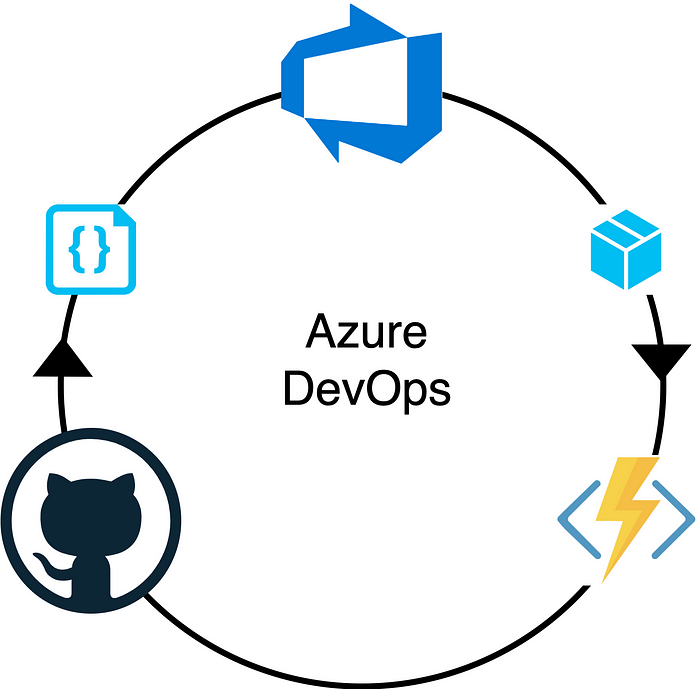
**Azure DevOps Guide**



Asimplied by the name “Dev-Ops” brings the Development and Operations together during the lifecycle of a project. As a result, it streamlines and automates the main project steps to make things reliable and faster. Main project development steps Devops can help with are,

* Building project artifacts
* Testing
* Source code scans
* Deployment

Once a Devops platform handles the above steps in a project, it will ensure there’s no human intervention is involved in any of those steps and improve the project development efficiency.

In order to understand how Devops can be implemented with [Azure Devops](https://learn.microsoft.com/en-us/azure/devops/?view=azure-devops&viewFallbackFrom=vsts) platform, we will start with creating a CI/CD pipelines for an Azure Function App.

**Prerequisites**

* Basic knowledge of Azure Function Apps.
* **An Azure subscription** — Will be used to deploy Azure resources.
* **An Azure Devops account** — Will be used to create CI/CD process.

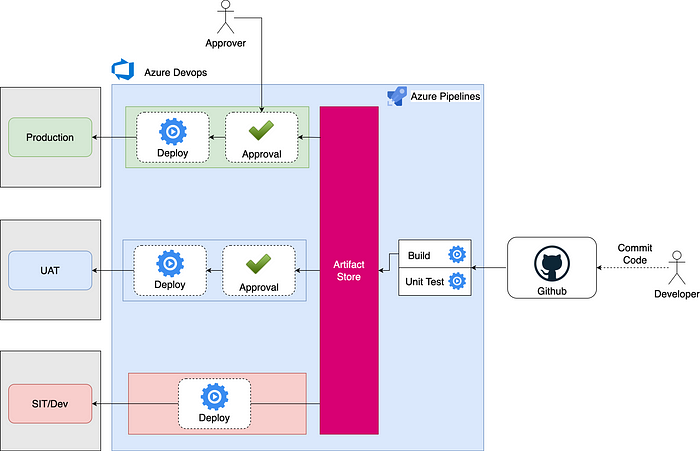
**You will learn**

* How to create an end-to-end Azure Devops pipeline for a project.

**Blog Outline**

1. Architecture
2. Project Setup
3. Create Azure Devops Project
4. Create & Configure Azure Devops Pipeline
5. Re-use Same Azure Devops Pipeline for Separate Project Branches
6. Azure Service Principal Deployment Permission
7. Assign Service Principal to Azure Devops
8. Create & Configure Azure Devops Release Pipeline
9. Test Azure Devops Pipeline
10. Self-Hosted Agents

**1. Architecture**



The above architecture provides a high-level overview of the Azure Devops flow and it’s components. This architecture will remain the same regardless of the project technology.

**Github**Github is the source control and provides integrations with Azure Devops to trigger pipelines upon code commits in a branch.

**Build Step**This represents the build scripts and commands that will be executed in the Azure Devops runtime agent to build project artifact and binaries. Azure Devops will keep a history of all the build artifacts to provide rollback capabilities.

The pipeline will halt upon any errors detected during the build step.

**Unit Tests**These needs to be implemented by the developer to make sure main functionality of the project remains in working condition at any given time.

Unit tests can be configured to be executed as part of builds steps and the pipeline will halt upon any errors detected during the unit test execution step.

**Artifact Store**Project artifact and binaries will be stored in Azure Devops built in artifact storage location.

**Approval Checks**Emails will be sent to relevant approvers before the deployment step to UAT or PRD environments. Only if approvers accept the deployment, the pipeline will proceed forward with the deployment.

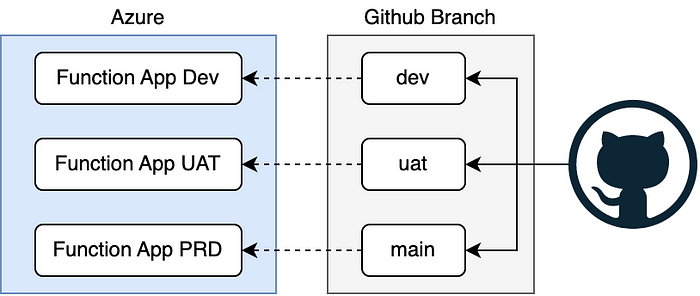
**Deployment**If there are no pending approvals, the deployment will be started into configured Azure environment utilizing the provided DevOps service principal.

**Azure DevOps Deployment Permission**[Azure service connection](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml) is used to get necessary permissions to deploy into Azure environments. A Service Principal with the required permissions to the Azure resources will be configured in Azure DevOps.

