The 8 Principles of Cloud Adoption

Microsoft Practice Development Playbook

Microsoft

aka.ms/practiceplaybooks

About this Playbook

The goal of this playbook is to help you accelerate or optimize your Azure-focused practice by teaching you fundamental principles you can apply for greater success in all your Microsoft Azure projects.

This playbook is the work of Jan Depping and Herman Keijzer. We wrote this to document our own experience helping Microsoft Partners to be successful with Azure.

When we started this project, we initially drew up a list of 19 principles that seemed to capture the spirit and best practices that made the Microsoft partners we were working with successful. We eventually merged these to the 8 cloud principles described in this playbook.

We hope that this information will be beneficial to other Microsoft partners using Azure as part of their offering.

ABOUT THE AUTHORS

Jan Depping works as a Partner Technology Strategist in the Netherlands. Jan has worked for many years in the service provider and public sector space. With a background in computer technology and psychology, he is always looking the optimal balance between people and product. His main focus is creating new business models with AI, Blockchain and Cloud Computing.

Herman Keijzer works as a Partner Development Manager on datacenter migration programs in western Europe. Before that he was a Partner Technology Strategist in Netherlands.

Herman has worked for most of his IT life in the service provider space and has many years of experience in offering services online. In the early days, these were IP services or ASP services; these evolved into online services, private cloud and now into public cloud services. His service provider point of view means he is always looking at how to run services both as efficiently as possible and with the highest level of quality.

Opsgility, a Microsoft Partner, provided additional technical writing services.

Using the playbook effectively

Quickly read through the playbook to familiarize yourself with the layout and content. Go over the content several times, if needed, then share with your team.

This playbook is organized as 8 principles. Each principle follows the same structure: an explanation, examples and guidance on how to apply the principle in practice, and further reading.

The examples and guidance are intended to illustrate the principle and provide a practical guide to implementing certain specific scenarios. They are not an exhaustive list of ways in which the principle can be applied. Once you understand the principle, you will find many other ways to apply it across your services.

This playbook assumes a reasonable level of knowledge of cloud computing and Microsoft Azure. In particular the reader should be familiar with Azure infrastructure services and have a basic knowledge of platform services.

TO GET THE MOST VALUE OUT OF THIS PLAYBOOK:

- ☑ Get your team together and discuss which pieces of the strategy each person is responsible for.
- ☑ Share the playbook with your technical and managed services teams.
- ☑ Leverage the resources available from Microsoft to help maximize your profitability.
- ☑ Share feedback on how we can improve this and other playbooks by emailing playbookfeedback@microsoft.com.

Table of Contents

July 2018

Introduction

This book is about how you can make the cloud work for you. This book will describe some guiding principles. Making use of these principles will help you to stay on track in your growth journey. They will enable quicker, better decisions that are aligned with your strategy and vision.

In our day to day work at Microsoft we work very closely with what Microsoft calls Managed Service Providers, or MSPs. We guide these partners in their journey of adopting Azure as one of their customer offerings, often transforming their existing hosting business in the process.

Running Azure workloads requires a different approach than managing workloads in your own datacenter. If you try to operate Azure applications using processes developed for on-premises hardware, you will not use the cloud to it potential and you will pay far too much for the cloud resources. You will also fail to take advantage of the opportunities for increased agility and faster, more efficient delivery that the cloud enables.

In this book we will guide through what we call the 8 principles of cloud computing. In our experience working with many Microsoft partners, these principles have been repeatedly observed to provide the foundations for successful cloud adoption.

Applying these principles is not a one-time task. They are a culture and mindset that drives all aspects of your cloud usage. You practice the principles during development and during operations. They become part of your agile DNA when practicing DevOps and CloudOps. Mastering these principles will become your IP that you can use in how you add value to your cloud offerings for your customers or how you consume cloud resources.

Living these principles will bring you many benefits, but do not treat them as fixed in scope or number. You should make the principles your own. Expand upon them based on your own experience. The goal is to develop a cloud mindset that drives every aspect of how your deliver services to your customers. We hope these principles will help to guide you towards that goal, enabling you to use the cloud to its true potential.

Principle 1: Build to Fail

The first principle is to embrace the potential failure of any component in a system. By doing so, you can deliver higher availability for the system as a whole.

In a traditional infrastructure, the principle that is often used is "built to last". IT systems are designed to run continuously and almost indefinitely. The availability of the system depends largely on the reliability of the underlying hardware. Hardware availability is achieved by duplicating many components, such as servers, power, network, and storage.

This approach does not adapt to cloud computing. In the cloud, you no longer have directly control over the hardware, power, or network. You trust your cloud provider to handle the infrastructure, providing you with a cloud platform to build and host your application.

"Build to Fail" is a mindset and an approach to building reliable applications in the cloud. With a "Build to Fail" mindset, you can recognize the potential for failure in every component of your application. Rather than trying to achieve the impossible and eliminate all possible failures, your approach should be to accept these failures as inevitable. By designing your application to be resilient to failure, you can deliver applications with extremely high levels of availability.

FAILURE IS INEVITABLE

Hardware will fail. Software has bugs. People will make mistakes. These failures can happen at any of several layers:

- **Utility level:** Outage of underlying infrastructure, such as power, cooling, network or hardware.
- **Service level:** Outage or degradation of any of the IaaS or PaaS services used to implement the application.
- **Software level:** Code bugs or configuration errors within the application itself.

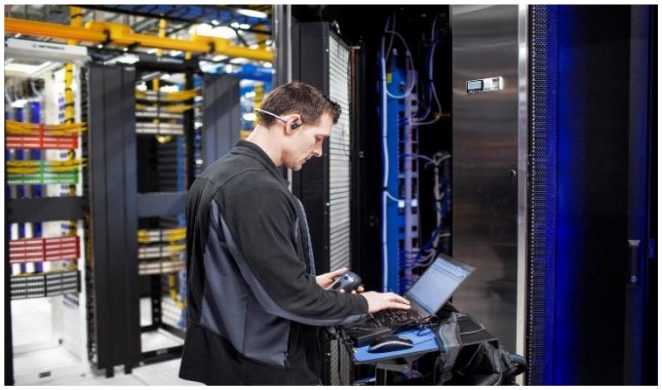
Software failures can occur either as a result of simple code bugs, or as a result of more fundamental mistakes in the design. Distributed systems in particular are complex and subject to a wide range of degraded behavior. The fallacies of distributed computing are a set of false assumptions which developers commonly make when designing distributed applications. These fallacies are:

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Each of these fallacies can lead to a potential failure. For example, an application that assumes infinite bandwidth may suffer performance degradation once actual usage exceeds the inevitably finite bandwidth available.

Failure is not limited to a sudden and complete outage, such as a power failure. Systems can also suffer degraded behavior, for example slow performance due to memory exhaustion or a spike in CPU utilization. These degraded experiences are also a type of failure.

You can never fully eliminate the possibility of failure within any component of your application. This is especially true in the cloud, where responsibility for the underlying infrastructure and services lies with the cloud provider, outside your immediate control. For each possible failure, you therefore need to treat the failure as an expected and inevitable event and embed resilience into your application design.



THE SEARCH FOR SPOF

A *Single Point of Failure* (SPOF) refers to any single point in the application, which if it fails, will cause a failure of the application itself. As we have seen, failures are inevitable, therefore each SPOF represents a risk to the application. Identifying and eliminating SPOFs is an essential step in implementing high-availability systems.

In traditional IT, eliminating SPOFs is achieved primarily through hardware redundancy, deploying redundant servers, power supplies, networks, and so forth. This is time-consuming, inflexible, and expensive—potentially *very* expensive, if separate facilities are required to prevent against data center-scale outages for business-critical applications.

In the cloud, you no longer have direct control over the hardware. Eliminating SPOFs is a shared responsibility between you and the cloud provider.

- **When using Infrastructure-as-a-Service**, Azure provides features such as availability sets and availability zones which enable your virtual machines to be distributed, thereby avoiding shared infrastructure which could act as a SPOF. You are responsible for understanding the available redundancy options and implementing the correct option for your application.
- **When using Platform-as-a-Service**, Azure implements redundancy to avoid SPOFs for you. You must still understand the levels of redundancy offered and make appropriate choices—for example, how many instances to deploy and whether they should be deployed across multiple availability zones.

You should also review your application architecture to identify SPOFs. For example, look for redundancy within the web content, application tier, database, and file store. Failure Mode Analysis (FMA) is a methodology for systematically identifying possible failure points and planning failure mitigations.

A popular approach to test and sharpen the software's ability to handle failure is the 'Chaos Monkey'. This approach was originally developed by Netflix, and has since been released as an open-source tool on Github and adopted by many other service providers. The idea of the Chaos Monkey is to randomly terminate services or instances to simulate failures at a much higher frequency than they naturally occur. This enables engineers to constantly test that their systems are resilient to failure, and that their automated responses are effective. For more information see the Principles of Chaos.

An important note: eliminating SPOFs does not guarantee your data is secure. For example, data may be corrupted by a code bug or virus regardless of the availability of your data platform. You should still back up your data regularly, and regularly test your processes for recovery from backup.

MICROSERVICES

A microservices architecture involves decoupling the application into small interconnected services, each of which performs a single function or role. These microservices are independently developed and deployed, and communicate with each other using standard protocols and well-defined interfaces.

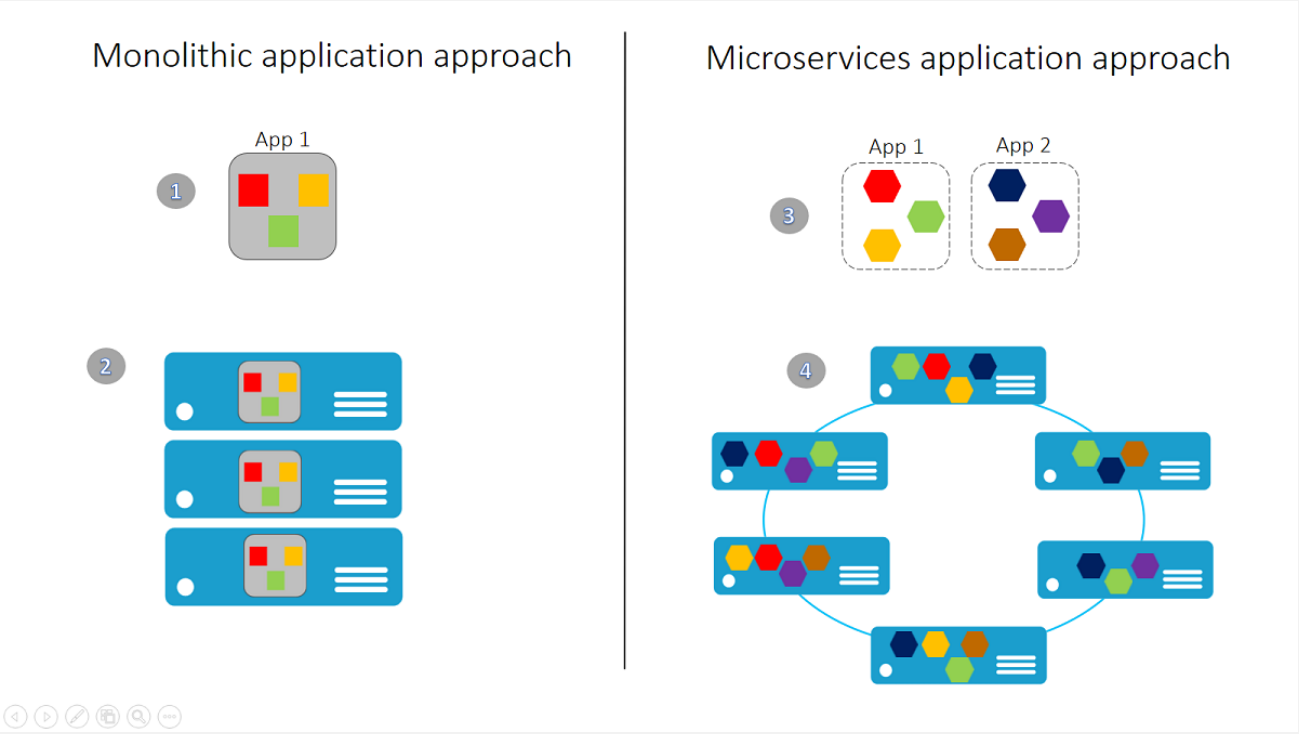
Microservices are deployed in redundant clusters (with more than one instance of each service). These clusters are spread across multiple VMs, decoupling the microservice deployment from the underlying infrastructure in such a way as to avoid single points of failure. The failure of any VM may impact particular microservice instances, but never an entire cluster for an individual microservice.

Azure supports several platforms for deploying and managing microservice applications, including Azure Service Fabric, Azure Kubernetes Service (AKS) and Web App for Containers. These services use either Service Fabric or containers as the platform for building microservices, and automate the process of orchestrating microservice instances as microservices are added, removed, updated, and scaled. In doing so, they provide a robust platform for delivering reliable applications that avoid SPOFs.

Microservice orchestration is an example of using software to automatically reduce exposure to SPOFs as deployments evolve over time. They also automatically detect and respond to failures, deploying new instances as required to maintain a redundant footprint for all microservices.

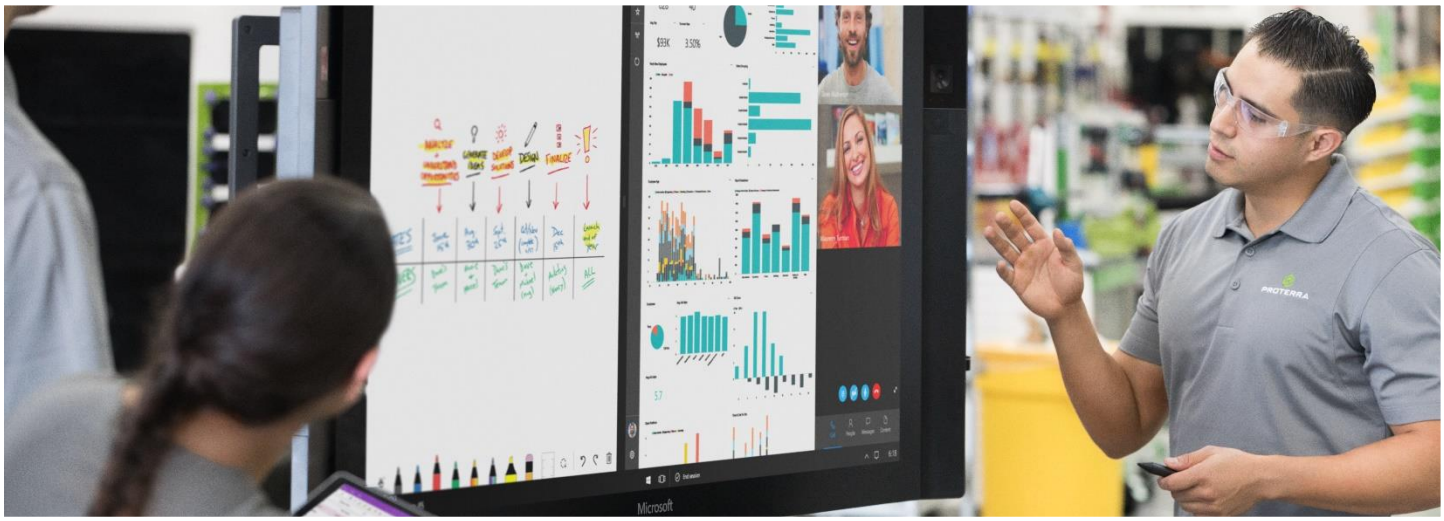
This automation is a critical component of the "Build to Fail" principle. Software should be 'smart' enough to recognize when it is not working properly and respond appropriately. For example, it can shut down a

UNDERSTANDING MICROSERVICES



1. A monolithic application contains domain-specific functionality and is normally divided into functional layers like web, business, and data.
2. You scale a monolithic application by cloning it on multiple servers/virtual machines/containers.
3. A microservice application separates functionality into separate smaller services.
4. The microservices approach scales out by deploying each service independently, creating instances of these services across servers/virtual machines/containers.

Source: https://docs.microsoft.com/azure/service-fabric/service-fabric-overview-microservices



malfunctioning or failed instance and deploy a new, clean instance to run the workload.

For more information on microservice architectures, see the microservices overview in the Azure Architecture Center.

DISASTER RECOVERY

When applying the "Build to Fail" principle for an Azure application, it is important to understand that the failure can happen in the application, but it can also happen in Azure. The Azure Engineering team employ a wide range of measures to prevent failures, but the possibility of failure cannot be eliminated entirely, and very occasionally a service you are using may experience disruption. Even more rarely, an entire region may be affected, for example in the event of a natural disaster.

Depending on the availability requirements of your application, you should plan for this. In addition to eliminating single points of failure, you should also plan how to recover from a region failure or other major outage. Resilience to routine failures is known as **high availability**; the ability to recovery from a large-scale disaster is known as **disaster recovery**.

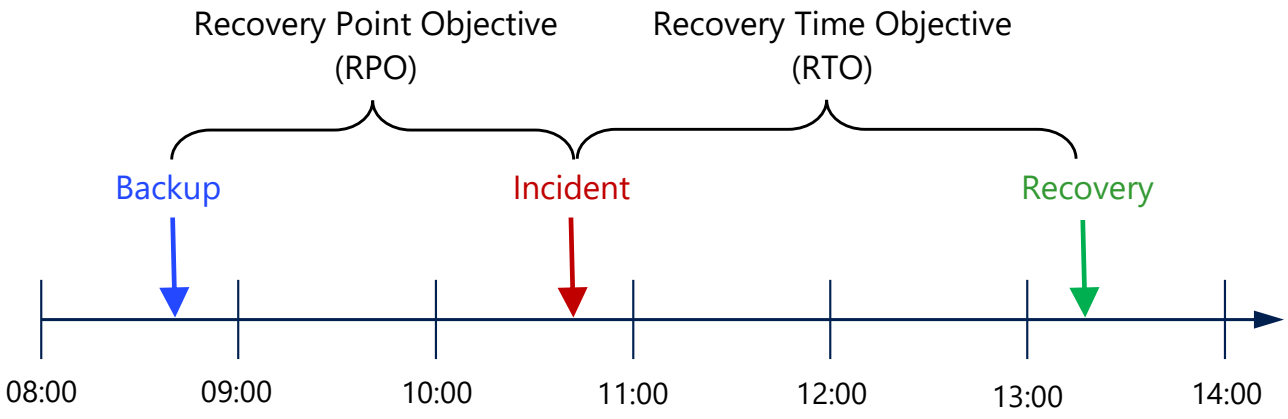
Your disaster recovery requirements are typically defined using two metrics, known as 'RTO' and 'RPO':

- **Recovery Time Objective (RTO)** is the time required after the disaster event for the service to be available once again. In other words, it's the maximum permitted outage duration.
- **Recovery Point Objective (RPO)** recognizes that a major disaster may result in data loss, even where backups are used. It defines the maximum permitted time window between the most recent backup and the disaster event. In other words, data older than the RPO should not be lost, but more recent data could be.

