@author MarcLab1

https://github.com/MarcLab1

**Develop Azure compute solutions (25-30%)**

**Implement IaaS solutions**

provision virtual machines (VMs)

configure, validate, and deploy ARM templates

configure container images for solutions

publish an image to the Azure Container Registry

run containers by using Azure Container Instance

**Create Azure App Service Web Apps**

create an Azure App Service Web App

enable diagnostics logging

deploy code to a web app

configure web app settings including SSL, API settings, and connection strings

implement autoscaling rules including scheduled autoscaling and autoscaling by

operational or system metrics

**Implement Azure functions**

create and deploy Azure Functions apps

implement input and output bindings for a function

implement function triggers by using data operations, timers, and webhooks

implement Azure Durable Functions

implement custom handlers

**Develop for Azure storage (15-20%)**

**Develop solutions that use Cosmos DB storage**

select the appropriate API and SDK for a solution

implement partitioning schemes and partition keys

perform operations on data and Cosmos DB containers

set the appropriate consistency level for operations

manage change feed notifications

**Develop solutions that use blob storage**

move items in Blob storage between storage accounts or containers

set and retrieve properties and metadata

perform operations on data by using the appropriate SDK

implement storage policies, and data archiving and retention

**Implement Azure security (20-25%)**

**Implement user authentication and authorization**

authenticate and authorize users by using the Microsoft Identity platform

authenticate and authorize users and apps by using Azure Active Directory

create and implement shared access signatures

**Implement secure cloud solutions**

secure app configuration data by using App Configuration Azure Key Vault

develop code that uses keys, secrets, and certificates stored in Azure Key Vault

implement solutions that interact with Microsoft Graph

**Monitor, troubleshoot, and optimize Azure solutions (15-20%)**

**Integrate caching and content delivery within solutions**

configure cache and expiration policies

configure cache and expiration policies for Azure Redis Cache

implement secure and optimized application cache patterns including data sizing,

connections, encryption, and expiration

**Instrument solutions to support monitoring and logging**

configure an app or service to use Application Insights

analyze and troubleshoot solutions by using Azure Monitor

implement Application Insights web tests and alerts

**Connect to and consume Azure services and third-party services (15-**

**20%)**

**Implement API Management**

create an APIM instance

configure authentication for APIs

define policies for APIs

**Develop event-based solutions**

implement solutions that use Azure Event Grid

implement solutions that use Azure Notification Hubs

implement solutions that use Azure Event Hub

import OpenAPI definitions

**Develop message-based solutions**

implement solutions that use Azure Service Bus

implement solutions that use Azure Queue Storage queues

| **#** | **Azure Product** | **Azure CLI** | **Example CLI** | **PowerShell Module** | **Example PS** |
| --- | --- | --- | --- | --- | --- |
| **1** | Syntax | az |  | PowerShell uses a verb-noun pair for the names of cmdlets |  |
| **2** | Log in to Azure | [az login](https://docs.microsoft.com/en-us/cli/azure/reference-index?view=azure-cli-latest#az-login) | az login -u johndoe@contoso.com -p VerySecret | [Az.Accounts](https://docs.microsoft.com/en-us/powershell/module/az.accounts/?view=azps-4.3.0#accounts) | Connect-AzAccount |
| **3** | Manage Azure subscription information | [az account](https://docs.microsoft.com/en-us/cli/azure/account) | az account list --o table | [Az.Accounts](https://docs.microsoft.com/en-us/powershell/module/az.accounts/Get-AzSubscription?) | Get-AzSubscription |
| **4** | Manage private registries with Azure Container Registries | [az acr](https://docs.microsoft.com/en-us/cli/azure/acr?view=azure-cli-latest) | az acr list -g MyResourceGroup -o table | [Az.ContainerRegistry](https://docs.microsoft.com/en-us/powershell/module/az.containerregistry/?view=azps-4.3.0#container_registry) | Get-AzContainerRegistry -ResourceGroupName "MyResourceGroup" |
| **5** | Manage Azure Container Services | [az acs](https://docs.microsoft.com/en-us/cli/azure/acs?view=azure-cli-lates) | az acs list-locations --subscription | [Az.Compute](https://docs.microsoft.com/en-us/powershell/module/az.compute/?view=azps-4.4.0#container_service?) | Get-AzContainerService -ResourceGroupName "myRG" |
| **6** | Manage Azure Active Directory | [az ad](https://docs.microsoft.com/en-us/cli/azure/ad?view=azure-cli-latest) | az ad group show --group | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.resources/?view=azps-4.4.0#active_directory) | Get-AzADGroup -First 100 |
| **7** | Manage Azure Advisor | [az advisor](https://docs.microsoft.com/en-us/cli/azure/advisor?view=azure-cli-latest) | az advisor recommendation list --category Performance | [Az.Advisor](https://docs.microsoft.com/en-us/powershell/module/az.advisor/get-azadvisorrecommendation?view=azps-4.4.0) | Get-AzAdvisorRecommendation -Category Performance |
| **8** | Manage Azure Kubernetes Services | [az aks](https://docs.microsoft.com/en-us/cli/azure/aks?view=azure-cli-latest) | az aks get-versions --location westus2 | [Az.Aks](https://docs.microsoft.com/en-us/powershell/module/az.aks/get-azaksversion?view=azps-4.4.0) | Get-AzAksVersion -Location westus |
| **9** | Manage Azure Media Services resources | [az ams](https://docs.microsoft.com/en-us/cli/azure/ams?view=azure-cli-latest) | az ams account list --resource-group --subscription | [Az.Media](https://docs.microsoft.com/en-us/powershell/module/az.media/Get-AzMediaService?view=azps-4.4.0) | Get-AzMediaService -ResourceGroupName "myRG" |
| **10** | Manage App Configurations | [az appconfig](https://docs.microsoft.com/en-us/cli/azure/appconfig?view=azure-cli-latest) | az appconfig list -g MyResourceGroup | [Az.AppConfiguration](https://docs.microsoft.com/en-us/powershell/module/az.appconfiguration/?view=azps-4.4.0#app_configuration) | Get-AzAppConfigurationStore |
| **11** | Manage App Service plans | [az appservice](https://docs.microsoft.com/en-us/cli/azure/appservice?view=azure-cli-latest) | az appservice ase list | [Az.Websites](https://docs.microsoft.com/en-us/powershell/module/az.websites/get-azappserviceplan?view=azps-4.4.0) | Get-AzAppServicePlan -Location "West US" |
| **12** | Manage Azure Backups | [az backup](https://docs.microsoft.com/en-us/cli/azure/backup?view=azure-cli-latest) | az backup container show.. | [Az.RecoveryServices](https://docs.microsoft.com/en-us/powershell/module/az.recoveryservices/?view=azps-4.4.0#recovery_services) | Get-AzRecoveryServicesBackupJob |
| **13** | Manage Azure Batch | [az batch](https://docs.microsoft.com/en-us/cli/azure/batch?view=azure-cli-latest) | az batch account list [--resource-group] [--subscription] | [Az.Batch](https://docs.microsoft.com/en-us/powershell/module/az.batch/?view=azps-4.4.0#batch) | Get-AzBatchAccount -AccountName "pfuller" |
| **14** | Manage Azure Billing | [az billing](https://docs.microsoft.com/en-us/cli/azure/billing?view=azure-cli-latest) | az billing period show --name --subscription | [Az.Billing](https://docs.microsoft.com/en-us/powershell/module/az.billing/?view=azps-4.4.0#billing) | Get-AzBillingInvoice -Latest |
| **15** | Manage Azure Content Delivery Networks | [az cdn](https://docs.microsoft.com/en-us/cli/azure/cdn?view=azure-cli-latest) | az cdn origin list | [Az.Cdn](https://docs.microsoft.com/en-us/powershell/module/az.cdn/?view=azps-4.4.0#cdn) | Get-AzCdnOrigin |
| **16** | Manage Azure Cognitive Services accounts | [az cognitiveservices](https://docs.microsoft.com/en-us/cli/azure/cognitiveservices?view=azure-cli-latest) | az cognitiveservices account list -g MyResourceGroup | [Az.CognitiveServices](https://docs.microsoft.com/en-us/powershell/module/az.cognitiveservices/?view=azps-4.4.0#cognitive_services) | Get-AzCognitiveServicesAccount |
| **17** | Manage Azure CLI configuration | [az configure](https://docs.microsoft.com/en-us/cli/azure/reference-index?view=azure-cli-latest#az-configure) | az configure --defaults group=myRG web=myweb vm=myvm | [Az.Accounts](https://docs.microsoft.com/en-us/powershell/module/az.accounts/set-azcontext?view=azps-4.4.0) | Set-AzContext -SubscriptionId "xxxx-xxxx-xxxx-xxxx" |
| **18** | Manage Azure Container Instances. | [az container](https://docs.microsoft.com/en-us/cli/azure/container?view=azure-cli-latest) | az container logs --name MyContainerGroup --resource-group MyResourceGroup | [Az.ContainerInstance](https://docs.microsoft.com/en-us/powershell/module/az.containerinstance/?view=azps-4.4.0#container_instances) | Get-AzContainerGroup -ResourceGroupName demo -Name mycontainer |
| **19** | Manage Azure Cosmos DB database | [az cosmosdb](https://docs.microsoft.com/en-us/cli/azure/cosmosdb?view=azure-cli-latest) | az cosmosdb list [--resource-group] [--subscription] | [Az.CosmosDB](https://docs.microsoft.com/en-us/powershell/module/az.cosmosdb/?view=azps-4.4.0#cosmos_db) | Get-AzCosmosDBAccount -ResourceGroupName {resourceGroupName} -Name {databaseAccountName} |
| **20** | Manage ARM template deployment at subscription scope | [az deployment](https://docs.microsoft.com/en-us/cli/azure/deployment?view=azure-cli-latest) | az deployment group list -g testrg | [Az.DeploymentManager](https://docs.microsoft.com/en-us/powershell/module/az.deploymentmanager/?view=azps-4.4.0#deployment_manager) | Get-AzDeploymentManagerArtifactSource -InputObject $artifactSourceObject |
| **21** | Manage Azure Managed Disks | [az disk](https://docs.microsoft.com/en-us/cli/azure/disk?view=azure-cli-latest) | az disk list [--resource-group] [--subscription] | [Az.Compute](https://docs.microsoft.com/en-us/powershell/module/az.compute/?view=azps-4.4.0#vm_disks) | Get-AzDisk -ResourceGroupName 'ResourceGroup01' -DiskName 'Disk01' |
| **22** | Manage Data Lake Analytics | [az dla](https://docs.microsoft.com/en-us/cli/azure/dla?view=azure-cli-latest) | az dla account list [--resource-group] [--subscription] | [Az.DataLakeAnalytics](https://docs.microsoft.com/en-us/powershell/module/az.datalakeanalytics/?view=azps-4.4.0#data_lake_analytics) | Get-AzDataLakeAnalyticsAccount -Name "ContosoAdlAccounaz t" |
| **23** | Manage Data Lake Store | [az dls](https://docs.microsoft.com/en-us/cli/azure/dls?view=azure-cli-latest) | az dls account list --resource-group myRG | [Az.DataLakeStore](https://docs.microsoft.com/en-us/powershell/module/az.datalakestore/?view=azps-4.4.0#data_lake_store) | Get-AzDataLakeStoreAccount -Name "ContosoADL" |
| **24** | Manage Azure Data Migration Service | [az dms](https://docs.microsoft.com/en-us/cli/azure/dms?view=azure-cli-latest) | az dms check-name --location westus2 --name MyService | [Az.DataMigration](https://docs.microsoft.com/en-us/powershell/module/az.datamigration/?view=azps-4.4.0#data_migration) | Get-AzDataMigrationProject -InputObject $myService |
| **25** | Manage Azure Event Grid | [az eventgrid](https://docs.microsoft.com/en-us/cli/azure/eventgrid?view=azure-cli-latest) | Get-AzEventGridDomain -ResourceGroup myRG -Name Domain1 | [Az.EventGrid](https://docs.microsoft.com/en-us/powershell/module/az.eventgrid/?view=azps-4.4.0#event_grid) | az ams account list --resource-group --subscription |
| **26** | Manage Azure Event Hubs | [az eventhubs](https://docs.microsoft.com/en-us/cli/azure/eventhubs?view=azure-cli-latest) | az eventhubs eventhub list --resource-group myRG --namespace-name mynamespace | [Az.EventHub](https://docs.microsoft.com/en-us/powershell/module/az.eventhub/?view=azps-4.4.0#event_hub) | Get-AzEventHub -ResourceGroup myRG -NamespaceName MyNamespaceName |
| **27** | Manage resource provider features | [az feature](https://docs.microsoft.com/en-us/cli/azure/feature?view=azure-cli-latest) | az feature list | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.resources/register-azresourceprovider?view=azps-4.4.0) | Register-AzResourceProvider -ProviderNamespace Microsoft.Network |
| **28** | Find commands | [az find](https://docs.microsoft.com/en-us/cli/azure/reference-index?view=azure-cli-latest#az-find) | az find "az storage" | [Az.Accounts](https://docs.microsoft.com/en-us/powershell/module/az.accounts/?view=azps-4.4.0#accounts) | Get-Command -Verb Get -Noun AzVM\* -Module Az.Compute |
| **29** | Manage function apps | [az functionapp](https://docs.microsoft.com/en-us/cli/azure/functionapp?view=azure-cli-latest) | az functionapp list --query "[?state=='Running']" | [Az.Functions](https://docs.microsoft.com/en-us/powershell/module/az.functions/?view=azps-4.4.0#functions) | Get-AzFunctionApp |
| **30** | Manage resource groups and template deployments | [az group](https://docs.microsoft.com/en-us/cli/azure/group?view=azure-cli-latest) | az group create -l westus -n myRG | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.resources/get-azresourcegroup?view=azps-4.4.0) | Get-AzResourceGroup -Name "EngineerBlog" |
| **31** | Manage HDInsight resources. | [az hdinsight](https://docs.microsoft.com/en-us/cli/azure/hdinsight?view=azure-cli-latest) | az hdinsight list [--resource-group] [--subscription] | [Az.HDInsight](https://docs.microsoft.com/en-us/powershell/module/az.hdinsight/?view=azps-4.4.0#hdinsight) | Get-AzHDInsightCluster |
| **32** | Managed Service Identities. | [az identity](https://docs.microsoft.com/en-us/cli/azure/identity?view=azure-cli-latest) | az identity list-operations [--subscription] | [Az.ManagedServiceIdentity](https://docs.microsoft.com/en-us/powershell/module/az.managedserviceidentity/?view=azps-4.4.0#managed_service_identity) | Get-AzUserAssignedIdentity -ResourceGroupName PSRG -Name ID1 |
| **33** | Manage custom virtual machine images. | [az image](https://docs.microsoft.com/en-us/cli/azure/image?view=azure-cli-latest) | az image builder show --name mytemplate --resource-group my-group | [Az.ManagedServiceIdentity](https://docs.microsoft.com/en-us/powershell/module/az.imagebuilder/?view=azps-4.4.0#vm_images) | Get-AzImageBuilderTemplate |
| **34** | Manage KeyVault | [az keyvault](https://docs.microsoft.com/en-us/cli/azure/keyvault?view=azure-cli-latest) | az keyvault list [--resource-group] [--subscription] | [Az.KeyVault](https://docs.microsoft.com/en-us/powershell/module/az.keyvault/?view=azps-4.4.0#key_vault) | Get-AzKeyVault |
| **35** | Manage Azure Kusto resources | [az kusto](https://docs.microsoft.com/en-us/cli/azure/kusto?view=azure-cli-latest) | az kusto cluster list --resource-group myRG | [Az.Kusto](https://docs.microsoft.com/en-us/powershell/module/az.kusto/?view=azps-4.4.0#kusto) | Get-AzKustoCluster -ResourceGroupName testrg |
| **36** | Manage Azure locks | [az lock](https://docs.microsoft.com/en-us/cli/azure/lock?view=azure-cli-latest) | az lock list | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.resources/get-azresourcelock?view=azps-4.4.0) | Get-AzResourceLock -ResourceGroupName "myRG" -AtScope |
| **37** | Manage assignments and definitions | [az managedservices](https://docs.microsoft.com/en-us/cli/azure/managedservices?view=azure-cli-latest) | az managedservices definition list | [Az.ManagedServices](https://docs.microsoft.com/en-us/powershell/module/az.managedservices/?view=azps-4.4.0#managed_services) | Get-AzManagedServicesAssignment |
| **38** | Manage Azure Maps | [az maps](https://docs.microsoft.com/en-us/cli/azure/maps?view=azure-cli-latest) | az maps account show --name MyMapsAccount --resource-group myRG | [Az.Maps](https://docs.microsoft.com/en-us/powershell/module/az.maps/?view=azps-4.4.0#maps) | Get-AzMapsAccount -ResourceGroupName myRG |
| **39** | Manage Azure Database for MariaDB servers | [az mariadb](https://docs.microsoft.com/en-us/cli/azure/mariadb?view=azure-cli-latest) | az mariadb db list -g testgroup -s testsvr | [Az.MariaDb](https://docs.microsoft.com/en-us/powershell/module/az.mariadb/?view=azps-4.4.0#database_for_mariadb) | Get-AzMariaDbServer |
| **40** | Manage the Azure Monitor Service. | [az monitor](https://docs.microsoft.com/en-us/cli/azure/monitor?view=azure-cli-latest) | az monitor action-group list [--resource-group] [--subscription] | [Az.Monitor](https://docs.microsoft.com/en-us/powershell/module/az.monitor/?view=azps-3.8.0#monitor) | Get-AzActionGroup |
| **41** | Manage Azure Database for MySQL servers. | [az mysql](https://docs.microsoft.com/en-us/cli/azure/mysql?view=azure-cli-latest) | az mysql db list -g testgroup -s testsvr | [Az.MySql](https://docs.microsoft.com/en-us/powershell/module/az.mysql/?view=azps-4.4.0#database_for_mysql) | Get-AzMySqlServer |
| **42** | Manage Azure Network resources. | [az network](https://docs.microsoft.com/en-us/cli/azure/network?view=azure-cli-latest) | az network nic list --query "[?dnsSettings.internalDomainNameSuffix=`{dnsSuffix}`]" | [Az.Network](https://docs.microsoft.com/en-us/powershell/module/az.network/?view=azps-4.4.0#networking) | Get-AzNetworkInterface |
| **43** | Manage resource policies. | [az policy](https://docs.microsoft.com/en-us/cli/azure/policy?view=azure-cli-latest) | az policy definition show --name MyPolicyDefinition | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.resources/?view=azps-4.4.0#policies) | Get-AzPolicyDefinition |
| **44** | Manage Azure Database for PostgreSQL servers. | [az postgres](https://docs.microsoft.com/en-us/cli/azure/postgres?view=azure-cli-latest) | az postgres db list -g testgroup -s testsvr | [Az.PostgreSql](https://docs.microsoft.com/en-us/powershell/module/az.postgresql/?view=azps-4.4.0#database_for_postgresql) | Get-AzPostgreSqlServer |
| **45** | Manage dedicated Redis caches for your Azure applications. | [az redis](https://docs.microsoft.com/en-us/cli/azure/redis?view=azure-cli-latest) | az redis list [--resource-group] [--subscription] | [Az.RedisCache](https://docs.microsoft.com/en-us/powershell/module/az.rediscache/?view=azps-4.4.0#redis_cache) | Get-AzRedisCache -Name "myexists" |
| **46** | Manage Azure Reservations | [az reservations](https://docs.microsoft.com/en-us/cli/azure/reservations?view=azure-cli-latest) | az reservations reservation list --reservation-order-id [--subscription] | [Az.Reservations](https://docs.microsoft.com/en-us/powershell/module/az.reservations/?view=azps-4.4.0#reservations) | Get-AzReservation -ReservationOrderId "1111aaaa" |
| **47** | Manage Azure resources. | [az resource](https://docs.microsoft.com/en-us/cli/azure/resource?view=azure-cli-latest) | az resource list --location westus | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.resources/?view=azps-4.4.0#resources) | Get-AzResource | ft |
| **48** | Manage user roles for access control with AAD and service principals | [az role](https://docs.microsoft.com/en-us/cli/azure/role?view=azure-cli-latest) | az role assignment list [--all] | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.resources/get-azadserviceprincipal?view=azps-4.4.0) | Get-AzADServicePrincipal |
| **49** | Manage Azure Search services | [az search](https://docs.microsoft.com/en-us/cli/azure/search?view=azure-cli-latestt) | az search service list --resource-group [--subscription] | [Az.Search](https://docs.microsoft.com/en-us/powershell/module/az.search/?view=azps-4.4.0#search) | Get-AzSearchService -ResourceGroupName felixwa-01 |
| **50** | Manage security with Azure Security Center | [az security](https://docs.microsoft.com/en-us/cli/azure/security?view=azure-cli-latest) | az security alert list | [Az.Security](https://docs.microsoft.com/en-us/powershell/module/az.security/?view=azps-4.4.0#security") | Get-AzDiscoveredSecuritySolution |
| **51** | Manage shared image gallery | [az sig](https://docs.microsoft.com/en-us/cli/azure/sig?view=azure-cli-latest) | az sig list [--resource-group] [--subscription] | [Az.ImageBuilder](https://docs.microsoft.com/en-us/powershell/module/az.imagebuilder/?view=azps-4.4.0#vm_images) | Get-AzImageBuilderTemplate |
| **52** | Manage snapshots | [az snapshot](https://docs.microsoft.com/en-us/cli/azure/snapshot?view=azure-cli-latest) | az snapshot list [--resource-group] [--subscription] | [Az.Compute](https://docs.microsoft.com/en-us/powershell/module/az.compute/Get-AzSnapshot?view=azps-4.4.0%22) | Get-AzSnapshot |
| **53** | Manage Azure SQL Databases and Data Warehouses. | [az sql](https://docs.microsoft.com/en-us/cli/azure/sql) | az sql db list --resource-group myRG --server myserver | [Az.Sql](https://docs.microsoft.com/en-us/powershell/module/az.sql/get-azsqldatabase) | Get-AzSqlDatabase -ResourceGroupName "myRG" -ServerName "server01" |
| **54** | Manage Azure Storage resources | [az storage](https://docs.microsoft.com/en-us/cli/azure/storage) | az storage account list -g myRG] | [Az.Storage](https://docs.microsoft.com/en-us/powershell/module/az.storage/?view=azps-4.4.0#storage) | Get-AzStorageAccount -ResourceGroupName "RG01" |
| **55** | Manage resource tags | [az tag](https://docs.microsoft.com/en-us/cli/azure/tag) | az tag list [--subscription] | [Az.Resources](https://docs.microsoft.com/en-us/powershell/module/az.accounts/Get-AzSubscription?) | Get-AzTag -Name "Department" |
| **56** | Show the versions of Azure CLI modules | [az version](https://docs.microsoft.com/en-us/cli/azure/reference-index?view=azure-cli-latest#az-version) | az version [--subscription] | [PowerShellGet](https://docs.microsoft.com/en-us/powershell/module/powershellget/get-installedmodule?view=powershell-7) | Get-InstalledModule Azure -AllVersions |
| **57** | Manage Linux or Windows virtual machines | [az vm](https://docs.microsoft.com/en-us/cli/azure/vm) | az vm list -g myRG | [Az.Compute](https://docs.microsoft.com/en-us/powershell/module/az.compute/?view=azps-4.4.0#virtual_machines) | Get-AzVM -ResourceGroupName "myRG" -Name "VirtualMachine07" |

* PowerShell we use “ or ‘
* CLI we can only use “

PowerShell commands

* Verb – describes which action to be taken (Get, Join, Connect)
* Use Tab a lot for possible commands (completers)
* Script commands run in parallel, not one at a time
* Whenever you don’t supply all the required parameters for creating anything, the complier will complain or a popup will appear asking for the necessary information
* start-transcript
  + starts a transcript(log file) and puts it into /Documents folder
* **Connect-AzAccount**
* **Get-Az, New-Az, Select-Az, Set-Az, Remove-Az**
* Get-AzContext
* Get-AzResourceGroup
* Get-AzResourceGroup | Where-Object { $\_.Location -eq 'eastus' } | select Location, ResourceGroupName
* Select-AzSubscription
* Select-AzSubscription -Subscription "Azure Subscription 1"
* New-AzVM
* $rsParams = @{ResourceGroupName = 'TutorialResources'}

$newRS1 = New-AZResourceGroup @rsParams

If you run the same command twice:

* + Provided resource group already exists. Are you sure you want to update it?

[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): n

* Remove resource group - Remove-AzResourceGroup -Name ExampleResourceGroup
* To connect to a VM
  + Get-AzRemoteDesktopFile -ResourceGroupName "RgName" -Name "VmName" –Launch
* To save the rdp file
  + Get-AzRemoteDesktopFile -ResourceGroupName "RgName" -Name "VmName" -LocalPath "C:\Path\to\folder"
* To create a new VM with parameters

$vmParams = @{

ResourceGroupName = 'TutorialResources'

Name = 'TutorialVM1'

Location = 'eastus'

ImageName = 'Win2016Datacenter'

PublicIpAddressName = 'tutorialPublicIp'

Credential = $cred

OpenPorts = 3389

}

$newVM1 = New-AzVM @vmParams

* ProvisioningState : Succeeded when New command the command was successful
* Remove Resource Group
  + Remove-AzResourceGroup –ResourceGroupName TutorialResources
* $resourcegroupname = “rs-vm-canadacentral-001”

Resource Groups

* The minimum is Name, Location
* New-AzResourceGroup -ResourceGroupName "Az204ResourceGroup" -Location "EastUS"

Storage Accounts

* The Minimum is Name, ResourecGroupName, Location, SkuName
* New-AzStorageAccount -ResourceGroupName "learn-df0aea7d-85bf-47c5-8724-58e53bfd85ac" -AccountName "stcanada001" -Location "canadacentral" -SkuName "Standard\_LRS"
* Set the current Storage Account
  + Set-AzCurrentStorageAccount -ResourceGroupName "RG01" -AccountName "mystorageaccount"
* Create a Storage Container inside the current storage account
  + New-AzStorageContainer container001
* Upload blob to container
  + Set-AzStorageBlobContent (have to specify container and blob file)
* Remove blob from container
  + Remove-AzStorageBlob (have to specify container and blob)
* Copy between 2 containers in the same storage
  + Start-AzStorageBlobCopy

KeyVault

* Minimum is Name, ResourceGroupName, Location
* New-AzKeyVault

WebApp

* New-AzWebApp

Cloud Shell

* Inherits the session from the browser, so you don’t have to login manually
* Shell.azure.com

PowerShell

* Create an App Service Plan
  + New-AzAppServicePlan
  + Minimum is Name, Location, ResourceGroupName
  + –Tier Free
* Get an AppService Plan
  + Get-AzAppServicePlan -ResourceGroupName "Default-Web-WestUS" -Name "ContosoASP"
  + Get-AzAppServicePlan -Location canadacentral
* Modify an AppServicePlan
  + Set-AzAppServicePlan -ResourceGroupName "learn-e04f35e4-013b-4276-8472-9cdc2877c87c" -Name "appserviceplan647" -PerSiteScaling $true
* Create an App Service
  + New-AzWebApp -Name webapp905 -Location canadacentral -AppServicePlan appserviceplan905 -ResourceGroupName rs-testing-001
* Publish-AzWebApp
  + ResourceGroupName rs-testing-001 Name webapp905 ArchivePath WebApplicationDemo.deps.zip
* Remove WebApp
  + Remove-AzWebApp -ResourceGroupName "Default-Web-WestUS" -Name "ContosoSite"
* Restart Web App
  + Restart-AzWebApp -ResourceGroupName "Default-Web-WestUS" -Name "ContosoSite"
* Start Web App
  + Start-AzWebApp -ResourceGroupName "Default-Web-WestUS" -Name "ContosoWebApp"
* Stop Web App
  + Stop-AzWebApp -ResourceGroupName "Default-Web-WestUS" -Name "ContosoWebApp"

To get the Tier of an app service plan

* (Get-AzAppServicePlan -Name appserviceplan905).Sku.Tier

OR

$asp1 = Get-AzAppServicePlan -Name appserviceplan905

Then

$tier1 = $asp1.Sku.Tier

$tier1

To get more info about an app Service

* (Get-AzWebApp -Name webappn905).