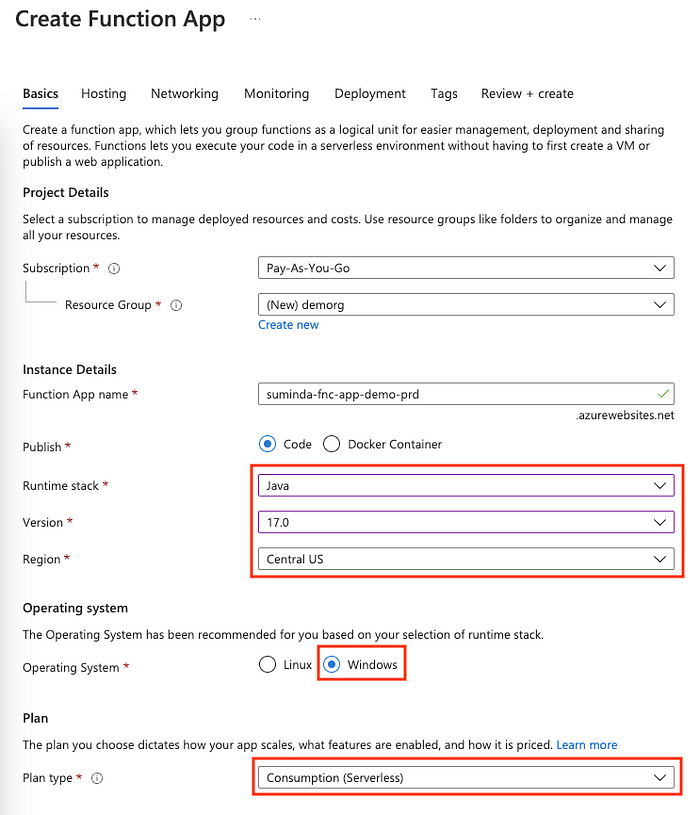
**2. Project Setup**

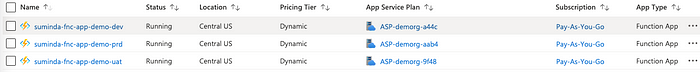
You can checkout or download the source code from [Github Repo](https://github.com/SumindaD/azure-function-app-demo" \t "_blank).

Please refer to below for relevant branch deployment target for Azure Function App in Dev, UAT and Production environment.



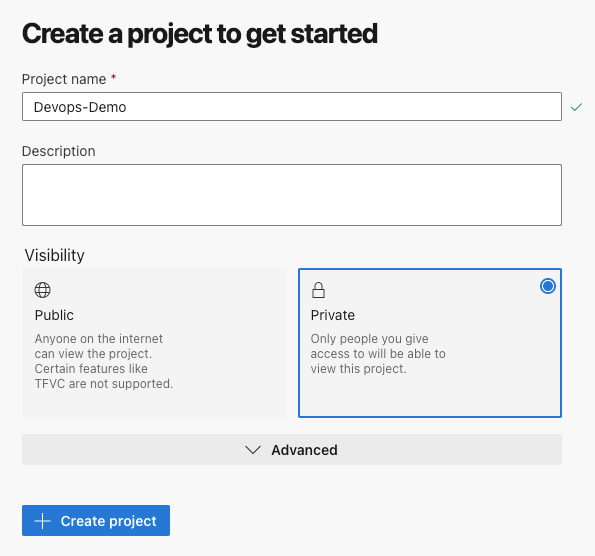
Create 3 Azure Function Apps for each environment with below configurations.





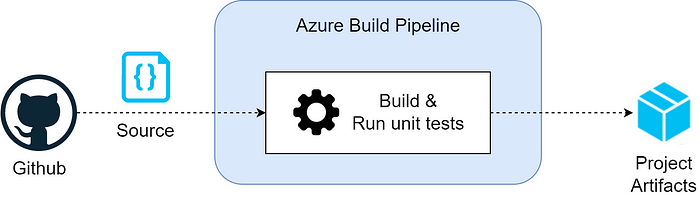
**3. Create Azure Devops Project**

Go to Azure Devops and create an Azure Devops Project as below.

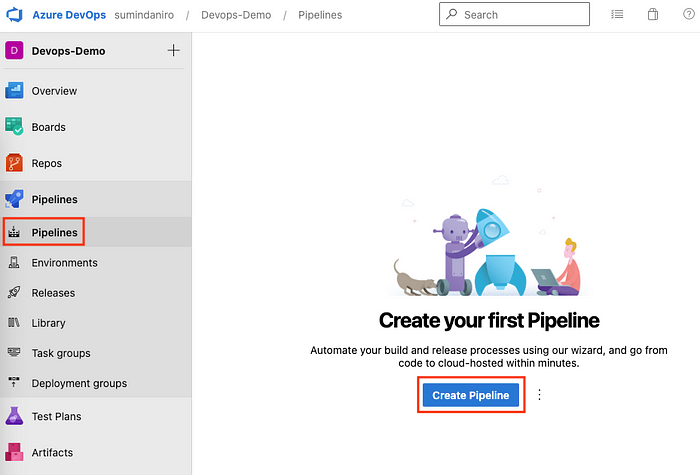


**4. Create & Configure Azure Devops Pipeline**

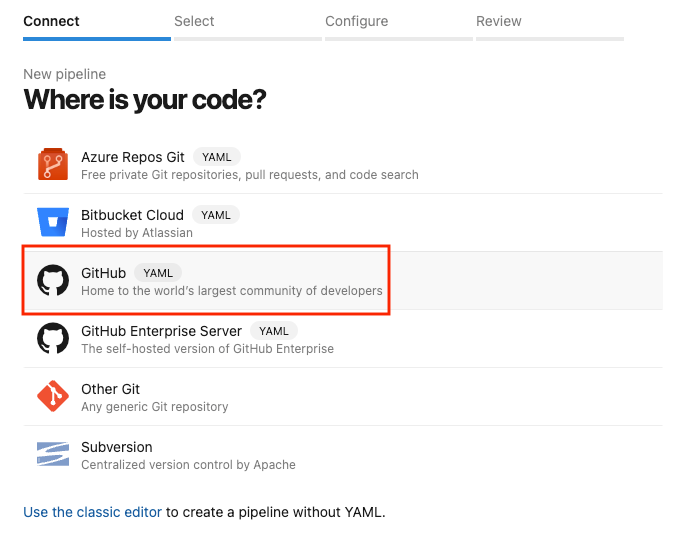
Azure Pipelines are used to build projects before the deployment.



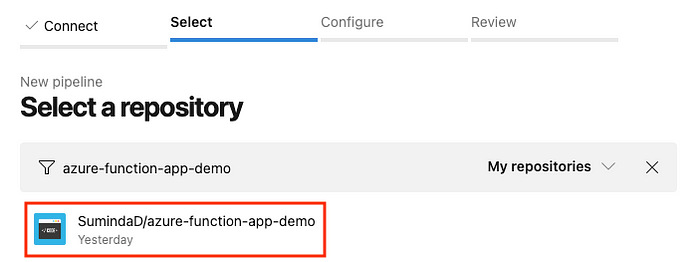
Goto Pipelines and create a new pipeline.



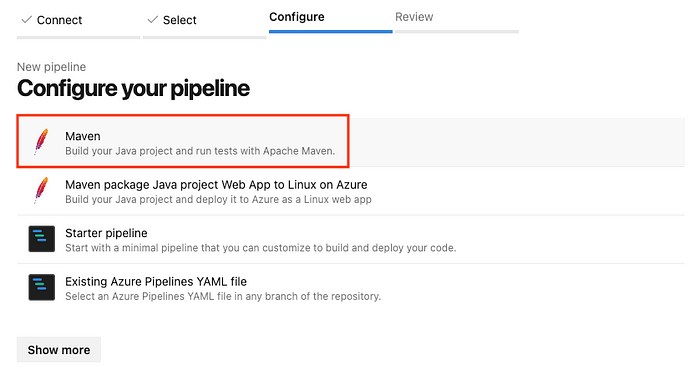
Select Github. You will be prompted to authorize Devops access to your Github account. Follow the instructions and provide approval.