In many cases, a business will initially declare than only an RPO of zero (no data loss) is acceptable. In practice, this requires synchronous replication of data across regions, which can significantly impact application performance, as well as cost. In practice, a more nuanced and pragmatic approach may be required. Azure SQL Database and Cosmos DB support a variety of options for data replication, which should be explored.

The ability to achieve a given RPO and RTO will depend on many factors, such as backup policies, the distance between sites, the data volumes and data churn rates involved, the network bandwidth available, and the throughput of databases and disks.

Azure Backup and Azure Site Recovery are powerful tools to enable disaster recovery for infrastructure solutions; for platform services the recovery options associated with each service must be individually explored.



UNDERSTANDING SLAS

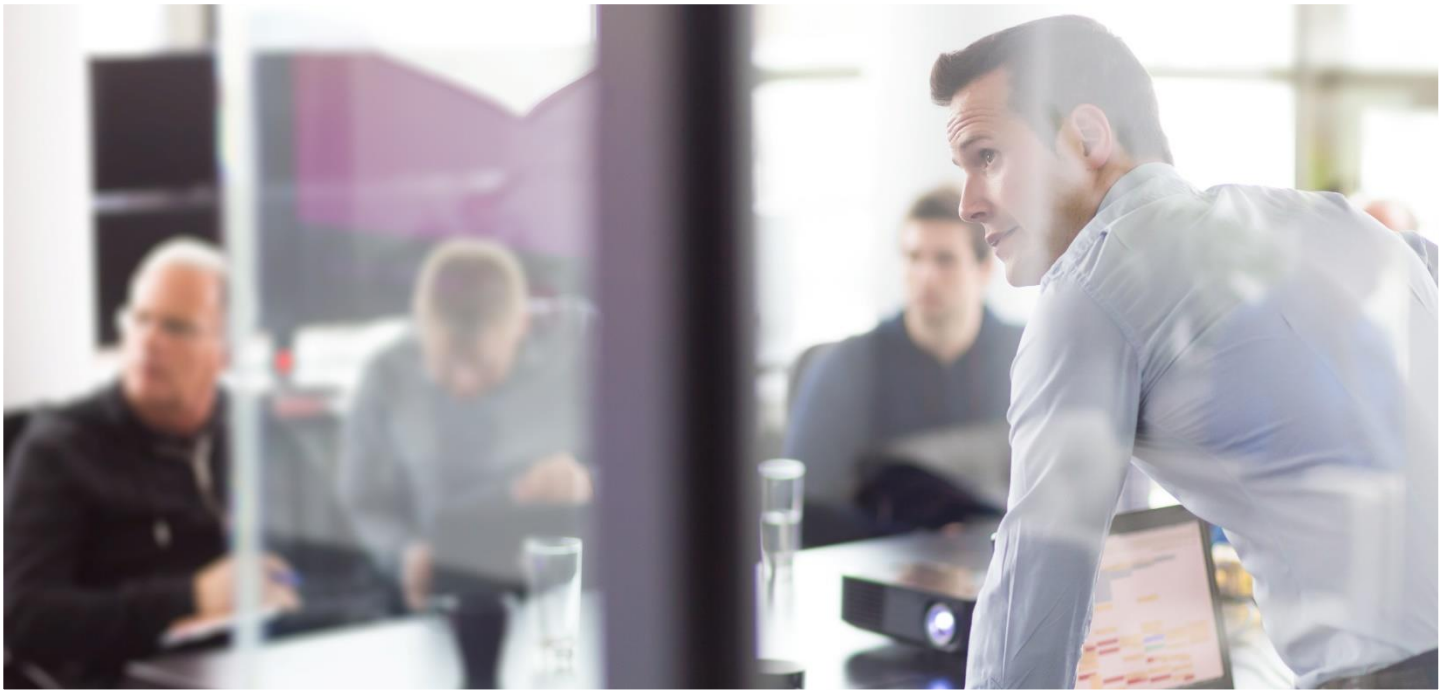
The resilience of your service to failure, and your ability to respond quickly when failures occur, has a direct impact on the Service Level Agreement (SLA) your service can offer.

SLAs are normally measured as the percentage availability of the service. To be meaningful, this must be measured over a fixed time interval, such as a week, a month, or a year. All Azure SLAs are measured monthly. The following table illustrates how much (or how little!) downtime is permitted based on the availability SLA.

For more information on Azure and SLAs, see:

- Azure SLA home page
- Azure Virtual Machines SLA
- Design to Survive Failures (Building Real-World Cloud Apps with Azure)

Availability %	Downtime per Year	Downtime per Month	Downtime per Week
90% ("one nine")	36.5 days	72 hours	16.8 hours
99% ("two nines")	3.65 days	7.2 hours	1.68 hours
99.9% ("three nines")	8.76 hours	43.2 minutes	10.1 minutes
99.99% ("four nines")	52.56 minutes	4.32 minutes	1.01 minutes
99.999% ("five nines")	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% ("six nines")	31.5 seconds	2.59 seconds	0.605 seconds



Using the Principle

In this section, we will briefly review some of the features and services available in Azure to implement the "Build to Fail" principle. Remember, the goal is to rely on software and automation rather than requiring manual intervention to respond to failures.

In the following two sections, we'll first look at the Azure features to support Built to Fail for Azure Infrastructure applications. We'll then discuss Build to Fail for cloud-native applications.

INFRASTRUCTURE RESILIENCY

Azure Resiliency describes the various options for virtual machine availability. These range from single VMs through availability sets and availability zones to deploying multiple application instances in paired regions.

Single VM

A 99.9% availability SLA applies to all Azure VMs using Premium storage, even individual VMs. As single VMs, they are still subject to SPOFs in the hardware, power supply, network or other supporting services. This SLA reflects Microsoft's confidence in the reliability of the Azure infrastructure and in the automated fault detection and recovery mechanisms (see 'Maintenance' below).

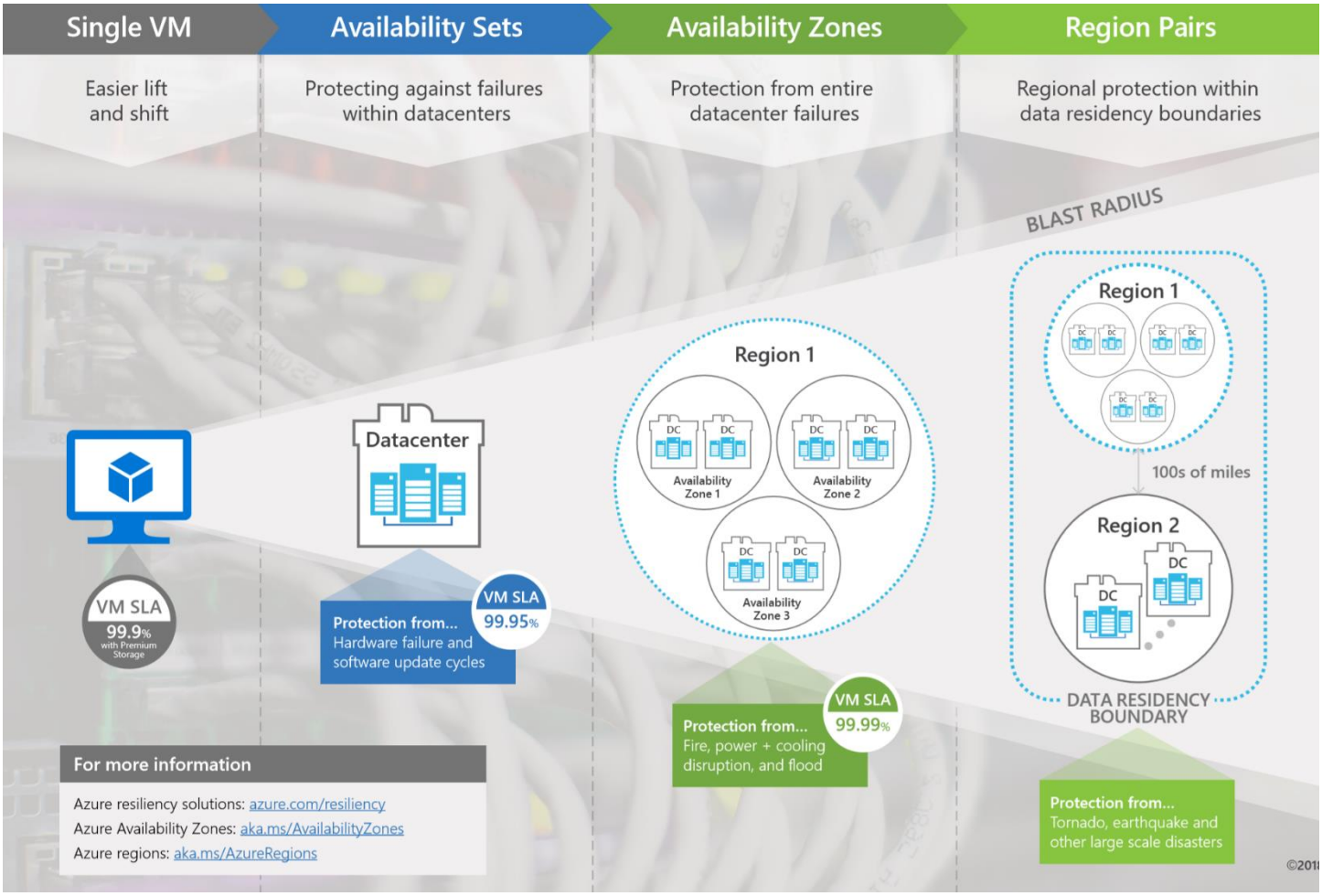
Availability Sets

Suppose you have two VMs sharing the same workload—for example, hosting a database. To avoid a SPOF, you would want these VMs to be deployed independently in the Azure data center, so they are not sharing the same physical host, network switch, or power supply. In addition, you would want an assurance that any updates to patch the VMs or their hosts will be rolled out separately, so both VMs are not impacted at the same time.

This is achieved by placing the VMs into an Availability Set. Each availability set defines a number of 'fault domains', which are physically separate locations without shared points of failure. VMs are spread evenly across fault domains, ensuring that any routine hardware failure within the datacenter can only impact a subset of the VMs, allowing the remainder to continue to deliver the application without interruption.

Availability sets also implement 'update domains'. These divide the VMs into separate groups so that platform updates are rolled out one group at a time. This prevents platform updates from causing application downtime.

When using availability sets, your VMs should be placed into the backend pool of an Azure load-balancer or



Application Gateway. These can be used to distribute the incoming traffic across the available VMs only.

Availability sets protect your application against routine failures within a datacenter, such as local power supply failures, network switch failures, or physical host failures. This is achieved by eliminating these components as SPOFs, and results in an availability SLA of 99.95%.

Availability Zones

As we have seen, availability sets protect your application against SPOFs within a datacenter. However, the application is still vulnerable to datacenter-wide outages such as a total power or cooling failure or flood. To defend against those risks, use availability zones. Availability zones is a high-availability offering that protects your applications and data from datacenter-wide failures. Availability zones are unique physical locations within an Azure region. Each zone is made up of one or more datacenters, and is equipped with independent power, cooling, and networking. To ensure sufficient redundancy, there's a minimum of three separate zones in all enabled regions.

While physically separate, availability zones within a region are sufficiently close to allow seamless networking and synchronous writes to storage across zones. Using availability zones therefore requires only incremental changes to your application and deployment architecture.

By spreading your VMs across availability zones, you can protect your application against large-scale failures impacting an entire datacenter. As well as protecting your VMs and disks, a wide range of other Azure services also support availability zones, such as Azure SQL Database, Service Bus, and VPN Gateway.

Availability zones are now available in 10 largest Azure regions, with more on the way. Where available, they offer an industry-best 99.99% VM uptime SLA.

For more information, see Availability Zones Overview.

	Single VM	Availability Set	Availability Zone	Paired Region
Scope of failure protected against	None	Server/rack	Datacenter	Regional disaster
Request routing	IP address	Load balancer	Zone-redundant load balancer	Traffic Manager
Networking	VNet	VNet	VNet	Global VNet peering
Data synchronization	n/a	Synchronous	Synchronous	Asynchronous*
SLA	99.9%	99.95%	99.99%	Per region

*CosmosDB supports cross-region synchronous replication as an option

Paired regions

Availability zones protect your applications against datacenter-scale outages by distributing your VMs and other services across multiple datacenters in an Azure region. They remain vulnerable only to very large-scale disasters, such as a major storm or earthquake, that would impact all the datacenters in a single region. To defend against even this eventuality, the application should be deployed to an Azure region pair.

Paired regions are selected pairings of Azure regions within an Azure geographic region (except Brazil South). They provide physical separation (at least 300 miles where possible) to protect against natural disasters impacting both regions. Other measures are also taken to prevent coordinated outages across regions pairs, such as regional isolation of services and sequential deployment of updates. Azure storage supports geo-replication of data between region pairs, and they are recommended locations when enabling geo-replication for other data stores such as Azure SQL Database.

Due to the physical distance between sites, data replication between regions is typically asynchronous, and networking is implemented independently. Traffic is distributed between endpoints at the DNS level, using Azure Traffic Manager.

In practice there are two ways to implement a paired region deployment:

- An active-active or active-passive architecture, where the application is simultaneously deployed to both regions
- A failover architecture, where the application is deployed to one region with the capacity to fail over to the paired region in the event of an outage, using a service such as Azure Site Recovery.

This choice is a trade-off between the cost and complexity of maintaining two parallel deployments and synchronizing data between them, versus the longer recovery time for a failover solution in the event of an outage. The appropriate choice will depend on the budget and the RTO requirements of the application.

System Maintenance and VM Reboots

There are several types of planned and unplanned maintenance events that can impact the availability of your virtual machines:

- **Planned maintenance without a reboot:** Microsoft performs periodic updates to maintain and improve the reliability, security and performance of the Azure platform. Most of these updates are rolled out using VM preserving maintenance, which simply pauses your VMs in-place, usually for less than 10 seconds and sometimes for up to 30 seconds. In some cases, your VM may be live-migrated to a new host, which again results in a short pause in VM operation, typically lasting no more than 5 seconds. Live migration is also used for pre-emptive maintenance, where Azure predicts a physical server may be about to fail and pro-actively migrates VMs to a new host.
- **Planned maintenance requiring a reboot:** Occasionally, planned Azure maintenance requires VMs to be rebooted. You will be given 30 days' advance notice, during which time you can choose when to apply the maintenance to your VMs. After this time the maintenance will be rolled out and your VMs rebooted. Azure provides APIs and notifications to help you manage planned maintenance, both from outside your application and from within the VMs themselves.
- **Unplanned downtime with service healing:** Where a physical server or other hardware fails unexpectedly, Azure will automatically detect the failure and re-provision the VM on a different server in an unaffected rack within the data center. The VM will reboot, and data on the temp disk will be lost. Data on the OS disk and data disks will be preserved.

For more information on events that can trigger a VM reboot, see Understand a system reboot for Azure VM.

Summary

The following table gives a comparison of the key features of each of the resiliency options. For more information, see Manage virtual machine availability.

CLOUD-NATIVE APPLICATION RESILIENCY

When designing cloud-native applications, you will first need to understand the resiliency options and features of each service you consume. You will then need to understand how these combine to deliver resiliency for the application as a whole.

There are many proven architectural best practices and design patterns which you can apply to improve the resiliency of your cloud-native application. A short and by no means complete summary of some common techniques is given below:

- **Throttling.** By throttling incoming requests, a service can protect itself from unexpected spikes in traffic and potential denial-of-service attacks which can otherwise have cascading effects on other application components.
- **Retries and the Circuit Breaker pattern:** Enable the application to handle transient failures by implementing retry logic. Use exponential back-off to prevent overloading the service being called and consider using the circuit-breaker pattern which cuts all calls to a service that is failing for a limited time to enable it more freedom to recover.
- **Prefer stateless services.** Where a service or component is stateless, it can be easily redeployed or scaled to handle failures or changes in load, for example using Azure App Service or VM Scale Sets with auto-scale. By removing local state and storing all state in a purpose-built service such as Azure Storage, Azure SQL Database, or CosmosDB, you can greatly simply and improve the resiliency of your application. By gathering state in a handful of purpose-built storage service, you can better leverage the resilience of those services.

  For microservice applications, note that Azure Service Fabric can be used to run both stateless and stateful services with high availability. Similarly, SQL Server can be deployed in containers orchestrated using the Azure Kubernetes Service (AKS) to provide a high-availability storage solution for microservices.

- **Use asynchronous signaling.** Consider a shopping cart application that sends a confirmation email when an order is placed. If the check out process sends the email synchronously, then the checkout will fail if the mail server used to send the email is unavailable, resulting in a lost order. Instead, the confirmation email should simply be queued as an asynchronous task to be completed later. Once the queue entry is written, the shopping cart application can complete the order. A separate process can read the queue and send the email later. In the event of a failed mail server, the emails can remain on the queue for later processing once the mail server is fixed.

  This is an example of asynchronous signaling. By decoupling tasks, failures in one part of the application do not block other parts of the application from working correctly. Decoupling also promotes code reuse, agility and ease of maintenance. A variety of Azure services are available to support asynchronous messaging between application components, such as Storage queues, and Service Bus. Queues can also help smooth out spikes in demand to a more steady-state load.

Further Reading

The Azure Architecture Center contains a wealth of detailed information to help you create resilient application designs. This has been written by the Azure Customer Advisory Team based on their real-world experience with many of Azure's most challenging customers. There is information for both infrastructure and cloud-native architectures. Start with the overview Designing reliable Azure applications, and from there dive into the more detailed materials. There is information on every aspect of building reliable systems, including defining requirements, architecture, testing, deployment, monitoring, disaster response and recovery. Be sure to review your design against the resiliency checklist for Azure services.

Principle 2: Self-Service

Self-Service is the principle that you can manage almost every aspect of your Azure resources yourself, without having to raise tickets or for others to respond.

A common cloud misconception is that because the service is operated by the cloud provider, you have much less control over the resources and infrastructure that deliver your applications and services. The fear is that the cloud is a 'black box' under the provider's control, and you will not have the visibility or control you need to pro-actively maintain the service and respond to incidents.

The reality is that Microsoft Azure provides many features that give you insight into the health and behavior of your resources and enable you to configure and manage those resources for yourself. It is certainly true that cloud providers such as Microsoft Azure take responsibility for their physical infrastructure, providing you with a set of virtual resources and services that run on top. However, those services still offer extensive monitoring and management capabilities. Once you master those capabilities, you should find you still have deep insight into your application health and that Azure provides all the control levers you need.

There are two parts to the Self-Service principle. The first is learning how to take full advantage of the monitoring and management features provided by Azure, so you can quickly and effectively manage your resources. The second is designing your application and supporting processes to promote self-service, so each member of each team can work efficiently and effectively, completing their tasks directly rather than having to raise tickets and wait for others to respond.

