**Azure Repository: -**

Azure Repos is a set of version control tools that you can use to manage your code. Whether your software project is large or small.

Using azure repository, we can either use git or Team foundation version control

when we create a project, default we get an empty repository. Or we can also create new repository.

Concept is same when we work with git or azure repos

From the local system you can add the changes in origin and then you can push changes to remote repository.

When we create new repository that time it creates default branch, and which is a master branch of that repository.

Let’s say that we are working on application code normally we would actually ensure that the master branch contains most recent copy of the production application code and now let’s say you have to work on feature of new feature of the application or maybe there is bug fix you have to work on instead of directly making these changes on main code that is our master branch what is a good development practice is like we have to create another branch which is feature branch. We can create multiple branches as per our feature requirement

It’s like taking a copy of our main code creating it another branch and then we can make changes on to newly created branch.so the advantages of this feature branch are you still have working copy of the main production master branch and we can compare new changes with existing changes at any point in time.

A later point in time you can merge changes of feature branch to the production branch that is your master branch.

Good practice is that

Create many short-lived feature branches

Delete the branches ones they are not required.

**Release Branch: -**

**Manage releases**

Use release branches to coordinate and stabilize changes in a release of your code. This branch is long-lived and isn't merged back into the main branch in a pull request, unlike the feature branches. Create as many release branches as you need. Keep in mind that each active release branch represents another version of the code you need to support. Lock release branches when you're ready to stop supporting a particular release.

Create a release branch from the main branch when you get close to your release or other milestone, such as the end of a sprint. Give this branch a clear name associating it with the release, for example *release/20*.

Create branches to fix bugs from the release branch and merge them back into the release branch in a pull request.

**Pull Request: -**

Suppose we have multiple developers who are working with application code they have their local repositories they have their master branch they go ahead and create multiple feature branches and you have your central repositories in azure repos and u have your master branch and then want to push the changes in master branch in the central repository by default you can do it very easily but let say you are a lead developer or senior developer to first go ahead and see what are the new changes and review the changes by the developer from the local repository on to that central repository because master branch central repository is very important because it has the production code so you want to ensure that make the right code has been pushed to the master branch.

so for this the developer can first go ahead and carry out the pull request this pull request feature is available in azure DevOps.

Here if we go to the branches with branch policy, we have different branch policies.

Branch policies are an important part of the Git workflow and enable you to:

* Isolate work in progress from the completed work in your main branch
* Guarantee changes build before they get to main branch
* Limit who can contribute to specific branches
* Enforce who can create branches and the naming guidelines for the branches
* Automatically include the right reviewers for every code change
* Enforce best practices with required code reviewers

**CICD Pipeline**

**Continuous Integration**: -

when we consider DevOps, life cycle the next step is after source control management is continues integration.

Considering the scenario, we have application has been built and we have many developers working on different module of the application we have central git/ azure repository which has main branch which has production code.

And we have each developer who has clone the code from same repository on their local machine and then they start working on feature branch or bug fixes etc. on their local machine and one of the developer needs to merge the code with central repository or main branch this can be done via pull request now in the next phase after the code has been merge on to the main branch the next step is go ahead and build the application so for building the application we have to create a deployable version of the application. This build will be deployed on web server for carrying out this build we have another person known as the build administrator.so this build administration take the code from the git or azure repository after the merge has been done the code from central repository and then build that particular source code either on their local machine or another server so for that they could use separate build machine as well. Once build has been done all test has been done that build software can be deployed on production environment.

So, the disadvantage of this build administrator is each time developer has to raise pull request and then build administrator will build the code and then deployed app in production etc.

So, when comes to continues integration this is not the ideal approach because in today’s time, we must look at the automation so for that we have built software in place.

This build software is actually taking the source code from main branch once a commit is in place.

And build the code automatically and then using another software we can publish build to the target machine.so the advantage of having built software is it helps to automate the entire build process.

So here you can automatically pick up the code and perform the build. The biggest advantage is that we can detect bugs in the development life cycle.

So, this is entire process of continuous integration.

Here you have the ability to ensure that as soon as the changes has made on to the git repository a build is automatically trigger and the build can be check is the pass or fail from the build process, so this is one of the stages in DevOps which is continuous integration.

