AZ104 – Azure Administrator

**Exam Tips:**

* **Core Exam Domains**:
  + IAM
  + Storage
  + Manage Azure Compute resources
  + Virtual Networking
  + Monitor and Backup
* Compounding services together
* 40-60 questions in 120 minutes (700 to pass)

Learning Path: [Exam AZ-104: Microsoft Azure Administrator - Learn | Microsoft Docs](https://docs.microsoft.com/en-us/learn/certifications/exams/az-104)

Learning Paths

# Prerequisites for Azure Admins

1. **Section 1**: Configure Azure resources with tools
   * **Units**:
     + AZ Portal
       - **Features**
         * Search resources/services.
         * Manage resources.
         * Create dashboards
         * CLI Shell
     + AZ Cloud Shell
       - **Features**
         * Requires mount a new or existing Azure Files share
         * Offers an integrated graphical text editor
         * Instant access to your resources.
         * Times out after 20 minutes
         * Requires a resource group, storage account, and Azure File share.
         * Persists $HOME using a 5-GB image held in your file share.
     + AZ PowerShell
       - Module you add to Windows PowerShell or PowerShell Core to connect to your Azure subscription and manage resources through the use of cmdlets
         * **ex**. New-AzVm command to create VM in Azure Sub
     + AZ CLI
       - command-line program to connect to Azure and execute administrative commands on Azure resources
         * controls diff resources such as:

RGPs

SAs

VMs

Az AD

Containers

* + - * Commands in the CLI are structured in groups and subgroups

1. **Section 2**: Use Azure Resource Manager
   * Learning Objectives:
     + - Identify the features and usage cases for Azure Resource Manager.
       - Describe each Azure Resource Manager component and its usage.
       - Organize your Azure resources with resource groups.
       - Apply Azure Resource Manager locks.
       - Move Azure resources between groups, subscriptions, and regions.
       - Remove resources and resource groups.
       - Apply and track resource limits.
   * **Units**:
     + Benefits of ARM
       - Deploy, manage, and monitor all the resources for your solution as a group
         * Repeatedly deploy resources in a consistent state
       - Manage infrastructure through templates
         * Define dependencies between resources
       - RBAC in RGPs
       - Apply tags to resources
         * Clarify billing with tags on RGPs
     + Review Terminology
       - Resource provider (ex. Microsoft.Compute)
         * Resource name = {resource-provider}/{resource-type}
     + Create RGPs
       - **Rules**:
         * Resources can only exist in one resource group & RGP cannot be renamed.

RGP has many different types (services).

RGP have resources from many different regions.

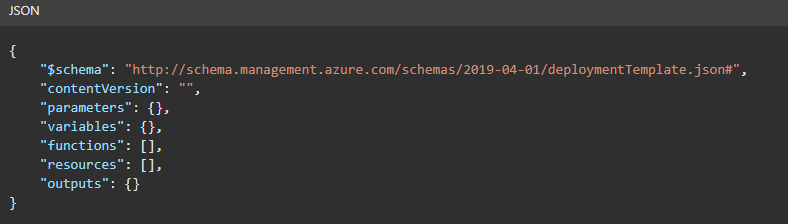
* + - * Factors when making RGP:
        + All the resources should share the same lifecycle. You deploy, update, and delete them together.

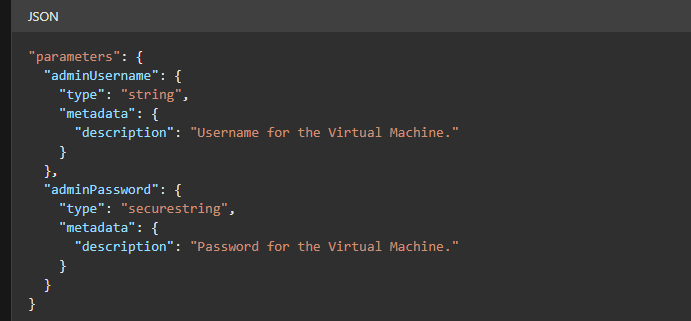
You can add or remove a resource to a resource group at any time.

You can move a resource from one resource group to another group.

* + - * + A resource group can be used to scope access control for administrative actions.
        + A resource can interact with resources in other resource groups.
    - AZ Resource Locks
      * Associate Locks with sub, RGP, resource
      * Locks are inherited by child resources
      * **Types**:
        + Read-only (prevent ANY changes)
        + Delete (prevent deletion)
    - Azure Resource Limits/Quotas
      * Limit to usage of resources (request needed to increase default limit)

1. **Section 3** – *Review of ARM Templates* 
   * **Units**:
     + Benefits
       - Consistent deployments
       - Help to express complex deployments
         * Ex. Dependency mapping (deploy resources in specific order)
       - Reduce manual human deployment errors
       - It's code (IaaC) so uses GIT
       - Promotes reuse of code
       - Linkable templates for more complex deployments
       - Simplify orchestration
     + Schema
       - JSON Document
         * Example code:



* + - Template Parameters
      * 
    - BICEP templates
      * Domain Specific language for Azure resource deployment
      * Used for ARM
      * Process:
        + Submit the Bicep to Resource Manager
        + Converts your Bicep template into a JSON template (Transpilation)
      * Benefits
        + Simpler syntax
        + Modules can be made from the template
        + Bicep auto detects resource dependencies
    - Quick start Templates
      * Azure Resource Manager templates provided by the Azure community

1. **Section 4** – Automate Azure tasks using scripts with PowerShell
   * Units:
     + AZ Portal, CLI and Powershell
       - AZ CLI
         * Via Cloud shell or through local install on OS (ex. Cmd.exe)

**Ex**. command: az vm create

* + - * AZ Ps
        + **Modules** added to Powershell that lets you connect to AZ Sub and manage subscriptions

Ex. New-AzVm

* + - * **Factors to choose Admin Tool**
        + Automation (Az Powershell/CLI)
        + Learning curve & Team skillset
    - Ps cmdlets
      * Command naming conventions:
        + "get" to retrieve data,
        + "set" to insert or update data,
        + "format" to format data,
        + "out" to direct output to a destination
      * Setting Variables
        + $variablename = value
        + New-AzVM
      * Loops
        + The comparison operators:

-lt for "less than",

-le for "less than or equal",

-eq for "equal",

-ne for "not equal", etc.

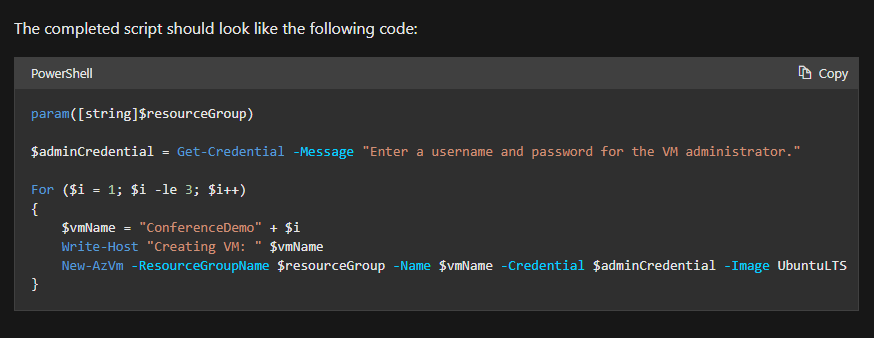
* + - * + For ($I = 1; $I –lt 3; $I++)

{

$I

}

* + - * + Example: Create 3 new VMs



1. **Section 5** – Control Azure services with the CLI
   * Units:
     + What is AZ CLI
       - Command-line program to connect to Azure and **execute** administrative commands on Azure resources.
         * Commands in the CLI are structured in groups (resource) and subgroups.

**Ex**. “storage” group --> subgroups including account, blob, and queue

1. **Section 6** – Deploy Azure infrastructure by using JSON ARM templates
   * Units:
     + Intro:
       - ARM templates allow you:
         * Specify your project's infrastructure in a declarative and reusable way (versioned/saved)
         * Define the infrastructure and configuration for your deployment (declarative syntax)
         * *Idempotent -->* you deploy the same template and get the same resource types in the same state
       - Benefits
         * Resource manager will orchestrate deployment of resources in correct order
         * RM will also validate templates before deployments
         * ARM templates can be integrated into CI/CD tools like AZ Pipeline
     + ARM Template Structure
       - schema --> defines location of JSON schema file that describes structure of resource-provider
       - contentVersion
       - apiProfile
       - **parameters** (defined during deployment)
         * The allowed types of parameters are:

string

secureString

integers

boolean

object

secureObject

array

* + - * variables
      * functions
      * resources (actual items being deployed)
      * output (values returned at end of deployments)

# Manage identities and governance in Azure

1. Section 1: Configure Azure Active Directory
   * Units:
     + Intro
       - Learning Objectives:
         * Identify the features and uses of Azure Active Directory.
         * Define the main Azure Active Directory components such as identity, account and tenant.
         * Compare Active Directory Domain Services to Azure Active Directory.
         * Identify features of Azure Active Directory editions.
         * Identify features and usage cases for Azure AD Join.
         * Identify features and usage cases for Self-Service Password Reset.
     + Describe Azure Active Directory benefits and features
       - AAD is an identity management service/directory for multi-tenant cloud
       - **Benefits**
         * Single Sign-on to cloud/on-prem apps
         * Works across OS
         * Protects on-prem web apps w/ secure remote access

uses MFA, Conditional access policies, and group-based access

* + - * + Protects sensitive data/apps (monitor suspicious activity)
        + Reduces costs and enhances security with self-service capabilities

delegate tasks to employees such as:

reset pwds

create/manage groups

* + - Azure Active Directory concepts
      * **Identity** (an object to auth.)
        + Examples:

User w/ username and password.

Applications that need authentication through secret keys or certificates.

* + - * **Account** (identity that has data associated with it)
      * **Azure AD Account**.
        + An identity created through Azure AD
        + Identities are stored in Azure AD and accessible to your organization's cloud service subscriptions.
      * **Azure subscription**.
        + Division used to pay for AZ services. Many subscriptions linked to 1 credit card.
      * **Azure tenant/directory**.
        + A dedicated and trusted instance of Azure AD
        + 1 Tenant created at subscription creation stage.

More instances of Azure AD can be created.

Tenant = single instance of Azure AD representing a single organization.

Tenant and Directory used interchangeably.

* + - AD Domain Services (on-prem server) vs. AzAD
      * Differences
        + Identity solution – AAD designed for internet-based apps using HTTP/s
        + REST API Querying – AAD can’t be queried with LDAP, only through REST APIs
        + Communication protocol – doesn’t use Kerberos auth. It uses HTTP/s protocols (ex. SAML, WS-Federation, OpenID, and Oauth)
        + Federation Services – AAD has fed services and third-party services (Facebook)
        + Flat structure – AD users/groups are created in flat structure (not as Org units or Group policies)
    - AAD Editions
      * Editions:
        + Free (w/ az sub)

provides user/group management

on-prem directory sync

basic reports

SSO for AZ, Office 365, SaaS apps

* + - * + Microsoft 365 Apps (w/ O365)

additional features

IAM features for O365 apps:

MFA, branding, group access management, self-service password reset

* + - * + Premium P1

additional features

hybrid users can use on-prem and cloud resources

advanced administrative

dynamic groups

self-service group management

MS ID Manager (on-prem IAM)

cloud right back (password reset by on-prem users)

* + - * + Premium P2

additional features

AAD ID protection

Privileged ID management

* + - Implement AAD Join
      * AAD enables SSO
      * AAD Join provides access to org apps/resources and simplifies Windows Deployments
        + **Benefits**

SSO to SaaS apps/services even when not connected to the domain network

Enterprise state roaming of user settings to sync across multiple devices

Users can access MS Store for Business

Windows hello

Restrict access to apps based on policies

Seamless access to on-prem resources when device has line of sight to on-prem domain controller

* + - * Connection Options to get device under control of AAD
        + Register device to AAD

allows you to manage device ID, authentication of device (can enable or disable a device)

* + - * + Join a device (extension of “Register”)

this provides the benefits of registering and changes the local state of the device

by changing the local state, the device can use an org account instead of personal

* + - Implement self-service password reset (SSPR)
      * This property must be specified per user in the AAD tenant directory
        + “selected” option allows you to create groups who have SSPR
      * Authentication methods
        + As AAD Admin, you pick authentication methods

email notification, a text, or code sent to user’s mobile or office phone, or a set of security questions.

1. Section 2: Configure user and group accounts
   * Units
     + LO
       - Configure users accounts and user account properties.
       - Create new user accounts.
       - Import bulk user accounts with a template.
       - Configure group accounts and assignment types.
     + Create User Accounts
       - AAD defines users as:
         * 1) Cloud ID (users exist only in AAD/cloud)

Ex. Admin account and users you manage yourself

Cloud IDs can also be in AAD or external AAD, if user is defined in another AAD instance (these accounts are deleted if primary instance in the directory is deleted)

* + - * + 2) Directory-synced ID

These users come from on-prem AD (Windows Server). Sync activity occurs via AAD Connect to add these users

* + - * + 3) Guest

User outside of Azure which are invited in

Ex. Account from other cloud service or MS accounts like XBOX live

Usually used when external vendor or contractors need to access your AZ resources

* + - Manage User accounts
      * Add users through AZ Portal
      * You can:
        + Add User profile (picture, job, contact)
        + Restore deleted users within 30 days
        + Sign in and audit log info
    - Create Bulk user accounts
      * Use a CSV File and upload it to AAD (must be Global or User Admin)
      * Consider:
        + Naming conventions for user names (ex. Kevin.chan@bmo.com)
        + Passwords (have a standard method to send new user their password (ex. Email to them)
      * \* PShell can also do bulk user uploads
    - Create group accounts
      * 2 Types of groups:
        + Security group

Manages member/computer access to shared resources specific to the group

Ex. Security group for specific security policy (give a set of permissions to members of the group based on need)

* + - * + MS 365 group

Enables collaboration between users with Mail, Files, Sharepoint, etc.