The Self-Service principle applies at several levels. It applies to individual resources, including resource diagnostics, deployment, and configuration. It also applies at the application or service level, for example using Azure Monitor Application Insights and Network Watcher for end-to-end application diagnostics, performance measurement and monitoring. This applies to both IaaS- and PaaS-based applications.

When applying the Self-Service principle in your own applications, a key aspect is ease and speed of use. Not only should the necessary controls be supported, they should be quick and easy to use. This requires investment in the right management tools to support your application. These may range from simple scripts to full-fledged management APIs and portals.

Common Operations team scenarios that should be fully supported include build and deployment, hot-fixing, scaling (up or down), configuring and patching virtual machines, configuring and adjusting monitoring and dashboards, and managing security settings to protect the service.

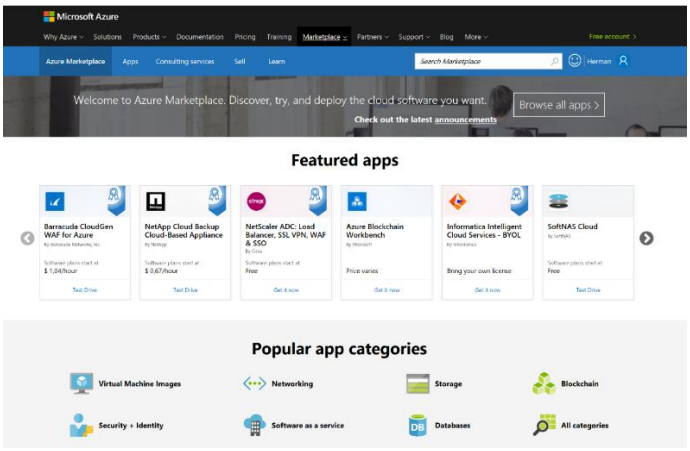
For Support teams, it is important to be able to investigate all customer settings. Subject to appropriate controls, the ability to work 'on behalf of' the customer to remedy issues quickly can be very useful.

Using the principle

Azure is not a 'black box', it is an open and flexible hosting platform. Take advantage of the wide variety of management features to gain full control over your resources.

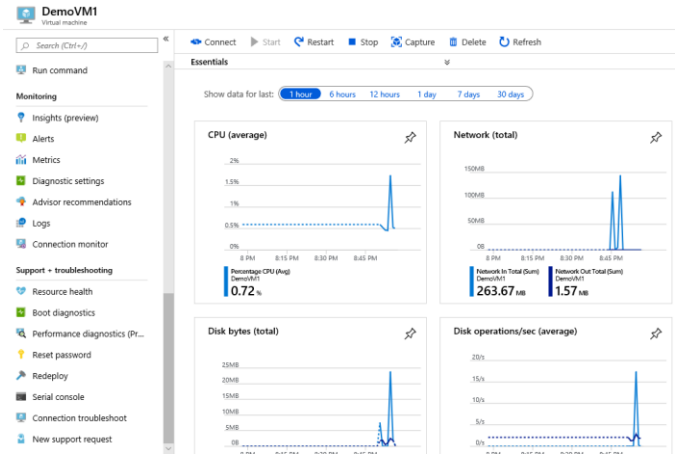
AZURE MARKETPLACE

The Azure Marketplace offers a wide range of third-party software and services. You can embed these in your application to accelerate your time-to-market and gain rich functionality. Each service deploys automatically into your subscription and enables you to manage the software for yourself.



VIRTUAL MACHINE SELF-SERVICE

Each service in Azure contains a range of features to help you manage that service. These features typically include configuration settings, monitoring metrics, and diagnostic logs. In some cases, additional service-specific management features are included. For example, in the screenshot below you can see how Azure Virtual Machines support a comprehensive range of monitoring, support and troubleshooting features:



We'll now take a closer look at just one of these features – the Azure Virtual Machine Serial Console.

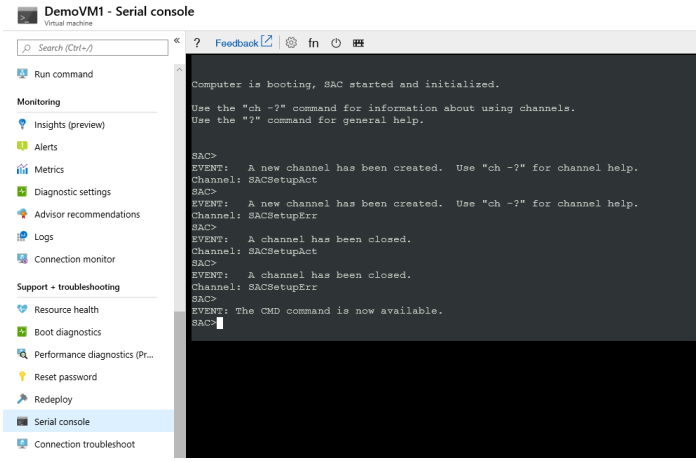
Serial Console

One of the differences between the cloud when compared to running your own virtualized environment using VMware or Hyper-V is that the cloud does not grant you full access to the underlying host hardware and hypervisor. Under normal circumstances, this is not a problem since your focus is on deploying and managing your virtual machines. However, occasionally a problem arises that blocks your remote access to the virtual machine, such as an OS misconfiguration causing a crash at boot time. Lower-level access to the VM is required to diagnose and resolve such issues.

The Azure Serial Console access is a very useful feature to address this scenario. It provides a text-based console interface to either Linux or Windows virtual machines,

providing low-level command-line access independent of networking configuration or operating system state.

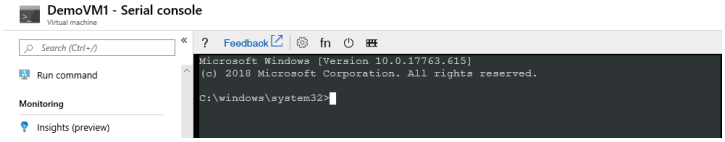
Serial Console is only available via the Azure Portal and requires Virtual Machine Contributor RBAC permissions or higher. It also requires that virtual machine boot diagnostics are enabled, which is used to log all access. For high-security environments where serial console access is not permitted, it can be disabled at subscription level for all VMs in that subscription.



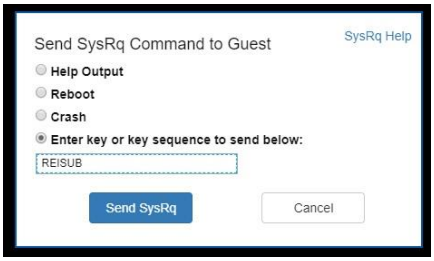
Common scenarios that can be resolved using the Serial Console include:

- Incorrect firewall rules or other network misconfiguration
- Filesystem corruption
- Remote access issues
- Interaction with the boot loader

On Windows, the Serial Console uses the Microsoft Windows Server Emergency Management Services command set. You can run a cmd.exe session in the Serial Console window, giving access to the more familiar Windows command line.



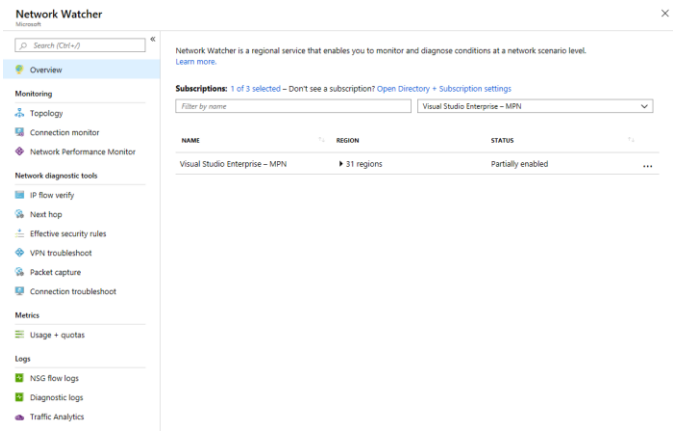
On Linux, you can access the GRand Unified Bootloader (GRUB mode) by restarting your VM with the Serial Console window open. This can also be used to access single user mode. The Serial Console for Linux can also be used to issue SysRq commands and Non-Maskable Interrupts (NMI).



For more information, see Azure Serial Console for Windows and Azure Serial Console for Linux.

NETWORK WATCHER

Another powerful collection of self-service tools is the Azure Network Watcher, which is a hub providing a central point of access for a wide range of network monitoring and diagnostics features.



Network troubleshooting can be time-consuming and difficult, requiring access to low-level diagnostics such as packet captures. Network watcher provides both powerful low-level tools to enable in-depth investigations as well as intelligent diagnostic tools to help speed up your investigations.

Key features of Network Watcher include:

- Network connection and performance monitoring, for both Azure and external networks
- Network topology mapping and next-hop diagnostics
- Connectivity diagnostics and monitoring
- VPN diagnostics
- Packet captures
- Diagnostic log configuration and analytics

For further information, see What is Azure Network Watcher?

Further Reading

In addition to the built-in management features, it can be useful to build custom scripts to integrate Azure tasks into your existing management tools. A common example is billing, where it can be useful to extract Azure usage information to feed into a billing system. As an example, this blog post by Dan Maas shows how to query the Azure rate card API using the Azure CLI.

Principle 3: Freedom of Choice

Choose the cloud hosting technology based on the full range available. Avoid tunnel-vision and sticking only with what you already know.

Azure offers a multiple technology platforms on which to build and host your applications. These range from IaaS VMs, to Azure App Service, to microservice and container orchestration services such as Azure Kubernetes Service and Azure Service Fabric, to fully serverless technologies such as Azure Functions and Azure Logic Apps. There is also a wide range of data storage options, from Storage Accounts to Azure SQL Database to CosmosDB. And this is only a sample to illustrate the range available!

Within each of these, there are again a range of options. There are over 100 different virtual machine SKUs to choose from, of various sizes across a range of VM families such as general purpose, compute optimized, memory optimized, storage optimized, and specialist hardware such as graphics cards. Within Azure App Service there is a range of service tiers, and a range of instances sizes within each tier. And so on.

Faced with all this choice, it is easy to become overwhelmed. A common pattern is for people to move towards their comfort zones, making technology choices based on their past experience and familiarity rather than the pros and cons of each option available and the technical and business requirements of the application.

The 'Freedom of Choice' principle means making the best technology choices from the full range available, based on the technical needs of your application and your business constraints. One such choice is the technology platform for the application, and the appropriate service SKU for that platform. Another is the choice of data storage to use.

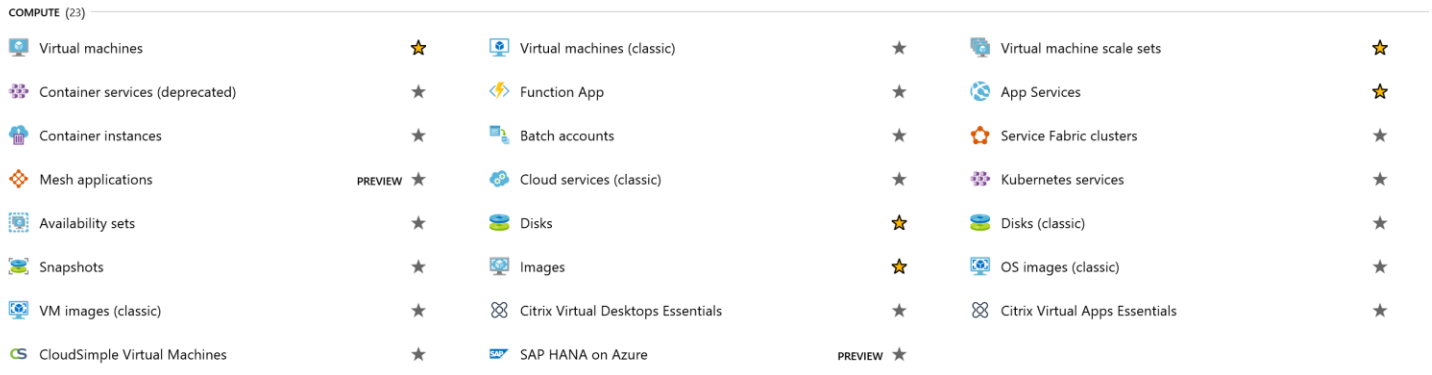
Freedom of Choice also means being aware of existing services that can save you 're-inventing the wheel'. Services such as Stream Analytics or Azure Bastion address specific scenarios in a highly targeted and efficient PaaS solution. It is usually preferable to consume such services rather than build your own equivalent service.

AZURE MARKETPLACE

Freedom of Choice includes the freedom to choose existing solutions over building your own. The Azure Marketplace provides a wide range of third-party offerings which can address many common scenarios. Why build an email service, when you can simply use the SendGrid service from the Marketplace? Even for simple scenarios such as SQL Server installed on a VM, why invest in automating the installation when you could use a pre-installed VM image from the Marketplace?

OPEN SOURCE

Freedom of Choice also means not being tied to Microsoft technologies. Azure supports a wide range of Linux distributions for both virtual machines and other platforms such as App Service. You can choose from a broad range of development languages, including C#, Java, Python, Ruby, PHP, Node, and more. And you can use a wide range of developer tools for source control,



Microsoft

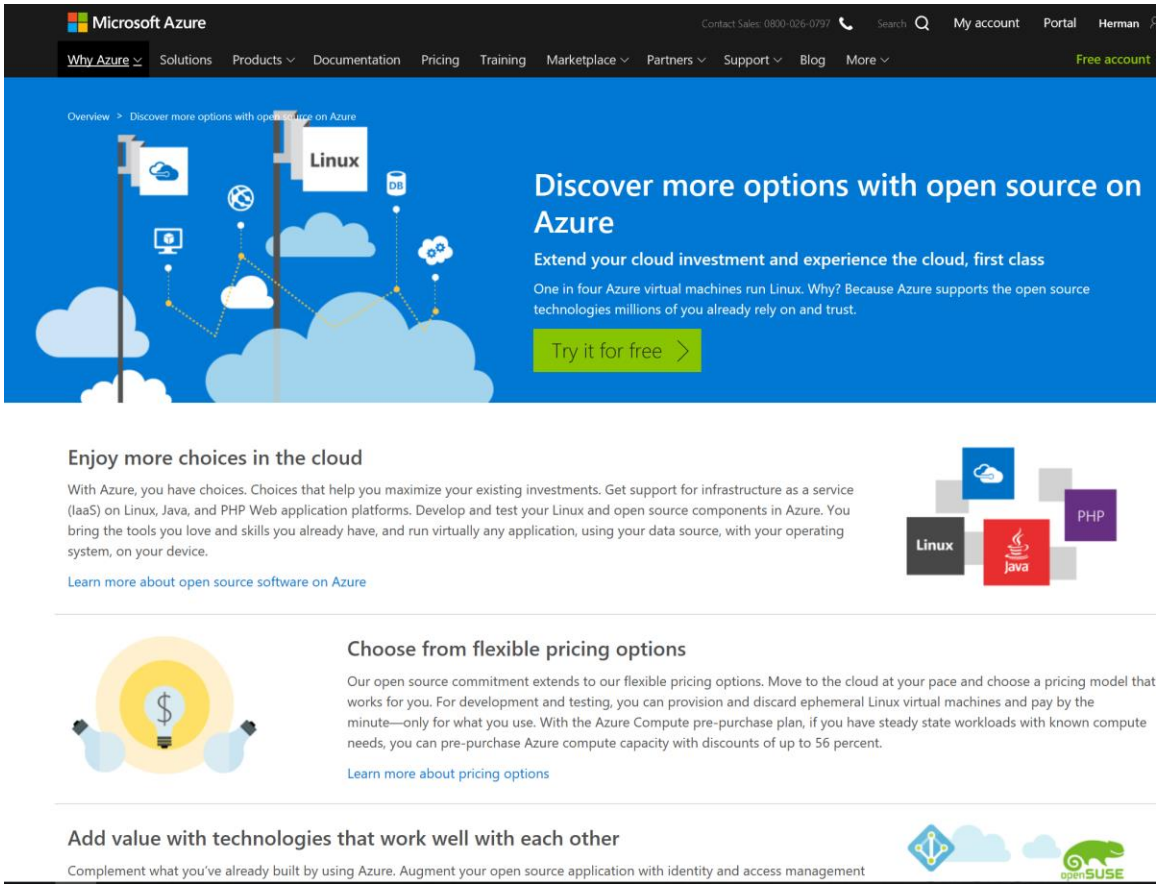
testing, build and deployment, such as Chef, Puppet, Ansible, Jenkins, and Terraform.

LOCK IN

A common concern raised by those new to cloud adoption is vendor lock-in. They are worried that migrating to the cloud will tie them to a cloud provider, reducing their future choices and leaving them exposed to pricing changes. Putting aside the reality that historically cloud prices have trended downwards rather than up, the ability to switch providers is sometimes cited as desirable.

Your flexibility to switch providers, and the work required to execute a switch, will depend on your technology platform choices. Virtual machines will run similarly on any cloud. Third-party deployment tools such as Terraform support multiple cloud providers, making it easier to re-use existing automation. Containers are by design extremely portable, and can easily be moved across cloud providers. In contrast, an application written to use technologies that are specific to a particular cloud (such as Azure Functions, for example) may require more effort to move.

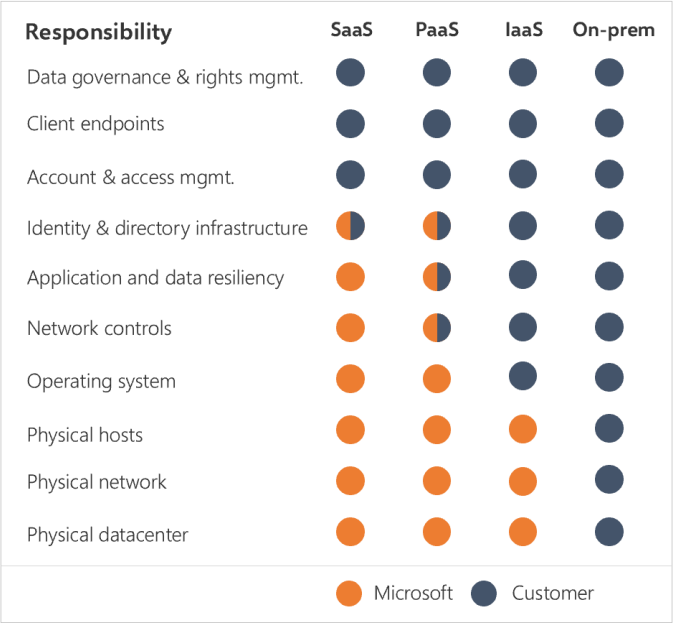
Moving data between providers can be more difficult, since it can require application downtime. Be sure to take into account the data volume and available bandwidth to calculate the time required to migrate all data. Services such as the Azure Data Migration Service can help, by providing incremental transfers while applications are still running, allowing you to keep downtime to a minimum.



Using the Principle

Take advantage of Microsoft guidance to help you choose the appropriate compute and data platforms for your application.

