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.

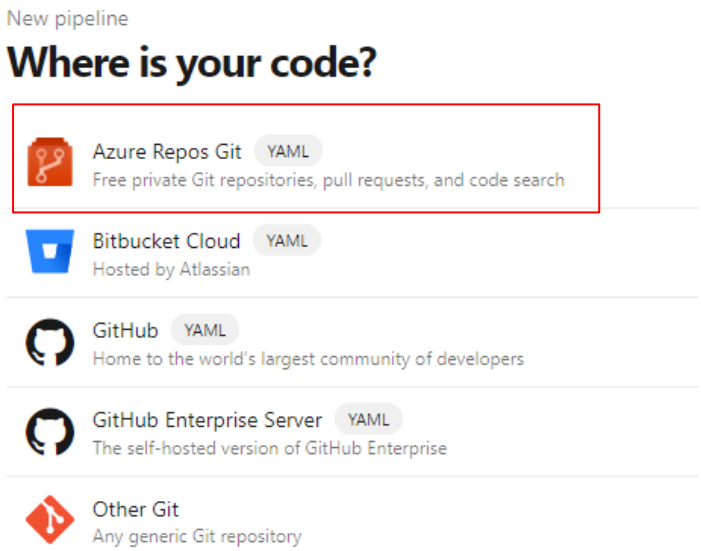
Explain how each of the above the requirements will be met using Azure DevOps configuration. Explain the steps with configuration details

A1 -Scenario

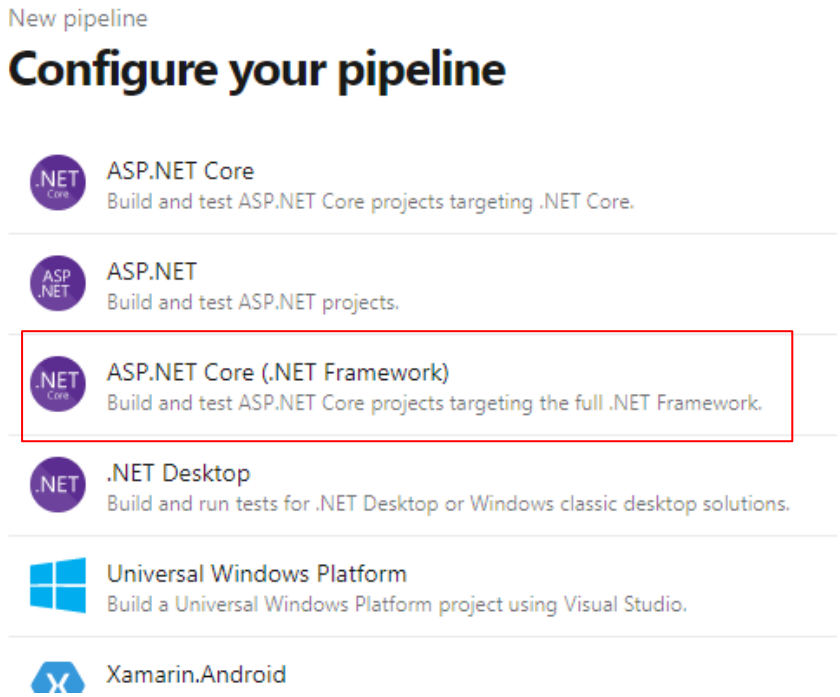
For the car rental company 'FastCarz' has code repository migrated to Azure repo git.

Pipeline setup:

Step 1: Starting the pipeline we setup repository for our pipeline, which is Azure Git in our case. To include the same we can navigate to Pipelines → New Pipeline →select repo to **Azure repo git** and select your repo



Step 2: Configure your build pipeline ASP.NET core (.NET Framework) from the pipeline suggestions. Pipeline can also be configured with Starter pipeline option and tasks can be added for .NET framework.