[Azure CLI](https://adamtheautomator.com/azure-cli/) Commands

* Azure CLI command starts with az you are using the current 2.0 version
* if it starts with azure you are using the older 1.0 Xplat version
* az login
* az account
* az account list
* az group
* az vm
* az storage account
* az keyvault
* az webapp
* az cosmosdb

Resource groups

* az group create --name TutorialResources --location eastus

(name can also be –n or --resource-group)

VM

* az vm create --resource-group TutorialResources \

--name TutorialVM1 \

--image UbuntuLTS \

--generate-ssh-keys \

--output json \

--verbose

* az group create --resource-group rg-smelly-922 --location westus

Show (similar to the Get command in powershell)

* az vm show --name TutorialVM1 --resource-group TutorialResources

List

* az group list
* az resource list
* az group list -o table
* az webapp list

Delete

* az group delete --name TutorialResources --no-wait

az resource delete -g MyResourceGroup -n MyVm --resource-type "Microsoft.Compute/virtualMachines"

Wait (wait until the deletion is complete or watch it progress)

* az group wait --name TutorialResources --deleted

Set the subscription

* az account set --subscription "Student 2"

Variables (very important to not use spaces)

* Set VARIABLE=value
* Set LOCATION=EastUS
* Set template="azuredepoly.json" (with or without quotations)
* az group create --name foo --location %LOCATION%
* az deployment group create --name blanktemplate --resource-group rg-arm-123 --template-file %templateFile%

Query

* az group list --query "[?location=='westus']"
* az group list --query "[?location=='westus'].name"
* az group list --query "[?location=='westus'].{name:name,location:location}"

Interactive mode

* az interactive
* INTELLISENSE!!!

Create container instance

* az container create --resource-group rg-container-111 --name containermy111 --image marclab1/aci-hello-world:v1 --dns-name-label aci-demo13131 --ports 80

**Develop Azure compute solutions (25-30%)**

**Implement IaaS solutions**

provision virtual machines (VMs)

configure, validate, and deploy ARM templates

configure container images for solutions

publish an image to the Azure Container Registry

run containers by using Azure Container Instance

Virtual Machines

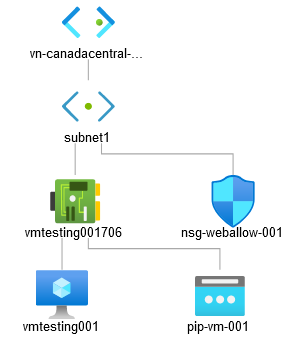
* IaaS
* Total control over the OS and software
* Supports marketplace and uploaded custom images
* VMs can be resized after creation
* Best suited for custom software requiring custom configurations and high degree of control
* Can run any application scenario – web apps, dbs, desktop apps, jumpboxes, gateways
* Customer provides the availability, security, and disaster recovery
* VMs are great but they lack scalability
* Customers are responsible for OS patches, but can automate using VM guest patching (public preview)
* Customers are responsible for antimalware, backups, configurations, etc;
* Administrator account – cannot be named “admin” or root; Cannot use password1 for password
* Not every VM size is available in every region
* Lift and shift – moving apps that cannot be containerized
* VMs don’t actually live inside VNets, we attach NIC(s) to VMs and connect the NIC to the VNet; Can have more than 1 NIC and connect different NICs to different subnets within the same VNet, but cannot be connect to multiple VNets
* Max Nodes = 1
* Each subscription has a limit of 20 VMs per region; Increased by contacting Azure support
* Virtual disks needed by the VM are .vhd files stored as page blobs in a storage account
* If you configure Managed disks (under Advanced section in Disks), then Azure handles the storage account automatically and storage account won’t appear in the vm configuration
* By default, RDP is enabled for Windows, ssh for Linux
* When you create a vm, an IP is assigned by default but not a FQDN (You can add one after)
* Ideally, you should not configure remote access over public IPs on production VMs. Instead, deploy a vpn and connect to your VM using its private IP. Use the public IP to publish only the service hosted in the VM, like an IIS service, while getting remote access to the VM using the private IP
* If your vms are highly available, or you want to scale them up and down, then you would typically use a load balancer. If your vm uses a load balancer, you should put your vms into an availability set
* A virtual machine can only be added to an availability set when created (so you have to create the availability set before you create the vms)
* When you create a VM in Azure, a new Virtual Network is also created with a default setting.
* Azure VMs in different subnets in the same virtual network can communicate by default
* When you create a vm, by default the nsg is attached to the nic (not the subnet)

SLA

* 99.99% - two or more instances deployed across two or more Availability Zones in the same Azure region
* 99.95% - two or more instances deployed in the same Availability Set or in the same Dedicated Host Group
* 99.9 - Single Instance Virtual Machine using Premium SSD or Ultra Disk for all Operating System Disks and Data Disks
* 99.5 - Single Instance Virtual Machine using Standard SSD Managed Disks for Operating System Disk and Data Disks
* 95 - Single Instance Virtual Machine using Standard HDD Managed Disks for Operating System Disks and Data Disks

Creating a VM behind the scenes

* Create Resource group
* Create Virtual network within RG (/16). Also create a subnet (/24)
* Create NSG around the subnet to secure the subnet level (apply the nsg to the subnet in the virtual network > subnet blade)
* Create storage account for diagnostics (storage v2, LRS). The virtual disks needed by the vm are vhd files stored as page blobs in this storage account
* Create the virtual machine. Change public inbound ports to none. We tell the nic to link to the virtual network we created and the subnet in the networking section. Click none under public ip. NIC network security group should also be set to none as we already have a nsg on the subnet level so we don’t need one on the nic. Enable OS guest diagnostics and set the storage account created previously instead of creating a new one. Under Extensions in the Advanced section, add MS Antimalware.
* Create public IP, give it a dns name; Then link it to the nic. You can do this in the networking blade of the vm or in the nic directly (Select vm > networking > select nic > IP Configurations > selct the IP > associate > select IP then click save)
* Add exception to the nsg around the subnet to allow in rdp traffic just from our machine (not the whole internet). We do this in the networking blade of the vm. Add inbound port rule > Source is IP address > IP address if your IPv4 IP address (google what is my ip). Destination is just the IP address of the vm (NIC Private IP: 172.16.0.4). RDP Port is 3389. Priority is 1000. Name is rdp\_test001
* Add another disk for data and attach to the vm. Click refresh if the disk doesn’t show up immediately. For a disk to be shared across 2 different VMs it has to be P15 minimum (256Gb)



Things you need to have remote access to a vm

1. Public ip
2. Nsg
3. Security rule (inbound)

VM Sizes

| **Type** | **Common sizes** | **Description** |
| --- | --- | --- |
| General purpose | B, Dsv3, Dv3, DSv2, Dv2, Av2, DC | Balanced CPU-to-memory. Ideal for dev / test and small to medium applications and data solutions. |
| Compute optimized | Fsv2 | High CPU-to-memory. Good for medium traffic applications, network appliances, and batch processes. |
| Memory optimized | Esv3, Ev3, M, DSv2, Dv2 | High memory-to-core. Great for relational databases, medium to large caches, and in-memory analytics. |
| Storage optimized | Lsv2, Ls | High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases. |
| GPU | NV, NVv2, NC, NCv2, NCv3, ND | Specialized VMs targeted for heavy graphic rendering and video editing. |
| High performance | H | Our most powerful CPU VMs with optional high-throughput network interfaces (RDMA). |

VM vs Container

* VM has more management
* Container has faster boot times, more efficient
* VM can have multiple OSs on 1 piece of hardware
* VM has partitions
* VM has independent OS/kernel, where containers share the kernel of the underlying OS
* Containers use far less disk space
* Containers are portable

JIT (Just In Time Access)

* JIT enables time limited access to a VM
* JIT leverages Azure Security Center or can be configured o the VM itself
* Cen restrict access to a given port (22, 3389, 5985, 5986 – the last 2 are Windows RM)
* Leverages RBAC for who can request access
* Provide audit history of who and where they connected from

JIT Process

* Configure JIT Access
* Request access
* Audit the activity

Ports (public Inbound port rules)

* http – 80
* https -443
* rdp – 3389
* ssh – 22
* Public inbound port rules will allow all IP addresses to access the vm by default (unless further configured in the networking tab)
* A VM would have ports 80 and 443 open if it were running a web server
* Port 3389 open for remote desktop protocol access in Windows
* Port 22 open for linux ssh access
* You always apply port rules to the nsg and not to the vm nic itself
* Can also create rules in the networking tab when you create a vm in the portal

Virtual machine with Windows Server 2016 Datacenter

* Save up to 49% with a license you already own (Azure hybrid benefit)
* OS Disk type Premium SSD > Standard SSD > Standard HDD
* Can attach additional data disks or existing disk (disk storage)
* Can set NIC Network Security Group
* Management section – diagnostics
* Advanced
  + extensions are scripts for Windows and
  + Cloud Init are scripts for Linux

Create a VM programmatically

1. Create a resource group
2. Create the vm
3. Ensure remote access port(s) is open
4. Retrieve the public ip address

Create a VM in CLI

* az vm create --resource-group TutorialResources --name TutorialVM1--image UbuntuLTS --generate-ssh-keys
* az vm create --resource-group 'rg-event-7777' --name 'MyVM' --image 'win2016datacenter' --admin-username 'username' --admin-password 'Password123!#@' --location 'EastUS'
* az vm create --resource-group myResourceGroupVM --name myVM --location eastus2 --image UbuntuLTS --generate-ssh-keys --zone 1

Create a VM in Powershell

* $username = 'demoadmin'
* $password = ConvertTo-SecureString 'password123!@#' -AsPlainText -Force
* $windowsCred = New-Object System.Management.Automation.PSCredential ($username, $password)
* New-AzVM -ResourceGroupName rg-test-123 -Name MyVM -Image Win2019DataCenter -Credential $windowsCred -OpenPorts 3389
* Get-AzPublicIpAddress -ResourceGroupName rg-test-123 -Name MyVM | Select-Object IpAddress

Create a VM in PowerShell (with parameters)

* $vmParams = @{

ResourceGroupName = 'TutorialResources'

Name = 'TutorialVM1'

Location = 'eastus'

ImageName = 'Win2016Datacenter'

PublicIpAddressName = 'tutorialPublicIp'

Credential = $cred

OpenPorts = 3389

}

* $newVM1 = New-AzVM @vmParams

Connect to the virtual machine using CLI

* Get the IP Address of the VM
  + az vm show -d -g rg-testing-123 -n vm123321 --query publicIps -o tsv
  + az vm list-ip-addresses --name MyVM --o table
* Connect to the vm
  + mstsc /v:52.235.59.44 (windows rdp)
  + ssh username@ipaddress
  + ssh ipaddress (if ssh keys are generated locally)

Connect to the vm using Powershell

* Get-AzRemoteDesktopFile -ResourceGroupName "RgName" -Name "VmName" –Launch

Linux vms and ssh public key in CLI

* The default option when creating a Linux vm is ssh public key
* Create ssh (t is type, b is bits)
  + ssh-keygen –t rsa –b 2048
* Passphrase is an additional layer of security (can be empty)
* Get the key file
  + cat ~/.ssh/id\_rsa.pub
* Paste the key into the SSH Public key box when you create the vm in portal. Use existing public key. Keys look like
  + ssh-rsa <key>
* Login to the vm via ssh with the private key
  + ssh azureuser@ipaddress OR ssh azureuser@ipaddress OR fqdn
  + Also, ssh -i id\_rsa azureuser@ipaddress
* Once logged in, use the hostname to validate you are on the vm

Azure Compute Units (ACUs)

* Way to compare CPU performance between different types and sizes of VM

Adding Managed Identities to the vm PowerShell or cli

* $vm = Get-AzVM -ResourceGroupName myResourceGroup -Name myVM
* Update-AzVM -ResourceGroupName myResourceGroup -VM $vm -IdentityType $SystemAssigned
* az vm identity assign -g myResourceGroup -n myVm --identities [system] assignedId
  + [system] is the default, can also be user

Running scripts on your VM

* Custom Script Extension
  + primarily used for post deployment configuration and software installation
  + Scripts can be downloaded from Azure storage or GitHub, or provided to the Azure portal at extension run time
  + Useful for post deployment configuration, software installation, and other configuration or management tasks
* Run command
  + enables virtual machine and application management and troubleshooting using scripts
  + available even when the vm is not available
  + Useful for vm and application management and for running scripts in virtual machines that are unreachable.
  + You don’t need access via ssh or rdp
  + Script time 90 minutes max
* Hybrid Runbook Worker
  + provides general machine, application, and environment management with user's custom scripts stored in an Automation account
* Serial Console
  + provides direct access to a VM, similar to having a keyboard connected to the VM
  + Useful when access to the virtual machine is needed regardless of the machine's network or operating system state.

When you want to grant VM access to specific resource groups in Azure Resource Manager

VM - > Access Azure Resource Manager

* Log into the VM that wants access, and open cmd
* run the Invoke-RestMethod cmdlet to make a request to the local managed identity for Azure resources endpoint
* the response is in JSON, extract the token
* call Azure Resource Manager using the access token

Managed identities for Azure resources and how to acquire access token via vm

* Managed identities provide automatically managed identity in Azure Active Directory
* can use this identity to authenticate to any service that supports Azure AD authentication
* only accessible within a vm, or client app running on a vm that uses managed identities
* URL
  + http:// 169.254.169.254:50432/metadata/identity/oath2/token
  + http://169.254.169.254/metadata/identity/oauth2/token
* Response is Json string, which contains access\_token
* The token is suitable for use as a bearer token in service-to-service calls requiring client credentials

ARM templates

* a JSON file that defines the infrastructure and configuration for your project.
* allows you to create and deploy an entire Azure infrastructure declaratively
* For example, you can deploy not only virtual machines, but also the network infrastructure, storage systems and any other resources you may need
* Consistent deployments
* Declarative syntax
* Repeatable results
  + templates can be versioned
* Extensible
  + calling Powershell and Bash scripts during deployment
* Automation
  + Orchestration ensuring dependencies are met and deployed in the correct order
  + Also automated into the release Pipeline CI/CD

ARM Template deployment scopes (4 options)

1. Resource group (most common)
2. Subscription
   1. Can include Resource Groups, Policy definitions and assignments, RBAC Permissions on Subscriptions, RBAC Custom Roles
3. Management group
   1. Can include Resource Groups, Policy definitions and assignments, RBAC Permissions on Management groups, RBAC Custom Roles
4. Tenant
   1. Can include RBAC Custom Roles and Policy Definitions

* The resourceGroup() function is supported for resource group deployments
* The subscription() function is supported for resource group and subscription deployments
* The reference() and list() functions are supported for all scopes
* Use resourceId() to get the ID for a resource deployed at the resources group

Deployment Process

* Done through Azure CLI, PowerShell, or Azure Portal
* Steps
  1. Create and upload JSON File
  2. ARM Converts to REST API operation
  3. Sends instructions to resource provider

Deploy ARM template file via CLI

* Create file Azuredeploy.json

{

"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",

"contentVersion": "1.0.0.0",

"resources": []

}

* set templateFile="azuredeploy.json"
* az deployment group create --name blanktemplate --resource-group rg-arm-123 --template-file %templateFile%
* az deployment group create --name blanktemplate --resource-group rg-arm-123 --template-file %templateFile% --parameters %paramFile%

Deploy ARM template via Powershell

* New-AzResourceGroupDeployment -Name storagedeploy2 -ResourceGroupName $rg

-TemplateFile $templateFile

ARM Templates Structure

* Github - Azure / azure-quickstart-templates

ARM Template File Construction

* Schema
* contentVersion
* Parameters
* Variables
* Functions
* Resources
* Outputs

Schema – req’d

* Sets the json schema or the version of the template used in the file
* Types

1. RG deployment (at the rg level)
   * https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#.
2. Subscription deployments (at the sub level)
   * + https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#
   * Management group template
   * Tenant template
   * Parameters template

contentVersion – req’d

* Template Internal version example 1.0.0.0

Parameters - Optional

* Used to make templates reusable by eliminating hardcoding of names
* Also allow you to customize based on environment
* If a parameter doesn’t have a ‘default value’ or ‘value’ then it will ask for input at runtime –unless you use a parameters.json template
* Each parameter has a name, a type (string, int, etc;) and metadata (optional)

Variables - optional

* Used to simplify templates by enabling you to write an expression once and reuse it throughout the template
* Are calculated at runtime

Parameters, Variables usage

* [parameters{‘subnetName’)]
* [variables(‘subnetName’)]

Functions - optional

* Used to dynamically create values
* 2 types
  1. User defined (custom)
  2. Predefined (concat, contains, greater, etc)
* User defined functions have their own parameters and cannot access the regular parameters of the arm template.
* User defined functions cannot access variables
* Parameters on your custom function cannot have default values
* Call a user defined function using **namespace.function(params)**
* User defined functional cannot call other user defined functions, only predefined ones
* You cannot use the reference() predefined function.

{

"namespace": "marclab1",

"members": {

"uniqueName": {

"parameters": [

{

"name": "namePrefix",

"type": "string"

}

],

"output": {

"type": "string",

"value": "[concat(toLower(parameters('namePrefix')), uniqueString(resourceGroup().id))]"

}

}

}

}

Resources – req’d

* Resource(s) that are deployed or updated by the template
* One or more resources (can you actually have zero wtf?)
* Type is the type of resource - Mocrosoft.Network/virtualNetworks

Outputs - optional

* used to return values from the template
* outputs have names, types and vaules

ARM Parameter templates

* A separate file with parameters
* The name of each parameter in your parameter file must match the name of a parameter in your template (template specifies type, min/max length, etc; parameter template specifies the actual unique values for the instance of this resource)
* Typical to use **.parameters.json** file extension instead of .json

Example - azuredeploy.json and azuredeploy.parameters.json

Creating and Deploying ARM templates

* On the creation screen after creating a VM, click on Download a Template for automation
* Can edit the template in JSON Format on this page before it is saved
* Templates are accessible via the Templates blade in Azure Portal
  + can deploy, edit, delete, share and pin to the dashboard

DependsOn

* When a resource depends on another resource
* this resource dependsOn blahblah resource

Copy

* Added to the resources section of your template to deploy multiple instances of the resource
* Useful for VM Scale Sets
* Limit – the count cannot exceed 800

CopyIndex()

* A function that returns the current iteration in the loop (used in conjunction with copy which runs a loop based on the number of copies
* You use the index as the name prefix (ie – 1storage000, 1storage000, 3storage000)

Containers

* any change that you make to a container is not persisted across reboots
  + if you need to make changes, modify the image container and then redeploy the container

Docker Images (DockerFile)

* an immutable (unchangeable) file that contains the source code, libraries, dependencies, tools, and other files needed for an application to run
* Read only file that contains all the data required by the container to run
* Includes:
  + Source code
  + Dependencies
  + Libraries
* Command to see all images - docker images

Docker Containers

* An instance of an image
* Run-time environment (not at the OS or hardware level)
* virtualized run-time environment where users can isolate applications from the underlying system
* command to see all running containers – docker ps

Creating a Container with Docker

* The Docker file is a:
* Text file with all the set of instructions for building the container image
* Consists of a series of layers
  + Write the DockerFile
  + Build the Image
  + Deploy the container

Dockerfile Properties (FW CR C)

* FROM
  + creates a layer from the ubuntu:18.04 Docker image
* WORKDIR
  + sets the working directory for any commands in the DockerFile
* COPY
  + adds files from your Docker client’s current directory
* RUN
  + builds your application with make
* CMD
  + specifies what command to run within the container

Procedure for creating an image

1. Create a directory for the new image (mkdir)
2. Create the Dockerfile
3. Open a command line (so you can run Docker commands)
4. Create the container image (docker build) – can tag here also
5. List the newly created image (docker image ls)

Docker commands

* Help
  + Docker –help
  + Docker build --help
* Build (a local image before pushing to Azure) – dot at the end means the source image is in the current working folder
  + docker build –t node-docker-demo:latest .
* Tag and image (before pushing)
  + docker tag SOURCE\_IMAGE[:TAG] TARGET\_IMAGE[:TAG]
  + docker tag glennc/fancypants:latest marclab1/fancypants:v1
  + When you don’t specify a tag it will default to latest
* Push
  + docker push marclab1/hello-seattle
* Login
  + Docker login
  + Docker login -u marclab1 –p xxxxxxxx
* List images in local docker
  + Docker images (or docker image ls)
* Remove local docker images
  + Docker images rm –f marclab1/fancypants
* Run locally (Start)
  + docker run --name MyUbunti1 -it Ubuntu
  + docker run -d -p 80:80 my\_image
* Remove the container
  + Docker rmi
* Show running containers
  + docker ps
    - CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
    - fde9e1433946 ubuntu "/bin/bash" About a minute ago Up 59 seconds MyUbunti1
* Pause/Unpause a container
  + docker pause fde9e1433946
  + docker unpause fde9e1433946
* Kill a container
  + Docker kill fde9e1433946
* Edit a container inside VS Code
  + Navigate to the local folder using *cd*, then **code ..**

Demo: there is a container in github and we want to get it into Azure

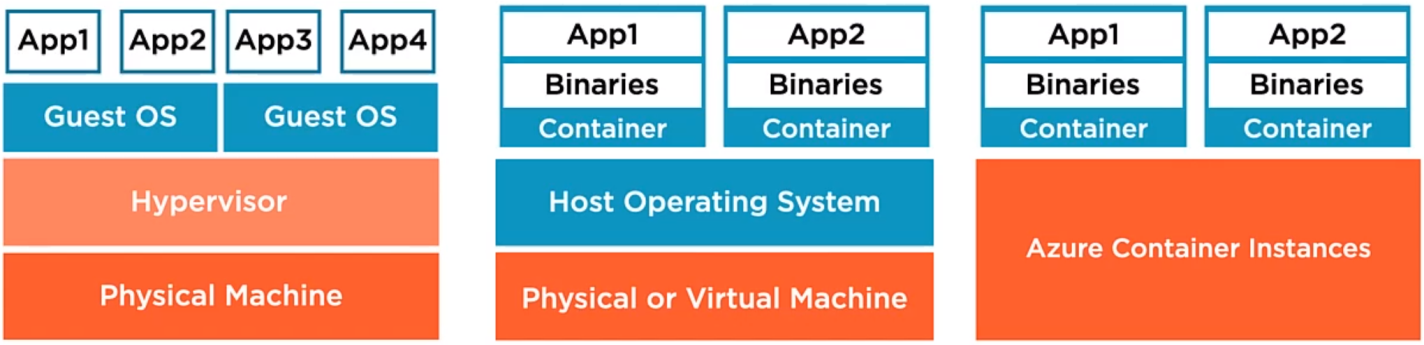
* Clone it into local folder
  + git clone https://github.com/Azure-Samples/docker-django-webapp-linux.git
* Then browse into the folder
  + Cd docker-django-webapp-linux
* Build that shit
  + docker build -t marclab1/django .
* Tag that shit with the url and version
  + Docker tag marclab1/django marclab1/django:v1
* Push that shit to docker
  + docker push marclab1/django:v1
* Push that shit to Azure Container Registry
  + az acr login --name containerregistrym111
  + In Access Keys blade, enable Admin User and may have to enter username and password
  + Tag it with the url of the acr registry
  + docker tag marclab1/fancypants:v1 containerregistrym111.azurecr.io/fancypants:v1
  + Now push to acr using docker
  + docker push containerregistrym111.azurecr.io/fancypants:v1

CLI docker App Service

* Az webapp create –plan %plan% --deployment-source-url image.azurecr.io/website:v1.0.0
  + Git repository URL to link with manual integration
* Az webapp create –plan %plan% --deployment-container-image-name images.azurecr.io/website:v1.0.0
  + multiple containers in the same container group - Linux only
  + Container image name from Docker Hub, e.g. publisher/image-name:tag.
* Az webapp config container set –docker-registry-server-url https”//images.azurecr.io –u admin -p admin
  + Set a web app container's settings.
  + –docker-registry-server-url is the container registry server url

Container Groups inside Virtual Networks

* deployments with Windows container groups are not supported in an Azure virtual network deployment
* Container groups deployed to a virtual network don't currently support exposing containers directly to the internet with a public IP address or a fully qualified domain name



Azure container instances (ACI)

* PaaS – no hardware to maintain
* Serverless containers
* A runtime for containers
* Only pay when it is running, billed by the second
* multi-tenant service, but containers are isolated and cannot see each other
* hypervisor isolation
* you can upload your container to azure for execution
* Linux or Windows containers
* Can be public or private
* TLS certs by default
* Monitoring and App Insights are enabled by default
* Ideally, only one app inside each container
* Uses Docker
* Ideal for simple use cases (web scripts, etc)
* Very ideal for burst scenarios
* Can specify cpu and memory sizes
* SLA - 99.9%
* Max nodes = 20
* 15GB max size
* Azure won’t let you deploy a container instance with the “wrong” OS (validation will pass but deployment will fail)
* If you don’t specify a dns, the container will only have an IP and not a FQDN

Container cli commands

* View logs
  + az container logs --resource-group myResourceGroup --name mycontainer
* Attach output streams
  + Az container attach
  + provides diagnostic information during container startup
* Get diagnostic events
  + Az container show

Azure Container Group

* ACI supports the deployment of multiple containers onto a single host using a container group
* Linux containers only (no support for Windows)
* A container group is useful when building an application sidecar for logging, monitoring, or any other configuration where a service needs a second attached process.

