Azure DevOps Labs

Pre-requisites: -

1.) User should have an Azure account.
2.) User should have GitHub Account
3.) Fork the Repo for the lab: https://github.com/rajnikhattarrsinha/java-tomcat-maven-example
4.) The location of name of the war file in pom.xml present in root directory of Git Repo to be used in lab below:

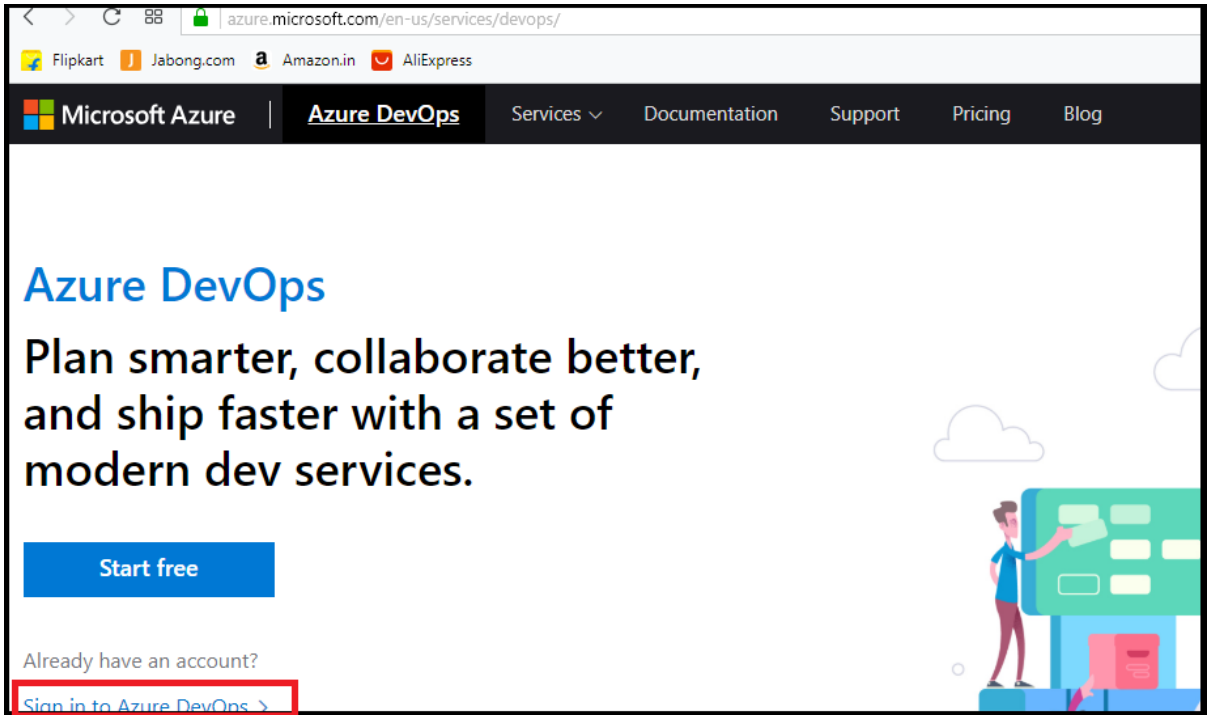
```
</dependencies>
<build>
  <finalName>demopipeline2503</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <version>2.3</version>
      <executions>
```

5.) Tomcat Server IP to be taken from Trainer
6.) Private Key for Tomcat Server is available with Trainer

