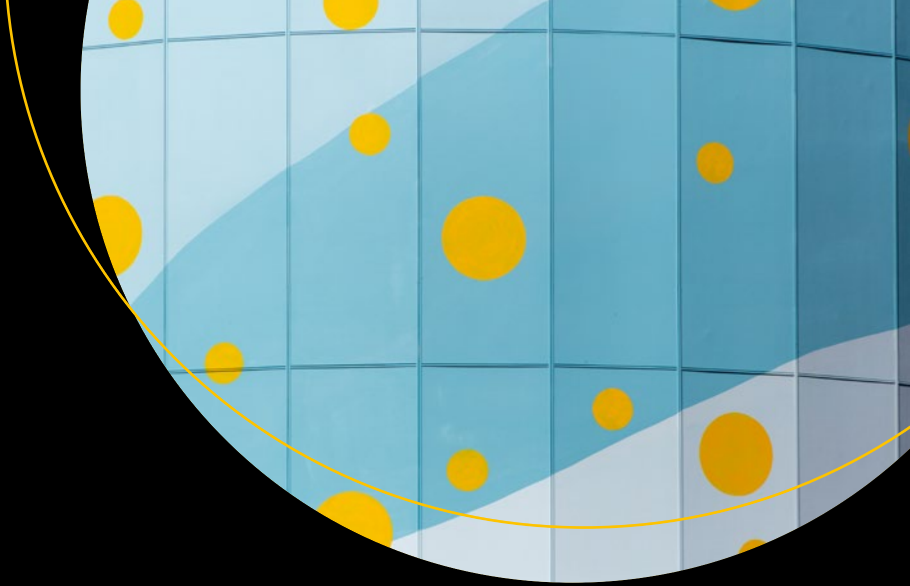


Azure Security Handbook

A Comprehensive Guide for Defending Your Enterprise Environment

Karl Ots

Apress®

Azure Security Handbook

A Comprehensive Guide for Defending Your Enterprise Environment

Karl Ots

Apress®

Azure Security Handbook: A Comprehensive Guide for Defending Your Enterprise Environment

Karl Ots
Zürich, Zürich, Switzerland

ISBN-13 (pbk): 978-1-4842-7291-6 ISBN-13 (electronic): 978-1-4842-7292-3
https://doi.org/10.1007/978-1-4842-7292-3

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Coordinating Editor: Jill Balzano

Cover photo courtesy of Gratisography

Distributed to the book trade worldwide by Springer Science+Business Media LLC, 1 New York Plaza, Suite 4600, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484272916. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

For my wife Annie.

Table of Contents

About the Author

Karl Ots is a cloud and cybersecurity leader with a decade of experience in Microsoft Azure security with large enterprises in fields such as technology, manufacturing, and finance. Karl is recognized as the global top technology visionary with the Microsoft Regional Director award. He is a patented inventor, a LinkedIn Learning instructor, and a Microsoft Azure MVP. He holds the Azure Security Engineer, SABSA Foundation SCF, and CISSP certifications.

Karl is a frequent speaker on cloud security topics at global conferences such as Microsoft Ignite or (ISC)^2 Security Congress. In his current role, he serves as Head of Cloud Security at EPAM Systems, a global engineering and consulting company. He hosts the Cloud Gossip podcast.

About the Technical Reviewer

As a Microsoft Technical Evangelist/Cloud Solutions Architect, **Mike Martin** is an Azure go-to for ISVs (independent software vendors). He's been active in the IT industry for more than 20 years and has performed work in almost all types of job profiles, going from coaching and leading a team to architecting and systems design and training. Today, he's primarily into the Microsoft Cloud Platform and Application Life cycle Management/DevOps, with an emphasis on security domains. He's not a stranger to both Dev and IT Pro topics; they even call him the perfect hybrid solution.

In January 2012, he became a crew member of AZUG, the Belgian Microsoft Azure User Group. As an active member, he's both involved in giving presentations and organizing events (like ITProceed, Techorama, and Global Azure Bootcamp, a.k.a. GAB). Mike was also a Microsoft Azure MVP (awarded five times since 2013, receiving his fifth in July 2017!) and Microsoft Azure Advisor.

Helping out in the community and introducing new and young people to the world of Microsoft and technology is also one of his passions.

Acknowledgments

There is never a perfect time to write a book. Yet, I found it oddly calming to write this book during one of the largest turmoils of my lifetime.

I want to thank my awesome wife Annie for giving me the motivation to always reach for the next level in my professional and personal endeavors.

I want to thank Jill and Joan from Apress for keeping me accountable yet giving me the time I needed to find my voice. I also want to thank Mike and Pablo from Microsoft for their efforts on making this book better.

I want to thank my colleagues, customers, and partners I have had the pleasure of working, failing, and learning alongside with.

Finally, I want to thank the grumpy old geezers at the chat group for never resisting a good pun.

Foreword

At Microsoft, we invest more than a billion dollars each year in research and development to help our customers secure their organizations' digital transformations. For us and many other technology providers, this is one of the fastest-growing technology areas. And while in the IT industry, we usually associate growth as something positive, the case of cloud security is bittersweet. On the one hand, implementing a secure cloud or hybrid footprint is a fundamental best practice. But this fast growth is fueled largely by the need to respond to, and to be one step ahead of, the growing number of malicious actors, including many private-sector offensive actors that are helping democratize cyber threats with their new hacking-as-a-service model.

Then, the COVID 19 pandemic drove a once-in-a-lifetime acceleration in cloud adoption and digital transformation. In the last sixteen months, companies worldwide and their security teams faced new challenges in their effort to support remote and hybrid-work scenarios, all in a short window of time. These unprecedented circumstances demanded a fast response from us at Microsoft. In my role leading community programs for Azure security engineering groups and bringing the voice of our customers into our engineering roadmaps, I saw first-hand an increase in feedback and insights coming from our customers and partners. In response, we are driving a fast evolution of our Azure security services and controls embedded in many Azure components and workloads. As a result, the list of security changes across Azure services and workloads over the last year is vast.

But in parallel to technical evolution, companies face the challenge of quickly upskilling their security teams. As a direct consequence, the need for cloud security experts in all cloud areas—identity and access control, data, applications, infrastructure, and the edge—is in high demand. Our technical libraries and training at `https://docs.microsoft.com` play an essential part in this effort. My colleagues in CxE also sustain a very active calendar of Azure security webinars that help educate security professionals on the latest releases in our services. You can check them out at `https://aka.ms/SecurityCommunity`. But even with all these investments in technical readiness, we can only cover so much ground.

This is where members of the security community bring their passion for sharing knowledge with others to play. Their contributions to our technical information complement and strengthen security knowledge in Azure. Further, their contributions often add different perspectives, insights, and scenarios to those covered in our Microsoft documentation.

As both a member of our Microsoft Regional Director[1] program and our Azure Private Security Community, where he represents Swiss Re as Vice President, Cloud Security Architecture, Karl Ots has a long history of sharing valuable feedback and insights with our security engineers. The combination of community leadership and private sector responsibilities experiences gives Karl a deep understanding of cloud security and the steps that companies should take when planning and implementing security for the cloud and multi-cloud with Azure.

In his book, Karl offers much-needed guidance to help organizations that are new to Azure, or cloud security in general, in planning the security architecture for their new cloud footprint. He also provides plenty of best practices and configuration decisions that together help decrease their attack surface by explaining how to take advantage of our Azure security services and native controls and integrate our solutions and centralize security operations in Azure. Additionally, Karl shares cloud and security agnostic recommendations that go beyond user guides, sharing lessons from his vast first hand experiences. And he maps many of our Azure controls and best practices to industry security benchmarks like those from CIS and NIST and security models like Zero Trust and Shift-Left.

I appreciate that his chapters guide you through key security domains and how to secure them in Azure. From identity and access management and their fundamental role in cloud security to protecting your cloud infrastructure and workloads, he touches on security basics and configurations without shying from pointing out where integration with 3rd party providers will help with specific needs.

Through this approach, Karl also brings several conversations that, while often overlooked, are fundamental when managing security operations in the cloud. For example, decisions such as the retention time for logs used for your security monitoring and analysis can have on your company's bill and the cloud shared responsibilities matrix[2], which showcases some of the benefits of moving to the cloud and highlights the importance of defining clear roles and responsibilities that may include workload

[1] https://rd.microsoft.com/

[2] https://docs.microsoft.com/azure/security/fundamentals/shared-responsibility

owners who had little or no security ownership over on-premises implementations. Karl provides complete perspectives and hence brings clarity to architecting security in Azure.

In closing his book brings perspective to readers of all levels of expertise with security in Azure. After more than three years working in Cloud Security and learning the intimacies of our Azure security services, it has even taught me several technical and non-technical considerations of planning a move to the cloud.

I want to thank you for this book Karl. You and many others in the security community play a crucial role in communicating the benefits that our Microsoft products and services offer our customers, adding your valuable expertise. Thank you for representing the voice of our customers and for influencing the future of our Azure security services and native controls.

Pablo Chacón

Sr. Program Manager – Private Security Communities Lead
Customer Experience Engineering (CxE), Cloud Security
Microsoft

July 2021

Introduction

You are about to enter the world of cloud security and learn from the mistakes I have made when working with hundreds of Azure projects with some of the largest organizations in the world.

This is the book I wish I had on hand when I started my cloud journey. It does not cover every single knob and control for every Azure service there is. That would be overwhelming. Instead, it covers the core pillars of cloud security with a reasonable sampling of pragmatically selected Azure services.

You might be a seasoned security professional taking your first steps into the public cloud world. Or you might be a cloud-native solution architect looking at securing your applications. I hope both of you will learn about each other's mindset by reading this book and the two worlds would not be as disconnected as they still are.

I wrote this book hoping that it would keep being valuable to you over the course of your Azure projects. On the first read-through, you might struggle to unlearn some existing habits and mental models. After embracing the cloud-native security mindset, I hope that you will keep returning to this book throughout your Azure adoption life cycle.

The first half of this book, Chapters 1-4 covers the core pillars of cloud security framework development. If you are looking at the Azure cloud through the central organization's looking glass, you will find this part the most familiar to you.

The second half of the book, Chapters 5-8 covers security controls for the most archetypical workloads. If you are a cloud solution builder, you will find this part of the book to your liking.

CHAPTER 1

Introduction to Cloud Security Architecture

Cloud computing is perhaps the largest information technology megatrend of this decade. The promises of commoditization of core infrastructure services, faster time to market, and adoption of brand-new technologies such as artificial intelligence or quantum computing are alluring.

Furthermore, the global disruptions of the COVID-19 pandemic in the beginning of this decade proved that the hyperscale cloud providers can answer the most unpredictable demand thrown at them.

> *The pandemic validated cloud's value proposition.*
>
> —Sid Nag, Research Vice President at Gartner[1]

The cloud computing model is now ready to mature from the experiments of early adopters and to become the mainstream computing platform of established enterprises. With this shift, we need to stop thinking about cloud security as an isolated domain of information security where some traditional rules cannot apply. We need to start looking at how the cloud can provide at least the same level of confidentiality, integrity, and availability of our information and assets as any other computing platform.

But with over hundreds of existing services, hundreds of announcements every year, and lack of skilled workforce, taming the beast of cloud security with traditional methods can seem overwhelming. The answer is what I like to call **cloud-native security**.

[1] www.gartner.com/en/newsroom/press-releases/2020-11-17-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-grow-18-percent-in-2021

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_1

In this chapter, we define cloud-native security and preface the subsequent chapters of this book. After reading this chapter, you will be able to describe what cloud-native security is and list what to include in a cloud security framework.

Cloud Security Responsibilities

Did you know that contrary to popular belief, the most common cloud security threats are not outside attacks, but rather misconfigurations?[2]

Based on this data, we can conclude the following points: First, the cloud can be just as safe (or unsafe) against outside attacks as our on-premises systems. Second, to fully secure public cloud platforms, we need to understand them deeply. This requires both upskilling existing information security office with cloud expertise and shifting the way security responsibilities are spread across the organization.

Shared Responsibility Model

In the world of on-premises computing, cybersecurity risks were perceived to be limited to the organization's network. In the era of cloud computing, both the impact and likelihood of potential risks are significantly higher.

Figure 1-1 describes how the security responsibilities are shared between the cloud provider (Microsoft) and the customer organization (us) across the various cloud service models: software as a service (SaaS), platform as a service (PaaS), infrastructure as a service (IaaS), and, finally, on-premises computing (our own datacenter). We can interpret this model from left to right or right to left.

Let us start by looking at this model from left to right. When using software as a service, we have the highest level of integrated security out of the box. The downside is that we are limited to what the platform offers: if the default settings or limited options for configuration do not meet our security requirements, we need to consider a cloud service model with a lower level of abstraction, such as platform as a service. As we move further to the right, we get lower level of integrated security, but we gain the possibility to implement additional controls to meet our requirements.

[2] According to Verizon DBIR 2020: https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf

Now let us analyze this model from right to left. This might feel more familiar approach when evaluating what needs to change when moving from traditional security to cloud security. As we move from on-premises to infrastructure as a service and further up in the cloud model abstractions, we give up control of configuring and operating said services. As we give up control, Microsoft assumes the responsibility of securing these services.

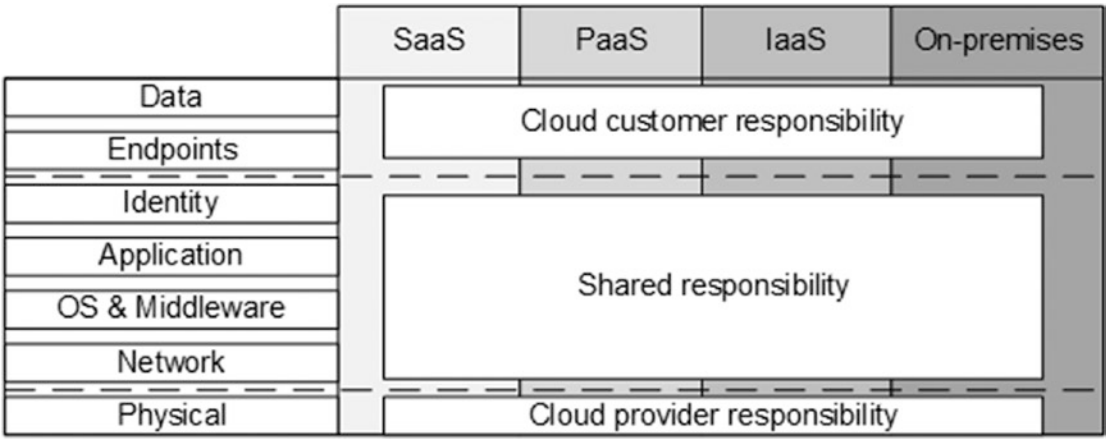


Figure 1-1. *Shared responsibility model of cloud security from the organization point of view*

Shifting Security Left

With the corresponding advent of DevOps methodology, security is now the responsibility of everyone who is part of the application development life cycle, not just security specialists.

Application Developers are no longer limited to services provided by central IT: they can adopt new cloud services on their own. At worst, this might lead into so-called shadow IT. At best, this expands your information security office in the orders of magnitude.

At the same time, the Application Developers are no longer protected by the outside perimeter of central teams: in Azure, they are responsible for more. There is no corporate firewall, access control or centralized audit logging to fall back on. If the developers have direct access to Azure Management pane, day-to-day operations, such as vulnerability management, patching, or monitoring, might fall under their responsibility, too.

Cloud-Native Security

As with any other paradigm-shifting technology, securing the public cloud is not as simple as extending or replicating existing security controls. Some of the differences are because of technical limitations, such as the changes in the network perimeter or access to certain detection and monitoring capabilities. Some changes are required because of the reasons your business units and application development teams are choosing the cloud: flexibility, faster time to market, or access to technologies that are not available in existing hosting platforms.

These differences eventually require you to shift how you implement security architecture. Rather than bolting your existing controls, tools, and processes on top of cloud workloads, you have a unique opportunity to build security into the very platform your workloads are hosted in!

Multi-cloud or Cloud-Native Security?

Multi-cloud solutions are built to be agnostic of the cloud platform they are hosted in. They are often built on services that have comparable services in other cloud providers, such as virtual machines or container hosting services. As such, multi-cloud solutions can use the least common denominator of the cloud services available in the cloud providers of your choice. Designing a multi-cloud solution means compromising features and integrated security in favor of interoperability and the ability to externalize the security controls from the cloud provider. In practice, multi-cloud is both cost prohibitive and slower to implement than building your solution with native platform-as-a-service components of a single cloud solution platform.

Similarly, multi-cloud security tools cannot offer the same level of granularity than those that are built natively for the individual cloud services. You might be able to standardize on a firewall appliance or a workload protection platform across cloud platform, but you will be only covering a subset of the available cloud services, particularly missing the platform-as-a-service services and native platform security features.

> Instead of adopting a multi-cloud security model, you should adopt a cloud-native security model.

Companies that are successfully leveraging multiple cloud service providers are often doing so to meet local regulatory requirements or to use existing skills to build native solutions for each of the clouds separately.

Cloud Security Architecture

Designing your cloud security architecture requires a careful balancing act between the brave new world of cloud and the existing internal and external security requirements. Figure 1-2 illustrates the difficulties in balancing these requirements.

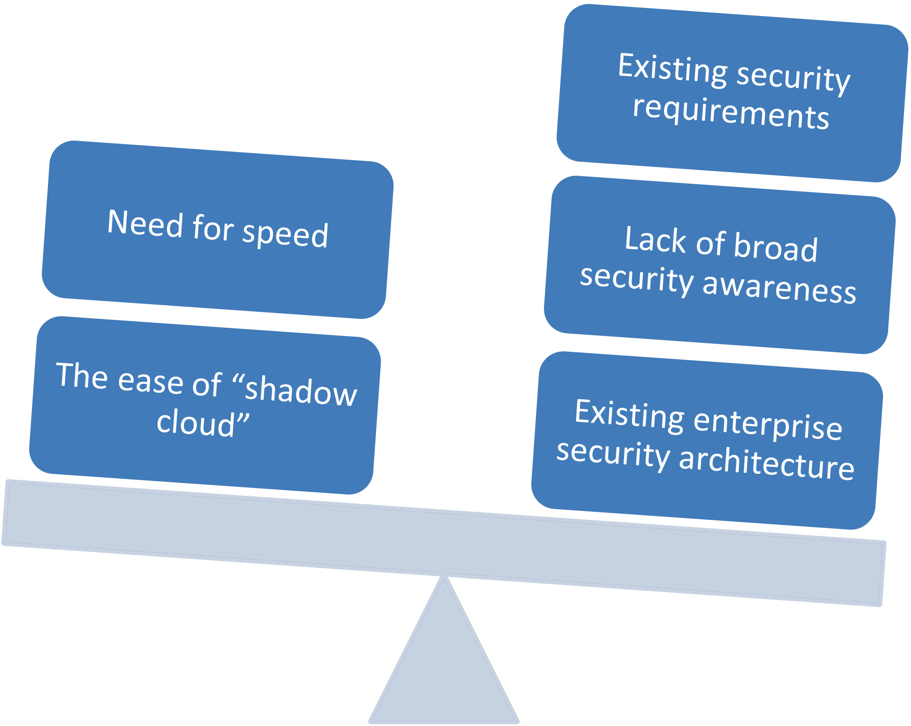

Existing security requirements
Need for speed
Lack of broad security awareness
The ease of "shadow cloud"
Existing enterprise security architecture

Figure 1-2. *Balance of enterprise cloud security*

Azure Security Building Blocks

To implement your cloud security architecture, you can conceptualize the required components as reusable and modular building blocks. Figure 1-3 describes the concept of Azure security building blocks.

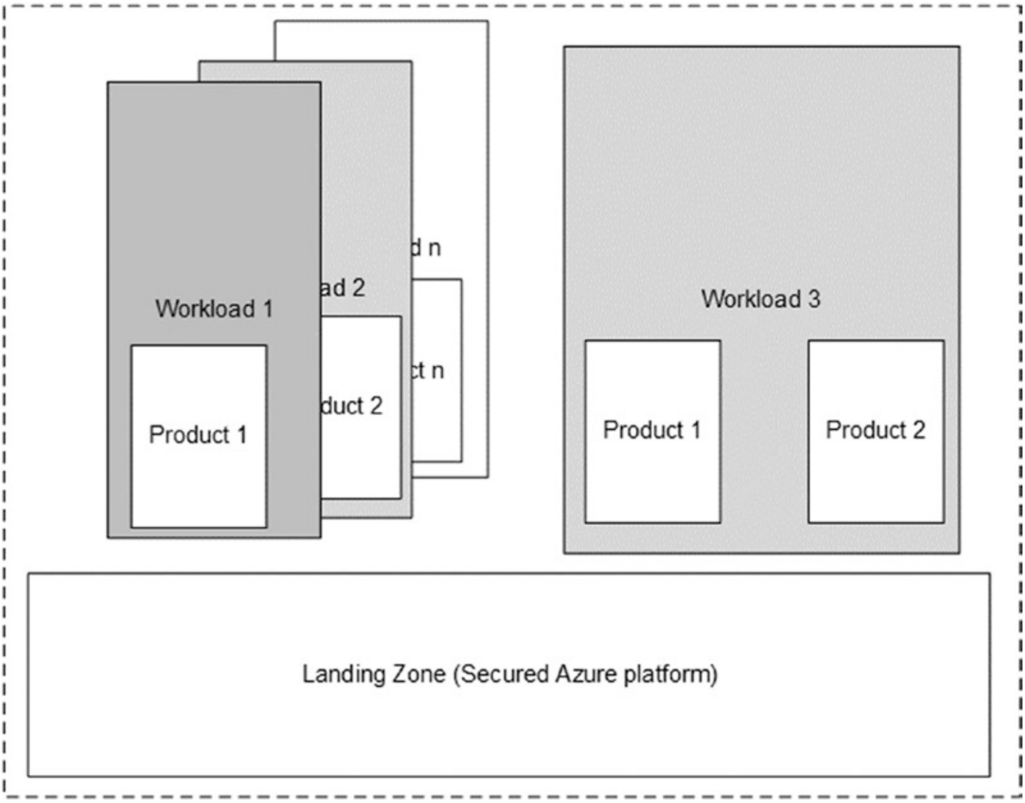


Figure 1-3. *Azure security building blocks*

Workloads (or applications) are typically a collection of one or more Azure services. Depending on your platform controls, these can be manually provisioned Azure services or instances of internal products.

Products are your internal artefacts for Azure services, preconfigured to meet your known good configuration. Typically, these are infrastructure-as-code artefacts, such as Azure Resource Manager templates, Bicep files, or Terraform modules. Building your security controls into the product artefacts enables you to get to a consistent and secure state from the beginning of a workload life cycle.

Landing zone is your secured Azure platform. This means everything from the way you get your Azure (enterprise agreement, cloud solution provider, or others) to networking topologies or enforcement of tenant-level security configuration using policy initiatives.

Landing Zone Security

Out of the cloud security building blocks, the landing zone is key from the security point of view. A landing zone is where you enforce security controls for your products to keep their secure state. Landing zone is also where you prepare for an audit by providing assurance that the workloads deployed to your landing zone are meeting your security requirements. You can have multiple landing zones: for example, your migrated legacy applications might stay within a centrally managed landing zone within a spoke of your centrally managed network. By contrast, your new and experimental workloads might stay within a separate landing zone with more freedom to reach fast time to market but offering less integration to your shared services (such as network).

While the landing zone implementation might vary, the overall approach of securing a platform for your workloads to run is consistent.

Identity and Access Management

The first component of your landing zone security is identity and access management (IAM). This consists of

- Integration with existing IAM processes
- Access control to the management and data plane
- Access control through the SecDevOps life cycle

We will discuss identity and access management in more detail in Chapter 2.

Detection and Monitoring

The next component of your landing zone security is detection and monitoring. This consists of

- Enforcing audit logging across the landing zone and any products or workloads deployed onto it
- Centralized logging architecture
- Integration with your security information and event management (SIEM) tool and your security operations center (SOC)

We will discuss detection and monitoring in more detail in Chapter 3.

Network Security

The last component of your landing zone security is the network perimeter. This consists of

- Securing cross-subscription, cross-region, and cross-cloud traffic
- Securing platform-as-a-service and infrastructure-as-a-service traffic
- Securing application-level traffic

We will discuss network security in more detail in Chapter 4.

Cloud Security Framework

To secure your cloud building blocks, you will need to define your cloud security framework. A cloud security framework defines the architecture, policies, and controls to secure your cloud environment.

Cloud Control Frameworks

You may use any existing control frameworks to get started building your cloud security framework on. However, simply following a control framework does not mean you are "done" with security. You need to select the appropriate controls for your cloud security framework according to your organization's risk appetite. Some control frameworks include

- National Institute of Standards and Technologies: Cyber Security Framework.
- Center for Internet Security's Microsoft Azure Foundations Benchmark provides prescriptive guidance for establishing a secure baseline configuration for Microsoft Azure.
- Cloud Security Alliance's Cloud Controls Matrix (CCM) is a cybersecurity control framework for cloud computing.
- Microsoft Azure Security Benchmark: Both documentation of best practices (Azure Security Baseline) and enforceable policy initiative.

Building Your Cloud Security Framework

To develop a cloud security framework that is both consistent with your existing security architecture and effective in securing emerging cloud services, incorporate it with your enterprise security architecture methodology, such as Sherwood Applied Business Security Architecture (SABSA). You can use SABSA to model cloud security in an agnostic manner, defining your control and enablement objectives before looking at control list from security frameworks or choosing your technical implementation.

Figure 1-4 illustrates the various inputs to the cloud security framework. Your cloud security framework should incorporate requirements from all these layers.

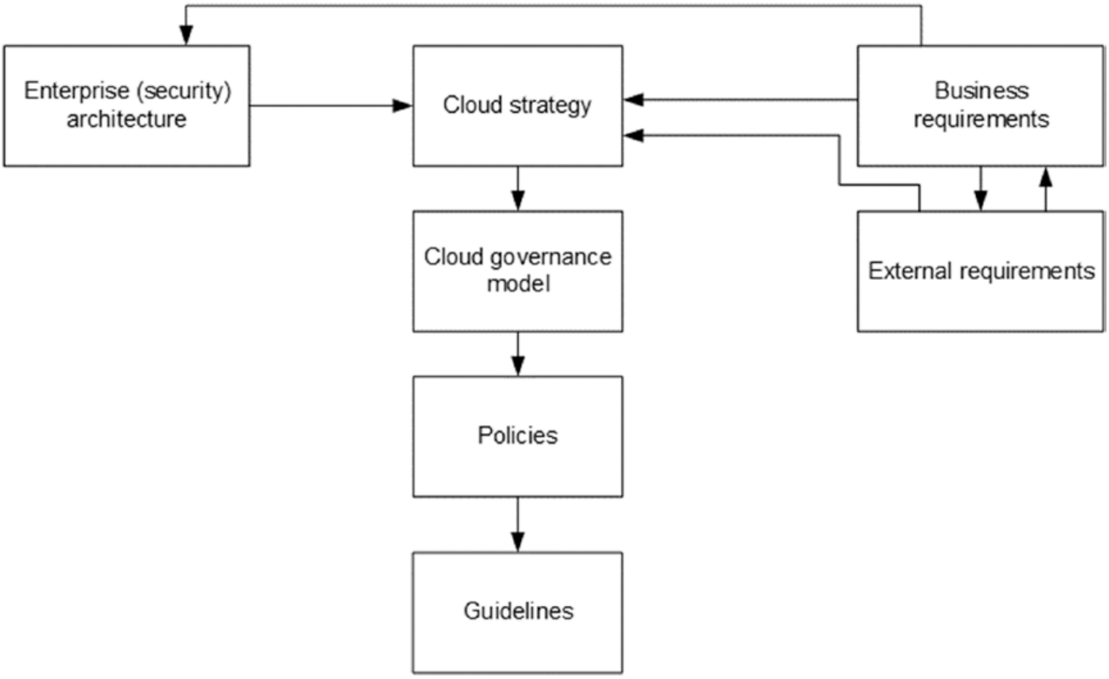


Figure 1-4. *Factors influencing your cloud security framework*

Summary

In this chapter, we introduced the key tenets of cloud-native security. We also learned about the key decisions you need to make to build your cloud security framework.

Chapters 2–4 of this book define the landing zones further. You should focus on these chapters first, if you are representing a central unit, such as enterprise architecture, information technology, or information security.

Chapters 5–8 cover how to secure your products. You may use these chapters also as a hardening guide to protect your individual workloads. If you are a cloud solution architect looking for a checklist to secure a specific Azure component, feel free to skip ahead to the specific chapter that covers your workload.

CHAPTER 2

Identity and Access Management

In this chapter, we are going to learn about one of the fundamental building blocks of cloud-native security – identity and access management. After reading this chapter, you will be able to design secure access control solutions for your Azure administrative and application access.

Azure Active Directory

Identity is the new perimeter in the public cloud world, and there is no identity and access management in Azure without Azure Active Directory (Azure AD or AAD). Regardless of your identity provider (IdP) solution, authentication to Azure subscriptions and resources is always performed by Azure Active Directory.

Overview of Azure Active Directory

Azure Active Directory (Azure AD or AAD) is an identity solution as a service (IDaaS), managed by Microsoft. As it is the sole supported authentication solution for Microsoft's own SaaS services, such as the Microsoft 365 suite ranging from Teams to Exchange Online, Azure AD is as ubiquitous in the cloud as Active Directory Domain Services (AD DS) is in the on-premises world. With support for single sign-on, Azure AD can extend to third-party SaaS services outside Microsoft, too.

Azure AD is delivered as an evergreen software as a service (SaaS) by Microsoft. As such, the operational obligations of IT departments are replaced by configuration and change management responsibilities. Microsoft is constantly adding features to Azure AD, so simply keeping up with, verifying, and rolling out new features is a challenge.

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_2

Identity Sources

Azure Active Directory supports multiple sources of identities, as described in Figure 2-1. You can provision users and other identities directly into Azure AD, or you can synchronize identities from your existing systems. If your organization is using Active Directory Domain Services, you can synchronize the identities with Azure Active Directory Connect synchronization services. This is the approach most established organizations are taking. As this combines the usage of on-premises and cloud services, it is referred to as **hybrid identity**.

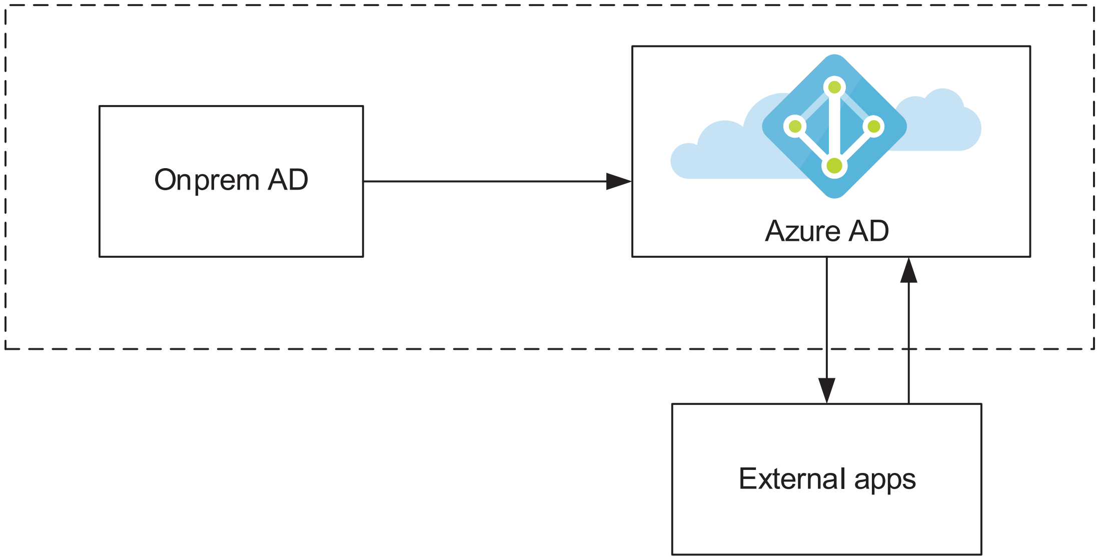


Figure 2-1. *Options for identity sources*

If your organization does not have traditional Active Directory Domain Services in use, you can use cloud-based HR systems to provision Azure AD users.

Furthermore, you can user management APIs of System for Cross-Domain Identity Management (SCIM) to provision users from your Azure AD into other cloud applications. This lets you use Azure AD as your main identity source for other services, such as Adobe, AWS console, Slack, or Salesforce.

Relationship Between Azure Active Directory and Azure

Each Azure subscription is linked to an Azure Active **Directory**, sometimes referred to as an **account** or a **tenant**. However, not all directories are linked to Azure subscriptions. This is the case when, for example, an organization is only using Microsoft SaaS services, but not (yet) any Azure services.

> Note that even if there has not been any formal usage of Azure, there could still be several Azure subscriptions linked to the Directory. Any new subscriptions created by users in your Directory are by default linked to the Directory. There is no approval needed from any elevated Azure Active Directory Role assignee.

As all Azure subscriptions are linked to an Azure AD, some Azure services leverage it in a highly integrated manner. This lets you control authentication in an effective and centralized manner across your Azure subscriptions. This link is important to understand across your whole application life cycle. If your Azure resources are moved to another Directory in case of mergers, outsourcing, or other business reasons, you need to remove and re-create the link between your services and Azure AD. As of the writing of this book, this applies to the following services:

- Azure SQL database
- Azure Key Vault
- Azure role-based access control (RBAC) assignments

Intelligent Security Graph

The Intelligent Security Graph is a collection of security-related data points gathered across the Microsoft 365 suite of services. As Azure AD is used for authentication for each of those services, the Graph can combine the data in an intelligent manner. Altogether, Azure AD is used for several hundreds of billions of authentications per month. In addition to data from their first-party services, Microsoft collects data from other sources, such as external feeds and dark markets. Altogether, Microsoft collects over eight million indicators of compromise (IoC) per day.

Before using this data from various sources, several privacy and compliancy boundaries need to be passed. The data is anonymized and synthetized before being fed into the Graph.

