Azure DevOps services

# **What is Azure DevOps?**

Azure DevOps **provides developer services for allowing teams to plan work, collaborate on code development, and build and deploy applications**. Azure DevOps supports a collaborative culture and set of processes that bring together developers, project managers, and contributors to develop software.

# **What are the benefits and features available in Azure DevOps?**

***Benefits:***

Azure DevOps provides DevOps teams with powerful tools.

* ***Timely Access to New Features***

Every three weeks, DevOps users receive access to new features. No need to scramble around and wonder what’s new. These are not to be confused with upgrades, and speaking of which:

* ***No Upgrades to Worry About***

Users need not worry about upgrading or patching up the toolchain because the Azure DevOps is a SaaS product. Companies that run on a CI/CD model no longer need to slow things down for the sake of upgrading.

* ***Reliability***

Azure DevOps is backed by 24 x7 support and a 99.9% SLA.

* ***Flexibility***

If your DevOps team doesn’t want or need the full suite of services, they can acquire them independently.

* ***It’s Platform-agnostic***

DevOps is designed to run on any platform (Linux, macOS, and Windows) or language (e.g., Android, C/C++, Node.js, Python, Java, PHP, Ruby, .Net, and iOS apps).

* ***It’s Cloud-agnostic***

Azure DevOps works with AWS and GCP.

***Features:***

Azure DevOps services offer a great selection of features for development teams. For instance:

* ***Dashboard Control***

Using the DevOps dashboard feature, you can quickly navigate to different areas of the project, add and manage dashboards, and configure dashboard widgets.

* ***Improved Source Control***

Azure DevOps systems support two popular types of source control: Git (distributed) or Team Foundation Version Control (TFVC), which is a centralized, client-server system. You can add and manage Azure Git tags, review, download, and edit files to see change history.

* ***Plan and Track Your Work***

Azure DevOps systems provide you with a couple of types of work items used to monitor features, requirements, user stories, tasks, bugs, and more. For planning purposes, you can access several kinds of backlogs and boards to support the main agile methods: Scrum, Scrumban, or Kanban. You can add and update relevant work items, manage product backlog, use sprint backlogs to plan sprints, and use Kanban boards to visualize the workflow and update statuses.

* ***Continuous Integration and Deployment (CI/CD)***

Many developers employ the practice of CI/CD, and Azure DevOps supports them. By using Azure pipelines, developers can automate many of the design processes, including defining builds and their steps, creating test instructions, and manage simultaneous releases.

* ***Support for Manual and Exploratory Testing***

Azure DevOps’ test features facilitate manual, exploratory, and continuous testing, including workflow customization, end-to-end traceability, criteria-based selection, and real-time charts that track test activity.

* ***Integrated Collaboration Services***

The feature that enables teams to collaborate across with the entire collection of Azure DevOps features and functions:

* Team dashboards
* Project wiki
* Discussion within work item forms
* Linking work items, commits, pull requests, and other artifacts that support traceability
* Alerts and change notifications managed per user, team, project, or organization
* The ability to request and manage feedback
* Analytics service, analytic views, and Power BI reporting
* ***Azure Cloud-hosted Services***

Azure provides DevOps teams with cloud-hosted services that support application development and deployment. These services can be used by themselves or in combination with Azure DevOps.

# **Azure Repos**

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 version control as soon as possible is recommended.

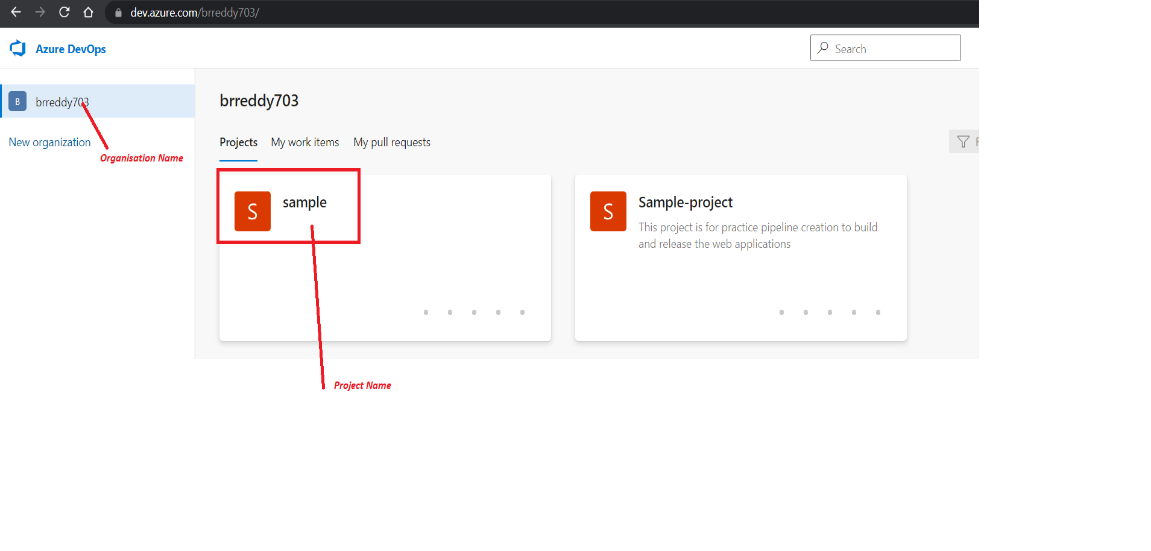
Creating Repository:

Prerequisites:

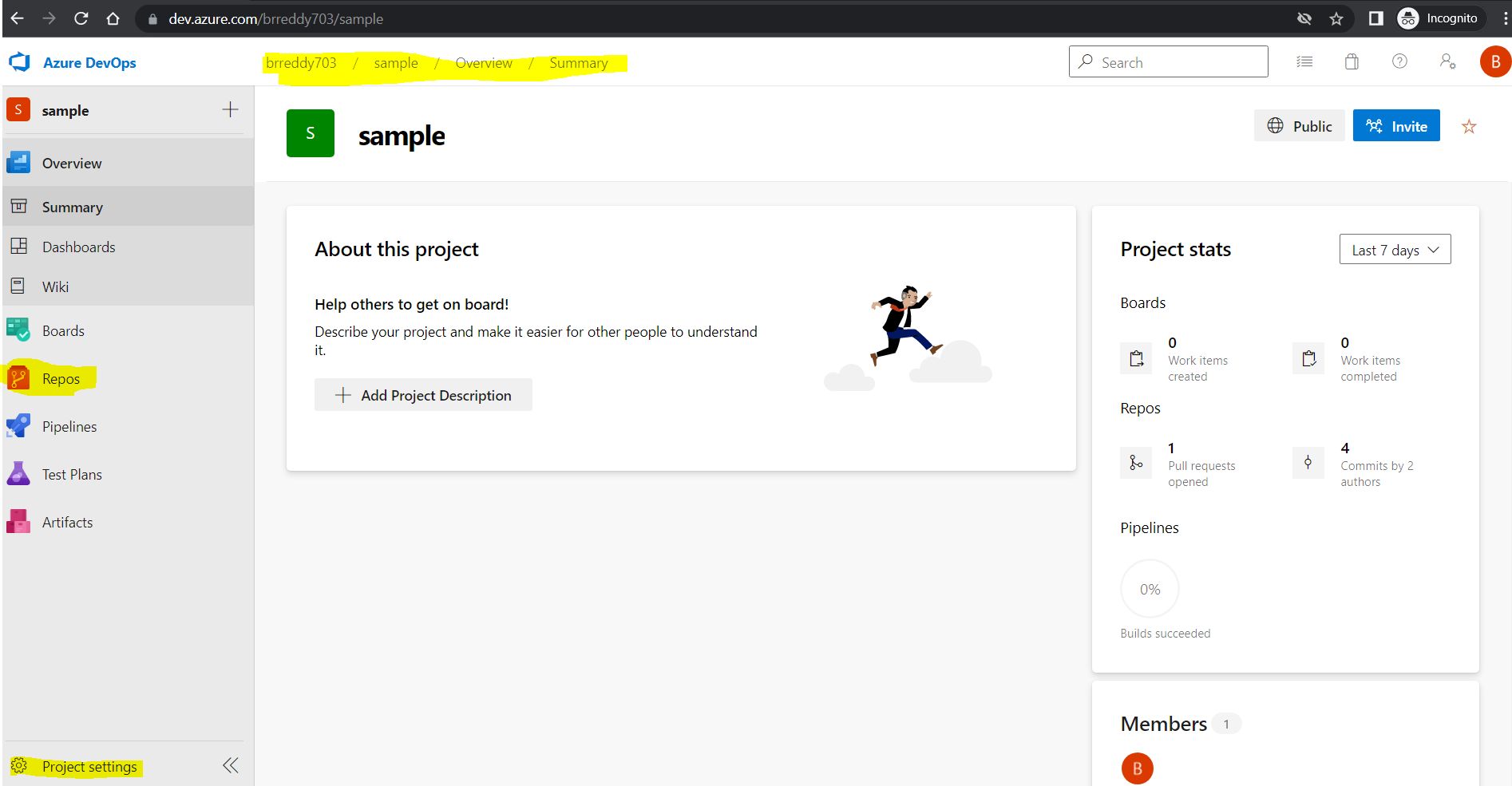
1. We should signup for Azure DevOps using dev.azure.com
2. Should be created one project in Azure DevOps Organisation

Steps to create Repo:

1. After login to <https://dev.azure.com> select project in which you need to create Repository

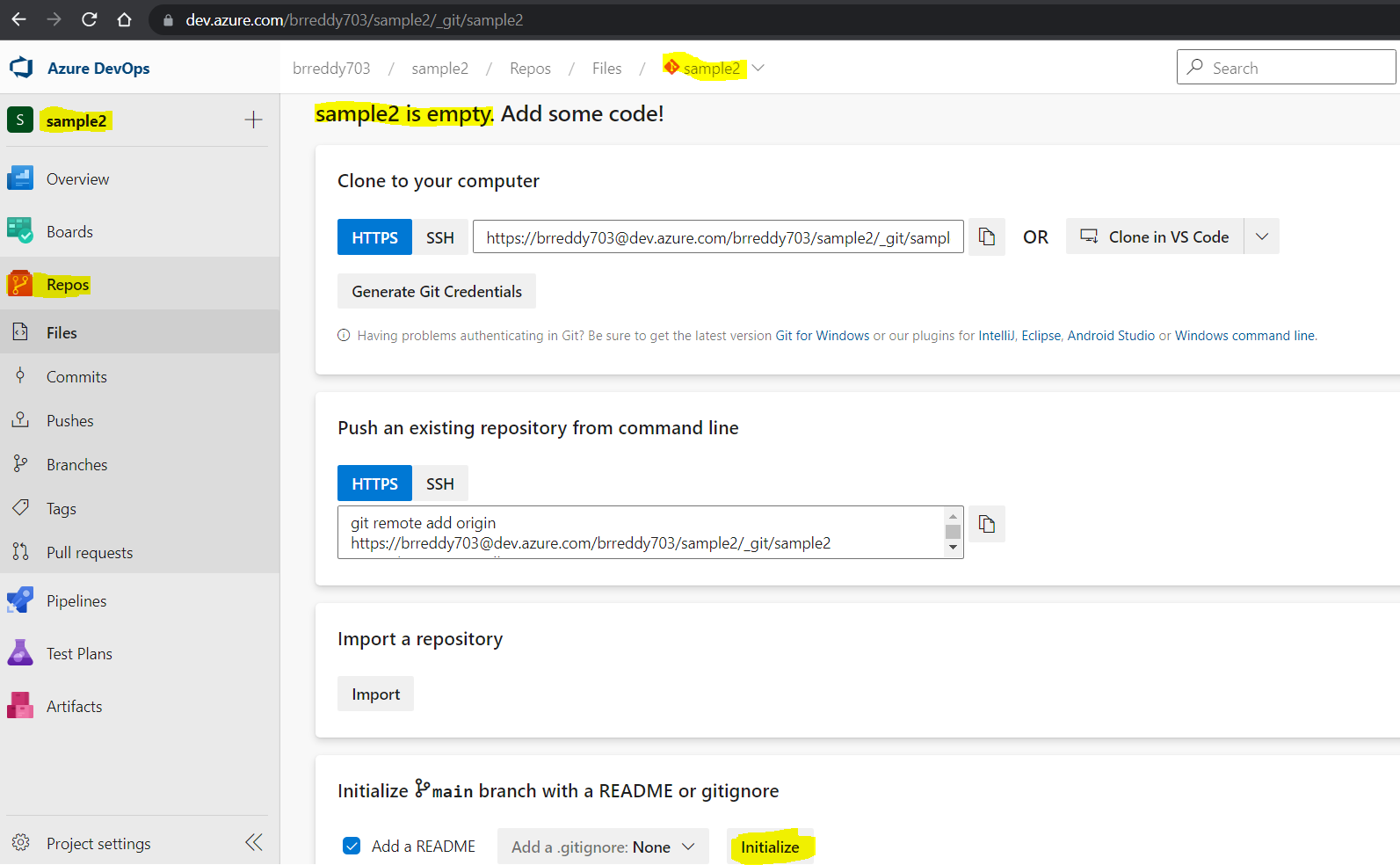


1. Click on project and it will take you to the project Summary page.

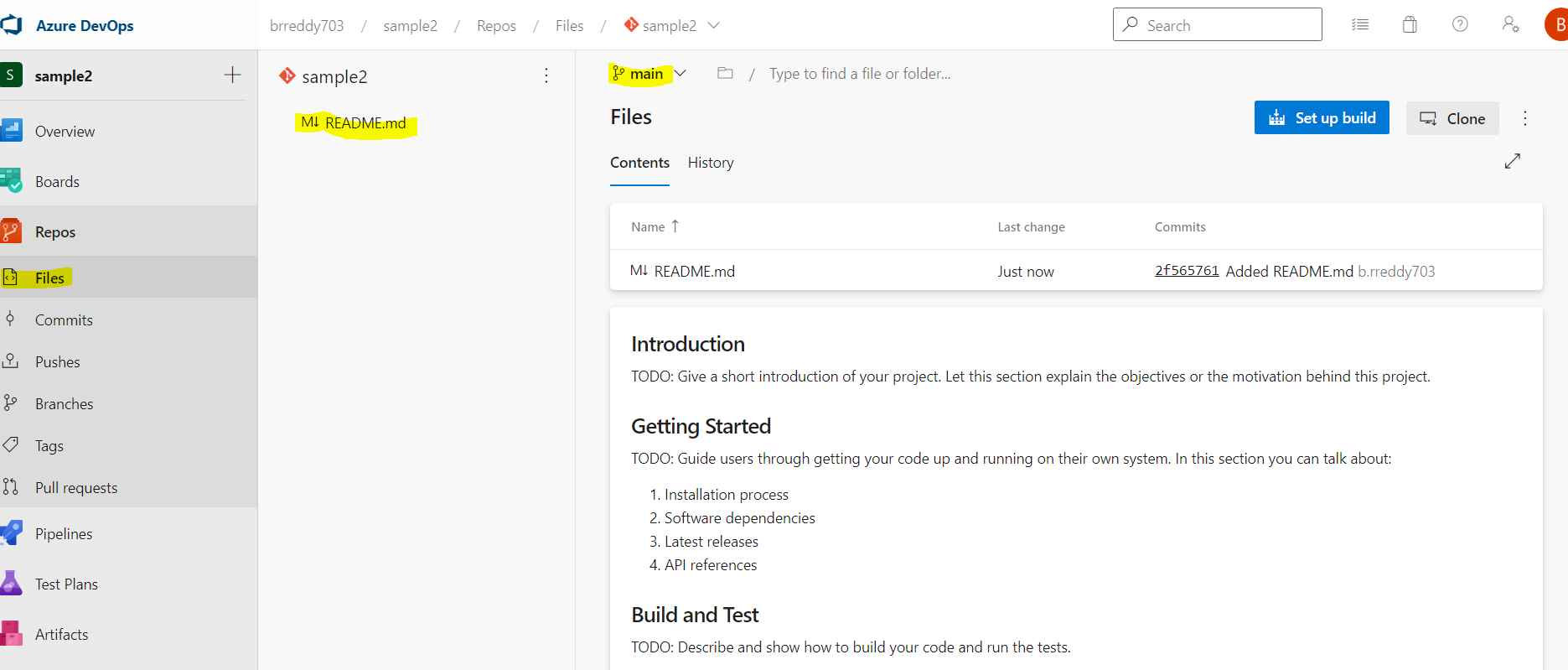


1. Click on the Repos from the left menu in summary page.

By default one empty repository will be available with Project name. Based on your requirement you can clone, add to your local machine using given commands. Else you can initialize in the browser itself by clicking on Initialize in the end.



1. Once you initialize it you can see page like below and it has ***main*** as a default branch. You can start development work or upload files to the newly created repos.



# **Azure Pipelines**

In this section we will see the step-by-step creation of pipelines.

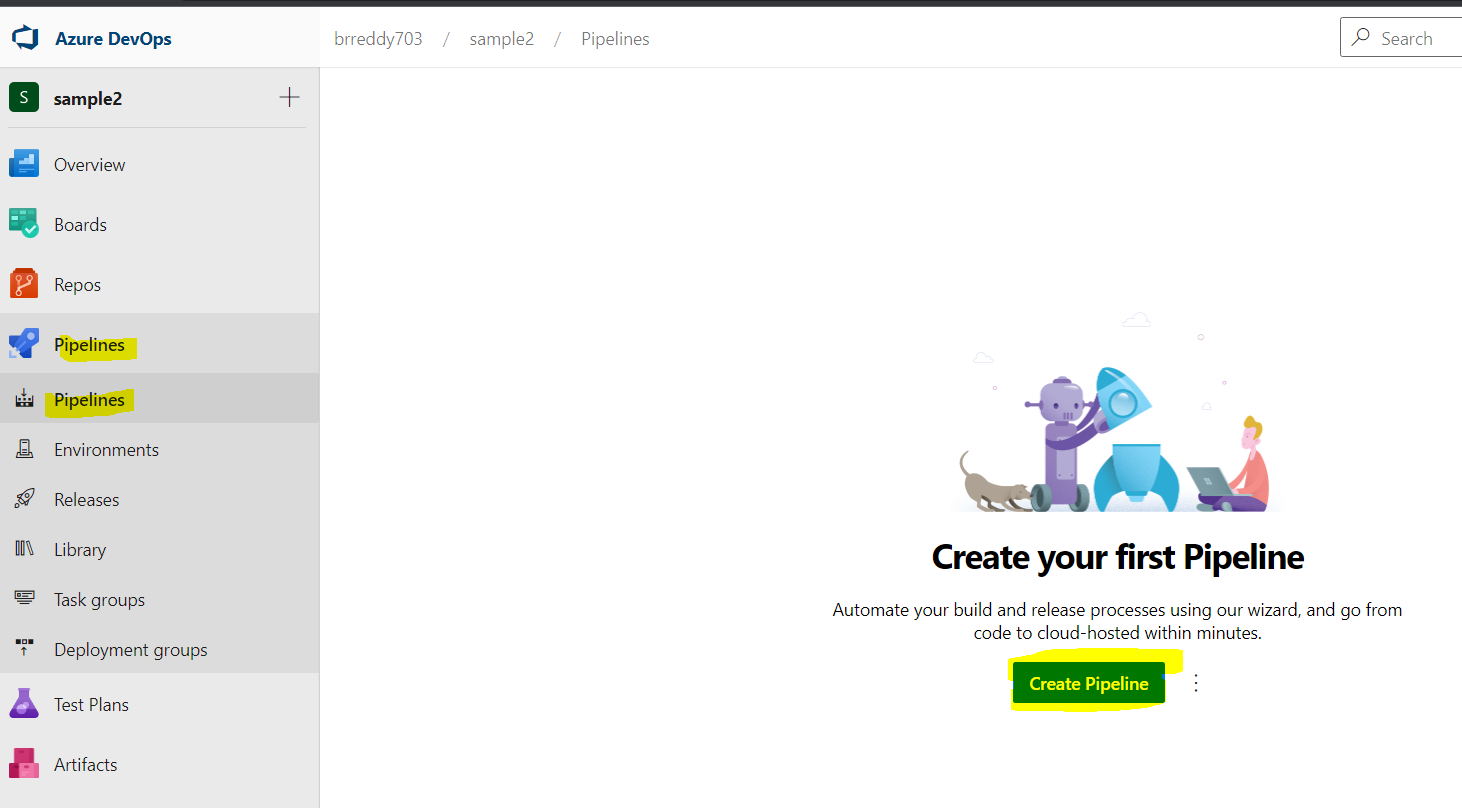
***Prerequisites:***

Before start to create a pipelines

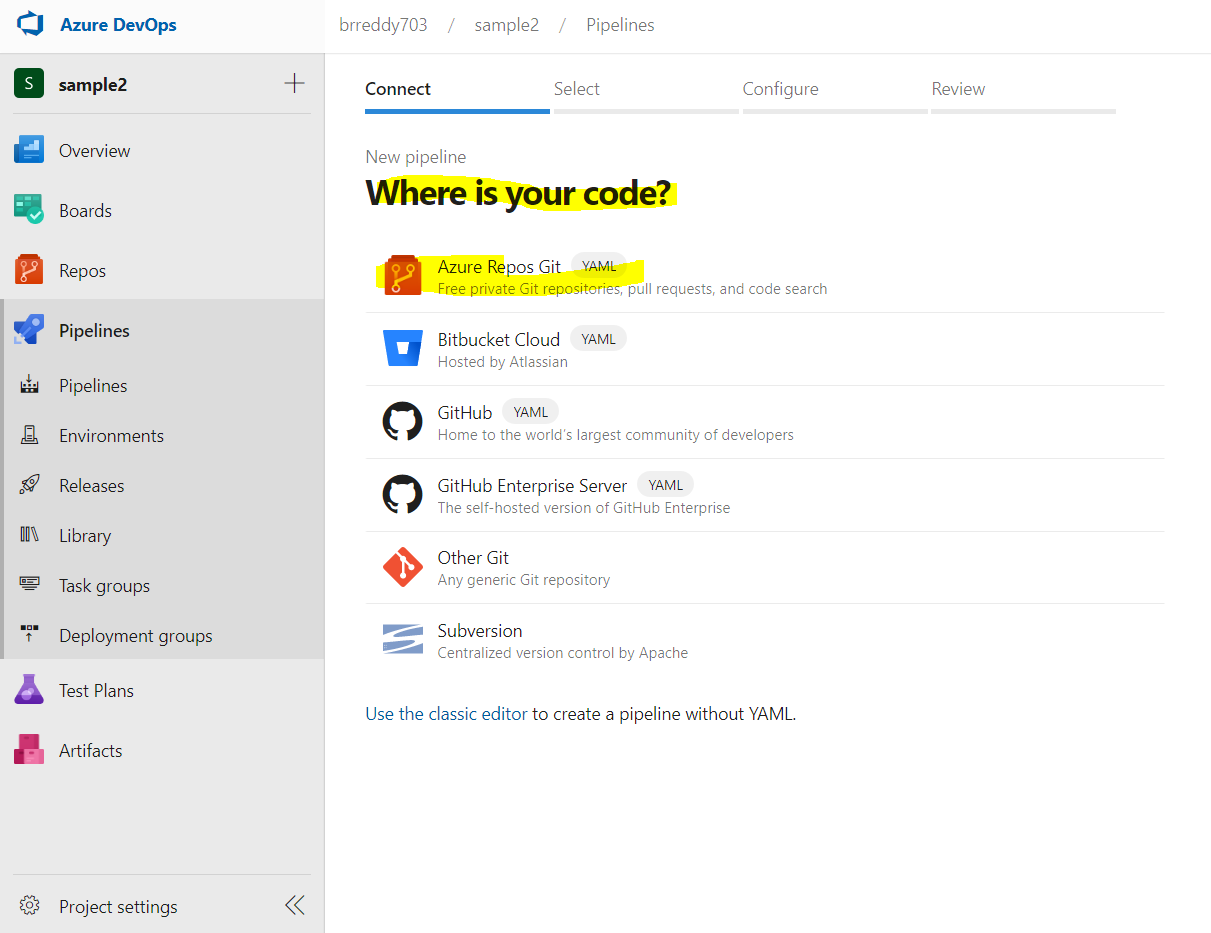
1. You should create one Project in Azure DevOps organisation. To [signup](https://azure.microsoft.com/en-us/services/devops/) azure DevOps
2. Should have one repository.

## **Creating Build pipeline:**

1. From the Project Summary click on Pipelines and Create New pipeline



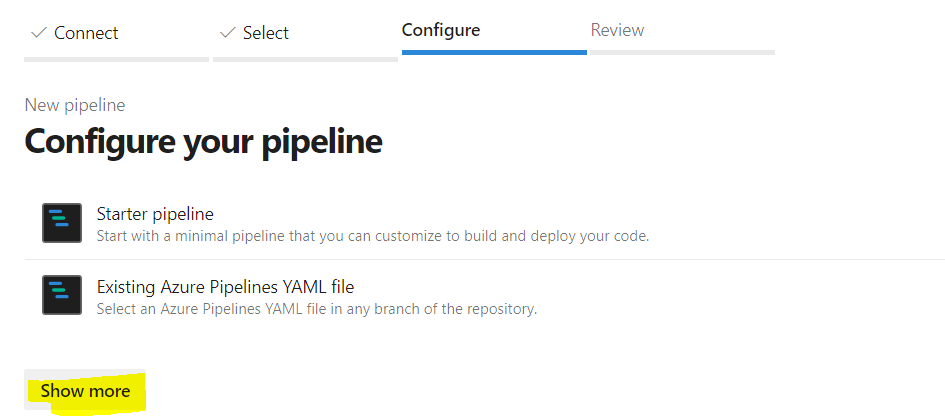
1. Next need to select Version control system in which your source code is saved. Select the Azure Repos Git



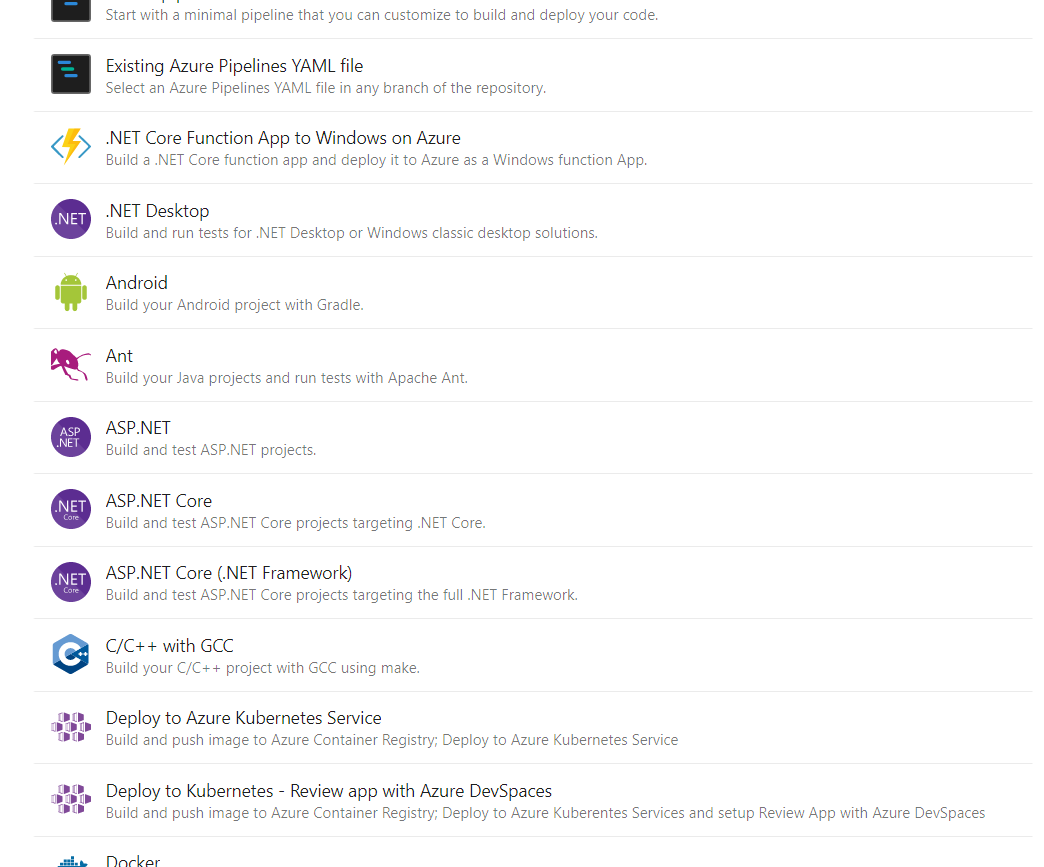
1. Need to select the repository from the list. It will list all repos in that Azure repos of selected Project.



