Tampere University

Samu Salko

BUILDING AN INFORMATION SECURE CLOUD ENVIRONMENT IN AZURE WITH AZURE BICEP

Case: ECB product from Evitec

Master of Science thesis
Faculty of Information Technology and Communication Sciences
Examiners: Kari Systä, David Hästbacka
February 2024

ABSTRACT

Samu Salko : Building an information secure cloud environment in Azure with Azure Bicep
Master of Science thesis
Tampere University
Master's Programme in Information Technology
February 2024

Cloud environments consist of various cloud services and cloud-native resources, which are partly managed by the cloud service provider. These cloud services differ from each other in level of control, customer responsibility, and costs. For hosting a SaaS product, the SaaS vendor needs to go through the tasks of identifying, combining and deploying the services and resources to create a cloud infrastructure, that fulfils the application's requirements and is information secure.

Designing and building an information-secure cloud environment requires acknowledging the security aspects related to cloud infrastructure. This master's thesis goes through a case study about Evitec's product hosted currently in AWS, aiming to create a suitable and information-secure environment for it in Azure. The transfer considers identifying the key services on the current cloud platform and identifying the corresponding services in Azure. Given the application processes personal data, it is crucial to take infrastructure security into account. This requires researching the security aspects of the cloud services and resources in question. The SaaS vendor is responsible for configuring the resource-specific configurations to be information secure.

The information security of the case study's environment is based on the theory about security in cloud environments and the specifications from Azure-native services and resources used. The evaluation of information security for the case study's environment was done with Microsoft Cloud Security Benchmark (MCSB), a list of security controls for Azure and AWS. MCSB consists of controls, which provide prescriptive best practices and recommendations for security in the cloud. Azure also offers automatic evaluation of MCSB compliance based on automated checks, which were useful in the study. With automatic evaluation, the environment's information security can also be upheld in the future. In the study, MCSB was limited to controls regarding infrastructure security and a single project level, which also means that they could be implemented in the Bicep scripts.

Azure Bicep is a cloud-native IaC tool for infrastructure management. Bicep is built on top of its predecessor, ARM templates, and thus works with the same ARM API. Bicep was used for deploying the cloud infrastructure of the case study. In addition, an open-source module library, called CARML, was used in the deployment scripts. Bicep was of great assistance, especially through the use of Microsoft Visual Code's extension for Bicep, which in practice described the resource declaration from the ARM API to the script developer. The extension also listed the compulsory and optional configurations and validated their values.

Keywords: Cloud, Azure, AWS, information security, cloud security, MCSB, IaC, Bicep

TIIVISTELMÄ

Samu Salko : Tietoturvallisen pilviympäristön pystyttäminen Azure Bicepin avulla
Diplomityö
Tampereen yliopisto
Tietotekniikan DI-ohjelma
Helmikuu 2024

Pilviympäristöt koostuvat monista pilvipalveluista ja -resursseista, jotka ovat osittain pilvipalveluntarjoajan hallinnoimia. Nämä pilvipalvelut eroavat toisistaan hallinnan tasossa, asiakkaan vastuissa ja hinnassa. SaaS-palvelun tarjoamiseksi palveluntarjoajan täytyy tunnistaa tarvitsemansa pilvipalvelut, muodostaa niistä toimiva kokonaisuus ja ottaa se käyttöön luodakseen pilveen infrastruktuuri, joka täyttää sovelluksen vaatimukset ja on tietoturvallinen.

Tietoturvallisen pilviympäristön suunnittelu ja pystyttäminen vaativat pilviympäristöön liittyvien tietoturvanäkökulmien tunnistamista. Tämä diplomityö käsittelee tapaustutkimusta, jossa Evitecin olemassa olevalle pilvisovellukselle AWS-pilviympäristössä pyritään kehittämään sopiva pilviympäristö Azureen. Pilvisovelluksen siirto vaatii käytössä olevien oleellisimpien pilvipalveluiden tunnistamista, ja niitä vastaavien palveluiden selvittämistä Azuresta. Koska sovellus käyttää henkilökohtaista tietoa, on erityisen tärkeää ottaa huomioon pilviympäristön tietoturvallisuus. Tämä vaatii tietoturvannäkökulmien tutkimista pilviympäristön ja käytettävien pilviresurssien kannalta. SaaS-toimittaja on vastuussa resurssikohtaisten asetusten konfiguroimisesta tietoturvallisiksi.

Tapaustutkimuksessa toteutetun pilviympäristön tietoturvallisuus perustuu pilviympäristöjen tietoturvallisuuden teoriaan sekä Azure-kohtaisten pilvipalveluiden ja -resurssien asettamiin vaateisiin. Ympäristön tietoturvallisuutta määritettiin Microsoftin kontrollilistalla MCSB:llä (Microsoft Cloud Security Benchmark). MCSB sisältää ohjauskohtia, jotka ohjaavat tietoturvan parhaisiin käytäntöihin ja suosituksiin Azuressa ja AWS:ssä. Azurella on lisäksi automaattista valvontaa, jota hyödynnettiin työssä MCSB:n noudattamisen mittaamiseksi. Automaattisella valvonnalla pilviympäristön tietoturvallisuutta voidaan jatkossa myös ylläpitää. Työssä MCSB rajoitettiin kontrolleihin, jotka liittyivät infrastruktuurin turvallisuuteen ja olivat yksittäisen projektin tasolla, mikä tarkoittaa, että ne voitiin toteuttaa Bicep-skripteillä.

Azure Bicep on pilvinatiivi IaC-työkalu, jolla voidaan hallita pilvi-infrastruktuuria. Bicep on kehitetty edeltäjänsä, ARM templaattien, päälle, ja täten sitä voidaan käyttää saman ARM-rajapinnan kautta. Bicep:iä käytettiin työssä pilvi-infrastruktuurin kehittämisessä ja pystyttämisessä. Lisäksi työssä hyödynnettiin avoimeen lähdekoodiin perustuvaa moduulikirjastoa, CARML:ia. Bicepistä oli hyötyä varsinkin Microsoft Visual Studion Bicep-lisäosan myötä, joka toi ARM:in resurssirajapinnat skriptien kirjoittajan näkyville. Lisäosa listasi lisäksi vaaditut ja mahdolliset konfiguraatiot ja validoi niille asetettujen arvojen oikeellisuutta.

Avainsanat: Cloud, Azure, AWS, tietoturvallisuus, pilviturvallisuus, MCSB, IaC, Bicep

PREFACE

I want to express my deepest gratitude to my supervisor Lauri Oikarinen, who showed interest and endless patience in guiding me on the journey from learning the ropes of cloud infrastructure to writing a master's thesis about it. Time passed by quickly, both the year in which the thesis was written and the "half-hour" meetings, in which 30 minutes were 60 minutes on Earth. Thanks to Evitec for the possibility to do a master's thesis for the company.

Thanks to Kari Systä for supervision and feedback, and for giving me the self-confidence needed in completing the thesis. Thanks and apologies to every student, colleague and friend, who was harmed along the way with excess information about my thesis subject. A big shoutout goes to my friends from Pöytyä: Matias, Lauri, Juho, and Lotta, who helped pull my head out of my dippa from time to time. Thanks for talking to me about my dippa go to my dear friends and colleagues in Tampere: Teemu, Niko, Samu, Simo, Heidi, Matti and Ilari.

Finally, I would like to thank the groups I hold dear, and who made these last 6,5 study years the best time of my life: Hassuja Poikia, HyVäfUKsiVeLIvErKoSTo, Hiukkanen, and TTYMI. To my Hassut Pojat Seppo, Jussi, and Tuomas, big thanks for regressing my sense of humour to the best and the worst level possible. It's been and always will be a master blast.

Tampere, 2nd February 2024

Samu Salko

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

ACL	Access Control List
ACR	Azure Container Registry
Amazon ECS	Amazon Elastic Container Service
Amazon RDS	Amazon Relational Database Service
AMI	Azure SQL Managed Instance
API	Application Programming Interface
ARM	Azure Resource Manager
AVN	Azure Virtual Network
AWS	Amazon Web Service
BYOK	Bring Your Own Key
CAF	Cloud Adoption Framework
CARML	Common Azure Resource Library
CIS	Center for Internet Security
CLI	Command Line Interface
CNAPP	Cloud-Native Application Protection Platform
CRUD	Create, Read, Update and Delete operations
CSP	Cloud Service Provider
CSPM	Cloud Security Posture Management
DevOps	Development and Operations
DNS	Domain Name System
DSL	Domain-Specific Language
ECB	Evitec Covered Bonds
GCP	Google Cloud Platform
HTTPS	Hyper Transfer Protocol over Secure Socket Layer
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code
IDaaS	Identity as a Service

ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LTR	Long-term retention
MCSB	Microsoft Cloud Security Benchmark
MDC	Microsoft Defender for Cloud
ME-ID	Microsoft Entra ID
MFA	Multi-factor authentication
NAT	Network Address Translation (Gateway)
NIST	National Institute of Standards
NSG	Network Security Group (Azure)
OS	Operating System
PaaS	Platform as a Service
PCI-DSS	Payment Card Industry Data Security Standard
PITR	Point-in-time restore
POC	Proof Of Concept
RBAC	Role Based Access Control
SaaS	Software as a Service
SFTP	Secure File Transfer Protocol
STR	Short-term retention
TCP/IP	Transmission Control Protocol / Internet Protocol
TLS	Transport Layer Security
VM	Virtual Machine
vNet	Virtual Network
VNM	Virtual Network Manager
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WAF	Web Application Firewall
WAF	Well-Architected Framework

1. INTRODUCTION

With cloud platforms and cloud services, the responsibility of information security is shared between the customer and the cloud service provider. Cloud services are subscription-based, and their features scale up with costs. There are many security features and cost-tiers, which either directly improve security or give the customer additional controls for security. But with services either fully or partially managed by the cloud service provider, what does the software vendor need to account for?

This thesis explores information security in the cloud with a focus on information security in infrastructure. Information security is a constantly evolving area, that needs to respond to changing security aspects. Multiple organizations, such as ISO (*International Organization for Standardization*), NIST (*National Institute of Standards and Technology*), and CIS (*Center for Internet Security*) produce and update guidelines for secure software development, both open-source and proprietary.

Infrastructure in the cloud consists of cloud resources and services, on which the application and business logic are implemented. Infrastructure security forms a large part of the application's information security. Designing and building secure infrastructure is therefore important. If done inadequately, it cannot be patched later by improving the application code. In addition changing existing infrastructure has a price tag. Design flaws in infrastructure can take some time to discover and the changes can reflect on other resources. Infrastructure changes take time to implement, can introduce new issues, and usually cause downtime to production. If a compromised infrastructure design is used in multiple environments, they need to be updated as well.

One solution for managing infrastructure deployments is Infrastructure as Code (IaC), which considers tools for scripting languages for automating infrastructure. One example of an IaC tool is Azure Bicep, which is a cloud-native tool for deploying infrastructure into Azure. The same script can deploy multiple similar environments with different parameters and can handle updating the infrastructure as well.

This thesis aims to answer the questions "*How to design an information-secure environment in Azure?* and "*How to evaluate the information security of an Azure environment?*". To find answers to these questions, this thesis introduces a case study, provided by Evitec. Evitec has an existing product in the Amazon Web Service (AWS) cloud environment, and

they aim to transfer it into Azure. This requires designing the corresponding architecture on Azure with the information security requirements in mind. In addition, they are interested, in how it could be done with Azure Bicep.

At the beginning of the thesis, the case study will be introduced in more detail, how the research is conducted, the constraints for the research are presented, and what the work aims to achieve. This is followed by introducing the most relevant cloud services and their respective effects on information security. In every service, the corresponding service and its controls in Azure will be discussed. Then, relevant frameworks for implementing information security and the MCSB control list are presented for designing and building information secure infrastructure. Then the thesis introduces the proof of concept architecture for the application, and the environment's information security is evaluated with the Microsoft Cloud Security Benchmark (MCSB). In the end, the results are presented, a summary is made of the environment's information security evaluation, and a reflection is made on what could have been different and what to consider in further studies.

2. RESEARCH TASK

In this master's thesis, information security in Azure cloud environments is researched and different tools are introduced for its evaluation. The research is conducted using a case study approach. In this chapter, the background for the case study is reviewed and the research area is restricted to create appropriate research questions.

2.1 Background

The case provider for this master's thesis is Evitec. Evitec has an existing product, Evitec Covered Bonds, later referred to as the product or ECB, which provides core functionalities needed for running a mortgage banking business. These functionalities include automatic and optimized pooling of mortgage loans for issuing and managing covered bonds and an automated credit rating agency, regulatory and business reporting, and analytics. [1]

The ECB product is a Software as a Service (SaaS) product hosted on Amazon Web Services (AWS) cloud environment, but plans are already made for transferring the product into an Azure environment. Building an information secure environment with modern infrastructure automation tools is critical for an application, that manages sensitive information from mortgage debtors and mortgage banks.

The business logic of the product is containerized with Docker and deployed to the cloud via AWS's ÏaC tool: AWS CloudFormation. The product's infrastructure is based on many AWS services, which need to be transferred to corresponding services in Azure. The main architecture is set for the product, but the security aspects need to be recognized and configured in the Azure environment. The product's restrictions are introduced in section 2.2 and the current architecture in section 5.1.

2.2 Research questions and restrictions

The goal of the product is to be securely hosted in an Azure environment. Designing a secure architecture and building the infrastructure are the main concerns when developing the SaaS product's Azure version. The current solution's infrastructure is managed via the AWS CloudFormation, which uses YAML files to declare the resources and con-

figurations for the application's infrastructure. A modern cloud-native IaC tool on Azure is Bicep, which can be described as the next iterative version of ARM.

An information-secure environment for the product is crucial for a mortgage bank pooling its debtors' mortgages into covered bonds, as there is sensitive data involved. This starts with the secure use of the infrastructure tools. In addition, it requires knowledge of the responsibilities and the security threats in the cloud environment. One needs to consider the resources' security controls, and their alignment with the virtualized infrastructure, such as networking. This brings us to the questions of: "*How to design an information-secure environment in Azure?*" and "*How to evaluate the information security of an Azure environment?*".

The case study revolves around the product and decisions considering its architecture and infrastructure in Azure. The information security considerations are limited to the ones regarding architecture or infrastructure. In addition, the scope will be limited to the project level and its immediate surroundings, which means that organization-level information security considerations are left out of the scope of the thesis, examples of which are areas concerning strategies, processes, or personnel. This also leaves out a large chunk of information security in the form of software security, which is important to note. With these limitations, the controls chosen for the study can be implemented in the Bicep scripts. These restrictions are valid in the designing process of the environment and the evaluation of the MCSB controls. Compliance with the MCSB controls is considered through the lens of infrastructure deployment and on the project level.

2.3 Research method

This master's thesis aims to find solutions to the research questions presented in the previous section through a case study. A case study is a detailed study of a specific subject through describing and analyzing a real-life case. Case studies are commonly used in social, educational, clinical, and business research. The case in this thesis is the transfer of Evitec's product, ECB, from AWS to Azure and the research questions are reflected upon it. In this thesis, the case study is an appropriate research method, since it provides a way of gaining concrete and deep knowledge about the case and its problem. [2]

After introducing the research questions and scope, the next step involves the building theoretical background around the case. The theory consists of introducing areas of information security and introducing their implementation in Azure. It also includes introducing related cloud components in Azure and inspecting their security controls. After this Azure's infrastructure tools will be introduced, which include tools for infrastructure automation, frameworks for architecture in Azure, and tools for evaluating information security.

The research is conducted using a proof of concept (POC) project representing the case. The POC project designs an architecture for Azure and deploys the infrastructure to Azure. The POC project's infrastructure's information security is evaluated and analyzed through different tools, reflecting on how secure the POC infrastructure is in the Azure environment and how suitable the infrastructure automation tool Bicep is for the task.

In the end, the results of the case study are presented, which depict, how suitable MCSB is for the task and the process of creating the Bicep scripts. In the summary, the results are analysed and ideas for further studies are introduced. The POC project does not aim to be a general solution for all kinds of cloud applications. It aims to find the corresponding functionalities and services from Azure compared to the current AWS environment. However, the theory and tools behind it represent good practices and guidelines for creating an information-secure environment in Azure.