**ARM Template:**

Infrastructure code is stored as part of repository and version control anyone in your team can run the code and setup similar environment. One way is to implement infrastructure as code in azure is to use azure resource manager template also called as arm template.

You can use code to automatically deploy these infrastructure so that you don’t have to login azure portal to manually create these resources.

**Benefits of Azure Resource Manager**

* Enables you to manage your infrastructure through declarative templates rather than scripts
* You can deploy, manage, and monitor all the resources for your solution as a group, rather than handling resources on individual basis
* Can help you define the dependencies between resources, so they are deployed in the correct order

So lets see how we can create arm template from visual studio.

I have visual studio in their we can create one arm template for web app.

Create new project their you can choose azure resource group as a project select it and create project. Here there is option to choose resource so now I want web app so I am selecting web app. You can get code automatically.

It has deploy.json file and also parameter.json file.

There is poweshell file to deploy template we wont using this powershell script we are using these two files in azure devops.

Azuredeploy.json file is where the resource is defined and the parameter.json file is the parameter that you want to supply for that particular file.

Suppose for your application you might want to name it differently for different environment in this case I want to name it armtemplate-dev for dev and for test armteplare-test etc

I can start creating resources from here

If I need to create new resource I can start typing in arm and typing the appropriate resources.

Like web app or whatever you want to create

Selecting that would flush out all the details there it requires so this is another way you can create an arm template so reset this back and go to resources write click on resources there is option add new resources so let me click it first choose azure web app where you want to deploy this web app.give appbase web app name

We need to create app service base plan. In background app service code get added.

So we have parameter appBaseName and also skuName and it has variable has created appBase Name is exactly same as parameter and overriding it with unique string this is automatically done so that there is no name conflix when we create new resources.

Under resources you can see that its adding appBase name from the parameter so that’s using the parameter function and getting the name from there

Its creating the server form using the app base sku name whatever you pass in as a parameter.

Also this has default value of F1 so even if you don’t specify 1 it should be fine it by default choose free tier

So now we have created arm template for web app and service plan now if we go back to parameter.json file this file is use to passing parameter to this particular json(main.json) file

You can also pass in parameter using inline arguments when invoking this template however keeping it as a parameter file keeps it nice and clean. We can use azure devops to populate parameter values from variables in azure devops. You could manually type parameter here or you could use automatically generated parameter values from here.

Once this has been done then we have to push code in azure repo and create pipeline to deploy resources. In azure we have to just mention json file path or publish json file in artifacts so we can copy for deployment. This is in brief idea about arm template.

**Azure Artifacts:**

Add fully integrated package management to your continuous integration/continuous delivery (CI/CD) pipelines with a single click. Create and share Maven, npm, NuGet and Python package feeds from public and private sources with teams of any size.

Azure Artifacts enable developers to consume and publish different types of packages to Artifacts feeds and public registries such as NuGet,org. You can use Azure Artifacts in conjunction with Azure Pipelines to deploy packages, publish build artifacts, or integrate files between your pipeline stages to build, test, or deploy your application.

Azure Artifacts is free for every organization up to 2 GiB of storage. Once you reach the maximum storage limit, you can no longer upload new artifacts and will need to either delete some of your existing artifacts, or [set up billing](https://docs.microsoft.com/en-in/azure/devops/organizations/billing/set-up-billing-for-your-organization-vs?view=azure-devops) to increase your storage limit.

**Azure agent machine:**

Remember we build our source code in our local machine like in visual studio has the ability to build our code into a format that can be run on our local machine. In order to run MS build tool which is part of visual studio to build our code is still needs to run somewhere so code needs to be taken from git repository it needs to be built on a machine so that build is in place.

So when comes to azure pipeline and azure cloud the way this is done is that you tell azure pipelines yes I want build dotnet code which is part of git repository when azure pipeline needs to build your code which is hosted in git repo

In order to provide that compute environment for building that application it will temporary build new virtual machine on cloud in azure pipeline itself you won’t see this virtual machine it will be running in background has part of azure pipelines and then it will actually be carried out that build process. So, all of the code will be automatically taken from the git repository on this virtual machine and then using build tools the code will actually built on this virtual machine and when’s the build is available once is success or failure this virtual machine is disposed of, so this is intimate environment which is use for building your application. this particular virtual machine actually known has an agent you could have an agent which is actually use to build your application this agent runs all the jobs, the steps scripts and the task that you actually specify as part of azure pipeline.