Afterwards select the relevant repository.



Select build configuration as “Maven”.



Generated Azure Pipelines file will be displayed. This file will be checked in to the project repository when clicking on “Save and Run”. For every project Azure Devops needs this “azure-pipelines.yml” file to,

1. Trigger the build pipeline.
2. Build the project.
3. Publish project to Azure Devops artifact storage.

Currently the generated “azure-pipelines.yml” file only contains the build step. We need to add “build” and “publish” steps. Please do following changes,

1. Make sure JDK version is set to “1.17”.



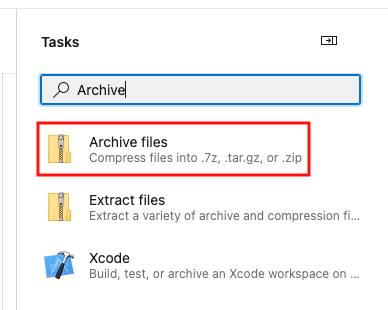
2. From Devops assistance menu select “Archive Files” step to add it to the “azure-pipelines.yml” file.

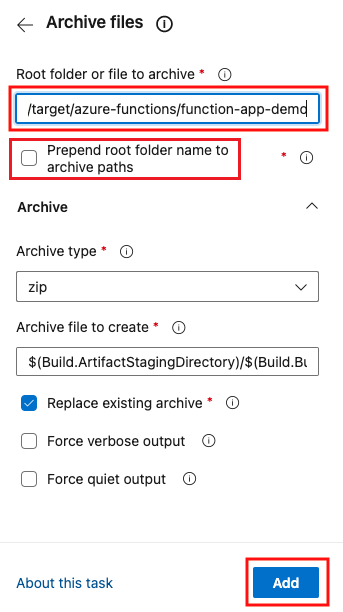
Change “Root folder or file to archive” path as below. The directory containing the project build files needs to be specified here.

Make sure to uncheck “Prepend root folder name to archive paths”.

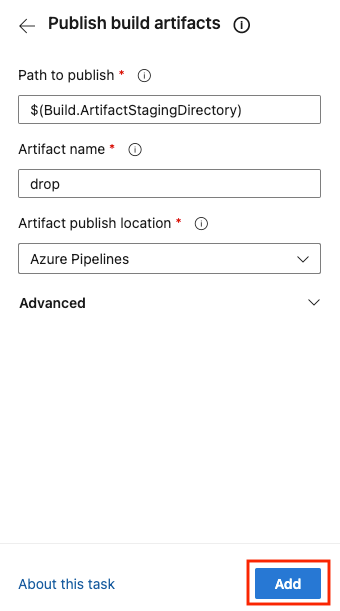
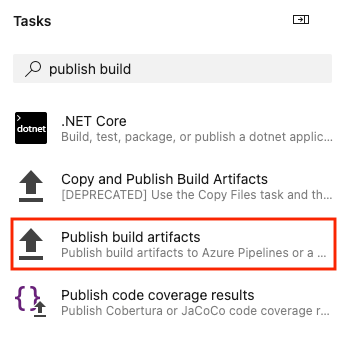
$(Build.SourcesDirectory)/target/azure-functions/function-app-demo

Click on “Add”.



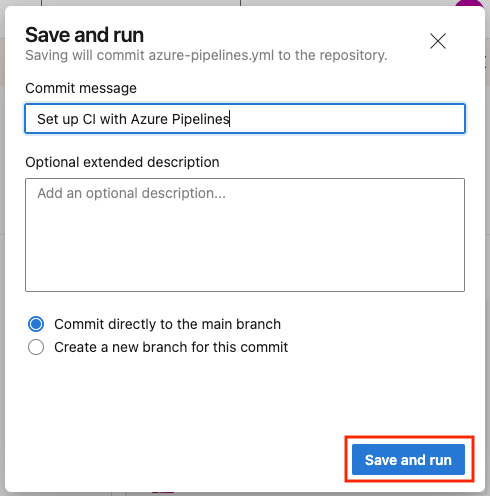


3. From Devops assistance menu select “Publish build artifacts” step to add it to the “azure-pipelines.yml” file and keep the settings as defaults.

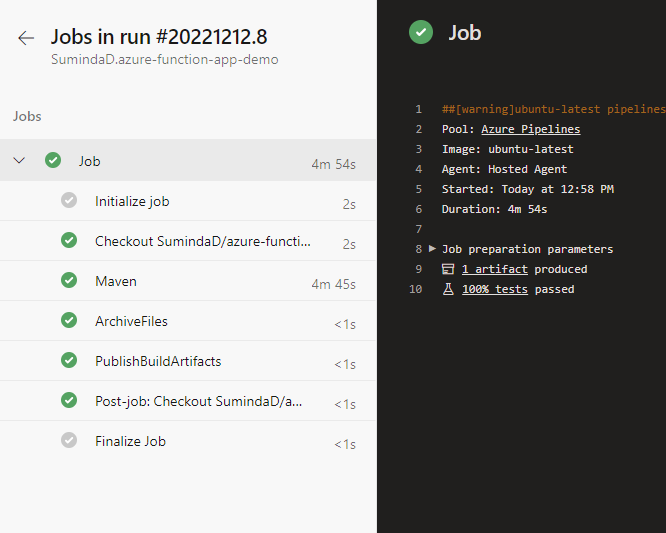


The “azure-pipelines.yml” file is completed now as below. Note the “trigger” value is “main”. This means Azure Devops pipelines will only be triggered if code is checked into “main” branch.

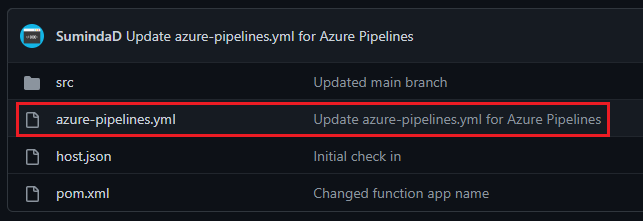
4. Click “Save and run” button to check in the “azure-pipelines.yml” file to the repository and trigger the build process as below.