Step 3: Once pipeline is created; it will open a yaml file to customize our build and deployment. Content of yaml file is given below customized as per our pipeline requirements.

```yaml
trigger:
- master

pool:
  vmImage: 'windows-latest'

variables:
  solution: '**/*.sln'
  buildPlatform: 'Any CPU'
  buildConfiguration: 'Release'

steps:
- task: NuGetToolInstaller@1

- task: NuGetCommand@2
  inputs:
    restoreSolution: '$(solution)'

- task: VSBuild@1
  inputs:
    solution: '$(solution)'
```

Trigger on every check-in on master branch

Run build based on pipeline variable defined above

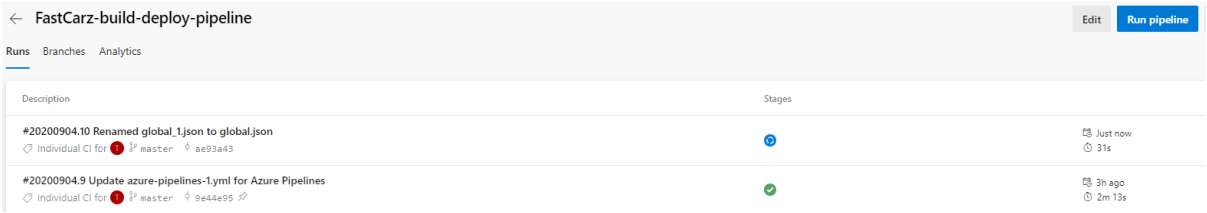
```
    msbuildArgs: '/p:DeployOnBuild=true /p:WebPublishMethod=Package /p:Package
AsSingleFile=true /p:SkipInvalidConfigurations=true /p:DesktopBuildPackageLoca
tion="$(build.artifactStagingDirectory)\WebApp.zip" /p:DeployIisAppPath="Defau
lt Web Site"'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'

- task: VSTest@2
  inputs:
    testSelector: 'testAssemblies'
    testAssemblyVer2: |
      **\*test*.dll
      !**\*TestAdapter.dll
      !**\obj\**
    searchFolder: '$(System.DefaultWorkingDirectory)'
    runOnlyImpactedTests: true
    runAllTestsAfterXBuilds: '10'
    testRunTitle: 'Unit tests'
    platform: '$(buildPlatform)'
    configuration: '$(buildConfiguration)'
    publishRunAttachments: false
    failOnMinTestsNotRun: true
    rerunFailedTests: false

- task: CopyFiles@2
  inputs:
    SourceFolder: '$(System.DefaultWorkingDirectory)'
    Contents: '*.zip'
    TargetFolder: '$(Build.ArtifactStagingDirectory)'

- task: PublishBuildArtifacts@1
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'drop'
    publishLocation: 'Container'
```

Run Tests

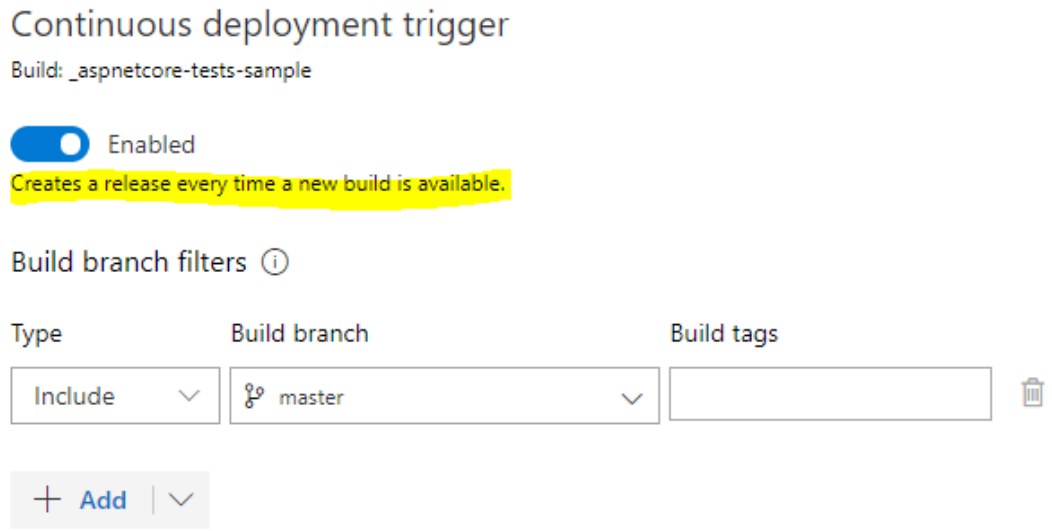
Copy generated .zip (or any) package to artifacts staging area

