**Q1 - SCENARIO**

A car rental company called FastCarz has a .net Web Application and Web API which are recently migrated from on-premise system to Azure cloud using Azure Web App Service

and Web API Service. The on-premises system had 3 environments Dev, QA and Prod.

The code repository was maintained in TFS and moved to Azure GIT now. The TFS has daily builds which triggers every night which build the solution and copy the build package to drop folder. Deployments were done to the respective environment manually. The customer is planning to setup Azure DevOps service for below requirements:

1. ***The build should trigger as soon as anyone in the dev team checks in code to master branch.***

*Manoj: Create build definition and enable continuous integration trigger against master branch to compile and check if the code on the server side integrates without errors along with delta from dev team*

1. ***There will be test projects which will create and maintained in the solution along the Web and API. The trigger should build all the 3 projects - Web, API and test. The build should not be successful if any test fails.***

*Manoj: Create build definition and enable continuous integration trigger against the targeted branch of code where the build contains tasks to restore nugget packages, compile code, run tests, create packages and drop it to shared path.*

*Disable the option* Continue on Error *under control option of Run test task (Visual Studio Test) to fail the build if test fails.*

1. ***The deployment of code and artifacts should be automated to Dev environment.***

*Manoj: Create a Release Pipeline linked to the (Dev) Build Pipeline where the release will pick the artifacts from and deploy to Dev environment, to have automated deployment configure the release pipeline with below properties:*

1. *Set* Continuous Deployment trigger *enabled in the artifacts configuration.*
2. *In the* Pre-deployment conditions *of the Dev Stage, select the trigger to be* After Release *with* Pre-Deployment Approvals *option* disabled
3. ***Upon successful deployment to the Dev environment, deployment should be easily promoted to QA and Prod through automated process.***

*Manoj: Extend the release pipeline from Dev to QA and Prod, configure the stages to have the necessary task to perform at individual stages, then configure the* Pre-deployment conditions *of the QA and Prod Stage as below:*

1. Pre-deployment conditions *of the QA, Select the trigger type to be* After Stage. *In the* Stages *drop down select* Dev
2. Pre-deployment conditions *of the Prod, Select the trigger type to be* After Stage. *In the* Stages *drop down select* QA
3. ***The deployments to QA and Prod should be enabled with Approvals from approvers only.***

*Manoj: In Release pipeline from Dev to QA and Prod to configure the deployments based on approvals configure* Pre-deployment conditions *of the QA and Prod Stage as below:*

1. Pre-deployment conditions *of the QA, Select the below*
   * *Trigger type to be* After Stage. *In the* Stages *drop down select* Dev
   * *Enable* Pre-deployment Approvals *option*

*Search and add the required approvers, optionally select the Timeout configuration if required.*

1. Pre-deployment conditions *of the Prod, Select the below*
   * *Trigger type to be* After Stage. *In the* Stages *drop down select* Prod
   * *Enable* Pre-deployment Approvals *option*

*Search and add the required approvers, optionally select the Timeout configuration if required.*

Explain how each of the above the requirements will be met using Azure DevOps configuration.

Explain the steps with configuration details.

**Q2 - SCENARIO**

Macro Life, a healthcare company has recently setup the entire Network and Infrastructure on Azure.

The infrastructure has different components such as Virtual N/W, Subnets, NIC, IPs, NSG etc.

The IT team currently has developed PowerShell scripts to deploy each component where all the properties of each resource is set using PowerShell commands.

The business has realized that the PowerShell scripts are growing over period of time and difficult to handover when new admin onboards in the IT.

The IT team has now decided to move to ARM based deployment of all resources to Azure.

All the passwords are stored in a Azure Service known as key Vault. The deployments needs to be automated using Azure DevOps using IaC(Infrastructure as Code).

1. ***What are different artifacts you need to create - name of the artifacts and its purpose***

*Manoj: Below are the artifacts need to be created:*

* *Resource Group: for deploying all the other resources*
* *Key Vault: to keep/save key and access it securely*
* *Virtual Network: To have defined IP address range and to avoid overlapping, also to establish the VM connectivity over internet*
* *Virtual Machine: To deploy and run the application*
* *Azure DevOps Project: To version control ARM template and to create deployment pipeline to enable auto deployment.*

1. ***List the tools you will use to create and store the ARM templates.***

*Manoj: Visual Studio 2015 and above or VS Code and Azure DevOps Project (Git repo)*

1. ***Explain the process and steps to create automated deployment pipeline.***

*Manoj: With the template checked-in into Git repo below steps are followed:*

* *Add Service connections to Azure Subscription*
* *Then setup pipeline with repository type Git and add ARM template deployment task and provide all the necessary details like:*
* *Name to the task: Display name*
* *Under Azure Details section:*

*Azure Resource Manager Connection: Select the value from the drop down which was added in service connection window*

