**AZURE INFRA GENERAL UNDERSTANDING**

* Elasticity, Agility and Economies of scale are the benefits that you can gain from using cloud services.
* A hybrid cloud is a public and private cloud combined to run your new applications on commodity hardware you rent from the public cloud and maintain your specialized mainframe hardware on-premises.
* PaaS services handle the IT management tasks for you, so you can focus on writing code.
* Azure Load Balancer provides networking capabilities to balance network traffic.
* Azure App Service is an HTTP-based service that enables you to build and host many types of web-based solutions without managing infrastructure.
* Azure Table Storage is a data storage model to store structured data.
* Azure Cache for Redis is a caching storage technology to improve performance of servers.
* Scaling out means adding additional systems, such as virtual machines. For example, you might create many virtual machines configured in the same way and use a load balancer to distribute work across them.
* Azure DevOps Overview ( a brief history of the service and how to set up an instance for the team)
* Azure Boards (What are azure boards and how u use them to manage ur work)
* Azure Repos ( used to store code and manage pull requests)
* Azure Pipelines ( used to set up fully automated builds and releases)
* Azure Artifacts ( used to set up our own organised/administered package management feeds)

Deploy and Configure Infrastructure

Analyze resource utilization and consumption

Create and configure storage accounts

Create and confi

Azure Infra Admin course

Contents

* Manage Azure subscriptions and Resources

Introduction to Accounts and subscriptions

* Implement and manage storage
* Deploy and manage virtual machines
* Configure and manage virtual networks
* Managing and securing Identities

Step 1: Pushed the code to git hub

A remote repository created in git hub.

Web api code stored in a dir where git initialized local repo.

Git add all the folders.

Got a warning msg

LF will be replaced by CRLF in Angularcrud/.deployment.

The file will have its original line endings in your working directory

you can turn this warning off by typing the following in the git command line

git config core.autocrlf true

Git commit

**Tags in Azure**

* Tags are used to logically organize resources into a group.
* Tags are also used to group billing data.
  + Eg: Name-Value pairs
  + Environment-Production
  + Eg: if multiple vms running on different organizations, Tags are used to group usage by Cost centre
* Limitations:
  + Each resource max 50 tags. Storage accounts support 15 tags.
  + Generalized Vms don’t support tags.
  + The user should be contributor to apply tags.

**Tags in Git/ Azure DevOps**

* Tags are used to refer specific points in history to mark release points.
* Lightweight tags are a pointer to specific commit.
* Annotated tags contain more information such as the tagger, message, and date.
* Tagging old commits can be done using the part of commit id.
* Retagging/Replacing old tags can be done using –f force option.
* Pushing tags to remote using --tags option to git push command.
* git checkout v1.4(tag name) will put the repo in detached state and changes made will not update the tag.
* A branch can be created from a Tag by right clicking an existing tag name in Azure repos.

**Manage Azure resources by using Azure PowerShell**

1. Used in deploying a resource (storage account)
2. Used to deploy a template to create a storage account
3. Used to deploy resources to multiple subscriptions or to different resource groups
4. Used to delete resources
5. Used to move a storage account from one RG to another RG
6. Used to lock a storage account
7. Used to tag resources and resource groups
8. Used to manage access to resources using RBAC

**ARM TEMPLATES**

ARM templates are used to implement infrastructure as code for Azure.

“$schema": Location of the JSON schema file

"contentVersion": Version of the template (such as 1.0.0.0)

"apiProfile": An API version that serves as a collection of API versions for resource types

"parameters": Defines inputs to deployment

"variables": Reused values to simplify repeating complicated expressions

"functions": User-defined functions that are available within the template

"resources": Resource types that are deployed or updated in a resource group or subscription.

"outputs": Values that are returned after deployment

Created templates using portal and visual studio for webapp. Customized template by adding resources like storage account. Deployed project to azure.