The most fundamental choices when choosing cloud technologies is whether to adopt an IaaS, PaaS, or SaaS approach, and whether the application will be cloud-only or a hybrid between on-premises and cloud. Each approach provides different advantages and different responsibilities, and typically requires a trade-off between many factors such as time, engineering effort, feature set, on-going management costs, backward compatibility, and so on. The following diagram summarizes the responsibilities of each approach.



COMPUTE

Even within IaaS and PaaS approaches, there are multiple technologies available. The Azure Architecture Center includes helpful guidance on choosing a compute service, including a detailed comparison of compute services. The decision tree below provides a useful guide to some of the questions you should ask and the range of compute options available. Remember, this will only guide your technology platform selection, there will still be other choices to make within that platform.

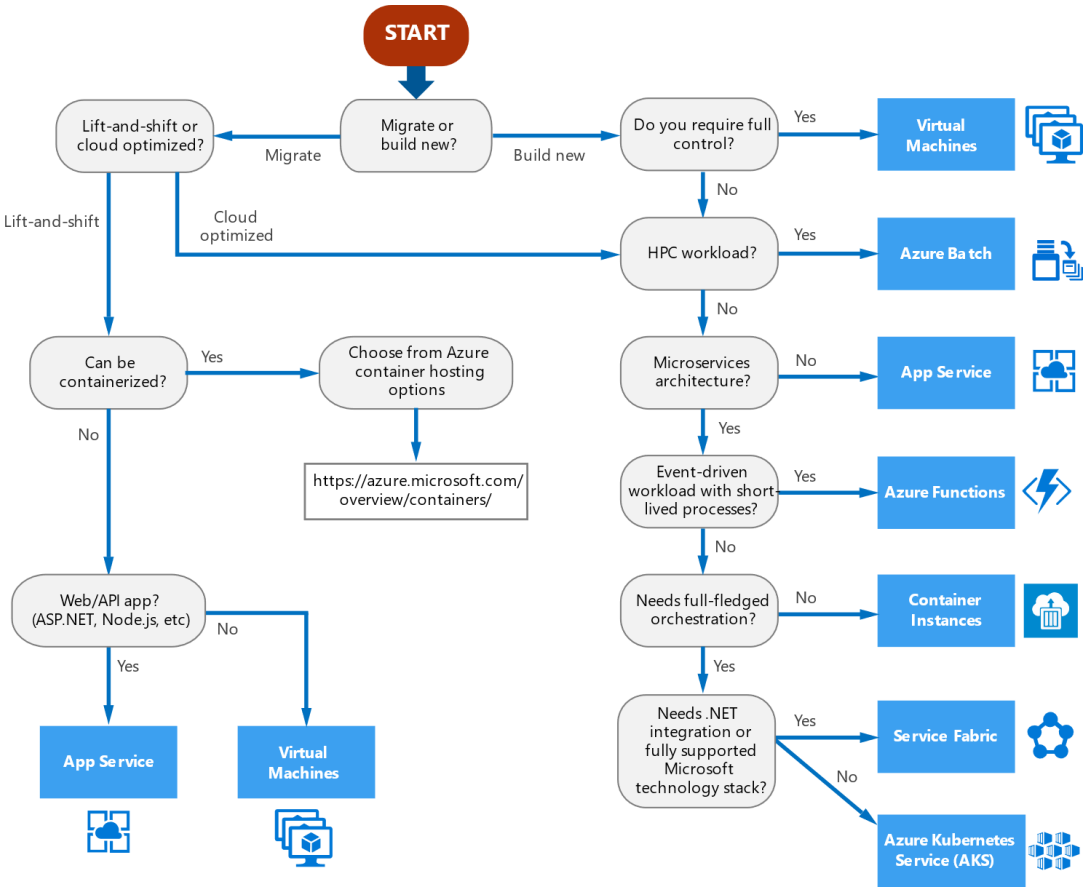
Suppose you have chosen an Infrastructure-as-a-Service approach using Azure VMs. You still face the challenge of choosing which VM family and instance size to adopt. Firstly, don't treat this like an on-premises hardware investment. You can easily change VM SKU later if you find your initial choice is sub-optimal. If you are migrating an existing workload, take advantage of Azure Migrate to assess your existing environment and make recommendations based on your existing machines and their utilization.

DATA

The Architecture Center also includes guidance on choosing a data store, including a data store comparison. Don't assume that your application must be tied to a single data store. Increasingly, cloud-native applications adopt a *polyglot storage* approach, using multiple storage technologies for different purposes.

For example, consider an e-commerce application. The data used by the system is stored across multiple storage types, each chosen for the data it stores best:

- **File Storage** is used to store product images and videos that are displayed to users.
- A **Relational Database** is used to store Product Orders where transactions and high consistency are needed.
- A **Document Database** is used to store Product Catalog information as well as Community Posts, allowing better scaling for a high volume of reads and writes.
- A **Key/Value** store is used by the Website to provide a much faster mechanism to reading and writing to store User Session and Shopping Cart information that is accessible across all instances of the website.
- A **Graph store** is used to store a Customer and Product social graph used for making product recommendations.



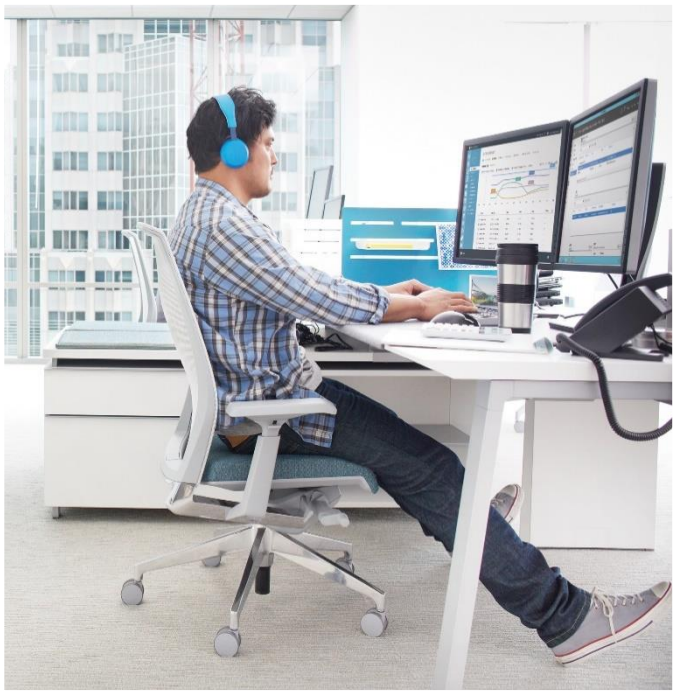
Source: https://docs.microsoft.com/azure/architecture/guide/technology-choices/compute-decision-tree

Further Reading

The Azure Architecture Center includes in-depth guidance on designing Azure applications, including how to choose between the various technologies available, for compute and storage.

The Architecture Center also includes guidance on microservices architecture, including how to choose between microservice compute options. The Azure documentation includes additional guidance on choosing between the various container services in Azure.

Choosing the right technology goes hand-in-hand with choosing the right application architecture. The Architecture Center also includes architecture patterns and reference architectures for common scenarios.



Principle 4: Trusted

Every human creation is developed twice: first in the mind and then in the physical world. The most important need for the shift from thinking to doing is trust.

This chapter is about trust, and the role it plays in enabling cloud adoption. We will explore how Microsoft ensures Azure is a trusted cloud platform and show how it enables you to build trustworthy applications.

Using technology to create smarter processes that help people to achieve more relies on trust. We need, as individuals, to have trust in technology, processes and other people in order to drive positive change.

Security, compliance and privacy are important prerequisites for cloud adoption, and we will explore how Azure supports each of these topics in depth. But trust is also something more. Trust is a feeling. Without it we stumble into our survival mechanisms: Fight, Flee or Freeze. In the case of technology adoption, lack of trust drives negative behaviors to avoid change, such as finding objections rather than finding solutions.

Trust cannot be taken for granted. To earn trust, we need to consistency demonstrate trustworthy behavior. These behaviors include competence, reliability, transparency and honesty. Trust can also be supported the testimony of others and an established reputation. But if you fail to maintain trust, it is hard to get it back, and loss of trust can be devastating for an organization.

Trust is not just about technology. It is also about people and processes. By building on a compliant platform such as Microsoft Azure and leveraging the controls Azure provides you, you can more easily build compliant solution for your applications.

SHARED RESPONSIBILITY MODEL

Cloud providers make huge investments to deliver secure and compliant cloud platforms. However, with the wrong design and implementation, you can still use a secure cloud platform to deploy insecure applications that do not meet compliance requirements, and which do not adhere to data privacy regulations.

Delivering trusted applications is therefore a shared responsibility between you and your cloud provider. Some areas are the responsibility of the cloud provider, such as the physical security of the datacenter or the reliability of the network. Other areas are your responsibility, such as ensuring your application maintains data privacy by not sharing personal data with unauthorized parties. In other areas, such as patch management, the responsibility varies: for IaaS deployments, VM patching is your responsibility, whereas for PaaS and SaaS services it is the responsibility of the cloud provider.

You are responsible for understanding which tasks belong to you and which are taken care of by the cloud provider. This will depend on the technology choices in your application design. You must then ensure that the appropriate tools, processes and training is in place to ensure your responsibilities are met.

SECURITY

Microsoft provides a multi-layered approach to security, from the physical datacenters and other infrastructure through a wide range of technical protections and security features to defend against a wide range of potential threats. Security is baked into the platform at every level. These features are supported by a team of more than 3,500 global cybersecurity experts working to protect your applications data in Azure.

Azure also provides a wide range of security features and controls which you can use to build secure applications. These cover a wide range of threats, allowing you to protect your network, servers, data, and user identities. Azure Security Center reviews your usage to recommend security improvements, as well as providing centralized visibility into your overall security posture.

While Azure Security Center started with an infrastructure focus, it has now expanded to cover a number of platform services. For example, it now supports building secure IoT solutions.

Microsoft's global scale provides another important security advantage. Microsoft uses machine learning to analyze vast sources of data including 18 billion Bing web

pages, 400 billion emails, 1 billion Windows device updates and 450 billion monthly authentications. The resulting insights enable a unique threat intelligence capability that protects Azure services and your applications.

COMPLIANCE

With over 90+ compliance offerings, Microsoft has the most comprehensive compliance coverage of any cloud service provider. These include global, country-specific, and industry-specific certifications. But what does this really mean?

As with security, compliance follows a shared responsibility model. Just because your application is hosted in Azure does not mean your application automatically inherits the compliance certifications that Azure has been awarded. Azure compliance—like any other cloud provider—is a solid foundation you can build upon. You still have overall responsibility for the compliance certification for your application.

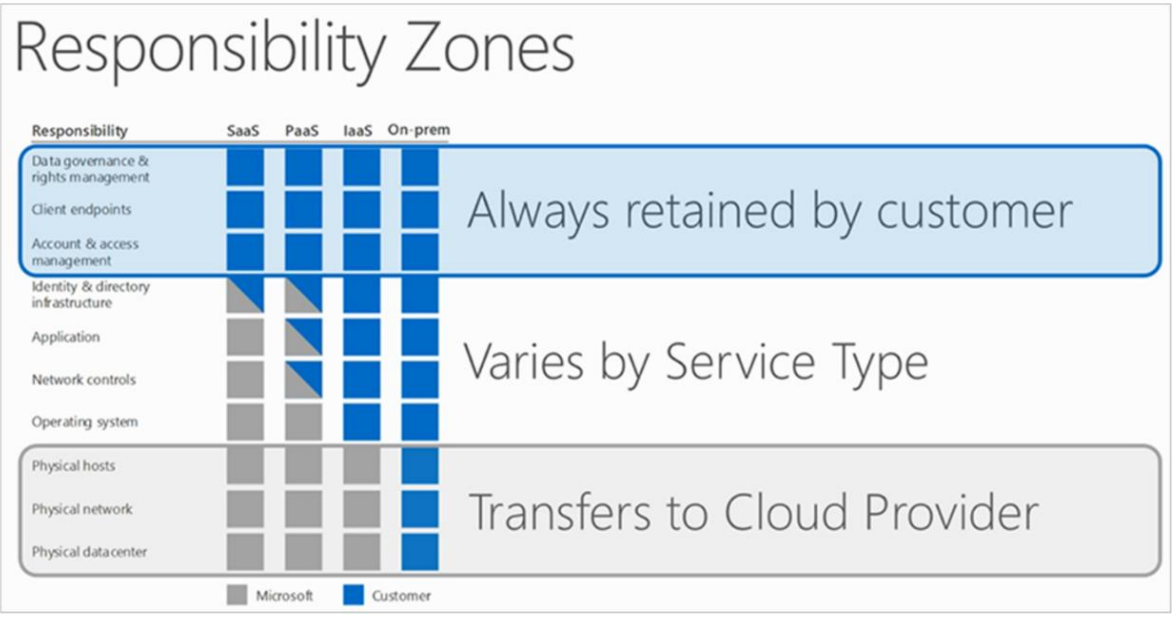
Whatever certifications your application needs to deliver, with such a broad range of compliance certifications, the chances are very high that Azure can provide you with a compliant solution.

PRIVACY

The core privacy principle for all Azure services is that you own your data. You have flexibility and choice in how and where you store, process and protect your data in Azure. Microsoft will never use your data for marketing or advertising purposes. And if you leave Azure, Microsoft has strict standards and processes for removing data from its systems.

Azure also complies with many external privacy standards, laws, and regulations, including: GDPR, ISO 27001, ISO 27018, European Union Model Clauses, HIPAA, HITRUST, FERPA, Japan My Number Act, Canada PIPEDA, Spain LOPD, and Argentina PDPA.

With GDPR top-of-mind for many organizations, it's important to understand how your relationship as a Microsoft customer relates to the GDPR roles of data controller, processor, and sub-processor. Where you are the data controller and store data in the Microsoft Cloud, then Microsoft takes the role of processor. Where you are the data processor, then Microsoft is a sub-processor. In each case, Microsoft accepts the responsibilities associated with these roles, as described in the Microsoft Online Service Terms. This clear approach enables you to have confidence in your ability to meet your GDPR obligations while storing personal data in the Microsoft cloud.



Azure Compliance Certifications

This table shows the current list of Azure compliance certifications. Click on the links in the table for further details.

Global	Government	Industry		Regional	
CIS Benchmark	CJIS	23 NYCRR Part 500	HDS (France)	BIR 2012 (Netherlands)	IT Grundschutz Workbook (DE)
CSA Cloud Control Matrix	CNSSI 1253	AFM + DNB (Netherlands)	HIPAA/HITECH	C5 (Germany)	LOPD (Spain)
CSA-STAR-Attestation	DFARS	APRA (Australia)	HITRUST	CCSL/IRAP (Australia)	MeitY (India)
CSA-Star-Certification	DoD DISA L2, L3, L5	AMF and ACPR (France)	KNF (Poland)	CS Mark (Gold) (Japan)	MTCS (Singapore)
CSA STAR Self-Assessment	DoE 10 CFR Part 810	CDSA	MARS-E	Cyber Essentials Plus (UK)	My Number (Japan)
ISO 20000-1:2011	EAR (US Export)	CFTC 1.31 (US)	MAS + ABS (Singapore)	Canadian Privacy Laws	NZ CC Framework (NZ)
ISO 22301	FedRAMP	DPP (UK)	MPAA	DJCP (China)	PASF (UK)
ISO 27001	FIPS 140-2	EBA (EU)	NBB + FSMA (Belgium)	EN 301 549 (EU)	PDPA (Argentina)
ISO 27017	IRS 1075	FACT (UK)	NEN-7510 (Netherlands)	ENS (Spain)	TRUCS (China)
ISO 27018	ITAR	FCA (UK)	NERC	ENISA IAF (EU)	
ISO-9001	NIST 800-171	FDA CFR Title 21 Part 11	NHS IG Toolkit (UK)	EU-Model-Clauses	IT Grundschutz Workbook (DE)
SOC 1	NIST Cybersecurity Framework (CSF)	FERPA	OSFI (Canada)	EU-U.S. Privacy Shield	LOPD (Spain)
SOC 2	Section 508 VPATS	FFIEC (US)	PCI DSS	GB 18030 (China)	
SOC 3		FINMA (Switzerland)	RBI + IRDAI (India)	GDPR (EU)	
WCAG 2.1		FINRA 4511	SEC 17a-4	G-Cloud (UK)	
		FISC (Japan)	Shared Assessments	IDW PS 951 (Germany)	
		FSA (Denmark)	SOX	ISMS (Korea)	
		GLBA	TISAX (Germany)		
		GxP			

Source: https://www.microsoft.com/trustcenter/compliance/complianceofferings

Using the Principle

Apply the 'Trusted' principle to build secure, compliant applications that meet your industry and regional compliance requirements and reduce your risk exposure.

A complete review of all security, compliance and privacy features in Azure would be far beyond the scope of this document. Instead, we will highlight some key features and provide a selection of references for further reading.

SECURITY

The Security Development Lifecycle

The Microsoft Security Development Lifecycle (SDL) was developed by Microsoft originally to help its own engineering teams develop secure applications. It covers the entire application development lifecycle, from design through development and testing to running services in production. The SDL uses in-depth reviews, best-practice engineering processes, and advanced code analysis tools to reduce security flaws at each stage of the cycle.

Microsoft first shared the SDL in 2008 to help its customers and others in the software industry to also build secure applications. It is constantly updated to reflect new threats, growing experience, and new scenarios such as the cloud, IoT and AI. Adopting some or all parts of the SDL is a proven way you can improve the security of your applications.

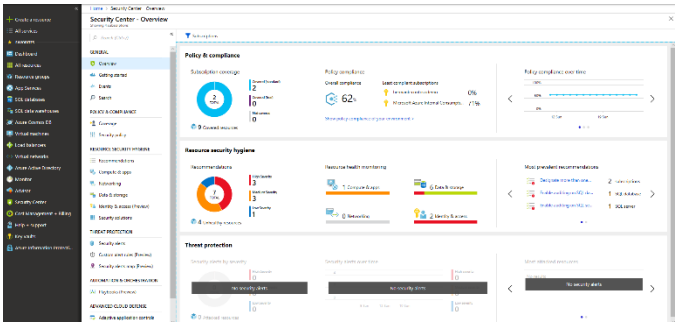
Security Center

Azure provides a huge range of services, each with a plethora of features. It can be hard to understand each service you are using in enough depth to be confident that your deployment follows all best practices. You've done your best, but how can you know your deployment is secure?