Conditional Access

Based on this large and diverse set of data, the Graph can provide us with risk ratings for each authentication event. These risk scores can be leveraged automatically to take a risk-based approach for securing access, such as blocking an authentication attempt by a user that is trying to sign in from geographically distant locations within a short amount of time (so-called impossible travel scenario). This is available as a feature called **conditional access**.

Conditional access lets you configure rules for automatically applying certain controls for any authentication events across our identity perimeter. These rules can be as straightforward as blocking users attempting to log in from devices infected with malware, or more complex, such as combining network risks with past behavior of the user attempting to authenticate.

After the conditional access evaluates the attempted authentication, it will either allow or deny access. It can even be configured to allow access after certain additional controls, such as multifactor authentication challenge, are satisfied.

Securing Your Azure AD

Securing your Azure AD is such a vast topic that it deserves its own book. For the purposes of this book, I am introducing you briefly to the Azure AD security topics that have the most direct impact on your Azure usage.

Directory Splitting

Directory splitting, or separation of users and subscriptions of the same organization into multiple Azure AD directories, is not considered a best practice in the cloud era. Microsoft recommends using a single Azure Active Directory per organization in the Cloud Adoption Framework,[1] as illustrated in Figure 2-2. Most of your isolation requirements can be met with Azure RBAC and separation of subscriptions, instead of Azure AD directories.

[1] https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/decision-guides/identity/

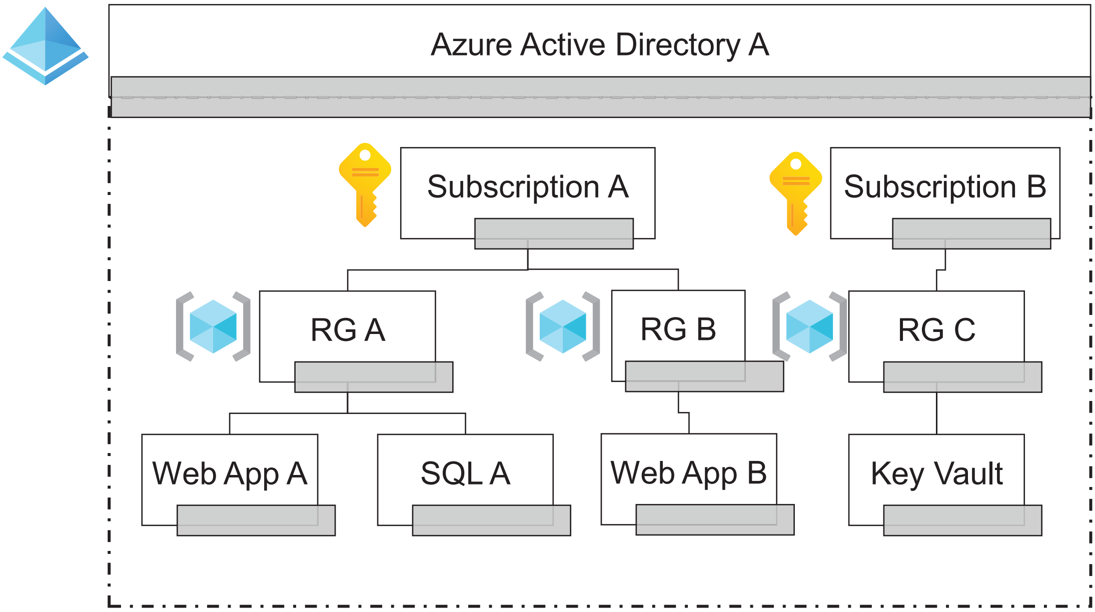


Figure 2-2. *The relationship between the Azure AD directory and subscriptions*

If your organization has already experimented with Azure usage, you might have multiple Azure AD directories. In that case, you might need to start your Azure security journey by consolidating your subscriptions under a single directory. One of the byproducts of this consolidation exercise is that you will need to re-create all RBAC assignments.[2] I encourage you to use this opportunity to enforce your new access control standards.

As always, there are some exceptions to this. You might operate in a business environment, such as joint ventures, where a separate Azure AD directory is necessary. Or you might be using one of the sovereign clouds, such as Azure US Government or Azure China. These sovereign clouds are already by nature disconnected from the global Azure infrastructure, including Azure AD.

[2] https://docs.microsoft.com/en-us/azure/role-based-access-control/transfer-subscription

Guest Management

You should set controls on Azure AD B2B guest management. Without proper controls, any regular member in your Azure AD can invite guests from virtually anywhere! The guests do not need to belong to the Azure ADs of your trusted partners or even any Azure ADs. Microsoft personal accounts and Google accounts are among the possible identity providers for guests.

Once a guest user has been invited and they have accepted the invitation, the guest can be assigned to any Azure AD administrative roles and Azure RBAC roles. This means that you need to be able to trust the identity of the guest user with limited visibility to their activities!

Note Guest users can be assigned to the same Azure AD administrative roles and Azure RBAC roles as native members of your Azure AD directory.

You can use the **Guests can invite** and **Members can invite** settings in your Azure AD to control who can invite guests. When both are set to No, only users with Azure AD administrative roles, such as Guest Inviter, can invite guests. Additionally, you can choose from the following **collaboration settings** to further control guest inviting:

- Allow invitations to be sent to any domain (most inclusive).
- Deny invitations to the specified domains.
- Allow invitations only to the specified domains (most restrictive).

As you do not have full visibility to the guest user identities, it is worth considering adding a conditional access policy to require multifactor authentication from guest user authentications to Azure Management plane (labeled **Microsoft Azure Management** in Azure AD Conditional Access).[3]

[3] https://docs.microsoft.com/en-us/azure/active-directory/external-identities/b2b-tutorial-require-mfa

Default Access Management

Contrary to Azure RBAC, every Azure AD user (including guest) has a set of default permissions.[4] I recommend you control the following permissions to secure your Azure environment:

- Users can register application.
- Ability to create security groups.
- Restrict access to Azure AD administration portal.

Privileged Access Management

Azure subscriptions trust their linked Azure AD directories. A key feature of this trust is the fact that Azure AD Global Admins can self-elevate themselves as User Access Administrator to any Azure resources that are linked to your Azure AD.[5]

Note Azure AD Global Admins can take over any Azure subscriptions linked to your Azure AD!

There are several controls available for you in securing privileged Azure AD users. First, to mitigate compromised on-premises users pivoting to the cloud,[6] or vice versa, do not synchronize accounts with high Active Directory privileges to Azure AD.

Second, you should define two or more emergency accounts (sometimes called Break-the-Glass accounts) that are as resilient to disruptions in Azure AD as possible. These users should be created directly in the cloud, that is, using your *.onmicrosoft.com domain instead of your domain. They should be exempt from your regular conditional

[4] https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/users-default-permissions#compare-member-and-guest-default-permissions

[5] https://docs.microsoft.com/en-us/azure/role-based-access-control/elevate-access-global-admin

[6] https://techcommunity.microsoft.com/t5/azure-active-directory-identity/protecting-microsoft-365-from-on-premises-attacks/ba-p/1751754

access policies, as well as phone-based multifactor authentication.[7] These accounts should have the highest level of monitoring controls applied, and their usage should be extremely limited.

Third, you should only use Azure AD accounts with your Microsoft Billing portals, to mitigate attacks against personal Microsoft accounts.

Finally, you should separate your user accounts from Global Admin accounts.

Conditional Access Policies

To complement your Azure RBAC access control, you should consider setting conditional access policies to protect the following authentications using MFA:

- Authentication of Azure AD administrators.
- Authentication of any user to Azure Management (through Azure Portal or a command-line interface).
- Authentication using a legacy authentication protocol. This setting is especially effective in mitigating password spray attacks, as it enforces MFA.

In addition to enforcing MFA, you can also control the network locations or devices allowed to perform the authentication.

Authorization: Azure Role-Based Access Control

All access to any Azure resources needs to be explicitly granted using Azure role-based access control (RBAC). Figure 2-3 decomposes the RBAC assignment, the key unit of access control in Azure.

[7] https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-resilient-controls

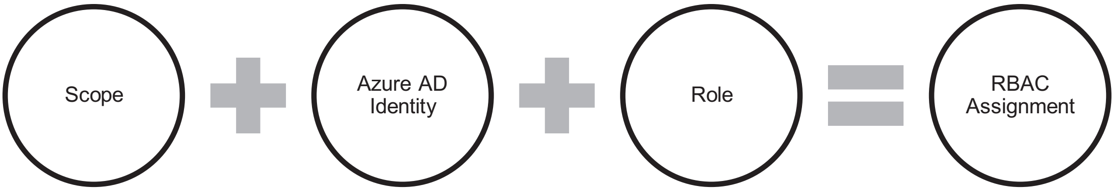


Figure 2-3. *Azure role-based access control*

In addition to role-based access control, Azure has recently introduced support for attribute-based access control (ABAC).[8] This gives you more granularity to access control, by requiring specific attributes from the scope or the Azure AD Identity, in addition to the RBAC role assignment. As of the writing of this book, ABAC is supported in Azure storage account. We will discuss ABAC in Chapter 5 of this book.

Scope

Azure role-based access control assignment starts with the **scope** of the assignment. A scope is the target of the assignment and can be one of the following:

1. Management group
2. Subscription
3. Resource group
4. Resource, such as a Cosmos DB account
5. Sub-resource, such as a subnet of a virtual network resource

Access assigned to a higher level of scope automatically grants access to lower levels. This inheritance applies throughout the life cycle of the RBAC assignment: access granted to a subscription will grant access to every resource group, resource, and sub-resource in that subscription starting from the assignment time. This means that this same RBAC assignment will automatically grant you access to every new resource any user will add to the subscription, anytime in the future.

[8] https://csrc.nist.gov/publications/detail/sp/800-162/final

Identity

Identity refers to the subject of the Azure RBAC assignment. The identity can be one of the following:

- **User** in the Azure AD linked to the subscription. Both member users and guest users are accepted.
- **Group** in the Azure AD linked to the subscription.
- **Application** registered to the Azure AD or created as a managed identity for Azure resources (see section managed identities for Azure resources for more details).

As the RBAC assignment subject is always an entity in your Azure AD, any changes to those entities are also reflected in the effect of RBAC assignment. For example, if a developer leaves your organization and their user is disabled in your identity provider, that information is synchronized to your Azure AD and the user will not be able to successfully authenticate, even if their RBAC assignment was not removed.

Note Regardless of your approach to Azure AD B2B, you should always avoid creating RBAC assignments to guest users that have not yet accepted the invitation to the Directory! The guest users can accept the invitation to join your Azure AD from a variety of identity providers, not all of which may meet your security requirements.

Role

As there are no default RBAC assignments in place for users, you need to always allow any kind of access explicitly to your Azure environment. A **role** of the Azure role-based access control defines a list of **Resource Provider actions** that are allowed or not allowed in the **scope** of the assignment. Identities can be assigned multiple roles simultaneously, in which case the resulting permission is the sum of all permission granted in the assigned roles.

Role Definition

Resource Providers are collections of APIs that provide functionality for all Azure services. This functionality may vary from listing resource properties to virtually any functionality possible with the Azure service you are working with. For example, the `Microsoft.Storage/storageAccounts/listkeys/action` provides the functionality to request the shared account keys of a storage account.

A role definition describes which Resource Provider operations are available for the assignee of the role in the action block of the role definition. The role definition can also include exclusions, in the notActions scope. If a user attempts to perform an action not permitted by their role, they will not succeed and are presented with an error message. The unsuccessful attempt is also logged in the activity log.

Note that the Reader role does not grant you access to read data stored in your Azure resources, such as the content of a storage account. The Reader role only grants access to the control plane. To manage access to content, you need to control expand the role definition to consider dataActions and notDataActions. With these, the role definitions can also control access to the **data plane** of Azure resources. The Storage Blob Data Reader role grants you access to read the data stored in the storage account.

The role definition syntax allows you to use wildcards. Typically, they are used to cover all operations of a Resource Provider, such as `Microsoft.Web/*`. But the wildcard can also be used to replace any other part of the Resource Provider operation path. In fact, the Owner and Reader roles are defined using this syntax.

Built-In Roles

There are several **built-in RBAC roles** available in Azure. Each Azure Product Group is responsible for designing, threat modeling, and updating their built-in roles whenever they introduce new functionality (new operations to their Resource Providers). I call these service-specific built-in roles. In addition, there are some general built-in roles that you should look at more closely, especially at the beginning of your Azure RBAC implementation.

General Built-In Roles

Owner grants you access to manage any operations in any Resource Provider. This includes the Resource Provider `Microsoft.Authorization`. In practice, this role lets you create, read, update, and delete any resource or RBAC assignment. The Owner role is a powerful role, and its usage should be minimized. Owner is the default role for any new subscriptions. The role definition at Listing 2-1 describes the explicit permissions of this role.

Listing 2-1. Permissions of the Owner built-in role

```
"permissions": [
    {
      "actions": [
        "*"
      ],
      "notActions": [],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

User access admin grants you access to the `Microsoft.Authorization` Resource Provider. Whenever possible, you should replace your justified Owner assignments with assignments to User Access Administrator. The role definition at Listing 2-2 describes the explicit permissions of this role.

Listing 2-2. Permissions of the User Access Administrator built-in role

```
"permissions": [
    {
      "actions": [
        "*/read",
        "Microsoft.Authorization/*",
        "Microsoft.Support/*"
      ],
      "notActions": [],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

Note Azure does not prevent you from creating new Owner and User Access Administrator role assignments, even while you have been assigned to either of the roles only temporarily using Privileged Identity Management. Any RBAC assignments you assign directly will stay in effect until explicitly removed.

Contributor grants you access to create, read, update, or delete any resource. When it comes to access control, the Contributor differs from the Owner role. The Contributor role has only read access to the `Microsoft.Authorization` Resource Provider. In practice, the Contributor is the most highly privileged role you should assign to developers. Once your RBAC practices mature, you should start replacing your Contributor assignments with either service-specific built-in roles or Contributor roles assigned in lower scopes. The role definition in Listing 2-3 describes the explicit permissions of this role.

Listing 2-3. Permissions of the Contributor built-in role

```
"permissions": [
    {
      "actions": [
        "*"
      ],
      "notActions": [
        "Microsoft.Authorization/*/Delete",
        "Microsoft.Authorization/*/Write",
        "Microsoft.Authorization/elevateAccess/Action",
        "Microsoft.Blueprint/blueprintAssignments/write",
        "Microsoft.Blueprint/blueprintAssignments/delete",
        "Microsoft.Compute/galleries/share/action"
      ],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

Reader role grants the /read action to any Resource Provider. In practice, this role has access to read control plane data, but not data plane. This means that the Reader role does not let you read the content of your resources, such as files stored in a storage account.

Listing 2-4. Permissions of the Reader built-in role

```
"permissions": [
    {
      "actions": [
        "*/read"
      ],
      "notActions": [],
      "dataActions": [],
      "notDataActions": []
    }
  ]
```

Service-Specific Built-In Roles

To grant access to manage data stored in a storage account, you should use one of the built-in service-specific roles for storage.[9] For read access, you should assign the Storage Blob Data Reader role, which contains the permissions defined in Listing 2-5.

Listing 2-5. Permissions for the Storage Blob Data Reader role

```
"permissions": [
    {
      "actions": [
        "Microsoft.Storage/storageAccounts/blobServices/containers/read",
        "Microsoft.Storage/storageAccounts/blobServices/
        generateUserDelegationKey/action"
      ],
```

[9] https://docs.microsoft.com/en-us/rest/api/storageservices/authorize-with-azure-active-directory#manage-access-rights-with-rbac

```
      "notActions": [],
      "dataActions": [
        "Microsoft.Storage/storageAccounts/blobServices/containers/blobs/
        read"
      ],
      "notDataActions": []
    }
  ]
```

Custom Roles

In addition to built-in roles, it is possible to define role assignments yourself. Custom roles are defined using the same syntax as built-in roles. They can also be assigned to the same identities or scopes as any built-in roles. Once you know the specific Resource Provider actions needed for your use cases, it is tempting to start replacing built-in role assignments with custom roles. While this presents the opportunity for granular access control and the application of the principle of least privilege, I recommend you consider custom roles only in exceptional cases.

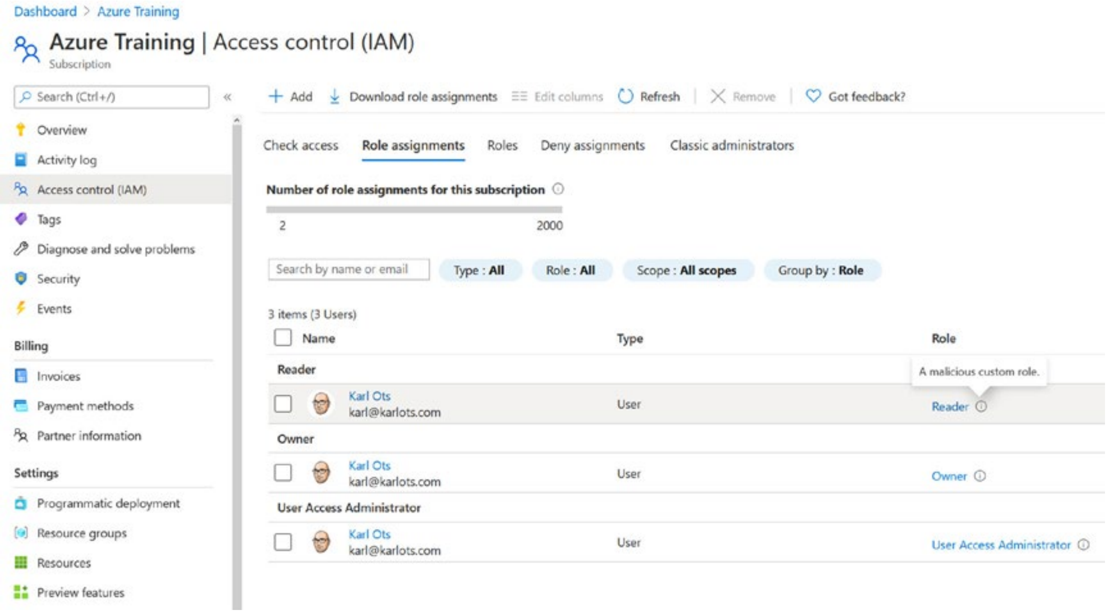
You are responsible for threat modeling and updating your custom roles. Whenever there is new functionality release to the Resource Providers, you need to re-evaluate whether your role definition is still resulting in the expected behavior. Tracking Resource Provider changes remains your responsibility, as there are no out-of-the-box alerts available for them.

Custom roles might present you with new governance challenges. Your organization needs to establish change management processes for any custom roles introduced to your Azure environment to maintain consistency on naming and control the number of roles present in your environments. There are even hard limitations on the number of custom roles you can have deployed simultaneously.[10]

[10] https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits#azure-role-based-access-control-limits

Account Manipulation

Uncontrolled use of custom roles could leave you vulnerable to account manipulation.[11] Custom role names can be tricked to use names already used by built-in roles as demonstrated in Figure 2-4.



Number of role assignments for this subscription

2 2000

Search by name or email | Type : **All** | Role : **All** | Scope : **All scopes** | Group by : **Role**

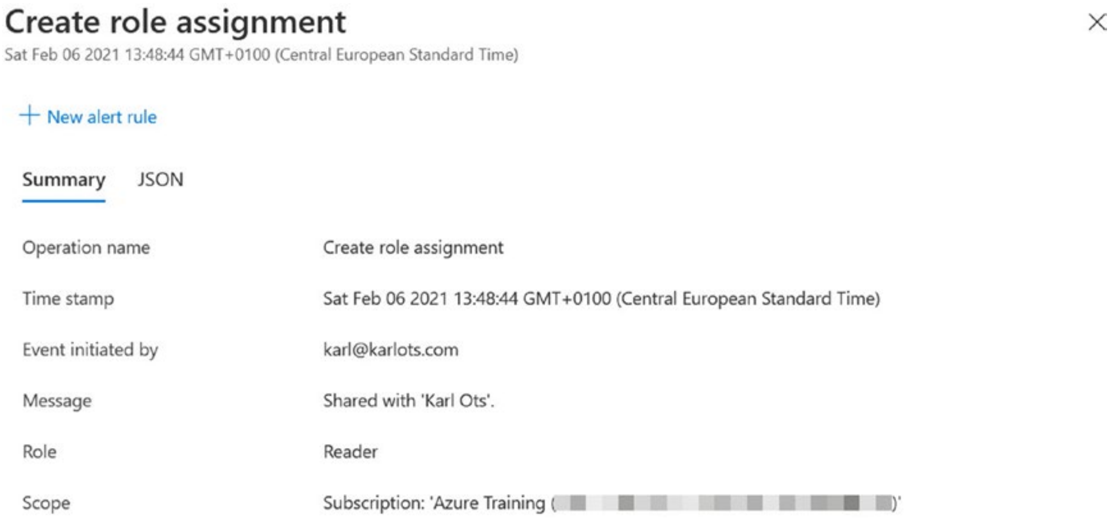
3 items (3 Users)

	Name	Type	Role
Reader			
□	Karl Ots karl@karlots.com	User	Reader ⓘ (A malicious custom role.)
Owner			
□	Karl Ots karl@karlots.com	User	Owner ⓘ
User Access Administrator			
□	Karl Ots karl@karlots.com	User	User Access Administrator ⓘ

Figure 2-4. *A malicious custom role assigned with an innocent-looking built-in role name*

Role assignments to custom roles can look the same as assignments to built-in roles. Figure 2-5 shows how the role assignment looks indistinguishable from a role assignment to a built-in role in the Azure activity logs.

[11] https://attack.mitre.org/techniques/T1098/



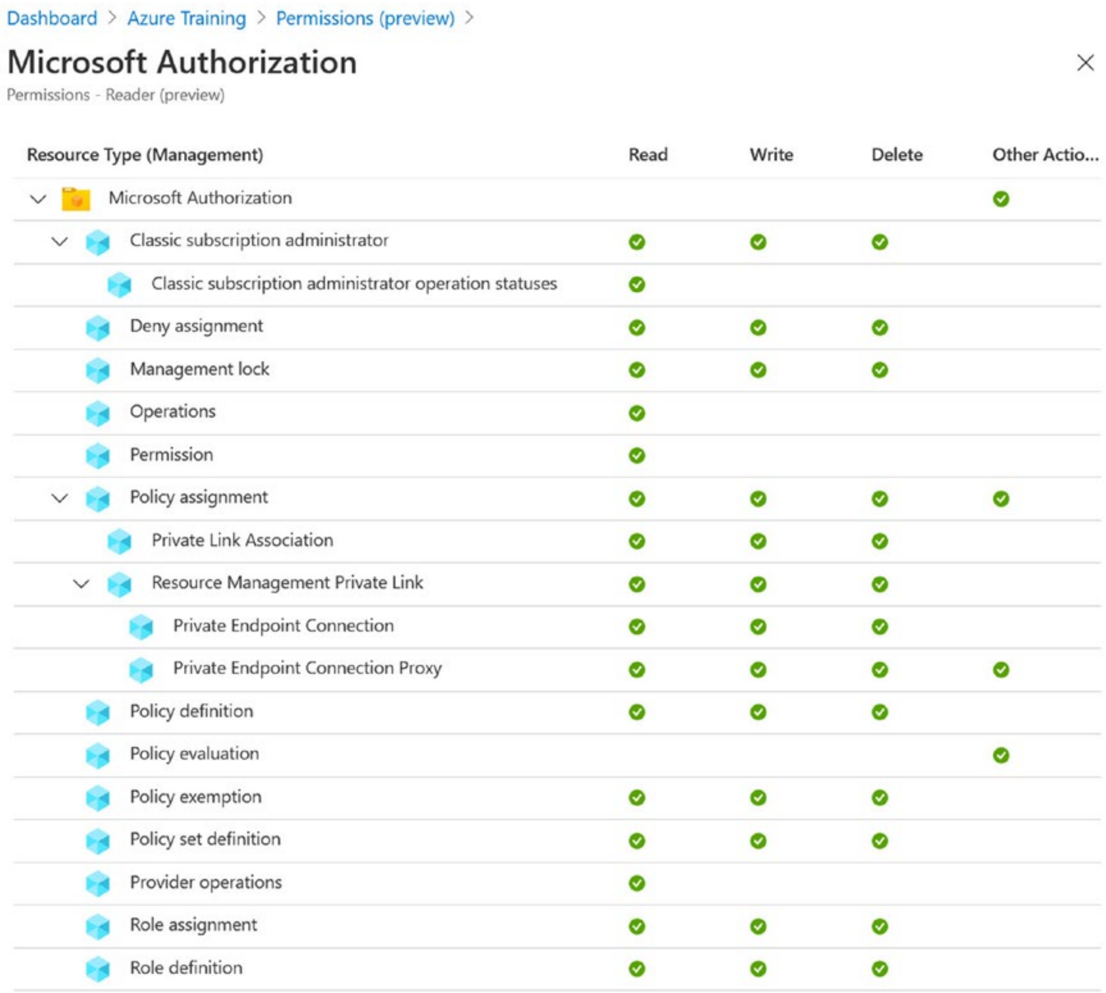
Create role assignment

Sat Feb 06 2021 13:48:44 GMT+0100 (Central European Standard Time)

Operation name	Create role assignment
Time stamp	Sat Feb 06 2021 13:48:44 GMT+0100 (Central European Standard Time)
Event initiated by	karl@karlots.com
Message	Shared with 'Karl Ots'.
Role	Reader
Scope	Subscription: 'Azure Training ([redacted])'

Figure 2-5. *Activity log entry of a role assignment to a custom role*

You should train your users and analysts to detect custom roles. Users with the built-in Reader role can enumerate the permissions for any role with the `az role definition list` command. Figure 2-6 shows the similar view in the Azure Portal. In this example, the malicious custom role has effectively the same permission as the built-in Owner role. The built-in Reader role does not allow any Write, Delete, or Other Actions.



Microsoft Authorization

Permissions - Reader (preview)

Resource Type (Management)	Read	Write	Delete	Other Actio...
Microsoft Authorization				✓
Classic subscription administrator	✓	✓	✓	
Classic subscription administrator operation statuses	✓			
Deny assignment	✓	✓	✓	
Management lock	✓	✓	✓	
Operations	✓			
Permission	✓			
Policy assignment	✓	✓	✓	✓
Private Link Association	✓	✓	✓	
Resource Management Private Link	✓	✓	✓	
Private Endpoint Connection	✓	✓	✓	
Private Endpoint Connection Proxy	✓	✓	✓	✓
Policy definition	✓	✓	✓	
Policy evaluation				✓
Policy exemption	✓	✓	✓	
Policy set definition	✓	✓	✓	
Provider operations	✓			
Role assignment	✓	✓	✓	
Role definition	✓	✓	✓	

Figure 2-6. *Enumerating permissions of a custom role in the Azure Portal*

EXERCISE

Consider the Azure environment described in Figure 2-7.

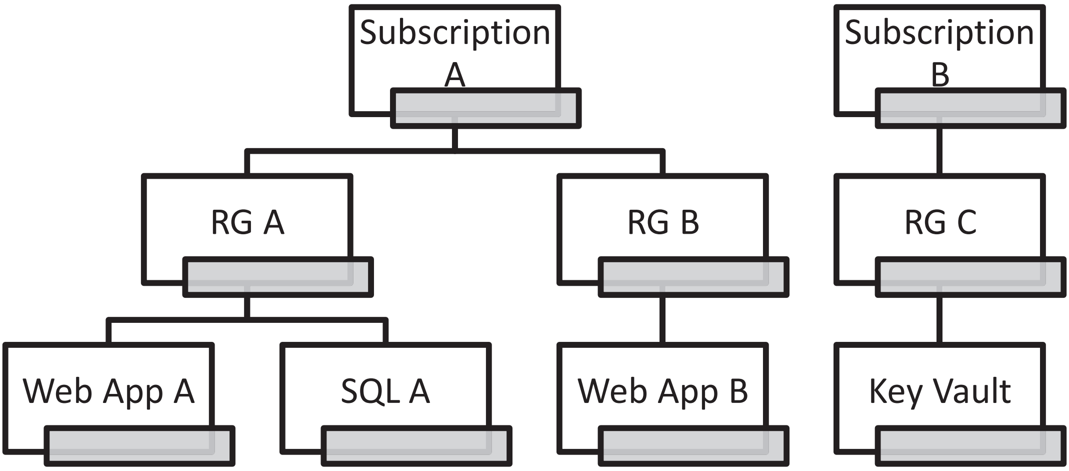


Figure 2-7. RBAC inheritance

You need to grant access to two Azure developers while ensuring the access is granted following the principle of least privilege:

1. Developer A needs to create, read, update, and possibly delete resources in resource group A.
2. Developer B needs to configure the settings of Web App B to read data stored in the Key Vault.
3. Web App B should be able to read data from SQL A.

Your answer should include both Azure role-based access control assignments and data-plane access control.

Advanced: How could you use policies and locks to further limit the access control of this environment?

Azure AD Administrative Roles

In addition to RBAC roles, you also need to consider whether your users in Azure need any Azure AD administrative roles. Azure AD roles do not have granular scopes, but rather provide elevated access across the whole directory. Key roles to consider from Azure perspective are

- **Application Administrator,** which grants the user access to manage enterprise applications, application registrations, and application proxy settings
- **Application Developer**, which grants the user access to create enterprise applications and application registrations. Users assigned to this role are automatically added as owners of the application they create. Applicable regardless of the value of the *Users can register applications* setting
- **Guest Inviter**, which grants the user access to invite Azure AD B2B Guest users. Only applicable when the directory user setting *Members can invite* is set to No

Note Assigning a user to the Application Administrator role gives them the ability to impersonate an application's identity, which can lead to an elevation of privilege.[12]

Assignment Life cycle

Azure RBAC assignment life cycle can vary based on your governance requirements and cloud security approach. RBAC assignments are perpetual, that is, they last throughout the life cycle of the scope they are assigned to. This can add complexity when evaluating appropriate roles and considering scope inheritance.

[12] https://docs.microsoft.com/en-us/azure/active-directory/roles/permissions-reference#application-administrator

Note You cannot edit RBAC assignments. Instead, you would need to delete the assignment you wish to modify and create a new one.

You have three main options when considering RBAC assignment life cycles.

The most common approach is to create **static RBAC assignments to each user** directly. This is the default approach and provides your application teams with the greatest agility. This agility comes with a cost of complexity and lack of central control, however. With this approach, your application development teams can manage the access themselves, bypassing your existing access management processes.

The next option would be to use **static RBAC assignments to AAD groups** with dynamic group membership assignments. In this approach, you pre-create RBAC assignments and accompanying AAD groups. Instead of making changes to the Azure RBAC assignments, access is granted by adding the users as members of the pre-assigned AAD groups. This is the likely approach when you want to continue utilizing your existing identity and access management concepts, processes, and tools. There are downsides to this approach, however. This approach requires a significant amount of upfront planning, in contrast to the perceived agility of cloud adoption. This approach also requires you to create a large number of Azure AD groups, which could result in reaching Azure AD service limits in larger enterprises.

The third option you have is to use **dynamic RBAC assignments using Privileged Identity Management** (PIM). This lets you minimize both the number of RBAC assignments and the need to create AAD groups per scope and per role. Managing access with PIM allows you to assign **just-in-time access** to your users: PIM access is automatically removed by the PIM system after the approved time.

Policies

Policies are an additional option for controlling allowed Resource Provider actions within the same **scope** as RBAC assignments. Once you are authenticated with Azure AD, and authorized to perform certain Resource Provider actions, your request is sent to Azure Resource Manager. Before the Azure Resource Manager sends your request to a respective Resource Provider, the request is filtered through the policy engine, which checks whether any policies are in effect for the scope you want to perform your action. Figure 2-8 illustrates the relationship between Azure RBAC roles and policies.

Note Even if the role you have been assigned grants you the permission to perform the requested action, a policy can still deny you from performing it. Even an Owner role assignment does not allow for changes denied by policies.

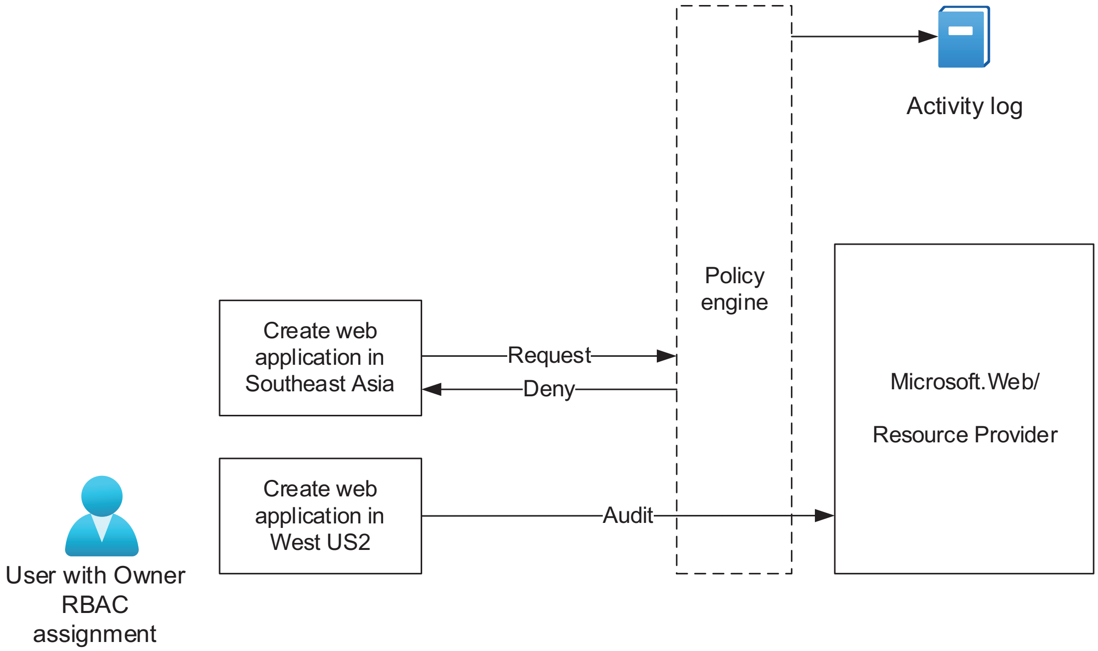

Activity log
Policy engine
Create web application in Southeast Asia
Request
Deny
Microsoft.Web/
Resource Provider
Create web application in West US2
Audit
User with Owner RBAC assignment

Figure 2-8. *Owner attempting to create a web app within a scope where Allowed Locations policy is in effect*

Policy Effects

A policy effect determines what happens when the policy is applied. As of the writing of this book, the following policy effects are supported:

- Append
- Audit
- AuditIfNotExists
- Deny
- DeployIfNotExists

- Disabled
- Modify

In addition to complementing access control, policies are one of the key Azure features used for monitoring and enforcing compliance. Several governance goals, such as geo-compliance, tagging, and service catalogues, can be achieved with policies. As we learn in Chapter 3, policies are also one of the core tools used for detection and monitoring.

Like RBAC assignments, policies are assigned to a **scope** and the same inheritance rules apply. Contrary to RBAC assignments, however, policies support exceptions to the scope. This allows you to assign policies in high scopes of the inheritance hierarchy to enforce controls across your whole Azure environment.

Locks

In addition to RBAC assignments and policies, you can control access to Azure resources using **locks**. Locks are recommended to prevent from accidental misuse of resources. Like RBAC assignments and policies, locks are assigned to a **scope**. Locks should not be considered effective controls against malicious users, as the locks themselves can be removed by Owners or User Access Administrators of the scope they are assigned in. There are two types of locks:

- **Delete** locks allow authorized users to perform Create, Read, and Update operations, but not Delete.
- **Read-only** locks allow authorized users to perform only Read operations. This lock effectively turns all RBAC assignments into Reader for the scope it is set in.

Note The Read-only lock also applies to the ***Microsoft.Authorization*** Resource Provider, effectively preventing you from making changes to RBAC assignments.[13]

[13] https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/lock-resources#considerations-before-applying-locks

Managed Identities and Service Principals

A common challenge when building cloud applications is how to securely manage the credentials of your application code for authenticating to other applications, APIs, or even Azure itself. The use cases may include

- Front-end to back-end access
- API to API access
- Resource deployment access

Managed identities for Azure resources automatically create and manage an identity in your Azure AD for your Azure resources without exposing the used credentials to you. Your application can use this identity to authenticate any service that supports Azure AD authentication.

If a malicious actor is able to access the authentication credentials of a service principal, they are able to use it anywhere without MFA. As of the writing of this book, service principal authentication bypasses any conditional access policies. Microsoft has communicated that they are working on enabling policies based on IP address range but have not committed to a timeline.

If you need to use service principals, use certificate credentials for authentication, instead of password credentials (client secrets).

Access Control Throughout the Secure Development Life cycle

What is the appropriate level of access to Azure environment for developers? The answer to this question spans from meaningful developer access in isolated sandbox environments to appropriate access in production.

Developer Sandbox Access

To enable faster time to market and agile evaluation of new cloud technologies, developers need a level of freedom that would have been unthinkable before the era of the cloud.

In staging and production environments, any infrastructure and configuration changes should be done through infrastructure as code, which can be templatized, version controlled, and tested. Before the developers know what to build, however, they need access to an environment with less restrictions where they can explore the available services. For the purposes of this book, I am calling these **sandbox environments**.

A sandbox environment in Azure can be as simple as a dedicated Azure subscription or a resource group. It might not necessarily differ from your other environments in any technical implementation. It is relatively easy and cost-effective to create Azure sandbox environments, but you should beware of lateral movement attacks! Even if there are no shared resources such as databases or network connectivity, your sandbox environments might still be lined to the same Azure AD tenant, allowing for account discovery attacks that are not limited to the sandbox environment.

Note Beware of account discovery, lateral movement, and denial of wallet attacks when managing developer sandbox access in Azure!

Your options for creating developer sandbox environments in Azure are summarized in Table 2-1.

Table 2-1. *Options for Azure sandbox environments*

Option name	Sandbox scope	RBAC roles	Azure AD roles
Business as usual	Subscription	Direct assignment to Contributor	AAD administrative roles according to standard processes Application registrations according to standard processes
Cost-optimized business as usual	Subscription with Dev/Test Offer type	PIM eligibility assignment to Contributor	AAD administrative roles according to standard processes Application registrations according to standard processes

(*continued*)

Table 2-1. (*continued*)

Option name	Sandbox scope	RBAC roles	Azure AD roles
Shared sandbox	Resource group in a subscription with a Dev/Test offer type	Direct assignment to Contributor	AAD administrative roles according to standard processes Application registrations according to standard processes

Continuous Deployment Pipeline Access

Most deployments to your Azure environments are not performed by a developer accessing Azure using their own credentials directly. Instead, the most common model for deploying to production would be to use a continuous deployment pipeline that performs the resource deployment in a controlled way. Centralized deployments to Azure can be done declaratively using various infrastructure-as-code template methods, or imperatively using Azure PowerShell or Azure CLI.

Figure 2-9 illustrates how the developers would typically delegate the deployment access to the continuous deployment pipeline. The benefit of this approach is that any changes to the application resources need to go through the continuous deployment pipeline. This method is often accompanied with a controlled execution of static and dynamic security testing and approval gates.

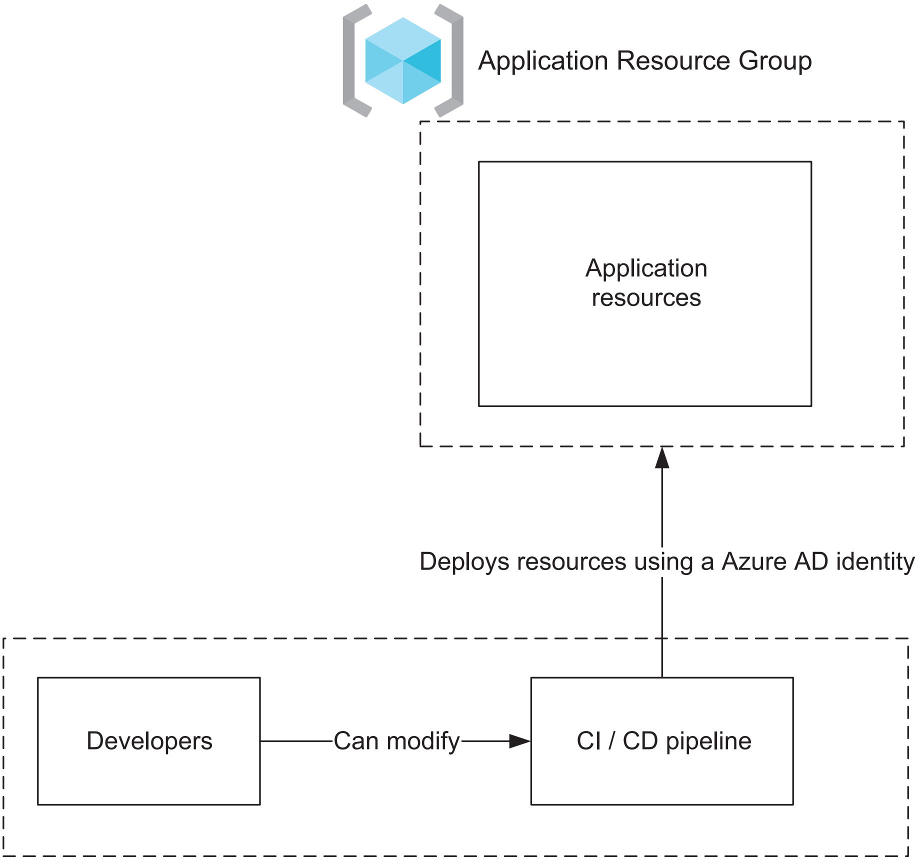

Application Resource Group
Application resources
Deploys resources using a Azure AD identity
Developers
Can modify
CI / CD pipeline

Figure 2-9. *Continuous deployment pipeline access to Azure*

To secure this approach, you need to control the identity used by the continuous deployment pipeline. As this is a system account, you can use either a managed identity or a service principal. If your deployment agent is running in Azure, you should use managed identities. If you are not able to use managed identities, you need to control access to the continuous deployment pipeline in a manner that prevents the developers accessing the service principal authentication credentials as this would let them impersonate the continuous deployment pipeline and deploy changes to the Azure resources without going through the security controls of your pipeline. You will also need to control access to modifying the continuous deployment pipeline.

Summary

In this chapter, you learned about Azure authentication and authorization models and how your approach will influence your cloud security architecture.

We looked more closely at Azure role-based access control and discussed how you can use it across the entire cloud application life cycle.

CHAPTER 3

Logging and Monitoring

Modern applications leverage a variety of cloud services and often span across IaaS, PaaS, and SaaS. In addition, the renewed complexity of applications themselves, the public cloud environments also generate a vast number of signals on their own. Monitoring these environments is different from traditional systems. For example, instead of locking the perimeter down and monitoring activities within your perimeter, you instead abide by the limitations of the cloud provider.

In this chapter, I will introduce you to platform, infrastructure, and application security monitoring and learn about the differences in the various log types. After reading this chapter, you will be able to describe end-to-end monitoring in Azure and select appropriate Azure monitoring tools for your environment.

Platform Monitoring

Traditionally, your applications are hosted in an infrastructure with a varying level of quality and access to monitoring signals. When you host your own infrastructure, you have the opportunity to combine data across your hosting environment. These could include CCTV feeds, physical access logs in and out of the datacenter building, HVAC systems, network components, as well as health of the hardware, operating systems, and applications of your applications. Integrating monitoring data from all these systems would be costly, but not impossible. After taking on such an exercise, you would be faced with massive number of signals to monitor. Identifying any security incidents or events would be like trying to find a needle in haystack.

This is different in an environment where some or all the parts are outsourced to a third party. If your datacenter is co-located with other customers, you will not have an unlimited access to the physical security information. Similarly, if you are consuming compute services from a hosting provider, they are not necessarily able to share the hypervisor-level monitoring data to protect their other customers. In some cases, the

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_3

company is providing you with physical infrastructure, and managing your servers might even be competing, and it will not be in their best interest to share everything in a transparent manner.

All this changes, with the move to public cloud. Everything in Microsoft Azure is software defined; you have signals available to you from across your hosting platform. The signals are available to you in a standardized manner, which makes it easier to correlate across signal types. And what's more, as Microsoft is responsible for the physical and host security, they provide you with reports with pre-correlated data and in some cases even alerts. Microsoft is constantly analyzing any malicious activity against their infrastructure. This lets them provide you with anonymized security intelligence information, even when your environment is not yet being attacked.

Activity Logs

Each Azure subscription automatically stores platform-level logs as activity logs. The activity logs are immutable and stored for 90 days. They can be further exported using diagnostic settings, ensuring longer retention. The activity log schema groups the log events to categories such as administrative, service health, and security. For a full and up-to-date list of categories, please refer to the activity log schema.[1]

As the activity logs are immutable, you get a reasonable audit trail of key platform events out of the box. The out-of-the-box audit trail contains events such as deletion of resources and changes in access control assignments.

Administrative Activity Logs

This is the main category for Azure activity logs. Azure activity logs monitor any write operations to the management plane of Azure Resource Manager and logs them in the administrative log category. The operations are logged regardless of whether they were successful or not.

[1] https://docs.microsoft.com/en-us/azure/azure-monitor/essentials/activity-log-schema

Note Activity log events are created for unsuccessful write operations, too.

Activity logs consider PUT, POST, and DELETE as write operations to be logged. The activity logs include the following information:

- The attempted operation and the outcome
- Identifying information about the user who initiated the operation, such as User Principal Name, authentication methods, and IP address used
- Additional information, such as a reason for a failed operation

Listing 3-1 illustrates the metadata available in an activity log event. Note that not all this information is populated when the source of the event is using a service principal.

Listing 3-1. A truncated example of an administrative activity log event.