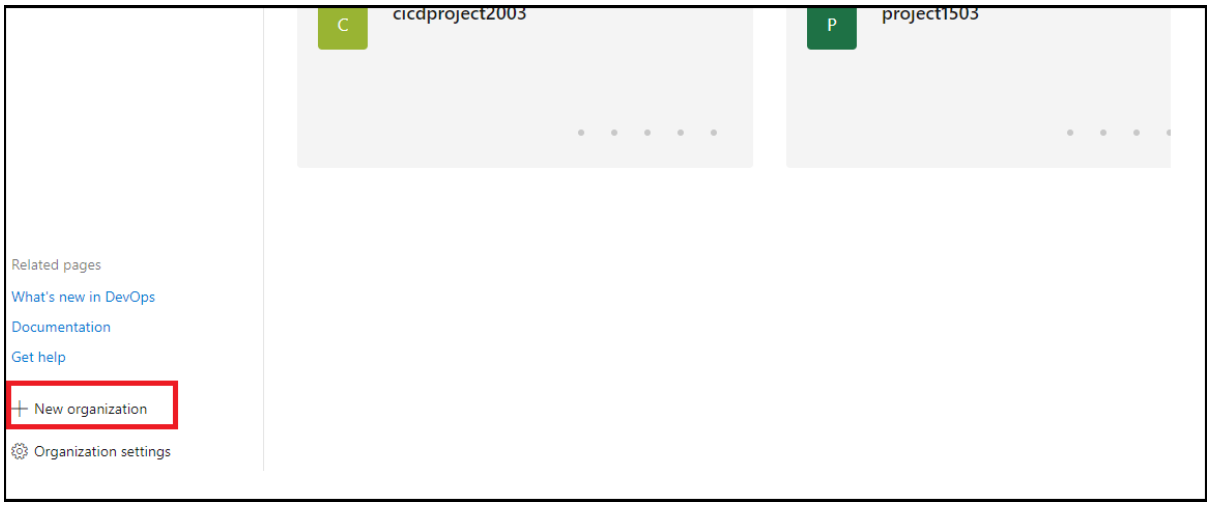
Lab 1:- Manual CICD Pipeline

Steps:-

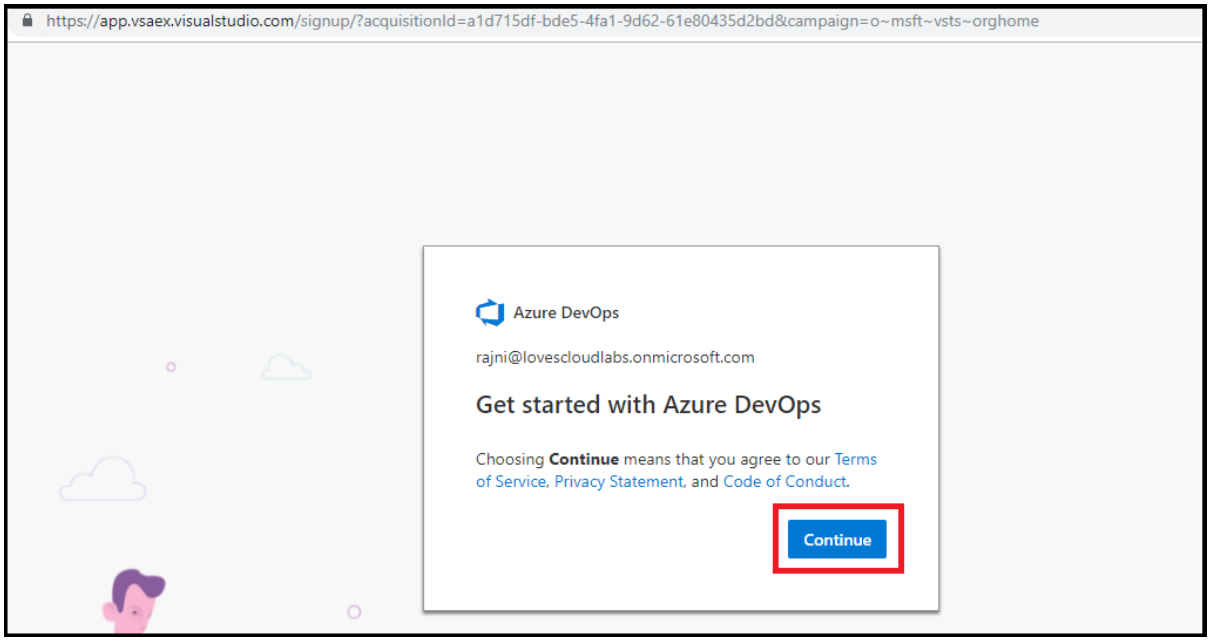
1.) Open the URL https://azure.microsoft.com/en-in/services/devops/ and Click on the link **Sign in to Azure DevOps>**