ACI Integrations

* Azure Kuberneties service
* Azure Logic Apps
* Azure Functions

Steps to run containers using ACI

1. Create as many images as your application needs to run correctly.
2. Upload or push your application images to a container registry.
3. Deploy the application.

Azure Container Registry (ACR)

* API for hosting images
* Microsoft’s implementation of a Docker registry service
* Managed private container registry service
* core component of CI/CD pipeline
* acts as a OCI repository (helm charts)
* built in automation functions
  + task scheduling, version control
* provides public and private repositories of container images
* different tiers(SKUs) can be purchased (basic, standard, premium)
* private registries only available for Premium SKU
* can be geo-replicated (with premium) - this is important both for resiliency and latency reasons
* can run jobs to build the container image
* ACR Tasks for container image automation
* encrypted at REST using an API key, but you can provide your own key (Customer-Managed key option)
* Basic version (SKU) default is 100Gb storage space, Premium is 500Gb
* No scaling options
* Other network integrations (LBs, app gateways) have to be configured separately
* Can deploy a repository from the container registry to a container instance or a web app
* Azure Container Registries have to be globally unique inside Azure
  + URL - containerregistrym111.azurecr.io

ACR Authentication

* ACR requires auth for operations
  + Azure A/D Identities (Users, Service principals)
  + ACR Admin
  + Orchestrators, tools and apps should use headless authentication
* CLI commands
  + Az acr login
  + Docker login
  + docker login myregistry.azurecr.io

ACR Role Based Authentication

* 2 types

1. People
   * Owner, Contributor, Reader
2. Service Principals
   * AcrPush, AcrPull, AcrDelete, AcrImageSinger

Azure container registry CLI

* Create the acr
  + az acr create --resource-group rg-containers-111 --name containerregistrymy111 --sku Basic
* Log into the acr
  + Az acr login –name containerregistrymy111
* Clone the git and create a folder locally
  + git clone https://github.com/Azure-Samples/acr-build-helloworld-node.git
* Build using ACR Tasks, tag and push in 1 step
  + Az acr build –image ipcheck:v44 –registry containerregistrymy111 .
* View repositories and images inside acr
  + az acr repository list --name containerregistrymy111
* Show tags inside repository image
  + az acr repository show-tags --name containerregistrymy111 --repository ipcheck
* Delete container image inside repo
  + az acr repository delete --name containerregistrymy111 --image ipcheck:v44

Az acr build vs docker build

* With acr build you don’t need a docker daemon on the local machine
* Can build, tag and push to acr in 1 step

Azure Web Apps for Containers

* Run Docker containers as azure web apps on Linux or Windows

Deploying containers in ACI from Container Registry

* Azure Container registry
* Docker Hub or other container registries
* Public or private registries
* Login server (containerregistrym111.azurecr.io)
* Auth credential (user name and password)

Creating a Service principal for ACI to pull from ACR

* SET ACR\_NAME =containerregistrymy111
* SET ACR\_REG\_ID=(az acr show --name %acr\_name% --query id --output tsv)
* Set sp\_name=acr-service-principal
* Set sp\_pass=(az ad sp create-for-rbac –name http//%acr\_name%-pull –scopes %acr\_reg\_id% --role acrpull –query password –output tsv)
* Set Sp\_appid=(az ad sp show –id http://%acr\_name%-pull –query appID –output tsv)

Running a container from ACR in ACI

* Set acr\_log=(az acr show –name --name %acr\_name% --query loginServer --output tsv)
* Az container create –resource-group rg-container-222 –name containerinstance222 –dns-name-label containerinstance222 –ports 80 –image %acr\_log%/ipcheck:v44 –registry-login-server %acr\_log% –registry-username %ap\_appid% –registry-password %sp\_pass%
* After pushing the Container Instance into the acr from the CLI, the Admin user must first be enabled for this registry in order to access the image during the contain instance creation. We do this in the Access Keys blade of acr

**Create Azure App Service Web Apps**

create an Azure App Service Web App

enable diagnostics logging

deploy code to a web app

configure web app settings including SSL, API settings, and connection strings

implement autoscaling rules including scheduled autoscaling and autoscaling by

operational or system metrics

Azure App Service

* PaaS – so less control over the hardware underneath
* Build enterprise grade web apps, mobile back ends, RESTful APIs in any platform
* Azure provides a fully managed secure and scalable platform (scale up and scale out)
* Supports multiple programming languages, runtimes, and containers (java, php, C#, Python)
* Focus specifically on code
* Linux or Windows (some programming languages only offer support for certain OSs)
  + **Python Linux only**
  + **ASP.NET Windows only**
  + .NET Core – either Win or Linux
* Custom domains (Blahblahblah.azurewebsites.net or another domain)
  + $11.99/year for App Service Domain
* SSO with Azure A/D
* Multiple deployment models – zip/war, ftp, github, onedrive sync, bitcucket, etc;
* Built in load balancing
* Not automatically backed up
* Lots of pricing options – different plans have different options (how many instances, how many deployment slots, etc)
* Supports content management systems such as Wordpress
* Can be monitored through Azure Monitor
* Host for container-based web apps (containers in Windows or Linux)
* Automatic platform management
* CI/CD integration
* Deployment slots
* SLA – 99.95% (with exception of Free and Shared tiers)
* Autoscaling both up and out
* Max nodes = 100
* url – webappname.azurewebsites.net
* will work with any Git repo

Deploy your web app into an app service using az webapp deployment

* Zip or war files
  + az webapp deployment source config-zip --resource-group rg-thomas-000 --name webappthomas000 --src clouddrive/webapp.zip
* ftp (ftp /s)
* cloud sync (from onedrive or dropbox folder)
* Continuous deployment (integrate with GitHub, BitBucket, or Azure Repos)
* Local Git repo
* External Git repo (deployment is only done once unlike the github CD option)
  + Az webapp deployment --repo-url $gitrepo –branch master –manual-integration
* ARM template
* Az webapp deployment source –help

ASP.NET vs .NET Core

* .NET Core is built for Windows, macOS, or Linux; ASP.NET is only Windows
* .NET Core has Higher performance than ASP.NET 4.x
* .NET Core has multiple versions per machine; ASP.NET only has 1 version per machine

App Service Plan

* Is a container – can hold zero or many App services
* Scope and scale of the App Service determines how much resources are available for your App Service(s)
* In App Service (Web Apps, API Apps, or Mobile Apps), an app always runs in an App Service plan
* Azure Functions also has the option of running in an App Service plan
* Can support Linux, Windows and container-based workloads
* Cannot mix Windows and Linux apps in the same App Service Plan
* Can use Azure AD, Google, Facebook, Twitter or other MS accounts for authentication
* defines a set of compute resources for a web app to run
* support CI/CD
* Tiers
  + Free (F)
  + Shared (D)
  + Basic (B)
  + Standard (S)
  + Premium (P)
  + Isolated (I)
* **B is the minimum tier to have SSL support**
* **F, D and B are not designed for production workloads**
* **Standard (S) is the recommended minimum for production workloads**
* **Shared (D) and up for custom domains**
* **Standard (S) and up supports auto scaling (scaling out)**
* **Standard (S) and up supports backups**
* **Standard (S) and up has slots (5 slots)**
* **Premium (P) will avoid cold starts with perpetually warm instances**
* **Isolated plans are inside a virtual network, so there is isolation at the network layer**

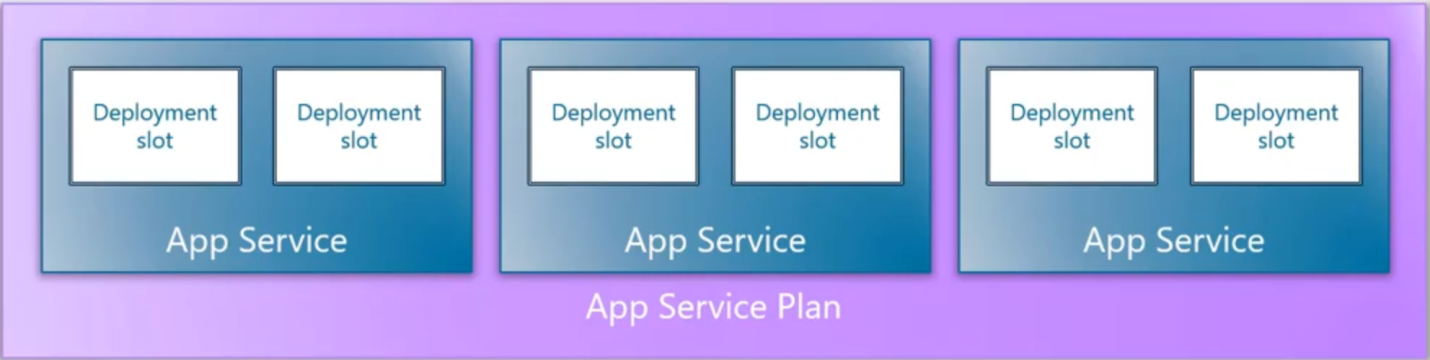
-> App Services must use an App Service Plan with the same OS, in the same region

-> resource groups created before January 21, 2021 **DO NOT** support having Windows and Linux apps in the same App service plan

-> resource groups created on or after January 21, 2021 support having Windows and Linux apps in the same App service plan

Deployment Slots

* One or more deployment slots per App Service
* Production slot is the default slot
* Staging
  + use a separate deployment slot instead of the default production slot when using an App Service Plan
  + Apps can be swapped between slots
* Can also use slots for load balancing to divide traffic (Traffic %) using separate slots
* errors occurred during a slot swap are logged in D:\home\LogFiles\eventlog.xml
* Az webapp deployment slots



Autoscaling for App Service Plans (Cannot scale for individual App Services)

* Different App Service Plans have different options
* In the Monitor Blade
* Manual Scale where an instance count can be set
* Autoscale (or Custom AutoScale)
  + Scale Rules – Metrics source, time aggregation, etc;
  + you configure the scale condition before you actually add the rule
* Settings that you configure in the Application Settings of an app service overwrite the values that you configure in the <appSettings> or <connectionStrings> in your Web.config or appsettings.json files.

Backup and Restoring App Services

* Handled by the standard back up service
* Requires an Azure Storage account to use as the backup
* Requires a non-free Standard or Premium plan for Snapshots or Backups
* Backup Configuration - storage, schedule, database (if your app uses a db)

Alerts can be configured from the Metric blade of App Service Plan

* Receive an Email when resources hit a certain threshold

App Service Environment (ASE)

* Customer VNet
* Inbound management traffic
* AppServiceManagement Service Tag
* Use firewall for outbound traffic control

CLI Commands for app service logging

* config : Configure logging for a web app.
* download : Download a web app's log history as a zip file.
* show : Get the details of a web app's logging configuration.
* tail : Start live log tracing for a web app.

Steps to create and deploy a web app CLI

* Create a rg
  + az group create --name rg-test-456 --location CanadaCentral
* Set Defaults (optional obv)
  + az configure --defaults group=rg-test-456 location=CanadaCentral
* Create app service plan (Basic is the default, is-linux is false)
  + az appservice plan create --name appserviceplan456 --resource-group rg-test-456
* Update app service plan to be free
  + az appservice plan update --name appserviceplan456 --resource-group rg-test-456 --sku FREE
* Create a storage account
  + az storage account create --resource-group rg-test-456 --name st123smell
* Create app insights
  + az monitor app-insights component create --app appinsights-webapp-456 --resource-group rg-test-456
* Create tags on a resource group
  + az tag create --resource-id /subscriptions/96c09541-ee61-4e57-94df-16d6ed62e757/resourcegroups/rg-test-456 --tags Class=AZ204
* Get the id of a resource
  + az appservice plan show --name appserviceplan456 --query id
* Create tags on a resource
  + az tag create --resource-id "/subscriptions/96c09541-ee61-4e57-94df-16d6ed62e757/resourceGroups/rg-test-456/providers/Microsoft.Web/serverfarms/appserviceplan456" --tags Class=AZ204
* Create an app service
  + az webapp create --name webapp456smelly --resource-group rg-test-456 --plan appserviceplan456 --runtime "DOTNETCORE|3.1"
* Delete the app service (also deletes the app service plan)
  + az webapp delete --name webapp456smelly --resource-group rg-test-456
* Deploy to the webapp
  + Set gitrepo=https://github.com/Azure-Samples/php-docs-hello-world
  + az webapp deployment source config --name webapp4545 --resource-group rg-webapp-4545 --repo-url %gitrepo% --manual-integration
* you have to delete the web app before you delete any appservice plan hosting that webapp. Both in cli and in portal you will get a warning

Create a web app PowerShell

* Connect-AzAccount
* New-AzResourceGroup -name rg-webapp-999
* new-azappserviceplan -name webapp999 -Resourcegroupname rg-webapp-999 -location CanadaCentral -Tier FREE
* new-azwebapp -name webapp999111 -Resourcegroup rg-webapp-999 –location canadacentral –appserviceplan webapp999

ASP.NET App Service access to users using the IApplicationBuilder Interface

1. Use Authentication -> Login
   1. Validate users
2. Use Authorization -> Authorization
   1. Permit users to access resouces
3. UseAzureAppConfiguration -> Dynamic App settings
   1. allows dynamic configuration to be reloaded without an app restart

App Service Configuration Blade

* Application Settings
  + Environmental variables
  + Connection strings
    - Configuring connection strings for the db
* General settings
  + Stack settings, stack & version
  + Platform settings
  + Debugging
  + Incoming client certificates
* Default documents
  + configures which web page is displayed at the root URL of your app
* Path Mappings

Securing a web app with SSL

* Need to use Basic App Service plan or higher
* Public vs private certificates
  + 4 options
    - Import app service certificate
    - upload certificate
    - import key vault certificate
    - create app service managed certificate
* Managed vs unmanaged certificates
* You can enforce HTTPS and minimum TLS versions (1.0, 1.1, 1.2)
* TLS/SSL settings blade in your app service
  + 2 types
    - SNI SSL (Server Name Indication) lets you run multiple SSL certificates from multiple domains on the same IP
    - IP Based SSL

Use a TLS/SSL certificate in your code in Azure App Service

* To access a certificate in your app code, add its thumbprint to the WEBSITE\_LOAD\_CERTIFICATES app setting
* This setting makes the specified certificates accessible to your Windows hosted app in the Windows certificate store, in Current User\My
* Also makes certificates accessible to your Windows or Linux container apps as files
* Prerequisite – app service plan has to be Basic or above

Azure App Service authN and authZ users

* App service has built in support for use authN and authZ

1. Can use http headers
   * HttpContext.Current.User property
     + Gets or sets security information for the current HTTP request
2. Can use ./auth/me http endpoint

WebApp Always on property

* Keeps your site up and running even if idle for too long (no cold start)
* Will prevent both idleTimeout and periodRestart
* Only available for App Service Plan Standard tier or above

Configuring a db connection string

* In the overview tab of a sql db, click on connection string
* In the configuration blade of the web app Add/Edit connection string

make sure to update password in the connection string

Type should be SQLAzure

Click save

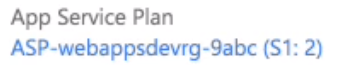
* I don’t know if this works with CosmosDB?
* Access in ASP.NET with Configuration.Get("Data:DefaultConnection:ConnectionString") where DefaultsConneciton is the connection string variable

Deploy code to app service web apps

* Configure Continuous Deployment
  + App Service pulls code from Github, ButBicket and Azure Repos
  + Authorize Azure App Service to retrieve the code
  + Can Enable continuous deployment
  + Can Disable continuous deployment

Scaling Azure App Service

* Vertical scaling
  + Upgrading cpu, ram, etc
  + This happens in the app service plan
* Horizontal scaling
  + If doing manual scaling, then set up Azure load balancer to handle 2 or more virtual machines
  + Auto scaling options (only available for Standard tier and above)
    - Scale out rules – average cpu > 70% for 5 minutes
    - Scale In rules – average cpu < 20% for 5 mins
    - Scale out schedule – 9:00AM – 3 instances
    - Scale in schedule – 5:00PM – 2 instances
    - Decrease count to x or decrease count by x
* Ideally, you want scale out rules and scale in rules to work in pairs and to be based off the same metric



* S1 is the App Service Plan Tier
* 2 is the instance count
* In the Scale up/down blade it will show you the max instance count (Free and Shared 1, Basic 3, Standard 10, isolated 100)

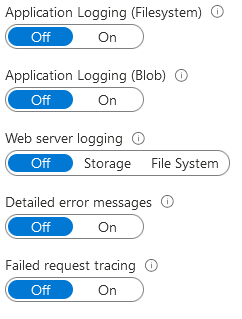
Designing Autoscaling Rules – Scale out blade

* Metrics (resource/custom)
* Time (schedule)
* Rules trigger actions such as increase or decrease VM instances
* Can also invoke actions such as sending emails to notify the team
* Or ping a webhook to start some automated process
* depends on Azure virtual machine scale sets
* can apply auto scale rules to only some Azure Resource types
* Current scale set is for host-based metrics that do not require additional agents to be installed or configured
* Vertical scaling affects the availability of the application because the application needs to be deployed in a virtual machine with the new resources assignment

Diagnostics == Logging

App Service Logging (Diagnostic Logging)

* In the App service Logs blade of the web app (under Monitoring)
* Logging is accessed locally on the vm where your instance is running
* 5 types
* Application Logging (File system)
  + Intended for debugging purposes
  + Turns itself off after 12 hours
  + Logs produced by the app itself
  + Windows or Linux
* Application Logging (blob)
  + Specified container inside blob storage account
  + Specify the retention period in days
  + .NET only, no support for Java, PHP, Node.js or Python
  + Windows or Linux
* Web Server Logging
  + Store diagnostic information generated from the web server itself
  + registers the raw HTTP transaction information for the requests made to the web server
  + system or file system options
  + Windows only
* Detailed Error messages
  + Also from the web server
  + HTTP status code 400 or greater (400-499 is client error, 500-599 is server error)
  + HTML files is generated, containing all information for each error
  + Windows only
  + Stored in the file system
* Failed request tracing
  + Also from the web server
  + detailed information about failed requests to the server
  + Windows only
  + Stored in the file system
* Deployment Logging
  + Logs for when you push content to an app
  + Windows and Linux
  + automatically enabled for you (so it doesn’t appear in the list in the portal)
  + no configurable settings
* You can use Blob Storage only with .NET application logs. If you use Java, PHP, Node.js, or Python, you need to use the application log file system option.
* Windows supports all 5 logging types, Linux only Application logging(Filesystem) and Deployment logging



Downloading logs

* 2 ways
  + FTP /S
    - Files are stored in the /LogFiles/ Folders
  + Azure CLI
    - az webapp log download --resource-group rg-test-000 --name webapp000
    - files are compressed into webapp\_logs.zip file

App Logging CLI

* Configure web server logging to the filesystem
  + Az webapp log config –name webapp1212 –resource-group rg-webapp-1212 –web-server-logging filesystem
* Configuring app logging to Azure Blob storage (Windows only)
  + Az webapp log config –name webapp1212 –resource-group rg-webapp-1212 –web-server-logging azureblobstorage
* Configuring app logging to the file system when working with Docker (Linux only)
  + Az webapp log config –name webapp1212 –resource-group rg-webapp-1212 –docker-container-logging filesystem
* Tail logs from app service app
  + Az webapp log tail –name webapp1212 –resource-group rg-webapp-1212
* Tail and filter logs from app service app
  + Az webapp log tail –name webapp1212 –resource-group rg-webapp-1212 –filter Error

Error Log levels

* Disabled - No errors are registered.
* Error - Critical and Error categories are registered.
* Warning - Registers Warning, Error, and Critical categories.
* Information - Registers Info, Warning, Error, and Critical log categories.
* Verbose - Registers all log categories (Trace, Debug, Info, Warning, Error, and Critical).

Log Streams

* In the Log streams blade of the web app
* CLI command
  + az webapp log tail --name appname --resource-group myResourceGroup

Transient Fault

* Any fault that is likely self-correcting and is caused by a temporary loss of connection

How to deal with transient faults

* App should log transient faults
* A retry strategy should be in space where needed
* Retry logic is already built into most SDK interactions
* Implement architectural patterns that help with transient faults
  + Retry pattern
  + Circuit breaker pattern

Docker environment variables for app service

* WEBSITES\_CONTAINER\_START\_TIME\_LIMIT
* WEBSITES\_ENABLE\_APP\_SERVICE\_STORAGE
* WEBSITE\_WEBDEPLOY\_USE\_SCM

App Service Plan metrics (these are the most common to help us identify performance issues)

* Cpu percentage
* Memory percentage
* Data in
* Data out
* Disk queue length
* http queue length

Azure App Service Mutual TLS Authentication

* Not supported on free or shared tiers
* Stores certificate on X-ARR-ClientCert header (Http request header)
* Available through the HttpRequest.ClientCertificate property
* Certificate value is Base64 encoded
* App code is required to validate certificate
* If you access your site over HTTP and not HTTPS, you will not receive any client certificate.
* So if your application requires client certificates, you should not allow requests to your application over HTTP

**Implement Azure functions**

create and deploy Azure Functions apps

implement input and output bindings for a function

implement function triggers by using data operations, timers, and webhooks

implement Azure Durable Functions

implement custom handlers

Azure Functions – an extension of Azure App Service

* Paas
* Azure Function App is a container for one or more functions
* Ideally a function does one thing
* Code + Events
* Every function has exactly one trigger, which is the event that cause the function to run
* Event driven services that execute on demand
* Serverless web hosting platform / serverless computing
* Runs a small piece of code instead of a full app
* Languages – C#, .NET .NET Core, Java, JS, Python, etc
* Event, timer, message, or another trigger driven event (http, schedule, etc)
* Typical usage scenarios - IoT, processing data, and more
* Micro or nano services
* Requires little maintenance from the IT team
* Azure Functions offers built-in integration with Azure Application Insights to monitor functions
* Functions use Triggers and Bindings to run
  + A trigger is an action or event from an upstream source that causes the function to execute its code
  + A function must have exactly one trigger
  + Binding are properties that connect functions to other Azure resources that can be an input or output or both
  + Allow you to pass data to and from
  + Bindings are optional, can have zero or many
  + Binding parameters must include type, direction, and the name of the binding
  + Functions can have multiple inputs and multiple outputs
* Bring your own dependencies (NPM or NuGet)
* Integrated security
* Can run inside Docker containers
* Flexible development
  + Can develop in the portal or Visual Studio Code or any other tool
  + Integration with Github, Azure DevOps
* Open-Source
* Built in Azure Application Insights to monitor functions (this service is optional and not free)
  + collects log, performance, and error data
* Best practices – functions should be stateless

App Service tiers

* Consumption (PAYG)
  + First 400,000Gb/s of memory and first 1 million executions are free
  + Automatic scaling
  + **5 minute default time limit for each function invocation, can be up to 10 mins**
    - **Only 2.5 mins for http and cannot be changed regardless of tier**
  + Cold start affecting latency can be an issue (can be up to a few minutes)
    - up to a 10-minute delay in processing new blobs if a function app has gone idle
  + **auto scale**
* App service plan (An existing App Service Plan)
  + you don’t pay for the number of executions, execution time, and memory used.
  + you do need to allocate resources for your functions explicitly and pay for them regardless if you are running functions or not (VM Instance, etc)
  + timeout duration is 30 minutes by default (can be set to -1 unbounded/unlimited)
  + Required if you run a custom image to run the function (ie – container)
  + **you need to turn on the Always On setting for your Function App to run properly**
    - Even with Always On enabled, the execution timeout for individual functions is controlled by the functionTimeout setting in the host.json project file
  + The Always on setting is only available for an App Service plan
  + **No autoscaling (customer is responsible)**
* Premium Plan
  + Your function apps run continuously, or nearly continuously; and no delay after being idle
  + **Eliminate cold starts**
  + **still 2.5min max for http**
  + **Can connect to VNet**
  + Timeout duration is 30 mins by default (can be set to -1 unbounded/unlimited
  + No max outbound connections per instance
  + **auto scale**
* **When running in a Consumption Plan or Premium Plan you should not enable Always On**
* **Enabling Always On on a Premium Plan will cause unnecessary scale out and generate 503 responses**

Azure Function Versions

* 1.x, 2.x, 3.x
* Backwards compatible
* 1.0 supports only .NET and Windows; 2.0 supports .NET Core and Windows/Linux

**Host.json**

* host.json metadata file contains global configuration options that affect all functions for a function app.
* Azure functions 2.x and later versions
* **functionTimeout property sets the timeout (min and max bounds)**
  + consumption – between 5 and 10
  + premium – 30 to -1 (unbounded)
  + app service – 30 to -1 (unbounded)

Functions CLI

* az functionapp create --name funapp123 --resource-group rg-test-123 --runtime dotnet --functions-version 3 --storage-account storagemy123 --consumption-plan-location CanadaCentral
* az functionapp delete --name funapp123 --resource-group rg-test-123

Function App Diagnostics

* Under the Diagnose and solve problems blade of Function App

Bindings

* A connection to data
* Bindings pass data to the function as parameters
* Configured in the function.json file
* 2 types of bindings

1. Input binding - An input binding is a connection to a data source. Our function can read data from these inputs.
2. Output binding - An output binding is a connection to a data destination. Our function can write data to these destinations. They are defined in JSON

* 3 properties are required in all bindings:

1. Name - Defines the function parameter through which you access the data
2. Type - Identifies the type of binding, i.e., the type of data or service we want to interact with.
3. Direction - Indicates the direction data is flowing, i.e., is it an input or output binding?