Publish artifacts to deployment pipeline

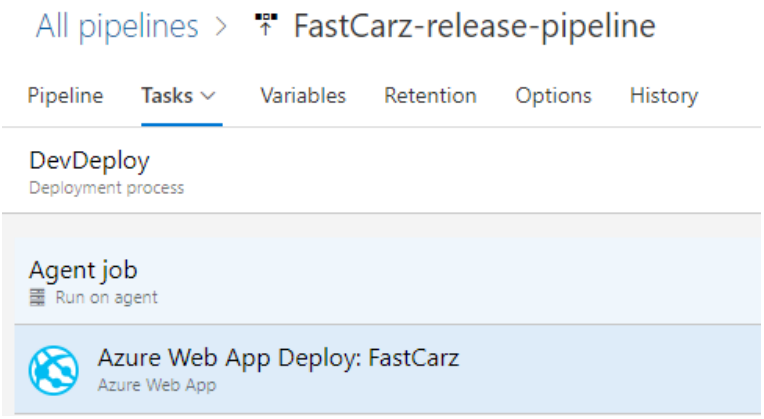
Step 4: Save changes, which will commit yaml file to master branch in azure repo. And the build will get triggered automatically as the commit will be to master branch.



Step 5. Creating Deployment pipeline. Go to release pipeline and create New Release Pipeline. Select Artifacts from the Build pipeline at first stage which are published. Here the trigger should be set so that it gets triggered for every new available build.

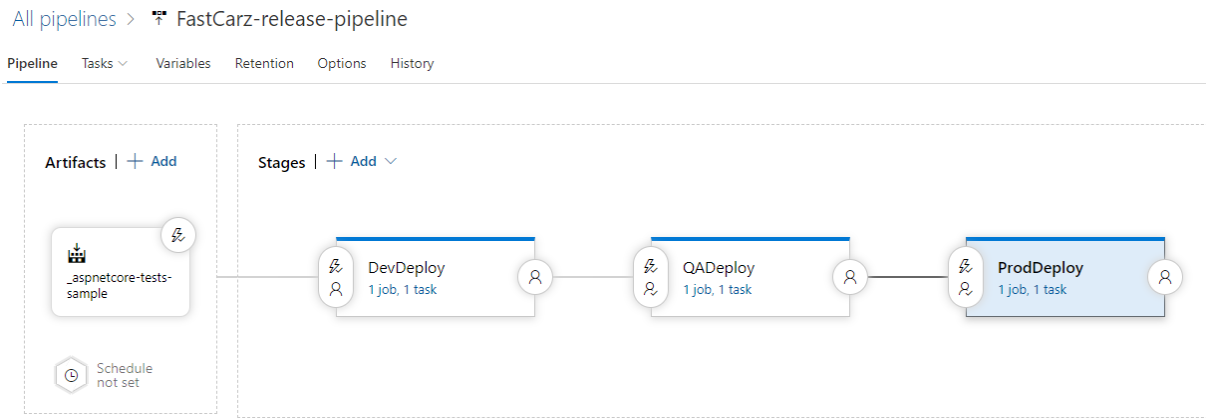


Step 6: Create first stage of Deployment: DevDeploy (rename the stage) and add deployment task to the stage. Provide subscription details and artifacts to deploy in configuring the deployment task.



Step:7: Configure/Clone same stages for QA and Prod environments.

Step 8: As additional approval setup for QA and Prod environments, click on Pre-deployment configuration icon, enable pre-deployment approval and setup approver account. Your pipeline should look like this.



When triggered the pipeline, it will automatically pull the artifacts published from build pipeline and will trigger DevDeploy stage. Once DevDeploy is completed, it will trigger QADeploy stage and wait for the approval to trigger the QA Release. Same will be applicable to ProdDeploy stage.