3. INFORMATION SECURITY IN CLOUD

This chapter goes through different aspects of information security and their adaptations in cloud environments. In addition, some Azure concepts are introduced through concrete examples. Traditionally in web applications, most of the security concerns are those of the application provider. Especially, if the servers are hosted locally (an on-premises solution). With remote virtual machines (VMs), some of the infrastructure's security is the responsibility of the provider of those VMs. For example, they need to provide the customer with a secure way of connecting to the VM through authentication. [3]

3.1 Information security principles

Information security means protecting data from unwanted usage or access, and it can be described through the CIA-triad, which consists of Confidentiality, Integrity, and Availability. Confidentiality refers to limiting access to data only to users, who are meant to view it. This can be implemented through authorization and authentication. Integrity refers to preventing unauthorized users from modifying data or parts of it in an unwanted way. Upholding integrity means restricting data modification to authorized users while having a way of reversing unwanted changes. Availability refers to the ability to access the data when needed, which can be for example implemented by scaling up compute resources when needed. [4]

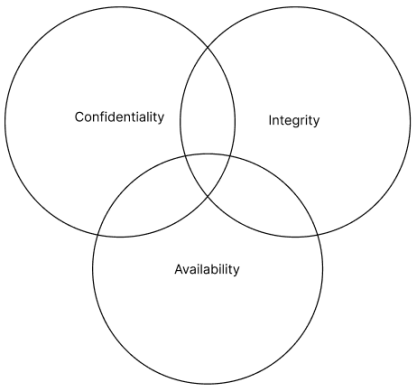

Figure 3.1. *The CIA triad*

Typically in traditional on-premise solutions, one organization is responsible for almost

everything, such as building, developing, maintaining, and updating the infrastructure and software of the solution. In cloud platforms, some of the responsibility is always outsourced to the Cloud Service Provider (CSP), for example, the management of the physical infrastructure. The responsibilities of the customer depend on the cloud service model, which depicts the level of virtualization for the customer. Virtualization means how much of the maintenance and configurations are managed by the cloud service provider. On the other hand, it affects, how much the customer has management responsibility and control over the platform. The cloud service models in order of increasing virtualization are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). [3]

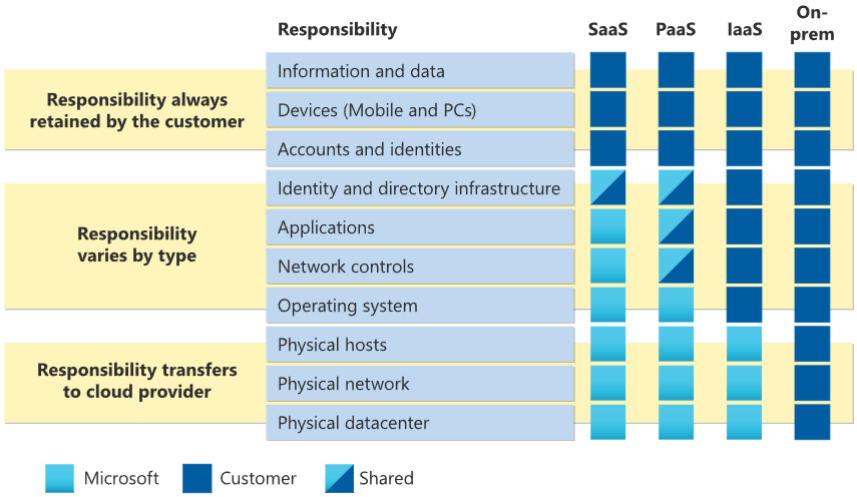

Figure 3.2. *Shared responsibility model according to Azure [5]*

In the IaaS model, the customer pays for the infrastructure (hardware) and their physical configuration, but can for example choose the operating system (OS), and configure the system settings from there on upward. The customer is therefore responsible for updating and security of the operating system and developing the application. In the PaaS model, the customer pays for a ready platform with an OS, so management of the OS is the responsibility of the cloud provider. In the SaaS model, the customer pays also for the application and takes responsibility for the data. In all of the service models, the customer is responsible for protecting the security of their data. [3, 5]

3.2 Azure management hierarchy

The basic hierarchy of the CSP often defines the basic hierarchy for administrative levels and inheritance for access controls. The access controls set upper on the hierarchy are inherited by their children. In Azure, the topmost hierarchical layer is the tenant. Usually, the tenant is an organization such as a school or a company, who is in charge of subscriptions, billing, and users. With the creation of the tenant, a Microsoft Entra ID

(ME-ID) is created for its users. Every user account in Azure is linked to the tenant via its ME-ID (formerly known as Azure Active Directory).

At the tenant level permissions are handled via Management Groups. The management groups control access at the organizational level, and they can be nested. After the management groups, comes the subscription level, which holds the licenses and billing information. At subscription level and lower hierarchy levels access controls are handled by role assignments. Role definitions and assignments are discussed in detail in chapter 3.6. Under subscriptions, there are resource groups, into which cloud resources are deployed. Lastly, there are individual resources. Azure's hierarchy is presented in figure 3.3. [6]

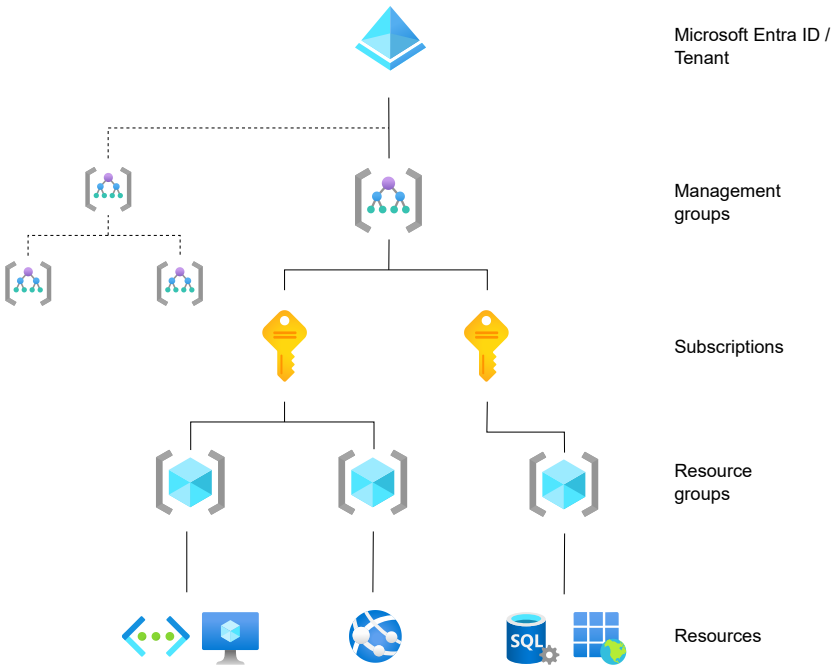

Figure 3.3. *Hierarchy of management levels in Azure. [7, edited]*

3.3 Zero trust policy

In more traditional on-premise solutions security could be based on the assumption, that if the internal network is secured by authentication, the services inside it wouldn't have to be. But with the increasing number of distributed services, connectivity between different services rises. Multiple resources lead to a bigger attack surface. If the services trust implicitly each other's location inside the network or a physical location, that poses a security threat. If an attacker managed to infiltrate one of those services, they potentially have ways of infiltrating other services too. [8]

This has given rise to a security model called *Zero trust*, which avoids implicit trust in other services and locations. In Microsoft's documentation the guiding principles for Zero Trust are defined as follows [9]:

- Verify explicitly - Every network call from one service to another is assumed untrustworthy, and it needs to be authenticated and authorized before responding.
- Use least privilege access - Grant identities the least privileges needed to complete the task and limit the privileges time-wise.
- Assume breach - Separate different resources and use threat detection services, so that in the case of a breach the damages are contained, minimized, and responded to.

3.4 Control plane and data plane in cloud

In traditional networking, a *plane* is an abstract conception of where certain processes take place. The control plane is the part of a network, that controls how data packets are forwarded, and the data plane forwards the data packets. [10] With this analogy in mind, these planes are used in managing cloud resources too.

In Azure and AWS most operations and services are divided into control planes and data planes as well, depending on the level of controls on cloud resources. The control plane provides the administrative APIs for create, read, update, and delete operations (often referred to as CRUD operations) on cloud resources, while the data plane offers resource-specific controls for CRUD operations on data for example. The planes are important to separate, as they require different sets of access permissions. The control plane is important for the cloud vendor for managing the cloud resources, while the data plane is important for the customer for accessing and viewing their data through an application interface. The planes in Azure are presented in the figure 3.4. [11, 12].

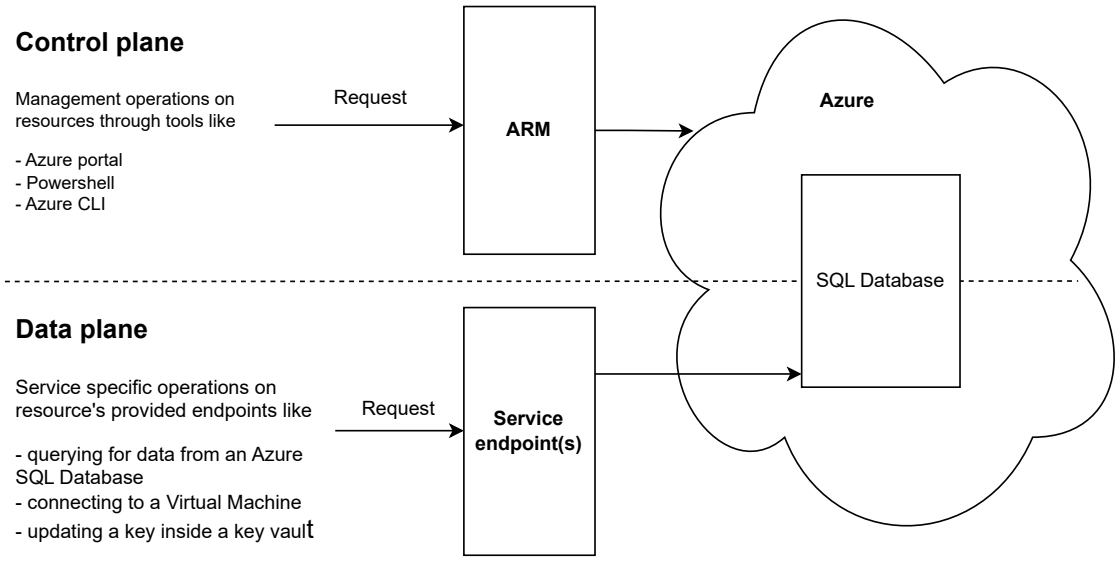

Figure 3.4. *Control and data plane in Azure. Control plane operations go through ARM and affect resource states in the cloud, while data plane operations go through the resource's service endpoints, and affect a resource's inner state.*

In Azure, all of the control plane requests are handled through the Azure Resource Manager (ARM), which manages the resources and their states in the Azure cloud. ARM is presented in more detail in chapter 4.3. The control plane operations in the figure 3.4 consider operations affecting the SQL Database resource, such as deploying it, changing access permissions for changing it, editing its storage encryption, and in general affecting the resource-specific configurations.

The data plane in Azure (similar to AWS) considers the service endpoints, that the resource offers. In the figure 3.4 the resource inside the cloud is an SQL Database, which holds relational data tables. One service endpoint is the domain name (server address name), to which the SQL query connections are established. The data plane operations consider the CRUD operations related to the data inside the SQL Database. [12]

3.5 Data security

Data security covers security specifically related to data storage and handling of data. It is a layer of information security focused on data storage to secure data at rest, in transit, and use. Securing data at rest means securing stored data, usually using strong encryption and access management. Securing data in transit means securing data transfer between two endpoints. Securing data in use means securing data from the perspective of the application. The application needs to decrypt the data for usage, so the application needs to implement restrictions on user access level. [13]

The data security life cycle has six phases [14]:

Creation phase: Either new digital content is generated or acquired, or existing content is altered or updated. The creation can happen within the cloud or externally after the data is imported into the cloud. It is during the creation phase that content should classified based on its sensitivity and value to the organization.

Storage phase: The storage phase involves the commitment of data to a storage repository, typically occurring simultaneously with the creation phase. During this stage, data protection measures should be aligned with the content's classification level. Various security controls, such as encryption, access policies, monitoring, logging, and backup mechanisms should be implemented to mitigate potential threats.

Usage phase: The usage phase covers viewing, processing, or usage of data in some activity, excluding its modification. Data in use is particularly vulnerable, as it may be transported to unsecured locations, such as individual workstations.

Sharing phase: Sharing is data made accessible to others, such as users, customers, or partners. Once shared, the data is no longer entirely under the organization's control, making the maintenance of security more challenging. Technologies can be implemented for data protection, such as data loss prevention technologies to prevent unauthorized sharing, and data rights management technologies to maintain control over the data.

Archiving phase: Data is moved from active use to long-term storage. Archiving typically means lowering the cost of data storage by lowering availability. Data in the archive must still be protected according to its classification needs.

Destruction phase: The permanent destruction of data using physical or digital means. The specific method for destruction depends on usage, data content, classification, and applications.

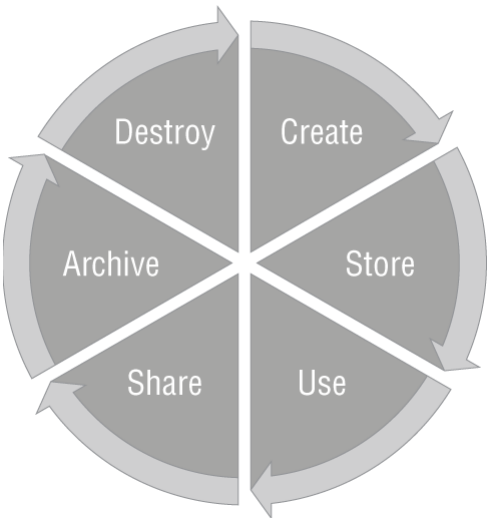

Figure 3.5. *Data life cycle [14]*

Data encryption is one of the fundamental parts of data security, and it can be done client-side or server-side. Client-side encryption means, that the client holds the means for encrypting and decrypting the data, which they upload to the server. Therefore the server only manages data in encrypted form. In server-side encryption, the client does not encrypt the data for encryption at rest purposes (dismissing encryption transit purposes), and it is the server's responsibility to manage the encryption for storage.

Data encryption in transit means securing the actual motion of data over a network through the means of encryption. SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols widely used for managing encryption in transit in applications such as email, instant messaging and HTTPS (Hyper Transfer Protocol over Secure Socket Layer) [15]. In SSL/TLS protocol the client and server make a handshake, in which they exchange information to create a shared encryption key for the session. This shared key is known only by the server and the client and is thereafter used for encrypting and decrypting the data packets. [16]

Typically each CSP has its cloud native data storage solutions. Cloud storage's strengths are that it scales well and is highly available. Copies of data are usually kept in multiple locations to reduce availability issues related to one location's network and ensure data preservation. Backups can also be distributed across multiple sites for better certainty of preservation than in the same physical location.

Data storage in modern software is typically divided into two categories; storage of data and storage of data files. This is mainly due to differences between storing structured and unstructured data. Storage of structured data is typically implemented via relational database solutions, such as an SQL database. Relational databases are not suited for storing unstructured data, such as documents, images, or video files, as these can hinder the performance of the database. Thus CSPs have managed native file storage services, which often offer folder structures in the cloud's network. In the managed file storage the CSP is responsible for hardware, storage, OS updates, and security, while the customer is responsible for managing data and access policies. Remote access to the files is granted via access policies to resources and users, who need them. [17]

Each cloud provider typically has a managed database as a service, where the CSP takes care of the infrastructure needed and updates the OS and database software. The customer is responsible for importing their data to the database and for operational usage. If the customer's data or data schema is not suitable for a managed database, they can spin up a VM and install their chosen database there in a more traditional lift-and-shift manner. [17]

Data security in Azure

For unstructured data storage Azure storage platform offers different services, that have roughly the same security features. Authorization to the storage accounts supports role-based authorization through ME-ID (which is discussed more in detail in section 3.6). Encryption at rest is enabled by default and managed by Azure. However, they also support custom keys for encryption in a method called *Bring Your Own Key* (BYOK). In addition, client-side encryption is supported. For data redundancy, multiple copies of the data are stored in different locations. The locations can either be configured to locate within the same data center or in multiple different regions depending on how well the data needs to be preserved. Starting from the default minimal option, in which the copies of the data are kept in the same data centre, Azure provides at least 99.999999999% (11 nines) durability of objects over a given year [18]. [19]

For hosting an SQL server there are three main alternatives: SQL Server on Azure VM, Azure SQL Managed Instance (AMI), and Azure SQL Database. The service with the most maintenance responsibility for the customer is the SQL Server on Azure VM (IaaS), representing the same deployment and install operations as one would have on an on-premise solution. This way the customer keeps the responsibility of maintenance but can take advantage of cloud possibilities with VMs, such as virtual networking and key vault integration. The SQL server version can be decided entirely by the customer, and therefore this solution supports applications, that are dependent on older SQL versions. It is a typical implementation of a lift and shift type of database migration, as the VMs database implementation can be made very similar to the on-premises database implementation.

AMI and Azure SQL Database are fully managed PaaS solutions, which means that the SQL Server will always represent the cloud's most recent version, and the updating will be automatically managed by the CSP. This is a good thing for security, as the SQL Server version will be updated frequently and tested by Microsoft. They both have similar security features presented in table 3.1. [20, 21]

Table 3.1. *Security features of Azure SQL Database and Azure SQL Managed Instance [22]*

Feature	Description
IP firewall rules	Both services offer access rule configuration based on the originating IP address.
Virtual network firewall rules	Both services offer access rule configuration based on the originating virtual network location.
Always encrypted	Both services support data at rest and in use is always encrypted by default.
Automated backups	AMI supports both system and user-initiated backups (BACKUP command), while SQL Database supports only system-initiated backups.
Application roles	Both services use database principals for enabling access to specific data to users, who connect through a particular application.
Row-level security	Both services support implementing restrictions to table data on the row level.
Authentication	Both services support authentication using SQL authentication, ME-ID, or Windows authentication for ME-ID principals.
Threat detection	Both services have additional threat detection services for detecting and alerting anomalous or suspicious database activities.
TLS protocol (Encryption-in-transit)	Both services support encrypting data in motion with the TLS protocol.
Key management with Azure Key Vault	Both services support key management via Azure Key Vault for managing their logic for encryption keys and their rotation (BYOK).

AMI represents a virtual abstraction of an SQL Server instance, where the customer has both the SQL Server configurations and database configurations available. It is a bit closer to the on-premises SQL Server implementation and is better for lift and shift types of migrations. SQL Database is an abstraction level higher and focuses on database-level controls, featuring serverless compute tiers for automatically scaling compute resources based on workload activity. [23, 24]

Backups and restoring data are essential parts of data security, whether the need for

restoring data comes from an application error, human error, or malicious intent. Backuping data is an inherently preventative method. Data backups in Azure are managed either by the resource's configuration or the Azure Backup service if the resource service is supported by it. [25]

Azure Backup supports Blob storage with two types of backups: continuous and periodic. Continuous backups protect the block blobs from accidental deletion or corruption, as the backups retain changes and allow restoration at a selected point in time. Periodic backups copy the data to a Backup vault periodically. The backup vault is an Azure storage account, and every copy of the data represents a recovery point in time, to which the database backup can restore the blob storage. [26]

Backups for Azure SQL Databases are configured through the resource's configuration. SQL databases provide short-term retention (STR) and long-term retention (LTR) of backups. Short-term retention allows PITR (point-in-time restore), which enables the user database to be restored at a given time since the last full backup. This is made possible through differential backups, which preserve the changes made since the last full backup. The differential backups can be configured to either 12 or 24-hour intervals. The STR backups can be configured to be preserved between 1 to 35 days. [27]

It is important to note, that an Azure SQL Server in itself cannot be backed up. Furthermore, if an SQL Server is deleted, SQL Databases hosted inside it alongside their STR backups are lost also. This is where the LTR backups come in. LTR backups are stored in a Blob storage for up to 10 years. LTR Backup timing and intervals are configured by using a combination of four parameters: weekly backup retention (W), monthly backup retention (M), yearly backup retention (Y), and week of the year (WeekOfYear). LTR backup intervals can only be configured by these parameters, and cannot be initiated manually. [28]

3.6 Identity and authentication management

Identity and Authentication Management (IAM) is the means of managing identities, their permissions, and their life cycles. This is a fundamental block of information security, as this is related to all the sections of the CIA-triad. Identities can be users, computers, services, or roles, that require access to a system or application. The identities are usually stored in the cloud system, and they can be categorized into different groups, which hold the same permissions. When a client sends a request for information, the identity needs to be authenticated. After authenticating the identity, the request is authorized, meaning that the authenticated identity's permission for taking a specific action is confirmed. IAM also handles the life cycles of roles and permissions, so that they can be granted and revoked as needed. [3]

IAM means managing access to different cloud resources by defining permissions for certain actions to certain identities. For example, from the development point of view, the platform architect should have sufficient permissions to create and manage the cloud resources in the software project's scope. *Platform architect* could be seen as a role, which has administrative permissions for CRUD operations on the data plane in the project's scope. Then there are software developers, who need access rights to modify and manage the actions and functionalities inside those cloud resources, but don't need the access to modify the cloud resources themselves. Thus *software developer* could be seen as one role, that has sufficient permissions for managing the cloud resource's data plane operations, while not having permissions for the control plane of the resource.