**.\Deploy-AzTemplate.ps1 -ArtifactStagingDirectory . -Location centralus -TemplateFile WebSite.json -TemplateParametersFile WebSite.parameters.json**

Later added code to project and deployed code with infrastructure.

Created multiple instances using copy{} --- specifies count and copyindex() –returns current iteration.

Setting resource deployment order. VM 🡪 Storage acc & NIC 🡪 public IP & vnet.

Used conditions to create new or existing resources, "condition": "[equals(parameters('newOrExisting'),'new')]",

**Agile Methodology & DevOps culture**

Agile methodology means

* Faster development
* shorter and stress-free release cycles
* easy and accurate rollbacks

One of the many ways to accomplish Agile is by using the tools which enable continuous integration, continuous deployment, test automation etc is DevOps.

the .cshtml file configures how the page looks

the .cs file determines the page’s behavior, or what it does

Program.cs 🡪 entry point for the program

Startup.cs 🡪contains code that configures app behavior and loads needed services and configuration the app is going to need to run

**Git**

Git one of the very important tool in DevOps and a successful agile workflow. It is

* speed through pointers,
* reliability through distributed architecture,
* support nonlinear development through easy branch-merge etc

|  |  |
| --- | --- |
| **Git** | **TFS** |
| Distributed Version control System | Centralised VCS |
| Frequent commits to local repo | no frequent check-ins until very sure of it |
| More often check in (even for small manageable code commits) | Less often check ins for Big code chunks |
| Tracking history is easier and hence reverts are also easier | Reverting is difficult, because we might want to revert partial code in the change set and that is not easy as there were no logical steps during the code check-ins. |
| Creating & switching branches and merging is fast | Creating & switching branches and merging is slow |

In GIT we have two kinds of merging fast-forward merge and 3-way merge.

**Fast-Forward Merge:** merge the histories with linear path from current branch tip to the target branch tip.

3-way Merge: merge two histories

git log : Show commit logs

git shortlog: Each commit will be grouped by author and title.

git show:

**Powershell commands**

**Commands to connect to azure account and to know the installed modules**

* Login-AzAccount or Connect-AzAccount
* Get-AzSubscription
* Get-AzSubscription -TenantId "xxxx-xxxx-xxxxxxxx"
* Select-AzSubscription –SubscriptionID “SubscriptonID”
* Get-InstalledModule -Name Az -AllVersions | select Name,Version

**Commands to create a new vm with required resources**

* New-AzResourceGroup
* New-AzVirtualNetwork
* New-AzVirtualNetworkSubnetConfig
* New-AzPublicIpAddress
* New-AzNetworkSecurityRuleConfig
* New-AzNetworkSecurityGroup
* New-AzNetworkInterface
* New-AzVMConfig
* New-AzVM

**Commands to get azure resource groups with ids and remove them**

* Get-AzResourceGroup –Name
* Get-AzResourceGroup | Where ResourceGroupName –like ags\*
* Get-AzResourceGroup |Sort Location,ResourceGroupName |Format-Table -GroupBy Location,ResourceGroupName,ProvisioningState,Tags
* Get-AzResource -ResourceGroupName "agsRG"
* Get-AzResource –ResourceType "microsoft.web/sites" –ResourceGroupName "agsRG"
* Remove-AzResourceGroup -Name

**Moving Resources from One Resource Group to Another**

Step 1: Retrieve existing Resource

$Resource = Get-AzResource –ResourceType "Microsoft.ClassicCompute/storageAccounts" -

ResourceName "agsStorageAccount"

Step 2: Move the Resource to the New Group

Move-AzResource -ResourceId $Resource.ResourceId -DestinationResourceGroupName "ags-NewRG"

**Commands to create storage account**

* $rg = New-AzResourceGroup -name agsrg -Location eastus
* $sa = New-AzStorageAccount -ResourceGroupName agsrg -Name agssotage -Location 'East US' -SkuName Standard\_LRS -Kind StorageV2 -AccessTier Hot -EnableHttpsTrafficOnly $true
* $sa | Format-List
* $sa.Context
* New-AzStorageContainer -Name agscontainer -Context $sa.Context
* Set-AzStorageBlobContent -File .\test.ps1 -Blob testpsfile -Context $sa.Context -Container agslabcontainer

**Storage account**

**Key Vault**

* Azure Key Vault is a cloud service used to manage keys, secrets, and certificates.
* Key Vault eliminates the need for developers to store security information in their code.
* It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked.

