**Project: Streamline the Software Development Lifecycle for ”TechCo” using Azure DevOps**

**Background:** "TechCo";isaleadingtechnologycompanyspecializinginsoftwaresolutions.Theyhaveateamofsoftwareengineersworkingonvariousprojectssimultaneously.However,thecurrentdevelopmentprocesslacksorganizationandcollaboration,leadingtodelaysandinefficiencies.Toaddresstheseissues,"TechCo";hasdecidedtoadoptAzureDevOpsandwantsthesoftwareengineeringteamtoutilizeitscapabilitieseffectively.

**Problem Statement:**

Asasoftwareengineerat"TechCo,"yourtaskistoleverageAzureDevOpstostreamlinetheSoftwareDevelopmentLifecycle(SDLC)foranewprojectcalled"ProjectX."Thisprojectaimstodevelopawebapplicationformanagingcustomerorders.

**Requirements:**

1.CreationofBoards:a.Createan"Epic"inAzureDevOpsfor"ProjectX."TheEpicshouldrepresenttheoverallgoaloftheproject.b.BreakdowntheEpicintousercentric"UserStories"withclearacceptancecriteria.EachUserStoryshouldrepresentaspecificfunctionalityofthewebapplication.c.CreatetaskswithineachUserStorytodefinethespecificworkitemsrequiredforimplementation.2.UtilizeRepos:a.CreateanewGitrepositoryinAzureDevOpstohostthesourcecodefor"ProjectX."b.Initializetherepositorywithabasicprojectstructurethatincludesnecessaryfilesandfoldersforthewebapplication.c.Collaborativelyworkonthesourcecodeusingbranchesandpullrequests.Ensurethateachfeatureorbugfixisdevelopedonaseparatebranchandmergedbacktothemainbranchafterreview.3.BuildCICDPipelines:a.ConfigureaContinuousIntegration(CI)pipelinetoautomaticallybuildthewebapplicationwheneverchangesarepushedtothemainbranchofthe repository.b.SetupautomatedtestingaspartoftheCIpipelinetoensurethecodequality.c.ImplementaContinuousDeployment(CD)pipelinetoautomaticallydeploythewebapplicationtoastagingenvironmentwheneverchangesaresuccessfullybuiltandpassthetests.d.Enablemanualapprovalgatesbeforedeployingthewebapplicationtotheproductionenvironment.

**Deliverables:**

1.CreateandorganizeEpics,UserStories,andTasksonAzureDevOpsBoardsfor"ProjectX."2.Developthesourcecodefor"ProjectX"onAzureDevOpsRepos,utilizingbranchesandpullrequestsforcollaboration.3.ConfigureaCIpipelinetobuildthewebapplication,includingautomatedtesting.4.ImplementaCDpipelinetodeploythewebapplicationtostagingandproductionenvironments,withmanualapprovalgates.

**Evaluation Criteria:**

1.ProperorganizationofEpics,UserStories,andTasksonAzureDevOpsBoards.2.EffectiveutilizationofAzureDevOpsReposforsourcecodemanagementandcollaboration.3.SuccessfulconfigurationofCIpipelineforbuildingthewebapplication.4.SuccessfulimplementationoftheCDpipelinewithstagingandproductiondeployments.s

**Note:**

YouareencouragedtoexploreandutilizeotherAzureDevOpsfeaturesthatcanenhancetheSDLCworkflow,suchasAzureReposWikifordocumentationorAzureArtifactsforpackagemanagement.

**Steps:**

1. Use Azure Boards App to create EPIC, Feature, User Story, Tasks, Test Cases, Defects and Execution of Sprints.

2. Create Maven project in Visual Studio and commit the same on Azure Repo.

3. Use Azure Repo to Create feature branch from main, commit code to feature branches, create pull request to merge into main branch.

4. Access for Agent Pool

5. Create a VM for configuring the Self Hosted Agent

6. Configure Agent on VM

7. Create Container based Web App for SonarQube Server Set up

8. Create Code based Web App for Deployment of our App

9. Install and Configure SonarQube in Azure DevOps

10. Install White source Mend Bolt tool

11. Create CI Pipeline

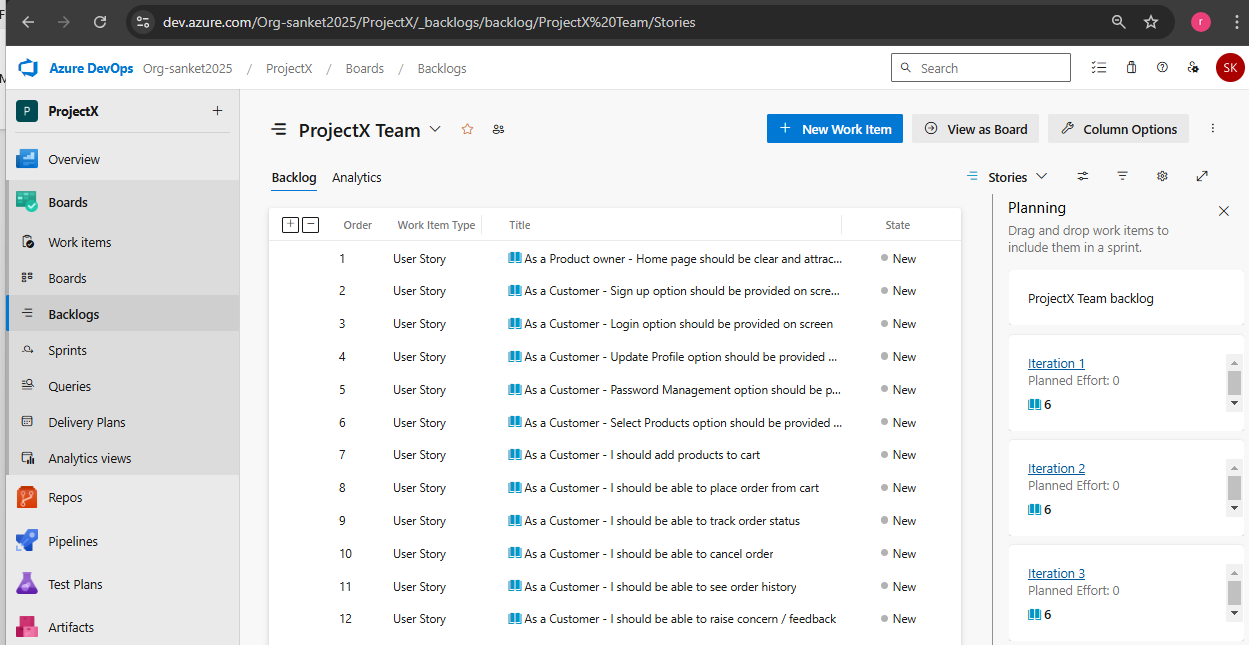
with maven package, test, SonarQube scan, White source Mend Bolt scan and publish

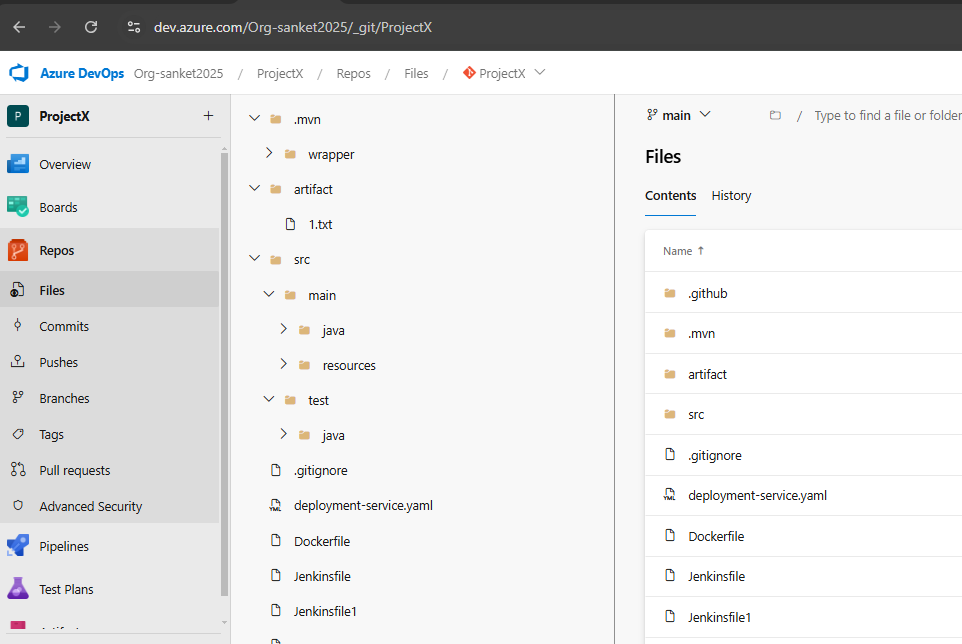
12. Create Release Pipeline

13. Triggered execution of CI and deploy on UAT, deploy on Prod after manual approval

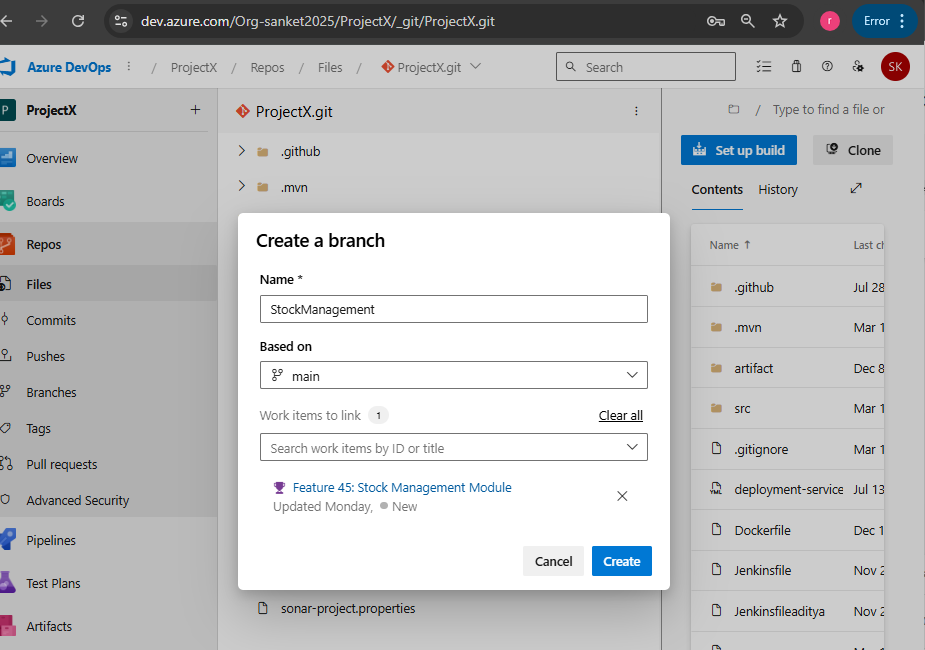
**Azure Boards:**

Using Import Work Items option, uploaded Epic, Backlog and Tasks into Azure Board

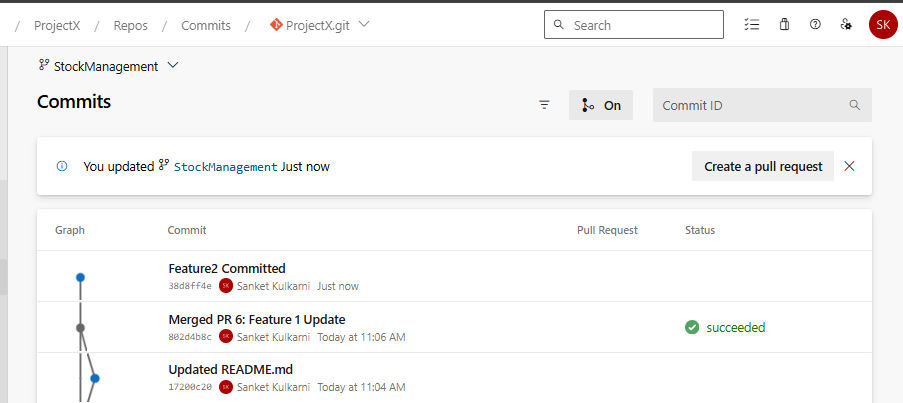
Backlog distributed across 3 sprints considering the priority and capacity of team 

**Azure Repo** -> Code commited to Azure Repository 

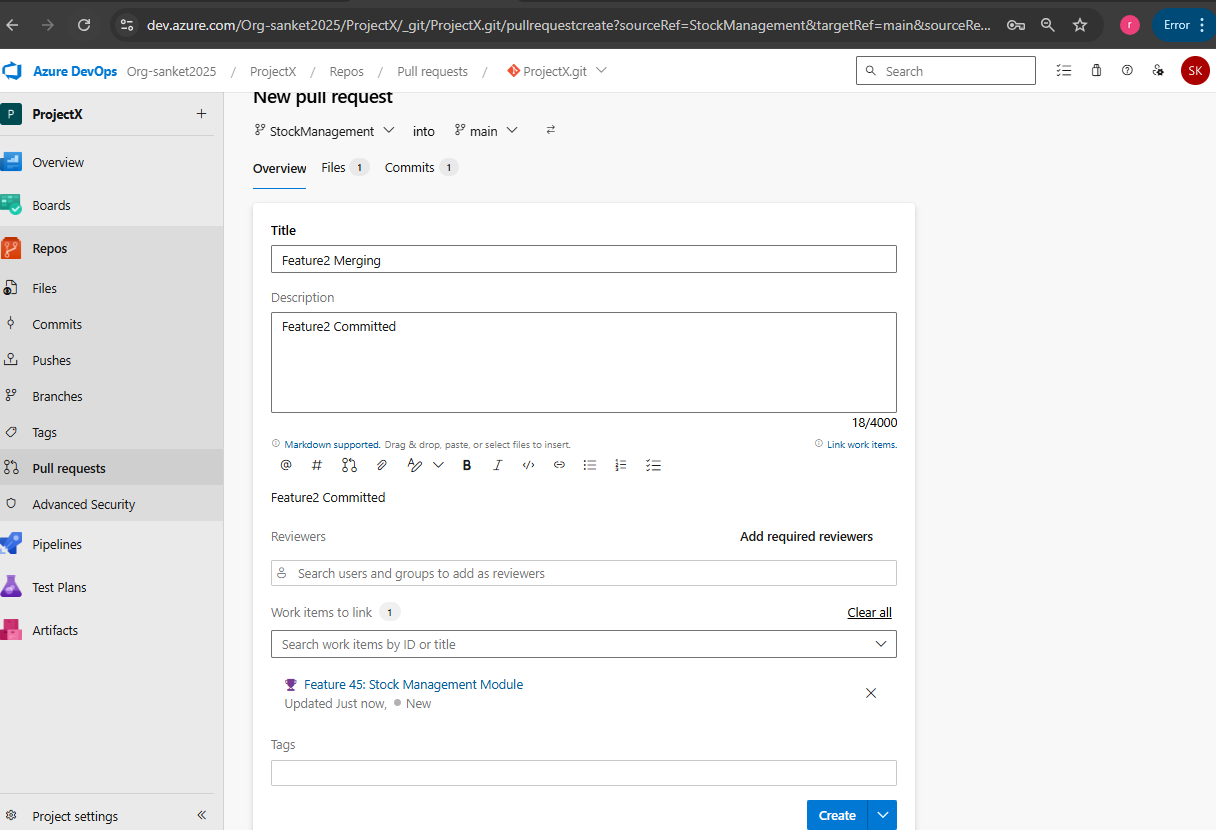
Create feature specific branch from main branch:



New feature developed and committed to feature branch.

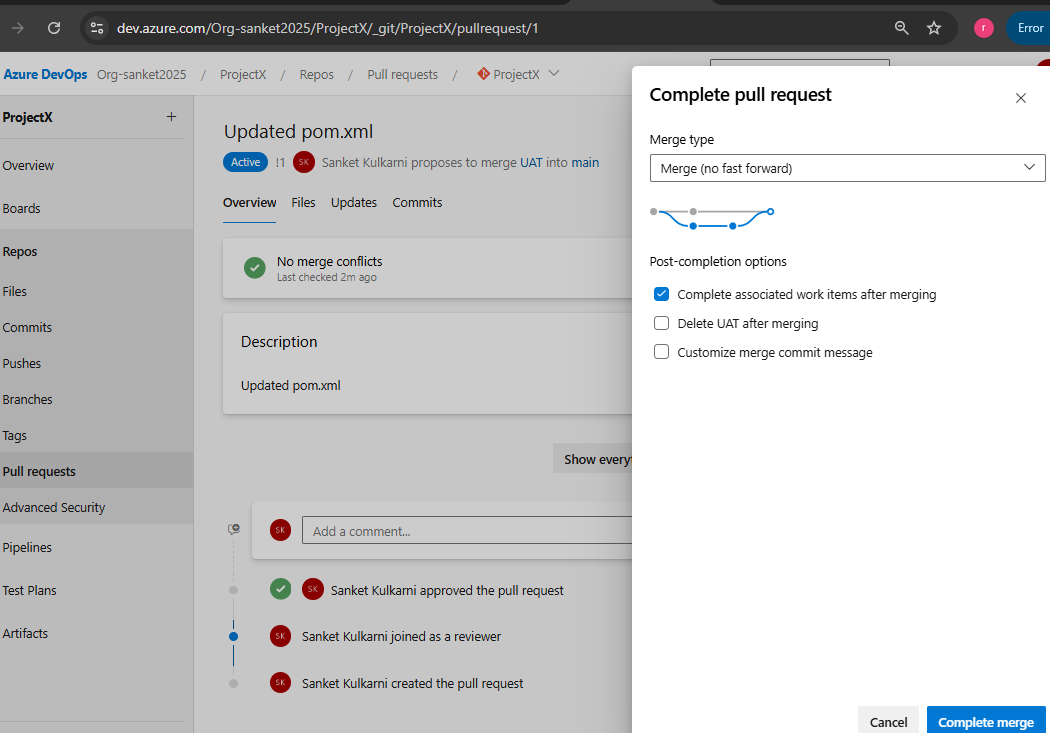


Merge Request for changes committed to Feature branch to main branch



Approve the pull request and complete.