5. The build process will complete similar to below.

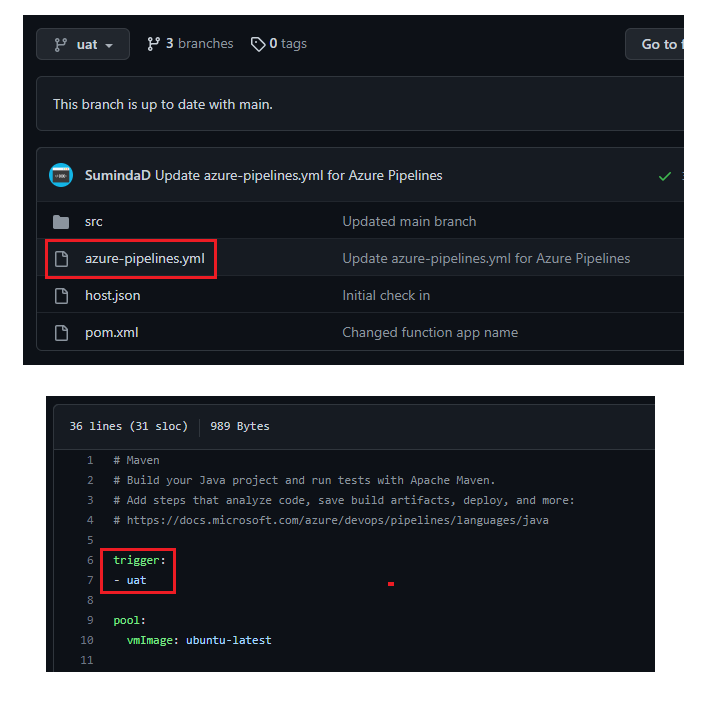


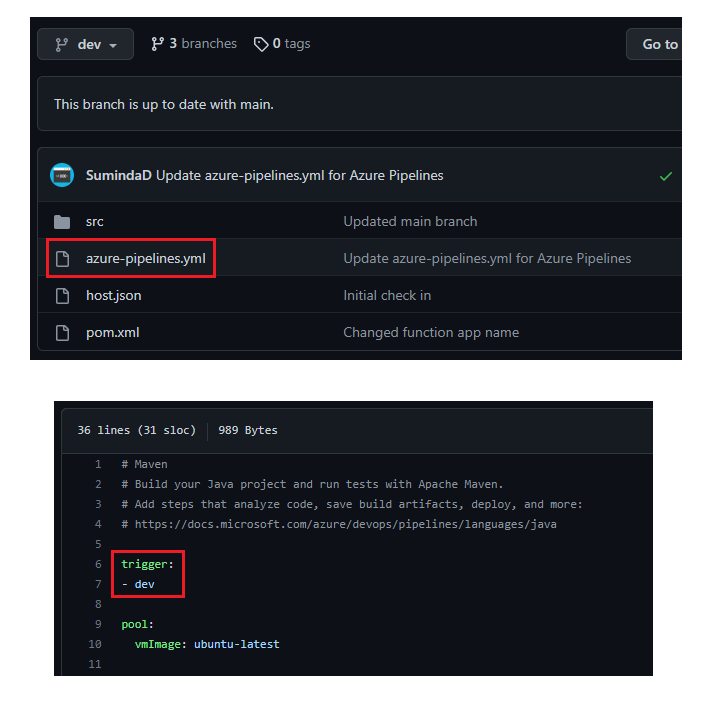
If we inspect the project repository, we can see that “azure-pipelines.yml” file is checked into the project repository as below.



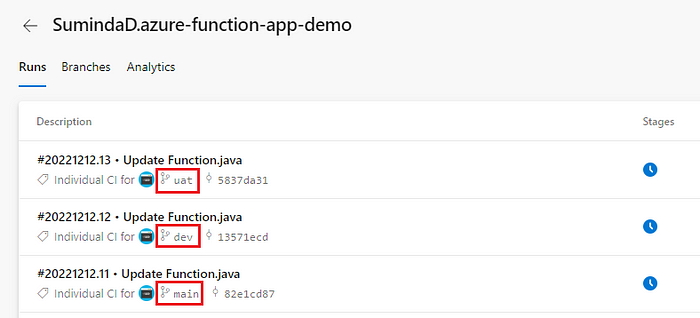
**5. Re-use Same Azure Devops Pipeline for Separate Project Branches**

In order to re-use the same build pipeline for separate branches, we need to check-in the “azure-pipelines.yml” file for each branch. Also, the “trigger” for each “azure-pipelines.yml” file needs to be changed for each branch. Please refer to below,





Once this is configured, any code commits for any branch will trigger the relevant Azure Pipeline build pipeline as below,



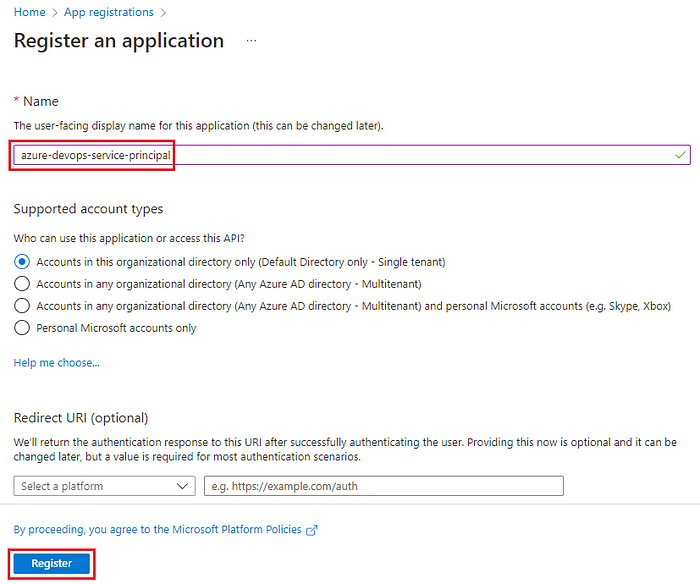
This will allow us to re-use the same Azure Build Pipeline for multiple branches.

**6. Azure Service Principal Deployment Permission**

[Azure service connection](https://docs.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml) is used to get necessary permissions to deploy into Azure environments. A Service Principal with the required permissions to the Azure resources will be configured in Azure DevOps.

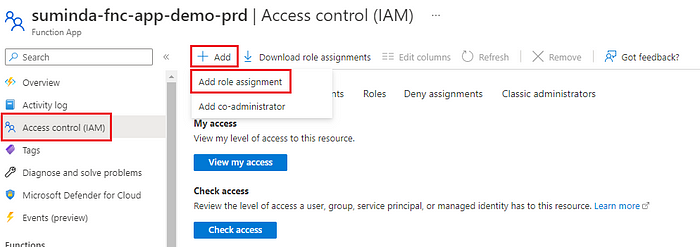
An Azure App Registration will need to be created in order get the required service principal with neccassary permissions.

1. Go to Azure Portal.
2. Go to “App registrations”.
3. Click “New registration”.
4. Enter name as “azure-devops-service-principal” and click “Register”.

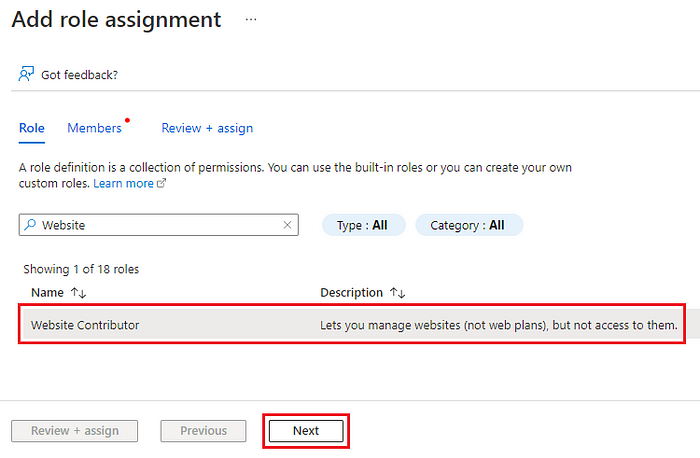


Once the Azure App Registration is created, it will also create a service principal. Now the permission to deploy binaries to the Function Apps needs to be given. Follow below steps for each and every Function App (PRD, UAT and DEV).

