**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.*

*Solution: In the Build Pipeline Edit menu:*

*Step1: Click on Triggers option.*

*Step2. Enable Continuous Integration, also select branch filter as master branch.*

*2) 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.*

*Solution: In the Build Pipeline Edit Menu:*

*Step1: Click on Triggers option.*

*Step2: If the Web and API Build and Test are configured in same pipeline in Test task click on the option “Fail the task if minimum number of tests are not run”.*

*Step3: If the Web, API Build and tests are run in separate pipeline then in the trigger select the trigger in last two pipeline as Build Completion and put the name of the Build Pipeline.*

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

*Solution:*

*Step1: In the Release pipeline select the Artifact feed either directly from the Build pipeline or from the Azure Universal Artifacts, enable continuous deployment trigger and set the Pre Deployment condition to After Release.*

1. *Upon successful deployment to the Dev environment, deployment should be easily promoted to QA and Prod through automated process.*

*Solution: Enable continuous deployment trigger and set the Pre Deployment condition to After Release.*

1. *The deployments to QA and Prod should be enabled with Approvals from approvers only.*

*Solution: In the edit release pipeline:*

*Step1: Select the Pre Deployment conditions.*

*Step2: Enable Pre Deployment Approvals and add the USER which has to be made the approver.*

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*

*2) List the tools you will to create and store the ARM templates.*

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

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

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

**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.

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

*2) The middle layer API should be deployed as containerized application images*

*3) The container images will use Azure Container Repository (ACR) as the private image repository*

*4) The CI/CD pipelines for microservices should be implemented using Azure DevOps services.*

*5) The Azure DevOps should be able to access ACR and download the container images for microservices deployment*

*6) The image should be deployed as templates such as <image\_name>:<build\_id>*

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