*Action: Create and Update*

*Resource Group: Selected the Resource group name*

*Location: Location name from drop down (Ex: North Europe)*

* *Provide ARM template details*

*Template location: select either linked artifact or URL to the file option*

*Template: path to ARM template*

*Template Parameter: path to Parameter file associated with ARM template*

*Template Override parameters: optionally one can provide values to override parameter values (one can use variable section to pass secure values like password)*

*Deployment mode: from the drop down select either Incremental, Completed or Validation only. Select Incremental.*

*The above Pipeline can be now triggered for deployment.*

1. *Create a sample ARM template you will use to deploy a Windows VM of any size*

*Manoj: I have considered the Azure environment where a subscription already exist with one resource group and Key Vault (with one key), I will be creating Network interface with default subnet (created after Virtual network creation), Network security group (NSG), Public IP address and Virtual Machine (size: Standard\_B2s, OSdisktype: Standard\_LRS, Image: MicrosoftWindowsServer:WindowsServer:2016-Datacenter:Latest, Admin User: devops, Admin Password: will be fetch from Azure Key Vault) using my ARM template*

*I will also use Parameter file to pass few parameters.*

* *

1. *Explain how will you access the password stored in Key Vault and use it as Admin Password in the VM ARM template.*

*Manoj: I have create a new Secrets within in my key-vault with Template Deployment option enabled. Then in the ARM template I create the below:*

1. *I will create a parameter type secureString (ex: adminPassword) and I will get the value in the parameter file using the below code snippet:*

"adminPassword": {

            "reference": {

                "keyVault": {

                "id": "/subscriptions/5e034370-f0d9-4d4f-b10e-007b8c4ebba5/resourceGroups/DevOpsTest/providers/Microsoft.KeyVault/vaults/cnm-key"

                },

                "secretName": "VM-Password"

            }

        }

**Q3 - SCENARIO**

A Toy Retail company ToyTrex has it retail application deployed as 3-tier application - Web App(UI), Web API(middle layer) and Database as Azure SQL.

The user load started increasing multiple fold every month and complex programs getting implemented, the application started performing poorly.

As a result, company decided to re-architect the middle layer as microservices using Azure Kubernetes Services.

The new architecture has below design decisions.

Manoj: I have not worked with Docker so intensively I have basic knowledge about it. I will answer the below with best of my knowledge.

1. *The middle layer should be implemented as Microservices using Azure AKS*

*Manoj: It is a complete re-architecting the middle and the DB if we have call it complete Microsservice based architecture. If the idea is to only re-define/break middle layer into independent functions and containerize to deploy. Firstly we will have to start with loosely couple functionalities and convert into API and make necessary changes in UI layer and deploy, likewise one will have break the monolith into microservices.*

1. *Break the middle layer into multiple projects and add docker support (add docker file) for all the projects. (right click on projects and Add Container orchestration support)*
2. *Modify the docker file in the project to use required image, set working directory and perform copy operation to move files into docker image (work dir)*
3. *The middle layer API should be deployed as containerized application images*
4. *The container images will use Azure Container Repository (ACR) as the private image repository*
5. *The CI/CD pipelines for microservices should be implemented using Azure DevOps services.*
6. *The Azure DevOps should be able to access ACR and download the container images for microservices deployment*
7. *The image should be deployed as templates such as <image\_name>:<build\_id>*

*Explain the DevOps configuration and steps in detail for above requirements*

*Manoj:*

* + - * *First create ACR service in Azure.*

Setting up connection b/w Azure DevOps and ACR

* + - * *If we are using Azure DevOps as repository for the project, one can add the service connection to Azure Container Registry from the project setting under service connection.*

Creating a build pipeline to build and push image to ACR

* + - * *Create a pipeline in Azure DevOps using Docker (build and push) task to build image and push it to ACR on Azure. In the build task select the ACR service connection and give a name to repository. Mention the path to docker file and set the build context.*

*Define the image name in the format like* ImageName:$(Build.BuildId) *$(Build.BuildId) is a system variable which will be replace by build number.*

* + - * *At the end of build we will see an image built and pushed to ACR repo mention in the task.*

Creating a release pipeline to download images from ACR and deploy it to Azure Container instance (ACI)

* + - * *In the Release pipeline select the Artifact source as Azure Container Registry, fill all the fields like Service connection, Resource Group, ACR, Repository Name, Version and click Add*
      * *Then using the Azure CLI task with type PowerShell and script location: Inline to create Azure Container Instance using the image from the ACR, also we will use variable section to pass values to inline script. (Variable are marked as <var.name>)*

az container create –resource-group $(<var.resourcegroupname>) –name $(<var.aciname>) –location $(<var.acilocation) –image $(<var.acrLoginServer>)/ImageName:latest –os-type windows –cpu 1 –memory 1 –registry-login-server $(<var.acrLoginServer>) –registry-username $(<var.acrName>) –registry-password $(<var.acrPassword>) –dns-name-label labelName –ports 80 –protocol TCP