1.Go to Azure Function App.  
2.Go to “Access control (IAM)”, click “Add” and select “Add role assignment”.

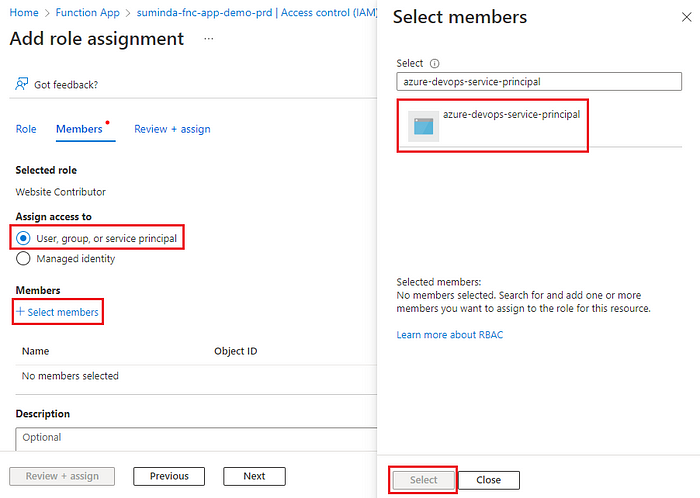


3.Select “Website Contributor” role and click “Next”. The Website Contributor role grants permissions to deploy code to Azure Function app.

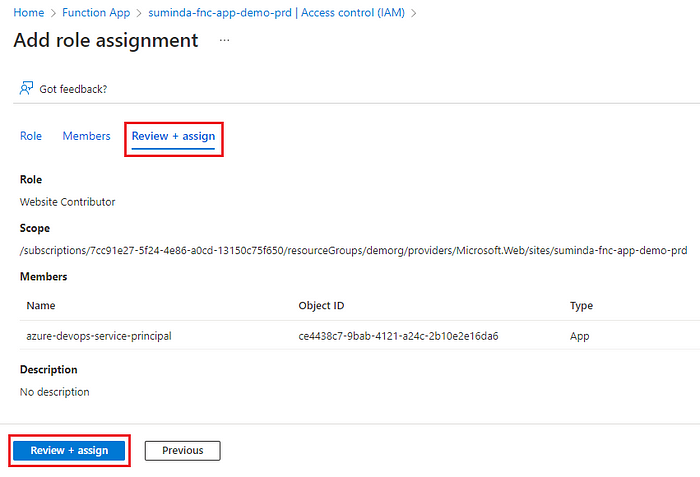


4.Make sure “User, group or service principal” radio button is selected and click “Select members” button.

5.Search for the Azure App Registration created earlier and click on “Select” button.



6.Click on “Review + assign”.

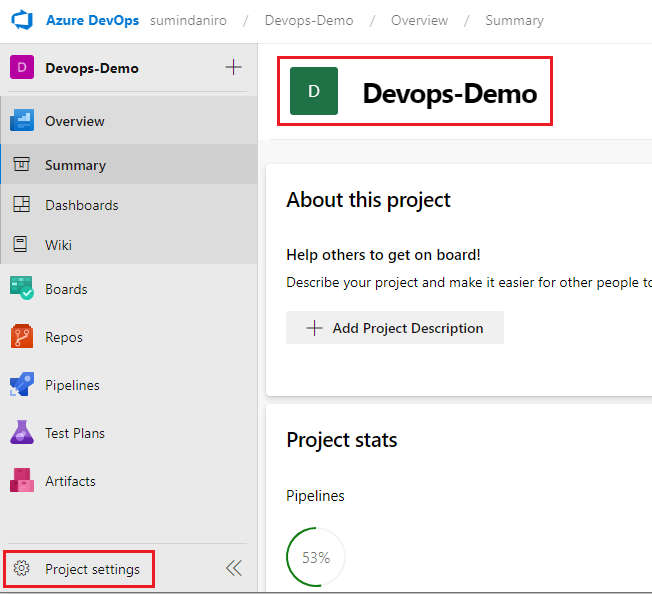


With the completion of the above step, using the service principal just created and given permission to, we need to configure Azure Devops Release Pipeline to use the service principal for deployments.

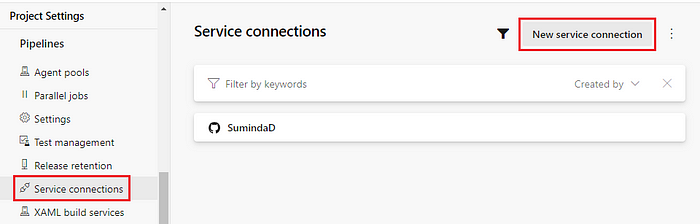
**7. Assign Service Principal to Azure Devops**

In order to get required permissions to deploy code for Azure Function Apps, need to assign the previously created service principal for Azure Devops.

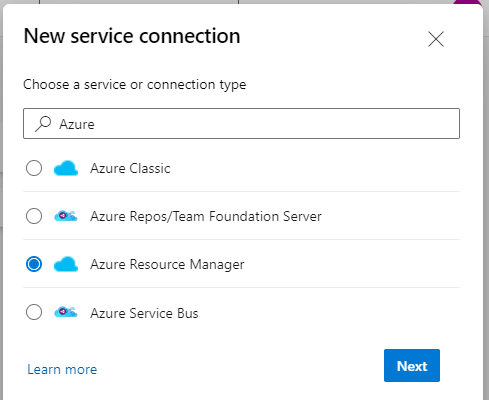
1.Go to “Devops-Demo” project and select “Project settings”.

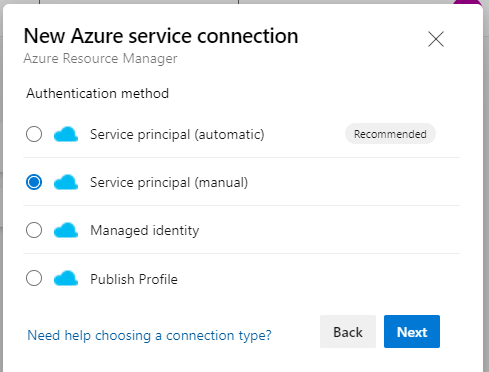


2.Click “Service connections” and “New service connection”.



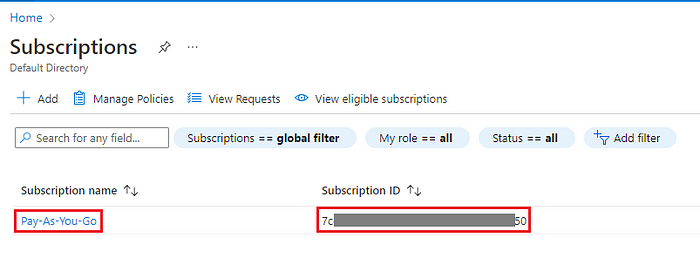
3.Select “Azure Resource Manager” and then “Service principal (manual)”.