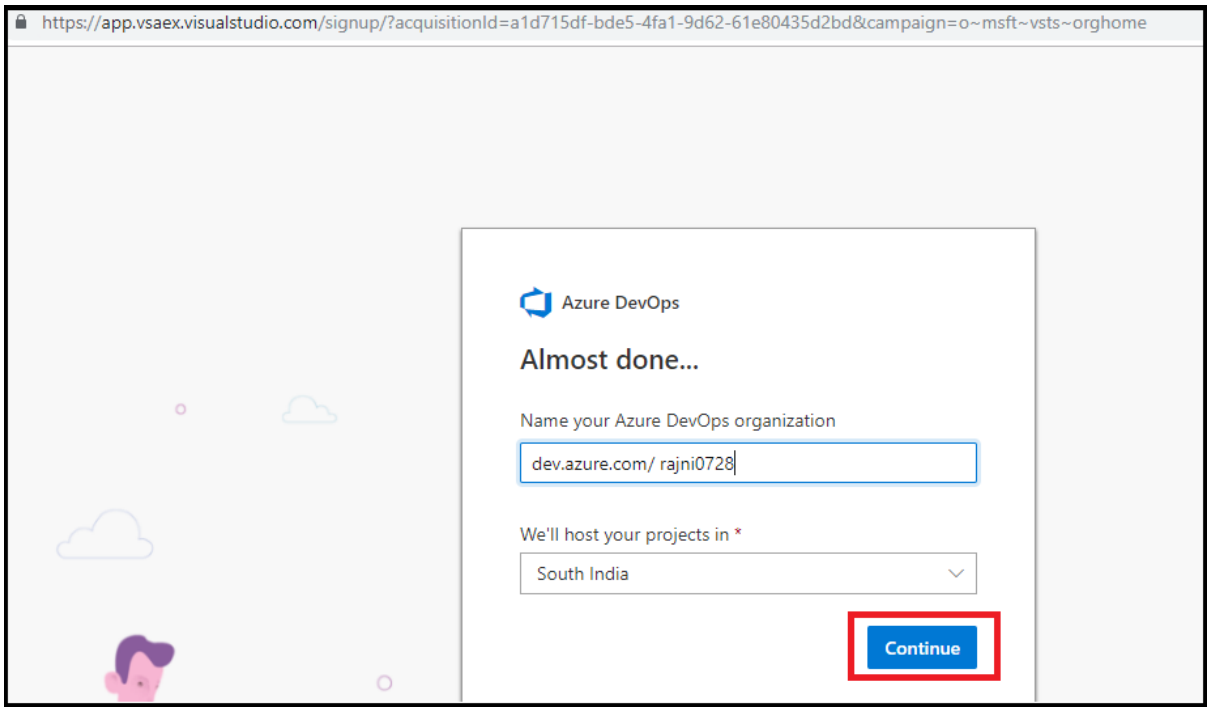
2.) Click on **New organization**



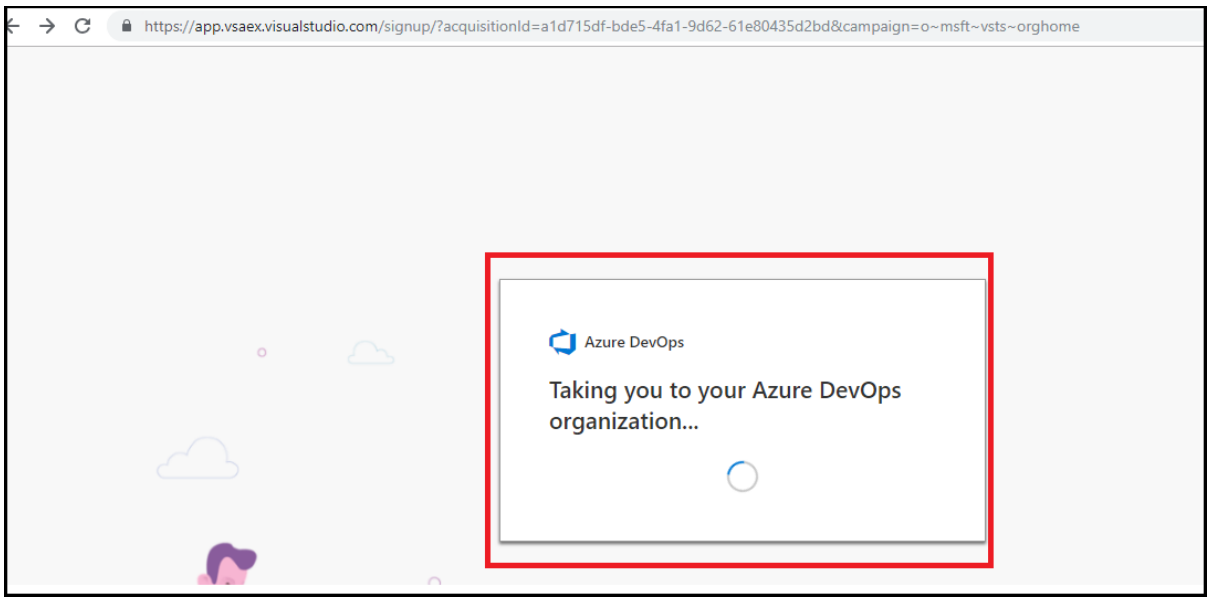
3.) Click on **Continue** on the pop up opened.