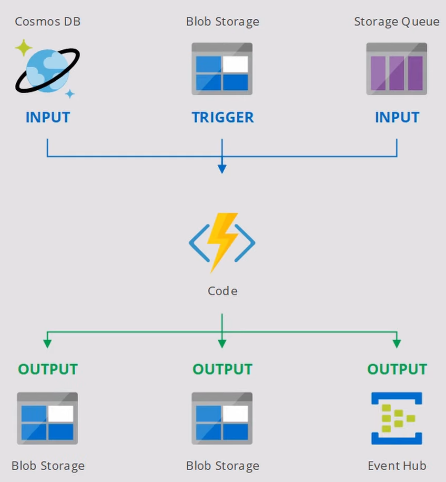
* Additionally, most binding types also need a fourth property:

1. Connection - Provides the name of an app setting key that contains the connection string

* Timer functions require a schedule property
  + "schedule": "0 \*/5 \* \* \* \*"

Common trigger types (always inputs)

* http (frequently used for APIs and respond to webhooks)
  + Can be customized via http methods (GET or POST) used to call the function
  + Secured via auth key
    - Anonymous – no key req’d
    - Function – key per this function (this is the default)
    - Admin – key per function app (All functions inside this function app)
* timer (schedule tasks)
* queue (run a function in response to a message on a queue
* storage (blob storage)
* cosmos db trigger (when a document is created or updated)
* service bus (topic & queue)
* events



* you need to install the extensions on your local environment before you can use bindings or triggers
* azure function runtime includes extension for http triggers and timer triggers, so no need to install anything

Trigger retries

* Max Retry Count is the maximum number of times an execution is retried before eventual failure
* value of -1 means to retry indefinitely
* The following triggers support retries at the trigger source
  + Azure Blob storage
  + Azure Queue storage
  + Azure Service Bus (queue/topic)
* By default, most triggers retry requests up to five times, including the first retry
* After the fifth retry, the Azure Queue storage will write a message to a poison queue.
* The default Service Bus queue and topic policy will write a message to a dead-letter queue after 10 attempts

Queue Trigger in Functions

* batchSize
  + The number of queue messages that the Functions runtime retrieves simultaneously and processes in parallel
  + Default is 16; Max is 32
  + Set batchsize to 1 to avoid parallel execution for messages received on one queue
* maxDequeueCount
  + The number of times to try processing a message before moving it to the poison queue
* newBatchThreshold
  + Whenever the number of messages being processed concurrently gets down to this number, the runtime retrieves another batch

Function.json file

* Each function has its own function.json file
* defines all triggers and bindings used by the function
* must be in a folder under the function name (funappthomas000/QueueTrigger/function.json)
* When using C#, function.json file is automatically constructed based on your code

{

"bindings": [

{

"authLevel": "function",

"name": "req",

"type": "httpTrigger",

"direction": "in",

"methods": [

"get",

"post"

]

},

{

"name": "$return",

"type": "http",

"direction": "out"

}

]

}

* Timer trigger, no binding

{

"bindings": [

{

"name": "myTimer",

"type": "timerTrigger",

"direction": "in",

"schedule": "0 \*/5 \* \* \* \*"

}

]

}

* Blob storage trigger

{

"bindings": [

{

"name": "myBlob",

"path": "samples-workitems/{name}",

"connection": "storagemy123\_STORAGE",

"direction": "in",

"type": "blobTrigger"

}

]

}

* Queue storage trigger & output

{

"bindings": [

{

"name": "myQueueItem",

"type": "queueTrigger",

"direction": "in",

"queueName": "messages-incoming",

"connection": "AzureWebJobsStorage"

},

{

"name": "$return",

"type": "queue",

"direction": "out",

"queueName": "messages-outgoing",

"connection": "AzureWebJobsStorage"

}

]

}

**Http requests**

* should be < 100mb
* url is limited to 4096 bytes
* **execution timeout is 230 seconds (2.5 mins)**

**HTTP trigger get and post and route**

* Get is req.Query()
* Post is req.body()
* By default the function is available at:

**http://<APP\_NAME>.azurewebsites.net/api/<FUNCTION\_NAME>**

Use route to change the url to a specific subdomain

Http responses

* HTTP 200 OK in case of Function 1.x runtime
* HTTP 204 No Content in case of Function 2.x

runtime

Http trigger methods

* By default, the function runtime accepts all http methods (get and post)

Blob blob trigger

* executes a function when a file is uploaded or updated in Azure Blob storage
* Under path
  + samples-workitems/{name}
    - samples-workitems represents the blob container that the trigger monitor
    - every type of file will cause the trigger to invoke the function with the parameter name
  + samples-workitems/{name}.png
    - would do the same but only for png files
* When a blob trigger function fails for a given blob, Azure Functions retries that function a total of 5 times by default.
* If all 5 tries fail, Azure Functions adds a message to a Storage queue named webjobs-blobtrigger-poison.
* No guarantee of time, can be delays up to 10 mins if using consumption plan

Best practices Azure Functions

* Avoid long running functions
* refactor large functions into smaller function sets that work together and return responses fast.
* For example, a webhook or HTTP trigger function might require an acknowledgment response within a certain time limit; You can pass the HTTP trigger payload into a queue to be processed by a queue trigger function
* Alternatively, you can use durable functions which are best suited for long-running function executions

Creating a Function from CLI

* Create the function app
* az functionapp create --resource-group AzureFunctionsQuickstart-rg --consumption-plan-location westeurope --runtime dotnet --functions-version 3 --name <APP\_NAME> --storage-account <STORAGE\_NAME>

Azure Timer Functions

* The basic format of the CRON expressions in Azure is:

{second} {minute} {hour} {day} {month} {day of the week}

* Examples

{second} {minute} {hour} {day} {month} {day of the week}

\* == every value

, == multiple values

\*/x == every x intervals

- == range

L == "Last" allowed for day of the week and day of the moth fields

W == "weekday" allowed for day of hte week

# == allowed for the day of the month field to specify the #th day of the month

every second --> \* \* \* \* \* \*

every minute --> 0 \* \* \* \* \*

every hour (at the 0th minute) --> 0 0 \* \* \* \*

every day --> 0 0 0 \* \* \*

Every 30 minutes during business hours --> 0 \*/30 9-17 ? \* 0-4

Every 30 minutes during business hours --> 0 \*/30 9-17 ? \* MON-FRI

Every minute on 11-11 --> 0 \* \* 11 11 ?

Every 30 seconds on sundays --> \*/30 \* \* ? \* SUN

Every month on every Monday, at noon --> 0 0 12 ? \* MON

Happy new years! --> 0 0 0 1 1 ?

Every month on the third Thursday of the Month, at noon --> 0 0 12 ? \* 5#3

Azure Function Storage accounts

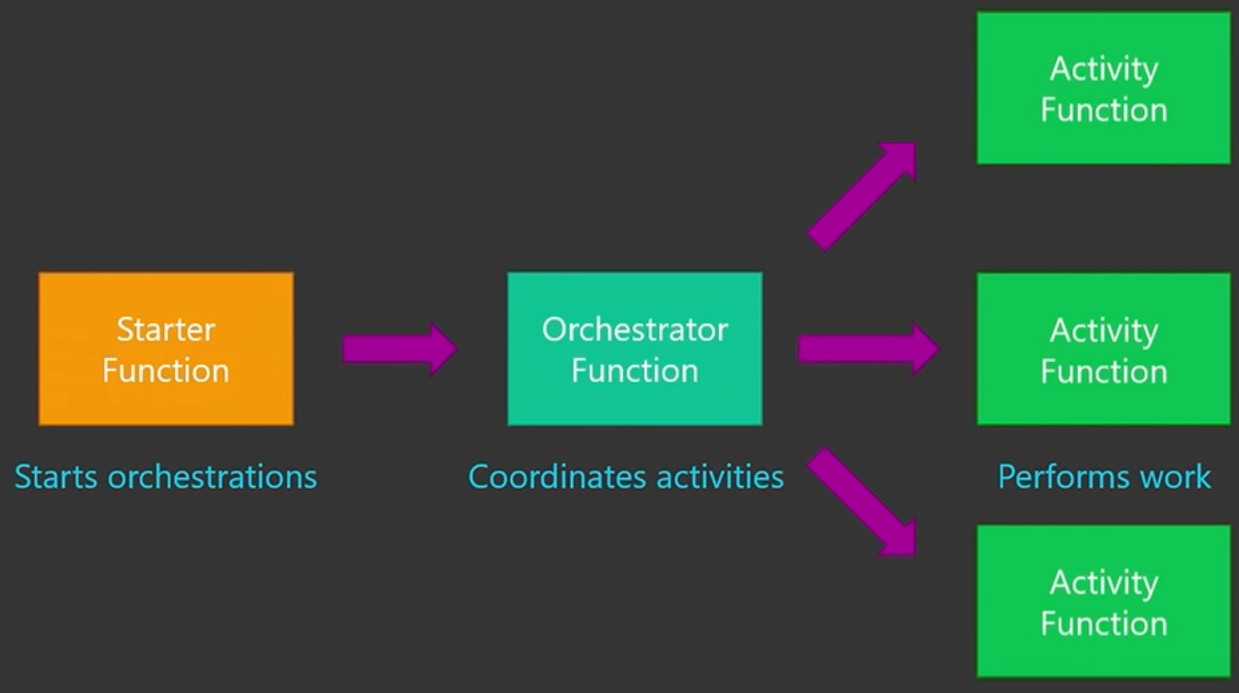
* Azure functions rely on storage accounts
* must create or link to a general-purpose Azure Storage account that supports Blob, Queue, and Table storage
* Has to be in the same region as the function (only when creating function in the portal?)
* Some storage accounts don't support queues and tables
  + blob, Azure premium storage and general purpose storage with ZRS
* Serverless computing typically requires a trigger
* Stateless function instances are created and destroyed on demand

Durable Functions

* an extension of Azure Functions and WebJobs
* serverless
* stateful orchestration of function execution
* can retain state between function calls.
* other functions can be called synchronously or asynchronously
* can manage restarts upon failure
* automatically manage checkpoints
* takes care of managing the state, checkpoint, and replays for you
* **a function that can run forever - no time limit for how long it runs (no timeout)**
* **typical usage for a function that has to run a longer period of time**
* supported languages – C#, JS, Python, F#, PowerShell
* Can be written in portal or Visual Studio
* 4 Logical types of functions
  1. Client (Starter or Entry function)
     + Can use any trigger type used in regular non-durable functions (http, queue, blob, etc)
  2. Activity
     + Implement a step or action in the workflow
     + Can use any input or output bindings
  3. Orchestration
     + Defines steps in the workflow
     + Controls the order that actions are executed
     + Can also handle errors
  4. Entity (not as common as the first 3)
* Durable Functions and Azure Logic Apps are built to manage state transitions and communication between multiple functions.
* Need the Microsoft.Azure.WebJobs.Extentions.DurableTask extention via NuGet

Durable Function triggers

* Orchestration trigger
  + Creating new instances of the function or resuming instances that were waiting for a task
* Activity triggers
  + When writing to activity functions
  + Allow communication between orchestration functions and activity functions
  + multithread



Durable Functions Application patterns

* Fan out/fan in
  + Single orchestrator function execute multiple functions in parallel
* Function chaining
  + pattern of executing a sequence of functions in a particular order
  + the output of one function needs to be applied to the input of another function
* Manual intervention (Human Interaction)
  + implemented using an orchestrator function
  + Example - Approval from a manager might be required for an expense report that exceeds a certain dollar amount
* Aggregator
  + Consolidates messages into single entity
* Monitoring
  + flexible recurring process in a workflow
  + uses timer
* **Async http (APIs)**
  + **HTTP endpoint trigger the long-running action. Then, redirect the client to a status endpoint that the client polls to learn when the operation is finished.**
* When using durable functions, you can pass information between the different functions in the workflow by using the binding mechanism

**Develop for Azure storage (15-20%)**

**Develop solutions that use Cosmos DB storage**

select the appropriate API and SDK for a solution

implement partitioning schemes and partition keys

perform operations on data and Cosmos DB containers

set the appropriate consistency level for operations

manage change feed notifications

**Develop solutions that use blob storage**

move items in Blob storage between storage accounts or containers

set and retrieve properties and metadata

perform operations on data by using the appropriate SDK

implement storage policies, and data archiving and retention

Cosmos DB

* globally distributed, multimodel database service
* turnkey global distribution – quickly replicate data to a new location
* elastic storage and throughput
* Semi structured, Non-relational db (nosql, JSON)
* Key-value pairs
* can scale throughput and storage across Azure regions.
* highly responsive at less than 10 ms latency and highly available at 99.999% or five nines.
* Automatically replicate data closest to the users and that's what makes it highly responsive and highly available.
* Typical uses include web, mobile, gaming, and IoT applications.
* Multiple APIs (SQL, MongoDB, Table API, Cassandra, Graph)
* Azure Free account allows you to create only 1 free tier cosmos db
* id is the unique field and is auto generated by cosmos, unless the user manually specifies the id field
* pricing is based on throughput (customizable) 400 RU/s is the default
* support multiple consistency options

NoSQL Databases

* Suited for large amounts of transactional volumes (many thousands of transactions per second)
* Fluid schema
* Multiple structures (key-value, graph, document, wide-column)
* Horizontal scaling and data partitioning for scalability
* Provides BASE (basic availability, soft state, eventual consistency) semantics
* Non-normalized data

Cosmos DB Features

* Integrated Analytics (Spark)
* Region support
* Schema-agnostic
* Automatic indexing
* Supports multiple SDKs



Account

* URL - blahblah.azure.com
* Read only Keys used to access cosmos
* Defines the consistency level, global distribution, and the API you are using

Database

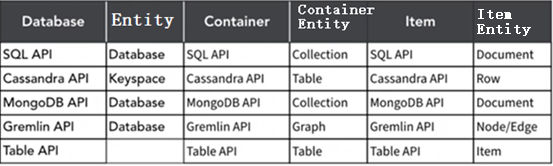
* One or many dbs per account
* Unit of management
* Can Provision throughput (performance)

Containers

* One or many containers per db
* Need to set the partition key
* Provision throughput (more common than Database level)
  + The minimum throughput for containers is 400 RU/s
* Throughput modes
  + Dedicated
  + Shared
* Properties
  + IndexingPolicy
    - How to index the items in the container
  + TimeToLive (TTL)
    - Delete items after a period of time automatically
  + ChangeFeedPolicy
    - Read changes made to an item in a container
  + UniqueKeyPolicy
    - Configure which property of the item is used as the unique key
* Once a container has been created you cannot change properties

Item

* Individual record
* Includes system defined properties
* Cannot mix APIs in a single Cosmos DB account
* Cannot change API once account has been created



Cassandra API Use cases

* You want to leverage the Cassandra Query Language (CQL) to query data
* Column based db that uses Key-value
* You have existing Cassandra db’s you want to migrate to the cloud
* You want to store data in a wide-column format (2D key-value store)
* SELECT id, productName FROM catalog.items

Mongo db use cases

* You want to leverage MongoDB API to query data
* You have existing Mongo db’s you want to migrate to the cloud
* You want to store data as JSON documents
* Lack of support for SQL like queries
* db.Items.find({},{productName:1,\_id:0})

Gremlin (Graph) API use cases

* Graph based view over the data
* Store graph relationships between data
* Relationships that require metadata to be stored for them
* Need to assign weight values to the relationships between data
* Can leverage Apache Tinkerpop’s Gremlin language for querying relationships
* g.V().hasLabel('product').has('productName', 'Industrial Saw').outE('boughtWith')

Table API use cases (replaces Azure Table Storage)

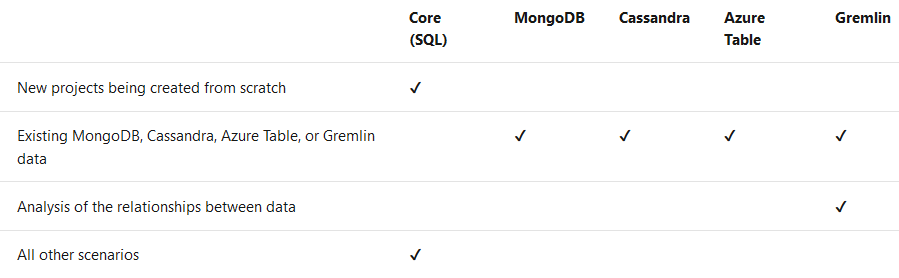
* Querying data using OData or LINQ queries
* can migrate from your current Azure Table Storage service with no code modification in your application
* can define your own indexes in your tables
* key-value pairs
* SELECT i.productName FROM Items i

Core SQL API Use cases

* The default for Cosmos
* Resembles a NoSQL
* Want to leverage a SQL-like language to query data
* Want to store data as JSON documents
* When the schema changes a lot
* Simple key-value pairs
* If no other uses cases fit, choose the SQL API
* SELECT \* FROM c where c.id="1"

How to select an SDK for Cosmos DB

* If you’re using SQL API, use the latest CosmosDB sdk for your platform
* If you’re using Mongo, Cassandra and Gremlin, use current sdks for those APIs
* If using Azure table API leverage the current table storage sdk



Cosmos DB RBAC

* DocumentDB Account Contributor
  + Can manage Azure Cosmos DB accounts.
* Cosmos DB Account Reader
  + Can read Azure Cosmos DB account data.
* Cosmos Backup Operator
  + Can submit a restore request for Azure portal for a periodic backup enabled database or a container. Can modify the backup interval and retention on the Azure portal. Cannot access any data or use Data Explorer.
* CosmosRestoreOperator
  + Can perform restore action for Azure Cosmos DB account with continuous backup mode.
* Cosmos DB Operator
  + Can provision Azure Cosmos accounts, databases, and containers. Cannot access any data or use Data Explorer.
  + Cannot access the keys that are required to access the data.
* applies to control plane operations only
* Data plane operations are secured using primary keys or resource tokens

Partitions

* Logical Partitions
  + All items stored in a logical partition share the same partition key
  + Limit of 20Gb by default
  + No limits in the number of logical partitions
* Physical Partitions
  + Automatically managed by Azure
  + A physical partition can contain 1 or more logical partitions
* Partitions exist inside containers

PartitionKey in Cosmos DB

* Items in our containers are divided into Logical partitions
* Must be set when we create a new container
* Logical partitions are mapped to physical partitions behind the scenes
* Keys are used to scale containers to meet performance needs
* The idea is to distribute the amount of storage or throughput efficiently (evenly)
* Partitions have a path, which is the property name (/CityName) and a value, which is the property value (Toronto)
* Cannot change partition key once created. Must create a new container and migrate records over to the new container should you need to change.

Partition key best practices

* Use a partition strategy to evenly spread throughput on partitions
* Provision throughput at the container for predictable performance (not at the db level)
* Use the serverless account type for dev workloads
* **Understand the link between consistency type and the number of RUs consumed**
  + **The strong and bounded staleness consistency levels consume approximately two times more RUs while performing read operations when compared to that of other relaxed consistency levels**
* have a partition key with many distinct values, such as hundreds or thousands, if that property doesn’t exist, use a synthetic key

Partition key considerations

* Serves as the means or routing your request to the correct partition
* Made up of both the key and value of the defined partition key
* Avoid hot partitions that receive most of the requests
* Should be a value that does not change for the item
* Should have many different values represented in the container
* Throughput should be distributed evenly across all your physical partitions
* Multi-item transactions require triggers or stored procedures
* You will want to minimize cross partition queries for heavier workloads
* Decide upon a partition key strategy before creating your container
* A field that will be load balanced would be an ideal primary key

**synthetic partitionKey**

* **concatenating multiple property values into a single artificial partitionKey property**
* ideal for workloads that have millions of documents and required performance needs of the application is to evenly spread the workload
* evenly spread across all partitions over time

CLI

* Create a sql api cosmos db account
  + Az cosmosdb create –name pluralsight –resource-group pluralsight
* Create a sql database
  + Az cosmos sql database create –account-name plauralsight –name sampled
* Create a sql database container
  + Az cosmos sql container create –resource-group pluralsight –account-name pluralsight –database-name sampled –name samplecontainer –partition–key-path /employeeid

C#

* NuGet Package Microsoft.Azure.Cosmos
* Create a new instance of the Cosmos Client
  + this.cosmosClient = new CosmosClient(EndpointUri, PrimaryKey)
* Create new db
  + Await client.CreateDatabaseIfNotExistsAsync(databaseId);
* Create new Container
  + Await database.createContainerAsync()
* Create new item, insert into container
  + Await container.CreateItemAsync<Family>
* Query with filters
  + .GenerateFilterCondition("PartitionKey", QueryComparisons.Equal, "Smith")

Cosmos ‘id’ field

* The system property that exists in every item in your container
* When you create a document, you need to provide an Id property of type string. This property

needs to identify your document inside the collection uniquely. If you don’t provide this property, Cosmos DB automatically adds it to the document for you

* 404 (Not found)
  + if you try to delete a document in the container that does not exist
* 409 (conflict)
  + If you try to insert an item in the container with a non-unique id

When to use id as the partition key

* When your app performs read workloads and uses parallelization techniques for getting the data
* Small read heavy containers or write heavy container of any size
* Cosmos DB can be configured for replicating data between available Azure regions
* Therefore, there is a trade off between consistency and performance

**Cosmos Composite Indexes**

* **Queries that have an ORDER BY clause with two or more properties require a composite index**
* If the composite index paths do not match the sequence of the properties in the ORDER BY clause, then the composite index can't support the query.
* The order of composite index paths (ascending or descending) should also match the order in the ORDER BY clause.
* The composite index also supports an ORDER BY clause with the opposite order on all paths.

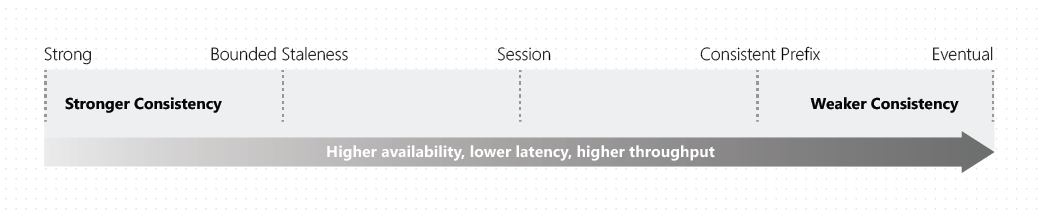
Data Consistency Levels (config’d in Default Consistency in the account level

* **Strong**
  + Also called immediate consistency
  + **guarantees that reads get the most recent version of an item**
  + only one that offers a linearizability guarantee **(in order)**
  + **data is consistent for all observers of the entity**
  + data has to be locked during the period of update or replication process to ensure that no other processes are updating the same data
  + **reduced availability (lag)**
* **Bounded staleness** 
  + **guarantees that a read has a max lag - either number of versions (K) or time (T)**
  + Maximum Lag (Operations) or Maximum Lag (time)
  + apps that expect low write latencies but require total global order guarantee
  + great for apps featuring group collaboration and sharing, stock ticker, publish-subscribe/queueing, etc;
* Session
  + The default consistency
  + Most balanced
  + guarantees that a client session will read its own writes in a session
  + best fits apps in which write operations occur in the context of a user session
* Consistent prefix
  + guarantees that updates are returned in order, but no guarantee you can read all the data
  + Example – if you write “A, B, C” you can read “A”, “A, B” or “A, B, C” but never “A,

C” or “B, A, C.”

guarantees that reads never see out-of-order writes

* Eventual
  + provides no guarantee for order of data
  + **the only one with out of order reads**
  + high availability and better performance (lowest latency)
  + different data (inconsistent) available in different regions
  + Use this consistency level if the order of the data is not essential for your app, such as non-threaded comments
  + When data is written, it will eventually be in sync with other instances
  + **all reads of the entity will eventually return the last updated value**



Consistency Levels for SQL APIs

* Configured on your account and applies to all databases and containers under that account
* Can also be request specific level
* Can be changed at any time
* For Gremlin and Azure Table APIs, CosmosDB uses account default consistency level
* For Cassandra write, the cosmos db account default consistency level is used
* For Cassandra reads, the client consistency is mapped to a Cassandra client driver
* For MongoDB, the write concern uses the account default consistency level
* For MongoDB, the read concern uses a mapping to a CosmosDB level read request

Throughput Considerations

* Both strong a bounded staleness reads will consume twice the normal amount of request units for a request, as Cosmos db will need to query 2 replicas to meet the criteria of the consistency level
* Avoid the 2 extremes (Eventual and Strong) – it is recommended instead to use Consistent Prefix or Bounded Statelessness, respectively
* If there are no write operations on the database, a read operation with **eventual, session,** or **consistent prefix** consistency levels is likely to yield the same results as a read operation with strong consistency level.

Cosmos DB throughput and scaling

* Vertical scaling –add more resources (cpu/memory)
* Horizontal scaling – add more servers
  + When your RU exceeds a physical partition, Azure will automatically add another physical partition and re-allocate your logical partitions

Request Unit (RU)

* Encapsulation of many of the resources needed for the db into a single unit
  + Cpu, memory, IOPS (input/output operations per second)
* 1 RU == 1kb item read operation from a Cosmos db container

Managing Cosmos DB throughput

* Provisioned
  + Setting a specific amount
  + Ideal for always on production implementation
  + Can be config’d at the db or container level
  + Evenly distributed to partitions
  + Requires 10RUs per GB of storage
  + Once RUs are consumed for a partition, future request will be rate limited
  + By default, it requires manual scaling to acquire more RUs
  + Can be set to Autoscale
    - Specify a max RU and CosmosDB will ensure that your data is available up to that throughput amount
    - The min is calculated as 10% of the max
* Serverless (preview)
  + Pay only for units consumed and the storage that you use
  + Ideal for off and on workloads
  + Max of 5000RUs
  + Requires a new account type
  + Only supports SQL (Core) API

CosmosDB Server-side concepts

* Stored procedures
  + Must be defined in js
  + Executes on a single partition, and it only has access to that partition
  + Partition key must be provided with the execution request
  + Supports a transaction model as all statements will be removed if it fails
* Triggers
  + Must be defined in js
  + Executing before(pre) or after(post) data is written to the db
  + Pre triggers can handle data transformation and validation
  + Post triggers can handle aggregation and change notifications
  + Not guaranteed to execute, as they have to be specified in a request
  + Errors in either pre or post trigger will result in data being rolled back
* User Defined Functions (UDF)
  + Must be defined in js
  + Enables you to define a custom function that can be leveraged in a query
  + Enables encapsulations of common logic in query conditions
* Change Feed
  + Enables you to react to data changes using server side code outside of the Cosmos DB engine
  + Enables notifications for any insert and update on your data
  + Deletes are not directly supported, but can leverage a soft delete flag
  + A change will appear exactly once in the change feed
  + Reading data from the db will consume throughput
  + Partition updates will be in order, but between partitions there is no guarantee
  + Not supported for Azure Table API
  + Change feed items come in the order of their modification time, guaranteed per logical partition key
* The first 3 are javascript based and running inside your CosmosDB environment
* Change feed can be run inside your own environment (on prem or Azure functions)

Azure Storage Account

* A collection of services (blob, queue, table, file and disk)
* Designed to allow customers to store files at high scalable in Azure
* High durability
* Accessed via http, Azure Rest API, powershell or azure cli
* Cheapest storage per GB
* Always encrypted at rest – you can choose your own key via Key Vault
* Lowercase characters only
* Hierarchy:
  + Account -> Container -> Blob
* 2 PB (Peta bytes) max storage amount
* Name has to be unique as it is globally accessible
* Account types
  1. general purpose storage account v2
     + Premium SSD ideal for unmanaged virtual machine disks
     + Standard HDD ideal for storing blobs, files, tables, queues, and Azure virtual machine disks
  2. blob storage account (legacy now)
* Performance
  1. Standard (Storage account v2)
  2. Premium (low latency scenarios – page blobs, file shares block blobs)
* Storage V2 supports Blob, File, Queue, Table, Disk, and Data Lake Gen2
* BlobStorage and **BlockBlobStorage** only supports blob storage
* FileStorage supports only File storage types
* So, we almost always choose Storage V2

Azure Storage V2 upgrading

* Upgrading to a general-purpose v2 storage account from a general-purpose v1 or Blob storage accounts can be done form the Azure portal, PowerShell, or Azure CLI
* Upgrade is done through the Configuration blade
* Storage V1 does not have all the features of Storage V2 or the lowest capacity pricing
* Storage V1 does not have hot, cool, achieve access tiers
* Storage V1 does not support ZRS replication
* This upgrade cannot be undone
* can configure a lifecycle management policy for automatically moving to the archive tier for those files that have not been accessed for some time
* A Blob storage account stores unstructured object data as block blobs and append blobs, but not page blobs

Copy files between 2 storage accounts

* You can copy blobs and containers between Storage Accounts in different regions or even

Subscriptions if you have enough privileges for accessing to both accounts

Blob copy CLI

* You have to be logged in to your Az cli account to use this command, so no need for SAS or auth that has to be provided
* Az storage copy –source $source –destination $destination
* az storage blob copy (runs asynchronously)
* az storage blob copy start-batch
* az storage blob copy start

--account-name stthomas000

--destination-container container2

--destination-blob "log.TXT"

--source-account-name stthomas000

--account-key zwGBGxUO9WTrKwhAI0yx+Sq8LlhzTsxOVawXkR60dP3RlqxfqW/6+J2lPGcUxByZDUVfipMaAYbdemYHJj3SFg==

--source-container container1 --source-blob log.TXT

Blob copy Powershell

* Start-AzureStorageBlobCopy

**AzCopy v10**

* **Automates data movement between storage accounts**
* Data movement is recoverable
* Azcopy –help
* azcopy login --tenant-id "eb34f74a-58e7-4a8b-9e59-433e4c412757"
* To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code AXFLDU8VH to authenticate
* Azcopy copy $source $destination
* azcopy copy "hello2.TXT" https://stthomas000.blob.core.windows.net/container1/?<sas token>
* azcopy copy "https://sourceaccount.blob.core.windows.net/sourcecontainer/\*?<source sas token>" https://destaccount.blob.core.windows.net/destcontainer/\*?<dest sas token>

Storage types in Azure

1. Blob (Containers)
2. Queue
3. Table
4. File
5. Disk

Storage account urls

* Blob - https://<storage-account>.blob.core.windows.net
* Files - https://<storage-account>.file.core.windows.net
* Queue - https://<storage-account>.queue.core.windows.net
* Table - https://<storage-account>.table.core.windows.net

Containers inside storage accounts and their public access levels

* Private – no anon access
* Blob – anon read access to individual blob files within the container, but not the container itself
* Container – anon read access at the container level and at the individual blob level
* https://storageexample021.blob.core.windows.net/container?comp=list (shows a list of files within the container/blob/private storage if access is set to container

Blob Storage (Blobs live inside containers)

* A blob is just a file (**B**inary **L**arge **Ob**ject)
* Unstructured data
* Highly scalable
* Inside a storage account (storage level), you create a container (which acts as a folder) and store files (blobs) inside that container inside the storage account
* MS recommends blob storage for most uses
* 3 storage tiers – Hot, Cool, Archive
* Set to no public access by default
* You can host a static website via blob storage ($web folder)
* PaaS
* Can save customized VM images for future provisioning inside blob storage
* Comparable to AWS S3
* Accessible via URL –
  + storageaccountname.blob.core.windows.net/containername/filename or blobname
* When you upload a file to a container you specify the type of blob:
  1. Block
     + Default blob type
     + Test and binary data
     + Optimized for uploading large amounts of data efficiently
  2. Page
     + Random access files up to 8TB in size
     + Best suited for VM disks
     + Optimized for random read and write operations
  3. Append bl
     + optimized for append operations
     + ideal for logging data from VMs, telemetry, syslog, media streams, etc;
     + cannot update or delete existing blocks

Blob Concurrency (multiple clients wanting to talk to the same blob)

* Optimistic concurrency
  + We use the identity of an object (Entity tag)
  + Conditional headers (if-match)
* Pessimistic concurrency
  + Locking is the acquiring of a lease

Azure Storage Blob change feed

* provides transaction logs of all the changes that occur to the blobs and the blob metadata in your storage account
* We can check the Enable blob change feed option in the Data protection blade of Blob storage
* provides ordered, guaranteed, durable, immutable, async, read only logs
* create, modify, and delete changes to blobs in your storage account
* files are located in the $blobchangefeed container of the storage account

Blob Snapshots

* Are like photographs, they can take images of files(blobs) at a particular time
* Useful to maintain previous versions of a blob
* Multiple snapshots can be created for a single blob
* Cannot delete a blob without deleting the blob snapshots (if they exist)

Blob Storage access tiers – the goal is to save money

* Hot (default)
  + frequently accessed data
  + discount on transaction because you are frequently accessing the data, but no discount on footprint (the actual data space)
  + instant access
  + 99.9
* Cool
  + infrequently accessed data (30 days)
  + discount on footprint but not on transactions
  + instant access
  + 99 GRS, 99.99 RA-GRS
* Archive
  + rarely accessed data (180 days)
  + data has to be rehydrated to be accessed
  + large discount on footprint, but have to pay for rehydration
  + there is a delay to access data (1-15 hours)
  + Data is stored offline
  + High latency for retrieval
* When you move from warmer to cooler tier (hot to cool or hot to archive), you are charged for the write operations on the destination tier
* When you move from cooler to warmer, you are charged for read operations from the source tier
* If you don’t manually configure the access tier for a blob(file), it inherits the access from its container or Storage Account. This practice is not recommended.

Rehydrate an archived blob to an online tier

* Standard priority
  + processed in the order it was received and may take up to 15 hours.
* High priority:
  + request will be prioritized over Standard requests and may finish in under 1 hour for objects <10GB in size
* Once a rehydration request is initiated, it cannot be canceled

Blob storage events

* Azure Storage events allow applications to react to events
* Blob storage events are pushed using Azure Event Grid to subscribers such as Azure Functions
* Asynchronous file uploads into blob storage are a great fit for events

Azure Storage Batch API

* batchClient.JobOperations.CreateJob()
* TaskSuccess
* TaskFailure
* OutputFiles
* CloudJob.CommitAsync(IEnumerable<BatchClientBehavior>, CancellationToken)

Storage Properties and Metadata

* System properties
  + information that the Storage services automatically adds to each storage resource
  + some can be modified, some are read only
  + name, URL, last modified, Etag, lease status, lease duration, etc;
* User-defined meta-data
  + Assign key-value pairs to an azure storage resource

3 steps to update metadata of blobs (updated v12)

1. GetPropertiesAsync– get the metadata (GP)
2. Metadata.Add – update the metadata
3. SetMetadataAsync – save the metadata (SM)

Properties and Metadata CLI

* az storage account blob-service-properties show --account-name stthomas000
* az storage account blob-service-properties update --account-name stthomas000 –add
* az storage blob metadata show –name stthomas000 --container-name container1
* az storage blob metadata update

Comparison of authentication services to access Azure Storage Accounts

* Managed Identity (fine grained control on folder level) and keys are managed internally
* Service Principal (fine grained control on folder level)
* Shared Signature Access (controls container level)
* Account Key / Access Key / Master Key (no control, all access to the storage account)

Blob leases

* A short block that the blob service sets on a blob or container item for granting access to that item
* Specifies you need to use the blob and no other users or processes can access the blob at the same time
* Leases can be timed or infinite (no time specified is infinite)
* AcquireLeaseAsync(null) is infinite as the lease will expire never
* Leases modes
  + Acquire
  + Renew
  + change
  + release
  + break
* A lease can be in 1 of 5 states
  + Available
  + Leased
  + Expired
  + Breaking
  + Broken

Blob Soft delete (In the Data protection blade)

* enables you to recover blobs, version and snapshots that were previously marked for deletion, including blobs that were overwritten
* Retention period is # of days (7 is the default)
* Applied at the storage account level

Azure Table storage (superseded by Cosmos DB Table API)

* 3 system properties of every entity

1. PartitionKey property
   1. a unique identifier for the partition within a given table, specified by the PartitionKey property
2. RowKey property
   1. The row key is a unique identifier for an entity within a given partition
   2. The “primary key” within a partition
3. Timestamp property

Table API C#

* Create the batch operation.
  + TableBatchOperation op = new TableBatchOperation();
* Execute the batch operation.
  + table.ExecuteBatch(op);
* Table Query
  + TableQuery.Where()
  + TableQuery.CombineFilters()

Storage replication in Azure

* Data in an Azure Storage account is always replicated 3 times in the primary region
  + synchronous- in region
  + asynchronous- different region
* Locally redundant storage (LRS)
  + copies your data synchronously 3 times within a single physical location in the primary region.
  + Least expensive
* Zone-redundant storage (ZRS)
  + copies your data synchronously across 3 Azure availability zones in the primary region.
* **Geo-redundant storage (GRS)**
  + 3 copies in the primary region within a single physical location using LRS, then copies your data asynchronously to a single physical location in the secondary region. Within the secondary region, your data is copied synchronously three times using LRS.
* Geo-zone-redundant storage (GZRS)
  + copies your data synchronously across 3 availability zones using ZRS, then copies your data asynchronously to a single physical location in the secondary region. Within the secondary region, your data is copied synchronously three times using LRS.
* Read access GRS (RA-GRS)
* Read Access GZRS (RA-GZRS)
* 3 copies of any data you run in Azure

Encryption

* Encryption in storage accounts is applied by default at the storage layer
* 2 types of encryption

1. Microsoft managed (default)
2. Customer managed

Azure Lifecycle policies

* Accessed in the Lifecycle management blade of storage account
* Azure Blob Storage lifecycle management offers a rich, rule-based policy for GPv2 and blob storage accounts.
* Use the policy to transition your data to the appropriate access tiers or expire at the end of the data's lifecycle
* For example, lets you transition blobs from cool to hot immediately if accessed to optimize for performance
* Rules are defined in the form IF – THEN

Azure Lifecycle policy CLI

* az storage account management-policy create
* az storage account management-policy delete
* az storage account management-policy show
* az storage account management-policy update

Azure Lifecycle Policy PowerShell

* Add-AzStorageAccountManagementPolicyAction
* New-AzStorageAccountManagementPolicyFilter
* New-AzStorageAccountManagementPolicyRule
* Set-AzStorageAccountManagementPolicy
* Remove-AzStorageAccountManagementPolicy

**Implement Azure security (20-25%)**

**Implement user authentication and authorization**

authenticate and authorize users by using the Microsoft Identity platform

authenticate and authorize users and apps by using Azure Active Directory

create and implement shared access signatures

**Implement secure cloud solutions**

secure app configuration data by using App Configuration Azure Key Vault

develop code that uses keys, secrets, and certificates stored in in AzureKey Vault

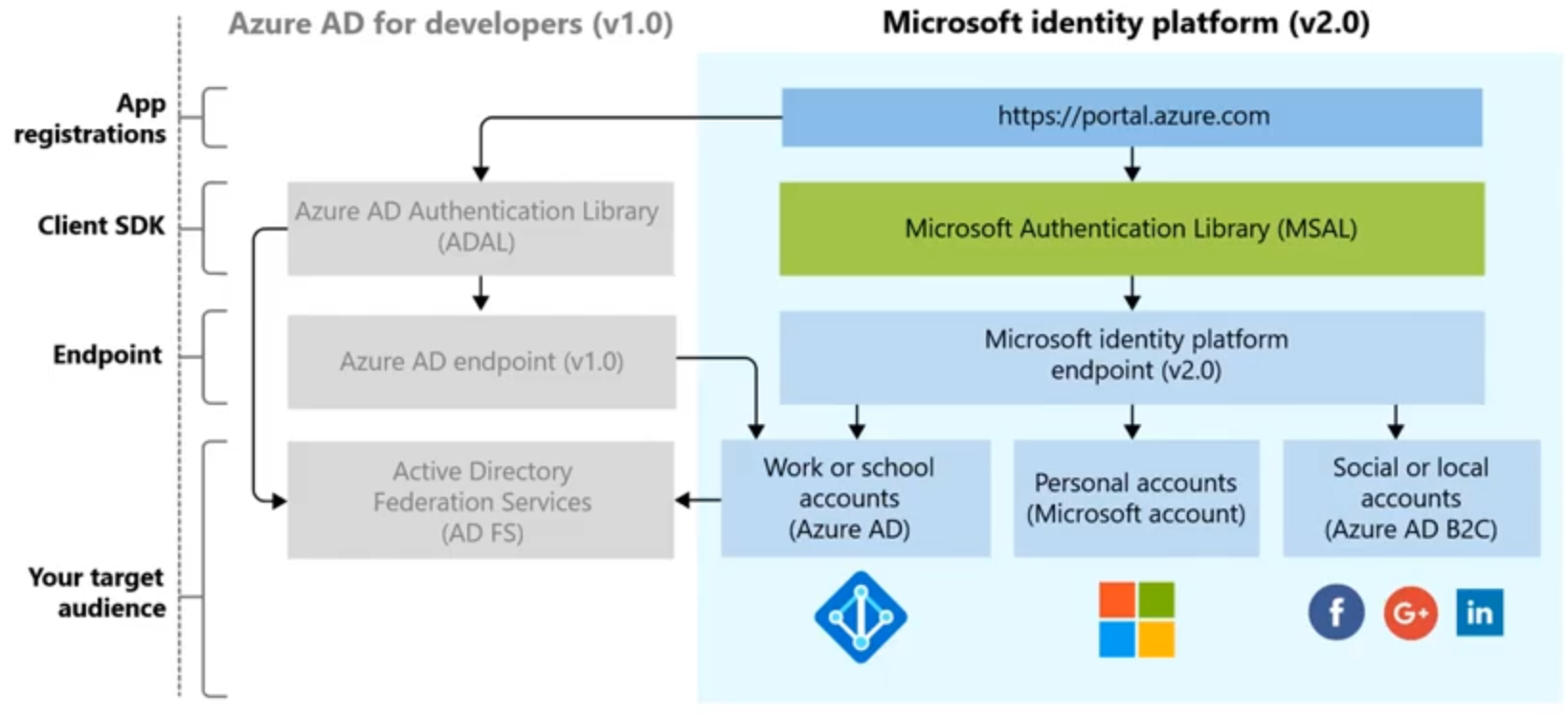
implement solutions that interact with Microsoft Graph

Authentication basics

* Apps request tokens from MS Identity
* Authentication happens over shared web surfaces to provide SSO
* Apps pass access tokens to APIs
* APIs validate access tokens before returning results
* **RECOMMENDATION – USE AN IDENTITY PROVIDER**

MS Identity platform v 2.0

* an evolution from the current Azure Active Directory version 1.0
* Consists of:
  + OAuth 2.0 and OpenID Connect standard-compliant authentication service that can auth identity types
    - Azure AD accounts
    - Personal MS accounts
    - Social or local accounts using Azure AD B2C
  + Open-source libraries (MSAL)
  + Application management portal
  + Application configuration API and (MS Graph API)
  + Developer content



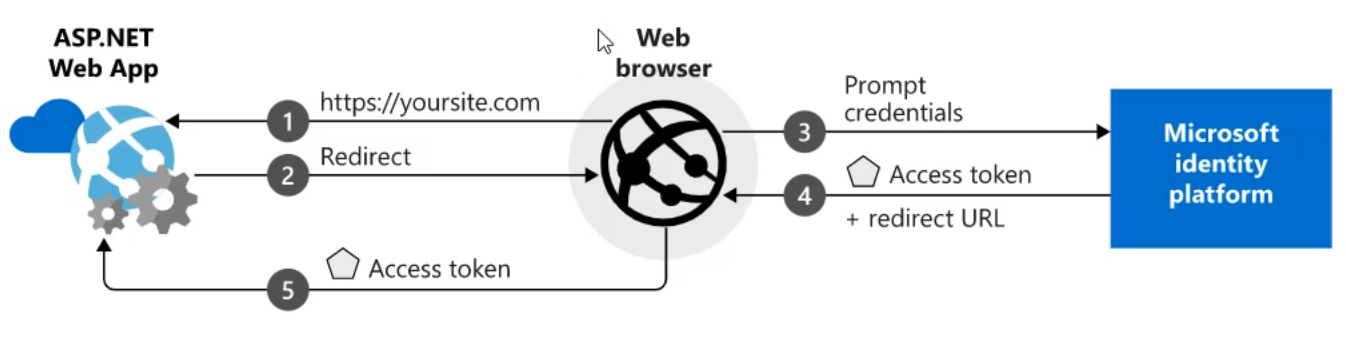
MS Identity platform ID tokens

* ID tokens are used by the client to **authenticate** the user
* They can be sent alongside or instead of an access token
* ID Tokens should be used to validate that a user is who they claim to be and get additional useful information about them
* Key-value pairs that are called claims (“key”: “value”}
* They can be sent alongside or instead of an access token
* should not be used for authorization purposes
  + use access token for authZ
* Id token is valid until the time expiration field (exp) passes out of its timeframe
* By default, an ID token is valid for 1 hour - after 1 hour, the client must acquire a new ID token
* id\_tokens are JWTs (JSON Web Tokens), meaning they consist of a header, payload, and signature portion

1. Header
   * Provides information about how to validate the token including information about the type of token and how it was signed.
2. Payload
   * Contains all the important data about the user or app that is attempting to call your service.
3. Signature (aka ID token signature)
   * Is the raw material used to validate the token

MS identity platform access tokens

* used by web APIs to perform authentication and authorization
* JWT
* issued by an authorization server as part of an OAuth 2.0 flow
* Access tokens are used to auth the app to call specific APIs



OAuth2

* industry protocol for authorization
* OAuth separates the role of the client from the resource owner
* Don’t have to store username/password
* Once the user has been authorized, the server sends a token confirming the identity of the client
* This access token must be refreshed once expired
* can use a refresh token for requesting a new token access without using credentials

OAuth authentication

1. Client
   1. uses the refresh token to get a new access token when it expires without needing to request a new authorization code
2. resource server
   1. The access token grants access to the resource hosted on the resource server
3. resource owner
   1. needs to authenticate the client before sending the authorization grant
4. authentication server
   1. The authorization grant or authorization code grants the client the needed rights to request an access token to the authorization server

Microsoft identity platform protocols

* OAuth versus OpenID Connect
  + OAuth for authz and OpenID Connect for auth
* OAuth versus SAML
  + OAuth for authZ and SAML for auth
* OpenID Connect versus SAML

MSAL – MS Auth Library

* Provides token based auth for web apps without having to work with OAuth
* Manages tokens

Azure Active directory

* Legacy v 1.0 solutions
* An Identity Provider
* Users and groups
* App Registrations
* Cloud based identity and access management solution
* It provides authentication, single sign on(SSO), application management, B2B identity services, B2C identity services and decide management
* The backbone of Azure and Office 365 and other SaaS apps
* Multi-tenant – identities from multiple tenants
* Tiers – free, basic (Office 365 Apps), Premium P1, Premium P2
* Free version can create 5,00,000 users or groups (they call them objects)
* **P1 and P2 has MFA and Conditional Access policies**
* P2 has identity protection
* Premium has SLA of 99.9%, free has no SLA
* Users are licensed on a per month/per user basis
* AAA – AuthN, AuthZ, Accounting (record keeping – ie who did what, who tried to sign on, access attempts both successful and failed, etc)
* Managed though the Azure Portal or Office 365 Portal
* Global Administrator is the highest privilege role
* All tokens used in Azure AD B2C are JSON web tokens (JWTs)

**Azure AD is an Identity Provider Solution**

* **Can authenticate a request**
* **Can also authorize a user to see if they have access**

Azure A/D authenticate and authorize users and apps

* In the A/D blade that student account do not have access to

Azure A/D App Manifest

* The definition of an app object within the MS identity platform which includes all configuration for allowed authentication and authorization integrations
* serves as a mechanism for updating the application object

Azure AD Conditional Access

* A way for admins to shape the authentication context to protect Azure AD Backend apps
* brings signals together, to make decisions, and enforce organizational policies
* Admin sets up specific policies to force users to do certain things to validate their identity under certain conditions
* Available only in P1, P2 tiers (Not in free)
* Example1 - trigger a MFA when the user logs in from a different location
* Example2 – allowing users to access company applications only from approved tablet devices

RBAC

* Create Users, Apps, groups
* Used to enforce authorization in applications
* When using RBAC, an administrator grants permissions to roles, and not to individual users or groups
* We assign them to objects in Azure with a specific role
* There are 2 sets of RBAC role sets in Azure, one for the Resources, and one in Azure Active Directory
* RBAC roles are used to manage access to Azure resources like VMs and storage accounts, Azure AD Administrator roles are used to manage administrative or sub administrative tasks that you would do within AD
* **RBAC can't be used for authentication or Identity Management**

Azure RBAC Built in roles

* Owner, contributor, reader
* An action defined what type of operations you can perform on a given resource type
  + Write enabled you to PUT, POST, PATCH, and DELETE
  + Read enables you to perform GET
* Can use powershell to get available roles
  + Get-AzRoleDefinition
* CLI
  + Az role definition list
  + az role assignment create

Azure Role definition vs role assignment

* az role assignment create command creates a new role assignment for a user, group, or service principal
* az role definition creates a custom role

ASP.NET RBAC

* Policy or Role
* [Authorize(Roles = "Administrator")]
  + Roles property indicates that users in any of the listed roles would be granted access.
* [Authorize(Policy = "Administrator")]
  + Policies are used for more granular and expressive control over authZ

App Registrations (A blade in Azure AD)

* You register an app with Azure AD in the Azure portal so the Microsoft identity platform can provide authentication and authorization services for your application and its users
* **The Microsoft identity platform performs identity and access management (IAM) only for registered applications**
* Application Registration is an account in Azure AD you can assign permissions to
* Used to represent an app so it can be accessed by another resource
* Global representation of an app for use in multiple tenants
* Supported account types – single tenant, multitenant, personal MS account only

To register an application with an Azure AD tenant using the portal

(3 steps exam question) – student account does not have access to AD so I copy things

1. Sign in to the Azure portal using either a work or school account or a personal Microsoft account.
2. If your account gives you access to more than one tenant, select your account in the upper right corner. Set your portal session to the Azure AD tenant that you want.
3. Search for and select Azure Active Directory (Step 1). Under Manage, select App registrations.
4. In App Registrations, Select New registration. (Step 2)
5. In Register an application, enter a meaningful application name to display to users.
6. Specify who can use the application.
7. Under Redirect URI (optional), select the type of app you're building: Web or Public client (mobile & desktop). Then enter the redirect URI, or reply URL, for your application. (Step 3)
8. When finished, select Register.

Service Principal

* Behind the scenes of every app registration is a service principal
* Is a local representation of an application registration
* Application account in the local Azure AD tenant
* A security identity object that can be used by a user created app, service or tool to have access to Azure resources
* helps us avoid having to create fake users in AD in order to manage authentication when we need to access Azure resources
* **has an expiration, which Managed Identity takes care of**
* when the resource doesn’t support Managed Identity, then we need to create Service Principal and manage it.

-> For each tenant, an app registration is created and there will be a local service principal to represent the app

Creating a service principal steps

1. Create AD Application Registration (by a user with the correct permissions in Azure AD)
2. Create Key
3. Assign App to Role
4. Login as application
5. Execute tasks

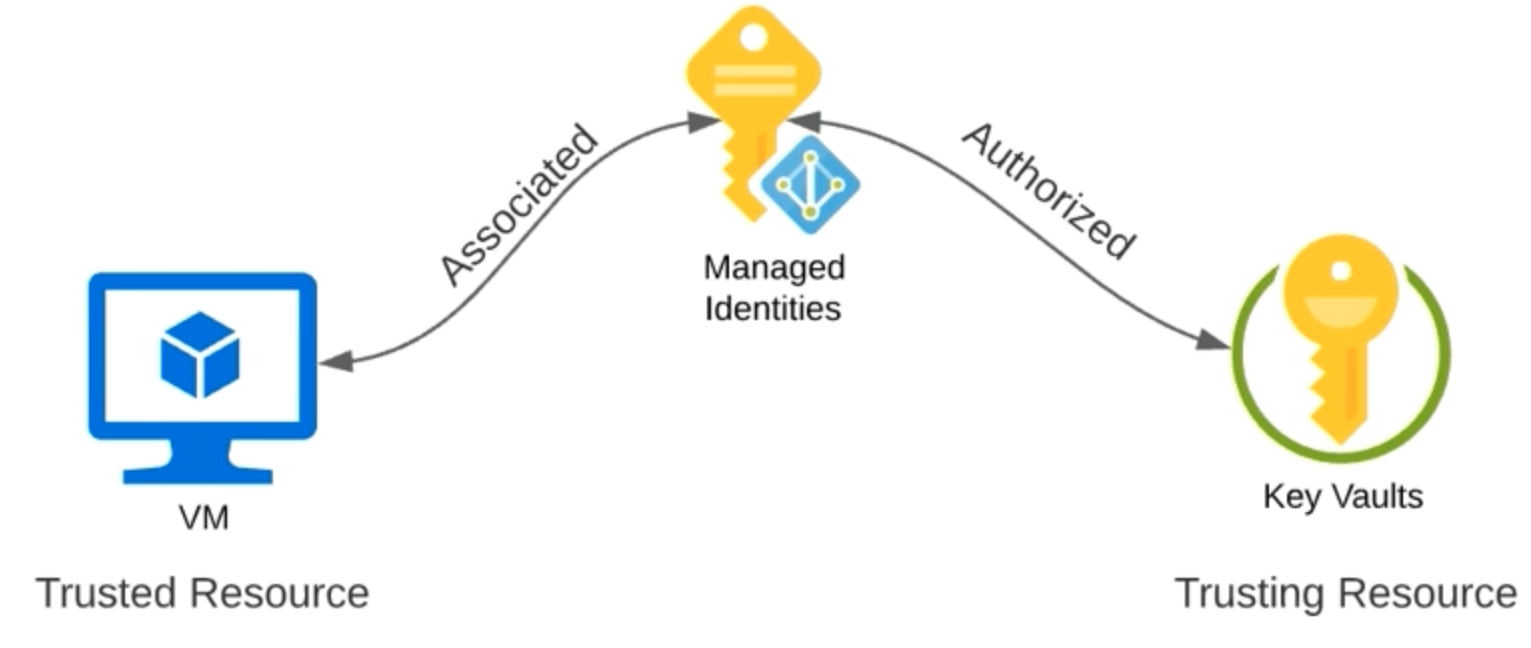
Vs Managed Identities (in the Identity blade of an Azure resource that supports managed identities)

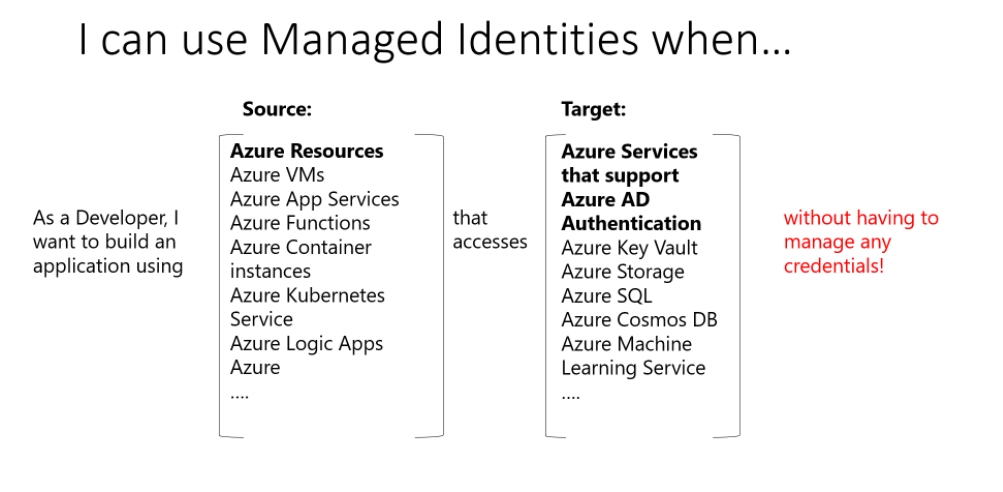
* A feature of Azure AD
* The preferred approach to managing identities for apps and automation access
* a layer on top of a service principal
  + Service Principal has an expiration and Managed Identity is a solution to that
* identities are used to obtain an access token from Azure AD for accessing a specific resource
* removes the need for you to manually create and manage service principals directly
* manages the creation and automatic renewal of a service principal on your behalf
* An Azure AD identity to an Azure Managed resource
* Managed Identity lets some Azure resource have an identity in Azure AD that can be given permissions to access other Azure resources
* This way we don’t need to store credentials or certificates
* Allow resources to authenticate to each other
  + Associate a managed identity with a resource (the trusted resource)
  + Assign permissions to the managed identity on another resource (trusting resource)
* Authorization is still separate
* No credentials to manage
* Restriction – when using a managed identity you can't restrict access by IP like you can with SAS token
* You also cannot use CDN
* 2 types - You either use one or the other (the other one is turned off)

1. System Assigned
   * Widely supported across Azure
   * Enabled directly on the Azure resource
   * Automatically attached to a single Azure resource
   * Deleted automatically when Azure resource is deleted (shared life cycle)
   * One system assigned identity per Azure resources, so cannot be shared
   * On/Off (Off is the default)
2. User Assigned
   * Supported by a growing list of services on Azure (some in preview)
   * Created as a standalone Azure resource
   * Must be deleted manually (no defined lifecycle)
   * Can be assigned to one or more Azure resources
   * 3 concepts
     + Client ID
       - unique identifier generated by Azure AD
     + Principal ID
       - ID of the service principal associated with the managed identity
     + Azure Instance Metadata Service (IMDS)
       - For requesting OAuth access token form your app deployed within the VM

->You usually use system-assigned managed identities when your workload is contained within the same Azure resource, or you need to independently identify each of the service instances, like VMs

->if you need to grant access to a workload that is distributed across different resources or you need to preauthorize a resource as part of a provisioning flow, you should use user-assigned managed identities.





How to configure Managed Identities Example

1. Create Identity
   1. Do this by creating a system assigned or user assigned identity (authentication)
   2. In the Identity blade of the trusted resource (webapp)
   3. Turn managed identity on (system or user assigned)
   4. Copy the Object ID property (we need this later)
2. EITHER Give permission to keyvault
   1. Assign permission to the client identity in the target Azure service (authorization)
   2. In the Access Policies blade of the trusting resource (keyvault)
   3. Click on Add access policy
   4. Under select principal, paste the object ID from the managed identity configured in step 1

2a. OR Give permission to storage account

* 1. Or in the IAM blade of a Storage Account
  2. Click on + Add role assignment on top of the page to add a new role assignment for your VM.
  3. Under the members tab, select managed identity, then select members

How Managed Identity works

* Managed Identity policies uses the managed identity to obtain an access token from Azure AD for accessing the specified resource.
* After successfully obtaining the token, the policy will set the value of the token in the Authorization header using the Bearer scheme

Managed Identities in the Azure portal

* Only shows user managed identities

Granting VM access to specific resource groups in Azure Resource Manager

* Enable a system assigned managed identities on the vm
* Grant your VM access to a resource group in the IAM blade of resource group
* Log into the VM that wants access, and open cmd
* run the Invoke-RestMethod cmdlet to make a request to the local managed identity for Azure resources endpoint
* extract token from response
* call Azure Resource Manager using the access token (Invoke-WebRequest cmdlet)
* Using managed identities for Azure resources, your code can get access tokens to authenticate to resources that support Azure AD authentication

**Azure AD App Manifest Attributes**

* appRoles
  + Specifies the collection of roles that an app may declare
* groupMembershipClaims
  + Configures the groups claim issued in a user or OAuth 2.0 access token that the app expects
  + Common uses are when you want Azure AD users must be able to login to a webapp
  + None, SecurityGroup, ApplicationGroup, DirectoryRole, All
  + SecurityGroup - groups claim will contain the identifiers of all security groups of which the user is a member
  + All - groups claim will contain the identifiers of all security groups and all distribution lists of which the user is a member
* optionalClaims
  + optional claims returned in the token by the security token service for this specific app
  + sid is Session ID, used for per-session user sign-out
* oauth2AllowImplicitFlow
  + Specifies whether this web app can request OAuth2.0 implicit flow access tokens
  + default is false
* oauth2AllowIdTokenImplicitFlow
  + Specifies whether this web app can request OAuth2.0 implicit flow ID tokens. The default is false.
* oauth2Permissions
  + Specifies the collection of OAuth 2.0 permission scopes that the web API (resource) app exposes to client apps
  + Returns a list
* signInAudience
  + Specifies what Microsoft accounts are supported for the current application
  + AzureADMyOrg, AzureADMultipleOrgs, AzureADandPersonalMicrosoftAccount, PersonalMicrosoftAccount
* Access to a key vault is controlled through two interfaces: the control/management plane and the data plane

Key Vault Access Policies

* Permission Models options
  + Vaults access policy OR
  + Azure RBAC

Control/Management Plane vs Data Plane

* Both planes use Azure AD for authentication

1. Control Plane
   * used for managing the Key Vault itself which includes operations like creating and deleting key vaults, retrieving key vault properties, and updating access policies
   * uses Azure RBAC for authZ
   * creating and deleting of resources
2. Data Plane
   * working with the data stored in a key vault and includes operations of add, delete, and modify keys, secrets, and certificates
   * AuthZ access to a data plane is provided through Key Vault access policies and Azure RBAC for KeyVault data plane operations
   * Please use Access Control (IAM) to configure access policy
   * Accessing resources, reading/writing resources

Management Plane - Azure RBAC (in the IAM blade)

* Security principal
  + User, groups of users, service principal, managed identity
* Role definitions
  + A collection of permissions
  + Owner, contributor, reader, user access admin
* Scope
  + Set of resources you want to apply certain actions to
  + Management group – subscription – resource group - resource
* Role assignments
  + Roles are Additive
  + Deny assignments take precedence over everything else (blocked access)

Accessing storage accounts in Azure

1. Access Keys
   * Root or administrator
   * 2 keys per account, so you can rotate without downtime as periodic regeneration of keys is recommended
   * This gives full root access, so not preferred from a security standpoint
   * Set at the storage account level
2. Shared access signature (SAS)
   * unique token generated for a client for a specific purpose
   * the preferred method of giving access as it allows for minimum privilege
     + what resources they can access
     + permissions to the resources
     + how long the SAS is valid
   * generated on the client, not on the server
   * grants access without sharing storage account keys
   * for best practices – always use https
   * SAS tokens are immutable
   * keys are in the form sp=blahblahblah
   * Unlike Managed Identities, cannot restrict by IP addresses or Azure Virtual Network subnets
   * SAS token can control on Container level only, Not on folder level like Service Principal
3. Stored Access Policies (SAP)
   * Only for Service level shared access signatures (and not Account SAS)
   * Set at the container/table/queue/file level rather than at the account level (SAS)
   * Lives within the control plane
   * Generated on the Server side
   * Can be modified
   * Allows you to define and control the validity and expiration of the policy without needing to modify the Service SAS token
   * Can associate up to 5 different SAPs
   * SharedAccessBlobPolicy Class can be added to the SAP on the container level

SAS Breakdown

* Storage Service Version
  + Sv=2017-07-29
* Signed Services
  + Ss=bfqt
  + (Blob, File, Queue, Table)
* SAS Signed Resource types
  + Srt=sco
* Storage Account level access
  + Container level access (read only)
  + Object (Container level access but can read and modify
* Signed Permissions
  + Sp=rwdlacup
  + Read Write, Delete, List, Add, Create, Update, Process

Azure Storage SAS

* A secure delegated access to resource in your storage account without compromising the security of your data
* Can specify services, resource types, permissions, date expiration, ip addresses and protocols
* Can create SAS token for service, container and item levels
* You append the SAS token to the URL for accessing storage resource
* 3 types SAS types
  + User delegation
    - only works for Blob Storage (at the blob/container level) and Azure Data Lake Storage Gen2
    - Key is generated from Azure A/D identity account
    - the preferred SAS type
    - does not use the access key
    - Cannot use either SAPs when working with user delegation SAS
  + Service
    - scoped to a specific single service in your storage account (only one of blob or queue or table or files)
    - Can apply to a container, and subsequently all blobs within that container; Or to a specific blob file
  + Account
    - Applied at the account level
    - any blobs, queues, tables AND/OR files within the storage account

SAS Best practices

* Always use https when creating or distributing an SAS
* Use user delegation whenever possible (limited to blob storage)
* Define an SAP for a service specific SAS
* Use near-term expiration on ad hoc service or account SAS
* Follow least privileged access for resources

SAS can have 2 forms

* Ad hoc SAS
  + Everything you need is in the token itself
  + Cannot be revoked; Have to regenerate key to invalidate the signature
  + time, permissions, IPs, https are specified in the SAS URI
* Service SAS (with stored access policy)
  + Defined on a resource container(blob, queue, file or table)
  + The SAS inherits the constraints in the SAP

Stored Access Policy

* Can only be done for service SAS
* Reused by Multiple SAS
* Defined on a resource container
* Permissions and validity period
* Service level SAS only
* Can specify time and permissions
* More flexibility than Ad Hoc SAS

App Configuration

* store all the necessary configuration for your cloud applications in a single repo
* can store key-value pairs and key vault references
* can also add a label attribute to a key (null by default)
* Endpoint URL https://appconfiguration-thomas-000.azconfig.io
* Integration with Managed Identities
* create references from Azure App Configuration items to Azure Key Vault items
* Values in the Azure App Configuration are encrypted
* limit of 10,000 characters, including the key, label, and value
* 2 components
  + App configuration
    - Free
      * 10Mb max size of store
      * 7 days
    - Standard
      * 1Gb max size
      * 30 days
    - Can switch from free to standard but NOT from Standard to free (cannot downgrade)
  + SDK

When to use App Configuration

* Micro server apps – AKS, Azure Service Fabric, any containerized apps
* Serverless apps – Azure functions, etc;
* CD pipelines

Azure Key Vault

* PaaS
* Managed service for securing sensitive information
* Central repo to store keys, secrets and certificates
* Types of secrets: tokens, passwords, certificates, etc
* Highly integrated with Azure services such as VMs, Logic apps, Web Apps, etc
* Certification management (public and private SSL/TLS certificates)
* HSM – hardware security models
* All disks created by Azure are encrypted by default
* Centralization
* Soft-delete
* Purge protection (retention period)
* should use managed identities authentication for accessing Azure Key Vault
* need to define a certificate policy before creating a certificate in Key Vault
* If you import a certificate into the Azure Key Vault, a default certificate policy is automatically created for you

When to use Azure Key vault

* Azure Key Vault provides better security than the Azure App Configuration service

Azure Keyvault Powershell

* New-AzKeyVault –Name keyvault1121 –ResourceGroupName rg-key-1121 –Location CanadaCentral

Azure Key Vault CLI

* az keyvault create –name keyvault1121 –resource-group rg-keyvalut-1121 –l CanadaCentral
* az keyvault secret set --name mykey1 --vault-name keyvaultmy222 --value dogs123
* az keyvault secret show --name mykey1 --vault-name keyvaultmy222 --query value
* az keyvault key create

Keyvault URL

* https://kvthomas000.vault.azure.net/mysecret1
* Secrets
  + {vaultBaseUrl}/secrets/{secret-name}/{secret-version}?{api-version query string parameter}
  + https://kvthomas000.vault.azure.net/secrets/mysecretname/4387e9f3d6e14c459867679a90fd0f79?api-version=7.2

Key Vault .NET

* string keyVaultName = Environment.GetEnvironmentVariable("KEY\_VAULT\_NAME");
* var kvUri = "https://" + keyVaultName + ".vault.azure.net";
* var client = new SecretClient(new Uri(kvUri), new DefaultAzureCredential());

Securing your key vault

* RBAC
* Access Policies – greater level of control than RBAC
* Networking
* Multiple Key Vaults

Pricing tiers

* Standard (software-protected)
* Premium (standard + optional HardwareSecurityModels-protected)

Configuring Auth for Azure Key Vault (3 options)

1. Azure A/D App Registration
2. Managed Identities (preferred option)
3. Key vault references

Azure Key Vault References (for Function and App Services)

* Uses managed identities under the hood
* Requires no code changes
* Steps
  1. Move the configuration to keyvault
  2. Deploy your app service or azure function
  3. Create a system assigned identity for your app
  4. Give GET KeyVault Secrets access to the app identity
  5. Update the configuration values with the KV reference syntax
  6. Verify your application functionality
* Syntax
  + SecretUri=secretUri @Microsoft.KeyVault(SecretUri=https://myvault.vault.azure.net/secrets/mysecret/
  + VaultName=vaultName;SecretName=secretName;SecretVersion=secretVersion

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

-secret version is optional

Soft-delete

* Enable retention period and accidental deletion
* Recovery of deleted vaults and key vault objects (keys, secrets and certificates)
* Automatically enabled (the ability to turn off in portal has been deprecated)
  + Can turn off in CLI, but this is expected to be deprecated late 2021
* Remains recoverable for 7 to 90 days (90 is the default)
* Use Purge to permanently delete a secret

Purge protection

* Enforce retention period and accidental deletion
* When enabled, a vault of object in the deleted state cannot be purged until the retention period has passed
* Disabled by default
* Enabling the purge protection for a vault is an irreversible action.
* az keyvault update --name keyvaultmy222 --resource-group rg-keyvault-222 --enable-purge-protection true

Always Encrypted using Key Vault

* Always Encrypted encrypts SQL db data at rest on the server, during movement between client and server, and while the data is in use.
* After you configure data encryption, only client applications or app servers that have access to the keys can access plaintext data
* In these types of situations, Enable Always Encrypted for the SecurityPin column using a certificate contained in Azure Key Vault and grant the WebAppIdentity service principal access to the certificate
* Using the Azure Portal, add Data Masking to the SecurityPin column and exclude the WebAppIdentity service principal

Microsoft Graph

* Microsoft Graph is a RESTful web API that enables you to access Microsoft Cloud service resources
* Microsoft Graph is the gateway to data and intelligence in Microsoft 365

Permissions

* User.Read
* User.ReadWrite

Constraints

* All
  + User.Read.All
* Shared
  + Mail.Read.Shared
* AppFolder
  + Permissions to read and write in one Drive folder

**Monitor, troubleshoot, and optimize Azure solutions (15-20%)**

**Integrate caching and content delivery within solutions**

configure cache and expiration policies

configure cache and expiration policies for Azure Redis Cache

implement secure and optimized application cache patterns including data sizing,

connections, encryption, and expiration

**Instrument solutions to support monitoring and logging**

configure an app or service to use Application Insights

analyze and troubleshoot solutions by using Azure Monitor

•implement Application Insights web tests and alerts

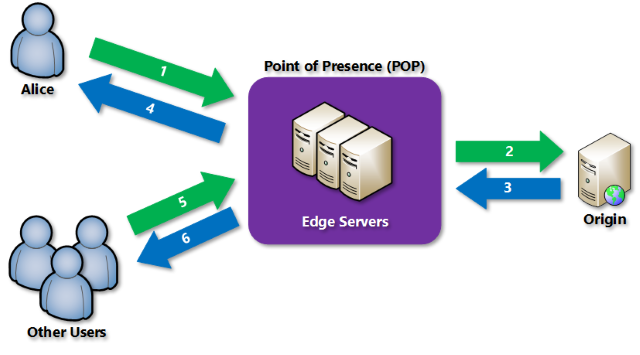
* Static vs dynamic content

Azure CDN (global)

* Globally distributed network
* The idea is to reduce asset load times, reduce hosting bandwidth, and increase availability and redundancy
* Protection from DDoS
* Acts as an Edge/Point of Presence and sits between the user and the source
* Azure CDN support 2 approaches for configuration caching
  + Global caching rule (overrides http cache headers)
  + Custom caching rules (overrides global caching rule)
* Custom caching rules can set custom behavior and duration by path preferred

-> CDN endpoint connects our CDN with our app

-> <unique endpoint name>.azureedge.net has to be globally unique



CDN Profile options

* Microsoft’s CDN or CDNs from Verizon or Akamai

CDN content types

* Static content (images, css files, js files)
* Dynamic content (changes on user interaction such as dashboards, query results)

Pricing Tiers (applied at the profile level)

* Standard Microsoft
* Standard Akamai
* Standard Verizon
* Premium Verizon

-> CDN profiles cannot be created with a student account

-> Can only configure caching rules only for Verizon and Akamai profiles

Http Headers

* 2 types
  + Cache-control (preferred)
  + Expires (legacy)
* Cache-control takes precedent over expires
* Standard Akamai
  + Max age
  + No cache
  + No store

CDN Advanced options

* Custom dns domain
* Compression
* Caching rules
* Geo-filtering
* Optimization

When to use CDN

* Caching static content that changes infrequently

Configuration options

* Custom DNS domain
* Compression
* Caching rules (Rules Engine)
* Geo-filtering
* Optimization
* Dynamic site acceleration

Azure CDN Cache Behavior (these are options when setting the expiration of the cache)

* Bypass cache
  + Do not cache and ignore origin-provided cache-directive headers
  + cache duration is automatically set to 0 seconds and cannot be changed
* Override
  + Ignore origin-provided cache duration; use the provided cache duration instead
  + cache durations range between 0 seconds and 366 days
* Set if missing
  + honor origin-provided cache-directive headers if they exist, else, use the provided cache duration
  + cache durations range between 0 seconds and 366 days

Azure CDN Query Strings

* Ignore query strings
  + Default mode
  + It’s really ignore all subsequent query strings
* Bypass caching for query strings
  + requests with query strings are not cached at the CDN POP node.
* Cache every unique URL
  + each request with a unique URL, including the query string, is treated as a unique asset

Azure CDN Query string handling

* Ignore query strings
* Bypassing caching
* Cache every unique url

Azure CDN Rules

* Each Azure CDN endpoint can have up to 25 rules
* Each rule can have up to ten match conditions and five actions
* Match conditions
  + Filter requests based on a specific IP address, country, or region.
  + Filter requests by header information.
  + Filter requests from mobile devices or desktop devices.
* Actions
  + URL redirect
    - Use this action to redirect clients to a new URL.

Integrate an Azure Storage account with Azure CDN

* You want to enable Azure Content Delivery Network (CDN) to cache content from Azure Storage
* Use a SAS on your Storage Account that grants restricted access rights to your Storage resources without exposing your account key
* CDN doesn’t support authN with Managed Identity

Dynamic site acceleration

* Dynamic site acceleration (DSA) is available for Azure CDN Standard from Akamai, Azure CDN Standard from Verizon, and Azure CDN Premium from Verizon profiles
* DSA includes various techniques that benefit the latency and performance of dynamic content
* Techniques include
  + route and network optimization
  + TCP optimization
* You can use this optimization to accelerate a web app that includes numerous responses that aren't cacheable.
* Examples are search results, checkout transactions, or real-time data. You can continue to use core Azure CDN caching capabilities for static data.

Azure Front Door (http, global)

* Global layer 7 load balancing
* Path based routing
* Allows you to cache content by providing a CDN
* SSL offloading
* Caching is performed at the routing level, so can route different paths in your URL to different back end pools hosting your app

Azure Front Door compression

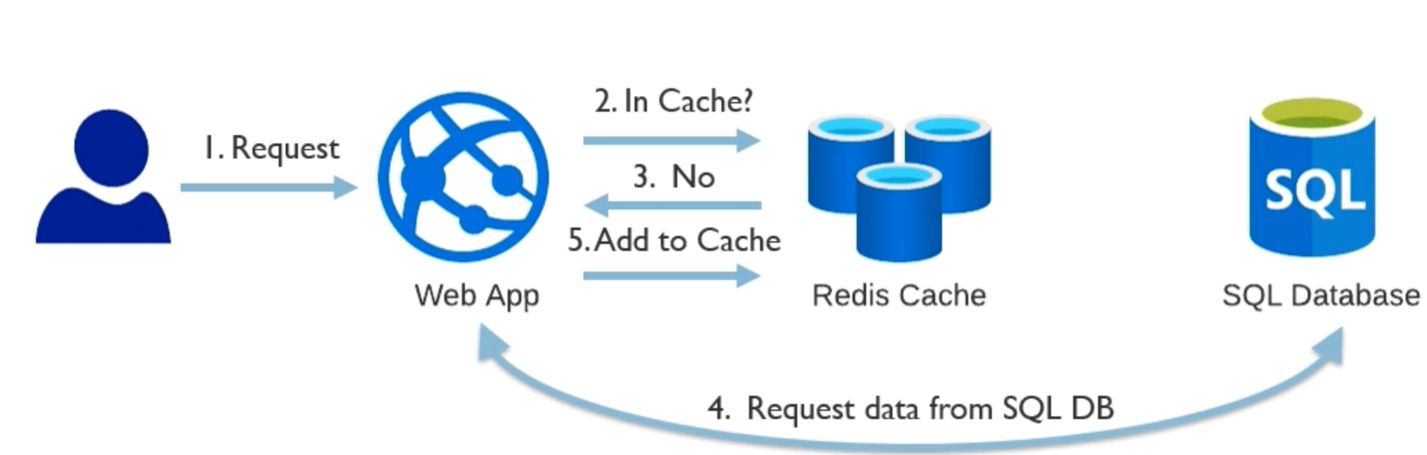
* Supports dynamic compression at the edge
* results in a smaller and faster response to your clients
* Compressed files must not be larger than 8Mb in size
* Supports compression types
  + GZip
  + Brotli
* This only works for supported MIME types from a fixed list maintained by MS
* You can purge cached content from all edge nodes to force new updated assets into cache

Azure Front Door Cache Purge Types

1. Single path (full path to the asset)
   1. Typically used for specific files
2. Wildcard purge (folder and subfolder)
   1. Typically projects
3. Root domain (all cached asset in root domain)

Azure Cache for Redis

* Distributed caching engine
* Provides an in-memory data store based on the Redis software
* Store data in web apps that persists the immediate session
* Managed by Microsoft
* Can support Apps in Azure or external Apps
* Full API support
* FQDN - <unique dns name> .redis.cache.windows.net
* If you don’t provide a TTL for the key-value pair, the entry in the cache won’t expire
* You can connect publicly (via Public IP addresses or can connect service endpoints), OR privately, using a private endpoint
* provides a session state provider that you can use to store your session state in- memory with Azure Cache for Redis instead of a SQL Server database



Azure Redis Cache Pricing Tiers

* Basic
  + no SLA
  + single node cache
  + 250MB – 53GB of memory sizes
* Standard
  + 2 replicated nodes (primary/replica)
  + 99.9% availability
  + 20,000 clients
* Premium
  + Redis cluster
  + Low latency
  + 99.95% availability
  + 100Gb memory
  + 40.000 clients
* Enterprise
  + Full Redis feature set
  + 99.99% availability
* Enterprise Flash
  + Fast non-volatile storage

Redis Cache scaling

* You can scale up a Redis Cache, but there are some restrictions on scaling down
* For example, You can't scale from a higher pricing tier to a lower pricing tier
* You can't scale from a Basic cache directly to a Premium cache – you have to scale to Standard first, then to Premium
* CLI scaling
  + az redis
* PowerShell scaling
  + Set-AzRedisCache -ResourceGroupName myGroup -Name myCache -Size 2.5GB
* If you want your Azure Cloud Services or Azure Web App applications to use two or more instances (processes dedicated to a Web App or web roles in a Cloud Service), you must configure where the application will store session state information.
* You need to Store session state information in Azure Redis Cache if you are using multiple instances

Azure Redis Cache supported Languages

* .NET
* Js
* Python

Reading/Writing values in Redis

* We typically use the StackExchange.Redis package available in NuGet
* When reading or writing values in the Azure Cache for Redis, you need to create a ConnectionMultiplexer object
* Using this object you can Use Redis as a database GetDatabase(), StringSet(), StringGet()
* Also use Redis as a messaging queue using GetSubscriber(), Publish() and Subscribe()

Common Azure Cache for Redis Use Cases

* User session storage for distributed apps
  + Maintain application state
    - Shopping cart
  + Session cookies or local storage
    - Limited data store
    - Slow performance
* Content caching
  + Cache static content
    - Images
    - Templates
    - Style sheets
  + Reduces server load
  + Redis Output Cache Provider for ASP.NET
* Distributed transactions
* Database caching
* Share session state across all ASP.NET web applications
* Save full HTTP responses for concurrent requests

When to use Azure Redis Cache

* Static content
* Most-accessed dynamic data
* In memory db’s
* Message queues using pub/sub pattern

Cache and Expiration policies

* Every cache entry has a TTL
* Once the TTL has expired, the latest file is fetched, and the TTL is reset
* Items stored in cache can be invalid by
  + Purging
  + TTL expiration

Controlling caching behaviors

* Using a caching policy
* Using the web.config file (if using a .NET web app) by setting the client cache property to control the TTL
  + cacheControlMaxAge is set like days.hours:minutes:seconds
* Programmatically (ASP.NET)
  + Response.Cache.SetExpires(DateTime.Now.AddHours(1));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetLastModified(DateTime.Now);

Default TTL (expiration duration)

* Azure CDN – 7 days
* Azure Front Door – between 1 and 3 days (selected randomly)
* Azure Cache for Redis – no default expiry, so items persist until deleted
  + Programitacally
    - \_cache.StringSet(“mykey”, “myvalue” new Timespan(3,0,0));
  + KeyDelete
    - Removes a single key from a Redis db
  + FlushAllDatabases
    - Remove all keys from all dbs

Default Caching behavior options

* Bypass cache
* Override
* Set if missing
* Not set

Redis Cache .NET

ConnectionMultiplexer connection =

ConnectionMultiplexer.Connect("contoso.redis.cache.windows.net,abortConnect=false,ssl=true,password=...");

IDatabase cache = connection.GetDatabase();

// Perform cache operations using the cache object...

// Simple put of integral data types into the cache

cache.StringSet("key1", "value");

cache.StringSet("key2", 25);

// Simple get of data types from the cache

string key1 = cache.StringGet("key1");

int key2 = (int)cache.StringGet("key2");

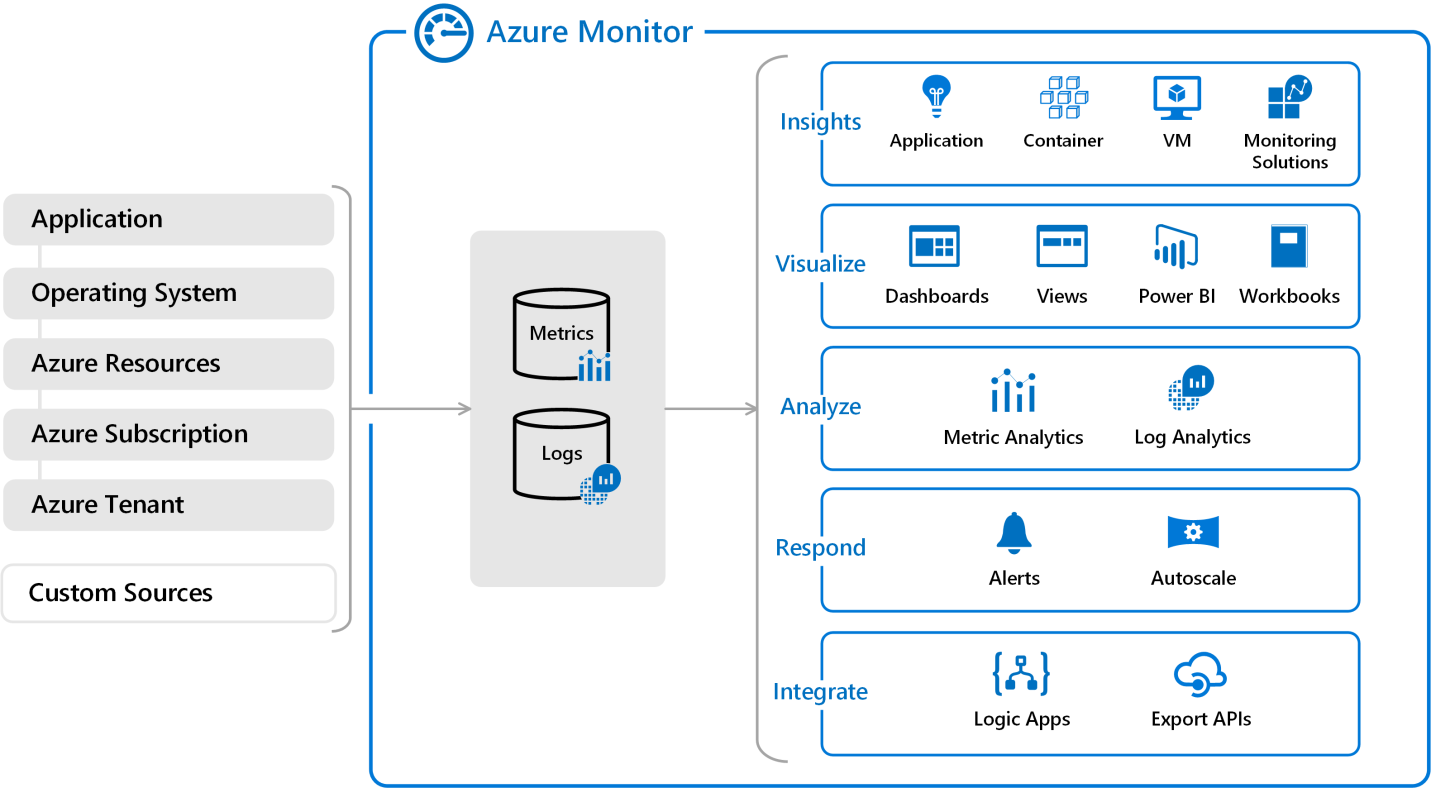
//Invalidate cache (delete keys)

Cache.KeyDelete(“key1)

* Azure Monitor is a tool composed of several elements that help you monitor and better understand the behavior of your solutions
* Application Insights is a tool for collecting information from your solutions
* Once you have the collected information, you can use the Analyze tools for reviewing the data and troubleshooting your application
* Depending on the information that you need to analyze, you can use Metric Analytics or Log Analytics.

Azure Monitor

* THE HEALTH OF YOUR APPS
* Collect, analyze, and act on telemetry from cloud and on prem environments to maximize your apps availability and performance
* monitor availability and performance of resources using metrics and logs
* service failure notifications
* help you create alerts and respond to those alerts
* monitors usage
* Insights -> Visualize -> Analyze -> respond -> Integrate
* In the middle - Metrics and Logs are types of data used by Azure Monitor
* On the left side - Application/OS/AzureResources/Azure Subscription/Azure Tenant/Custom Sources/ are the sources of monitoring data that populate the data stores above
* On the right side - is the different functions that Azure monitor performs with the collected data



Metric Analytics

* Point of time information
* reviewing the standard and custom metrics sent from your application
* a numerical value related to some aspect at a specific time of your app
* examples - CPU usage, free memory, and the number of requests
* Can create custom metrics
* Lightweight – so they can be used near real time
* can use metric telemetry to get different application metrics like: requestsPerSecond, requestsInQueue, and use these values to know when to scale

Vs Log Analytics

* analyzing the trace, logs, events, exceptions, and any other message sent from your application
* more complex than metrics because they contain much more information
* Kusto Query Language to analyze collected data
* Can refine query results by adding where clauses (piping)
* They can contain different kinds of data and may be structured or free form test with a timestamp
* Example – information about exception thrown during application execution
* Remember to enable diagnostic logging for Azure App Service before you query logs

Metrics Capabilities

* Correlate infrastructure issues
* Detect and diagnose issues across applications and dependencies
* Support operations with smart alerts and automated actions
* Create visualizations with Azure dashboards and workbooks

What type of data does Azure Monitor collect?

* Data about the performance and functionality of the apps source code
* Data about the OS on which our app is running
* Data about the operation of an Azure resources
* Data about the operation of tenant-level Azure services, such as Azure A/D

-> Azure Monitor can collection log data from any REST client that allows to create custom monitoring scenarios, including on prem solutions(Custom Sources in the Azure Monitor chart)

Diagnostic settings

* integration with Azure Monitor
* Platform logs are automatically generated, but to store them you must enabled diagnostics settings and specify one or more destinations (3 options)
  1. Log analytics workspace
     + Logical storage unit where log data is collected and stored
     + Allows logs to be analyzed by other monitoring
     + Powerful log queries
     + Use advanced Azure Monitor features such as alerts and virtualizations
     + Requirement
       - does not have to be in the same region as the resource being monitored
  2. Azure Storage
     + Allow you to select retention in number of days
     + Less expensive option
     + Logs can be kept indefinitely
     + Requirements
       - has to be in the same region as the resource being monitored IF the resource if regional
       - should not use storage that has existing non-monitoring data
  3. Event Hub
     + Allow you to stream data to external systems
     + Requirements
       - has to be in the same region as the resource being monitored IF the resource if regional
       - Requires manage, send and listen permissions

Azure Application Insights

* A blade in Azure Monitor
* Also each Azure resource will have its own Monitoring blades
* Use it to monitor your live applications, how traffic is moving within your App
* Both Azure and non-Azure
  + When using App Insights, there is no requirement to run your application in Azure. You only need to use Azure for deploying the Application Insights resource that you use for analyzing the information sent by your application
* Multiple ways to analyze and react to what is happening
* Distributed logging
* All data inside App Insights is stored in Log Analytics workspace
* Code based and codeless deployments
* Check performance of server machine like cpu or memory usage (to see if we need to scale up)
* Uses a log analytics workspace to log data
* Tracks request rates, dependency rates, exceptions, page views, load performance, http calls, host diagnosis diagnostic trace logs, etc;
* Languages - .NET, Java, JavaScript, Node.js, etc
* Different ways to review logs – metrics explorer, dashboards, visuals studio , power BI, Rest API
* Multiple Apps use the same App Insights
* Can be enabled for already deployed apps without having to modify code
* Can be done during deployment of resource or after, can be changed, can be enabled or disabled, and turned on with deployment of code
* Only available for Windows based Apps (no support for Linux)
* Azure will create the DefaultResourceGroup-CCAN resource-group with a Log analytics resource inside when you create a function resource (and likely other resources). That log analytics is the same scope as Azure Monitor > Logs blade

What does Application Insights monitor?

* Request rates
* Dependency rates
* Exceptions
* Load performance
* http(s) calls
* user and session counts
* performance counters
* host diagnostics
* diagnostic trace logs
* custom events

App Insights SDK

* ASP.NET, ASP.NET Core, Node.JS, React, Java EE, Python
* App Center Diagnostics collects data from mobile and desktop apps and export data to Application Insights

App Insights Telemetry

* Request
  + Generated to log a request received by your app.
* Exception
  + Typically represents an exception that causes an operation to fail.
* Dependency
  + Represents a call from your app to an external service or storage such as a REST API or SQL. In ASP.NET, dependency calls to SQL are defined by System.Data.

Application Insights provides three additional data types for custom telemetry

* Trace
  + used either directly, or through an adapter to implement diagnostics logging using an instrumentation framework that is familiar to you, such as Log4Net or System.Diagnostics.
* Event
  + typically used to capture user interaction with your service, to analyze usage patterns.
* Metric
  + used to report periodic scalar measurements.
  + Typically required when the app wants automatically scale with the number of policy actions that it is performing

App Insights Sampling

* Is the recommended way to reduce telemetry traffic, data costs and storage costs
* reduces traffic and data costs, and helps you avoid throttling
* Sampling is an effective way to reduce charges and stay within your monthly quota
* Can set in the portal (Usage and Estimated Costs blade)
* or in ASP.NET SDK in the .config file; or in the Java SDK in the ApplicationInsights.xml file
* 3 types
  + Adaptive sampling
    - default sampling when you use the ASP.NET or ASP.NET Core (on by default)
    - automatically adjusts to the volume of telemetry that your app sends
    - operates automatically in the SDK in your web app so that telemetry traffic on the network is reduced
  + Fixed-rate sampling
    - reduces the volume of telemetry sent from server and from your users' browsers
  + Ingestion sampling
    - happens at the Application Insights service endpoint by discarding some of the telemetry that arrives from your app

Using Application Insights, how can I see collected telemetry from my apps?

1. Smart Detection
   1. Automatically warns you of potential performance problems and failure anomalies in your web apps
2. Application Map
   1. Helps spot performance bottlenecks or failure hotspots across all components of the distributed app
3. Live Metrics
   1. Real time information about performance
4. Failures
   1. Provides details about issues detected inside your app like exceptions and server errors

Live Metrics Stream

* Can Monitor your live in-production web application or function in the portal
* Get exceptions as they happen
* Diagnose issues live

Application Insights Web Test

* Use App Insights to monitor the availability and responsiveness of web apps that have been deployed on the Azure platform
* This monitoring is configurable with multiple test types for web apps
* Visual Studio Enterprise license is req’d for creating multistep web test

Azure App Insights Usage

* Users
  + How many people used your app and its features
* Sessions
  + How many sessions, where a session is counted after a ½ hour of inactivity, or 24 hours of continues use
* Events
  + How often certain pages/feature of your app are used
* Funnels
  + progression through a series of steps in a web application is known as a funnel
  + correlation
* Cohorts
  + defined by an analytics query
  + more granular filters
* Impact
  + analyzes how load times and other properties influence conversion rates for various parts of your app
  + discovers how any dimension of a page view, custom event, or request affects the usage of a different page view or custom event
* Retention
  + how many users return to your app
  + how often they perform particular tasks or achieve goals
* User Flows
  + visualizes how users navigate between the pages and features of your site
  + repetitive actions

Alerts and Action Groups

* alert rules have conditions and actions
* action groups are a collection of notification preferences defined by the owner of an Azure subscription
* action groups are used by azure monitor and service health to notify users that an alert has been triggered
* Each action is made of up of the following properties
  + Type – notification or action performed
  + Name – unique identifier of the group
  + Action – additional action like webhook

Metrics and Alerts CLI

* Az monitor alert
* Az monitor metrics alert

App Insights Code based deployments

* You need an existing Application Insights instance already set up
* Define Instrumentation Key
  + Unique identifier for the app insights instance
* specify the logging level
* App Insights Package

Types of Availability Tests

* URL Ping test
  + A single url test that you can create in Azure portal
  + Can specify frequency, can enable 3x retries, send alerts,
* Multi-step web test
  + A recording of a sequence of web requests which can be played back to test more complex scenarios
  + Can specify frequency, can enable 3x retries, send alerts,
* Custom track availability tests
  + Send info about availability of an app or function using TrackAvailability() method from the sdk

App Insights CLI

* az monitor app-insights --help
* Delete
* az monitor app-insights component delete --app funapp123 --resource-group rg-test-123

Application insights .NET classes

* You send messages to Application Insights by using the TelemetryClass class
* You send custom events using the TrackEvent() method
* GetMetric() returns a metric from the azure.applicaitoninsight namespace
* TrackValue() adds a value to the return of GetMetric()

**Connect to and consume Azure services and third-party services (15-**

**20%)**

**Implement API Management**

create an APIM instance

configure authentication for APIs

define policies for APIs

**Develop event-based solutions**

implement solutions that use Azure Event Grid

implement solutions that use Azure Notification Hubs

implement solutions that use Azure Event Hub

import OpenAPI definitions

**Develop message-based solutions**

implement solutions that use Azure Service Bus

implement solutions that use Azure Queue Storage queues

API Management (APIM)

* Azure service to create consistent and modern AI gateways for existing back-end services
* Provides secure, scalable API access for your applications
* Layer of protection
* Allows us to create policies
* Unify all back-end APIs in a common interface that can be offered to external users
* You add back-end APIs to an APIM instance
* Grant access to APIM instance in Developer portal
  + https://apiminstance.developer.azure-api.net
* Need to associate a product to your API for publishing it
  + 2 types - Starter and unified

Authentication for APIs

* A subscription manages the keys that a developer can use for accessing your API
* Can assign different users to the sub using the Users param in the new subscription panel
* 3 scopes for applying a subscription
  + Product
    - All APIS config’d in the product assigned to the sub
  + All APIs
    - In the APIM instance
    - When using this approach, don’t have to associate the back end API with an API
  + API
    - A single API in the APIM instance using a sub key

API Management Components

1. API Gateway
   1. Accepts API calls and routes them to your backends
   2. Verifies API keys, JWT tokens certificates and other credentials
   3. Encourages quotas and rate limits
   4. Caches backend responses
2. Azure Portal
   1. The administrative interface where you set up your API program
   2. Defines or imports API schema
   3. Set up policies like quotas or transformations on the APIs
   4. Package APIs into products
   5. Manage users
   6. Publish the developer portal in the Portal overview blade of API Management
3. Developer portal
   1. Web user interface where they can read API documentation
   2. Read AP docs
   3. Create an account to sub and get APIs keys
   4. Try out AP via the interactive console
   5. Access analytics

Versions

* Allow to present groups of related APIs to the devs
* Use version to handle breading changes in your API safely
* Typically used for breaking/major changes

Revisions

* Allow you to make changes to the APIs in a controlled and safe way, without disturbing your API consumers
* Typically used for non-breaking/minor changes
* Each version can have multiple revisions
* You can use revisions without using versions OR use versions without using revisions

Products and Groups inside API Management

* Products are how APIs are surfaced to devs, and have 1 or more APIs, title, description and terms of use
* Groups are used to manage the visibility of products to devs
* 2 products are created by default
  + Starter
    - does not req admin approval
    - 5 calls a minute
  + Unlimited
    - req’s admin approval

Products

* Can be open or protected. Protected products must be subscribed to before they can be used
* When a product is ready for use by devs, it can be published for devs who use it
* Subscription approval is configured at the product level. Devs need this subscription to access products

Azure API Management Groups

* Administrators
  + Manage APIs
  + Can also create custom groups or leverage external groups in associated A/D tenants
* Developers
  + Granted access to the dev portal and build apps that call operations of the API
  + Can be created to invited to join by admins
  + Can also sign up from dev portal
  + Each dev is a member of one or more groups and can sub to the products that grant visibility to those groups
* Guests
  + Unauthenticated developer portal users with certain read only access, such as the ability to view APIs but not call them

API Management CLI

* az apim api create --service-name MyApim -g rg-testing-111 --api-id MyApi --path '/myapi' --display-name 'My API'

API Management Powershell

* New-AzApiManagementBackendProxy -Url $ApiUrl
  + creates a new Backend Proxy Object which can be piped when creating a new Backend entity
* AzApiManagementApi
  + imports an Azure API Management API from a file or a URL in Web Application Description Language (WADL), Web Services Description Language (WSDL), or Swagger format.
* New-AzApiManagementBackend
  + creates a new backend entity in Api Management.
* New-AzApiManagement
  + creates an API Management deployment in Azure API Management.

API Management pricing tiers

* Consumption – 99.95 SLA
  + No internal caching system, have to use external (Redis)
* Developer – 1 scale unit so no SLA
* Basic – 2 scale units, 99.95
* Standard – 4 scale units, 99.95
* Premium – 12 scale units, 99.99
* (Isolated) – in preview, 99.99

API Management cache types

* Internal
  + cache provided within the API management service
  + limited in size based on the AP management tier
  + not available for consumption tier (use External - Azure Redis cache instead)
* External
  + Redis compatible
* Both internal and external cashing types of configured in the API management policies

APIM access restriction policies

* Check HTTP header - Enforces existence and/or value of a HTTP header.
* Limit call rate by subscription - Prevents API usage spikes by limiting call rate, on a per subscription basis.
* Limit call rate by key - Prevents API usage spikes by limiting call rate, on a per key basis.
* Restrict caller IPs - Filters (allows/denies) calls from specific IP addresses and/or address ranges.
* Set usage quota by subscription - Allows you to enforce a renewable or lifetime call volume and/or bandwidth quota, on a per subscription basis.
* Set usage quota by key - Allows you to enforce a renewable or lifetime call volume and/or bandwidth quota, on a per key basis.
* Validate JWT - Enforces existence and validity of a JWT extracted from either a specified HTTP Header or a specified query parameter

Authentication to APIM

* Validate credentials for a caller of an API
* Options
  + **Basic Auth**
  + **Client Certificate Auth**
    - API Management allows to secure access to the back-end service of an API using client certificates
* Managed Identity Auth
  + Managed Identity is preferred because calls are secure and do not require passwords
* <authentication-basic username="username" password="password" />
* <authentication-certificate thumbprint="thumbprint" certificate-id="resource name"/>
* <authentication-managed-identity resource="resource" client-id="clientid of user-assigned identity" output-token-variable-name="token-variable" ignore-error="true|false"/>

Securing backend API

* Use OAuth 2.0 authZ with Azure AD
* validate-jwt management policy to pre-authorize requests in API Management

Azure APIM Policies

* Each policy has 4 sections
  + Inbound
    - statements to be applied to the request go here
    - Example - Forward the name of the product associated with the subscription key in the request to the backend service.
    - Example - Forward the user id associated with the subscription key in the request as well as the region where the proxy processing the request is hosted
    - Setting variables
  + Back End
    - statements to be applied before the request is forwarded to the backend service go here
  + Outbound
    - statements to be applied to the response go here
  + On-Error
    - not present in policies by default
* Are defined using XML format
* Allow changing the behavior of the API through configuration
* Also a collection of statements that are excluded sequentially on the request or response of an API
* Can be used to format conversion from XML to JSON
* Restrict the amount of incoming calls
* Enforces the existence and/or value of an http header
* Caches response according to the specified cache control configuration
* 512 max body size

Access restriction policies

* Limit call rate by key (incoming call rate)
* Validate JWT tokens
* Set usage quota by key
* Check http header presence
* Limit call rate by subscription

Advanced policies

* Mock response
* Forward request
* Retry
* Set request method
* trace

Transformation policies

* Convert xml to json
* Convert json to xml
* Find and replace string in body
* Set backend service
* Set query string parameter

Caching policies

* Store to cache
* Get from cache
* Remove value from cache

Policy scope

* Global scope – affects all APIs within the instance of API management
* Product scope – manages access to the product as a single entity
* API scope – affects only a single API
* Operation scope – affects only one operation within the API
* Inbound policies execute when a request is received from a client
  + Includes information inside the request that has to be forwarded to backend or outbound
* Backend policies execute before a request is forwarded to a managed API
* Outbound policies execute before a response is sent to a client
* On error policies execute when an exception is raised

API Management caching policies

* Cache-lookup …
* caching-type
  + internal to use the built-in API Management cache. Not available on consumption plan
  + external to use the external cache as Azure Cache for Redis prefer-external to use external cache if configured or internal cache otherwise.
* downstream-caching-type
  + none - downstream caching is not allowed.
  + private - downstream private caching is allowed.
  + public - private and shared downstream caching is allowed.

Set back-end policy

* set-backend-service policy to redirect an incoming request to a different backend than the one specified in the API settings
* use of when conditions to decide between versions

Example policy

<policies>

<inbound>

<!-- statements to be applied to the request go here -->

</inbound>

<backend>

<!-- statements to be applied before the request is forwarded to

the backend service go here -->

</backend>

<outbound>

<!-- statements to be applied to the response go here -->

</outbound>

<on-error>

<!-- statements to be applied if there is an error condition go here -->

</on-error>

</policies>

API Management advanced policies

<choose>

<when condition="Boolean expression | Boolean constant">

<!— one or more policy statements to be applied if the above condition is true -->

</when>

<when condition="Boolean expression | Boolean constant">

<!— one or more policy statements to be applied if the above condition is true -->

</when>

<otherwise>

<!— one or more policy statements to be applied if none of the above conditions are true -->

</otherwise>

</choose>

On-Error Policy

* The on-error section is not present by default in API policies
* Policies allow in on-error
  + choose
  + set-variable
  + find-and-replace
  + return-response
  + set-header
  + set-method
  + set-status
    - sets the HTTP status code to the specified value
  + send-request
  + send-one-way-request
  + log-to-eventhub
  + json-to-xml
  + xml-to-json
  + limit-concurrency
  + mock-response
  + retry
  + trace

The context variable in policy expressions

* Available implicitly (so we don’t have to declare it)
* provides access to the request and/or response parameter of the API call
* also provides access to information about your APIM service instance
* context.Response.StatusCode()
* context.LastError.Message.Contains()

Configure an API to use client certificate for gateway authentication

* Target
  + http(s) endpoint
  + Azure Logic App
* Gateway credentials
  + None
  + Basic
  + Client cert

Messages vs Events

* Messages contain raw data from a source system, to be consumed by the receiving component
* Messages are expected to be handled by a receiver
* Messages can follow publisher/subscriber (1-to-many) or a producer/consumer (1-to-1) model
* With messages, the publisher application has the expectation that the message is processed or stored by the subscriber
* Events are lighter that messages
* Events are lightweight notification of a condition or change of state
* Events are used in broadcast communications – publisher and subscribers
* **Asynchronous file uploads are a great fit for events**
* If a message is published without any knowledge or expectations of how it will be handled, then it’s deemed to be an event
* The publisher doesn’t know (or care) how events are handled

Types of Events

1. Discrete events
   1. Report state change form a system and enable subscribers to take action
   2. Event Grid
   3. Example – account was created or image was uploaded
2. Series Events
   1. Report a condition and enable subscribers to analyze a condition over time
   2. Event Hub
   3. Example - telemetry
3. User notification
   1. Prompt use or their device for attention
   2. Notification Hub

Event

* The data messages passing through event grid that describe what has taken place
* What happened
* Change of state in the source
* The smallest amount of information that fully describes something happened in the system
* Does not contain the saw data, only the metadata to describe it
* Limited to 64 kb data
* When you need to work with events, you configure an event source to send events to a topic
* Any event handler that needs to process those events subscribes to that topic

Event Schema

[

{

"topic": string,

"subject": string,

"id": string,

"eventType": string,

"eventTime": string,

"data":{

object-unique-to-each-publisher

},

"dataVersion": string,

"metadataVersion": string

}

]

Topics

* And endpoint where an event source sends events
* An event source can have one or more topics, specified by the publisher
* 1 to many
* Collection of related events (that Event Grid will take care of)
* A publisher will send its events to a topic

Events handler

* Who handles the event (an app or service)
* Can be an Azure service or http webhook
* Where Event Grid will send the notification to
* Azure Functions, Event Hubs and Service Bus, Web hooks, and Storage Queues

Publisher

* User or organization that sends events
* Several publishers on Azure – App configuration, App Service, Blob Storage, Event Hubs, IoT Hub, Container Registry, M/L Resource Groups, Service Bus, etc;
* Can create your own publishers
* Where it happened

Event source

* Where the event happened
* responsible for sending events to Event Grid
* the resources generating the events
* Every event has an event source

Subscription

* Event routing
* Event grid routes and filters events to handlers
* Apps subscribe to a topic

Event Subscription

* The endpoint to route events
* define which events on a topic an event handler wants to receive
* How you inform the event grid what events you are interested in listening to
* When creating the subscription, you provide an endpoint to handle the received events

Event Delivery

* Event grid supports exponential back off retry policy
* Expires after 24 hours

Publisher/Subscriber model

* messages are published to an intermediary broker rather than communicating directly with consumers
* 1 to many
* An event signifies something changed
* Publishers has no expectation of what happens with event
* Subscriber determines what to do with event
* frequently associated with stateful application (that care about ordering)

Producer/Consumer model

* Each message the producer produces will be consumed by a single consumer

Polling

* Manually checking for messages

**Azure Event Grid**

* Takes information in, filters it, then passes it along to an event handler so it can be processed
* Runs of top of Azure Service Fabric
* Designed to process or route events
* **1MB max size of event**
* Event routing service for sending messages between services
* Reactive programming, apps can react to changes
* takes an event source such as IoT Hub, or storage, and sends the event handler for processing
* Event sources (input) -> Event handlers (output)
* connects data sources and event handlers
* **uses publish/subscribe model;** 1 to many model, as many subscribers as are listening for events
* **does not guarantee the order of events because each event is handled independently**
* publishers emit events, and subscribers consume (Azure services or custom)
* filtering of events that match rules
* designed for event based and near-real time apps
* can send HTTP requests to notify events that happen in publishers
* can trigger Azure Function
* **a distribution system, not a queueing mechanism. Events that are pushed in are pushed out immediately (unless we send undelivered events to a storage account – dead lettering)**
* **cannot be configured to keep messages infinitely, pair it with another messaging system for that (service bus or storage queue)**
* dynamically scalable up and down, all the way to zero if required (pay for what you use)
* serverless, low cost
* The first 100,000 operations per month are free.
* supports dead lettering for events that aren’t delivered to an endpoint
* **eliminates the need for constant polling**
* **at least once delivery**
* **does not supporting event capture**
* **No native support for partitioning, but partitioning can be achieved with filtering**
* Need to enable event grid on the subscription level
  + az provider register --namespace 'Microsoft.EventGrid'
* Types
  1. System topics
     1. Built in products such as Azure Storage, Azure Event Hubs, Azure Service Bus, etc;
  2. Custom topics
     1. Application and third party topics
     2. SAS or key authentication
  3. Partner topics
     1. Integration with SaaS and PaaS products from 3rd parties
     2. 0Auth is an example

Use Event Grid when you need

* Simplicity in connecting sources to subscribers in Event Grid
* Advanced filtering where subscription have close control over the vents the receive from a topic
* Fan out – unlimited amount of endpoints
* Pay per event
* Reliability – event grid retries event delivery for up to 24 hours for each subscription

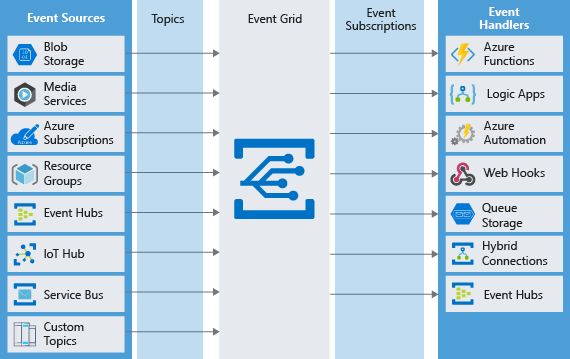
EventGrid CLI commands

* az eventgrid topic create --resource-group $myResourceGroup --name $myTopic --location westus2
* az eventgrid event-subscription create
* Az eventgrid event-subscription create –source-resource-id $topicid –name $name –endpoint-type servicequeue –endpoint $endpoint

When setting up an Event Grid solution (Workflow)

1. Create Topic
2. Send publisher events
3. Add subscriber info with filtering

* Every event has the same metadata schema
* The data property contains event specific information
* You need to create the Event Grid Topic before the Event Grid Subscription(s)
* You can create an Event Grid Topic without an Event Grid subscription, but not vice versa



* Event Grid and Event Hubs are essentially for event based communication

Event Grid supports two ways of validating the subscription

* ValidationCode handshake (programmatic)
  + Recommended If you control the source code for your endpoint
  + Ie webhooks events to Azure functions
* ValidationURL handshake (manual)

Event Grid Security

* Custom topics use either Shared Access Signature (SAS) or key authentication.
* Microsoft recommends SAS, but key authentication provides simple programming, and is compatible with many existing webhook publishers.
* SAS tokens have an expiration time

Event Grid with Blobs

* Common Blob storage event scenarios include image or video processing, search indexing, or any file-oriented workflow.
* Asynchronous file uploads are a great fit for events.
* When changes are infrequent, but your scenario requires immediate responsiveness, event-based architecture can be especially efficient

Event Schema properties

* topic, subject, id, event type, and data

Event subscription properties

* Destination
  + The object that defines the endpoint.
    - endpointType
      * type of endpoint for the subscription (webhook/HTTP, Event Hub, or queue).
    - endpointUrl
      * The destination URL for events in this event subscription.
* Filter
  + An optional field for filtering the types of events.
    - includedEventTypes
      * Match when the event type in the event message is an exact match to one of these event type names. Raises an error when event name does not match the registered event type names for the event source. Default matches all event types.
      * Microsoft.Storage.BlobCreated
    - subjectBeginsWith
      * A prefix-match filter to the subject field in the event message. The default or empty string matches all.
    - subjectEndsWith
      * A suffix-match filter to the subject field in the event message. The default or empty string matches all.
    - isSubjectCaseSensitive
      * Controls case-sensitive matching for filters.
    - enableAdvancedFilteringOnArrays
      * Enables using arrays for keys in advanced filtering. For more information, see Advanced filtering.

**Azure Event Hub**

* Hosted event telemetry stream processing platform
* Big data pipeline
* Used to inject events and stream them to another service
* Designed to process data
* can receive and process millions of events per second and provide low-latency event processing
* An Event publisher can send events to the event hub by using any of these protocols: AMQP 1.0, Kafka 1.0 (or later), or HTTPS
* Can send a single event or a batch of events (limited to 1Mb of data per publication)
* sending and receiving data
* Event Hubs accept endpoint for injection of data and don’t provide a mechanism for sending data back to publishers.
* Event Hubs keep data for up to 7 days
* Event Hub - can receive and process millions of events per second
* Capable of receiving and processing millions of events per second
* Application logging
* **Order is maintained**
* **No duplicate detection (only Service Bus has this)**
* **Supports long polling**
* All Event Hub triggers are long-polling triggers, which means that the trigger processes all the events and then waits 30 seconds per partition for more events to appear in your Event Hub
* **publish/subscribe model**
  + Any resource that sends events to an event hub is a publisher
  + consumers read data through an event hub; Multiple consumers for the same event hub
  + There is always a $default consumer group
* Telemetric
* **At least once delivery**
* Low latency
* Ideal for big data scenarios
* **Events are stored in a partition, specified by the partition key provided as one of the data of the event**
  + **Cannot be changed after creation**
* **Cannot be configured to store events for an infinite period of time**
* Use SAS for security
  + available at the namespace and event hub level
* once Azure Event Hub receives the data, it can deliver to Event Grid, or store events for archiving in a storage account or data lake
* Event Hubs can only accept events from the same region, regardless of the resource type doing the sending

Use Event Hub when you need

* Authentication of a large amount of publishers
* Need to save stream of events to data lake or blob storage
* Aggregation/analytics of event stream
* Reliable messaging or resiliency

Namespace

* Must be created before creating the Event Hub
* Container for one or more Event Hubs
  + 10 event hubs per namespace (Basic and Standard Tiers)
* URL
  + <namespace name> .servicebus.windows.net

Event Publisher/Producer

* Resources that send data to the Event Hub

Partitions

* Like a bucket or queue for incoming event messages
* Always stored in the order received
* Events are not deleted after read (only after message retention period has ended)
* Event Hubs decides which partition to send events to
  + The partition can be specified with a partition key
* Between 2 and 32 partitions in Standard
* Up to 2000 partitions per capacity unit in Event Hubs Dedicated
* cannot change number of partitions after resource creation

Event Retention

* The default value and shortest possible retention period is 1 day (24 hours).
* For Event Hubs Standard, the maximum retention period is 7 days.
* For Event Hubs Dedicated, the maximum retention period is 90 days.
* If you change the retention period, it applies to all messages including messages that are already in the event hub.

Consumer groups

* View of an Event Hub
* Connect to receiver apps
* Apps that read data from Event Hub
* Using consumer groups, can have different event receivers or consumers, accessing different views (state, position, or offset) of the partitions in the event hub
* There can be at most 5 concurrent readers on a partition per consumer group; however it is recommended that there is only one active receiver on a partition per consumer group

Event Hub CLI commands

* Create the namespace
  + Az events namespace create –name namespace1121 –resource-group rg-event-1121 –l canadacentral –sku Basic
* Create the Event Hub
  + Az eventhubs eventhub create –name eventhub1121 –namespace rg-event-1121 –message-retention 1

Sending events to Event Hub solution (Workflow)

1. Install .NET SDK (via NuGet)
2. Obtain Connection Info (namespace name and key)
3. Open connection (EventHubProducerClient)
4. Prepare data (convert to binary)
5. Send data (single or batch events)

Reading events from Event Hub solution (Workflow)

1. Install .NET SDK (via NuGet)
2. Obtain Connection Info (namespace name and key)
3. Open connection (EventHubConsumerClient)
4. Retrieve data (the connection to Event Hubs remains open until the next event comes in)
5. Decode data

Tiers

* Basic – 1 consumer group, 100 brokered connections, capture not available, 1 day message retention
* Standard – 20 consumer groups, 1000 brokered connections, capture available, up to 7 days message retention
* Dedicated - up to 1000 consumer groups, capture included, up to 90 days message retention

Throughput Units

* Capacity measurement for Event Hub
* Can specify throughput units between 1 and 20 in any tier
* Can enable auto inflate

Event Receiver

* The app that is part of the consumer group that reads the data

Event Hub options

* Apache Kafka
  + Let’s existing Apache Kafka clients send data to Event Hub
  + Fully managed PaaS offering
  + Only available on standard & Dedicated tiers
  + Enabled by default
* Capture
  + Save incoming event to Azure Storage or Azure Data Lake
  + Time and Size Windows for how frequently data will write to storage

**Azure Notifications Hubs Solutions**

* Send push notifications to users
* Send to multiple platforms – iOS, Android, Windows
* Abstraction over platform notification services
* Cross platform (both front and back end)
* Multiple delivery formats (push to user, push to device, localization, silent push)
* Telemetry
* Scalable
* Notification Hubs have to belong to a namespace (typically one namespace per application)
* .NET SDK to send pushes

Notification Hub Components

* Notification Hub itself
  + Holds the push information for a single app
* Notification Namespace
  + Holds a collection of Notification hubs for a single region

Common uses

* Breaking new
* Sports notifications
* Location based coupons for targeted users
* MFA
* Newly assigned tasks
* Work items

Namespaces

* Collection of Notification Hubs (one or more Hubs)
* Typically only 1 Namespace per application
* 1 hub per app environment
* Credentials at namespace level
* Billing at namespace level

Sending notifications (Workflow)

1. Setup PNS (Platform Namespace Service)
2. Setup ANH (Azure Notification Hub Instance)
3. Map PNS to ANH (apply keys)
4. Register devices using .NET SDK
5. Send pushes using .NET SDK via web API

How it works

1. Mobile app checks for notifications
2. Platform Notification Service (PNS) responds with temp handle
3. Mobile app sends request to app backend
4. App backend stored handle and sends notifications to Notification Hub
5. Notification Hub sends notification to PNS
6. PNS sends notification to mobile app

Notification Hub CLI

* Az notification-hub namespace create –resource-group rg-thomas-000 –name notificationnamespace000 –location CanadaCentral –sku Basic
* az notification-hub create –resource-group rg-thomas-000 –namespace-name notificationnamespace000 –location CanadaCentral –name notificationhub000

Tiers

* Free
* Basic
* Standard

.NET

* NotificationHubClient hub = NotificaitonHubClient.CreateClientFromConnectionString()
* hub.SendWIndowsNativeNotificaitonAsync(windows pay load)

Messaging benefits

* Encourages application logic modularity
* Enables fault tolerance between modules (when one component goes down we don’t lose any data)

Messages

* A message is in binary and contains Json, XML or just text
* async

Selecting a Messaging based Service

* If the total storage for queue needs to be over 80GB? Use Queue Storage
* If you need logs for all transactions executed against queue? Queue Storage
* Track the progress of messaging processing? Queue storage
* If you need support for receiving messages without polling (ie AMQP protocol)? Use Service Bus
* Guarantee messaging ordering (FIFO)? Service bus
* Detect duplicate messages? Service bus
* Messages up to 256 Kb? Service bus
* Need support topic based notifications (1 to many)? Service bus
* Need to support publishing and consuming in batches? Service Bus

**Azure Service Bus (vs Queue Storage)**

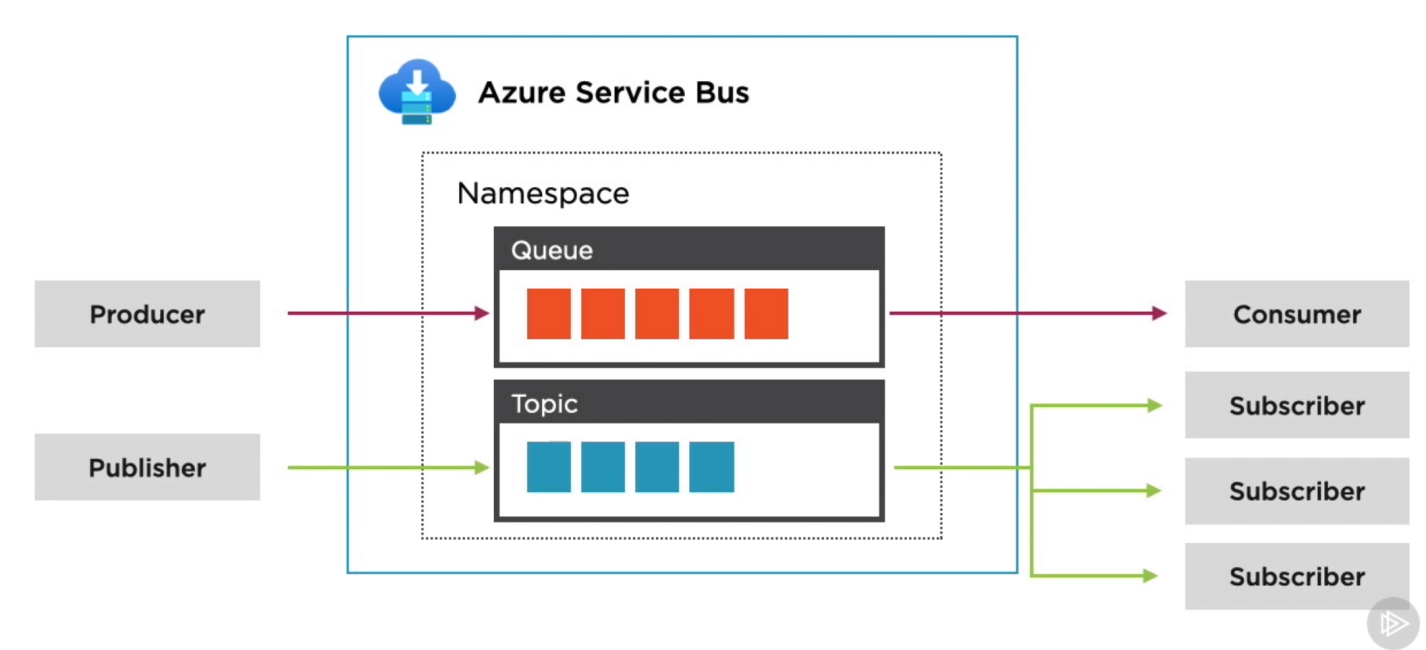
* Service Bus Namespaces are a container for service bus queues
* A brokered messaging system, storing messages in a broker until the consuming party is ready to receive the messages
* Commonly used for order processing and financial transactions
* Multiple modes of messaging
* **in order delivery (optional)**
* **FIFO**
* reliable asynchronous message delivery system
* **At least once delivery**
* advanced features – ordering, batching, dead letter queue**, duplicate detection**
  + **the only messaging service with duplicate detection**
* High value enterprise messaging
* **Can store messages for unlimited period of time**
* Sending messages or topics (and not events)
* Supports http/https and AMQP protocols
* Azure Service Bus namespaces are constrained to a single region.
* Includes messaging for both queues and topics
* **Messages sit in the queue until they are read**
* **Once a message is read it is destructed**
* **Can enable partitioning on either topic or queue**
* Can facilitate highly secure communication across hybrid cloud solutions and can connect existing on-premises systems to cloud solutions
* Sender and receiver act independently
* Optional polling (long-polling – set to 1 minute by default)
* **Queues within Azure service bus follow the producer/consumer model**
* **Topics within Azure service bus follow the publisher/subscriber model**
* URL Structure
  + https://<namespace name>servicebus.windows.net/<queuename>
* two ways to authenticate and authorize access to Azure Service Bus resources: Azure Activity Directory (Azure AD) and Shared Access Signatures (SAS)

Use Service Bus Queues if you

* Need to handle messages larger than 64 KB but less than 256 KB.
* Queue size will not grow larger than 80 GB; Use Azure Storage Queue otherwise
* 1 to 1 relationship between sender and receiver
* Want to publish and consume batches of messages

Use Service Bus topics if you

* need multiple receivers to handle each message (1 to many)
* publish/subscribe model
* Consumers create a subscription to a topic (a subscription is a dedicated queue into a topic)
* Every subscriber needs its own configuration options
* Each receiver will have its own queue to pull messages from

****

Basic Tier

* Does not support topics
* Consumption model

Standard Tier

* Pricing is pay as you go
* Messages up to 256KB
* Shared resources
* Does not Support geo-disaster or availability zones

Premium Tier

* Pricing is fixed based on messaging units
* Dedicated resources
* Supports geo-disaster and availability zones
* Messages up to 1Mb
* Supports integration with Event Grid

Azure Service Bus CLI Commands

* Create a namespace
  + az servicebus namespace create --resource-group ContosoRG --name ContosoSBusNS --location eastus
* Create a topic
  + Az servicebus topic create –namespace-name pluralsight –name testtopic –resource-group pluralsight
* Delete a topic
  + Az servicebus topic delete –namespace-name pluralsight –name testtopic
* Crete a queue
  + Az servicebus queue create –namespace-name pluralsight –name testqueue –resource-group pluralsight
* Delete a queue
  + Az servicebus queue delete –namespace-name pluralsight –name testqueue
* Get connection string for the namespace used to connect to the queue and send/receive messages
  + az servicebus namespace authorization-rule keys list --resource-group rg-thomas-000 --namespace-name ContosoSBusNS --name RootManageSharedAccessKey --query primaryConnectionString --output tsv

Azure Service Bus Topic Filters

* Each newly created topic subscription has an initial default subscription rule. If you don't explicitly specify a filter condition for the rule, the applied filter is the true filter that enables all messages to be selected into the subscription
* Boolean filters
  + all or none of the messages are selected
* SQL filters
  + More sophisticated rules
  + sys.Label LIKE '%bus%'` OR `user.tag IN ('queue', 'topic', 'subscription')
* Correlation filters
  + matched against user and system properties in the message
  + checked for equally and equally only (! or < not allowed)
  + very simple and efficient

Service Bus Dead letter Queue (for queue or topic)

* holds messages that can't be delivered to any receiver, or messages that couldn't be processed
* a separate queue
* default value is after 10 attempts

Azure Service Bus Namespace and Policies

* SAP (Shared Access Policy) is created at the namespace level
* RootManageSharedAccessKey is created by default and allows the user to Manage, Send and Listen to messages

Service Bus message counters

|  |  |
| --- | --- |
| ActiveMessageCount | Number of messages in the queue or subscription that are in the active state and ready for delivery. |
| ScheduledMessageCount | Number of messages in the scheduled state. |
| DeadLetterMessageCount | Number of messages in the dead-letter queue. |
| TransferMessageCount | Number of messages pending transfer into another queue or topic. |
| TransferDeadLetterMessageCount | Number of messages that failed to transfer into another queue or topic and have been moved into the transfer dead-letter queue. |

* Message count – is the total messages in the queue with various statuses (scheduled, deferred, dead-lettered, in transfer, transfer dead-lettered

Service Bus important properties

* SessionId (group-id)
  + For session-aware entities, this application-defined value specifies the session affiliation of the message
* MessageId (message-id
  + The message identifier is an application-defined value that uniquely identifies the message and its payload
* DeliveryCount
  + Number of deliveries that have been attempted for this message
  + Read only
* CorrelationId
  + Enables an application to specify a context for the message for the purposes of correlation
  + reflecting the MessageId of a message that is being replied to
* SequenceNumber
  + a unique 64-bit integer assigned to a message as it is accepted and stored by the broker
  + Read only
* ReplyToSessionId
  + This value augments the ReplyTo information and specifies which SessionId should be set for the reply when sent to the reply entity

Service Bus Security

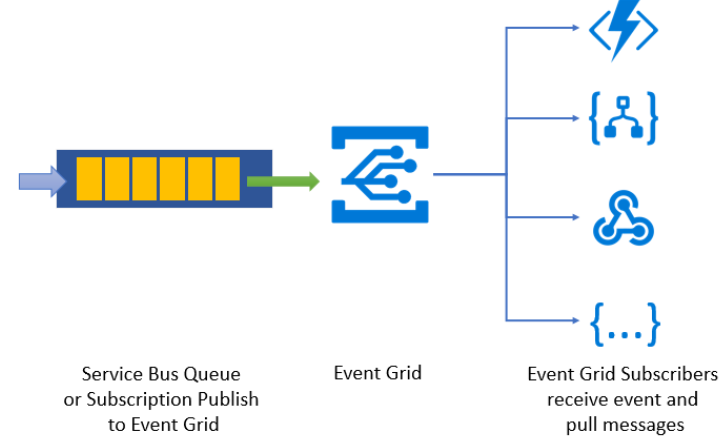
* We use SAS to access Service Bus Queue’s and Topics
* Root SAS created by default on the Namespace (RootManagedSharedAccessKey)
* Can create custom SAS on Queue, Topic or Namespace
* SAS options – manage, send, listen

.NET

* Using topic client, call subscriptionClient
* RegisterMessageHandler which is used to receive messages continuously from the entity
* Send message
  + ServiceBusMessage message = new ServiceBusMessage("Hello world!");
  + await sender.SendMessageAsync(message);
* Receive message from queue
  + await args.CompleteMessageAsync(args.Message);

Azure Event Grid and Azure Service Bus together

* Azure Event Grid is commonly used to respond to Service Bus Events

****

| **Service** | **Purpose** | **Type** | **When to use** |
| --- | --- | --- | --- |
| Event Grid | Reactive programming | Event distribution (discrete) | React to status changes |
| Event Hubs | Big data pipeline | Event streaming (series) | Telemetry and distributed data streaming |
| Service Bus | High-value enterprise messaging | Message | Order processing and financial transactions |

Use the services together

* In some cases, you use the services side by side to fulfill distinct roles.
* For example, an e-commerce site can use Service Bus to process the order, Event Hubs to capture site telemetry, and Event Grid to respond to events like an item was shipped.

Azure Service Bus 🡪 Event Grid integration overview

* To enable the feature, you need the following items:

1. A Service Bus Premium namespace
   * with at least one Service Bus queue or a Service Bus topic with at least one subscription.
2. Contributor access to the Service Bus namespace.
   * Navigate to your Service Bus namespace in the Azure portal, and then select Access control (IAM), and select Role assignments tab.

**Azure Storage Queue (vs Azure Service Bus)**

* **Once a message is read it is destructed**
* **1 to 1 messaging service**
* Cheaper than service bus
* Higher storage limits than Service bus
* Fully managed part of the storage account (requires a general purpose v2 storage account)
* Messages up to 64Kb in size
* Maximum queue size is 500 TB
* Messages exist in single queue (cannot span queues)
* **customizable time to live for messages (7 days is the default), can be customized to never expire**
* Data redundancy (as with storage accounts)
* **No duplicate detection (only service bus has this)**
* url structure
  + https://**nameofstorageaccount**.queue.core.windows.net/**queuename**
* Data within a queue is encrypted by default
* Interactions are done via http/https or in the portal
* You can only access an Azure Queue using the REST API or using the .NET Azure Storage SDK
* **Does not guarantee ordering**
* **Messages in Storage queues are typically first-in-first-out, but sometimes they can be out of order**
* **Therefore, No FIFO guarantee**
* Is the only messaging service that allows messaged to be edited before they are deleted

**Use queue storage if you**

* need an audit trail of messages that pass through the queue
* **Need > 80Gb**
* Want to track progress for processing a message
* Don’t need to maintain ordering

Azure Storage Queue CLI Commands

* az storage queue create –name myqueue
* az storage queue delete –name myqueue
* az storage message peek –queue-name myqueue (doesn’t affect visibility)
* az storage message get –queue-name myqueue (does affect visibility)
* az storage message delete –id <idofmessage> --pop-receipt <pop-receipt> --queue-name <queue-name>

Visibility timeout

* Specifies the new visibility timeout value, in seconds, relative to server time. The default value is 30 seconds

Messages are delivered to consumers, but not immediately deleted from the queue (they become invisible)

Visibility timeout determines when messages will become visible again

This ensures fault tolerance for your applications

Queue security

* Shared key
* SAS
  + No more than 5 stored Secure Access Policies
* Azure AD
* Firewall and Virtual Networks

Queue .NET commands v 11

* CreateCloudQueueClient()
* CreateQueueClient()
  + Create a queue
* QueueDescription.LockDuration
  + Sets duration of a peak lock (default is 30 seconds)
* PeekMessage
  + Peeks the next message In the queue
* ReceiveMessages (Dequeue the next message)
  + When you call ReceiveMessages, you get the next message in a queue. A message returned from ReceiveMessages becomes invisible to any other code reading messages from this queue. By default, this message stays invisible for 30 seconds, then goes back into the queue
* DeleteMessageAsync
  + delete the message from the queue
* **you must call DeleteMessage after ReceiveMessage to finish removing the message from the queue (both v11 and v12)**
* **otherwise, the message will be deleted when it expires**

Queue libraries new vs old

**.NET v12 QueueClient**

.NET v11CloudQueueClient