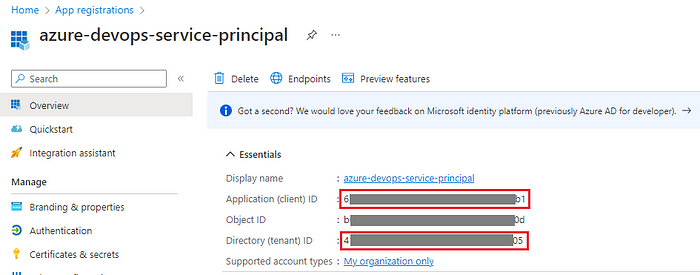


4.Following details are required to configure service principal.

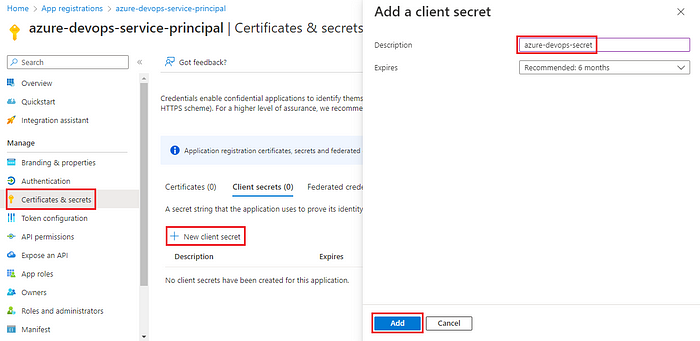
* **Subscription Id** & **Subscription Name** — Get from Azure Subscriptions as below.

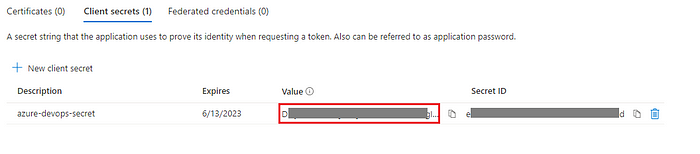


* **Service Principal Id** (Application client ID) & **Tenant ID** — Get from Azure App Registration “azure-devops-service-principal” Overview page as below.



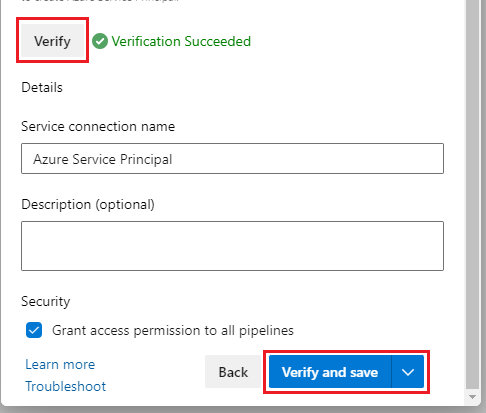
* **Service principal key** — Go to “Certificates & secrets” in Azure App Registration “azure-devops-service-principal” and generate a new secret as below and copy it.





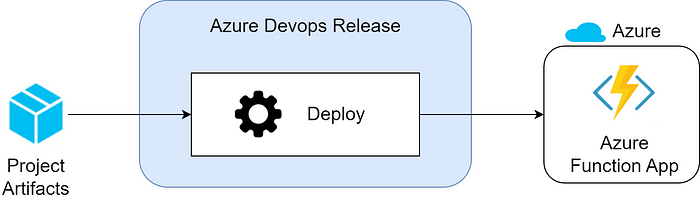
* **Service connection name** — Provide a service connection name as needed. In this example, “Azure Service Principal” is given.

5.Select “Grant access permission to all pipelines” check box and click “Verify”. This will make sure all the connection information is accurate and display successful message. Click “Verify and save”.

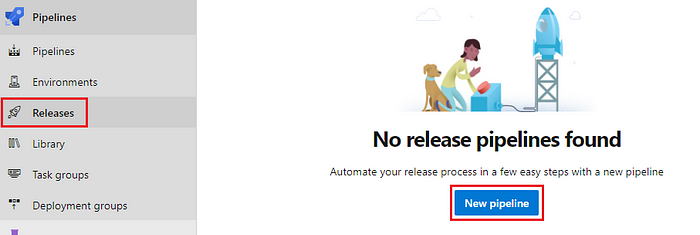


**8. Create & Configure Azure Devops Release Pipeline**

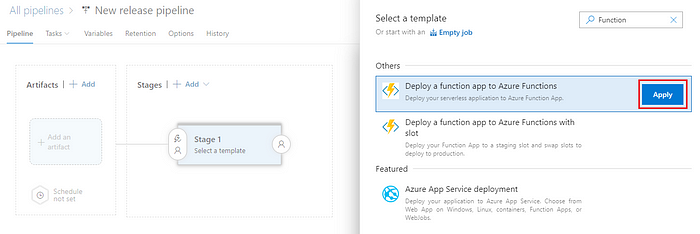
Azure Devops release pipelines allows to deploy the artifacts built via Azure Pipelines to the runtime environment.



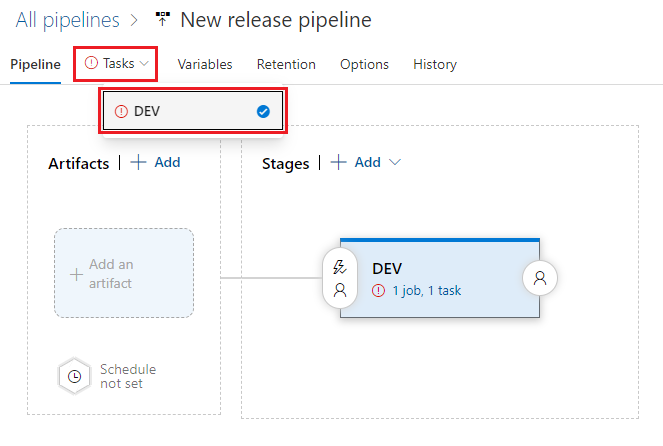
1.Go to “Releases” and click “New pipeline” as below.



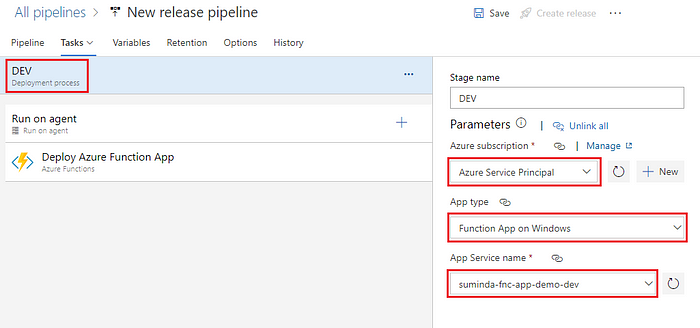
2.Select “Deploy a function app to Azure Functions” as below.