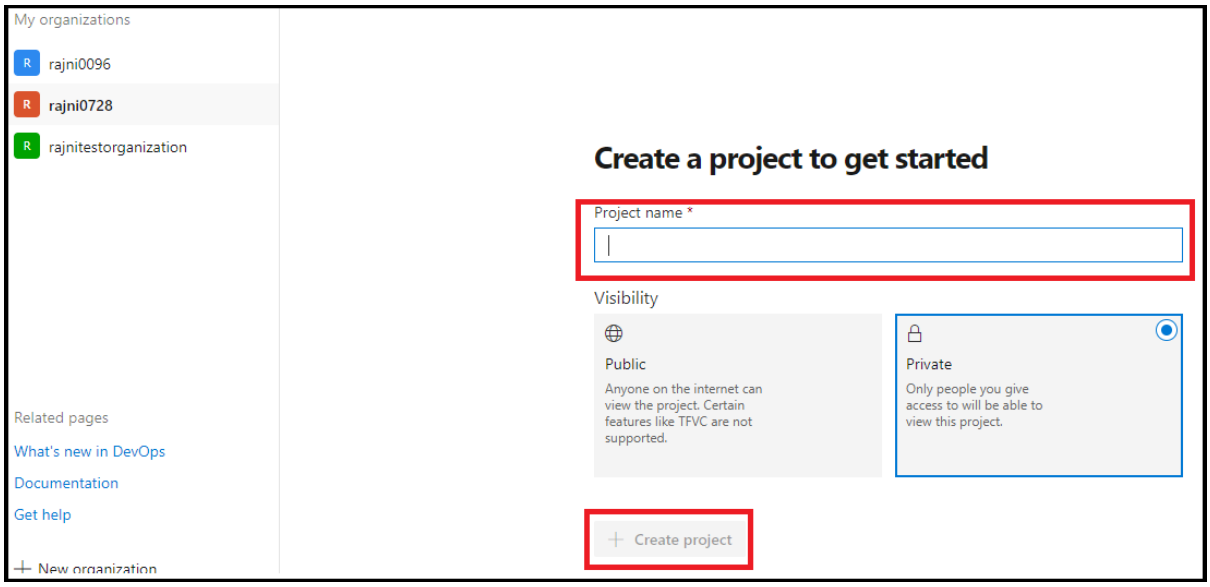
4.) Click on **Continue**



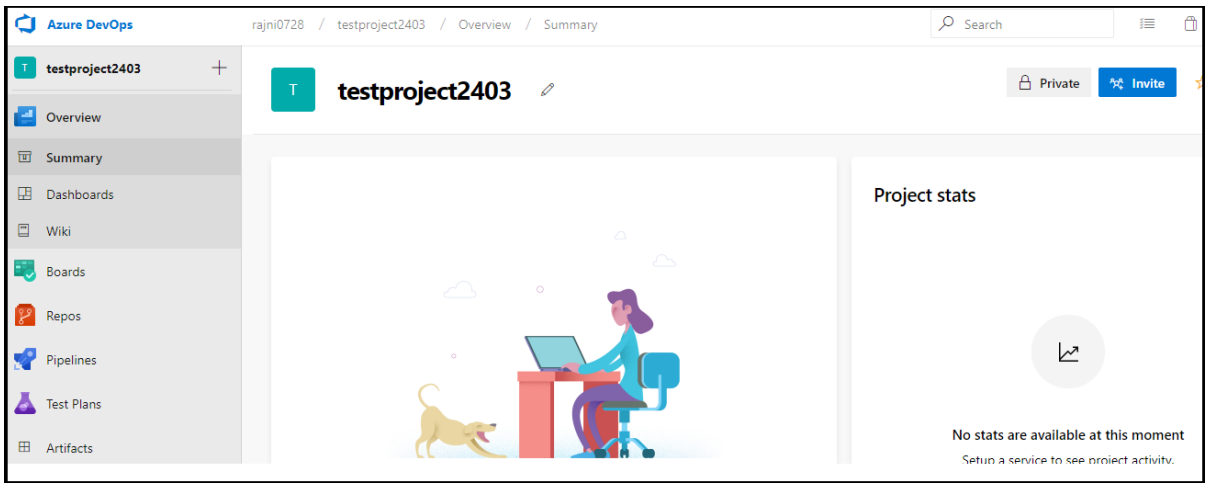
5.) The following screen should be displayed



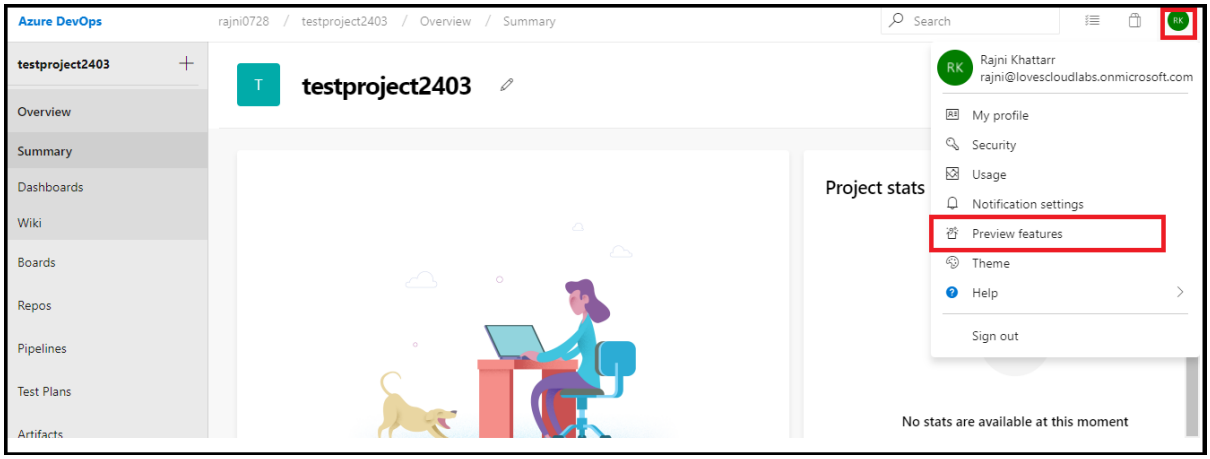
6.) Enter **Project name** and keep **Visibility** as Private selected by default. Click on **Create project.**



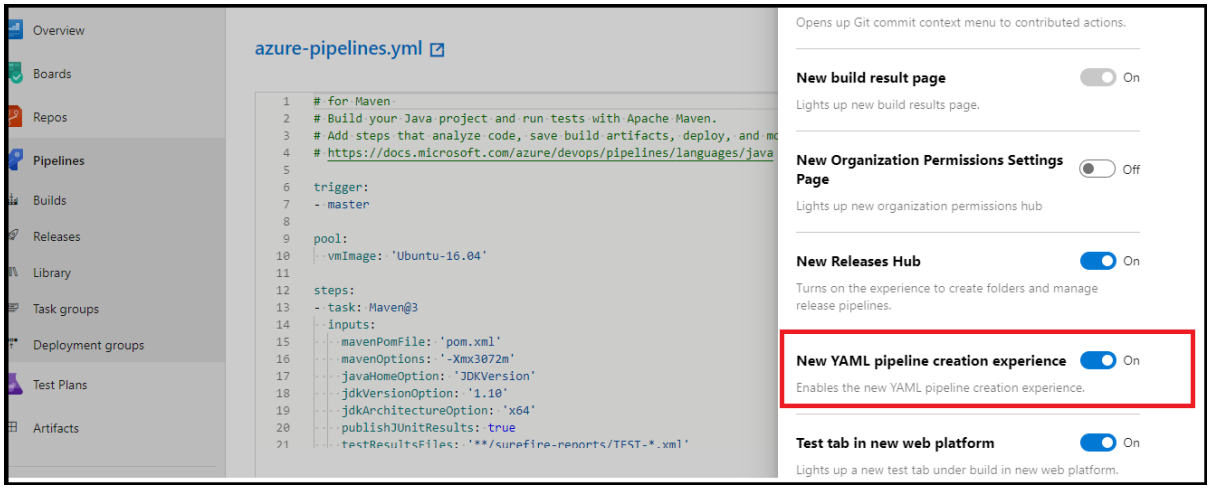
7.) Following screen should be displayed once project is created.



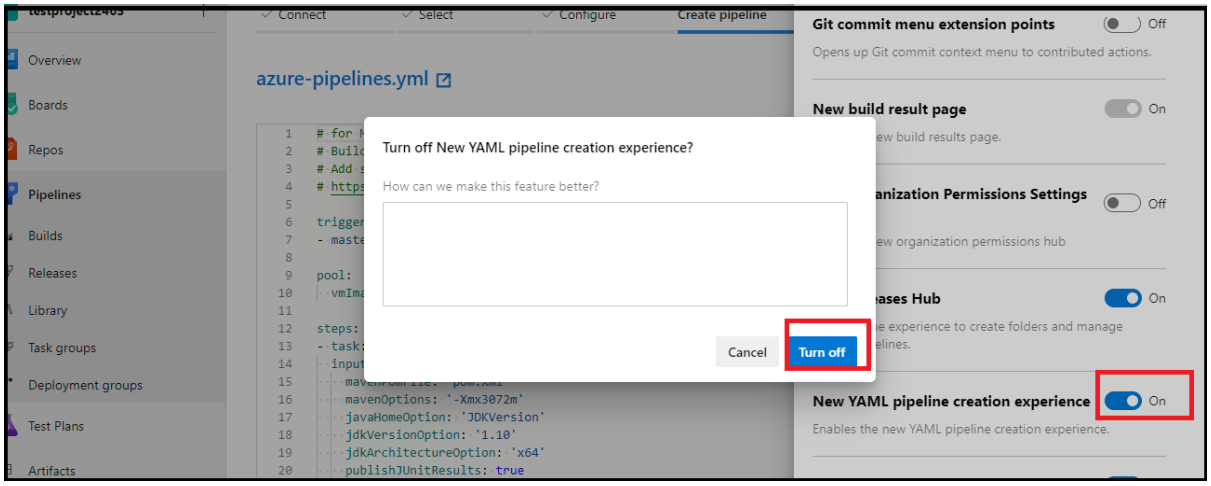
8.) Click on the Account Name initials on Right Top Corner and Click on **Preview features**



9.) Scroll down and Toggle Left the feature **New YAML pipeline creation experience**



10.)Click on **Turn Off** button

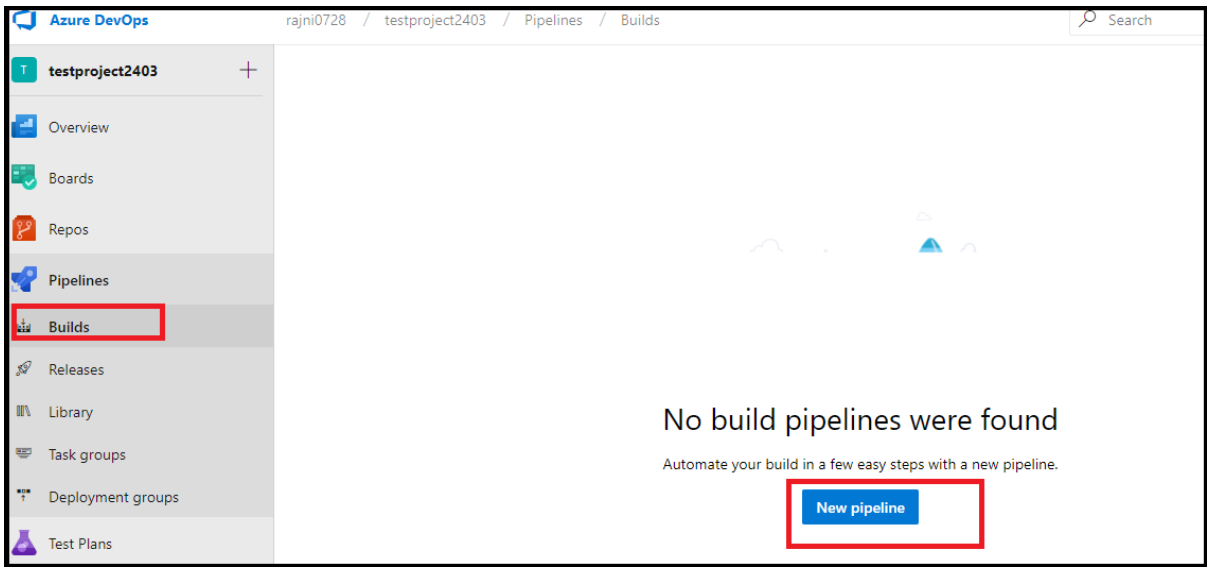


11.) Sign Out and Sign In again.

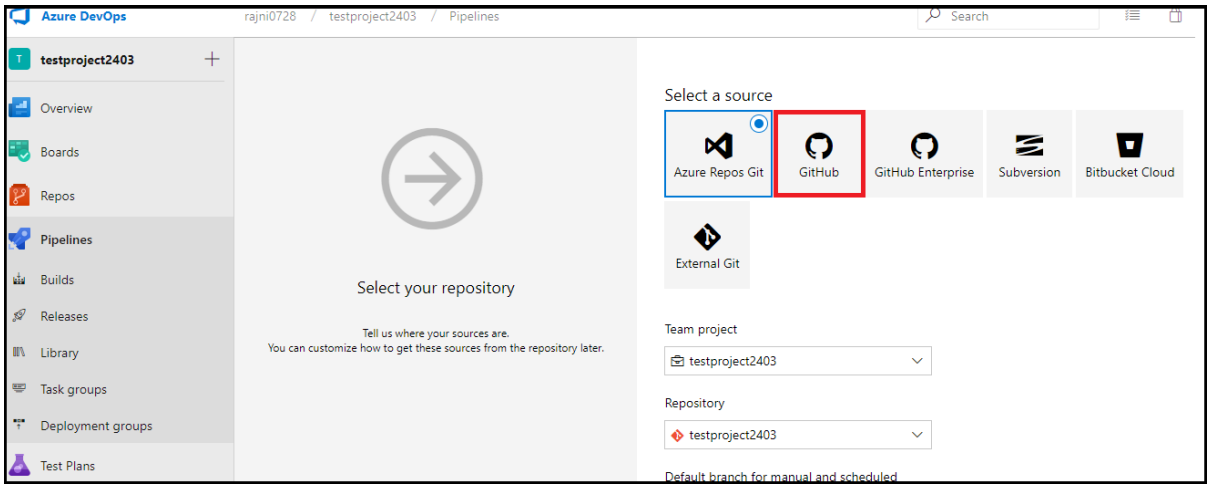
12.) Select your project

13.)Click on **Pipelines -> Builds**

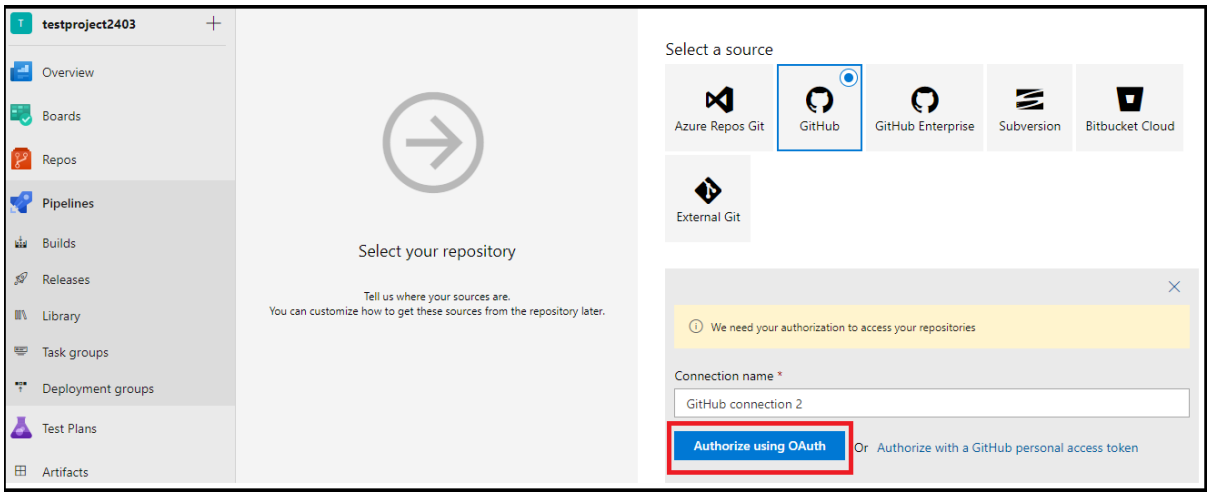
14.)Click on **New Pipeline**



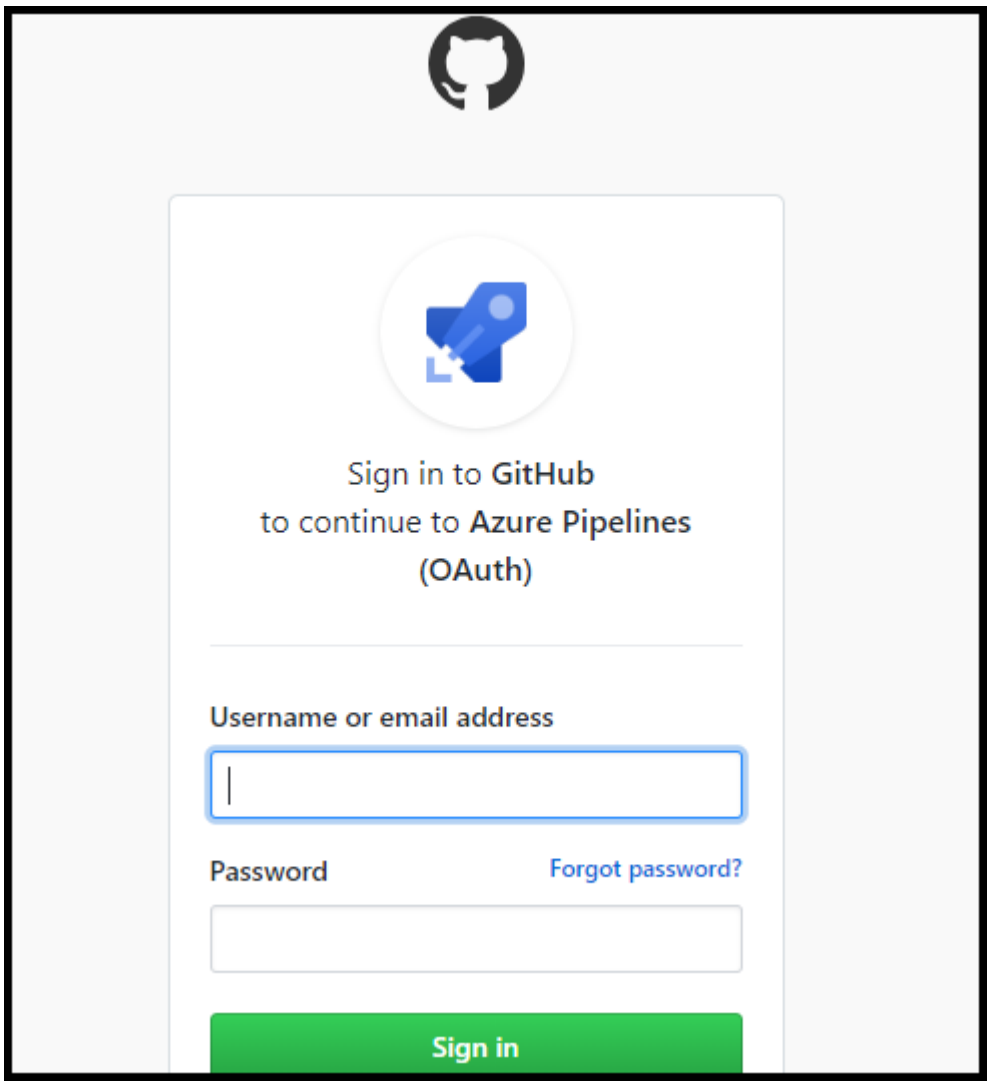
15.)Select **GitHub**



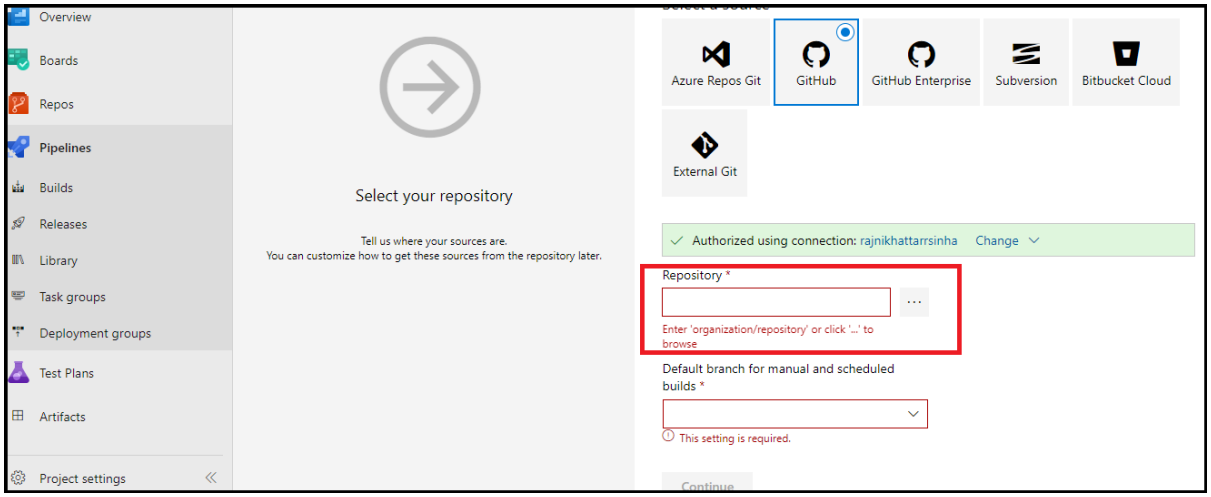
16.) User would need to create a new Service Connection for **GitHub** and Click on **Authorize using OAuth**