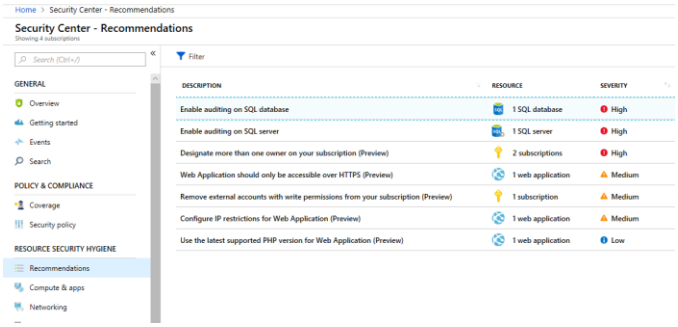
Microsoft's field, support, and engineering teams have extensive experience in helping customers to be successful in their Azure adoption. This includes securing applications against a wide range of threats. The challenge lies in scaling this expertise and making it accessible so that all customers can benefit.

Enter Azure Advisor and Azure Security Center. These services are designed to capture Microsoft's hard-won experience and make it accessible to all Azure customers.

They do this by analyzing your deployments against an extensive library of best practice rules and making recommendations on how you can improve.



These rules are written by the engineering team based on their in-depth knowledge of each Azure service, together with the best practices and mistakes they see when supporting customers. Each rule comes with a test, clear explanation, and specific recommendation of a change you should consider. Recommendations in Azure Advisor are grouped into cost saving, availability, performance, and security. These security recommendations are also exposed in Azure Security Center.



Security Center also provides a summary of compliance with Azure policy. This enables you to see at a glance whether your usage is continuing to meet the governance rules defined for your organization.

Security Center integrates with Management Groups to enable a single at-a-glance report into your security stance across multiple subscriptions, representing any team or group within your organization. This enables security metrics and reviews to be implemented at scale across the organization.

Microsoft

Just-In-Time VM access

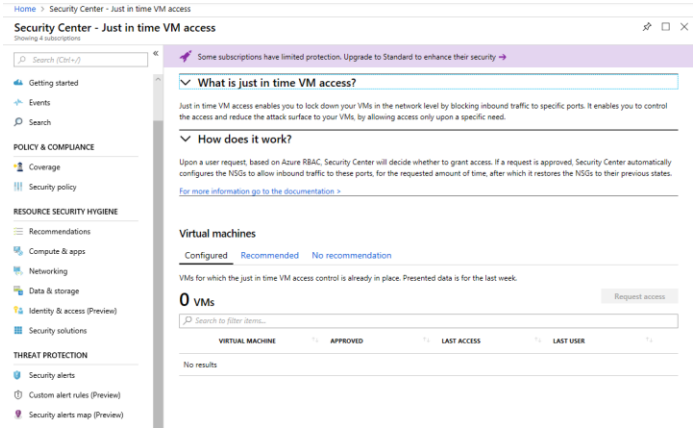
In addition to recommendations and reporting, Security Center also includes several powerful security features. One of these is Just-In-Time VM access, or JIT for short.

When running VMs in Azure, you sometimes need access to do maintenance, such as patching, deploying an update, or investigating an issue. This access is a potential security vulnerability, since it could also be targeted by an attacker. To reduce the attack surface, the endpoint used to access the VM should be closed when not in active use, and only opened for limited periods to authorized personnel.

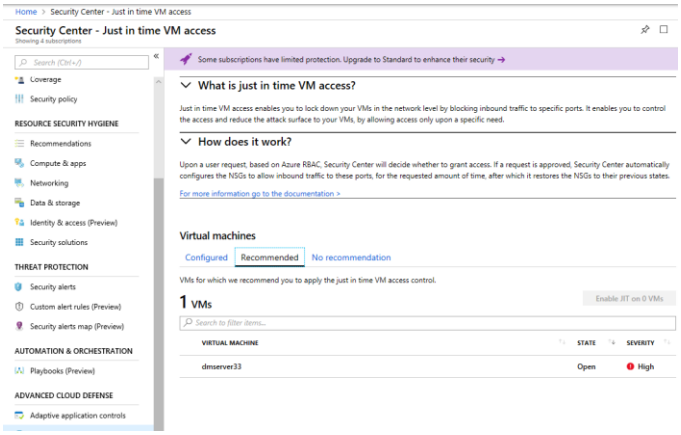
The purpose of JIT is to make this easy. JIT allows you to lock down your VM maintenance endpoints. When access is needed, an engineer makes an access request. If their role-based access control (RBAC) permissions are sufficient, access is granted and the endpoint opened. After a pre-defined time interval, say 2 hours, access is automatically restricted once more.

JIT can be configured for an individual VM from the VM blade in the Azure portal (under 'Configuration'). Alternatively, you can also configure JIT from the JIT blade within Azure Security Center. This approach has the advantage of allowing central configuration of JIT settings across multiple VMs.

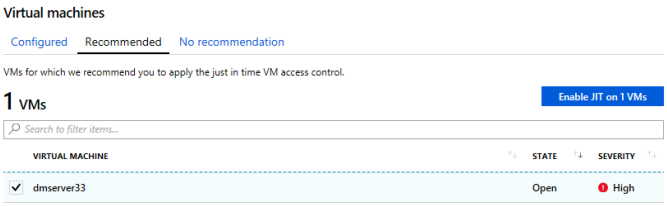
To start, open Azure Security Center and click on 'Just-in-time VM access' in the left nav.



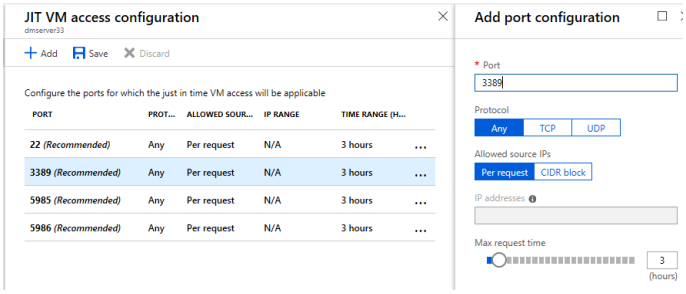
Click the 'Recommended' tab to see Security Center recommendations on servers where JIT should be deployed.



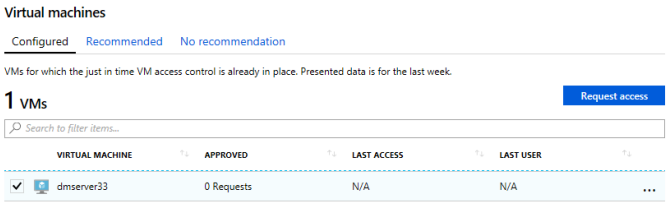
Select the VMs you with to use with JIT, and click the 'Enable JIT on *n* VMs' button.



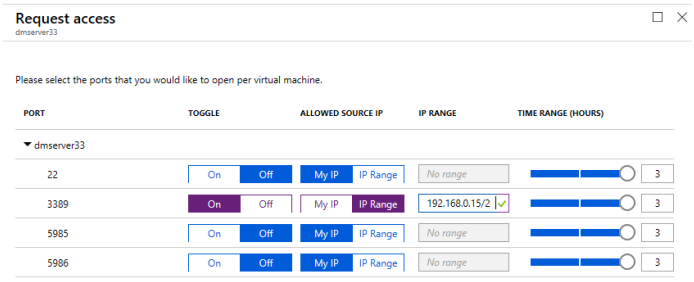
In the JIT configuration, you can select which ports and protocols are controlled via JIT and the maximum time access can be requested for. You can also which control source IP addresses are permitted access—you can restrict access to a specific CIDR (IP address range) block as part of the JIT configuration, or allow the source IP to be configured in each JIT request.



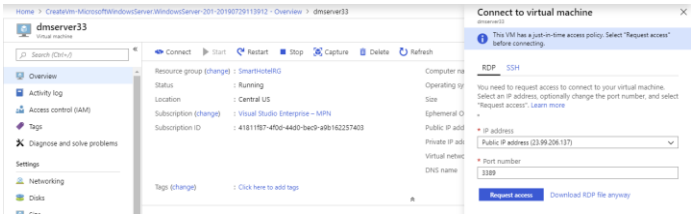
To request access, navigate to the 'Configured' tab to see the VMs with JIT enabled. Select one or more VMs and click 'Request access'.



When requesting access via Security Center, you specify the ports, allowed source IP range and the duration. These must all lie within the constraints imposed by the initial JIT configuration.



You can also request access directly from the VM blade. Simply click the 'Connect' button for the VM in the usual way—the 'Connect to virtual machine' prompt is updated to provide a 'Request access' button. This approach makes the access request immediately using default settings.



Of course, enabling JIT and requesting access can also be automated by using the Azure Powershell cmdlets:

```
$VMName = "dmserver33"
$RG = "demoRG"
$min = 60

#Get my IP address
$ip = Invoke-RestMethod http://ipinfo.io/json |
Select -exp ip

#Request access
Invoke-ASCJITAccess -VM $VMName -ResourceGroupName
$RG -Port 3389 -Minutes $min -AddressPrefix $ip
```

Access requests will only be approved if the requestor has write access to the VM enabled in Azure's role-based access control (RBAC) permissions, and all access requests are logged in the Azure Activity Log.

COMPLIANCE

Azure also includes a range of tools to help simplify and accelerate your compliance journey.

The Service Trust portal is available to existing Office365, Dynamics, and Azure customers at https://servicetrust.microsoft.com/. Access requires you to log in with your subscription credentials. In this portal, you can find information about Microsoft's implementation of controls and processes to protect Microsoft cloud services. This includes a range of audit reports for several Azure certifications.

The Service Trust portal also includes access to the Compliance Manager tool. This is a workflow-based risk assessment tool that helps you manage your compliance activities. It includes:

- Real time risk assessments on Microsoft cloud services
- Actionable insights to improve your data protection capabilities
- Simplifies compliance process through built in control management and audit-ready reporting

Further Reading

The central hub for Azure compliance information is the Azure Trust Center. This is the launch pad where all information regarding compliance regarding azure can be found.

For comprehensive documentation on all aspects of Azure security, see the Azure Security documentation.

Microsoft has also provided a detailed white paper on best practices for securing Azure applications.

There are many established principles and best practices for ensuring systems are secure by design. See for example the OWASP Security By Design principles and the security design guidance in the Azure Architecture Center.

For more information on Azure Compliance, see the Overview of Microsoft Azure Compliance white paper.

Principle 5: Continuous Change

The cloud is constantly evolving. Your applications should also evolve to take advantage of best practices and new features.

Traditional application development and deployment follows a linear flow:

- The requirements are defined, and handed to the Development team
- The code is written, and handed over to the Test team
- The application is tested, and after a bug-fix phase, handed over to the Operations team
- The application is deployed and managed in production

This linear, waterfall-style development process runs in parallel with hardware provisioning. This typically takes 3-6 months, and each software release is a multi-month project. Once purchased, the hardware is inflexible and acts as a constraint on future changes.

In the cloud, the hardware constraints are removed. Provisioning takes only minutes and can be easily adapted to changing requirements. In addition, the cloud is constantly evolving, with new features and new services being released every week. To take advantage, a more dynamic approach to both infrastructure and application architecture is required.

The 'Continuous Change' principle means embracing a dynamic, changing view of your application. Application are no longer in 'maintenance mode', being managed by Operations but no longer changing. Instead, they are constantly evolving to adapt to changing business needs, adopt new cloud features and services, and resolve underlying issues.

For example: where today you use IaaS VMs, tomorrow you might use a PaaS service. Where today you are deployed in US data centers only, tomorrow you might expand to Europe and Asia. Where today you use an on-premises network security appliance, tomorrow you use a virtual appliance from the Azure Marketplace. And so on.

Embracing Continuous Change requires and promotes continuous learning. Teams need to invest in their cloud skills and keep abreast of new cloud features and services.

The benefits of Continuous Change are applications that are responsive to changing business needs. They make efficient use of cloud resources. They require less time and effort to update and maintain.

Using the Principle

Adopt Agile processes, a DevOps team structure, and a continuous integration/continuous deployment pipeline to enable Continuous Change.

For a service to be continually evolving means that it is constantly updated. This isn't possible if each code release follows a multi-month waterfall development process. Embracing the principle of Continuous Change requires you to work differently.

Three significant changes are required to fully embrace Continuous Change:

1. Adopt an Agile development methodology
2. Create a DevOps team structure
3. Provide a continuous development / continuous deployment (CI/CD) pipeline

Let's look at each in turn.

AGILE METHODOLOGIES

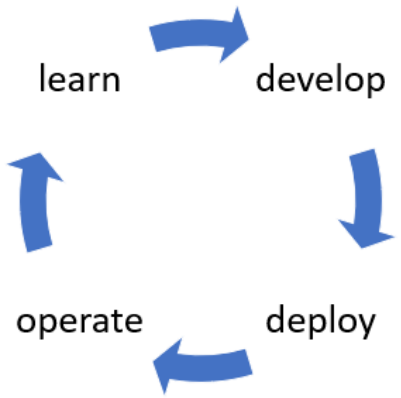
Agile development methodologies are characterized by short development cycles and rapid feedback loops, in which software is developed in short, iterative 'sprints', each delivering incremental value to production. This is a fundamentally different approach to software development favoring a reactive, iterative approach instead of relying on large up-front plans. Collectively, these approaches are known as Agile development methodologies, and their principles are captured in the Manifesto for Agile Software Development, first published in 2001.

There are a variety of Agile methodologies to choose from, such as Lean software development, Kanban, or Scrum. Whichever you choose, the essential requirement for enabling Continuous Change is to avoid long waterfall projects and embrace short, iterative development cycles.

DEVOPS

A DevOps team structure breaks down the boundaries between Development, Test and Operations teams. Instead a single, combined team is responsible for the end-to-end application from design through implementation and into production. This provides developers with first-hand experience of production issues, accountability for overall availability, and the authority to prioritize work to deliver on the overall service health *and* feature set.

A DevOps team should follow a DevOps workflow, as shown below. It is a continuous cycle of learning from and improving the production environment. DevOps thereby promotes ongoing investment in service quality, with investments prioritized based on real-world production experiences.

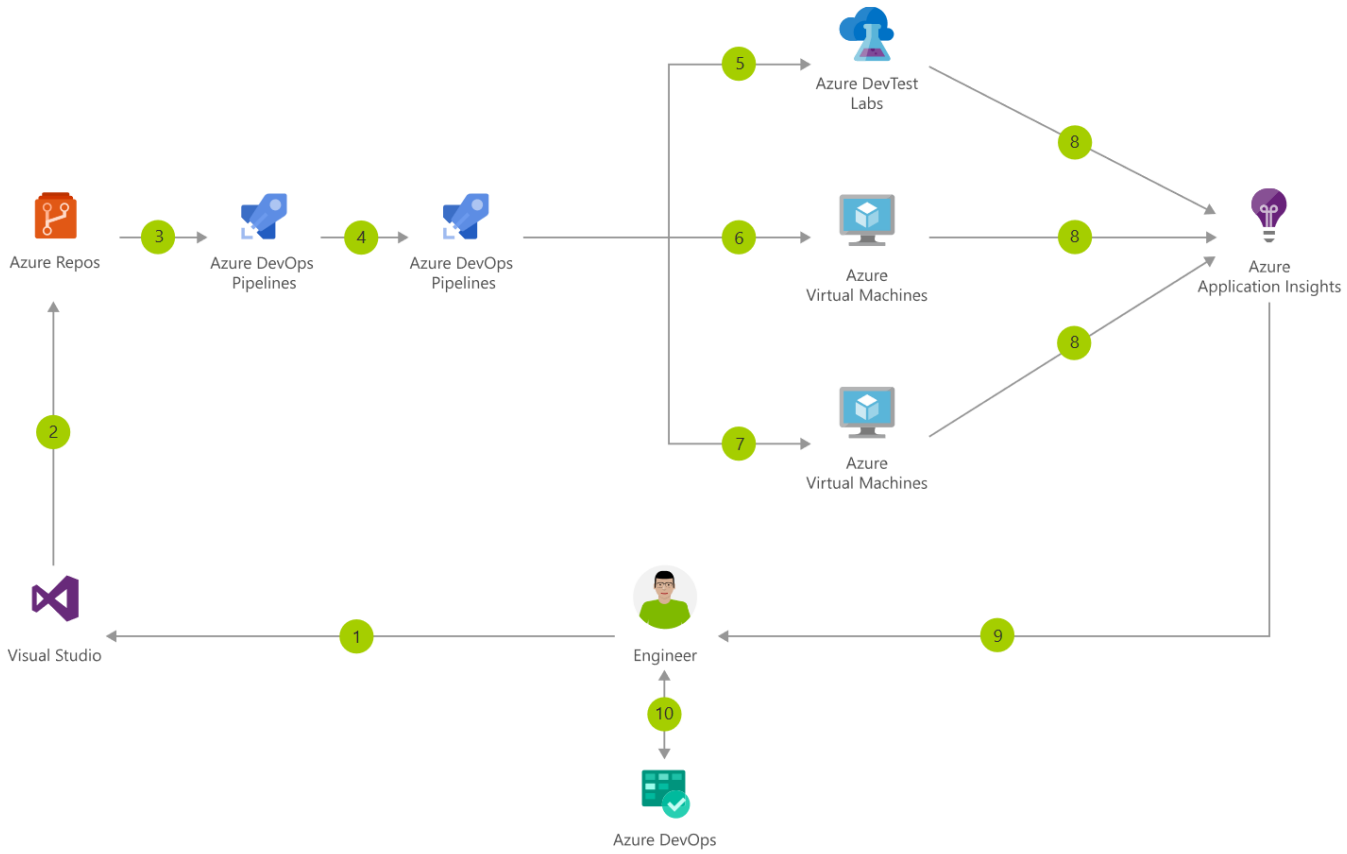


While each cycle delivers only incremental improvement, the cumulative effect can be transformative. For an inspiring analogy, see how continuous improvement has refined a Formula 1 pit stop between 1950 and the present day.

This virtuous cycle is not limited to the application code. It includes all aspects of the application, including the cloud infrastructure and services that the application runs on, and can be updated and scaled as required. This will be discussed further in the next principle, 'Software-Defined'.

CONTINUOUS DEVELOPMENT / CONTINUOUS DEPLOYMENT

To enable successful Agile development by DevOps teams, the right tools are needed. They are best supported by a continuous integration / continuous deployment (CI/CD) release pipeline. CI/CD provides an automated pipeline by which each code change is automatically compiled and deployed to a test environment. Once



Source: https://azure.microsoft.com/solutions/architecture/cicd-for-azure-vms/

tested—where possible, with automated testing—the release moves to a staging environment and then to a production environment.

CI/CD also ensures an up-to-date, releasable code-base is always available. A customer-impacting bug can be prioritized and a fix release quickly.

An effective CI/CD pipeline enables rapid releases to production with minimal overhead. Should an unexpected issue arise, a CI/CD pipeline also makes it straightforward to roll back to the previous release.

Apply cloud principles is not something only for the operations. When architecting the solutions, you should already reckon with the cloud principles. When doing so you will reap the benefits on cost, agility and more.

Many Azure PaaS services, such as Azure App Service, include native support for CI/CD. However, CI/CD can also be used with Azure virtual machines.

When building and design one should always have the cloud in mind. Incorporating cloud principles in the design and Built of cloud services is a skill set that is not only part of the Cloud operations but also for the DevOps.

