**Scenarion1:**

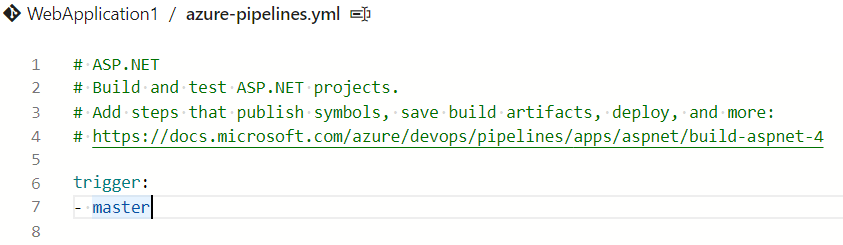
**1) The build should trigger as soon as anyone in the dev team checks in code to master branch.**

Solution:

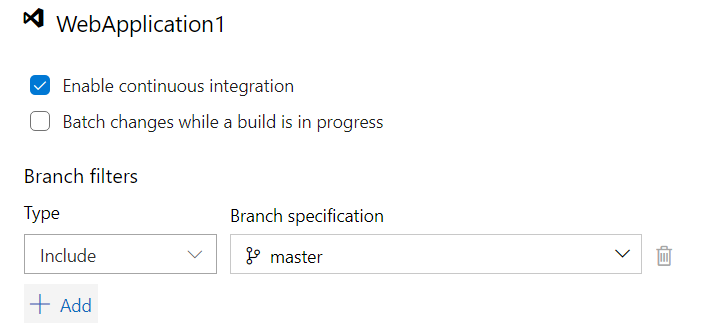
We should enable continuous integration in triggers section by check the “Enable Continuous integration” or

trigger:

- master



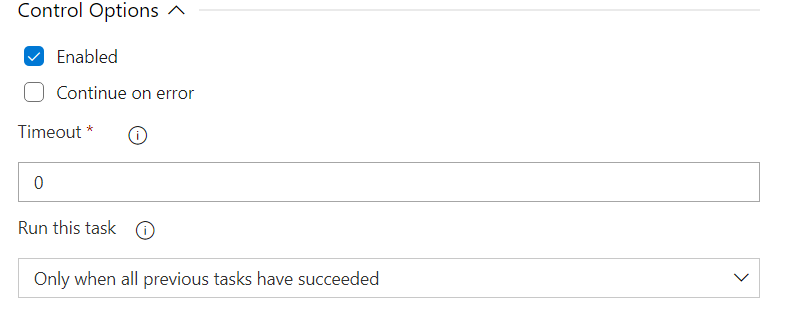
OR



**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:

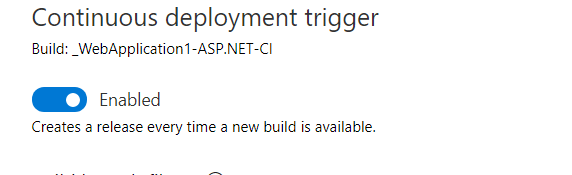
By using Condition expression, we can control the build tasks or under control options by select the task run when previous tasks are succeeded.



Maybe we can build and test in single task using PowerShell or cli commands. However, I have not done this before.

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

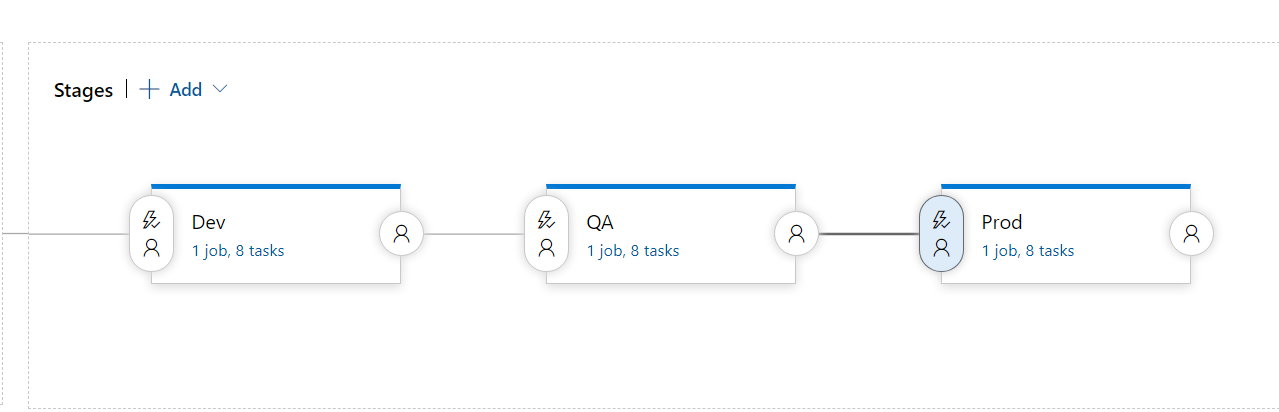
Create Release pipeline with three stages Dev, Qa and Prod. Add the Artifactory from the CI pipeline published Artifactory and Check Enable the continuous deployment trigger.



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

**Solution:**

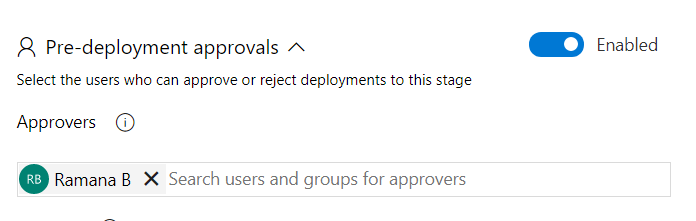
In CD pipeline we create multiple stages as Dev, QA and Prod as below



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

**Solution:**

We should enable the pre deployment approvals and should provide the approver details.



**Scenarion2:**

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

The below artifact files will be generated form CI pipeline.

**.zip** – which has the html, xml, executable files (ex: .dll, jar, war files)

ARM template or Terraform configuration files – which will deploy the Azure resources

Test results will be published to CI pipeline logs and .xml to store the code coverage & test results.

.**ps1** - files of PowerShell scripts to configure the app settings pre and post of app deployments.

1. **List the tools you will create and store the Terraform templates.**

VScode to write terraform configuration files

Azure Repos as Source code Management

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

* Create the service connections to make connectivity between Azure DevOps & Target environment.
* We will create release pipeline.
* Integrate build artifacts to release pipeline
* Create multiple stages for different environments usage.
* Enable the continuous deployment
* Configure variables for each stage.
* Configure approvals to control the deployments.

Steps to create Deployment pipeline:

1. Select “Release” under Pipelines, Click on + New and select create “new release pipeline”
2. Select one of the jobs from predefined templates.
3. Add the build artifiact refernce by giving the CI pipeline name.
4. Create the stages for new environment deployment.
5. Create varaibles and make availability to use different stage levels.
6. Terraform installer
7. Terraform init
8. Terraform plan
9. Terraform apply steps can be configured to create Azure resources
10. Use the Azure App service deploy task to deploy code.

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

1. We can create Azure Key vault and secrets manually in Azure.

2. In terraform configuration file we can use data source provider to get secrets.

3. use the same secret in VM creation configuration section.

OR

If we are using different subscription for terraform and Azure Key vault, then create variable groups in Azure DevOps library section.

Make available to the deployment pipeline and use the replace tokens for terraform configuration file.