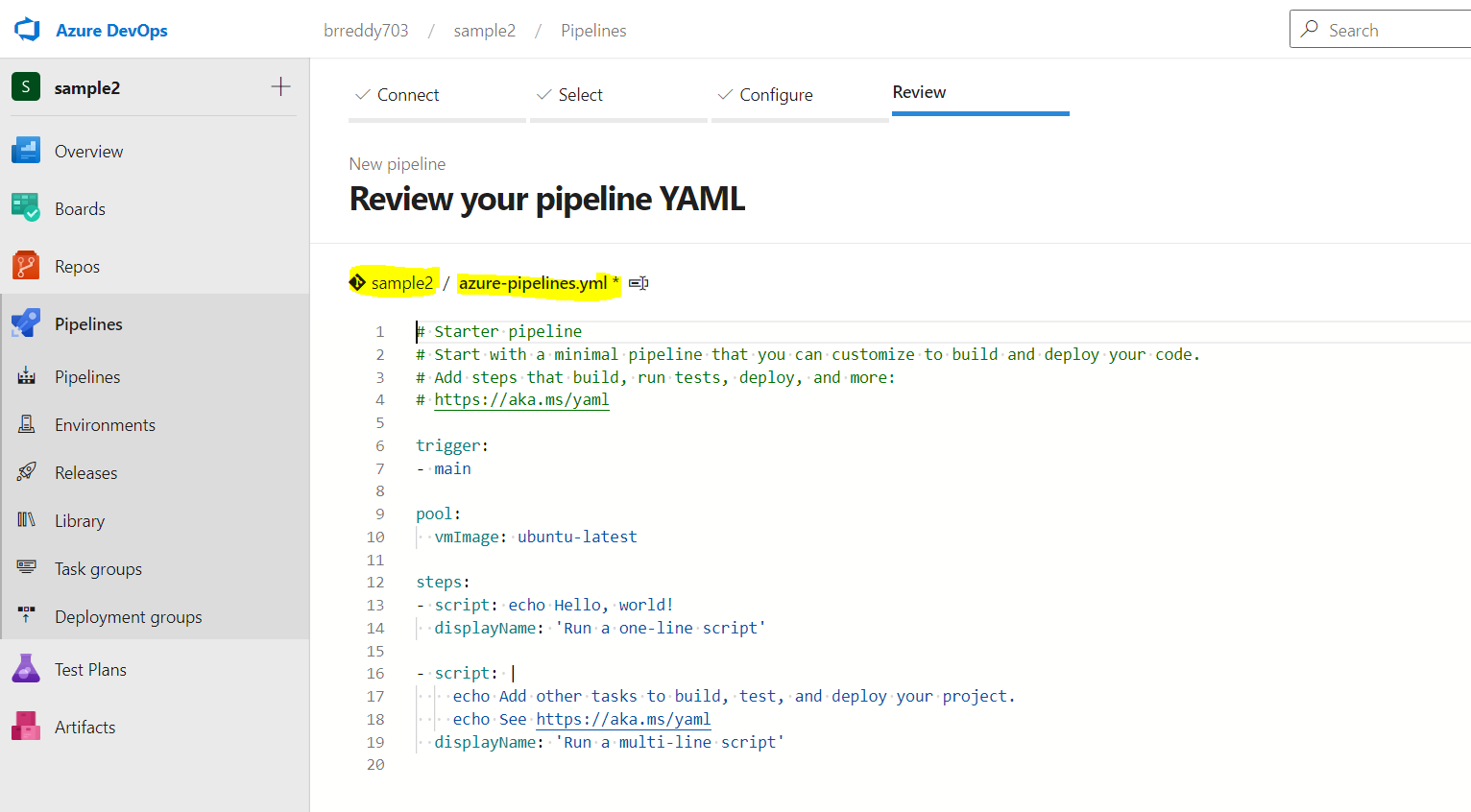
1. Now it will show the available pipeline template to create. Click on Show more for more templates if not displayed.



After click on Show more will display more templates. You can select templates based on your requirement. As of now I am selecting Starter pipeline only.



1. Once you select the template it will take us to the pipeline editor. By default pipeline’s yaml file name is ***azure-pipelines.yml*** and it will saved to main branch of repo.

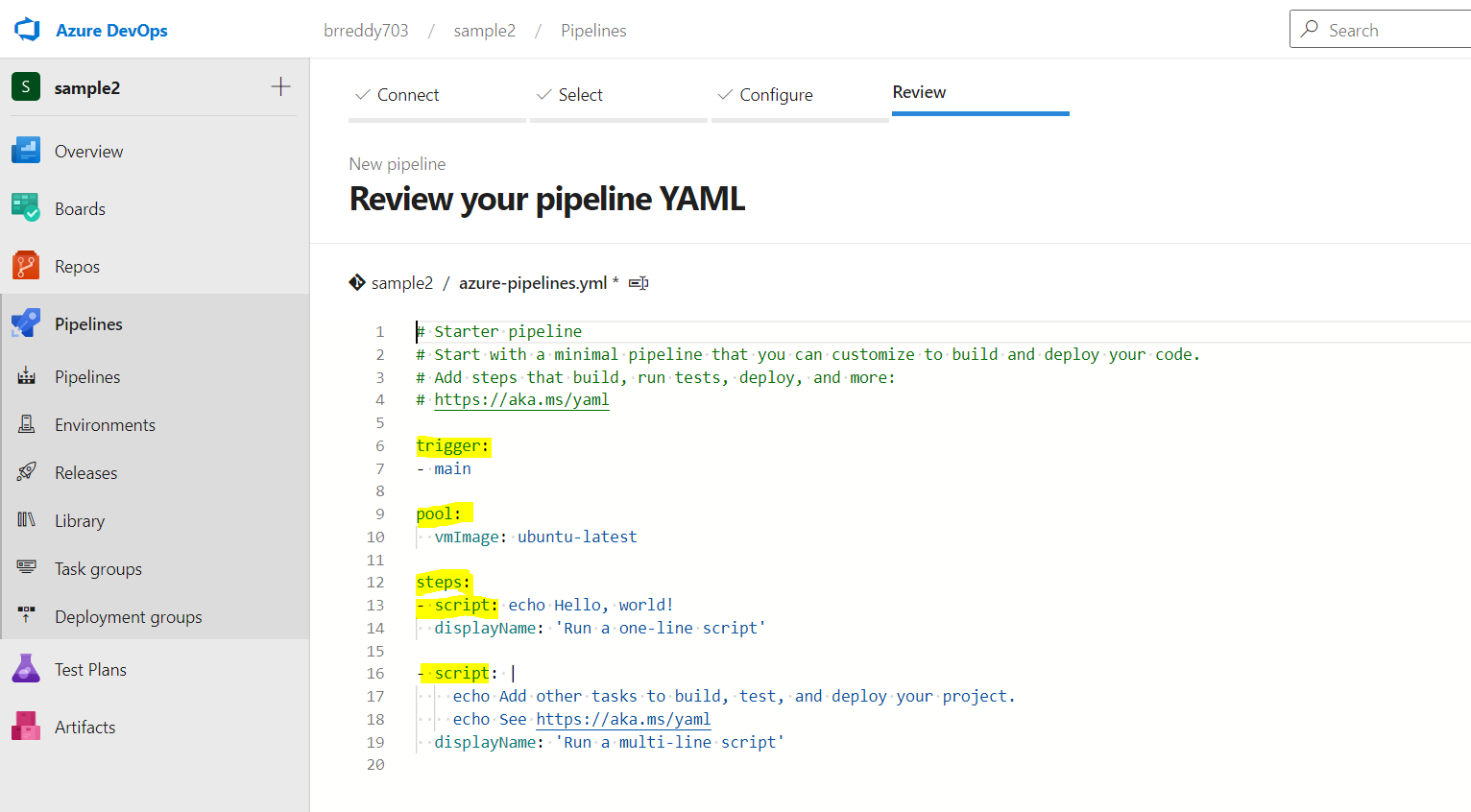


1. Here we can see the different sections in pipeline

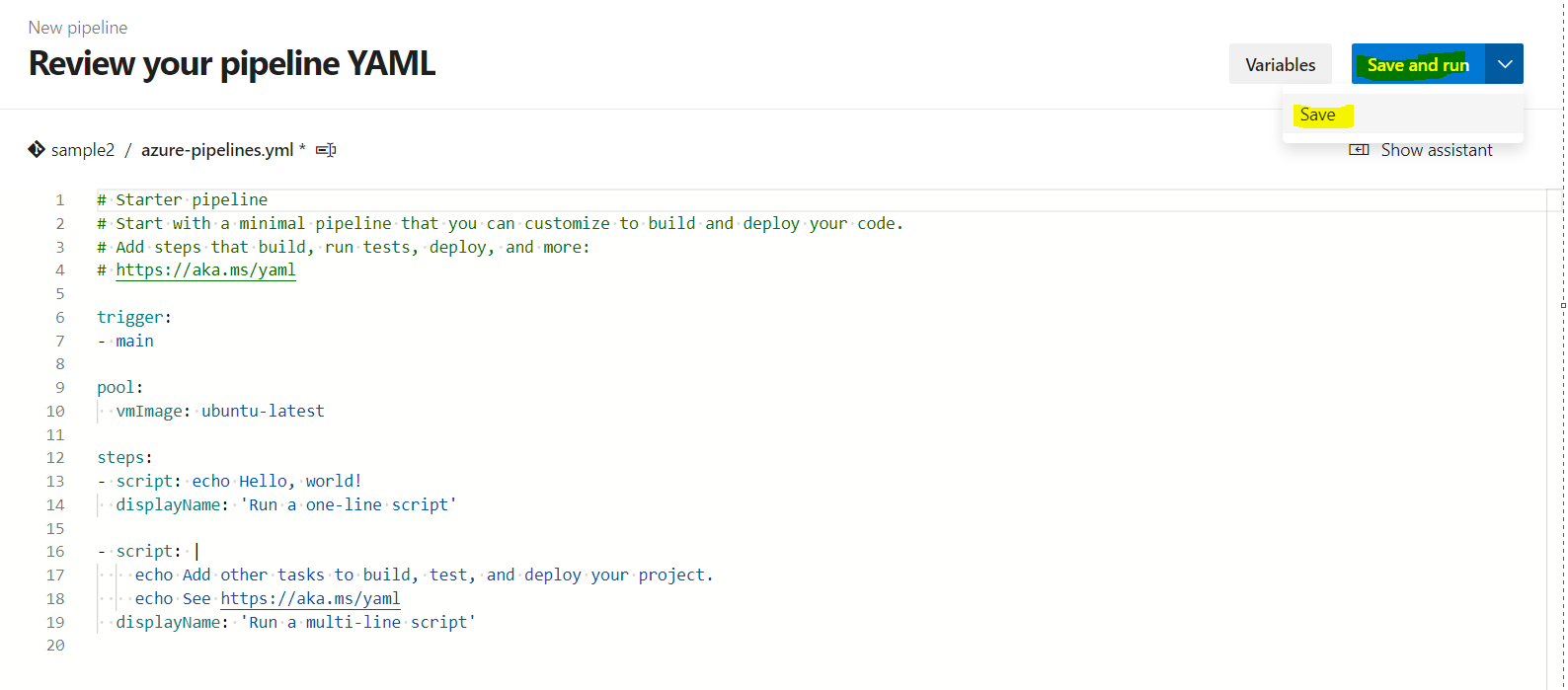
***Trigger:*** *#triiger is an event to run the pipeline. Any commit happens to main branch this pipeline will run.*

***Pool****: # agent pool It has agents on which pipelines tasks will be executed.*

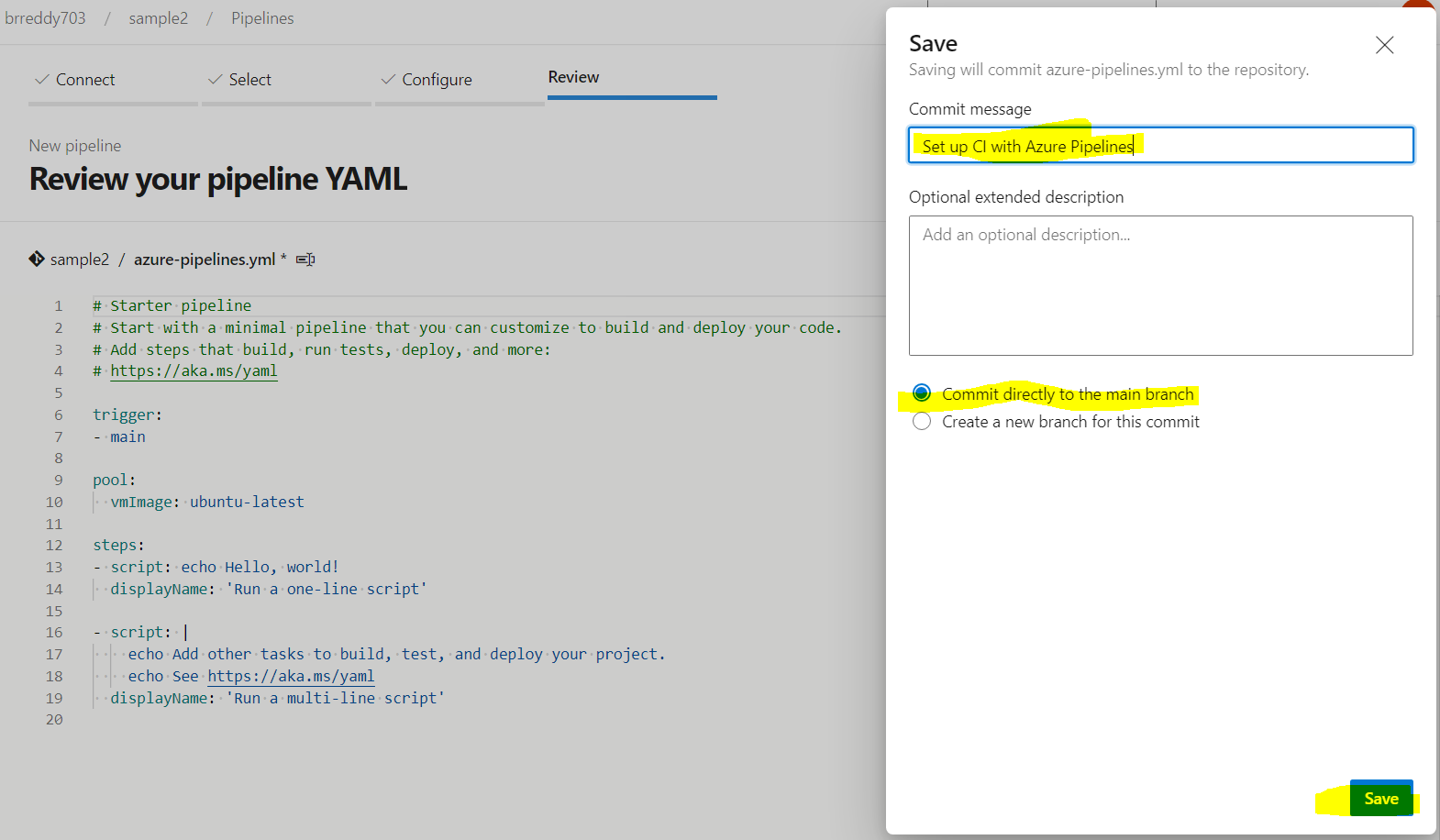
***Steps:*** *# Actual execution tasks will define in this section.*



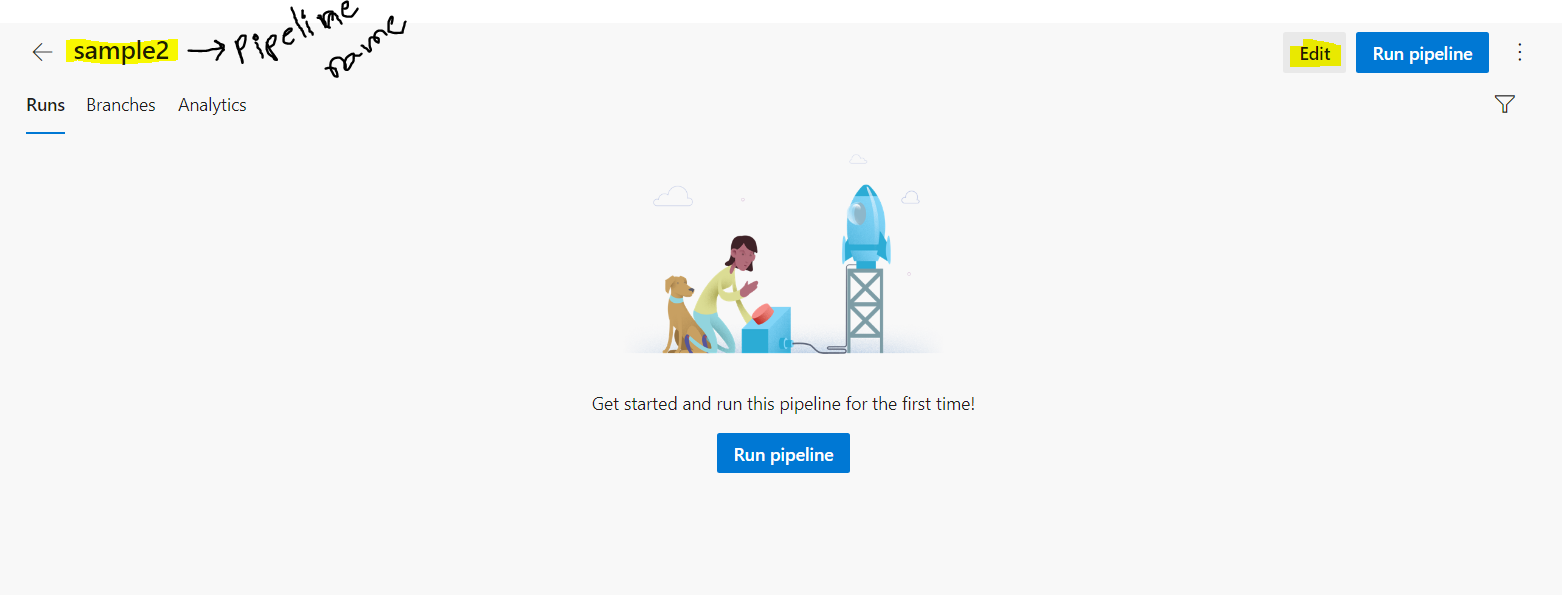
1. Once you done with pipeline edit. You can save and run or save it.by default pipeline name will be repository name.



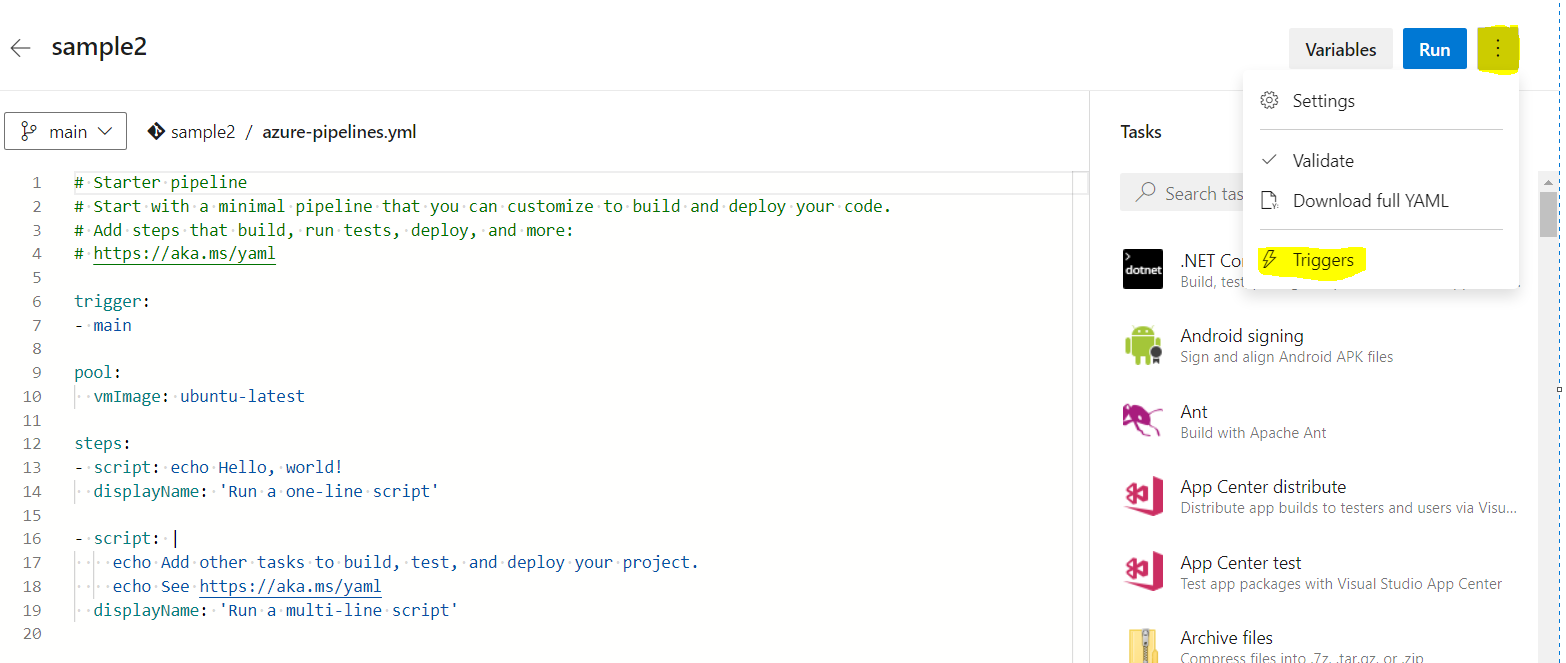
1. To rename the yaml pipeline, you need to select only save option. Click on save and enter commit message. Click on Save in the end.



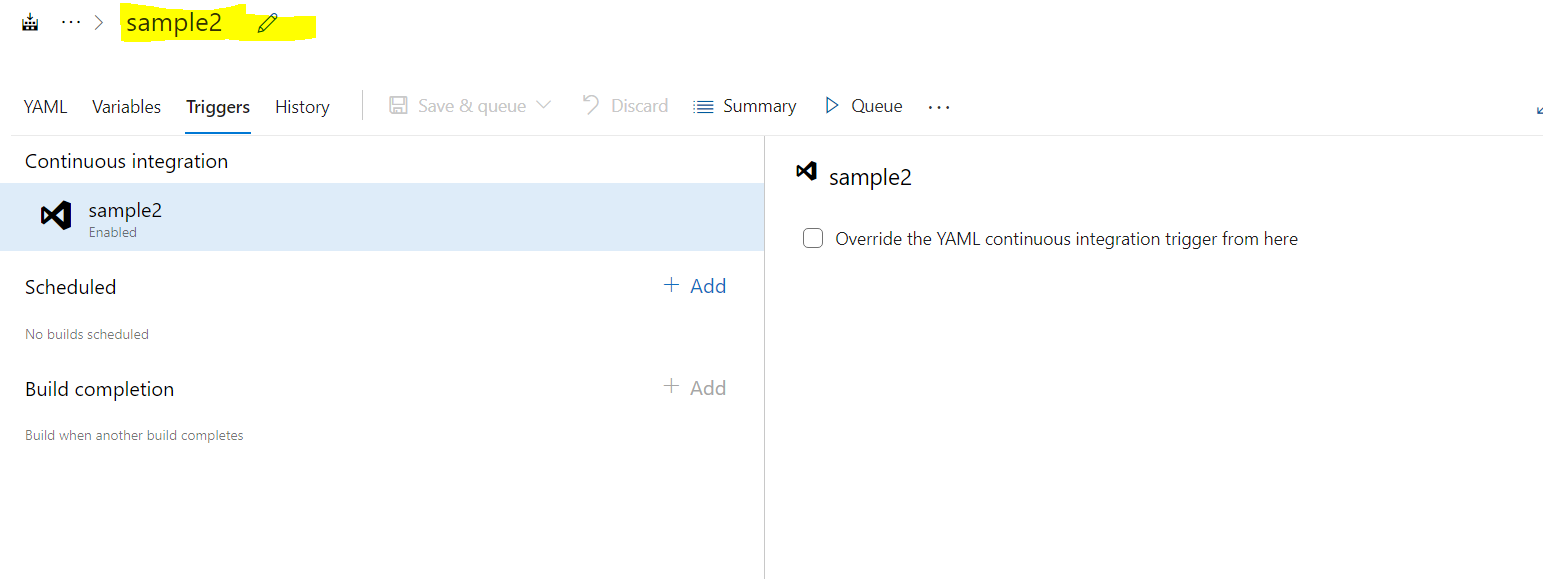
1. After saving it will take us to pipeline run page. Click on edit to go to pipeline editor.