APPLICATION MIGRATION

When migrating on-premises applications to the cloud, we often refer to the 6 R's

- **Rehost:** Redeploy as-is using cloud virtual machines
- **Refactor:** Make minimal changes to adapt the application to the cloud, for example using Azure SQL Database rather than SQL Server in a VM
- **Rearchitect:** Decompose the application to migrate each component to an appropriate cloud service within a cloud-optimized architecture, retaining code where possible and writing new code as necessary.
- **Rebuild:** Start afresh and create a new cloud-native application to replace the on-premises version
- **Replace:** Decommission the application and adopt a SaaS service instead.
- **Retire:** Decommission the application and migrate any existing users to alternative services.

For may migrations, this is not a one-time decision. For example, an application may initially be rehosted in Azure, since this approach offers the faster migration with low risk and minimal up-front investment.

Applying the Continuous Change principle will see this migration evolved over time, being refactored and rearchitected to make greater use of cloud-native technologies.

This evolution occurs post-migration, as part of the 'Optimize' phase of a typical migration project.

Further Reading

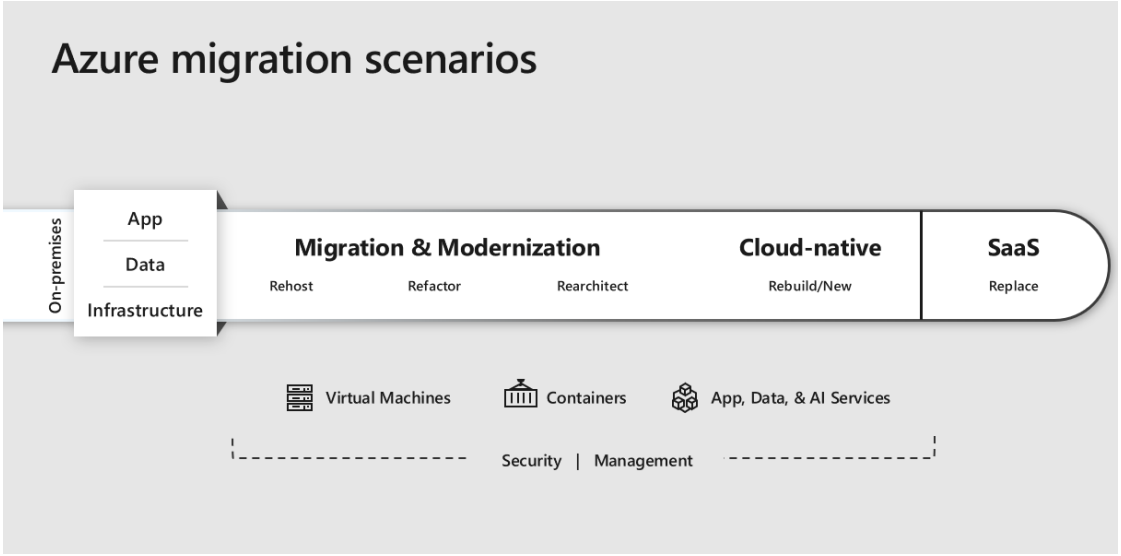
Azure provides extensive features for delivering CI/CD pipelines that enable DevOps teams and Agile methodologies. For more information, see DevOps on Azure.

The following books provide an insightful perspective on the challenges and power of adopting DevOps. The first provides a practical guide, whereas the second adopts a novel format (literally, as a novel!) to convey the value DevOps can bring.

- *The DevOps Handbook*, ISBN 978-1-942788-00-3
- *The Phoenix Project*, ISBN 978-0-9882625-0-8

An interesting perspective on the need to adopt DevOps to take advantage of the cloud is provided in DevOps and Cloud: Like Chocolate and Peanut Butter.

Download the Effective DevOps e-book from O'Reilly to learn more on building a DevOps culture in your organization.



Principle 6: Software-Defined

Use automation for deployment and common operations tasks for consistency, agility and efficiency.

Our 6th principle is that every aspect of your application should be Software-Defined.

From a customer point of view, every Azure resource is managed remotely using APIs, command line tools and management portals. Microsoft is responsible for all physical infrastructure. This means that every aspect of your application deployment can be automated. This is supported by a variety of tools and scripts from both Microsoft and third parties.

In a software-defined environment, there are no manual steps required during deployment. Deploying a completely new environment will require environment-specific parameters to be defined. These include VM sizes and instance counts, and external resources such as domain names and SSL/TLS certificates. With these prerequisites in place, the remainder of the deployment is fully automated, resulting in a fully-functioning application.

This automation is a key enabler of the continuous integration / continuous deployment pipeline we discussed in Principle 5. The scripts used to automate environment creation and deployment are managed similarly to application code—they can be stored in a source code repository and changes can be version controlled. This is known as an 'Infrastructure as Code' (IaC) approach.

Like many of the other principles, the Software-Defined principle requires a change of mindset. It doesn't suffice simply to automate your infrastructure provisioning and deployment. Any changes to the infrastructure or the application should be made using the automation.

For example: to change the size of a VM, the script or parameter file used by the automation to define the VM should be updated, checked in, and deployed. Changing the VM size in any other way—such as via the Azure portal—would mean the automation and the deployed environment become out of sync, and the change will be overwritten the next time the deployment automation is run. Direct changes to the environment should be reserved for critical break-fix changes only, which are then back-ported to the automation.

In addition, the Software-Defined principle does not stop at deployment. All on-going operations tasks should also be automated. This includes deployment of updates, scaling the service up or down, and any other common maintenance tasks. You can also automate how your application responds to events. For example, to send an email summary of daily usage, or to automatically reboot a VM in the case of an out-of-memory alert.

Automation isn't new. Developers and IT Professionals have been using code and scripts to automate common and repetitive tasks for many years. What is new is that virtualized cloud environment makes it possible to automate *everything*. Also new is that the cloud drives an increased impetus to adopt DevOps processes and CI/CD pipelines, which depend on automation.

You can't fully implement the Continuous Change principle without first implementing the Software Defined principle.

There are many benefits to a Software-Defined approach. First, humans make mistakes. Relying on manual steps in deployment or operations is to accept these mistakes as an inevitable cost. With properly tested automation, these processes can be repeated day and night with complete consistency and without error. An additional benefit is that automation reduced deployment times and frees valuable staff time for higher-value tasks. Finally, the automation scripts provide a precise record of the deployment itself. Like well-commented code, the infrastructure becomes self-documenting.

Using the Principle

Automated deployment and management using Azure Resource Manager templates to implement an Infrastructure as Code approach.

INFRASTRUCTURE AS CODE

A range of tools to implement an Infrastructure as Code approach to Azure deployment, from both Microsoft and third parties. These include:

- Azure Resource Manager templates (Microsoft)
- Desire State Configuration (Microsoft)
- Puppet (Puppet Labs)
- Chef (Chef)
- Ansible (Red Hat)
- Terraform (Hashicorp)

Common to any automated deployment is the concept of idempotence. What does this mean? An idempotent script or operation has the property that end state is always the same, regardless of the starting condition. For example, you might have a script to create a VM with a given name. If the VM doesn't exist, it is created. If the VM already exists, the script shouldn't create a second one, it should just update the existing VM to match the settings in the script. Thus the end result is the same, regardless of the initial state.

Idempotency is a critical feature of any IaC implementation. It allows the same scripts to be used for both initial deployments and updates, and removes the need for endless 'if-then' conditional logic. Idempotency also makes error handling much simpler, since any operation can simply be re-tried without having to worry about any incomplete state. The script is said to be 'declarative', in that it declares the 'goal' state and let's the platform figure out the changes required to get there.

When deploying an application in Azure, your first concern is to provision the Azure resources and ensure they are connected correctly. This can be achieved using Azure Resource Manager templates. Each template defines a list of resources, including all resource properties and dependencies.

Your second concern is to ensure the correct applications and code are deployed into those resources. Here, your approach will depend on the service being used. For example, Azure App Service can integrate with many common source code control systems to automatically retrieve and publish the application. For Azure VMs, you can run deployment scripts as part of the VM provisioning. Azure Automation State Configuration uses the PowerShell Desired State Configuration language to provide an idempotent scripting language for deploying server components and custom applications. It can be used with both Azure VMs and on-premises servers.

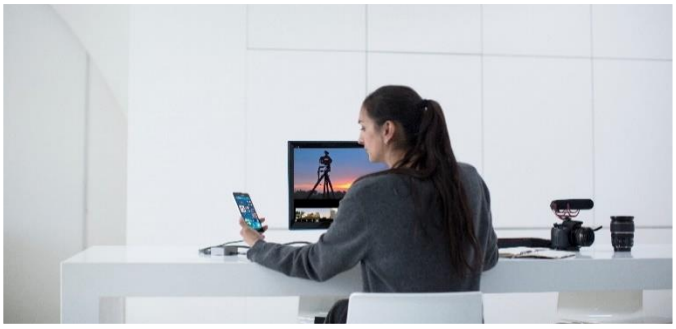
Azure Resource Manager Templates

Most important of all these IaC technologies are Azure Resource Manager templates, which we'll simply call templates for convenience. Each template is a text file, formatted in JSON, that lists a collection of resources to be deployed.

Templates are a declarative language and can be used both for new deployments and to update existing deployments. While the template focuses on resources and their properties, they can also reference additional scripts to be run on your VMs during VM provisioning.

Each template can be parameterized. For example, this enables the same template to be used for development, staging and production environments, with different parameter values being used to size each environment appropriately. These parameters can be specified using a parameters file, stored alongside the template, consistent with a fully-automated, Software-Defined approach.

Microsoft provides substantial in-depth documentation on how to write templates, together with a comprehensive set of examples in the Azure QuickStart templates library. For the purposes of illustration, let's look at a simple template to deploy a virtual network, subnet, and network security group.



The structure of a template is as follows:

```
{
  "$schema":
"https://schema.management.azure.com/schemas/2015-
01-01/deploymentTemplate.json#",
  "contentVersion": "",
  "apiProfile": "",
  "parameters": {  },
  "variables": {  },
  "functions": [  ],
  "resources": [  ],
  "outputs": {  }
}
```

We will be concerned only with the parameters, variables and resources sections. The remaining sections are either boiler-plate text or relate to more advanced templates outside the scope of this example.

The parameters section is used to define any template inputs. The value of each parameter must be specified when the template is deployed.

```
"parameters":{
     "virtualNetworkName": {
            "type": "string",
            "minLength": 1,
............"maxLength": 20
            "metadata": {
                "description": "The name of the
Virtual Network."
            }
     }
},
```

Each parameter definition can specify constraints, such as the min/max length of strings or the min/max value of integers.

The variables section is internal to your template. It allows you to define values that you will use later in the template. Variables allow you to make your template more readable, and de-duplicate values that appear more than once in your resource definitions.

```
"variables": {
     "vnetPrefix": "10.0.0.0/16",
     "subnetName": "hostsubnet",
     "'subnetPrefix'": "10.0.0.0/24",
     "NSGName": "MyNSG",
},
```

In the resource section you describe the Azure resources you want to deploy. Each resource definition comprises some common metadata such as the resource name, type, and API version, followed by a collection of type-specific resource properties.

```
"resources": [
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "name": "[variables('NSGName')]",
      "apiVersion": "2018-08-01",
      "location": "[resourceGroup().location]",
      "tags": {},
      "properties": {
        "securityRules": [
          {
            "name": "RemoteDesktop",
            "properties": {
              "protocol": "TCP",
              "sourcePortRange": "*",
              "destinationPortRange": "3389",
              "sourceAddressPrefix": "*",
              "destinationAddressPrefix": "VirtualNetwork",
              "access": "Allow",
              "priority": 300,
              "direction": "Inbound"
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/virtualNetworks",
      "name": "[parameters('vnetName')]",
      "location": "[resourceGroup().location]",
      "apiVersion": "2016-03-30",
      "dependsOn": [
        "[variables('NSGName')]"
      ],
      "properties": {
        "addressSpace": {
          "addressPrefixes": [
            "[variables('vnetPrefix')]"
          ]
        },
        "subnets": [
          {
            "name": "[variables('subnetName')]",
            "properties": {
              "addressPrefix": "[variables('subnetPrefix')]",
              "networkSecurityGroup": {
                "id":
"[resourceId('Microsoft.Network/networkSecurityGroups',
variables('NSGName'))]"
              }
            }
          }
        ]
      }
    }
 ],
```

In the above example, notice how the 'dependsOn' declaration is used to ensure the NSG is deployed before the virtual network.

EVENT HANDLING

We have seen how you can automate resource deployment using Azure Resource Manager templates and discussed how VM configuration can be automated using PowerShell DSC. Another automation scenario is responding to alerts or other events.

The first step is to detect the event. Events can be generated by many Azure services. Examples include Activity Log alerts, metric alerts, log alerts, or other triggers such as a web request. Azure Event Grid is a fully-managed service designed to consume events from any Azure or external service and trigger appropriate actions.

Those actions can be implemented either in your application, or using a variety of serverless technologies available in Azure. These include Logic Apps, which provides rapid, graphical development of process flows and integration with a wide variety of external service, and Azure Functions which enables serverless deployment of fully customize code in a variety of languages.

Further Reading

To learn more about the Azure Resource Manager template language, see the Azure Resource Manager template structure and syntax reference documentation. The Azure quickstart templates provide a wide range of template examples you can use to get started. The Azure Resource Manager template reference documentation provides details on how to describe each resource type in a template.

The Azure documentation pages provide further information about Azure Functions, Azure Automation, Event Grid and Logic Apps.

Principle 7: Pay-per-Use

The cloud changes not only how companies implement their IT projects, but also changes fundamentally how IT is paid for. Adapting to this new billing model is a key to success.

Traditional IT requires up-front capital expenditure on hardware, software licenses, and facilities, plus on-going operational costs for staffing, power, and connectivity. These are known as 'CapEx' and 'OpEx' respectively. In the cloud, CapEx costs are borne by the cloud provider. As an Azure customer, your Azure bill is based on your metered resource usage and is typically treated as OpEx.

This shift from up-front to metered billing is a very fundamental change, with far-reaching implications for successful cloud adoption. The Pay-per-Use principle describes how to adapt your cloud usage to optimize your costs. This requires new skills, new responsibilities, and cultural change, as we shall now discuss.

Accurately pricing an Azure solution isn't always easy. The billing model can be complex, and pricing can vary based on many variables. The ability to accurately price an Azure solution, and to identify cost-saving opportunities, are skills that require both study and experience.

Cost optimization needs to be factored into your application designs. Architects and developers need to consider cost alongside other requirements such as performance, scale or availability. There is an interplay between each of these requirements, making it more difficult to retro-fit a cost optimization later without impacting these other factors. They need to be considered together at design time—something many architects will not be used to.

Last, cost optimization requires cultural change. Organizations need to become cost-aware, with each team accountable for their Azure costs and with clear targets and reporting. Cost optimization is not a one-time task, rather it is an on-going activity, since requirements and the available Azure features evolve over time. An optimized application can only be delivered and maintained, and cloud waste minimized, if cost awareness is baked into the organizational mindset.

Using the Principle

Learn how to forecast, monitor and optimize your costs to take advantage of Pay-per-Use cloud billing.

UNDERSTAND AZURE BILLING

Before you can optimize your Azure costs, you need to understand in detail how you Azure costs are calculated.

How Azure Billing Works

Each Azure resource is billed separately. For example, for a single virtual machine, there are separate resources for the machine itself (which determines the CPU and memory characteristics), the disk, and the network. Each appears as a separate line item on your Azure bill.

Each resource also has its own billing model, which is described on the Azure website pricing page for that resource. Resources are typically billed based on the length of time they are created for, or how much they are used, or both. It is important to understand the billing model for each of the resources in your application.

For example, a virtual machine is charged based on how long the machine is used for. Network usage is based on the volume of data using the network. An ExpressRoute circuit, used for connecting Azure with on-premises networks, has two billing models to choose between: either a fixed monthly charge only, or a lower fixed monthly charge plus a charge for each GB of outbound data.

Discounts

Azure also offers a variety of discount plans, which can offer very substantial cost savings. Taking full advantage of the available discounts is essential to optimizing the costs of any deployment. Key amongst these discounts are Azure Reservations and Azure Hybrid Benefit.

Azure Reservations allows you to pre-purchase selected Azure resources, at a discounted rate. Reserved VM Instances allow you to pre-purchase Azure virtual machines, while Reserved Capacity allows you to pre-purchase Azure SQL Database, Azure SQL Data Warehouse, and Azure CosmosDB. Reservations are paid for in advance and used to discount eligible resources from your monthly bill. This advance payment model can be useful where an organization is not yet ready to shift expenditure from CapEx to OpEx for internal reasons.

Azure Hybrid Benefit allows you to use existing Windows Server and SQL Server licenses to discount the cost of running corresponding Azure resources. Windows Server licenses are discounted against Windows VMs, and SQL Server licenses against Windows VMs running SQL Server or against Azure SQL Database.

Azure Reservations and Azure Hybrid Benefit can be combined for even greater savings.

COST FORECASTING

There are several tools available to help you calculate Azure costs. Some of the most commonly-used tools are described below. Many third-party cost management tools also include cost forecasting and optimization capabilities.

Azure Pricing Calculator

The Azure pricing calculator is the most commonly-used pricing tool. It allows you to specify each resource used by your application, together with the resource parameters that impact on pricing, such as SKU, region, and applicable discounts. The calculator provides per-resource pricing, in a currency of your choice.

As an example, let us calculate the cost of a VM with several data disks, as described in one of the Azure quickstart templates. According to the template, the following resources are used:

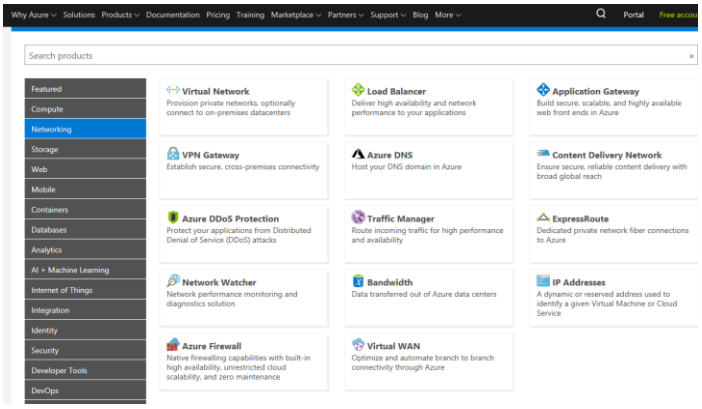
- Microsoft.Storage/storageAccounts
- Microsoft.Network/publicIPAddresses
- Microsoft.Network/virtualNetworks
- Microsoft.Network/networkInterfaces
- Microsoft.Compute/virtualMachines

The virtual machine also uses managed disks (Microsoft.Compute/disks), although these are specified implicitly in the virtual machine properties, rather than as explicit resources within the template.