Ex. Allow external users to view your word docs

* + - * 3 Methods to Add members to a group
        + Assigned (add specific user to group)
        + Dynamic User (auto add/remove members based on the changing of a member’s attributes)

Ex. If user’s state changes, they gain/lose permissions

* + - * + Dynamic Device (SG only)

Auto add devices based on their attributes

* + - Create admin units
      * Use admin units in org to restrict the admin scope
      * **Ex**. University has IT teams who control access/users/policies for each Department
        + The Central Admin could create a admin role w/ powers over AAD users for 1 Department for each IT Team

Add users for the admin unit with only the desired users

Add the department’s IT team to the role, along with its scope

* + - * **Considerations**:
        + Use AZ Portal, PS, CLI or MS group to manage admin units
        + AZ Portal requires Global/Privileged role admin
        + Admins units scope to only management permissions

Won't prevent users from using default user permissions to look at other users, group, resources outside their admin unit

1. Section 3: Configure Subscriptions
   * Units:
     + Identify Regions
       - Region = contains multiple datacenters in close proximity with low-latency networks
         * Ex. CACN, CAEA (60+ regions)
       - Info on Regions
         * Azure most global regions
         * Regions allow for flexibility and scale by bringing them closer to users
         * Regions preserve data residency and have compliance/resiliency options
         * User chooses region the resource is deployed in
         * Some resources are region-specific
         * AAD, Az Traffic manager, Az DNS don’t require a region
         * Regions are paired together
       - Info on Region Pairs
         * Physical isolation (300 miles apart from other datacenters of a region pair)
         * Platform-provided replication (geo-redundant storage)

Auto replicate to region pair

* + - * + Region recovery order (1 of the pairs is prioritized during outage)
        + Sequential updates (planned AZ updates roll out to pairs sequentially)
        + Data residency (region pairs will meet the requirements of tax/law of the area)
    - Implement AZ Subscriptions
      * Subscriptions are a unit linked to an AZ account (you bill based on the sub)
      * Subs help you group your cloud service resources
        + Controls how resource usage is reported, billed, paid for

Diff payment setup per subscription

* + - * + Example:

You have a sub based on a testing environment

* + - * Subs are associated with AAD so users need to auth before accessing them
    - Obtaining a sub
      * Methods:
        + Enterprise Agreement

Upfront monetary agreement to add Azure (99.95% SLA)

* + - * + Reseller

Get Azure through Open licensing program (flexible way to purchase cloud services

* + - * + Partners (they will design and implement AZ cloud solutions for you)
        + Personal free account
    - Identify sub usage
      * Tiers
        + Free

Credit for 30 days + 12-month trial

* + - * + Pay as you go

Charges you monthly for services you used in a billing period (used by individuals, small to large companies)

* + - * + Enterprise

Buy services/software licenses under 1 agreement

* + - * + Student

Use Student account to sign up (basically Free tier)

* + - Implement cost management
      * AZ Cost Management shows organizational cost and usage patterns with advanced analytics
        + It reports based on your usage while considering negotiated prices/AZ Hybrid benefit discounts

These usage costs are internal and external (AZ Marketplace charges)

* + - * + Predictive reporting is used to help users understand their spending (and anomalies)
      * AZ CM helps plan/control expenses through:
        + Cost Analysis tool

View aggregated costs/spending trends (monthly, quarterly, yearly)

* + - * + Budget tool

Thresholds on spending can be set

* + - * + Recommendations to optimize spending by identifying underused resources/show less expensive options
        + Export CM data to external tools

Or store cost CSV files in Az Storage

* + - Apply resource tagging
      * Tag resources to organize them into categories (key, value pair)
        + Can be used to tag based on billing data
      * Considerations:
        + 1 resource/RGP can have max 50 tags
        + Tags to RGP aren’t inherited by its resources
    - Apply Cost savings
      * Ways to save
        + Reservations

Pay ahead for a resource (ex. 3 tears for a VM, SQL DB, or CosmosDB, etc.)

Reduce your Pay-as-you-go (PATG) costs by max 72%

Reservations give you billing discounts and don’t affect runtime state of resources

* + - * + AZ Hybrid Benefits

For users w/ licenses with Software Assurance

Maximizes value of existing on-prem Windows Server/SQL server license investments by moving them to Azure (use AZ Hybrid benefit savings calculator)

* + - * + AZ Credits

Monthly free credit to try new solutions (ex. VS subscriber gets free MS Azure)

* + - * + AZ Regions have diff pricing
        + Budgets

Helps you account for the cost of subscribing/using resources for a specific period of time

Monitor spending progress over time

Set budget thresholds to alert you (won’t stop the resource usage however)

* + - * + Pricing Calculator

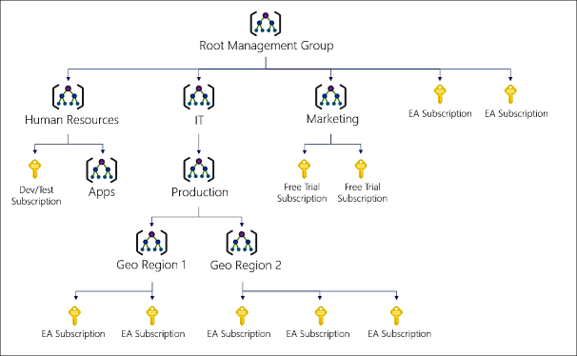
Estimates in all areas of Azure (compute, networking, storage, web, and databases)

1. Section 4: Configure Azure Policy
   * Units:
     + Create Management Groups
       - Management groups are used to group subscriptions which you can then apply your governance policies to
         * They enable:

Org alignment of your subscriptions into hierarchies/groups

You can target policies and spending budgets across subs which inherit down the hierarchy

Set compliance/cost reporting by Departments/Teams

**Ex**. 

* + - * All subs in a management group inherit conditions/policies
        + Ex. All subs and their resources within the SBX management group inherit conditions
      * Can be added through Portal, Pshell, AZ Cli
        + Upon creation, management group has these values:

**Management Group ID** --> the directory unique id that is used to submit commands on this management group (can’t be changed)

**Display Name** (can be changed)

* + - Implement AZ Policies
      * Policies enforce compliance rules over your resources, so they are compliant with standards and service level agreements (scans your resources)
      * Advantages:
        + Real-time enforcement of evaluation/compliance of your resources
        + Apply policies at scale across your organization’s management groups

Aggregate policies in an initiative

Exclude scans based on scope

* + - * + Real-time remediation of resources
        + Useful to govern:

Multiple team/environments

Multiple subs

Standardize cloud resources

Manage regulatory compliance, cost control, security and design

* + - Create Az Policies
      * Steps:
        + Browse Policy Definitions (if, then JSON files)

**Ex**. Prevent VMs from being deployed with public IPs

* + - * + Create Policy Initiative to group your policy definitions

**Ex**. Ensure all Key Vaults are compliant

* + - * + Scope Initiative (exclude based on scope too)
        + View Evaluation/Remediation tasks
    - Determine Compliance
      * Once a policy is set, you can use “Compliance” blade to view # of non-compliant resources

1. Section 5: Configure RBAC
   * Units:
     + Implement RBAC
       - Auth system built on AZ Resource Manager
         * Manages how/what users can access based on their role

Ex. Allow app/users to access resources in an RGP

* + - * Concepts:
        + Security Principal is an object that reps the “thing” trying to access resources

Ex. User, Service Principal Name (SPN), Managed ID

* + - * + Role Definitions are the set of permissions/actions

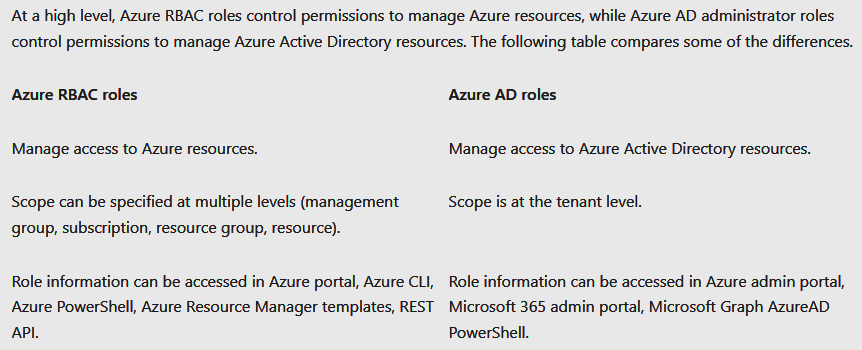
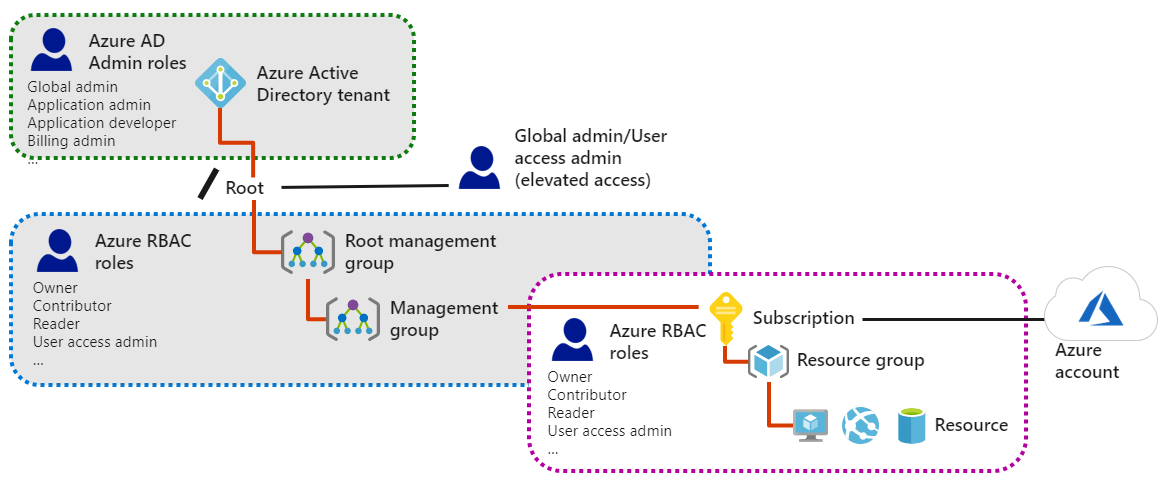
Ex. Contributor, Reader

* + - * + Scope = level of access
        + Assignment = attach role def to a sec. principal at a scope (you can also use “deny assignment” to restrict actions
      * Ultimately, RBAC lets you segregate duties by need
    - Create a role definition
      * A role def’s are a set of permissions in a JSON file (name of the role is based on a use case)
        + Ex. Reader or contirbutor role
        + Example: [Create a role definition - Learn | Microsoft Docs](https://docs.microsoft.com/en-us/learn/modules/configure-role-based-access-control/3-create-role-definition)

Action vs. Not Action (denied permissions)

Scope role def to the management group, subscription, rgp.

\* /subscriptions/[subscription id]/resourceGroups/[resource group name]/[resource]

* + - Create a role assignment
      * Role assignment = the process of scoping a role definition to a user, group, SPN, managed id to grant them access
        + Resource inherits role assginment from parent resource
    - Compare Azure roles to AAD roles
      * 
    - Apply RBAC
      * Azure AD Admin roles are used to manage resources in Azure AD, such as users, groups, and domains.
      * Azure RBAC roles provide more fine-grained access management to Azure resources.
        + 
    - Determine RBAC roles
      * 4 Fundamental RBAC roles
        + Owner

Full access to **all** resources + **giving** access to others.

The Service Administrator and Co-Administrators are assigned Owner role at the subscription scope.

* + - * + Contributor.

Can create/manage all types of Azure resources but **can’t grant** access to others.

* + - * + Reader

Can view existing Azure resources.

* + - * + User Access Administrator

It can manage user access to Azure resources.

* + - * Custom roles can be created with its own specific permissions

1. Section 6: Create Azure users and groups in Azure Active Directory
   * Units:
     + User Accounts in AAD
       - In AAD, user accounts are granted default permissions when first created
         * User account access = type of user + role assignment + ownership of ind. objects
       - Permissions = access rights or a group/user
       - “Roles” have diff permissions attached to them which are to be assigned to a user
       - Types of Roles:
         * Administrator roles

Controls what other users can do (it manages identities/roles of others)

Example tasks:

Create/edit users

Assign admin roles

Reset passwords

Manage licenses

* + - * + Member users

They are native member of AAD org who have default permissions tied to them

**Ex**. New hires are given this account (internal users)

They can’t manage other users

* + - * + Guest user

Restricted permissions (**ex**. Invite external user to collab with you)

You can send a link for them to share your work (**ex**. Sharepoint)

* + - * Adding/Delete Users w/ AZ CLI, PShell
        + az ad user create
        + New-AzureADUser
        + az ad user delete
        + Remove-AzureADUser

\* After deleting user, you have 30-day window to restore (account is suspended during this time)

* + - Manage app and resource access by using Azure Active Directory groups
      * Use org groups to manage your cloud, on-prem apps, and resources
        + Resources can be in AAD org or external (ex. SaaS apps, Sharepoint, on-prem, etc.)
      * Access Management in AAD
        + AAD roles --> manages AAD resources (users, groups, licensing, app registration)
        + RBAC --> manages AZ resources (VMs, SQL DB, SA, KV)
      * Diff assignment types to provide access rights
        + Direct --> assign 1 user a role
        + Group --> assign a group to a role
        + Rule-based --> use rules to determine a group membership based on user or device properties

A user/device’s account must meet the group membership requirements

1. Section 7: Secure your Azure resources with Azure RBAC
   * Units:
     + What is Azure RBAC
       - 2 Concerns of IAM
         * Ensure removal of access from outgoing-users
         * Balance autonomy and central governance

Ex. Giving teams power over managing a VM, but no control over the network

* + - * AZ subscriptions
        + 1 subscription associated with 1 AAD directory

Users/groups in that directory can manage resources of that subscription

* + - * + Azure AD Connect lets you extend on-prem Active Directory to the cloud

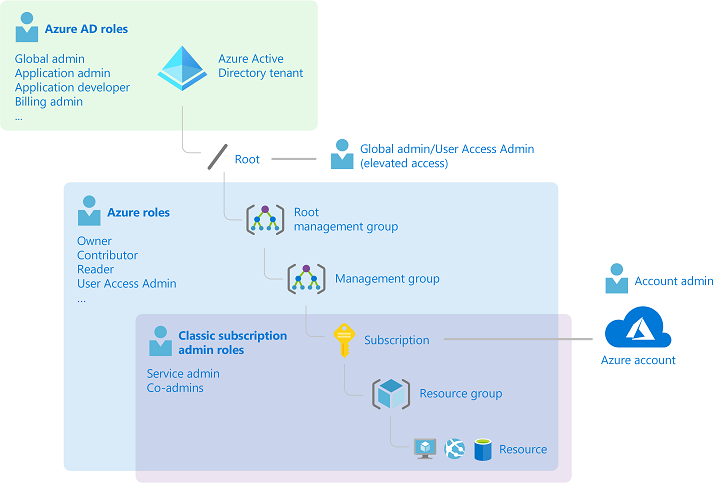
Ex. Employee uses existing work id to manage AZ subs

If on-prem AD account is disabled, they lose access to Azure subs connected through AD Connect

* + - * Azure RBAC
        + Is an authorization system built on AZ Resource Manager (ARM)

Let's you grant exact accesses to your users based on need

* + - * + Diagram of how you can scope roles with RBAC (role assignments)



Child scopes inherit the parent’s role

* + - * How does RBAC work
        + You create role assignments for your users, groups, apps

3 elements of a role assignment

WHO: Security Principal (SPN)

The user, group, app

WHAT: Role definition

The set of permissions (ex. Contributor)

Owner - full access + able to give out access

Contributor – create/manage resource, but can’t grant access

Reader - view resources

User Access Administrator - Lets you manage user access to resources

\*Custom Role

WHERE: Scope

Location (sub, management group, RGP, resource)

1. Section 8: Allow users to use AAD self-serve to reset password (SSPR)
   * Units:
     + Self Service Pwd Reset
       - Allows you to reset your password
     + How it works
       - Portal checks the browser local settings and renders SSPR page to appropriate language
       - Verify username and uses Captcha
       - Authenticates user
         * Types: \*free tier doesn’t have phone call options

Security questions

Mobile app code

Email

Mobile phone

Office phone

* + - * + Admin specifies which methods are used together
        + **Recommendations**

2+ auth reset request methods.