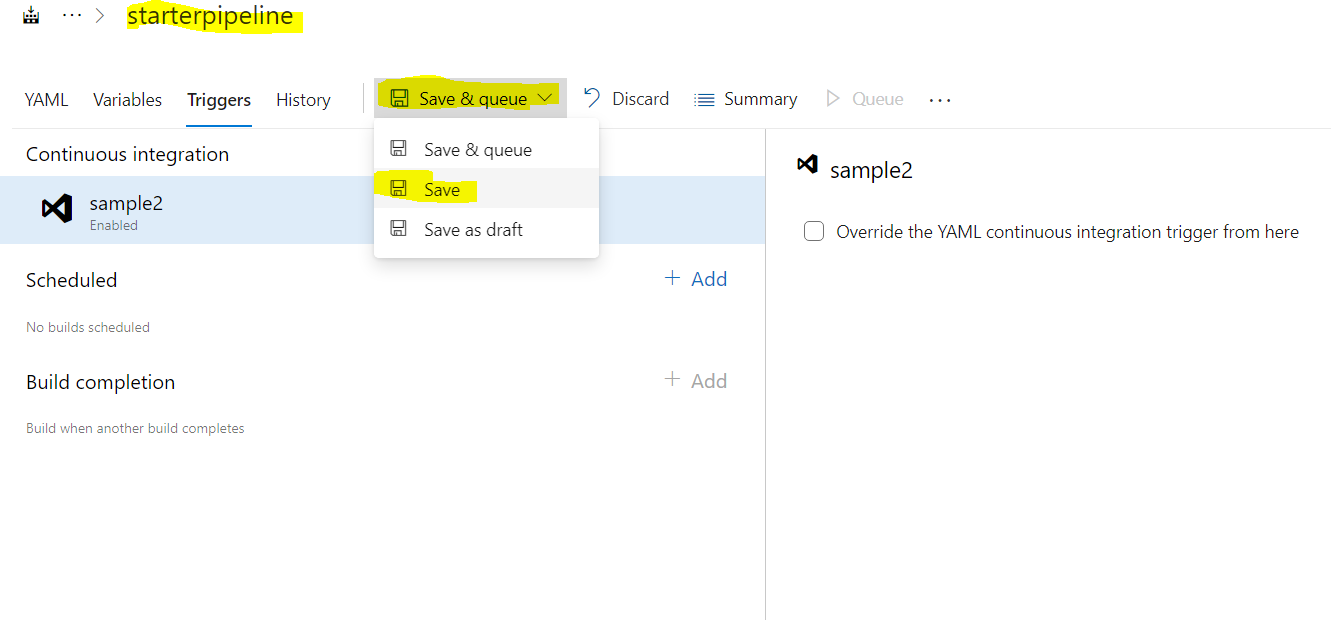
1. Click on three vertical dots. And select Triggers

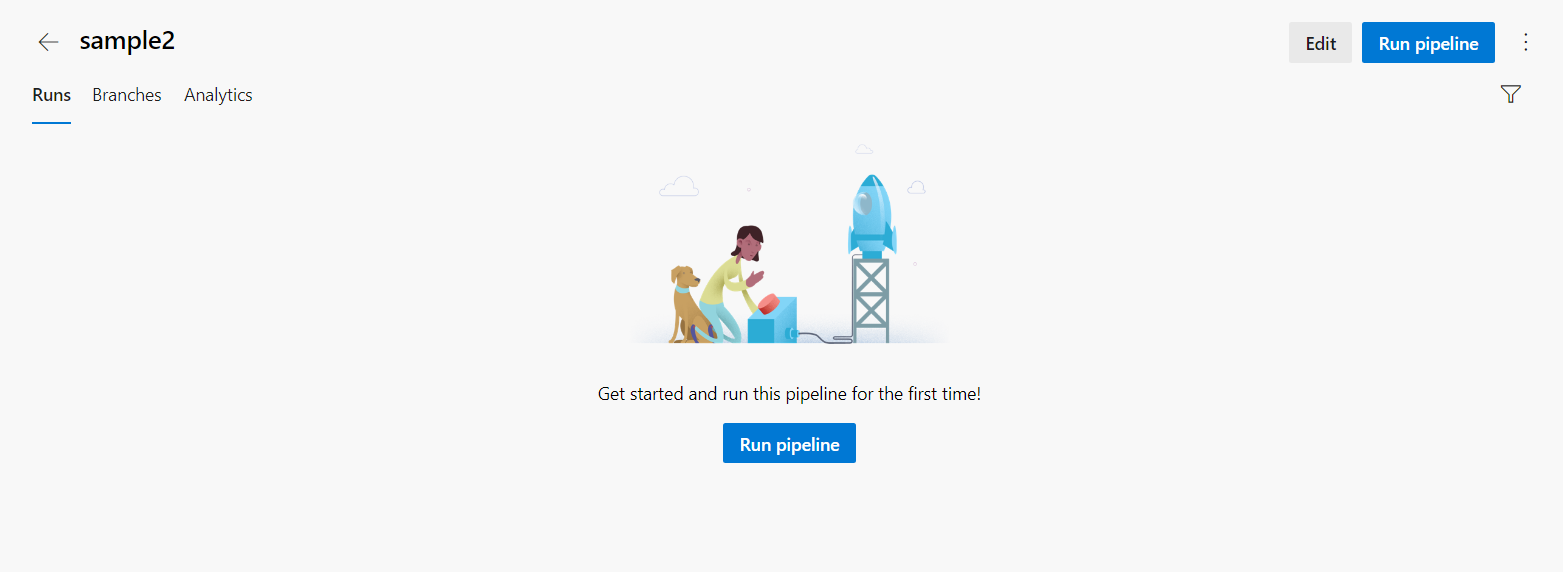


1. It will open gui editor page. Select the pipeline same then you can enter your own pipeline name what ever you want.



And then select save.





Now, click on Run Pipeline.

Here we can see the RUN logs

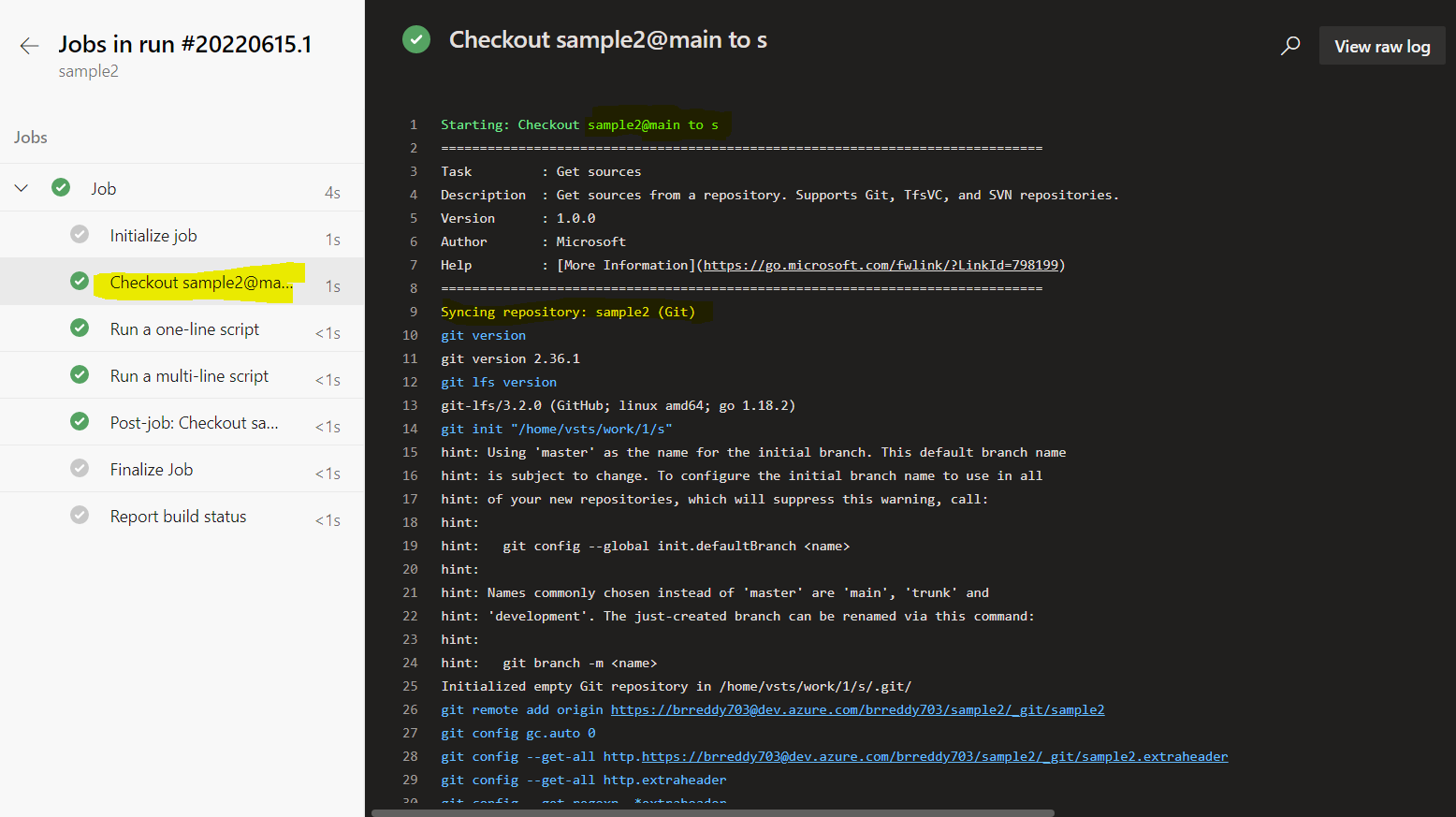
Here we can see the pool name and agent name as well.



In Initialize job we can see the Agent details on which pipeline tasks are executing.



In checkout step, files in main branch of Repository will be synced to agent machine folder.



In Summary page of pipeline, we can see artifact files if any generated.

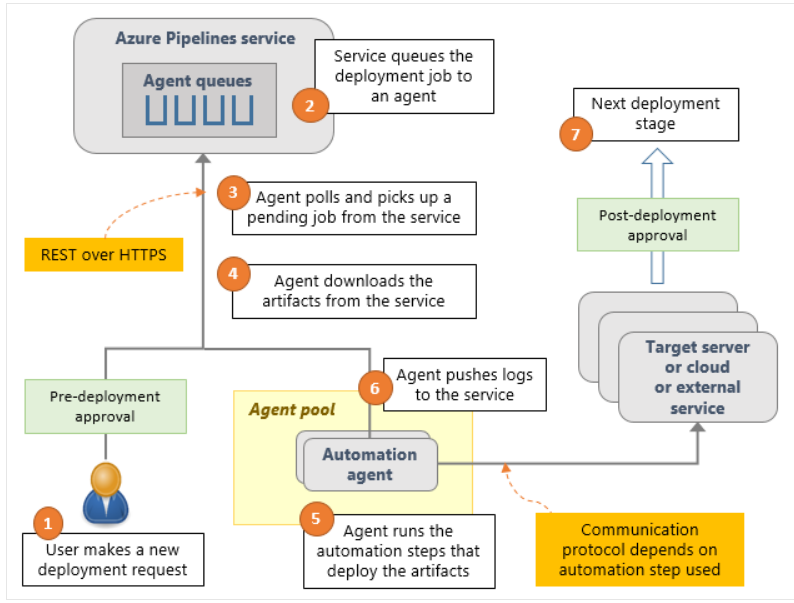
This is the completed process of creating, running and monitor pipelines.

## **Creating Release Pipeline:**

**Release pipelines** in Azure Pipelines help your team **continuously deliver** software to your customers at a faster pace and with lower risk. You can **fully automate** the testing and delivery of your software in multiple stages all the way to production. Or, set up semi-automated processes with **approvals** and **on-demand deployments**.

### **How do release pipelines work?**

Release pipelines store the data for your pipelines, stages, tasks, releases, and deployments in Azure Pipelines.

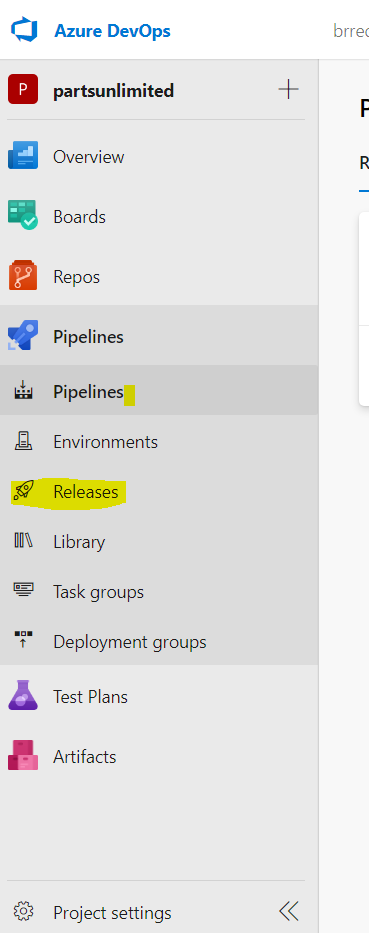


Azure Pipelines runs the following steps as part of every deployment:

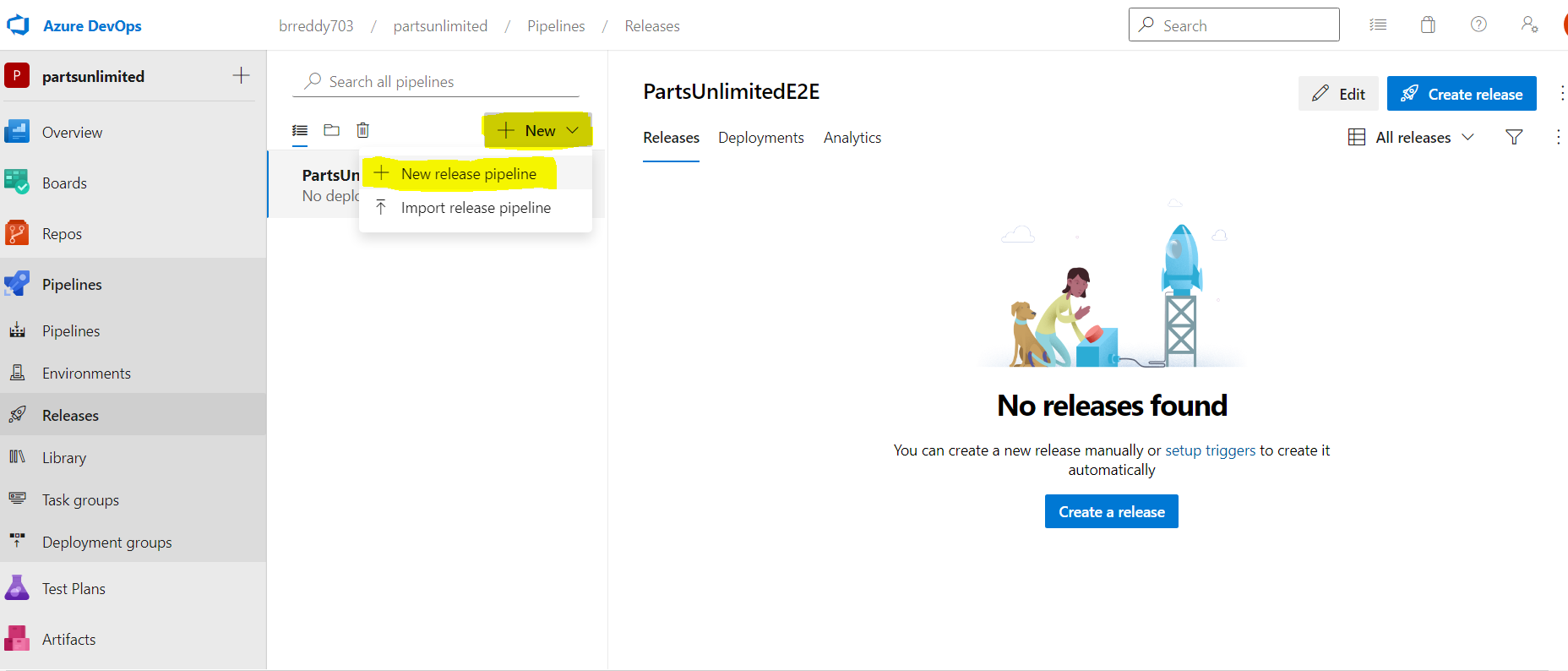
1. **Pre-deployment approval**: When a new deployment request is triggered, Azure Pipelines checks whether a pre-deployment approval is required before deploying a release to a stage. If it's required, it sends out email notifications to the appropriate approvers.
2. **Queue deployment job**: Azure Pipelines schedules the deployment job on an available [automation agent](https://docs.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops). An agent is a piece of software that can run tasks in the deployment.
3. **Agent selection**: An automation agent picks up the job. The agents for release pipelines are exactly the same as the agents that run your builds in Azure Pipelines. A release pipeline can contain settings to select an appropriate agent at runtime.
4. **Download artifacts**: The agent downloads all the artifacts specified in that release, provided you haven't opted to skip the download. The agent currently understands two types of artifacts: Azure Pipelines artifacts and Jenkins artifacts.
5. **Run the deployment tasks**: The agent then runs all the tasks in the deployment job to deploy the app to the target servers for a stage.
6. **Generate progress logs**: The agent creates detailed logs for each step while running the deployment, and pushes these logs back to Azure Pipelines.
7. **Post-deployment approval**: When deployment to a stage is complete, Azure Pipelines checks if there's a post-deployment approval required for that stage. If no approval is required, or upon completion of a required approval, it proceeds to trigger deployment to the next stage.

### **Creating Classic Release Pipeline step by step:**

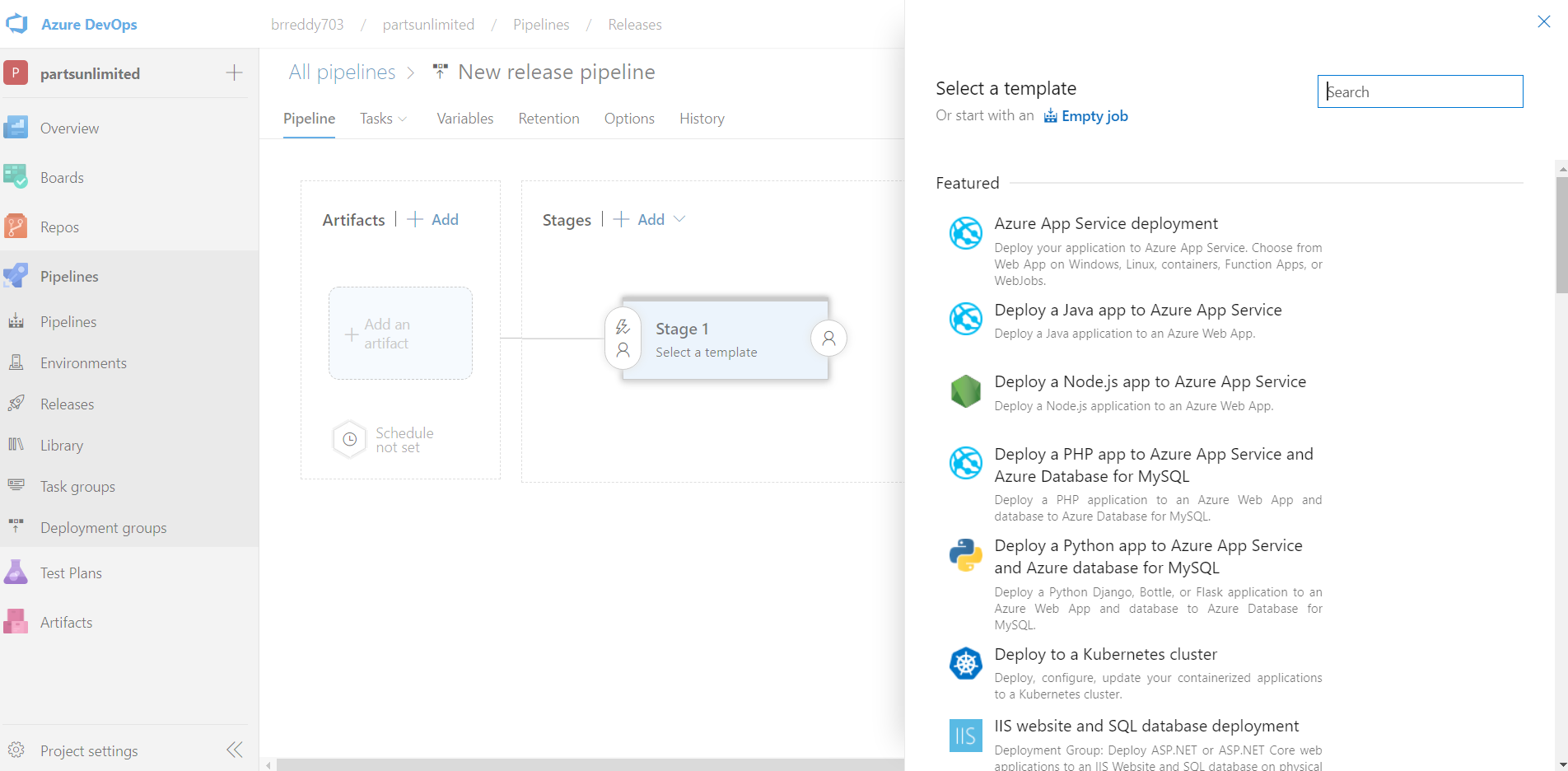
Go to Pipeline section in Azure DevOps services. Select the Release option



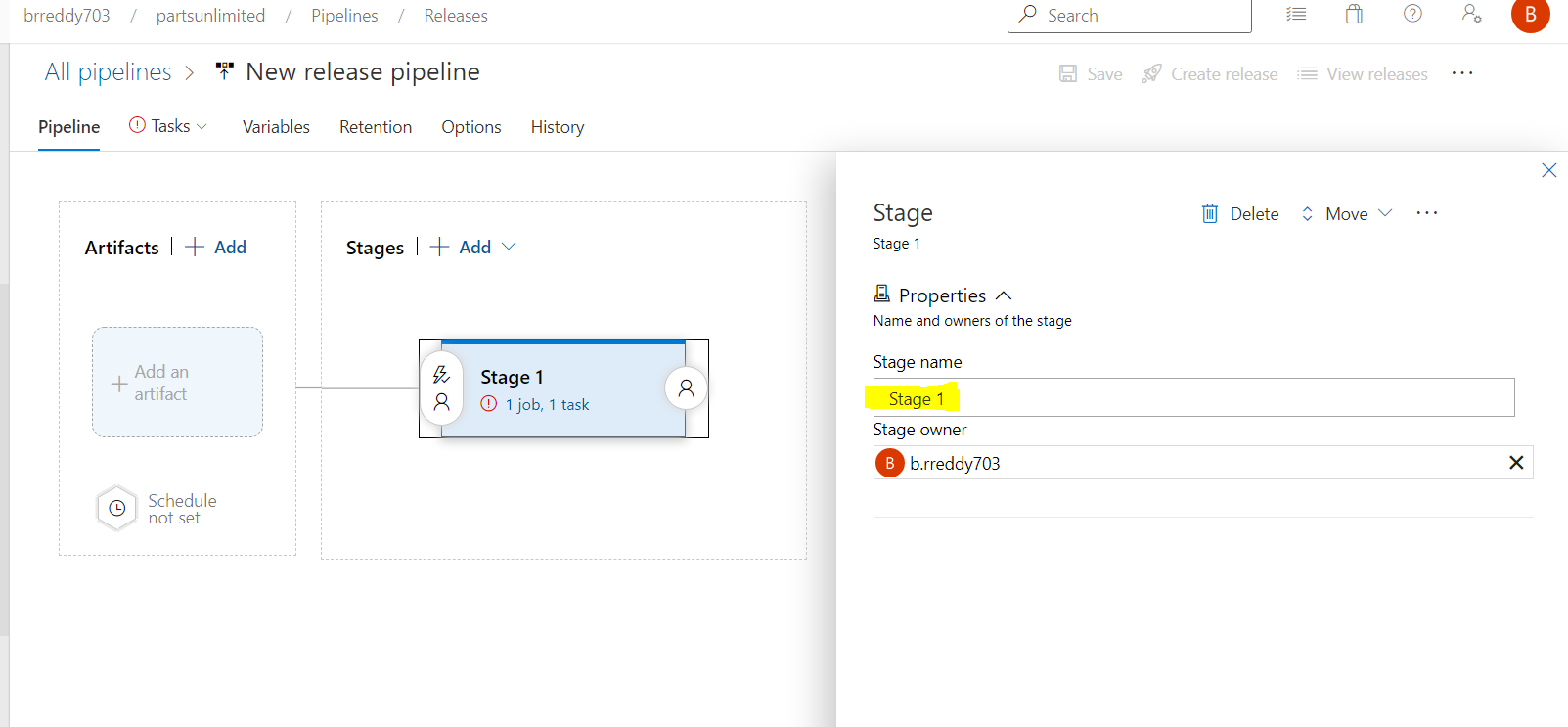
Click on New and select New release pipeline.



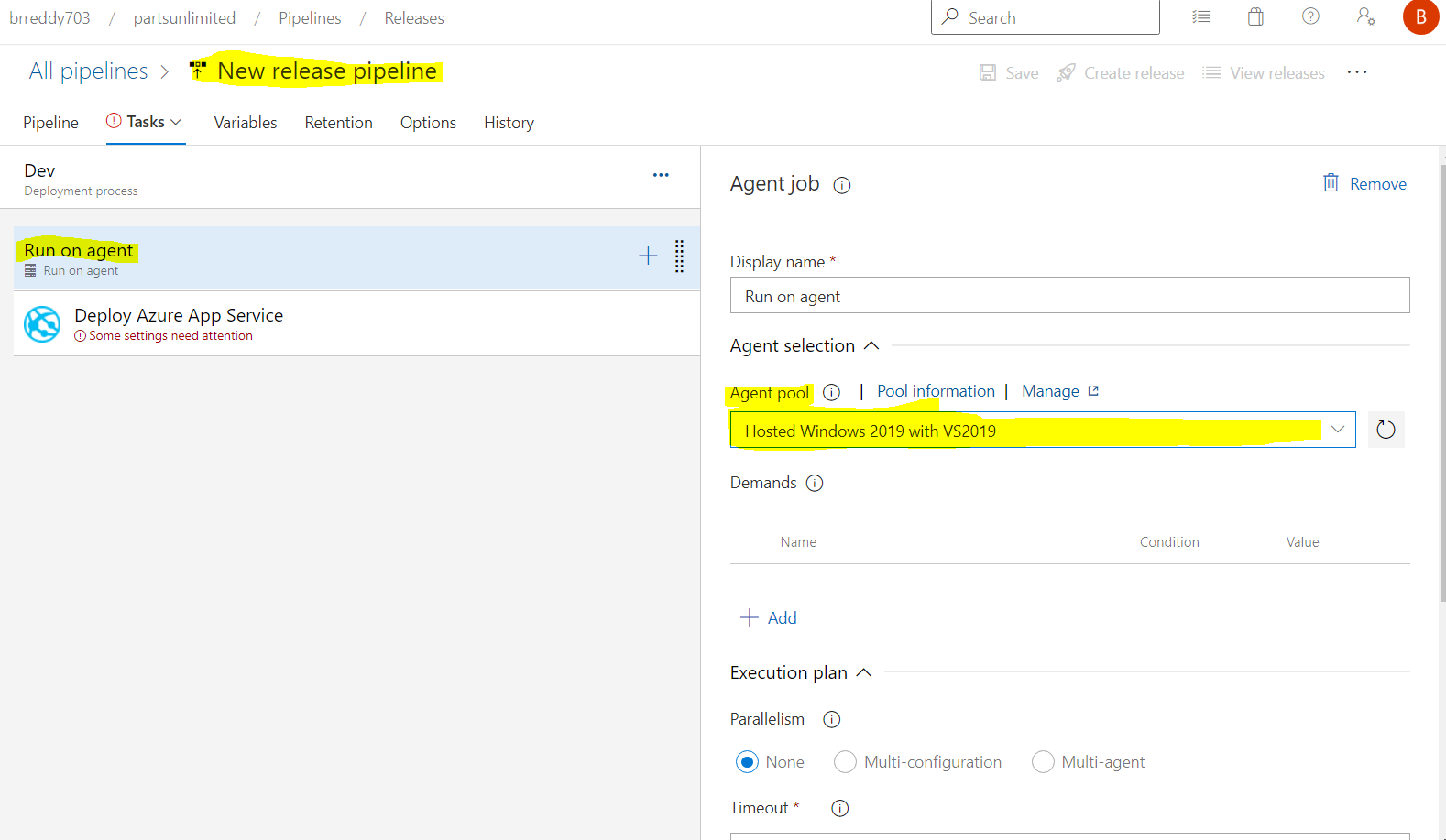
Then below screen will open



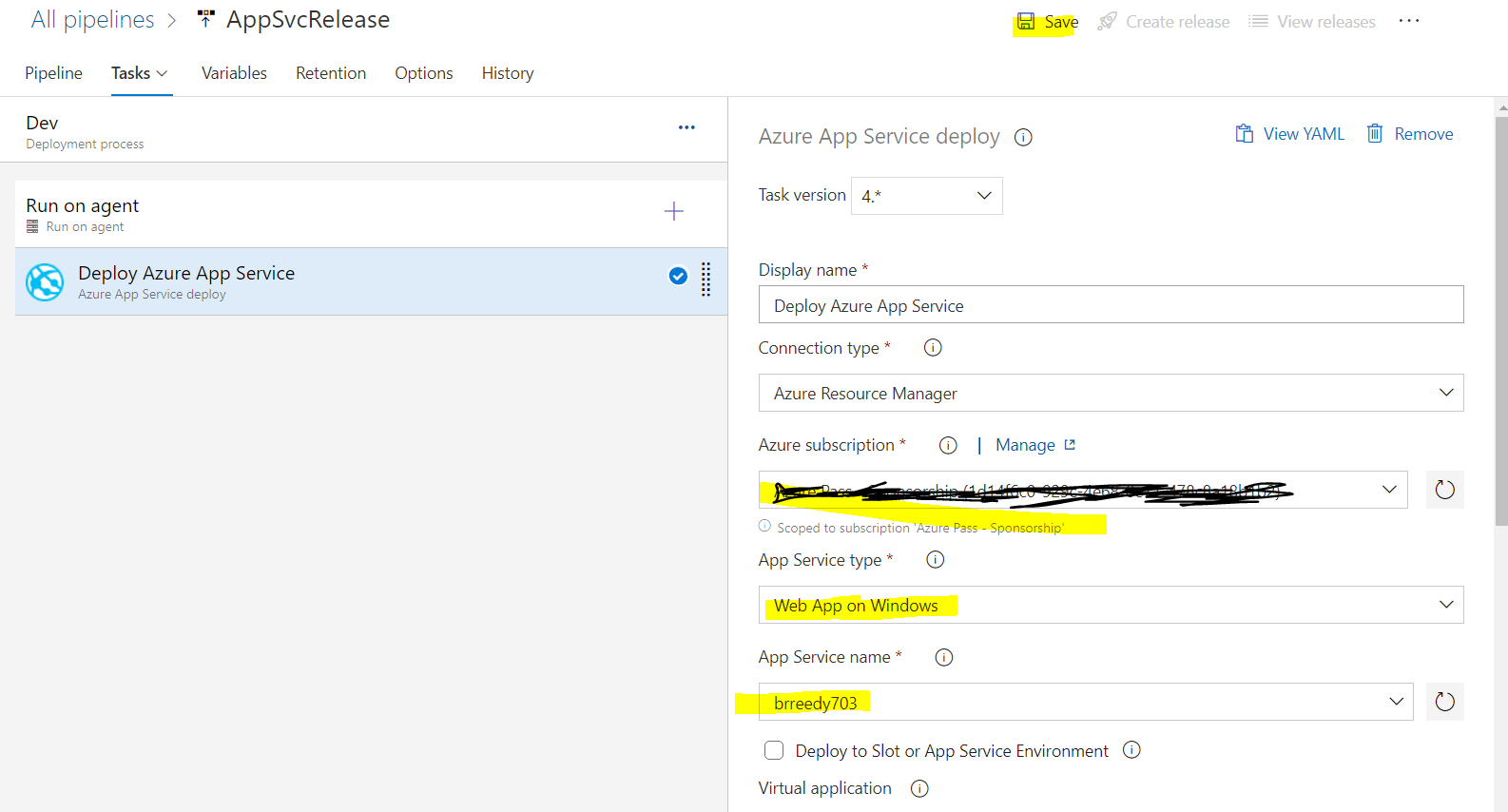
Select one of template under featured list. Here I selected Azure Appservice deployment and then click on apply.you can change the stage name what ever you want.



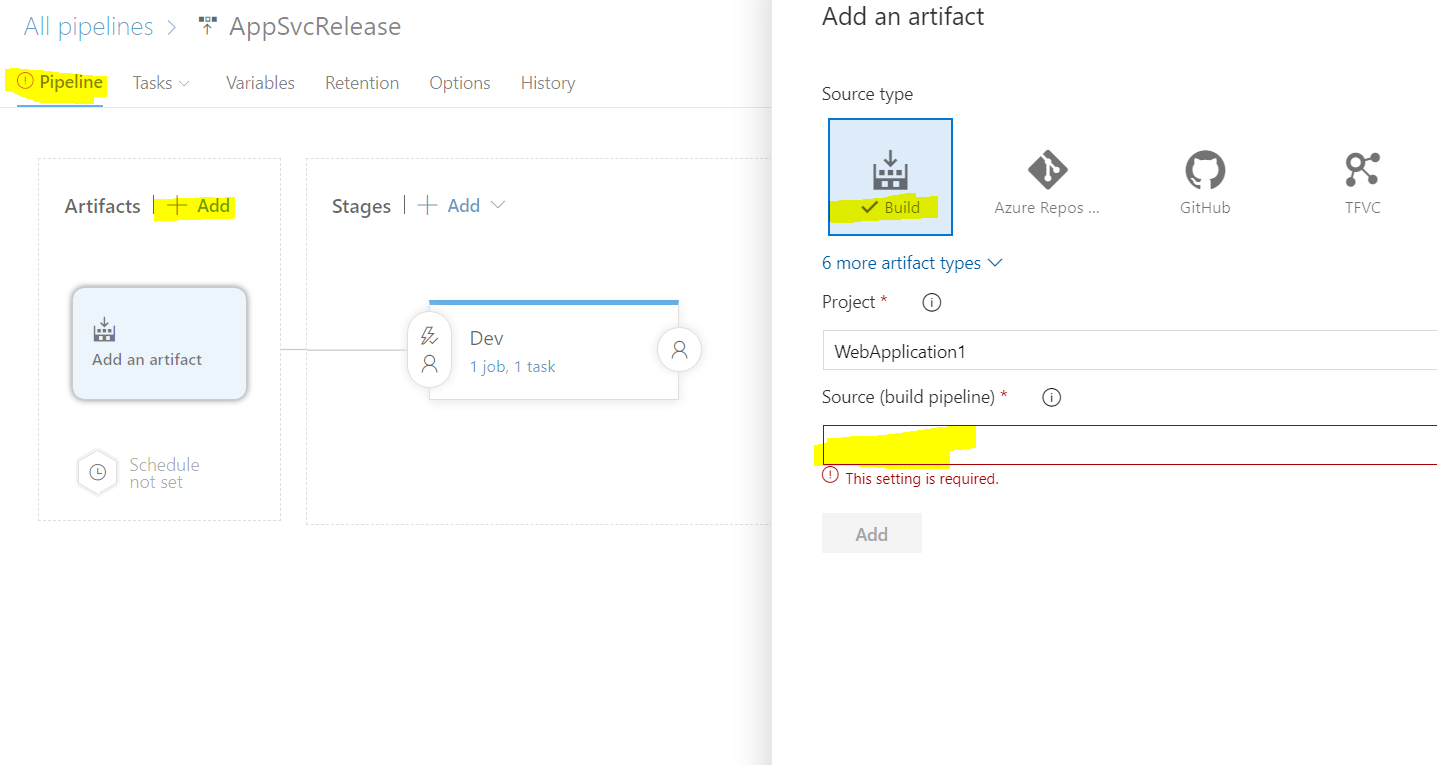
Select the agent from the agent pool and give the release pipline name

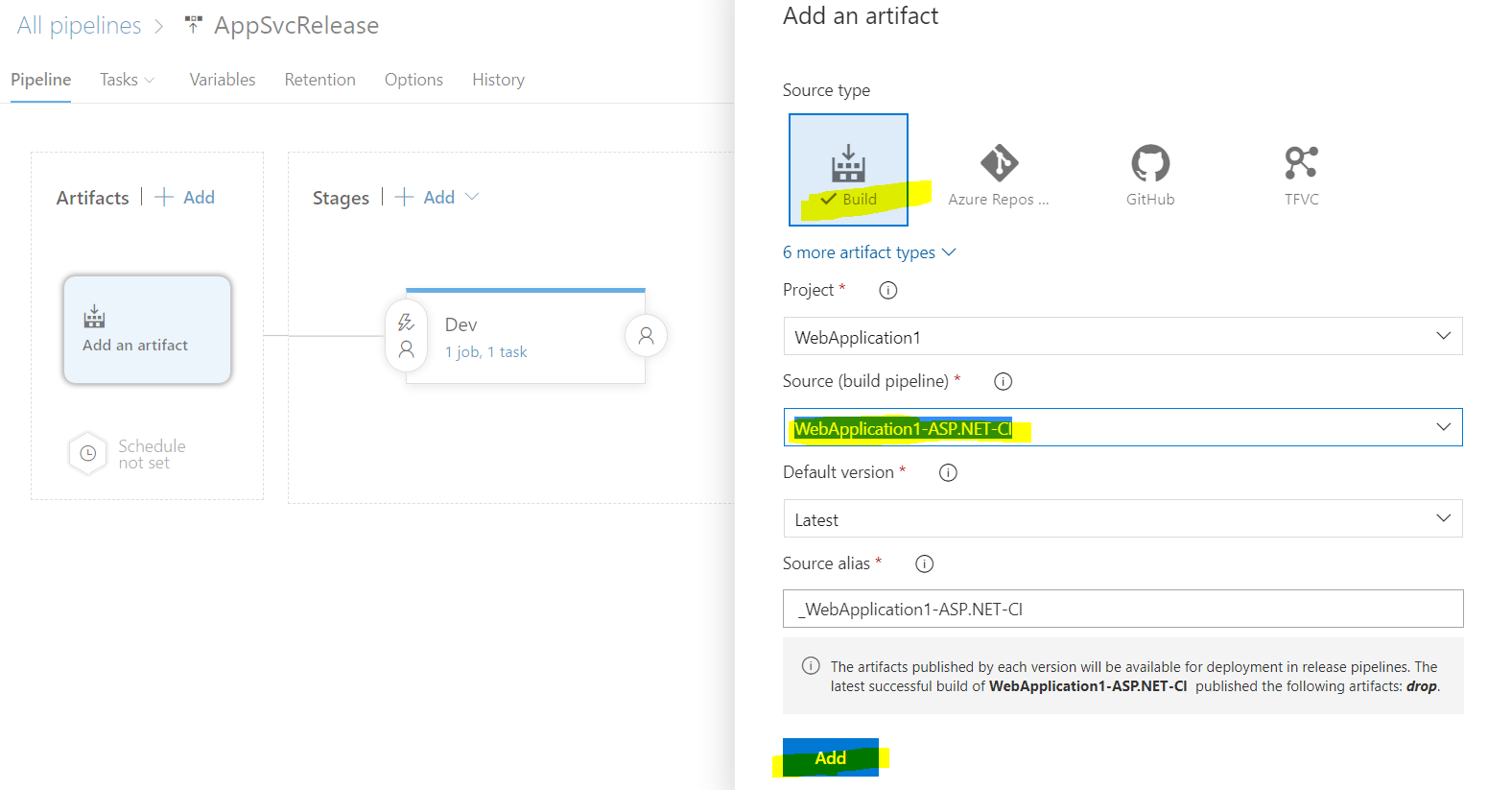


Select the Deploy Azure App Service task and select the Azure Subscription,app service type and name of the app service. Finally click on Save

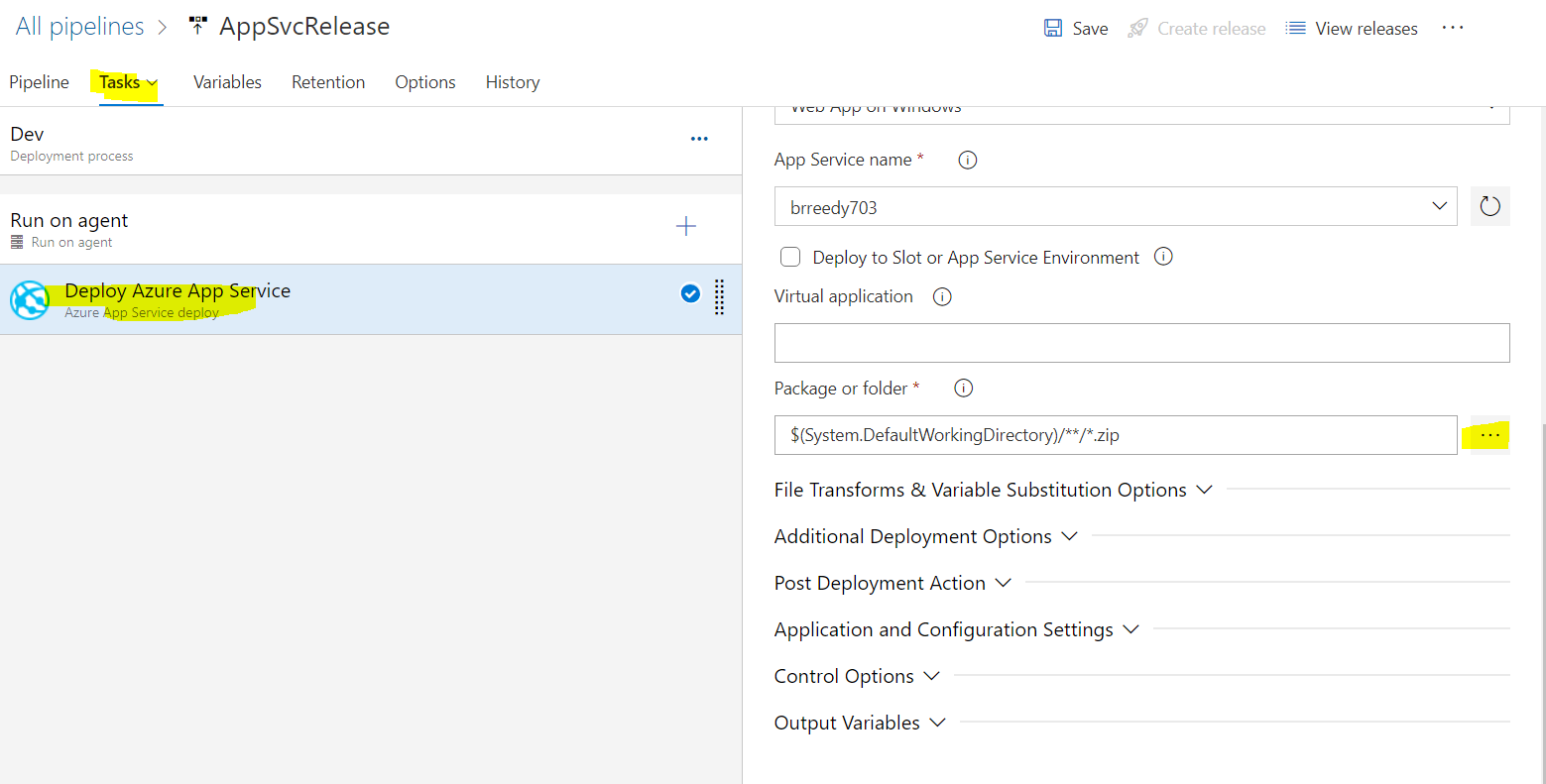


Now need to add artifacts which are generated by Build pipeline.

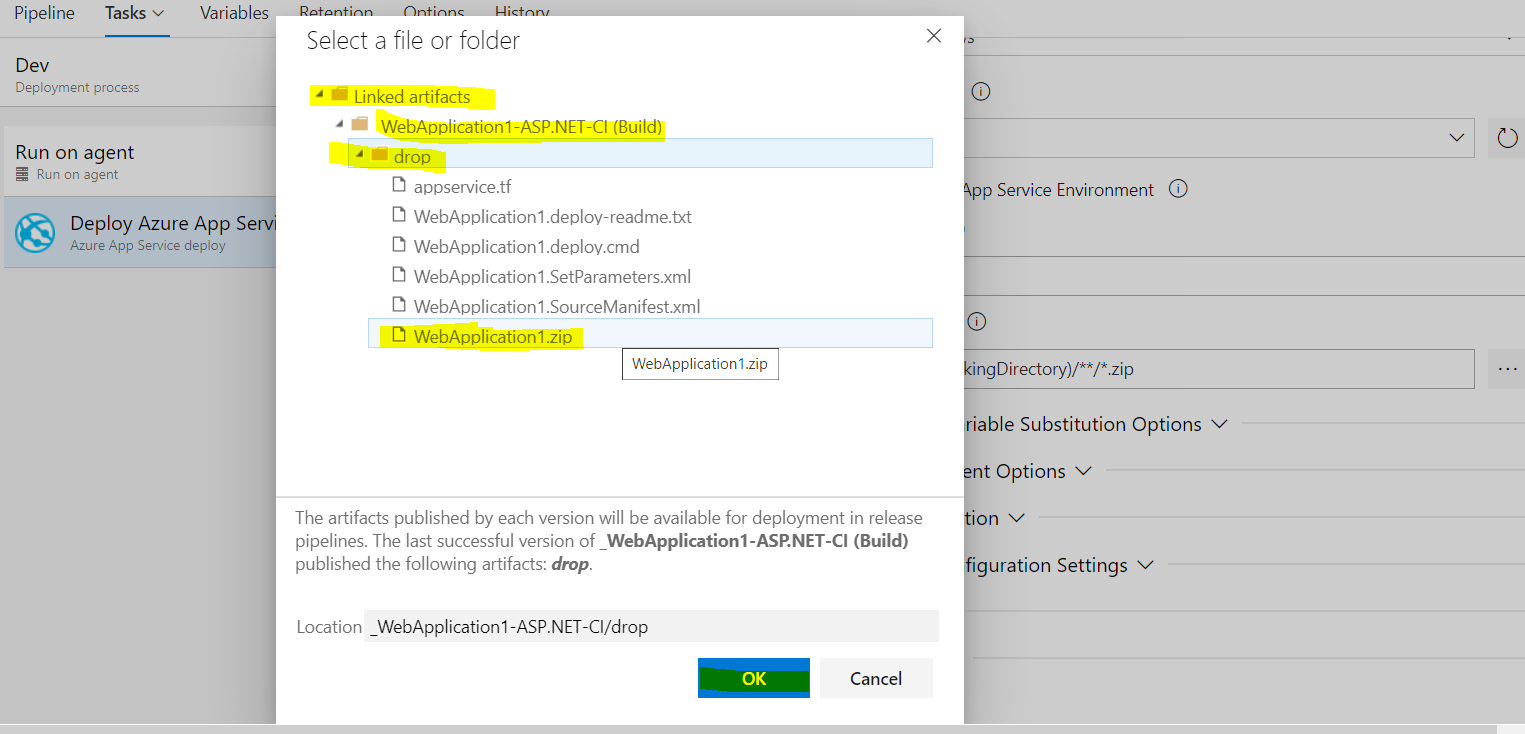




After adding come back to Task and select the depploy azure app servcie, selecet the folder of packages which need to be deployed.



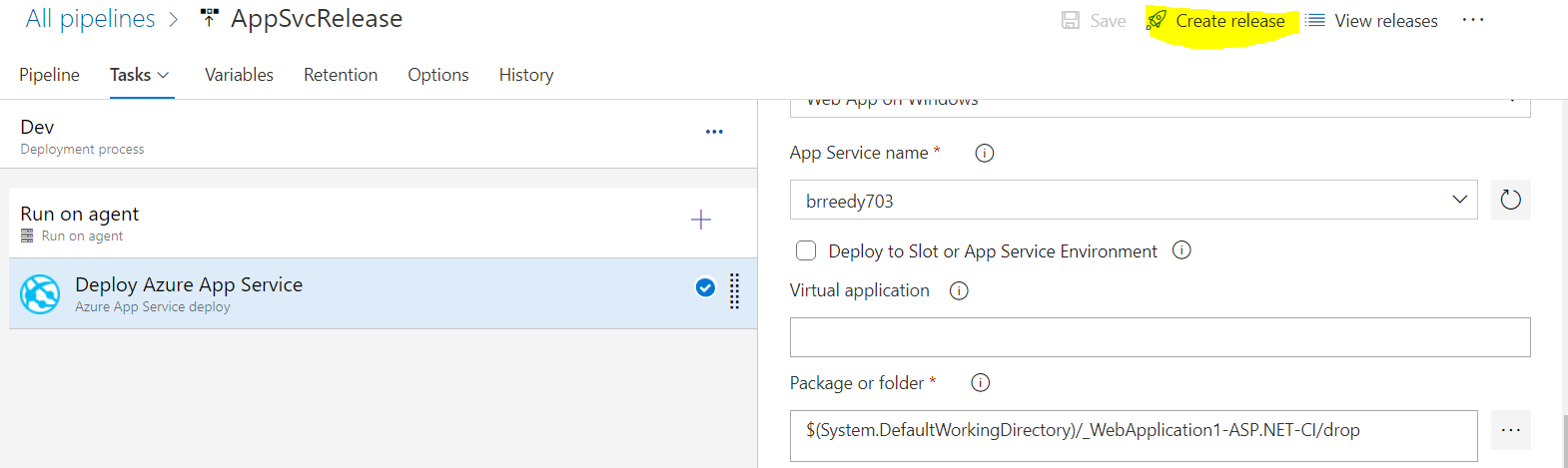
Select the package and click on OK.



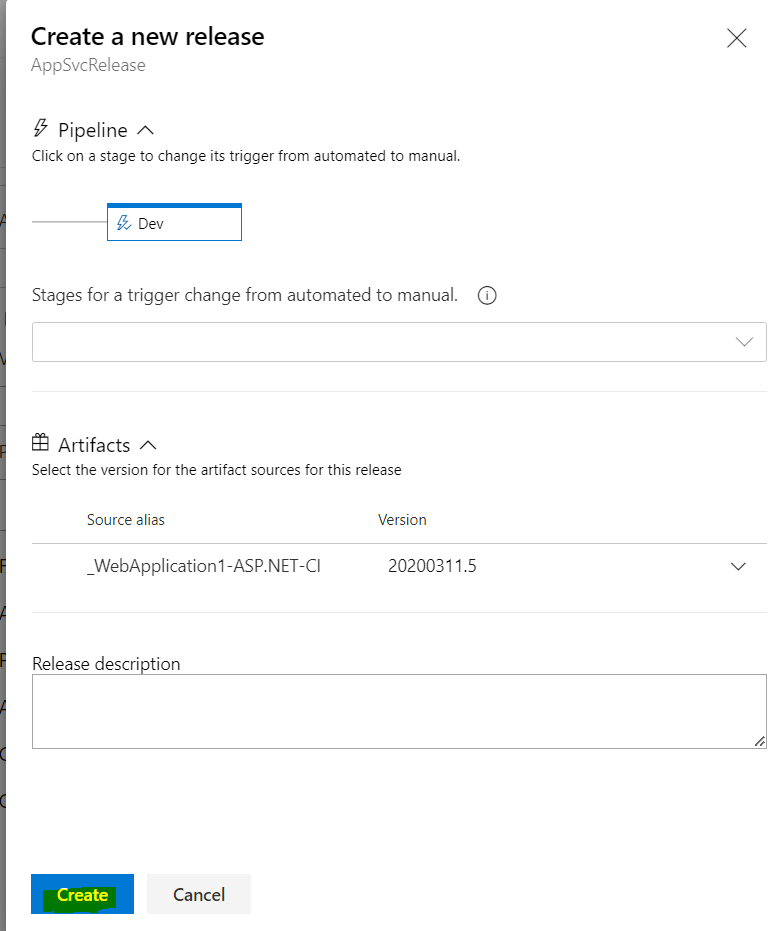
After adding click on Save and enter the commit message.

Now pipeline is ready, now we need to create release to deploy the packages.

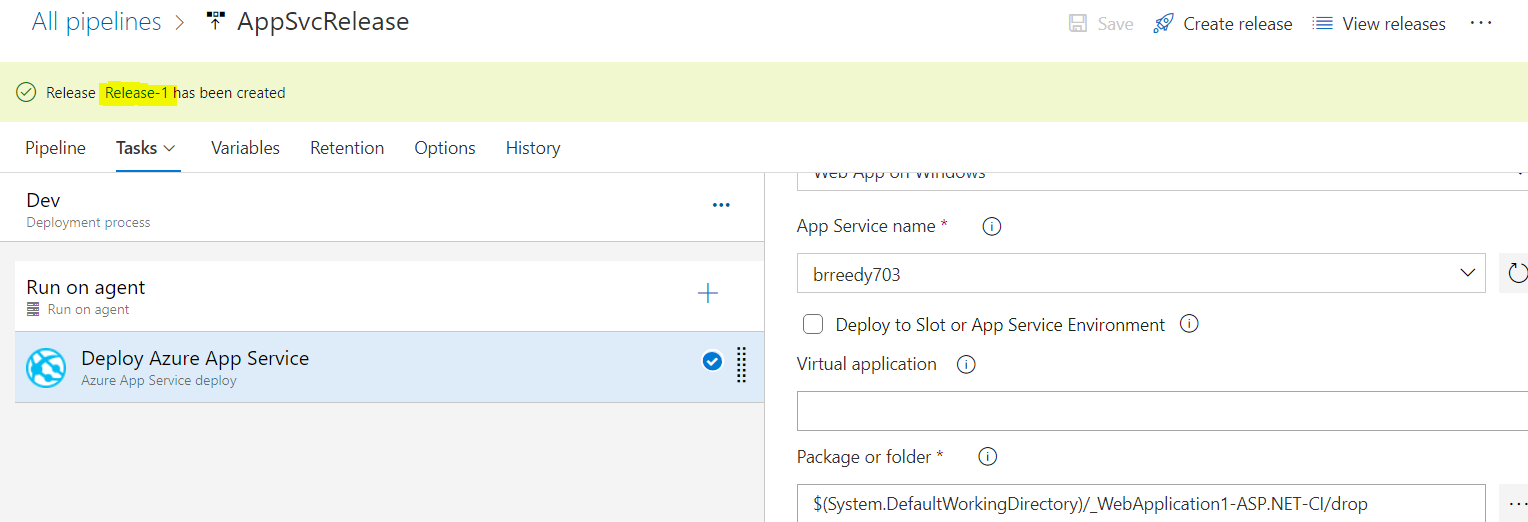
For that click on Release.



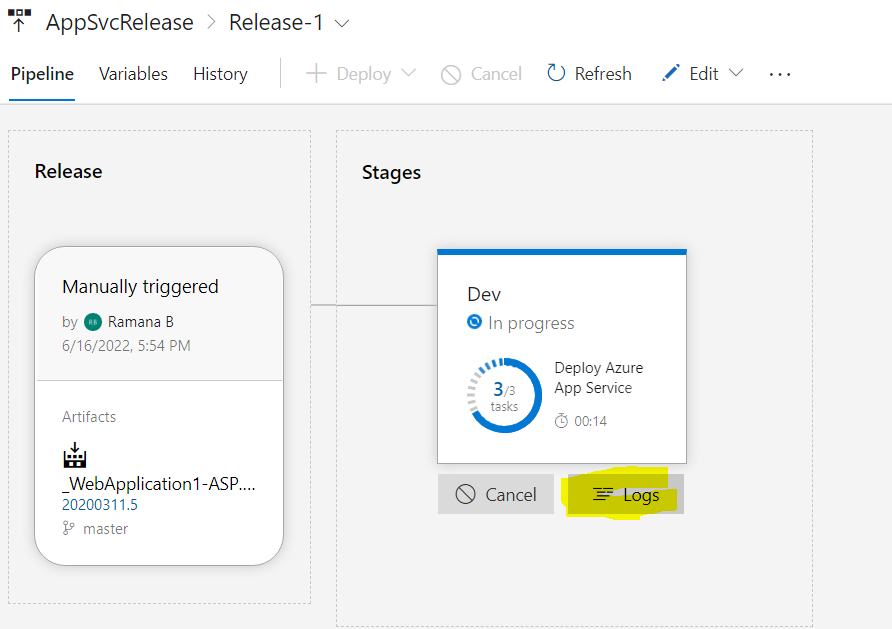
Verify the details and Create.

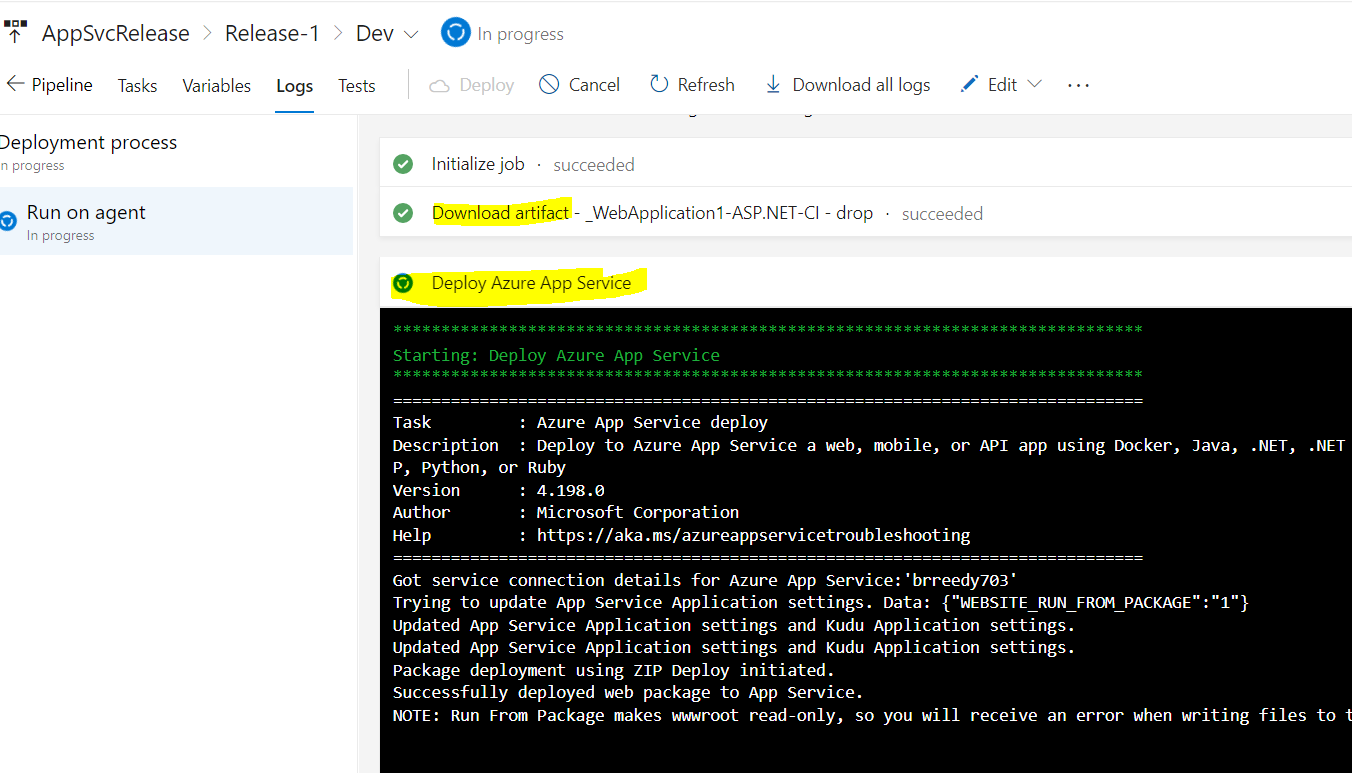


After Create, release pipeline will be running.

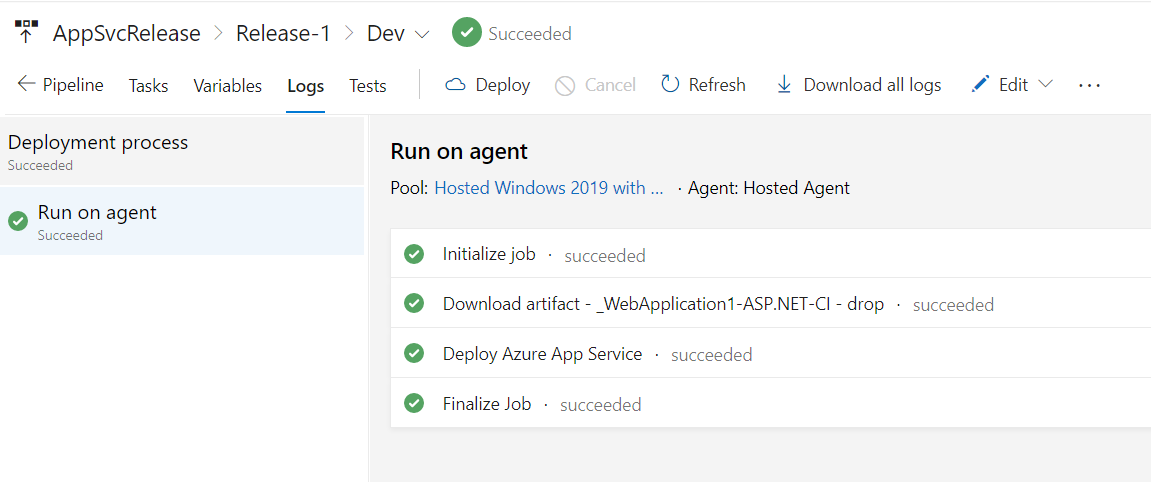


Click on logs to view the run logs.

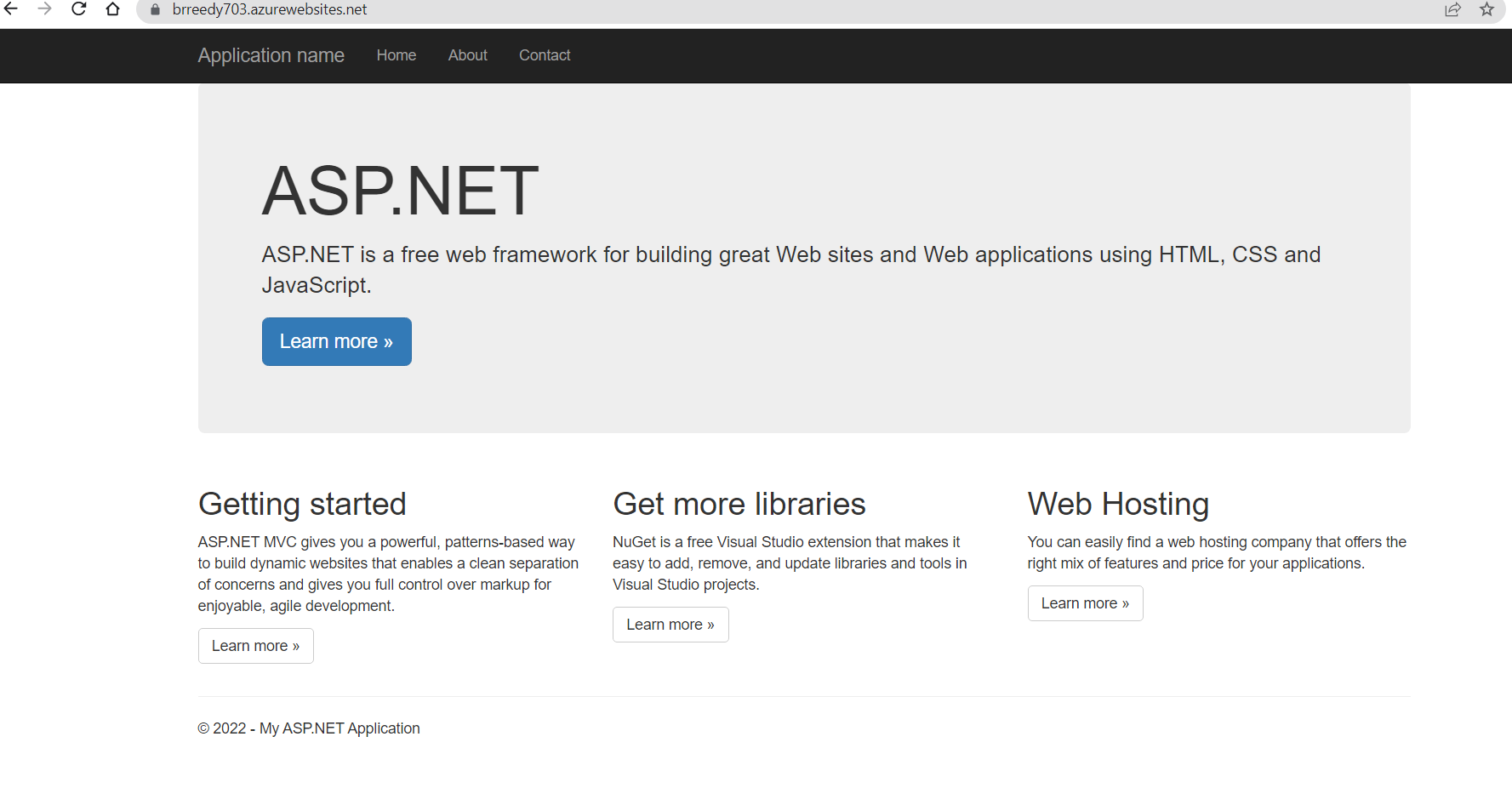




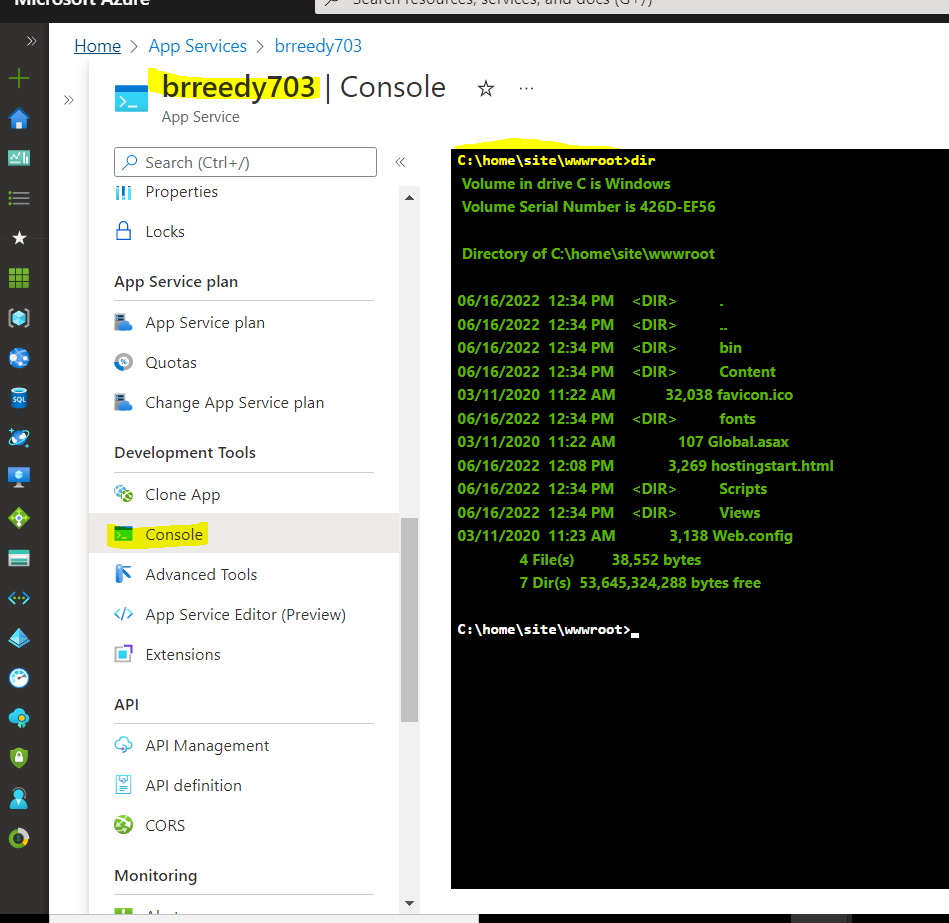
Release pipeline is successfully completed.



To verify the changes in app service browse the app service in azure.



To check the files been deployed app service you can go to app service🡪console and run dir command, as I selected windows app service.



## **End to End YAML pipeline for Application build and deployment**

### **Project Overview**

Once downloaded you can open the SmartHotel360.WebSite.sln file, located in the Source folder to open the solution in Visual Studio 2017 15.5 or higher. This solution contains two projects:

* The SmartHotel360.Website project: An [ASP.NET Core](https://github.com/brreddy7/SmartHotel360-Website/blob/master/www.dot.net) website which is a web app developed using React+Redux and server-side rendering.
* The SmartHotel360.WebsiteFunction project: An [Azure Function](https://azure.microsoft.com/services/functions) used to analyze photos of pets using the Cognitive Services Vision API and Azure Cosmos DB.

### **Get the source code**

Go to below github link and download or clone the code to local machine

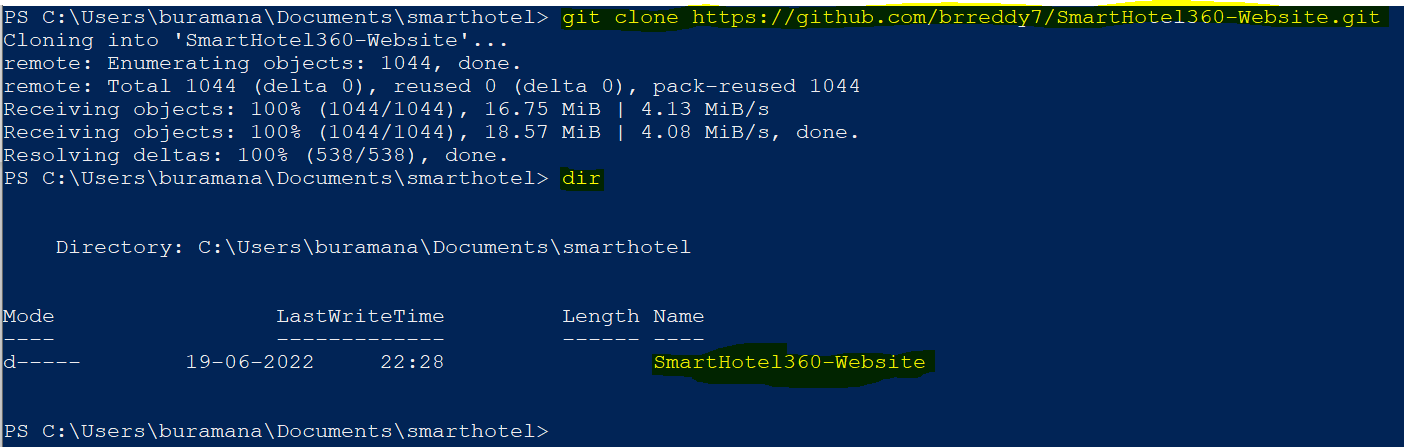
<https://github.com/brreddy7/SmartHotel360-Website>



and exapand the zip file if you downloaded. OR

Use the below command to clone

***git clone*** [***https://github.com/brreddy7/SmartHotel360-Website.git***](https://github.com/brreddy7/SmartHotel360-Website.git)



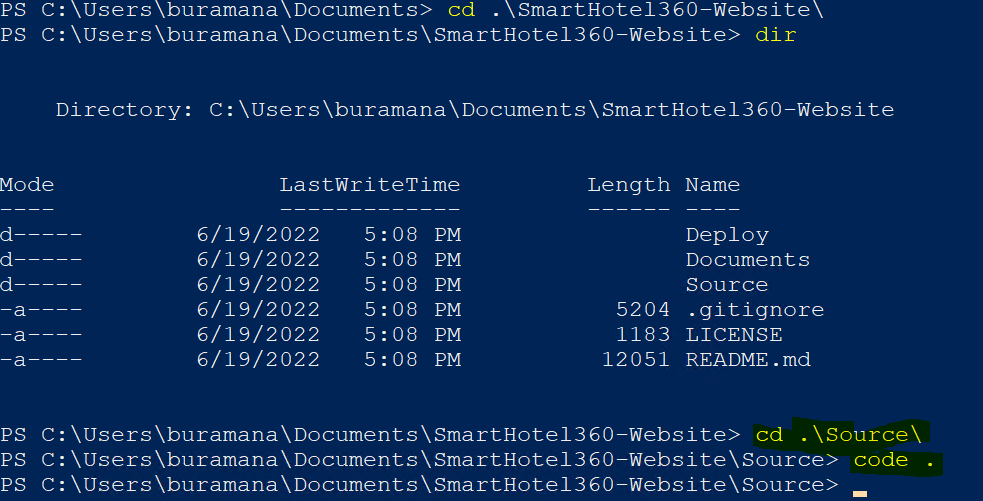
Go inside the directory and we can see the multiple folders. Source folder has the actual .net code.



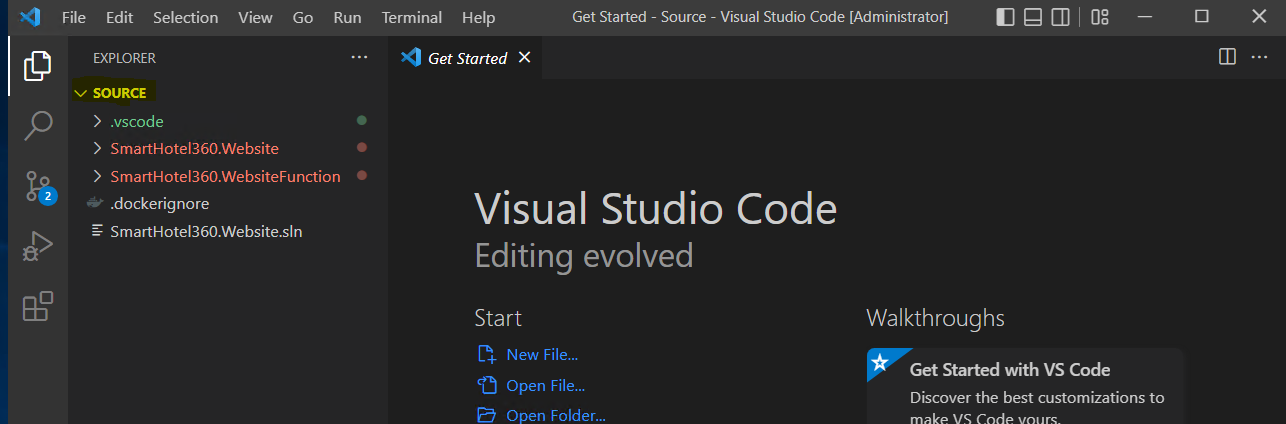
To run application in local machine. Install .net core and .net framework and visual studio code. Open the ***Source*** folder in ***vs code***.

Run - ***cd Source***

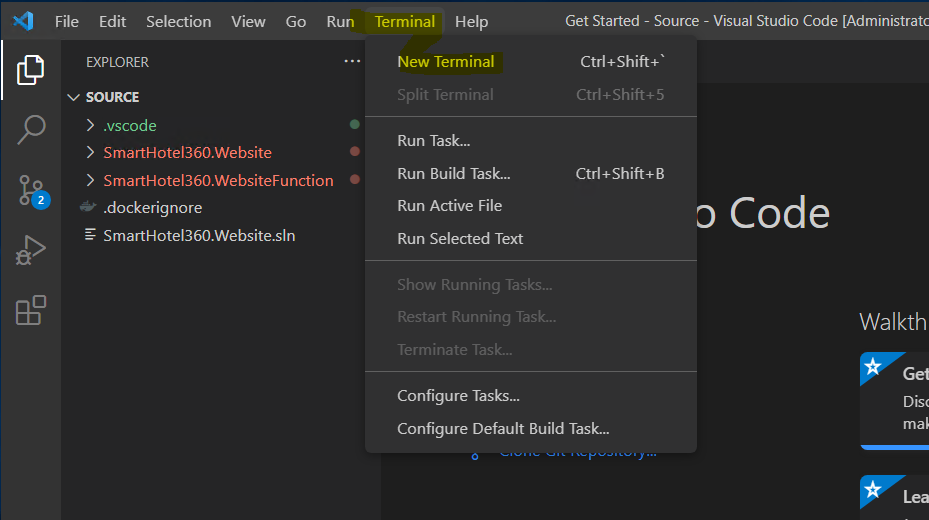
Then do - ***code .*** *#VS code should be installed*



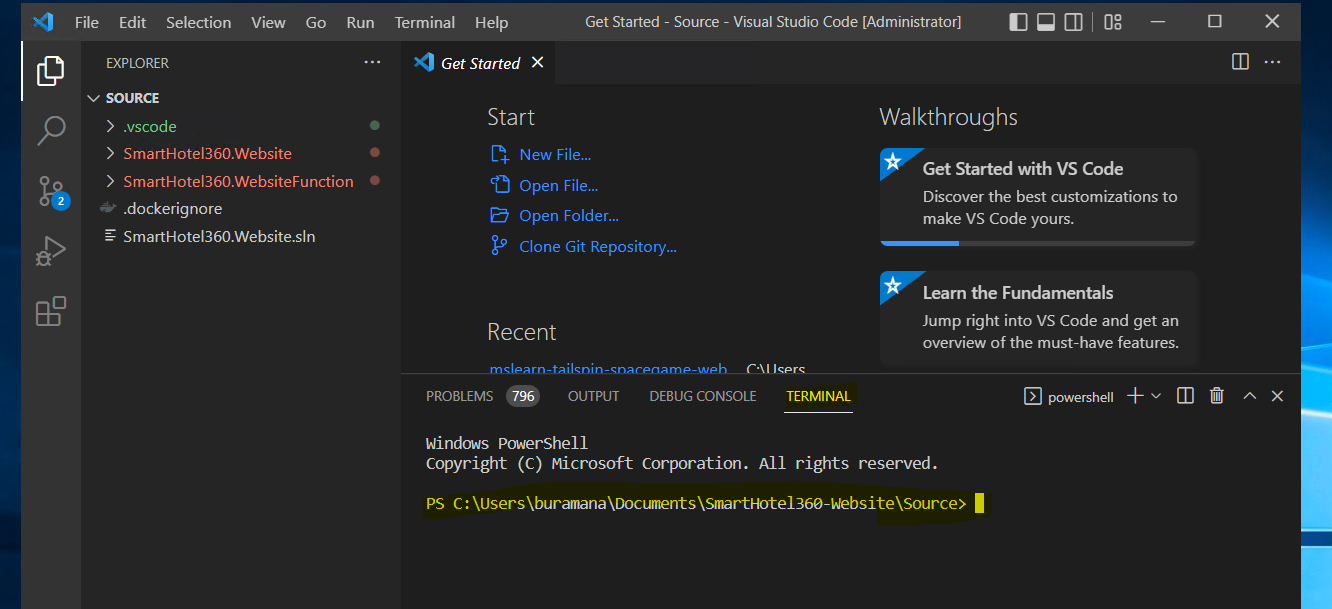
Source folder will be opened in VS Code



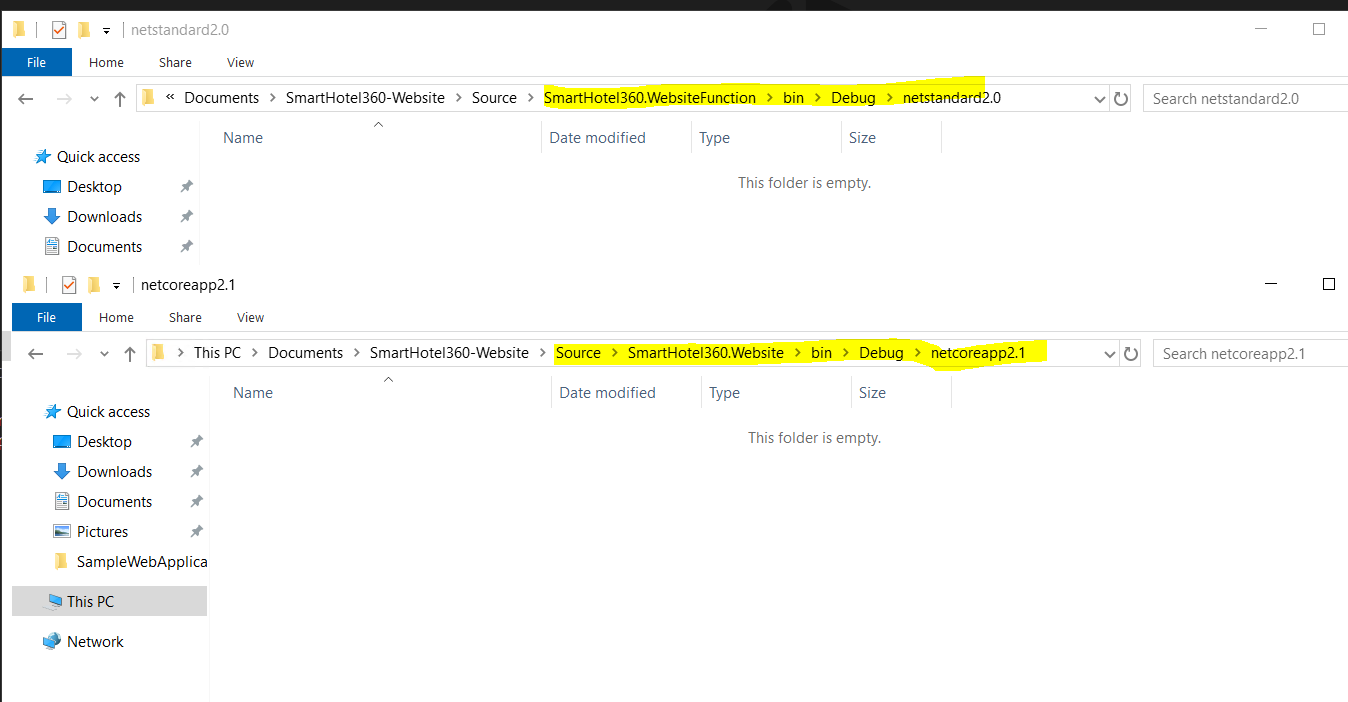
Open the terminal



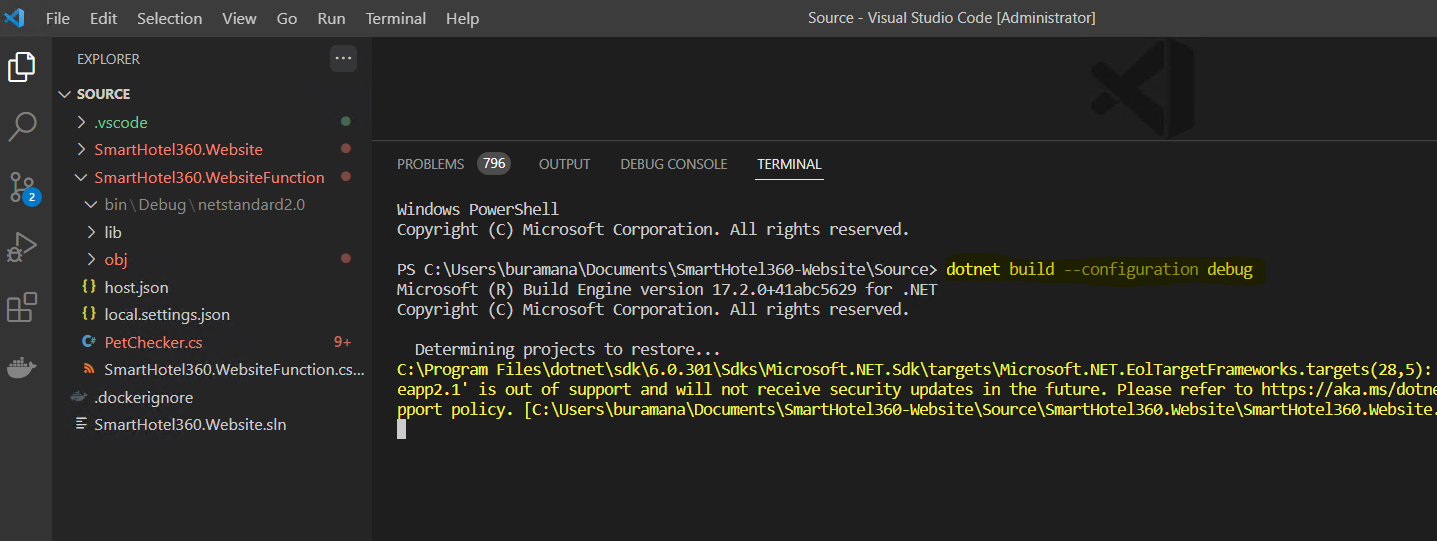
Command line will be started.