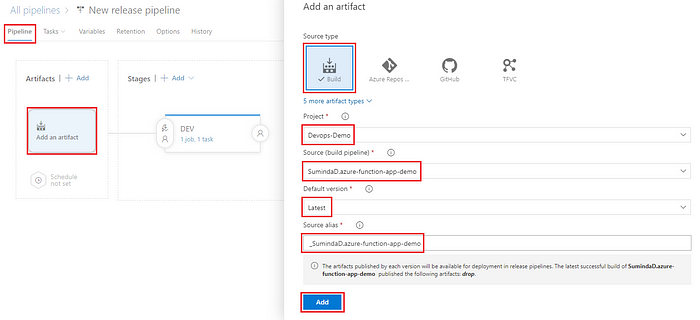
3.Enter “Stage name” as “DEV” and go to “Tasks” and select “DEV”.



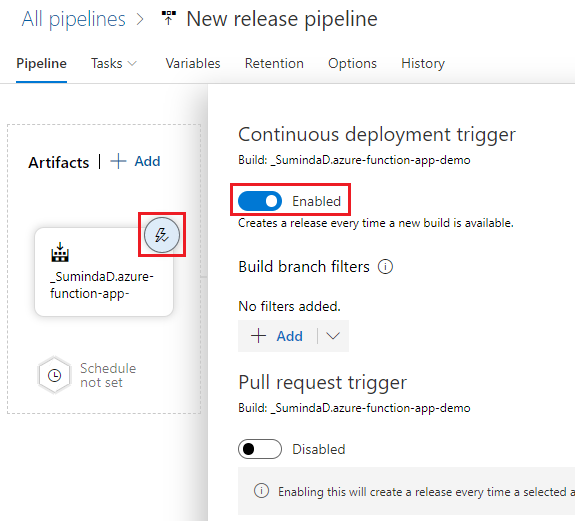
4.Select the Azure Service Principal to be used here and the Azure Function app to deploy the code to as below.



5.Select “Pipeline” and configure artifact as below. This will configure which build pipeline needs to be used in order to get the build artifact.



7.Enable “Continuous deployment trigger” for build artifact. This will trigger a new release automatically whenever a new build is available.

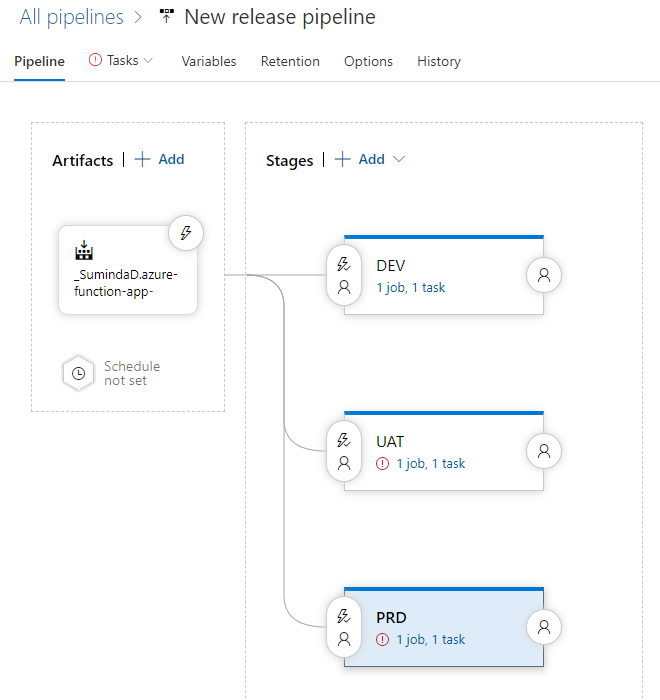


6.Click “Save”.

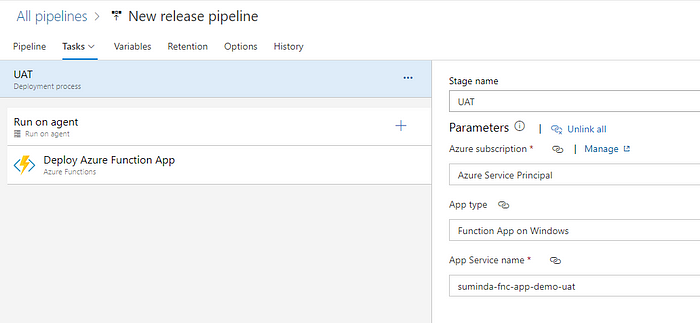
**Add Multiple Stages for UAT and PRD**

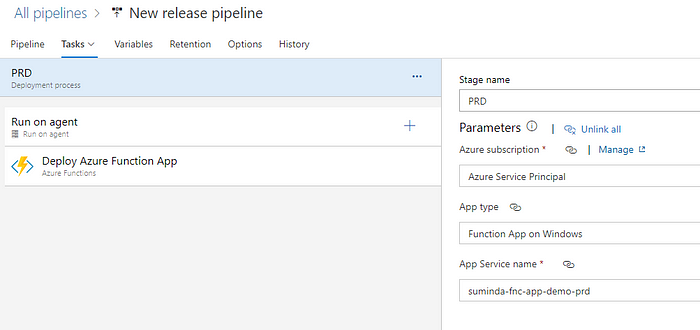
DEV release pipeline is now complete! Let’s add for UAT and PRD as well.

Follow the step 2 and create 2 new stages for UAT and PRD as below.



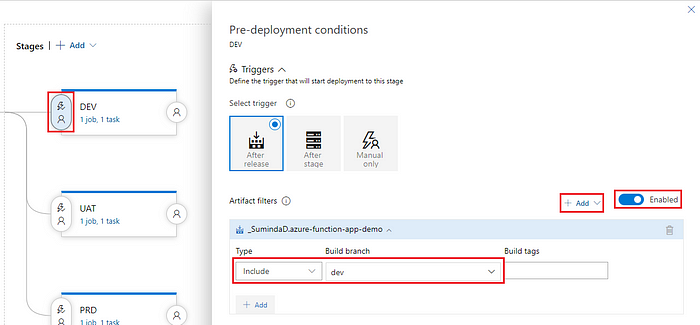
Follow step 3 & 4 and configure UAT and PRD stages as below.

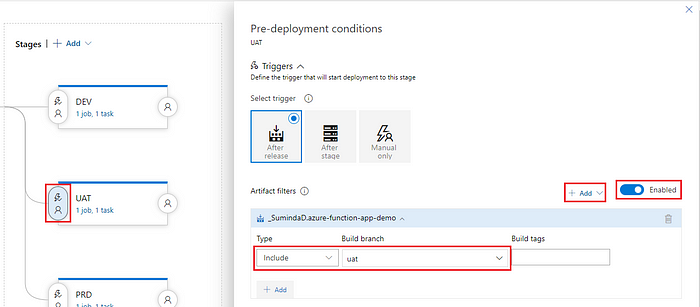


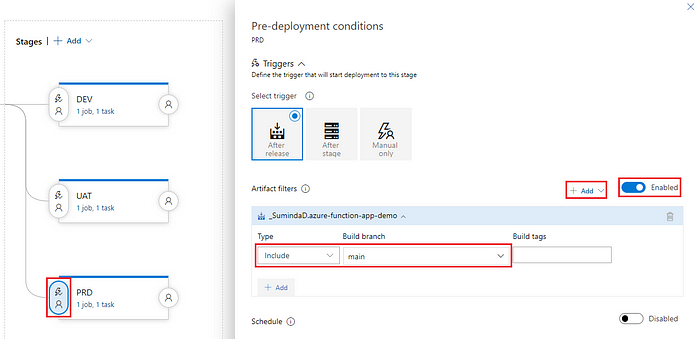


**Add Artifact Filters**

In order to make sure only relevant build is deployed to the relevant Azure Function App, we need to configure artifact filters for each DEV, UAT and PRD stage as below.