Mobile app notification or code as the primary method

The mobile phone method isn't a recommended method (SMS fraud)

Security question least recommended method

* + - * Password resets
      * Notification sent to show password change
        + 2 Options

Notify the user on pwd reset

Notify admins when other admins reset their passwords

# Implement and manage storage in Azure

1. Section 1: Configure Storage Accounts
   * Units:
     + Implement AZ Storage
       - Attributes
         * Durable/highly available

Data redundancy ensures data is safe in the event of hardware failure/unexpected outage (replicated across regions and data-centres to protect against natural disasters)

* + - * + Secure/encrypted data
        + Scalable

Scales to meet data storage and performance needs

* + - * + Azure manages your hardware maintenance, updates, and issues for your SA’s
        + Accessible through HTTP/s

Supports multiple language SDKs/API integration as well as Powershell and AZ CLI

Portal can also be used to work with your data

* + - * 3 Categories of Az Storage
        + 1) Storage for VMs

Disks = persistent block storage for VMs (IaaS)

Files = fully managed file shares in cloud

* + - * + 2) Unstructured data

Blobs

Highly scalable, REST-based cloud object storage

Data Lake Stores

Hadoop Distributed File System (HDFS) as a service

* + - * + 3) Structured data

Tables (key/value, autoscaling NoSQL)

CosmosDB (global DB)

Az SQL DB (DaaS on SQL)

* + - * 2 Tiers of General SAs
        + Standard

Uses HDD and low cost per GB

Used for bulk storage/not accessed often

* + - * + Premium

Uses SSD, low-latency performance

Used for VM disks w/ intensive apps (Dbs)

* + - Explore Az Storage Services
      * **Blob** storage containers

Used for massive amounts of unstructured data (text/binary)

* + - * + Use Cases:

Serve image/docs to browser

Store files for distributed access

Stream video/audio

Store backups to be restored later or archived

Store data to be used by on-prem/Azure services

* + - * + Accessed through HTTP/s (SA’s API)
      * Azure **Files**
        + Used for network file shares for cloud or on-premises deployments.

The shares use SMB and NFS protocol to let multiple VMs share the same files with read/write access

Can also access through REST API

* + - * + The URL of the file includes a SAS token that is accessible globally
        + Use Cases

On-prem apps can use Azure File shares (helps with migration)

Config files, DevTools can be store and accessed by VMs

Diag logs can be saved and accessed here

* + - * Azure Queues
        + Stores messages (64kb) between app components asynchronously.
        + Use Case:

Cx wants to upload pics and you want to create thumbnails of the them

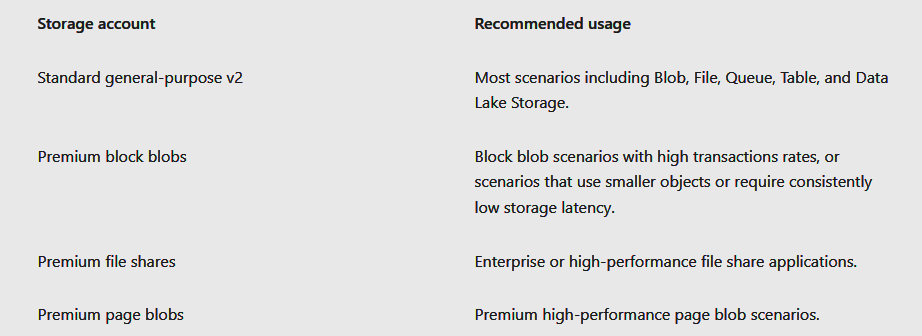
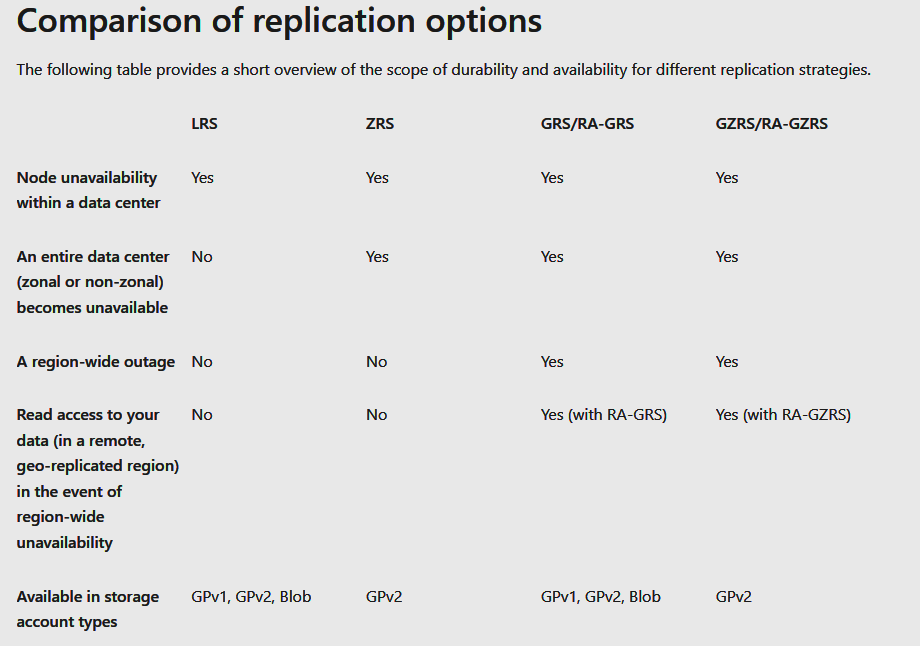
When Cx finishes upload, a message is sent to the queue which triggers the thumbnail creation

Each part is separate (more scalable and controllable)

* + - * Azure Tables
        + A NoSQL store for schemaless storage of structured data.

Now part of CosmosDB

Stores structured, non-relational data

* + - Determine SA kinds
      * Diagram:
        + 
    - Determine Replication Strategy
      * Diagram
        + 
      * **Local Redundant Storage** (LRS)
        + Lowest-cost option, but least durable (replicas might be lost/unrecoverable)
        + Use Cases:

Data can be easily reconstructed

Data is constantly changing

App is restricted to replicating data only in 1 country (data gov laws)

* + - * **Zone Redundant Storage** (ZRS)
        + Syncs replication across 3 storage clusters in 1 region

Each cluster is physically separated in its own availability zone w/ autonomous, separate utilities/networking capabilities

Ensures access to data if zone becomes unavailable (performance and low-latency)

Not avail. In all regions

Changing from ZRS to other option requires physical data movement

* + - * **Geo-redundant** (GRS)
        + Replicates data to secondary region to protect against regional outage (99.99% durable)

Data can be recovered even in complete regional outage/failure

How it works:

First replicated with LRS, then replicated in GRS secondary region which also LRS replicates

Primary and secondary regions are acros separate fault domains and upgrade within a storage scale unit (basic replication unit in the datacenter)

Two options of GRS

GRS

Replicate data to other data center in second region, but only in read-only if MS initiates failover from primary to second region

Read-access GRS

Replicates data from prim to second region, and also lets you read from second region (doesn’t need the MS failover to trigger this option)

* + - * **Geo-zone Redundant** (GzRS)
        + Combines Zone-redundant w/ regional outage protection of Geo-redundant

Data replicated in 3 availability zones in both Primary and Secondary regions

Each region is also paired

You can still read/write even if availability zone is unavailable/unrecoverable (99.99% durable)

Same scalability as the other options

Option to read data in second region like RA-GRS) --> used in region disaster

* + - * + Used for applications requiring consistency, durability, high availability, excellent performance, and resilience for disaster recovery
    - Access Storage
      * All objects stored in Az Storage has unique URL
        + Default Endpoint

//{object name}.{storage type}.core.windows.net

Ex. //kevstorageaccount.blob.core.windows.net

* + - * + Objects location = {default endpoint}/mycontainer/myblob
      * Custom domain can be made to access BLOB storage
        + 2 Options

1) Direct CNAME mapping

CNAME record

blobs.contoso.com

Replaces this target

contosoblobs.blob.core.windows.net

2) Intermediary Mapping w/ asverify

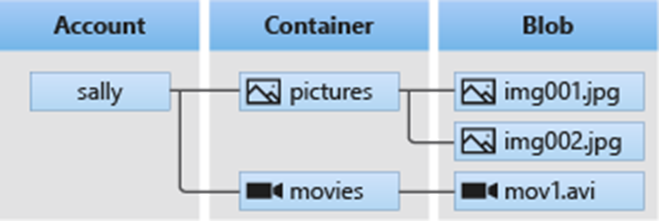
To avoid downtime as domain updates

Prepend “asverify” to let Azure recognize custom domain without modifying dns during the mapping process

Ex. Asverify.blobs.contoso.com

* + - Secure SAs
      * Use Firewalls/Virtual Networks
        + This restricts access to SA from specific Subnet/VNETs/Public Ips
        + The subnets/VNETs must exist in the same region or is a region pair

1. Section 2: Configure blob Storage
   * Units:
     + Implementing blob storage
       - Usually used to store text/binary (ex. Documents, images/video/audio, app installation files, backups, data used for analysis)
       - 3 Parts of Blob Storage resources
         * 1) Storage Account
         * 2) Container inside SA (**categorizes** that it holds)
         * 3) Blobs inside containers (the file **itself**)



* + - Create Blob containers
      * Container = grouping of a set of blobs (files)
        + You can have unlimited containers w/ unlimited blobs (blobs must be in containers)
      * **Components**
        + Name (lowercase, numbers, hypens)
        + **Public Access Levels**:

\*\*\* **Default** setting = private

Use **private** so prevent anonymous access

Use **blob** to allow anon-public access to blob

Use **containers** to allow anon-public read/list to entire container

* + - Create Blob Access Tiers
      * Tiers are based on usage patterns
        + **Hot**

Frequent access to objects in SA (DEFAULT)

* + - * + **Cool**

Infrequent access to objects and stored for 30 days

Cost effective if you don’t need to access the data often

* + - * + **Archive**

For data that can tolerate several hrs of retrieval latency (remains archived for 180 days)

Most-cost effective if you’ll rarely use the data

* + - Blob Lifecycle management rules
      * Blob Lifecycle policy is used to change your data’s to optimized access tiers/delete it based on data’s lifecycle
        + *Use Cases*:

**Change** blobs to cooler storage to optimize performance/cost

**Delete** blobs at end of lifecycle

Define rules to be run per day at SA level

Apply rules to containers and blobs

* + - Determine blob object replication
      * Based on rules you set, **object replication** asynchronously copies block blobs in a container (its versions, metadata and properties) to another container in *different region*
      * Scenarios
        + **Minimize latency**

Object replication reduces latency for user read requests by copying data into a region closer to the user

* + - * + **Increase efficiency** for compute workloads because they can consume same set of block blobs in a diff region
        + **Optimize data distribution**

You can process/analyze data in 1 region, then only copy over the results to another region

* + - * + **Optimize costs** by archiving copied data using life-cycle management policies
      * Considerations for Object Replication
        + Blob **versioning** must be enabled on both source/destination accounts
        + Doesn't support blob **snapshots** (won’t replicate those)
        + Source and destination accounts can be in **diff tiers** (hot or cold only)
        + Configuring object replication creates a replication policy that specs source/destination account

The **Rep Policy** has 1+ rules that spec the source/destination container and specs the which block blobs are going to be replicated

* + - Upload blobs
      * 3 Types of blobs
        + 1) *Block*

Blocks of data that make 1 blob

Ex. Text/binary data like files, images, video

* + - * + 2) *Page*

Up to 8 TB and used for frequent read/write operation

VMs use page blobs as OS and data disks

* + - * + 3) *Append*

Like block blobs, but specified for “append operations”

**Ex**. Diagnostic log data

* + - * Blob upload tools
        + 1) *AzCopy*

CMD tool in Win/Linux to copy data across containers/SAs

* + - * + 2) *AZ Storage Data Movement Library*

A .NET library for moving data

* + - * + 3) *AZ Data Factory*

Using an Account key, SAS token, SPN, or Managed ID to copy data

* + - * + 4) *Blobfuse*

Virtual file system drive for AZ Blob. Let's you access existing blob storage through Linux file system

* + - * + 5) AZ Data Box Disk

Service to transfer on-prem datato Blob storage

Used when dataset is massive or has network constraints that make uploading hard

This service lets you request SSDs that you can copy your data to then ship to MS to be upload to Blob storage

* + - * + 6) AZ Import/Export

Let's ou export data from SA to HDD which you then give to MS which ships back to you with your data

* + - * + 7) AZ Storage Explorer App
    - Storage Pricing
      * Models
        + Performance tiers

Storage tier determines amount of data stored & the cost to store it

Cooler tier = per-GB cost decreases

Data access costs

This cost increases as the tier gets cooler

Cool/archive tiers get charged higher per-GB data access

Transaction costs

Charge increases as tier gets cooler

Geo-replication data transfer costs

GRS/RA-GRS configured accounts get charged per-GB

Outbound traffic (exiting AZ region)

Billing for bandwidth usage based on per-GB basis

Consistent with general-purpose SAs

Changing storage tier

Changing from **cool to hot** incurs a charge equal to reading **all** the data existing in the SA

But, hot to cool only charges equal to writing data into the cool tier (GPv2 only)

1. Section 3: Configure Storage Security
   * Units:
     + Review of Storage Security Strategies
       - AZ Storage security capabilities:
         * **Encrypted data** (SSE)
         * **Authentication** (AAD and RBAC)

You can assign RBAC roles for SA to SPNs and use AAD to authorize resource management operations (ex. Key management)

AAD integration is supported for data operation on blob/queue services

* + - * + **Data in transit** between app and Azure

Secured by Client encryption, HTTPs, SMB

* + - * + **Disk encryption** of OS and VM data disks

Encrypted with Azure Disk Encryption

* + - * + **SAS tokens** can be used to grant access to data objects in Storage
      * Auth Options
        + Options to ensure only authorized sources can request against secured storage resources

**AAD** (IAM service)

Assign fine-grained access to users, groups, apps based on RBAC roles (ex. Contributor)

**Shared authorization key**

Using access keys on your account, to produce an encrypted signature that is passed with the request

**Shared access signature (SAS)**

Delegates access to a resource with specific permissions for a time interval

**Anon access to containers/blobs**

Make them public to any user without authentication

Read requests to public containers don’t need auth.