**Load Balancer**

* Azure load balancer is a layer 4 load balancer that distributes incoming traffic among healthy virtual machine instances.
* Load balancers uses a hash-based distribution algorithm.
* It uses a 5-tuple (source IP, source port, destination IP, destination port, protocol type) hash to map traffic to available servers.
* Load balancers can either be internet-facing where it is accessible via public IP addresses, or internal where it is only accessible from a virtual network
* Azure load balancers also support Network Address Translation (NAT) to route traffic between public and private IP addresses.

**AKS (Azure Kubernetes Service)**

AKS is the quickest way to use k8s service in Azure.

It manages your hosted k8s environment making it quick and easy to deploy and manage containerised application without container orchestration expertise.

It also helps in maintaining auto patching, auto scaling, auto updates.

It helps to create resources and infrastructure inside AKS**.**

**WEB JOBS**

WebJobs are a feature of Azure App Service that can run a script at a specific time. In my case, I would like to hit a specific URL of my website at the same time every day.

**My power shell script just loads the URL specified.**

$progressPreference = "silentlyContinue";

$result = Invoke-WebRequest -Uri ("https://www.google.com") -Method Get -UseBasicParsing;

**APP Services:**

* Azure App Service is an HTTP-based service that enables you to build and host many types of web-based solutions without managing infrastructure.
* App service --- > website

App service Plan --- > server

Scale Up: When your application or service requires a more powerful CPU or more memory to run faster.

Scale out: When you need additional virtual machines to speed up your application.

Scale in: When you're using excess capacity that you don't need.

Deployment slots are live apps with their own hostnames. App content and configurations elements can be swapped between two deployment slots, including the production slot.

How to link Azure storage to your App Service application?

Azure Storage:

* A single Azure subscription can host up to 200 storage accounts, each of which can hold 500 TB of data.
* Azure storage includes four types of data:
  + **Blobs**: object (images or documents, files, videos and audio, backup data, on-prem data) storage for text and binary data (unstructured data)

Azure Storage supports three kinds of blobs:

**Block blobs: hold** media files or image files for websites upto 5TB

**Page blobs:** hold random-access (backing storage) files up to 8 TB for VHDs for VMs

**Append blobs: hold logging information** for an application running on multiple VMs

* + **Files**: Managed file shares for cloud or on-premises deployments.
    - Accesses files using SMB protocol
    - Store shared configuration files for VMs, tools, or utilities
    - Log files such as diagnostics, metrics, and crash dumps.
    - Shared data between on-premises applications and Azure VMs to allow migration of apps
* **Queues**: A messaging store for reliable messaging between application components.
  + - queues are used to loosely connect different parts of your application together
  + **Tables**: A NoSQL store for schemaless storage of structured data. This service has been replaced by Azure Cosmos DB
  + Azure Storage provides three distinct account options with different pricing and features supported.

**General-purpose v2 (GPv2):** support all of the latest features

**General-purpose v1 (GPv1):** cool storage and archive storage are not supported

**Blob storage accounts:** support all the same block blob features as GPv2,

**Application Insights telemetry data model**

<https://docs.microsoft.com/en-us/azure/azure-monitor/app/data-model>

**Azure CLI commands**

<https://docs.microsoft.com/en-us/cli/azure/reference-index?view=azure-cli-latest>

