open-source project that defines standards for container formats and runtime. OCI specifications ensure that container images are portable across different container runtimes and platforms. The OCI format specifies how to package and distribute container images and how to run them on different platforms. OCI sets definitions for:

Image This concerns the format and layout of a container image. A container image is a lightweight, standalone, and executable package that contains everything needed to run an application, including the application code, runtime, libraries, and dependencies. The OCI Image Specification provides a standard way of packaging and distributing container images, which can be used across different container runtimes.

Runtime This defines the format and behavior of a container runtime. A container runtime is responsible for starting and managing a container instance, including creating and isolating namespaces, managing cgroups, and interacting with the host system. The OCI Runtime Specification provides a standard way of interacting with a container runtime, which allows containers to be portable across different container engines and platforms.

An example of a Dockerfile that contains the specifications according to OCI is as follows:

```
# Use a base image that supports the OCI image format

FROM scratch

# Set the image label according to OCI specification

LABEL org.opencontainers.image.title="My App" \

    org.opencontainers.image.description="This is my app" \

    org.opencontainers.image.authors="Jeroen Mulder " \

    org.opencontainers.image.created="2023-03-31T14:00:00Z" \

    org.opencontainers.image.version="1.0"
```

```
# Add the application binary to the container image

ADD myapp /usr/bin/myapp

# Set the default command for the container

CMD ["/usr/bin/myapp"]
```

In the example, we start with a scratch base image, which is an empty image that provides no file system or runtime environment. Next, we set various image labels using the LABEL instruction, following the OCI image specification. Then we add our application binary to the container image using the ADD instruction. Finally, we specify the default command to run when the container is started using the CMD instruction.

We also need container orchestration. Kubernetes is the most popular container orchestration platform, which has become an industry standard for containerized application deployment and management. Kubernetes provides a rich set of features for scaling, scheduling, and managing containerized applications across different cloud platforms.

Lastly, we need a mechanism to register our containers, where we can centrally store containers and distribute container images. Docker Hub is the most widely used container registry, but there are other popular alternatives, such as Google Container Registry Amazon ECR, and ACR. These registries allow seamless integration with container orchestration platforms like Kubernetes.

In all of this, security plays a big role too. The Center for Internet Security provides a benchmark for secure container deployment, which is widely adopted in the industry. The benchmark provides best practices and guidelines for securing container images, orchestrators, and hosts. Think of:

Use trusted base Always use images from trusted sources and verify the integrity of the image by verifying its checksum or digital signature.

Use the latest version of container Keep your container images up to date to ensure that security patches are applied and vulnerabilities are addressed.

Limit container Containers should run with the least number of privileges necessary to perform their intended function. For example, avoid running containers as root.

Use image scanning Use tools such as vulnerability scanners or static code analysis tools to scan container images for known vulnerabilities.

Apply security Regularly update container images with the latest security patches to address known vulnerabilities.

Implement network Use network segmentation to isolate container traffic and prevent attackers from moving laterally within your infrastructure.

Use secure configurations: Follow secure configuration practices, such as disabling unnecessary services, restricting access to sensitive resources,

and enforcing secure communication protocols.

Monitor container Monitor container activity for suspicious or malicious behavior, such as attempts to access sensitive resources or unusual network traffic.

Implement access Use access controls to limit the permissions of containerized applications and ensure that users and applications only have access to the resources they need to perform their tasks.

All these industry standards will help to ensure that containerized applications are portable and can be easily deployed across different cloud platforms, making container technology an attractive option for modern application development and deployment. But how will we develop and deploy a containerized application to the cloud? In the next section, we will go through the essential steps.

Guided plan to develop and deploy containers

If we want to start working with containers, it's recommended to follow these this ten steps approach:

Step Choose a cloud provider. Select a cloud provider that supports container services. Popular options include Amazon Web Services Microsoft Azure, and Google Cloud Platform Evaluate their offerings, pricing, and features to find the best fit for your requirements.

Step Set up an account. Create an account with your chosen cloud provider. This typically involves providing your contact and billing information.

Step Familiarize yourself with container concepts. Get acquainted with container technology and its key concepts, such as container images, containers, and container orchestration. Understand the benefits of containers, such as portability, scalability, and isolation.

Step Select a container orchestration platform. Choose a container orchestration platform that will manage your containers. The most popular options are Kubernetes and Docker Swarm. Kubernetes has become the de facto standard and offers a rich ecosystem and community support.

Step Set up a container registry. A container registry is a repository for storing and managing container images. Most cloud providers offer their

own container registries, such as AWS ECR, ACR, and GCP Container Registry. Create a registry to store your container images.

Step Build container images. Create Dockerfiles or build configuration files that define the steps to build your container images. These files specify the base image, dependencies, and commands required to run your application within a container.

Step Containerize your application. Use tools like Docker to build container images from your application code. This involves packaging your application, its dependencies, and any required configuration into a container image.

Step Push container images to the registry. Push the container images you've built to the container registry you set up earlier. This step makes the images available for deployment and sharing with your team.

Step Create a cluster. Set up a cluster within your chosen container orchestration platform. This involves provisioning virtual machines or worker nodes that will run your containers. The orchestration platform will manage the scheduling, scaling, and monitoring of containers across the cluster.

Step Deploy containers. Use the orchestration platform's CLI or GUI to deploy your containers to the cluster. Define the desired number of instances, resource requirements, networking, and any other necessary configurations.

Obviously, from this point onwards, we should also monitor the health and performance of our containers using the tools provided by the cloud provider or third-party monitoring solutions. And we must set up automatic scaling rules to scale your container instances based on resource utilization or incoming traffic.

Still, the very first step is to choose a container platform that best suits our needs. There are many options available, including Docker, Kubernetes, and Amazon ECS, as we have discussed in the first section of this chapter. Evaluate each platform's features and pricing and choose the one that best fits your requirements. That can be quite straightforward: if our environment runs completely in Azure and there are no plans to deploy in other clouds, then Azure seems to be a logical place to start hosting our containers. It really depends on the cloud strategy of an organization.

Next, we can develop our application using your preferred programming language and framework. We must make sure that we design the application with containers in mind, ensuring that it is modular and can run in a containerized environment. This is really the job of an application or software architect.

Once the application has been designed and coded, we can start developing the container images. We can use Docker, for instance, with a Docker image. A Docker image is a self-contained package that includes your application and all its dependencies. We use a Dockerfile to define the image and its contents. An example of a Dockerfile is given.

We start with a base image that includes the Python 3.9 runtime and dependencies for our application. We then set the working directory to /app and copy the contents of the current directory into the container. We

install any required dependencies using pip and set the default command to run our app.py script using the Python interpreter:

```
# Use a base image that includes the runtime and dependencies for your application

FROM python:3.9-alpine

# Set the working directory to /app

WORKDIR /app

# Copy the current directory contents into the container at /app

COPY . /app

# Install any required dependencies

RUN pip install --no-cache-dir -r requirements.txt

# Set the default command to run when the container starts

CMD [ "python", "./app.py" ]
```

We can build this image using the docker build command:

```
docker build -t hello-world .
```

Once the build is completed, we can run the container:

```
docker run -p 5000:5000 hello-world
```

This will start the container and map port 5000 in the container to port 5000 on the host. We can then access the Hello, world! application by navigating to http://localhost:5000 in a web browser.

Of course, this is a very simple example and probably does not require a lot of testing, but testing the image is a necessity. After creating the Docker image, test it to ensure that it runs as expected. Use a local Docker environment to test the image and make any necessary changes. When we make changes to our application, we must also update the Docker image and redeploy it to the container platform. Rolling updates is a methodology to minimize downtime and ensure that your application is always available. Having a central place where images are stored and distributed – the container registry – is essential to stay in control of image consistency.

We can deploy our application to the designated cloud provider using our chosen container platform. Follow the documentation provided by your cloud provider to set up a cluster or a single instance for your application. The reason to set up a cluster is because we want our containers to be able to scale in case our application is used more. As our application grows, we may need to scale it horizontally by adding more instances, adding more instances automatically or manually.

Now we have got everything set up, and we must start to monitor the behavior of our containers and make sure that everything keeps running as designed and secure. In the final section of this chapter, we will discuss monitoring and logging in much more detail.

Developing and deploying a containerized application to the cloud involves choosing a container platform, developing your application, creating a Docker image, testing the image, choosing a cloud provider, setting up a container registry, deploying the application, and next scaling, monitoring, and updating the application. Once we have developed and deployed containers, we must manage our container platform and the applications that are hosted on the platform. That is the topic of the next section.

Exploring container management using a single pane of glass view

We discussed the deployment of containers, but how do we manage these container platforms and the containers that hold the applications, especially in multi-cloud settings? We need to make sure that containers are secure, scalable, and highly available. This section will discuss how container management is done and the tools available in Azure, AWS, GCP, and Alibaba Cloud. We will also discuss best practices, pitfalls, and security management in container environments.

Managing containers involves various tasks such as creating, deploying, scaling, and monitoring them. Container orchestration tools help to automate these tasks, making it easy to manage containers at scale. Some of the popular container orchestration tools include Kubernetes, Docker Swarm, and Apache Mesos:

Kubernetes is the most popular container orchestration tool, and it is available in all major cloud providers. Kubernetes automates container deployment, scaling, and management. It provides a platform to deploy and manage containers in a distributed environment. Kubernetes is highly customizable and provides various deployment options, such as rolling updates, blue-green deployments, and canary deployments.

Docker Swarm is a native clustering and scheduling tool for Docker containers. It is integrated with the Docker engine, making it easy to use

for Docker users. Docker Swarm provides automatic load balancing and scaling, making it easy to manage containers at scale.

Apache Mesos is another popular container orchestration tool. It provides a platform to manage containers across multiple clusters. Apache Mesos provides resource isolation and sharing, making it easy to run different workloads on the same cluster.

To manage containers across multiple clouds, we can use a multi-cloud container management platform. These platforms provide a unified view of container management across different clouds. A multi-cloud container management platform will allow us to manage containers from a single interface, regardless of the cloud provider. An example is Kubernetes Federation. Kubernetes Federation enables the management of multiple Kubernetes clusters across different cloud providers. Federation provides a single pane of glass view for container management across different clouds.

Another solution is to use a container management platform such as Rancher. Rancher is a container management platform that provides a unified view of container management across multiple clouds. Rancher allows us to manage Kubernetes clusters across different clouds, including Azure, AWS, GCP, and Alibaba Cloud. Moreover, with Rancher, we can manage containers from a single interface. Next, it offers features such as automated scaling, load balancing, and monitoring, making it easy to manage containers at scale.

One final tool that we must discuss is Tanzu Mission Control by VMWare. This is a centralized management platform offered by VMware that enables organizations to manage and deploy Kubernetes clusters across

multiple cloud environments, just like Rancher does. It provides a single control plane for managing Kubernetes clusters running on different cloud providers, including AWS, Microsoft Azure, and GCP. Tanzu Mission Control also enables organizations to manage the entire container lifecycle, including building, deploying, and scaling containerized applications.

In terms of best practices, we already mentioned the use of container orchestration and registry for images in the first section. But there is more that we can do to ensure that container environments are managed in a proper way. Best practices include:

Use resource When running containers, it is important to set resource limits such as CPU and memory usage to prevent containers from consuming too many resources and affecting other applications running on the same host.

Monitor container Monitoring container performance is critical to ensuring that they are running efficiently and reliably. You should monitor key metrics such as CPU usage, memory usage, and network performance to identify any performance issues and address them proactively.

Use configuration management Configuration management tools such as Ansible and Puppet make it easier to manage and deploy containerized applications. These tools automate the process of configuring and deploying containers, making it easier to manage containers at scale.

Use version Version control is critical to managing container images and configurations. You should use version control tools such as Git to track

changes to container images and configurations and roll back to previous versions if necessary.

Managing security in container platforms

There is one topic that we must discuss in more detail, and that is managing the security of container environments. Security starts with using a minimal and trusted base image for our containers, such as those provided by the cloud provider or a reputable source. This reduces the attack surface and ensures that the container does not include any unnecessary software or libraries that can introduce vulnerabilities. This also implies that we must keep our containers up to date, making sure that we regularly apply updates to our containers with the latest patches and security fixes to mitigate known vulnerabilities. This includes not only the application code but also the underlying operating system and any dependencies.

Deploying a base image for a container is one, but we also need to configure the containers. This means that we have to ensure that containers are secure by default. For example, avoid running containers as root, disable unnecessary services and ports, and enforce strong authentication and access control. We must implement secure networking practices such as encrypting communication between containers, using firewalls to restrict access to sensitive ports, and using network policies to control traffic flow.

An important task is to monitor the container platform and the containers running on that platform to discover any suspicious activity, such as unauthorized access or attempts to exploit vulnerabilities. We can use tools such as logging and auditing to track container activity and investigate any security incidents. In the final section, we will discuss this in more detail since this is of great importance.

Other best practices to secure containers are:

Use container isolation techniques such as namespaces and cgroups to limit the resources and permissions available to each container. This can help prevent containers from interfering with each other or accessing sensitive data or resources.

Use container image scanning tools to detect any vulnerabilities or security risks in container images before they are deployed. This can help prevent the deployment of vulnerable containers that can be exploited by attackers.

Use access control and authentication mechanisms such as role-based access control and multi-factor authentication to control access to container environments and containers. This can help prevent unauthorized access and reduce the risk of data breaches.

Overall, managing security in container environments and containers in the cloud requires a comprehensive and proactive approach that includes implementing best practices for configuration, monitoring, and access control. This all fits into the concept of DevSecOps, as represented in Figure

Figure 8.2: The DevSecOps approach

DevSecOps is an approach to software development that integrates security into the entire software development lifecycle, from design to deployment and beyond. It emphasizes collaboration between development, operations, and security teams to ensure that security is a shared responsibility and is addressed at every stage of the development process.

Techniques that should be included in DevSecOps are Software Component Analysis Static Application Security Testing Interactive Application Security Testing (IAST), Dynamic Application Security Testing and penetration testing:

SCA is a process that involves analyzing the software components used in an application to identify any potential security vulnerabilities or compliance issues. Obviously, this includes container development. SCA tools scan the

software codebase and generate a list of components, including open-source libraries, third-party components, and custom code, along with their versions and dependencies. The SCA tool compares this list against a database of known vulnerabilities and checks for any licensing or compliance issues. The results of the analysis are typically presented in a report that includes a summary of the findings, along with recommendations for remediation.

SAST is a process that analyses the source code of an application to identify security vulnerabilities and coding errors before the code is compiled or deployed. SAST tools use a set of rules and heuristics to scan the codebase and identify potential security issues such as SQL injection, cross-site scripting and buffer overflows. SAST is relevant when using containers, as it helps identify security vulnerabilities and misconfigurations that could be exploited by attackers, such as image tampering, resource abuse, and container escape attacks. SAST can be integrated into the container image build process to ensure that any potential security vulnerabilities or coding errors are identified before the image is deployed.

DAST is the process of testing an application from the outside to identify vulnerabilities that may be exploitable by an attacker. DAST tools simulate attacks against the running application to identify potential security weaknesses such as input validation issues, authentication and authorization problems, and misconfigured security settings. It can help identify security vulnerabilities in the containerized application at runtime. Since containers are often used to deploy micro-services-based architectures, it can be challenging to ensure that all the micro-services are secure and that there are no vulnerabilities in the communication between them. DAST tools can help identify these vulnerabilities by testing the application from the outside.

IAST focuses on identifying vulnerabilities in applications during runtime. IAST involves instrumenting the application code or runtime environment to

gather security-related data while the application is executing within a container. This data includes information about code execution, data flow, inputs, and outputs.

Pen testing can be performed in container environments in the cloud to identify potential security weaknesses in the containerized applications and their underlying infrastructure. In these tests, we simulate attacks on the containerized applications and infrastructure to identify any potential security weaknesses using tools such as Metasploit or Nmap.

One topic that also must be discussed here is the Software Bill of Materials Containers make it easy to deploy and scale applications across different environments. However, this also introduces new security challenges, particularly around supply chain security. An SBOM is a detailed inventory of all the components used in a software application, including its dependencies and their versions. It helps organizations understand the security risks associated with the software they are using, as well as identify vulnerabilities and potential points of attack in the software supply chain.

For container users, having an SBOM can provide visibility into the components used in the container image, including the operating system, libraries, and other software packages. This information can be used to identify and remediate any vulnerabilities or compliance issues associated with these components.

In terms of who benefits from an SBOM when using containers, it would be the organization deploying the containers, as well as any downstream consumers of the container image. This includes developers, operations teams, security teams, and compliance teams that need to ensure the security and compliance of the software they are using. Additionally, SBOMs are becoming increasingly important in regulatory compliance, particularly in

industries such as healthcare and finance. As such, collecting and maintaining the SBOM has become a mandatory artifact in DevSecOps.

Managing security in container environments and containers in the cloud fits in well with the DevSecOps approach because it emphasizes a proactive, collaborative, and automated approach to security. By incorporating security best practices into the container development process, teams can ensure that security is not an afterthought but is instead built into the entire process. This includes implementing security testing and scanning as part of the CI/CD pipeline, using automation to enforce security policies, and collaborating between teams to identify and address security risks.

In the DevSecOps approach, security is not a separate function that is added at the end of the development process but is instead integrated into every aspect of the process. This ensures that security risks are identified and addressed early in the development cycle, reducing the risk of vulnerabilities and security incidents. By managing security in container environments and containers in the cloud as part of the DevSecOps approach, teams can build more secure and resilient applications that are better able to withstand cyberattacks and protect user data.

Deep dive into container monitoring and log management

With so many moving parts, it is important to have a robust monitoring strategy in place to ensure that everything is running smoothly. This section is all about monitoring and log management. We will start with the best practices for implementing the monitoring of containers in our cloud environments.

When we discuss monitoring, we have to start debating what we will be monitoring in the first place. Sometimes a simple 'heartbeat' is a certain workload, alive or not, can be sufficient, but in most cases, we want to know much more. For instance, how the workload is performing. That is a matter of choosing the right metrics. Metrics such as CPU usage, memory usage, and network traffic are all important, but we should also consider other metrics like container restarts, container uptime, and resource limits. These metrics can help you identify potential issues before they become serious problems.

Setting metrics for container monitoring involves identifying the Key Performance Indicators that are relevant to the application and its infrastructure. The metrics that we choose will depend on specific use cases and what we want to achieve with container monitoring. Steps to consider in defining metrics are:

Identifying application This includes identifying the resources that the application needs to run, such as CPU, memory, disk space, and network bandwidth.

Defining key performance KPIs should be relevant to the overall performance and health of the application. Examples of KPIs include CPU usage, memory usage, network traffic, and disk I/O.

The following Python script uses the Docker SDK for Python to get the container stats and then defines functions to extract the CPU usage, memory usage, and network I/O metrics from the stats. The script then enters an infinite loop to continuously print out these metrics every second for the specified container ID. You can customize this script to monitor other metrics as needed; it is just an example:

```
import docker

import time

client = docker.from_env()

def get_container_stats(container_id):

    container = client.containers.get(container_id)

    stats = container.stats(stream=False)

    return stats

def get_cpu_usage(stats):
```

```
    cpu_stats = stats['cpu_stats']

    precpu_stats = stats['precpu_stats']

    cpu_delta = cpu_stats['cpu_usage']['total_usage'] -
precpu_stats['cpu_usage']['total_usage']



    system_delta = cpu_stats['system_cpu_usage'] -
precpu_stats['system_cpu_usage']

    cpu_percent = round((cpu_delta / system_delta) * 100.0, 2)

    return cpu_percent

def get_memory_usage(stats):

    memory_stats = stats['memory_stats']

    memory_usage = memory_stats['usage'] / (1024 ** 2)

    memory_limit = memory_stats['limit'] / (1024 ** 2)

    memory_percent = round((memory_usage / memory_limit) * 100.0, 2)

    return memory_percent
```

```
def get_network_io(stats):

    network_stats = stats['networks']

    rx_bytes = network_stats['eth0']['rx_bytes']

    tx_bytes = network_stats['eth0']['tx_bytes']

    return (rx_bytes, tx_bytes)

# Example usage

container_id = 'your-container-id-here'

while True:



    stats = get_container_stats(container_id)

    cpu_usage = get_cpu_usage(stats)

    memory_usage = get_memory_usage(stats)

    rx_bytes, tx_bytes = get_network_io(stats)

    print(f'CPU Usage: {cpu_usage}%')

    print(f'Memory Usage: {memory_usage}%')
```

```
    print(f'Network In: {rx_bytes} bytes')

    print(f'Network Out: {tx_bytes} bytes')

    time.sleep(1)
```

Since containers literally can come and go in cloud environments and are continuously updated, it is essential to have real-time monitoring in place. We need to know what is always happening in containers to identify issues and troubleshoot problems. This means that we must choose tools that provide real-time monitoring and alerting to ensure that we are always aware of what is happening in our containers. Examples of such tools are Prometheus, Grafana, New Relic, Datadog, and Sysdig.

If we are running operations in multi-cloud, we as an administrator do not want to get into various dashboards to monitor containers. The use of a centralized dashboard is therefore highly recommended to keep track of all our containers in one place. It allows us to see the status of all your containers briefly and quickly identify any issues that need attention. Use a dashboard that integrates with the monitoring tools to make it easy to monitor and manage containers.

We talked about metrics, but then we also need alerts if metrics are not met. Hence, we need to set up alerts to notify us when there is an issue that needs attention, allowing us to quickly respond and resolve the problem. We must set up alerts for important metrics such as CPU usage and memory usage, as well as for critical events like container crashes.

After we have determined the relevant KPIs, we need to define thresholds and alerting rules for each metric. This means setting limits for acceptable levels of each metric and configuring your monitoring system to alert you when these limits are exceeded. For example, we may want to set a threshold for CPU usage at 80% and receive an alert when CPU usage exceeds this threshold.

Now that we have set metrics, thresholds, and alerting rules, we must be able to continuously monitor the application and adjust the metrics as needed. This means regularly reviewing the defined metrics and thresholds to ensure that they are still relevant and effective. Failing the regular review is probably the most common mistake in monitoring containers, but there are more pitfalls.

We can, for instance, define too many metrics. In that case, we are over-monitoring. Monitoring too many metrics can lead to information overload and make it difficult to identify the most important issues. We must choose the metrics that are most important for the application and focus on those.

A lack of integration between monitoring tools and container orchestration tools can make it difficult to get a complete view of your containers. Make sure that monitoring tools integrate with the chosen container orchestration tools to ensure a complete view of the containers and the applications running in these containers.

The last major pitfall may seem obvious, but it happens all the time: alerts that are not followed up. Alerts are useless if we do not act on them. Make sure that you have a plan in place for responding to alerts and resolving

issues quickly. Assign responsibilities to team members and make sure that everyone knows what to do when an alert is triggered.

Collecting and analyzing logs

Logs are a crucial source of information when it comes to monitoring containerized applications. They provide insights into the behavior of your applications, thus helping you identify potential issues and troubleshoot problems when they occur. By collecting and analyzing logs, you can gain a better understanding of your application's performance and usage patterns, as well as detect security threats and other issues.

The first task at hand is to set a logging strategy before we start collecting logs. This includes identifying which logs to collect, how frequently to collect them, and where to store them. We will also need a place to store our logs. A centralized logging solution is preferred and a best practice in container monitoring. This enables us to collect and store logs from all your containers in a single location, making it easier to analyze and troubleshoot issues.

Logfiles only make sense if they are well-structured. Structured logging involves adding additional metadata to logs, making it easier to analyze and search for specific information. By using structured logging, wean improves the accuracy of your monitoring and troubleshooting efforts. This also means that we might need to remove information that is not required to keep in logs. With log rotation, we can automatically delete or archive old logs to free up storage space. This is essential for managing the large volumes of logs generated by containerized applications.

All of this must be done in real-time to enable fast detection of and response to issues, reducing downtime and minimizing the impact on your users. Some popular tools to facilitate this are as follows:

Elasticsearch and Kibana are open-source tools for log collection and analysis. Elasticsearch provides a powerful search engine for storing and indexing logs, while Kibana offers a graphical interface for visualizing and analyzing log data.

Fluentd is an open-source log collector that supports a wide range of log formats and provides flexible routing and filtering options. It is highly customizable, making it an ideal tool for managing complex containerized applications.

Logstash is another open-source log collection and processing tool that is highly customizable. It supports a wide range of input and output plugins, making it easy to integrate with other tools and platforms.

What do these tools collect in terms of logs and events? There might be differences, but in general, these tools collect logs of the following events:

Real-time logs of container events, including container creation, start, stop, and deletion. These logs can be used to monitor the container lifecycle and troubleshoot issues.

Logs are generated by applications running in containers. This can include log files generated by web servers, databases, and other application components.

Network traffic logs, including logs of inbound and outbound traffic, as well as traffic between containers and other network resources.

Security logs about security-related events in containers, such as failed login attempts, suspicious network traffic, and other security-related events.

Performance logs using container performance metrics, such as CPU usage, memory usage, and disk I/O. This can help identify performance bottlenecks and optimize container performance.

These are just a few examples of the types of logs that tools can provide in containers. The specific logs available will depend on the configuration of the container environment and the logging options enabled. Containers generate a vast number of logs, including application logs, system logs, and container runtime logs. Correlating these logs allows you to gain a holistic view of your containerized application's behavior. Correlating logs is, therefore, essential for troubleshooting and debugging.

Now, the next question is how do we make sure that logs are collected from our containers and shipped to a store that keeps the logs? In the following example, we use a script that tells Fluentd to collect logs from a Docker container and send them to a centralized logging environment.

First, make sure you have Fluentd installed on the host where the Docker container is running. The installation instructions can be found at:

Next, we create a Fluentd configuration file, for with the following content:

```
  @type forward

  port 24224


docker.**>

  @type copy


    @type elasticsearch

    hosts :

    logstash_format true

    flush_interval 10s
```

This configuration file sets up a Fluentd forward input plugin to receive logs from the Docker container on port It then sends the logs to Elasticsearch using the Elasticsearch output plugin.

Replace and with the hostname and port of the designated Elasticsearch instance.

Start Fluentd using the configuration file in a sudo command:

```
sudo fluentd -c /etc/fluentd/conf.d/docker.conf
```

Start your Docker container with the following additional flags:

```
docker run -d --log-driver=fluentd --log-opt fluentd-address=:24224 --log-opt tag=docker.
```

This command tells Docker to use the Fluentd logging driver and specifies the address and tag for Fluentd to use. Replace with the hostname or IP address of the host where Fluentd is running, with a name for the container, and with the name of the Docker image you want to run.

With these steps, the Docker container logs should now be sent to Elasticsearch via Fluentd. We can use Kibana or another Elasticsearch client to visualize and analyze the logs.

As with monitoring, there are pitfalls we must avoid in collecting logs. One of the biggest pitfalls of log collection and analysis is the overwhelming amount of data that can be generated. This can lead to

difficulties in managing and analyzing logs, as well as increased storage costs.

Another common pitfall is insufficient log retention, which can result in the loss of valuable data. It is important to establish a log retention policy that meets our specific needs and ensures that we retain logs for a sufficient period.

Of course, security also plays a significant role in logs. Collecting and storing logs can create security risks, as logs may contain sensitive information that could be exploited by attackers. It is important to implement adequate security measures, such as encryption and access controls, to protect logs and prevent unauthorized access.

Collecting and analyzing logs is a critical part of container monitoring in the cloud. By following best practices and using the right tools, we can gain valuable insights into the performance and behavior of containerized applications, enabling us as administrators to identify and troubleshoot issues quickly and effectively. However, we must keep our eyes open to pitfalls, one of them doing too much. It is all about making the right choices.

This concludes our chapter about containers. The next cloud-native technology that we will explore is the concept of serverless functions. It is the topic of the next chapter.

Conclusion

In this chapter, we discussed the concepts of containers. We learned how we could use container technology to develop, provision, and manage applications in the cloud. First, we studied what a container exactly is and what the benefits of deploying applications using container technology are. Then we looked at the container offerings of the major cloud providers AWS, Azure, GCP, and Alibaba Cloud. All of them offer both managed and unmanaged services to deploy and manage containers on their platforms. We learned that we need a container orchestration platform to run containers. Kubernetes has evolved to be the industry standard.

We looked extensively at the DevSecOps process to build applications and deploy them using containers. In the chapter, we stressed that security best practices must be bolted into the container development process to ensure that security is not an afterthought but built into the entire process. This includes implementing security testing and scanning as part of the CI/CD pipeline and using automation to enforce security policies.

The final part of this chapter was about setting up monitoring and logging. We learned that setting the right metrics in monitoring is crucial. When it comes to monitoring and logging, the biggest pitfall is that we get swamped by data if we are not precise enough in setting the right metrics.

Points to remember

Containers have unique properties that require careful consideration during the application design and deployment process. Think of micro-services, resource allocation, and security settings. It is essential to take measures to secure containers, such as using secure container images, applying security patches, and limiting container privileges. A base image can help to keep container deployments consistent.

Major cloud providers offer a variety of managed container services, but these might not fit the multi-cloud approach. We can use a multi-cloud container management platform. These platforms provide a unified view of container management across different clouds. A multi-cloud container management platform will allow us to manage containers from a single interface, regardless of the cloud provider.

Monitoring is essential in managing containers, but we must make sure that we set the right metrics to prevent an overload of monitoring data and logs. Metrics like CPU usage, memory usage, and network traffic are all important, but we should also consider other metrics like container restarts, container uptime, and resource limits. These metrics can help in identifying potential issues.

Questions

We discussed an open-source project that defines standards for container formats and runtime. What is the name of that project?

What processes are recommended to include in DevSecOps?

What are Elasticsearch and Kibana?

Answers

OCI

SCA, SAST. DAST, Pen testing

Open-source tools for collecting and analyzing logs

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

Building and Managing Serverless

Introduction

Pay only for what you really use. Serverless does not mean that there are no servers involved, but that we only pay for server resources that we really need, such as typically CPUs, memory, and disk space for storage. But how do we deploy apps and services on serverless infrastructure?

In this chapter, we will learn how to define our environments as functions that we can deploy and manage as serverless environments. We will learn about event-driven architectures and discuss the various serverless frameworks and technologies, including the offerings from major cloud providers. In the final section, we will extensively talk about defining metrics to monitor serverless environments in multi-cloud settings.

Structure

In this chapter, we will discuss the following topics:

Understanding the serverless concept

Developing and provisioning serverless functions from architecture

Using CI/CD for serverless deployments

Managing multi-cloud environments with serverless frameworks

Following best practices in serverless

Deep dive into monitoring serverless

Understanding the serverless concept

Serverless computing is rapidly gaining popularity due to its many benefits. In this section, we will discuss what serverless computing is and what benefits it offers. We will also explore serverless concepts in AWS, Azure, GCP, and Alibaba Cloud.

The term serverless has gone through different meanings over time:

Early web Described applications that didn't require dedicated servers, relying on client-side technologies and using server-side infrastructure for data storage.

Backend as a Service Refers to third-party cloud services handling common backend functionalities, relieving developers from managing servers.

Function as a Service Associated with the execution of individual functions triggered by events without the need to manage underlying server infrastructure.

Serverless Encompassing a wider range of cloud services beyond functions, such as managed databases and other services that abstract away infrastructure management.

So, the first obvious question is: what is serverless computing? To answer that question, we first need to solve one misunderstanding. Serverless does not mean that there are no servers involved. It is still code, and it still needs to run on a machine. Serverless computing, or FaaS, is a cloud computing model that allows developers to create and run applications without the need to manage the machines, which are typically the virtual servers in the cloud. In this model, the cloud provider manages the infrastructure, including server management, scaling, and availability.

In a serverless architecture, developers write code in the form of functions that are triggered by events such as HTTP requests, database changes, or message queues.

Note that in the following section, we will explore this event-driven architecture in more detail. These functions run in a stateless environment, where the cloud provider manages the underlying infrastructure, and the developer is only charged for the execution time of the functions. This is the main difference between building and managing application code in a traditional way or the cloud-native way using containers and serverless functions.

Traditionally, building and managing applications required developers to focus on server management, infrastructure maintenance, and scaling. This process was time-consuming and required significant resources, including money and manpower. Let us clarify this in an example. In developing and deploying applications, we also needed to define the infrastructure that the code required to run. We needed to think of the network and how the various components of the application – the webserver, database, and application server, for instance – would

communicate with each other and how users would approach the server with the application. We needed to specify the capacity of the server: how much CPU and memory it would need to run the code and, at the same time, perform well enough. At some stage, the application code and the infrastructure would need to merge to create the desired environment.

But we were not yet done. We also needed to manage the infrastructure from that point onward. If the application would grow due to usage, the server would need to be expanded or the database enlarged. All of this would consume a lot of time.

Serverless computing simplifies this process, allowing developers to focus on writing code and building applications. With serverless, developers do not need to worry about infrastructure management, scaling, and availability anymore. This approach significantly reduces development time, reduces costs, and allows developers to focus on building the applications. To summarize, the benefits of using cloud-native concepts such as serverless are:

Serverless computing is cost-effective since developers only pay for the time their code runs. With traditional cloud computing, developers pay for the entire infrastructure, regardless of the actual usage.

Serverless functions allow for the automatic scaling of applications in response to user demand. This allows developers to build applications that can handle high traffic without worrying about infrastructure management.

Developers can focus on writing code and building applications without worrying about infrastructure management. This approach significantly reduces development time and increases productivity. This also means that the effort of managing resources is heavily reduced. Since the cloud provider manages the infrastructure, developers can reduce the time and resources spent on maintenance.

In terms of security, there's no server to scan to gather vulnerabilities and exploit them. We basically always have a clean server without the need to patch.

All cloud providers offer a variety of services that provide serverless functions and computing. Here are the most used ones:

AWS Lambda is Amazon's serverless computing service that allows developers to run code in response to events. Lambda supports several programming languages, including Java, Python, Node.js, and C#. Developers can create and deploy Lambda functions in minutes, and the service automatically scales applications in response to user demand.

Lambda functions can be triggered by various events, such as changes to data in an Amazon S3 bucket, updates to a database, or incoming messages from an Amazon Simple Queue Service queue. This event-driven architecture enables us to build serverless applications that respond to events in real time.

Other use cases involve, for instance, the Internet of Things where functions execute custom business logic based on the incoming sensor

data, perform real-time analytics, and trigger actions or alerts. In other words, Lambda functions can be used for a lot of different cases.

Lambda is likely the best-known AWS serverless service, but of course, AWS has more to offer. Amazon API Gateway is a fully managed service to create, deploy, and manage APIs at any scale. It provides features like throttling, caching, and security and integrates with other AWS services such as Lambda, DynamoDB, or S3.

AWS Step Functions is worth mentioning too. This is a serverless workflow service that lets us coordinate distributed applications and micro-services using visual workflows. We can define your workflows using AWS Step Functions state machines, which support multiple AWS services like Lambda, Fargate as a serverless service to orchestrate containers, or Glue for serverless data integration.

One more popular service is Amazon Simple Notification Service This is a fully managed messaging service that we can use to send notifications from the cloud to mobile devices, email, or HTTP endpoints. It supports multiple protocols, such as SMS, email, or HTTPS, and integrates with other AWS services like Lambda or S3.

Azure Functions is Microsoft's serverless computing service. Like Lambda, it supports several programming languages, including Java, Python, Node.js, and C#. Next to the basic Functions, we can use a variety of serverless solutions in Azure. This includes Azure Durable Functions as an extension of Azure Functions, which allows us to create stateful workflows and long-running tasks. It defines and orchestrates workflows using a declarative programming model.

Other services are Azure Logic Apps and Event Grid. Logic Apps is a serverless workflow orchestration service that allows us to create and run workflows that integrate with a wide range of services, including Office 365 and Salesforce. We can use pre-built connectors or create our own custom connectors. Azure Event Grid is a serverless event routing service that allows us to subscribe to and react to events from various Azure services and third-party sources.

The main serverless service of GCP is Google Cloud Functions, which is comparable to Lambda and Azure Functions, enabling event-driven architectures. Moreover, GCP offers some fully managed serverless services. A good example is Cloud Run: a fully managed serverless container platform to run stateless HTTP containers in a serverless environment. We can deploy containerized applications whilst Cloud Run will automatically scale the application based on traffic.

App Engine is a fully managed serverless platform to deploy and scale applications without worrying about infrastructure. It supports several programming languages, including Python, Java, Node.js, PHP, and Go.

A similar service to AWS SNS is Cloud Pub/Sub. It is a fully managed messaging service that enables sending and receiving messages between independent applications. It supports multiple protocols, like HTTP, gRPC, or Pub/Sub Lite, and integrates with other GCP services like Cloud Functions or Cloud Run.

Alibaba Cloud offers very similar services to AWS. The main serverless proposition is Function Compute, which we can use to run code in response to events. Function Compute supports several programming languages, including Java, Python, Node.js, and C#. Alibaba Cloud also offers a serverless API Gateway and message service. Where AWS offers DynamoDB as a serverless NoSQL database, Alibaba Cloud provides a service called Table Store as a fully managed NoSQL database service that provides real-time data access and storage. It can handle large amounts of data with automatic scaling and disaster recovery.

With serverless, developers can build and deploy applications without worrying about infrastructure management, scaling, and availability. AWS Lambda, Azure Functions, Google Cloud Functions, and Alibaba Cloud Function Compute are all good examples of serverless options. But the question arises: where do we start; how do we build and deploy these serverless services into our cloud environments? We will discuss that in the next section.

Developing and provisioning serverless functions from architecture

Developing and deploying serverless applications in the cloud requires a different approach than traditional application development. In this section, we will discuss the steps involved in developing and deploying serverless applications in the cloud, from architecture to deployment. Obviously, this process starts with defining the architecture. It is a misunderstanding that when we use serverless concepts, we do not have to bother about the architecture. Serverless options are often managed by the cloud provider, but we have to make sure that these serverless components fit into our architecture.

Defining the architecture can be cumbersome. Serverless architecture is event-driven, which means that the code is executed in response to events. The architecture consists of a set of functions that are triggered by events. These functions can be written in different programming languages, depending on the cloud provider. Event-driven architecture is an architectural pattern that emphasizes the production, detection, consumption, and reaction to events. What are the events in this case? Events are significant occurrences or changes in the system or the environment that may require action. In EDA, services or components communicate with each other by producing or consuming events. Figure 9.1 represents an event-driven architecture:

Figure 9.1: High level presentation of an event-driven architecture

EDA is often used in cloud computing environments as it provides an efficient and scalable way to build and operate complex distributed systems. But this leaves one question unanswered: what would we use EDA for? We can think of several use cases, such as:

IoT EDA is an ideal architecture for building Internet of Things applications. IoT devices generate a vast amount of data and events, which can be challenging to manage and process in real time. With EDA, IoT devices can publish events to the cloud, which can be processed by serverless functions, triggering other events and actions in response. For example, an IoT-enabled smart home could use EDA to turn on the lights when a motion sensor detects movement or to alert the homeowner when a door is left open.

Financial EDA is increasingly being used in the financial services industry to improve data processing and analysis, fraud detection, and real-time decision-making. Financial institutions can use EDA to consume events from multiple sources, including market data feeds, transaction systems, and customer interactions, to detect and prevent fraud, identify trading opportunities, and manage risk. A bank could use EDA to monitor transactions for suspicious activity, triggering an alert to the security team or temporarily blocking the account until the issue is resolved.

EDA is also well-suited for e-commerce applications that require real-time processing of large volumes of data. EDA can be used to process events from multiple sources, including customer interactions, inventory systems, and payment gateways, to provide a seamless and personalized experience for customers. Think of an e-commerce site that uses EDA to trigger personalized recommendations for customers based on their browsing and purchase history or to update inventory levels and pricing in real time.

In the healthcare industry, EDA is used to improve patient care and outcomes. EDA can be used to consume events from multiple sources, including medical devices, electronic health records, and patient interactions, to provide real-time monitoring, diagnosis, and treatment. An example is the hospital that uses EDA to monitor patients in real-time, triggering alerts to medical staff when a patient's vital signs fall outside of normal parameters.

EDA is a powerful and flexible architecture pattern that is well-suited to many business cases. EDA enables organizations to process, analyze, and act on large volumes of data and events in real time, providing significant benefits for businesses and their customers. But it all starts with a well-defined architecture.

Next, we have to choose our cloud provider. There are several cloud providers to choose from, including AWS, Azure, Google Cloud, and Alibaba Cloud. As we have learned in the previous section, each cloud provider has its own set of services, pricing models, and programming languages. It is important to choose a cloud provider that meets the specific requirements of the project. In fact, this is not different from anything else that we do in the cloud. With IaaS and containers, we always have to verify and validate if services are the right fit for our needs and requirements.

After defining the architecture and the selection of the cloud provider, we can really start to create and test the functions. Each function is a piece of code that performs a specific task. Functions can be written in different programming languages, depending on the cloud provider. It is essential to keep the functions small and stateless, as this makes them easier to manage and scale. Let us use an e-commerce site as an example, where the customer buys a good and automatically triggers the creation of an invoice that has to be sent to the customer. In this example, we use Azure Functions to create that function.

The code in C# - be aware that this is just an example - creates an Azure Function called CreateInvoice that is triggered by an HTTP POST request. When a customer buys a good, the e-commerce site sends an HTTP POST request to the Azure Function with the order information in the request body. The Azure Function then generates an invoice based on the order information and adds it to a queue for processing:

```
using System;

using System.IO;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Azure.WebJobs;
```

```
using Microsoft.Extensions.Logging;

using Newtonsoft.Json;

public static class CreateInvoice

{

    [FunctionName("CreateInvoice")]

    public static async Task Run(

        [HttpTrigger(AuthorizationLevel.Function, "post", Route = null)]
HttpRequestMessage req,

        [Queue("invoices")] IAsyncCollector invoiceQueue,



        ILogger log)

    {

        log.LogInformation("Creating invoice...");

        // Deserialize the request body to get the order information

        string requestBody = await req.Content.ReadAsStringAsync();
```

```
    dynamic data = JsonConvert.DeserializeObject(requestBody);

    // Generate the invoice

    string invoice = GenerateInvoice(data);

    // Add the invoice to a queue for processing

    await invoiceQueue.AddAsync(invoice);

    log.LogInformation("Invoice created and added to the queue.");

    return new OkResult();

}

private static string GenerateInvoice(dynamic data)

{

    // Generate the invoice using the order information



    string invoice = "Invoice for order #" + data.orderId + ":\n";

    invoice += "Item: " + data.itemName + "\n";

    invoice += "Price: $" + data.itemPrice + "\n";
```

```
        invoice += "Quantity: " + data.quantity + "\n";

        invoice += "Total: $" + (data.itemPrice * data.quantity) + "\n";

        // TODO: Add logic to send the invoice to the customer

        return invoice;

    }

}
```

Note that we will need to add logic to send the invoice to the customer, which could be done using Azure Logic Apps or This also makes clear that we need to test the function in its entire setting. Testing is a critical step in the development process, as it ensures that the functions are working as intended. Each function should be tested in isolation as well as in the context of the entire application.

The final step is the actual deployment to the cloud. This involves uploading the code to the cloud provider and configuring the triggers. Triggers can be configured to respond to events such as HTTP requests, database changes, or message queues.

Using CI/CD for serverless deployments

Typically, developers will integrate the development and deployment of functions into CI/CD pipelines. Serverless functions are designed to be highly scalable and flexible, which makes them a great choice for CI/CD pipelines. For instance, we can use Azure DevOps to deploy Azure Functions to different environments, such as development, staging, and production. This can be done by creating a release pipeline that packages the Azure Functions code and deploys it to the appropriate environment. Of course, we can also use Azure DevOps to manage the CI/CD pipeline and trigger deployments automatically when code changes are made.

Testing is more convenient in CI/CD too. Before deployment, we should perform unit testing, for example, with unit testing frameworks such as NUnit or xUnit. This can help to catch bugs and ensure that serverless functions are working correctly before deploying them to production. Next, we have to execute integration testing since the function will be part of an entire environment. For this, we can use popular tools such as Postman or Selenium.

Once the functions have been deployed, it is essential to monitor them. Monitoring ensures that the functions are running correctly and that there are no issues. It is also essential to scale the functions as needed. Serverless applications are designed to scale automatically in response to user demand. This ensures that the application can handle high traffic without any issues. In the final section of this chapter, we will extensively

discuss how we can monitor our serverless components in cloud environments.

Managing multi-cloud environments with serverless frameworks

This book is about multi-cloud and assumes that we develop, deploy and run applications in multi-cloud settings. How would that work with serverless concepts? It is possible to use serverless concepts in a multi-cloud environment, although it can introduce some additional complexities and challenges.

Using a framework is a common approach to working with serverless concepts in a multi-cloud environment. A framework can help you abstract away the underlying infrastructure and provide a consistent interface to deploy and manage your serverless functions across multiple cloud providers. There are multiple frameworks that will help us in architecting, deploying, and managing serverless concepts in multi-cloud. In this section, we list the most important ones:

Serverless This is a popular open-source framework that supports multiple cloud providers, including AWS, Azure, and Google Cloud. It provides a simple and consistent way to define and deploy serverless functions, as well as manage other related resources, such as APIs, databases, and event sources. An example of a multi-cloud solution using the Serverless Framework is provided in Figure

Figure 9.2: Example of working with serverless functions using a Serverless Framework

In the example from the Serverless Framework, we see that traffic to order something in a webshop is split into two streams. Fifty percent is sent to Azure, and the other half to AWS. Orders can be made either using a mobile app or through a browser via the Internet. Both architectures use the same format; in Azure, we see Azure Functions that pick up the order from the cart and next checks the inventory. In AWS Lambda, functions are used to execute the same tasks.

Open Functions as a Service This is another open-source framework that supports multiple cloud providers, as well as on-premises environments. It

has become very popular over the past years. It provides a platform for building and deploying serverless functions using Docker containers deploying to Kubernetes and supports a wide range of programming languages: Go, Java, Python, C#, Ruby, Node.js, and PHP. We can also bring existing micro-services written with custom frameworks like Express.js, Vert.x, Flask, ASP.NET Core, FastAPI and Django. This makes OpenFaaS very powerful.

OpenFaaS allows for invoking functions through events from Apache Kafka, AWS Simple Que Service PostgreSQL, Cron, and Message Queuing Telemetry Transport as the standard for IoT messaging.

Figure 9.3 shows a conceptual design for deploying OpenFaaS functions. Each function is built into an immutable Docker image before being deployed:

Figure 9.3: Conceptual design for OpenFaaS using Docker images

Apache Kafka deserves a bit more explanation since it has gained popularity in defining and designing event-driven architectures in the cloud. Apache Kafka is an open-source distributed event streaming platform that provides a highly scalable, fault-tolerant, and reliable way of processing and storing real-time data streams. It was initially developed by LinkedIn and later donated to the Apache Software Foundation. Kafka is based on the publish-subscribe messaging model, where data is sent from producers to Kafka topics, and consumers subscribe to these topics to receive data. It is designed to handle large amounts of data and can process millions of messages per second. The power of Kafka lies in the fact that it can process data streams in real time, which is essential in EDA. Thanks to the increasing popularity of OpenFaasS, there is a very active community that posts developments and best practices in open-source repositories on GitHub. Find everything about OpenFaaS, including templates and scripts in various languages, at

Two other serverless frameworks that are worth mentioning are Apache OpenWhisk and Kubeless:

Apache OpenWhisk: Like the Serverless Framework and OpenFaaS, OpenWhisk is an open-source serverless platform that supports multiple cloud providers, including IBM Cloud, AWS, and Google Cloud. It provides a flexible and extensible architecture for building and deploying serverless functions, as well as event-driven workflows and APIs. The principle of OpenWhisk is shown in Figure

Figure 9.4: Architectural overview of Apache OpenWhisk

OpenWhisk is a loosely coupled architecture that makes OpenWhisk scalable. Each layer is designed to scale independently. Actions, rules, and triggers are created and managed through REST endpoints. To invoke an action, the event source needs to call the trigger REST API. One of the building blocks in OpenWhisk is Kafka, which handles the events streaming.

Since OpenWhisk is open source, documentation and templates can be found at

Kubeless: This serverless framework runs on top of Kubernetes and supports multiple cloud providers, including AWS, Azure, and Google Cloud. It provides a platform for building and deploying serverless functions using Docker containers. The architecture for Kubeless is shown in Figure

Figure 9.5: Architectural Overview of Kubeless

Using a framework will help abstract away the differences between different cloud providers and provide a consistent way to develop and deploy serverless functions. However, it is important to choose a framework that supports the cloud providers and programming languages that we have chosen to build and run our environments. Next, it is strongly recommended to follow best practices for secure coding, monitoring, and management. Let us review some of these best practices.

Following best practices in serverless

The serverless frameworks that we discussed in the previous section help in creating and deploying functions that are agnostic to a specific cloud provider, preventing a lock-in. To avoid being locked into a specific cloud provider, it is important to use cloud-agnostic serverless technologies and frameworks. This means choosing tools and services that are compatible with multiple cloud providers, such as Kubernetes, Docker, and Apache OpenWhisk.

By using cloud-agnostic tools, we can easily move applications and services between cloud providers as required. This is enabled by common standards. Using common standards and protocols can help ensure compatibility and interoperability between cloud providers. For example, using APIs that conform to RESTful architecture principles can make it easier to integrate with different cloud providers. Also, using common languages and frameworks, such as Python or Node.js, can help ensure that serverless functions are portable across cloud providers.

Although we will talk extensively about monitoring in the final section of this chapter, we already stress the need to implement monitoring and logging that is capable of monitoring serverless functions. Monitoring and logging are essential to detect and diagnose issues in your serverless applications. It is recommended to use a centralized monitoring and logging platform that can work across multiple cloud providers. Tools such as Prometheus, Grafana, and ELK Stack can help to monitor and troubleshoot serverless functions across different clouds.

When we talk about best practices, we must address security, which is paramount in serverless environments. We must implement proper security measures, such as authentication, access control, and encryption, to protect data and applications. Best practices include secure coding and deployment and regularly scanning applications for vulnerabilities. In event-driven architecture and serverless environments, data is transmitted between services, and if proper security measures are not implemented, there is a risk of data breaches or unauthorized access. Attackers can exploit vulnerabilities in serverless functions or event streams to gain access to sensitive data or execute malicious code.

As event-driven architectures and serverless computing rely on third-party services, there is always a risk of service outages, which can cause downtime and loss of revenue. It is crucial to have a backup plan or failover strategy in place to ensure high availability. This can involve replicating data and functions across different cloud providers, implementing automated failover mechanisms, and regularly testing your backup and recovery procedures.

To simplify the management of serverless applications in a multi-cloud environment, we should consider using a cloud management platform that can provide a unified view and control of the applications across different cloud providers. Platforms such as VMWare's CloudHealth and RightScale can help you manage your serverless functions, as well as other cloud services, across multiple clouds.

Serverless functions are very powerful, but deploying and managing these environments requires careful consideration and implementation of best practices. To conclude this section, we will summarize these practices briefly:

Set up monitoring and Serverless environments generate a lot of data, and it is important to set up monitoring and logging to track performance, identify issues, and debug problems. Cloud providers offer native services with the required capabilities. Think of AWS CloudWatch and X-Ray. In multi-cloud settings, we might prefer third-party solutions to monitor and log serverless functions.

Optimize function Since serverless functions are billed based on usage, it is important to optimize their performance to minimize costs. Use techniques such as function caching, code optimization, and event-driven architectures to improve performance.

Use security best Serverless environments can be vulnerable to security threats, so it is important to use security best practices like network isolation, encryption, and least privilege access controls. As always, we must consider and implement Identity and Access Management to manage access to serverless resources. Remember that everything in the cloud is an identity, and thus we must define what an identity is allowed to do and where it has access. This includes serverless functions.

Implement scaling Serverless environments are designed to automatically scale based on usage, but it is still important to implement scaling strategies to ensure optimal performance. Use techniques such as auto-

scaling, sharding, and load balancing to ensure that your serverless functions can handle any level of traffic.

Test Serverless environments can be complex, so it is important to thoroughly test code before deploying it to production. Cloud providers like AWS provide tools such as Lambda Layers to manage dependencies and ensure consistent environments across development, testing, and production. For testing serverless in multi-cloud environments, Selenium might be a good option. Be aware that we only mention a few tools. Obviously, there are many more. Selecting the right one that fits our requirements will take time.

Use version Serverless environments can be highly dynamic, so it is important to use version control to manage changes to the code and configuration. Git and GitHub are likely the most popular tools to utilize as repositories with version control. Many companies, especially the larger ones, use Innersource, a software development approach that brings open-source development practices and methodologies inside an organization's boundaries. Innersource enables internal teams to collaborate and share code across teams and business units, using the same principles as open-source development. It is a way to promote transparency, reuse, and knowledge sharing within an organization.

When it comes to security, other best practices can be added. Think of:

A monorepo, short for the monolithic is a software development approach where multiple projects or applications are stored within a single version control repository. In the context of serverless, a monorepo can be used to manage multiple Lambda functions or serverless applications within a single repository. This approach simplifies code sharing, dependency

management, and deployment processes, as all the related serverless components are kept together.

Choke points or In serverless architectures, choke points or functions refer to the components or services that handle a large portion of the workload or act as the central processing hub. These functions typically have higher computational requirements or are responsible for coordinating and orchestrating other serverless functions. Choke points should be carefully designed and optimized to ensure they can handle the anticipated load without becoming a bottleneck.

Granule Granule policies, also known as granular access policies, refer to fine-grained permissions that can be assigned to serverless functions or resources within a serverless architecture. With granule policies, you can control and restrict the actions that individual functions or resources can perform. This allows for precise access control, enhancing security and minimizing the risk of unauthorized or unintended operations.

Serverless functions should be designed and optimized to execute quickly and efficiently. Techniques such as reducing unnecessary dependencies, optimizing resource utilization, and implementing caching mechanisms can help improve performance and reduce execution time.

By implementing these best practices, we can effectively manage serverless environments in the cloud and ensure optimal performance, scalability, and security. However, we promised that we would look at monitoring in more detail as a critical component of our architecture. That is the topic of the final section.

Deep dive into monitoring serverless

Monitoring serverless functions in a multi-cloud environment requires a different approach than monitoring in a single-cloud environment. In this section, we will discuss the preferred approach and how to set metrics for monitoring. If we work in a multi-cloud setting, we will need a multi-cloud monitoring solution. Multi-cloud monitoring solutions such as Datadog or Dynatrace enable monitoring serverless functions across multiple clouds from a single dashboard. These solutions can provide insights into metrics like latency, error rate, and invocation count. However, whatever tool we will use, we must specify the metrics first.

Custom metrics are important for monitoring serverless functions in a multi-cloud environment. Metrics provide visibility into the behavior and performance of your application, helping you identify and diagnose issues before they become critical problems. These metrics should be relevant to the application and help measure the performance and behavior of the application. For example, if the application is processing orders, we may want to monitor the number of orders processed per second, the average processing time per order, and the number of failed orders. We may also want to monitor metrics related to resource utilization, such as CPU usage and memory usage. Let us review the most important metrics:

Invocation metrics are used to track the number of times a serverless function is invoked, as well as the duration of each invocation. These metrics can help to identify potential bottlenecks and optimize the

performance of serverless functions. Some invocation metrics to monitor include:

Invocation The number of times a function is invoked.

Invocation The amount of time it takes for a function to complete an invocation.

Error The percentage of invocations that result in errors.

Cold start The percentage of invocations that require a cold start, which can impact the latency of the first request.

Resource metrics track the utilization of serverless resources such as memory, CPU, and network bandwidth. These metrics help to optimize resource usage and ensure that we have allocated enough resources to handle the workload of the application. Some resource metrics to monitor include:

Memory The amount of memory used by a function during an invocation.

CPU The amount of CPU time used by a function during an invocation.

Network The amount of network bandwidth used by a function during an invocation.

Error metrics track the occurrence and frequency of errors in serverless functions. These metrics are essential to troubleshoot issues that may be

impacting the performance and reliability of the application. Some error metrics to monitor include:

Error The percentage of invocations that result in errors.

Error The type of errors that occur, such as timeouts or memory errors.

Error The specific error message associated with each error.

Cold start metrics track the time it takes for a serverless function to initialize and execute its first request. Cold starts can impact the latency and performance of an application, so it is important to monitor and optimize them. Some cold start metrics to monitor include:

Cold start The amount of time it takes for a function to initialize and execute its first request.

Cold start The percentage of invocations that require a cold start.

Cost metrics track the cost of running serverless functions. These metrics will help to identify cost optimizations and ensure that costs are not exceeding the allocated budgets. Some cost metrics to monitor include:

Execution The amount of time a function is executed, which impacts the cost of running the function.

Memory The amount of memory used by a function, which impacts the cost of running the function.

Invocation The number of times a function is invoked, which impacts the cost of running the function.

Once we have identified the key metrics that we want to monitor, the next step is to choose a monitoring solution. There are several monitoring solutions available, including open-source tools like Prometheus, commercial tools like Datadog and New Relic, and cloud-native tools like AWS CloudWatch and Google Cloud Operations. When choosing a monitoring solution, consider factors like cost, ease of use, scalability, and integration with your existing toolchain.

We have defined the metrics and selected our monitoring tooling. The next step is to set up metrics collection. This involves instrumenting code to emit metrics and sending those metrics to the monitoring solution. Most monitoring solutions provide libraries and SDKs for various programming languages that make it easy to instrument your code. For example, CloudWatch provides the CloudWatch Logs Agent for collecting logs and metrics from Amazon EC2 instances; remember that serverless does not mean that there are no servers involved, and on-premises servers, while Google Cloud Operations provides an SDK for collecting metrics from Google Cloud Functions.

In addition to monitoring metrics, it is also important to log events and errors in the application. Logging provides a record of what happened in your application, making it easier to diagnose issues and troubleshoot problems.

We will need to define alarms and alerts at this stage. Alarms and alerts notify us when a metric exceeds a certain threshold or when a specific event occurs. For instance, we may want to set an alarm when the number of failed orders exceeds a certain threshold or when CPU usage exceeds a certain percentage.

After defining the metrics and configuring monitoring and logging, the final step is to monitor and analyze the collected metrics and logs. It is recommended to define log filters: allowing to search and filter through logs to find specific events or errors. By analyzing metrics and logs, we will be able to identify and diagnose issues before they become critical problems. Most monitoring solutions provide dashboards and visualization tools to make the life of an administrator easier.

Some other best practices also include using distributed tracing and log aggregation. Distributed tracing will help to track requests as they move through multiple services and clouds. This is important for identifying bottlenecks and issues in complex, distributed systems. With Log aggregation, we can collect logs from multiple sources and store them in a single location.

By implementing these best practices and setting up relevant metrics, we will monitor serverless functions in an effective way, identifying issues before they impact the application and, with that, the end user. This concludes the part about creating the interoperable multi-cloud. The next part of this book will be about securing the multi-cloud, starting with identity and access management.

Conclusion

In this chapter, we discussed serverless concepts that we can use in multi-cloud. We have learned that in serverless concepts, we typically only pay for server resources that we really need, such as CPUs, memory, and disk space for storage. We discovered that serverless functions are important components in event-driven architectures where an event triggers an action that must be executed. An example is a user who buys something in a webshop: the purchase triggers the action to verify if the purchased item is in stock, next pick it up for delivery, and prepares the invoice to the customer. All these actions can be performed by functions.

Major cloud providers offer a variety of serverless options, such as Azure Functions and AWS Lambda. We also learned that we could have serverless concepts cloud-agnostic, operated across multiple cloud platforms. Serverless frameworks are a good help to guide us in architecture, developing, and deploying event-driven environments in multi-cloud. OpenFaaS has gained a lot of popularity as a framework. Most frameworks use serverless functions in combination with containers.

The final part of this chapter was about setting up monitoring and logging. We looked at the required metrics to monitor serverless functions, such as invocation metrics. Defining these metrics, tracking the number of times a serverless function is invoked, as well as the duration of each invocation, is crucial in serverless and event-driven architecture. These metrics can help to identify potential bottlenecks and optimize the performance of serverless functions.

Serverless is a fast-evolving technology. As more businesses recognize the cost-saving and efficiency benefits of serverless architectures, we can expect to see increased adoption across various industries. We also intensively discussed security as a major concern in serverless computing. Future developments might focus on enhancing security measures, such as better isolation of functions, improved access controls, and advanced threat detection mechanisms.

Remember that serverless computing doesn’t exist in a vacuum. It’s often used in conjunction with other technologies like containers, microservices, and machine learning.

This chapter concludes the section about cloud-native technologies. The next section will discuss implementing and managing security in multi-cloud.

Points to remember

The most important point to remember is that being serverless does not mean that there are no servers involved. There is still infrastructure that must be managed. In serverless concepts, the infrastructure, such as servers and networks, is typically managed by the cloud provider.

Serverless architecture is typically event-driven, which means that the code is executed in response to events. The architecture consists of a set of functions that are triggered by events.

Major cloud providers offer a variety of serverless functions. Serverless frameworks will help to define the architecture and enable portability in multi-cloud settings. A popular framework is OpenFaaS, which is open-source. It provides a platform for building and deploying serverless functions using Docker containers deploying to Kubernetes and supports a wide range of programming languages. A very active community manages a vast library of templates to get started with OpenFaaS in a fast yet very comprehensive way.

As with everything, proper monitoring is essential. Specific metrics for monitoring serverless functions in the event-driven architecture are invocation metrics, measuring how often functions are invoked and how they perform.

Remember our first point: There is still infrastructure and, thus, resources involved. Resources invoke costs. Part of monitoring serverless functions is monitoring costs and optimizing the performance of serverless functions.

Questions

Name two other serverless offerings in Azure.

What would we use Apache Kafka for?

What does OpenFaaS stand for, and what is it?

What do we measure with cold start

Answers

Event Grid and Logic Apps.

Handling streaming events in real-time.

Open Functions as a Service - open source serverless framework.

Amount of time it takes for a function to initialize and execute its first request.

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

Managing Access Management

Introduction

In the previous chapters, we discussed various cloud infrastructure concepts with IaaS, containers, and serverless. Whatever we choose as infrastructure, we must keep in mind that we are hosting data and applications on this infrastructure. We do not want just anyone to be able to access that data and the applications. Hence, we need access management, which is a complicated topic in multi-cloud. We will learn about that in this chapter, exploring identities, how to define roles and permissions using role-based access, and how we can control access to our resources in the cloud.

Structure

In this chapter, we will discuss the following topics:

Exploring the basics of access management

Using cloud tools for access management

Understanding and working with PAM

Creating and storing secrets

Defining, implementing, and managing role-based access control

Monitoring access control

Exploring the basics of access management

Access management plays a vital role in ensuring security and protecting sensitive data across multiple cloud platforms. In such an environment where organizations utilize different cloud service providers simultaneously, managing access becomes complex due to the diverse set of resources, applications, and users involved.

Access management provides a centralized control mechanism for managing user identities, permissions, and access rights across multiple cloud platforms. It allows us to enforce consistent security policies, ensuring that only authorized users can access specific resources or perform certain actions. Centralized control simplifies administration, reduces the risk of misconfigurations, and enhances security posture.

Users are required to authenticate themselves before accessing cloud resources, typically through username-password combinations, multi-factor authentication, or even biometrics. Strong authentication mechanisms prevent unauthorized access attempts and protect against password-related attacks, such as brute-forcing or credential stuffing.

Access management typically enables fine-grained authorization and follows the principle of least privilege, granting users only the permissions necessary to perform their tasks. But access management also involved segmentation and isolation. Multi-cloud environments often involve multiple tenants or business units within an organization. Access

management allows for the segmentation and isolation of resources and data between different tenants, preventing unauthorized access or data leakage between them. This isolation is critical for maintaining data confidentiality, integrity, and regulatory compliance.

It will be no surprise that access management in multi-cloud environments is complex. By adopting best practices such as using a unified identity management system, implementing a Role-Based Access Control model, and using a Cloud Access Security Broker organizations can simplify access management and reduce the risk of unauthorized access, data breaches, and compliance violations. In this chapter, we will extensively discuss all these topics. The final section will be about monitoring access management.

Let us first recap our definition of multi-cloud: multi-cloud refers to the use of multiple cloud computing services from different vendors or providers. In a multi-cloud environment, organizations can select different cloud services based on specific requirements, such as cost, performance, availability, and geographic location. As we have already discussed in the first chapter of this book, multi-cloud offers several advantages, such as the ability to avoid vendor lock-in, reduce single-point-of-failure risks, and optimize workload placement. However, managing access to cloud resources across multiple clouds can be challenging, as each cloud provider has its own access management tools and policies.

Understanding managed identities

Before we start exploring access management, we have to study identities in the cloud. More specifically: managed identity. After all, we will provide access to resources in our clouds, and access is provided to identities. What do we mean by managed identity?

A managed identity is a feature provided by cloud service providers that enables applications running in the cloud to authenticate and access cloud resources without the need for explicit credentials. In essence, a managed identity is a virtual identity associated with a specific resource, such as a virtual machine or a container instance. The cloud service provider manages the identity and provides it with credentials that can be used to access cloud resources. When an application needs to access a resource, it uses the managed identity to authenticate and access the resource.

Managed identities are important in the cloud because they help to simplify authentication and access management. Instead of managing explicit credentials for each application, we can use managed identities to grant permissions to specific resources. This reduces the risk of credential theft and makes it easier to manage access to cloud resources.

Another important benefit of managed identities is that they are automatically rotated and renewed by the cloud service provider, reducing the risk of credential theft or misuse. This helps to ensure the security and compliance of cloud resources. Later in this chapter, we will study secrets that are related to managed identities.

Challenges in access management

Now we can start talking about providing and managing access. The following are some of the challenges that we face when managing access to cloud resources in a multi-cloud environment:

Each cloud provider offers its own access management tools, which can be different in terms of syntax, semantics, and capabilities. This can make it difficult to develop consistent access policies and enforce them across multiple clouds.

In a multi-cloud environment, user identities and access privileges need to be managed across multiple cloud providers, as well as on-premises and legacy systems. This requires a unified identity management approach that can synchronize identities, roles, and permissions across multiple clouds.

Managing compliance and audibility across multiple clouds can be challenging, as different cloud providers may have different compliance requirements and audit trails. Organizations need to develop a unified compliance and audibility framework that can provide visibility and control across multiple clouds.

In the previous chapters, we studied cloud-native services such as containers and serverless options. Access management is a critical aspect of securing cloud-native services, as these services often involve multiple

components, and the management of access to each component needs to be properly controlled.

To address these challenges, it is best to adopt best practices for multi-cloud access management. Some of them are as follows:

First and most important of all, we should use a centralized identity management system that can synchronize user identities, roles, and permissions across multiple clouds. This can help ensure consistent access policies and reduce the risk of unauthorized access.

We should develop and adopt an RBAC model in which we assign roles to users based on their job functions and responsibilities. This can help simplify access management and reduce the risk of human errors.

The least privilege must be the default. Users are only granted the minimum level of access needed to perform their job duties. This will reduce the risk of data breaches and limit the scope of potential damage.

We must use a unified policy management system that can provide a single view of access policies across multiple clouds. This is to ensure consistent access policies and to simplify compliance and auditability.

Finally, a CASB is a security tool that can provide visibility and control over cloud usage and enforce security policies across multiple clouds. CASBs help to detect and prevent security threats, monitor compliance, and provide audibility across multiple clouds.

Basically, a CASB is a tool that supports us in keeping our clouds secure. In the next section, we will explore various tools that we can use for access management.

Using cloud tools for access management

Access management is about granting access to identities. Hence, we also need to manage identities in our cloud environments. Before we get into tools to manage identities, we need to underpin an important principle. We can use the tools that cloud service providers offer. But be aware that a cloud provider is typically only responsible for the security of the cloud, while the customer is responsible for security in the cloud, including access.

A solution to manage identities is Identity as a Service which is something different than CASB. We will explain the difference, making clear why we need both. This is very helpful if a user needs to have access to multiple systems that are hosted in different environments, which is typically the case in multi-cloud.

IDaaS solutions provide a centralized platform for managing user identities, access controls, and authentication across multiple cloud services and applications. Some popular IDaaS solutions include Okta, Microsoft Azure Active Directory, and Google Cloud Identity. These solutions provide a range of features, such as single sign-on multi-factor authentication and user provisioning and de-provisioning. A reference architecture for IDaaS is shown in Figure

Figure 10.1: High-level example architecture for IDaaS using AD as Identity Provider

CASB solutions, on the other hand, provide a layer of security and access control between cloud services and end-users. CASB solutions can be used to monitor and control user access to cloud services, enforce security policies, and provide visibility into user activities. Some popular CASB solutions include Netskope, McAfee Skyhigh, and Symantec CloudSOC. These solutions can provide features such as Data Loss Prevention threat detection and response, and granular access controls. A high-level reference architecture for a CASB solution is shown in
Figure

Figure 10.2: High-level architecture for CASB

When it comes to multi-cloud environments, IDaaS and CASB solutions can help organizations manage user identities and access controls across multiple cloud services and applications. This can help improve security and compliance, as well as simplify the management of user access.

In addition to IDaaS and CASB solutions, there are also other cloud tools and services that can be used for access management in multi-cloud environments. For example, cloud-native Identity and Access Management services such as AWS IAM and Google Cloud IAM can be used to manage access to cloud resources and services within their respective cloud environments. These services can provide features such as fine-grained access controls, identity federation, and audit logging.

Implementing IDaaS and CASB solutions involves several key steps:

The first step is to define the requirements for the solution, including the scope of the cloud services and applications to be covered, the types of access controls and policies needed, and the user groups to be managed.

Once the requirements are defined, the next step is to evaluate different IDaaS and CASB solutions based on features, scalability, cost, and ease of integration with existing systems.

Next, a detailed implementation plan should be developed that includes timelines, resource requirements, and integration points with existing systems.

After the implementation plan is in place, the solution can be configured according to the requirements defined earlier. This involves setting up policies, access controls, and authentication mechanisms and integrating the solution with existing user directories.

The solutions must be tested thoroughly to ensure it is working as expected. After testing is complete, the solution can be deployed in production.

Managing IDaaS and CASB solutions requires ongoing monitoring and maintenance. Some key points to consider include:

Regular review of Access policies should be reviewed periodically to ensure they remain relevant and effective.

Monitoring user User activity should be monitored to detect and respond to suspicious behavior or policy violations.

Continuous The solution should be continuously improved based on user feedback, changing requirements, and emerging threats.

Integration with other The solution should be integrated with other security and management systems, such as SIEM and ITSM tools, to ensure a holistic view of security and compliance across the organization.

Training and Users must be trained on the use of the IDaaS and CASB solutions and made aware of policies and controls to ensure compliance and reduce risk.

This final step may look obvious, but it will take some time to train users to adopt standards in access management, especially when we have applied the principle of least privilege. In some cases, this means that users cannot do everything that they were used to. They will only be granted access to the activities they need to perform for their job. There might be exceptions where user rights have to be elevated. Moreover, in that case, we want to be fully in control. That is the essence of privileged access management, the topic of the next section.

Understanding and working with PAM

Privileged Access Management is a security practice that involves controlling and monitoring the access of privileged users to critical systems and data. Privileged users are individuals or accounts that have elevated access privileges, such as system administrators, database administrators, and application developers.

Before we start exploring the details of PAM, it's good to have a deeper understanding of the Zero Trust concept as a major trend in IAM and PAM. Zero Trust is a security framework that assumes no inherent trust in any user or device, whether inside or outside the network perimeter. It emphasizes the principle of never trust, always In a Zero Trust model, every user, device, and network component must be authenticated and authorized before accessing resources, regardless of their location or network segment. IAM solutions ensure that the right individuals have appropriate access to the right resources at the right time, whereas PAM focuses specifically on managing and securing privileged accounts, which have elevated access rights and permissions.

In a Zero Trust model, IAM plays a crucial role in implementing strong authentication and authorization mechanisms. IAM solutions ensure that users are properly identified, authenticated, and authorized before accessing resources, regardless of their location or network segment. IAM helps enforce the Zero Trust principle by consistently verifying user identities and access privileges throughout the organization's infrastructure.

Zero Trust frameworks often incorporate privileged access controls as a critical component. Privileged accounts pose a higher risk due to their extensive access rights. Zero Trust principles demand that even privileged users or accounts should not be automatically trusted, and their access must be continuously verified and authorized based on contextual factors. PAM solutions provide the necessary controls to secure and manage privileged access, aligning with the Zero Trust philosophy of never assuming trust and constantly validating access.

To we need PAM solutions to elevate the rights of users and systems in a controlled manner. Implementing PAM will reduce the risk of insider threats, external attacks, and accidental or intentional data breaches by controlling access to privileged accounts and monitoring user activity. It also is a tool to meet regulatory compliance requirements by providing audit trails and documentation of privileged user activity. With PAM in place, administrators can manage the process of granting and revoking access privileges, ensuring that access is provided only when needed and revoked promptly when no longer required.

With the increasing adoption of multi-cloud environments, PAM has become even more critical. Organizations must manage multiple cloud services and platforms, each with its own privileged access management mechanisms. Without a centralized approach to PAM, organizations face the risk of inconsistent access controls, a lack of visibility into privileged user activity, and potential security breaches.

There are several tools available for PAM in cloud environments, including:

CASBs provide a centralized approach to managing access controls across multiple cloud services and platforms. They enable organizations to define policies and access controls and enforce them consistently across different cloud environments.

IAM solutions support centralized management of user identities and access privileges across multiple cloud services and platforms. We need IAM to manage access controls, enforce policies, and monitor user activity.

Lastly, we can use specific PAM solutions, providing a centralized approach to managing privileged access across multiple cloud services and platforms. These tools help us in managing and controlling access to privileged accounts, enforce policies, and monitor user activity.

There are several risks associated with PAM, including the risk of insider threats, external attacks, and data breaches. These risks can be mitigated by implementing a robust PAM program that includes access controls, monitoring of user activity, and regular audits. PAM solutions must comply with regulatory requirements and industry standards. Some of the regulations and standards include the Payment Card Industry Data Security Standard the Health Insurance Portability and Accountability Act and the General Data Protection Regulation Naturally, PAM solutions must also adhere to specific organizational policies and guidelines.

All cloud providers offer their own native PAM solutions. The most common ones are as follows:

Azure

Azure Active Directory Privileged Identity Management Azure PIM enables organizations to manage and control access to privileged resources in Azure AD and other Azure services. It provides just-in-time access, role-based access control, and privileged access approval workflows.

Azure AD Identity Azure AD Identity Protection provides real-time detection and remediation of identity-based risks, including compromised accounts, suspicious sign-in activity, and risk events.

More information about Azure identity and access solutions can be found at

AWS

AWS Identity and Access Management AWS IAM enables organizations to manage access to AWS services and resources. It provides fine-grained access control, centralized user management, and policy-based access.

AWS Secrets AWS Secrets Manager helps organizations to manage and secure secrets such as database credentials, API keys, and encryption keys. It provides automated secret rotation, encryption at rest, and fine-grained access control.

More information about AWS identity and access solutions can be found at

GCP

GCP Identity and Access Management GCP IAM enables organizations to manage access to GCP services and resources. It provides fine-grained access control, centralized user management, and policy-based access.

GCP Cloud Identity-Aware Proxy GCP IAP provides secure access to applications running on GCP. It provides context-aware access control, multi-factor authentication, and secure remote access.

More information about GCP's identity and access solutions can be found at

Alibaba Cloud

Resource Access Management (RAM): RAM is a centralized access control service that enables organizations to manage access to Alibaba Cloud resources. It provides fine-grained access control, centralized user management, and policy-based access.

More information about Alibaba Cloud's identity and access solutions can be found at

In addition to these native PAM solutions, there are also third-party PAM solutions available for Azure, AWS, and GCP, such as CyberArk,

BeyondTrust, and Thycotic. These solutions offer more advanced features and functionality for managing privileged access in the cloud.

Privileged session and privileged behavior analytics

We discussed IAM, PAM, and Zero Trust, but there are two more concepts that will help us in managing identities and access: Privileged Session Management (PSM) and Privileged Behavior Analytics (PBA).

PSM focuses on securing and monitoring privileged user sessions when accessing critical systems, applications, or sensitive data. The main components of PSM are:

Secure PSM solutions enforce secure access controls by acting as intermediaries between privileged users and target systems. They establish a secure session through a proxy or gateway, often employing techniques like protocol isolation, session isolation, and application-level encryption. This ensures that privileged access is protected against interception, unauthorized access, or tampering.

Just-in-Time PSM implements the concept of just-in-time privilege, where users are granted temporary elevated access only for the duration of their specific task or session. This minimizes the exposure of privileged credentials and reduces the risk of misuse or compromise of privileged accounts.

Session monitoring and PSM solutions enable comprehensive monitoring and recording of privileged sessions. They capture detailed logs of all activities performed during the session, including commands executed,

files accessed, and changes made. Session recordings serve as valuable audit trails for forensic analysis, compliance reporting, and post-incident investigations.

Session termination and access PSM solutions facilitate the automatic termination of privileged sessions once the authorized task is completed or after a predefined period of inactivity. This ensures that access is promptly revoked, reducing the risk of unauthorized access to critical resources.

With PBA, we can analyze the behavior and activities of privileged users to detect anomalous or suspicious actions that may indicate insider threats, compromised accounts, or malicious activities. PBA solutions leverage machine learning algorithms and advanced analytics techniques to identify patterns and deviations from normal privileged user behavior. Typically, PBA involves the following components:

Baseline PBA solutions establish a baseline profile of normal privileged user behavior by analyzing historical data, including user activity, access patterns, and usage patterns. This baseline is continuously updated and refined over time.

Anomaly PBA solutions monitor privileged user activities in real time and compare them against the established baseline. They look for deviations, outliers, or unusual behavior that may indicate potential security threats, such as unusual access attempts, unauthorized system changes, or suspicious data transfers.

Risk scoring and PBA solutions assign risk scores to detected anomalies based on their severity and contextual factors. High-risk activities trigger

alerts or notifications to security teams, enabling them to investigate and respond to potential security incidents promptly.

Behavioral profiling and adaptive PBA solutions employ adaptive learning techniques to enhance the accuracy of anomaly detection. They continuously update behavioral profiles based on new data and feedback, adapting to evolving privileged user behavior patterns and improving the effectiveness of threat detection.

By integrating PSM and PBA into our PAM solution, we can strengthen our security posture by securing privileged sessions, reducing the attack surface, and proactively detecting and mitigating potential insider threats or unauthorized activities by privileged users. These components work together to provide granular control, real-time monitoring, and behavior-based insights into privileged access, ensuring the confidentiality, integrity, and accountability of critical systems and data.

So far, we talked about identities and granting access to these identities in a controlled way. To ensure that identities and access to these identities are protected, we need to attach secrets to these identities and their access tokens. In the next section, we will study how we can implement secrets and keys.

Creating and storing secrets

One of the most important aspects of security in cloud computing is managing access to resources. This requires careful management of secrets - any sensitive information such as passwords, API keys, or access tokens that can grant access to systems or services.

Let us recap shortly what we learned so far. Access management is about controlling who can access cloud resources and what actions they can perform on those resources. To enable this, we need secrets and keys to authenticate users and applications, encrypt data in transit and at rest, and establish secure connections between cloud resources. If secrets and keys are not properly managed, it can lead to unauthorized access to sensitive information, data breaches, and other security incidents.

For example, if a user's password is compromised, an attacker could use that password to gain access to cloud resources and potentially steal sensitive data. Similarly, if encryption keys are not properly protected, an attacker could potentially decrypt sensitive data that is being transmitted or stored in the cloud.

Proper secrets and key management help prevent these types of security incidents by ensuring that secrets are properly protected and access is tightly controlled. This includes generating keys securely, rotating them on a regular basis, and properly destroying them when they are no longer needed. It also involves monitoring access to secrets and auditing usage to

identify potential security issues. Most important: don’t store secrets in code repositories. This common mistake often leads to security breaches.

In the following sections, we will briefly discuss the various key management offerings of AWS, Azure, GCP, and Alibaba Cloud. Most services include the creation of keys, adding cryptographic parameters, storing the keys in a protected vault, and using automation to rotate the keys in regular intervals.

Although we will focus on the native tools to store secrets in the various clouds, we have to mention Hashicorp Vault as a widely used open-source tool for secrets management that can be used across various platforms, including AWS, Azure, GCP, and Alibaba Cloud.

Managing secrets and keys in AWS and Azure

In AWS, there are a few different services that can be used for managing secrets. The most commonly used service to store and manage secrets is AWS Secrets Manager. This service also automatically rotates secrets on a regular basis.

To use AWS Secrets Manager, you will need to follow these steps:

We can create a secret using the AWS Management Console, AWS CLI, or AWS SDK. When creating a secret, we will need to specify the type of secret, for instance, a database password. Next, we must specify the value of the secret and any additional metadata that we want to associate with the secret.

Now we can grant access to the secret. Controlling access to secrets is done through AWS Identity and Access Management policies. It is recommended to create policies that grant specific permissions to users or roles, such as the ability to retrieve a secret or rotate it.

AWS Secrets Manager can automatically rotate secrets on a schedule that we define. Scheduling is the best practice, but we can also manually rotate secrets using the AWS Management Console, AWS CLI, or AWS SDK. The code is an example in Python to demonstrate how we can create a secret using a Boto3 SDK. It should be noted that we use Boto3 since it makes it easy to integrate a Python application, library, or script with AWS services:

```
import boto3
from botocore.exceptions import ClientError


def create_secret(secret_name, secret_value):
    """
    Creates a new secret in AWS Secrets Manager.


    :param secret_name: The name of the secret.
    :param secret_value: The value of the secret.
    :return: The ARN of the created secret.
    """
    # Create a Secrets Manager client
    client = boto3.client('secretsmanager')


    # Create the secret
    try:
        response = client.create_secret(
            Name=secret_name,
            SecretString=secret_value
        )
        return response['ARN']
    except ClientError as e:
        print(f"Error creating secret: {e}")
        raise
```

Now we can call the function with the desired name and value of our secret:

```
arn = create_secret(’my-secret’, ’my-secret-value’)

print(f"Created secret with ARN: {arn}")
```

This will create a new secret in AWS Secrets Manager with the name my-secret and value my-secret-value and return the ARN of the created secret. It should be kept in mind that we will need to have the appropriate IAM permissions to create secrets in AWS Secrets Manager in order to use this code.

More information on AWS Secrets Manager can be found at

The way of working in Azure is comparable to the example that we discussed in AWS. The most commonly used service for storing and managing secrets is Azure Key Vault. Like AWS Secrets Manager, Key Vault automatically rotates secrets on a regular basis.

Creating a secret in Key Vault is done through the Azure Portal, Azure CLI, or by using Azure PowerShell. We will need to specify a name, subscription, resource group, and location. Next, we specify the type of secret, the value of the secret, and any additional metadata that you want to associate with the secret.

The PowerShell code will look like this:

```
# Connect to Azure using Azure PowerShell module

Connect-AzAccount
```

```
# Define the name of the Key Vault and the secret to be created

$keyVaultName = "my-key-vault"



$secretName = "my-secret"

$secretValue = "my-secret-value"

# Get the Key Vault

$keyVault = Get-AzKeyVault -VaultName $keyVaultName

# Create the secret

$secret = Set-AzKeyVaultSecret -VaultName $keyVaultName -Name
$secretName -SecretValue $secretValue

# Output the secret ID

Write-Output "Created secret with ID $($secret.id)"
```

We only need to replace the values for and $secretValue with our desired values and then run the script. The script will connect to Azure using the Azure PowerShell module, get the key vault with the specified name, create a new secret with the specified name and value, and then output the ID of the created secret.

Controlling access to secrets is done through Azure Active Directory policies.

Azure Key Vault will automatically rotate secrets on a schedule that we define. There is an option to do this manually, but it is strongly advised to automate this as it will increase the level of security.

One remark: the script assumes that the Key Vault already exists, so do make sure that you have a Key Vault that you can use:

```
az keyvault create --name --resource-group --location
```

More information on the service Azure Key Vault can be found at

Managing secrets and keys in GCP and Alibaba Cloud

In GCP, the primary service used for managing secrets is Google Cloud Key Management Service Google Cloud KMS allows you to create and manage cryptographic keys for use in encrypting data, as well as storing and managing secrets. To manage secrets in GCP, you can follow these steps:

First, we must create a key ring, a container for cryptographic keys, which can be used to encrypt and decrypt data. We can create a key ring using the GCP console or the Cloud SDK.

Within a key ring, we can create one or more keys. A key is a named object that represents a cryptographic key. You can create a key using the GCP console or the Cloud SDK. This may sound more complex than in AWS and Azure, but it is not very hard, as we will learn in the example. The code using the GCP SDK is written in Python. It will create a key ring and then create a key in KMS:

```
from google.cloud import kms_v1

# Set the parameters for the new key ring and key

project_id = "your-project-id"

location_id = "us-central1"
```

```
key_ring_id = "my-key-ring"

key_id = "my-key"



purpose = kms_v1.CryptoKey.CryptoKeyPurpose.ENCRYPT_DECRYPT

# Create a new key ring

client = kms_v1.KeyManagementServiceClient()

key_ring_parent = f"projects/{project_id}/locations/{location_id}"

key_ring = client.create_key_ring(request={"parent": key_ring_parent,
"key_ring_id": key_ring_id})

# Create a new key in the key ring

key_parent = f"{key_ring.name}/cryptoKeys"

key = client.create_crypto_key(request={"parent": key_parent,
"crypto_key_id": key_id, "purpose": purpose})

# Print the details of the new key

print(f"Created key {key.name} with purpose {key.purpose}")
```

We have to replace the placeholders and my-key with our desired values. Note that we have to make sure that we have the appropriate permissions to create key rings and keys in KMS. This code creates a new key ring in the specified project and location and then creates a new key in the key ring with the specified ID and purpose (in this case, Finally, it prints the details of the new key, including its name and purpose.

Once you have created a key, we can use it to encrypt and decrypt secrets. When creating a secret, we must specify the key to use for encryption.

Access to secrets is controlled by using Identity and Access Management policies.

Google Cloud KMS can automatically rotate secrets on a schedule. We should also define these schedules as part of the process. An alternative to KMS is Google Secret Manager.

It is important to ensure that your GCP IAM policies are properly configured to control access to secrets. Additionally, we must ensure that the cryptographic keys are properly protected and that we have a process in place for rotating keys on a regular basis.

More information on GCP KMS can be found at

Lastly, we will look at Alibaba Cloud. Like GCP, the primary service used for managing secrets is Alibaba Cloud KMS. To manage secrets in Alibaba Cloud, we will have to follow these steps, that by now you will recognize:

We create a key using the Alibaba Cloud console or API. When creating a key, we can specify the type of key and the key size.

Once you have created a key, we can use it to encrypt and decrypt secrets using the Alibaba Cloud console or API.

Access is controlled through policies in Resource Access Management

Alibaba Cloud KMS can automatically rotate secrets on a schedule.

Please note that in Alibaba Cloud, KMS is mainly used for key management rather than secrets. A more suitable service for managing secrets would be Alibaba Cloud Resource Access Management

More information about Alibaba Cloud’s KMS can be found at

Avoiding pitfalls in managing secrets and keys

Now that we have a good overview of the various services that cloud providers offer, we must also realize that managing secrets in the cloud can be a complex task. There are several common pitfalls that organizations should be aware of to ensure that their secrets are properly managed and protected.

One of the most critical aspects of managing secrets is ensuring that access to those secrets is properly controlled. Organizations should use IAM policies or similar tools to ensure that only authorized users or systems can access secrets. Failure to properly control access can result in unauthorized access to sensitive information. However, there is much more that we need to do to keep our secrets really secrets.

Secrets should be encrypted both in transit and at rest to protect against interception and unauthorized access. It is important to ensure that encryption is properly implemented and that keys are properly protected. Weak encryption can allow attackers to easily access secrets, even if access controls are properly configured. It is all part of proper key management. Keys should be generated securely, rotated on a regular basis, and properly destroyed when no longer needed. We have stressed a couple of times that we must define rules to rotate secrets and keys automatically. Secrets should be rotated on a regular basis to reduce the risk of unauthorized access. Failure to properly rotate secrets can result in an attacker being able to access a secret for an extended period, even if access controls are properly configured.

We have discussed the various services that cloud providers offer. These services do not take away the responsibility to have processes in place for key management. Many cloud services offer default configurations that may not be appropriate for all use cases. It is important to carefully review and customize default configurations to ensure that they meet the specific needs of your organization.

Be aware that storing and managing keys is an important part of audit controls. Monitoring and auditing are critical components of any secret management strategy. Organizations should monitor access to secrets and audit usage to identify potential security issues. We will extensively talk about this in the final section of this chapter. But first, we have one more task to fulfill as administrators: we must create a proper RBAC model.

Defining, implementing, and managing Role Based Access Control

Role-Based Access Control is a widely used access control mechanism that restricts system access based on the roles or positions of individual users within an organization. RBAC helps organizations to manage access to critical resources, applications, and systems, thus preventing unauthorized access and reducing the risk of data breaches. It is an essential step in identity and access management.

RBAC defines the roles, privileges, and permissions that are associated with each role in an organization. These roles are then assigned to users, who are granted access to the resources and applications that they need to perform their jobs. RBAC typically involves three components: roles, permissions, and users. Roles are defined based on the job functions, positions, or responsibilities within the organization. Permissions are the actions that users are allowed to perform, and users are the individuals who are assigned to specific roles.

Implementing RBAC involves several steps, including identifying the roles and permissions needed for each role, defining the rules and policies that govern access control, and implementing the RBAC model into the organization's systems and applications. The following steps can be followed to implement RBAC:

Identify the roles that are necessary for the organization and define the job functions and responsibilities associated with each role.

Define the permissions that are required for each role, including the actions that users are allowed to perform.

Assign roles to users based on their job functions and responsibilities.

Implement the RBAC model into the organization's systems and applications, including access control policies and rules.

Be aware that RBAC policies can vary greatly among different cloud providers. Also, RBAC policies can be applied at different levels, including at the management group, subscription, resource group, and resource levels.

Once we have implemented the model, we must monitor user activity, review access rights, and update access policies as needed. This includes managing user accounts, reviewing user activity logs, and modifying user access permissions as necessary. The principle of least privilege should always be followed. Of course, it's crucial to perform continuous reviews and updates of roles and permissions as a part of security hygiene.

Least privilege is the concept of granting users and applications only the minimum level of access necessary to perform their required functions. In a multi-cloud environment, where data and applications are distributed across multiple clouds and providers, applying the principle of least privilege is essential to ensure the security and integrity of the organization's data and resources.

Multi-cloud environments can be complex and challenging to manage, and the risk of security breaches and data loss is high. Access controls are essential to limit the exposure of sensitive data and applications to unauthorized access, and the principle of least privilege provides an effective approach to achieving this goal. By granting users and applications only the permissions they need, the risk of malicious activities and unauthorized access is significantly reduced.

Implementing the least privilege in multi-cloud environments requires careful planning and attention to detail. It involves identifying and documenting the different levels of access required by users and applications and defining policies and procedures for granting and revoking access. Additionally, it involves regularly monitoring and reviewing access logs to ensure compliance and identify any potential security threats.

To ensure that our RBAC model and least privilege are safeguarded, follow these steps:

Monitor user activity to identify any potential security risks or unauthorized access attempts.

Regularly review user access rights and permissions to ensure that they are appropriate for the user's job function and responsibilities.

Update access policies as needed to reflect changes in the organization's structure, job functions, or security requirements.

RBAC is an effective access control mechanism that can help organizations to manage access to critical resources, applications, and systems. Defining, implementing, and managing RBAC involves identifying roles and permissions, defining access control policies and rules, and monitoring user activity to ensure that access rights are appropriate and secure. However, as we have concluded, proper monitoring is critical. That is the topic of the last section.

Monitoring access control

Multi-cloud environments can present unique challenges when it comes to monitoring access control. With multiple cloud providers, each with its own access control mechanisms, it can be difficult to get a comprehensive view of who has access to what resources. It requires a comprehensive approach that includes centralized IAM systems, RBAC, access control logging and monitoring, regular policy reviews, and automation. By implementing these best practices, organizations can ensure that access to critical resources is properly managed and monitored and that security risks are identified and addressed in a timely manner.

First, we must look at monitoring our keys and secrets. Keys and secrets are used to authenticate and authorize access to resources, applications, and services in the cloud. As such, monitoring and securing these keys and secrets is critical to maintaining the security of the entire environment. Some specific requirements for monitoring keys and secrets in multi-cloud environments include:

Implementing a centralized key and secret management system can help to ensure that keys and secrets are properly managed and secured across all cloud environments. This can also provide a single point of control for monitoring and auditing access to keys and secrets.

Keys and secrets should be encrypted and stored securely using industry-standard encryption algorithms and secure storage systems. This will

prevent unauthorized access and ensure that keys and secrets are only accessible to authorized users.

Access to keys and secrets should be controlled and monitored, with access granted only to authorized users and roles. The principle of least privilege must be adhered to. Access rights may be elevated following a strict process and using PAM. Access should be monitored and audited to ensure that any unauthorized access attempts are quickly detected and addressed.

Keys and secrets should be rotated regularly, with new keys and secrets generated and old ones revoked or retired. Unauthorized access will be avoided by ensuring that keys and secrets are not compromised due to long-term use.

Keys and secrets management in multi-cloud environments should comply with regulatory and compliance requirements, such as PCI-DSS, HIPAA, and GDPR. This can help to ensure that the organization meets its legal and regulatory obligations.

By implementing these requirements, organizations can ensure that keys and secrets are properly secured and managed across all cloud environments. But this is only about the keys and secrets. Obviously, we also must monitor resources and access to these resources themselves.

Implementing a centralized IAM system that can manage access across multiple cloud environments can help to simplify access control and reduce the risk of unauthorized access. This can also provide a single

point of control for monitoring access across all cloud environments. Other topics we should address are:

RBAC can help to ensure that users have only the necessary access permissions for their job functions and responsibilities.

Implementing logging and monitoring of access control events will help to detect and respond to unauthorized access attempts. Any security incidents must be quickly identified and addressed.

Regularly reviewing access policies and permissions can help to ensure that they are up-to-date and aligned with the organization's security policies and compliance requirements.

It is strongly recommended to simplify access control through automation. For example, automation can be used to automatically provision and de-provision access permissions, ensuring that access is properly managed and controlled.

What do we need to manage and monitor access control? We discussed the various tools already with IAM, PAM, and CASB solutions. Additionally, organizations implement SIEM solutions: Security Information and Event Management. SIEM systems are designed to collect, analyze, and correlate security events and alerts across multiple sources, including cloud services. They can provide visibility into access control events and help detect and respond to security incidents.

While SIEM systems can provide valuable insights into access control events in multi-cloud environments, they may not be the best option for

managing access control directly. SIEM systems are typically used for security monitoring and incident response and may not provide the granular control required for access management. Instead, IAM and PAM solutions are better suited for managing access control in multi-cloud environments.

We have talked a lot about monitoring in the past chapters. Using centralized systems for this is almost a necessity in multi-cloud environments. A popular word to describe this centralized monitoring is a single pane of glass In multi-cloud, this single pane has become crucial to have one integrated view of all our resources in various clouds. That is the topic of the next chapter.

Conclusion

In this chapter, we learned that access management in multi-cloud environments can be challenging. Proper access management is crucial for ensuring the security of sensitive data and resources. It requires the use of various tools and techniques, including IAM, PAM, and CASB.

IAM is essential for managing user identities and controlling access to cloud resources. IAM enables organizations to define user roles and permissions and ensure that access is granted only to authorized users. PAM is important for managing access to administrative accounts and other high-risk areas. PAM enables organizations to enforce tighter controls around privileged access, reducing the risk of data breaches. In addition, organizations should use secrets and keys to further secure access to cloud resources. Secrets and keys can be used to authenticate users, applications, and devices accessing cloud resources. This helps to prevent unauthorized access and reduce the risk of data breaches.

Cloud Access Security Broker is a security solution that sits between an organization's on-premises infrastructure and cloud provider to provide visibility, control, and protection of cloud resources. CASB solutions can monitor access and usage of cloud resources and enforce policies to ensure compliance with security regulations.

The final section was about monitoring which is critical to detect and respond to security incidents. Organizations must monitor access to cloud

resources to ensure that only authorized users are accessing them. This includes tracking access attempts, identifying failed login attempts and suspicious activity and analyzing access logs for potential security incidents.

Points to remember

Access management in multi-cloud environments is complex but critical for ensuring the security and integrity of cloud resources. Organizations should adopt best practices such as using a unified identity management system, implementing a Role-Based Access Control Model, and using a Cloud Access Security Broker

Be aware that access control is a critical component in audits. Access management should comply with regulatory and compliance requirements, such as PCI-DSS, HIPAA, and GDPR.

The least privilege must be the default. Access rights might be elevated using a strict process with Privileged Access Management Privileged users are individuals or accounts that have elevated access privileges, such as system administrators, database administrators, and application developers.

One of the most important aspects of security in cloud computing is managing access to resources. This requires careful management of secrets - any sensitive information such as passwords, API keys, or access tokens that can grant access to systems or services. Cloud providers offer services for key management that must be explored and implemented to protect resources in cloud environments.

The best practice is to have keys automatically rotated. All key management offerings of cloud providers include automatic rotation of keys and secrets.

Organizations must develop an RBAC, including the definition of roles and permissions that are needed for each role.

Questions

What are the typical services that are captured in a CASB system?

What does the least privilege mean?

What are keys and secrets used for?

Answers

Compliance check, data security, threat protection, and data loss prevention.

Minimal rights that an identity needs to perform a job.

Keys and secrets are used to authenticate and authorize access to resources, applications, and services in the cloud

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

Managing Security

Introduction

When we have environments, data, and apps distributed in various clouds, including on-premises-private clouds, we need to have security in place. Security is not a menu, though. We either do it or not, but there is nothing in between. With distributed environments, we must use frameworks that cover the various cloud technologies and tools that can handle various clouds, and we need a single pane glass of view to keep track of it all. That is the topic of this chapter.

Structure

In this chapter, we will discuss the following topics:

Working with cloud security frameworks

Choosing the security tools

Managing security through one lens

Keeping track and up to date with security trends

Working with cloud security frameworks

As we have learned in this book, organizations are increasingly adopting cloud computing to harness its scalability, flexibility, and cost-efficiency. As businesses evolve, they often find themselves relying on multiple cloud service providers simultaneously, giving rise to the concept of a multi-cloud environment. While this approach offers numerous advantages, it also introduces unique security challenges that must be effectively managed. With multiple cloud providers involved, each with its own security frameworks and practices, ensuring a cohesive and robust security posture becomes a complex task.

Each cloud service provider has its own set of security tools, configurations, and access controls, making it challenging to establish a unified security strategy. Moreover, ensuring compliance with various regulatory requirements across different cloud platforms adds another layer of complexity, not to mention the criticality of the secure exchange of data and information between different cloud providers and on-premises systems. Establishing secure communication channels and implementing encryption mechanisms becomes crucial to protect sensitive data from unauthorized access and interception.

Furthermore, managing identity and access control across multiple clouds poses a significant challenge. Maintaining a central identity management system that seamlessly integrates with various cloud platforms while maintaining strong authentication and authorization mechanisms becomes essential to prevent unauthorized access and data breaches.

Additionally, monitoring and detecting security incidents across multiple clouds can be intricate. The ability to have centralized visibility and threat intelligence across the entire multi-cloud environment is crucial for early detection and prompt response to potential security breaches. Security frameworks will help us in setting up comprehensive security management.

Before we dive into the cloud security frameworks, we must learn what the most common frameworks are that we can or even must use in the cloud. Moreover, just as important is the question: why do we need to adhere to the principles and guidelines in these frameworks? The most common frameworks are:

Cloud Security Alliance (CSA) security guidance: CSA provides a comprehensive framework for secure cloud computing. It covers a wide range of security domains, including governance, risk management, data protection, and compliance.

National Institute of Standards and Technology (NIST) cybersecurity framework: NIST's framework provides guidelines for managing and improving cybersecurity risk. While not specific to cloud computing, it can be applied to secure cloud environments effectively.

ISO/IEC 27001: This international standard outlines the requirements for establishing, implementing, maintaining, and continually improving an Information Security Management System It includes specific controls and practices for cloud security.

Payment Card Industry Data Security Standard (PCI DSS): PCI DSS is a set of security standards designed to protect credit card data. It includes

requirements for securing cardholder data in cloud environments that handle payment card information.

FedRAMP: The Federal Risk and Authorization Management Program is a U.S. government program that provides a standardized approach to security assessment, authorization, and continuous monitoring of cloud services. It is primarily used for federal agency cloud deployments.

CIS controls: The Center for Internet Security Controls provide a set of best practices for cybersecurity. While not specific to the cloud, it includes guidelines that can be applied to secure cloud environments effectively.

GDPR: The General Data Protection Regulation is a European Union regulation that sets rules for the protection of personal data. It includes requirements for the secure processing and storage of data in the cloud.

These frameworks and standards provide organizations with guidelines, controls, and best practices to ensure the security of cloud computing environments. The choice of framework depends on factors such as industry regulations, compliance requirements, and specific security needs of the organization. Failing to be compliant with mandatory security guidelines and frameworks will definitively lead to severe consequences. These consequences can vary depending on the specific regulations and the jurisdiction in which the organization operates. Think of:

Non-compliance with security regulations can result in legal penalties, fines, or sanctions imposed by regulatory bodies or government agencies. The severity of the penalties can vary, but they can be substantial, especially for major breaches or repeated non-compliance.

Failure to comply with security regulations can expose organizations to lawsuits from affected individuals, customers, or business partners. This can lead to legal damages, settlements, or other financial liabilities. Consequences can be severe. Think of data breaches, fines, penalties, lawsuits, but also disruption of business and loss of market access. As an example, the European Union's GDPR requires companies to meet certain data protection standards to operate within the EU. Non-compliant companies may face restrictions in accessing the EU market, limiting their growth opportunities and putting them at a competitive disadvantage compared to compliant competitors.

Security breaches or non-compliance can result in significant reputational damage to an organization. News of a breach or failure to protect sensitive data can erode customer trust, lead to loss of business, and damage the company's brand image.

Non-compliance with security regulations can disqualify organizations from participating in certain contracts, partnerships, or business opportunities. Many clients and partners require proof of compliance as a prerequisite for engagement.

In the event of non-compliance, organizations may incur significant costs to investigate and address the issues, implement security improvements, and demonstrate compliance. This can involve investing in new technologies, hiring external experts, or conducting audits.

Failure to comply with security regulations can result in increased regulatory scrutiny, including audits and inspections. This can be a time-consuming and resource-intensive process, diverting attention and resources from core business operations.

Considering all of this, it is highly recommended to understand and adhere to the relevant security regulations and compliance requirements to mitigate these risks. Having said that, we have to realize that this is not an easy task. How do you determine to what frameworks our organization should adhere to, and how do you implement the guidelines from the relevant frameworks?

First, let us analyze what frameworks are relevant by following these crucial steps:

The first step is to identify the relevant industry-specific regulations and legal requirements that our organization must comply with. For example, if we handle payment card information, PCI DSS compliance is necessary. Similarly, if our organization operates in the healthcare industry, HIPAA compliance will almost certainly be applicable. This step helps narrow down the frameworks that align with the industry's specific requirements.

Next, we have to evaluate the security needs, objectives, and risk tolerance. Consider factors such as the type of data you handle, the cloud infrastructure, the sensitivity of the information, and also the security budget. This assessment will help us understand our unique security requirements and prioritize the frameworks that align with the organization's goals. Be careful, though, with the budget allocation: security is not something that we can do afterward. It has to be top of mind, and that will require investments. Breaches are way more expensive that taking accurate security measures.

We should consider the best practices in the industry that we work in or work for. Hence, we must review industry best practices and standards to identify frameworks that are widely accepted and recognized. Look for frameworks developed by reputable organizations such as the CSA, NIST, or ISO. Figure an overview of a lot of security and compliance frameworks which cloud

providers such as AWS and Azure are compliant with. In this case, we took the frameworks that Azure listed to be compliant with. You will recognize the differences between global standards, industry standards, and national standards such as US Government and more regional frameworks.

Figure 11.1: Compliance framework overview for Azure

For an overview of the compliance programs of the various providers, we recommend visiting https://aws.amazon.com/compliance/programs/ for AWS, https://azure.microsoft.com/en-us/explore/trusted-cloud/compliance for Azure, https://cloud.google.com/security/compliance for GCP and https://www.alibabacloud.com/trust-center for Alibaba Cloud.

In the next and final chapter of this book, we will study compliance in the cloud in more detail, but here is a brief overview:

We cannot be all specialists. It is advised to consult with security and compliance professionals who have expertise in the industry. They can provide valuable insights and guidance based on their knowledge and experience. These professionals can help us understand the specific frameworks that are commonly adopted in our industry and can offer recommendations based on the organization's needs.

Assess the regulatory compliance requirements for the organization. Determine the specific controls, certifications, or attestations needed to meet those requirements. This evaluation will help to identify frameworks that provide the necessary controls and guidelines for compliance.

Based on the above steps, prioritize the frameworks that align with the industry requirements, organizational needs, and compliance obligations. Consider the maturity, comprehensiveness, and practicality of each framework. It may be necessary to adopt multiple frameworks to address various aspects of security.

Keep in mind that the security landscape and regulatory environment evolve over time and at a mind-blowing pace. Regularly review the framework selection to ensure they remain up to date with the latest industry practices and regulations. Periodically reassess the organization's needs and revise the framework selection as necessary.

Remember that selecting the appropriate frameworks requires careful consideration of our specific organizational context and compliance obligations. We must involve relevant stakeholders, such as IT, legal, and compliance teams, in the decision-making process to ensure a comprehensive and well-informed approach.

Example of implementing CIS guidelines for Azure

We have selected the relevant frameworks, but that is only half of the work. Now we have to implement the guidelines coming from these frameworks. Let us take CIS as an example. CIS separates the controls into three categories: basic, foundational, and organizational, as shown in Figure

Figure 11.2: The CIS Framework, version 7 (taken from

Next, the CIS framework provides a set of security controls and recommendations per platform to enhance the security posture of your Azure infrastructure. For this, we use the CIS Benchmarks. Some key steps and best practices for implementing the CIS benchmark guidelines in Azure would, for example, include:

Identity and Access Management (IAM):

Enforce strong password policies and multi-factor authentication for user accounts.

Implement Azure Active Directory Privileged Identity Management to manage privileged access.

Follow the principle of least privilege by assigning permissions based on the user's roles and responsibilities.

Regularly review and update access controls to ensure only authorized individuals have access to resources.

Configuration and vulnerability management:

Continuously monitor and enforce compliance with Azure Defender for Cloud Security Policies to detect and remediate misconfigurations.

Regularly apply security updates and patches to virtual machines and Azure services using Azure Defender for Cloud or Azure Update Management.

Enable vulnerability assessment and integrate it with Azure Defender for Cloud to identify and remediate security vulnerabilities in your environment.

Monitoring and logging:

Enable Azure Monitor and configure logging and alerting for critical resources and security-related events.

Centralize logs using Azure Monitor Log Analytics and leverage Azure Sentinel for advanced threat detection and response.

Regularly review logs, investigate security events, and establish incident response processes.

Networking:

Implement Network Security Groups and Azure Firewall to control inbound and outbound traffic at the network level.

Utilize Azure Virtual Network Service Endpoints and Private Link to secure access to Azure services.

Configure Azure DDoS Protection Standard to mitigate Distributed Denial of Service attacks.

Implement Azure Bastion for secure remote access to virtual machines.

Data protection:

Implement encryption for data at rest and in transit using Azure Disk Encryption, Azure Storage Service Encryption, and Azure VPN Gateway.

Utilize Azure Key Vault to securely store and manage cryptographic keys, secrets, and certificates.

Implement backup and disaster recovery strategies for critical data using Azure Backup and Azure Site Recovery.

Application security:

Follow secure coding practices and regularly update and patch applications running on Azure.

Utilize Azure Web Application Firewall or Azure Front Door to protect web applications from common attacks.

Implement Azure API Management to secure and manage APIs.

Incident response:

Establish an incident response plan that outlines roles, responsibilities, and procedures for responding to security incidents.

Test and simulate incident response scenarios to ensure readiness.

Leverage Azure Defender for Cloud threat intelligence and automated response capabilities.

The CIS framework provides detailed guidelines for specific controls, configurations, and security measures. It is advisable to review the complete

CIS Azure Foundations Benchmark and consult Azure documentation and resources for step-by-step implementation guidance tailored to your specific Azure environment.

Obviously, if we run workloads in AWS, GCP, and Alibaba Cloud, too, or on on-premises stacks, then we will need to do the exact same exercise for these clouds. The benchmarks for all platforms can be found at

Now we have selected the frameworks and implemented the guidelines, and we need tools to manage our security. That is the topic of the next section.

Choosing the security tools

This section aims to provide a method to select tools and discusses some best practices in multi-cloud. However, it does not aim to dive deep or provide a comprehensive overview of all tools that are available. While cloud providers such as Azure, AWS, and GCP offer built-in security features, organizations must take a proactive role in enhancing security. It's about the data of the organization and that of its customers. Cloud providers will hand a toolbox, but it's up to the organization to use these tools or choose differently.

First of all, it is important to understand the concept of shared responsibility. The components of this concept are:

Infrastructure security: Cloud providers are responsible for the security of the underlying infrastructure, including data centers, networks, and hardware. They implement physical security measures, network isolation, and protection against external threats, such as DDoS attacks. Additionally, they ensure the availability and reliability of their cloud services.

Platform security: Providers offer security features and controls within their cloud platforms, including IAM, authentication mechanisms, and encryption services. They secure the platform components, such as virtual machines, containers, and databases, to protect against common vulnerabilities and exploits.

Compliance and certifications: Cloud providers adhere to industry standards and regulations, obtaining certifications like ISO 27001, SOC 2, and PCI DSS. They offer compliance frameworks, security audits, and reporting capabilities, ensuring their platforms meet the required security and privacy standards.

Customers of these providers, organizations that host their workloads and data in the cloud, have their own responsibility to enhance the security that cloud providers offer. As said at the beginning of this section, providers such as AWS, Azure, GCP, and Alibaba Cloud offer giant toolboxes with a lot of security tools. It is up to the organizations to use these tools, use different tools, or not use security tools at all. Let us be very clear about this: security is intrinsic and should never be neglected. In general, it means that organizations will have to collaborate closely with their provider:

Understand the organization's security requirements, data sensitivity, and compliance obligations. Perform a comprehensive risk assessment to identify potential threats and vulnerabilities specific to your multi-cloud environment.

Select security tools that can provide centralized visibility and control across multiple cloud platforms. Look for solutions that integrate with APIs provided by Azure, AWS, and GCP, allowing us to manage and monitor security configurations, access controls, and compliance policies from a single interface.

Implement robust IAM practices across all cloud platforms. Utilize RBAC, enforce strong authentication mechanisms (for example, multi-

factor authentication), and regularly review and audit user access privileges.

Leverage encryption mechanisms provided by cloud providers, such as Azure Key Vault, AWS and Google Cloud KMS. Encrypt sensitive data at rest and in transit, both within and between cloud providers.

Deploy network security tools, such as firewalls, intrusion detection/prevention systems, and Virtual Private Networks to safeguard network traffic between on-premises systems and the multi-cloud environment. Implement network segmentation and monitor network activities for anomalous behavior.

Utilize threat intelligence feeds and Security Information and Event Management solutions to detect and respond to security incidents. Implement continuous monitoring, log analysis, and real-time alerts to identify potential threats and intrusions across multiple cloud platforms.

Conduct periodic security audits and vulnerability assessments to identify and remediate any weaknesses. Monitor compliance with industry regulations and standards applicable to your organization, such as GDPR, HIPAA, or PCI DSS.

Educate employees on cloud security best practices, social engineering risks, and data handling policies. Promote a security-conscious culture within the organization and encourage reporting of potential security incidents or breaches.

The security tool landscape is continually evolving, so it would be extremely challenging to provide an exhaustive list. Organizations should review their security landscape very frequently, recommended every quarter. We must evaluate specific security requirements, industry regulations, and cloud provider integrations to choose the most suitable security tools for our multi-cloud environment. Let us explore some key security tools, including SIEM and Security Orchestration, Automation, and Response

SIEM tools collect, correlate, and analyze security event logs and data from various sources across cloud environments. They provide centralized visibility into security events, helping identify potential threats, detect anomalies, and respond to security incidents. SIEM tools enable real-time monitoring, log management, and advanced analytics to support proactive threat detection and incident response. Popular tools are Splunk, IBM QRadar, LogRhythm, McAfee Enterprise Security Manager, and Elastic. Some cloud providers offer native SIEM. Azure Sentinel is a good example of this.

SOAR platforms combine security orchestration, automation, and incident response capabilities. They streamline and automate security processes, allowing organizations to respond rapidly and effectively to security incidents across multi-cloud environments. SOAR tools integrate with various security systems and technologies, facilitating automated incident investigation, response coordination, and remediation workflows. Some popular products on the market are Demisto (Palo Alto), Swimlane, and Splunk Phantom.

We should also mention Cloud Security Posture Management CSPM tools enable organizations to assess and manage their cloud security posture by

continuously monitoring cloud configurations and detecting misconfigurations or security gaps. These tools provide insights into compliance violations, insecure configurations, and potential risks, helping organizations maintain a strong security posture across multiple cloud platforms. Palo Alto Networks Prisma Cloud, Dome9 (now part of Check Point Software Technologies), and Rapid7 are examples of popular CSPM tools. Native environments such as AWS Security Hub also include CSPM capabilities.

When we use containerization in our cloud environments, we must also consider specific security tools for containers. Container security tools focus on securing containerized environments, such as those based on Docker or Kubernetes. These tools ensure container image integrity, vulnerability scanning, runtime protection, and compliance enforcement. They help organizations secure applications and data running in containers across multi-cloud deployments. Aqua Security, Sysdig Secure, is Twistlock by Palo Alto are a few examples. Google Cloud Security Command Center includes capabilities specifically for container security.

One of the most critical aspects of cloud security is observability, which makes sense: you need to be able to ‘see’ before you can spot something. Observability provides real-time visibility and insights into the behavior and state of cloud environments. It will help us detect, investigate, and respond to security threats, incidents, and vulnerabilities effectively.

Observability enables proactive threat detection by continuously monitoring and analyzing cloud infrastructure, applications, and user activities. It helps identify security incidents, such as unauthorized access attempts, abnormal behavior, or indicators of compromise, allowing organizations to respond promptly and mitigate potential risks. How

would that work in practice? Think of the security team that uses observability tools to monitor network traffic, log events, and user access patterns across multiple cloud providers. They detect a sudden spike in network traffic to a critical server and investigate the incident promptly, discovering a potential Distributed Denial of Service attack and taking necessary measures to mitigate the threat.

Some popular tools for observability that are not platform specific include:

Sysdig Secure: Provides container and cloud-native security with built-in observability and threat detection capabilities.

Datadog: Offers cloud monitoring and observability solutions to gain visibility into cloud infrastructure, applications, and security metrics.

Dynatrace: Provides AI-driven observability and Application Performance Monitoring for cloud environments.

New Relic: Offers observability and monitoring solutions to gain insights into cloud infrastructure, applications, and user experience.

Google Cloud Operations: Provides a suite of observability tools, including monitoring, logging, and tracing, to enhance visibility in Google Cloud Platform

Observability allows organizations to conduct in-depth incident investigations and root cause analyses by capturing and correlating relevant data from different cloud sources. The security team will use the

tools to analyze access logs, authentication events, and user activity data. This will help to identify the compromised user account and trace the attack's origin, helping them patch the vulnerability, revoke unauthorized access, and implement additional security measures. Tools they can use for this are:

Elastic Observability (ELK Stack): Provides log management, monitoring, and APM capabilities for cloud environments, enabling comprehensive observability and analysis.

AWS CloudTrail: Offers visibility into user activity and API usage in AWS, facilitating incident investigation and auditing.

Azure Monitor: Provides monitoring and observability capabilities for Azure resources, including logs, metrics, and diagnostics.

GCP Cloud Logging and Cloud Monitoring: Enables log management, monitoring, and observability in the Google Cloud Platform.

Lastly, observability supports compliance requirements by providing visibility into cloud configurations, access controls, and data handling practices. It allows organizations to monitor adherence to security policies, regulatory frameworks, and industry standards, ensuring continuous compliance and simplifying audit processes.

In highly regulated industries like healthcare and finance, compliance with industry-specific regulations and standards is of utmost importance. Maintaining compliance ensures the protection of sensitive information, promotes transparency, and helps prevent fraudulent activities.

Observability plays a vital role in supporting and enhancing compliance efforts in these industries.

As an example, we can think of the financial institution that utilizes observability tools to monitor access controls, encryption settings, and data handling practices across their multi-cloud infrastructure. They ensure compliance with regulations like PCI DSS by actively monitoring and auditing cloud environments and promptly addressing any non-compliant configurations.

In this section, we discussed a lot of different tools with different focus areas. Many organizations will use a set of tools to cover their security needs and requirements. This will bring some serious challenges for administrators: how can they keep track of it all, especially in multi-cloud settings? In the next topic, we will discuss the concept of the single glass-of-pane view, managing security through one lens.

Managing security through one lens

With organizations increasingly adopting multiple cloud providers to leverage different services and capabilities, it becomes crucial to have a unified view of security across all cloud environments. The principle of a single pane glass view provides a comprehensive and centralized approach to managing security in a multi-cloud environment. In this section, we will explore how this principle can be implemented, the tools that can be used, and the benefits it offers. We will also discuss how AI can help in this.

First, let us look at the different layers of defense that we must implement. It is shown in Figure

Figure 11.3: Layers of defense in the cloud

The concept of a single pane glass view entails having a centralized security management platform that aggregates and correlates security information from various cloud providers. It allows security teams to monitor, analyze, and enforce security policies consistently across all cloud environments, regardless of the underlying cloud service providers. This approach enables organizations to gain better visibility, control, and compliance while minimizing complexities and ensuring a holistic security posture.

To implement the single pane glass view principle, organizations can utilize cloud-native security tools or adopt third-party security platforms that offer multi-cloud management capabilities. These tools typically provide features such as:

Cloud security monitoring: Tools that enable continuous monitoring of cloud environments, collecting logs, events, and metrics from multiple cloud providers. This allows security teams to detect and respond to security incidents promptly.

Centralized policy management: Platforms that offer a unified interface for defining and managing security policies across multiple cloud environments. This ensures consistent application of security controls, reducing the risk of misconfigurations or security gaps.

Threat intelligence integration: Integrating threat intelligence feeds into the security management platform enhances the ability to detect and respond to emerging threats across all cloud environments. This ensures proactive threat hunting and faster incident response.

Compliance and audit support: Tools that provide automated compliance checks against industry standards and regulations, enabling organizations to maintain regulatory compliance across multiple cloud environments. They can generate comprehensive reports for audits and streamline compliance processes.

Identity and access management: A centralized identity and access management solution that integrates with different cloud providers' identity services, allowing organizations to enforce uniform access controls and user authentication mechanisms across all clouds.

It's good to realize the pros and cons of both cloud-native and third-party tools when implementing security in the cloud. Pros of cloud-native could be integration and native insights. Cloud-native security tools are designed specifically for the cloud environment and seamlessly integrate with native cloud services and infrastructure. They often provide out-of-the-box compatibility with cloud platforms, simplifying deployment and management. Native tools offer native visibility and insights into the cloud environment, providing deep integration with cloud-specific APIs and telemetry data. They can leverage platform-specific features and capabilities, enabling organizations to gain granular visibility and control over their cloud assets.

The cons might be limited functionality, lack of possibilities for customization, and vendor lock-in. Third-party security tools are often designed to address a wide range of security challenges across different cloud platforms. They may offer advanced features, such as threat intelligence, advanced analytics, or specialized compliance modules, providing comprehensive security coverage. In addition, these tools are typically designed to work across multiple cloud environments, offering flexibility and compatibility for organizations adopting multi-cloud or hybrid cloud

architectures. They can integrate with various cloud platforms and provide centralized management and visibility.

Also, third-party tools offer more customization options, allowing organizations to adapt the tools to their specific security requirements. They may provide APIs and extensions for integration with existing security infrastructure, enabling seamless workflows and custom integrations.

Next to integration challenges, a downside of this might be the costs. Third-party security tools often come with additional licensing or subscription costs. By adopting a single pane glass view approach to multi-cloud security, organizations can reap several benefits:

A unified view of security across multiple clouds provides a holistic understanding of the organization's security posture, enabling proactive identification and mitigation of risks.

With centralized security policy management, organizations can streamline security operations, reducing complexity and eliminating the need for disparate tools and processes for each cloud provider.

The ability to enforce consistent security policies and controls ensures that security standards are uniformly applied across all cloud environments, reducing the likelihood of misconfigurations and vulnerabilities.

Centralized security monitoring and incident management enable quick detection, analysis, and response to security incidents, regardless of the cloud provider involved.

Organizations can maintain regulatory compliance more efficiently by having a centralized platform that helps automate compliance checks, generate reports, and streamline auditing processes.

Introducing Cloud Security Posture Management

Cloud Security Posture Management can help with our security challenges in multi-cloud. CSPM refers to the practices, tools, and technologies used to continuously assess, monitor, and manage the security posture of cloud environments. It focuses on identifying misconfigurations, vulnerabilities, and compliance violations within cloud infrastructure, applications, and services. CSPM helps organizations ensure that their cloud deployments adhere to security best practices, industry standards, and regulatory requirements.

CSPM tools scan cloud environments, including infrastructure-as-a-service platform-as-a-service and software-as-a-service to identify misconfigurations in security settings. This includes analyzing network configurations, access controls, encryption settings, storage configurations, and more. By identifying and remediating misconfigurations, organizations can reduce the risk of security incidents. Examples of such tools are Check Point, Aqua Security, Fugue, and Palo Alto, but there are many more. The basic capabilities of CSPM are shown in Figure

Figure 11.4: Basic capabilities of CSPM

CSPM tools assess cloud environments against regulatory requirements, industry standards, and security best practices. They provide automated checks to ensure that configurations align with specific compliance frameworks such as PCI DSS, HIPAA, GDPR, and CIS benchmarks. Compliance monitoring helps organizations maintain a strong security posture and demonstrate adherence to regulatory obligations.

With CSPM, we can perform vulnerability scanning within cloud environments to identify security weaknesses and potential entry points for attackers. They analyze cloud infrastructure, applications, and services for known vulnerabilities and provide recommendations for remediation. Regular vulnerability assessments are essential to proactively address security risks and protect against exploits.

Lastly, CSPM enables continuous monitoring of cloud environments, collecting logs, events, and metrics from various cloud service providers. They leverage machine learning and behavioral analysis techniques to detect anomalous activities, potential threats, and security incidents. Continuous monitoring ensures timely detection and response to security events, enhancing overall cloud security.

Good tools will not only identify security issues but also provide remediation recommendations. They offer guidance and automation capabilities to help organizations fix misconfigurations, apply security patches, and enforce security policies consistently across cloud deployments. Remediation workflows streamline the process of addressing security gaps and reducing the attack surface.

Using ML and AI, CSPM tools will assess and quantify the risks associated with identified security issues. They prioritize vulnerabilities and misconfigurations based on their potential impact, severity, and exploitability. Risk assessment enables organizations to allocate resources effectively and prioritize remediation efforts for the most critical security issues. How can ML contribute to this?

ML algorithms can analyze data and detect security threats through a process known as anomaly detection. It does so by following a number of steps:

Training ML algorithms require a training phase where they learn from historical or labeled data. In the context of security threat detection, this training data typically consists of examples of normal, legitimate behavior and known security threats. The algorithm analyzes the data to identify patterns, correlations, and features that distinguish normal behavior from anomalous or malicious activities.

Feature once the training data is collected, ML algorithms extract relevant features from the data. These features are specific attributes or characteristics that help describe the data and provide useful information for the algorithm to make predictions. In security threat detection, features can include network traffic patterns, system log events, user behavior, or application usage statistics.

Model based on the extracted features, ML algorithms create a model that represents the learned patterns and relationships between normal and malicious behavior. The model can take various forms, such as decision trees, neural networks, or support vector machines, depending on the specific algorithm used.

Real-time once the model is created, it can be deployed to analyze real-time or streaming data. Incoming data is fed into the ML algorithm, which compares the observed patterns and features against the learned model. The algorithm then assigns a probability or score to each data point, indicating the likelihood of it being a security threat or an anomaly.

Anomaly ML algorithms excel at identifying deviations from normal behavior. If the algorithm detects data points that significantly deviate from the learned patterns, it flags them as potential security threats or anomalies. These anomalies could indicate malicious activities, such as unauthorized access attempts, unusual network traffic, or abnormal user behavior.

Feedback ML algorithms continuously improve their performance by incorporating feedback. When security analysts or domain experts review and validate the algorithm's detections, they can provide feedback on

misclassified instances. This feedback is used to refine and update the model, making it more accurate over time.

ML algorithms will adapt and learn from new data over time. As the threat landscape evolves, the algorithm can be retrained with updated datasets to account for emerging threats or changes in normal behavior. This adaptive learning allows the algorithm to stay effective and up-to-date in detecting evolving security threats.

But it is not only tools and technology. It is a culture. A way of thinking. It is important to involve all relevant stakeholders, such as IT and security teams, as well as business leaders and end users, in order to ensure the success of the program. All of these stakeholders must be aware of the threats, the risk, and the potential damage that breaches will cause. It is essential to regularly review and update your posture management policies and procedures in order to stay ahead of evolving threats and compliance requirements. It means that we have a serious task of keeping up with the security trends, the topic of the final section of this chapter.

Keeping track and up to date with security trends

Let us start by saying that it is extremely hard to keep up with trends when it comes to security in multi-cloud. To understand this, we must first acknowledge that types of attacks in the cloud are rapidly evolving. Yet, we can identify some common types of attacks that very often start with misconfigurations as a cause of security breaches in multi-cloud environments. Attackers often exploit misconfigured access controls, insecure default configurations, or inadequate security settings.

Misconfigurations can lead to data breaches due to vulnerabilities in cloud services, weak authentication mechanisms, or compromised credentials. Attackers may target sensitive data stored in multi-cloud environments, such as Personally Identifiable Information or intellectual property. There are many ways to exploit these vulnerabilities, for instance, through insider threats where malicious or compromised insiders with legitimate access can abuse their privileges to steal or manipulate data, disrupt services, or perform unauthorized actions.

These are just examples. But how do we recognize vulnerabilities, and more important, how can we protect our environments from breaches? We extensively discussed tools, but we should also have knowledge of how attacks work. TTP is a process that helps us understand tactics, techniques, and procedures. A TTP analysis will help security teams detect and mitigate attacks by understanding the way threat actors operate. A framework that certainly helps in keeping track of identifying threat actors is Mitre ATT&CK, which can be found at

TTPs, as presented in Mitre ATT&CK, are an essential framework for identifying and understanding security threats in multi-cloud environments. They provide a structured approach to analyzing and categorizing the methods employed by attackers, enabling security teams to effectively detect, prevent, and respond to threats. Let us know more:

Tactics refer to the high-level objectives or goals that attackers aim to achieve. Understanding the tactics employed by threat actors helps security teams anticipate potential attacks and develop appropriate defensive measures. Some common tactics in multi-cloud security include unauthorized access, data exfiltration, privilege escalation, lateral movement, and denial of service.

Techniques describe the specific methods and tools used by attackers to execute their tactics. By studying the techniques employed in past attacks or known attack patterns, security teams can recognize similar activities or behaviors within their multi-cloud environments. Techniques may include exploiting misconfigurations, conducting phishing campaigns, using malware, leveraging insider access, or exploiting vulnerabilities in cloud services.

Procedures outline the step-by-step processes or sequences of actions followed by attackers to achieve their objectives. Procedures encompass the tools, infrastructure, and actions taken by threat actors during an attack. Understanding the procedures helps security teams identify the Indicators of Compromise associated with specific attack campaigns, enabling early detection and response. Procedures may involve

reconnaissance, initial compromise, persistence, lateral movement, data exfiltration, or covering tracks.

By analyzing and mapping TTPs, security teams can identify patterns, signatures, or anomalies indicative of potential threats or ongoing attacks in multi-cloud environments. Some examples of ways that TTPs help in threat identification are listed as follows:

Signature-based detection: TTPs enable the creation of signatures or rules that match known attack patterns. Security tools can then use these signatures to identify and alert malicious activities that align with specific TTP.

Anomaly detection: By understanding the expected behavior and TTPs associated with various threats, security systems can detect anomalies or deviations from the norm. Unusual network traffic, abnormal user behaviors, or unauthorized actions can be flagged as potential threats.

Threat intelligence: TTPs serve as valuable information for threat intelligence gathering. Security teams can leverage public or private sources to gather insights on known TTPs associated with specific threat actors or campaigns. This intelligence helps in proactively identifying emerging threats and taking appropriate preventive measures.

Security incident response: During incident response, understanding the TTPs associated with an attack can help security teams investigate, contain, and remediate the incident effectively. TTPs provide valuable context, allowing analysts to trace the attack chain, identify affected systems, and develop appropriate response actions.

It is important to note that TTPs alone may not provide complete protection against evolving threats. Regular updates to TTP knowledge, continuous monitoring, and a multi-layered security approach that combines TTP analysis with other security measures are crucial for effective threat identification and mitigation in multi-cloud environments.

Now, imagine that we all have to do this manually. With a rapidly evolving landscape and an increasing number of attacks, there is simply no way to keep track of all security breaches manually. We need to automate to keep our environments continuously protected and, by doing so, compliant. That is the topic of the final chapter.

Conclusion

Managing security in a multi-cloud environment presents numerous challenges due to the complex nature of distributed systems and the diverse set of cloud service providers, such as AWS, Azure, GCP, and Alibaba Cloud. In this chapter, we learned how we can address these challenges effectively, for instance, by leveraging security frameworks such as the Center for Internet Security benchmarks, along with appropriate tool selection and the implementation of a single pane glass view. Additionally, Cloud Security Posture Management solutions can play a crucial role in managing security across multiple cloud platforms.

We extensively discussed the single pane of glass concept, which is crucial in multi-cloud. This refers to a unified dashboard or interface that provides a centralized view of the security posture across all the cloud environments being used. A single-pane glass view helps security teams gain visibility into the overall security landscape, identify potential vulnerabilities or misconfigurations, and respond swiftly to emerging threats.

In the final section, we studied the concept tactics, techniques, and procedures to understand how we can identify possible new threats. We had a closer look at the Mitre ATT&CK framework that will help us in keeping up with security trends.

Points to remember

The first step is to understand what security frameworks apply to our business, industry, or geographical location where our business resides. We must learn what the most common frameworks are that must use in the cloud and to which principles and guidelines our organization must adhere.

The CIS benchmarks provide guidelines and guardrails for various cloud platforms. They are a great help in defining the security policies in an organization that uses the cloud.

Cloud platforms are very well protected, yet organizations are still responsible for how they protect their own workloads when they use cloud technology. Azure, AWS, GCP, and Alibaba Cloud offer built-in security features and tools, but organizations must take a proactive role in enhancing security.

In multi-cloud, we should adopt the principle of single pane glass view: a centralized security management platform that aggregates and correlates security information from various cloud providers. It allows security teams to monitor, analyze, and enforce security policies consistently across all cloud environments, regardless of the underlying cloud service providers.

To understand security threats, we must understand how attacks work by learning Tactics, Techniques, and Procedures The Mitre ATT&CK is a good framework that will help in this.

Questions

What are the key capabilities of CSPM?

Name three popular tools for observability.

What are CIS Benchmarks?

Answers

Discovery and observability, continuous threat detection, misconfiguration management and remediation, and devsecops integration

Datadog, New Relic, Sysdig, Dynatrace

Guidelines and guardrails to implement security policies in the cloud.

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

Automating Compliancy

Introduction

Part of security is compliancy, which has become a very important topic for almost every organization entering the cloud. Governmental bodies, certification authorities, and auditors are setting compliancy guardrails to allow usage of major cloud providers. These guardrails tend to change every now and then, leaving organizations with tasks to stay compliant with their environments, including data and apps. A recent example of this is the implementation of the Software Bill of Materials by the US government. Automating compliancy can be a way to keep in control of these developments, keeping track of all environments on multi-cloud platforms.

Structure

In this chapter, we will discuss the following topics:

Understanding compliance in multi-cloud

Automating governance

Using RPA for automating compliance

The next step: using AI for compliance checking

Understanding compliance in multi-cloud

In the previous chapter, we already discussed some elements of compliance. Let us discuss this a bit further in this section before we start thinking of automating our cloud environments to ensure compliancy.

As we have learned in the previous chapter, compliance is a critical concern for organizations operating in regulated industries, such as healthcare, finance, and government. Meeting compliance requirements across multiple cloud providers poses unique challenges due to variations in regulatory frameworks, audit procedures, and data protection laws. However, automation can play a pivotal role in ensuring compliance in multi-cloud environments by streamlining and enhancing governance practices. We will go over some key topics when it comes to compliance.

The first step towards compliance is to identify the applicable regulatory requirements that govern our organization's operations. We must understand the specific compliance standards, industry regulations, and data protection laws that pertain to our business. Examples include HIPAA, GDPR, PCI-DSS, and ISO 27001. From these frameworks, we need to establish a comprehensive set of governance policies and controls that align with the identified regulatory requirements. These policies should cover areas such as data privacy, access controls, encryption, incident response, and audit trails. Now, these may vary between cloud providers, which makes it a bit more complex in multi-cloud settings. Be sure to clearly define and document the expectations for each cloud provider regarding compliance obligations.

It’s crucial to keep in mind the ever-changing nature of the regulatory landscape. A periodic review of this content might be necessary to ensure it stays up to date with any changes in regulations.

Automation can significantly streamline the enforcement of governance policies and controls across multiple cloud providers. The advantages of automation versus manual are significant. By implementing policy-driven automation tools, organizations can continuously evaluate the compliance status of their cloud resources and automatically remediate any non-compliant configurations. This reduces manual effort, minimizes human error, and ensures consistent adherence to compliance requirements. Let’s take a closer look at the main benefits of automation:

Automation allows for the execution of repetitive tasks and processes with minimal human intervention. It enables the cloud infrastructure to perform actions quickly and consistently, reducing the time required to complete tasks. In contrast, manual actions may involve human errors, delays, and inefficiencies.

Automation facilitates the scaling of cloud resources, such as virtual machines, storage, and networking. With automation, you can define rules and triggers that automatically adjust resources based on demand. Manual actions can be time-consuming and error-prone when it comes to scaling resources effectively.

Automation ensures consistent deployment and configuration of cloud resources, following predefined templates and best practices. By eliminating manual intervention, you reduce the risk of configuration errors or inconsistencies. Manual actions, on the other hand, rely on human interpretation and may lead to inconsistencies and non-standard configurations.

Automation helps in ensuring the high availability and reliability of cloud services. It enables proactive monitoring, automatic fault detection, and recovery mechanisms. Manual actions may lack the ability to respond quickly to issues and may require human intervention, resulting in potential downtime or reduced service availability.

Automation can contribute to cost optimization by enabling the efficient utilization of cloud resources. It allows for automated scheduling, resource allocation, and cost monitoring, leading to better cost management and optimization. Manual actions may overlook cost-saving opportunities or may not be as effective in resource optimization.

In the following sections, we will study more closely how we can automate governance policies and, with that ensure compliance. The benefits of automation are clear:

Automated monitoring solutions enable continuous compliance checking in multi-cloud environments. These tools can proactively scan the cloud infrastructure, identify potential compliance violations, and trigger alerts or automated remediation actions. By continuously monitoring the compliance posture of cloud resources, organizations can address issues promptly and maintain a robust compliance framework.

Automation simplifies the auditing and reporting processes required for compliance. Instead of manually gathering data and generating compliance reports, organizations can leverage automation tools to collect and consolidate information from different cloud providers. This enables faster and more accurate audits, as well as the ability to produce comprehensive compliance reports in a timely manner.

Automation facilitates effective change management and documentation, which are crucial for maintaining compliance in dynamic multi-cloud environments. By automating the provisioning and configuration of cloud resources, organizations can ensure that changes are tracked, documented, and auditable. Additionally, automation can enforce approval workflows and maintain a history of all modifications, simplifying compliance assessments and demonstrating compliance to auditors.

Automation provides the scalability and flexibility needed to adapt to evolving compliance requirements in multi-cloud settings. As regulations change or new compliance standards emerge, organizations can update their automated governance workflows and policies accordingly. Automation allows for swift and efficient adjustments, ensuring that compliance measures remain up to date across all cloud providers.

So, by harnessing automation in multi-cloud environments, organizations can achieve and maintain compliance with greater ease, efficiency, and accuracy. Automation not only streamlines governance practices but also enhances the organization's ability to respond to regulatory changes, mitigate risks, and demonstrate a strong commitment to compliance.

One topic that we have not addressed yet, in terms of compliance, is SBOM. The US government introduced the concept of SBOM through an executive order for the purpose of improving cybersecurity in the software supply chain. The specific executive order is Executive Order 14028, titled Improving the Nation's signed by President Joe Biden on May 12, 2021.

The primary motivations behind the introduction of SBOM as an act by the US government include:

Supply chain With the rise of sophisticated cyber threats, the software supply chain has become a potential target for attackers to infiltrate and compromise critical systems. SBOM aims to enhance the security of the software supply chain by increasing transparency and visibility into the components and dependencies used in software applications.

SBOM enables organizations to track and monitor the software components they use, including known vulnerabilities associated with those components. This helps in prioritizing patching and mitigation efforts, reducing the risk of exploitation through vulnerable components.

SBOM facilitates effective incident response and remediation processes. In the event of a security incident or vulnerability disclosure, having an accurate and up-to-date SBOM enables organizations to quickly identify and address affected software components. It streamlines the process of understanding the potential impact and applying necessary fixes or mitigations.

SBOM aims to establish accountability and compliance requirements for software vendors and suppliers. The government can set standards and guidelines regarding SBOM inclusion and accuracy, ensuring that software providers take responsibility for the components they incorporate into their products. This encourages a culture of transparency, accountability, and security throughout the software supply chain.

SBOM almost forces organizations to automate compliance checking, in this specific case, for all the software that we use. Be aware that this applies to all software, regardless of if it is used in the cloud or not. Figure 12.1 shows all the aspects of SBOM:

Figure 12.1: Components for SBOM

You might wonder why it is important to discuss SBOM at this place. The relationship between SBOM and compliance in the cloud is closely tied to transparency, security, and risk management. SBOM is a detailed inventory that lists all the components and dependencies of a software application or system. It provides comprehensive visibility into the software supply chain, including open-source libraries, third-party components, and their associated vulnerabilities.

When it comes to compliance in the cloud, SBOM plays a crucial role in several areas:

Vulnerability SBOM enables organizations to identify known vulnerabilities in their software components. By maintaining an up-to-date SBOM, organizations can assess the compliance of their cloud infrastructure with vulnerability management standards and guidelines. This allows them to proactively address vulnerabilities, apply patches, or implement compensating controls to reduce security risks and maintain compliance.

Open Many cloud applications and services rely on open-source software components. Complying with open-source licenses and obligations is essential to avoid legal and compliance issues. SBOM provides visibility into the open-source components used in the cloud infrastructure, allowing organizations to track licenses, verify compliance, and ensure adherence to license obligations.

Supply The cloud ecosystem involves multiple vendors, suppliers, and partners. Assessing the security posture of these entities is crucial to maintain compliance. SBOM helps in mapping the software supply chain, allowing organizations to identify potential security risks or vulnerabilities introduced by third-party components. This enables organizations to assess the compliance and security practices of their suppliers and make informed decisions about their cloud service providers.

Risk assessment and SBOM assists in risk assessments and compliance audits by providing a comprehensive overview of the software components and their associated risks. Auditors can review the SBOM to evaluate compliance with regulatory requirements, security standards, and industry best practices. The detailed information in the SBOM helps auditors assess the effectiveness of vulnerability management, security controls, and risk mitigation strategies.

Incident response and In the event of a security incident or data breach, having an up-to-date SBOM can significantly aid incident response and remediation efforts. It allows organizations to quickly identify and assess the impact of the incident on the affected software components. With this information, organizations can take appropriate actions to remediate the issue, update the SBOM, and demonstrate compliance with incident response requirements.

Overall, SBOM provides essential transparency and visibility into the software supply chain, which is crucial for maintaining compliance in the cloud. It helps organizations manage vulnerabilities, ensure open-source compliance, assess supply chain security, facilitate risk assessments and audits, and improve incident response capabilities.

Now that we have a deep understanding of what compliance is and what benefits we can gain by automation, we can really start implementing automation. But where and how to start? That is the topic of the next section.

Automating governance

If there is one thing that we have seen throughout this book, it is that traditional governance practices struggle to keep pace with the dynamic and complex nature of multi-cloud environments. However, through the application of automation, organizations can streamline governance processes, improve compliance, enhance security, and maximize operational efficiency. In this section, we will explore the challenges associated with governance in multi-cloud settings, discuss potential approaches to automate governance, and outlines best practices for successful implementation.

Governance in multi-cloud settings introduces several challenges that need to be addressed to ensure effective management and control:

Multi-cloud environments consist of different cloud providers, each with its own set of policies, APIs, and interfaces. Coordinating governance across these diverse, heterogenous platforms can be complex and time-consuming.

The distributed nature of multi-cloud deployments often leads to a lack of visibility into resource utilization, security posture, and compliance status. Without proper monitoring and control mechanisms, organizations face increased risks.

Organizations operating in industries with strict regulatory requirements face the challenge of ensuring compliance across multiple clouds, each with its own compliance frameworks and standards.

Maintaining consistent governance policies and controls across different cloud providers is difficult due to variations in service offerings, capabilities, and configurations.

To overcome the challenges associated with multi-cloud governance, we should implement best practices, starting with Infrastructure-as-Code Implementing IaC allows us to define and manage the cloud infrastructure using machine-readable configuration files. By codifying governance policies and controls, we can automate the deployment and management of cloud resources across multiple clouds. This is what we call a declarative way of working, and this includes policy-driven automation: leveraging policy-driven automation tools, organizations can define and enforce governance policies consistently across different cloud providers. These tools continuously evaluate cloud resources against predefined policies and automatically remediate any non-compliant configurations.

In the previous chapter, we stressed the importance of the single pane of glass view. This is absolutely crucial in enforcing and maintaining compliance. Implementing a centralized monitoring and management platform provides a holistic view of the multi-cloud environment. By aggregating data from various cloud providers, organizations can gain insights into resource utilization, security vulnerabilities, and compliance status, enabling proactive governance.

However, this is just the beginning. To successfully automate governance in multi-cloud settings, we must have a governance strategy that aligns with business objectives, regulatory requirements, and cloud provider capabilities. Identify key stakeholders and establish clear governance policies and controls. The steps are crucial in defining the governance strategy are:

Start by clearly defining your governance objectives. Identify the specific compliance, security, and operational requirements that need to be met across your multi-cloud environment. Consider factors such as data protection, access controls, resource allocation, cost management, and regulatory compliance.

Develop a set of governance policies that align with your objectives. These policies should outline rules, standards, and best practices for managing your multi-cloud environment. Consider aspects such as security controls, data classification, encryption standards, user access management, Service-Level Agreements and incident response protocols.

Roles and Determine the roles and responsibilities of individuals or teams responsible for implementing and enforcing governance policies. This may include cloud architects, security teams, compliance officers, and operations personnel. Clearly define the scope of each role and establish communication channels for collaboration.

Standardize governance practices, naming conventions, and resource configurations across different cloud providers. This ensures consistency and simplifies automation efforts. Utilize cloud-native automation tools and services provided by cloud providers. These tools are specifically

designed to integrate seamlessly with the underlying infrastructure and offer built-in governance capabilities.

Implement continuous monitoring and compliance checking mechanisms to proactively identify and remediate any violations. Leverage automation to trigger alerts, perform remediation actions, and maintain compliance posture.

Conduct regular audits and reviews to assess the effectiveness of governance automation. Evaluate the accuracy of policies, identify areas for improvement, and adjust automation workflows as necessary.

This is a full cycle, as shown in Figure From the review, we might need to adjust our policies and even reconsider our strategy:

Figure 12.2: Governance cycle

A final or additional step might be continuous learning and adaptation. With cloud technologies evolving rapidly, it's crucial for organizations to keep learning and adapting their strategies to stay ahead.

Can we automate all of this? Not everything but enforcing that policies are followed, and roles and actions are monitored can definitively be automated. By leveraging approaches such as Infrastructure as Code, policy-driven automation, and centralized monitoring, organizations can enhance governance practices, improve compliance, and optimize operational efficiency. We can use the principles of Robotic Process Automation to do so, as we will learn in the next section.

Using RPA for automating compliance

The rapid adoption of cloud technology has revolutionized the way businesses operate, providing flexibility, scalability, and cost-effectiveness. However, as organizations transition their operations to the cloud, ensuring compliance with industry regulations and internal policies becomes increasingly complex. Manual compliance monitoring processes often fall short of keeping up with the dynamic nature of cloud environments. To address this challenge, many organizations are turning to RPA as a powerful tool to automate compliance monitoring in the cloud.

You might be wondering how we can use RPA for automating compliance in multi-cloud. Compliance monitoring in the cloud involves tracking, validating, and reporting adherence to regulations, policies, and security controls. It encompasses various aspects such as data privacy, access controls, encryption, and auditing. Manual monitoring can be time-consuming, error-prone, and costly. RPA can help overcome these challenges by automating routine tasks, data collection, analysis, and reporting.

What would be the benefits of using RPA in compliance monitoring? To start with, it will increase accuracy. RPA bots perform repetitive tasks with precision, reducing the risk of human error in compliance monitoring. They can gather data from multiple cloud platforms, databases, and applications, ensuring accurate and consistent monitoring across the organization.

RPA eliminates the need for manual data collection, validation, and analysis, enabling faster and more efficient compliance monitoring. Bots can work 24/7, ensuring continuous monitoring without the limitations of human resources, thereby improving overall efficiency. This efficiency is increased through scalability: RPA allows organizations to scale compliance monitoring efforts easily. As the cloud environment grows, RPA bots can be deployed to handle increased data volume, monitor additional systems, and adapt to evolving compliance requirements without significant effort.

Of course, there is a cost element as well. By automating compliance monitoring, organizations can reduce labor costs associated with manual efforts. RPA bots can perform monitoring tasks at a fraction of the time and cost required by human resources, allowing employees to focus on higher-value activities.

RPA bots can automatically extract and analyze audit logs from various cloud platforms, detecting unauthorized access attempts, unusual activities, and potential compliance violations. Alerts can be generated for further investigation or corrective actions. Let us look at an example of an RPA bot collecting logs in Azure using Logic Apps. Azure Logic Apps, in this case, is used to build an RPA bot that automatically extracts log files from Azure workloads. Logic Apps provide a visual designer and a wide range of connectors to automate workflows and integrate with various services, including Azure. Of course, there are other ways to do this; Azure Logic Apps is just one option for building RPA bots in Azure. We may also consider using other tools like Azure Automation, Power Automate, or even Azure Functions, depending on requirements and preferences.

In this very basic example, we use a Logic App workflow that extracts log files from Azure Blob Storage and saves them to a designated location:

We create a new Logic App in the Azure portal.

Next, we add a trigger to start the workflow. In this case, use the When a blob is added or modified (properties only) trigger from the Azure Blob Storage connector. Select the storage account and container where the log files are stored.

Add an action to retrieve the log file contents. Use the Get blob content action from the Azure Blob Storage connector. Provide the storage account and container information, as well as the blob path from the trigger output.

Add any necessary actions to analyze or process the log file contents. We can use various connectors and actions based on your requirements. For example, we could use the Extract text from PDF action if the logs are in PDF format or perform custom analysis using Azure Cognitive Services or Azure Functions.

Finally, we add an action to save the log file to a designated location. Use the Create file action from the Azure Blob Storage connector or choose a different connector based on your desired destination. Provide the necessary information, such as the destination folder and file name.

Once we have defined the Logic App workflow, it will automatically trigger whenever a new log file is added or modified in the specified Azure Blob Storage container. The bot will then retrieve the log file, perform any required analysis, and save it to the designated location.

In code, using the Python SDK, it will look like this:

```
from azure.identity import DefaultAzureCredential

from azure.storage.blob import BlobServiceClient

# Configure the storage account connection string and container name

storage_connection_string = ""

container_name = ""

# Initialize the BlobServiceClient using the connection string

blob_service_client =
BlobServiceClient.from_connection_string(storage_connection_string)

# Get a list of blobs in the container

container_client =
blob_service_client.get_container_client(container_name)

blobs = container_client.list_blobs()
```

```
# Iterate over the blobs and download them

for blob in blobs:

    # Download the blob contents



    blob_client = container_client.get_blob_client(blob.name)

    blob_content = blob_client.download_blob().readall()

    # Perform analysis or processing on the log file contents

    # For example, you can use regular expressions, data parsing, or any
required analysis logic

    # Save the log file to a designated location

    save_location = ""

    with open(f"{save_location}/{blob.name}", "wb") as file:

        file.write(blob_content)
```

In this code, we need to replace with the connection string of our Azure Storage account and with the name of the container where the log files are stored. Provide the with the desired folder where you want to save the log

files. Moreover, we have to make sure that we have the required dependencies installed. It is recommended to install them using pip.

In AWS, we can use AWS Lambda and AWS SDKs to automate the extraction of log files from AWS workloads, using Python and AWS SDK for Python (Boto3). In GCP, the best options are Google Cloud Functions and Google Cloud Storage Client Libraries to automate log file extraction.

This is just a basic example of the usage of RPA bots. In this section, we discussed the native technology of cloud providers, but a lot of organizations might choose to use third-party tools to automate compliance. Examples are Cloud Custodian, Dome9, or Prisma Cloud. The primary functions of these tools include:

Cloud security and compliance These tools continuously monitor cloud infrastructure, services, and resources to identify security vulnerabilities, misconfigurations, and compliance violations. They provide real-time visibility into the security posture of cloud environments, helping organizations proactively address risks and ensure compliance with industry standards and regulatory requirements.

Automated policy Tools like Cloud Custodian, Dome9, and Prisma Cloud allow organizations to define and enforce security policies and best practices across their cloud deployments. They enable the automation of policy enforcement, ensuring that cloud resources adhere to specific security guidelines and configurations. This helps prevent security incidents and maintain a secure cloud environment.

Threat detection and incident These tools employ advanced threat detection techniques and machine learning algorithms to identify suspicious activities, indicators of compromise, and potential security breaches within cloud environments. They provide alerts, notifications, and actionable insights to security teams, enabling them to respond quickly to security incidents and mitigate threats.

Compliance reporting and Tools like Cloud Custodian, Dome9, and Prisma Cloud assist organizations in generating compliance reports and conducting audits. They provide predefined compliance frameworks and standards, enabling organizations to assess their cloud environments against specific regulatory requirements. These tools simplify the process of preparing for audits and demonstrate compliance with regulatory bodies.

Security automation and These tools offer automation capabilities to streamline security tasks and remediation processes. They can automatically fix misconfigurations, apply security patches, and remediate security issues identified within cloud environments. This helps organizations improve operational efficiency and reduce the manual effort required for security management.

As we can see from this brief overview of the capabilities that these third-party tools have, we can utilize RPA for a lot more. For instance, to run user access reviews: RPA can automate the process of reviewing user access privileges in the cloud. Bots can collect user access data, compare it against defined roles and policies, identify any discrepancies, and generate reports for management and compliance teams. Overall, RPA is a good practice to streamline the generation of compliance reports by

automatically collecting data from different sources, formatting it, and generating comprehensive reports. This reduces the time and effort required for manual report generation and ensures accuracy and consistency. As we have learned, consistency is key to compliance. With RPA bots, we can enforce compliance policies by monitoring cloud resources and configurations. They can identify deviations from defined policies, trigger remedial actions, or generate alerts for the IT or compliance teams to take appropriate measures.

We have to take some important considerations into account. First and most important of all, we have to make sure – as with anything – that security measures are in place to protect RPA bots and the data they handle. Access controls, encryption, and secure communication protocols should be implemented to prevent unauthorized access or data breaches. Next, RPA bots should be regularly updated to accommodate changes in compliance requirements, cloud platforms, and regulations. Maintenance tasks, such as patching and monitoring, should be performed to ensure the bots' effectiveness and reliability.

While RPA can automate many compliance monitoring tasks, human oversight and governance remain crucial. Organizations should establish clear guidelines, policies, and processes for managing RPA bots, reviewing their output, and addressing exceptions or complex scenarios.

Automating compliance monitoring in the cloud using RPA offers numerous benefits to organizations, including enhanced accuracy, improved efficiency, scalability, and cost reduction. It is essential to carefully plan and implement RPA solutions while considering security, maintenance, and human oversight aspects. By embracing RPA in compliance monitoring, organizations can achieve higher levels of

compliance and governance in their cloud environments. In the final section, we will learn how AI and machine learning can be a great aid in this.

The next step: using AI for compliance checking

No modern book about cloud technology can do without discussing how AI could help us in managing our environments in the cloud. So, in the final section of this book, we will look at the possibilities that AI will bring to multi-cloud management, especially in ensuring compliance. AI can certainly play a significant role in managing and controlling compliance in multi-cloud environments by providing automated and intelligent solutions.

Most chapters in this book ended with a section about monitoring. With multi-cloud environments can become very complex, and we will need solid monitoring to know the actual status of our workloads and what possible consequences there might be if one of our workloads, connections, databases, or APIs fails. AI algorithms can continuously monitor various data sources, including logs, configurations, and user activities across multiple cloud platforms. By analyzing this data, AI can identify any non-compliant actions, policy violations, or security breaches in real time. It can generate alerts and notifications for immediate action, enabling organizations to proactively address compliance issues.

Before we have a closer look at AI, it's good to explore emerging trends in cloud compliance.

We can expect to see an increase in regulations specific to cloud computing, addressing data privacy, security, cross-border data transfers,

and vendor accountability. Compliance frameworks, such as GDPR and CCPA, are likely to evolve and become more cloud-centric.

Data residency and sovereignty concerns are gaining prominence as organizations grapple with cross-border data transfers and conflicting jurisdictional requirements. Compliance regulations may address the storage and processing of data within specific geographic boundaries, leading to the development of cloud solutions that offer localized data centers or mechanisms to control data residency.

With the increased reliance on cloud service providers, there is a growing demand for transparency and accountability from cloud service providers regarding their compliance practices. Organizations are seeking assurance that CSPs adhere to stringent compliance requirements and follow best practices for data security, privacy, and governance. We can anticipate more rigorous audits, certifications, and third-party assessments of CSPs.

As organizations adopt multi-cloud and hybrid cloud strategies, compliance challenges become more complex. Compliance frameworks will need to address the unique considerations associated with managing compliance across multiple cloud providers and on-premises infrastructure. Standards for interoperability, data portability, and consistent compliance controls are expected to evolve.

Of course, automation will play a crucial role in ensuring cloud compliance. Organizations will increasingly leverage technologies such as IaC and Continuous Compliance to automate the deployment, configuration, and enforcement of compliance controls. Integrating compliance into DevOps and adopting DevSecOps practices will become

essential for maintaining compliance in highly dynamic cloud environments.

As cloud-native architectures, such as containers and serverless computing, gain wider adoption, compliance frameworks will need to adapt accordingly. Regulations will likely include specific guidelines for securing cloud-native applications and microservices architectures, addressing container security, secrets management, and secure development practices.

The need for real-time compliance monitoring and continuous assessment will grow. Tools and technologies that provide real-time visibility into compliance posture, automated scanning for misconfigurations, and monitoring of cloud environments for policy violations will become increasingly important.

As AI and machine learning applications become more prevalent in the cloud, compliance frameworks will evolve to address the ethical and responsible use of AI. Regulations may include guidelines on transparency, explainability, fairness, and accountability in AI algorithms and models deployed in the cloud.

AI can assist in enforcing compliance policies by automatically evaluating cloud resources and configurations against predefined compliance rules and standards. It can identify any deviations and suggest appropriate remedial actions to bring the environment back into compliance. This will help us in maintaining a consistent and secure configuration across their multi-cloud infrastructure. AI can also perform automated risk assessments by analyzing the security posture of different cloud providers and identifying potential vulnerabilities or weaknesses in the

infrastructure. It can evaluate factors such as encryption practices, access controls, data privacy measures, and regulatory compliance. This helps organizations identify areas of improvement and ensure adherence to compliance requirements.

However, the best part about AI is that it learns, and by doing so, it can become predictive. AI can leverage historical compliance data and patterns to perform predictive analytics. By analyzing past compliance violations and incidents, it can identify potential future risks and provide recommendations to mitigate them. This proactive approach allows organizations to anticipate compliance challenges and take preventive measures in advance.

Most monitoring systems in the cloud already work with AI. Examples are Azure's Sentinel, AWS GuardDuty, and Google Cloud Security Command Center They all use AI and machine learning techniques to enhance security operations, detect and respond to threats, and provide actionable insights:

Threat Employing AI algorithms to analyze large volumes of security event data in real time. Sentinel, GuardDuty, and Cloud SCC can automatically identify patterns, anomalies, and suspicious activities that may indicate a potential security breach. AI-powered threat detection helps in discovering both known and unknown threats, including sophisticated and evolving attack techniques.

Behavioral By applying machine learning, these systems create baselines of normal behavior for users, devices, and applications in the environment. They then compare ongoing activities against these baselines to identify deviations that could indicate malicious behavior. AI

algorithms help in detecting unusual patterns, privileged access abuse, insider threats, and other anomalous activities. This can be very simple: if someone, for example, logs in from London at 10 AM, this person cannot log in to systems from the other side of the world at 10.05. Systems will detect and flag this.

Anomaly Sentinel, GuardDuty, and Cloud SCC utilize AI-based anomaly detection techniques to identify abnormal events or deviations from expected behavior. It can detect anomalies in user behavior, network traffic, application usage, system logs, and more. By continuously learning from data patterns, these systems improve their anomaly detection capabilities over time, making them more accurate in recognizing potential threats.

Automated threat With AI integration, these systems enable automated threat response actions. It can trigger automatic remediation actions or orchestrate security workflows based on predefined playbooks. For example, if a specific threat is detected, Sentinel or one of the other systems can automatically block an IP address, quarantine a compromised device, or initiate incident response procedures.

Threat intelligence and This one specifically applies to Azure Sentinel, which integrates with threat intelligence feeds, both from Microsoft and third-party sources. AI algorithms analyze and correlate this information with real-time security events to provide context and prioritize threats effectively. It helps security analysts in threat-hunting activities by surfacing relevant indicators of compromise and supporting proactive detection of emerging threats.

Data analysis and AI-powered data analysis and visualization techniques provide meaningful insights into security events. It can automatically categorize and tag security events, perform entity extraction, and identify relationships between various entities. This helps analysts quickly identify important security events and investigate incidents more efficiently.

Sentinel, GuardDuty, and Cloud SCC certainly share capabilities, but keep in mind that each service has its own unique features and integrations specific to their respective cloud platforms.

Overall, AI brings automation, scalability, and intelligence to compliance management in multi-cloud environments. It enhances the ability to monitor, enforce, and report compliance, thereby reducing human effort, improving accuracy, and ensuring a higher level of security and adherence to regulatory requirements.

This brings us to the end of this chapter and this book. By now, you should have a good understanding of managing multi-cloud environments, the challenges that multi-cloud brings along, and how you could overcome these challenges.

Conclusion

The most important lesson from this chapter is that compliance is a critical concern for organizations operating in almost every industry, but especially in healthcare, finance, and government. Meeting compliance requirements across multiple cloud providers poses unique challenges due to variations in regulatory frameworks, audit procedures, and data protection laws that vary from continent to continent and even from country to country. Compliance frameworks will offer guidance in applying the right policies. Next, we learned how we can use automation to enforce and monitor these policies.

We learned that Infrastructure as Code policy-driven automation, and centralized monitoring organizations can help us to governance practices and improve compliance and how we can use the principles of RPA to do so. We studied examples of automated bots in Azure, AWS, and GCP.

In the final section, we discussed to possibilities of integrating AI and machine learning to help us in managing compliance by providing automated and intelligent solutions. Most major cloud providers already offer sophisticated solutions, such as Azure Sentinel, AWS GuardDuty, and GCP Security Command Center, that all utilize AI to monitor workloads on their platforms.

Points to remember

Although this chapter focused on automation, the most important first step in managing compliance is identifying the applicable regulatory requirements that govern our organization's operations. We must understand the specific compliance standards, industry regulations, and data protection laws that pertain to our business. Examples include HIPAA, GDPR, PCI-DSS, and ISO 27001.

Since monitoring compliance can become extremely complex, automation becomes key. By implementing policy-driven automation tools, organizations can continuously evaluate the compliance status of their cloud resources and automatically remediate any non-compliant configurations.

SBOM is or will be, a mandatory document in many industries.

Keep in mind that not everything can be automated (yet). However, enforcing that policies are followed and roles and actions are monitored can definitively be automated.

The principles of RPA are good guidance in automating security and compliance. RPA can be used for tracking, validating, and reporting adherence to regulations, policies, and security controls. RPA can help by automating routine tasks, data collection, analysis, and reporting.

On top of RPA, AI and machine learning are recent developments that are great enhancements in monitoring, including actioning of events, for instance, in threat detection. AI can trigger automatic remediation actions or orchestrate security workflows based on predefined playbooks. In this case, we will have to work with both RPA and AI. Major cloud providers offer already sophisticated tools to enable this.

Questions

What are GDPR, HIPAA, and PCI-DSS?

We discussed six stages in the governance cycle. Name these six stages.

What are the security suites of cloud providers that offer AI capabilities?

Name three aspects of SBOM.

Answers

Compliance frameworks.

Objectives, policies, roles and responsibilities, standardization, monitoring, review.

Azure Sentinel, AWS GuardDuty, GCP Security Command Center.

Software components, compliance requirements, libraries, author name, supplier name, open-source components, version string, licenses, dependencies

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

A

access control

monitoring

Access Control Lists (ACL) 26

access management

challenges 212

cloud tools, using 214

exploring 210

managed identities 211

Active Directory (AD) 64

AI for compliance check

anomaly detection 275

automated threat response 275

behavioral analytics 275

data analysis and visualization 275

threat detection 274

threat intelligence and hunting 275

using

Alibaba Cloud

about 191

data quality 137

options 87

secrets and keys, managing

Alibaba Cloud framework

about

examples 14

reference link 15

Alibaba Cloud Interconnect 108

Alibaba Cloud OpenAPI 108

Alibaba Cloud Service Mesh 114

Alibaba Cloud’s KMS

reference link 226

Alibaba cloud storage options

ApsaraDB 51

block storage service (BSS) 51

elastic file system (EFS) 51

object storage service (OSS) 51

table store 51

Alibaba Container Service

for Kubernetes (ACK) 84

Amazon Machine Image (AMI) 85

Amazon QuickSight 134

Amazon Redshift 134

Amazon SageMaker Data Wrangler 134

Amazon Web Services (AWS)

about 169

data quality 134

methods 86

secrets and keys, managing 222

anomaly detection 252

antipatterns

about 10

inaccurate out-of-the-box
security assumptions 10

IT as cloud provider 10

Apache Kafka 199

Apache Mesos 172

Apache OpenWhisk 199

API Management (APIM) 144

Application Delivery Network (ADN) 31

application-level load balancing 30

Application Performance
Monitoring (APM) 246

Application Programming
Interface (API) 97

Aria components

reference link 83

Artifact generation

packaging 149

versioning 149

artificial intelligence (AI) 133

automation

benefits

autonomous systems (ASes) 39

autoscale 75

autoscaling

using 79

AWS App Mesh 107

AWS CAF

about

reference link 8

AWS CloudTrail 247

AWS DataSync 53

AWS Direct Connect 107

AWS Glue DataBrew 134

AWS Lambda 190

AWS outposts 82

AWS Secrets Manager

reference link 223

AWS Storage Gateway 56

Azure

data quality 135

options 86

secrets and keys, managing

Azure Active Directory
(Azure AD) 241

Azure API Management 107

Azure Arc 82

Azure CAF 10

Azure Container Instances (ACI) 165

Azure Container Registry (ACR) 165

Azure Data Box

about 52

formats 126

Azure Databricks 136

Azure Data Factory 135

Azure Data Lake Storage 135

Azure Function 191

Azure Key Vault

reference link 224

Azure Kubernetes Service
(AKS) 165

Azure Load Balancer 31

Azure Monitor 247

Azure Stack 82

Azure StorSimple 56

Azure Synapse Analytics 135

Azure Virtual Desktop (AVD) 9

B

Backend as a Service (BaaS) 188

BigQuery 136

Border Gateway Protocol (BGP) 39

business agility 15

business metrics

measuring 19

C

Californian Consumer Privacy Act
(CCPA) 124

Capital Expenditure (CAPEX) 75

Center for Internet Security (CIS) 237

centralized policy management 249

choke points or functions 203

CIS guidelines for Azure

implementing

Client-Side Encryption (CSE) 138

Cloud Access Security Broker (CASB)

about 210

implementing 215

key points 215

Cloud Adoption Framework (CAF)

about 35

adopt 6

comparing

govern 6

operate 6

plan 5

prepare 6

reference link 13

strategy 4

cloud brokers

benefits 99

limitations 100

Cloud Centre of Excellence (CCoE) 6

Cloud Dataflow 136

cloud data loss prevention 136

Cloud Data Management Interface (CDMI) 97

Cloud Dataprep 136

cloud gateways

benefits 99

limitations 100

cloud-native

organizing, with DevOps 153

cloud-native concepts

about

benefits 189

containerization 142

micro-services 142

serverless 142

cloud network operations

cloud performance

geographic location 76

instance types 76

load balancing 76

network latency 75

resource allocation 75

scalability 75

software optimization 76

storage performance 76

third-party services 76

workloads performance
metrics, monitoring 76

cloud provider organization

pitfalls, avoiding in secrets
and keys management 227

Cloud Security Alliance (CSA) security guidance 236

Cloud Security Command Center
(Cloud SCC) 274

cloud security frameworks

Center for Internet Security (CIS) 237

Cloud Security Alliance (CSA) security guidance 236

Federal Risk and Authorization Management Program (FedRAMP) 237

General Data Protection
Regulation (GDPR) 237

ISO/IEC 27001 237

National Institute of Standards
and Technology (NIST)
cybersecurity framework 237

Payment Card Industry Data Security Standard (PCI DSS) 237

working with

cloud security monitoring 249

Cloud Security Posture Management
(CSPM)

cloud service providers (CSPs) 273

cloud tools

used, for access management 214

command-line interface (CLI) 87

compliance and audit support 249

compute assets

deploying 85

managing 85

compute models

comparing, in public clouds

key considerations

rightsizing, in autoscaling 78

rightsizing, in deployment 78

Configuration Management Database
(CMDB) 19

containerization 142

container log management 181

container management

exploring, with single pane
of glass view 173

container monitoring 181

container platform

best practices 174

security, managing

containers

cons 74

data security 80

deploying

developing

developing, with industry
standards 167

isolation 80

pros 74

provisioning, with industry
standards

Container Service for Kubernetes (ACK) 165

container technology

about 164

container services, from cloud
providers 165

key consideration 163

pitfalls 164

risks 164

Content Delivery Networks (CDNs) 76

Continuous Delivery (CD)

about 147

additional testing 149

approval and release 149

deployment 149

User Acceptance Testing (UAT) 149

Continuous Deployment (CD)

fully automated release 149

Continuous Integration and Continuous Delivery (CI/CD) pipeline

about 147

approval and release 148

artifact generation 148

build and test 148

code integration 147

deployment 148

deployment environment, testing 148

used, for serverless deployment 196

Continuous Integration (CI)

about 147

automated tests 149

code analysis 149

code build 148

compilation 148

cross-site scripting (XSS) 176

Customer-Managed Keys (CMEK) 138

D

data

securing 138

data access

auditing 65

managing 65

monitoring 65

data access controls 65

Datadog 246

data encryption 65

data governance

managing

data integrity 123

Data Loss Detection (DLD) 136

Data Loss Prevention (DLP) 214

data migration

planning

data portability

about 101

examples 102

data quality

dimensions 133

improving 133

in Alibaba Cloud 137

in AWS 134

in Azure 135

in GCP 136

Data Quality Services (DQS) 135

data strategy

defining

days payables outstanding (DPO) 18

Demilitarized Zone (DMZ) 38

DevOps

used, for organizing cloud-native 146

DevOps capabilities

API Management (APIM) 144

apps and database migration 145

cloud-based development 144

containerization 144

DevSecOps 145

Document As Code 145

event-driven architecture 145

full stack and grand stack 145

PaaS deployment 145

SaaS integration 145

DevOps for cloud-native architecture

best practices 151

DevSecOps 175

DevSecOps platform

cultural shift to DevOps 147

infrastructure automation 147

Kubernetes capabilities across clouds 146

open platform 146

platform and apps monitoring 147

service mesh deployment 147

DevSecOps risk strategies

access control and privilege management 146

continuous security testing 146

incident response and recovery 146

secure coding practices 146

secure configuration management 146

Direct Attached Storage (DAS) 46

direct connections

about 127

benefits 40

setting up 40

Distributed Denial of Service (DDoS) attacks 242

Distributed File System (DFS) 48

DNS load balancing 31

Docker Swarm 172

Domain Name Services (DNS) protection 37

Dynamic Application Security Testing (DAST) 176

Dynatrace 246

E

e-commerce application 193

Edge

deployment concepts

Elastic Block Store (EBS) 55

Elastic Compute Cloud (EC2) 72

Elastic Compute Service (ECS) 72

Elastic Container Instance (ECI) 165

Elastic Container Registry (ECR) 165

Elastic Container Service (ECS) 165

Elastic File System (EFS) 50

Elastic Kubernetes Services
(EKS) 165

Elastic Load Balancing 31

Elastic Observability (ELK Stack) 247

Enhanced Interior Gateway
Routing Protocol (EIGRP) 27

Environmental, Social, and Governance (ESG) 8

Event-driven architecture (EDA)

about 192

e-commerce application 193

financial services 193

healthcare 193

IoT applications 193

F

Federal Risk and Authorization Management Program (FedRAMP) 237

financial services 193

Firewall as a Service 37

Function as a Service (FaaS) 188

G

GCP Cloud Interconnect 108

GCP Cloud Logging and
Cloud Monitoring 247

GCP endpoints 108

GCP framework

about 13

lead 11

learn 11

scale 12

secure 12

GCP KMS

reference link 226

GCP storage options

cloud bigtable 50

cloud datastore 51

cloud filestore 50

cloud SQL 51

cloud storage 50

General Data Protection Regulation (GDPR) 237

Google Anthos 83

Google Cloud Functions 191

Google Cloud Operations 247

Google Cloud Platform (GCP)

about 247

data quality 136

deployment options 87

secrets and keys, managing

Google Cloud Transfer Service 53

Google Container Registry (GCR) 168

Google Kubernetes Engine
(GKE) 165

governance

automating

granule policies 203

GraphQL 105

GraphQL, React, Apollo,
and Neo4j Database (GRAND) 145

H

Hard Disk Drives (HDDs) 76

healthcare 193

Health Insurance Portability
and Accountability Act (HIPAA) 217

High Performance Computing (HPC) 77

hybrid cloud storage solution

AWS Storage Gateway 56

Azure StorSimple 56

Google Anthos 56

I

Identity and Access Management (IAM) 249

Identity as a Service (IDaaS)

about 213

implementing 215

key points 215

Indicators of Compromise (IOC) 254

Information Security Management System (ISMS) 237

Infrastructure as a Service (IaaS) 3

Infrastructure as Code (IaC) 265

infrastructure management

automating

Input/Output operations
Per Second (IOPS) 59

Interactive Application
Security Testing (IAST) 176

Intermediate System to Intermediate
System (IS-IS) 27

Internet Business Exchange (IBX) 98

Internet of Things (IoT) 193

Internet Service Providers (ISPs) 37

Internet Small Computer Systems
Interface (iSCSI) 46

interoperability

approach 98

challenges 97

defining 100

requisites 103

versus portability 104

interoperability solutions

creating, in hybrid cloud

creating, in public cloud

IoT applications 193

ISO/IEC 27001 237

K

Key Management Service (KMS) 224

Key Performance Indicators (KPIs) 178

Kubeless 200

Kubernetes 172

Kubernetes Container Service (ACK) 143

Kubernetes, deployment strategies

blue/green deployment 151

canary releases 151

rolling updates 150

Kubernetes Federation 173

L

Lambda functions

usage 190

load balancing

about 31

application-level load balancing 30

DNS load balancing 31

network-level load balancing 30

Logical Unit Number (LUN) 46

logs

analyzing

collecting

M

---

machine learning (ML) 137

managed identity 211

Message Queuing Telemetry Transport (MQTT) 198

metrics example

Bandwidth 59

CPU usage 60

Input/Output operations Per Second (IOPS) 59

Latency 59

Queue depth 59

Throughput 59

microservices

about

releases, managing

microservices, architectural artifacts

API development and integration 154

application modernization 154

legacy migration 153

Minimal Viable Products (MVP) 9

Mitre ATT&CK

reference link 254

monitoring

best practices 20

monolithic approach

reference link 164

monorepo 203

Multi-access Edge Compute (MEC) 37

multi-cloud

compliance 262

monitoring 18

multi-cloud business challenges

exploring

multi-cloud environment

managing, with serverless framework 197

multi-cloud networks

direct connections, setting up

managing 38

VPN, setting up

multi-cloud storage assets

data, protecting from data loss 54

managing 52

third-party storage products, using

multi-factor authentication
(MFA) 241

Multi-Lateral Peering Exchange (MPLE) 37

multiple cloud platforms APIs

API compatibility 106

authentication and authorization 106

documentation and support 107

error handling 106

error logging 106

feature parity 106

governance and compliance 107

performance and latency 106

versioning and compatibility 107

Multi-Protocol Label Switching (MPLS) 35

N

National Institute of Standards and Technology (NIST) cybersecurity framework 237

NetApp XCP 55

Network as a Service (NaaS) 42

Network Attached Storage (NAS) 49

network building blocks

about 24

load balancing 31

routing 27

virtual firewalls

virtual switching

Network File System (NFS) 50

network interface cards (NIC) 24

network-level load balancing 30

Network Security Groups (NSG) 242

New Relic 246

Next Generation Firewall (NGFW) 37

O

Object Storage Service (OSS) 138

Open Compute Project (OCP)

about 114

working with 116

Open Container Initiative (OCI)

about 166

image specification 166

runtime specification 166

Open Functions as a Service (OpenFaaS)

about 198

reference link 199

Open Shortest Path First (OSPF) 27

Open Systems Interconnection (OSI) 27

Operational Expenditure (OPEX) 75

operational technology (OT) 5

P

Payment Card Industry Data
Security Standard (PCI DSS) 237

penetration (Pen) testing 176

Personally Identifiable
Information (PII) 254

Platform as a Service (PaaS) 86

Point of Presence (PoP) 37

portability

versus interoperability 104

Privileged Access Management (PAM)

about 216

solutions 218

tools 217

working with 217

Privileged Behavior Analytics (PBA)

about 219

components 220

Privileged Identity Management (PIM) 241

Privileged Session Management (PSM)

about 219

components 219

public cloud on premises propositions

AWS outposts 82

Azure Arc 82

Azure Stack 82

exploring 81

Google Anthos 83

VMWare, extending 83

public clouds

compute models, comparing

R

Rancher 173

Recovery Point Objective (RPO) 97

Recovery Time Objective (RTO) 97

Relational Database Service (RDS) 50

Remote Procedure Call (RCP) 105

Representational State Transfer (REST) 105

Resource Access Management (RAM) 226

Resource Orchestration Service (ROS) 143

Robotic Process Automation (RPA)

about 267

using, for automating compliance

role-based access control (RBAC)

about 210

defining 227

implementing 228

managing 229

routing 27

Routing Information Base (RIB) 27

Routing Information Protocol (RIP) 27

S

scaling 15

secrets

creating 221

storing 221

Secure Access Service Edge (SASE) 35

Secure File Transfer Protocol (SFTP) 127

Secure Socket Layer (SSL) 39

Secure Web Gateway 37

security

managing, through one lens

Security Information and Event Management (SIEM) 244

Security Orchestration, Automation, and Response (SOAR) 245

security tools

selecting

security trends

tracking

serverless

about 142

data security 80

isolation 80

serverless architecture 188

serverless computing 188

serverless concept

about

Backend as a Service (BaaS) 188

Function as a Service (FaaS) 188

serverless architecture 188

web applications 188

serverless deployment

Continuous Integration and Continuous Delivery (CI/CD) pipeline, using 196

serverless framework

about 198

best practices

used, for managing multi-cloud environments 197

serverless functions

developing 196

monitoring

provisioning 196

serverless options

cons 74

pros 74

Server Load Balancing (SLB) 31

Server Message Blocks (SMB) 48

Server-Side Encryption (SSE) 138

Service-Level Agreements (SLA) 266

service mesh 112

service mesh functionality

failure recovery 113

load balancing 113

metrics 113

monitoring 113

service discovery 113

service mesh solutions

istio 114

linkerd 114

Simple Notification Service (SNS) 190

Simple Object Access Protocol (SOAP) 105

Simple Queue Service (SQS) 198

Simple Storage Service 50

single sign-on (SSO) 213

Software as a Service (SaaS) 29

Software Bill of Materials (SBOM)

about 259

benefits 263

compliance 264

Software Component Analysis (SCA) 175

software-defined data center (SDDC) 32

software defined local area

networks (SD-LAN)

about 35

deployment concept 36

deployment concepts

Software-Defined Networks (SDN)

about 32

concept 34

software defined wide area
networks (SD-WAN) 35

Solid-State Drives (SSDs) 76

SSE with customer-managed
keys (SSE-CMK) 138

Static Application Security
Testing (SAST) 176

Storage Area Network (SAN) 46

Storage Lifecycle Management (SLM) 57

storage lifecycles

managing

storage scaling

automating

storage tiering

automating 61

managing

storage virtualization types

about

array-based 48

fiber channel 48

file system 48

host-based 48

OS-level 48

public cloud, storage concepts 49

Sysdig Secure 246

T

third-party storage products

AWS 54

Azure 55

GCP 55

using 54

threat intelligence integration 249

Transport Layer Security (TLS) 127

U

uniform resource locator (URL) 28

user-defined routing (UDR) 27

V

virtual firewalls 29

virtual interfaces (VIF) 41

virtual local area networks (vLAN) 25

virtual machines (VMs)

about 72

cons 73

data security 80

isolation 80

pros 73

Virtual Private Gateway (VPG) 39

Virtual Private Network (VPN) 244

virtual SAN (vSAN) 47

virtual switching

VMWare

extending 83

W

web application firewall (WAF) 242

web applications 188

Well-Architected Framework (WAF) 35

Windows Server Software-Defined (WSSD) 81