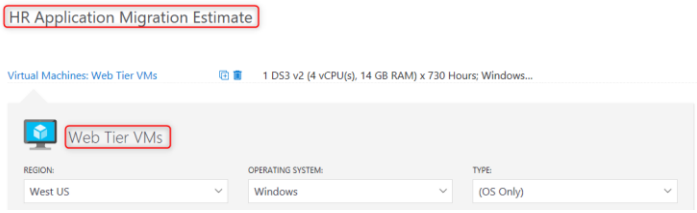
The virtual machine size and data disk type will both impact the pricing. These values are specified as template parameters, so a range of values could be used in practice. Let's assume the default values defined in the template are used: Standard_DS3_v2 for the VM SKU and StandardSSD_LRS for the data disk type. The number and size of data disks are determined by template variables (5 data disks, each 1024 GB).

These dependencies on parameters, variables, and implicitly-defined resources illustrate the importance of properly understanding the template and paying attention to detail when calculating costs.

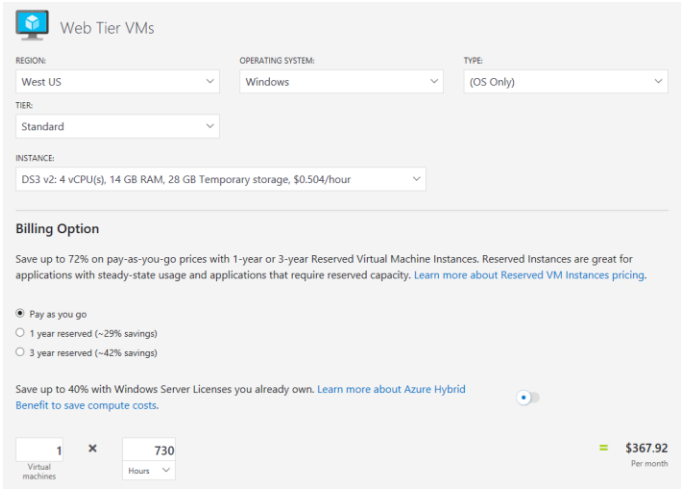
To calculate the costs, open the Azure pricing calculator and add each of the resource types identified.



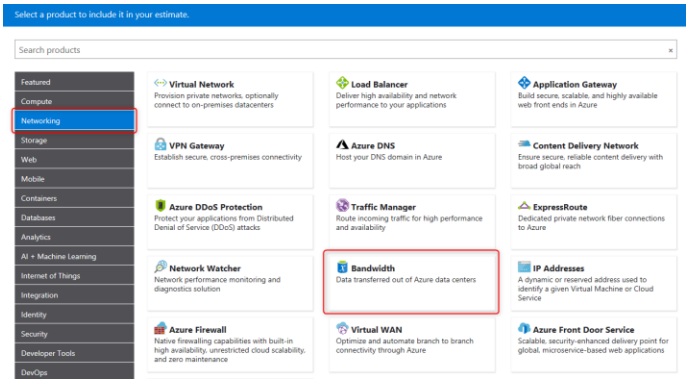
Note that you can change the name of your price estimate and provide a name for each resource included. This is very useful, for example to describe the different tiers of your application as 'Web Servers', 'Database Servers' rather than simply 'Virtual machines'.



It is essential that every pricing calculator field is filled in correctly, since all fields exposed by the calculator have the potential to impact the price.

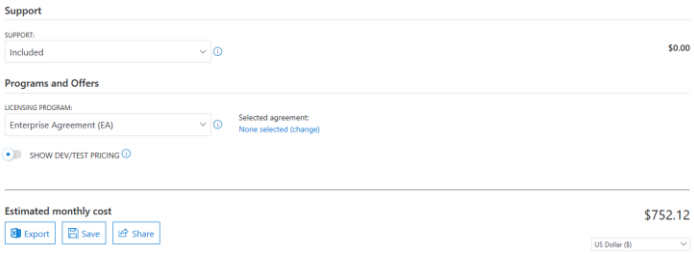


Take care to ensure network bandwidth is specified correctly. Data transferred across peering connections between virtual networks is specified within the virtual network resource. Egress bandwidth (for example, for web traffic served from a web server) is priced separately, using the 'Bandwidth' type within the calculator, which must be added to your estimate even though it is not a resource type within your deployment.



The pricing calculator lets you save your estimates, and export them as an Excel spreadsheet. The total price can be shown in the currency of your choice. Those using Enterprise Agreement (EA) or Cloud Solution Provider (CSP) subscriptions can specify their subscription type, so

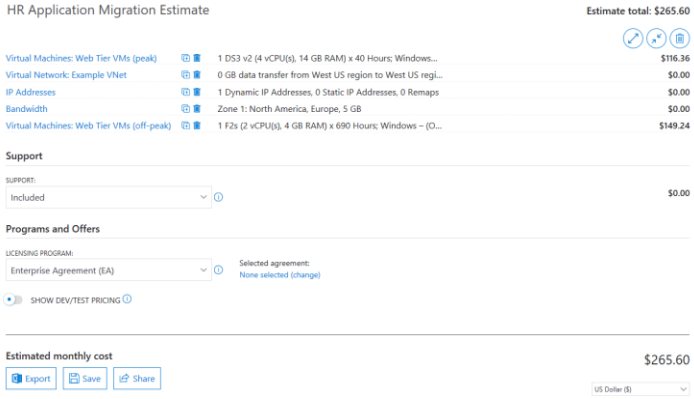
the pricing calculator applies the correct discounts. Support plans can also be included.



The price shown above would be the monthly price if you run this VM 24x7.

Suppose now that this capacity is only required for 40 hours per month. During the remaining hours, a much smaller F2s VM suffices. In addition, suppose that the data disk size can be reduced from 1024 to 256 GB.

To estimate this scenario, change the hours for the existing VM, and add a new F2s VM for the remaining hours. Only specify the data disks on one of the VMs, to avoid double-counting.

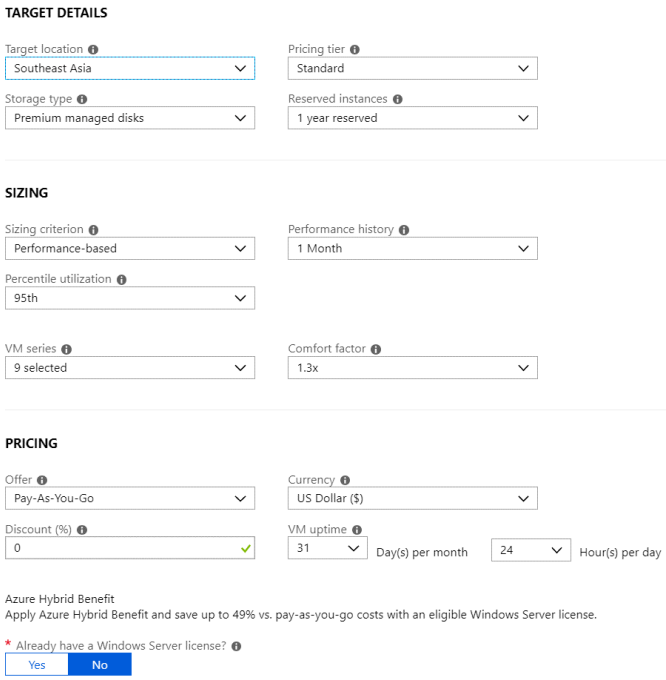


In this scenario, the total monthly cost is reduced from $752.12 to $265.60, representing an annual saving of over $5,800, or 65%. Using hybrid benefit (if eligible) and reserved instances for the F2s VM could result in further savings.

Azure Migrate

The Azure Migrate tool helps you assess on-premises environments and migrate them to Azure. As part of the migration assessment phase, it provides estimates for what the on-premises workload will cost to run using Azure VMs.

This estimate is calculated based on a variety of parameters, which you can control to fine-tune the pricing. For example, you specify the VM tier, hours of operation, and discounts.



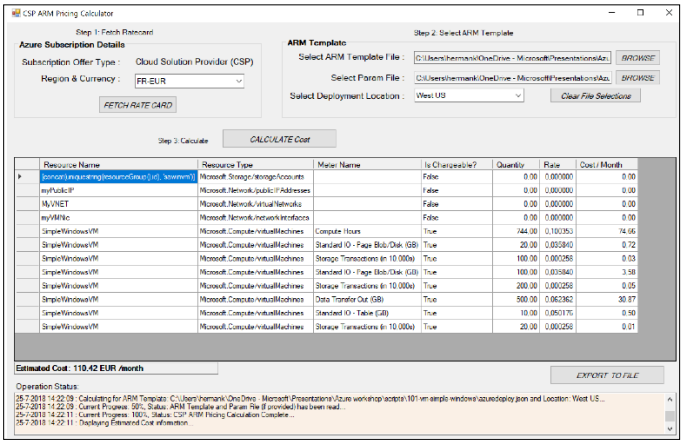
The size of VM can be based either on the size of the on-premises machines, or based on their utilization, using performance metrics gathered by Azure Migrate. Since many on-premises servers are heavily under-utilized, sizing VMs based on utilization can offer substantial savings.

CSP ARM Pricing Calculator

The CSP ARM Pricing Calculator is an Azure pricing tool aimed specifically for Azure CSP partners. It takes as input an Azure Resource Manager template and parameters file, and provides a pricing calculation based on the resources

defined by those files. The calculation is based on the CSP rate card, so takes account of any CSP discounts.

The tool is available for free on Github.

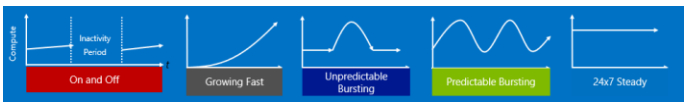


RIGHT-SIZING

Right-sizing your deployment footprint means deploying sufficient resources to meet your needs, while ensuring you do not unnecessarily over-provision resources and thus spend more than you have to. Right-sizing applies to the VM size and type, disk size and type, and the number of VMs and disks deployed.

Unlike traditional infrastructure, the flexibility of Azure enables all of these parameters to be updated over time as your workload needs change. With traditional infrastructure, hardware is typically over-provisioned to handle worst-case scenarios, such as failures and an anticipated future peak demand. This leads to CPU utilization that is often under 10%. In contrast, the ability to scale Azure VMs on demand means workloads can target a CPU utilization of up to 80%. For optimal cost-savings, workloads should be monitored and scaled up or down to keep the utilization within this range.

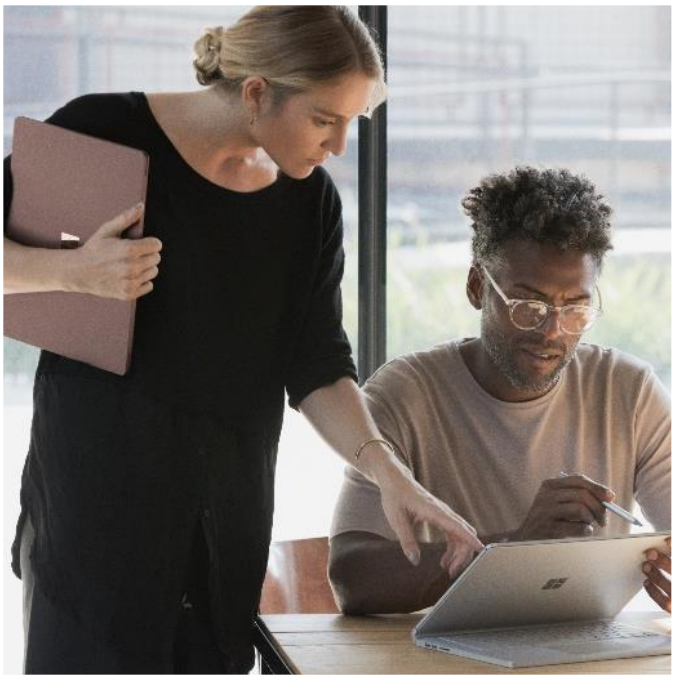
Applications rarely experience a constant load, 24x7. More usually, demand will vary based on user behavior. The following diagram shows some typical usage patterns. Right-sizing enables your application to maintain efficient utilization regardless of the usage pattern.



In addition, Azure VMs are based on the latest Intel and AMD processors, on hardware and networks that are highly tuned for optimal performance. This allows further optimization by choosing a VM family that offers sufficient CPU capacity and a suitable CPU/memory ratio.

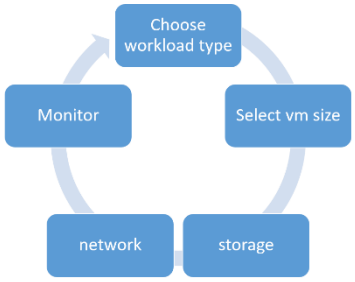
Azure VMs run on a range of different Intel and AMD processors, with new SKUs being introduced regularly to support the latest and most powerful models. To enable a comparison of CPU capacity across VM families, Microsoft publishes a table of 'Azure Compute Unit' (ACU) values for each VM family. This measures the relative CPU performance across VM families, with the A1 SKU benchmarked as 100 ACU, allowing you to compare performance.

SKU Family	ACU \ vCPU	vCPU: Core
A0	50	1:1
A1-A4	100	1:1
A5-A7	100	1:1
A1_v2-A8_v2	100	1:1
A2m_v2-A8m_v2	100	1:1
A8-A11	225*	1:1
D1-D14	160	1:1
D1_v2-D15_v2	210 - 250*	1:1
DS1-DS14	160	1:1
DS1_v2-DS15_v2	210-250*	1:1
D_v3	160-190*	2:1**
Ds_v3	160-190*	2:1**
E_v3	160-190*	2:1**
Es_v3	160-190*	2:1**
F2s_v2-F72s_v2	195-210*	2:1**
F1-F16	210-250*	1:1
F1s-F16s	210-250*	1:1
G1-G5	180 - 240*	1:1
GS1-GS5	180 - 240*	1:1
H	290 - 300*	1:1
L4s-L32s	180 - 240*	1:1
M	160-180	2:1**



Choosing the Right VM

Choosing the right VM is an important part of any initial deployment. Even more important is to treat this process as a continual cycle, adjusting according to measured performance and changing demand.

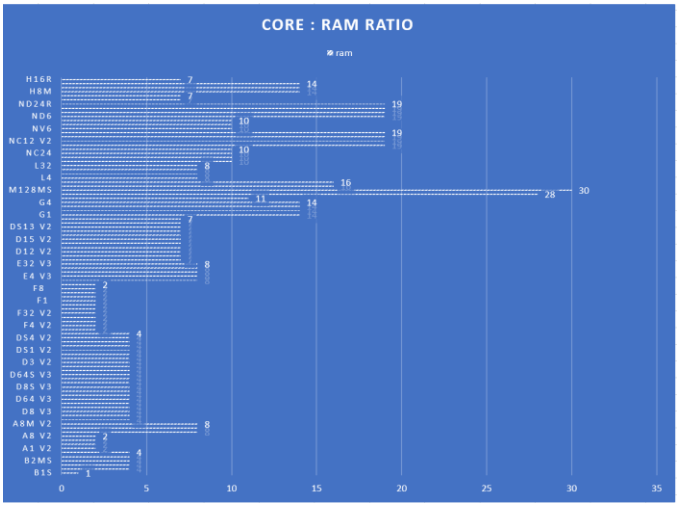


The first step is to choose the workload type. This determines which VM family to use. Azure supports a wide range of VM families, designed for different workloads. The following table illustrates the VM types available (source):

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, DSv2, Dv2, DS, D, Av2, A0-7	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2, Fs, F	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, DS, Dv2, D	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Ls	High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases.
GPU	NV, NC, NCv2, ND	Specialized virtual machines targeted for heavy graphic rendering and video editing. Available with single or multiple GPUs.
High performance compute	H, A8-11	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

Use the table to identify the workload type. The table will then show which VM sizes and types are suitable for that workload. If you do not know the workload type, start with the General Purpose type.

Having chosen the VM type, the second step is to choose the initial VM SKU (family and size) to deploy. This will be based on the number of CPU cores and how much memory your workload requires. Within each VM family, the ratio of CPU to memory is fixed—a larger VM with double the CPUs will also have double the memory. However, between different VM families, the ratio of CPU to memory can vary considerably. This is illustrated in the following diagram:

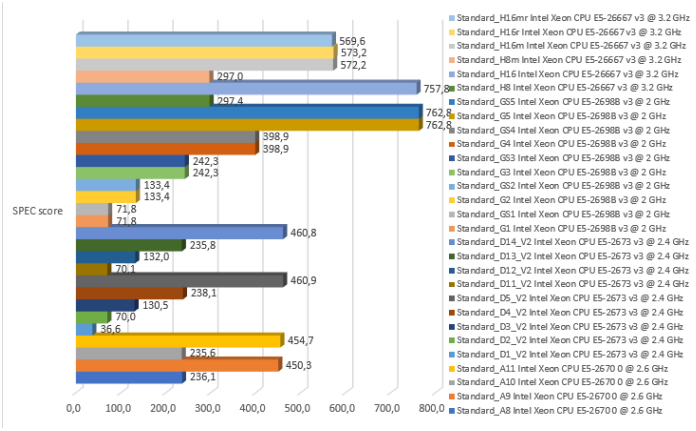


When selecting a VM size the first question is in most cases is how much memory is needed, followed by how much CPU capacity. Understand if these requirements are sensitive to the number of concurrent users, since using a larger number of smaller VMs is usually preferable to a small number of large VMs, since it offers better scalability and resiliency.

Bear in mind your target CPU and memory utilization, and avoid the temptation to over-provision. A better rule of thumb is 'less is best'. If the VM turns out to be to small you can always redeploy with more capacity.

Measurements from proof-of-concept deployments or on-premises environments can help inform your VM choice. However, bear in mind that Azure CPUs will be different from those in your own datacenter. To help you make valid comparisons, Microsoft publishes compute

benchmark scores for each Azure VM SKU, computed by running SPECint 2006 on Windows Server:



Also important is to remember that when running VM's in a on-premise environment, the hypervisor may be configured to use only a fraction of a physical CPU core for each virtual core. Ratios of 1:4 are not uncommon, whereas in Azure each physical core supports only 1 or in some cases 2 virtual cores (see the earlier table showing ACU values). A workload that previously required 4 vCores on-premises may therefore run in a single vCore in Azure, since both represent a single physical core. Assessing on-premises environments using Azure Migrate will take these factors into account.

The next step is to consider the VM storage. This may influence the VM choice, since the choice of VM SKU will determine the size of the temporary disk available, the number of data disks supported, and the maximum supported disk throughput and IOPS. In addition, only certain VM SKUs (denoted by an 's' suffix) support Premium SSD drives, and only certain VM SKUs support premium storage disk caching for increased performance.

Similarly, your networking requirements will also influence your VM choice. The number of network interfaces and the network capacity of the VM vary according to the VM family and size.

The following table illustrates the storage and network limits for the DS_v3 VM family (link). Similar tables are available for each VM family.