* az login
* az login --service-principal -u http://azure-cli-2016-08-05-14-31-15 -p ~/mycertfile.pem --tenant contoso.onmicrosoft.com
* az account list –all
* az account set -–subscription (Set a subscription to be the current active subscription)
* az account show –subscription
* az group list –subscription
* az group create -l westus -n MyResourceGroup
* az group deployment create -g MyResourceGroup --template-uri https://myresource/azuredeploy.json --parameters @myparameters.json
* az acr
* az ad
* az aks
* az appconfig
* az appservice
* az backup
* az configure
* az deployment ( manage ARM deployments at subscription scope)
* az disk
* az identity (Managed Service Identities)
* az image
* az keyvault
* az network
* az resource
* az role
* az storage
* az tag
* az vm
* az vmss
* az webapp

**Daily CLI commands**

* az login
* az account list –-all
* az account set –-subscription <id>
* az group list –-s <id>
* az group delete --name <RGNAME>
* az group create --location <southindia> --name <agsRG>
* az acr create -n <agsacr> -g <agsRG> --sku <Standard> --admin-enabled true

**Azure Active Directory**

**Domain Services Overview**

**Active Directory Domain Services (ADDS)**

Legacy ADS since 2000

Kerberos &LDAP functionality

**Azure Active Directory (AAD)**

Service built for cloud

Can sync with on-premises

**Azure Active Directory Domain Services(AADDS)**

Managed domain services

Maintained by Microsoft

Domain Join,Group policy,LDAP,Kerberos,NTLM all supported.

**Azure AD features**:

**Enterprise Identity solution:** create a single identity for users and keep them in sync across the enterprise.

**Single sign-on:** Provide single sign on access to applications and infrastructure services

**Multifactor Authentication**: enhance security with additional factors of authentication

**Self service:** Empower your users to complete password resets themselves as well as request access to specific apps and services.

organisation

|  |
| --- |
| AD Tenant |

**Application Registartion in Azure AD to provide secure sign-in and authorization**

**Azure Active Directory**

1. **App registrations > New registration**.
2. **Register an application Name**
3. **Supported account types**

**Accounts in this organization**

**Accounts in any organization**

**Accounts in any organization and personal Microsoft accounts**

1. **Enter the redirect URL for application and register**

Azure AD assigns a unique application (client) ID to your app.

You can add additional capabilities like branding, certificates and secrets, API permissions

Add credentials to your web application

From the app's **Overview** page, select the **Certificates & secrets** section to add certificates and secretes.

Add permissions to access web APIs

**API permissions Microsoft APIs APIs my organization uses My APIs**

**Azure Resource Manager Vs classic deployment models**

Azure devOps scenarios

1. Change build pipeline behavior based on branch

I have a build pipeline on Azure DevOps that creates a package when the build runs successfully.I would like to change behavior if the build is run from another git branch, for example when we run it from master build a release package but when we run it from a dev or feature branch release a prerelease package. I think I will need to use variables and some condition checks?

You can check the [predefined](https://docs.microsoft.com/en-us/azure/devops/pipelines/build/variables?view=azure-devops&tabs=yaml) variable Build.SourceBranch and use a [custom condition](https://docs.microsoft.com/en-us/azure/devops/pipelines/process/conditions?view=azure-devops&tabs=yaml) in the package task:

steps:

- task: Create a release package

condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/heads/master'))

- task: Create a pre-release package

condition: and(succeeded(), eq(variables['Build.SourceBranch'], 'refs/heads/dev'))

Also add a ps step to set the prerelease variable using "##vso[task.setvariable variable=testvar;]testvalue"

|  |  |  |  |
| --- | --- | --- | --- |
| **2159** | **2159** | **Kiran Kumar V** | **Kiran Kumar V** |
| **Date** | **Topics / Tasks Assigned** | **Timeline to complete the Topics /  Task (Date)** | **Status of the Tasks**  **(Completed / In Progress / Pending)** |
| 06/11/2019 | Preparing for the DevOps Written Assesment | 06/11/2019 | In progress |
| 06/11/2019 | Started learning the web application deployment using AKS( Azure Kubernetes Service) | 07/11/2019 | In progress |

**ARM TEMPLATES**