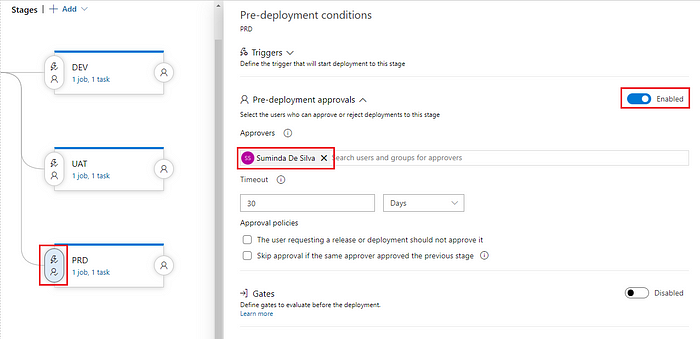


**Add Pre-deployment Approvals**

It is a good practice to make sure Production and UAT stages require user approval before triggering an actual deployment.

Approvals can be specified in “Pre-deployment conditions” as below. Once this is enabled, whenever there’s a new release, approvers will get an email to provide approval for the deployment.

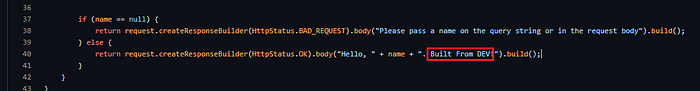
In this example only the PRD stage has pre-deployment approvals configured.

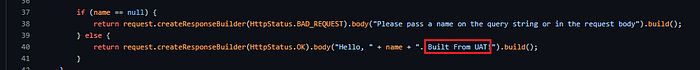


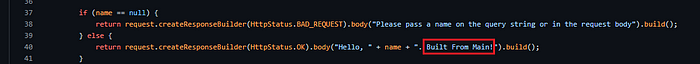
Azure Release Pipeline configuration is now complete.

**9. Test Azure Devops Pipeline**

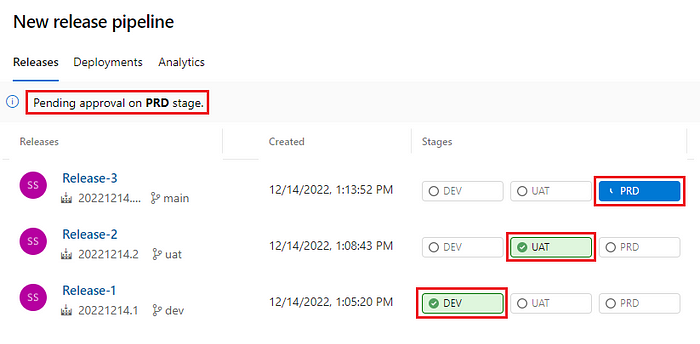
Go to each Github branch and make a code commit which will uniquely identify each build as below.



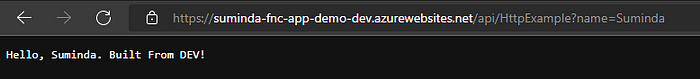


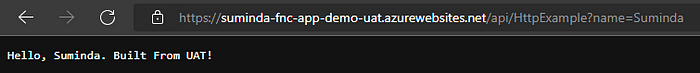


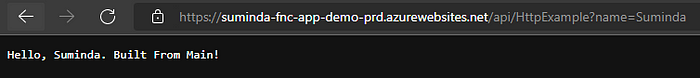
Once these changes are committed, after a while 3 new releases will be created. Dev and UAT branch builds will be automatically deployed, and PRD stage will be kept pending approval as below.



Once approval is granted and PRD stage also deployed. Go to each function app API endpoint and make sure each build is deployed as expected.







As you can see the 3 builds have been successfully deployed to the 3 separate environments!

**10. Self-Hosted Agents**

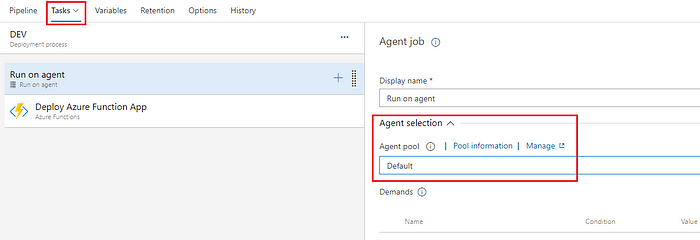
By default, Azure Devops uses Microsoft hosted agents for both Build and Release pipelines. But when it comes to release pipeline, most enterprises have Azure resources deployed in private networks which makes it impossible to deploy into them from Microsoft hosted agents since the Azure resources are not accessible over public internet.

As a solution Azure Devops allows us to install it’s agent into a Virtual Machine which can be hosted inside the organization’s Azure private network. This will allow Azure Devops to deploy builds via the Azure Devops agents into private Azure resources. These Virtual machines are called [Self-hosted agents](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser).

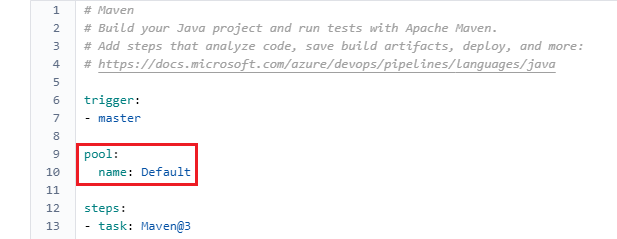
In order to configure a Windows based agent, following these [steps](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops).

It is recommended to run self-hosted agent as a windows service.

Once Self-hosted agent is configured, following configuration needs to be done in Release Pipeline to make sure Self-hosted agent is actually being used.



Additionally, if we require Build pipeline also to use Self-hosted agent, it needs to be specified in the “azure-pipelines.yml” file as below.



**11. Summary**

Hope you had a clear understanding of how Devops can be utilized to help project development.

Cheers!

**12. References**

[Azure DevOps documentation | Microsoft Learn](https://learn.microsoft.com/en-us/azure/devops/?view=azure-devops&viewFallbackFrom=vsts)

[Service connections in Azure Pipelines — Azure Pipelines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml)

[Deploy an Azure Pipelines agent on Windows — Azure Pipelines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops)

[Deploy an Azure Pipelines agent on Windows — Azure Pipelines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops)

[Azure Pipelines Agents — Azure Pipelines | Microsoft Learn](https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser)