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.

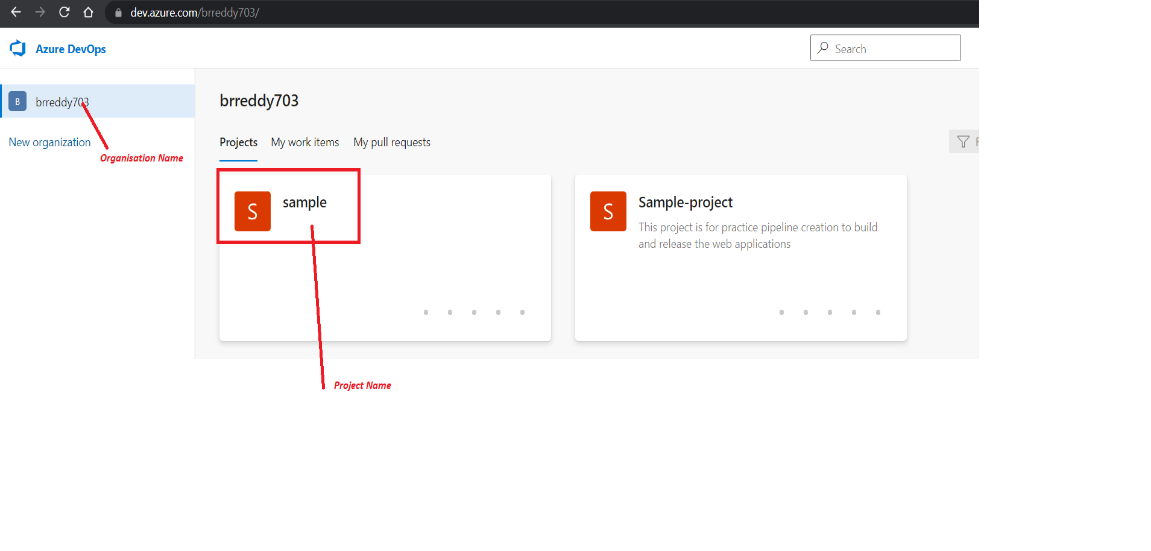
Creating Repository:

Prerequisites:

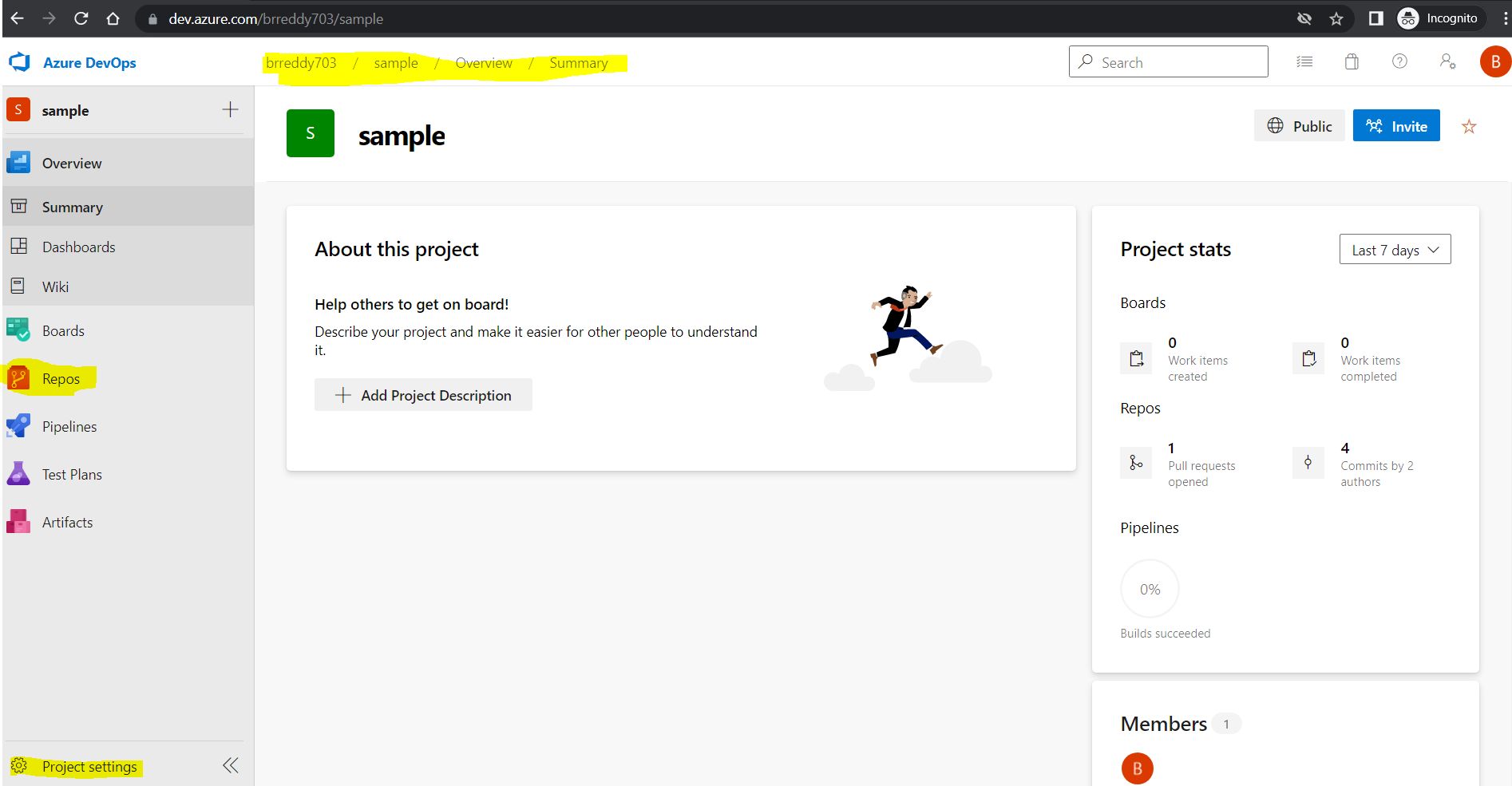
1. We should signup for Azure DevOps using dev.azure.com
2. Should be created one project in Azure DevOps Organisation

Steps to create Repo:

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

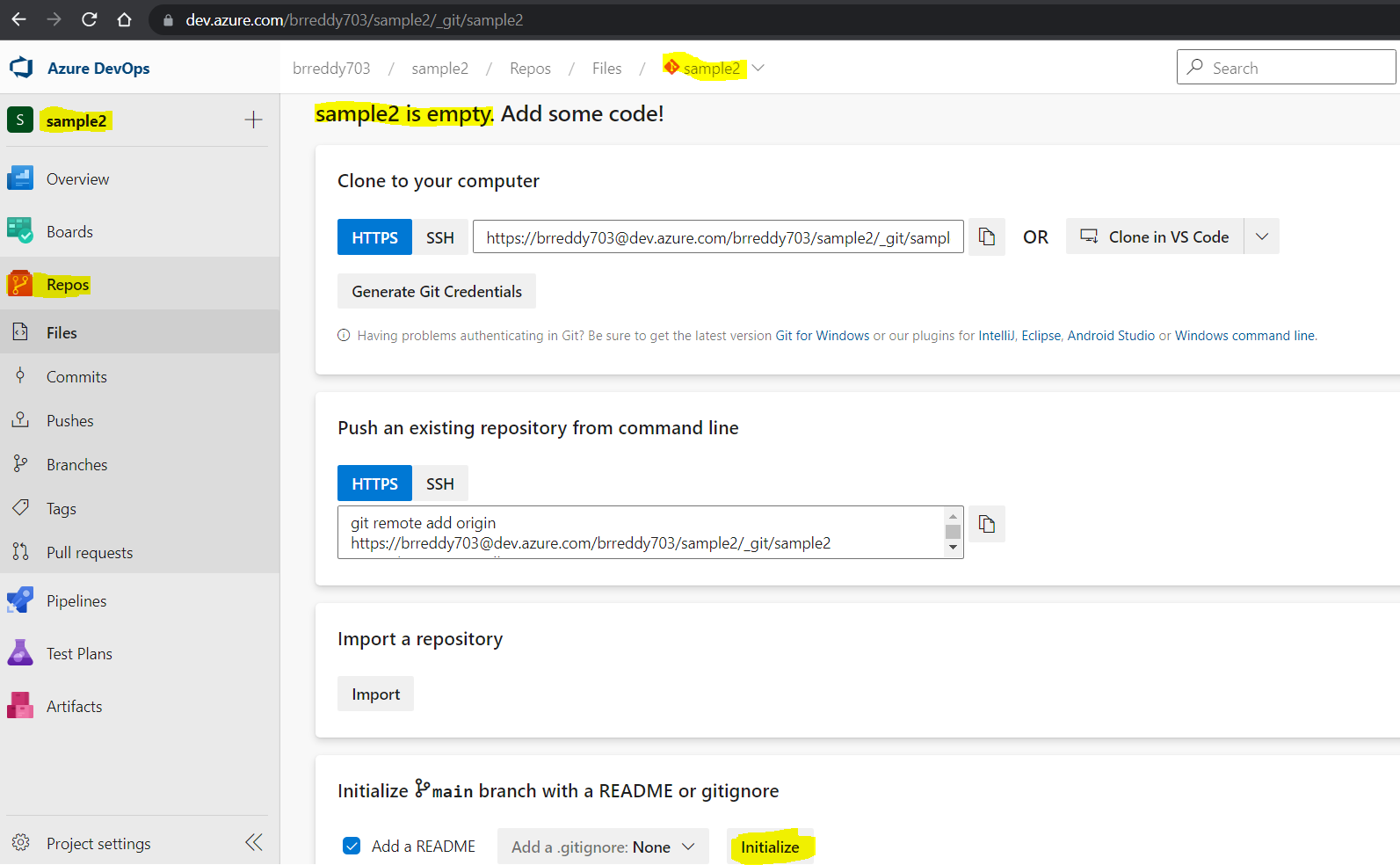


1. Click on project and it will take you to the project Summary page.

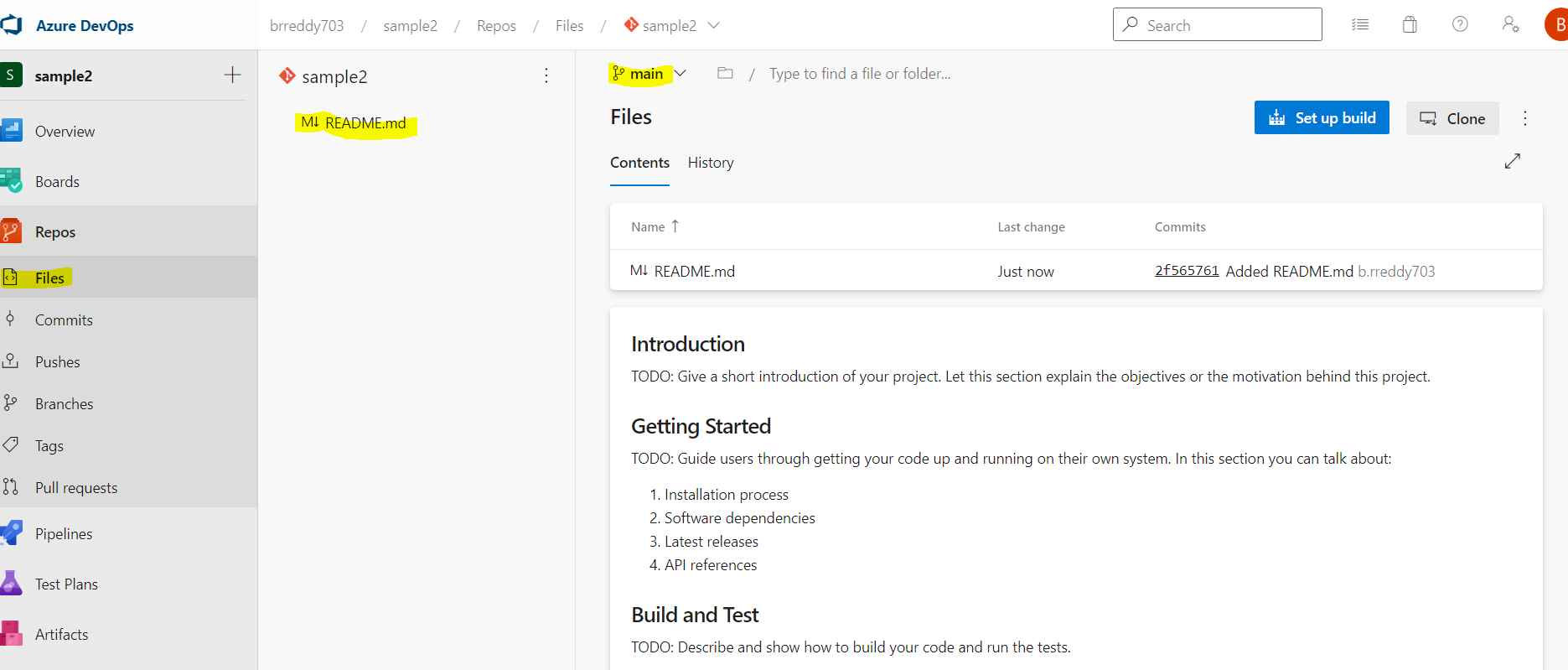


1. Click on the Repos from the left menu in summary page.

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.

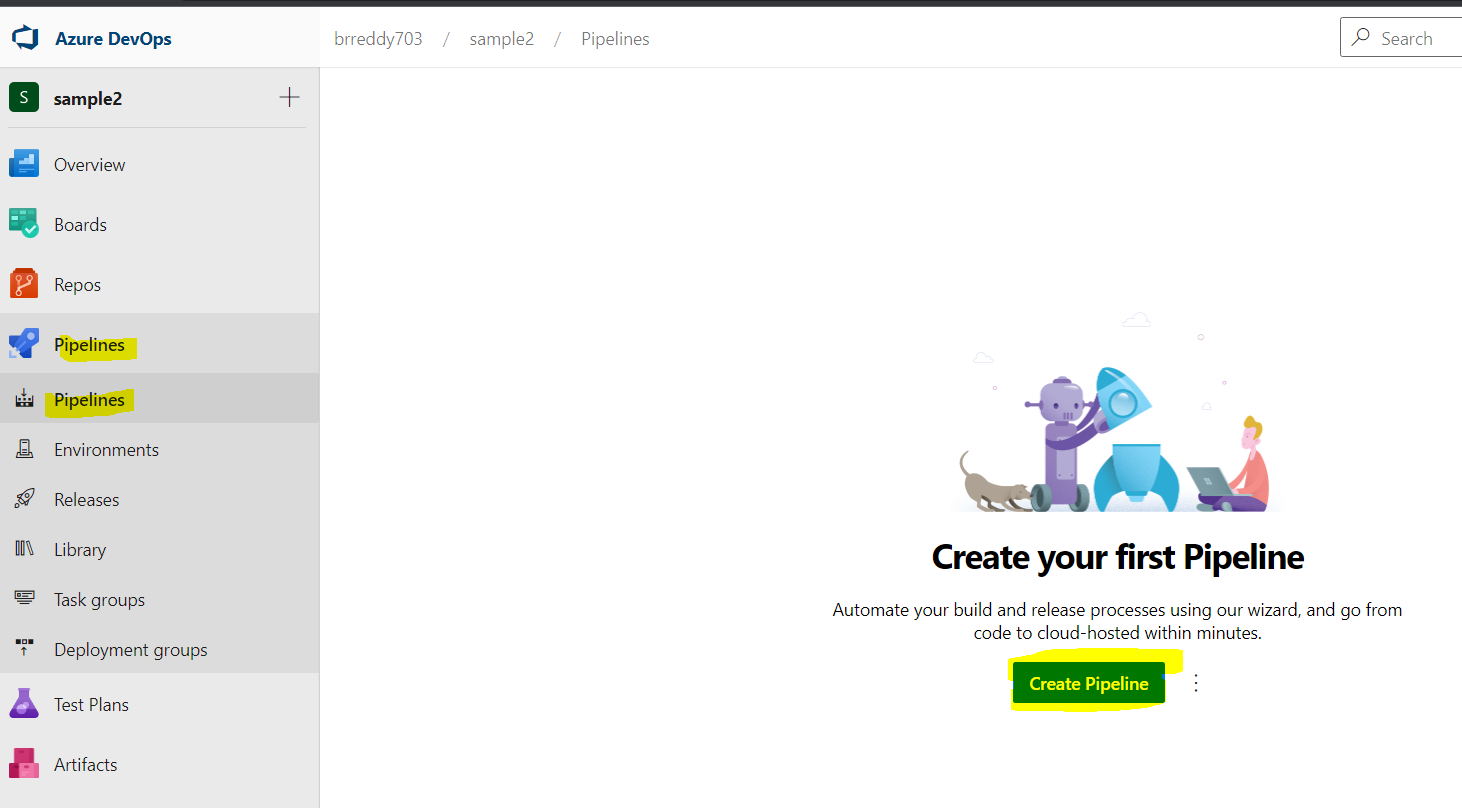
***Prerequisites:***

Before start to create a pipelines

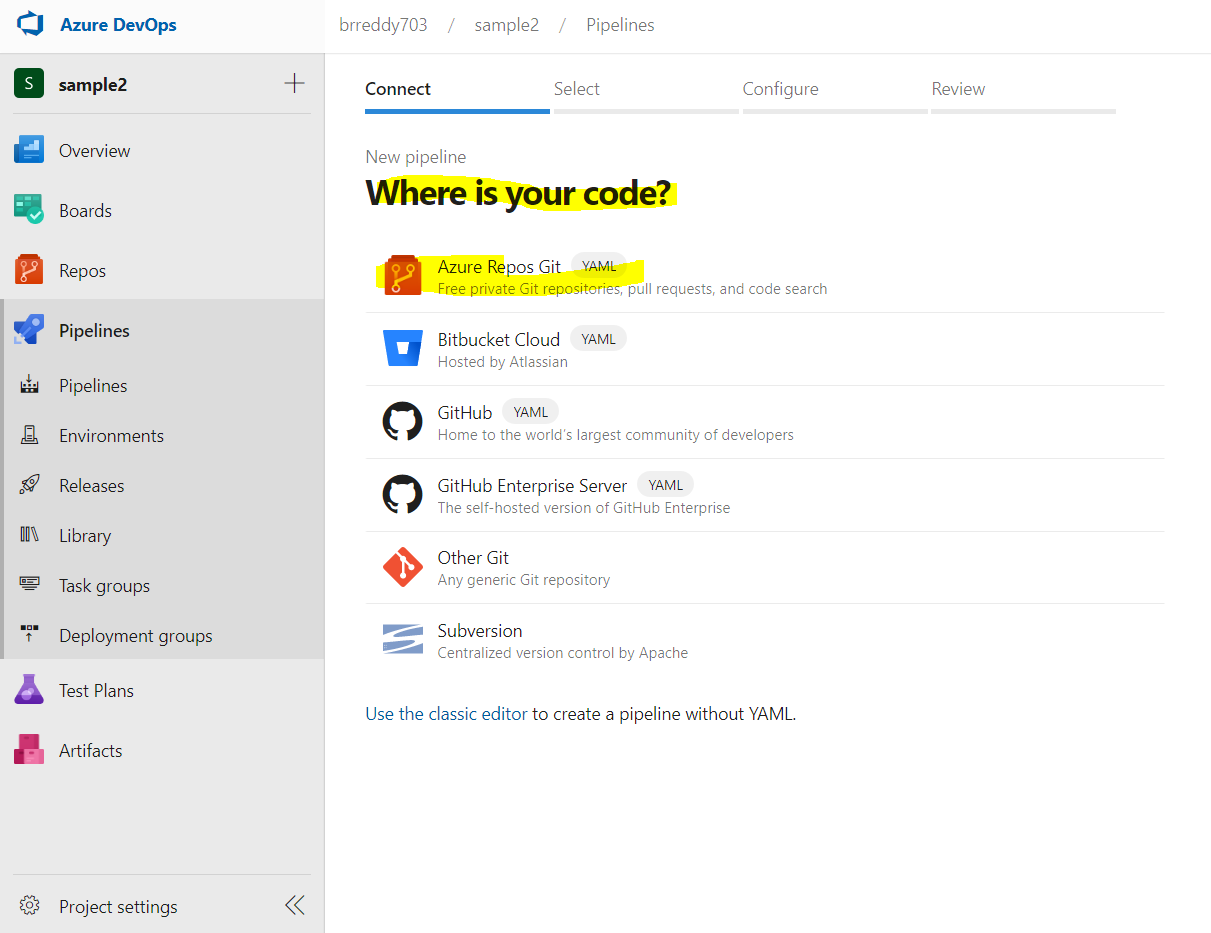
1. You should create one Project in Azure DevOps organisation. To [signup](https://azure.microsoft.com/en-us/services/devops/) azure DevOps
2. Should have one repository.

***Creating Starter pipeline:***

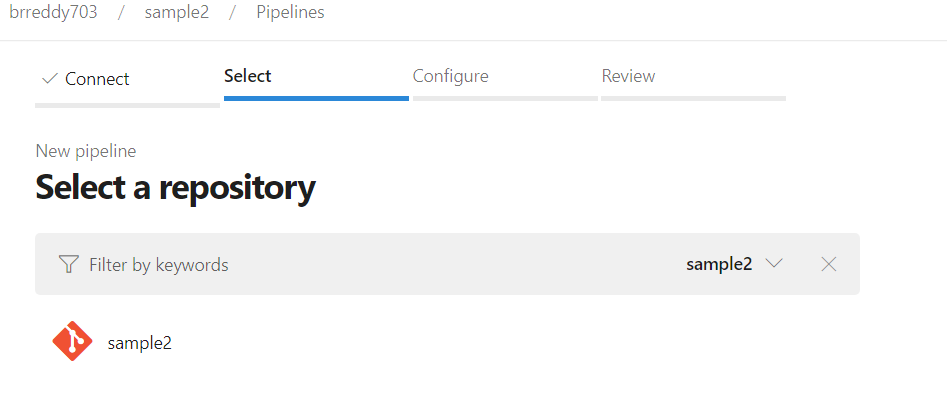
1. From the Project Summary click on Pipelines and Create New pipeline



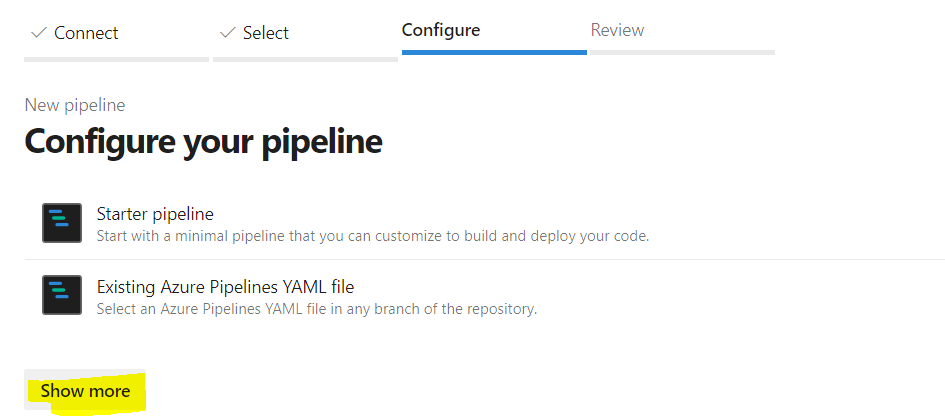
1. Next need to select Version control system in which your source code is saved. Select the Azure Repos Git



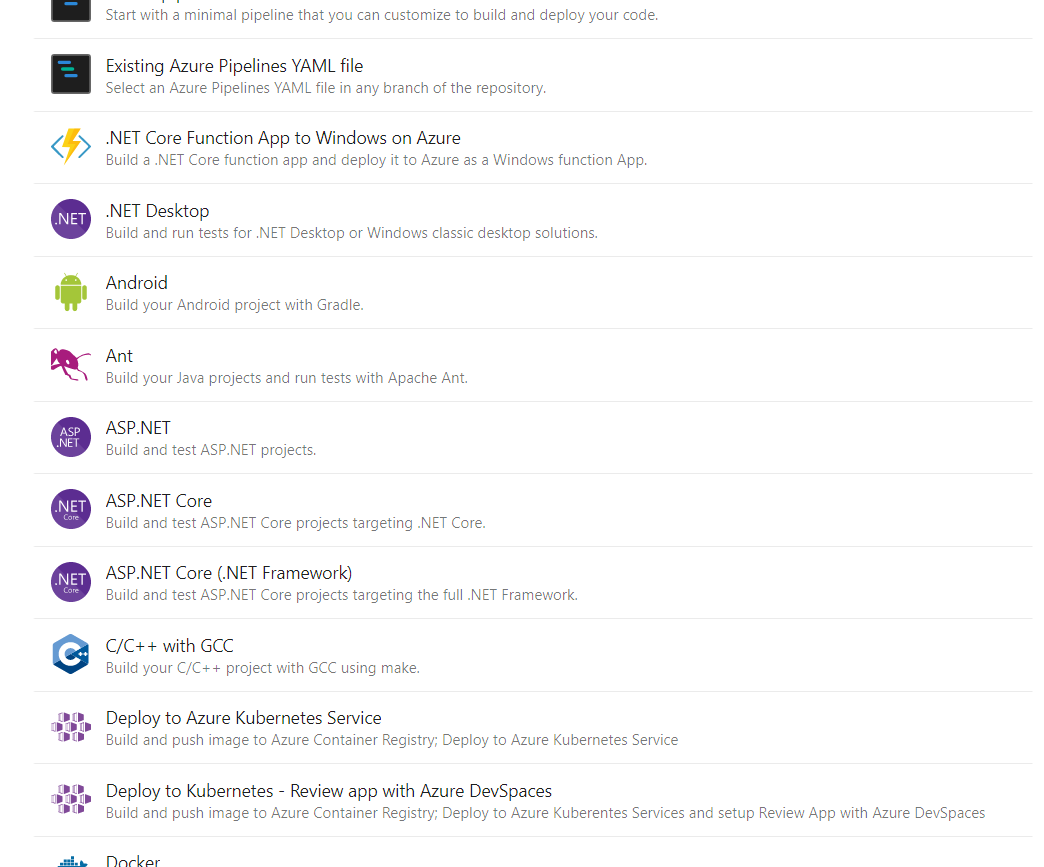
1. Need to select the repository from the list. It will list all repos in that Azure repos of selected Project.



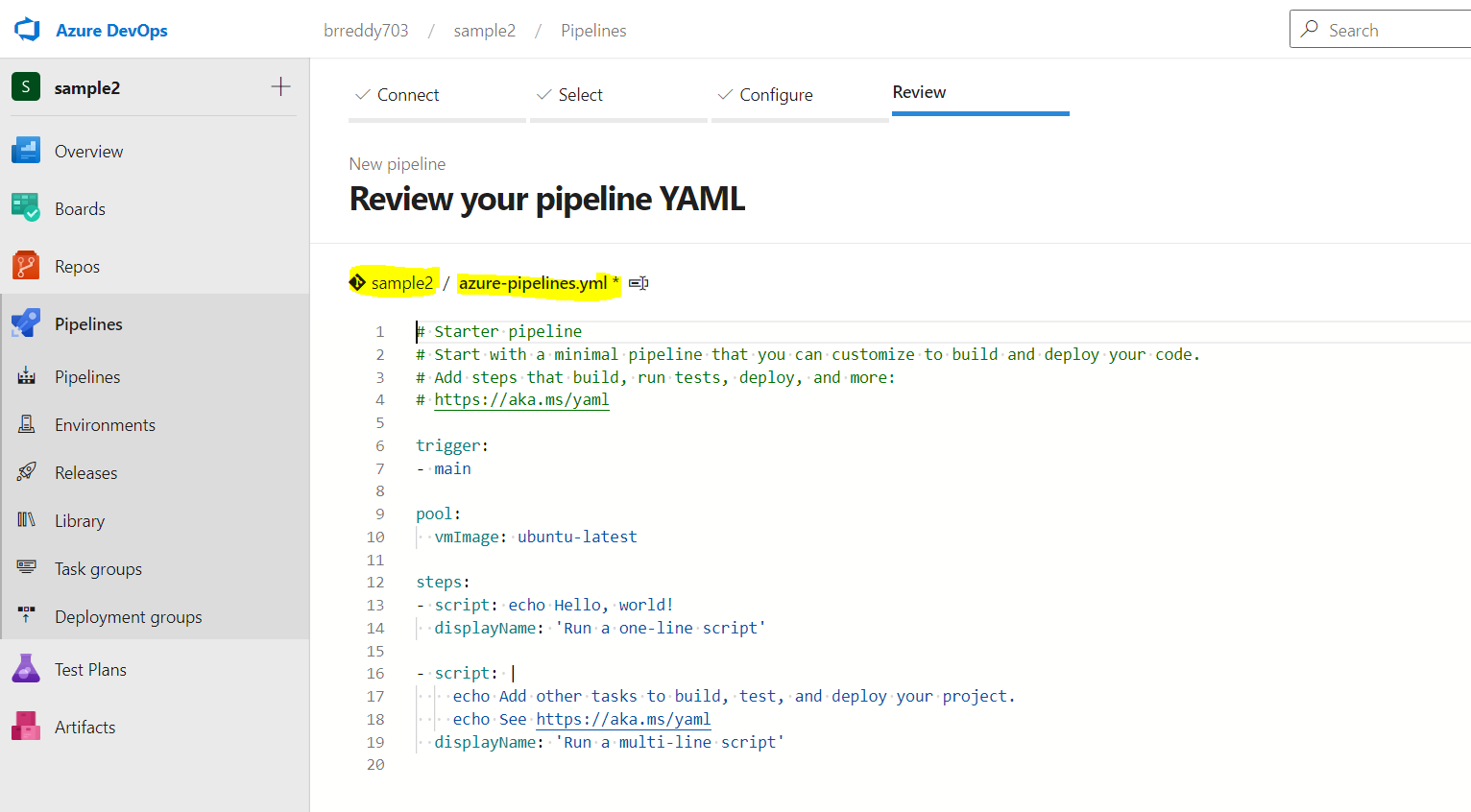
1. Now it will show the available pipeline template to create. Click on Show more for more templates if not displayed.



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.

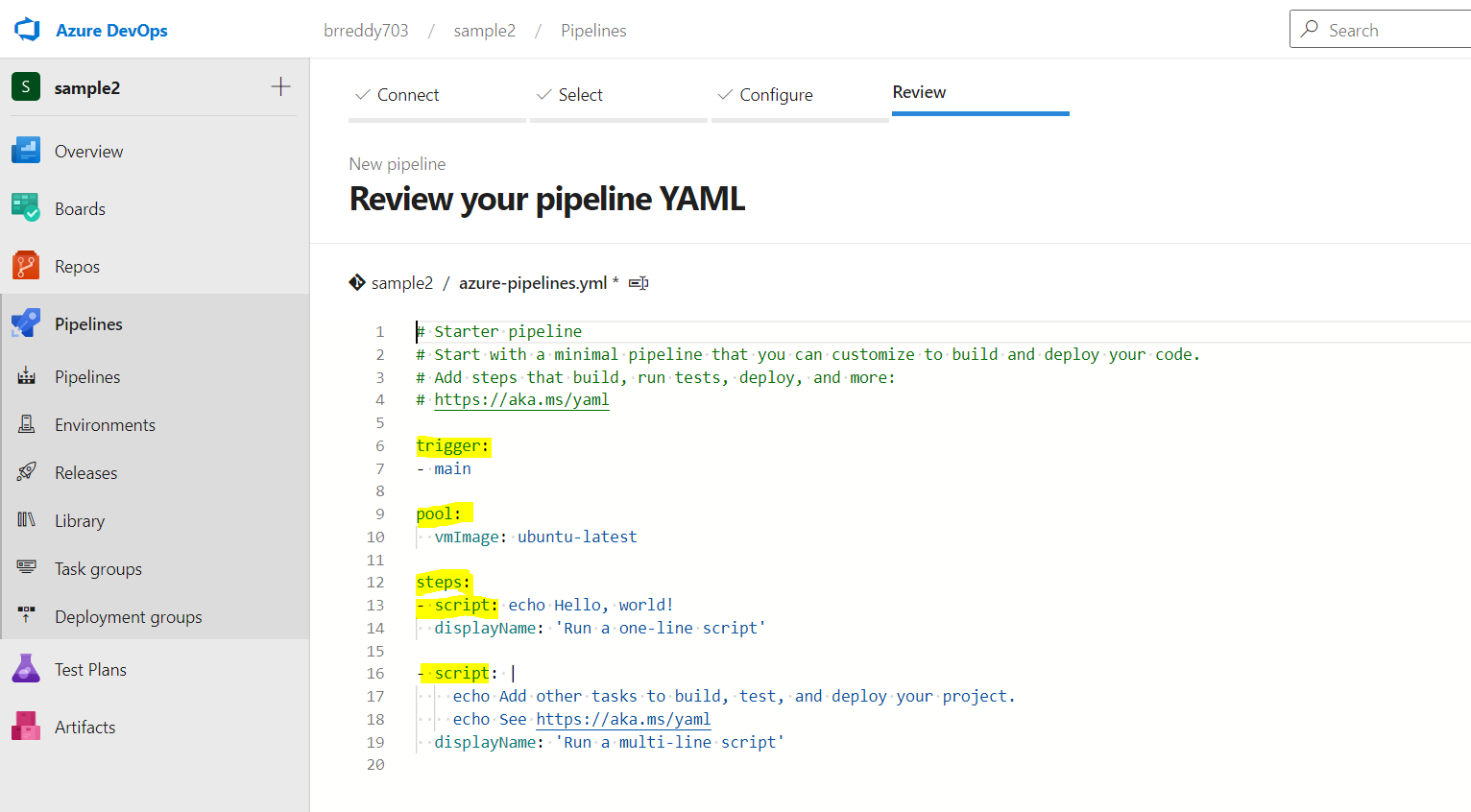


1. Here we can see the different sections in pipeline

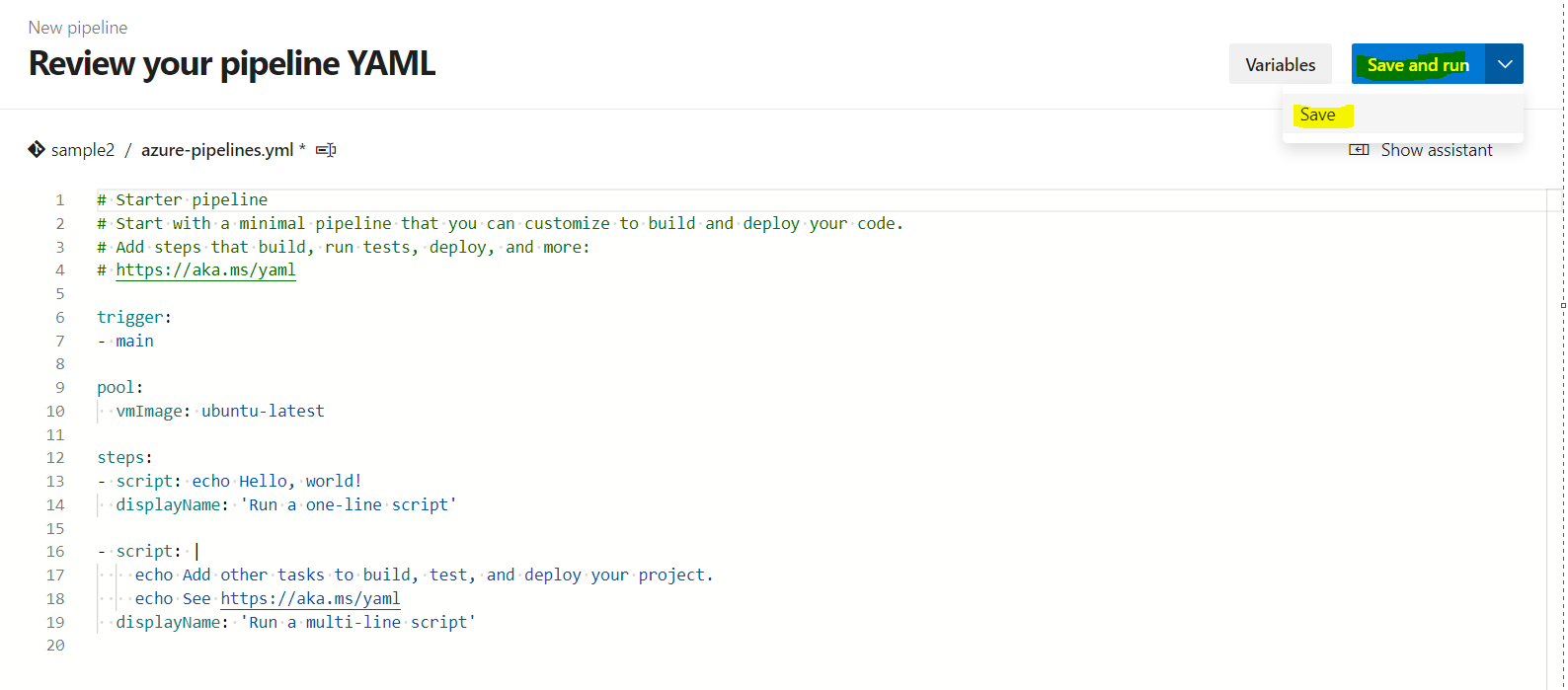
***Trigger:*** *#triiger is an event to run the pipeline. Any commit happens to main branch this pipeline will run.*

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

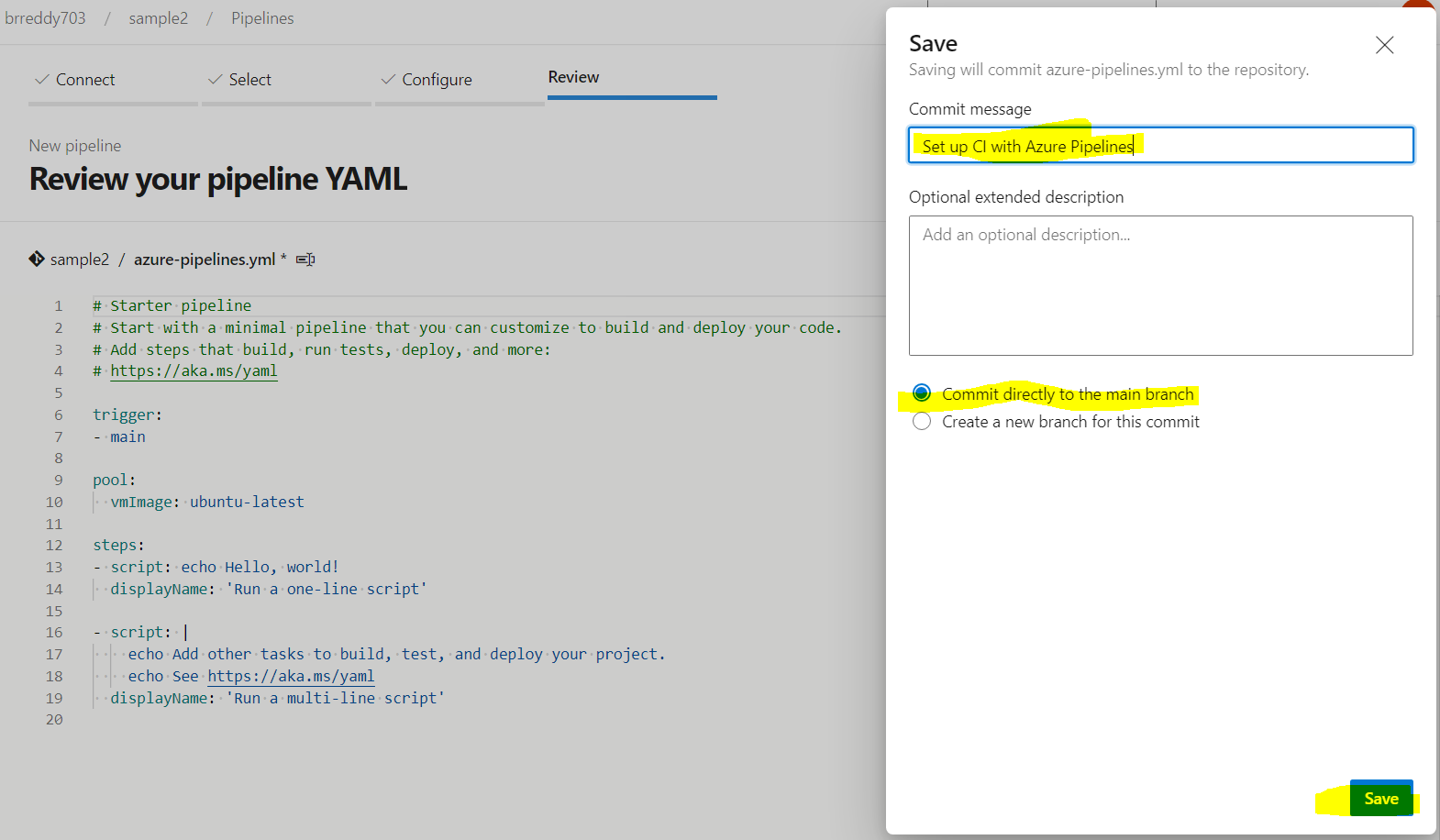
***Steps:*** *# Actual execution tasks will define in this section.*



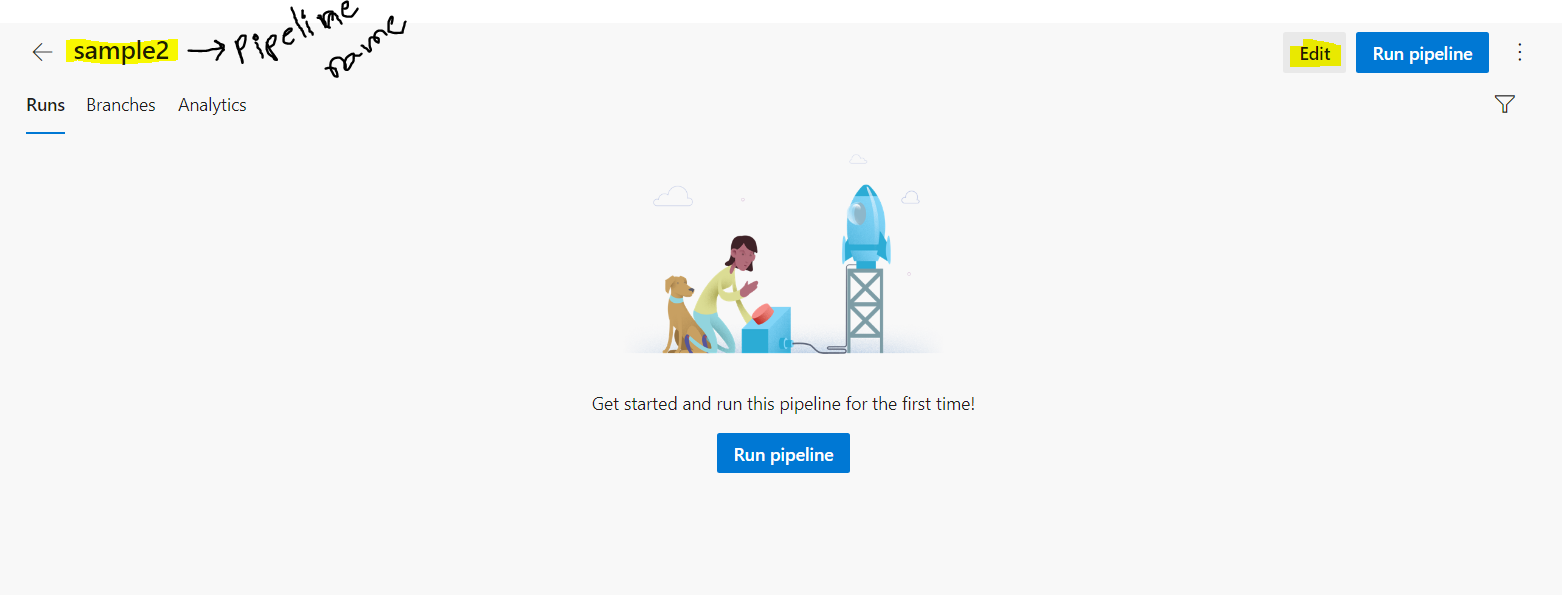
1. Once you done with pipeline edit. You can save and run or save it.by default pipeline name will be repository name.



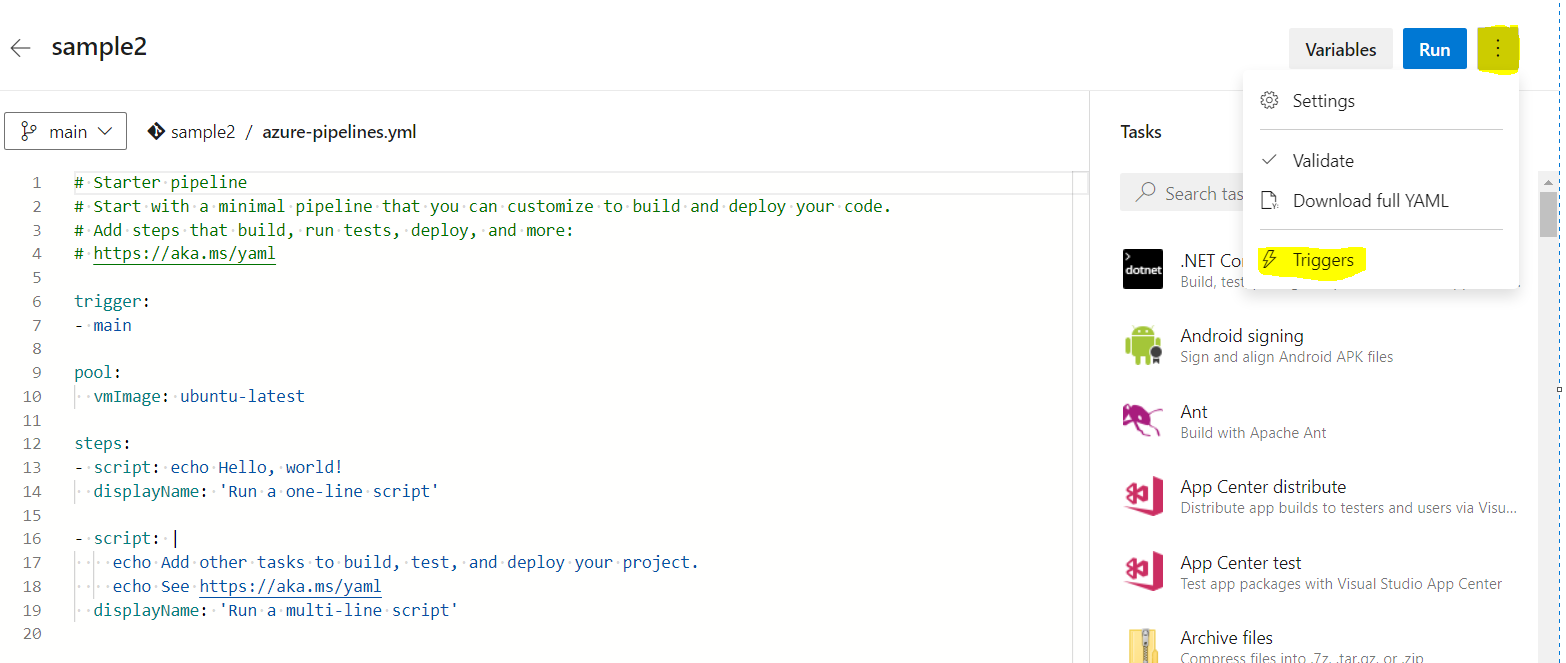
1. To rename the yaml pipeline, you need to select only save option. Click on save and enter commit message. Click on Save in the end.



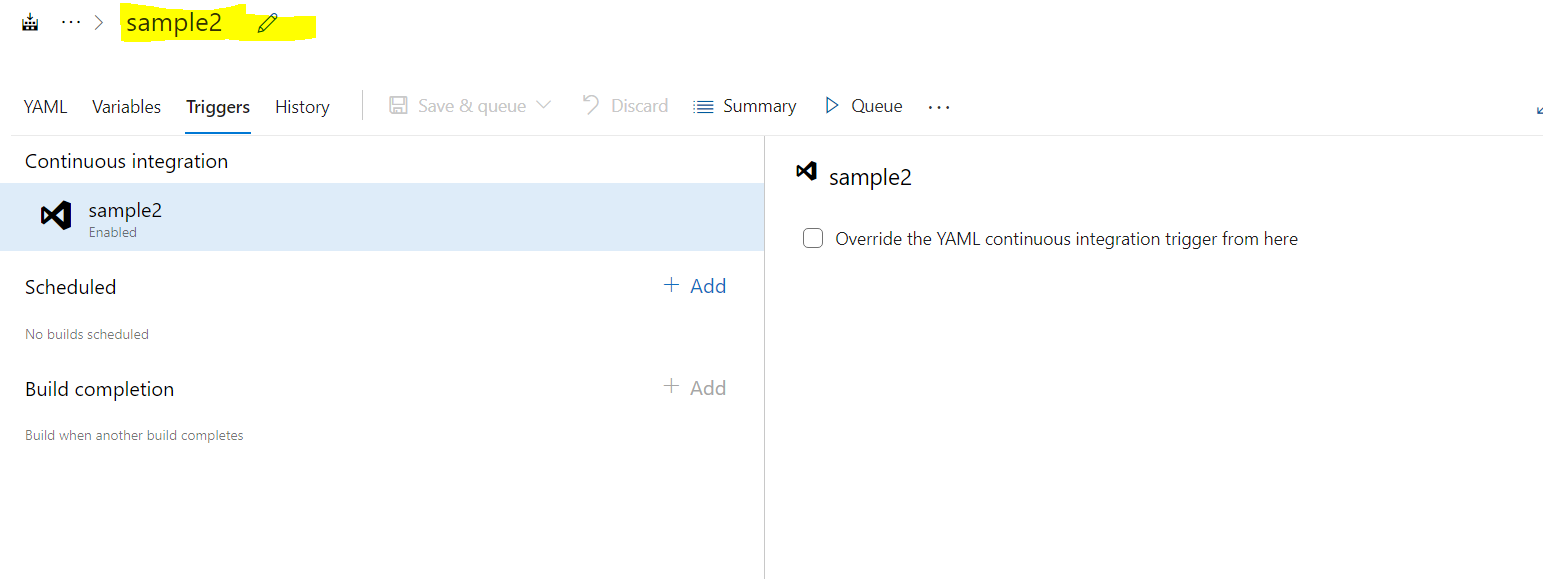
1. After saving it will take us to pipeline run page. Click on edit to go to pipeline editor.



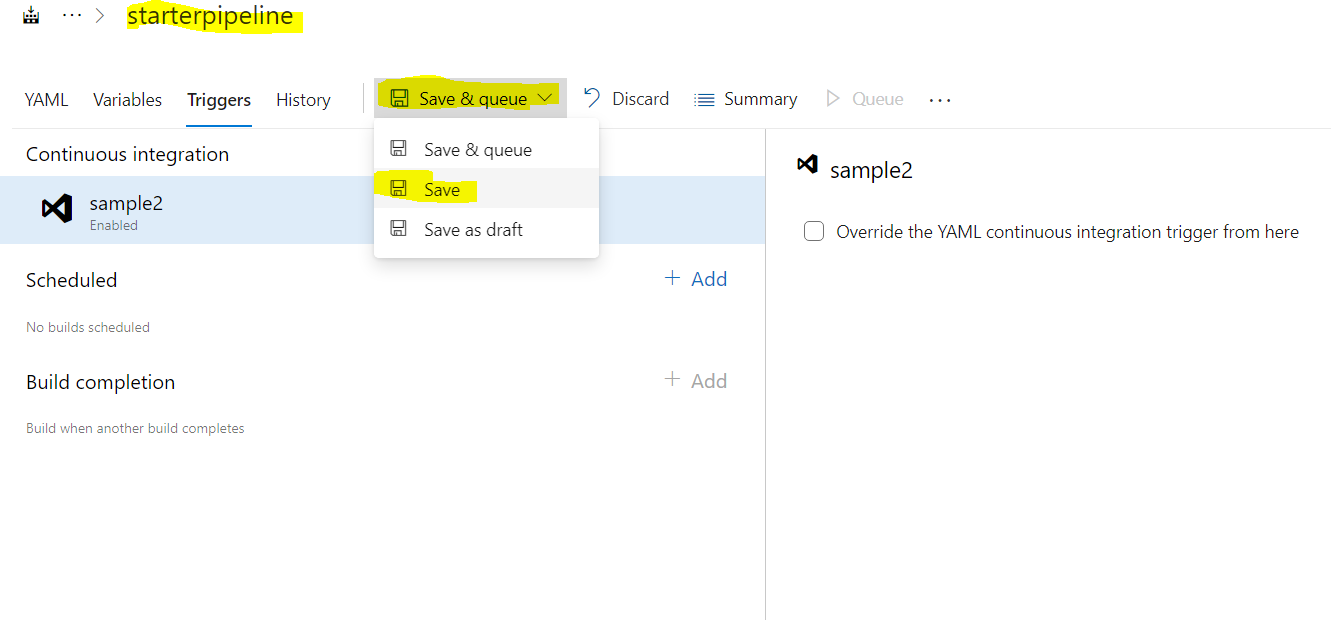
1. Click on three vertical dots. And select Triggers

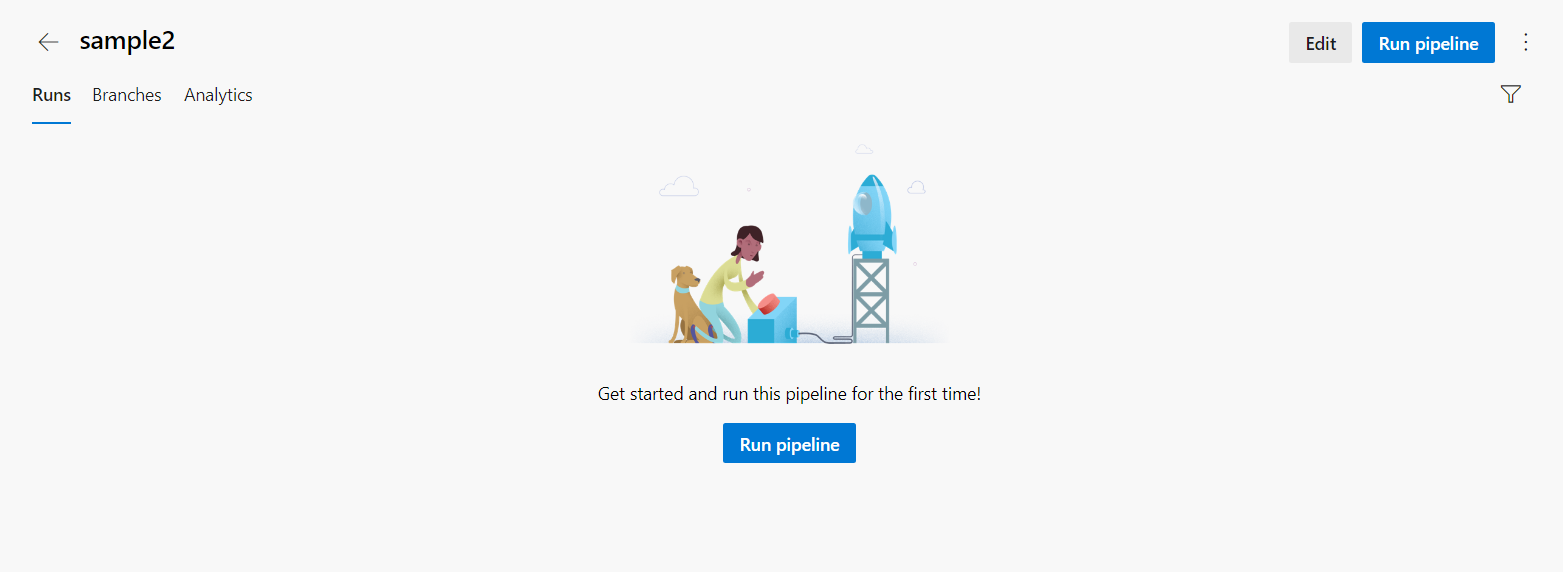


1. It will open gui editor page. Select the pipeline same then you can enter your own pipeline name what ever you want.



And then select save.





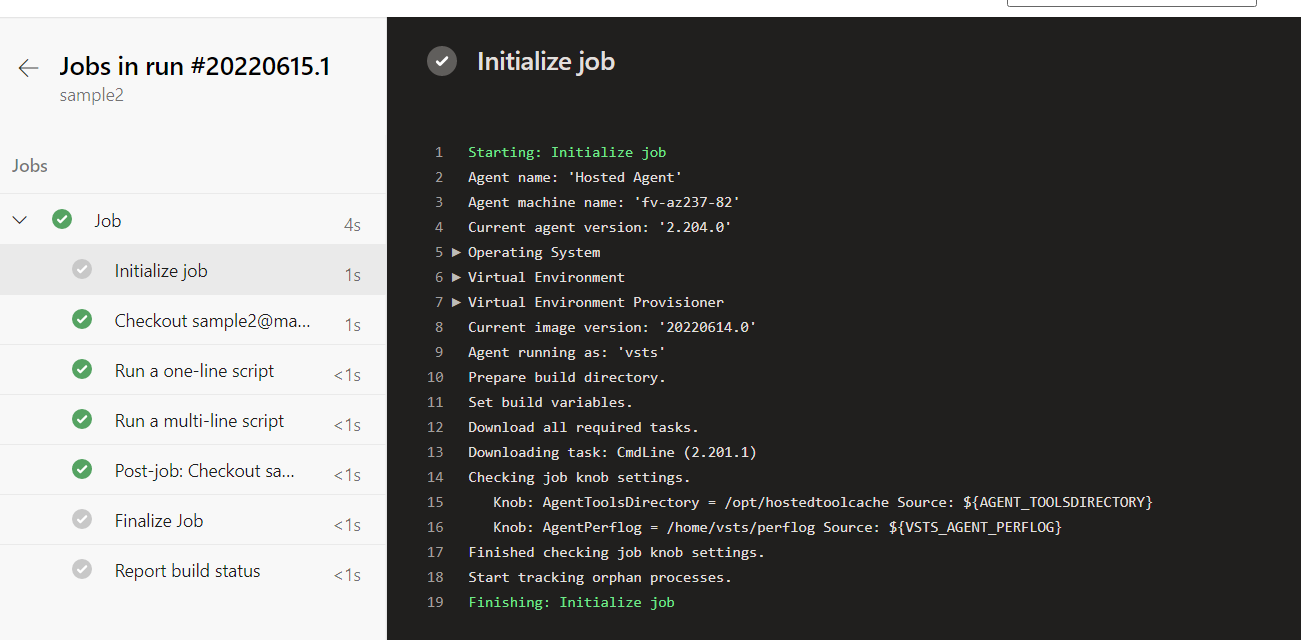
Now, click on Run Pipeline.

Here we can see the RUN logs

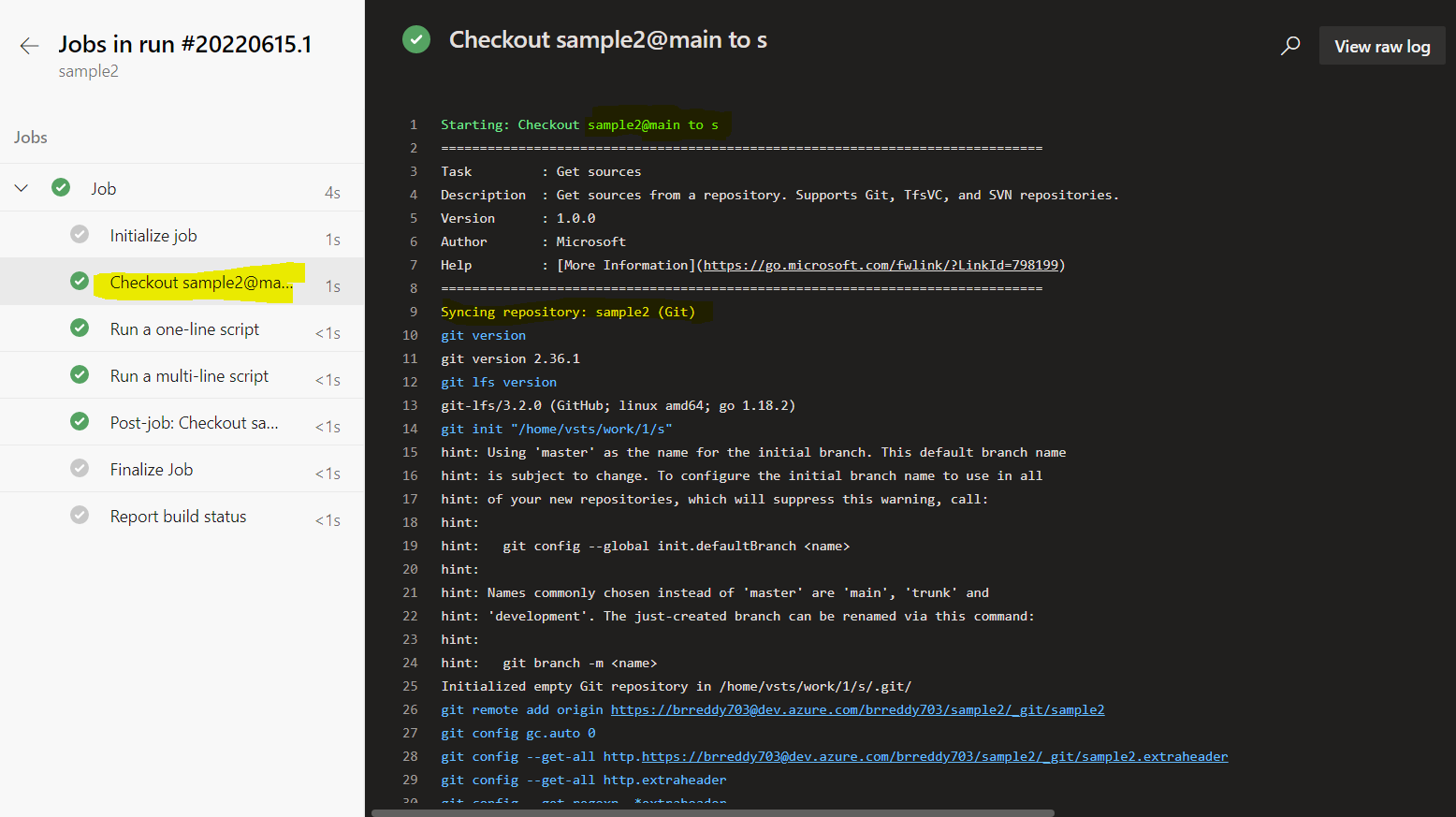
Here we can see the pool name and agent name as well.



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.