**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.*

*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.*

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

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

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

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

Explain the steps with configuration details.

Solution:  
1. We need to checkout the repository with master branch in jenkins job and set the build job trigger to "poll csm" so every time a developer pushes changes to master branch job will automatically triggered.

2.We schedule web build first to run after which it is successful , it should trigger API job in dwostream and after API job is successful it should trigger test job and finally we can send notification that all builds passed. If any one build fails it won't trigger other builsd and then send notificatio that paticular build is failed likewise.

3.Once the jenkins runs properly and create artifact we can trigger deployment job to deploy the previously generted artifact on dev environment

4.If the deployment CD job is successful we can mark that build as promote build by using promoted build plugin in available in CI server and send notification to QA and PRod to refer that build as stable build and start testing and deployment to prod.

5.The deployment CD pipeline should be triggered with an option called Manual Approval which will allow users to take decision whether this build should go into QA or Prod enviroment or not

**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.*

Solution:

1. We need to created artifact for every resource we have in our infrastructure. For example: Virtual N/W which will give a separate network with full access. Subnets which will logically divide our network to put different layers of app like web, API, db etc. NSG which will provide security rules to access machines in subnets etc.

2. We can store ARM templates i storage service provided by Azure

3. Once our infra is created on Azure cloud we can simply launch our CI and Cd pipeline in the Azure DevOPs service and build our code then deploy using CD pipelines.

4.We need to create resource group first and then the vm using azure ARM template. In the template we need provide schema which will have pathto json file and then vm related data like username, subnets, storage etc.

5.We need to create a role to access key from vault service to APM template service and also need to specify which resouce is using this

**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*

Solution:

1. We need to write dockerfile for the API middle layer and for every build the docker image should be created with the latest changes in API code. Once this image is created we cn deploy containers created from latest image with all changes in API and can do deployment on Azure AKS.

2.All the images created in continuos integration job sholud be tagged with build number and commit-id and should be stored in private image repositoy.

3.The CI pipeline will be responsible for building docker image for evry commit in API middle layer whereas the CD pipeline of Azure DevOPs service will be responsible for deploying conatiners created from newly created docker images from CI pipeline.

4. We nee provide authentication in Azure DevOps service so that it can push docker image once image is built in CI and pull from private image repository in CD pipeline for deploying to some snvironment.

5. We should tag the docker images in every build as <image\_name>:<build\_id> where build\_id will be uniuqe every time and this way using same variables we can just access or pull image from private image repository in CD pipeline and finally can deploy the microservice container in enviroment.