```
"claims": {
 "http://schemas.microsoft.com/claims/authnmethodsreferences":
        "pwd,mfa",
        "ipaddr": "{IP address}",
        "name": "Karl Ots",
    },
    "category": {
        "value": "Administrative",
    },
    "eventTimestamp": "2021-02-14T13:37:59.86869Z",
    "operationName": {
        "value": "microsoft.insights/diagnosticSettings/write",
        "localizedValue": "Create or update resource diagnostic setting"
    }
}
```

Activity Logs: Authorization

Administrative category also stores authorization events. The authorization events are stored for all scopes within your subscription (subscription, resource group, resource). The following RBAC authorization events are logged to activity log in the administrative category:

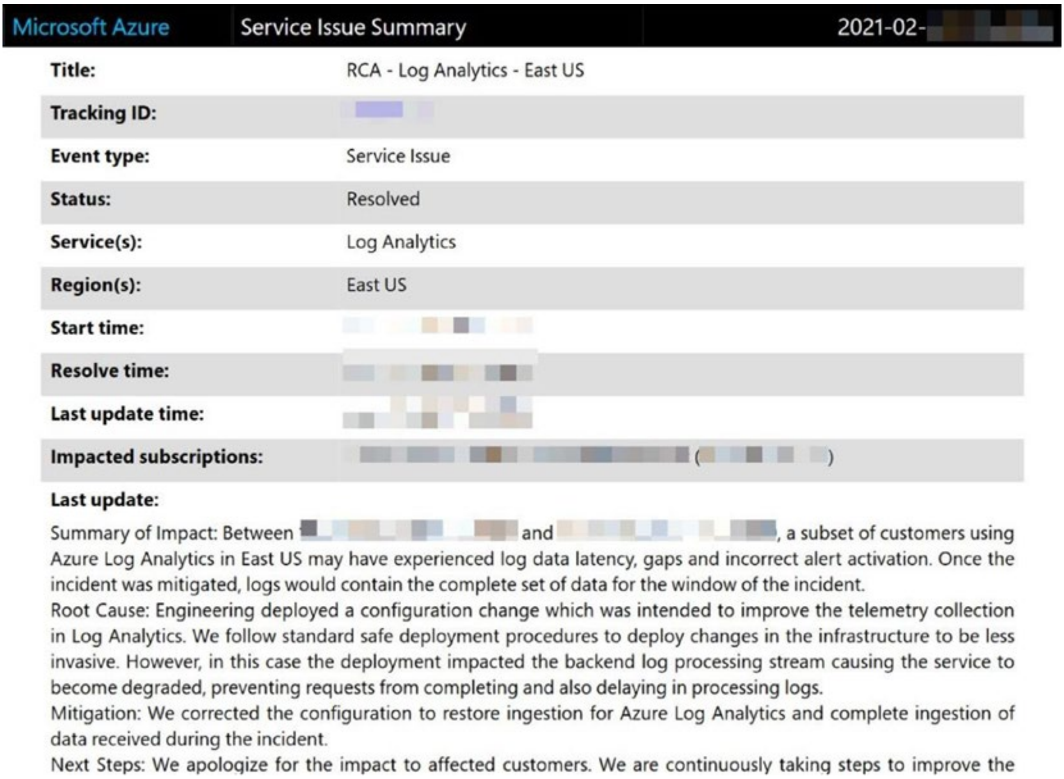
- Create role assignment.
- Delete role assignment.
- Create or update custom role definition.
- Delete custom role definition.

Service Health

Events in this category are emitted by Azure service health. These events originate from Microsoft's own monitoring of their infrastructure. As Microsoft is responsible for both the hardware and software (in case of platform as a service), you could compare these events to events emitted from both your datacenter operations team and your managed services team. You can use service health to set up alerts and get notified when there is an Azure service outage that is impacting you. The possible events include

- **Service issues**: Problems in the Azure services that affect service types you are using in the Azure regions you are using them. You are informed of service issues in real time. Once the issues have been resolved, you are able to export the reports that describe in detail which of your services were impacted, how, and how long.
- **Planned maintenance**: Upcoming virtual machine maintenance that can affect the availability of your services.
- **Health advisories**: Changes in Azure services that require actions from you, such as upgrading the middleware frameworks you are using in your Azure App Service application.
- **Security advisories**: Security-related notifications or violations that may affect the availability of your Azure services.

In case of any broader outages, Microsoft publishes root cause analyses to Azure service health. Both the RCA and regular service issue reports can be exported as a PDF to be stored outside of the Azure systems. Figure 3-1 illustrates the level of detail available in the report.



Microsoft Azure Service Issue Summary 2021-02-[redacted]

Title:	RCA - Log Analytics - East US
Tracking ID:	[redacted]
Event type:	Service Issue
Status:	Resolved
Service(s):	Log Analytics
Region(s):	East US
Start time:	[redacted]
Resolve time:	[redacted]
Last update time:	[redacted]
Impacted subscriptions:	[redacted] ([redacted])

Last update:

Summary of Impact: Between [redacted] and [redacted], a subset of customers using Azure Log Analytics in East US may have experienced log data latency, gaps and incorrect alert activation. Once the incident was mitigated, logs would contain the complete set of data for the window of the incident.
Root Cause: Engineering deployed a configuration change which was intended to improve the telemetry collection in Log Analytics. We follow standard safe deployment procedures to deploy changes in the infrastructure to be less invasive. However, in this case the deployment impacted the backend log processing stream causing the service to become degraded, preventing requests from completing and also delaying in processing logs.
Mitigation: We corrected the configuration to restore ingestion for Azure Log Analytics and complete ingestion of data received during the incident.
Next Steps: We apologize for the impact to affected customers. We are continuously taking steps to improve the

Figure 3-1. *A service issue report downloaded from Azure service health*

Security

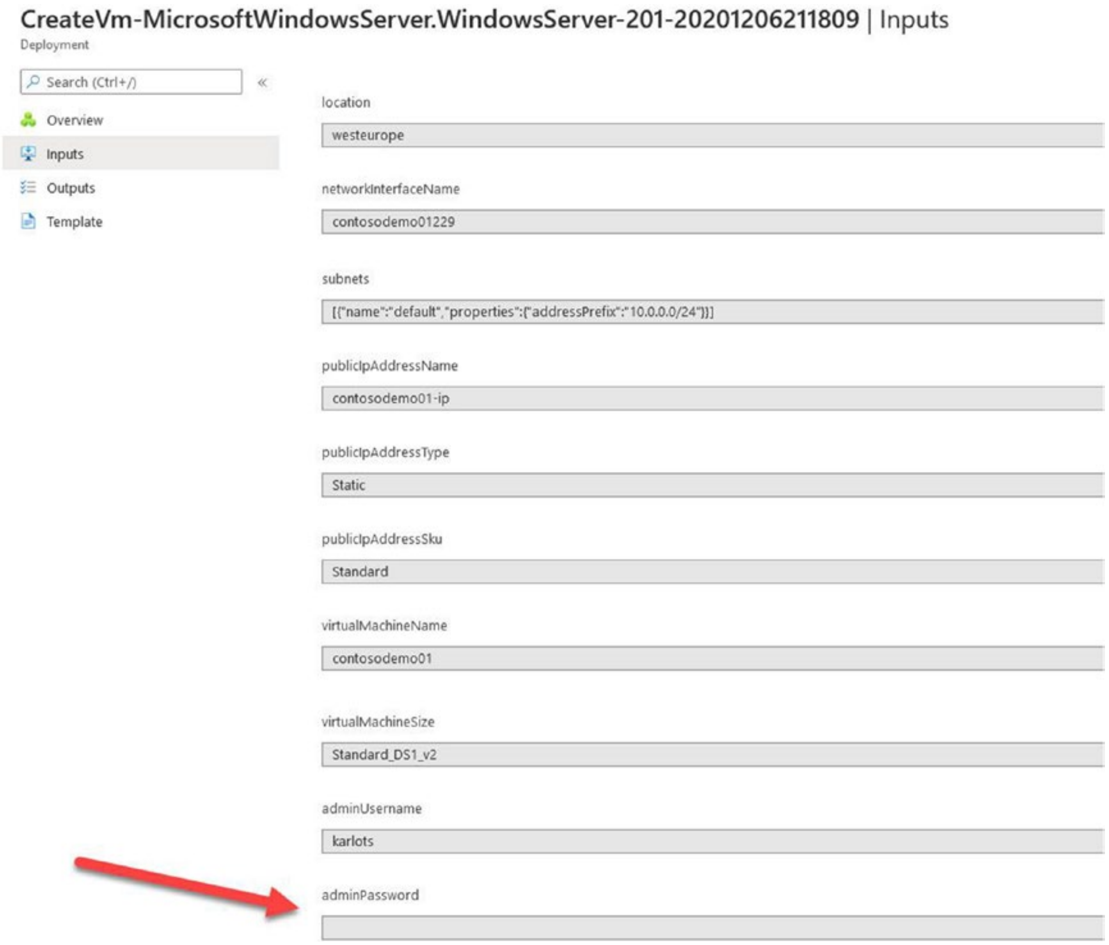
This category contains the record of alerts and security incidents generated by Azure Security Center. Alerts are individual security threats that require your attention. Security incidents combine multiple alerts into a single view. Both alerts and incidents describe the impacted resources, as well as a line the potential attack on the MITRE ATT&CK matrix.

Policy

This category contains records of all effect action operations performed by Azure policy engine. Whenever a new Azure Resource Manager request is evaluated against a policy, it is logged into the activity log. Compliance policies that evaluate existing resources are also logged, so you should be wary off signal noise in this activity log category. I recommend you consider the policy activity log events as complimentary logs for audit and record-keeping purposes. For security posture management perspective, you should monitor policies with Azure Security Center.

Deployment History

Resource groups store management activities into deployment history. The deployment history contains information about the template used for deployment, such as defined resources, used parameters or secrets, and any output values. As deployment history is visible in the Azure Portal to users with reader access or above, it is a best practice to secure any sense to do parameters by using the secure string data type in your templates. By using the secure string type, the values of the parameters are not logged into deployment history, as illustrated in Figure 3-2.



CreateVm-MicrosoftWindowsServer.WindowsServer-201-20201206211809 | Inputs

Deployment

location: westeurope

networkInterfaceName: contosodemo01229

subnets: [{"name":"default","properties":{"addressPrefix":"10.0.0.0/24"}}]

publicIpAddressName: contosodemo01-ip

publicIpAddressType: Static

publicIpAddressSku: Standard

virtualMachineName: contosodemo01

virtualMachineSize: Standard_DS1_v2

adminUsername: karlots

adminPassword:

Figure 3-2. *Deployment history view in the Azure Portal: the adminPassword parameter is defined as a secure string type*

Deployment history is not a write-once-read-many (WORM) log store: deployment history events can be manually deleted like any other resources. Deployment history is stored as metadata in the resource group until they are deleted over the resource group reaches the maximum limit. The maximum number of deployment history items per resource group is 800.

Note Only Azure Resource Manager Deployments provide template history; other infrastructure-as-code methods that use the AZ CLI, such as Terraform, are not trackable in the deployment history. Additionally, the Azure Portal deploys resources using arm templates, so manual deployments from the portal are locked in the deployment history.

If your resource group has 800 deployments stored in its deployment history, the subsequent deployments will fail. To mitigate this, you can manually delete deployment history items, or you can rely on the automatic deletions feature, introduced in 2020. The automatic deletions feature deletes deployment history items in a first-in, first-out manner. The feature aims to keep the number of deployment history items at around 750, but this number is subject to change. If you do not want to use deployment history at all, you can override the name of your deployments. When you deploy a template with the same name as one in the deployment history, the existing deployment history item is replaced.

By correlating the event ID found in your deployment history with the activity logs, you can identify the Azure AD identity used to perform the deploy the template. Listing 3-2 illustrates this.

Listing 3-2. Excerpt from an activity log event which correlates to a deployment history item

```
  "correlationId": "7a34bd43-f8aa-46d6-af5e-a37a73e1a3eb",
"operationId": "a36eefdd-17cb-4abc-a8b7-da79652e121d",
"operationName": {
    "value": "Microsoft.Resources/deployments/write",
    "localizedValue": "Create Deployment"
```

Azure AD Monitoring

Azure AD provides two main log categories. **Activity logs** include activities such as user sign-ins and changes made to Azure AD resources (users, groups, roles etc.). **Security logs** include correlated information from Azure Information Protection. Security logs include risky sign-in logs and reports for users that are flagged as risky users. Risk profiles from Azure Information Protection can also be used as conditions when creating conditional access policies.

Azure Information Protection correlates user sign-in information with Microsoft's internal and external threat intelligence sources. Some of the risks are evaluated in real time. If a user attempts to **log in from an anonymous IP addresses** (such as from a Tor network exit node), they are immediately flagged for sign-in risk. Another real-time risk is labeled "**unfamiliar sign-in properties**," and it compares properties such as IP address and physical location to the user's history.

Most of the sign-in risks are calculated offline. These include

- **Atypical travel**, which identifies sign-ins from physically distant locations where the user would not have had time to travel across these locations during the time elapsed between the sign-ins.
- **Malware linked IP address**, which detects sign-ins from IP addresses that are within the known infected addresses, such as bot networks.
- **Password spray**, which detects sign-in attempts using the same password against multiple users, to perform a brute-force attack while avoiding attempted user accounts to be logged out.

Some Azure services allow you to configure Azure AD as their authentication system. These services include Azure App Service, Azure Databricks, Azure Kubernetes Service, and Azure SQL database.

Note Whenever your application is integrating with Azure AD, you should monitor Azure AD sign-in logs against that application!

EXERCISE

You are acting as the security analyst investigating the impact of an ongoing nation-state attack against organizations in your region. You have learned that the adversaries have added additional credentials for service principles with existing elevated privileges[2] to gain persistent access. You are tasked to assess whether your environment is impacted. List the steps you need to take and logs you need to query to find this out.

Bonus question

What are the required privileges for your investigation?

Infrastructure Monitoring

For some Azure services, you're responsible for securing the virtual machine image. In these cases, you need to consume the logs from your infrastructure into a centrally managed location. Monitoring your Azure-based virtual machines typically requires installing a monitoring agent. There are multiple agents available natively in Azure, and the branding is evolving. There are multiple Cloud Workload Protection Platforms (CWPP) vendors, who offer other agent-based monitoring, vulnerability management, and protection solutions that you could use, too. If you choose to use a third-party agent, you should take network considerations into account in your deployment plans.

At the time this book went to press, the most comprehensive native monitoring agent from security perspective is the Log Analytics agent. The Log Analytics agent is named Microsoft Monitoring Agent or OMS agent in some documentation pages. For Windows, it is the same agent used by System Center Operations Manager. The Log Analytics agent collects Windows event logs from Windows virtual machines and Syslog from Linux virtual machines.

In addition to log collection, Azure Security Center's Azure Defender for Servers includes a vulnerability assessment scanner by Qualys. Once installed, the Qualys agent collects artifacts from the host virtual machine and sends them to the Qualys cloud service of your region for analysis. The findings of the vulnerability assessments performed by Qualys are available in Azure Security Center.

[2] https://attack.mitre.org/techniques/T1098/001/

If your virtual machine is configured as a Docker host, Azure Security Center provides you with recommendations to fix vulnerabilities in your container configurations. Azure Security Center uses Center for Internet Security (CIS) Docker Benchmark to perform these assessments.

Container images stored in the Azure Container Registry should also be scanned for vulnerabilities. Azure Defender for Container registries uses Qualys to perform the vulnerability scanning. Images are scanned whenever they are pushed to or pulled from the Azure Container Registry.

Application Monitoring

In this section, I describe how you can monitor the security of your applications and data plane.

Centralized Log Architecture

In this section, I present various options for centralized logging. The most suitable centralized logging architecture for your need varies based on the footprint you have in Azure. Your Azure usage might be split across multiple Azure clouds, Azure Active Directory tenants, and billing accounts.

Enterprise Environment Considerations

The most common scenario includes collecting logs across your organization's Azure subscriptions into a single centralized Log Analytics Workspace. Whether you are collecting all platform, infrastructure, and application logs into this centralized log store varies based on your unique requirements.

Activity log collection can be enabled centrally, if the identity used for enabling the log export has access to the target Azure subscription, as well as the centralized Log Analytics Workspace.

Any logs that require setting up a diagnostic setting required access to be reversed. The identity creating the diagnostic setting in the target subscription needs to have access to the centralized Log Analytics Workspace **at the time the diagnostic setting is created**.

Figure 3-3 illustrates the centralized log structure concept. In this example, the centralized log store (Log Analytics Workspace) is deployed to a separate subscription.

This enables for a granular access control, providing access to the centralized security team for security posture management as well as security operations monitoring purposes.

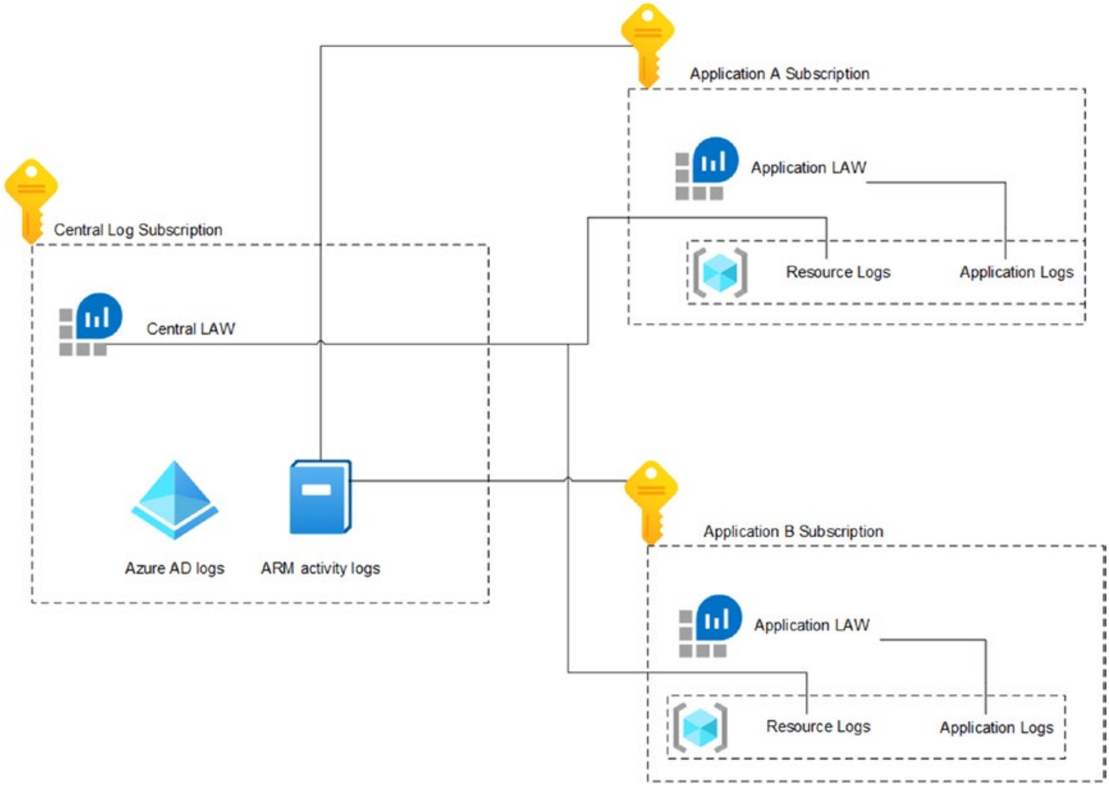


Figure 3-3. *Centralized log structure*

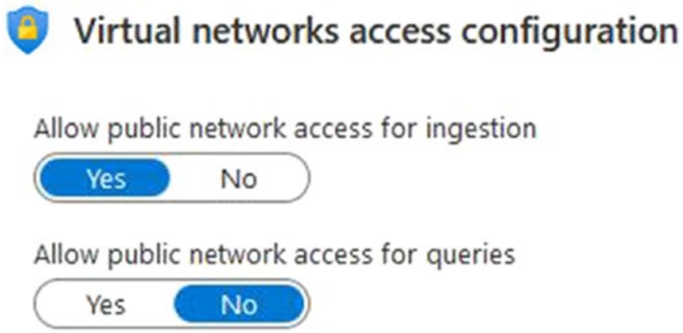
This centralized log store is then configured to ingest platform logs from sources such as Azure AD, ARM Activity Logs, and Resource Logs. Access control and log retention are set separately for the centralized log store and the application-specific log stores. While key platform and infrastructure logs are stored in the centralized log store, this approach supports more verbose logging through the usage of subscription-specific log stores. I consider at least the following Resource logs relevant for centralized logging purposes:

- SQL audit logging, such as password resets and security logs
- App Service antivirus logging

- Key Vault access logs
- Web Application Firewall logs

Securing the Centralized Log Store

To secure the centralized Log Analytics Workspace, you should control the public network access for ingestion and queries, as illustrated in Figure 3-4. You should also consider creating a diagnostic setting to audit any access to the centralized Log Analytics Workspace.



Virtual networks access configuration

Allow public network access for ingestion

[✓ Yes] No

Allow public network access for queries

Yes [✓ No]

Figure 3-4. *Network isolation settings for Log Analytics Workspace*

Depending on data stored in your centralized log store and your regulatory requirements, you might need to provide audit logs of your audit log access too. Log Analytics Workspace can emit audit logs using diagnostic settings. The specific log type is labeled *LAQueryAuditLogs*.

Note Remember to keep a balance between usefulness for a company-wide audit and usefulness for application-specific requirements!

Complex Environments

In addition to the global Azure cloud, you might be using more than one Azure sovereign cloud to meet regulatory requirements. Azure US government cloud, Azure cloud in China, and Azure cloud in Germany are the most common examples. These sovereign clouds have their own network and identity parameters. In addition to technical connectivity issues, your regulatory requirements might prevent you from exporting logs from these sovereign clouds to global Azure infrastructure.

Whether or not you are using sovereign clouds, you might be using more than one Azure Active Directory tenant. This could be due to your business requirements, such as past or upcoming acquisitions, or for any number of reasons. If that is the case, you need to build a solution that integrates across the trust boundary of these tenants. You will effectively consume these logs in the same way as you would consume logs from Azure to a SIEM hosted by a third party.

Security Posture Monitoring

In this section, I discuss strategic choices your organization needs to take for security posture monitoring. There are several multi-cloud cloud security posture management (CSPM) vendors[3] in the market who offer visibility to a broad set of public cloud environments. Regretfully, most of these are focusing on only a small set of services that are available across all your cloud platforms. While this approach dust gives you a single plane of glass, you are only gaining visibility to the least common denominator across the cloud platforms.

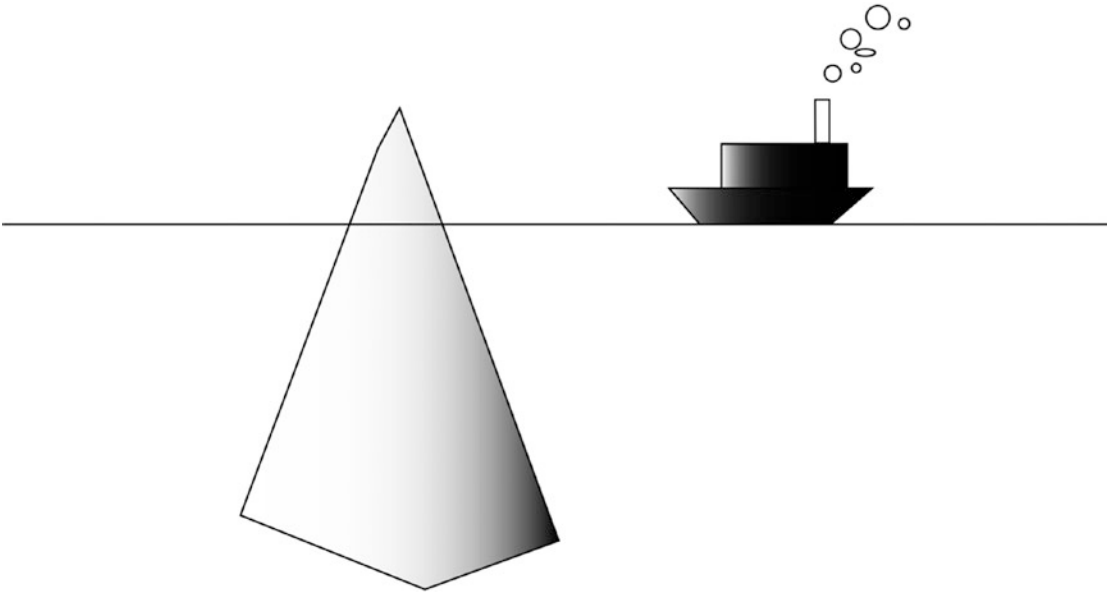


Figure 3-5. *Most cloud security posture monitoring tools see only a subset of you your cloud environment*

[3] Including but not limited to CheckPoint CloudGuard Dome 9, Palo Alto Prisma Cloud, and Trend Micro Cloud Conformity.

If your cloud security posture management tool reports 100% compliance against your standards, you need to set this number in context. Without knowing how many of your cloud resources are not covered within the reports, you are effectively only seeing the tip of that proverbial iceberg, and you do not have a complete picture of the potential risks, as illustrated in Figure 3-5.

When choosing a cloud security posture monitoring approach, you need to align with your cloud strategy. Specifically, you need to understand the range of existing upcoming cloud services in use within your organization. If you are mostly consuming capacity and services that are similar across cloud vendors, multi-cloud cloud security posture management tools could give you a reasonable coverage across your environment. If you are using platform-as-a-service services, you might need to build additional security monitoring capabilities for the individual clouds you are working with anyway.

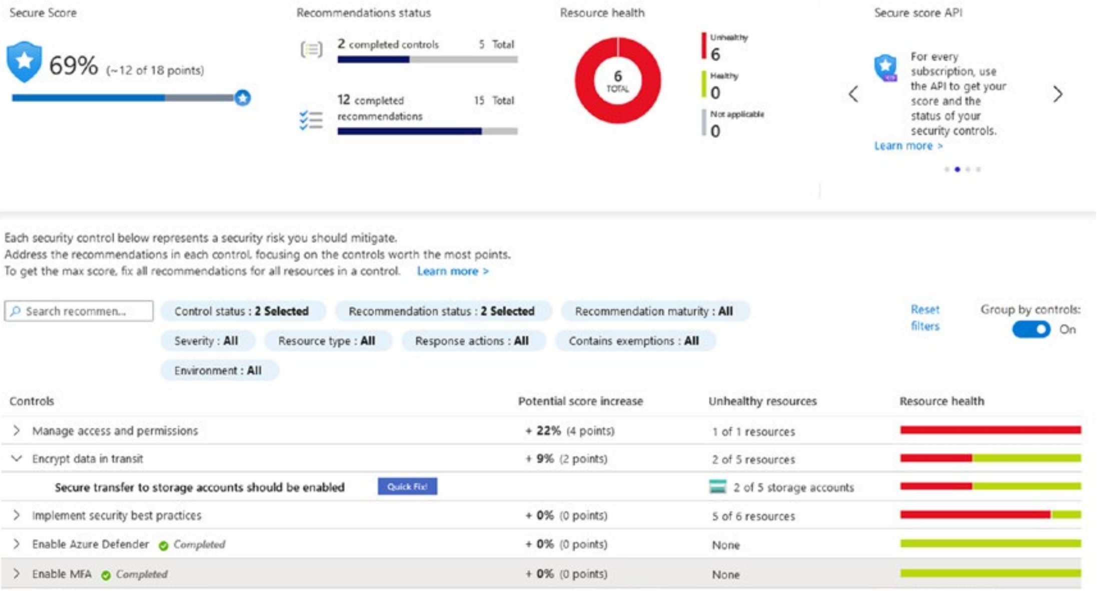
Security Posture Monitoring Using Azure Security Center

Azure Security Center is an Azure-native tool that helps you monitor your Azure security posture. Azure Security Center comes built into Azure subscriptions, providing a compliance review against Microsoft set of security best practices, the Azure Security Benchmark (ASB). Azure Security Center can be used by the central security team and by individual application development teams. Azure Security Center set of recommendations conform to Azure role-based access control. Therefore, developers and stakeholders see the security posture information of the resources that they have access to normally.

The central security teams can either use the same Azure-native interface as the application teams do or export the Security Center information to a centralized location. Azure Security Center data can be consumed into custom reports within the Security Center user interface using Azure monitor workbooks. Instead of the Security Center, you can use existing reporting tools at your disposal. Continuous Cloud Optimization Power BI Dashboards project[4] is an example of such reporting outside the native Security Center interface.

[4] https://github.com/Azure/ccodashboard

Azure Security Center helps application teams secure their environment according to Microsoft best practices. Azure Security Center displays security recommendations based on the used Azure resources. Each recommendation provides steps to implement remediating security controls. The proposed security controls are assigned severities (low, medium, high) and are using the Microsoft secure score. Implementing a security control with a significant impact on your security posture improves your secure score more than a smaller improvement. Secure score is a way to gamify this federated security posture management. From an operational perspective, it is relatively straightforward to communicate how secure your environment is using the secure score.



Secure Score

69% (~12 of 18 points)

Recommendations status

2 completed controls 5 Total

12 completed recommendations 15 Total

Resource health

6 TOTAL

Unhealthy 6

Healthy 0

Not applicable 0

Secure score API

For every subscription, use the API to get your score and the status of your security controls.

Learn more >

Each security control below represents a security risk you should mitigate.
Address the recommendations in each control, focusing on the controls worth the most points.
To get the max score, fix all recommendations for all resources in a control. Learn more >

Search recommen...

Control status : 2 Selected

Recommendation status : 2 Selected

Recommendation maturity : All

Severity : All

Resource type : All

Response actions : All

Contains exemptions : All

Environment : All

Reset filters

Group by controls: On

Controls	Potential score increase	Unhealthy resources	Resource health
Manage access and permissions	+ 22% (4 points)	1 of 1 resources	
Encrypt data in transit	+ 9% (2 points)	2 of 5 resources	
Secure transfer to storage accounts should be enabled Quick Fix!		2 of 5 storage accounts	
Implement security best practices	+ 0% (0 points)	5 of 6 resources	
Enable Azure Defender ✓ Completed	+ 0% (0 points)	None	
Enable MFA ✓ Completed	+ 0% (0 points)	None	

Figure 3-6. *Screenshot of Azure Security Center recommendations in the Azure Security Center blade of the Azure Portal*

Since 2021, the standard Security Center policy initiative is the Azure Security Benchmark. Azure Security Benchmark is based on Microsoft best practices and mapped into industry standard security controls, such as the ones from Center for Internet Security (CIS) and the National Institute of Standards and Technology (NIST).

Security Policy Initiatives

In addition to the standard policy initiative, you can use Azure Security Center with your own compliance policy initiatives. These policy initiatives are assigned the same way as Azure policies, that is, to the management group, subscription, and other scopes within Azure Resource Manager. Once assigned, conformance against your security policies can be viewed within the native Azure Security Center, specifically in the compliance blade.

You can use your custom security policy the same way you will do the default Security Center policy initiative. If you want to operationalize the custom security policy in a similar manner as the standard Security Center policy initiative, you can provide your own remediation steps and severity information.

Microsoft manages a list of built-in security policy initiatives to help you meet industry or regional regulatory compliance. As of the writing of this book, these policy initiatives include, but are not limited to, the following standards:[5]

- HIPAA HITRUST
- UK NHS
- SWIFT CSP CSCF
- CMMC

Security Policy Architecture at Scale

Implementing and enforcing security policies across your Azure environment is a complex topic. On one hand, you want to ensure a consistent security posture and visibility across all your applications. On the other hand, you want to provide room for exception management and multiple levels of granularity. Security policy management therefore ends up being a technical exercise of privileged access and Azure policy assignment(s). Microsoft's best practices are described in the Azure Security Center Enterprise Onboarding Guide.[6]

Figure 3-7 illustrates the cumulative effects of these policy initiative assignments. In this figure, the Azure Security Benchmark is assigned in the root management group. This ensures that users with previous access to management groups and subscriptions

[5] https://docs.microsoft.com/en-us/azure/governance/policy/samples/#regulatory-compliance

[6] https://github.com/Azure/Azure-Security-Center/blob/onboarding/Onboarding

are not able to modify the integrity of the baseline security policy initiative. Next, applicable policies for standards and compliance are assigned in the management group layers according to your governance model. These built-in policy initiatives can include industry-specific standards or regional compliance requirements. And finally, application teams should be empowered to assign custom policies in their respective environments.

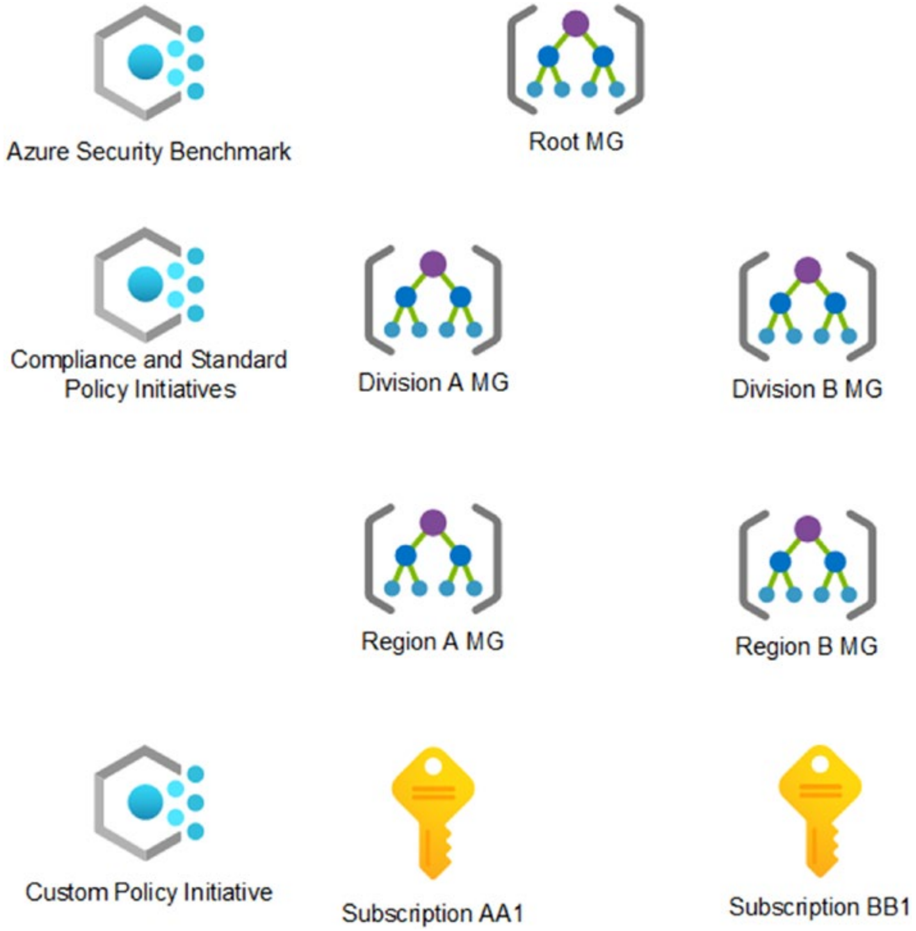


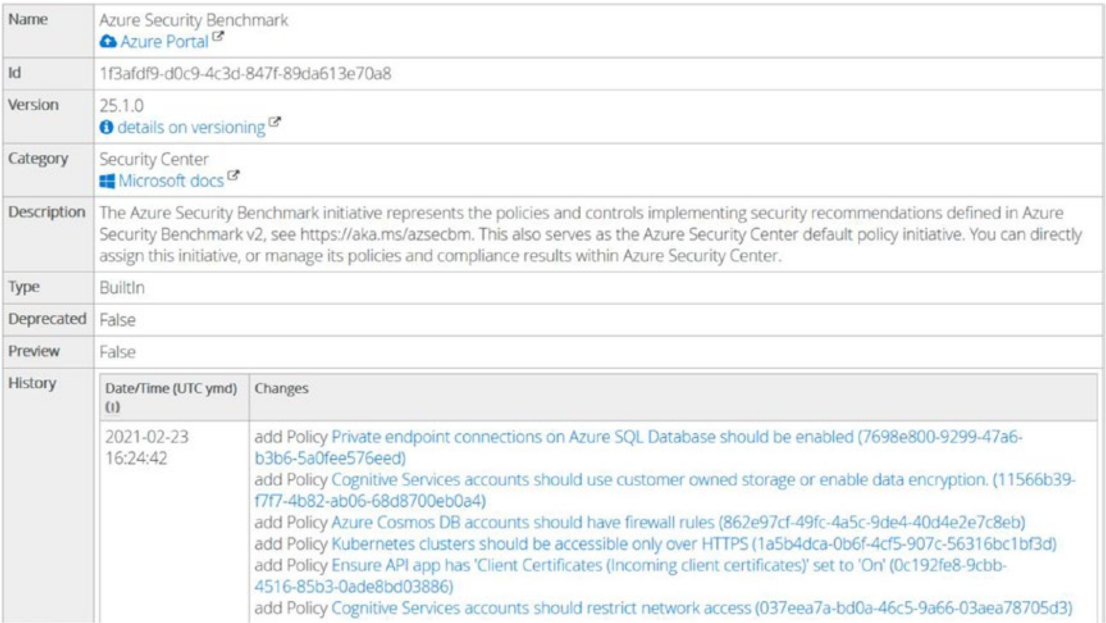
Figure 3-7. *An example of nested security policy initiative structure*

Change Tracking of Security Policies

Due to the evergreen nature of Azure, and specifically Azure Security Center, the standard security policies assigned to your Azure environment will change over time. You might have regulatory requirements requiring you to keep an audit trail of which policies have been in effect and when.

Microsoft keeps track of policy deprecations and other changes in the Azure Security Center release notes documentation.[7] You can use this information as a baseline to meet your change tracking requirements. If you are using a custom policy initiative, you are in control of the changes. In that case, your change tracking information will come from your version control and deployment history of the policy initiative.

A more technical solution for policy initiative change tracking is the AzAdvertiser project.[8] This unofficial tool periodically exports built-in Azure policies and policy initiatives from a live Azure subscription. Each policy or initiative is exported in full, so changes can be tracked down to the policy definition level. Beyond manual change tracking illustrated in Figure 3-8, AzAdvertiser provides a RSS feed of any changes.



<table>
<tr><td>Name</td><td>Azure Security Benchmark
Azure Portal</td></tr>
<tr><td>Id</td><td>1f3afdf9-d0c9-4c3d-847f-89da613e70a8</td></tr>
<tr><td>Version</td><td>25.1.0
details on versioning</td></tr>
<tr><td>Category</td><td>Security Center
Microsoft docs</td></tr>
<tr><td>Description</td><td>The Azure Security Benchmark initiative represents the policies and controls implementing security recommendations defined in Azure Security Benchmark v2, see https://aka.ms/azsecbm. This also serves as the Azure Security Center default policy initiative. You can directly assign this initiative, or manage its policies and compliance results within Azure Security Center.</td></tr>
<tr><td>Type</td><td>BuiltIn</td></tr>
<tr><td>Deprecated</td><td>False</td></tr>
<tr><td>Preview</td><td>False</td></tr>
<tr><td>History</td><td><table>
<tr><th>Date/Time (UTC ymd) (i)</th><th>Changes</th></tr>
<tr><td>2021-02-23 16:24:42</td><td>add Policy Private endpoint connections on Azure SQL Database should be enabled (7698e800-9299-47a6-b3b6-5a0fee576eed)
add Policy Cognitive Services accounts should use customer owned storage or enable data encryption. (11566b39-f7f7-4b82-ab06-68d8700eb0a4)
add Policy Azure Cosmos DB accounts should have firewall rules (862e97cf-49fc-4a5c-9de4-40d4e2e7c8eb)
add Policy Kubernetes clusters should be accessible only over HTTPS (1a5b4dca-0b6f-4cf5-907c-56316bc1bf3d)
add Policy Ensure API app has 'Client Certificates (Incoming client certificates)' set to 'On' (0c192fe8-9cbb-4516-85b3-0ade8bd03886)
add Policy Cognitive Services accounts should restrict network access (037eea7a-bd0a-46c5-9a66-03aea78705d3)</td></tr>
</table></td></tr>
</table>

Figure 3-8. *Changelog of the Azure Security Benchmark policy initiative, as displayed in Azadvertizer.net*

These sources give you a baseline of changes in the policies. But what about the effective scope of these policies? To answer that question, you would need to complement the policy information with any exceptions you might have. Azure Security

[7] https://docs.microsoft.com/en-us/azure/security-center/release-notes

[8] www.azadvertizer.net/

Center supports exempting resources or even recommendation categories. Exporting this exemption state can be done using Azure resource graph.[9]

Azure Tenant Security Scan

In addition to Security Center recommendations and policies, Microsoft built the Secure DevOps kit for Azure (AzSK) to automate security scanning across the application life cycle. The Secure DevOps kit for Azure's security tested Azure PowerShell to check against misconfigurations. This approach was beneficial for automation purposes and allowed you to control the security of your resources in staging environments, too. In 2021, Microsoft announced the deprecation of Secure DevOps kit for Azure, citing advancements in Azure-native security capabilities such as Azure policy and Security center.

The replacing service is Azure Tenant Security Solution (AzTS). Azure Tenant Security Solution is built by the same team in Microsoft who built the Secure DevOps kit for Azure. It provides comparative security scan coverage for continuous assurance but lacks standalone mode and Azure DevOps pipeline integration. As the AzTS uses Azure Security Center, Azure policy, and Azure resource graph, it is scalable to tens of thousands of Azure subscriptions but, as of 2021, is still lacking some features that are available in the PowerShell-based approach of Secure DevOps kit for Azure.

Summary

In this chapter, you learned that security monitoring at scale in Microsoft Azure is a complex topic. You need to balance between different requirements, level of granularity, storage types, and the signal to noise ratio. To succeed in meeting your business requirements, you need to clearly differentiate where centralized logging and monitoring responsibilities and application team responsibilities begin.

When implemented successfully as part of a comprehensive security architecture, both security posture monitoring and operational security monitoring can provide improved results when compared to on-premises monitoring.

[9] https://docs.microsoft.com/en-us/azure/security-center/exempt-resource#find-recommendations-with-exemptions-using-azure-resource-graph

CHAPTER 4

Network Security

In this chapter, I present the various network controls available to you in Azure environments, their function, and how to apply them. I discuss the best practices on implementing these controls for infrastructure-as-a-service and platform-as-a-service workloads.

Azure Virtual Networks

In this section, I introduce you to the key concept of Azure networking – the Azure virtual network.

Microsoft Global Network

To provide cloud services, Microsoft operates a physical network of over 160 **datacenters** globally. The datacenters are grouped into **regions** which operate within interconnected regional networks. These regional networks are connected to[1] through Microsoft global-wide area network (WAN) over private fiber-optic cables.

[1] https://docs.microsoft.com/en-us/azure/expressroute/expressroute-locations-providers#expressroute-locations

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_4

Customer traffic from the Internet to Microsoft's global WAN enters or exists through **point of presence** locations, with even broader physical footprint than of datacenter regions. Alternatively, enterprise customers can create ExpressRoute connections to connect to the Microsoft global WAN without leaving their dedicated network. ExpressRoute locations are co-location facilities where Microsoft Enterprise edge (MSEE) devices are located.

When connecting through the Internet, network traffic is protected with Azure DDoS Protection. With Azure DDoS Protection, Microsoft applies traffic monitoring and real-time mitigation against network-layer attacks, such as network flushing, before the traffic is routed to customer instances.

IP Addresses in Azure

IP addresses under your control are considered resources in Azure. As such, they are deployed into a resource group in a subscription. IP addresses are subject to the same role-based access control and Azure policies like any other resources. Figure 4-1 illustrates the different resource types related to IP addresses and how the access control and life cycle of IP addresses are independent.

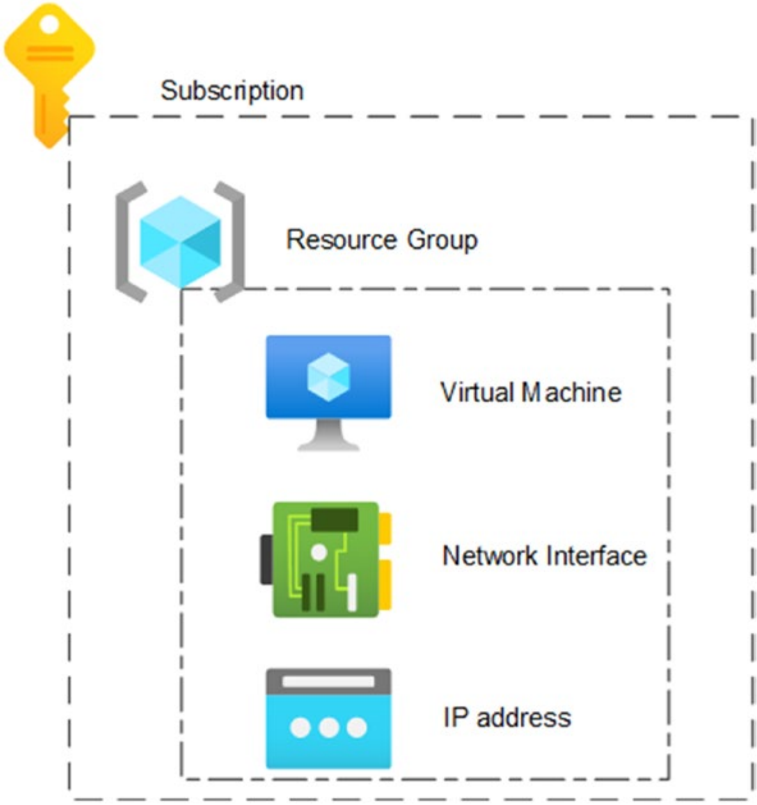


Figure 4-1. *IP address resource*

While managed as independent resources, IP addresses are often associated to other resource types, such as a

- Virtual machine network interface
- Internet-facing load balancer
- VPN gateway
- Application Gateway

Azure IP addresses support either dynamic or static address allocation. You can use both IPv6 and IPv4 addresses in Azure. The actual IP address is tied to the life cycle of the IP address resource. If you want to reallocate an IP address, you need to disconnect its association to another resource, before deleting it. Once that IP address is no longer associated with the resource, you are free to associate it with another resource.

Note When you delete an Azure IP address resource, the IP address is released to be used by other Azure customers!

Azure Virtual Network

A key component in Azure networking is the Azure virtual network. Virtual networks bring a level of control similar to on-premises networks into that multitenant cloud world. Any resource deployed into a virtual network are isolated from other cloud users and even your own virtual networks. You can control inbound and outbound traffic, IP addresses, and routing for Azure virtual networks. Figure 4-2 illustrates the virtual network isolation.

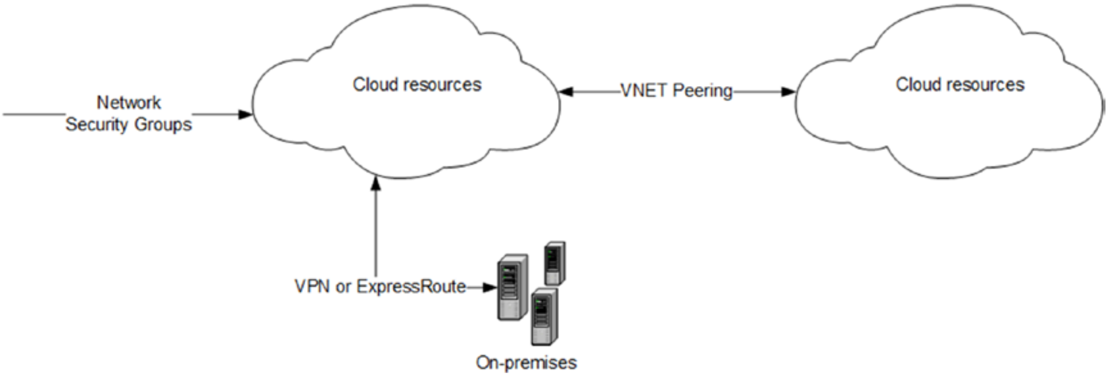


Figure 4-2. *Virtual networks isolate network connectivity within the cloud, as well as inbound or outbound to/from on-premises or public networks*

To control network traffic within Azure, Azure virtual networks can communicate with other Azure virtual networks by using virtual network peering. For example, an application deployed to multiple Azure regions could leverage virtual network peering to extend the logical virtual network across regions.

Azure virtual networks are used to communicate with on-premises resources, too. You can connect your virtual networks to other networks outside of Azure using the Azure virtual private network (VPN) or Azure ExpressRoute.

Like IP addresses, virtual networks exist within the Azure Resource Manager as resources. Therefore, access control and life cycle considerations should be kept in mind when planning network topologies. Virtual network resources have several properties to be configured, such as IP address spaces, gateways, or user-defined routes. Crucially, subnets are not considered properties of virtual network resources, but rather sub-resources. Role-based access control can be granted for in the sub-resource scope, too. The logical hierarchy is illustrated in Figure 4-3.

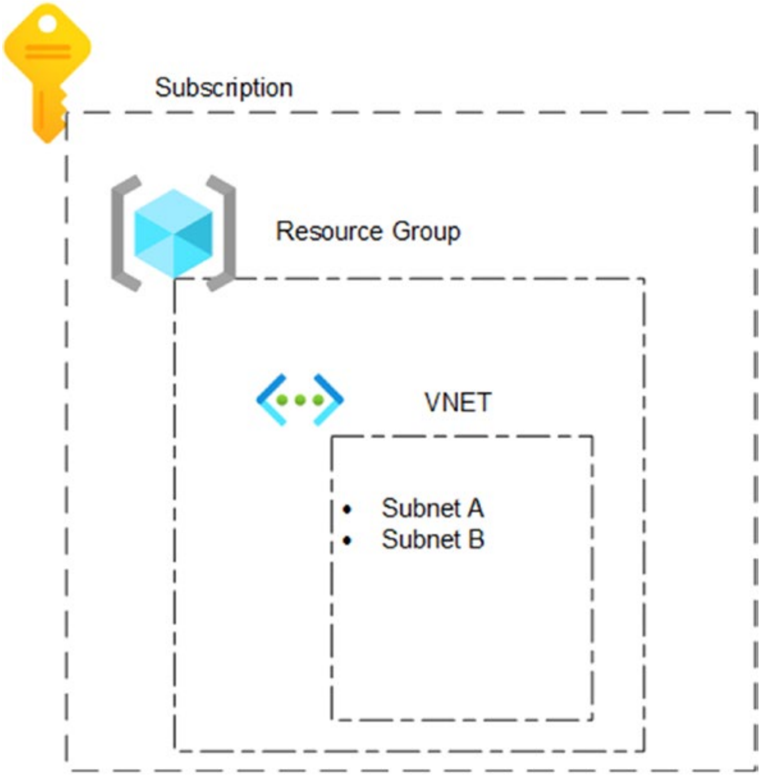


Figure 4-3. *Logical hierarchy of virtual network resource and subnet sub-resource in Azure Resource Manager*

Network Controls for Infrastructure as a Service

Network Security Groups

The native way to control network traffic to and from resources in your Azure virtual network is the network security group. Like an access control list, a network security group consists of a list of security rules that allow or deny traffic. If you are familiar with VLAN segmentation, virtual network subnets and network security groups offer a comparable solution in Azure. For each network security group security rule, you can specify

- **Traffic source or destination**: IP address range, service tag, or application security group
- **Protocol**: TCP, UDP, ICMP, ESP, AH, or any
- **Port range**
- **Rule priority** between 100 and 4096 (lower numbers are higher priority)
- **Rule action** (either allow or deny)

Service tags are Microsoft-managed lists of IP addresses[2] for multitenant Azure services. You can use service tags to create network security group rules that deny or allow traffic to the public endpoints of Azure PaaS services, while still preventing access to or from the Internet. For example, the service tag **AppService.WestEurope** can be used to allow outbound traffic to Azure App Service. Microsoft updates service tags and IP address ranges periodically. Service tag information is available for download as JSON files, as well as programmatically through the Service Tag Discovery API.

Application security groups are logical groupings of security rules for your own applications. For example, if all your front-end virtual machines need connectivity blocked to the Internet but allowed to your back-end virtual machines, application security groups can be used.

Azure evaluates incoming or outgoing traffic against network security groups. Traffic flows are interrupted when connections are stopped, and no traffic is flowing in either direction.

Note Existing connections are not interrupted when you remove a security rule that enabled the flow. NSG flows are evaluated for new connections.[3]

[2] www.microsoft.com/en-us/download/details.aspx?id=56519

[3] https://docs.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview#security-rules

Network security groups can be assigned in the subnet or the network interface scope. For inbound traffic, Azure processes NSG rules configured in the subnet scope first and then the rules in a network security group associated to the network interface. For outbound traffic, the order of processing is reversed. Figure 4-4 illustrates the combination of different network security group destination and source types and cumulative network security group rules.

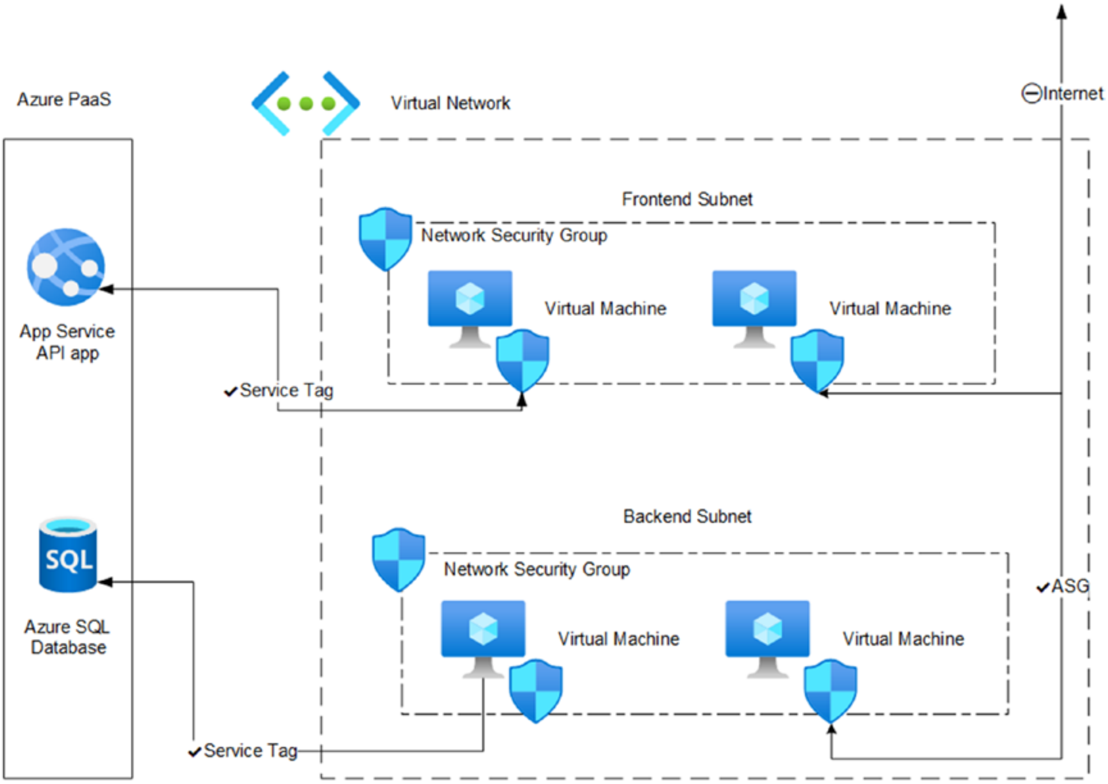


Figure 4-4. *Network security groups, service tags, and application security groups*

Note Network security groups cannot be assigned at a virtual network scope, but only on the subnet scope!

Securing Administrative Access to Virtual Machines

One of the key attack vectors for workloads running in public cloud is misconfigurations. Exposed management ports are often scanned by adversaries as part of the initial discovery tactics.[4]

As the IP address ranges of Azure datacenters are relatively easy to obtain, it is safe to assume that if you leave a management port exposed to the Internet, it will be picked up by adversaries in the matter of minutes. Once discovered, your virtual machines are susceptible to password spray attacks. Azure DDoS Protection basic or other protocol-level network protections will not help you to prevent this. To protect yourself from this and still allow legitimate access, you need to introduce additional network controls.

As part of Azure Defender for Servers, Azure Security Center just-in-time (JIT) access to virtual machines reduces the exposure of your virtual machine management ports by managing the network security group or Azure Firewall rules on demand. Once configured, Azure security center JIT enforces that inbound traffic is denied. To open management port access from their network location, users need to perform a just-in-time activation in Azure Portal. This verifies user role-based access control, audits the access, and opens the management ports for a predetermined time. Specifically, the requesting user needs to have access to the `Microsoft.Security/locations/jitNetworkAccessPolicies/initiate/action` resource provider action in the scope of the virtual machine resource group.

As discussed earlier, network security groups interrupt traffic flows when connections are stopped, so just-in-time access to virtual machines does not prevent your privileged users from keeping their connections open even after the just-in-time access time has ended.

An alternative to Azure Security Center just-in-time virtual machine access would be to create a point to site VPN connection to the virtual network of the virtual machine. Azure point-to-site VPN supports most client operating systems. The point-to-site VPN access can be authenticated using Azure AD (Windows 10), RADIUS, or certificate authentication. It supports the following protocols:

- OpenVPN®
- Secure Socket Tunneling Protocol (SSTP)
- IKEv2 VPN

[4] https://attack.mitre.org/techniques/T1046/

As an alternative to client-based point-to-site VPN, you can use Azure Bastion. Azure Bastion provides HTML5-based web client that is streamed to your administrative users through the Azure Portal. Behind the scenes, Azure Bastion is a Microsoft-managed and hardened service that is deployed into a separate subnet in your virtual network. Azure Bastion then automatically manages network security group rules to allow connectivity between the Azure Bastion subnet and your virtual machines.

Azure Security Center just-in-time access is the easiest of these to set up. It reduces the exposure time and source addresses of administrative access. However, it does not remove them altogether. Azure point-to-site VPN adds an additional layer of protection by keeping that management ports exposed and requiring user authentication for the VPN gateway connection. Azure Bastion removes any need for exposed IP addresses. Based on your user experience and performance requirements, however, Azure Bastion might not meet your needs. As with all cloud security, securing management port access comes down to balancing your need for agility and control.

Securing Outbound Access from Virtual Machines

The federated nature of Azure networking adds complexity in monitoring and securing outbound network traffic of your virtual machines. The default NSG rules allow outbound Internet connectivity on any port. For virtual machines that are not connected to your centralized virtual networks, you might not have a straightforward way to determine malicious traffic from legitimate traffic.

Azure Security Center and Azure Firewall allow you to filter out and alert on outbound communication with malicious IP addresses. Azure Security Center's adaptive network hardening also provides recommendations for narrowing down any NSG rules, based on your actual traffic patterns.

For some Azure offer types, outbound connectivity is limited even further. Specifically, outbound connectivity over port 25 (SMTP) is disabled for most offers. When you are using one of the paid subscription types (pay as you go, CSP, and EA), you can request opening of this port through support.

Network Controls for Platform as a Service

In this section, I discuss the network controls available in Azure application and storage PaaS services.

Application PaaS Networking

Most Azure application platform-as-a-service resources are multitenant by nature. From the networking perspective, this means that the services are by default reachable by anyone through the public endpoints. Furthermore, controlling outbound network traffic can also be quite limited. Having the option to deploy your platform-as-a-service resources inside your virtual network often requires committing to a higher pricing tier.

Controlling Inbound Traffic to App Service

Azure App Service is one of the most widely used application platform-as-a-service resource types in Azure. In the multitenant App Service, inbound traffic can be restricted using Access Restrictions,[5] a feature that provides allow and deny rules for inbound access. Access Restriction rules are set separately for management operations and content. Access Restriction rules are evaluated separately from your App Service instance, by the managed App Service front end, so any denied traffic never reaches your application code. Access Restriction rules can be set to limit incoming traffic based on

- IP address ranges
- Virtual network service endpoints
- Service tags
- HTTP headers

Controlling Outbound Traffic from App Service

Outbound traffic from Azure App Service can be secured through virtual network integration or the App Service Hybrid Connections feature.

[5] https://docs.microsoft.com/en-us/azure/app-service/app-service-ip-restrictions

App Service virtual network integration allows **outbound** connectivity from your App Service to your virtual network. Your application can then reach resources in the virtual network. Virtual network integration does not grant inbound private connection from your virtual network. App Service can be configured to route all its outbound traffic to the virtual network.

Figure 4-5 illustrates the Azure App Service networking controls. First, Access Restrictions are used to deny inbound connectivity from the Internet and allow inbound connectivity from the specific IP address range. Second, virtual network integration is configured for controlling outbound traffic from the App Service application. Third, integration subnet is configured to allow traffic from the App Service endpoint, and the back-end subnet's network security group is configured to allow traffic from the integration subnet. And finally, all outbound traffic is enforced to go through the virtual network using the `WEBSITE_VNET_ROUTE_ALL` setting in App Service.

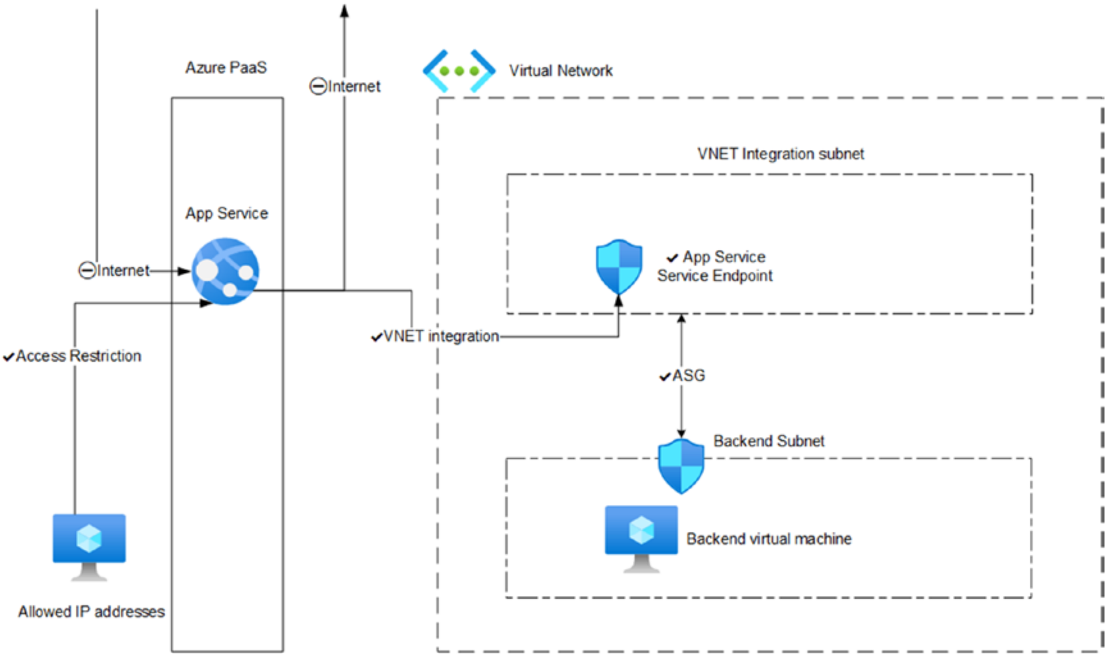


Figure 4-5. *App Service networking controls*

Cross-Network Connectivity

App Service Hybrid Connections enables access to your on-premises resources, without changes to on-premises inbound firewall rules. Instead, Hybrid Connections depends on a relay agent to be installed on your on-premises network and outbound TCP connectivity to Azure over port 443. Once configured, any DNS requests in your App Service application that match your Hybrid Connection endpoint will be redirected through the Hybrid Connection relay.

Finally, the single tenant tier of Azure App Service, App Service Environment, is in fact placed in a virtual network that you control yourself. This means that you can use the same virtual networking connectivity options that you can for any other virtual network resources, such as virtual machines.

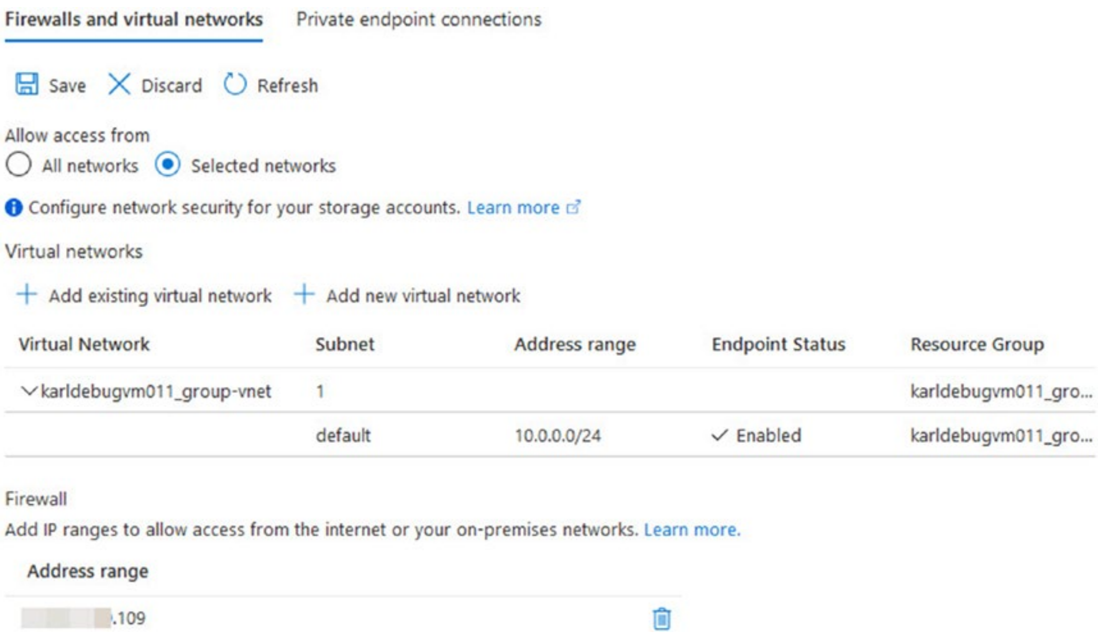
Network Controls for Data Platform as a Service

You can store your data on several platform-as-a-service data stores in Azure. Just like application platform as a service, data platform-as-a-service resources are often hosted in a multitenant environment. Gaining access to full virtual network injection capabilities is either not possible at all or available only in the highest pricing tiers.

By default, most multitenant Azure data services are available through the public endpoints in an unrestricted manner. It is our responsibility as the cloud consumer to configure traffic restrictions. When combined with limitations on the other parts of our workloads, such as legitimate end users or continuous deployment pipelines, this can become quite complex.

Storage Account Firewall

Most Azure data services offer a variation of a **firewall** functionality. For example, the firewall of Azure storage account lets you control inbound traffic by creating access rules that target IP address ranges or virtual networks. This is done by changing the **default network access rule** to deny all incoming traffic by default. Once access is restricted, you can configure storage account to allow access only from certain public IP address ranges. Additionally, private addresses can be granted by allowing virtual network access, provided that the target virtual network allows outbound traffic from the virtual network to storage account service endpoint. Figure 4-6 illustrates a storage account that denies default public access and allows access from a specific subnet and a single public IP address.



Firewalls and virtual networks | Private endpoint connections

Save | Discard | Refresh

Allow access from

○ All networks ● Selected networks

Configure network security for your storage accounts. Learn more

Virtual networks

+ Add existing virtual network + Add new virtual network

Virtual Network	Subnet	Address range	Endpoint Status	Resource Group
karldebugvm011_group-vnet	1			karldebugvm011_gro...
	default	10.0.0.0/24	✓ Enabled	karldebugvm011_gro...

Firewall

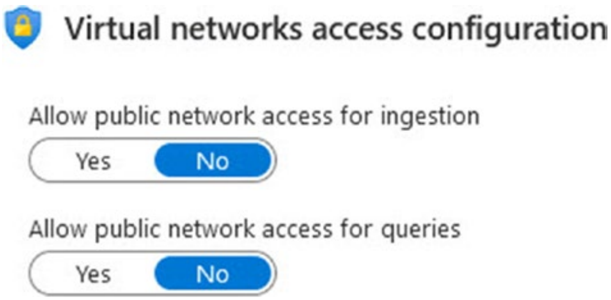
Add IP ranges to allow access from the internet or your on-premises networks. Learn more.

Address range
[redacted].109

Figure 4-6. *Storage account firewall configuration*

Azure Monitor Network Isolation

The firewall functionality of Azure Monitor service, including Log Analytics Workspace and Application Insights, allows to configure separate network rules for data ingress and egress. Figure 4-7 illustrates this functionality. Like for storage account, you need to allow egress traffic from your network to Azure monitor. To do this, you can use the `AzureMonitor` service tag.



Virtual networks access configuration

Allow public network access for ingestion

Yes [✓ No]

Allow public network access for queries

Yes [✓ No]

Figure 4-7. *Azure monitor network isolation controls*

Private Endpoints

As an alternative to IP address or virtual network access rules, traffic between Azure platform-as-a-service resources and your virtual network cannot be secured using private endpoints.

Private endpoints expose your instance of a multitenant Azure service as private IP addresses within your virtual networks. Using private endpoints protects from data exfiltration and gives you more control of both inbound and outbound traffic. Instead of service tags or service endpoints, your access rules can be set in a more granular level.

Private endpoint configuration is more complex than just enabling access rules or creating service endpoints. As all traffic between your private endpoint resource and your network is within your control, you can no longer rely on Microsoft to resolve your resource names. Creating a private endpoint connection requires you to also configure private DNS zones. For some services, such as Azure storage, you will need to configure the DNS zones for each storage endpoint.

While adding an additional layer of control, private endpoints also add an additional price component. As the traffic is being transferred across your dedicated connection, there is a data transfer fee for both inbound and outbound data.

Azure Firewalls

One of your key network control decisions is about whether to use Azure-native controls or third-party controls. Azure-native controls provide tighter integration and fast time to market. If you want to build more complex networks or leverage existing licenses or skills come up, you can bring in network virtual appliances from existing vendors through the Azure marketplace. The best solution depends on your cloud strategy. If you plan to take advantage of modern platform-as-a-service components, the benefits you gain from third-party network virtual appliances are more limited. However, if your cloud strategy involves a large footprint on infrastructure as a service, bringing in advanced network virtual appliances can be beneficial.

Azure Firewall is a stateful, Layer 3–7 firewall that controls virtual network access. It is a separate service from the stateless network security group rule sets and provides additional capabilities, such as

- Application FQDN filtering rules
- Network traffic filtering rules

- Threat intelligence
- Outbound SNAT support
- Inbound DNAT support
- Forced tunneling
- TLS inspection
- Intrusion detection
- Intrusion prevention

Azure Web Application Firewall

Azure Web Application Firewall is a managed, OSI Layer 7 firewall. Azure WAF is based on OWASP Core Rule Set,[6] updated and managed by Microsoft, protecting your web applications against

- SQL-injection protection
- Cross-site scripting protection
- Common Apache and IIS misconfigurations

Azure Web Application Firewall functionality can be deployed to Azure Application Gateway or Azure Front Door. The firewall rules can be managed centrally using Web Application Firewall Policies. Azure Front Door is deployed on Azure network edge locations, inspecting incoming requests before they even reach Microsoft global network.

Network Monitoring

Azure networks provide logs for both our security information and event management (SIEM) and forensic investigation needs. As with other services, the logging needs to be consciously turned on.

[6] https://owasp.org/www-project-modsecurity-core-rule-set/

Security information and event management should consume logs from NSG diagnostic logs and Azure Firewalls. NSG diagnostic logs contain element information about which security rules are applied to which virtual machines, based on Mac addresses. In addition, Azure activity logs contain logs about configuration changes to public IP instances or NSGs.

Azure Firewall provides log event for each new connection (either accepted or denied). These logs contain evaluated application and network rules. Each log event stores information about the connection source, protocol, ports, and the result of the rule evaluation. If threat intelligence mode is configured, Azure Firewall also logs that read intelligence alerts.

Azure Web Application Firewall (WAF) Access logs provide further application-level information. WAF Access logs contain

- httpMethod
- requestUri
- RequestQuery
- sslProtocol and sslCipher
- UserAgent
- originalHost

Furthermore, Azure Web Application Firewall's Firewall logs contain information about evaluated firewall rule sets.

Logs Supporting Forensic Investigation

Azure network watcher is a service that provides network security group flow log information and point-in-time packet capture functionality.

All virtual network traffic is true network security group. Whenever Azure virtual network traffic passes through a network security group rule set, flow logs can be created. NSG rules are evaluated at OSI Layer 4. The NSG flow logs include the following information:

- Source IP, port, and protocol
- Destination IP, port, and protocol
- Traffic direction
- Traffic decision, indicating either denied or allowed traffic

You should enable NSG flow log collection and export the logs to your intrusion detection systems.

For investigation purposes, Azure network watcher also supports packet capture. This is relatively intrusive agent-based approach, though. Network watcher packet capture is done within a virtual machine extension `AzureNetworkWatcherExtension`. The packet capture can be started manually or based on virtual machine alerts.

Alternative Network Monitoring

As part of Azure security center, Threat Protection for Azure network layer can alert on anomalous activities. These network alerts are raised based on traffic analytics from packet headers and Microsoft Threat Intelligence databases of malicious traffic addresses. Threat Protection for Azure network-layer alerts includes

- Network communication with a malicious machine detected
- Possible incoming SQL brute-force attempts detected
- Possible outgoing port scanning activity detected
- Suspicious incoming RDP/SSH network activity from multiple sources

For raw traffic analytics, you need to use third-party network virtual appliance (NVA) solutions hosted in virtual machines within your virtual networks. Azure marketplace offers third-party solutions for both network packet brokering as well as security analytics and network performance management.[7]

Cloud Adoption Framework

The Cloud Adoption Framework is a series of Microsoft best practice articles and artifacts. As part of the Cloud Adoption Framework, several network architectures are discussed, including traditional hub and spoke and modern networking. The Cloud Adoption Framework's network topology and connectivity section provides guidance for

[7] https://docs.microsoft.com/en-us/azure/virtual-network/virtual-network-tap-overview#network-packet-brokers

- Defining Azure network topology using Azure-native or traditional network solutions
- Connectivity to/from Azure
- Connectivity with Azure PaaS services
- Connectivity with other cloud providers
- Landing zone planning
- Network segmentation
- Traffic inspection
- Private endpoint and DNS configuration at scale

For many key scenarios, such as hub and spoke network topology, the Cloud Adoption Framework also provides infrastructure as code samples and reference architectures.

EXERCISE

You are tasked to secure an application hosted in Azure App Service. One of your requirements is the ability to provide logs of each failed attempts of access. Propose a network architecture that meets this requirement.

Summary

Azure networking at the enterprise scale is immensely complex. You need to solve cross-subscription, cross-region, cross-premises, and cross-cloud connectivity, all while balancing between user flexibility, control, and signal to noise ratio.

While it can be tempting to try to replicate on-premises networking topologies and operations, it is simply not technically feasible. You need to select alternatives that still satisfy your risk appetite and redesign your networking approach for cloud era. If you are compelled to replicate on-premises networking in the cloud world, consider that only a partial solution. If "traditional" workloads need to be segmented in the hub and spoke model, could we find new workloads that suit the cloud networking model better?

CHAPTER 5

Workload Protection – Data

In this chapter, we discuss hardening of data workloads. After reading this chapter, you will be able select and implement appropriate controls for securing data workloads as part of your organization's security policy framework.

I have selected Azure Key Vault, Azure Blob storage, and Azure SQL database as the services covered in this chapter. There are, of course, a plethora of other Azure services that support your data workloads available, such as Azure CosmosDB, Databricks, and HDInsights. Based on my experience working with enterprise organizations, the services I have selected are relevant to the broadest set of Azure users. If the data service you are looking for is not covered, you should be able to apply the core monitoring, networking, and access controls presented in this chapter for your needs.

Azure Key Vault

Azure Key Vault is an Azure-native service for storing and managing secrets, certificates, and cryptographic keys. Logically, Azure Key Vault is natively linked to the Azure Active Directory of your Azure subscription. Physically, Azure Key Vault instances are deployed to shared pools of hardware security modules (HSM) in Azure regions. For additional compliance needs, you may also use managed HSM and dedicated HSM offering of Azure Key Vault, which provides the same set of core functionality and controls, while providing you with additional isolation.

Several Azure resources integrate natively with Azure Key Vault using managed identities for Azure resources for authentication. For example, Azure App Service can consume certificate data from Azure Key Vault. Data services that support Bring Your Own Key (BYOK) encryption use Azure Key Vault to hold the cryptographic keys by default.

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_5

Access Control

Access to Azure Key Vault is authenticated using Azure Active Directory. This limits the potential access to users and security principles which are either native to your Azure Active Directory or invited as Azure Active Directory Guests. Effectively, Azure Key Vault is as secure as your Azure Active Directory.

Azure Key Vault authorization is performed using two alternative permission models: access policies (classic) or Azure data-plane role-based access control (modern). The permission models are not cross-compatible, meaning that your Azure Key Vault can only use one of the permission models. For any new Azure Key Vaults, you should use the data-plane role-based access control model.

In both permission models, management-plane access is granted using Azure role-based access control. A Contributor for an Azure Key Vault resource can switch between the permission models by editing the `properties.enableRbacAuthorization` property. Once in access policy mode, a Contributor can create, read, update, and delete access policies.

Note Contributor can switch between permission models and, by extension, manage access to the data stored in Azure Key Vault.

In the access policy permission model, the access scope is the entire Key Vault resource. An access policy is simply a combination of allowed operations, combined with the Azure Active Directory principal. There are 40 Key Vault operations configurable in access policies: 16 key permissions, 8 secret permissions, and 16 certificate permissions. There are also access policy templates available, which you can use to standardize authorization. As the scope of the access policy is the entire Key Vault resource, you cannot grant granular access two individual secrets, keys, or certificates. As such, it is not considered a best practice to share Key Vault resources across applications.

In the Azure data-plane role-based access control, you can grant access within a more granular scope. Specifically, keys, secrets, and certificates are considered sub-resources and can be used as a scope for role-based access control assignments. There are built-in roles available, which you can assign either to the Key Vault resource or any of its sub-resource scope:

- **Key Vault Administrator**: Perform all data-plane operations on a key vault and all objects in it, including certificates, keys, and secrets. Cannot manage Key Vault resources or manage role assignments.
- **Key Vault Reader**: Read metadata of Key Vaults and its certificates, keys, and secrets. Cannot read sensitive values such as secret contents or key material.
- **Key Vault Contributor**: Management-plane operations only. It does not allow access to keys, secrets, or certificates.
- **Key Vault Secrets Officer**: Perform any action on the secrets of a key vault, except manage permissions.
- **Key Vault Secrets User**: Read secret contents.
- **Key Vault Certificates Officer**: Perform any action on the certificates of a key vault, except manage permissions.
- **Key Vault Crypto Officer**: Perform any action on the keys of a key vault, except managing permissions.
- **Key Vault Crypto User**: Perform cryptographic operations using keys.

Network

Network access to the data plane of Azure Key Vault is by default unrestricted: authorized applications and users can access the Key Vault resource from any network location allowed within your Azure Active Directory Conditional Access. To set up network controls, you need to enable the Key Vault Firewall.

The least restrictive mode for the Key Vault Firewall is the **Allow Trusted Microsoft Services to bypass this firewall** option. This denies public network access and allows it from most multitenant Microsoft services. The list of trusted services includes, but is not limited to, Azure App Service, Azure SQL database, and Azure storage. To verify whether your target service is included, consult the documentation.[1] If the multitenant Azure service you are using is not included in the list, such as Azure DevOps agent pools, you need to configure the Key Vault Firewall with network allow lists.

[1] https://docs.microsoft.com/en-us/azure/key-vault/general/overview-vnet-service-endpoints#trusted-services

Azure Key Vault Firewall's **network allow lists** support both public IPv4 address ranges and Azure virtual networks. Additionally, you can control access to your Key Vault using **private endpoints**.

Note Azure Key Vault Firewall rules apply to any data-plane operations, including access from the Azure Portal.

Logging

Azure Key Vault provides security logs for management and data planes. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Firewall configuration changes (`properties.networkAcls`)
- Permission model changes (`properties.enableRbacAuthorization`)
- Role-based access control changes (`roleAssignments/write`)
- Access policy changes (`properties.accessPolicies`)

Azure Key Vault does not keep data-plane logs by default. To enable data-plane logging, you need to configure `AuditEvent` resource logs (under diagnostic settings) to be sent to your centralized log store. These logs include create, read, update, and delete (CRUD) operations to the Key Vault data plane. These logs include

- Operation name
- Caller IP address
- Caller identity

Additionally, Azure Defender for Key Vault analyzes data-plane logs and creates Security Center alerts for anomalies. These alerts include

- Unusual application accessed a key vault.
- Suspicious policy change and secret query in a key vault.
- Unusual operation pattern in a key vault.

Best Practices

To enforce proper operational best practices, you should ensure that keys, secrets, and certificates stored in Azure Key Vault are properly rotated. You can do this natively by configuring versions and expirations or any content you store in the Key Vault. You can also create custom workflows for advanced scenarios by subscribing to Azure Event Grid events from Azure Key Vault.[2]

To mitigate the risk of lost keys and user error, you can enable the soft delete and purge protection features. This keeps your deleted data recoverable for up to 90 days.

Azure Blob Storage

Azure Blob storage is an object storage service, built for storing unstructured data such as binary files. As one of the first platform-as-a-service components released in Microsoft Azure, Blob storage is a core component for most applications built natively for the cloud.

A variant of Azure Blob storage, Azure Data Lake Storage Gen 2, supports a hierarchical namespace filesystem. This makes Azure Data Lake Storage Gen 2 optimal for analytics workloads. Azure Data Lake Storage Gen 2 supports most of the security features of Azure Blob storage.[3]

Access Control

Access to Azure Blob storage is controlled using Azure Active Directory or shared keys (sometimes called Storage Account keys). **Whenever possible, you should use Azure Active Directory-based authentication**. If you are dealing with legacy workloads or third-party solutions, you might need to revert to using shared keys.

Public read-only access is often required for workloads serving web application content in Blob storage. Users with sufficient privileges (or using shared keys) can enable public access per blob item or container level.

[2] https://docs.microsoft.com/en-us/azure/key-vault/general/event-grid-overview

[3] https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-supported-blob-storage-features

Data-Plane Role-Based Access Control

You should manage access to the data plane of Azure storage account using data-plane role-based access control roles. The available built-in roles are available for you to assign in an account or container scope:

- **Storage Blob Data Owner**: Used to manage POSIX access control for Azure Data Lake Storage Gen 2
- **Storage Blob Data Contributor**: Used to grant read/write/delete permissions to Blob storage resources
- **Storage Blob Data Reader**: Used to grant read-only permissions to Blob storage resources
- **Storage Blob Delegator**: Used to create a shared access signature that is signed with Azure AD credentials for a container or blob

Shared Key Access

Shared keys, sometimes referenced to as account keys, are the oldest way of authorizing access to Azure Blob storage. With shared keys, a user gets full control to the storage account. This access allows for create, read, update, and delete operations for any data stored within the Blob storage account. Additionally, shared case also allows to change any access control settings, such as setting a blob as publicly available. It only takes the name of the storage account and the shared key to gain full access to the storage account.

Any user that has access to the `Microsoft.Storage/storageAccounts/listkeys/action` operation in your storage account can use the shared key. The built-in Contributor role is an example of a typical role that has this access.

Note Contributor role has access to read and regenerate shared keys. By extension, Contributor can manage access to the data stored in Azure Blob storage.

You should avoid using shared key authorization whenever possible. If you are sure you are not using shared key authorization, you can configure the `AllowSharedKeyAccess` property in your storage account resource to prevent its usage altogether. This property forces Azure Active Directory authorization for the storage account. Crucially, this property is something a Contributor can change!

If you need to use shared keys, you should rotate the keys periodically. You can set your own key rotation policies by using the Storage Account Key Operator Service Role or automate the key rotation using the `az keyvault storage` command.[4]

Delegated Access

Some scenarios require you to share the content of your Blob storage to external users while still applying granular access control. For example, a media bank application could allow public access for sample content but would require additional controls for paid content. This is where user delegation shared access signature (SAS) comes in.[5]

In this model, you control access via a shared access signature token, which is a key that provides restricted access to data in your storage account. Together with a URI to a file in your Blob storage, anyone using the SAS can access your data.

Azure Active Directory identity is used to sign the SAS, effectively **delegating** the identity's permissions to the holder of the SAS key. You can limit the effective permissions even further by setting granular permissions to storage account, container, or blob level using SAS parameters. Table 5-1 describes the available options for limiting the permissions using SAS parameters. Notably, access can be limited for a certain time period and even with network controls.

Prior to the introduction of the user delegation model, SAS keys were created and signed using shared keys. This is still possible for backward compatibility purposes. However, its usage should be avoided. The delegated permissions used for creating SAS keys with shared keys would effectively be the root access, potentially exposing your storage accounts.

[4] https://docs.microsoft.com/en-us/azure/key-vault/secrets/overview-storage-keys#manage-storage-account-keys

[5] https://docs.microsoft.com/en-us/rest/api/storageservices/delegate-access-with-shared-access-signature

Anonymous Access

You might have business requirements to expose data in your Blob storage account to anonymous users. By default, containers and blobs can be set publicly accessible by an administrator (or the holder of a shared key). The available public access levels per container are

- No public read access for container or blobs (default setting)
- Public read access for blobs only
- Public read access for container and its blobs

You can prevent this by enforcing the `AllowBlobPublicAccess` property. When the property is set to false, anonymous access is not allowed, regardless of the public access level.

Note You can prevent public anonymous access by setting the `AllowBlobPublicAccess` property to false!

Table 5-1. SAS permissions

Permission	Allowed operations
Read	Read the content, block list, properties, and metadata of any blob in the container or directory
Create	Write a new blob, snapshot a blob, or copy a blob to a new blob
Write	Create or write content, properties, metadata, or block list. Snapshot or lease the blob
Delete	Delete a blob
List	List blobs
Move	Move a blob or a directory and its contents to a new location
signedIp	Specifies an IP address or a range of IP addresses from which to accept requests
signedProtocol	The protocol permitted for a request made with the SAS. Possible values are HTTPS+HTTP (default) or HTTPS
signedKeyStartTime	The start of the lifetime of the user delegation key
signedKeyExpiryTime	The end of the lifetime of the user delegation key

Network

Network access to Azure Blob storage is notoriously unprotected by default. To protect your Blob storage, you can set multiple controls in place.

First, you can limit public anonymous access by configuring the `AllowBlobPublicAccess` property. We discussed anonymous access in the previous section. To continue sharing access to blobs after enabling the property, you can use SAS tokens, which you can further control by configuring allowed IP addresses in the `signedIp` field.

Next, you can enforce your connections are always encrypted in transit, by setting the `supportsHttpsTrafficOnly` to true and `minimumTlsVersion` to the newest TLS version.

And finally, you can enable the **Azure Blob storage firewall**. This changes the value of default action within network rule set from allow to deny. After enabling the firewall, you need to specify allowed network locations with **network allow lists**. The network allow lists support both public IPv4 address ranges and Azure virtual networks. Additionally, you can control access to your Blob storage using private endpoints. This eliminates exposure to public networks and helps you prevent data exfiltration.

Logging

Azure Blob storage provides security logs for management and data planes. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Network control configuration changes (`networkAcls`, `supportsHttpsTrafficOnly` and `allowBlobPublicAccess`)
- Role-based access control changes (`roleAssignments/write`)
- Shared key usage (`listKeys` operation used to get the shared keys)

Azure Blob storage does not keep data-plane logs by default. To enable data-plane logging, you need to configure `StorageRead`, `StorageWrite`, and `StorageDelete` categories of resource logs (under diagnostic settings) to be sent to your centralized log store. The log schema includes

- **REST operation** and HTTP status.
- **Caller IP address** and port.
- **Caller identity**: OAuth, Kerberos, SAS Key, Account Key, or Anonymous. In case of OAuth or Kerberos, further details of the requestor identity are stored.[6]

Additionally, you can use Azure Defender to analyze data-plane logs and create Security Center alerts in case of anomalies. For Azure Blob storage, these alerts include[7]

- Access from a suspicious IP address.
- Anonymous scan of public storage containers.
- Potential malware uploaded to a storage account. This alert is based on hash reputation analysis.

Backup and Disaster Recovery

Azure Blob storage helps you protect the availability of your data in two ways. First, you control data availability by selecting a redundancy type for your storage account, as listed in Table 5-2.

[6] https://docs.microsoft.com/en-us/azure/storage/blobs/monitor-blob-storage-reference#resource-logs-preview

[7] https://docs.microsoft.com/en-us/azure/security-center/alerts-reference#alerts-azurestorage

Table 5-2. *Storage redundancy options*

Tier	Number of copies	Durability
LRS: locally redundant storage	3 copies on separate nodes within a region	99.999999999% (11 9's)
ZRS: zone-redundant storage	3 copies across separate availability zones within a region	99.9999999999% (12 9's)
GRS: geo-redundant storage	6 copies: 3 in the primary region and 3 in the secondary region	99.99999999999999% (16 9's)
GZRS: geo-zone-redundant storage	6 copies, 3 across separate availability zones in the primary region and 3 locally redundant copies in the secondary region	99.99999999999999% (16 9's)

And second, you can enable soft delete and point-in-time restore to protect the data in your Blob storage against accidental deletion, corruption, or ransomware attacks.

Note Selecting a redundancy tier is not a substitute for backups. You are always responsible for adding sufficient backup and disaster recovery controls for your application needs.

Best Practices

While the data stored in Blob storage is always encrypted with Microsoft-managed keys (MMK) using AES 256-bit encryption, your organization might have requirements to control the key length, operations, or storage. To do that, you need to specify the storage account encryption type as customer-managed keys (CMK). Customer-managed keys can be stored either in Azure Key Vault or Azure Key Vault managed hardware security module (HSM). Using customer-managed keys, Azure Storage supports RSA and RSA-HSM encryption keys of sizes 2048, 3072, and 4096.[8]

[8] https://docs.microsoft.com/en-us/azure/storage/common/customer-managed-keys-configure-key-vault?tabs=portal#add-a-key

To support multitenancy or isolation requirements, you can encrypt different encryption scopes in your storage account using multiple keys. For example, you can use Microsoft-managed keys to encrypt most of the storage account, but encrypt an individual container or blob using your own encryption key. Azure supports up to 10 000 of these encryption scopes per storage account.[9]

As an operational-based practice, you should limit access to the list storage account keys operation to a minimum. If you are not able to prevent it altogether, you should closely monitor usage of this operation with custom activity log alerts.

While Azure Defender alerts you for malware based on hash comparison, Azure does not have built-in support for anti-malware scanning of Blob storage files. To implement your own logic for scanning uploaded files, you can look at Azure Event Grid. Your scanning workflow should start when `Microsoft.Storage.BlobCreated` event is triggered.

EXERCISE

Your organization is about to launch the next-generation product. As part of the launch campaign, you are asked to secure the storage account that stores the marketing images for the campaign website and product datasheets, available for interested customers after signing up. How would you configure the network and access controls of the storage account?

Bonus Exercise

A week before the public launch, you receive a new requirement. You would need to provide early access to product data sheets to an internal test group. What would you need to change in the storage account configuration to grant the access securely?

Azure SQL Database

Azure SQL database is another key data service in Microsoft Azure. Azure SQL database is a managed relational database as a service, based on Microsoft SQL Server engine. Like Blob storage, Azure SQL database has been available from the very beginning.

[9] https://docs.microsoft.com/en-us/azure/storage/blobs/encryption-scope-overview#key-management

This means that the service is mature and supports most of the Azure security controls. However, given its history, you might find that some of the controls are implemented differently than in newer services, such as Azure Key Vault.

There are variants of Azure SQL database in terms of database technology and tenancy. In this book, we are focusing on the multitenant SQL database and leaving Azure SQL Managed Instance out of the conversation. Similarly, we will not be discussing security controls for Azure database for MySQL, PostgreSQL, or MariaDB. Finally, some key controls such as Azure Active Directory authentication are also available in Azure Synapse Analytics (formerly known as Azure SQL DataWarehouse). Microsoft documentation pages usually clearly indicate any differences between Azure Database versions.

Access Control

Authentication to Azure SQL database is controlled by either of the two authentication methods: Azure Active Directory authentication or SQL authentication. **Whenever possible, you should use Azure Active Directory authentication**.

SQL Authentication

As a best practice, avoid using SQL authentication whenever possible and use Azure AD as a centrally managed identity provider instead. Similar to Blob storage shared keys, a Contributor can reset the logical server admin login credentials.

You can prevent the usage of SQL authentication by setting the logical server's `azureADOnlyAuthentications` property to true. This property enables **Azure AD authentication only mode** and disables SQL authentication in the server level. Effectively, no SQL login can connect to your Azure SQL database after setting the Azure AD only mode.

Azure AD Authentication

You should use Azure Active Directory authentication to manage identities of your database users in a central location. This reduces the operational responsibilities of your application team: they do not need to worry about password rotation policies or storing passwords themselves. When using Azure Active Directory authentication, your users also benefit from all that security trades of your Azure Active Directory, such as conditional access and multifactor authentication.

Azure Active Directory authentication is configured in the logical server as an Azure AD group assignment. There is a lower and upper limit of one assignment for the logical server Administrator role. Further role assignments should be granted in the data plane: in the database using Transact-SQL statements. Listing 5-1 illustrates how the database users can be created from the native Azure Active Directory.

Listing 5-1. Assignments of an AAD user, an AAD group, and a managed identity to the database.

```
CREATE USER [alice@contoso.com] FROM EXTERNAL PROVIDER;
CREATE USER [FINANCE-ADMINS] FROM EXTERNAL PROVIDER;
CREATE USER [financewebappmsi] FROM EXTERNAL PROVIDER;
```

Authorization

Once your users are successfully authenticated using Azure AD, authorization to data and operations within Azure SQL database should be done using standard database roles with principle of least privilege in mind. You can further limit access with role-level security.[10] As of the writing of this book, there are no data-plane role-based access control roles available that would map Azure RBAC assignments to database authorization.

Control-Plane Role-Based Access Control Roles

While there are no data-plane RBAC roles available for Azure SQL database, there are some management-plane RBAC roles which can be useful for you. The built-in roles for Azure SQL database are

- **SQL DB contributor**: Used to manage SQL databases, but not access to them. Excludes configuration of security-related policies

[10] https://docs.microsoft.com/en-us/sql/relational-databases/security/row-level-security?view=sql-server-ver15

- **SQL server contributor**: Used to manage SQL logical servers and databases, but not access to them. Excludes configuration of security-related policies
- **SQL security manager**: Used to manage the security-related policies of SQL servers and databases, but not access to them

Network

To protect network access to your SQL database, you can set multiple controls in place.

First, you can enforce your connections are always encrypted in transit, by setting the `minimalTlsVersion` to the newest TLS version.

Next, you can enable the logical server firewall. This changes `publicNetworkAccess` allow to deny. After enabling the firewall, you need to specify allowed network locations with `firewallRules` and `virtualNetworkRules`. Additionally, you can control access to your Azure SQL database logical server using private endpoints. This eliminates exposure to public networks and helps you prevent data exfiltration.

Finally, you can create IP firewall rules in the database level. This is done in the data plane, using Transact-SQL statements. If there are conflicts between the logical server and database firewall rules, **the database firewall rules prevail**. As a best practice, you should always use database-level firewall rules when you have more than one database in a logical server.

Figure 5-1 illustrates how firewall conflicts are handled. If an incoming request originates from an IP address that is part of the allowed database-level addresses, the request is accepted, regardless of the firewall settings of the logical server.

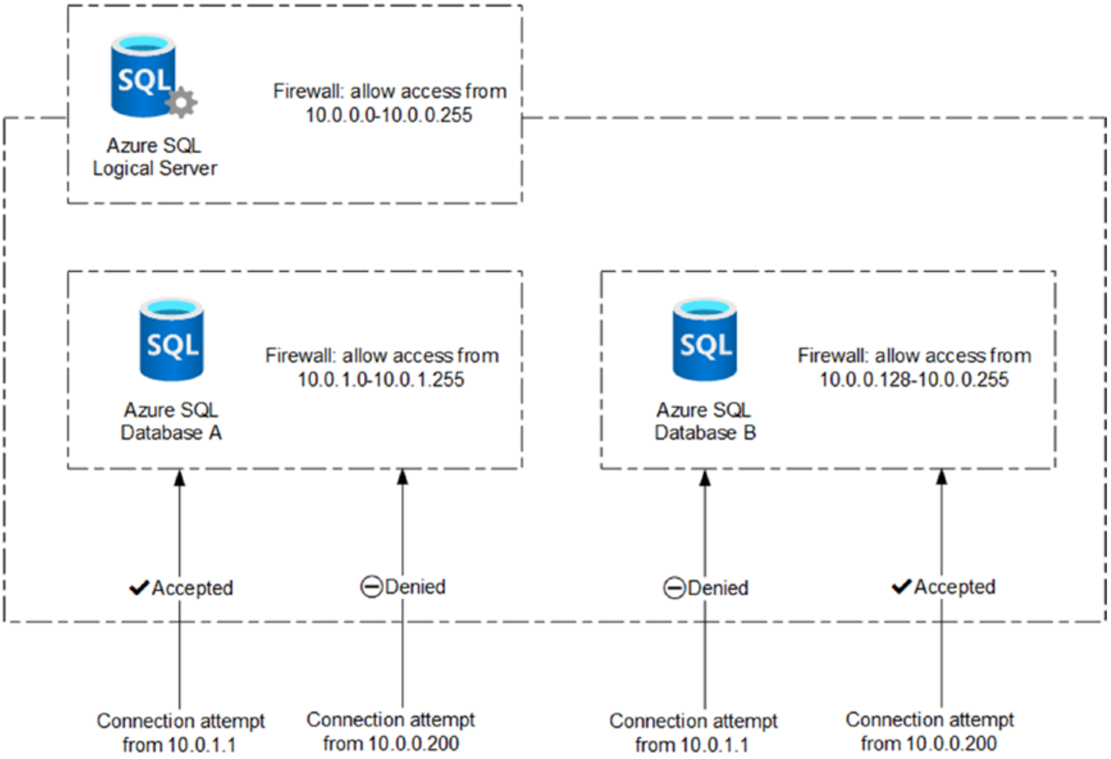

SQL
Azure SQL Logical Server
Firewall: allow access from 10.0.0.0-10.0.0.255
SQL
Azure SQL Database A
Firewall: allow access from 10.0.1.0-10.0.1.255
SQL
Azure SQL Database B
Firewall: allow access from 10.0.0.128-10.0.0.255
✓Accepted
⊖Denied
⊖Denied
✓Accepted
Connection attempt from 10.0.1.1
Connection attempt from 10.0.0.200
Connection attempt from 10.0.1.1
Connection attempt from 10.0.0.200

***Figure 5-1.** Azure SQL database's firewall hierarchy*

Logging

Azure SQL database does not keep data-plane audit logs by default. To enable audit logging, you need to configure the `auditingSettings` property to store the logs to your centralized log store. Azure SQL database auditing tracks database events and writes them into an audit log. As a best practice, you should set this only in the server level. If you need additional logs to be created with different settings per database, you can enable these per database, too. The audit log fields include[11]

- Name of the action, such as INSERT
- Source IP of the client application and service principal information of the login

[11] https://docs.microsoft.com/en-us/azure/azure-sql/database/audit-log-format#subheading-1

- T-SQL statement that was executed
- Information type and sensitivity label of the data queried

Note When you use Azure AD authentication, failed login records will not appear in the SQL audit log, but rather in the Azure AD sign-in logs!

Additionally, you can use Azure Defender to analyze data-plane logs and create Security Center alerts. For Azure SQL database, these alerts include[12]

- Login from a suspicious IP
- A possible vulnerability to SQL injection due to a faulty SQL statement in your database
- Potential SQL injection
- Potential SQL brute-force attempt

Finally, Azure Defender includes recurring scanning against SQL vulnerability assessment rules,[13] based on Microsoft best practices.

Backup and Disaster Recovery

As a managed PaaS service, Microsoft is responsible for most of the high availability implementation of Azure SQL database. Microsoft offers 99.99% availability service-level agreement (SLA) for Azure SQL database out of the box. If you choose business critical tier with zone-redundant deployments, you get availability SLA of 99.995%, recovery point objective of 5 seconds and recovery time objective of 30 seconds.

By default, Azure SQL database comes with a point-in-time restore feature, allowing you to restore the content of your entire database to its earlier state. The point-in-time restore timeframe is set to 7 days by default, so it is a good idea to change it to the maximum allowed 35 days. Your application team can restore the content in a self-

[12] https://docs.microsoft.com/en-us/azure/security-center/alerts-reference#alerts-sql-db-and-warehouse

[13] https://docs.microsoft.com/en-us/azure/azure-sql/database/sql-database-vulnerability-assessment-rules

service manner by replacing the current database or restoring the content to a new database – even in another geographic region.

To fulfill backup requirements that are over 35 days and up to 10 years, you need to configure the long-term backup retention feature, which leverages the full database backups that are created for the point-in-time restore (PITR) feature. The LTR policy for each database in SQL database can also specify how frequently the LTR backups are created (weekly, monthly, yearly).

Note The BACKUP Transact-SQL statement is not available in Azure SQL database.

To protect your application from regional datacenter downtime, you can configure auto-failover groups, which configure the active geo-replication and failover logic for Azure SQL database. You can initiate failover to another region manually or you can automate it based on a policy you define (`failoverWithDataLossGracePeriodMinutes`).

Best Practices

While the data stored in Azure SQL database is always encrypted with Microsoft-managed keys (MMK) using AES 256-bit encryption, your organization might have requirements to control the key length, operations, or storage. To do that, you need to specify the storage account encryption type as customer-managed keys (CMK). Using customer-managed keys, Azure SQL database supports RSA encryption keys of sizes of up to 4096 bits.

Azure SQL database supports some data governance features, such as automatic data discovery and classification. If you are not already using a similar solution, you should evaluate the usage of these features.

Finally, Azure SQL database provides server-side data protection for sensitive data. This feature – dynamic data masking – obfuscates the query result data before sending it to the clients. You can customize it to recognize your existing sensitivity labels or configure the masking manually per column.

Summary

In this chapter, we looked at the controls available to you for protecting Azure data services. As we learned, they share many basic controls, such as Azure AD authentication support and the concept of an IP address and virtual network firewall. However, details such as data-plane RBAC support or the options for enforcing Azure AD authentication differ from service to service.

For all the data services we covered in this chapter, the built-in control-plane RBAC role Contributor should be considered as a privileged role.

Finally, even though available logging options differ vastly in each service, the support for tracking core control-plane actions in the activity log and support for Azure Defender alerts for the data-plane actions provide a consistent set of controls that you can enforce at scale.

CHAPTER 6

Workload Protection – Platform as a Service

In this chapter, we discuss protecting platform-as-a-service workloads. After reading this chapter, you will be able select and implement appropriate controls for securing platform-as-a-service workloads as part of your organization's security policy framework.

Azure App Service

Azure App Service is a fully managed service for hosting web applications, REST APIs, and serverless applications. With App Service, Microsoft is responsible for maintaining the underlying virtual machine infrastructure, as well as patching and maintaining the operating system and middleware frameworks. In addition to the full web app feature set, App Service also provides opinionated flavors for REST API and mobile scenarios.

There are two hosting variants of Azure App Service. The more feature complete variant is using Windows server. Native Linux application stacks and container support are available in the second version, descriptively named App Service on Linux. App Service on Linux supports most of the security features of Azure App Service, hosted on Windows.

Similar to Azure SQL database, there are also tenancy variants of Azure App Service. In free and shared pricing tiers, your applications run on shared virtual machines, possibly with other Microsoft customers. In basic, standard, and premium tiers, your applications are hosted on virtual machines dedicated to you. And finally, the isolated tier provides you with even more control by running your applications in dedicated Azure virtual networks. If your security policies require you to host your applications

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_6

in environments you fully control, you should use the isolated tier. This allows you to deploy the App Service applications to your own App Service Environment,[1] instead of Microsoft-managed App Service plans.

Most of the security controls covered in this chapter are also available in Azure Logic Apps and Azure Logic Apps Integration Service Environment.

Access Control

Azure App Service provides several built-in features in the space of access control. First, incoming requests can be forced to provide authentication claims from federated identity providers, such as Azure Active Directory. This feature is sometimes referred to as Easy Auth or App Service authentication. For the purposes of this book, I am referring to it as built-in authentication.

You can also leverage the managed identity features natively with Azure App Service. This lets your application authenticate Azure platform service data workloads or first- or third-party APIs that support Azure Active Directory.

Finally, you can use the Key Vault reference or the Azure App Configuration[2] service to abstract your secret management away from your application code.

Built-In Authentication

Azure App Service built-in authentication implements federated identity, meaning that the identity provider you choose manages the authentication and authorization flow. You can use any OpenID Connect provide, but App Service has built-in support for Azure Active Directory, Facebook, Google, and Twitter and identity providers.

[1] https://docs.microsoft.com/en-us/azure/app-service/environment/overview

[2] https://docs.microsoft.com/en-us/security/benchmark/azure/baselines/app-config-security-baseline?toc=/azure/azure-app-configuration/TOC.json

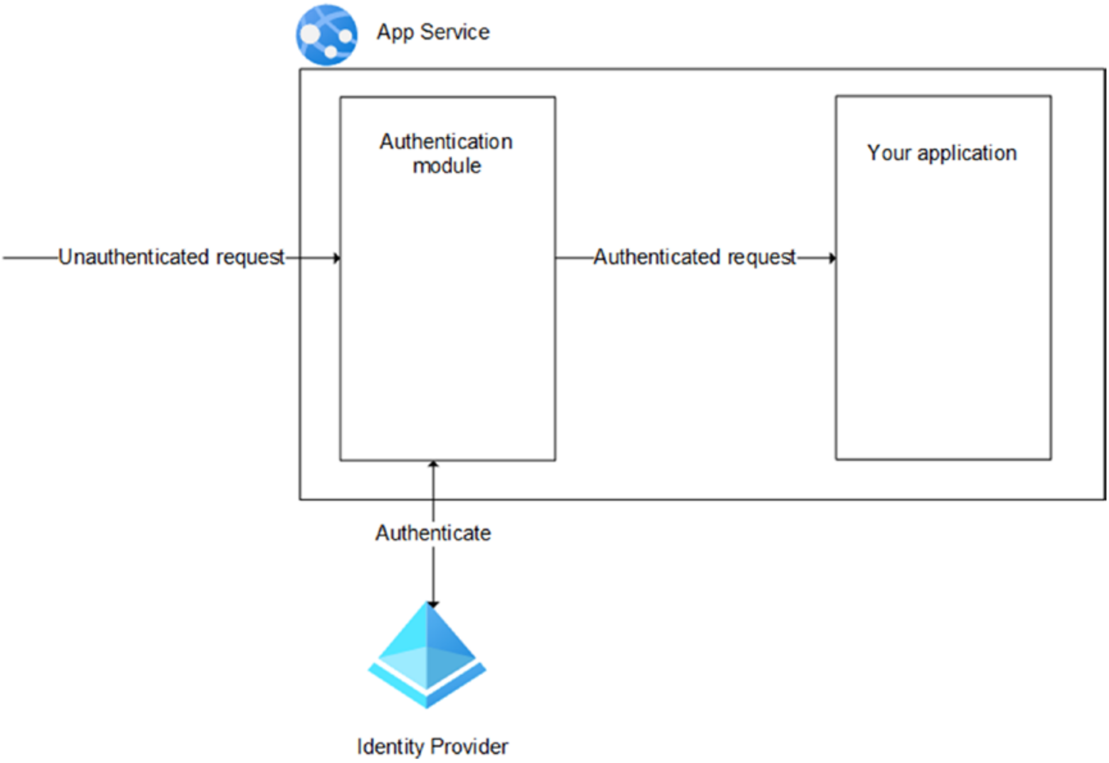
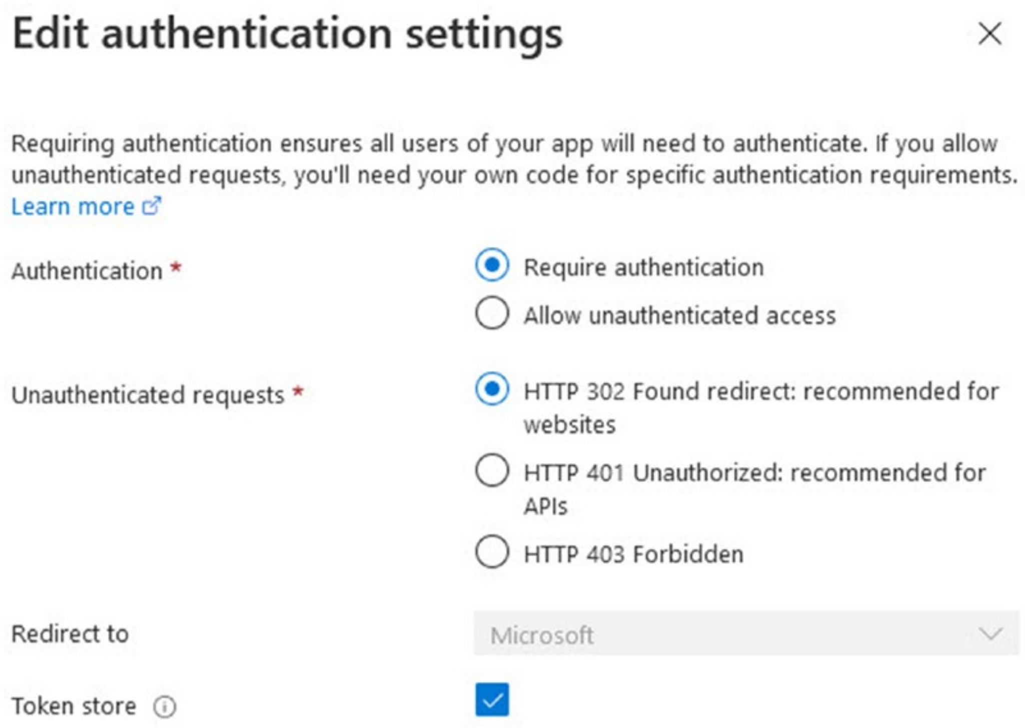


Figure 6-1. *App service built-in authentication*

Figure 6-1 illustrates the built-in authentication. After enabling the feature, your App Service instance passes all incoming requests through the authentication module. This module authenticates users with your selected identity provider and manages the authenticated sessions. It validates, stores, and refreshes tokens in a dedicated token store. App Service makes the user claims available for your application code by injecting them into headers.

Based on your application requirements, you can either require authentication for all requests or allow unauthenticated users access and provide them with a dedicated user experience before asking them to log in. This is configured in the authentication settings, as illustrated in Figure 6-2.



Edit authentication settings

Requiring authentication ensures all users of your app will need to authenticate. If you allow unauthenticated requests, you'll need your own code for specific authentication requirements. Learn more

Authentication *
● Require authentication
○ Allow unauthenticated access

Unauthenticated requests *
● HTTP 302 Found redirect: recommended for websites
○ HTTP 401 Unauthorized: recommended for APIs
○ HTTP 403 Forbidden

Redirect to: Microsoft

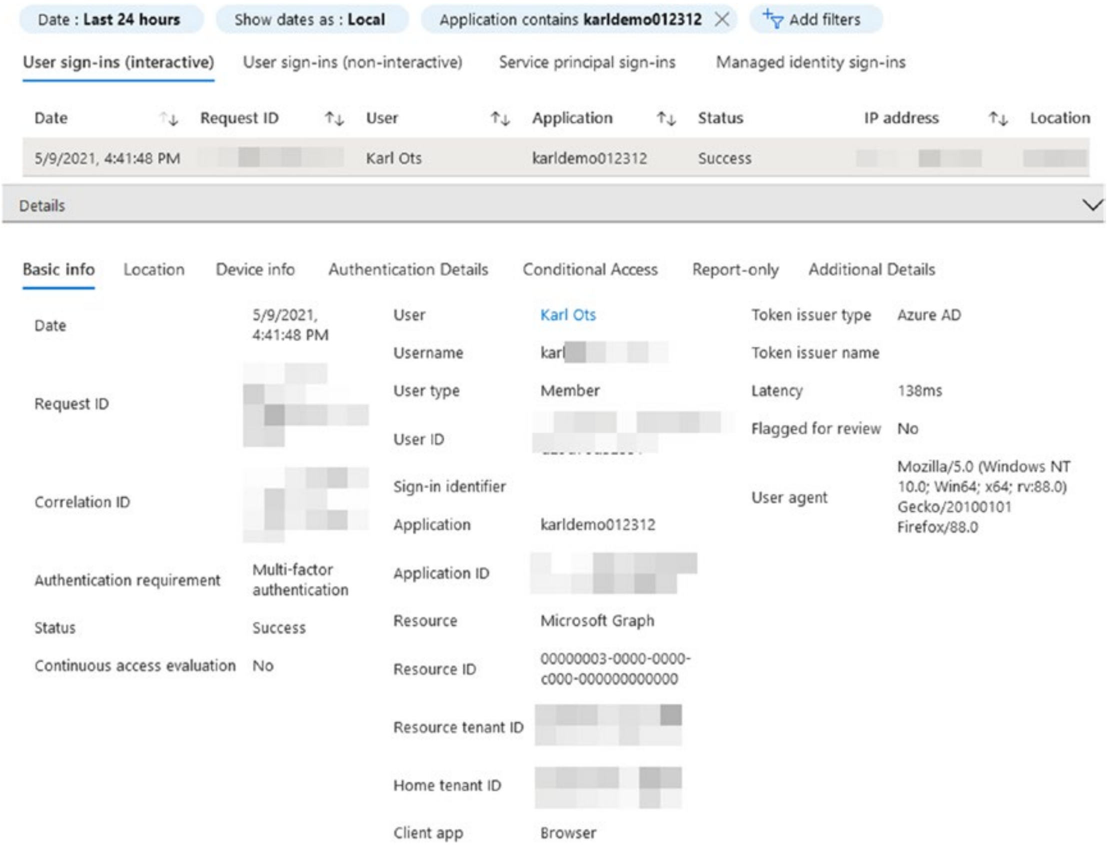
Token store ☑

Figure 6-2. *Available settings for handling unauthenticated request*

After authentication, you need to authorize your user. You can do this in code based on the user claims. You can enrich your authorization logic by queering your token store for additional details. You might also connect to an authorization API to verify your user's authorization level.

Relying on the platform features to manage your authorization logic is also one of the possibilities. For applications hosted in the Windows flavor of App Service, you can use `Web.config` to define authorization rules.

For applications hosted on either of the App Service flavors, you can use Azure Active Directory's application user assignments and application roles to authorize user access. Additionally, you can use Azure AD Conditional Access, as discussed in Chapter 2. This allows you to control application access by requiring multifactor authentication and limiting access by endpoint management state or network locations. By using Azure Active Directory as your identity provider, you get to manage, secure, and monitor your applications' authentications in a central location. Figure 6-3 illustrates how Azure App Service authentication logs are available in Azure Active Directory sign-in logs.



Date : **Last 24 hours** | Show dates as : **Local** | Application contains **karldemo012312** | Add filters

User sign-ins (interactive) | User sign-ins (non-interactive) | Service principal sign-ins | Managed identity sign-ins

Date	Request ID	User	Application	Status	IP address	Location
5/9/2021, 4:41:48 PM	[redacted]	Karl Ots	karldemo012312	Success	[redacted]	[redacted]

Details

Basic info | Location | Device info | Authentication Details | Conditional Access | Report-only | Additional Details

Date	5/9/2021, 4:41:48 PM	User	Karl Ots	Token issuer type	Azure AD
Request ID	[redacted]	Username	karl[redacted]	Token issuer name	
Correlation ID	[redacted]	User type	Member	Latency	138ms
Authentication requirement	Multi-factor authentication	User ID	[redacted]	Flagged for review	No
Status	Success	Sign-in identifier		User agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
Continuous access evaluation	No	Application	karldemo012312		
		Application ID	[redacted]		
		Resource	Microsoft Graph		
		Resource ID	00000003-0000-0000-c000-000000000000		
		Resource tenant ID	[redacted]		
		Home tenant ID	[redacted]		
		Client app	Browser		

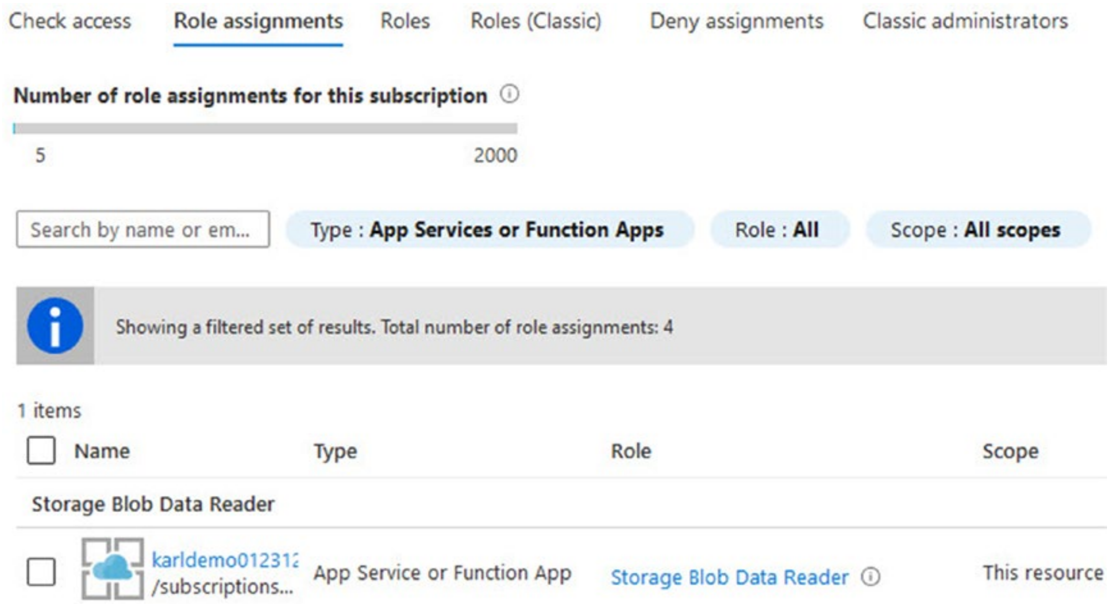
Figure 6-3. *Azure AD application sign-in logs*

Storage Access

You can grant your Azure App Service app access to your Azure data workloads using managed identities. This minimizes the use of connection strings and complexity of managing and rotating access keys.

You add a managed identity to your App Service simply by adding the `identity` property to your resource definition. Alternatively, you can add it at runtime using the `az webapp identity assign` command.

Once your App Service has a managed identity, you can use that as a target for your data-plane role-based access control assignments. For example, you can assign your App Service application a Storage Blob Data Reader role in the scope of your Blob storage account as illustrated in Figure 6-4.



Number of role assignments for this subscription

5 2000

Type : **App Services or Function Apps** Role : **All** Scope : **All scopes**

Showing a filtered set of results. Total number of role assignments: 4

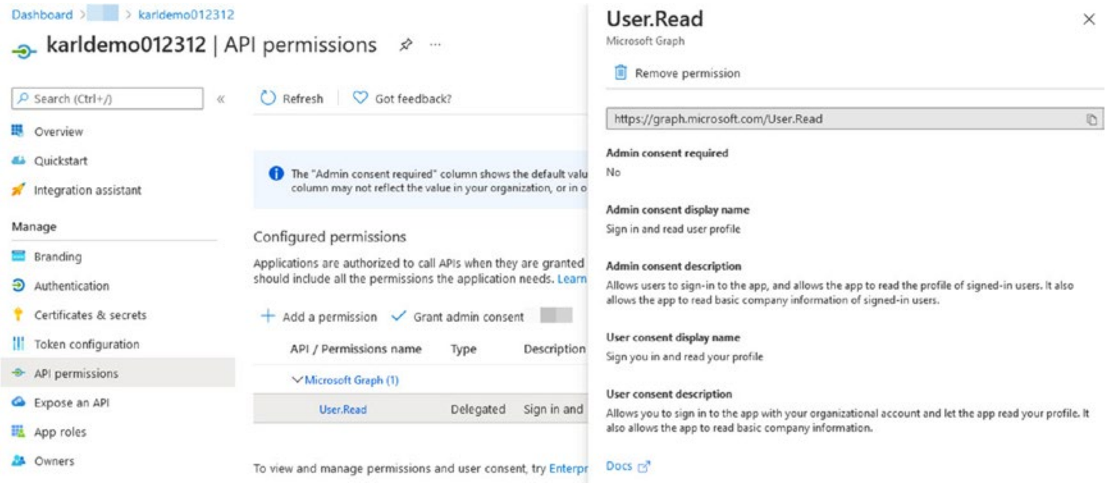
1 items

□	Name	Type	Role	Scope
	Storage Blob Data Reader			
□	karldemo01231 /subscriptions...	App Service or Function App	Storage Blob Data Reader	This resource

Figure 6-4. *Data-plane RBAC assignment to an App Service using managed identity*

API Access

Your application will most likely need to connect to other APIs to perform its functionality. These APIs could support simple API key authentication, in which case you should store those securely in Key Vault. If the APIs are exposed as applications to your Azure Active Directory, you can configure the API access as API permissions of your application in Azure Active Directory. Figure 6-5 illustrates granted Microsoft Graph permissions on our sample web application's application registration.



The "Admin consent required" column shows the default valu
column may not reflect the value in your organization, or in o

Configured permissions

Applications are authorized to call APIs when they are granted
should include all the permissions the application needs. Learn

API / Permissions name	Type	Description
∨ Microsoft Graph (1)		
User.Read	Delegated	Sign in and

To view and manage permissions and user consent, try Enterpr

User.Read
Microsoft Graph

https://graph.microsoft.com/User.Read

Admin consent required
No

Admin consent display name
Sign in and read user profile

Admin consent description
Allows users to sign-in to the app, and allows the app to read the profile of signed-in users. It also allows the app to read basic company information of signed-in users.

User consent display name
Sign you in and read your profile

User consent description
Allows you to sign in to the app with your organizational account and let the app read your profile. It also allows the app to read basic company information.

Figure 6-5. *API permissions of an Azure AD application*

Key Vault access

Azure App Service stores environment variables in application settings. Application settings are encrypted at rest and transmitted to your application at runtime while encrypted in transit. For Windows applications, application settings are available as settings in `Web.config` or `appsettings.json`. For Linux applications, the application settings are available as environmental variables.

You can store the content of your application settings in Azure Key Vault and access them using the managed identity of your App Service. The value of your application setting in that case would follow the syntax defined in Listing 6-1.

Listing 6-1. Application setting Key Vault reference

```
@Microsoft.KeyVault(VaultName=myvault;SecretName=mysecret)
```

Network

In this section, we explore the network controls available for Azure App Service multitenant models. As discussed earlier, isolated tier of Azure App Service, sometimes called App Service Environment, is placed in a virtual network that you control yourself. This means that you can use the same network controls as you can with virtual machines. This gives you more flexibility and control over your network but comes with added responsibilities and running costs.

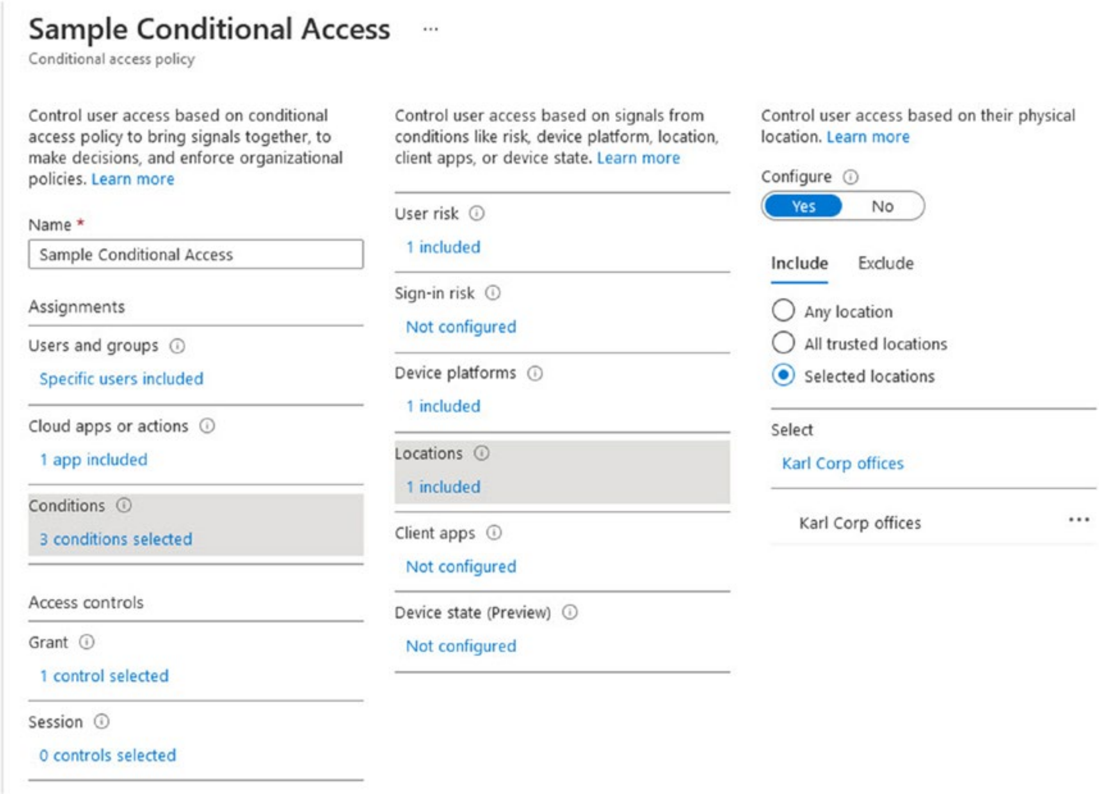
Controlling Inbound Traffic to App Service

By default, all inbound traffic to App Service apps is allowed. In the multitenant App Service, inbound traffic can be restricted using Access Restrictions or Azure AD Conditional Access.

When there is one or more entries to the allowed traffic source list, an implicit "deny all" rule exists at the end of the list. Access Restriction rules can protect both management operations (the App Service management portal, sometimes called Kudu portal) and the actual application content. Access Restriction rules can be set to limit incoming traffic based on

- IP address ranges
- Virtual network service endpoints
- Virtual network service tags
- HTTP headers
- Azure Front Door instance ID

Azure AD Conditional Access lets you set additional conditions which each authentication attempt needs to meet to allow authentication, after successfully authenticating using username and password. These conditions can include device management state conditions, such as requiring the devices where the authentication is being performed to be enrolled to your organization's domain. The conditions can also include enforcement of multifactor authentication. Lastly, the conditional access can require the authentication attempt to originate from a certain network location (a country or allow list of IP address ranges). Figure 6-6 illustrates the network conditions of a conditional access rule, in this case assigned to the Azure AD application that is used as the identity for our App Service built-in authentication.



Sample Conditional Access ···

Conditional access policy

Control user access based on conditional access policy to bring signals together, to make decisions, and enforce organizational policies. Learn more

Name *

Sample Conditional Access

Assignments

Users and groups ⓘ

Specific users included

Cloud apps or actions ⓘ

1 app included

Conditions ⓘ

3 conditions selected

Access controls

Grant ⓘ

1 control selected

Session ⓘ

0 controls selected

Control user access based on signals from conditions like risk, device platform, location, client apps, or device state. Learn more

User risk ⓘ

1 included

Sign-in risk ⓘ

Not configured

Device platforms ⓘ

1 included

Locations ⓘ

1 included

Client apps ⓘ

Not configured

Device state (Preview) ⓘ

Not configured

Control user access based on their physical location. Learn more

Configure ⓘ

[✓ Yes] No

Include Exclude

○ Any location

○ All trusted locations

● Selected locations

Select

Karl Corp offices

Karl Corp offices ···

Figure 6-6. *Conditional Access network controls*

Controlling Outbound Traffic

You can control outbound traffic from Azure App Service through virtual network integration or the App Service Hybrid Connections feature.

App Service **virtual network integration** allows outbound connectivity from your App Service to your virtual network. Your application can then reach resources in the virtual network. If your application requires, you can route all outbound traffic to that virtual network.

Figure 6-7 illustrates the Azure App Service networking controls. First, Access Restrictions are used to deny inbound connectivity from the Internet and allow inbound connectivity from the specific IP address range. Second, virtual network integration is configured for controlling outbound traffic from the App Service application. Third, integration subnet is configured to allow traffic from the App Service endpoint, and the back-end subnet's network security group is configured to allow traffic from the integration subnet. And finally, all outbound traffic is enforced to go through the virtual network using the `WEBSITE_VNET_ROUTE_ALL` setting in App Service.

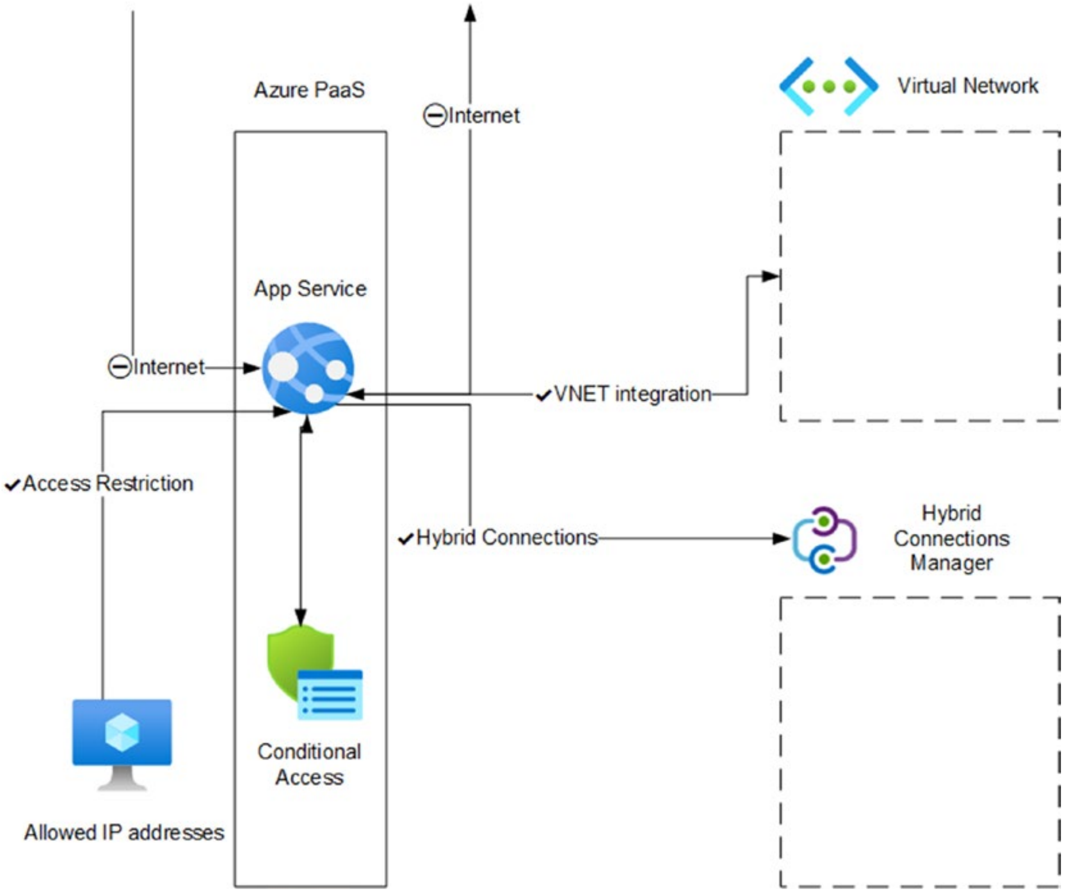


Figure 6-7. App Service networking controls

App Service Hybrid Connections enables access to your on-premises resources, without changes to on-premises inbound firewall rules. Instead, Hybrid Connections depends on a relay agent to be installed on your on-premises network and outbound TCP connectivity to Azure over port 443.

Web Application Firewall

Azure Web Application Firewall (WAF) is a managed, OSI Layer 7 firewall. Azure Web Application Firewall functionality can be deployed to Azure Application Gateway or Azure Front Door. Azure WAF is based on OWASP Core Rule Set, updated, and managed by Microsoft. You can customize the actions taken for each individual rule in the Microsoft-managed default rule set. Azure WAF protects your web applications from common exploits and vulnerabilities by inspecting inbound web traffic for attacks such as SQL injections, cross-site scripting, malware uploads, and DDoS attacks.

In addition to blocking malicious traffic reaching your application, you can collect the data-plane logs from your Web Application Firewall and store them in your centralized log storage. The relevant log names vary based on which service you are using Web Application Firewall with, but they consist of `AccessLog` and `FirewallLog`. The former log stores all requests, and the latter stores requests that match your WAF rules.

In addition to OWASP WAF rules, you can add custom rules to Azure WAF. These rules can include

- IP address range allow/deny lists
- HTTP parameter-based access control
- Geographic filtering access control

To force all incoming traffic to your App Service to go through the Azure Web Application Firewall, you need to use App Service Access Restrictions feature to allow access only from the certain IP addresses (Application Gateway) or from a list of Front Door ids.

Encryption in Transit

By default, Azure App Service accepts both incoming HTTP and HTTPS traffic. To redirect all HTTP to HTTPS, you need to configure the `requireHttps` property. To enforce minimum TLS version, configure the `minTlsVersion` property. Both properties are available through Azure Resource Manager in the `Microsoft.Web` Resource Provider.

Logging

App Service provides security logs for management and data planes. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Administrative operations, such as deleting or restarting a web app
- Built-in authentication changes (write operation to Microsoft.Web/sites/Config/authsettings and Microsoft.Web/sites/Config/authsettingsV2)
- Role-based access control changes (`roleAssignments/write`)

Note App Service does not provide detailed change information in the activity logs. It only provides information about the scope of changes (such as changes to Config or Authsettings). To manage changes (or removal of) to Access Restrictions, you need to set compensating controls in place, such as limiting changes of the App Service using Locks or RBAC.

App Service does not keep data-plane logs by default. To enable data-plane logging, you need to configure the App Service resource logs (under diagnostic settings) to be sent to your centralized log store. These logs include

- **AppServiceAuditLogs**: Login activity via FTP and App Service management portal Kudu.
- **AppServiceIPSecLogs**: Allowed and denied requests made to the web app. Depends on Access Restrictions being enabled.
- **AppServiceAntivirusScanAuditLogs**: Logs from the antivirus scan using Microsoft Defender. The scan is performed once per day and cannot be started manually.
- **Application logs:** Include the logs generated from your application framework or custom code. From the security perspective, these may include authentication and authorization traces of the built-in authentication (EasyAuthModule_32/63).

- **Web server logs**: Raw HTTP request data. Each log message includes the HTTP method, resource URI, client IP, client port, user agent, and response code.

Additionally, you can use Azure Defender to analyze data-plane logs and create Security Center alerts in case of anomalies. For App Service, these alerts include[3]

- Access from a suspicious IP address
- Detected file download from a known malicious source
- NMap scanning detected
- Potential reverse shell detected

EXERCISE

You are tasked to secure the front-end application of an internal business solution. The application is hosted in Azure App Service, and its use should be restricted to all internal company employees. Additionally, access is only allowed from the company network. Which Azure services will you configure and how?

Bonus Exercise

After an incident, you are tasked to investigate whether at any point was your internal business solution unprotected. How would you ascertain if the application were accessed from outside the company network?

Azure Functions

Azure Functions is a serverless compute service. Functions can execute your application code in more granular units than a complete web application. Typically Functions host web APIs, scheduled workloads, or event-driven applications. Functions can be hosted in Consumption, Premium, or App Service plans. The availability of security controls becomes gradually better with each pricing plan upgrade.

[3] https://docs.microsoft.com/en-us/azure/security-center/alerts-reference#alerts-azureappserv

Access Control

By default, access to Function apps is controlled using key-based access control, which can be scoped to the root level as admin keys or app level as Function keys. This is not a desirable method of authentication, as you would often need to expose the key to your clients. If you use key-based authentication, you should store the key sin Azure Keu Vault. To configure this, set the `AzureWebJobsSecretStorageType` to `keyvault` in the Function app application settings and populate the `AzureWebJobsSecretStorageKeyVaultName` setting with your vault name.

You should avoid using the host keys (either function keys or admin keys) and use built-in authentication instead. Alternatively, you can perform the access control logic in Azure API management and set the API management instance as the only allowed source of traffic using Access Restrictions.

To call Azure data services from your Functions app, you can assign your Function apps managed identities and grant the data-plane RBAC access to their target data services.

Network

By default, all inbound traffic to Functions apps is allowed. You can control inbound network access using Access Restrictions. Access Restrictions is available for all Functions tiers: Consumption, Premium, and App Service plan. As for App Service, Access Restriction rules for Functions can be set to limit incoming traffic based on

- IP address ranges
- Virtual network service endpoints
- Virtual network service tags
- HTTP headers
- Azure Front Door instance ID

To allow access only through Azure Front Door's Web Application Firewall, you should use Access Restrictions with Azure Front Door instance ID. To allow access only through you Azure API management instance, you can use Access Restrictions with IP address ranges as traffic source.

To reach resources within virtual networks, you need to enable the virtual network integration feature. Virtual network integration is available in Premium and App Service plans.

Logging

Function apps provide security logs for management and data planes. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Administrative operations, such as deleting or restarting a web app
- Role-based access control changes (`roleAssignments/write`)

Function apps do not keep data-plane logs by default. To enable data-plane logging, you need to configure `FunctionAppLogs` resource logs (under diagnostic settings) to be sent to your centralized log store. These logs include your self-generated logs and runtime-generated logs. They consist of

- Name of the function that logged the message
- Exception details
- The log message

Note There are no AppServiceIPSecLogs available for Function apps. To monitor traffic to/from Function apps, you need to enforce the Function app to pass another service, such as Web Application Firewall or virtual network.

Best Practices for Azure Compute PaaS

Applications hosted in Azure Functions are generally greenfield applications and often stateless. Applications hosted in App Service can also be refactored applications from other hosting providers. If you are working with a modern application, you might have the luxury of storing the code, configuration, and infrastructure as code. In that case, redeployment of your application is a preferred approach over restoring a backup. If you are storing state in your application, you can use the App Service Backup feature. App Service Backup backs your application configuration and file content to Azure Blob storage account. App Service Backup is available in standard and higher pricing tiers.

There are also several limitations, such as firewall-enabled storage accounts not being supported.

Whenever you set Access Restrictions, you should also protect your App Service management portal (KUDU).

If your application is a RESTful API, you should protect it with cross-origin resource sharing (CORS). Both App Service and Functions apps support managing the allowed CORS origins, which you should use to control the list of domains from which you allow requests. With CORS enabled, responses include the `Access-Control-Allow-Origin` header.

By default, plain text FTP deployments are allowed for both App Service apps and Functions. If you need FTP to deploy your solution, you should enforce your App Service and Function apps to use FTPS (`ftpsState` property). Even better if you can set the property to Disabled.

You should enforce the `remoteDebuggingEnabled` property to `false` to disable remote visual studio debugging capability.

Summary

In this chapter, we looked at the controls available to you for protecting Azure compute PaaS services. As we learned, they share many basic controls, such as built-in authentication, Access Restrictions, and managed identities. However, details of network controls and logging support differ from service to service.

In the context of security features, App Service can be considered as a superset of Functions. Within the full-fledged App Service, the Windows variant of web app offers the most security features.

CHAPTER 7

Workload Protection – Containers

In this chapter, we discuss protecting container workloads in Azure. After reading this chapter, you will be able to select and implement appropriate controls for securing container workloads as part of your organization's security policy framework.

Containers can be hosted in multiple services in Azure, collectively referred to as container-as-a-service services. For the purposes of this book, I consider App Service for containers a platform-as-a-service service, as it includes further and opinionated web services such as scaling, load balancing, and authentication. Likewise, I consider virtual machines purely as infrastructure as a service, despite their support for nested virtualization and, thus, capability to act as a container host.

Container Security

Container security is a broad and nuanced topic. To set the context for this book, I will briefly explain the core security concepts. If you are familiar with container security, feel free to skip ahead to the sections covering specific Azure container-as-a-service security.

Build Security

Comprehensive container security starts with securing the container image. When a container image is built, it often uses a base image instead of defining the container image from scratch.

A base image is a parent image, typically containing at least the operating system and often purpose-built for specific workloads. For example, instead of a full-blown Ubuntu image, the more lightweight Alpine Linux could be used as a base image, reducing the attack surface. Standardizing and controlling the base image used for your container

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_7

applications is as important as standardizing and controlling so-called golden images for virtualized workloads. Furthermore, container images can build upon multiple layers of base images. This means that you can introduce opinionated base images on top of the generic Linux distributions, such as the enterprise standard base image for a Python application, which includes the necessary middleware in addition to the operating system.

After building your container image, you should scan it for vulnerabilities and remediate them before making your image available, such as pushing the image to your container registry.

Once you are satisfied with your newly built container image, your next task is to distribute it securely to your container workloads. In the cloud context, assurance of the integrity of the images is crucial. A common solution for this is to sign the images before distributing them. This allows you to limit your container hosts to run only signed images.

Registry Security

In addition to controlling the base images and the resulting application container images, you will need to control the usage of container registries within your organization. You can either rely on a single, centrally managed, private container registry, a distributed approach where teams manage their container registries according to your cloud security framework, or combine these in a mixed model. By consolidating your container registries, you can perform and act on vulnerability scans centrally.

Containers can be hosted in multiple services in Azure, each with their own sets of security controls. Often the application teams host their container applications in multiple or even all these services. Therefore, in large enterprises, one of the trickiest operational challenges is to gather an up-to-date inventory of container images that are running in your environment. You need to solve this to be able to mitigate any security issues, such as finding of new vulnerabilities in the images.

Once you are assured of the origin and the security state of the images in your container registries, you are ready to move on to container runtime security.

Runtime Security

The runtime security considerations can be split into the host hardening and application security.

Your host hardening responsibilities vary depending on the Azure service you are using to run your container images in. If you are hosting the container in infrastructure as a service, you are responsible for the host and should secure it using checklists such as the CIS Docker Benchmark.[1]

For container-as-a-service services, you need to understand the specifics of the individual services you choose to use. Azure Kubernetes Service provides a significant number of additional controls to you, but also leaves you responsible for configuring many of them. Azure Container Instance, on the other hand, takes care for patching and hardening the underlying container host, but limits the controls you have available for protecting the workload.

Your choice of container hosting service also dictates which controls are available to you for securing your container workloads. If you need to run multiple containerized applications in the same environment, Azure Kubernetes Service is a likely fit for you. By choosing Azure Kubernetes Service, however, you are responsible for a larger set of controls, such as network segmentation and enforcing the containers are executed using least privileges.

Finally, there are some application-level controls you need to apply in any service you are using to host your container workloads. These include access control to the registry, secret management, monitoring, and data-plane network control.

Azure Container Registry

Azure Container Registry is a service for storing, distributing, scanning, and managing container images. Azure Container Registry is based on the open source Open Container Initiative (OCI) Distribution Specification and supports Docker images, Helm charts, OCI images, and OCI artifacts. Azure Container Registry also supports automating build tasks using the ACR tasks feature.

[1] www.cisecurity.org/benchmark/docker/

Wherever you end up running your container images in Azure, you should use Azure Container Registry or a similar private container registry to control your container image life cycle.

Access Control

Access to container registry is controlled using Azure Active Directory or admin keys (sometimes called master keys). **Whenever possible, you should use Azure Active Directory-based authentication**. If you are dealing with legacy workloads or third-party solutions, you might need to revert to using admin keys.

Data-Plane Role-Based Access Control

Access to the Azure Container Registry control plane is controlled using Azure role-based access control. The default RBAC roles Owner, Contributor, and Reader allow you to manage access to, configure, and the content of your Azure Container Registry.

Note The Reader role grants access to the data plane. Specifically, it lets you pull images from the Azure Container Registry. Keep this in mind when considering the resource group placement and RBAC access of your Azure Container Registry instance.

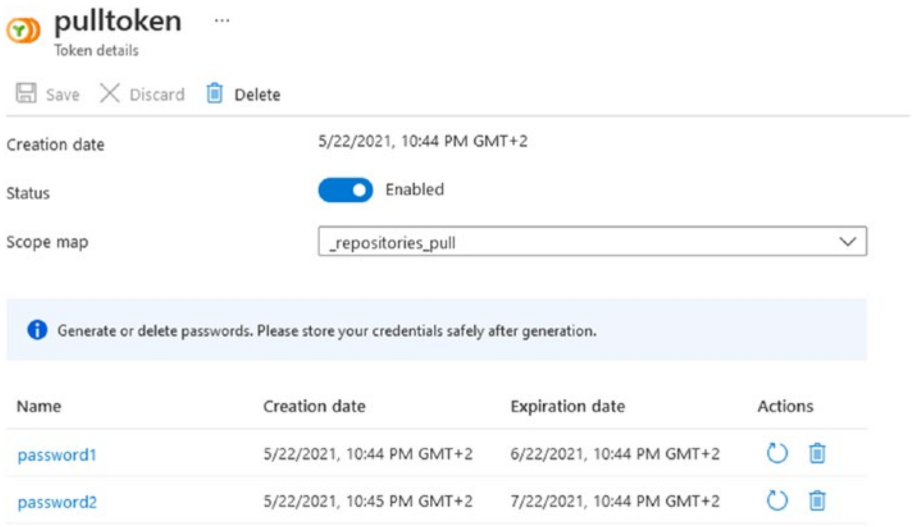
The following built-in roles complement the default roles and grant access to the data plane of Azure Container Registry:

- **AcrPush**: Grants access to push and pull images
- **AcrPull**: Grants access to pull images
- **AcrDelete**: Grants access to delete images or other artefacts
- **AcrImageSigner**: Grants access to sign images if content trust is enabled. Typically used together with the AcrPush role, as part of automated CI/CD access

Access Control Without Azure Active Directory

In addition to Azure Active Directory authentication, Azure Container Registry supports authentication with an admin account. The admin account is similar to the shared key access of Azure Blob storage: it applies to the whole registry and provides both push and pull access. Admin user is meant for testing purposes. Using admin user is not recommended, and it is disabled by default. Regretfully, as we went to press, the portal experiences of some Azure services such as Azure Container Instance and App Service Web App for Containers are still using the admin account to control access to the Azure Container Registry.

When Azure Active Directory authentication is not viable, instead of admin account, you can control third-party access to containers in your Azure Container Registry using repository-scoped tokens, as illustrated in Figure 7-1. This method is similar to the delegated access of Azure blob storage. A token along with a generated password lets the user authenticate with the registry using `docker login` simple authentication. You can set an expiration date for a token password and revoke them when needed.



Creation date 5/22/2021, 10:44 PM GMT+2

Status Enabled

Scope map _repositories_pull

Generate or delete passwords. Please store your credentials safely after generation.

Name	Creation date	Expiration date	Actions
password1	5/22/2021, 10:44 PM GMT+2	6/22/2021, 10:44 PM GMT+2	
password2	5/22/2021, 10:45 PM GMT+2	7/22/2021, 10:44 PM GMT+2	

Figure 7-1. *Azure Container Registry repository-scoped tokens*

Lastly, Azure Container Registry can also be configured to allow anonymous pull access. When enabled, the feature allows anonymous pull operations to all the artefacts in your repository, so it should not be enabled in a registry you are using to manage your private container images. To prevent anonymous pull access, enforce the `anonymousPullEnabled` property to `False`.

Network

Network access to the data-plane Azure Container Registry is unrestricted by default: the container images can be pulled from any network location. To protect network access to your Azure Container Registry, you can set multiple controls in place.

First, you can enable the registry firewall. This changes `publicNetworkAccess` from Enabled to Disabled. After enabling the firewall, you need to specify allowed network locations with `ipRules`. Additionally, you can control access to your registry using private endpoints. This minimizes exposure to public networks and helps you prevent data exfiltration.

Note Azure Container Registry supports a maximum of 100 network access rules and 10 private endpoints per instance.

To allow access from other Azure services without using IP rules or private endpoints, you can enable the Allow trusted services feature (configuring `networkRuleBypassOptions` to `AzureServices`). This enables access from other Azure Container Registries or Azure Machine Learning workspaces. Other registries can either import images directly or use images in your registry as a base image to build their application images. Azure Machine Learning workspaces can use your registry to deploy or train their models.

Enabling the Allow trusted services feature does not allow access from other managed Azure services, such as App Service, Azure Container Instances, or Azure Defender image scanning. To control access to those services, consider applying compensating controls.

Logging

Azure Container Registry provides security logs for management and data planes. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Administrative operations, such as deleting or restarting the registry
- Role-based access control changes (`roleAssignments/write`)

- Data-plane Access control changes, such as changes to the `adminUserEnabled` property or someone performing the `Microsoft.ContainerRegistry/registries/listCredentials/action` operation
- Network control operations, such as changes to `publicNetworkAccess`

Azure Container Registry does not keep data-plane logs by default. To enable data-plane logging, you need to configure the Azure Container Registry resource logs (under diagnostic settings) to be sent to your centralized log store. These logs include

- **ContainerRegistryLoginEvents**: Logs from registry authentication events and success status, including the identity and IP address
- **ContainerRegistryRepositoryEvents:** Logs from operations on content in registry repositories. The following operations are logged: push, pull, untag, delete (including repository delete), purge tag, and purge manifest. Includes identity and IP address information

To log network access to your Azure Container Registry, enable network security group flow logs for the subnets where the private endpoints that can access your Azure Container Registry are placed.

Additionally, you can use Azure Defender to automatically scan your images using Qualys. Once Azure Defender is enabled, Azure Defender pulls the images from your container registry and scans them in an isolated sandbox with the Qualys scanner. Images are scanned on push or import. If an image has been pulled in the last 30 days, it is also scanned weekly. The scan results are published as recommendations in Azure Security Center. To manage signal noise, you can also suppress findings by adding Disable rules.

Disable rules can be set using the following filters:

- Finding ID
- Finding category
- CVSS v3 score
- Severity
- Patchable status

Note Azure Defender cannot currently perform image vulnerability scanning in a registry that restricts networks access to private endpoints, virtual networks, or IP addresses.

Best Practices

In this section, we discuss additional security controls and best practices.

Encryption at Rest

While the data stored in Azure Container Registry is always encrypted with Microsoft-managed keys (MMK) using AES 256-bit encryption, your organization might have requirements to control the key length, operations, or storage. To do that, you need to specify the encryption type as customer-managed keys (CMK) and manage the encryption keys using Azure Key Vault. Using customer-managed keys, Azure Container Registry supports RSA encryption keys of sizes up to 4096 bits.

Automate Base Image Updates

You can set up an Azure Container Registry Task to track a dependency on a base image when it builds an application image. When the updated base image is pushed to your registry, or a base image is updated in a public repository, Azure Container Registry Task can automatically build the application images based on it. If your base image is hosted in another Azure Container Registry, the Task is triggered immediately. If your base image is stored in a public repository, such as Docker Hub or Microsoft Container Registry, the Task is triggered at a random interval between 10 and 60 minutes.

Image Signing

To enable support for signed images, Docker content trust, enable content trust on your Azure Container Registry. This sets the `TrustPolicy status` to `enabled`. Once enabled, signed images can be pushed to the container registry. When a signed image is pulled, the Docker client of the pull host verifies the integrity of the image.

High Availability

As a managed PaaS service, Microsoft is responsible for most of the high availability implementation of Azure Container Registry. Microsoft offers 99.9% availability SLA for Azure Container Registry out of the box. To improve the resiliency of your registry, you can configure it in the zone redundancy. To improve global performance and resiliency against regional outages, you can implement geo-replication to other regions.

EXERCISE

You are designing security controls for a high-priority business application. The application includes a containerized version of your latest machine learning models. The container images are stored in Azure Container Registry. You are required to provide an audit trail of each user that has had access to the model. How will you configure the container registry?

Bonus Exercise

You are required to provide network access logs of each successful and unsuccessful attempt at downloading the container image. How will you configure the container registry, and which data sources will you use for this?

Azure Container Instance

Azure Container Instance is one of the container-as-a-service options for running containers in Azure. With Azure Container Instance, Microsoft takes care of hardening and operating the underlying operating system, networking, and log integration. You are left with managing the container image: application and its supporting middleware. Compared to platform as a service, Azure Container Instance provides you with fewer features and looser platform integration. For example, network features are not as advanced, and there is no built-in authentication.

Azure Container Instance applications are defined in Container Groups, which are functionally like Kubernetes Pods.

Access Control

To create the Azure Container Instance, you need to authenticate the Azure Container Registry. As the Azure Container Instance resource does not exist at that point, it cannot be assigned a managed identity to access Azure Container Registry. To access the Azure Container Registry to pull your container image for the Azure Container Instance, you can use either service principal or repository-scoped tokens. Both approaches rely on a username and password to authenticate. The least privileged method of authenticating would be to use a token, which would be assigned the `repositories_pull` scope map. Service principal should be avoided as an authentication method for Azure Container Registry, as adversaries may add additional service principal credentials to maintain persistent access.[2]

Network

To protect access to and from your Azure Container Instance application, you can place it in a virtual network and limit it to use only a private IP address. The virtual network placement lets you protect an Internet accessible application in an Azure Container Instance by deploying a Web Application Firewall in front of your application. Additionally, with user-defined routes, you can enforce all outbound application traffic to go through a firewall.

Note Azure Container Instance is unable to access a network-protected container registry to pull images.

If you prefer to host the network controls in a sidecar and deploy it together with your application, you can expose the Azure Container Instance with a public IP address and even configure Azure to create a fully qualified domain name for it.

Whether you use private or public IP addresses, you will need to expose your application port on the IP address from the Container Group.

[2] https://attack.mitre.org/techniques/T1098/001/

Logging

Azure Container Instance provides security logs for management and data planes. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Administrative operations, such as deleting the Azure Container Instance resource
- Role-based access control changes (`roleAssignments/write`)

For data plane, container diagnostic events can be pulled with the `az container show` command. These logs include container deployment events, such as image pull, or container restart. Application-level logs from within your application can be pulled with the `az container logs` command.

To send application logs to your centralized log store, you need to implement log ingestion in the application. If you use Log Analytics Workspace, you can add your workspace credentials in the `properties.diagnostics` field of your Container Group deployment configuration. As of the writing of this book, this integration did not yet support managed identities.

Azure Kubernetes Service

Azure Kubernetes Service is a managed Kubernetes service in Azure. It is not, however, platform as a service. Microsoft is responsible for creating, configuring, and operating the **Kubernetes control plane** of your Azure Kubernetes Service. This includes Kubernetes API servers, Etcd, kube-dns, and other system components in the kube-system namespace. You are still responsible for parts of your Azure Kubernetes Service, such as network controls or agent node patching. Figure 7-2 illustrates this.

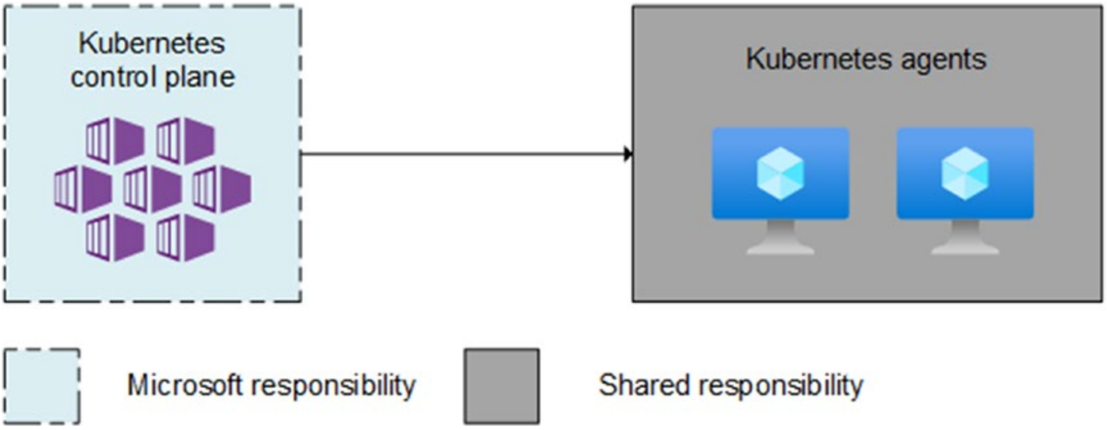


Figure 7-2. *Shared responsibility of Azure Kubernetes Service*

Access Control

Administrative access to Azure Kubernetes Service can be controlled using Azure role-based access control, Azure Active Directory, and Kubernetes role-based access control. Figure 7-3 illustrates these access controls.

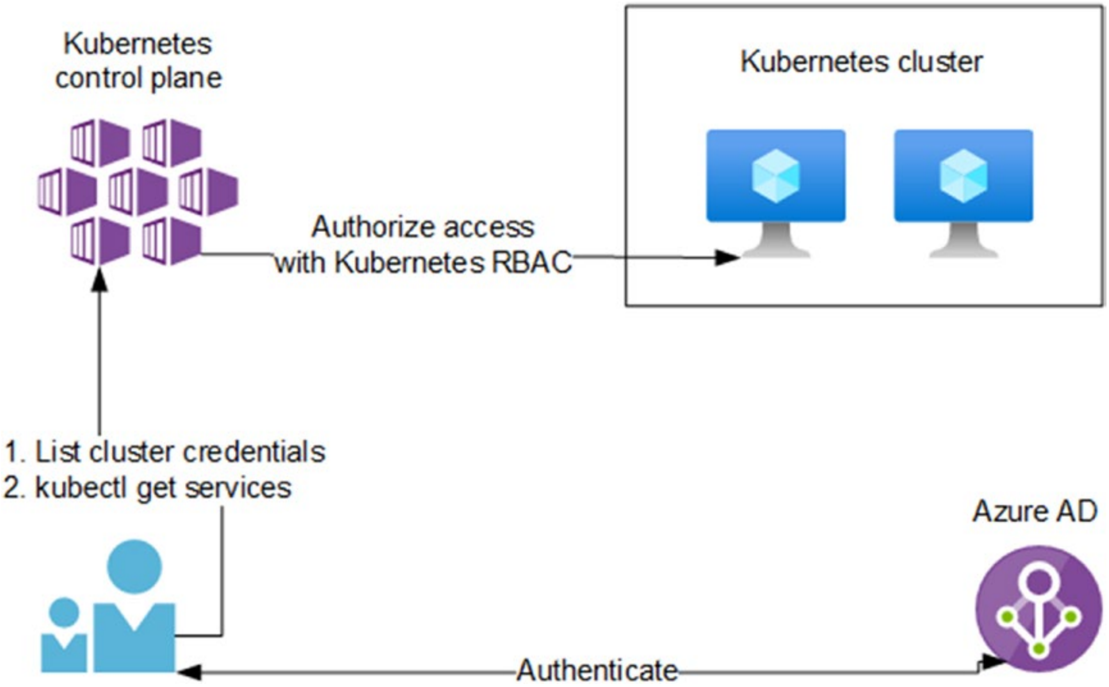


Figure 7-3. *Azure Kubernetes Service access control*

You can use the kubectl tool to authenticate the Azure Kubernetes Service clusters. To authenticate through headless scenarios, such as from a continuous deployment pipeline, you can use the kubelogin Kubernetes credential plugin.

Azure RBAC

Kubernetes control plane can perform operations such as creating, updating, or deleting services in your Kubernetes cluster. Access to the Kubernetes control plane is controlled using Azure role-based access control. The built-in Azure Kubernetes Service Cluster User role grants access to list the Kubernetes **cluster user credentials** and download the kubeconfig file. The built-in Azure Kubernetes Service Cluster Admin role grants access to list the Kubernetes **cluster admin credentials** and download the kubeconfig file. You should use the Azure Kubernetes Service Cluster User role to control access to the Kubernetes control plane.

Note The admin user bypasses Azure AD sign-in to the Kubernetes control plane. The Contributor role can use the listClusterAdminCredential action to get the admin user credentials!

Azure Active Directory Authentication and Kubernetes RBAC

After downloading the cluster user credentials using Azure RBAC roles, you can authenticate your users using Azure Active Directory and authorize them with Kubernetes RBAC.[3]

You can enable Azure Active Directory integration on Azure Kubernetes Service clusters with the --enable-aad option, either at the cluster creation or at cluster upgrade. You will also need to specify an Azure AD group that will have access to sign in to the cluster.

To further control access within the cluster, you can define Kubernetes RBAC Roles and assign them to users or groups using RoleBindings. The scope of a Kubernetes RBAC assignment is a namespace or the entire AKS cluster.

[3] https://kubernetes.io/docs/reference/access-authn-authz/rbac/

To prevent admin users bypassing the Azure Active Directory authentication, you disable local accounts by enforcing the `disableLocalAccounts` property. As of the writing of this book, the `disableLocalAccounts` property was in early preview.

Network

You can protect the administrative access to your Kubernetes control plane by adding network controls that prevent public access. You can also add multiple network controls to protect your applications running in Azure Kubernetes Service. Let's look at these in more detail.

Kubernetes Control Plane Network Controls

By default, your Kubernetes control plane and your Kubernetes API server and possible Kubernetes dashboard are publicly accessible. You can control network access to the Kubernetes control plane with authorized IP ranges. When you configure the `authorizedIPRanges` property, only requests made to the API server from IP address that you have explicitly listed are allowed.

To keep the traffic between the Kubernetes control plane and your cluster nodes in a private network, you can use the Private Cluster feature by enabling the `enablePrivateCluster` property. With Private Cluster, the control plane communicates with the cluster nodes through Azure Private Link. This prevents Kubernetes control plane for users outside the cluster virtual network, or without a jumpbox.

Application Network Controls

To meet your application business goals and enterprise security requirements, you can set multiple network controls in place. These allow you to

- Control ingress and egress traffic (north-south)
- Control traffic between your cluster namespaces, nodes, or pods (east-west)

To control ingress traffic, you can implement an ingress controller. An ingress controller can be a container hosting web application proxy logic, such as nginx, or it can stay outside your cluster altogether. In the former case, you can configure your application load balancing, authentication, and encryption in transit within the cluster.

In the latter case, you can use Azure Kubernetes Service's integration with Azure Application Gateway. With Application Gateway, you can configure encryption in transit and Web Application Firewall using platform-as-a-service components.

To control east-west traffic, that is, traffic within your cluster and between your microservices, you can implement a service mesh such as Istio, LinkerD, or Open Service Mesh. A service mesh can control traffic flows within your cluster. As a service mesh is typically implemented as a proxy, it can also encrypt your traffic in transit.

Logging

Azure Kubernetes Service provides security logs for management plane and Kubernetes control plane. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Administrative operations, such as deleting the Azure Container Instance resource
- Role-based access control changes (`roleAssignments/write`)
- Data-plane access control changes, such as someone performing the `Microsoft.ContainerService/managedClusters/listClusterAdminCredential/action` operation
- Network control operations, such as changes to `authorizedIPRanges`

Azure Kubernetes Service does not keep Kubernetes control-plane logs by default. To enable Kubernetes control-plane logging, you need to configure the Azure Kubernetes Service resource logs (under diagnostic settings) to be sent to your centralized log store. These logs include

- **kube-audit** category contains all audit log data for every audit event, including get, list, create, update, delete, patch, and post.
- **kube-audit-admin** category is a subset of the kube-audit log category, excluding the get and list audit events from the log.
- **guard** category contains Azure Active Directory authentication logs.

Additionally, you can use Azure Defender to analyze data-plane logs and create Security Center alerts. For Azure Kubernetes Service, these alerts include[4]

- Exposed Kubernetes dashboard detected
- Privileged container detected
- Suspicious request to Kubernetes API

Best Practices

Microsoft's Patterns and Practices team has published a reference implementation of a baseline Azure Kubernetes Service cluster,[5] which is a good collection of best practices.

Microsoft applies **daily patches** (including security patches) to Azure Kubernetes Service virtual machine hosts (nodes). Some security updates, such as kernel updates, require a node reboot to finalize the process. This reboot process does not happen automatically. A Linux node that requires a reboot creates a file named `/var/run/reboot-required`. You are responsible for scheduling the reboots as needed. You can use Kured[6] (KUbernetes REboot Daemon) by weaveworks for this.

You are responsible to keep your Kubernetes version updated and staying within the one-year support window.[7] To upgrade the **Kubernetes version** of your cluster, you need to perform Azure Kubernetes Service upgrade operation, which deploys a new node with the new Kubernetes version, cordons and drains your old node, schedules your Kubernetes pods in the new node, and finally deletes your old node.

In addition to manually upgrading your Azure Kubernetes Service cluster, you can configure auto-upgrade on your cluster. To control when the cluster upgrade operations are performed, you can also configure Planned Maintenance. Planned Maintenance allows you to limit all Azure Kubernetes Service maintenance operations (including cluster upgrades) to a specific weekly time.

As of the writing of this book, auto-upgrades and Planned Maintenance features were in early preview.

[4] `https://docs.microsoft.com/en-us/azure/security-center/alerts-reference#alerts-akscluster` and `https://docs.microsoft.com/en-us/azure/security-center/alerts-reference#alerts-containerhost`

[5] `https://github.com/mspnp/aks-secure-baseline`

[6] `https://github.com/weaveworks/kured`

[7] `https://kubernetes.io/blog/2020/08/31/kubernetes-1-19-feature-one-year-support/`

Summary

In this chapter, we looked at the controls available to you for protecting Azure container-as-a-service workloads.

As we have learned, their support for Azure Active Directory is consistent, but there are differences in supporting Azure-native network controls. We have also established that the use of container-as-a-service workload introduced the need for governance and architecture. Specifically, container registries and Kubernetes in-cluster networking controls require careful planning and cross-team collaboration.

CHAPTER 8

Workload Protection – IaaS

In this chapter, we discuss protecting infrastructure-as-a-service workloads in Azure. After reading this chapter, you will be able to select and implement appropriate controls for securing infrastructure-as-a-service workloads as part of your organization's security policy framework.

While you can certainly replicate your on-premises datacenters into Azure, the cloud's distributed and software-defined nature means that your experience will be different, no matter your level of execution. To mitigate the differences in performance and cost, you will need to understand what you are paying for when deploying Azure services. For example, an Azure virtual machine is more than just physical hardware, but includes highly available deployment, host updates, storage, and networking capabilities. To benefit from your existing expertise, processes, and licensing, I recommend you apply those in a centrally managed infrastructure-as-a-service environment.

This chapter focuses on securing infrastructure-as-a-service capabilities in self-service manner, where application development teams can deploy and configure their own virtual machines and other services, while relying minimally on a centralized team for configuration and management. This approach is attuned to the cloud-native approach of shifting security responsibilities left.

If you are planning to build a centrally managed shared service, emulating the capabilities and service level of that of on-premises, you will still benefit from the structure of this chapter. You might solve some of the topics differently from my recommendations, however.

Specifically, your network topology, choice of firewall technologies, and update management operations might vary.

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3_8

Access Control

Administrative access to Azure virtual machines is controlled using Azure role-based access control, virtual machine extensions, and the various options for user access for logging in to the virtual machine resources.

Azure Role-Based Access Control

Management plane access to Azure virtual machines is controlled using role-based access control. The **Contributor** and **Virtual Machine Contributor** roles grant access to create and manage virtual machines, disks, snapshots, and extensions.

The Virtual Machine Contributor role is slightly more limited than Contributor, limiting access on virtual network and storage resources. You should use Virtual Machine Contributor when you want to grant your application teams access to create virtual machines on their own and let them use existing network resources. If you want to let them only configure the virtual machine but not create a new one, you should grant the role in the virtual machine resource scope.

Note The Contributor and Virtual Machine Contributor grant the user access to reset or create local administrator passwords by using the VMAccess virtual machine extension!

For operational access in shared environments, you should assign the access according to the least privilege principle. To assign access to forensic investigators, use the **Disk Snapshot Contributor role**, which lets the user manage virtual machine snapshots, but does not grant access to modify, restart, or delete the virtual machine resource itself. To assign access to perform backup and restore operations, you should use the **Disk Snapshot Contributor, Disk Restore Operator, Site Recovery Contributor**, and **Site Recovery Operator roles**.

Automation Access

You can use several tools to automate your Azure virtual machine guest operating system configuration management. If you build your own virtual machine images, you can include any configuration management tools you would like to. If you use images from the Azure Marketplace, however, you will need to take the virtual machine onboarding into account when planning your automation.

Whether you are using an agent-based automation tool, such as Chef or Puppet, or an agentless automation tool, such as Ansible, you can onboard your virtual machines using Azure virtual machine extensions. Some configuration management tools are provided as native extensions, but for others, you can use Azure Custom Script Extension to download and execute configuration scripts in the virtual machines.

If you perform the extension installation post-deployment, the user installing the extension will require Virtual Machine Contributor access to the virtual machine resource. You can also include the extension installation as part of your infrastructure-as-code deployment. In that case, your continuous deployment pipeline will have to have the same Azure RBAC access.

Virtual Machine Login Access

The data-plane access for virtual machines is controlled using virtual machine login access control.

Azure allows users to log in using a local administrator account by default. While the Azure Portal experience prevents you to use some of the most common usernames, such as admin or root, it will still expose you to brute-force and password-spray attacks.

The Azure-native way of managing login access to virtual machines is to use Azure Active Directory login. With Azure Active Directory login, authentication is performed using Azure Active Directory and authorization using Azure role-based access control. In practice, Azure Active Directory login is implemented as a virtual machine extension, which installs and manages the required packages on your Linux or Windows virtual machines.

Once you have configured your virtual machine to use Azure Active Directory login, you can control access to it by assigning Azure role-based access control roles. The following data-plane roles are applicable for virtual machine login access:

- **Virtual machine administrator login**: This role grants access with administrative privileges, including the ability to elevate privileges with the sudo command in Linux virtual machines.
- **Virtual machine user login**: This role grants access with user privileges.

When your users have one of the earlier-mentioned Azure role-based access control roles assigned, they can log in to the virtual machines using Remote Desktop Protocol (Windows). To log in to Linux virtual machines, they will need to use the `az ssh` extension of Azure Command-Line Interface. After a successful authentication, standard SSH clients can be used.

To support a broader set of existing workloads, you might want to use Azure Active Directory Domain Services (Azure AD DS) or traditional Active Directory Domain Services (AD DS) to control login access to your virtual machine. If you have extended your Active Directory to the cloud by hosting domain controllers in Azure, you can join your virtual machines to the domain and continue using your established login methods. This requires some considerable network planning, however, so this is mostly suitable for centrally managed infrastructure-as-a-service scenarios. For self-managed virtual machines, it is recommended to use Azure Active Directory login.

Network Controls

Network access to the virtual machine management ports (RDP and SSH) is unrestricted by default: virtual machines with public IP addresses can be accessed remotely by anyone. To protect your virtual machines from port scanning, brute-force logins, and other threats that arise from open Internet access, you can set multiple controls in place.

Self-Managed Virtual Machines

First, you should only use private IP addresses for your virtual machines. This recommendation applies regardless of whether you are securing self-managed virtual machines or virtual machines in centrally managed shared services.

If you are not able to use private IP addresses, you should use Azure Security Center just-in-time access (JIT) to reduce the public exposure of your virtual machine management ports. Once configured, Azure Security Center JIT enforces that inbound traffic is denied. To open management port access from their network location, users need to perform an activation in the Azure Portal, which authorizes the user, audits the access, and opens the management ports for a predetermined time to the IP addresses defined by the user requesting access.

Next, you will need to control access to the virtual network. In the case of self-managed virtual machines, you can do this by setting up a virtual private network (VPN) connection to the virtual network your virtual machine is deployed in. Figure 8-1 illustrates such a scenario. In this case, you will need to deploy and configure a VPN gateway resource in a dedicated subnet, which will need to have access to your virtual machine.

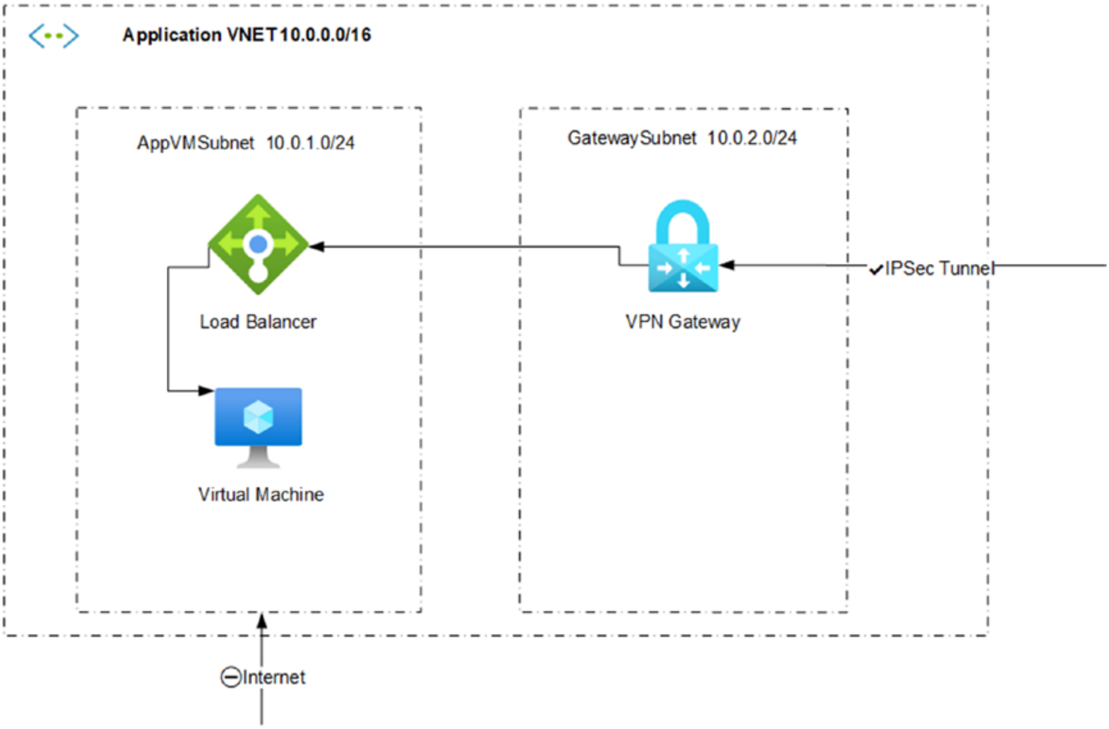


Figure 8-1. *Controlling network access for self-managed virtual machines*

Finally, you can implement adaptive network hardening[1] to let Azure Security Center automatically monitor and manage your network security group rules.

Azure point-to-site VPN supports most client operating systems. The point-to-site VPN access can be authenticated using Azure AD (Windows 10), RADIUS, or certificate authentication.

As an alternative to client-based point-to-site VPN, you can use Azure Bastion. Azure Bastion provides HTML5-based web client that is streamed to your administrative users through the Azure Portal. Azure Bastion is jumpbox service that is deployed into a separate subnet in your virtual network, just like the VPN gateway. Azure Bastion is managed and hardened by Microsoft, as a service.

[1] https://docs.microsoft.com/en-us/azure/security-center/security-center-adaptive-network-hardening#what-is-adaptive-network-hardening

Centrally Managed Virtual Machines

Controlling network access for centrally managed virtual machines is different from that of self-managed virtual machines. Figuvre 8-2 illustrates the network topology and network controls in a centrally managed environment. This approach requires further planning regarding IP address inventories, central firewall management, and Azure subscription governance. This approach is detailed further in the Microsoft Cloud Adoption Framework.[2]

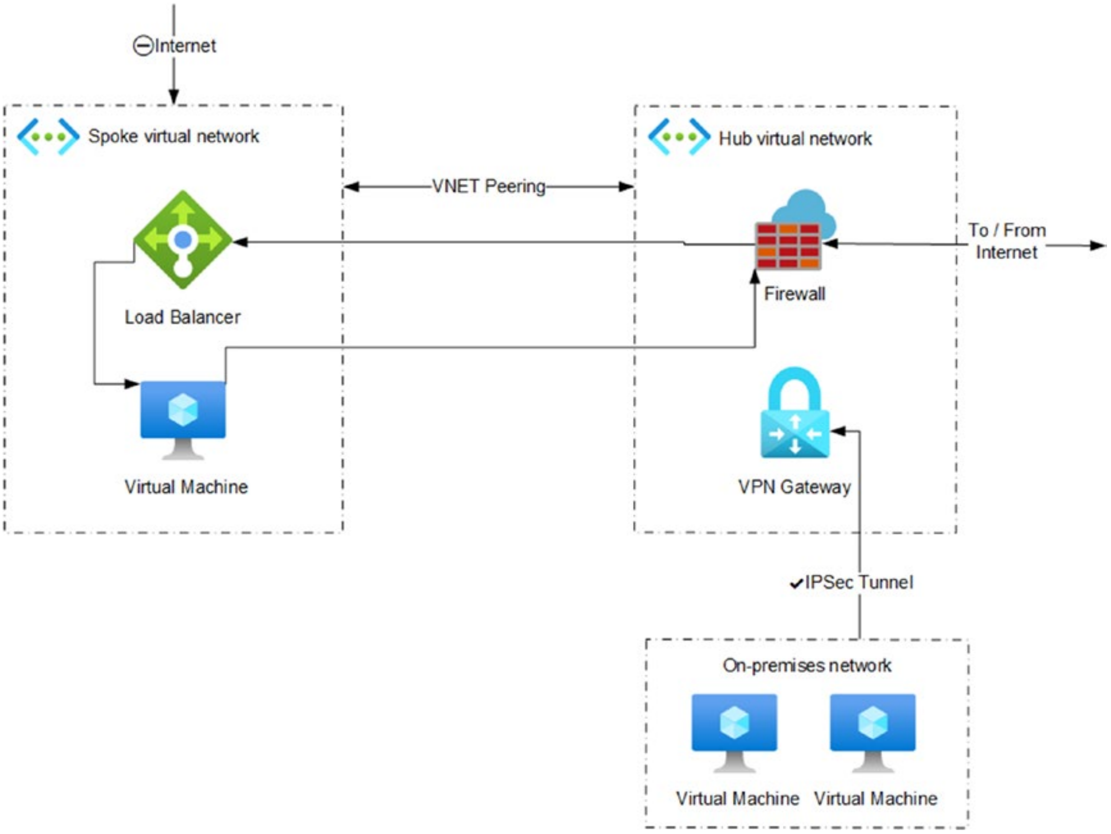


Figure 8-2. *Network controls for centrally managed virtual machines*

[2] https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/enterprise-scale/network-topology-and-connectivity

Web Application Access

Based on your cloud strategy and available application modernization investments, you might not be able to re-platform or re-architect your applications to be suitable for hosting in platform as a service or container as a service. Therefore, in addition to providing administrative access, application workloads hosted in your virtual machines might need to be accessible by its users directly.

Figure 8-3 illustrates how you can expose your web applications hosted in virtual machines to users, without exposing the virtual machine directly to the Internet via a public IP address.

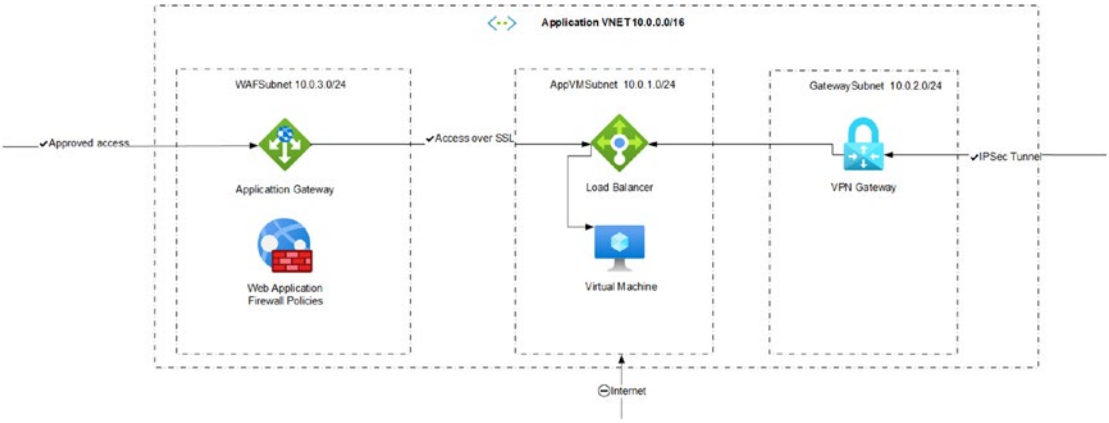


Figure 8-3. *Network controls for web applications hosted in Azure virtual machines*

Monitoring and Detection

Azure virtual machines provide security logs for management and data planes. Management-plane, or platform, logs are created within Azure activity log. These logs include

- Administrative operations, such as deleting, shutting down, or restarting the virtual machine
- Role-based access control changes (`roleAssignments/write`)
- Data-plane access control changes, such as changes to SSH keys or using the VMAccess extension to reset local administrator credentials

Azure virtual machines do not keep data-plane security logs by default. To enable data-plane security logging, you need to enable Azure Defender, provision the Azure Monitor agent on the virtual machine, and configure the Azure Security Center data collection level.

Azure Defender is enabled by configuring the Security Center Standard pricing tier for the virtual machines of your environment. This can be done in infrastructure as code or enforced using Azure Policy by defining the `Microsoft.Security/pricings` resource with the name `VirtualMachines` and setting the `pricingTier` property to `Standard`.

Next, you need to install the Azure Monitor agent (Log Analytics agent), which reads security-related configuration and event logs from the virtual machine and transmits them to your Log Analytics Workspace. Azure Security Center then uses these logs to provide you with security recommendations.

Finally, you need to select the security logs that are collected. The collected data includes operating system type and version, operating system logs (Windows event logs), running processes, machine name, IP addresses, and logged in users by default. The security category event logs are not collected by default. To enable their collection, you need to configure Azure Security Center data collection level. You can select from the following options:

- **Minimal** covers only events that might indicate a successful breach and important events that have a very low volume (such as failed login attempts).
- **Common** provides a full user audit trail, such as logins and sign-outs, Kerberos operations, and security group changes.

Vulnerability Management

Azure Defender includes a built-in vulnerability scanner by Qualys. The vulnerability scanner is deployed as an extension: `LinuxAgent.AzureSecurityCenter` or `WindowsAgent.AzureSecurityCenter`, respectively. Once the extension is installed, the Qualys scans the virtual machine and sends the results for analysis in the Qualys' cloud service hosted closed to your region (either in the United States or Europe). Subsequent scans are performed every four hours, but you may also trigger scans manually.[3]

[3] https://docs.microsoft.com/en-us/azure/security-center/deploy-vulnerability-assessment-vm#trigger-an-on-demand-scan

Qualys can identify affected machines within 48 hours of the disclosure of a critical vulnerability. Any findings will be available in the Azure Security Center as under the recommendation **Vulnerabilities in your virtual machines should be remediated**.

You may also use any other vulnerability management or endpoint detection and response solution, such as Microsoft Defender for Endpoint.

Endpoint Protection

Azure Defender includes a license to Microsoft Antimalware for Azure, which identifies viruses, spyware, and other malicious software. Microsoft Antimalware uses the same platform as Microsoft Security Essentials (MSE), Microsoft Forefront Endpoint Protection, and Microsoft System Center Endpoint Protection. It offers both real-time protection and scheduled scanning. Microsoft Antimalware for Azure is installed as a virtual machine extension and is only available for Windows.

You may also use any other endpoint protection solution. Major vendors that offer endpoint protection as virtual machine extensions include Trend Micro, Symantec, McAfee, and Sophos.

Azure Defender Alerts

Additionally, you can use Azure Defender to analyze data plane logs and create Security Center alerts. For virtual machines, these alerts include[4]

- A logon from a malicious IP has been detected.
- Event log or command history log was cleared.
- Detected the disabling of critical services.
- Possible credential dumping detected.
- Suspected Kerberos Golden Ticket attack parameters observed.
- Behavior similar to ransomware detected.

[4] https://docs.microsoft.com/en-us/azure/security-center/alerts-reference#alerts-windows

- Detected persistence attempted.
- Disabling of `auditd` logging
- Failed SSH brute-force attack.

Backup and Disaster Recovery

Azure provides four tiers of high availability for VM-based applications: single-instance deployments with SSD or HDD, availability sets, and availability zones. When you deploy a single instance of a virtual machine and use HDD Managed Disks, Microsoft guarantees the virtual machine connectivity with a 99.5% service-level agreement (SLA). If you deploy a single instance with solid-state disks, Microsoft offers SLA of 99.9%. If you deploy two or more instances, you have further options. Availability sets offer SLA of 99.95% while deploying your virtual machines to separate fault and update domains within the same Azure datacenter region. Availability zones provide SLA of 99.99%, while deploying your virtual machines to two or more zones in a datacenter region. Each zone consists of one or more datacenters equipped with independent power, cooling, and networking.

To protect the data and the state stored in your applications running in virtual machines, you can use Azure Backup and Azure Site Recovery. Both store their data in the Azure Recovery Services Vault, which can be configured for either locally redundant storage (LRS) or geo-redundant storage (GRS).

Azure Backup provides a native backup solution for Azure virtual machines. To enable Azure Backup, add it to your Azure Recovery Services Vault and configure the retention policy. When restoring data, you can choose to create a new virtual machine, replace the existing virtual machine, or perform a cross-region restoring the secondary (paired) Azure region.

Azure Site Recovery replicates workload's virtual machines from your primary region to a secondary region. When an outage occurs at your primary region, you can automate failover to the secondary location. After the primary region is running again, you can revert to it. Azure Site Recovery has SLA-guaranteed recovery time objective (RTO) of two hours. Azure Site Recovery integrates with other Azure infrastructure as a service, such as storage and networks. As it is application-aware, you can customize the failover and recovery of multi-tier applications running on multiple virtual machines by configuring Recovery Plans.

EXERCISE

You are responsible for architecting the resiliency of an infrastructure-as-a-service application. The application is deployed to three tiers: front end, back end, and data. Your business requirement is to design a solution that is available 99.95% of time. How will you design the solution?

Guest Operating System Management

When it comes to **operating** infrastructure as a service in the public cloud, you will face several impediments. In contrast to platform as a service, there are still a significant number of controls you are responsible for in an ongoing basis, and most often your application development teams do not have the capacity or expertise to perform them. So how do you balance between fast time to market of a self-service infrastructure as a service and the operational responsibilities that this comes with?

You may start by ensuring your application development teams are using the latest versions of operating system images that meet your requirements. You can approach this by standardizing on a set of approved images from the Azure Marketplace or by publishing your own images using the Azure Shared Image Gallery service.

Next, you can automate operating system patching in Azure to support both self-service and centrally managed infrastructure-as-a-service scenarios.

Finally, remember that you do not need to do the impossible. To deliver appropriate operational controls with an appropriate cost that also fits your enterprise security architecture is a complex topic. You may very well end up in a solution that promotes self-service infrastructure as a service for a limited number of scenarios and encourages your users to take advantage of your centrally managed infrastructure as a service, where operational responsibilities are taken care for them.

Operating System Image Management

To ensure a secure baseline for your operating system configuration, you can control the operating system images used in your virtual machines. The images offered by Microsoft are a good starting point as they are always up to date, but their security configurations are close to defaults. As such, their configuration needs to be hardened before operational usage. If you are providing a centrally managed infrastructure as a service, you may use these images as a source before applying your own configuration management and hardening. For example, you might use a Microsoft-provided image and control the hardening of the images using domain policies. As this depends on the virtual machine having network access to the domain controllers, it might not be suitable for all network topologies, however.

If you would like to control the original image, you can use Azure Shared Image Gallery to provide your application teams access to private images you have approved or built yourself (such as your golden image). Images in the Shared Image Gallery are considered sub-resources in Azure Resource Manager, so you can assign your application teams access to them using role-based access control. Once they have access to an image, they can create virtual machines in their subscriptions using that image.

You can also use images built and managed by other vendors. Specifically, the Center for Internet Security (CIS) provides hardened images through the Azure Marketplace. The CIS hardened images come with an hourly license fee that is billed in addition to the virtual machine infrastructure cost. The images are provided for both Level 1 and Level 2 CIS Benchmark profiles.[5] The images are configured using local group policy, so they are a good fit for a self-managed infrastructure-as-a-service scenario.

Note By using the CIS hardened images from the Azure Marketplace, you delegate the responsibility for both operating system hardening and vulnerability patching to CIS.

Self-Managed Patching

To keep your self-managed infrastructure as a service patched, you can automate patching of your virtual machine operating systems using the automatic VM guest patching feature. This automatically downloads and applies critical and security patches and reboots your virtual machines if required. Managing other updates is still left for the responsibility of the application development team.

[5] www.cisecurity.org/cis-hardened-images/microsoft/

The automatic patching is performed during off-peak hours of your virtual machine time zone. The patches will be installed within 30 days of release. Automatic VM guest patching is a good option for you, if your virtual machines are not automatically turned off during off-peak hours.

Note Not all workloads are suitable for automatic patching! Before going to production, validate your approach and follow best practices detailed in the Cloud Adoption Framework.[6]

You can also evaluate the usage of Hotpatching for Windows Server Azure Edition, which allows for automating the patching of security updates without virtual machine reboots.

As we went to press, automatic virtual machine guest OS patching and Hotpatching were still in preview.

Centrally Managed Patching

To centrally update your Azure virtual machines, you can certainly leverage your existing processes and tools, provided that all networking requirements are met. The most straightforward example of that would be if you would centrally manage a landing zone for virtual machines and give your application development teams access only to the data plane, that is, inside the virtual machine resources. In that case, you can continue to use System Center Updates Publisher or Windows Server Update Services (WSUS) as you would have before.

If your infrastructure-as-a-service workloads are not completely under central control, you might benefit from using Azure Update Management, a native service to manage operating system updates at scale. Azure Update Management can manage not only manage critical and security updates, like automatic VM guest patching, but any other updates, too.

Azure Update Management consumes data from Log Analytics agents running in your virtual machines and uses automation runbooks to compare the state of your

[6] https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/manage/azure-server-management/

virtual machines to available updates in private or public update sources. For Windows virtual machines, this means WSUS or Microsoft update. For Linux virtual machines, the options are private or public repositories. Windows virtual machines are scanned twice a day for available updates, while Linux virtual machines are scanned hourly.

When updates are available, you can deploy them to all or a subset of the virtual machines that you manage with Azure Update Management. As update management is using Azure Automation, you can also schedule update deployments programmatically, such as to install critical updates on Sundays.

Summary

In this chapter, we discussed the controls available to you for protecting Azure infrastructure-as-a-service workloads.

As we learned, your options vary based on how autonomous your application development teams are. If you are giving the application development teams a high degree of freedom in terms of network topology and deployment options, you essentially approach securing virtual machines as you would any other Azure resources, that is, by enforcing security controls on self-service resources. This gives your application delivery teams more flexibility and potentially reducing time to market. Your infrastructure-as-a-service environment will be more heterogeneous, however. Rather than attempting to build central policies that control your cloud virtual machines like you have previously, with this approach, you should focus on limiting any exposure to the application development team's own environment. For example, you might do well to isolate them from any internal networks (either on Azure or on-premises).

If you are taking more responsibilities and offering a centrally managed service, you will be able to build a more homogeneous infrastructure-as-a-service environment. You might even be able to reuse your existing process and tools, providing an indistinguishable virtual machine experience to your end users. As they might not even need to get access to the Azure Resource Manager or the Azure Portal, the fact that the resources are deployed to Azure might become a trivial implementation detail to them.

Index

K. Ots, *Azure Security Handbook*, https://doi.org/10.1007/978-1-4842-7292-3