Dsv3-series [1]

ACU: 160-190

Premium Storage: Supported

Premium Storage caching: Supported

Dsv3-series sizes are based on the 2.4 GHz Intel Xeon® E5-2673 v3 (Haswell) processor or the latest 2.3 GHz Intel XEON ® E5-2673 v4 (Broadwell) processor that can achieve 3.5GHz with Intel Turbo Boost Technology 2.0 and use premium storage. The Dsv3-series sizes offer a combination of vCPU, memory, and temporary storage for most production workloads.

Size	vCPU	Memory: GiB	Temp storage (SSD) GiB	Max data disks	Max cached and temp storage throughput: IOPS / MBps (cache size in GiB)	Max uncached disk throughput: IOPS / MBps	Max NICs / Expected network bandwidth (Mbps)
Standard_D2s_v3	2	8	16	4	4000 / 32 (50)	3200 / 48	2 / 1000
Standard_D4s_v3	4	16	32	8	8000 / 64 (100)	6400 / 96	2 / 2000
Standard_D8s_v3	8	32	64	16	16000 / 128 (200)	12800 / 192	4 / 4000
Standard_D16s_v3	16	64	128	32	32000 / 256 (400)	25600 / 384	8 / 8000
Standard_D32s_v3	32	128	256	32	64000 / 512 (800)	51200 / 768	8 / 16000
Standard_D48s_v3	48	192	384	32	96000 / 768 (1200)	76800 / 1152	8 / 24000
Standard_D64s_v3	64	256	512	32	128000 / 1024 (1600)	80000 / 1200	8 / 30000

[1] Dsv3-series VM's feature Intel® Hyper-Threading Technology

Having chosen your VM SKU, you will then need to consider the storage requirements of the VM. Each VM comes with an OS disk and temporary disk. You have full control over additional data disks.

For each disk, you will need to specify the type. There are 4 types available as described in the following table (link). Pricing varies according to disk type. As with VMs, so be sure to choose a type that meets your needs without being tempted to over-provision.

Disk comparison

The following table provides a comparison of ultra solid-state-drives (SSD) (preview), premium SSD, standard SSD, and standard hard disk drives (HDD) for managed disks to help you decide what to use.

	Ultra SSD (preview)	Premium SSD	Standard SSD	Standard HDD
Disk type	SSD	SSD	SSD	HDD
Scenario	IO-intensive workloads such as SAP HANA, top tier databases (for example, SQL, Oracle), and other transaction-heavy workloads.	Production and performance sensitive workloads	Web servers, lightly used enterprise applications and dev/test	Backup, non-critical, infrequent access
Disk size	65,536 gibibyte (GiB) (Preview)	32,767 GiB	32,767 GiB	32,767 GiB
Max throughput	2,000 MiB/s (Preview)	900 MiB/s	750 MiB/s	500 MiB/s
Max IOPS	160,000 (Preview)	20,000	6,000	2,000

The choice of disk type and size will determine the maximum disk throughput and IOPS. The following table

illustrates how throughput and IOPS vary with disk size for Premium SSD disks.

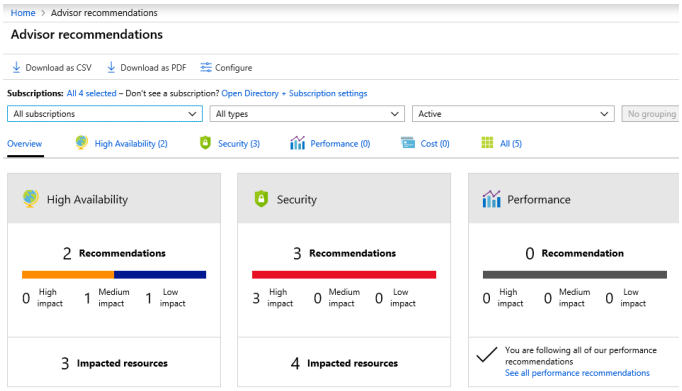
Premium SSD sizes	P4	P6	P10	P15	P20	P30	P40	P50	P60	P70
Disk size in GiB	32	64	128	256	512	1,024	2,048	4,096	8,192	16,384
IOPS per disk	Up to 120	Up to 240	Up to 500	Up to 1,100	Up to 2,300	Up to 5,000	Up to 7,500	Up to 7,500	Up to 16,000	Up to 18,000
Throughput per disk	Up to 25 MiB/sec	Up to 50 MiB/sec	Up to 100 MiB/sec	Up to 125 MiB/sec	Up to 150 MiB/sec	Up to 200 MiB/sec	Up to 250 MiB/sec	Up to 250 MiB/sec	Up to 500 MiB/sec	Up to 750 MiB/sec

Remember that the maximum disk throughput and IOPS is determined by both the per-disk limits and the per-VM limits, and take both sets of limits into account.

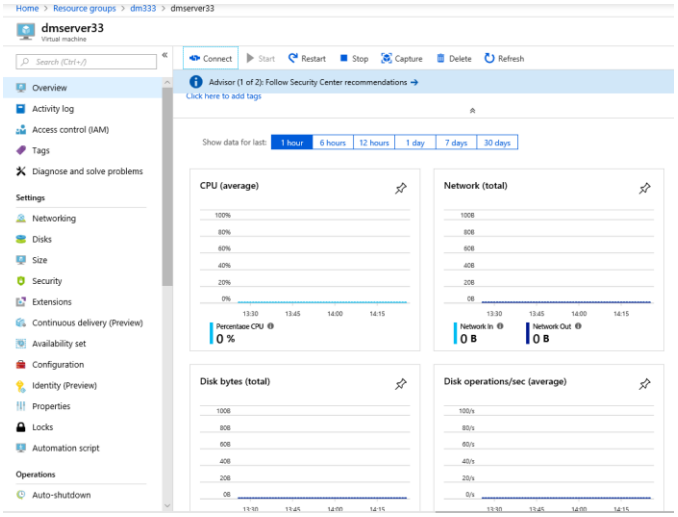
Monitoring and Updating VM Size

Ongoing monitoring is essential to validate the VM size chosen, and update as necessary to maintain a cost-efficient utilization.

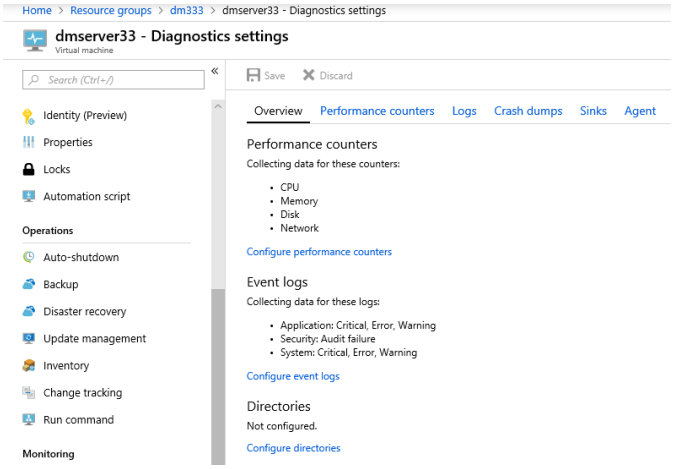
A first step is to review Azure Advisor, to check for alternative VM size recommendations. These may appear under either the 'Performance' or 'Cost' categories.



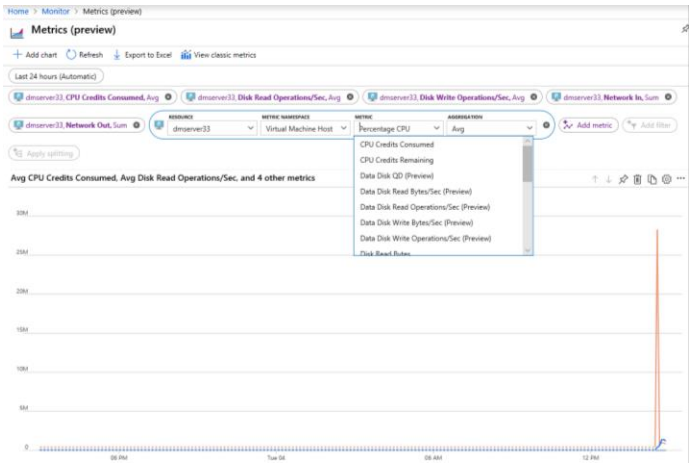
To understand the VM performance, you will need to review the VM performance metrics. Basic metrics can be seen on the VM Overview blade in the Azure portal:



The default metrics are gathered from the Azure hypervisor. For deeper analysis, you can enable additional performance counters from the VM Diagnostic Settings blade. These are gathered by the VM Agent running on the VM.



You can review the VM metrics in Azure Monitor. First select the subscription, resource group and VM, then select the counters to view. This data can also be exported for offline analysis.



Based on these metrics, you can determine if the VM is optimized for the current workload or whether it should be changed to offer better utilization and lower cost.

Re-sizing a VM requires the VM to be rebooted. For a single-VM deployment, this will result in service downtime. The time and effort to re-size will vary, depending on whether the new VM size is available in the same hardware cluster as the existing VM—see this blog post for more information.

For load-balanced workloads, re-sizing VMs can be achieved without downtime by adding new VMs to the load balancer as old VMs are removed. The Azure Load Balancer does not current support connection draining (waiting for existing connections to close before removing the VM from the load balancer). This means removing the existing VM will terminate existing sessions.

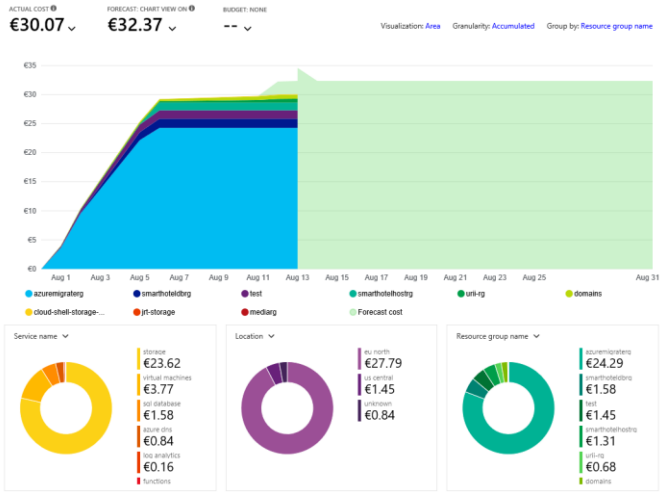
A workaround is to use the load balancer health checks to enable connection draining. Add a custom health check page to your application, such as healthcheck.html, and configure this page in your load balancer health probe. Before removing the VM from the load balancer, rename the health check page (for example, to _healthcheck.html) on the affected VM. This will result in the load balancer health probe receiving an HTTP 404 error, causing the load balancer to direct new connections to other VMs. Wait for any existing connections to close, then remove the VM from the backend pool. With this approach, VMs can be resized without user impact.

COST MANAGEMENT

Azure Cost Management is an Azure service designed to help you understand and manage your Azure costs. It supports cost analysis, forecasting, budgets, and alerting. It also provides cost optimization recommendations.

Visibility

Azure Cost Management provides a view into the costs of the Azure environment. These reports can be analyzed and the subscription or resource group level. They can also be aggregated across subscriptions or filtered using resource tags to create a wide range of views.



Tracking usage and costs trends is provided by the Cost Analysis area of the tool, which provides a report of costs against time. When first used, the report will have no groups or filtered applied, so this shows the all-up cost for the entire Azure environment. The report can be filtered by the various Azure services consumed by this subscription or by groups that you can add. Some examples of groups are departments or applications that you have identified using Azure Tags.

Creating a cost-conscious culture requires visibility and accountability to be pushed down to the engineers responsible for developing and operating the service. Access controls ensure that teams can access the cost management data relevant to them.

Monitoring and Alerting

Monitoring usage and spending is critically important for cloud infrastructures because organizations pay for the resources they consume over time. When usage exceeds agreement thresholds, unexpected cost overages can quickly occur.

Azure Cost Management allows you to alert stakeholders automatically to spending anomalies and overspending risks. You can define alerts based on budgets and thresholds, providing early warning of unexpected expenditure allowing you to take early corrective action.

NAME	SCOPE	RESET P...	START D...	END DATE	BUDGET	CURRENT SPEN...	PROGRESS	
SafetyNet	b1b3fa6f-c9d...	Monthly	10/1/2018	9/30/2020	$2,000.00	$3,459.37	172.97%	...
ExecAlert	b1b3fa6f-c9d...	Monthly	10/1/2018	9/30/2020	$5,000.00	$3,459.37	69.19%	...

CSP Cost Management

Cost Management in Azure was originally provided via the standalone Cloudyn service, which Microsoft acquired in 2017. This service is gradually being replaced by the Azure Cost Management service.

At this time, Azure Cost Management does not support CSP subscriptions, so CSP Partners should continue to use Cloudyn for their cost management scenarios. New Cloudyn registrations are limited to Microsoft CSP Partner administrators only. Existing Cloudyn customers can continue using Cloudyn for a limited period. A roadmap of future changes will enable CSP partners to access Azure Cost Management.

FURTHER READING

The prices of Azure resources vary per region. The third-party tool https://azureprice.net provides cross-region comparisons for each VM SKU. This enables you to choose the most cost-effective region for your workload—especially valuable for workloads without geographical constraints or latency requirements.

Microsoft Azure Support publishes technical guidance on monitoring and troubleshooting performance bottlenecks for Azure VMs.

Principle 8: Scalable

Leverage the agility of the cloud with scalable applications that maximize resiliency, flexibility and cost-efficiency.

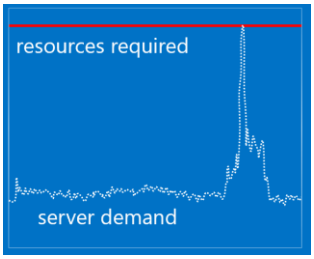
Azure provides almost unlimited scale for storage, compute, network capacity and many platform services. In addition, the ability to dynamically scale the deployment footprint of each application removes the need for large-scale up-front investment and allows you to scale reactively rather than trying to forecast future requirements.

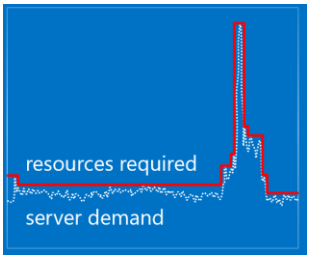
This provides an unprecedented opportunity. New business can start small and local, and grow into large-scale, global applications, without requiring global engineering teams and off-shore facilities.

To take advantage of this opportunity, your application must be architected with scalability in mind. For example, it is desirable to use 'stateless' VMs which offload application state to a separate storage services, such as Azure SQL Database. This makes it easier to 'scale in and out' by adding and removing VMs.

Where this is not possible, you will need to 'scale up and down' by replacing existing VMs with larger or smaller SKUs. This makes it more difficult to avoid application downtime when scaling, since it requires the existing servers to be rebooted. It also introduces inherent limits, since it is not possible to scale up indefinitely.

The ability to scale is closely related to the 'Pay-per-Use' principle, since one of the motivations is to maintain an optimized deployment and avoid cloud waste. But this is not the only motivation. The Scalability principle is also motivated by the ability to grow your business over time and handle unexpected spikes in demand without degrading the service.





Just as with IaaS VMs, most PaaS services support varying the service tier and/or number of instances to enable scaling. This includes Azure App Service, Azure SQL Database, Application Gateway, and more. In some cases, such as Azure Firewall and Azure Load Balancer, scaling is handled transparently by the platform.

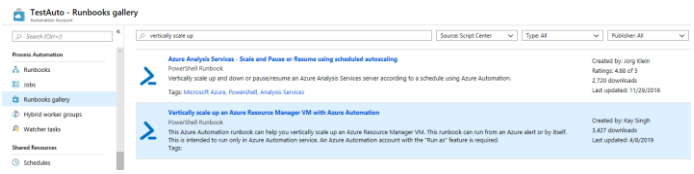
Using the Principle

Use built-in features and custom scripts to automate scale changes and minimize the need for manual interventions.

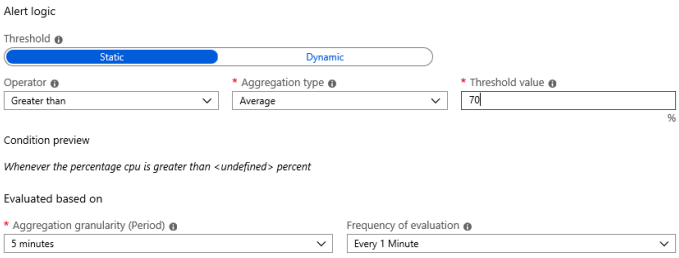
Scaling can be manual, but where possible should be automated. Some services, such as App Service and VM Scale Sets, support autoscaling natively as part of the platform. In other cases, you can implement automated scaling using other platform tools, such as Azure Monitor and Azure Automation.

As an example, consider resizing a VM automatically based on CPU utilization. A similar approach can be used for autoscaling based on an alternative trigger, such as a timer. This guide is based on this example.

To begin, create an Azure Automation account and import the 'Azure Automation Vertical Scale' runbooks from the runbook gallery.



Open the runbook, click 'Edit', and publish the runbook. Then add a webhook to the runbook so it can be triggered from an alert rule. Next, identify the VM to scale and add an alert from the virtual machine settings. Choose 'Percentage CPU' as the alert metric and define the alert thresholds. For the alert action, create an action group that triggers the webhook defined earlier.



Elevated CPU will now trigger the alert, which will call the webhook to start the runbook. This runbook will then scale up the VM. The CPUSTRES tool can be used to simulate high CPU load for testing.

Further Reading

The Azure Monitor Autoscale documentation explains how Azure Monitor autoscale rules can be used to implement autoscaling of VM Scale Sets and App Service deployments. It also explains how autoscale rules can be used trigger custom actions vial webhooks.

An example of autoscaling the number of VMs, without using a VM Scale Set, is given in this TechNet article on autoscaling for Remote Desktop Services hosts.

Summary

Thank you for taking the time to read this playbook. We hope it has given you a valuable insight into best practices for Azure adoption that will enable you to deliver successful, cloud-optimized services.

In this playbook we presented what we have found to be 8 principles of successful cloud adoption. We believe that by applying these principles as you develop your Azure practice, you will avoid many pitfalls and will be better positioned for success.

The cloud changes every aspect of IT, and so as experienced IT professionals we must challenge our experience and assumptions, and be willing to learn and adopt new ways of working. The principles described in this playbook can help you on this learning journey.

These cloud principles are not intended as a recipe, or rigid formula. They are just a starting point that illustrates how to think about how to best develop and operate cloud-based services.

This is not an exact science but an art. The cloud is constantly evolving and changing so the principles will also evolve. The way you apply the principles and learn from your experiences to develop your own unique approach to the cloud will be your IP and your differentiator.

We hope we inspired you to take another look at the cloud and make it yours. Use these principles as a starting point, and go on to create your own cloud principles, so you can become the master of the cloud.

Jan & Herman