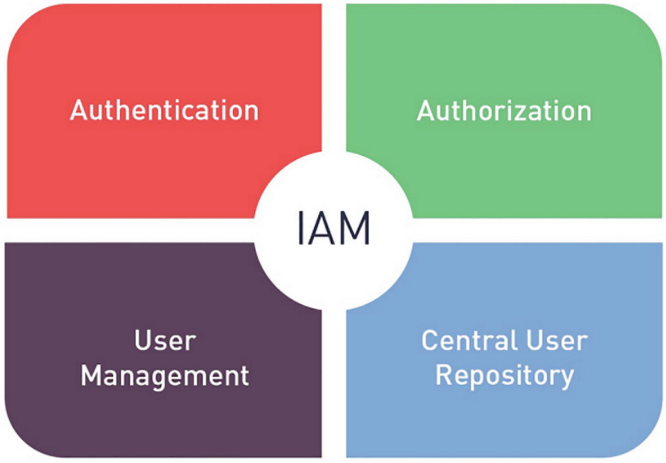

Figure 3.6. *IAM and its main functions. [29]*

Managing access in a software project can be a challenging task with multiple moving parts: users with too much access, users with too little access, former employees with active credentials, resources, or services left too open from the development phase. With the rise of microservice architecture, the amount of resource management has risen significantly, and it can be quite tedious to handle manually. This is why IAM aims to automatize and separate identity management from the actual development.

Development processes require higher access levels than the end-users need. For example, accessing the database directly is typically restricted from end-users (or connections from the public internet), but it is a necessity for the application's backend to store and manage information. Databases for different purposes can also have different access policies. A database purely for development could have lighter restrictions than a database in production use. In line with *Zero trust*, one should aim to limit access privileges to the least ones needed. It is clear, that the different needs require identification, categorization, and grouping mechanisms.

Different cloud providers typically offer their own IAM services. In Microsoft's Azure, there's ME-ID, which handles the identities, roles, and permissions regarding cloud re-

sources in Azure. in AWS the corresponding service is AWS IAM. However, this does not mean, that the IAM service would be limited within the cloud service or a cloud application. For example, ME-ID is used as an IAM service for many of Microsoft's SaaS applications, such as Outlook and OneDrive. ME-ID can also be used as an independent third-party IAM solution, thus it can be referred to as an Identity as a Service (IDaaS). [30, 29]

IAM in Azure

ME-ID functions with security principals, which represent an entity to which permissions are granted. A security principal can represent a user, group, service principal, or managed identity, presented in figure 3.7. A service principal is an application, which grants tokens for authenticating and granting access to specific Azure resources by using the service principal's credentials. Managed identities are very similar in functionality to service principals, but there are no credentials assigned to them. Managed identities can be assigned to resources, which can then use the managed identities for accessing other resources. [31]

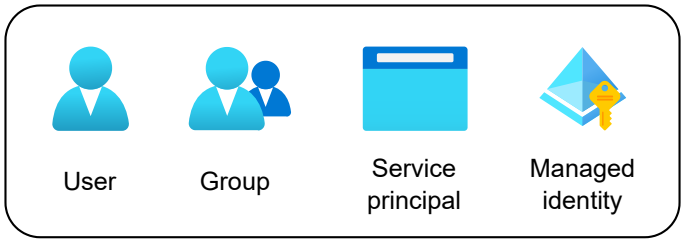

Figure 3.7. *Security principles in Azure [7, edited]*

Permissions of a security principal are managed by assigning them roles. A role is a collection of rules, and by assigning them to a security principal, they get access permissions defined by the rules defined in the role. The roles can be defined on different hierarchical scopes (management group, subscription, resource group, or resource), and they are inherited downwards. This is called Role Based Access Control (RBAC), and it lifts the access management from individual service level one abstraction layer higher. A security principal can have multiple roles assigned to them, which can for example reflect different roles in a project or roles in multiple projects.

The roles are divided into *Microsoft Entra roles* and *Azure roles*. The Microsoft Entra roles are at the top of the role hierarchy, and they manage permissions on the directory level such as creating or editing users, assigning administrative roles to others, managing user licenses, and managing domains. Then there are Azure roles, which are at the heart of the Azure RBAC authorization system. Through Azure roles, one can set resource-specific permissions from either built-in roles or custom-defined roles. [7]

Groups in Azure mean entities, to which security principals can be added as members. In Azure, the groups are divided into two kinds: security and Microsoft 365. Security-typed groups are used for managing the permissions of security principals. To them, one can assign administrative roles and Azure roles, which were discussed before. *Microsoft 365*-typed groups are meant for more consumer-like purposes, sharing access to Microsoft services such as mailbox, calendar, and SharePoint sites. [32]

There are three different types of memberships to Azure groups depending on how the group assignment is made: *assigned, dynamic user, and dynamic device*. First of which means manually adding a security principal into a group, or removing them from it. *Dynamic user* is an automatic way of assigning group members based on whether the conditions set for being a group member are met. If the attributes of a user change, the system will look into dynamic groups, if the user meets a group's rule requirements. Depending on the current group status and requirements, the user is added to the group or removed from it. *Dynamic device* works similarly to the dynamic user, but on a machine identity.

A managed identity represents an identity, to which Azure roles can be assigned for granting access permissions. By assigning the identity to a resource, it can be used for authentication and authorization inside the Azure IAM. For example, a built-in role *AcrPull* holds the set of rules needed for reading an Azure Container Registry (ACR), thus granting access to pulling images from it. By assigning the role *AcrPull* to either a user or a resource, it can authenticate and authorize itself through ME-ID for reading/pulling images from the ACR. The scope of the role is defined in the assignment phase, and the role permissions are inherited downwards in the following order (greatest first): tenant, subscription, resource group, resource. The role assignment is illustrated in figure 3.8.

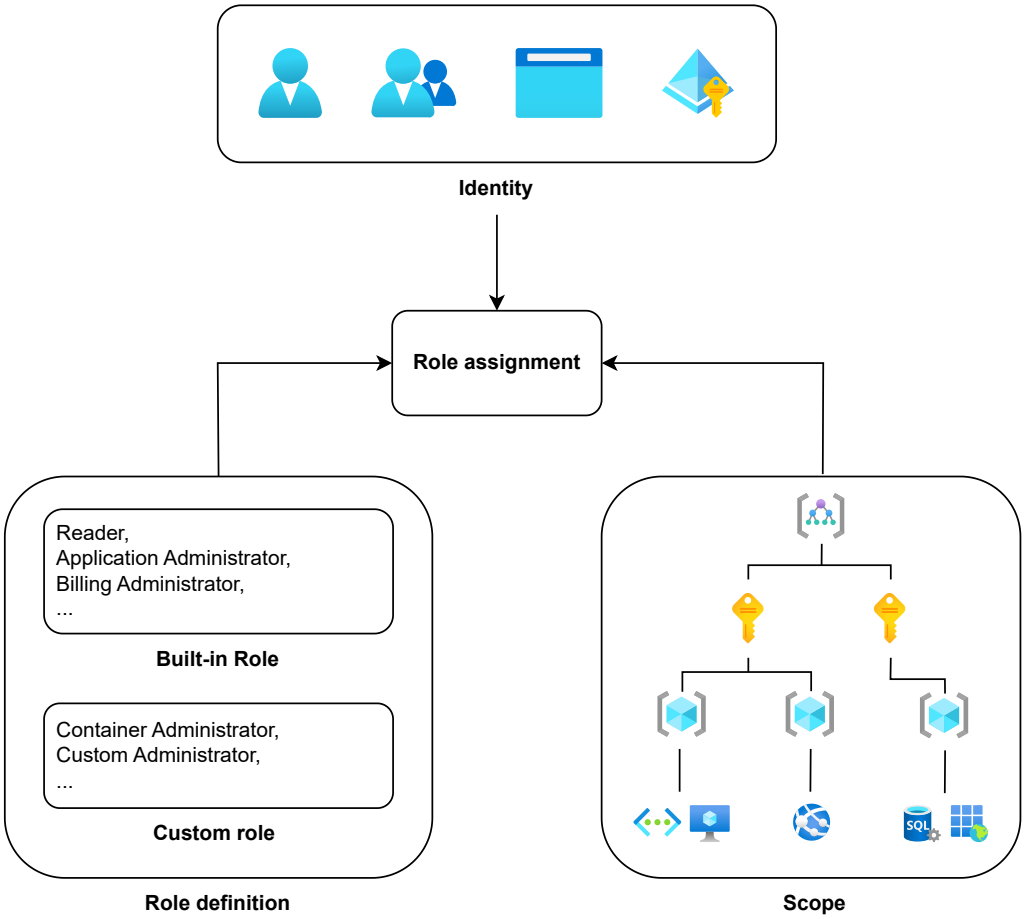

Figure 3.8. *Role assignment in Azure [7, edited]*

There are two kinds of managed identities in Azure: system-assigned identities and user-assigned identities. System-assigned identities are assigned to resources on deployment, and they are bound to the resource's lifetime. After the creation of a resource, the system-assigned identity can be assigned roles depending on its needs. With user-assigned identities, the identity is created separately from the resource deployment. This way the user-assigned identity's life cycle is different from the resource's life cycle. Multiple user-assigned identities can be assigned to a resource to grant it different sets of rules. Similarly, a single user-assigned identity can be shared across multiple different resources.

3.7 Network security

Network security means controlling network access to the resources by allowing certain connections through while blocking unwanted or unauthorized access. Different cloud service models mean different levels of responsibility over the network controls for the customer. The physical network layer consists of access between the network hardware and the physical host servers and storage within the cloud provider's data centres. The

physical network layer is the cloud provider's responsibility. The customer's responsibilities include virtual networking, which depicts the configuration of connections between different cloud resources within the cloud provider, as well as connections from outside networks to the cloud's virtual network (vNet). [33]

Internet Protocol (IP) addressing is an important part of Internet communication, as it lays the foundation for the TCP/IP (Transmission Control Protocol / Internet Protocol) Internet communication protocol. For a client to make a connection through the public internet to a cloud resource, the client needs to have an IP address to connect to. IPs are typically used via Domain Name System (DNS) registries, which act as registries for finding an IP address for a domain name. Using the DNS registry one can connect to a resource with the domain name without knowing the IP address. The DNS servers can be hosted either publicly, where they are publicly accessible, or privately, where they are only accessible inside a limited network. The private DNS servers can be used for networking inside the same network.

IP addresses can be either private or public addresses. Private IP addresses are used to communicate with resources within the same network, thus they need to be unique within the same network. Public IP addresses can be contacted through the internet, and they need to be unique globally. Subnets are segmentations of a network, which have a portion of the network address range appointed to them. [34]

Connecting different networks is relevant both in traditional solutions and cloud solutions. Often cloud applications are some form of hybrid solutions, where a physical network is connected to the cloud application's virtual network via a secure tunnel, such as using a Virtual Private Network (VPN) connection. A VPN connection creates a point-to-point tunnel between two nodes, and encrypts the communication inside it, hiding information about the sender's IP address and personal data. This way connections from an office's physical network can be more secure, and the cloud network can be more hidden from the public network. [35]

Network topology is the arrangement of a network consisting of nodes and their links. It is an important aspect of network security, as different nodes inside the network topology have different networking needs and a need for security. From the aspect of subnets inside virtual networks, some subnets need to be able to communicate with other subnets, and some need to be able to connect to the public internet.

From the aspect of a web application, there may be multiple similar environments for development, staging, and production. Managing multiple public entry point nodes for these private environments would not be very effective. Here a hub and spoke-topology (or star topology) can be useful, where one node acts as the central hub, to which all the other nodes, called spokes, are connected. For example, one virtual network can act as the hub network, allowing public access to it and forwarding it to spoke virtual

networks. The spoke networks contain different environments for development, staging, and production. Therefore the spoke nodes are not available directly from the public internet, and they can only be reached through the shared hub's public endpoints. This limits the attack surface of the network, and optimizes costs, as only the hub network needs to be secured for attacks from the public internet. On the other hand, if the hub node fails to work, the whole system is unavailable from the public internet. Therefore the hub acts as the single point of failure. [29, 36]

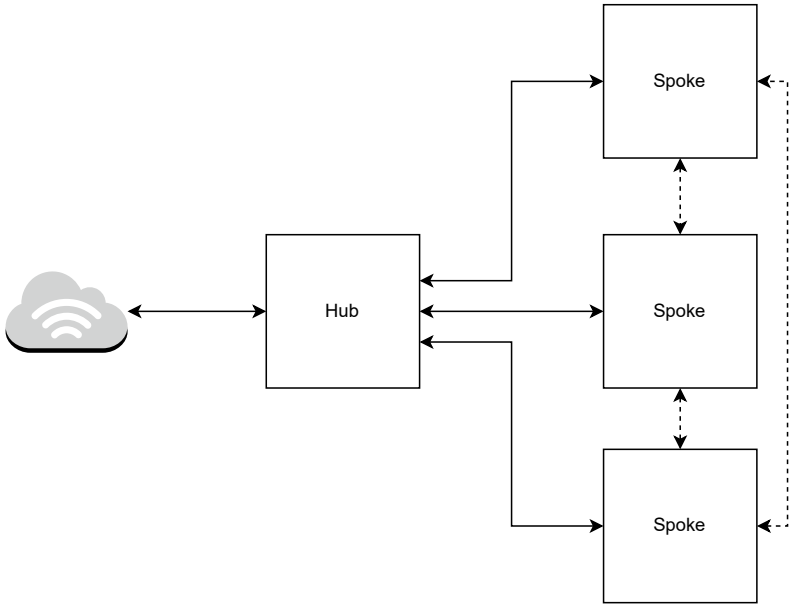

Figure 3.9. *Hub and spoke -topology, where traffic from outside network goes to the spokes only through the central hub. The spokes can be interconnected.*

A virtual network is a software-based network inside the customer's cloud environment, where most of the common cloud resources are. Through them, the customer can configure, how different cloud resources can communicate with each other, and how connections can be made between the virtual network and the outside world. Subnets represent segments of the virtual network, which are more granular tools to manage communication between the cloud resources. Subnets are appointed a portion of the virtual network's IP address range, so they have unique IP addresses in their use within the virtual network [37]. The subnets can be either public or private, depending on whether access to the public internet is allowed. An example of a simple virtual network configuration is presented in figure 3.10. [3][38]

Virtual networks are abstractions of a network (both physical network and some of the network controls). They have public IP addresses, that make connections from the internet possible, as well as private IP addresses, which make connections within the address space possible. Depending on the cloud service model, the responsibility of network controls varies greatly. In both the on-premises and IaaS models the customer is fully

in charge of the network controls. With the difference of responsibility for the physical network equipment, in both the models the customer needs to configure all the network controls, hosts, IP addresses, and regulations usually on the OS level of the host computer. In the PaaS model, the customer does not have access to the OS level but has additional tools provided by the platform provider to configure and regulate the communication related to the cloud resources.

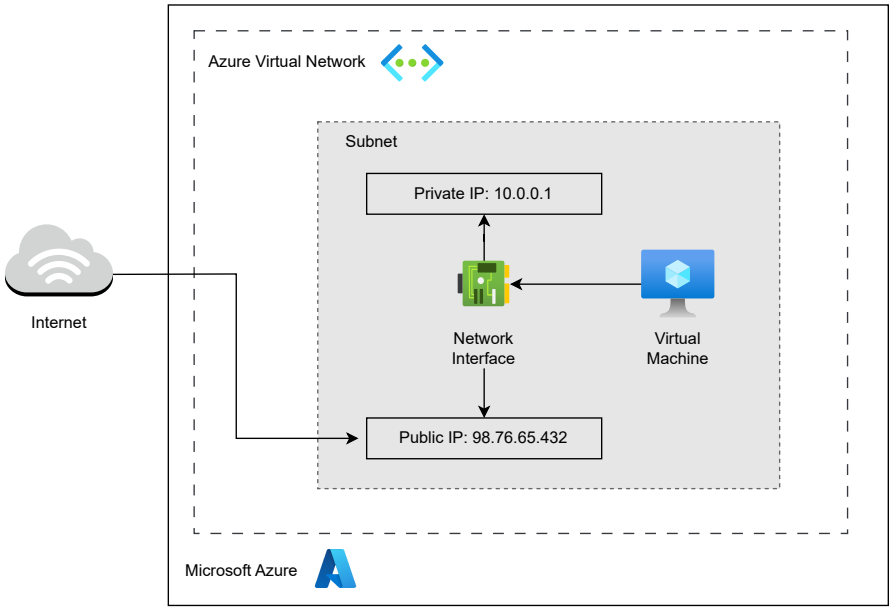

Figure 3.10. *An example of a virtual network configuration with a VM. In this example, the network interface is separate from the VM.*

Access control to virtual networks is based on rules, which either allow or block access to the virtual network. The rules can be configured on the sender's IP address (or address space), which for example makes it possible to allow access from a specific location and block other connections. Access should also be limited to specific endpoints, which have specific cloud resources listening to them. Leaving unnecessary ports open weakens the overall security and can lead to potential ways of exploitation. Rules can also control, what kind of connections are allowed to an endpoint by specifying the communication protocols allowed. The communication protocols define, how the connection is established and data moved between the sender and the receiver. The rules can also target the direction of the traffic, limiting the rule to outbound (outgoing) or inbound (incoming) traffic. [39][40]

Gateway is an important concept in handling traffic between different networks. Gateways are nodes located inside a network, that act as a boundary for inbound or outbound traffic. Inbound traffic means connections, that originate outside the given network, and are directed to the given network. Outbound traffic is the opposite; connections that originate from within the given network, and are directed to other networks. As private IPs can be used for communication only within the network, gateways are needed to route traffic from the public IPs of the resources to their private IPs within the network.

When a resource within a network creates a connection to another network, it needs a unique public IP address, as the private IP address can only be addressed within the same network. To solve this problem the NAT (Network Address Translation) gateway was created. A NAT Gateway distributes resources within the network public IPs to use for creating outbound connections to other networks. When the request is responded to, the NAT Gateway can translate the response's target public IP address to the private one. This way the private address of the resource is not revealed to the other networks.

Network controls in Azure

Azure Virtual Network (AVN) is a resource that implements the virtual network in Azure. An AVN contains subnets, which act as logical containers for resources inside the virtual network. An important aspect of AVNs is, that by default all the resources inside an AVN can communicate outbound to the internet. This is important to notice, as it does not follow the principle of least access or the zero access policy. For inbound communication, the resource needs to have a public IP address or a public load balancer. Azure's network segmentation hierarchy is presented in figure 3.11. [41].

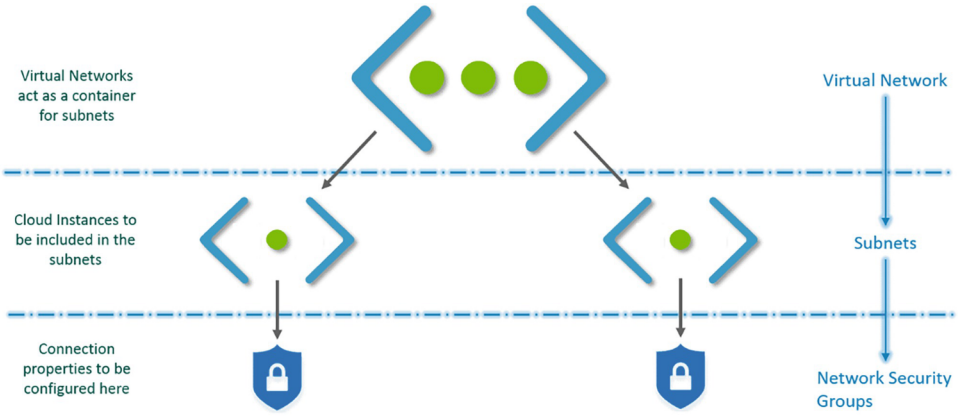

Figure 3.11. *Segmentation of virtual networks and subnets in Azure. [29]*

In Azure, most of the network traffic controls are configured inside Network Security Groups (NSGs), which are created inside a resource group. NSGs can be attached to subnets or resources' network interfaces, such as the network interface of a VM or a container. NSG contains a list of Access Control List (ACL) rules, which apply to the resources inside the security group. They are used to either allow or deny access to a given destination from a given source. The ACL rules have a priority number to describe the order, in which they are taken into consideration. A lower priority number gives higher priority to the rule, and it overrules a higher priority number. [39]

One example architecture of a virtual network configuration in Azure is presented in the figure 3.12. This represents a possible architecture for a web application running in the cloud, with one VM handling business logic, and a SQL database for storing the informa-

tion. First of all, there is one *public subnet* and one *private subnet*. The publicity of these subnets is defined by the NSGs, which are attached to them. On the public subnet, inbound traffic is allowed to the load balancer, which forwards it to the VM. Outbound traffic from the NSG is allowed only from the NAT Gateway. The NSG on the private network denies all traffic, except for inbound traffic from from the load balancer. Outbound traffic is allowed through the NAT gateway, which translates the private IP of the requester to a public one. Traffic outside the private network is allowed only to the VM. Thus the SQL Database is only accessible from within the private subnet. The only machine with direct access to the SQL Database is the VM, and other communication is prohibited.

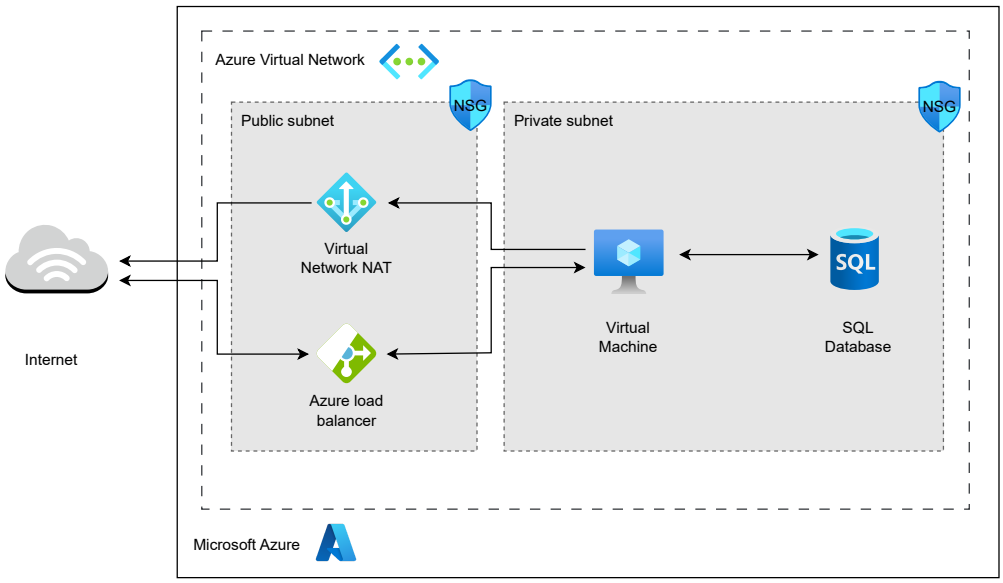

Figure 3.12. *Example of an architecture featuring virtual networks, where the public load balancer makes a connection possible to a VM inside a private subnet.*

In the figure 3.12 traffic from the internet to the VM is routed through a public Azure Load Balancer, which translates public IP addresses into private IP addresses within the virtual network. Load balancers can be either public or internal, depending on whether they are used for load-balancing traffic within a single virtual network or traffic from an outside network to the given network. This makes it possible to create an inbound connection from the internet to the VM. In the public subnet, there is also a NAT Gateway to translate the VM's private IP address to a public one for outbound connections. [42]

3.8 Monitoring

Monitoring is the practice of measuring and collecting data from sources related to our environment, and analyzing it against specific metrics and thresholds to verify, that the cloud is fully available and working properly. Monitoring can also be used to detect anomalies in resource states, connections, or user activities. This way monitoring helps to maximize the availability and performance of the environment and its resources. With the

three biggest CSPs, the security monitoring is largely managed by security monitoring services; Azure Monitor in Azure, Amazon CloudWatch in AWS, and Google Cloud Operations Suite in Google [43].

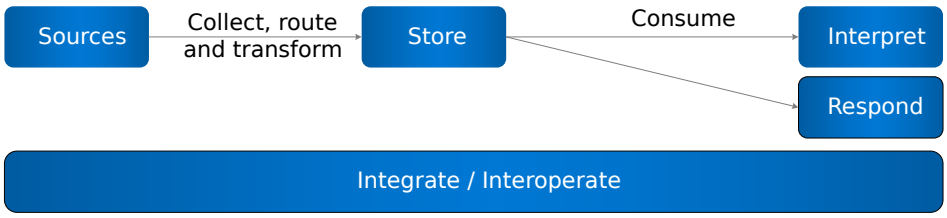

Figure 3.13. *A high-level view of the monitoring process in Azure [44]*

Security monitoring is inherently a proactive method for identifying and responding to threats and vulnerabilities within the cloud service before they take place. Monitoring resources and workloads measures, that they are functioning as expected and they are in a healthy state, meaning that they are not running out of memory or storage for example. Monitoring can also be done against a set of regulatory policies to check if the cloud environment is compliant with the policies. These policies can be set by the CSP or the vendor, and they can for example represent the environment's compliance to different regulations.

Security monitoring is also reactive, as it collects data from events, that have happened, to analyze and produce alerts based on the triggers set. These alerts can be reacted upon with either automatic or manual measures. Monitoring user activities can mean analyzing suspicious actions to identify stolen credentials or mischievous acts. Monitoring of user activities should be done in the name of increasing transparency and accountability, not for spying on employee actions. [45]

Monitoring in Azure

Azure Monitor is the default monitoring service in Azure. It collects data from cloud resources, infrastructure, and custom sources to a centralized monitoring service. It covers collecting data, data analysis, alerting, and data visualization. As soon as an Azure resource is created, Azure Monitor begins collecting metrics and activity logs from it, which can be viewed in the Azure portal from the resource or the Azure Monitor service. Depending on the resource, other data collection may be automatically enabled. [44]

A Log Analytics workspace is a unique environment for storing log data from Azure Monitor and other Azure services, e.g. Microsoft Cloud Defender. In addition to collecting activity logs and metrics by default, one can configure more resource-specific data collection through the resource's *Diagnostic settings*. Azure monitor uses the Log analytics

workspaces for further analysis and visualization. From a security perspective, it is important to make sure logs (data plane) and audit logs (control plane) are collected and stored in a controlled manner. [46, 47]

Through Azure Monitor, alerts can be configured based on different metrics. For instance when containers are not available or CPU usage exceeds a certain threshold. These alerts can be used for preventing or minimizing application downtime and reducing availability issues. In addition, they can be analyzed with the help of AI tools in search of anomalies, which could indicate security risks or potential breaches. [48]

4. BUILDING SECURE INFRASTRUCTURE

Separate cloud components and services were discussed in chapter 3, and those are supposed to form a union, which is reliable, efficient, and secure. In this chapter, the building of secure infrastructure will be discussed along with tools and frameworks. The frameworks are limited to the scope of the case study (presented in section 2.2). Then the infrastructure automation is discussed, and Azure's tools for infrastructure automation are introduced.

Evaluation of information security is a complex task. It is ever-changing and needs to consider a huge scope of everything from the hardware of the application environment to the end-users nature. Even though the cloud the responsibility is shared with the CSP, there is still a lot to take care of. And from the attacker's perspective, all they need to find is one unlucky compromised port, through which they gain elevated access. Or the more traditional phishing attack, where the application's security is compromised by the correct password of a user with elevated access rights.

There are many industry standards for information security of software, which consider many aspects of possible risks and threats. As the case study subject is based on a highly regulated industry, finance, it needs to meet high compliance requirements. In this study, the focus will be looking into the best practices set by different industry standards by using guidelines, checklists, and technological tools. Although the standards cover areas well before the actual secure development of applications or technological decisions, in this study they will be limited to infrastructure-related controls, or controls that can be implemented during the infrastructure deployment phase.

4.1 Designing a secure environment in Azure

Microsoft's Well-Architected Framework (WAF) is a set of principles, that are meant to guide towards good cloud architecture in Azure. WAF consists of five main areas, which are: reliability, security, cost optimization, operational excellence, and performance efficiency. For this study, the security aspect of the WAF is relevant for designing a secure environment. The security pillar of WAF consists of design principles, common security patterns, and a checklist of security recommendations for aligning the workload with the Zero Trust model. Another framework from Azure is the Cloud Adoption Framework

(CAF), which includes many guides for different cloud adoption scenarios. [49, 50]

Though WAF and CAF will not be referenced directly in this study, they are worth mentioning, as they are included in the Microsoft Cloud Security benchmark, which is discussed more in detail in chapter 4.5.

4.2 Infrastructure automation

Infrastructure automation is the process of automating the provisioning of infrastructure through means of software tools and methods. With every cloud service model (IaaS, PaaS, SaaS) the responsibility for physical infrastructure belongs to the CSP. However, with PaaS and IaaS there are configurations regarding the environment, such as networking or authentication methods to resources, of which the customer is responsible. With IaaS vendors the responsibility for configurations is more extensive, as the configurations need to be made from the virtualized network level upwards.

Setting up and maintaining complex infrastructure manually leads to situations, where the infrastructure is difficult to reproduce and to document. Manually managing the infrastructure is also time-consuming and error-prone, as a lot of critical configuration steps are done by hand. The order in which the configurations are made can be important for the infrastructure to work correctly, and that can be difficult to retrace later. Scaling the infrastructure later gets harder, as the frequent updates need to be configured many times to different parts of the infrastructure. Duplicating resources by hand needs to reflect on their dependencies, and in large systems IP address management alone can get messy. [51].

IaC tools are created for automating the infrastructure deployment and provisioning by providing a way to describe the wanted infrastructure as scripts. The IaC approach can be either imperative or declarative. Imperative IaC means writing the script as imperative commands, which are executed in the given order. This means, that the IaC developer needs to know exactly, how and in what order resources need to be deployed. In declarative IaC the script is written in a declarative manner, which describes the target state for the infrastructure. In the declarative manner, the details of how and in which order resources are deployed, are left for the CSP to decide. This affects both the creation of new resources, and updating old ones. [51]

IaC scripts can be version-controlled like code through version-control tools such as GitHub or GitLab. This brings infrastructure provisioning to version control and makes rollbacks to previous versions easier. In addition, having the code in version control makes the scripts retraceable, meaning that changes in the scripts can be tracked and located time-wise. If for example problems arise in a resource's configuration, the changes in the resource deployment can be traced and the reasons for the change explored.

4.3 Azure Resource Manager

ARM is the deployment and management service for Azure. ARM can be accessed through different Application Programming Interfaces (APIs): Azure portal (web application), Azure Powershell (terminal), Azure CLI (Command Line Interface), and REST clients. ARM authenticates and authorizes each request forwarding it to the appropriate Azure service. Functionalities initially released through APIs are represented later in the Azure portal. [52]

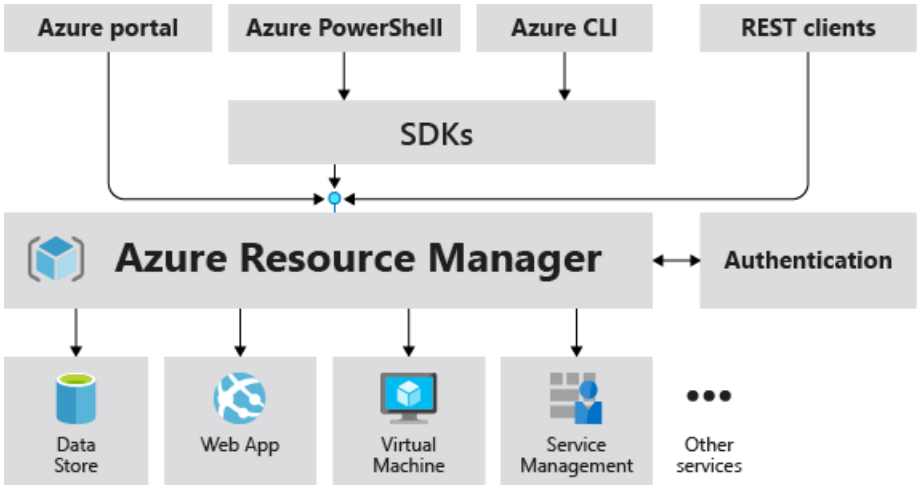

Figure 4.1. *Azure management layer [52]*

ARM templates are an IaC solution for deploying resources to Azure. The templates are JavaScript Object Notation (JSON) files written in a declarative manner. These templates can be deployed into ARM, after which ARM can deploy resources based on the template. ARM templates are idempotent, which means one can deploy the same template many times and get the same environment as a result. The state of the environment (and the necessary changes to it) are managed by ARM. ARM templates are tested for possible errors by ARM to prevent the deployment from failing and ending up in a half-finished state. The difference between the declarative ARM template deployment and multiple imperative REST API calls is illustrated in figure 4.2. [52]

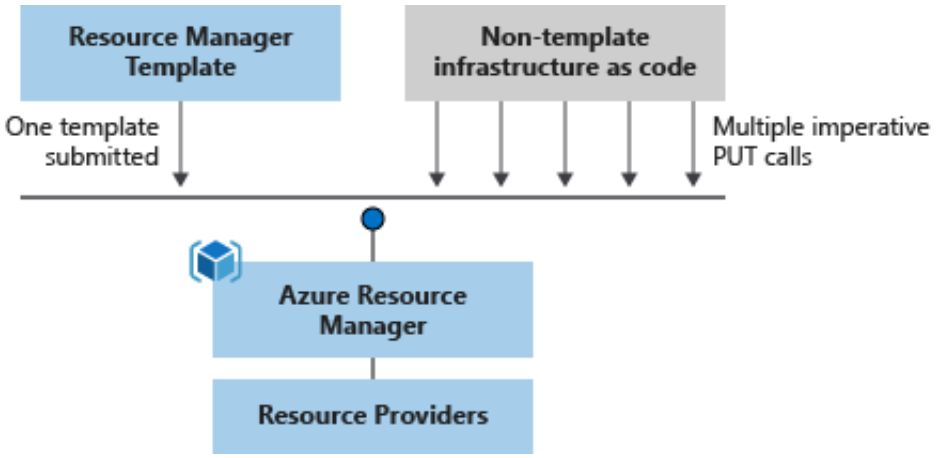

Figure 4.2. *Deployment with ARM templates compared to multiple REST API calls [52]*

An ARM template consists of the following sections: parameters, variables, functions, resources, and outputs. These are declared as attributes of the main JSON object. The sections are described in the table 4.1.

Table 4.1. *ARM template sections [52]*

Section	Description
Parameters	Input values for variables used in the template. These can be either compulsory inputs or optional inputs, where optional ones have configured default values.
Variables	Programming variables, which can be referenced later in the template.
User-defined functions	Custom function declarations, which can be used in the template.
Resources	Declarations for Azure resources to deploy.
Outputs	Values to print out after the deployment. For example domain names or IP addresses.

Azure portal supports exporting ARM templates for existing resources. These are created programmatically, and therefore they are not always practical to use. In addition, JSON format was originally meant for transferring data between machines, not so much for human reading and maintaining. With ARM templates usually consisting of hundreds of lines of JSON formatted text, the authoring experience lacks in format.

When deploying ARM templates, ARM looks into the template for resource declarations and compares them to the existing resources. If the resources have no changes in their attributes, ARM will do nothing to them. If they have changes, ARM will try to update

those attributes. There might be a case, where the attribute of an existing resource is not possible to change without deleting and recreating the resource. This can be for example the location attribute of a resource or the name of an SQL server. Or when the resource acts as a container for other resources, and deleting it would result in deleting its children. In this case, ARM will throw out an error. In these situations, the template's declaration either needs to declare a new resource (to add completely) or the old resource needs to be manually deleted before deployment. If the declaration contains changes to existing resources, it is important to notice, that ARM will not *save* any attributes based on the current state of the resource if there are changes, rather it will base the new resource attributes completely on the template's declaration.

ARM template deployment has two different modes depending on what to do with resources, which are not declared in the deployed template: Complete mode and Incremental mode. In complete mode, all the resources, that are not declared in the template, are deleted. This might be problematic if the environment consists of many IaC templates meant to deploy different resources. The incremental mode leaves the resources outside of the template untouched and only adds the resources declared to the resource group. Incremental mode is the default deployment mode. [53]

Before deploying infrastructure changes, one may be interested in the effects it has on the resources. For this ARM deployments have a flag *What-if*, which is used to preview the changes the deployment is going to make to the state of the infrastructure. When an ARM template is deployed with the what-if flag, the ARM API does not make changes to the infrastructure; it only reports the changes to the resources, which would be changed according to the template and the infrastructure's current state. [54]

4.4 Azure Bicep

Lack of readability in ARM templates gave rise to a new open-source IaC tool: Azure Bicep. Bicep is a Domain Specific Language (DSL) with a simpler format compared to ARM templates. It has been built on top of ARM templates, as it is converted into ARM templates for actual deployment. Bicep has been improved on the aspect of the authoring experience, with the focus being on readability and simplicity of syntax. [55]

Bicep files have achieved parity with ARM templates in terms of functionalities. They are deployed to ARM similarly to ARM templates, where ARM transpiles them into ARM templates. In addition, Bicep files get their resource definitions from the same interface as the ARM templates, so updates on resources to the ARM API are immediately usable in the Bicep templates as well. [55]

Comparing code snippets from ARM and Bicep templates shown in appendix A, one can see how the Bicep syntax differs from the ARM format. The snippets are from the

code editor, Visual Studio Code (VS Code), and use Microsoft's official extensions for colouring the keywords. Both snippets declare a network interface. The Bicep template is more concise, as it has fewer quotation marks, function calls and brackets. In other words, these keywords and characters are included in the syntax of the Bicep language to decrease the amount of *code bloat*. Code bloat considers code, that is perceived unnecessary for the application purpose [56].

4.5 Microsoft Cloud Security Benchmark

The Microsoft Cloud Security Benchmark v1 (MCSB) is a control framework set by Microsoft for cloud security in Azure and AWS. MCSB was rebranded from the Azure Security Benchmark (ASB) in October 2022. It provides security controls categorized into control domains, which present the security principles behind the controls, and the recommended actions to fulfil the principle both in Azure and AWS. Together these are aimed towards the security best practices and recommendations for a cloud environment. MCSB focuses on cloud-centric control areas and thus is a suitable framework for evaluating the case study environment. It takes into consideration Azure's frameworks, such as WAF and CAF, and other industry benchmarks, such as *CIS Controls (v7.1 and v8)*, *NIST SP 800-53 r4* and the Payment Card Industry Data Security Standard (*PCI-DSS v3.2.1*). The MCSB recommendations have been mapped to the corresponding controls by their IDs in the aforementioned industry benchmarks. [57]

First, we will go through the MCSB v1 recommendations list, and limit it to those that fit the scope of the case study; building an information-secure environment with Bicep. There are a few main causes for excluding security controls from the scope of the study, and they are presented in the table 4.2.

Table 4.2. *Reasons for excluding controls from the MCSB v1*

Reason	Description
Too high level	Control is too high level in the management hierarchy, e.g. control considers enterprise-level actions when the scope is the environment of a single product.
Too software specific	Control considers mostly software technical decisions and thus is not in the scope of the project.
Too data dependent	Control is reliant on data for open consideration.

The MCSB v1 controls are presented in the appendix B along with short explanations for exclusion from the process. The criteria relevant to the case study are discussed more in detail. The security principles behind them are explained and their implementation in the POC environment is presented in the section 5.3.

4.6 Microsoft Defender for Cloud

Another tool used for evaluating information security in the cloud environment is a cloud-native application protection platform (CNAPP): Microsoft Defender for Cloud (MDC). One of MDC's features is cloud security posture management (CSPM), which inspects and evaluates the security posture of an Azure tenant [58]. The security posture is evaluated according to Security policies set for the subscription. The security policies come from security standards and Azure recommendations. Security standard is a set of rules, which contain compliance conditions for those rules and actions to take if the conditions are not met. There are ready-made Azure security standards (collections of security policies) for existing security standards, and one can create custom Azure security standards. [59, 60]

MDC calculates a *Secure score* for subscriptions based on security standards on the tenant's Azure subscriptions, AWS accounts and GCP (Google Cloud Platform) projects. By default MDC has one security standard enabled; the MCSB. At the time of writing, there are 235 security policies defined for automated compliance measuring for MCSB. Resources affected by a security policy are checked and deemed either *healthy* or *unhealthy*. Inside a security standard, the security policies are grouped into security controls. Depending on the maximum score of the security control and the amount of healthy and unhealthy resources based on its security policies, the secure score for a security control is calculated according to 4.1: [61]

$$\text{Secure score for a security control} = \frac{\text{Max score}}{\text{Healthy} + \text{Unhealthy}} * \text{Healthy}. \tag{4.1}$$

The secure score of a subscription is calculated from the security scores of the security controls according to 4.2:

$$\text{Secure score for a subscription} = \frac{\sum \text{current score of security control}}{\sum \text{maximum score of security control}}. \tag{4.2}$$

These secure scores can be taken into account when evaluating the security of an Azure environment. And for the MCSB v1 controls they are configured as a security baseline for recommendations by default. Even though there are automated control checks for the MCSB v1 in the form of a security standard, which is enabled for every subscription by default, it is noted in the Azure portal, that the security recommendations alone don't

guarantee compliance. It is up to the customer to evaluate and validate the effectiveness of the controls in a regulatory environment.

5. CASE ECB

This section describes the POC project and the current architecture it inherits from Evitec. First, the current architecture of the ECB product is presented from the AWS environment, after which a corresponding POC architecture for Azure is presented. The POC project aims to find solutions to the research questions, which were introduced in the chapter 2. Key points of the POC environment are discussed in more detail, and then the environment will be evaluated through the lens of the MCSB control list. The list is limited to the scope of the study, which was presented in chapter 4.5

The POC project may and probably will differ from the actual project's end result, but it aims to give insight into how information security could be taken into account in the designing of the infrastructure for the Azure environment.

5.1 Current architecture in AWS

The ECB product is currently hosted on AWS, and its high-level infrastructure is illustrated in figure 5.1. The architecture diagram was largely put together by Matti Valkeinen, who studied its suitability for fitting in multiple different cloud platforms in their master's thesis *Cloud Infrastructure Tools For Cloud Applications* [62]. This architecture is not the complete design, but it illustrates the main components of the architecture. The infrastructure is managed by AWS's cloud-native IaC tool, CloudFormation.

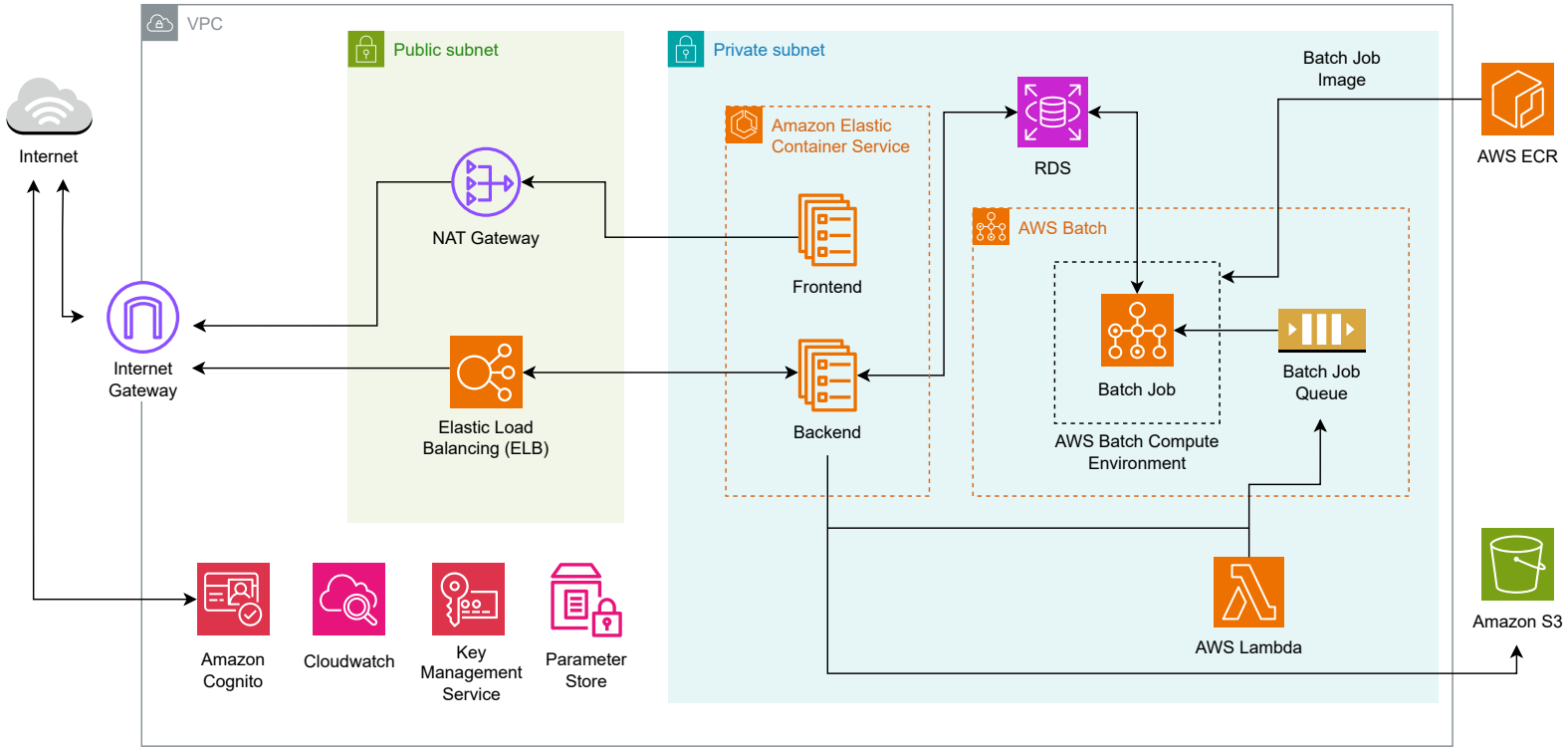

Figure 5.1. *High level architecture of the current AWS environment [62, edited]*

Comparing the AWS resources to the Azure resources, there are a lot of similarities. First off the architecture is wrapped around a Virtual Private Cloud (VPC), which has similarities to Azure's Virtual Networks. Network controls on the subnet level are configured via ACLs, which hold a set of connectivity rules. On the resource level, the traffic rules can be set by Security Groups, which work similarly to Azure's Security Groups. The application is placed inside a private subnet using a container orchestration service, Amazon Elastic Container Service (Amazon ECS), for hosting and managing the containerized application [63]. The concept of container orchestration and competitive container orchestration tools are also discussed in the thesis of Matti Valkeinen. [62]

Inbound traffic to the application from the public internet is allowed through an Internet Gateway, which is placed on the edge of the VPC. It forwards the requests to a load balancer, Elastic Load Balancing (ELB), which in turn forwards the request to the Amazon ECS. Depending on the amount of traffic, there can be multiple instances of containers, to which the network can be distributed. Outbound traffic from the application is handled with the NAT Gateway.

The database solution for the AWS environment is the Amazon Relational Database Service (RDS), which is a managed database relational database service. Access to the database is managed by IAM policies and Security Groups. The resources, which can connect to the RDS are the backend containers from the Amazon ECS and the Batch Job instances. The batch jobs are in charge of financial computing and analytics, which require a lot of computing resources periodically. There are multiple different batch jobs for different causes such as creating reports, calculating analytics, importing data to the database, and exporting reports periodically. Thus the batch jobs are defined as container images in the AWS Elastic Container Registry (ECR). [64]

Authentication in the AWS environment is managed by the Amazon Cognito service, which is the AWS' identity management management service. It manages RBAC for both users and resources through user pools and identity pools. [65]

5.2 POC architecture in Azure

The POC environment is based on the application's current architecture, which was presented in chapter 5.1 alongside the ECB product's high-level AWS architecture and its services. The POC architecture is presented in figure 5.2, and it shows similar traits to the AWS architecture on a high level. The AWS services have been replaced by the most similar functioning services in Azure, and they have been deployed via Bicep scripts. Therefore, the infrastructure deployments are automated fully with Bicep.

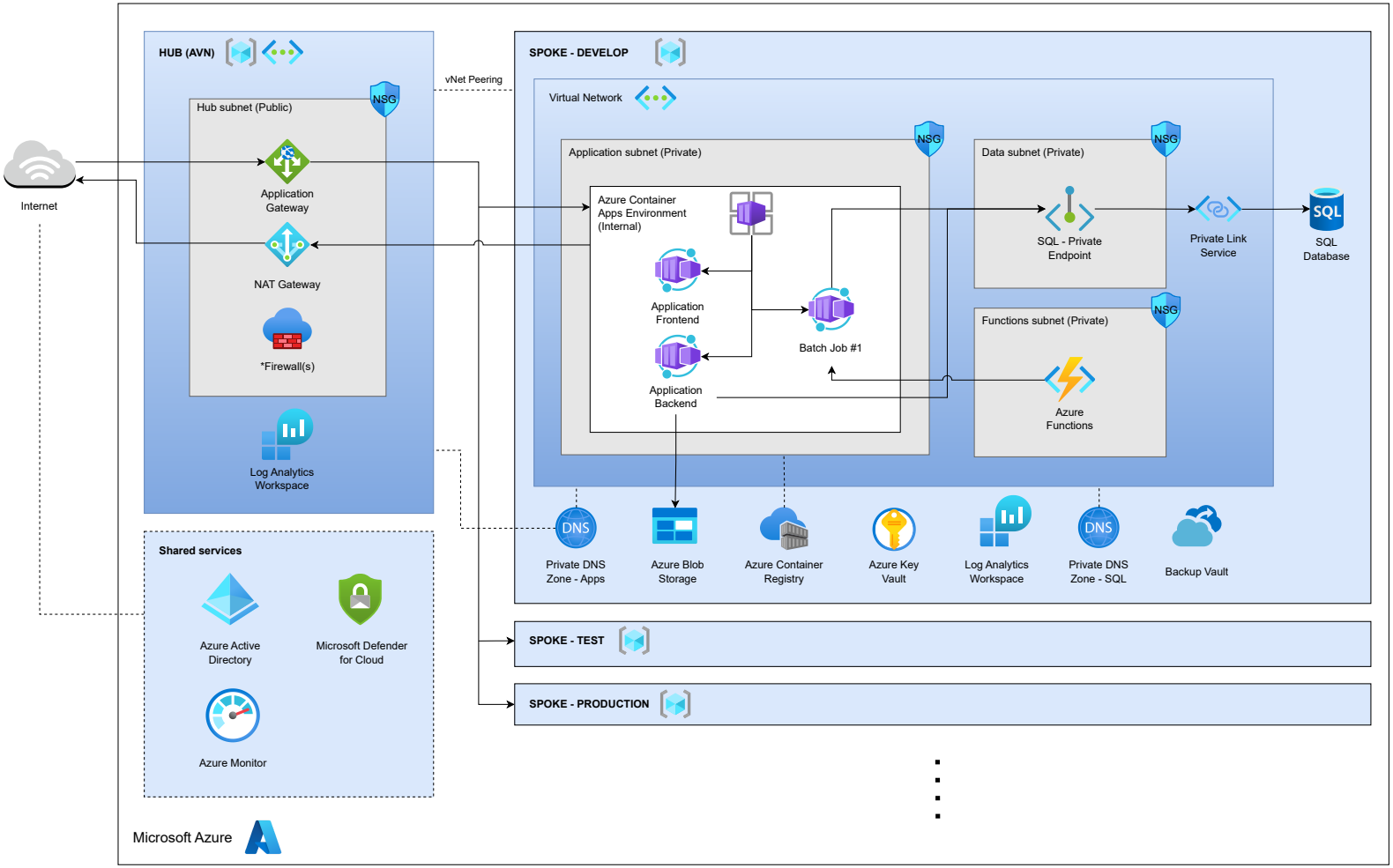

Figure 5.2. *Architecture of the POC environment*

The POC architecture presents the hub and spoke pattern for multiple application environments, which are grouped into resource groups. The resource groups divide the pattern into reasonable groups: Hub, Spoke-develop, Spoke-test, and Spoke-prod, which contain vNets for their resources. In the POC environment, the grouping of resources into resource groups was done to fit the environment into a single subscription (and its budget). The grouping of the application environments could also be changed from resource groups to subscriptions, which would for example benefit managing and optimizing the costs of different environments. The spoke-develop environment might not need the batch jobs block for instance, and this could be removed from the environment to optimize costs.

The subnets and resources in the spoke vNets do not allow any traffic to them from the public internet, which means they are private. The spoke vNets are peered into the hub network, which means that the spoke networks are attached to the hub networks in a way, that allows the resources between them to use the private addresses for communication. This makes it possible, that the Application Gateway can forward traffic to the containers inside a private subnet in the Spoke vNet. Traffic from the public internet is allowed only to the hub vNet's Application Gateway, which forwards it to the spoke vNet's container apps' managed environment. Through the private DNS zone resources inside the spoke network and the hub network can resolve the container app's (either front-end or back-end) private IP address from its domain name.

All of the application logic resides inside the Azure Container Environment. The frontend container serves the frontend code as static content and the backend container runs the backend API, which responds to HTTPS requests from the frontend app run in the client's device. These are accessible from the public internet through the Application Gateway, which has the WAF enabled for edge network protection. In addition to the WAF, the Azure Virtual Network Firewall could be enabled for further protection.

In addition to the frontend and backend containers, container jobs will be running inside the container apps environment. These containers act as the application's batch job platforms. These app jobs can be triggered either manually, by a schedule or by an event. Sharing the same container app network allows the container apps and container jobs to share networking and logging configurations. One possible way of triggering the containers is through Azure Functions, which can enable additional events and trigger options for more flexible use. [66]

The SQL Database is in the most restricted corner of the networking architecture; it is accessible through a private endpoint, which is located in its own private data subnet. Connections to this subnet are allowed only from the container app environment's (integrated) subnet. The containers that need access to the SQL Database are the backend container and the batch job containers. Access to the SQL Database is granted to the

containers via their assigned system identities using Microsoft Entra Authentication.

The logging is centralized on the spoke level, and the logs are gathered inside a shared Log Analytics workspace. The spoke resource's audit logs and resource metrics can be queried and further analyzed from there through the help of Azure Monitor. In addition, the networking traffic metrics are collected there from the flow logs produced by the spoke NSGs. [67]

5.3 Evaluation of MCSB controls in the POC environment

Although the Microsoft Defender for Cloud has built-in policies for checking compliance with the MCSB controls, automatic controls don't cover, how to best address the design questions and processes. The purpose of this inspection is to address the controls and the security principles behind them. In this section, the MCSB controls included in the study's scope are presented and established, as to how they comply with the controls.

5.3.1 Network Security

MCSB's Network Security covers controls to secure and protect networks and networking both inside the virtual network and outside it [68]. This might be one of the biggest aspects of security for the application, as it defines, how and from where the resources can be accessed. It is important to recognize, what network interfaces are facing the public internet and what are only accessible from within the virtual network or the subnet.

NS-1 Establish network segmentation boundaries

Security principle This control enforces network segmentation into virtual networks and subnets. Workloads storing and processing highly sensitive data should be isolated into separate virtual networks. NSGs should be used to implement traffic controls inside networks.

Implementation The virtual network division is visible in the high-level architecture in the figure 5.2. The hub and spoke -pattern separates each application environment into its separate virtual networks. In addition, subnet-level segmentation is done inside the spoke networks to separate the application subnet from the database subnet. NSGs are attached to each subnet to define its publicity and connections allowed. A simplified networking graph is illustrated in figure 5.3.

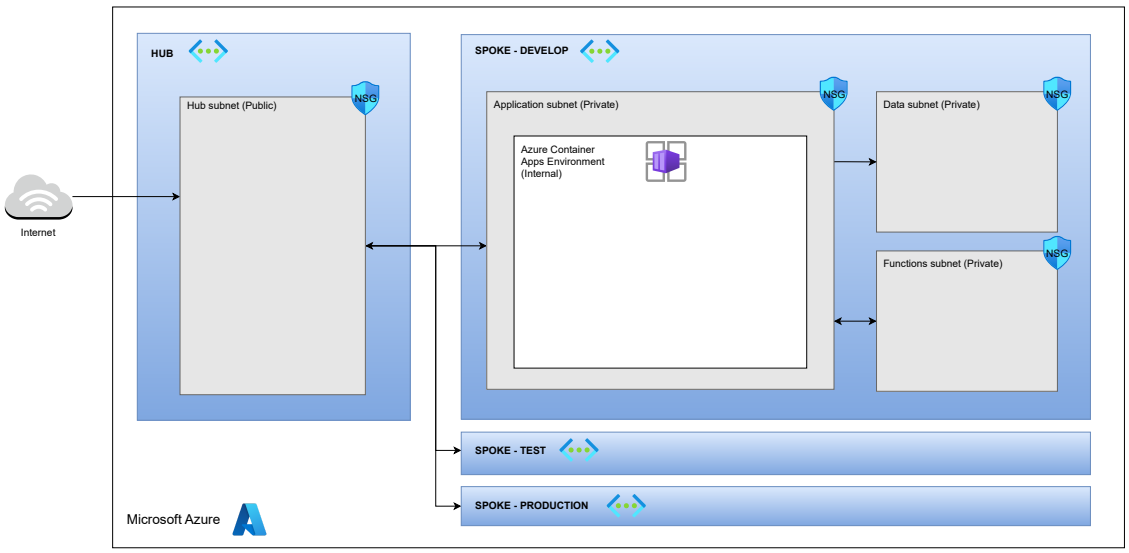

Figure 5.3. *POC environment's networking on subnet level.*

For private subnets, all connections outside the virtual network are denied, and as an exception, only the necessary gateway connections are allowed. As the hub network is peered to the spoke-networks, it essentially means that the traffic between is moving inside the Azure backbone using private IP addresses. For public subnets traffic from everywhere is essentially allowed, but only to the gateway components' resources.

NS-2 Secure cloud-native services with network controls

Security principle Secure cloud services by establishing private access points for resources and restrict public network access when possible.

Implementation Private endpoints are established for the SQL server, Key vault and storage account. In other components, the traffic is kept inside the virtual network using private IP addresses and private DNS zones. Traffic from the public internet is limited only to the hub component. The application containers (frontend and backend) are accessed through the application gateway, which forwards the requests into the container apps environment. The container apps environment is hosted internally within the spoke-vNet, and the containers are accessible only through the private DNS server inside the spoke-vNet and the peered hub-vNet.

NS-3 Deploy firewall at the edge of enterprise network

Security principle Deploy a firewall at the edge of the enterprise network for advanced filtering of network traffic.

Implementation Azure Firewall would deployed to the hub network, and traffic controls set up for necessary internet traffic. The firewall provides L3-L7 level filtering for traffic, which covers protection for attacks from the networking layer to the application layer in the communication model. [69]

Notes The Azure Firewall is an expensive tool for the POC budget and therefore left out of the POC environment implementation. In addition, depending on the CSP model of the product, even the Hub vNet might be already a part of inner networking, in which case the firewall would be implemented on the edge of the outer network, and therefore the responsibility of the IaaS vendor.

NS-6 Deploy web application firewall

Security principle Deploy a web application firewall (WAF) and configure the appropriate rules to protect web applications and APIs against application-specific attacks.

Implementation A WAF is established inside an application gateway in the hub network's public subnet, which forwards traffic to the container network running the application. WAF is a tool for centralizing security management of the application, and provides security against the *OWASP Top Ten* most critical security risks to web applications (The Open Worldwide Application Security Project) [70]. From here the DDoS protection could be turned on for application-level protection against DDoS attacks. [71]

NS-7 Simplify network security configuration

Security principle Use tools to manage complex network environments for simplicity and centralization.

Implementation This control could be best fulfilled with the use of Azure Virtual Network Manager (VNM), which is assigned to a vNet. Through the VNM one can create virtual network groups, which group together vNets for configuring network topologies and security controls. In addition to the hub and spoke topology, mesh topology is also supported.

Notes In the POC project, there are two networks implemented, the hub network and a spoke network, which are peered directly from the vNets' configuration. In the case of multiple spoke networks, this would be more complex to set up, and would not scale well because of avoiding overlapping IP address spaces within the hub network.

NS-9 Connect on-premises or cloud network privately

Security principle Use private connections for secure communication between different networks.

Implementation The hub and spoke topology is linked through vNet peering, which means that the traffic between private resources stays inside Azure's network. Virtual network peering also supports peering remote networks into the virtual network, creating a secure tunnel for communication between the networks. A VPN tunnel could also be established with the help of an Azure VPN Gateway.

Notes VPN Gateways to remote networks are not implemented in the POC environment.

NS-10 Ensure Domain Name System Security

Security principle Ensure that DNS security configuration protects against known risks. From the scope of the case study, this considers isolating private DNS resolution from the public network.

Implementation Private DNS zones are established for the SQL server and the container apps environment. The private DNS zone for the Container Apps environment is linked to both the spoke network and the hub network. This way the application gateway, in the hub network, can resolve the private DNS records for the container IPs. The private DNS zone for SQL is only linked to the spoke network, as it is only needed for the communication between the SQL Server and the backend container.

5.3.2 Identity Management

Identity management controls cover establishing secure identities and access controls using identity and access management services. This includes machine identities for resources and users requiring authentication. Authorization should be based on role assignments or group memberships. [72]

IM-1 Use centralized identity and authentication system

Security principle Use a centralized identity and authentication system for governing identities and authentications for cloud resources and non-cloud resources.

Implementation ME-ID is used for identity management and authentication for both human users and machine identities. In the POC project's scope, this considers access to

cloud resources and Azure SQL Server. Cloud resources have currently one infrastructure administrator (the thesis writer) authorized for reading and making changes. More would be authorized for making changes in the event of multiple infrastructure administrators, and developers could be granted reading rights for the resources. The backend container is granted reading and writing rights for the application database inside the Azure SQL Server.

IM-3 Manage application identities securely and automatically

Security principle Use managed application identities instead of creating human accounts for applications.

Implementation The POC environment uses both system-assigned identities and managed identities for accessing resources. These identities are assigned on deployment, so the Bicep script deployer needs to have permission to manage roles and identities on the subscription level. Database permissions are granted to the backend container and the batch job container(s) through their system-assigned identities. The container images located in the private ACR are fetched using managed identities permitted for the *image pull* action.

Notes An interesting problem arose when the container app deployment needed an identity for the *image pull* operation during deployment. This identity could not be system-assigned, as the identity needed an assigned role with permissions during the actual deployment. This was resolved by assigning the *image pull* role to a managed identity, which was then assigned to the container app during deployment. Henceforth the container app had an identity with sufficient permissions during deployment.

IM-4 Authenticate server and services

Security principle Authenticate remote servers and services from your client side via the server's certificate and a trusted certificate authority.

Implementation The Application Gateway uses a certificate from the Key vault for TLS connections. The certificate is provided by a trusted certificate authority and will act as a signature to prove the authenticity of the remote server.

IM-6 Use strong authentication controls

Security principle Enforce strong authentication controls such as strong passwordless authentication or multi-factor authentication (MFA).

Implementation MFA authentication is required for ME-ID sign-ins. This covers both developer/administrative access through Azure Portal and application user logins.

Notes The customer, Evitec, has a tenant-wide policy set for requiring MFA access for its users, which enforces this control tenant-wide.

IM-8 Restrict the exposure of credentials and secrets

Security principle Ensure that application developers securely handle credentials and secrets.

Implementation In the POC environment, the credentials and secrets are stored inside spoke-specific key vaults. This includes secrets needed in the deployment phase or resources. From the application's perspective, the backend container is authorized through its system identity to access the key vault secrets. In the event of integrations with third-party services, those credentials can be stored into the key vault, and used by the application from there.

5.3.3 Privileged Access

Privileged access covers controls concerning privileged access to the tenant or the resources. This considers administrative use and privileged access workstations. This section has been ruled out of the POC environment inspection, as it is primarily higher on the abstraction level managing tenant-level policies and actions. [73]

5.3.4 Data Protection

Data protection controls cover data protection at rest and in transit. It begins with identifying and classifying data according to its sensitivity levels for actions. In addition, it enforces monitoring of data use and auditing. [74]

DP-3 Encrypt sensitive data in transit

Security principle Protect the data in transit using encryption, especially in external and public network traffic.

Implementation HTTPS connections are required in communication from the client to the application gateway. The SQL Servers require a minimum TLS version 1.2 to encrypt data in transit. Secure File Transfer Protocol (SFTP) is required for file transfers with the Storage account.

DP-4 Enable data at rest encryption by default

Security principle All the data at rest should be encrypted by default, as this makes it harder for the attacker to read or modify the data.

Implementation Services in the POC environment, in which encryption at rest is enabled by default are the Azure Container Registry, Azure SQL Server (configurable on the database level), the Azure Key vault, the Azure Monitor (including the Log Analytics Workspace), and the Azure Storage. It is important to notice, that if there are Virtual Machines in the environment, their data disks aren't necessarily encrypted at rest by default. [75]

DP-8 Ensure security of key and certificate repository

Security principle Ensure the security of the key vault service through the means of access control, network security, logging, monitoring, and backups.

Implementation The key vault in Spoke requires ME-ID authentication and is accessible only within Spoke's network through a private endpoint. The key vault's metrics from both the data lane and control lane are collected to the spoke's Logs analytics workspace configured via a diagnostic setting. Soft-delete is enabled for the key vault, which protects its data from unauthorized or accidental deletion.

5.3.5 Asset Management

Asset management covers controls to ensure security visibility and governance of cloud resources. It includes tracking of assets and access management to cloud resources. It is limited out of the study's scope as too high level, as it is not considered to belong to the level of a single project. [76]

5.3.6 Logging and Threat Detection

Logging and threat detection covers controls for collecting logs for conducting analysis for threat detection. Audit logs are collected to keep track of changes and who made them. [77]

LT-1 Enable threat detection capabilities

Security principle For supporting threat detection scenarios monitor all known resource types for known and expected threats and anomalies.

Implementation Microsoft Cloud Defender has service-specific offerings for the following services in the POC environment, The main components in the POC environment and their corresponding service-specific support from Microsoft Defender for Cloud are presented in figure 5.1.

Table 5.1. *Microsoft Defender for Cloud support for POC environment main services, * marks partial support*

Service	Microsoft Defender support
Azure SQL Server	Microsoft Defender for SQL
Azure Storage Account	Microsoft Defender for Storage
Azure Key Vault	Microsoft Defender for Key Vault
Azure Container Registry	Microsoft Defender for Containers*
Azure Resource Manager	Microsoft Defender for Resource Manager
Azure DNS	Microsoft Defender for DNS
Azure Functions	Microsoft Defender for App Service*
Azure Container Apps Environment	Not supported

These services can be enabled for additional threat detection features related to the services. Even though there is a service called *Microsoft Defender for Containers* it supports mostly Kubernetes Service (on Azure and elsewhere), but it does not support the Container Apps service used in the POC environment. [78, 79, 80, 81, 82, 83, 84]

Notes These additional threat detection capabilities are not enabled in the POC environment, as only enabling them does not bring much to the table for the study.

LT-3 Enable logging for security investigation

Security Principle Enable logging for cloud resources to meet the requirements for security incident investigations and security and compliance purposes. This covers the logging of activities both in data and control planes.

Implementation Diagnostic settings are configured for resources for collecting data from the resource's Activity Log (control plane) and resource-specific logs (data plane). These logs are collected to a workspace in the Spoke's resource group, which is located in the Spoke's resource group. From there the Azure Monitor and MCD can use the workspace for their functions.

LT-4 Enable network logging for security investigation

Security Principle Enable logging for network services to support network-related incident investigations, threat hunting and security alert generation.

Implementation Diagnostic settings are configured for NSGs to collect resource logs. NSG flow logs are created for NSGs inside Hub and Spoke networks for collecting log data from ingress and egress traffic. The logs from the application gateway are collected to a Log Analytics workspace inside the hub network, and these also include WAF logs.

5.3.7 Incident Response

Incident response covers controls in the incident response life cycle - preparation, detection and analysis, containment and post-incident activities. This set of controls is deemed too high level for the scope of the project's consideration and therefore left out of it. [85]

5.3.8 Posture and Vulnerability Management

Posture and vulnerability management requires defining the security baselines and controls to enforce them. It covers tools for continuous evaluation of the cloud security posture and security configurations through configuration tracking, reporting and scanning. [86]

PV-1 Define and establish secure configurations

Security principle Defining the security configuration baselines for each resource type in Azure. Alternatively one can use configuration management tools for automatic evaluation against a certain baseline.

Implementation The MCSB v1 is defined as the configuration baseline for the POC environment. The resource-specific security baselines are also documented on Azure for every resource.

PV-2: Audit and enforce secure configurations

Security principle Enforcing continuous monitoring and alerts for deviations against the predefined security configuration baselines. For cloud resources, one can set controls to deny the deployment for a non-compliant configuration.

Implementation As aforementioned, the MCSB controls are enabled in the POC environment by default. MCSB controls are a group of Azure policies, called an *Initiative*. Other built-in initiatives can be used as security configuration baselines, such as *CIS v2.0*

Azure Foundation Benchmark or *ISO 27001:2013* for example [87]. One can use built-in Azure policies or initiatives, or create their own for enforcing their security configuration baselines. [88]

5.3.9 Endpoint Security

Endpoint security controls cover in endpoint detection and response and anti-malware detection service for cloud environments [89]. These controls are left out of the case study due to being too software-specific.

5.3.10 Backup and Recovery

Backup and recovery controls ensure that data and configuration backups for different services and resources are performed, validated and protected. [90]

BR-1 Ensure regular automated backups

Security principle Ensuring backups from business-critical resources are made and enforced in resource creation through policies.

Implementation The critical resources requiring backups in the POC environment are the SQL Database(s) and the Blob Storage. The SQL Database backup configuration for STR backups is set to a week (PITR possible for 7 days) and LTR backups are made monthly. For the Blob storage, the continuous PITR-enabled backups are configured to 30 days and the periodical Backup vault copies are made every 30 days.

BR-2 Protect backup and recovery data

Security principle Ensure that the backup data and operations are protected.

Implementation Backup vaults are automatically encrypted with 256-bit AES encryption. Azure RBAC is used for backup operations by assigning the role *storage account backup contributor* to the storage account. Soft-delete is enabled for the Backup vault to safeguard data against accidental or malicious deletion. Soft-delete keeps a record of deleted Backup vaults, which can be restored for up to 14 days.

Notes One interesting notion is, that the forming of LTR backups for the Azure SQL Database does not seem to support manual triggering or custom scheduling. This is problematic, as banking days are important to follow in the making of backups, and they are affected by national holidays. In the case that the LTR backups don't support a way for customizing the schedule by a holiday calendar, the Azure SQL Database might be

unsuitable for the ECB product.

5.3.11 DevOps Security

DevOps (development and operations) security covers the controls related to the security engineering and operations in the DevOps processes. However this is deemed too software-specific for the study's scope, so it is left out of the study. [91]

5.3.12 Governance and Strategy

Governance and strategy cover guidance for defining and implementing various security strategies for establishing responsibilities, unified technical strategy and supporting policies and standards. However, these are deemed to be too high level to be included in the scope of the study. [92]

6. RESULTS

This chapter presents the results, which came from the case study presented in chapter 5. The information security of the environment is discussed along with the role of the MCSB controls in the security evaluation. In addition experiences from Bicep as an IaC tool are discussed.

6.1 Information security of the POC environment

The architecture for the environment in Azure (figure 5.2) holds all the key components compared to the AWS architecture (figure 5.1). The infrastructure responds to the needs of the ECB product, which resembles an architecture for a typical web application with periodic batch jobs. MCSB was suitable for the security evaluation of the POC environment because it is a security control list specified for Azure. Therefore it holds controls and recommendations, which are directly manageable from Azure. It is also more compact and concrete compared to other industry standards.

In the Azure portal, the Microsoft Defender for Cloud gives a secure score of 79% for the environment. Furthermore one can see, from which recommendations the missing 21% comes and what recommended actions to take. In this case, 14% is related to SQL vulnerability assessments; their configuration and the resolving of their findings. Another potential 7% increase in score would come from enabling DDOS protection for virtual networks.

Azure portal also shows the regulatory compliance to the MCSB v1 controls through the automated controls and their assigned policies. It states, that there are 47 passed controls out of the total of 63 controls for the MCSB. As a percentage, this means that 74.6% of the automated control checks are passed in the POC environment. This is quite a high percentage, even though the study included only 22 controls out of the 86 available in the MCSB v1, which is about 20% of the MCSB controls. It can be interpreted, that many of the controls outside the study scope were neither practical nor possible to supervise with automated control checks. Such as MCSB controls related to the section *identifying and classifying data*, which cannot be measured through automated tests. Many of the controls in MCSB v1 involved defining processes or strategies both at the project level and at the organization level, which are not involved in technical security decisions.

However, the automated controls include many resource-specific Azure policies, which could be inspected more thoroughly. For example, for the control *NS-2 Secure cloud services with network controls* there are 32 automated assessments (Azure policies) only for the Azure environment. Furthermore, it covers 16 assessments for AWS and 12 assessments for GCP environments. These assessments contain various resource-specific configuration checks, which may pose a threat, if not configured correctly.

In the case study 22 controls from a total of 86 controls in the MCSB v1 were included for manual evaluation and validation. These were limited to the study's scope, which focused on the secure infrastructure. The MCSB v1 controls and the exclusion reasons are presented in the table 4.2.

In figure 6.1 the MCSB controls included in the case study are presented compared to the total number of controls. They are grouped per section so that they depict the coverage of the section's security controls. From the figure, it can be seen, that the section *Network Security* was the biggest factor in the POC environment's security. In the MCSB list, it holds the second most controls (10), while the section containing the most controls is the section *Governance and strategy*. However, in the case study 7 controls out of 10 were chosen for inspection from *Network security*, while all of the controls in *Governance and strategy* were excluded from the study for being too high level. This seems reasonable, as *Network security* is closely tied to infrastructural decisions, while *Governance and strategy* holds mostly defining and implementing of organisation-level processes and strategies.

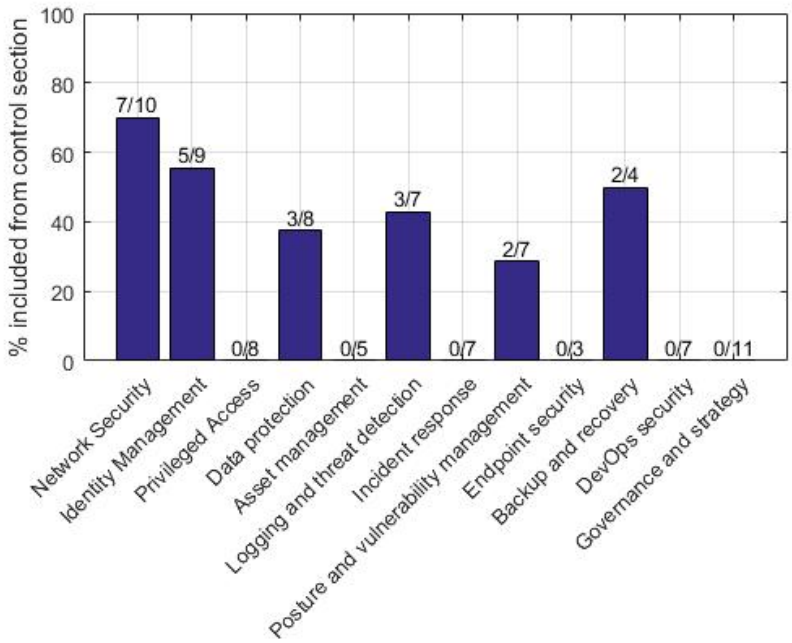

Figure 6.1. *Coverage of MCSB control sections. The bars display the proportion of the included controls over the section's controls. Above the bars is the section's number of controls included compared to the section's total number of controls.*

The most important control sections in the case study can be measured by the number of controls chosen in the study. By this definition, the most important control sections were *Network security*, *Identity management*, *Data protection* and *Logging and threat protection*. Together these sections covered 83% of the security evaluation in the case study. The control sections by the controls chosen for the study are presented in the figure 6.2. The MCSB control sections, which had zero controls selected for the study were left out of the figure.

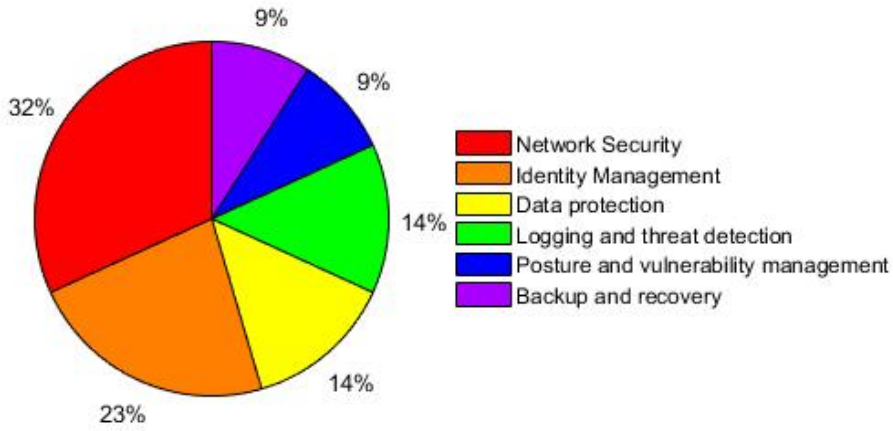

Figure 6.2. *The number of MCSB controls by section selected for the study. The sections with no controls were left out of the figure.*

6.2 The process of creating Bicep scripts

The POC environment's infrastructure (illustrated in figure 5.2) was deployed with Bicep scripts. The Bicep scripts were created from scratch, and they took a considerable amount of time compared to deploying through the Azure portal. However, using the Visual Studio Code editor and Microsoft's extension for Bicep sped up the coding process, as it gave information and explanations about the required and optional attributes for every resource.

The Bicep scripts were created using the Common Azure Resource Library (CARML for short), which is a community-driven (open source) library for reusable modules both for ARM and Bicep base deployments, and tools related to deployments like pipelines and useful Powershell scripts. The CARML modules act as an abstraction layer between

configuring the parameters and deploying the actual resources. They smooth out the configuration needed for a resource by using common configuration patterns and assert secure default values for optional resource configuration attributes. Furthermore, they can deploy multiple related resources at once, which wouldn't be meaningful to deploy separately. The POC environment's Bicep scripts act like module wrappers, which pass configuration parameters to the CARML modules. [93]

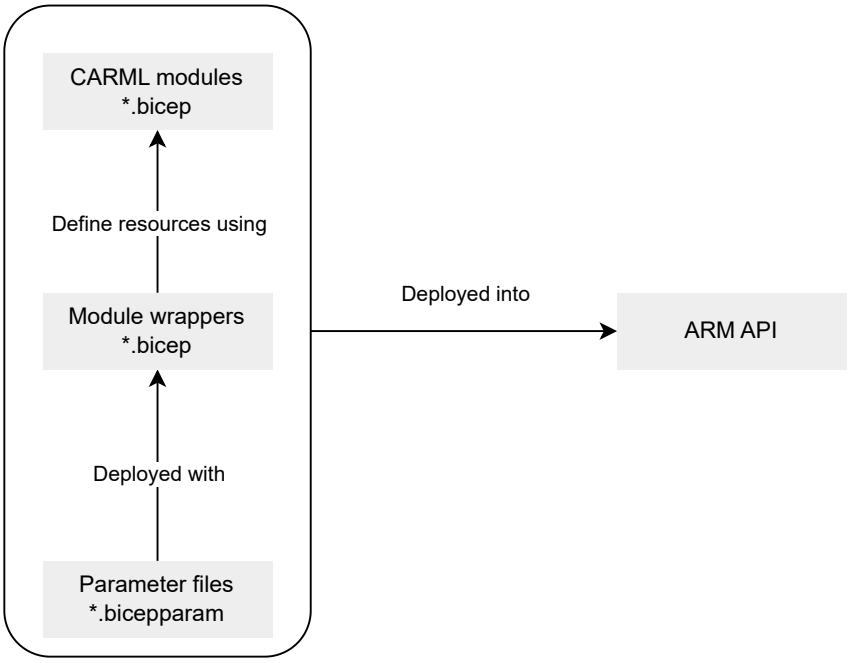

Figure 6.3. *The Bicep scripts for the POC environment consist of Module wrappers, CARML modules and parameter files.*

There were some issues with using the modules from CARML. In the beginning, it was updated in the project regularly, which led to problems due to the library not being functional at times. There were problems with files and file paths being renamed somewhere, which led to errors in the modules' reference paths, and caused some modules to be unusable. However, this is better managed by updating the library in phases and using versioning on different module wrappers and modules, so they can be updated separately.

At the beginning of the project, there was a plan, that the different application environments (development/testing/production) could be defined using a single parameter file. However, the bicepparam files supported only 1-1 relations between the bicep file and a parameter file. This resulted in creating a parameter file for every bicep module wrapper. This isn't all bad, as the configurations are in their respective files. However, common shared parameters were repeated in multiple parameter files, which means changing one shared parameter would require changing in multiple locations. Bicep does support JSON loaders for parameter files, which would make it possible for a single (huge) parameter file per environment, from which the separate Bicep files could parse only the parameters

required for its deployment. This would unify configuring shared parameters, such as the resource group name and location, but the parameter file would be quite long, and the JSON format isn't too convenient for reading.

7. SUMMARY

The goal of this thesis was to create an information-secure environment in Azure for the ECB product and take advantage of the modern IaC tool Bicep while doing so. In the thesis, a POC environment is created for security evaluation. The MCSB v1 was chosen as a security standard for the evaluation. It is limited to the controls related to infrastructure security and single project level. This forms a list, through which the security of a cloud environment should be implemented through the deployment of infrastructure.

The first part of the thesis goes through fundamental security areas and concepts, which are needed both in and out of cloud environments. Along with the high-level theory, Azure services are brought along to give concrete examples of configurations and to define important tools needed from the security perspective. Concepts introduced included service models, control and data planes, Azure services and resources with their technicalities.

The second part covers the frameworks and tools used for establishing information security in the cloud. The frameworks include WAF and CAF, which the MCSB enforces. Evaluating information security is not a straightforward task, however by limiting the MCSB control list to its infrastructural aspects, one gets a list that is possible to implement through infrastructure deployments. Thereafter Bicep is used for actual deployments of infrastructure in an iterative manner for achieving the wanted POC environment.

The third part of the thesis is the case of Evitec. The current environment of the ECB product in AWS is introduced with its AWS services. Then the POC environment with corresponding features is displayed and analyzed in light of the relevant MCSB controls. This way the resource configurations of the environment are based on the best practices and recommendations on Azure. With Microsoft Cloud Defender the environment's security baseline is monitored and evaluated, and the recommendations help to keep the environment secure.

As a whole the thesis provided answers to the questions of information security of an environment in the cloud; how to design it and how to evaluate it. The MCSB list is a concrete tool for keeping the environment in check. It needs to be noted, that the list comes from Azure, and many of the recommendations increase the costs of the environment in the form of subscribing to additional services. Security has a price tag and further cost estimation for it is needed. This is left out of this study, as it is a topic on its

own, but for example, the control *LT-1 Enable threat detection capabilities* recommended to turn on additional service-specific offerings for different services. There are over ten additional services, which range in cost from $5/Month (*Defender for Resource Manager*) to $15/Month. Together these services can increase the subscription cost by $102/Month.

Bicep did well as an IaC tool, as the deployments flowed well, and the errors were mainly pretty easy to solve. It was quickly discovered, that nested resources need to be deployed inside the parent resource. For example, a problem will arise, if a virtual network is declared as empty, and then the subnets declared after it separately. It will work fine for the first time. But with the upcoming supposedly *idempotent* re-deployments the ARM API will try to delete the subnets within the virtual network and recreate them, even when their properties haven't changed. This was solved by passing the subnet declarations as the virtual network properties, and the subnets passed on as its properties. This way the ARM API could handle the virtual network deployment correctly and no longer tried to recreate the subnets on each deployment.

As an Azure native tool and with Microsoft's VS Code extensions for it, the writing of Bicep scripts was pretty convenient. The most time-consuming part was studying the properties and their mappings through the CARML modules to the resource's final state in Azure. From an IaC novice developer, it can be said, that the Bicep syntax was easy to learn and use. However, the module wrappers included some programmatic logic like loops and conditionals, which makes them more flexible and reusable. Still, the Bicep scripts created are not as reusable as expected. Parts of them can be used in other environments, but as a whole, they cannot be used through changing parameters to fit deploying different environments. But hopefully, they may have an impact on infrastructure deployments for ECB products in the future.

REFERENCES

[1] Evitec. *Evitec Covered Bonds*. 2022. URL: https://evitec.com/solutions/evitec-covered-bonds/ (visited on 03/18/2023).

[2] Shona McCombes. “What is a case study? | Definition, examples & methods”. In: *Scribbr* (June 2023). URL: https://www.scribbr.com/methodology/case-study/.

[3] Eyal Estrin. *Cloud Security Handbook: Find Out How to Effectively Secure Cloud Environments Using AWS, Azure, and GCP*. Packt Publishing, Limited, 2022.

[4] Jason Andress. *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*. Syngress, 2014.

[5] Microsoft. *Shared responsibility in the cloud*. 2022. URL: https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility (visited on 03/18/2023).

[6] Azure. *What are Azure management groups?* 2023. URL: https://learn.microsoft.com/en-us/azure/governance/management-groups/overview (visited on 09/06/2023).

[7] Microsoft. *What is Azure role-based access control (Azure RBAC)?* 2023. URL: https://learn.microsoft.com/en-us/azure/role-based-access-control/overview (visited on 04/26/2023).

[8] Luca Ferretti et al. “Survivable zero trust for cloud computing environments”. In: *Computers & Security* 110 (2021), p. 102419.

[9] Microsoft. *Zero Trust security*. 2022. URL: https://learn.microsoft.com/en-us/azure/security/fundamentals/zero-trust (visited on 03/24/2023).

[10] Cloudflare. *What is the control plane? | Control plane vs. data plane*. 2023. URL: https://www.cloudflare.com/learning/network-layer/what-is-the-control-plane/ (visited on 12/01/2023).

[11] Amazon Web Services. *Control planes and data planes*. 2023. URL: https://docs.aws.amazon.com/whitepapers/latest/aws-fault-isolation-boundaries/control-planes-and-data-planes.html (visited on 12/01/2023).

[12] Microsoft. *Azure control plane and data plane*. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/control-plane-and-data-plane (visited on 12/01/2023).

[13] Mustafa Toroman and Tom Janetscheck. *Mastering azure security : keeping your microsoft azure workloads safe*. eng. Second edition. Birmingham, England ; Packt Publishing, 2022. ISBN: 1-80324-292-2.

[14] Kevin L. Jackson and Scott Goessling. *Architecting Cloud Computing Solutions: Build cloud strategies that align technology and economics while effectively managing risk*. eng. Birmingham: Packt Publishing, 2018. ISBN: 9781788470742.

[15] Wikipedia. *Transport Layer Security*. 2023. URL: https://en.wikipedia.org/wiki/Transport_Layer_Security (visited on 01/07/2024).

[16] SSL.com. *What is SSL/TLS: An In-Depth Guide*. 2023. URL: https://www.ssl.com/article/what-is-ssl-tls-an-in-depth-guide/ (visited on 01/07/2024).

[17] Pushpa Herath. "Ensuring Data Security with Azure Storage". eng. In: *Azure Cloud Security for Absolute Beginners*. United States: Apress L. P, 2021, pp. 123–136. ISBN: 1484278593.

[18] Azure. *Azure Storage redundancy*. 2024. URL: https://learn.microsoft.com/en-us/azure/storage/common/storage-redundancy (visited on 01/07/2024).

[19] Azure. *Introduction to Azure Storage*. 2023. URL: https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction (visited on 09/18/2023).

[20] Microsoft. *What is Azure SQL?* 2023. URL: https://learn.microsoft.com/en-us/azure/azure-sql/azure-sql-iaas-vs-paas-what-is-overview?view=azuresql (visited on 09/19/2023).

[21] Microsoft. *Azure threat protection*. 2023. URL: https://learn.microsoft.com/en-us/azure/security/fundamentals/threat-detection#threat-protection-features-other-azure-services (visited on 09/20/2023).

[22] Microsoft. *An overview of Azure SQL Database and SQL Managed Instance security capabilities*. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-sql/database/security-overview (visited on 09/20/2023).

[23] Microsoft. *What is Azure SQL Managed Instance?* 2023. URL: https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/sql-managed-instance-paas-overview (visited on 09/20/2023).

[24] Microsoft. *What is Azure SQL Database?* 2023. URL: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview (visited on 09/20/2023).

[25] Microsoft. *What is the Azure Backup service?* 2024. URL: https://learn.microsoft.com/en-us/azure/backup/backup-overview (visited on 01/12/2024).

[26] Microsoft. *Overview of Azure Blob backup*. 2024. URL: https://learn.microsoft.com/en-us/azure/backup/blob-backup-overview (visited on 01/12/2024).

[27] Microsoft. *Automated backups in Azure SQL Database*. 2024. URL: https://learn.microsoft.com/en-us/azure/azure-sql/database/automated-backups-overview (visited on 01/12/2024).

[28] Microsoft. *Automated backups in Azure SQL Database*. 2024. URL: https://learn.microsoft.com/en-us/azure/azure-sql/database/long-term-retention-overview (visited on 01/12/2024).

[29] Sagar Lad. "Security Processes". In: *Azure Security For Critical Workloads : Implementing Modern Security Controls for Authentication, Authorization and Auditing*. Berkeley, CA: Apress, 2023, pp. 135–171. ISBN: 978-1-4842-8936-5. DOI: 10.1007/978-1-4842-8936-5_6. URL: https://doi.org/10.1007/978-1-4842-8936-5_6.

[30] Microsoft. *Shared responsibility in the cloud*. 2023. URL: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis (visited on 03/24/2023).

[31] Peter De Tender. *Demystifying Service Principals – Managed Identities*. 2021. URL: https://devblogs.microsoft.com/devops/demystifying-service-principals-managed-identities/ (visited on 01/26/2024).

[32] Microsoft. *Learn about groups and access rights in Azure Active Directory*. 2023. URL: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/concept-learn-about-groups (visited on 09/07/2023).

[33] Microsoft. *Azure network security overview*. 2023. URL: https://learn.microsoft.com/en-us/azure/security/fundamentals/network-overview (visited on 04/24/2023).

[34] Gilbert Held. *The ABCs of TCP/IP*. CRC Press, 2002.

[35] Azure. *What is a VPN?* 2023. URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-vpn (visited on 10/31/2023).

[36] Pethuru Raj, Anupama Raman, and Harihara Subramanian. *Architectural patterns : uncover essential patterns in the most indispensable realm of enterprise architecture*. eng. 1st edition. Birmingham, England: Packt, 2017. ISBN: 1-78728-749-1.

[37] Vladimir Stefanovic and Milos Katinski. “Azure Administration”. In: *Pro Azure Administration and Automation: A Comprehensive Guide to Successful Cloud Management*. Berkeley, CA: Apress, 2021, pp. 19–37. ISBN: 978-1-4842-7325-8. DOI: 10.1007/978-1-4842-7325-8_2. URL: https://doi.org/10.1007/978-1-4842-7325-8_2.

[38] AWS. *Connect to the internet using an internet gateway*. 2023. URL: https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html (visited on 04/26/2023).

[39] Microsoft. *Network security groups*. 2023. URL: https://learn.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview (visited on 04/26/2023).

[40] AWS. *Control traffic to subnets using Network ACLs*. 2023. URL: https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html (visited on 04/26/2023).

[41] Microsoft. *What is Azure Virtual Network?* 2023. URL: https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview (visited on 05/28/2023).

[42] Microsoft. *What is Azure Load Balancer?* 2024. URL: https://learn.microsoft.com/en-us/azure/load-balancer/load-balancer-overview (visited on 01/08/2024).

[43] Microsoft. *What is cloud monitoring?* 2023. URL: https://www.crowdstrike.com/cybersecurity-101/observability/cloud-monitoring/ (visited on 11/12/2023).

[44] Microsoft. *Azure Monitor overview*. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-monitor/overview (visited on 11/13/2023).

[45] Umar Mukhtar Ismail, Syed Islam, and Shareeful Islam. “Towards Cloud Security Monitoring: A Case Study”. In: *2016 Cybersecurity and Cyberforensics Conference (CCC)*. 2016, pp. 8–14. DOI: 10.1109/CCC.2016.8.

[46] Microsoft. *Log Analytics workspace overview*. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-workspace-overview (visited on 12/19/2023).

[47] Microsoft. *Diagnostic settings in Azure Monitor*. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-monitor/essentials/diagnostic-settings (visited on 12/22/2023).

[48] Microsoft. *What are Azure Monitor alerts?* 2023. URL: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-overview (visited on 12/19/2023).

[49] Azure. *Microsoft Azure Well-Architected Framework*. 2023. URL: https://learn.microsoft.com/en-us/azure/well-architected (visited on 10/09/2023).

[50] Azure. *Microsoft Cloud Adoption Framework for Azure*. 2023. URL: https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/overview (visited on 10/10/2023).

[51] Adora Nwodo. *Beginning Azure Devops : Planning, Building, Testing, and Releasing Software Applications on Azure*. eng. Hoboken, NJ: John Wiley & Sons, Inc., 2023. ISBN: 9781394165896.

[52] Azure. *What is Azure Resource Manager?* 2023. URL: https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/overview (visited on 10/10/2023).

[53] Azure. *Azure Resource Manager deployment modes*. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deployment-modes (visited on 10/17/2023).

[54] Azure. *ARM template deployment what-if operation*. 2023. URL: https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/deploy-what-if?tabs=azure-powershell (visited on 10/17/2023).

[55] Azure. *What is Bicep?* 2023. URL: https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview (visited on 10/12/2023).

[56] Wikipedia. *Code bloat*. 2023. URL: https://en.wikipedia.org/wiki/Code_bloat (visited on 01/26/2024).

[57] Azure. *Overview of Microsoft cloud security benchmark (v1)*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/overview (visited on 11/26/2023).

[58] Azure. *Cloud security posture management*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/concept-cloud-security-posture-management (visited on 11/16/2023).

[59] Azure. *What is Microsoft Defender for Cloud?* 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-cloud-introduction (visited on 11/16/2023).

[60] Azure. *Security policies in Defender for Cloud*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/security-policy-concept (visited on 12/20/2023).

[61] Azure. *Secure score*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/secure-score-security-controls (visited on 12/19/2023).

[62] Matti Valkeinen. *Cloud Infrastructure Tools For Cloud Applications : Infrastructure management of multiple cloud platforms*. 2022.

[63] Amazon Web Services. *What is Amazon Elastic Container Service?* 2023. URL: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html (visited on 11/20/2023).

[64] Amazon Web Services. *Amazon RDS Security*. 2023. URL: https://aws.amazon.com/rds/features/security/ (visited on 11/20/2023).

[65] Amazon Web Services. *What is Amazon Cognito?* 2023. URL: https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html (visited on 11/20/2023).

[66] Jobs in Azure Container Apps. *Microsoft*. 2024. URL: https://learn.microsoft.com/en-us/azure/container-apps/jobs (visited on 01/10/2024).

[67] Flow logging for network security groups. *Microsoft*. 2024. URL: https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-nsg-flow-logging-overview (visited on 01/10/2024).

[68] Microsoft. *Security Control: Network Security*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-network-security (visited on 11/26/2023).

[69] OSI model. *Wikipedia*. 2024. URL: https://en.wikipedia.org/wiki/OSI_model (visited on 01/12/2024).

[70] OWASP Top Ten. *OWASP*. 2024. URL: https://owasp.org/www-project-top-ten/ (visited on 01/12/2024).

[71] What is Azure Web Application Firewall? *Microsoft*. 2024. URL: https://learn.microsoft.com/en-us/azure/web-application-firewall/overview (visited on 01/12/2024).

[72] Microsoft. *Security Control: Identity Management*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-identity-management (visited on 12/01/2023).

[73] Microsoft. *Security Control: Privileged access*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-privileged-access (visited on 12/03/2023).

[74] Microsoft. *Security Control: Data protection*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-data-protection (visited on 12/18/2023).

[75] Microsoft. *Azure Data Encryption at rest*. 2023. URL: https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-atrest (visited on 12/17/2023).

[76] Microsoft. *Security Control: Asset management*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-asset-management (visited on 12/18/2023).

[77] Microsoft. *Security Control: Logging and threat detection*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-logging-threat-detection (visited on 12/19/2023).

[78] Microsoft. *Overview of Microsoft Defender for Azure SQL*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-sql-introduction (visited on 12/20/2023).

[79] Microsoft. *Overview of Microsoft Defender for Storage*. 2023. URL: Overview%20of%20Microsoft%20Defender%20for%20Storage (visited on 12/20/2023).

[80] Microsoft. *Overview of Microsoft Defender Key Vault*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-key-vault-introduction (visited on 12/20/2023).

[81] Microsoft. *Overview of Microsoft Defender for Resource Manager*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-resource-manager-introduction (visited on 12/20/2023).

[82] Microsoft. *Overview of Microsoft Defender for DNS*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-dns-introduction (visited on 12/20/2023).

[83] Microsoft. *Overview of Microsoft Defender for App Service*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-app-service-introduction (visited on 12/20/2023).

[84] Microsoft. *Overview of Container security in Microsoft Defender for Containers*. 2023. URL: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-containers-introduction (visited on 12/20/2023).

[85] Microsoft. *Security Control: Incident response*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-incident-response (visited on 12/23/2023).

[86] Microsoft. *Security Control: Posture and vulnerability management*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-posture-vulnerability-management (visited on 12/23/2023).

[87] Microsoft. *Azure Policy built-in initiative definitions*. 2023. URL: https://learn.microsoft.com/en-us/azure/governance/policy/samples/built-in-initiatives (visited on 12/29/2023).

[88] Microsoft. *Regulatory Compliance in Azure Policy*. 2023. URL: https://learn.microsoft.com/en-us/azure/governance/policy/concepts/regulatory-compliance (visited on 12/28/2023).

[89] Microsoft. *Security Control: Endpoint security*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-endpoint-security (visited on 12/28/2023).

[90] Microsoft. *Security Control: Backup and recovery*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-backup-recovery (visited on 12/28/2023).

[91] Microsoft. *Security Control: DevOps security*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-devops-security (visited on 12/28/2023).

[92] Microsoft. *Security Control: Governance and strategy*. 2023. URL: https://learn.microsoft.com/en-us/security/benchmark/azure/mcsb-governance-strategy (visited on 12/29/2023).

[93] CARML. *Azure Resource Modules*. 2023. URL: https://github.com/Azure/ResourceModules (visited on 12/29/2023).

APPENDIX A: DIFFERENCES IN ARM AND BICEP TEMPLATES

In this appendix code snippets are shown from both ARM and Bicep templates. Snippets are from visual Visual Studio Code (VS Code). The whole template isn't shown, but both snippets are a part of a virtual machine declaration, and they depict the part, in which the network interface is declared.

```
// NIC for Virtual Machine
{
  "type": "Microsoft.Network/networkInterfaces",
  "apiVersion": "2023-04-01",
  "name": "NICForVM",
  "location": "[parameters('location')]",
  "properties": {
    "networkSecurityGroup": {
      "id": "[resourceId('Microsoft.Network/networkSecurityGroups', 'NSG_VirtualMachine')]"
    },
    "enableAcceleratedNetworking": false,
    "ipConfigurations": [
      {
        "name": "[format('{0}-ipconfig-VM02-01', parameters('namePrefix'))]",
        "properties": {
          "subnet": {
            "id": "[resourceId('Microsoft.Network/virtualNetworks/subnets', parameters('VNName'), parameters('subnetName'))]"
          },
          "privateIPAllocationMethod": "Dynamic",
          "publicIPAddress": {
            "id": "[resourceId('Microsoft.Network/publicIPAddresses', 'VM02-public-IP')]"
          }
        }
      }
    ]
  },
  "dependsOn": [
    "[resourceId('Microsoft.Network/networkSecurityGroups', 'NSG_VirtualMachine')]",
    "[resourceId('Microsoft.Network/publicIPAddresses', 'VM02-public-IP')]"
  ]
},
```

Figure A.1. *Code snippet from an ARM template for a network interface*

```
// NIC for Virtual Machine
resource NICForVM 'Microsoft.Network/networkInterfaces@2023-04-01' = {
  name: 'NICForVM'
  location: location
  properties: {
    networkSecurityGroup: {
      id: NSG_VirtualMAchine_res.id
    }
    enableAcceleratedNetworking: false
    ipConfigurations: [
      {
        name: '${namePrefix}-ipconfig-VM02-01'
        properties: {
          subnet: {
            id: subnet.id
          }
          privateIPAllocationMethod: 'Dynamic'
          publicIPAddress: {
            id: VMPublicIP.id
          }
        }
      }
    ]
  }
}
```

Figure A.2. *Code snippet from an Bicep template for a network interface*

APPENDIX B: MCSB V1 EXCLUDE REASONS

The Microsoft Cloud Security Benchmark v1 is a set of controls and best practices for implementing information security practices in Azure, by Microsoft. It is not the whole truth, but gives a great cloud native aspect from the perspective of Azure services. The controls have been limited to the scope of the case study. The MCSB v1 controls are presented below at the table B.1, and the ones excluded from the study scope are reasoned in the *Exclude reason* column.

Table B.1. *MCSB v1 recommendations and excludations*

ID	Recommendation	Exclude reason
NS	**Network Security**	
NS-1	Establish network segmentation boundaries	
NS-2	Secure cloud native services with network controls	
NS-3	Deploy firewall at the edge of enterprise network	
NS-4	Deploy intrusion detection/intrusion prevention systems (IDS/IPS)	Too high level
NS-5	Deploy DDOS protection	Too high level
NS-6	Deploy web application firewall	

NS-7	Simplify network security configuration	
NS-8	Detect and disable insecure services and protocols	Too high level
NS-9	Connect on-premises or cloud network privately	
NS-10	Ensure Domain Name System (DNS) security	
IM	**Identity Management**	
IM-1	Use centralized identity and authentication system	
IM-2	Protect identity and authentication systems	Too high level
IM-3	Manage application identities securely and automatically	
IM-4	Authenticate server and services	
IM-5	Use single sign-on (SSO) for application access	Too software specific
IM-6	Use strong authentication controls	
IM-7	Restrict resource access based on conditions	Too high level
IM-8	Restrict the exposure of credential and secrets	
IM-9	Secure user access to existing applications	Too software specific
PA	**Privileged Access**	
PA-1	Separate and limit highly privileged/administrative users	Too high level
PA-2	Avoid standing access for user accounts and permissions	Too high level
PA-3	Manage lifecycle of identities and entitlements	Too high level
PA-4	Review and reconcile user access regularly	Too high level

PA-5	Set up emergency access	Too high level
PA-6	Use privileged access workstations / channel for administrative tasks	Too high level
PA-7	Follow just enough administration (least privilege) principle	Too high level
PA-8	Determine access process for cloud provider support	Too high level
DP	**Data Protection**	
DP-1	Discover, classify, and label sensitive data	Too data dependant
DP-2	Monitor anomalies and threats targeting sensitive data	Too high level
DP-3	Encrypt sensitive data in transit	
DP-4	Enable data at rest encryption by default	
DP-5	Use customer-managed key option in data at rest encryption when required	Too high level
DP-6	Use a secure key management process	Too high level
DP-7	Use a secure certificate management process	Too high level
DP-8	Ensure security of key and certificate repository	
AM	**Asset Management**	
AM-1	Track asset inventory and their risks	Too high level
AM-2	Use only approved services	Too high level
AM-3	Ensure security of asset lifecycle management	Too high level
AM-4	Limit access to asset management	Too high level

AM-5	Use only approved applications in virtual machine	Too software specific
LT	**Logging and Threat Detection**	
LT-1	Enable threat detection capabilities	
LT-2	Enable threat detection for identity and access management	Too high level
LT-3	Enable logging for security investigation	
LT-4	Enable network logging for security investigation	
LT-5	Centralize security log management and analysis	Too high level
LT-6	Configure log storage retention	Too high level
LT-7	Use approved time synchronization sources	Too high level
IR	**Incident Response**	
IR-1	Preparation - update incident response plan and handling process	Too high level
IR-2	Preparation - setup incident contact information	Too high level
IR-3	Detection and analysis - create incidents based on high-quality alerts	Too high level
IR-4	Detection and analysis - investigate an incident	Too high level
IR-5	Detection and analysis - prioritize incidents	Too high level
IR-6	Containment, eradication and recovery - automate the incident handling	Too high level
IR-7	Post-incident activity - conduct lesson learned and retain evidence	Too high level
PV	**Posture and Vulnerability Mgmt**	

PV-1	Define and establish secure configurations	
PV-2	Audit and enforce secure configurations	
PV-3	Define and establish secure configurations for compute resources	Too software specific
PV-4	Audit and enforce secure configurations for compute resources	Too software specific
PV-5	Perform vulnerability assessments	Too software specific
PV-6	Rapidly and automatically remediate vulnerabilities	No VMs used on project
PV-7	Conduct regular red team operations	Too high level
ES	**Endpoint Security**	
ES-1	Use Endpoint Detection and Response (EDR)	Too software specific
ES-2	Use modern anti-malware software	Too software specific
ES-3	Ensure anti-malware software and signatures are updated	Too software specific
BR	**Backup and Recovery**	
BR-1	Ensure regular automated backups	
BR-2	Protect backup and recovery data	
BR-3	Monitor backups	Too high level
BR-4	Regularly test backup	Too high level
DS	**DevOps Security**	
DS-1	Conduct threat modeling	Too software specific

DS-2	Ensure software supply chain security	Too software specific
DS-3	Secure DevOps infrastructure	Too software specific
DS-4	Integrate static application security testing into DevOps pipeline	Too software specific
DS-5	Integrate dynamic application security testing into DevOps pipeline	Too software specific
DS-6	Enforce security of workload throughout DevOps lifecycle	Too software specific
DS-7	Enable logging and monitoring in DevOps	Too software specific
GS	**Governance and Strategy**	
GS-1	Align organization roles, responsibilities and accountabilities	Too high level
GS-2	Define and implement enterprise segmentation/separation of duties strategy	Too high level
GS-3	Define and implement data protection strategy	Too high level
GS-4	Define and implement network security strategy	Too high level
GS-5	Define and implement security posture management strategy	Too high level
GS-6	Define and implement identity and privileged access strategy	Too high level
GS-7	Define and implement logging, threat detection and incident response strategy	Too high level
GS-8	Define and implement backup and recovery strategy	Too high level
GS-9	Define and implement endpoint security strategy	Too high level
GS-10	Define and implement DevOps security strategy	Too high level

GS-11	Define and implement multi-cloud security strategy	Too high level