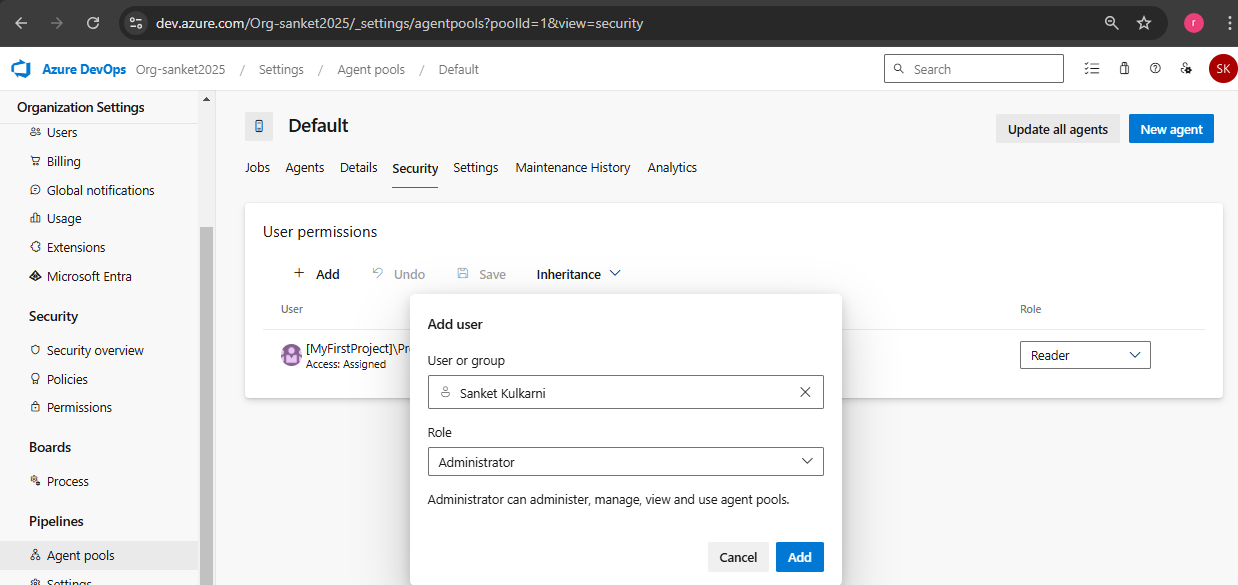
Complete Merge / pull request



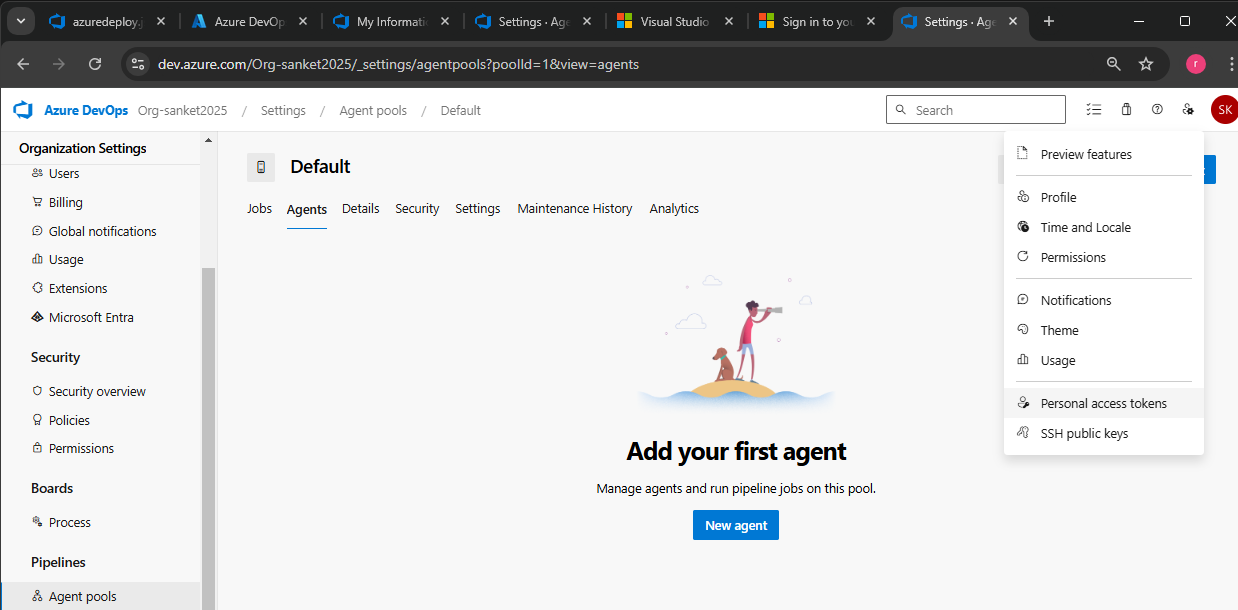
**Self Hosted Agent**

Go to Organization -> Settings -> Agent Pools -> Security

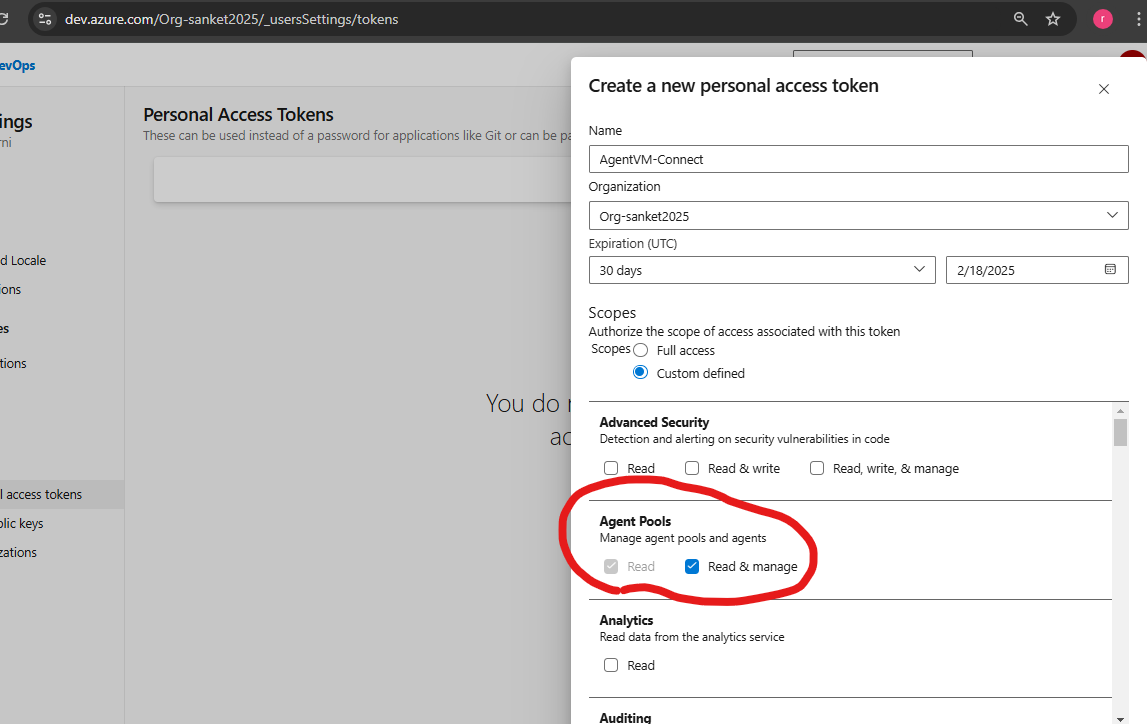
Add the account which will be used for registering the Self Hosted Agent



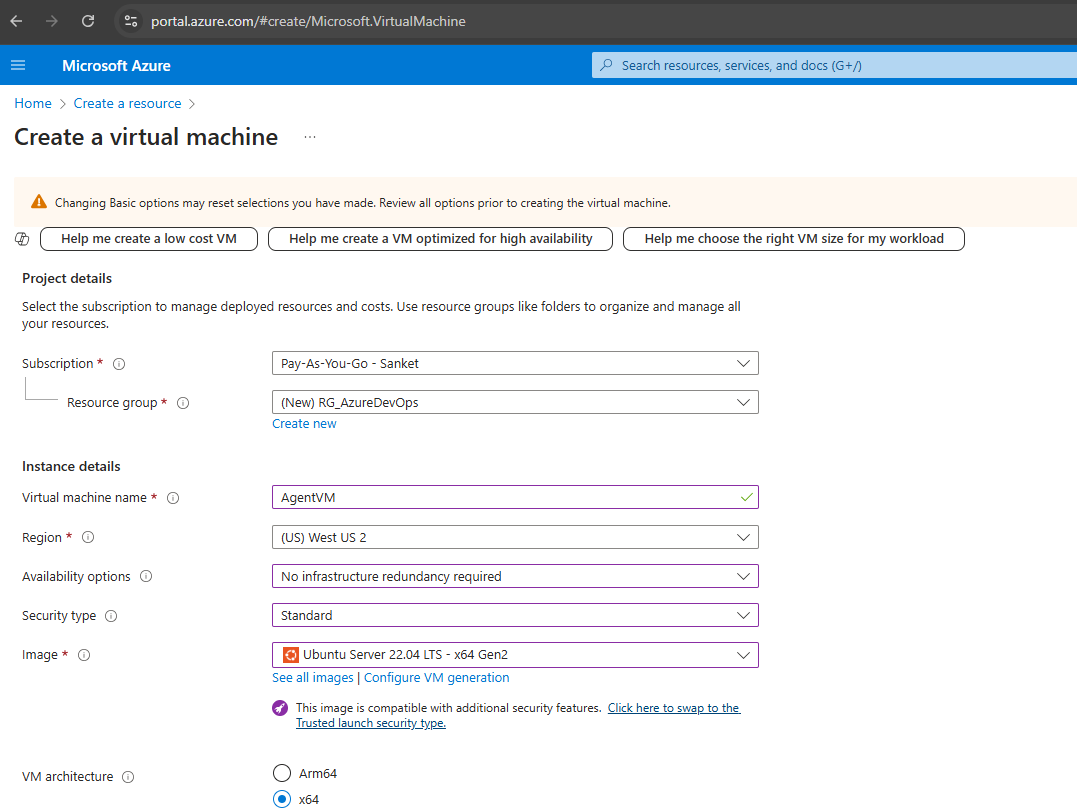
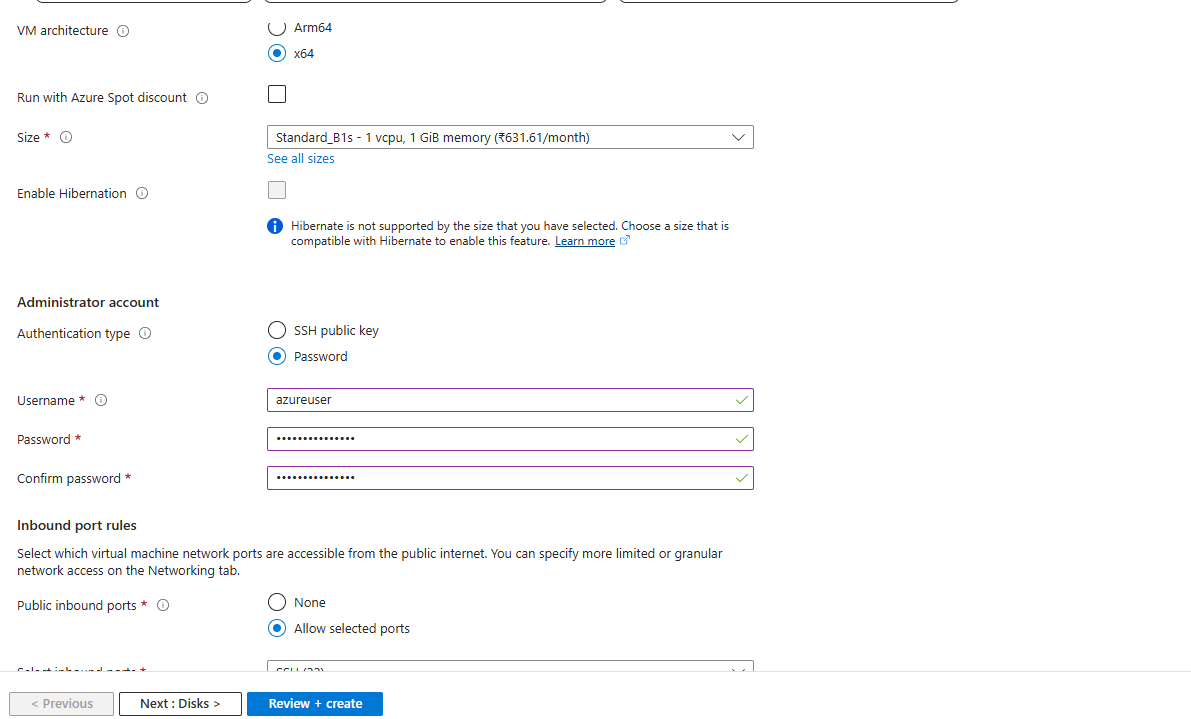
Create PAT for Agent Pool Access in Agents tab

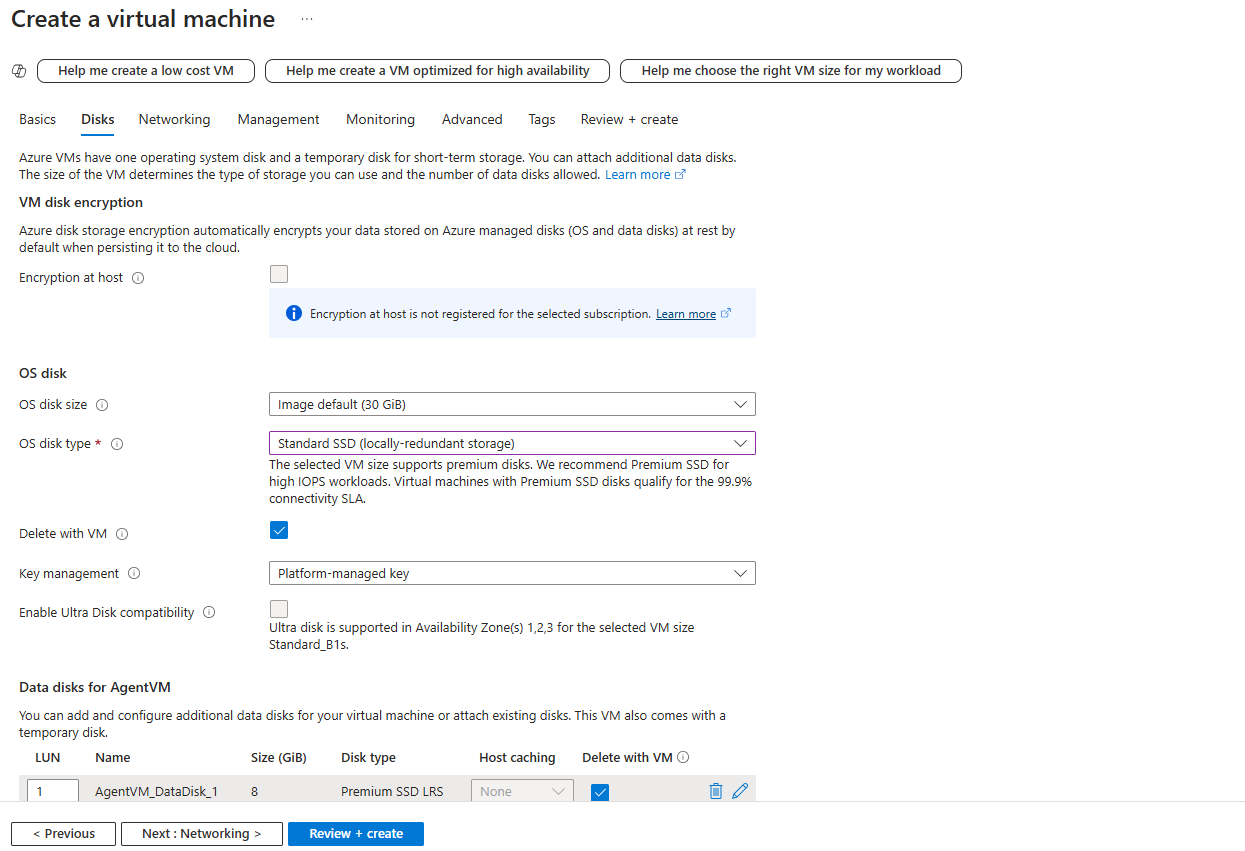
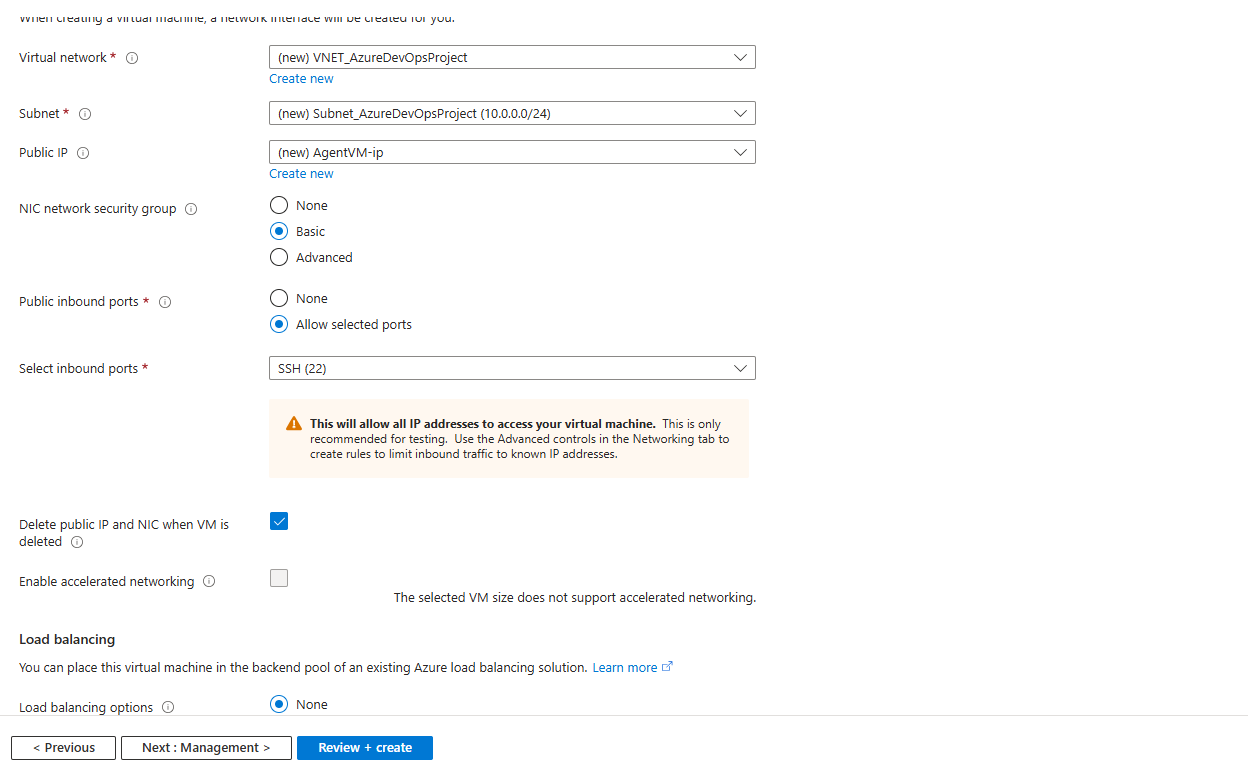


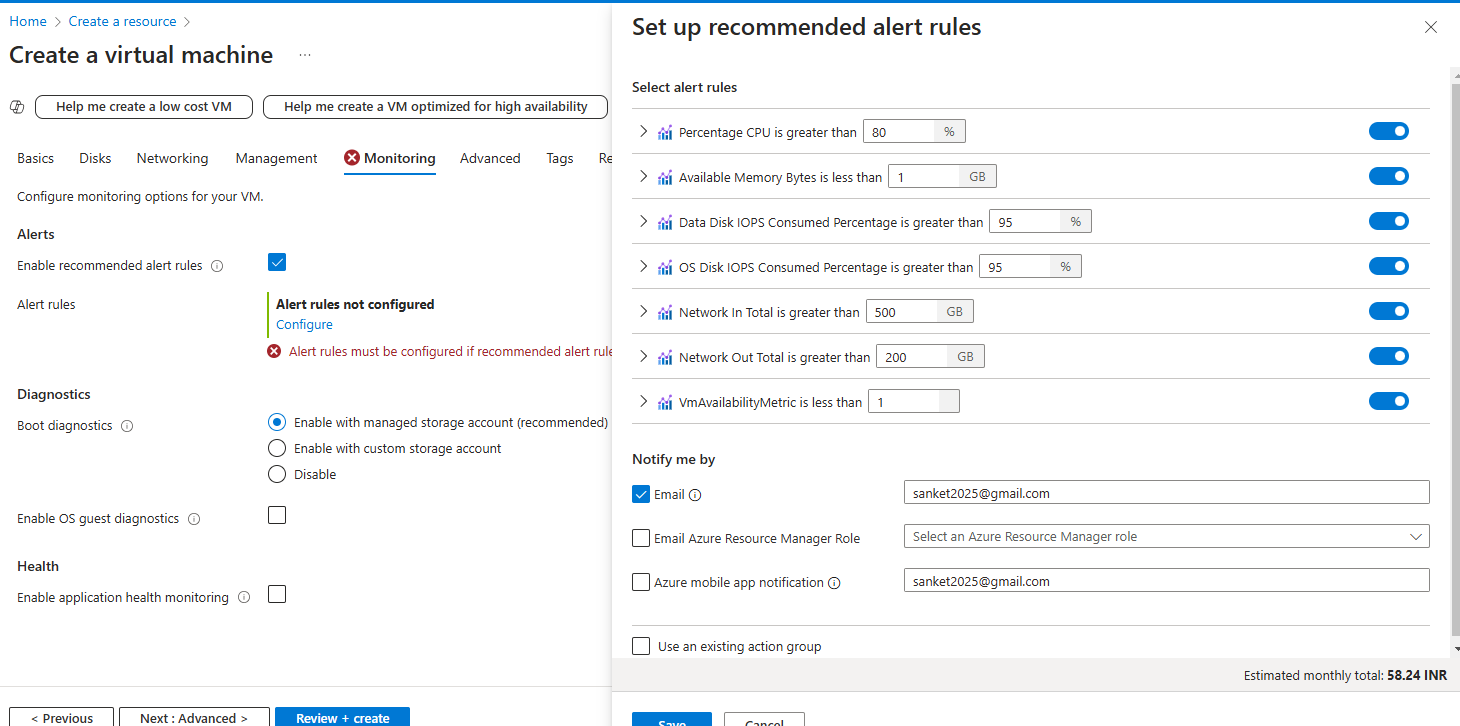
Select Agents Pool Accesses as shown below



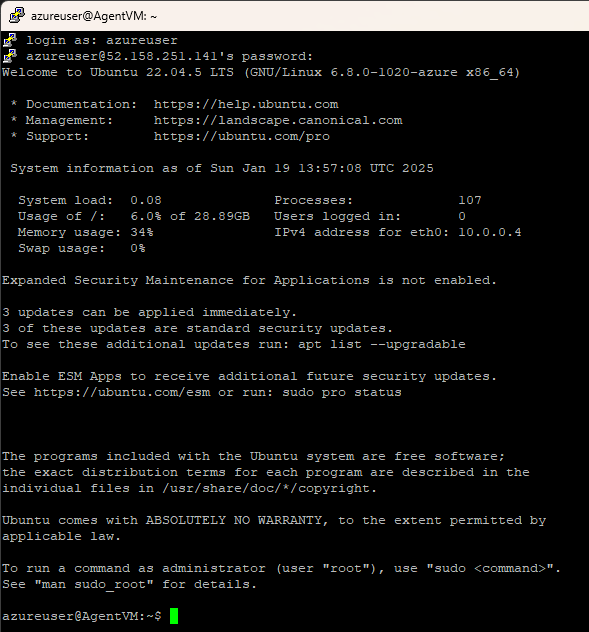
**VM Creation for Self-Hosted Agent**

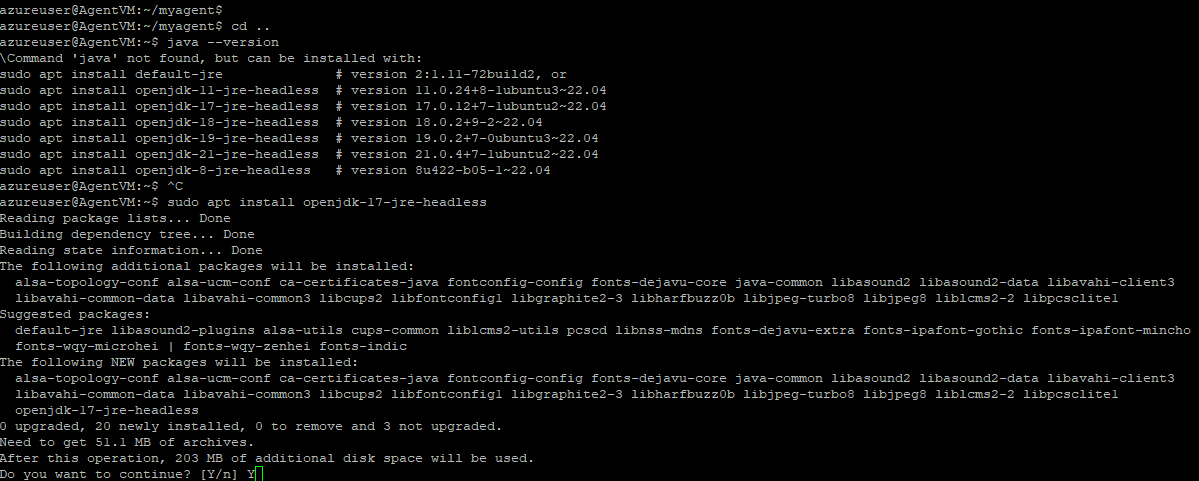
 

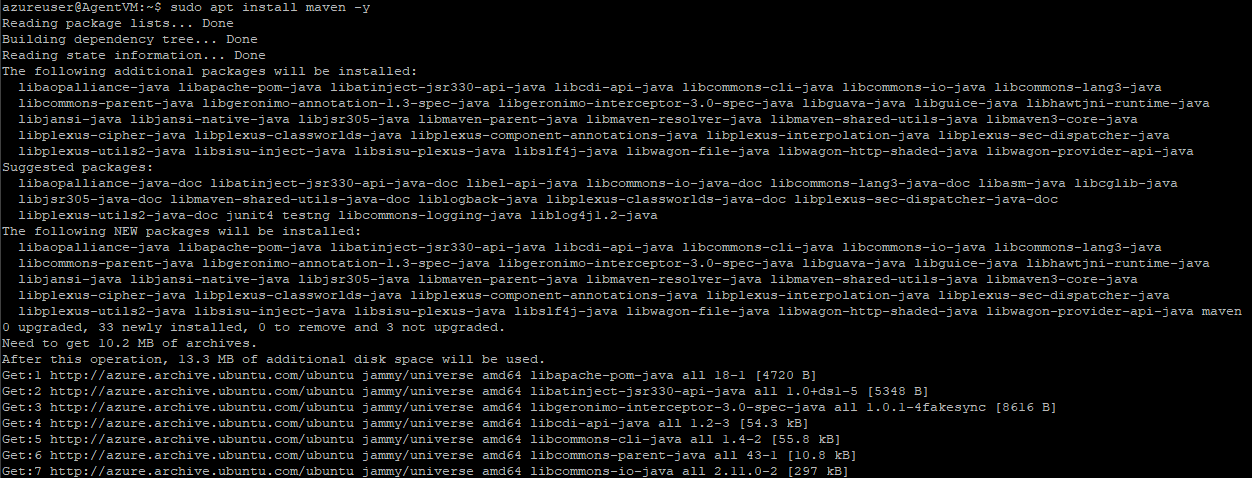


**Login to Agent VM using Putty**



Install Java on the VM



Install maven on the VM

To Configure this VM as an Agent for our pipelines

Execute Below commands

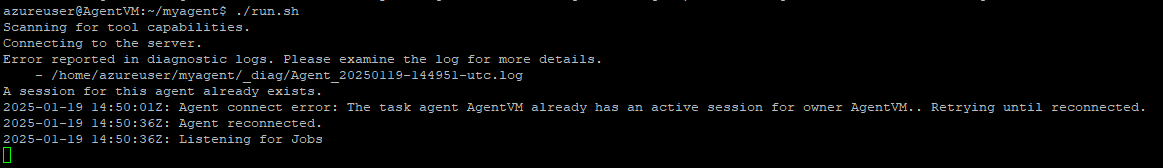
wget <https://vstsagentpackage.azureedge.net/agent/4.248.0/vsts-agent-linux-x64-4.248.0.tar.gz>

mkdir myagent && cd myagent

tar zxvf ../vsts-agent-linux-x64-4.248.0.tar.gz

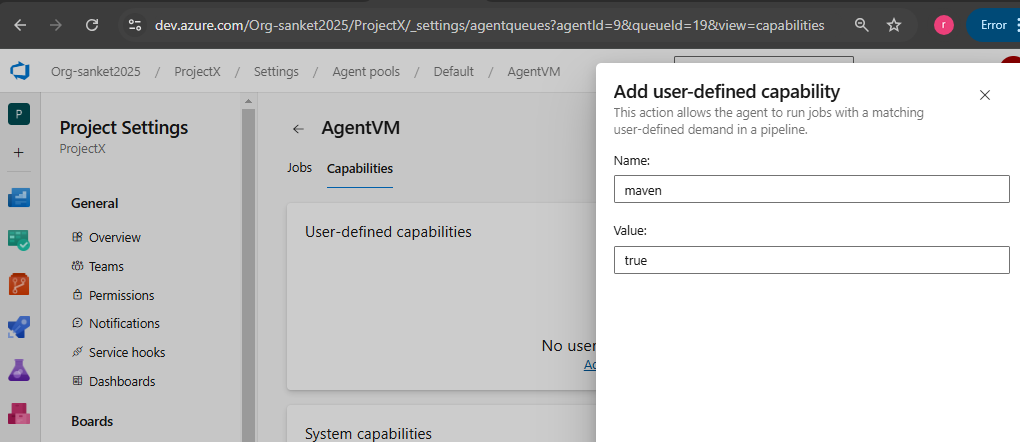
./config.sh -- Enter this command, Enter Y when asked for PAT and enter the PAT generated earlier and other inputs

Execute Command to enable the agent -> ./run.sh

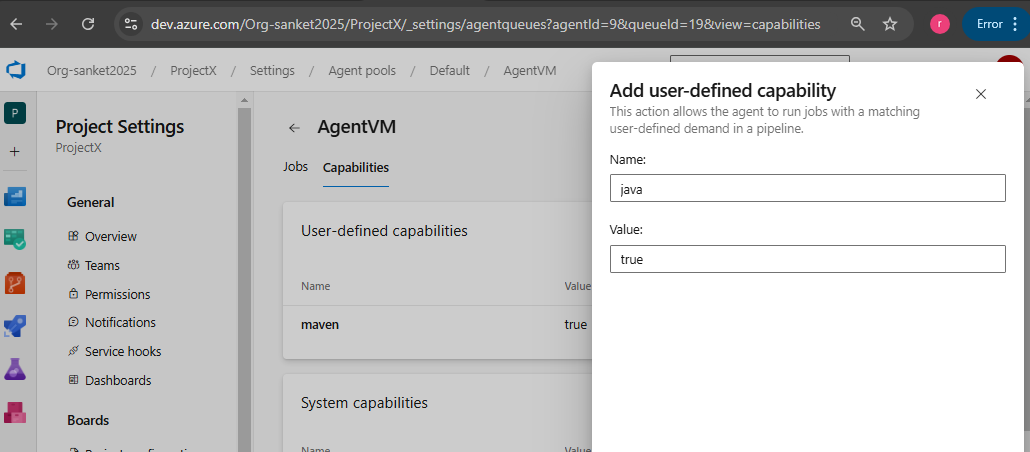


**Add Capabilities for the Agent** -> maven and java

Maven

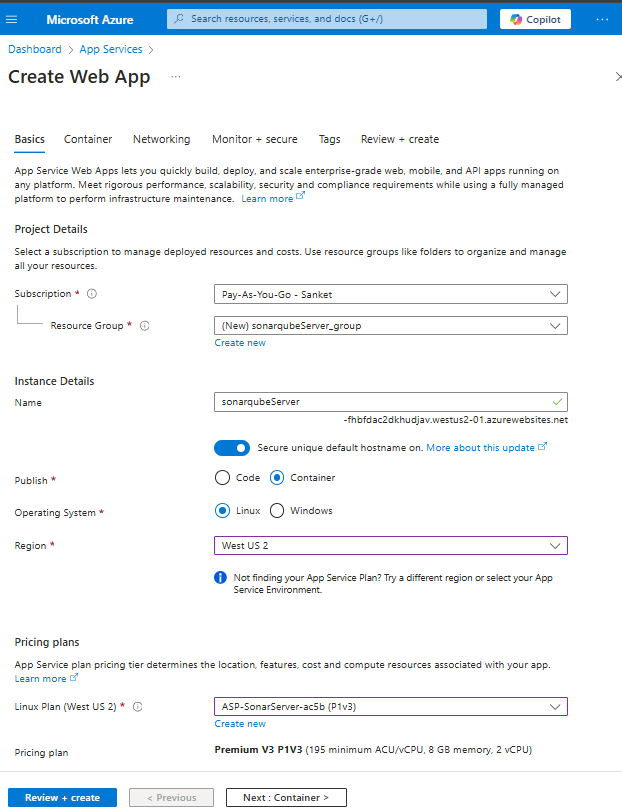
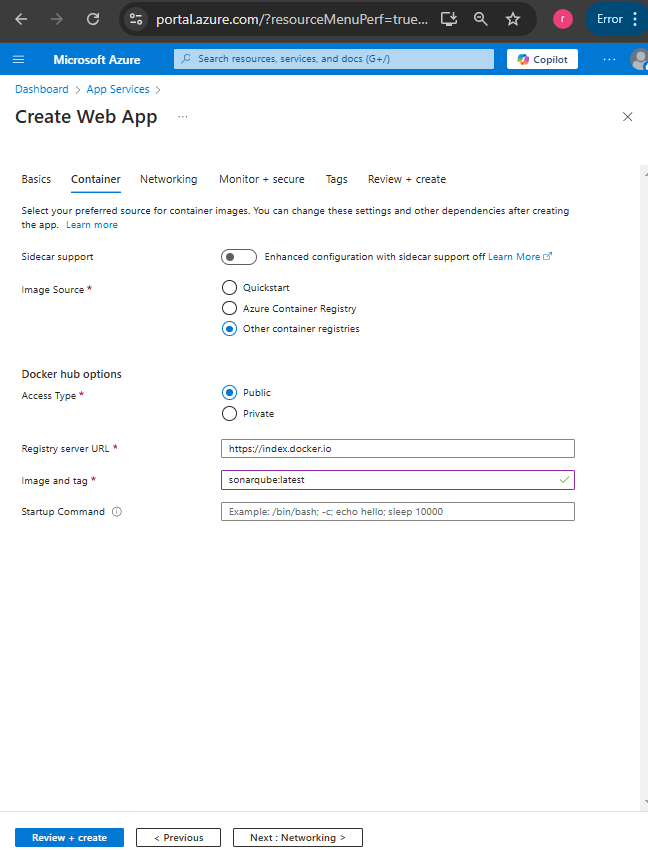


Java

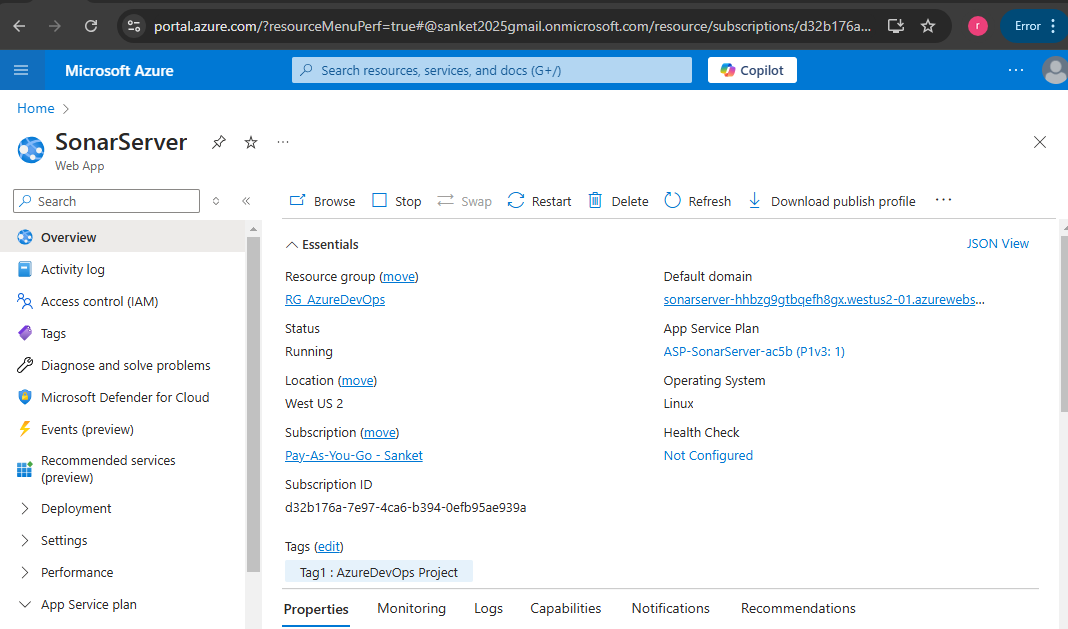


Web App for **SonarQube Server** (Similarly create Web app with code option to deploy our UAT and Prod Applications)

Creating Web app with the container image for the same – Refer container image and tag details here <https://hub.docker.com/_/sonarqube/tags>

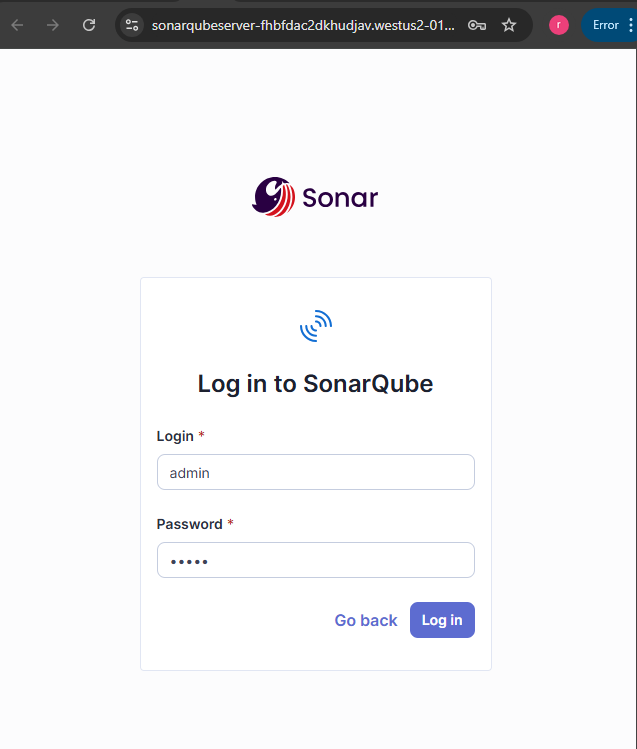
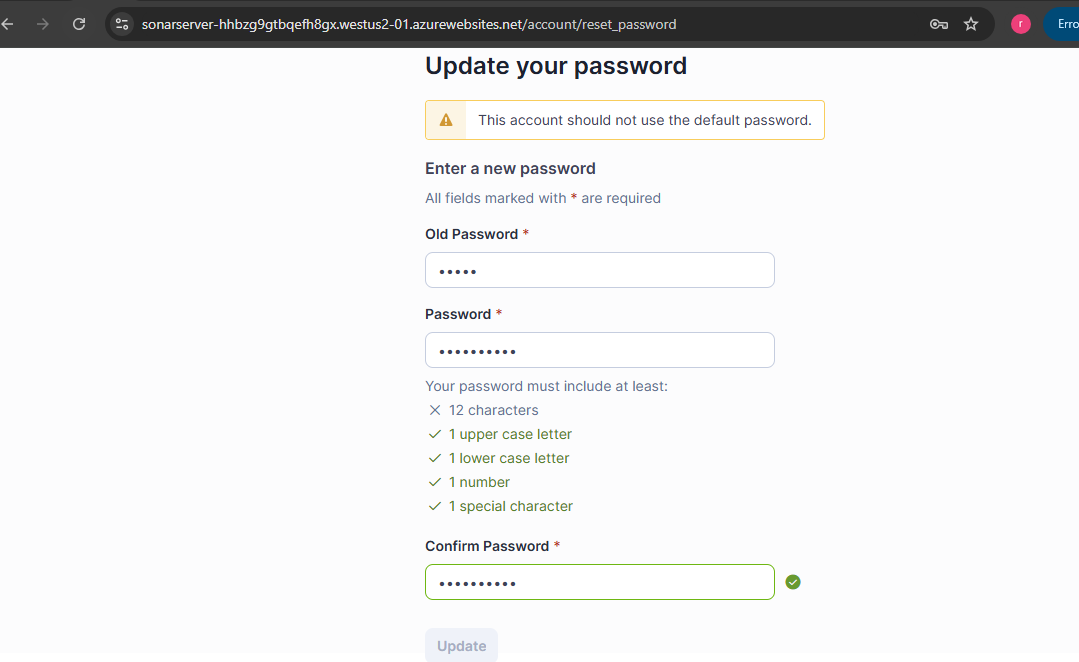
 

As its very slow changed the Plan type to Premium as below



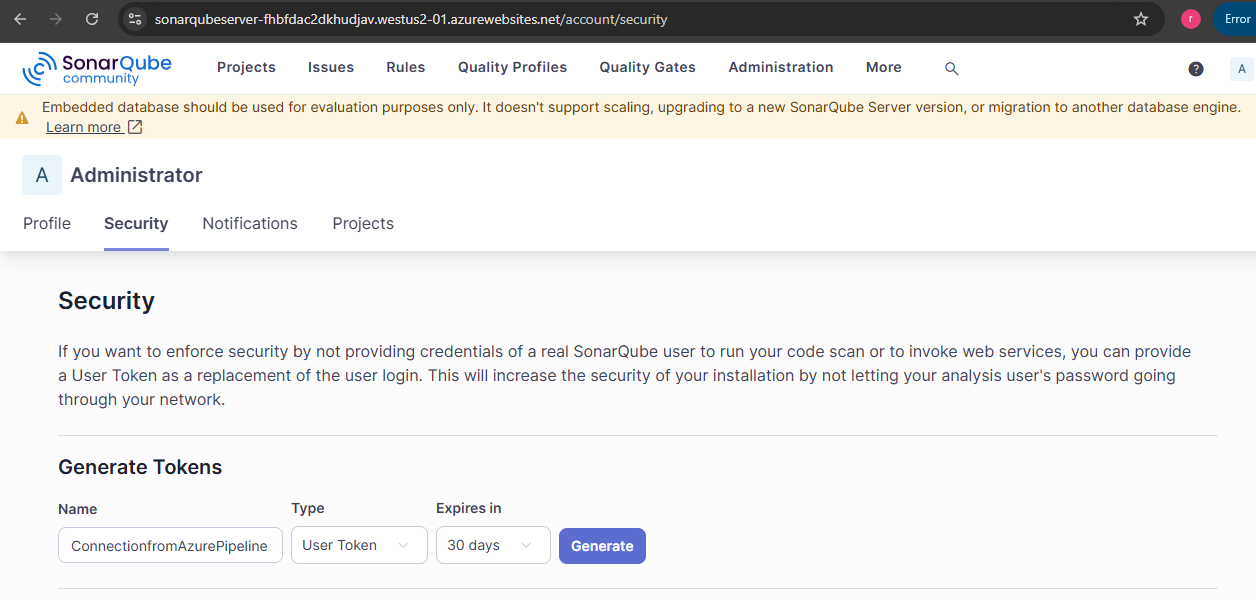
Click on Browse / Default Domain to open the sonar server URL -> Enter default credentials as below

Username – admin password – admin and update new password -> Login with new password

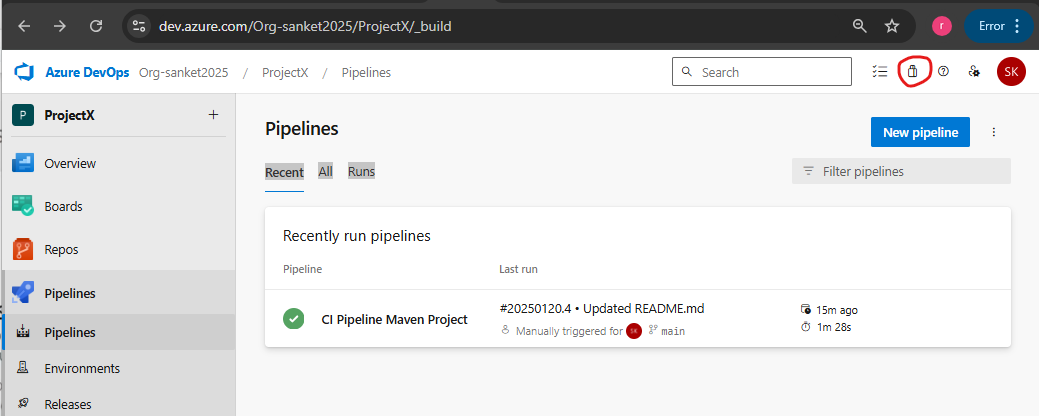
 

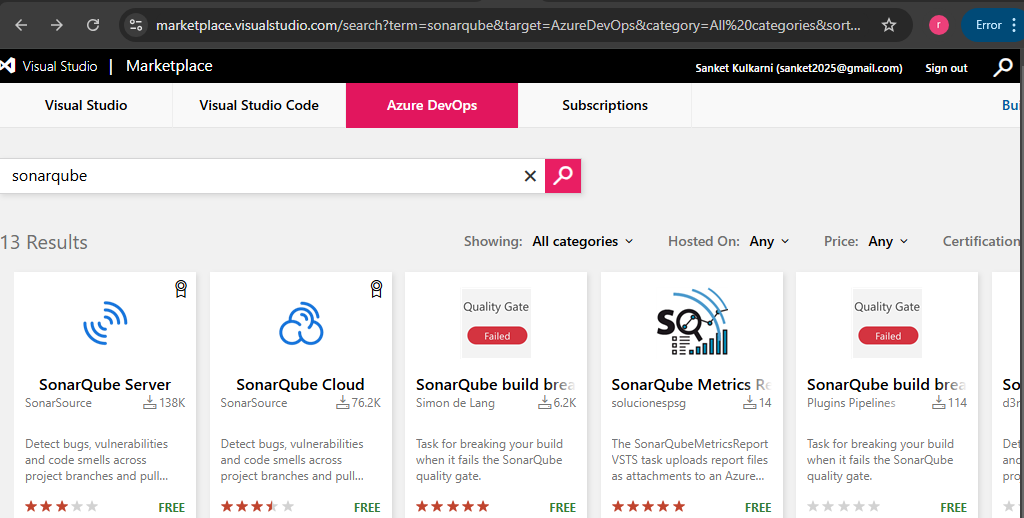
Create a Token which can be used to connect to SonarQube server from Azure Pipeline

Click on Symbol A at the top right corner -> My Account -> Security -> Generate Tokens -> Add details and Generate

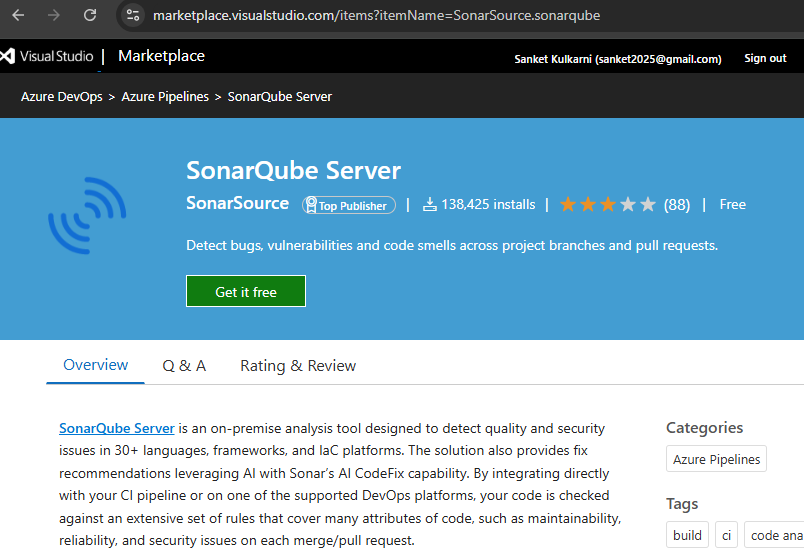
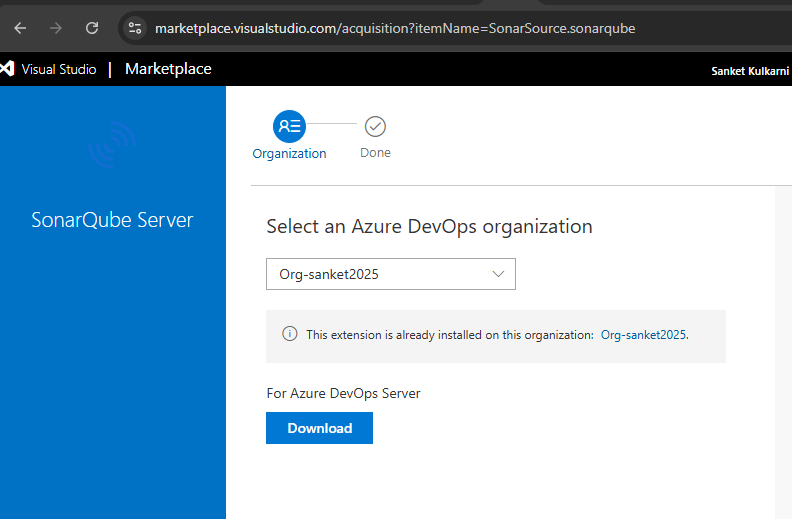


Install SonarQube Cloud Extension / Plugins in Azure Devops: Click symbol shown below -> Browse Marketplace

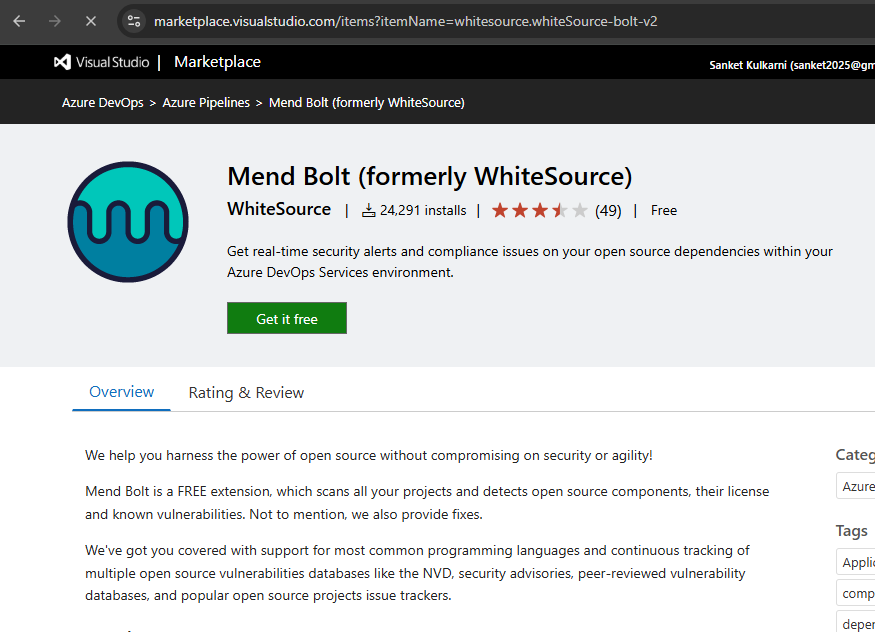


Search SonarQube and click on the symbol

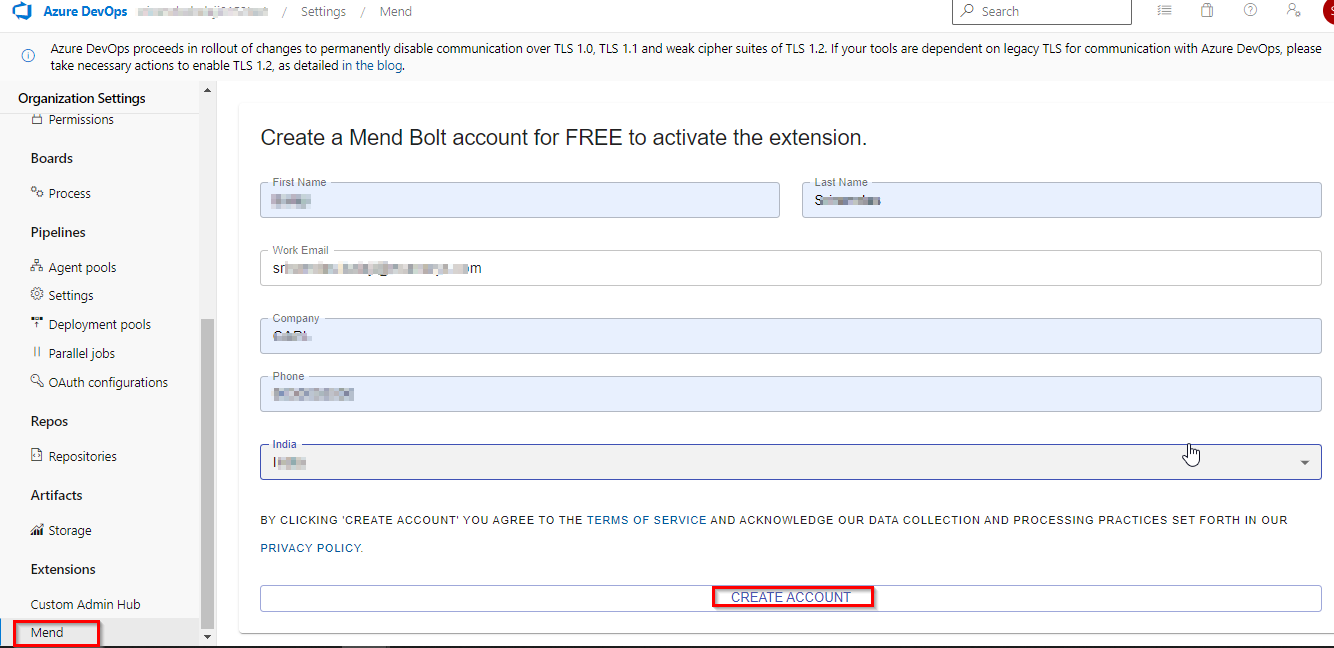
Click on get it for free Select Organization and click on Install

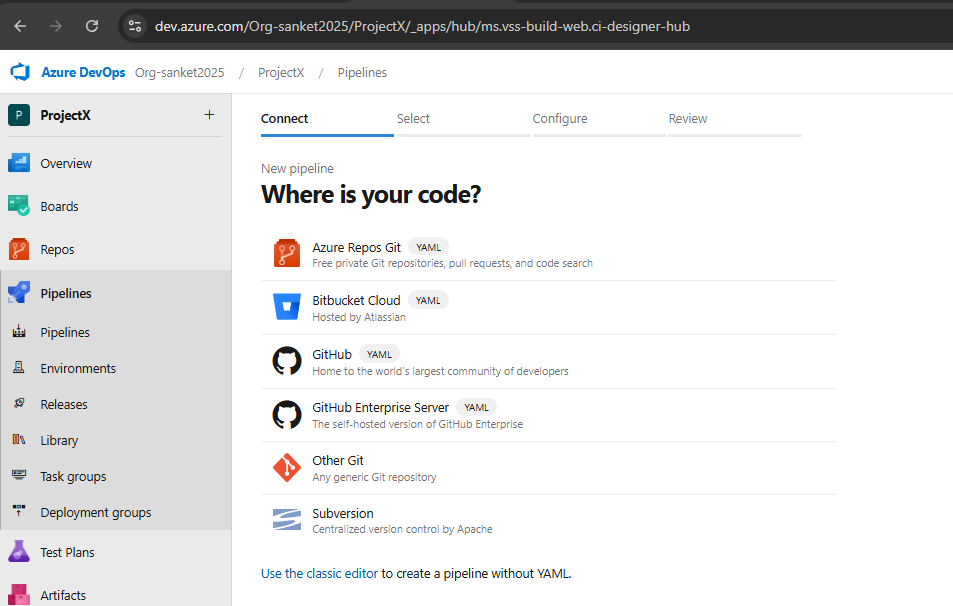
Install Mend Bolt (WhiteSource) also in same way



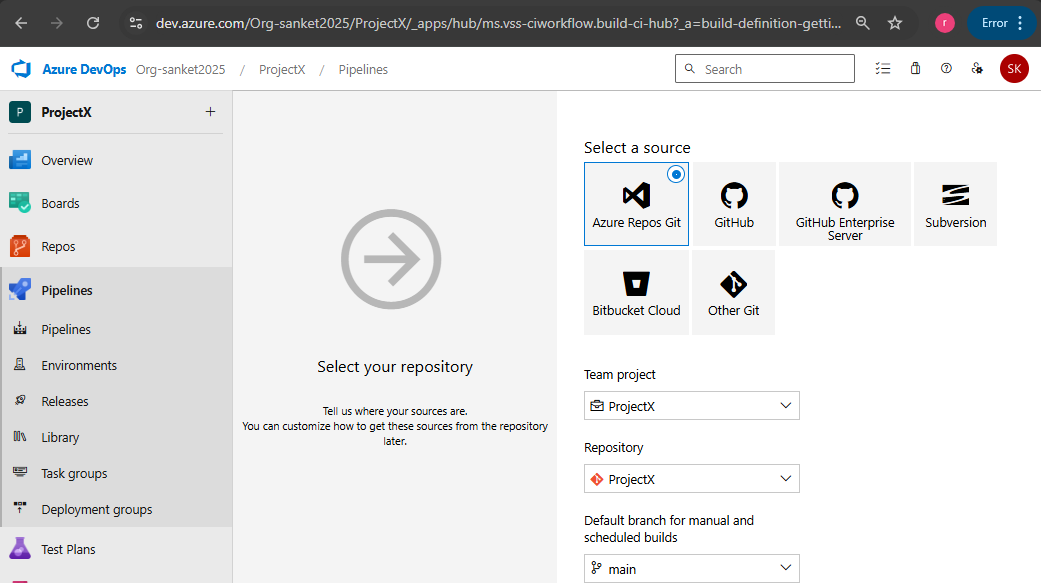
Go to Organization Settings -> at the bottom click on Mend -> Fill the form and Click on Create Account



Go to Pipeline Menu -> Create Pipeline -> Select Option at the bottom – Use the classic editor

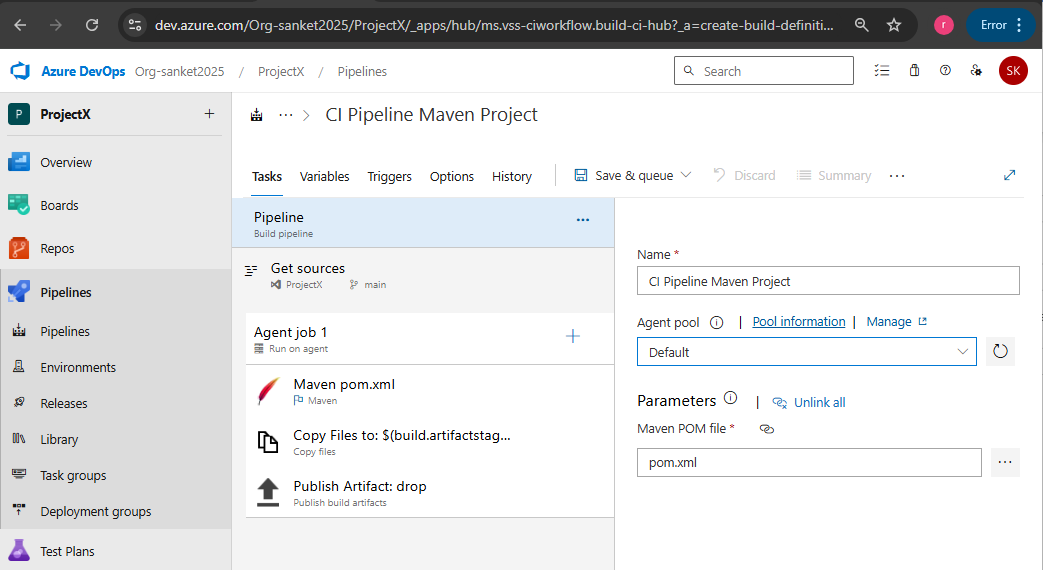


Select Repository and other details as shown below –> Click on Continue

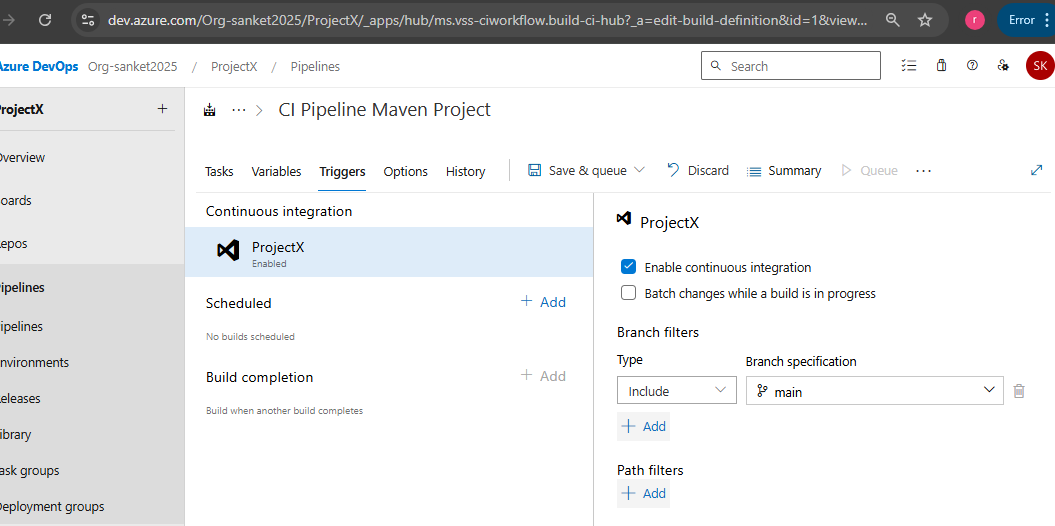


Search Maven Task and click on Apply

Edit the Name as required, Select Agent Pool as Default (for which we configured Self Hosted Agent earlier)

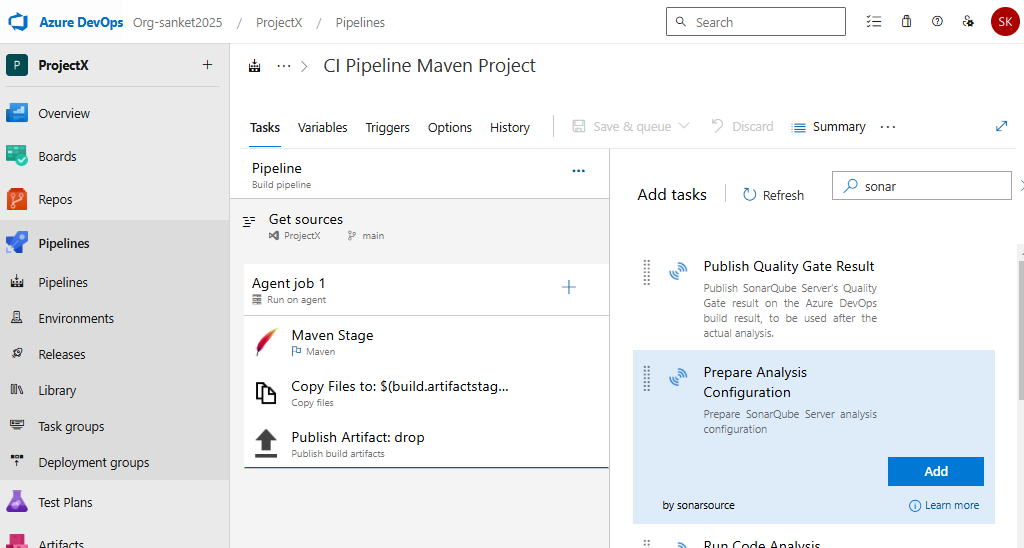


Add Trigger for CI Pipeline



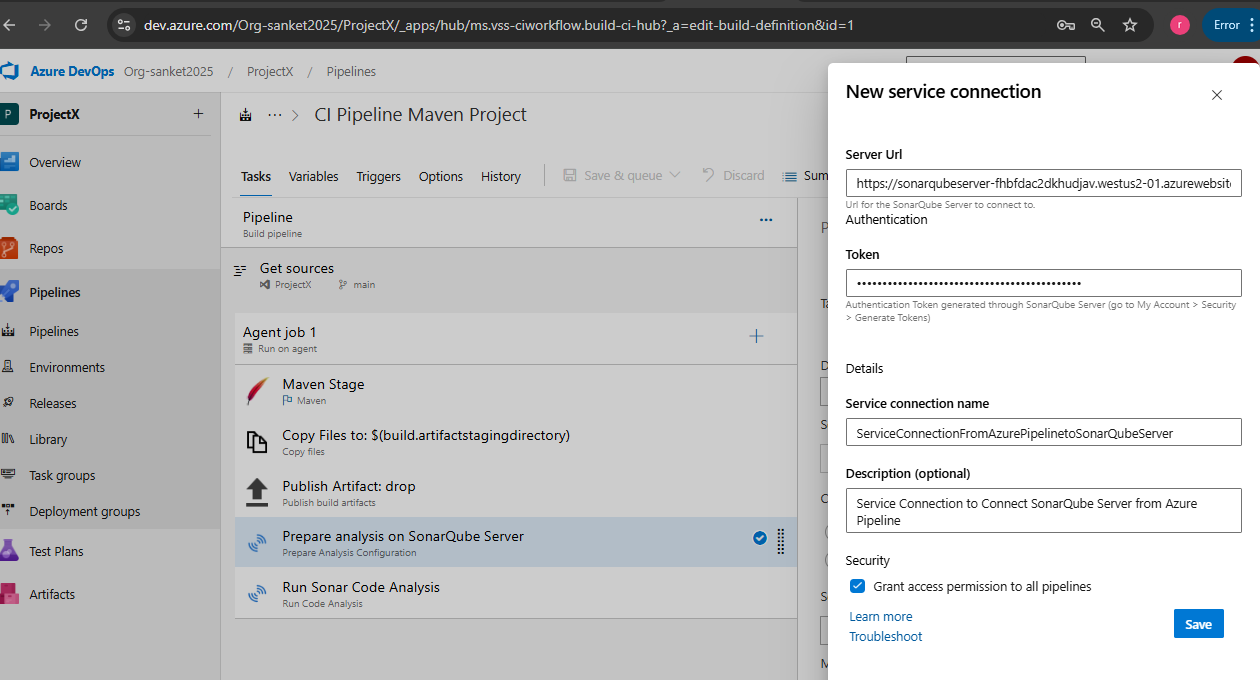
Add SonarQube Tasks in Azure Pipeline

Click + symbol to add new task and select Sonar and click on add for Prepare Analysis Configuration

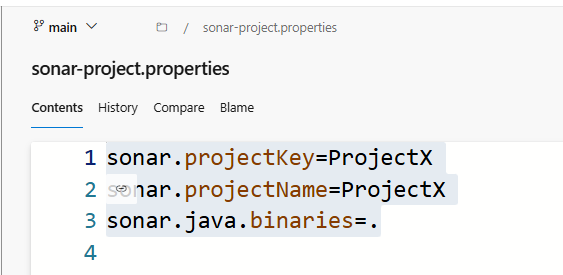


Click on +New button to add a new Service Connection as below -> Click on Save

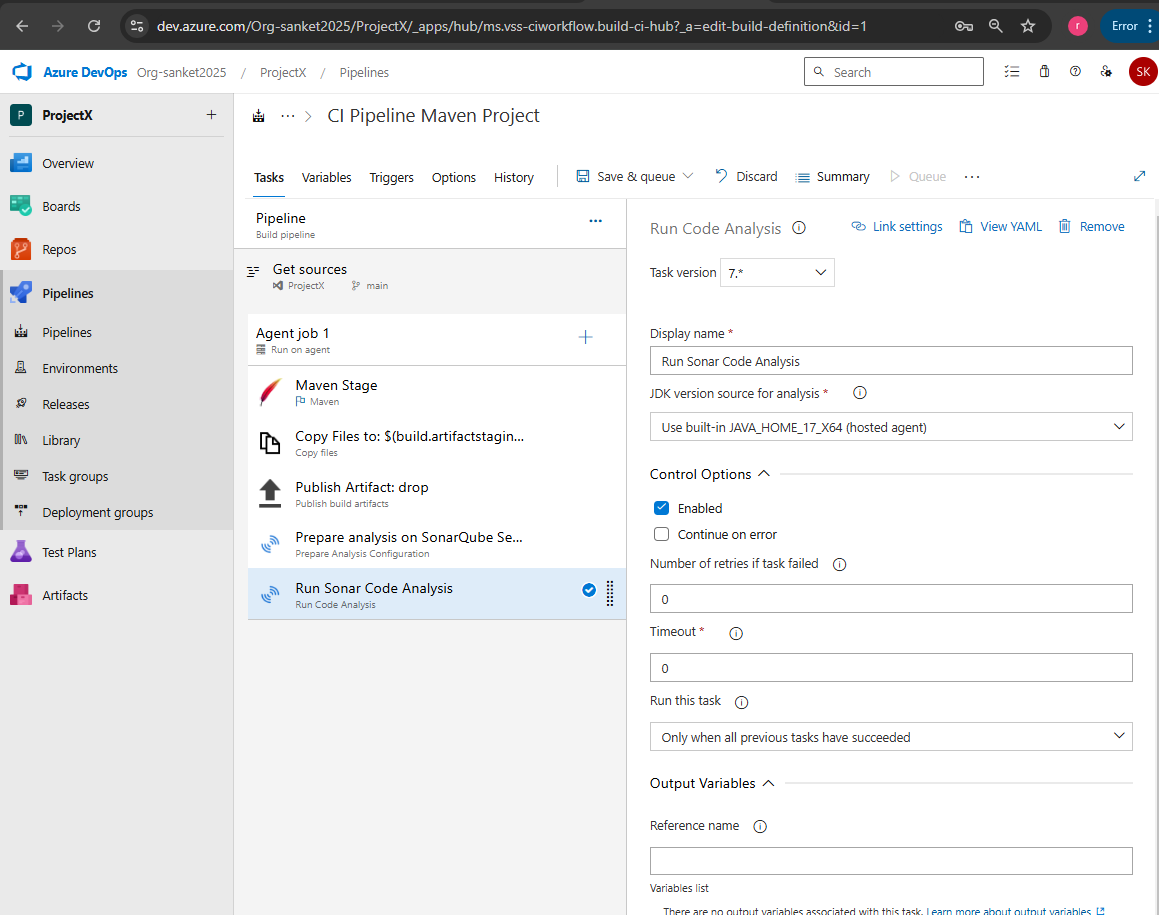
Server URL : Take URL from Resource name (Web App created for SonarQube Server) including Https://



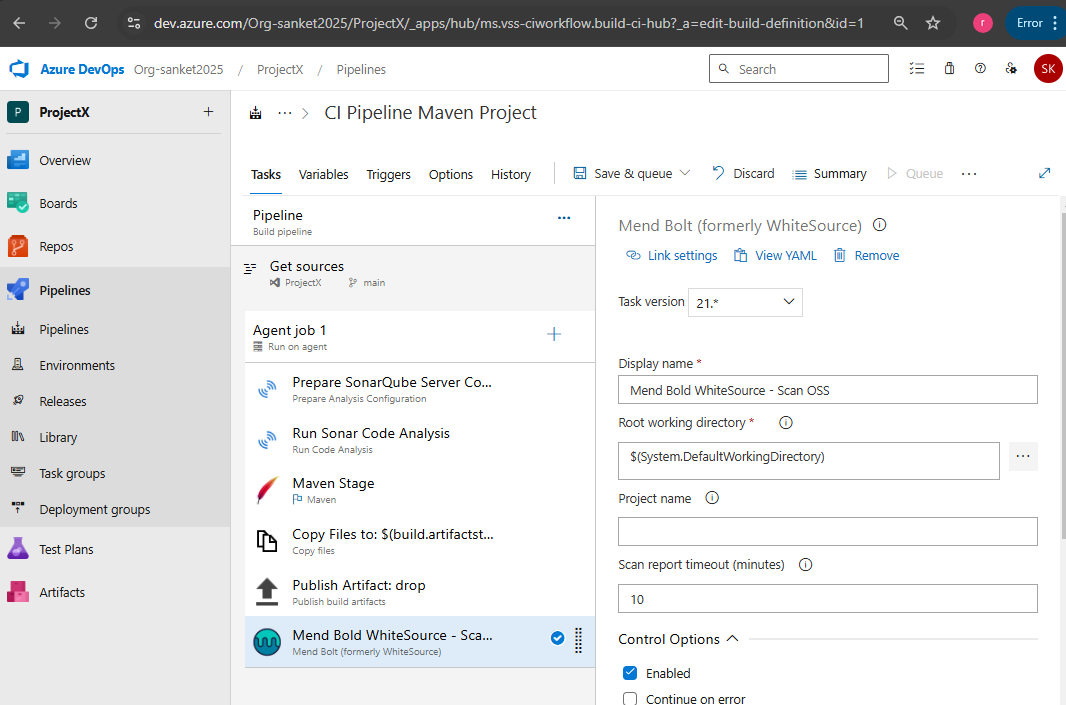
Sonar-project.properties file is as below (available in Repo)



Add another task to run sonar Code Analysis and click on Save

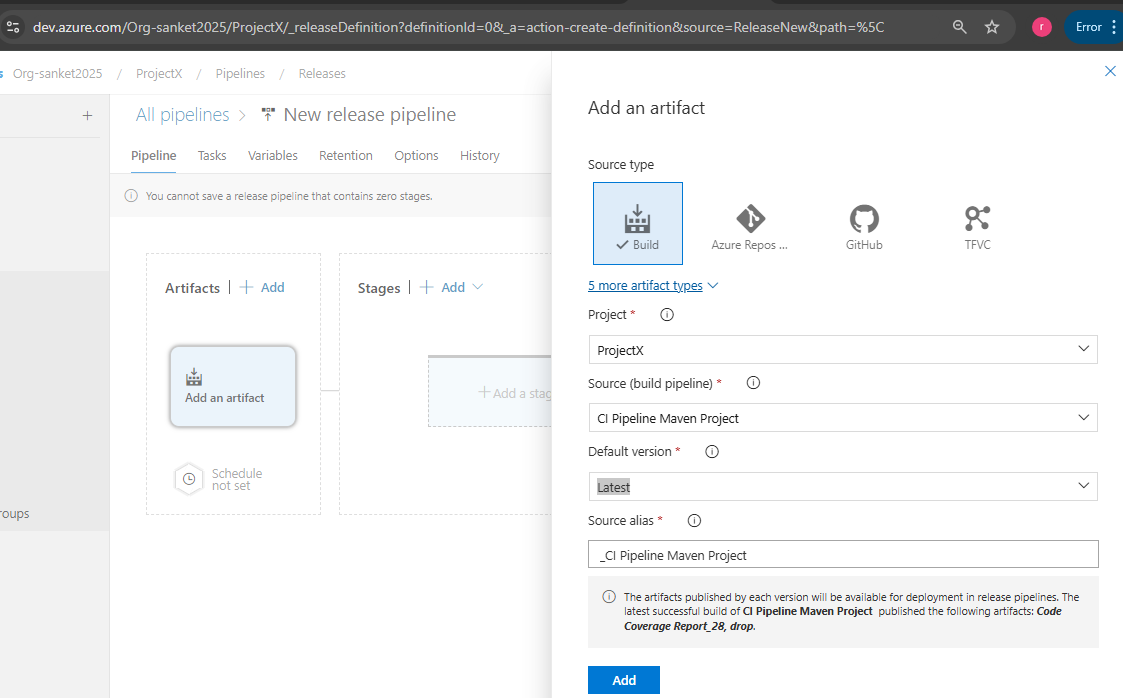


Add another task for Mend Bolt

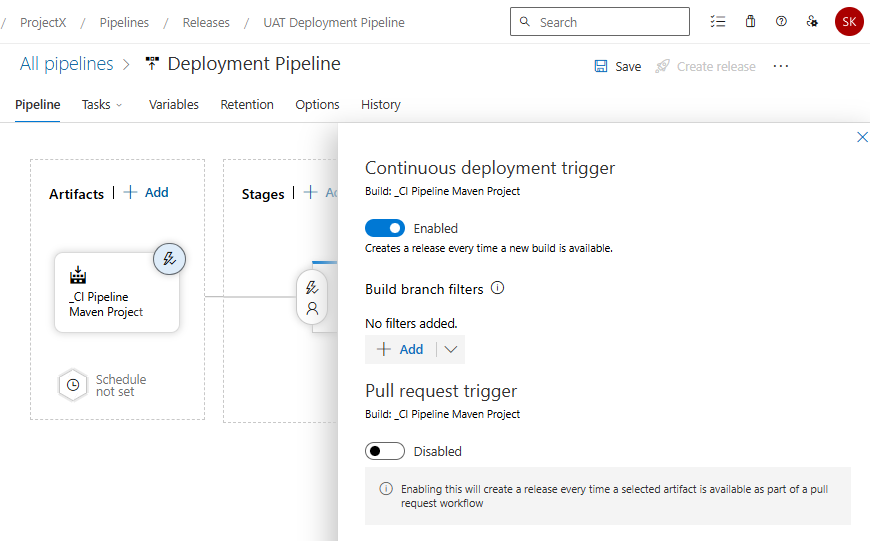


**Release Pipeline**: Go To Pipelines -> Release ->

Click on Add Artifact -> select details as shown below and click on Add

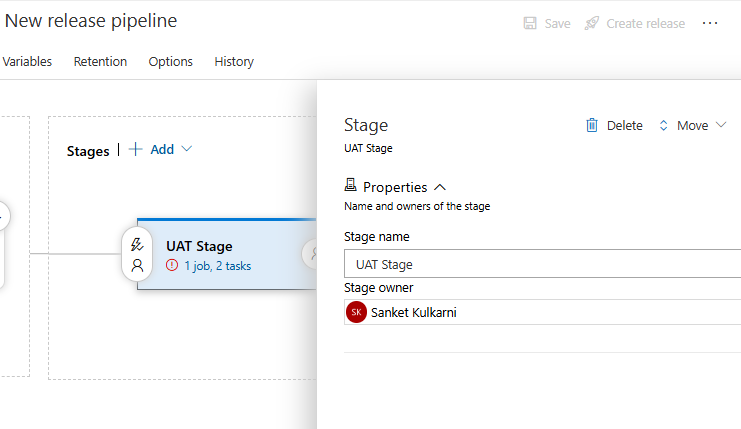
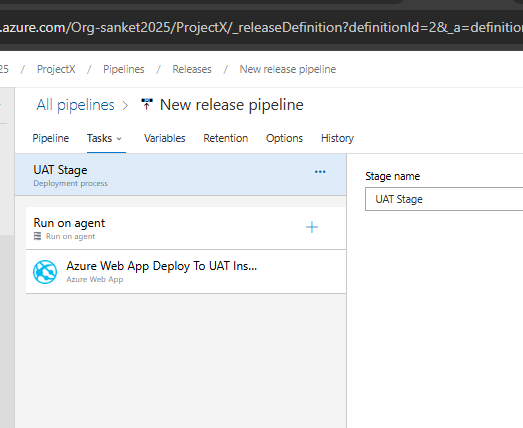


Add Trigger for Continuous Deployment

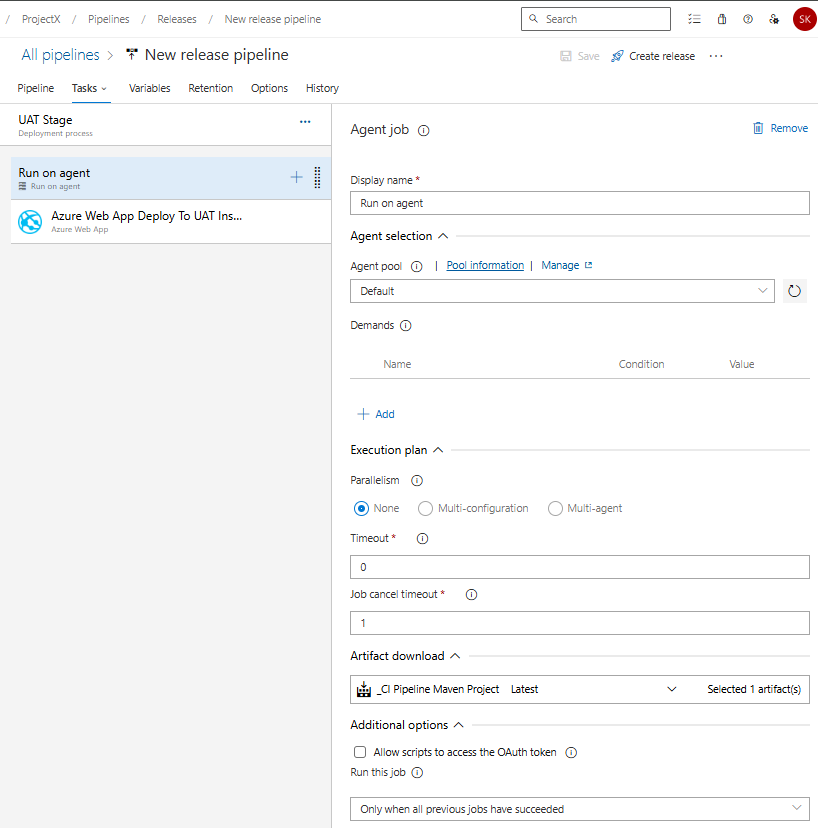


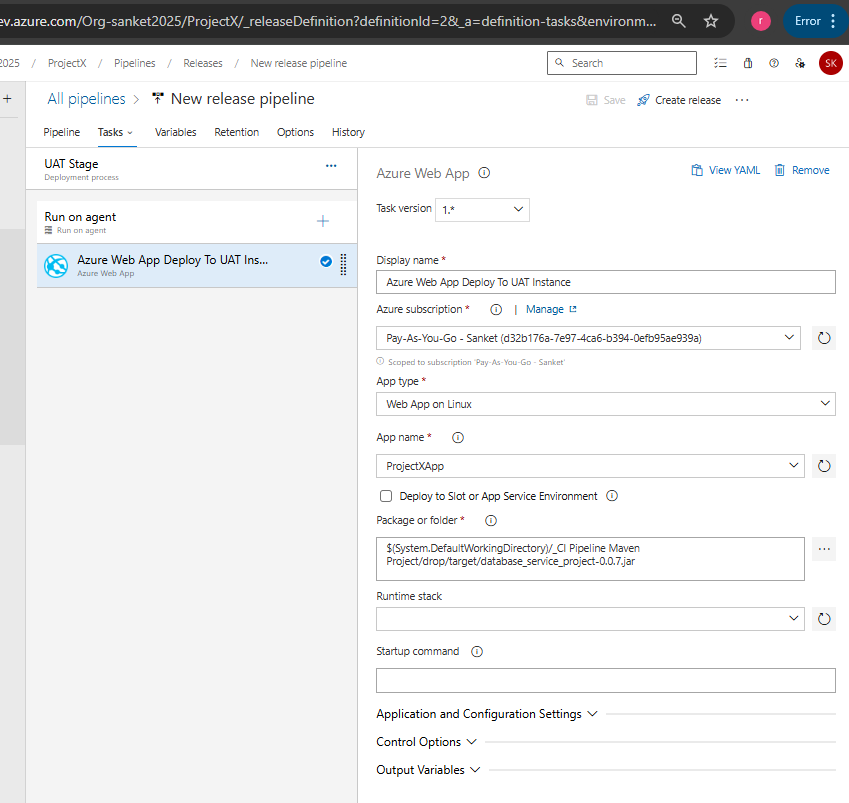
Click on New Stage -> Select Azure Web App Deployment-> Click on Apply

-> Modify Stage Name: UAT Stage

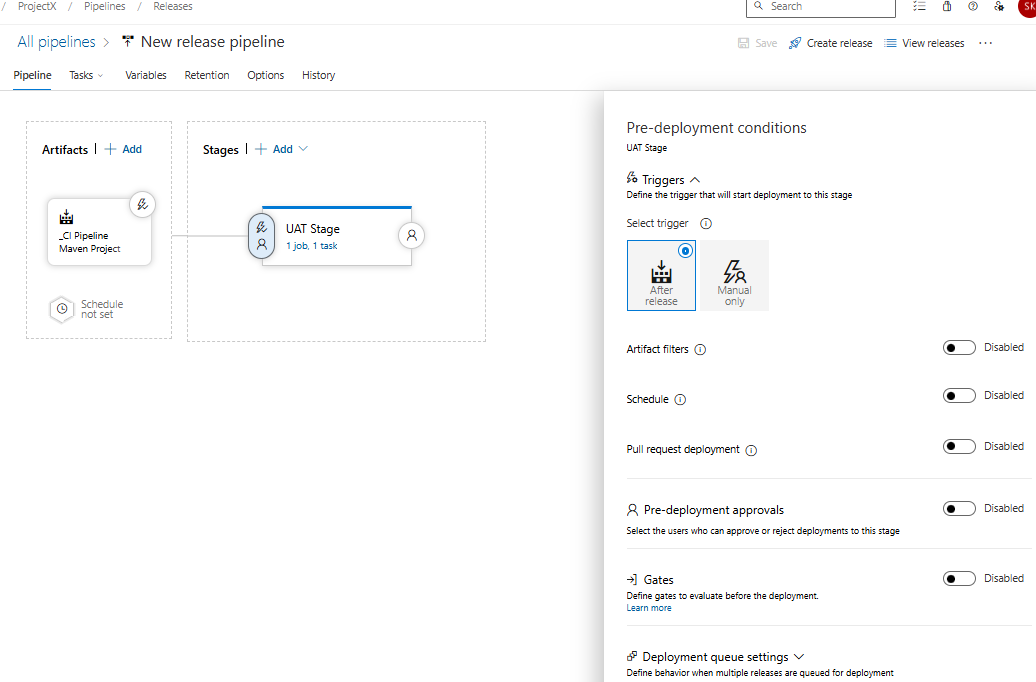
 

Open Each Task one by one and update details as required -> IF already not present, Authorize pipeline to connect to the Azure App service (It will create service principal) -> Click on Save, do the changes in next steps as per the screenshots ( Copy Slot name, app service name, resource group name from the created app service.

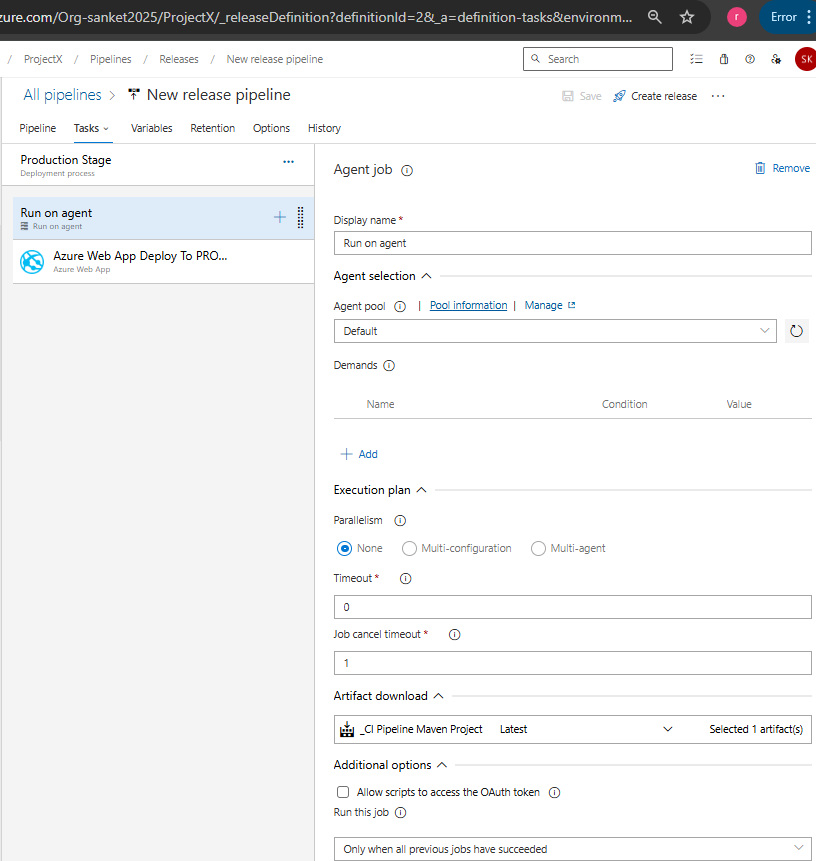


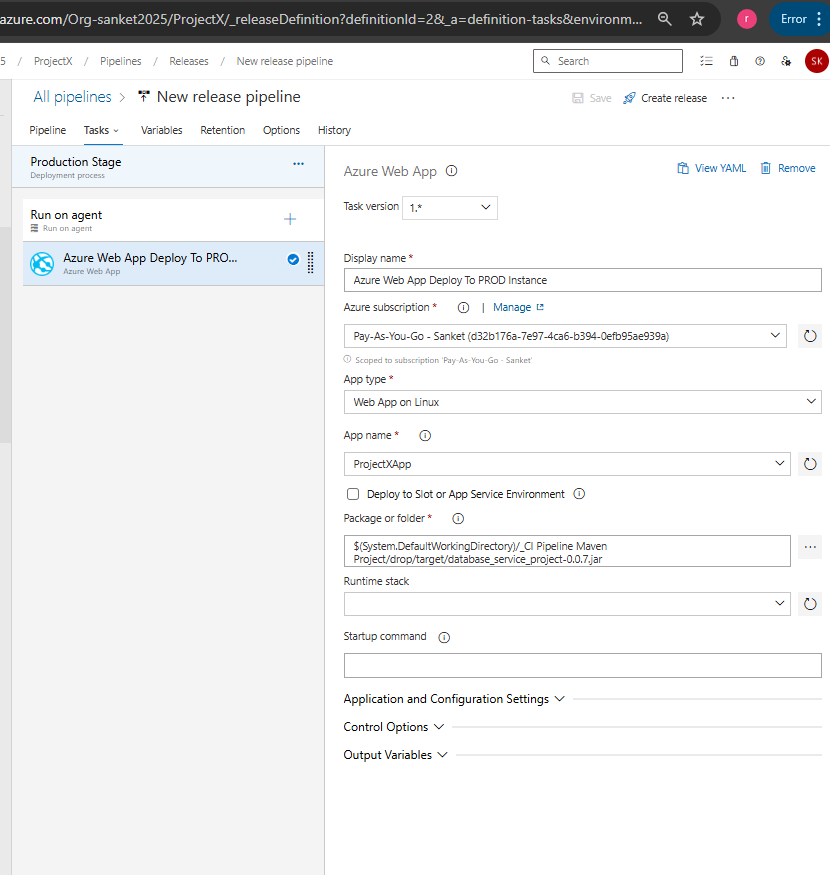


Just Create Trigger from Build Artifact generated. No pre-deployment conditions / approvals required for UAT Stage

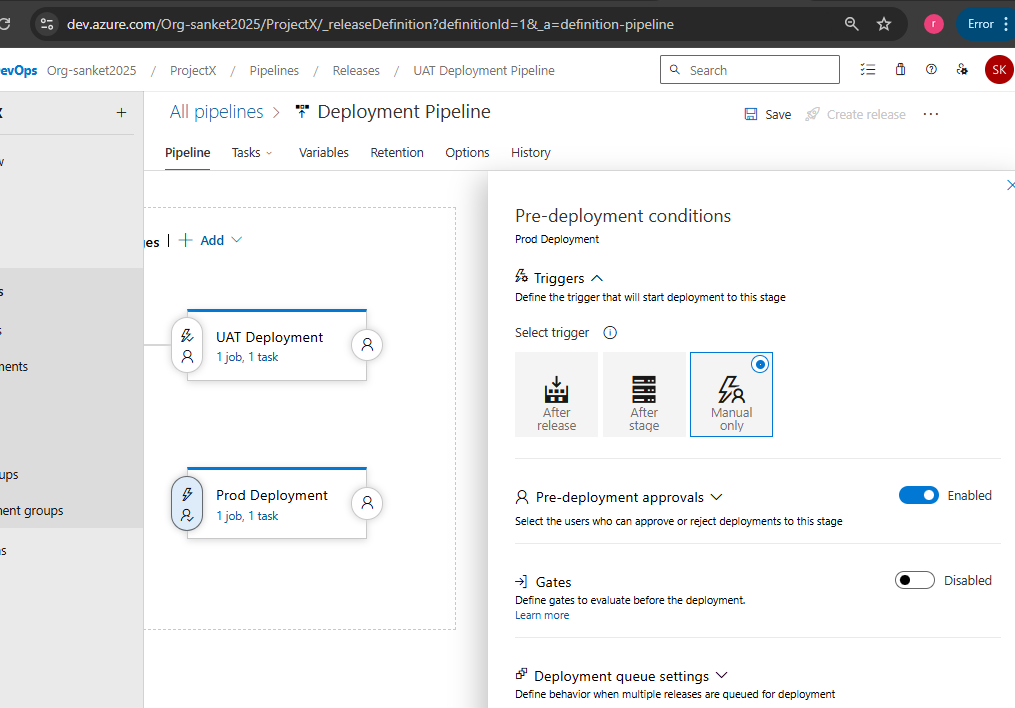


Click on Clone Stage and rename the new copy as Production Stage and Update each stage as below

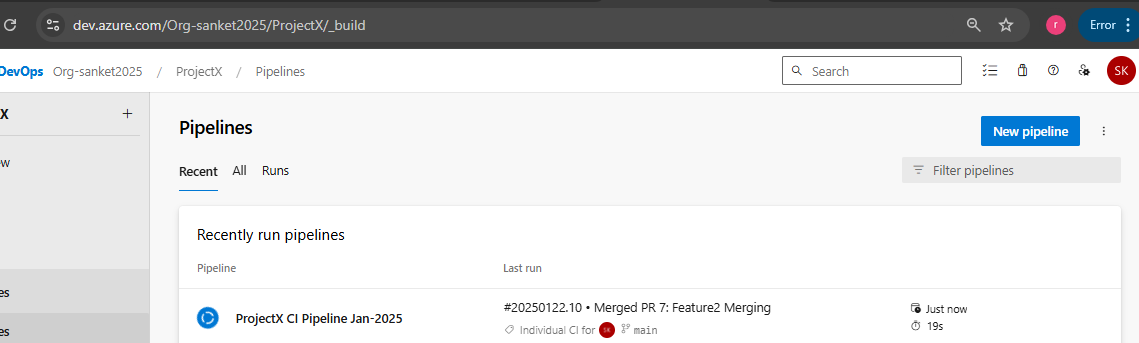




Pre-Deployment for Prod -> Allow Manual Deployment only. Also required pre-deployment approval.

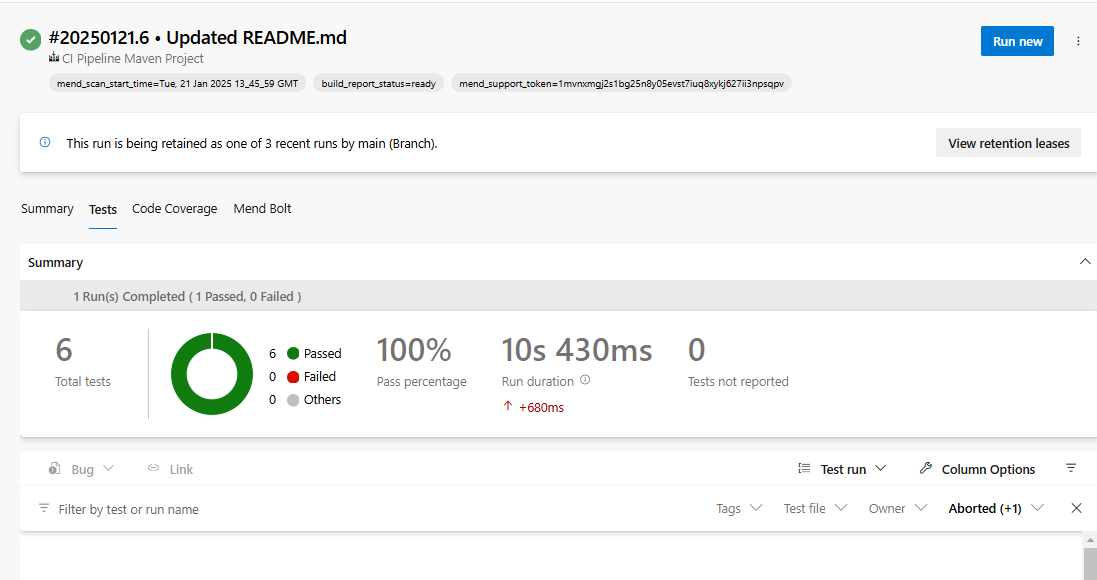


CI Pipeline triggered automatically once pull request completed on main branch (pipeline was renamed earlier to debug some issues)

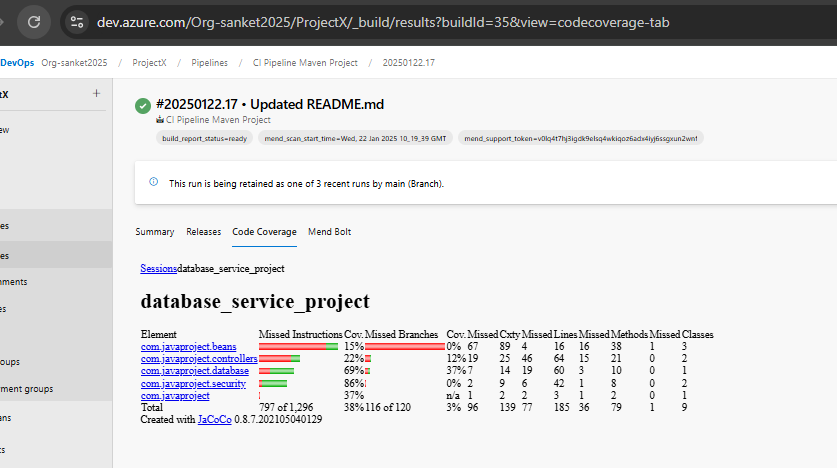


**CI Pipeline Tasks executed** successfully and below reports are generated

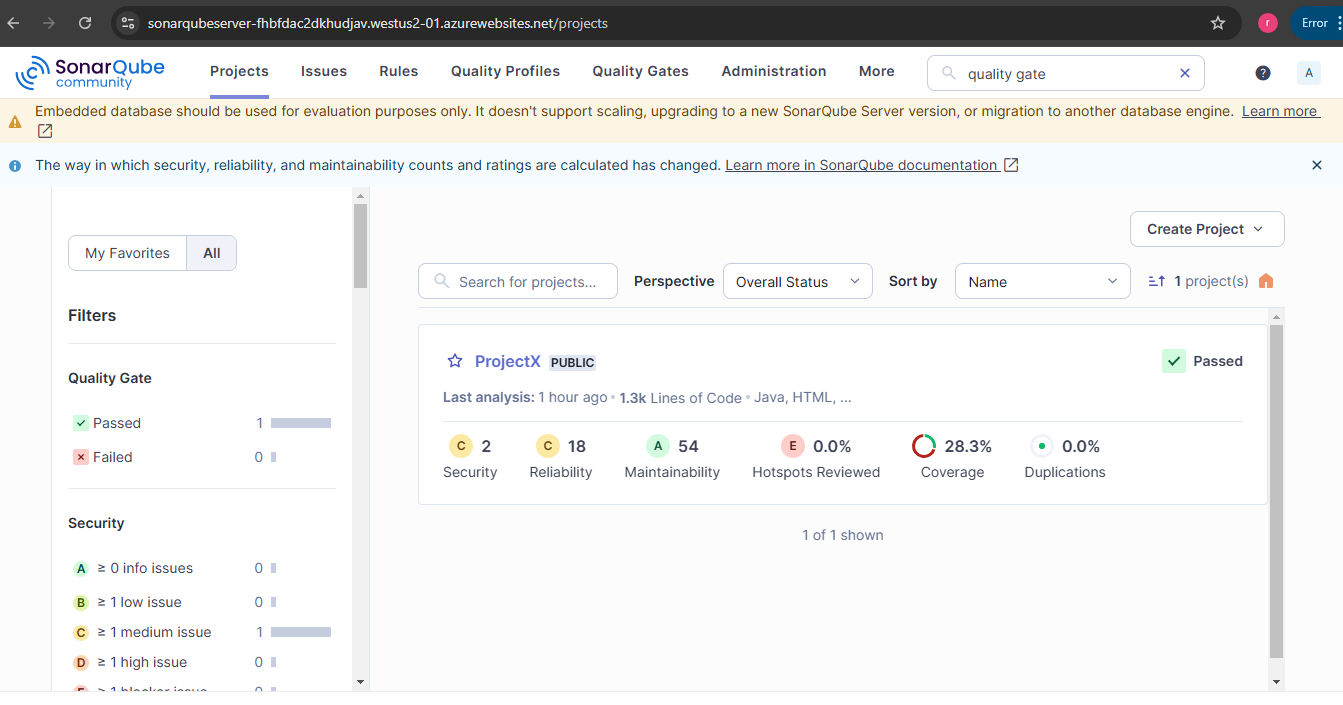
Unit Test case Report



Code Coverage

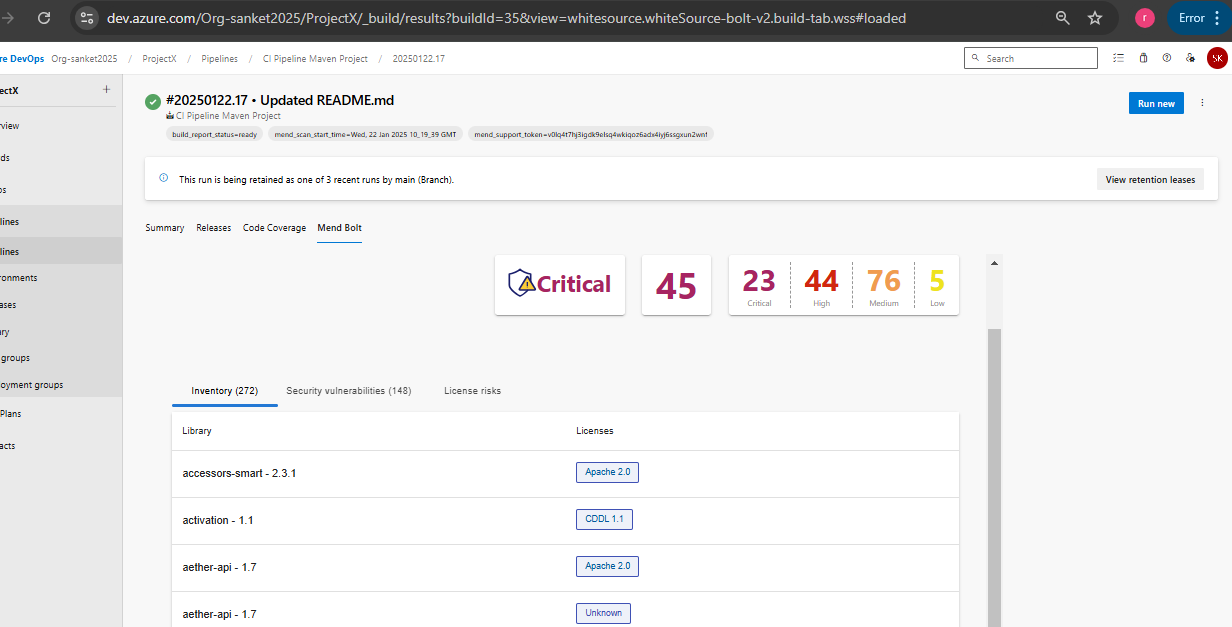


SonarQube Server portal – Output

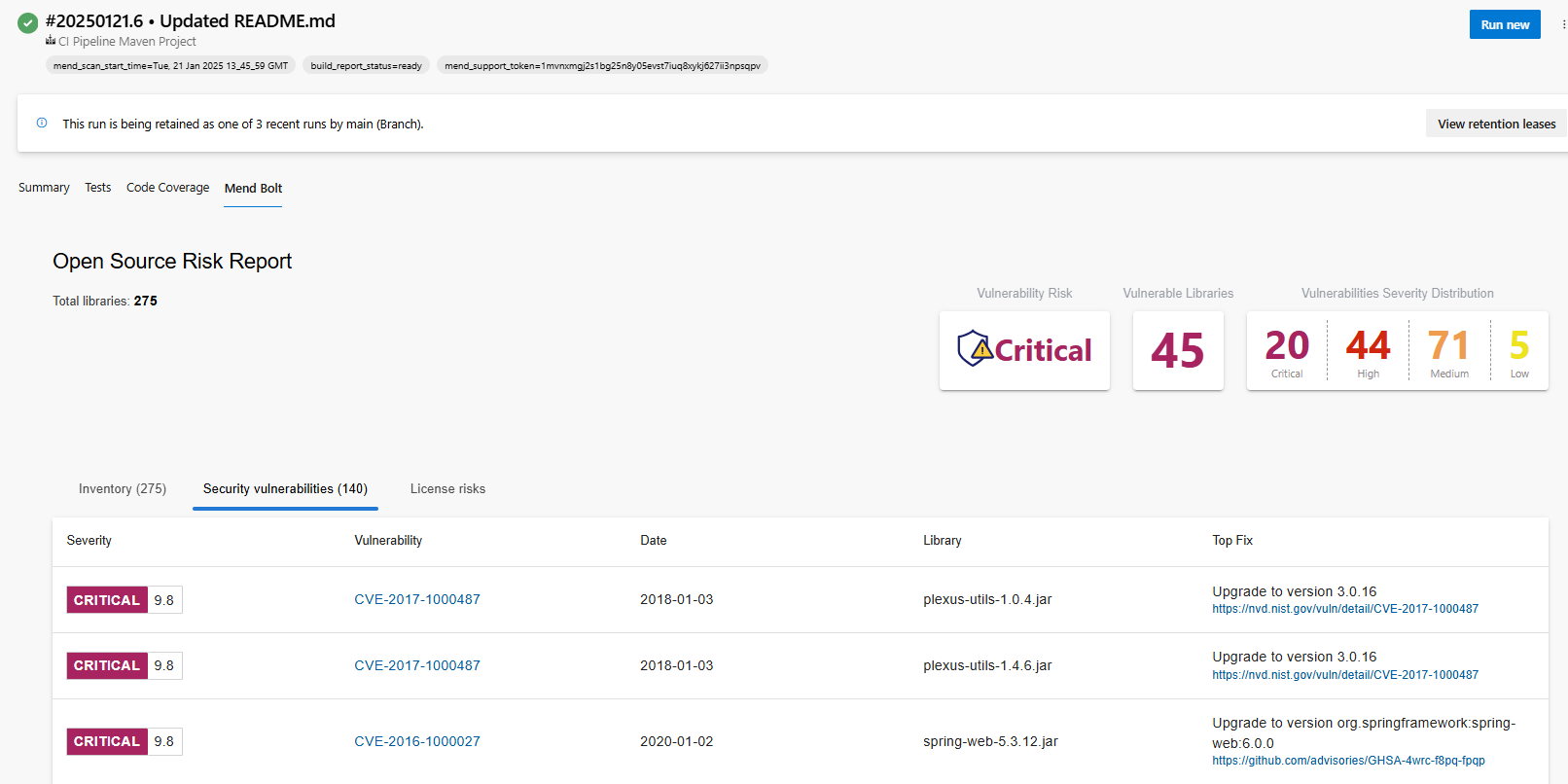


Mend Bolt Output – Go to Mend Tab in executed job instance

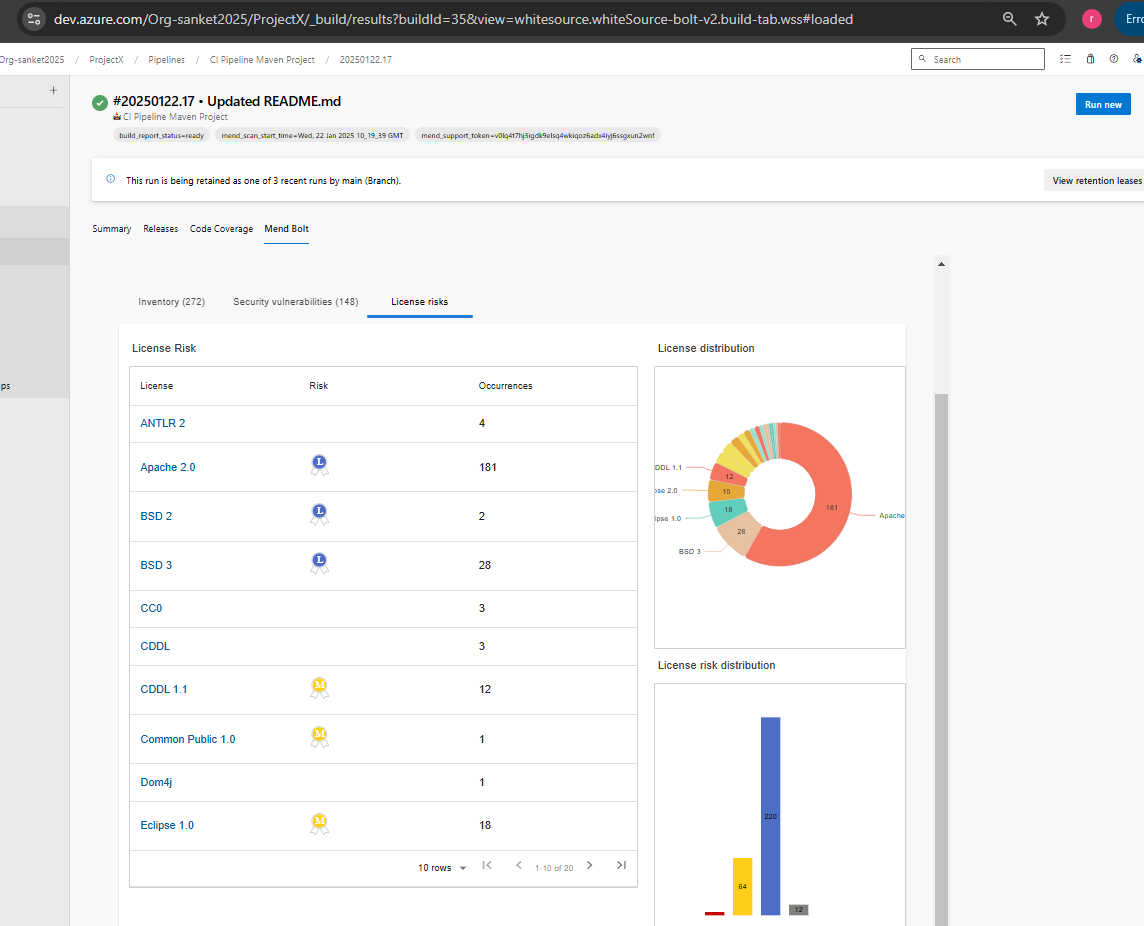
Mend Bold Inventory



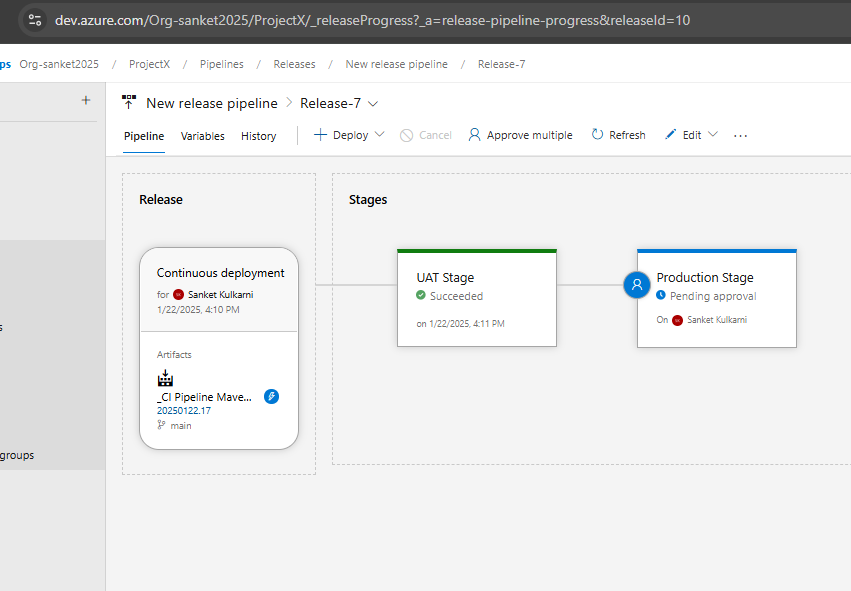
Mend Bold Security Vulnerability



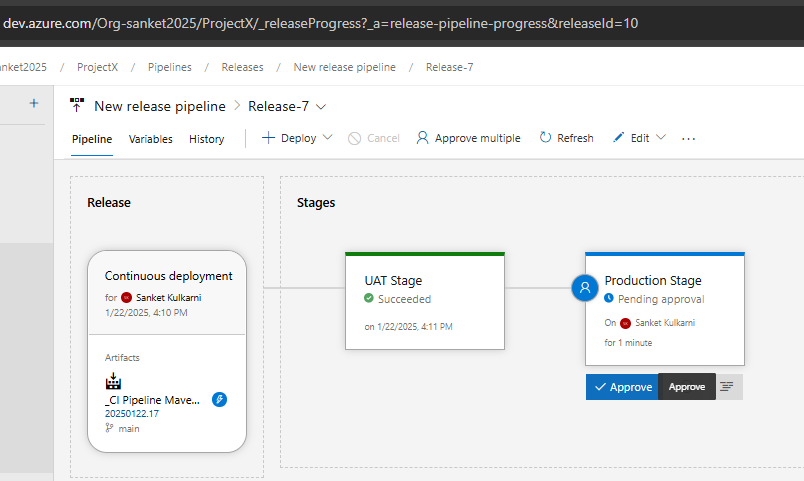
Mend Bold License Risk

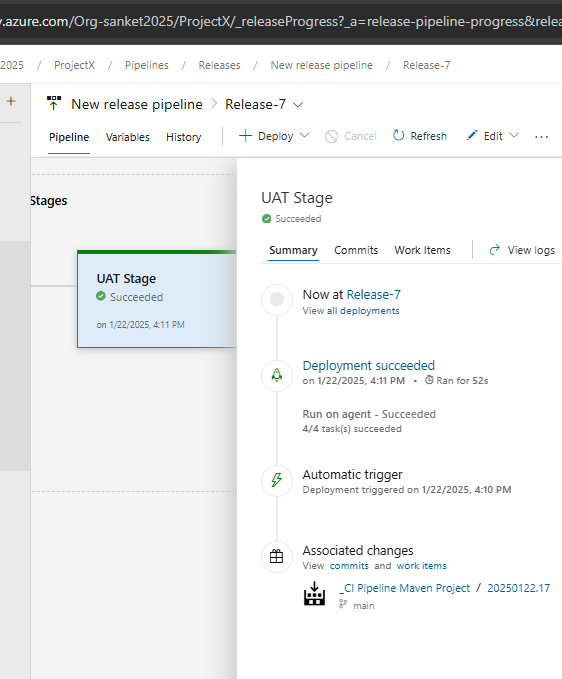


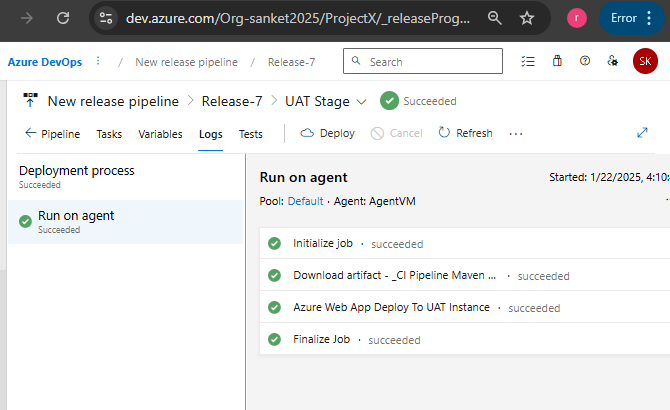
**Release Pipeline** - UAT Stage triggered automatically after CI pipeline completed and artifact generated

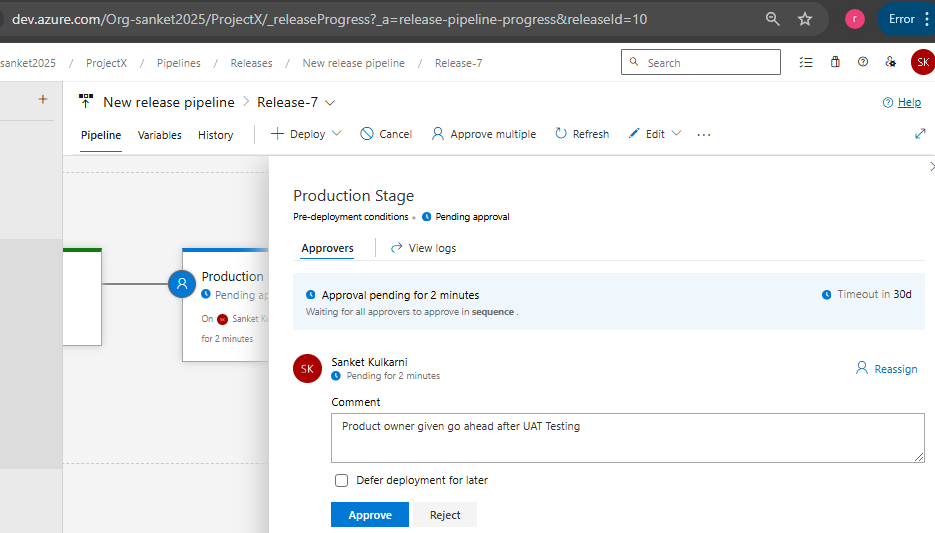


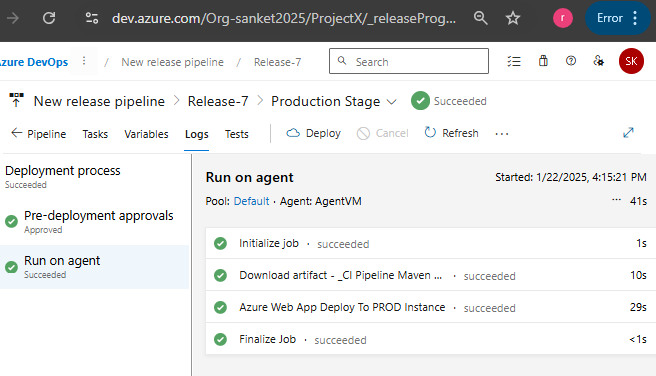
After UAT Stage successfully completed, Manual Approval required for Prod Deployment







Approval Given by Authorized person -> Release pipeline Prod Stage will start



Web App Service URL opened to check the deployments