And when it comes for type of virtual machine which is being spun up, so there are different images available. So in your pipeline you can specify what will be image for application build.

For example, if we are building .net application as part of azure pipeline so we can use window images which you have in place in that already visual studio installed so using visual studio we can build our .net application. likewise, it also has other agents in placed it has one base on mac OS and another based on ubuntu

Apart from that we could also create our own custom agent as well. In simple term Azure pipeline will create an environment which is basically simulation of our local desktop environment for building the application.

**Variable Group: There** is something known as library u can see there is variable group. Click on variable group then give name of variable group.

If you scroll down, we can add variables even after saving the variable group.

Let me add variable **secret** and give the value **appvalue if you don’t want to** show variable value to everyone then there is option -change variable type to secret click on this you can’t see variable value. I am saving this variable.

Once you have variable group in place if we go back on library you can see your variable group and you can add multiple variables.

In another tab I am going to create another pipeline to show on how to fetch variables in the variable group.

Chose azure git repo simple code on repository I just want to show you simple yaml code that can use and access secrets of the variables

Now I m replacing yaml code

In variable section we have to specify what is the variable group that you want to use in the pipeline and then I just want to show or echoing what is the value of the secrets.

Now let me run our pipeline and here you can see variable value.

Variables:

1. # Docker
2. # Build and push an image to Azure Container Registry
3. # https://docs.microsoft.com/azure/devops/pipelines/languages/docker
5. trigger:
6. - master
8. pool:
9. vmImage: 'ubuntu-latest'
11. variables:
12. - group: demogroup
13. stages:
14. - stage: demostage
15. jobs:
16. - job: Test
17. steps:
18. - script: echo $(secret)

And for Azure keyvault secrets

You can see there is option to link secrets from an azure key vault as variables

Enable this option link secretes from an azure keyvault

Then it will ask you to choose subscription

You have to choose subscription I don’t have subscription so I can’t show you

Once you choose your subscription then you can choose keyvault name as you want.

You can add keyvault secrets here after authorizing the keyvault

1. trigger:
2. - master
4. pool:
5. vmImage: 'ubuntu-latest'
7. variables:
8. - group: demogroup
9. stages:
10. - stage: demostage
11. jobs:
12. - job: Test
13. steps:
14. - task: AzureCLI@2
15. inputs:
16. azureSubscription: 'Test Environment (20c6eec9-2d80-4700-b0f6-4fde579a8783)'
17. scriptType: 'bash'
18. scriptLocation: 'inlineScript'
19. inlineScript: 'az vm create --resource-group newgrp1 --name demovm3 --image win2016datacenter --admin-username demousr --admin-password $(newpassword)'

**My Use case CI pipeline:**

**Yaml File configuration:**

**Use case .net core application**:

First thing is trigger:

any changes in master branch in our source code repository Is triggers the pipeline we can mention different branches here. Also, we can use filter as per the different folders of the branches as well

we have lot of flexibility when it comes to defining on what should trigger your pipeline

Pool-

Use image of windows latest now we are using .net core-built sort of pipeline, so it is using windows latest image to spin up a virtual machine to build our code. Because we are not building our code we need to right software in place on that virtual machine means it needs to have visual studio or at least build tools in that machine to build asp.net core base application.

Variables: -

Here we have some variables in places this variable can be use within the yaml configuration file

Steps:

Now we have set of steps in placed

First step is task to install the NuGet tool

When it comes to .net when it comes to packages all of this manages by NuGet one of the core task when it comes to building .net base application is to ensure that u restore all of the package dependency for your project if .net project depends upon another package in order to run NuGet needs to be used to get all of the packages from site nuget.org and then build your application along with the packages so that you have bundle solution that can be run. Again, depending upon type of programing languages that your application if you have java application then you need to use maven etc.

This task is nothing but inbuild task that define azure pipeline, so the first step is download or install NuGet tool in agent machine.

Next step: -

Use that NuGet tool to restore our visual studio solution.

First step – restore all packages.

And then next step is using visual studio building our entire application.

It is checking build platform solution and configuration

All these arguments are inputs of build task.