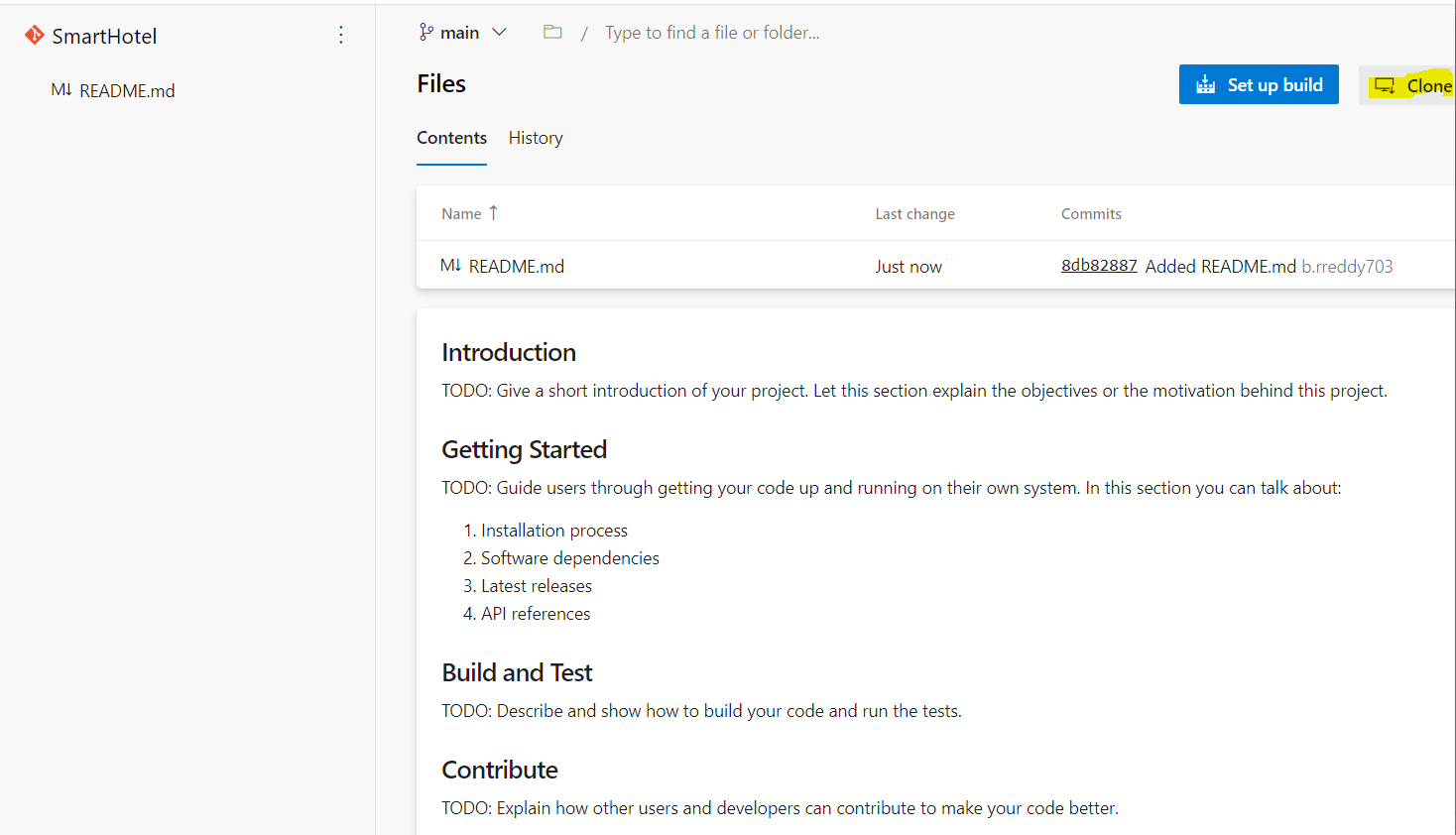
Now we need to build the .net code using ***dotnet cli*** commands. Before doing that verify are there any files in ***bin*** folder of ***SmartHotel360.website*** or ***SmartHotel360.WebsiteFunction***



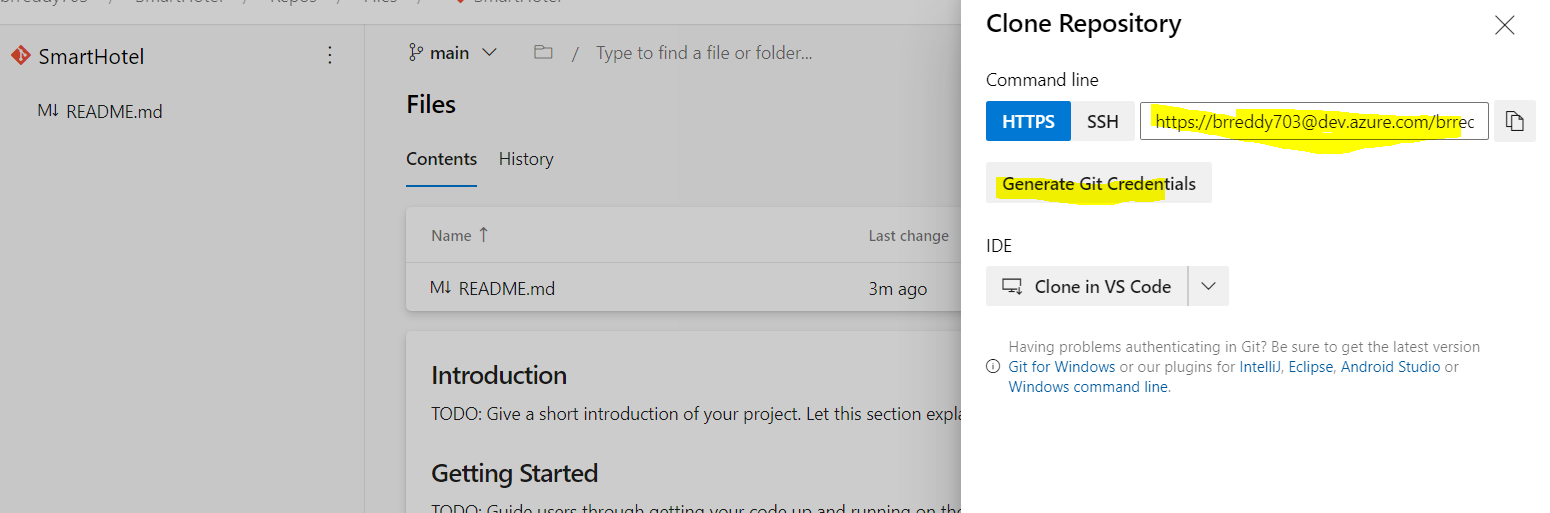
Now start build the solution using ***dotnet build –configuration debug***



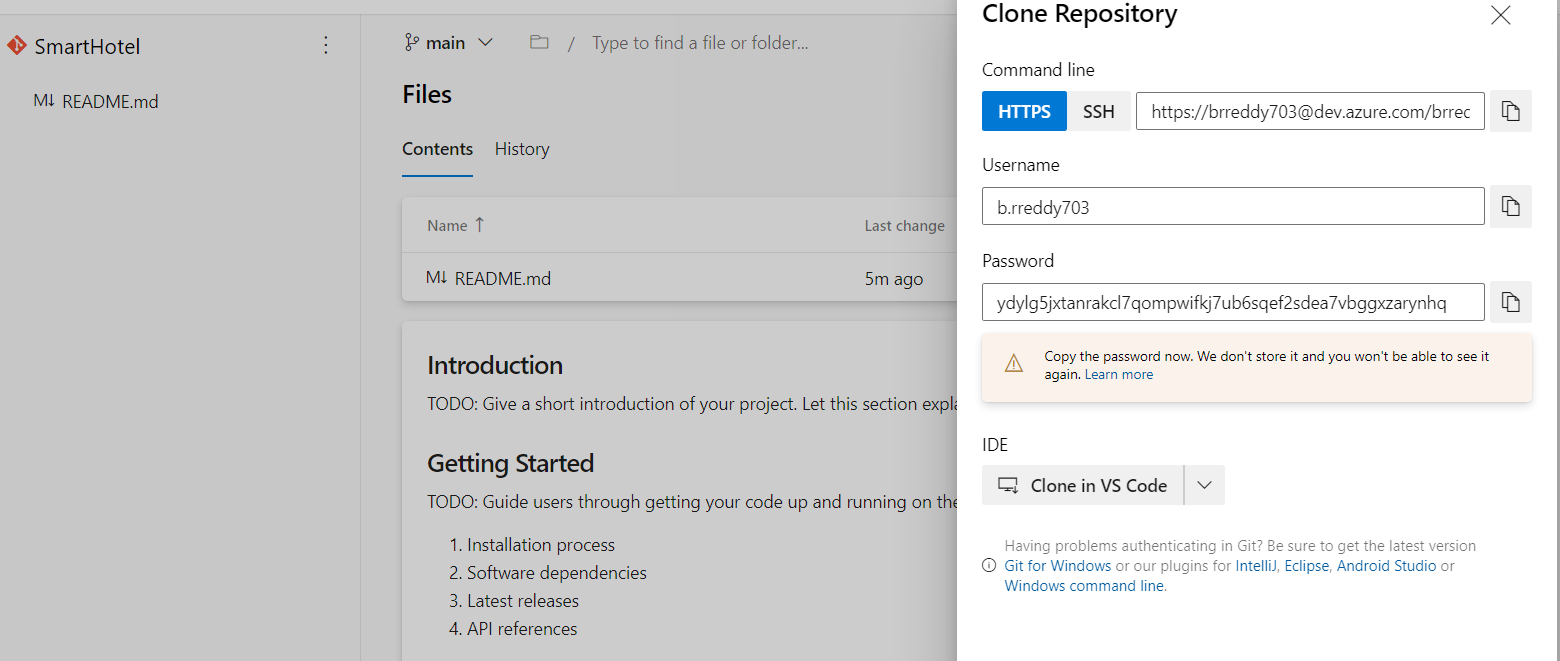
After verify the files, you can push the local repo to Azure DevOps Repos. For that should created one project and repository in azure devops.



Get the repositor url ,to get it click on clone .



Click generate git credentials, will be useful while push code to this repo.



and run the below commands in local machine.

***#****To add remote repo to local. Upstream is alias name of remote repo.*

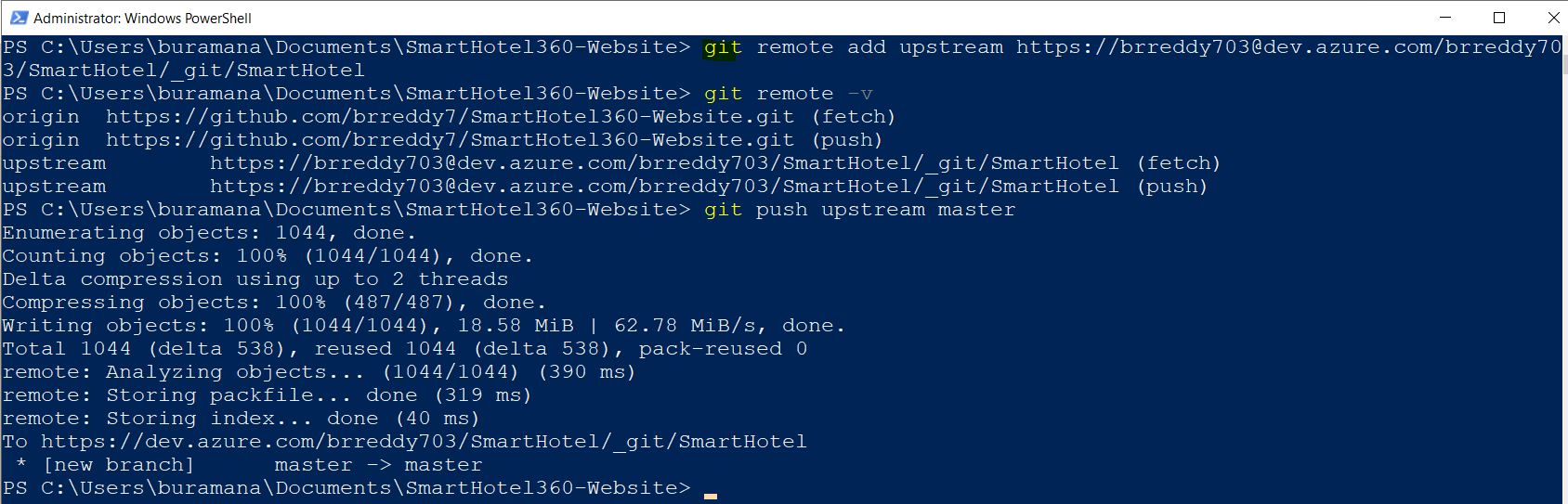
***git remote add upstream <git url>***

*#To check whether remote repo being added or not*

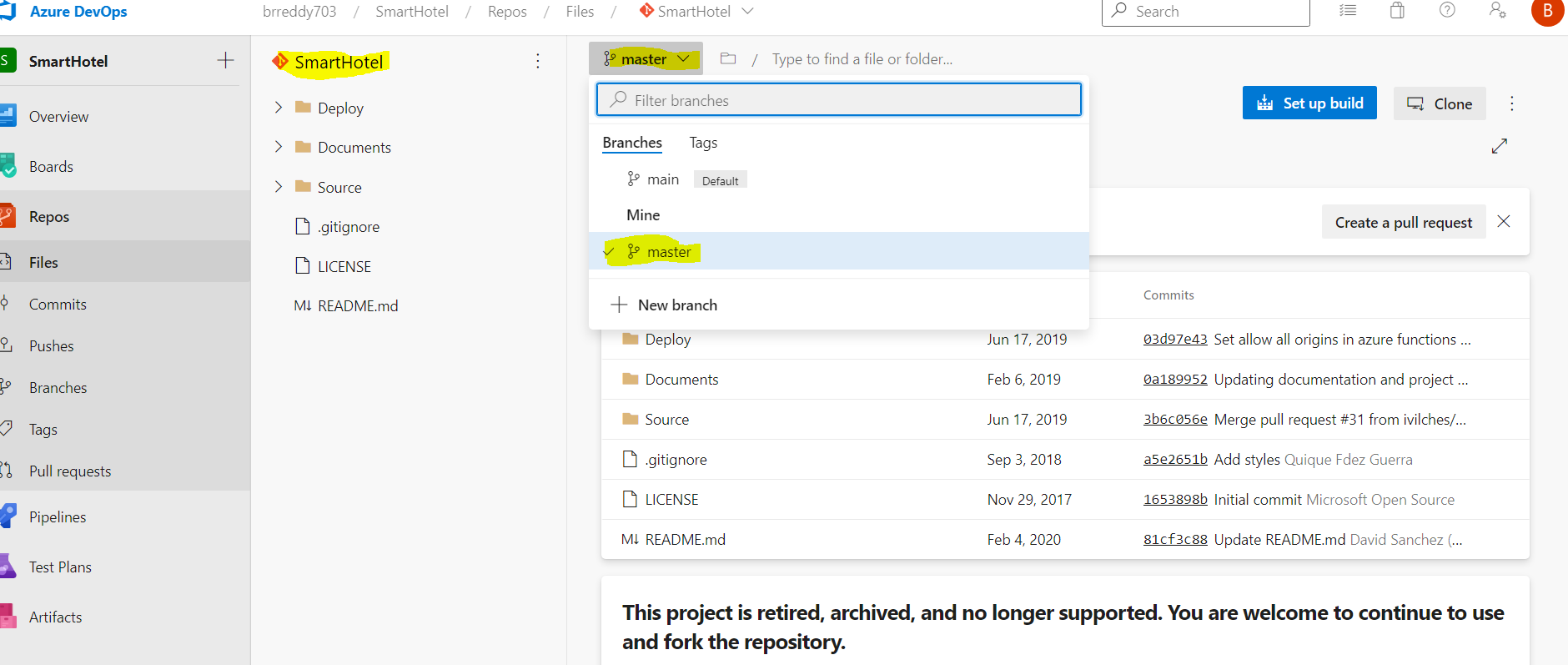
***git remote -v***

*#To push to Remote or central repo from local*

***git push upstream master***



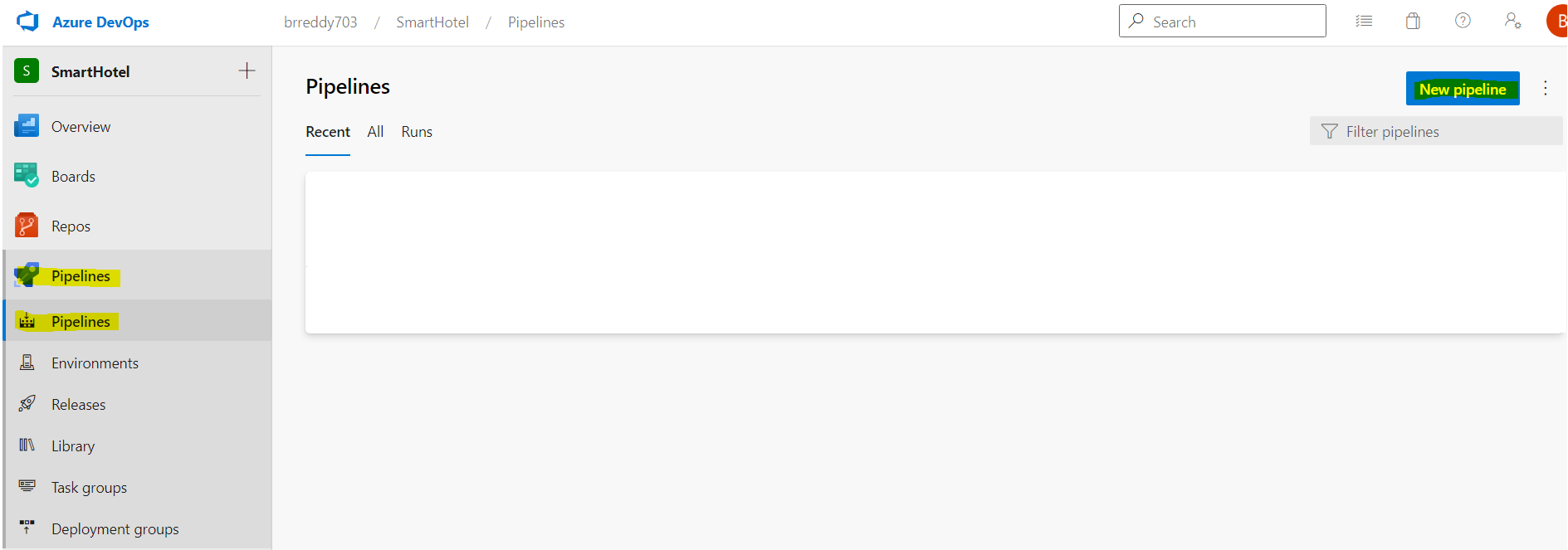
Now we can see the files in azure repo in master branch. Because I pushed to master branch.



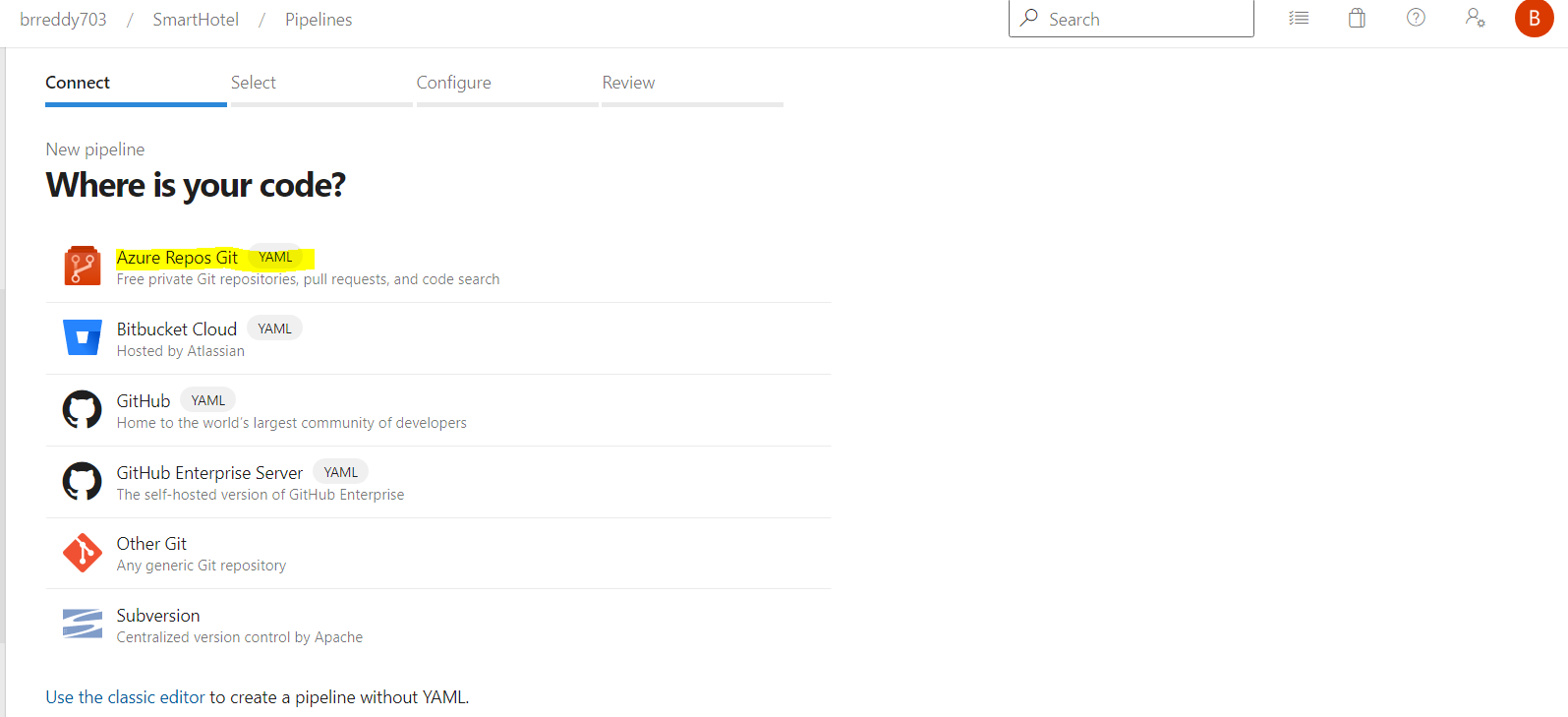
Code is ready. Next, Need to create pipeline for build and deploy.

### **Create Build and Deployment pipeline using YAML based pipeline**

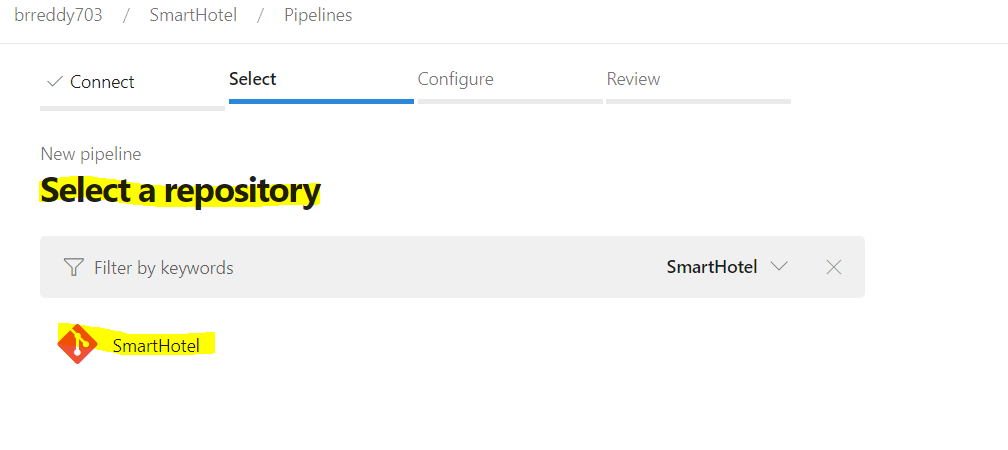
Go to the pipelines. Click on Pipelines and select newpipeline



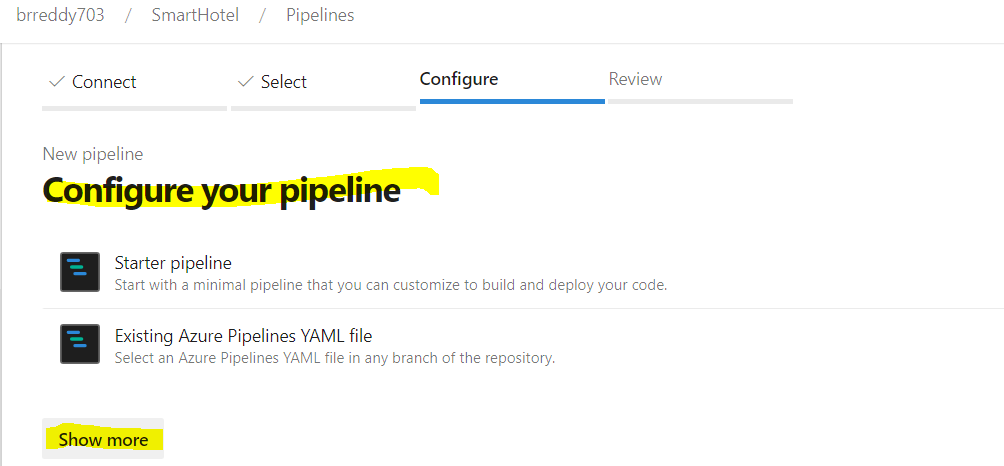
Select the repository where our code is. Select Azure Repos Git



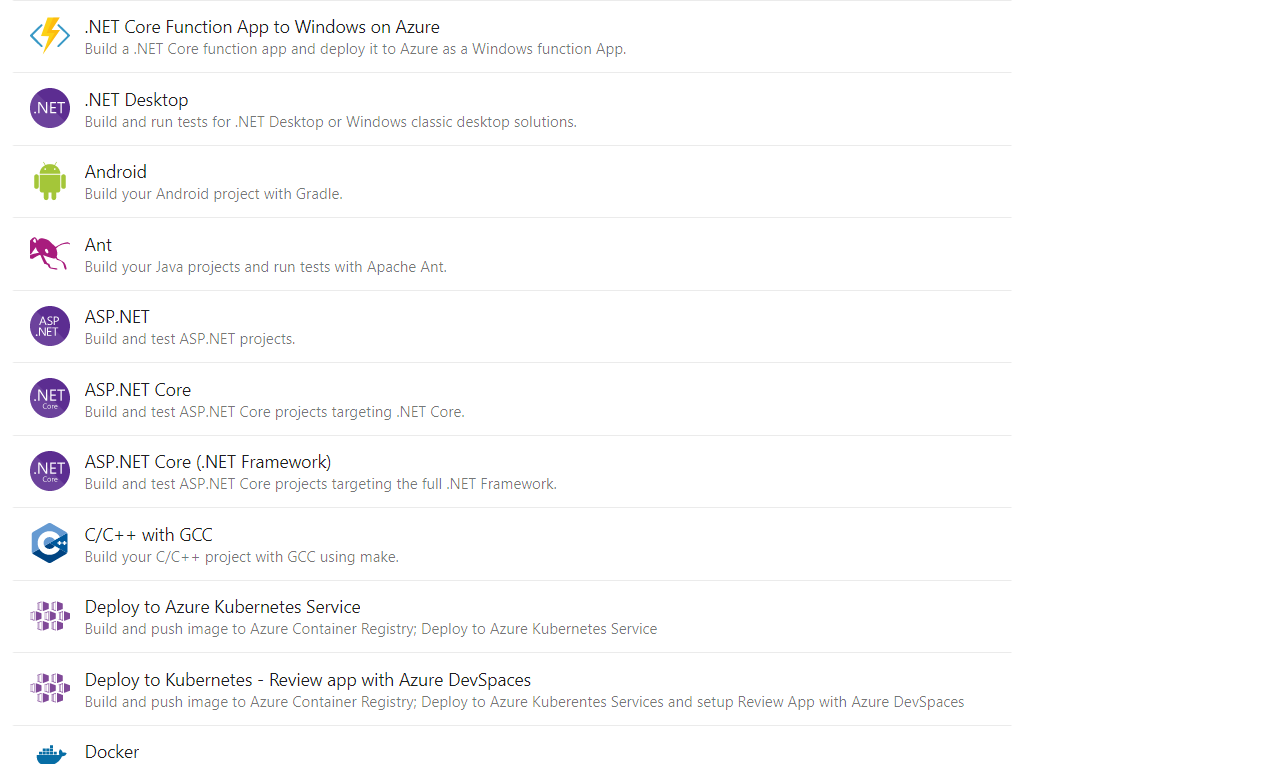
Select the repository name.



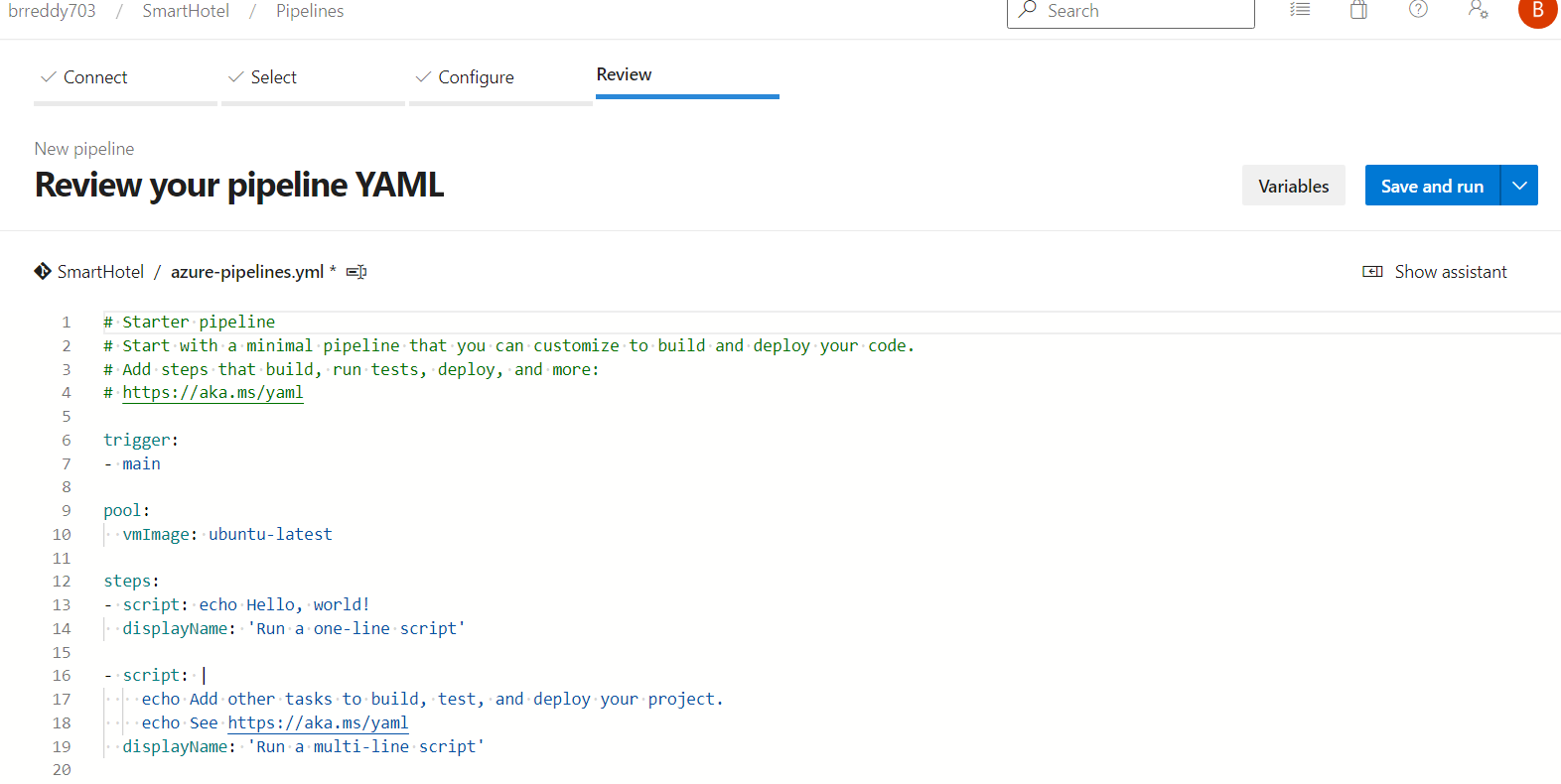
Will appear in built in templates. Click on show more to view more templates.



Based on our requirement we can select one template and then modify it.

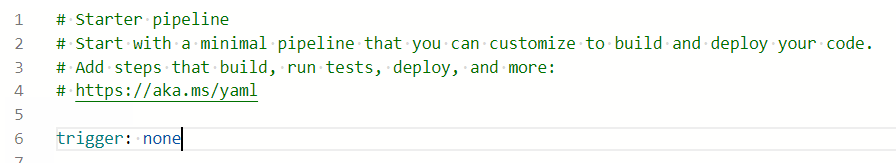


For this document, I am selecting starter pipeline and YAML starter pipeline will be look like below.

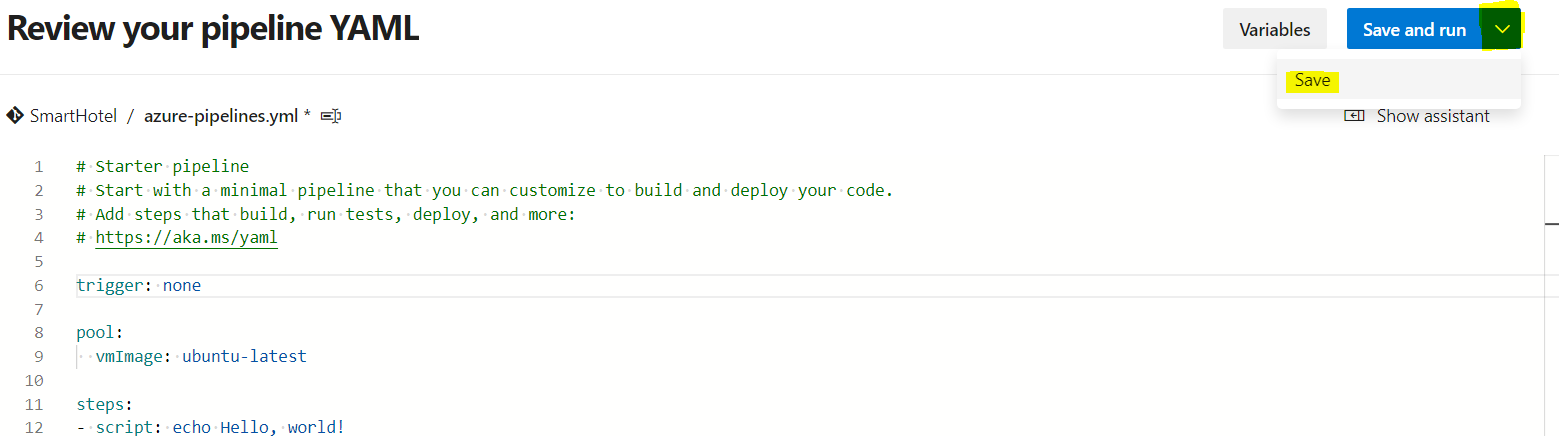


First I will edit the pipeline name. By default yaml based pipeline name will be repository name. To rename pipeline. Disable automatic trigger.

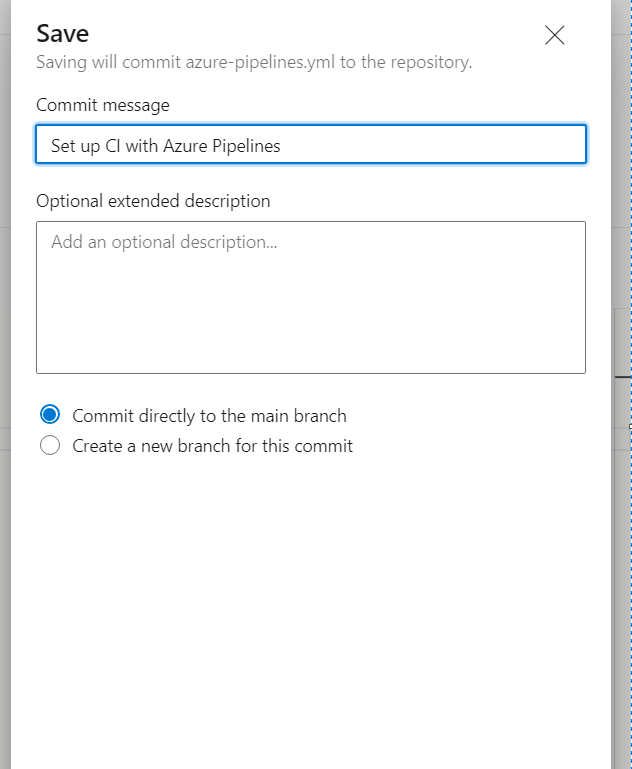
Make trigger section to ***none*** instead of ***main***



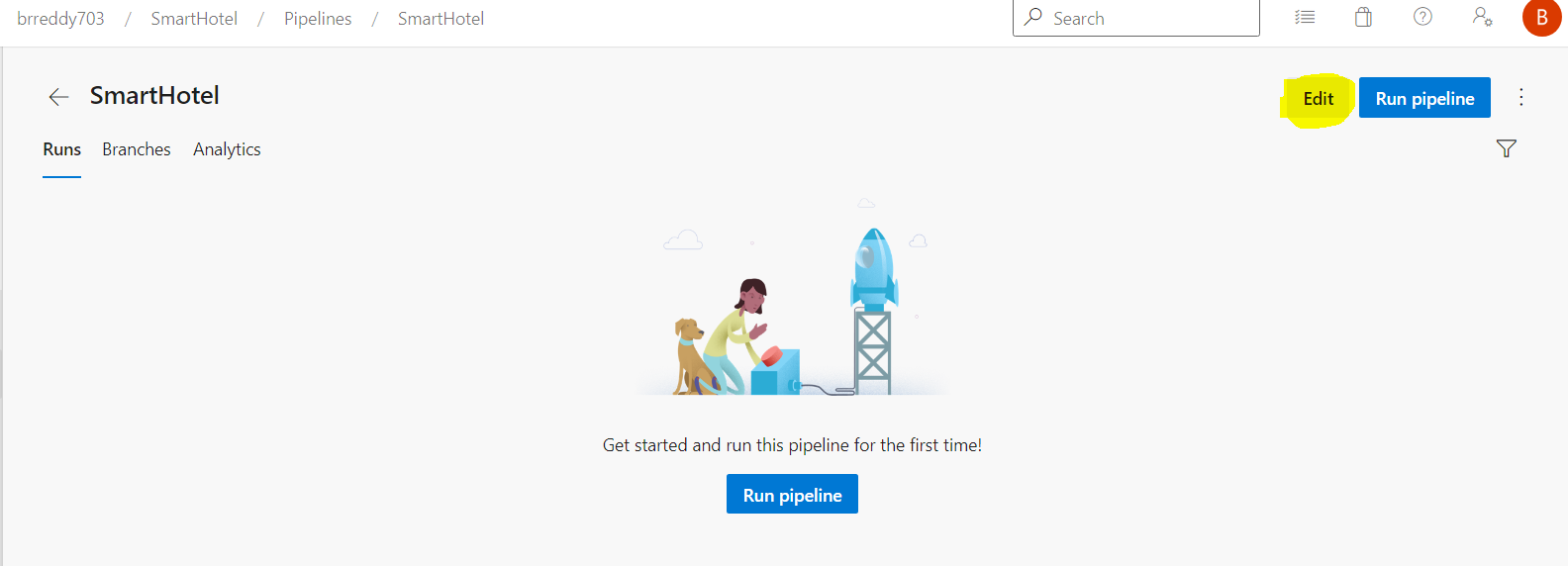
Then click on ***down arrow*** beside Save and run, then click on ***Save***.



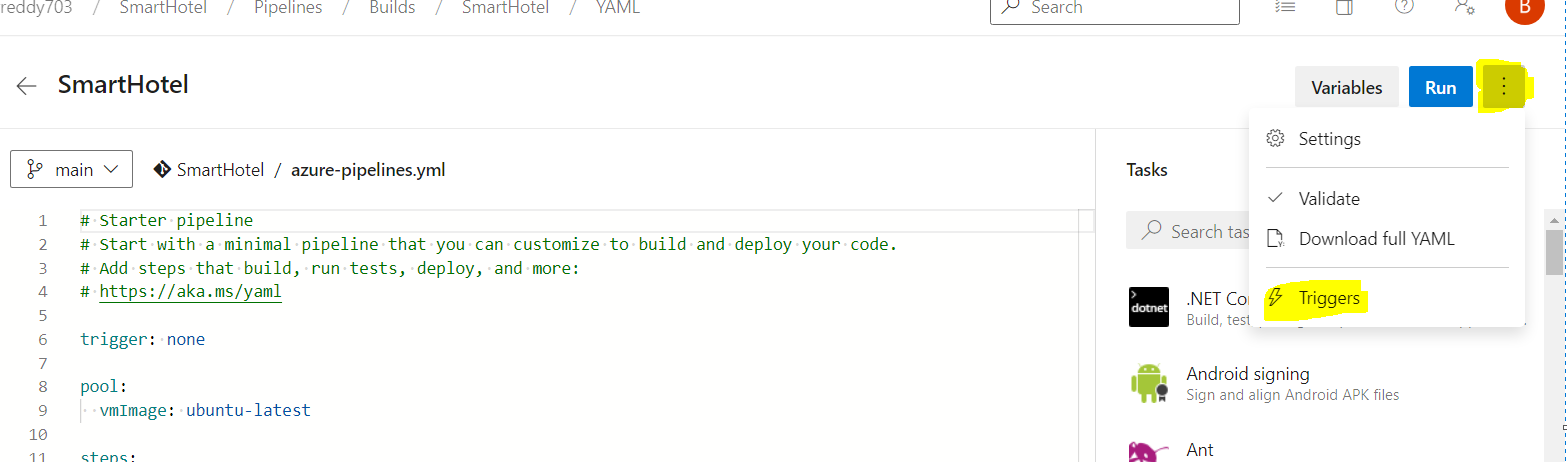
Enter some commit message and then click on Save button in the bottom



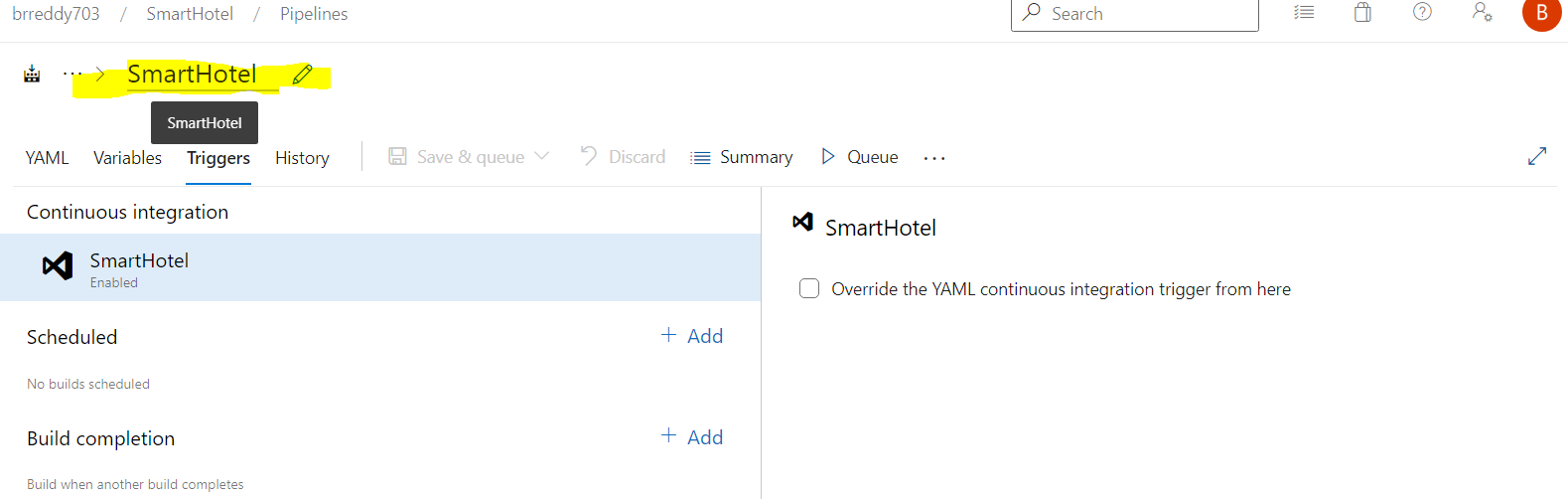
Now click on edit,



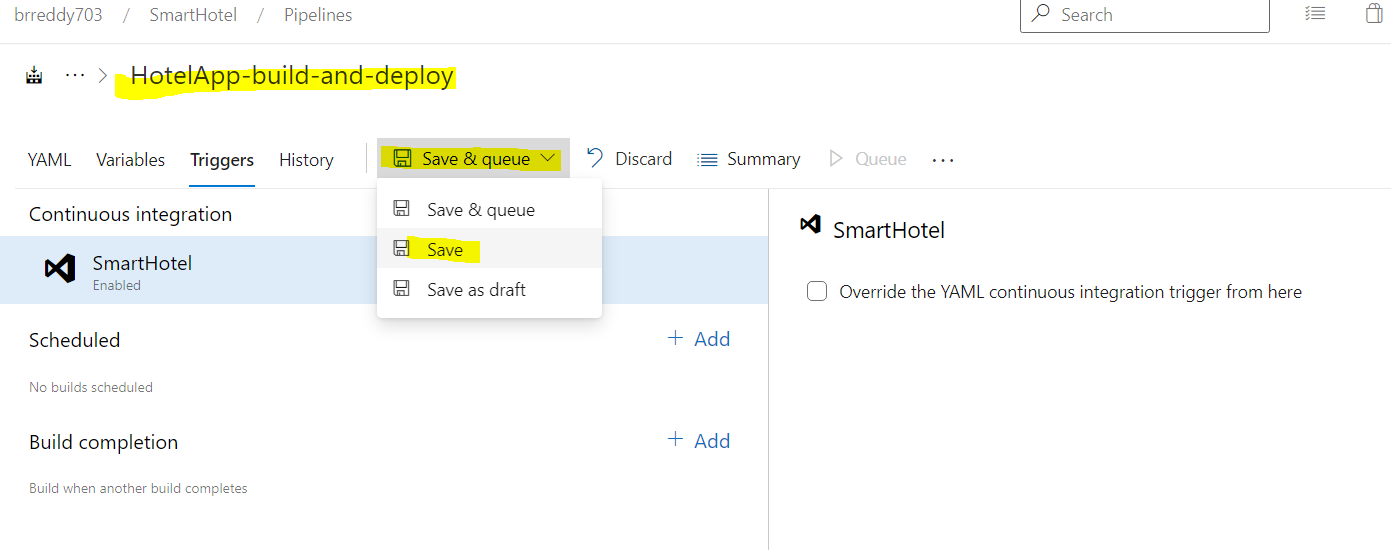
Then click on three dots beside Run. Then select triggers



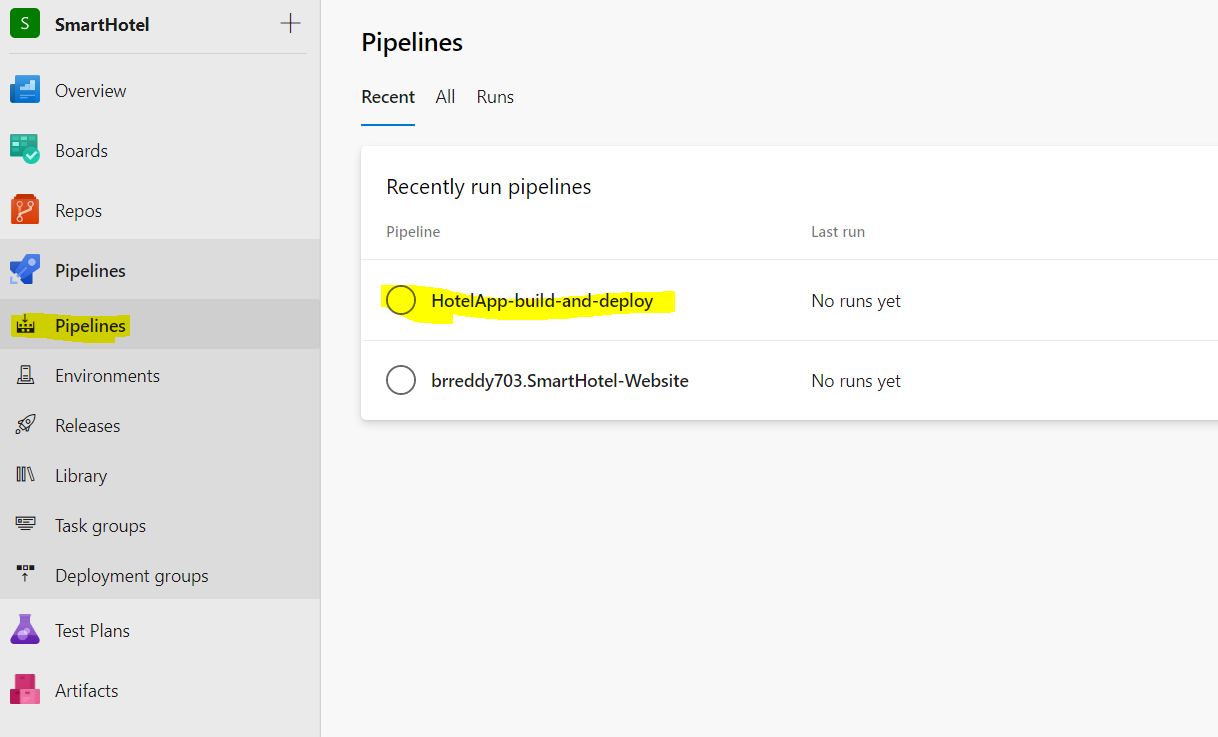
Now on top pipeline name will be there, rename it to what ever the name you want.

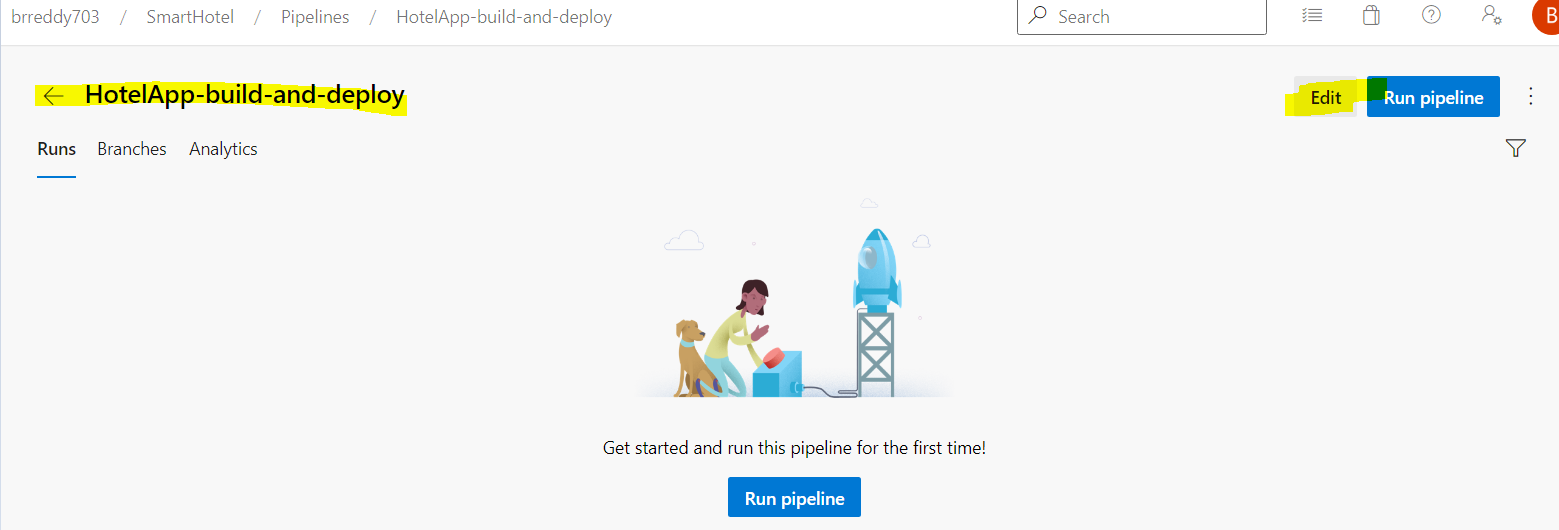


After rename you can Save and enter commit message. click save.

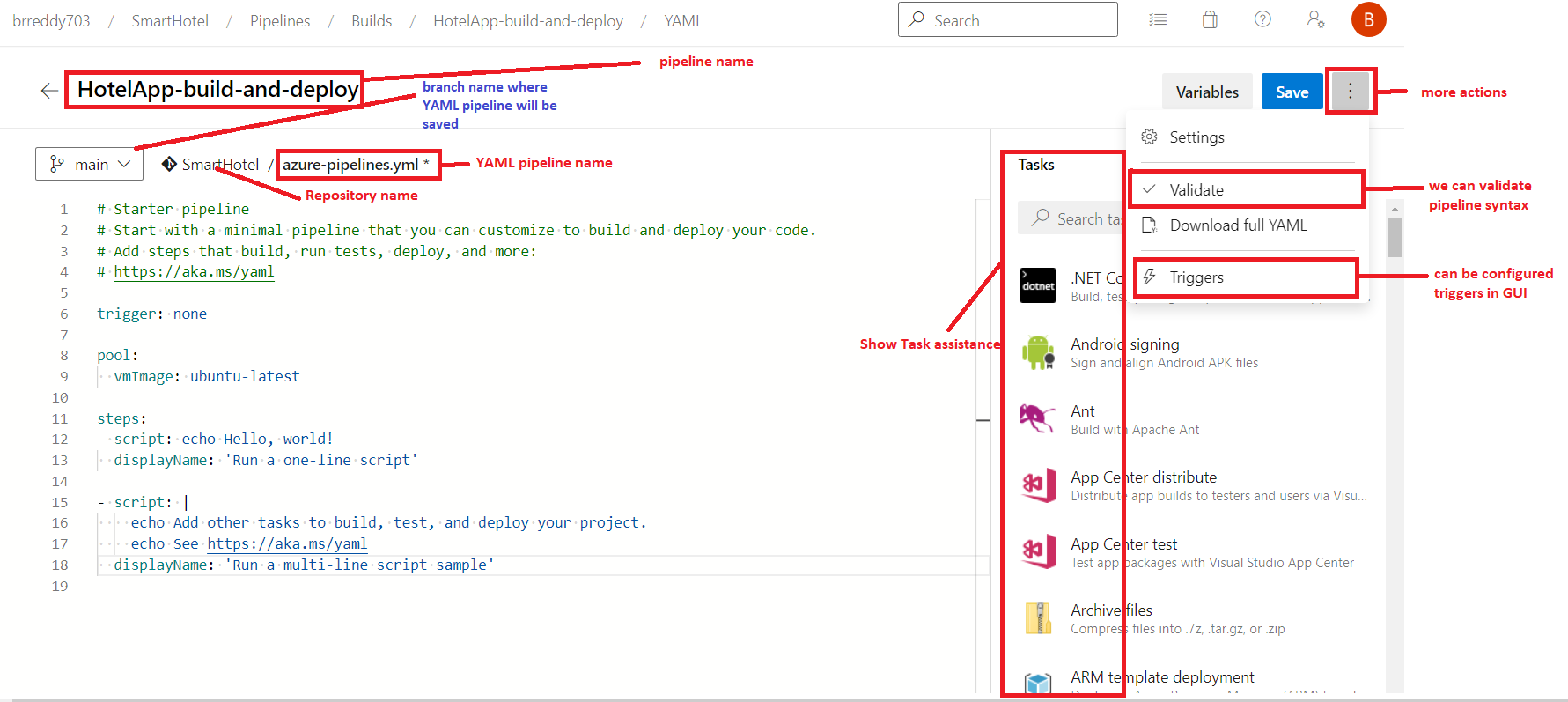


Click on Pipelines under Pipelines section to get back to list of pipelines. Here you can see the newly created pipeline is there.select that pipeline and click on edit to open pipeline editor.





Different options in pipeline editor





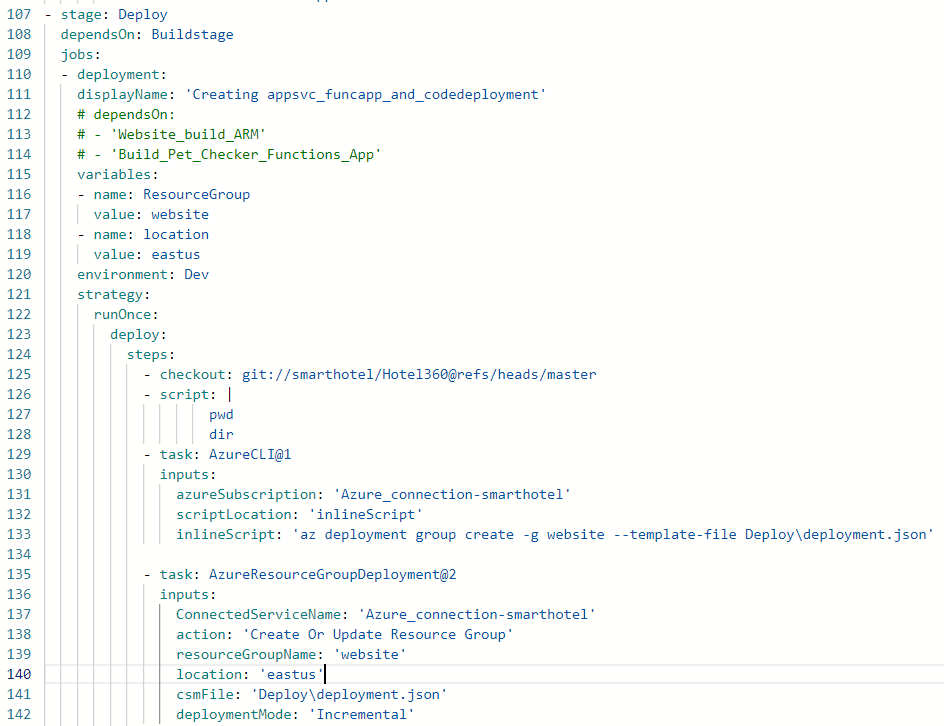
Here is the ***azure-pipelines.yml***  for pipeline.

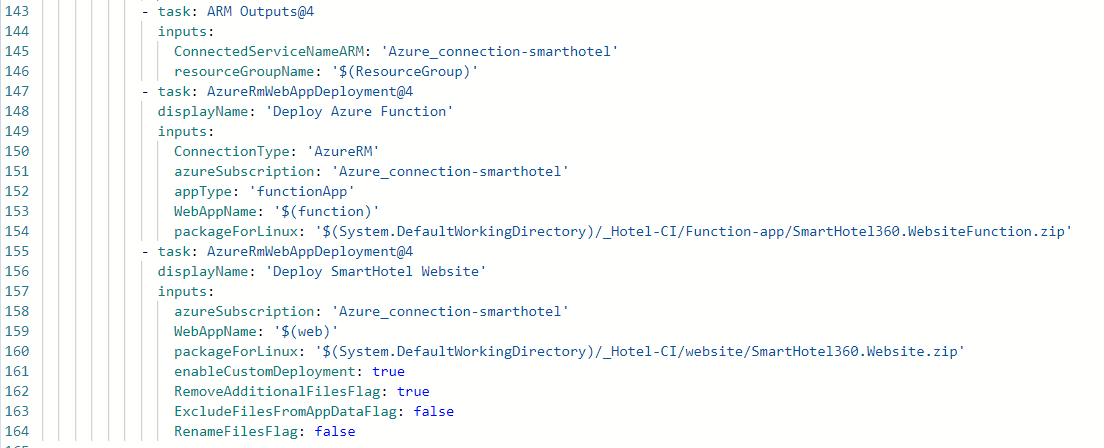
Here is the Complete YAML Pipeline.



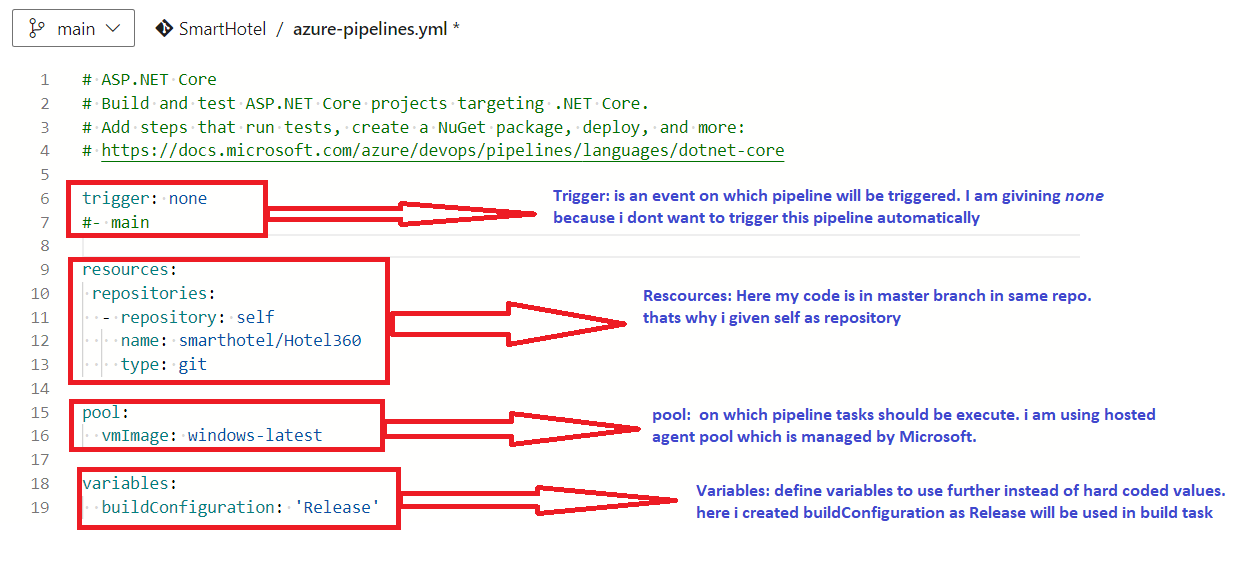






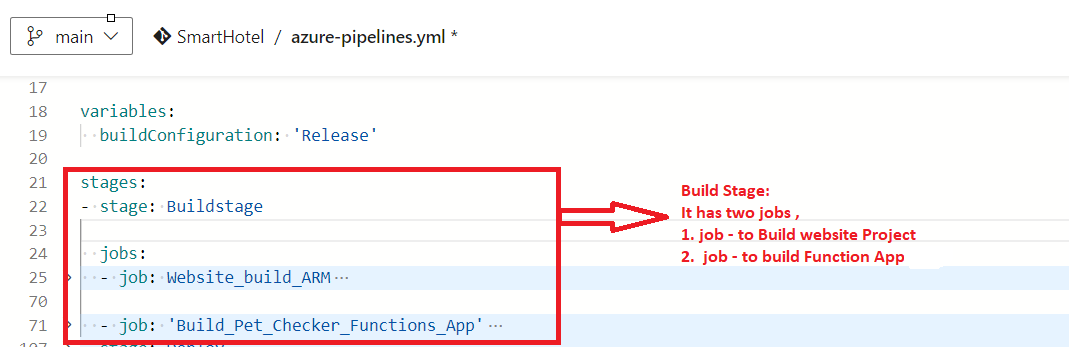


Here is the detailed information of each step of YAML pipeline.

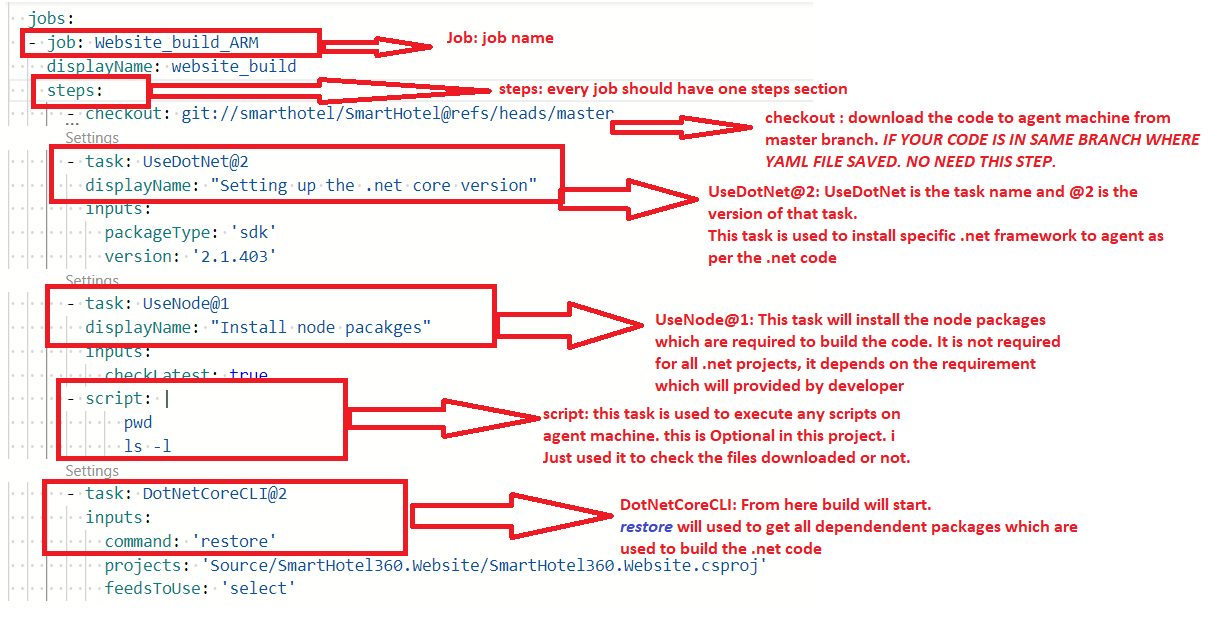


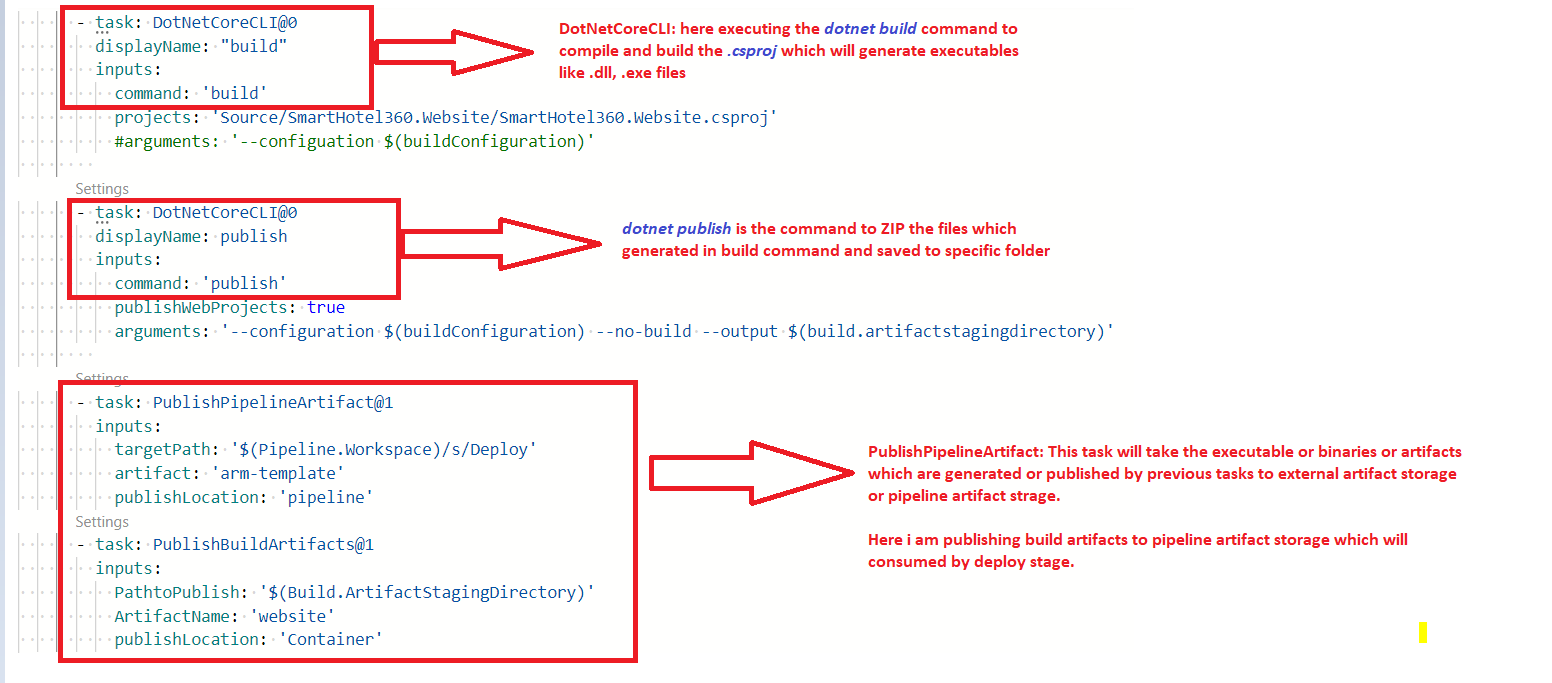
Here I created multi staged YAML build pipelined.



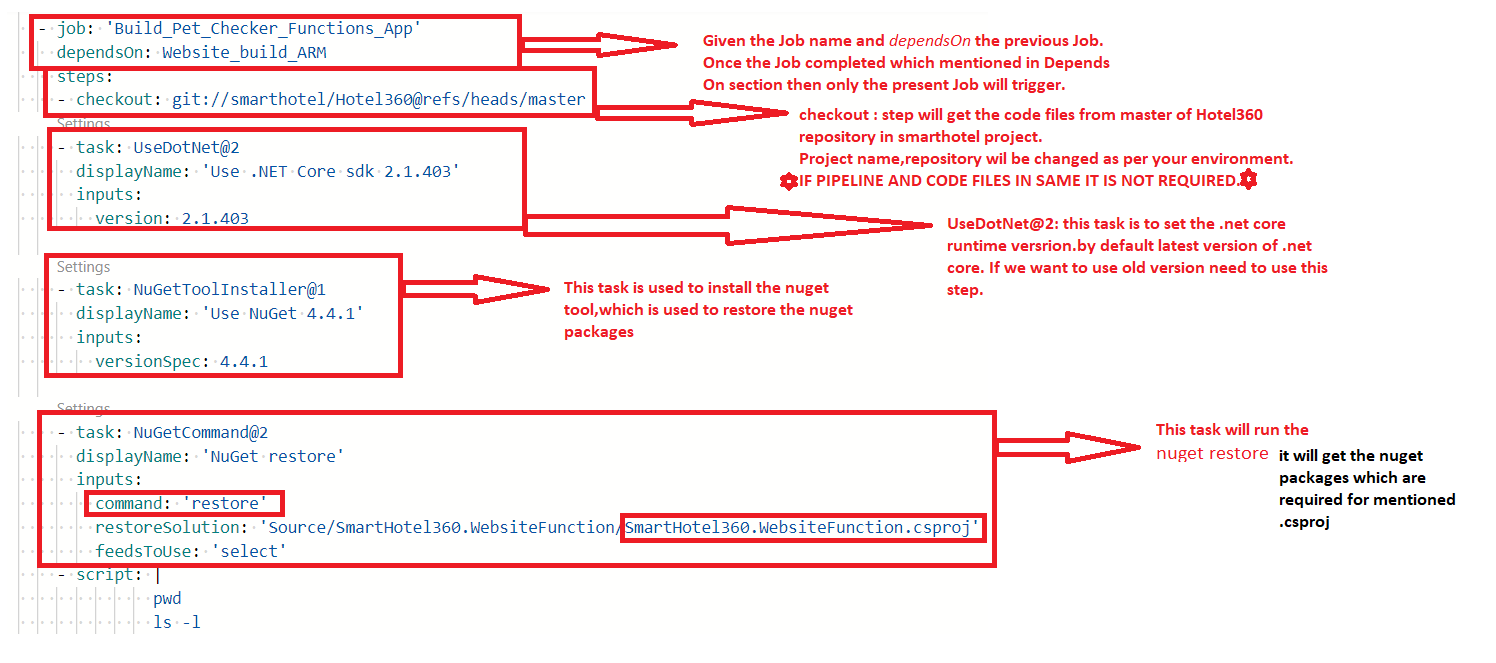


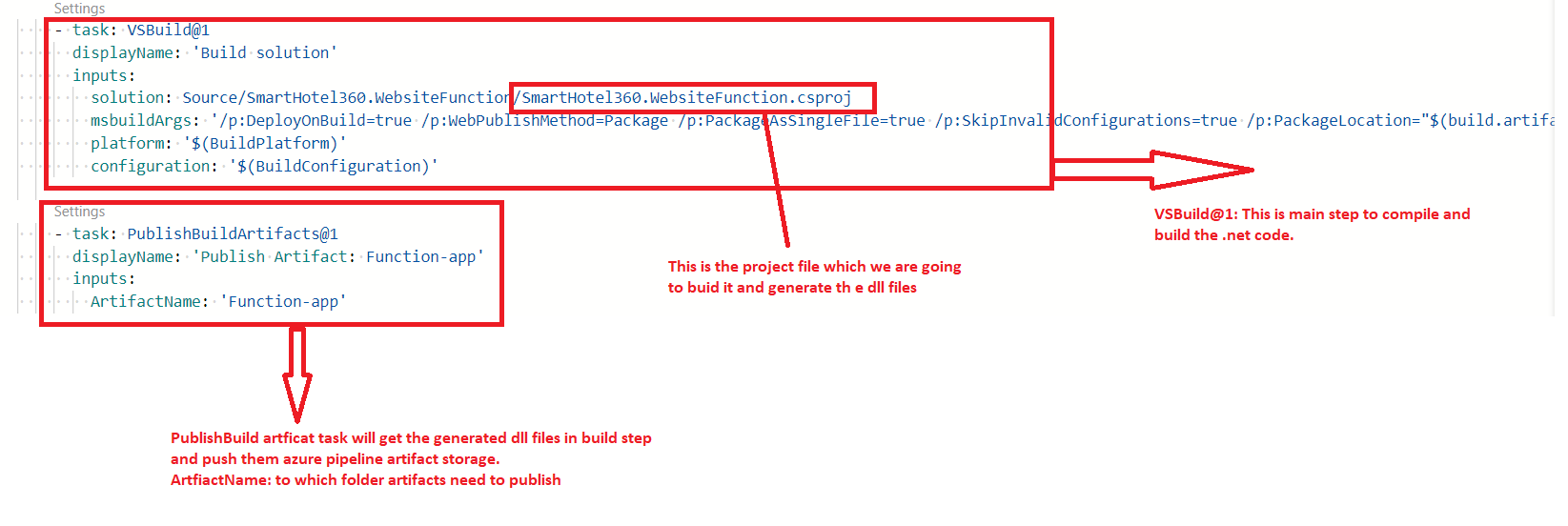
Website Build job





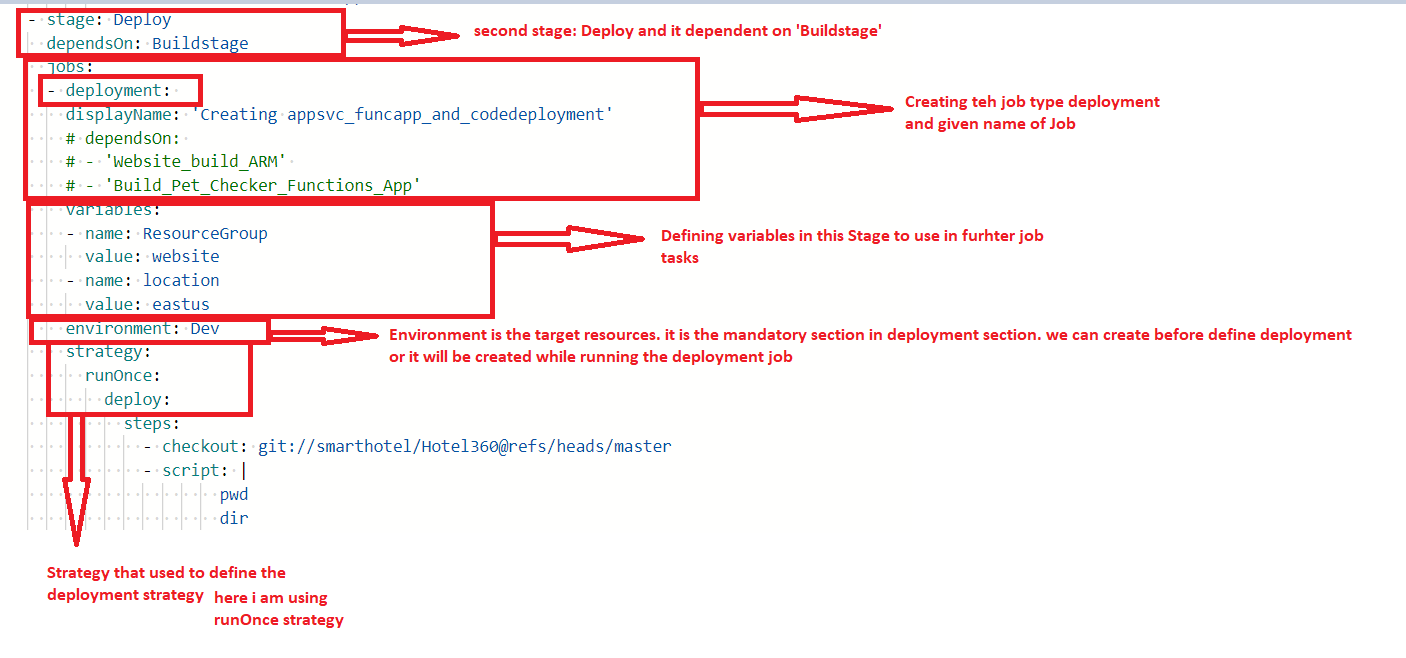
Another Job is Build the Function App

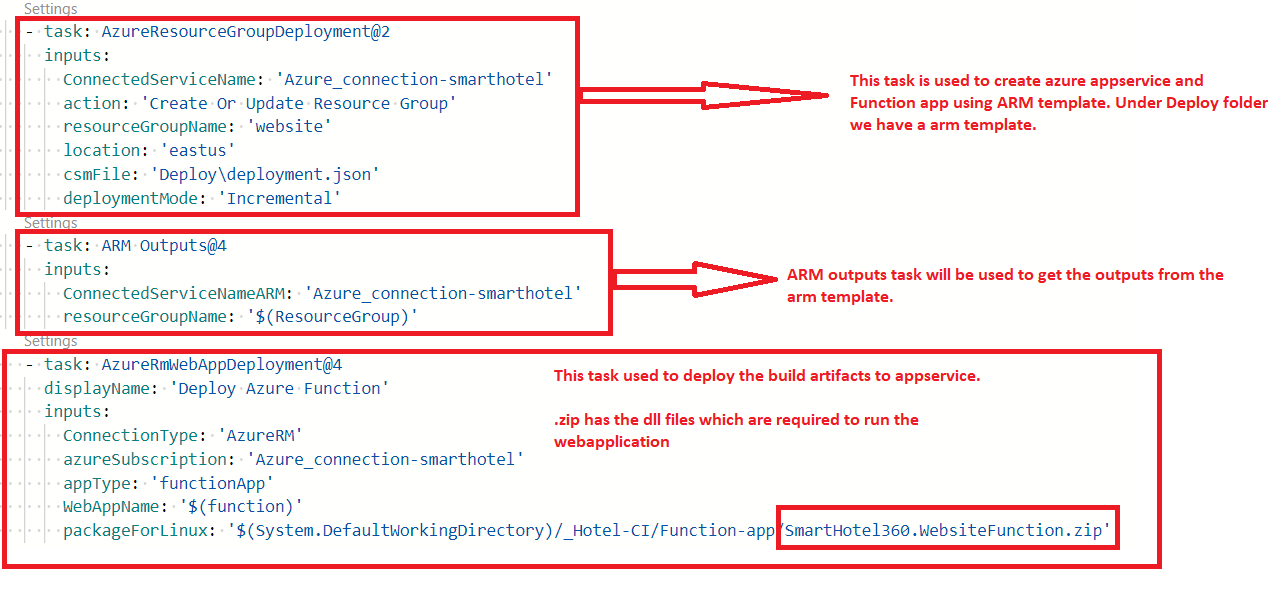


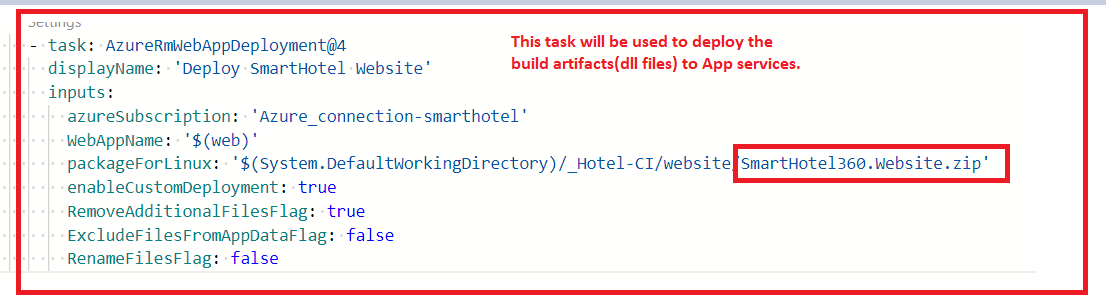


Till this Build stage completed. Now we need to create App service and deploy the artifacts to Appservice and azure function app.

Next Stage is Deploy stage:







Save the pipeline and run it.

See the progress of the pipeline.

# **Important Configurations in Pipelines**