* + - * Create SAS
        + SAS is a URI that grants restricted access rights to AZ Storage resources for a specific time interval which you can share with a client

**SAS gives you control of the type of access you grant:**

Account-level SAS = delegates access to multiple storage services (blob, file, queue, table)

Service-level = delegates to only 1 type of storage service

Interval where SAS token is valid can be changed

Specific permissions (ex. Read/write, but not Delete)

Specify which IP addresses that the SA will accept SAS from

Ex. Only IP addresses belonging to your organisation

Restrict the protocol which Storage will accept SAS from

Ex. Only let clients using HTTPs to use SAS

* + - * Identify URI and SAS parameters
        + URI consists of {**storage**-**resource**}/{**SAS**-**token**}

**Ex**. https://myaccount.blob.core.windows.net/?restype=service&comp=properties&sv=2015-04-05&ss=bf&srt=s&st=2015-04-29T22%3A18%3A26Z&se=2015-04-30T02%3A23%3A26Z&sr=b&sp=rw&sip=168.1.5.60-168.1.5.70&spr=https &sig=F%6GRVAZ5Cdj2Pw4txxxxx 

* + - * Determine Storage Service Encryption
        + AZ Storage Service Encryption (**SSE**) --> protects data by ensuring your org security and compliance standards are met

Auto encrypts your data before adding it to Az-managed disks, blobs, queues, table, or files (decrypts before retrieving)

All transparent to user and encrypted on 256-bit AES encryption (strongest cipher)

* + - * Create Cx Managed Keys
        + Key Vaults manage/store/generate your encryption keys

Keys can be created, disabled, audited, rotated, and have defined access controls

These keys can be used with SSE, but the SA and KV must be in the same region, tho different subs

* + - * Apply Best Practices
        + **Risks of SAS**

If SAS is compromised, anyone can use it

An app that uses SAS, but can’t get a new one before it expires, is locked out (hinders its work)

* + - * + **Recommendations to reduce risk**

Always use HTTPs to create/distribute SAS tokens

Reference stored access policies (if possible)

These policies let you revoke permissions w/o regenerating storage account keys (set the expiration date far out)

Use near-term expiration time on “unplanned SAS”

Ensures key is only valid for short amount of time (can prevent long operations)

Clients should renew SAS automatically before expiration dates

Set SAS start/end times at least 15 minutes in the past

This ensures that clock skew doesn’t occur

You can also not set these times

Be specific with what needs to be accessed

Provide minimum permissions

Your account will be billed for all usages (including actions done by users with SAS)

Validate data being written by SAS operations

SAS is not always the right choice

Use Storage Analytics to monitor apps

Logging and metrics to check for spikes in auth failures due to outage in SAS or removal of stored access policy

1. Section 4: Configure Azure files and Azure File Sync
   * Units:
     + Compare files to blob
       - File storage is used by VMs/cloud services/on-prem apps to share file data across app components via mounted shares
         * **Common uses of “file” storage**:

**Replace/supplement** traditional file servers/NAS devices

OS (Windows, Mac, Linux) can directly mount Az File shares to be **accessed anywhere** in the world

Az Files makes it **easy to move** apps to the cloud that need file share to store app/user data

**Az File Sync** lets you replicate file shares to cloud/on-prem for performance and distributed caching of the data being used

**Store shared app settings** and config files

**Share diagnostic** logs, metrics, etc. In a centralized location

Store **tools/utilities** for developing VMs or cloud services

* + - * Files vs. Blobs (when to use)
        + Chart:
        + 
        + Other comparisons

Az Files are true directory objects; Az Blobs are flat namespace

Az Files accessed through file shares; Az Blobs are accessed through containers

Az Files provide shared access for multiple VMs; Az Blobs exclusive to 1 VM

* + - Manage Files shares
      * You need to store your file share in a SA
        + Map files shares (Windows)

Simply connect file share to Win server in VM (use Port 445)

* + - * + Mount in Linux

Create entry in /etc/fstab and connect file share there

* + - * + Secure transfer required

Only allow requests to SA through secure connection (ex. HTTPs)

* + - Create files share snapshots
      * Snapshots capture point in time of your file share
        + Retrieve snapshots at the individual file level to restore individual files

These snapshots are taken incrementally (use most recent to restore share)

* + - * Use cases:
        + Ensure protection against app error and data corruption

Apps use the file share are constantly writing, reading, processing the data so bugs can occur

Take a snapshot before deployment of new app code to ensure that bugs introduced by new code don’t impact your previous versions of the file share data

* + - * + Protect against deletion/unintended changes (undo)
        + General backup purposes (ex. Audit or disaster recovery)
    - Implement file sync
      * Az File Sync (AFS) centralizes org’s file shares (similar to an on-prem file server)
        + Changes Windows Server into a quick cache of file shares
      * Advantages of AFS:
        + **Ease in “lift/shift”** applications that need access to Azure and on-prem systems

Provides write access to the same shared data across Windows Servers and Azure Files

* + - * + **Branch offices** need to backup files & you can setup a new server that connects to Az Storage
        + **Backup/Disaster Recovery** through Az Backup

Az Backup backs up on-prem data which can retore file metadata or recall data after disaster

* + - * + File archiving of non-used data (Cloud Tiering)
    - Identify file sync components
      * Definitions:
        + **Storage Sync Service** (Top-level resource for AFS)

It is peered to the SA resource and can be deployed to RGP

Distinct top-level resource from the SA resource is needed since SSS can create a sync relationship with multiple SAs via multiple sync groups

Multiple SSS per sub

* + - * + **Sync group**

Defines sync topology for a set of files (ex. Accounting)

Cloud Endpoints within a sync group are kept in sync with each other

**Ex**. If you have 2 sets of files to manage with AFS, you need two sync groups with their own endpoints

Multiple sync groups per SSS

* + - * + **Registered server**

A relationship between Server and SSS (many servers to 1 SSS instance)

* + - * + **AFS Agent** (downloadable package that lets Win Server sync with AFS)

**3 Components**

*FileSyncSvc.exe* --> background monitors changes to server endpoints to initiate sync sessions to Azure

*Storage sync* --> file system filter that determines the tier of files to Az Files

PShell management cmdlets

Lets you interact with the Microsoft.StorageSync Azure resource provider

* + - * + **Server endpoint**

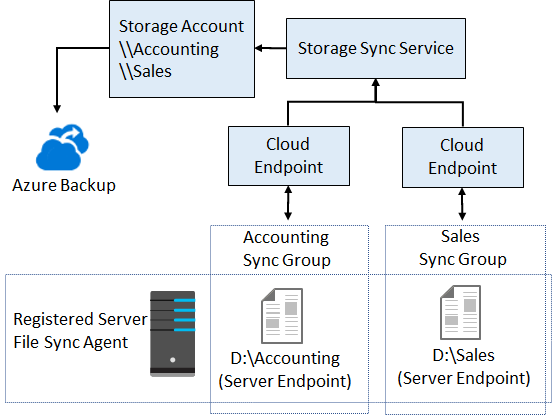
A specific location on a registered server (ex. Folder on server)

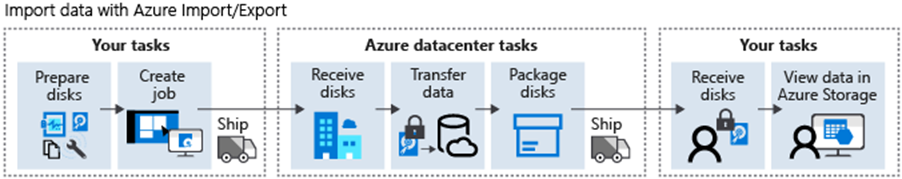
Multiple server endpoints can exist if namespaces do not overlap

* + - * + **Cloud endpoint**

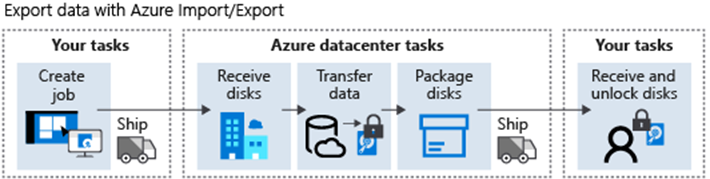
Is a File share that is part of a sync group

1 Az File share per Cloud point and its sync group

* + - * **Diagram**:
        + 
    - Deploy Azure File Sync
      * Steps:
        + 1) Deploy SSS from Az Portal and provide name, sub, RGP, location
        + 2) Prep Windows Server to use AFS (temp disable Internet Explorer Enhanced Security and get latest PShell version)
        + 3) Install AFS Agent
        + 4) Register Windows Server with SSS

1. Section 5: Configure Storage with Tools
   * Units:
     + AZ Storage Explorer
       - Windows app that you connect to and can access internal and external SAs
         * Access external SAs through Connection access key
     + AZ Import/Export
       - Used to securely import large amounts of data to Blob storage and Files storage by shipping disk drives to Az Data-center
         * Can also be used to move data to on-prem sites
       - **Use cases**
         * Migrate massive amounts of data to Azure
         * Quickly distribute content to customer sites
         * Backup on-prem data to Az blobs
         * Recover data from blobs and move to on-prem location
       - **Import**:
         * 

Use WAImportExport tool for copying

* + - * **Export**:
        + s
      * Import/Export Tool (WAImportExport)
        + Let's you prepare the drives you will send and also repairs the drives when they return

Uses:

Copy data to the hard drives you are shipping out

Repair returning drives

* + - AzCopy
      * Cmdlet to copy data between file systems and SAs, or between SAs
        + Can be run as a background job so ideal for long transfers
      * Features:
        + Supports Az Data Lake Storage Gen2 APIs
        + Supports copy entire account to another account
        + List/remove files and blobs in a path
        + Supports wildcard patterns
        + Every AzCopy instance creates a job and a log file
      * Auth Options
        + AAD

Use *.\azcopy* login to sign in using AAD

* + - * + SAS tokens can be appended to the blob path
      * Basic syntax:
        + azcopy copy [source] [destination] [flags]

1. Section 6: Create Az SA
   * Units:
     + Deciding how many SAs are needed
       - What are Az Storage?
         * 4 services: Blobs, Files, Queues, and Tables
       - **Storage Accounts** 
         * The container that groups a set of these services together (part of a RGP).
         * *Settings*:

Subscriptions.

Location.

Performance (determines data services in your SA and the HDD used to store the data)

Standard

Lets you use the 4 services and uses magnetic disk drives

Premium

Additional services (block blobs, append blobs, specialized file storage for premium file shares) and uses SSD

Replication

The strategy used by the SA to replicate their data against disasters

Minimum = 3 copies of data

Minimum replication = Locally Redundant (LRS)

Doesn’t account for total data-centre failure

Can be upgraded to GRS

Access Tier

Hot, cool, archived tiers for how often blobs can be accessed in the SA

Secure transfer options (HTTPs is needs to be enabled)

VNETs

Security feature to allow inbound only from specified VNETs

* + - * + How to determine when to create a new SA.

**Factors**:

Data diversity.

Could be due to reasons such as:

Region specific data.

Sensitive information

Proprietary vs. Public data

Deparment

Cost sensitivity.

Settings chosen for SA changes cost (ex. LRS/GRS, Hot Tier, Premium performance)

Tolerance for management overhead.

Each SA requires time/attention from an administrator

Complexity of the data itself, per SA also impacts

* + - * Choosing Account Settings
        + **Main 3 Settings**

**Name**

**Deployment Model** (system Azure uses to organize your resources – defines how API will create, maintain, configure the resources

1) ARM template (current)

2) Classic (legacy that uses Az Service Management API)

\*\*\* SA, VM, VNETs can be deployed with either model

**Account kind** (set of policies that determine which data services are included)

3 Kinds of SA

**StorageV2** (general purpose v2)

*Current* and *recommended* offering that supports ALL storage types AND features

**Storage** (v1)

*Legacy* offering that supports ALL storage types BUT not all features

**Blob storage**

*Only* offers block blobs and append blobs

* + - * Choosing a Creation Tool
        + Tools:

Az Portal

Az CLI

Az Pshell

Management client libraries (incorporates creation into a client app)

* + - * + Criteria to choose

1 time = Portal

Automation = Script vs. API

scripts are faster

If you have an existing client app, management libraries might be better (needs IDE)

1. Section 7: Control access to Azure Storage with shared access signatures
   * Units:
     + 4 Authorization options for Azure Storage (w/ clients over HTTP/s)
       - **1) Public Access**
         * Anons have public read-access to your blob files in containers
         * **2 Separate settings that impact public access**

**Config at SA level** with “*AllowBlobPublicAccess*” is true setting

Blob data is only public accessible if the container’s settings are also set to public access

**Config at Container level** (anon access only available if SA has also allowed this

*2 Types of Container settings*:

1) Public read access for **only** blobs

2) Public read access for container **and** its blobs

\*\*\* Anon access is controlled at container level

Both required to enable anon-public access (don’t need to share keys or SAS)

* + - * 2) AAD
        + Secures Az Storage w/o storing credentials in code

2 step approach

1) Auth a SPN that returns Oauth 2.0 token

2) Pass the token to Az Storage to enable authorization to the resource

\*\*\* used for running app w/ managed identities or SPN

* + - * 3) Shared key
        + Az Storage creates two 512-bit access keys for each SA to grant client access (root access)

Store these keys in Key Vaults (KV) which can rotate these keys

* + - * 4) SAS
        + Grants granular access (read-only, read-write, expiration time) to the resource

3 Types of SAS

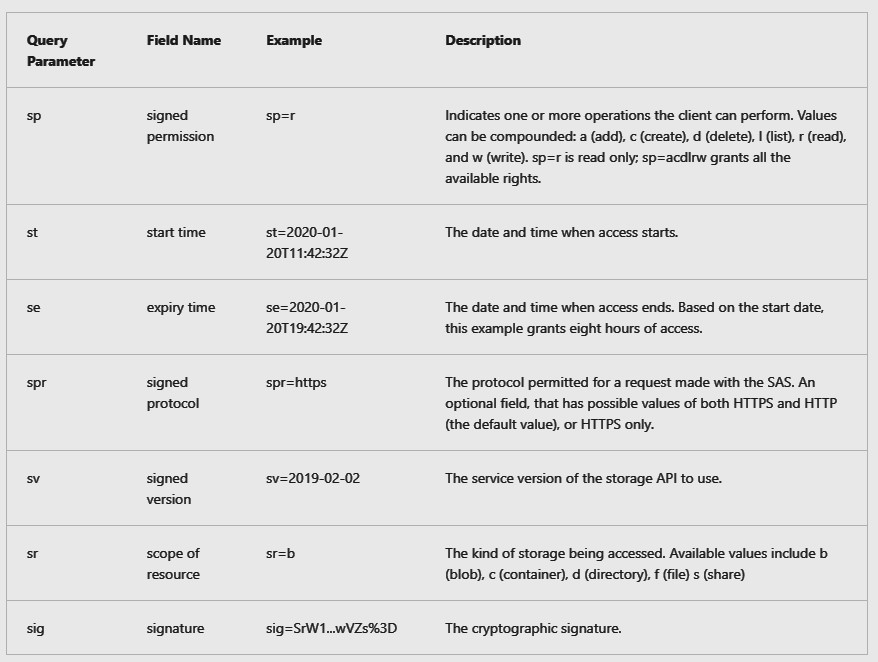
User delegation – used for blobs and secures w/ AAD credentials

Service – secures using SA key (delegates access to the 4 storage services b, q, t, f)

Account – also secures using SA key (same access delegation)

but also controls service-level operations like (Get service stats)

* + - * + You can associate Service SAS with Stored Access Policy which controls start time, expiration time, and permissions (lets you control expiration/revocation of SAS
    - Use SAS to Delegate Access to Az Storage
      * SAS has 2 Components
        + 1) URI pointing to SA(s)
        + 2) Token indication how resource is accessed



* + - Use stored access policies to delegate access to Azure Storage
      * Stored Access Policies are used with the 4 storage service types (blob, file, queue, tables)
        + They can be assigned to either the container or individual blobs inside it
        + 4 Properties

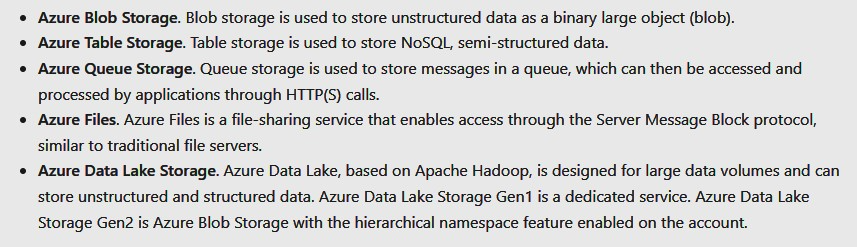
Id = name of stored access policy

Start-time = value of the date/time when policy starts

Expiry-time = value of date/time when policy expires (requests to SA will fail w/ 403 error)

Permissions = string of permissions in “acdlrw” (ex. read/write)

1. Section 8: Upload, download, and manage data with Azure Storage Explorer
   * Units:
     + Connect Azure Storage Explorer to a storage account
       - Az Storage Explore is a GUI app to access, manage SAs’ and their containers’ data
         * **ex**. Blobs, Tables, Queue, Files, Data Lake Storage



* + - * Connection Types
        + 1) Add Az account w/ AAD

This connection type accesses the data layer (only can create Data Lake blob or standard blob containers

* + - * + 2) Use SAS URI

This connection let’s you use the SAS URI of the SA you wnat to access (bqtf)

* + - * + 3) Via SA name and Key

You need to get the correct access key of the specific SA

* + - * + 4) Manage Data Lake Storage Gen1

You need the URI of the data lake (you can access resources even if they are not in your subscription)

# Deploy and manage Azure compute resources

1. Section 1: Configure VMs
   * Units:
     + Review cloud services responsibilities
       - **IaaS business Scenarios using VMs**
         * *Test/Dev* can be done quickly since the environment can be creating/dismantled in the VM

Scaling can also be done easily

* + - * + *Website hosting*.

Running websites on VMs is cheaper

* + - * + *Storage, backup and recovery* expenses can be avoided

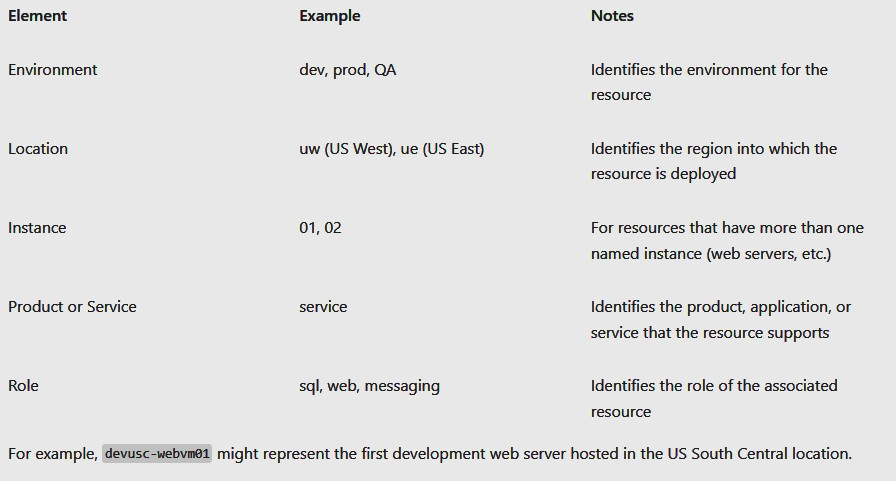
VMs can manage demand and growing storage needs

* + - * + *High-performance computing* to solve complex problems
        + *Big data analysis*
        + *Extended data center*

Avoid the need for physical hardware/space by using Azure VMs

Or, connect physical network to Azure

* + - Plan/provision VMs
      * 1) Consider the VNETs that will provide connectivity between VMs and other cloud resources
        + Also, allow access to external/on-prem services through the VNETs
      * 2) Name the VM
        + Good conventions:



**ex**. devcc-web01

* + - * 3) Determine Location
        + Consider:

Location can limit options (**ex**. Hardware and configurations)

Price difference per region

* + - * 4) Pricing Options
        + 2 Types of Costs:

A) Compute

These are the usage costs (priced per hr, but billed per minute)

OS and VM size are also considered due to licenses

B) Storage cost

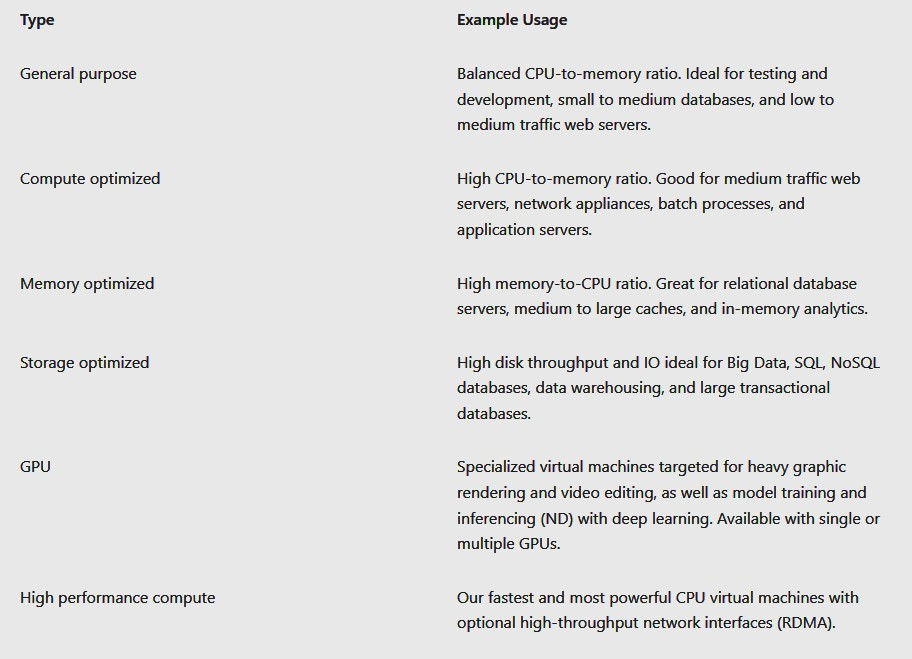
Storage used by disks have two payment options

*Consumption-based*: pay for compute capacity per second (best for short-term/unpredictable workloads)

*Reserved VM instances*: RI allows for advanced purchasing of VM for 1-3 yrs

Get up to 72% savings (can be exchanged/returned for a fee)

(best for VMs that you can predict/run long-term)

* + - Determine VM size
      * Workload Options:
        + 
    - Determine VM storage
      * All Az VMs use 2+ disks (OS disk and Temp disk)
        + OS Disks:

This disk is pre-installed with an OS and registered as a SATA drive as C:

* + - * + Temp Disk

For non-critical data (used during maintenance event or VM redeployment)

On Windows, disk is labelled D: drive and stores pagefile.sys.

On Linux, disk is labelled /dev/sdb and is formatted/mounted to /mnt by the Agent.

* + - * Other disks:
        + Data disk (stores app data/other)

They are SCSI drives

VM’s size determine how many of these disks can be attached

* + - * + VM storage options

Az Premium storage (SSD) is used for I/O intensive workloads

Multiple premium storage disks can be attached to a VM

256 TB per VM

80000 I/O operations per second per VM

Disk throughput 2000 MB/sec

* + - * + Managed disks

Azure Virtual Hard Drive (VHD)

Stored as page blobs (random IO storage objects)

“managed” because it is an abstraction over page blob, blob containers, and Az SAs

Az creates and manages these disks for you (SSD, premium SSD, Standard SSD, Standard HDD)

* + - Creating VMs in the portal
      * Select OS image
      * Additional Config
        + Basic (proj details, admin account, inbound port rules)
        + Disks (OS disk type, data disks)
        + Networking (VNETs, load balance)
        + Management (Monitoring, Auto-shutdown, back up)
        + Advanced
    - Connect to VMs
      * Windows --> RDP
      * Linux --> use SSH w/ PuTTY
      * Bastion
        + A PaaS service that is provisioned in a VNET
        + Provides secure RDP/SSH to VM from portal over SSL (VMs won’t need public IP when using this connection type)

VMs must be in the specific VNET

* + - Connect to Windows VM
      * RDP (Powershell --> Get-AzRemoteDesktopFile)
        + GUI based
      * Windows Remote Management (WinRM)
        + Cmdline session
        + Can be used to run non-interactive Powershell scripts

You can use certificates with WinRM

Steps:

1) create KV

2) Create SSL certificate

3) Upload certificate to KV

4) Id URL of certificate in KV

5) Reference the URL in VM’s configuration

\*\*\* Port 5986

* + - Connect to Linux VM
      * SSH public key/password required to authenticate
      * SSH Connections:
        + Utilize “public-private key pair” == SSH key

Public key = placed on Linux VM

Private key = placed in local system

* + - * + Process = VM checks if client has the private key to auth.

Keys can be reused by multiple VMs

Keys must be at least 2048 bit in SSH-RSA format

1. Section 2: Configure VM availability
   * Units:
     + Plan for Maintenance and Downtime
       - 3 Scenarios/Events:
         * Unplanned Hardware Maintenance

Az Platform predicts hd/platform component about to fail (issues an event)

Az will “Live Migrate” VMs to healthy physical machine to preserve and pause the VM

* + - * + Unexpected downtime

Unexpected failure (ex. Local network fail, local disk fail, rack level fail

Az will auto migrate your VM to healthy physical machine (reboots VM and loss of temp drive)

* + - * + Planned Maintenance

Periodic updates to Azure platform to improve infrastructure (rarely impact VMs)

\*\* no auto update of VM OS/software tho

* + - Setup Availability Sets
      * A feature used to ensure that a group of related VMs are deployed to reduce downtime during failures/not upgrade all at the same time
        + These VMs do identical operations
        + These VMs are run across multiple physical servers, racks, storage units, and networks
      * Avail. Sets allow for:
        + Redundancy
        + Can configure each app tier w/ separate Avail. Sets
        + Combine load balancers w/ Avail. Sets
        + Use managed disks w/ VMs
      * Service Level Agreements (SLA)
        + VMs w/ 2+ instances in 2+ Avail. Zones are guaranteed by MS to be available 99.99%
        + 99.95% availability for VMs w/ 2+ instances in same Availability Set
        + Single instance VM w/ premium storage for OS and data disks are guaranteed to have 99.99% connectivity
    - Update and Fault domains
      * Help to maintain high availability and fault tolerance during upgrades)
        + Each VM in Avail. Set is in 1 update domain and 1 fault domain
      * Update Domain
        + Group of nodes that are updated together during service upgrade rollouts (allows for incremental rolling out of upgrades)

1 Update domain has:

1 VM + 1 physical hardware that is updated/rebooted at the same time

Only 1 update domain reboots during planned maintenence

5 update domains by default (but up to 20)

* + - * Fault Domain
        + Group of nodes repping 1 physical unit of failure (group of VMs w/ common hardware that share single point of failure)

Have 2 fault domains mitigate against these failures

**Ex**. Server rack used by multiple VMs

* + - Availability zones
      * Protects apps from data-centre failures
      * Considerations:
        + Availability zones are physical locations in an Azure Region

Zone = 1+ independent data-centres

Minimum 3 zones per Azure Region (**ex**. 3 data centres in Canada Central)

Zones are physically separated to protect against data-centre fails

Zone-redundancy replicates apps across these Availability zones (99.99% VM uptime SLA)

* + - * Implementation
        + 1 Availability Zone = Fault + Update Domains

Ex. 3 VMs across 3 zones in 1 region = 3 update & fault domains

* + - * + Az Services w/ Availability zones:

1) Zonal service (pins resources to a specific zone (ex. VMs, managed disks, IP addresses)

2) Zone-redundant service (auto replicated across zones (ex. Zone-redundant storage, SQL)

* + - Vertical and Horizontal Scaling
      * Vertical (scale up/down)
        + Increase/decrease the size of the VM itself (based on workload)

Ex. Inc. VM size to cope w/ higher demand during the weekend

* + - * Horizontal (scale out/in)
        + Inc./dec. # of VMs
      * Considerations:
        + Vertical has more limits since it depends on availability of hardware which can vary by region

Also forces VM to reboot

* + - * + Horizontal is more flexible (scale sets)
        + Reprovisioning = remove and replace an existing VM
    - Scale Sets
      * Scale sets are a compute resource where you can deploy/manage sets of IDENTICAL VMs automatically based on demand
      * **Benefits**:
        + All VMs created have same OS image/configuration
        + Scale sets support using:

Load balancers for layer-5 traffic distribution

App gateways for layer-7 distribution and SSL termination

* + - * + Scale sets run multiple instances of a VM to reduce failure impacts
        + Scale sets can match you Cx demand
        + Supports up to 1000 VM instances
    - Creating Scale Sets
      * Parameters:
        + Initial instance count (# of VMs in scale set: 0-1000)
        + Instance size for each VM in the set
        + Az Spot Instance (low-priority VMs are allocated from Azure’s excess compute capacity)

These VMs enable several types of workloads to run at a reduced cost

* + - * + Use managed disks to show the abstraction of a disk (instead of underlying storage accounts)

Unmanaged disks will expose SAs and VHD blobs

* + - * + Enable scaling beyond 100 instances

If “yes”, you can span multiple placement groups instead of just 1 group with 100 max

* + - * + Spreading algorithms (deploy at max spread)
    - Implement auto scale
      * Auto scaling in/out to meet demands
      * **Benefits**:
        + *Auto adjust capacity* by using rules to define acceptable performance levels
        + *Scale out/in* based on application demand

**Ex**. Scale out during the week, but scale in during the evenings

* + - * + *Schedule events*

Auto inc/dec at fixed times

* + - * + *Less overhead* to monitor/optimize app performance
    - Configure auto scale
      * **Parameters**:
        + Min/Max # of VMs for auto scale
        + CPU usage threshold to trigger scale out/in
        + # of VMs to increase/decrease by when event is triggered

1. Section 3: Configure VM extensions
   * Units:
     + Implement VM extensions
       - Extensions can be used to automate creating, maintaining, and removing VMs
         * They can be managed by Az CLI, Pshell, ARM templates, Portal
         * They can be bundled w/ new VM deployment or existing VMs/system
     + Implement custom script extensions (CSE)
       - Used to automate launch/execute VM custom tasks post configuration
         * Example tasks: Stop VM, install software component, etc.
       - CSE is installed through Portal and you provide a Pshell script to run
         * Scripts can be downloaded from Az storage, github, or Az Portals

**Ex**.

Set-AzVmCustomScriptExtension -FileUri https://scriptstore.blob.core.windows.net/scripts/Install\_IIS.ps1 -Run "PowerShell.exe" -VmName vmName -ResourceGroupName resourceGroup -Location "location"

* + - * Considerations:
        + Timeout

CSE have 90 mins to run so deployment can’t exceed this time (VM must be running during this time)

* + - * + Dependencies (networking or storage access) for CSE
        + Failure events

What will script do in the event of an error (**ex**. Run out of diskspace or sec restrictions)

* + - * + Sensitive data

This may need to be provided to the script

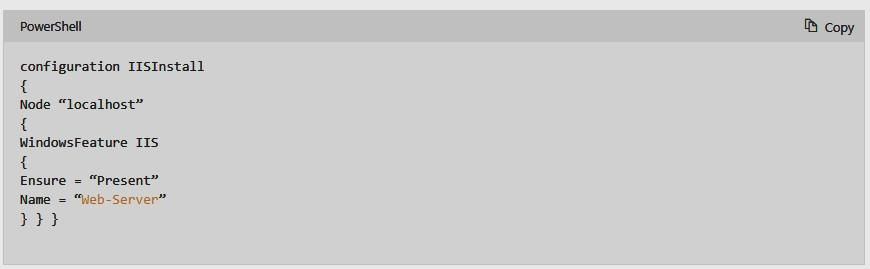
* + - Implement desired state configuration (DSC)
      * A management platform in Windows Pshell that enables deploying/managing config data for software services
        + It also manages the environment where these services run
        + DSC provides:

Set of Windows Pshell language extensions

Windows Pshell cmdlets

Resources that can be used to specify how software env is to be configured

DSc can provide means to maintain/manage existing configs

* + - * Configurations are “easy-to-read" scripts that describe an environment w/ specific characteristics
        + Ex. Characteristic = specific windows feature to be enabled
        + Ex2. Characteristic = deploy sharepoint
      * Dsc is used when your CSE isn’t working
      * **Example**: *installing IIS on local host* (in a PS1 File)
        + 
        + **Configuration** block = name of the configuration
        + **Node** block = defines the computer/VMs you are configuring (ex. “localhost” is the comp being configured)
        + **Resource** blocks configure the resource properties

**Ex**. “windows feature” is a resource block

This block is indicating the name (Web-server) of the role/feature you are trying to ensure is added or removed

“*ensure*” indicates role is added (“present or absent”)

1. Section 4: Config App Service Plans
   * Units:
     + Implement Az App Service Plans (ASP)
       - ASP defines a set of compute resources for a web app to run
         * These resources are analogous to server farm in normal web hosting
         * 1+ apps can use the same ASP
       - The resources in the ASP are in the same region as the plan
         * **ASP defines**:

Region

# of VM instances

Size of VM instances

* + - * How it works:
        + Web apps in App Services are added to automatically to an ASP
        + When the app runs, all the VM instances in the ASP are run

If multiple apps per ASP, they all use those VM instances

Multiple deployment slots for an app are also run on those instances’

The diag logs, perform backups, WebJobs all use the CPU and memory of these instances

ASP is a scale unit of App Service apps

ASP can be configured to scale out too

* + - * Considerations:
        + You can potentially save money by running multiple apps in ASPs

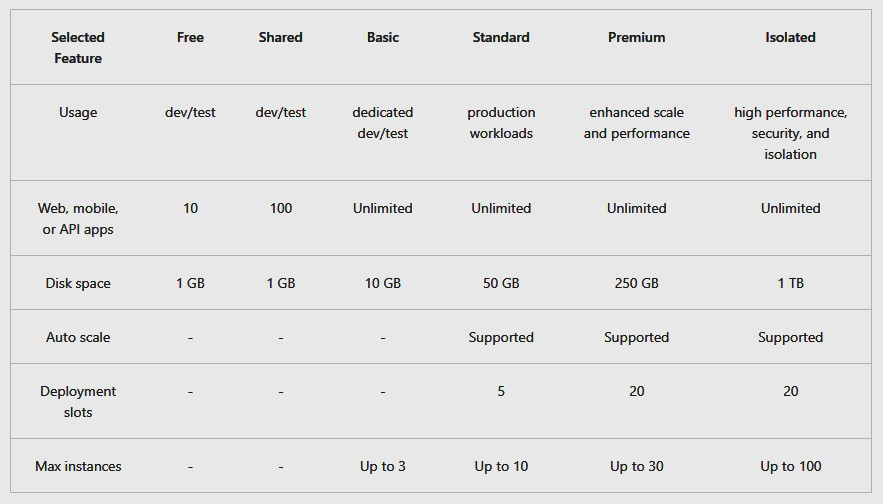
But the ASP’s resources must be able to handle the load!

* + - * + Isolate app into a new ASP when:

App is resource intensive

Scale app independently

App needs resources in diff regions

* + - Determine ASP pricing
      * 
      * Tiers/Plans:
        + Free/shared

ASPs that run on the same VM as other apps whom may belong to other customers

Use this for dev/testing as there is no SLA for these plans

Metered on a per app basis

* + - * + Basic

Designed for apps w/ low traffic requirements (no auto scale)

Priced based on size/# of instances

Built in network load balancing support to auto distribute traffic across instances

Also supports Linux containers

* + - * + Standard

Designed for prod workloads

Priced same as basic

Built in network load balancing support

Auto scale to adjust # of VM instances according to traffic needs

Also supports Linux

* + - * + Premium

Designed for enhanced performance in prod apps

Better hardware for VMs (SSDs, faster processors, 2x memory-to-core ratio)

Supports higher scaling w/ higher instance counts

* + - * + Isolated

Designed for running mission critical workloads (must be run in a VNET)

Cx can run their app in private, dedicated Az Data-Centre w/ high-end hardware

Private environment used with Isolated Plan = App Service Environment (ASE)

Scale up to 100 instances

* + - Scale up/out the ASP
      * **Web App Scaling Methods:**
        + *Scale up* = increase CPU, memory, disks, gain additional features (ex. Dedicated VM, custom domain/certs, staging slots, autoscaling, etc)

Changes your pricing tier

* + - * + *Scale out*= inc. 3 of VM instances (30 based on tier)

ASE lets you scale to 100

Configure this automatically (rules/schedule) or manually\

* + - * Scale up/down based on your need!
      * If your app depends on other services (**Ex**. SQL, Storage), scale these separately
    - Configure ASP scaling
      * Auto-scaling Settings (scale in/out):
        + Rules on when to scale include trigger:

1) Metric-based = scale based on workload (ex. CPU usage, avg. response time and requests)

2) Time-based = schedule when to scale your instances (ex. Trigger webhook every 7 PM)

* + - * Considerations:
        + Have minimum instances to ensure app is always running
        + Max instances limits cost
        + Auto-scaling between Min and Max
        + Always use scale-out and scale-in rule
        + Choose appropriate metric-based triggered
        + Default instance count
        + Always configure auto-scale notification settings

This setting defines what notification should arise based on an auto-scale event

1. Section 5: Config App Services
   * Units:
     + Implement Azure App Services
       - AAS brings together websites, mobile backends, web APIs
       - **Benefits of AAS**:
         * Multiple language and framework support (**ex**. NET, Java, Ruby, Node, PHP, Python, Pshell)
         * DevOps optimized (CI/DC)

Using AzDO, github, bitbucket, docker, Az Container Registry

Allows for testing and staging environments and managing of apps in AAS w/ Pshell or Az CLI

* + - * + Global scaling and high availability

Host apps in MS global data-centre infrastructure

* + - * + Connect with SaaS platforms and on-prem data

Connect to 50+ enterprise systems (ex. SAP), SaaS services (ex. Salesforce), Internet services (ex. FB), on-prem data (using hybrid connections and VNETs)

* + - * + Security and Compliance

It is ISO, SOC, PCI compliant

Auth through AAD or social Oauth (google, fb, twitter, MS)

Restrict access through IP addresses and Managed Service Identities

* + - * + App Templates accessed through MS Marketplace

**ex**. Wordpress, Joomla, Drupal

* + - * + VS Code integration

Tools dedicated to use VS

* + - * + API and mobile features

It provides turn-key CORS support for REST APIs and simplifies mobile app scenarios by enabling auth, offline data sync, push notifications, etc.

* + - * + Serverless code (run code/script w/o provisioning or managing infrastructure – pay as you go)
    - Create an App Service (Az AS)
      * Parameters:
        + Name (unique = webapp.azurewebsites.net)
        + Publish (hosted w/ code or Docker Container)
        + Runtime stack
        + OS
        + Region
      * App Settings
        + “Always on” --> load app even without traffic

Requires continuous WebJobs/using CRON to trigger WebJob

* + - * + ARR Affinity

If multi instances, ensure that client is routed to same instance during session

* + - * + Connection string
    - CI/CD
      * CD --> push out features/bug fixes quickly w/o impacting users in automated deployments
        + Options:

AzDO

Push code, run tests, generate releases, and push to PROD automatically

GitHub

Connect to the repo to automate deployments from PROD branch

Bitbucket

* + - * Manual Deployment/Manual Push
        + Git (Az AS web apps have a git url you can push to)
        + CLI

“webapp up” --> cmd to package and deploy app

* + - * + VS --> has a Az AS deployment wizard
        + FTP/S (traditional way of pushing code)
    - Create Deployment Slots
      * Use a separate deployment slot instead of the PROD one (for Standard, Premium, Isolated AS plan tier)
        + Deployment slot == live app w/ own hostname
      * Benefits of separate deployment slots
        + Validate app changes in staging slot first before going to PROD slot
        + Eliminates downtime when deploying app by swapping slots when instances are available/warmed up

Configure “Auto swap” if pre-swap validation isn’t needed

* + - * + After swap, the “staged slot” becomes the new PROD slot (you can swap back)
      * Enables: zero cold starts and zero downtime!
    - Add Deployment slots
      * New Deployment slots can be empty or cloned
        + 3 Types of Deployment slot Categories

1) slot-specific settings/config strings

2) CD settings

3) AS Auth settings

* + - * + **Settings that are swapped:** *(asterisk (\*) will become unswapped soon)*

General settings, such as framework version, 32/64-bit, web sockets

App settings (can be configured to stick to a slot)

Connection strings (can be configured to stick to a slot)

Handler mappings

Public certificates

WebJobs content

Hybrid connections \*

Service endpoints \*

Azure Content Delivery Network \*

* + - * + **Settings that aren't swapped:**

Publishing endpoints

Custom domain names

Non-public certificates and TLS/SSL settings

Scale settings

WebJobs schedulers

IP restrictions

Always On

Diagnostic settings

Cross-origin resource sharing (CORS)

Virtual network integration

* + - Secure an Az AS
      * Process:
        + Authenticates/Authorization Module runs in sandbox that your app code runs in

It monitors incoming HTTP requests to your app code:

It authenticates users w/ specific provider

Validates/stores/refreshes tokens

Manages auth session

Injects id info to request headers

* + - * + Config Settings

Behaviours

Allow Anon requests (no action)

Defers authorization of unauthenticated traffic to app code

For authenticated requests, AS will pass this info to the HTTP headers

Allows for flexibility on how you want to allow sign-in

Allow only Authenticated requests

Redirects all anon requests to “/.auth/login/<provider>” that you have chosen

Returns 401 error if anon request comes from native mobile app

You don’t need to write auth code

\*\* not desirable for apps wanting a publically available home page

* + - * + Logging/tracing

All logs about authentication/authorization are stored

Details on “failed request tracing” allows you to see what role the auth module had in failing the request (stored in EasyAuthModule\_32/64)

* + - Create Custom Domain Names
      * Az auto assigns a subdomain (azurewebsites.net) and IP address when web app is created
      * Config steps:
        + 1) Reserve domain name (can be bought through Az Portal)
        + 2) Create DNS Records for you Az web app’s domain

Types of DNS record

A (Address) record maps domain name to IP

CNAME record maps domain name to another domain name

Users see first domain name in their browser

**ex**. chankevin.com is mapped to chankevin.azurewebsites.net

* + - * + 3) Enable custom domains and use portal to validate it
    - Backup an Az AS
      * Create a backup snapshot manually/scheduled
      * Info you retain in a backup
        + App configuration
        + File content
        + DB connected to your app (ex. SQL, Azure Database for MySQL, PostgreSQL, MySQL in-app)
      * Considerations
        + Can only backup if AS is in Standard/Premium tier
        + Config backups manual/schedule
        + Need SA + container to be of same sub as app you want to backup to
        + Full backup is default
        + Partial backup can also be selected
        + Can exclude files you don’t want to backup
        + Backup up to 10GB
        + Firewall enabled SA for backups is not supported
    - App Insights
      * Part of Az Monitor (for live apps)
        + Auto detect performance anomalies and helps you diagnose issues and user patterns
      * Integrated with AzDO and works for multiple language platforms
      * Can monitor/analyze data from mobile apps w/ VS app centre
      * **Features/What it monitors**:
        + Request rates, response times, failure rates

**ex**. Which web pages are popular, when they are accessed, where users are coming from

* + - * + Dependency rates, response times, failure rates

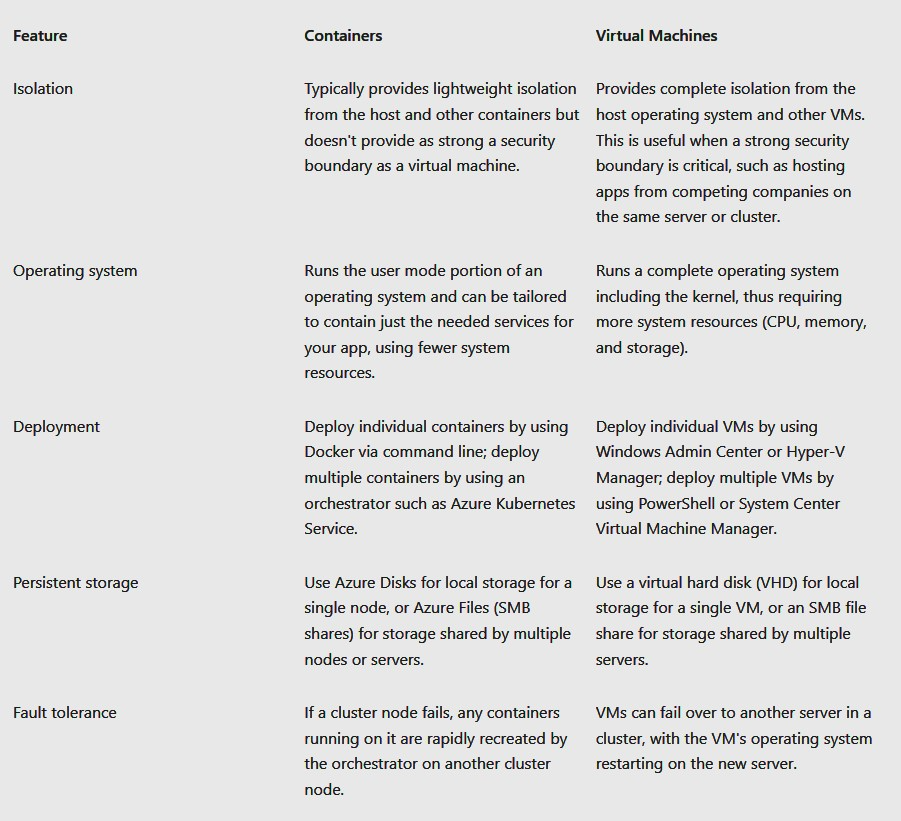
Find out if external services are slowing your web app

* + - * + Exceptions

Analyze aggregated stats and drill into stack trace/related requests

* + - * + Page views/load performance
        + User and session counts
        + Performance counters

CPU, memory, network usage

* + - * + Host Diags
        + Diag trace logs
        + Custom events/metrics
    - Config Az Container Instances (ACI)
      * ACI vs. VMs
        + 
      * Advantages of ACI vs. VM:
        + More flexibility and speed when developing/sharing code
        + Simplified app testing
        + Streamlined app deployments
        + Higher workload density = improved resource utilization
    - Review of ACIs
      * ACI allow for faster and simpler packaging, deploying, and managing cloud apps
        + Fastest way to fun containers w/o managing VMs or adopting high level services
        + **Features**:

*Fast Startup time*

No need to provision VMs = faster startup

*Public IP Connection and DNS names* (directly expose to internet w/ IP address and FQDN)

*HyperV – level security*

Container app is isolated in container (like in VM)

*Custom sizes* based on dynamic scaling

*Persistent storage*

Support for File Share mounting

*Linux/Windows Containers*

ACI can be either Windows or Linux containers

*Co-scheduled Groups*

Support for scheduling multi container groups that share host machine resources

VM deployments

ACI can be deploy into Az VNET

* + - Implement Container Groups
      * Container groups are a collection of containers that scheduled on the same host machine
        + They share a life-cycle, resources, local networking, storage (like Kubernetes)

**Ex of Container group**:

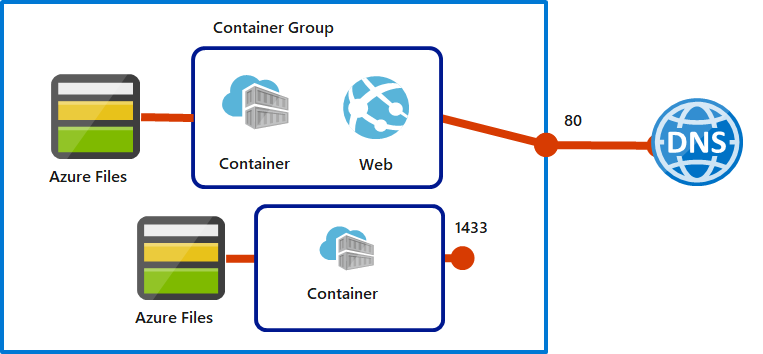
Is scheduled on a single host machine

Is assigned a DNS name label

Exposes a single public IP w/ 1 exposed port

Consists of 2 containers (1 on 80 port, 1 on 1433 port)

Has 2 Az file shares (each container mounts one of the shares locally)



* + - * + Deployment Options

1) ARM templates

Best when you need to also deploy additional az services like File Share w/ ACI

2) YAML file

Best when only ACI being deployed

* + - * + Resource allocation

Container group is allocated resources per container instance

ex. 2 ACI in 1 Container group request CPU, 2 are added to the group-based

* + - * + Networking

Container group can share:

External facing IP

1+ ports on that IP

DNS Label w/ FQDN

Exposing port of a container in the group is required for external clients to access it

No port mapping since containers in a group share a port namespace

* + - * Common Scenarios
        + Container groups best support the division of a single functional task into smaller container images

Each image could be supported by diff teams and have sep resource requirements

Examples:

1 container for web app; 1 container for pulling from source control

1 app container and 1 logging container

1 app container and 1 monitoring container

Frontend and backend containers

* + - Docker Platform
      * This platform enables hosting of apps within a container
        + Consists of:

App Executable code

Runtime env (.NET Core)

System Tools

Settings

* + - * + The Platform ensures that the containerized software is always run the same way (no matter locally, on Win, Linux, or in the cloud)

Software can be developed in Docker container, shared w/ QA, then deployed in PROD cloud.

Scaling would be done using ACIs

* + - * Terms:
        + Container

An instance of a Docker image (reps the execution of a single app, process, or service)

Consists of:

Contents of Docker image

Execution environment

Standard set of instructions

Scaling means that mulitple containers instances are created that are identical/passed diff parameters to

* + - * + Container image

The package with dependencies and info needed to create a container

ex. Frameworks, deployment/execution config for container runtime

Image comes from multiple base images to form container’s file system

Image can’t be changed once created

* + - * + Build

Action of building a container image based on Dockerfile

Use Docker build command

* + - * + Pull

Download container image from container registry

* + - * + Push

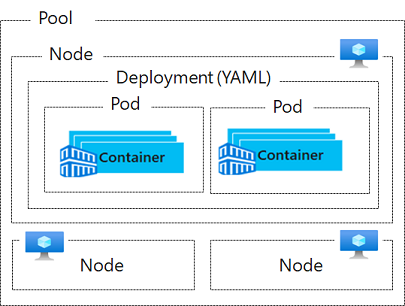
Upload container image to registry

* + - * + Dockerfule

Text file that contains instructions to build Docker image (like Batch script)

1st line = ids the base image

Rest = build actions

1. Section 6: Config Az Kubernetes Service (AKS)
   * Units:
     + AKS Terms
       - **Pools** = group of nodes w/ identical configurations
       - **Nodes** = individual VMs in containerized app
       - **Pods** = single instance of an app
         * (1 pod can have many containers)
       - **Container** = lightweight, portable executable image w/ software and dependencies
       - **Deployment** = pods managed by Kubernetes
       - **Manifest** = YAML file for deployment
       - 
     + Explore AKS cluster and node architecture
       - **1 AKS cluster has 2 components**
         * 1) *Az-managed nodes*

They provide the core AKS services and orchestration of app workloads

Auto created and configured when you create an AKS cluster

Only pay for running agent nodes

* + - * + 2) *CX-managed nodes*

They run your app workloads

* + - * **Nodes and Node pools**
        + 1 AKS Cluster has many nodes (VMs) that run the kub node components and the container runtime

*Kubelet*

The AKS agent that processes the orchestration request from the AZ-Managed node and schedules running of the requested containers

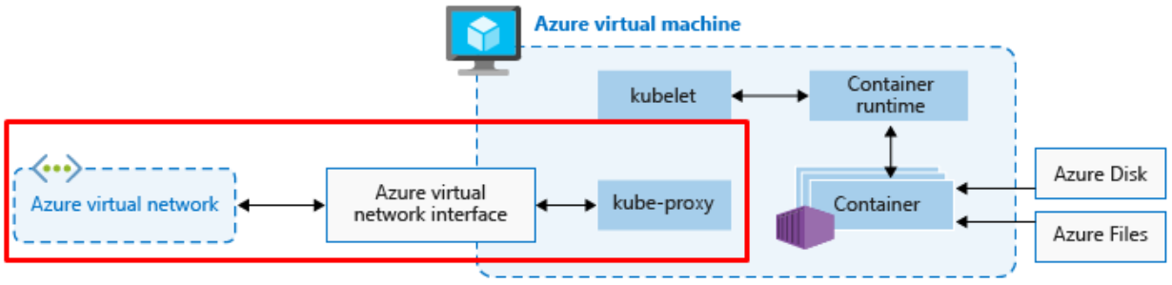
VNETs are handled by “*kube-proxy*” on each node

Proxy routes network traffic and manages IP addresses for services and pods

Container runtime

Component that allows containerized apps to run w/ additional resources (Vnet, Storage)

AKS clusters use “containerd” (newer) and “Moby” for container runtime



* + - * + Nodes of same config are grouped into *node pools*

1 AKS cluster has 1+ node pools

Initial # of nodes are defined in AKS cluster creation

Default node is created which contains the underlying VMs that run your agent nodes

* + - Config AKS Networking
      * AKS provides an abstraction layer to virtual networking in order to all apps to communicate w/ each other
        + Nodes are connected to a VNET and can be provided outbound/inbound connectivity for pods by the “kube-proxy” component on each node does this

Use Load Balancer (LB) to distribute traffic

Ingress controllers for complex routing of app traffic

Security filtering w/ Kubernetes network policies

* + - * + Azure platform simplifies vnetting for AKS Clusters by:

Auto create AZ LB when Kubernetes LB is created

Configure NSG rules when ports to pods are created

Config external DNS as new ingress routes are configured

* + - * Services that Kubernetes uses to logically group pods and provide networking
        + Service Types:

Cluster IP

Creates an Internal IP in cluster that allows for internal-only app connectivity within the cluster

NodePort

Creates a Port mapping on underlying node that allows app direct access with the node ip and portable

Load Balancer

Creates Az LB, configs external IP, connects pods to LB backend pool

Allows CX traffic to reach your apps

ExternalName

Creates a specific DNS entry for app accessed

* + - * Pods
        + Pods run an instance of your app (1 instance = 1 pod)
    - AKS Storage
      * How apps running in AKS store their data
        + Volumes:

Reps a way to store, retrieve data across pods, through their life-cycle (LC)

Can be disposed of after their pod’s LC ends

Use:

Az Disks

Creates a Kub DataDisk resource (premium or standard)

Only available per single node

Az Files

Used to mount an SMB share backed by Az Storage to pods

They let you share data across nodes and pods (premium or standard)

* + - * + Persistent Volumes

Created to exist beyond the LC of an individual pod

Az Disks or Files can provide Persistent Volume (determined which one by concurrent access to the data/performance tier)

Can create this *statically* (by user) or *dynamically* (by API)

Dynamic provisioning of this volume uses a *StorageClass* to id which type of Az Storage is used

* + - * + Storage Classes

Defines tiers of storage-resource

Defines “reclaimPolicy”

Behaviour of the Az storage resource when its pod is deleted/persistent volume isn’t needed anymore

4 Initial StorageClasses:

*default –* uses Az StandardSSD for Managed Disk

Reclaim policy ensures the disk will be deleted when persistent volume is deleted

*managed-premium* – uses Az Premium for Managed Disk

Reclaim policy does the same as default

*azurefile* – uses Az Standard storage for File Share

*azurefile-premium* – uses Az Premium for File Share

PVC requests either Disk or File Storage of a StorageClass

PersistentVolume is *bound* to a PersistentVolumeClaim (1:1)

* + - AKS Scaling
      * Manual scaling of pods or nodes
        + Lets you define set amount of resources to use (fixed cost)

ex. # of nodes

* + - * + You must define replica or node count and then the Kub API schedules creating/draining pods and nodes
      * Horizontal pod autoscaler (HPA)
        + Monitors resource demand and auto scales # of replicas (default = check every 30 secs)

Works w/ AKS clusters (for Kubs 1.8)

* + - * + You define the min/max replicas
        + You define the metric that causes scaling (ex. CPU usage)
      * Cool down of Scaling Events
        + Set amount of time that the autoscaler must wait between scale events

Default = Scale up (3 mins), Scale down (5 mins)

* + - * Cluster autoscaler
        + Scales the number of nodes based on the requested compute resources in the node pool

Checks every 10 secs

* + - * + Used with HPA

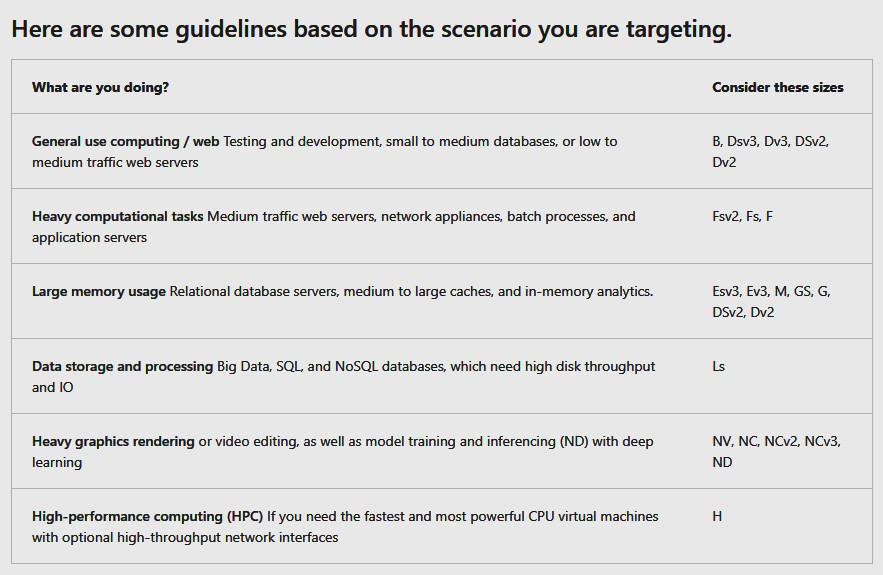
HPA --> scales based on # of pods

Cluster Autoscaler --> scales based on # of nodes

* + - * Scale out events
        + Cluster autoscaler checks whether pods have enough compute resources to run. If not, it will scale out node pool to provide additional resources
      * Scale in events
        + If Cluster autoscaler notices node that passes a threshold of not being needed for 10 mins, it will schedule it for deletion

The autoscaler will then decrease the number of nodes

* + - Configure AKS scaling to Azure Container Instances

1. Section 9: Configuring Virtual Machines
   * Units:
     + Basics of VMs
       - Resources used by Win VMs
         * VM w/ CPU/memory resources
         * Az SA for vHDD
         * Virtual disks for OS, apps, and data
         * VNET to connect VM to other resources
         * Network interface to communicate with Vnet
         * Public IP address
       - Choose the image for your OS + other software
       - Size your VM based on power, memory, storage, and cost
       - **Processor type by use cases:**
         * 
       - Mapping storage to disks
         * VHD replicate physical disks for VMs
         * 2 Default VHDs

1) OS disk ([c://](file:///c://)) w/ 2048 gb

2) Temp disk ([d://](file:///d://))

* + - * Data is stored in the C drive but should be added to dedicated *data disks* (32,767 gb)
        + Max storage of the data disk is based on VM size
        + You can migrate your data from physical disks to data disk
      * Unmanaged vs. Managed Disks
        + Unmanaged

You are responsible for SAs that hold the VHDs (1 SA = 20,000 I/O operations per sec)

1 SA can support 40 VHDs at full throttle

* + - * + Managed

Azure manages your SAs (you only specift disk type and size of the disk)

Benefits:

More reliable

Azure will ensure VHDs for priority VMs are stored in Az Storage with resiliency

Better security

Uses RBAC to restrict who can access the VHDs

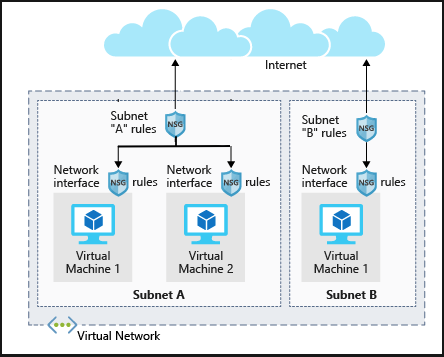
Snapshot supported

Create snapshots of VHD

Used for creating duplicate VMs or backup

Backup support

* + - * VMs require VNETs to manage traffic
    - Configuring VNET settings for VM
      * Process to open ports:
        + Create NSG
        + Create inbound rule to allow traffic on port 20 and 21 for active FTP support
      * NSGs
        + They provide enforcement and control over network traffic (optional software firewall to filter in/outbound traffic to the VNET
        + They are associated to a network interface, a subnet, or both
      * Security group rules
        + Each rule ids the source and destination range, portocol, port, direction, direction (in/out), and allow or deny traffic



* + - * How Azure uses network rules
        + Inbound

Az processes SG on subnet, then SG on network interface

* + - * + Outbound

Az processes network interface, then subnet

* + - * + The NSG rules are evaluated by “priority order” --> lowest number first

DENY will stop all evaluation of rules

DENY ALL is the last priority rule

1. Section 10: Host a Web App w/ Az App Service
   * Units:
     + Create a web app in the Azure Portal
       - Az AS is a fully managed web app hosting platform (PaaS)
       - Deployment slots
         * Each slot can be used for a diff purpose (ex. Staging, prod)
         * Swap between the slots
       - CI/CD Support
         * Azure portal provides you with CI/CD solutions such as AzDO, GitHub, Bitbucket, FTP, Git repos

You can define your own build and release process, run test cases, build releases, and deploy apps (every time you commit new code)

* + - * Integration w/ Visual Studio (VS) allows for publishing web app via Web Deploy tech
        + AS also supports FTP-based publishing
      * App Service also have the ability to scale up/down/out your web apps
        + Inc/dec resources based on usage (scale up/down)
        + Or Scale out for more # of machine instances running your web app
    - Deploy code to App Services
      * Automated deployments (CI)
        + AzDO

Push up code to the cloud, run the tests, generate a release, and push your code to Az Web App

* + - * + GitHub
        + Bitbucket
        + OneDrive
        + Dropbox
      * Manual
        + Git
        + az webapp up (creates a new App Service Web App for you)
        + Zip Deploy (az command)
        + WAR Deployment
        + Visual Studio deployment wizard

GFTP/s

1. Section 10: Protect your virtual machine settings with Azure Automation State Configuration
   * Units:
     + What is Az Automation State Configuration (AZ ASC)
       - Used to ensure that VMs in a cluster are in a consistent state (same software/configs)
         * Built on Powershell so you can consistently deploy, monitor, and update the state of your resources
       - Why use AZ ASC?
         * Addresses consistency of config for your servers that run your services

It centrally manages DSC artifacts/process

Az ASC can target nodes to auto receive the configs of the “pull server” so they conform to the desired state

* + - * + Az Monitor can use logs to review compliance of the nodes
      * What is Pshell DSC?
        + A declarative management platform that Az ASC uses to config, deploy, and control systems

It is better than normal Pshell in managing the use cases and allowing for “Idempotence” (an operation that has the same effect every time)

Normal Pshell struggles to manage logical/error handling

ex. Overwriting a user’s access

* + - * LCM (local config manager)
        + A component of the Win Management Framework (WMF) on Win OS
        + It updates the node’s state to match desired state

Process:

Get the current node state

Test/compare Node’s state to desired state (use DSC script .mof file)

Set/update node to match desired state

* + - * + LCM Modes

1) Push mode

Admin pushes the configs to 1+ nodes which the LCM then checks to ensure that the state of the node matches the desired configuration

2) Pull mode

LCM on each node regularly (15 mins) pulls from server to check the latest config destails

Pros of each Mode:

Push is easy to set up w/ no dedicated infra

Often used to test DSC functionality

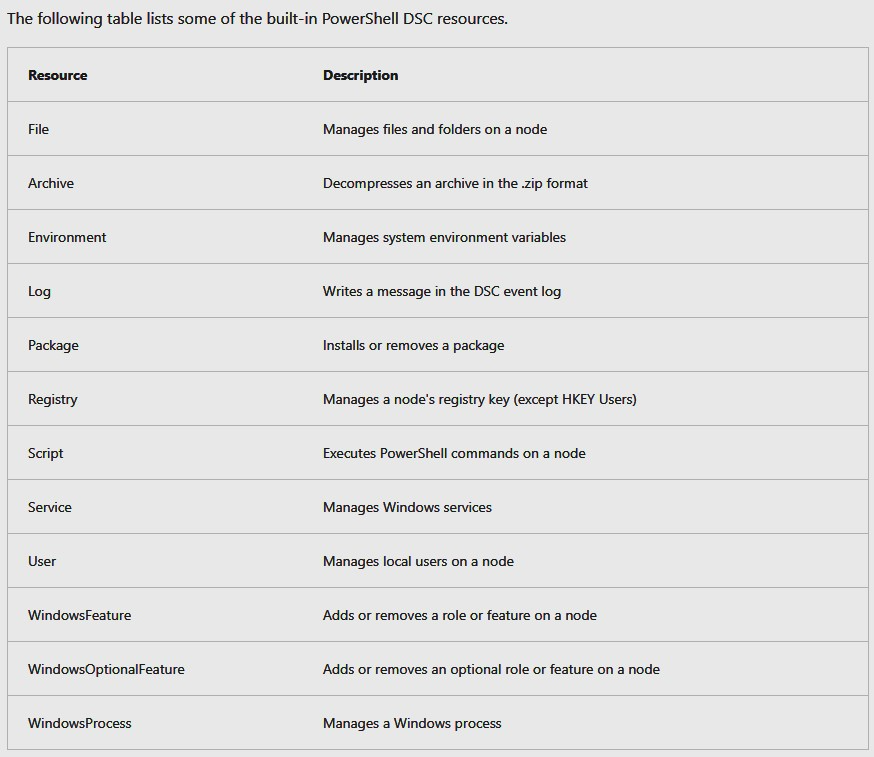
And to get newly imaged machine up to desired state

Pull is used for enterprise deployments

LCM will regularly pull from server

This will remediate any individual machines quickly

Continuous compliance for your security and regulatory obligations

* + - Using Pshell DSC to achieve desired state
      * Get-DSCResource | select Name, Module, Properties
        + 
      * Anatomy of a DSC Code Block
        + Configuration MyDscConfiguration { ##1

Node "localhost" { ##2

WindowsFeature MyFeatureInstance { ##3

Ensure = 'Present'

Name = 'Web-Server'

}

}

}

MyDscConfiguration -OutputPath C:\temp\ ##4

* + - * + 1) **Configuration** block = description of desired state

Has Parameters like a function

* + - * + 2) **Node** block = determines name of the .mof files generated during compilation
        + 3) Resources

Specify the resources to configure

ex. The “WindowsFeature” resource ensures that a Windows feature (“web-server”) is installed

* + - * + 4) **MyDscConfiguration**

This invokes the “function” aka Config block

It compiles the block into a MOF document (managed object format which is a compiled language)

* + - * Configuration data in a DSC script
        + Config data block provides data that the config process might need for each node

It is an array of nodes within an array called “AllNodes”

* + - * Secure credentials in a DSC Script
        + Use a “PSCredential” object in the DSC script to encrypt your credentials
      * Push the Config to a Node
        + “Start-DscConfiguration -path [D:\](file:///D:/)”

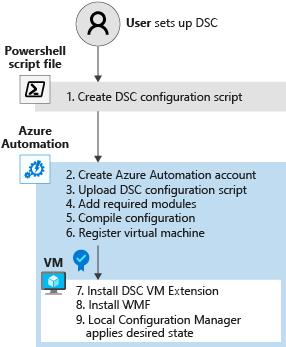
the path is to a directory filled with .mof files you need

* + - * Pull the config to a node
        + Used if you have multiple VMs

You can config an Az Automation account to act as the pull service

Ensure that necessary Pshell modules are added to your Automation Account

**ex**. xSmbShare (how to check for file share) module



# Configure and manage virtual networks for Azure administrators

1. Section 1: Configure VNETs
   * Units:
     + Plan VNETs
       - A VNET is a representation of your network in the cloud
         * It logically isolates your Az Cloud dedicated to your subscriptions

VNETs manage the Virtual Private Networks (VPNs) in Az and can link together VNETs or on-prem

You can also use DNS server setttings for VNETs and segment into Subnets

* + - * + Use Cases:

Create dedicated private cloud-only VNET

Only communicate securely within your cloud space (config endpoint connections for VMs and services)

Securely extend your data centre w/ VNETs (Hybrid cloud)

Use site-to-site (S2S) VPNs to scale your data-centre capacity

S2S VPNs use IPSEC to securely connect corporate VPN gateway to Azure

Enable hybrid cloud

* + - Create Subnets
      * Logical subdivision of VNET’s IP CIDR range
      * Considerations
        + Service Requirements

Each service requires specific routing into/out of subnets (may even require their own subnet)

Enough space

ex. Connect VNET to on-prem using Az VPN Gateway, the gateway requires a subnet

* + - * + Network Virtual Appliances (NVA)

They can be used to route traffic between subnets (tho, by default, all subnets can talk to each other)

* + - * + Service endpoints

Limit access to Az Resources to specific subnets w/ VNET service endpoints

This can not only filter resources from interacting w/ each other, but also the internet

You can selectively add Service Endpoints to subnets

* + - * + NSGs

Contains allow/deny rules for each subnet it is attached too (1:1)

* + - * + Private Links

Private connection from VNET to Az PaaS, cx-owned, or MS Partner services

Simplifies network architecture and eliminates data exposure to the public internet

* + - * \*\* Azure reserves 5 IP addresses by default:
        + x.x.x.0: Network address
        + x.x.x.1: Reserved by Azure for the default gateway
        + x.x.x.2, x.x.x.3: Reserved by Azure to map the Azure DNS IPs to the VNet space
        + x.x.x.255: Network broadcast address
    - Create VNETs
      * **Default**: can create up to 50 VNETs per sub per region. But up to 500 by contacting Azure support.
        + IP spaces of the VNET cannot overlap another one
    - Plan IP Addressing
      * Two Types:
        + Private IP

Used for communications within Az VNET and VNET to on-prem network (using VPN gateway or ExpressRoute (ER))

* + - * + Public IP

Communication to Internet and Az public-facing services

* + - * Static v. Dynamic addressing
        + Static is best for:

DNS name resolution --> requirement to update host record when changing IP address

IP-address-based security models requiring apps/service to have static IP

TLS/SSL certs linked to IP

FW ules that allow/deny traffic using IP ranges

Role-based VMs (Domain controller, DNS servers)

* + - Create public IP addressing
      * Parameters:
        + IP version (IPv4, IPv6 or both)
        + SKU (basic or standard)
        + Name of configuration
        + IP address assignment

Dynamic

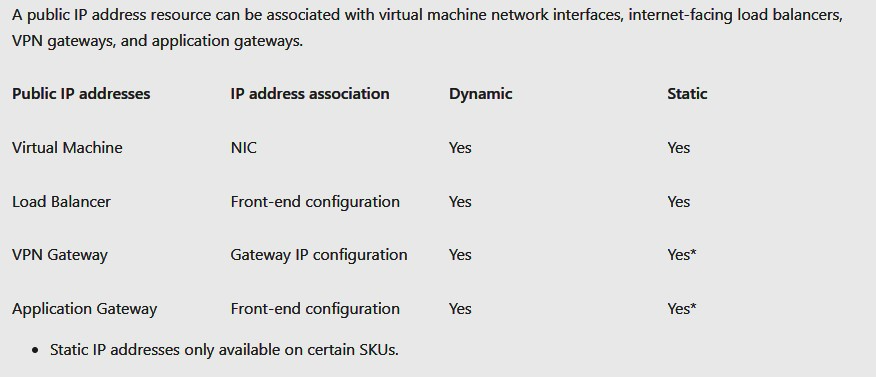
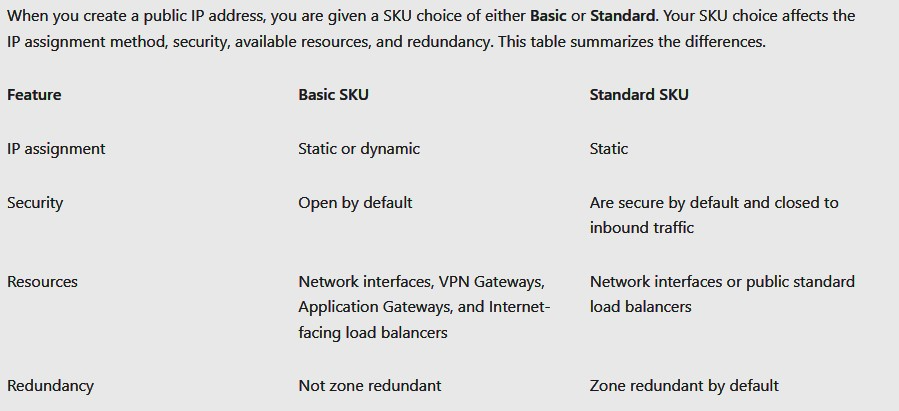
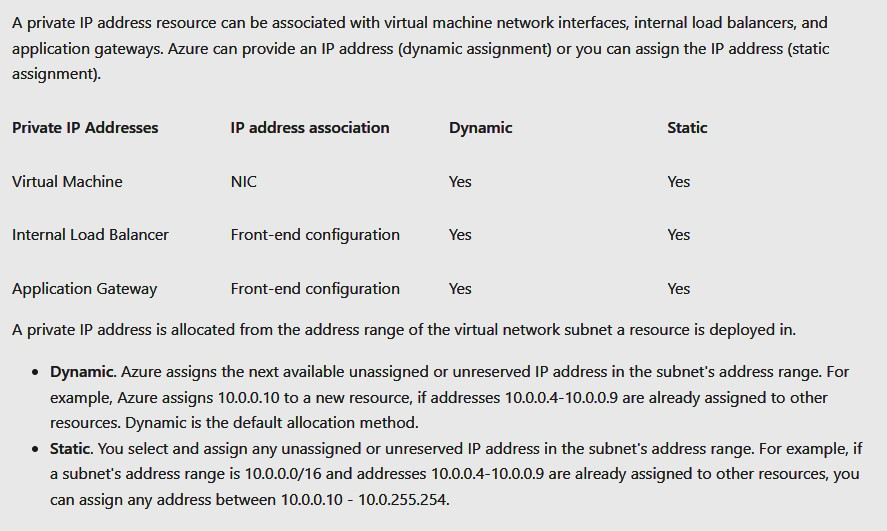
Assigned after a public IP is associated w/ Az resource and resource has started 1st time

This type of address will change after VM is stopped + deallocated, and restarted

Otherwise, it will remain the same

Static

Assigned at resource creation

* + - Associate public IP
      * 
      * Address SKU
        + 
    - Associate Private IP
      * 
      * Private IP is allocated from CIDR of the VNET’s subnet a resource is deployed in
        + Dynamic (default)

Assigns to next available unassigned/unreserved IP in the subnet’s range

* + - * + Static

You assign manually