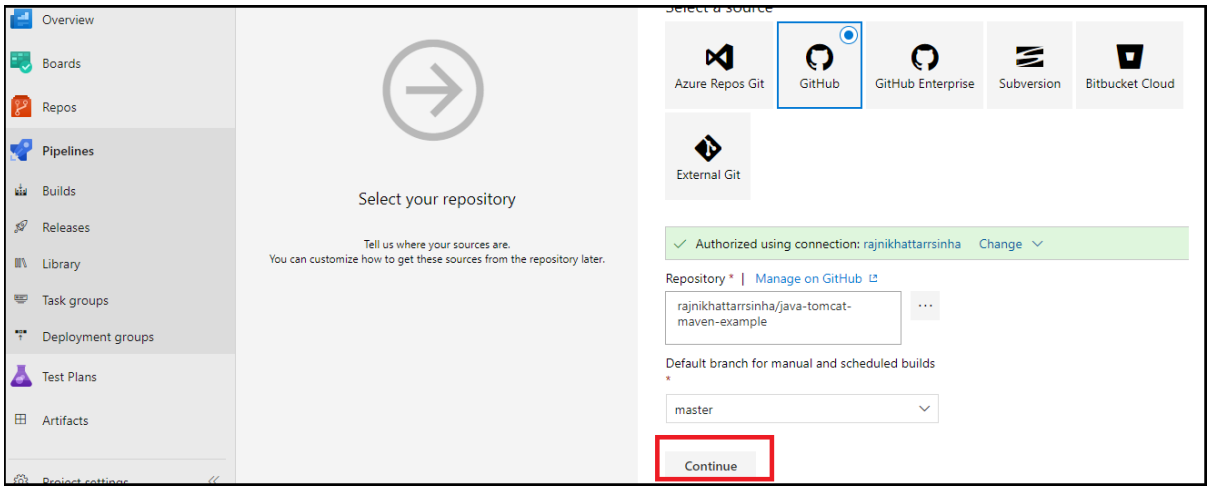
17.)Enter **Username**, **Password** and Click **Sign in.**



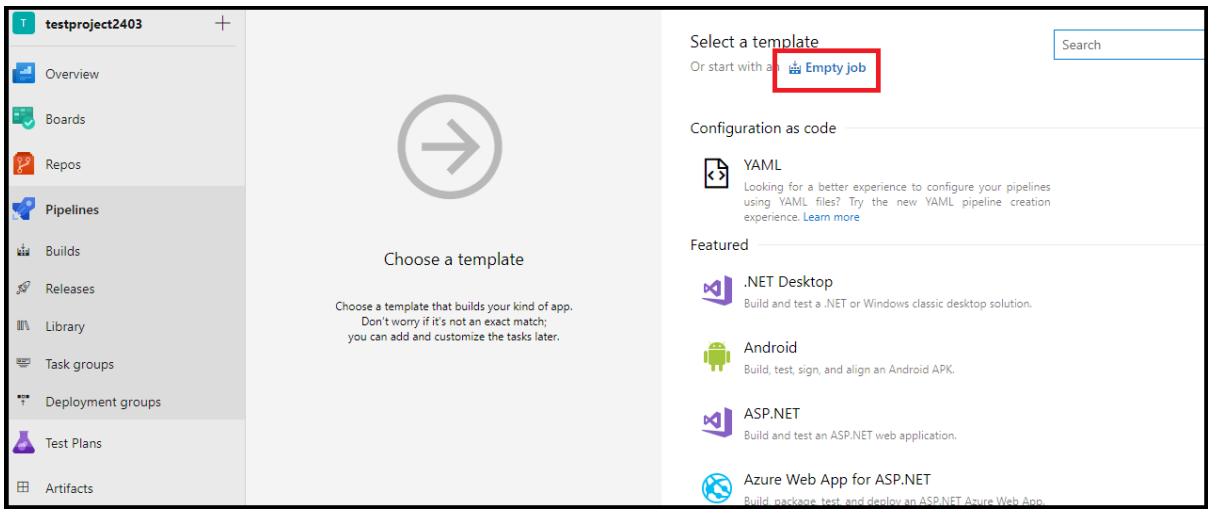
18.)Select **Repository** by clicking on … and searching your repo forked during Pre-requisites.
Select **Default Branch** as master



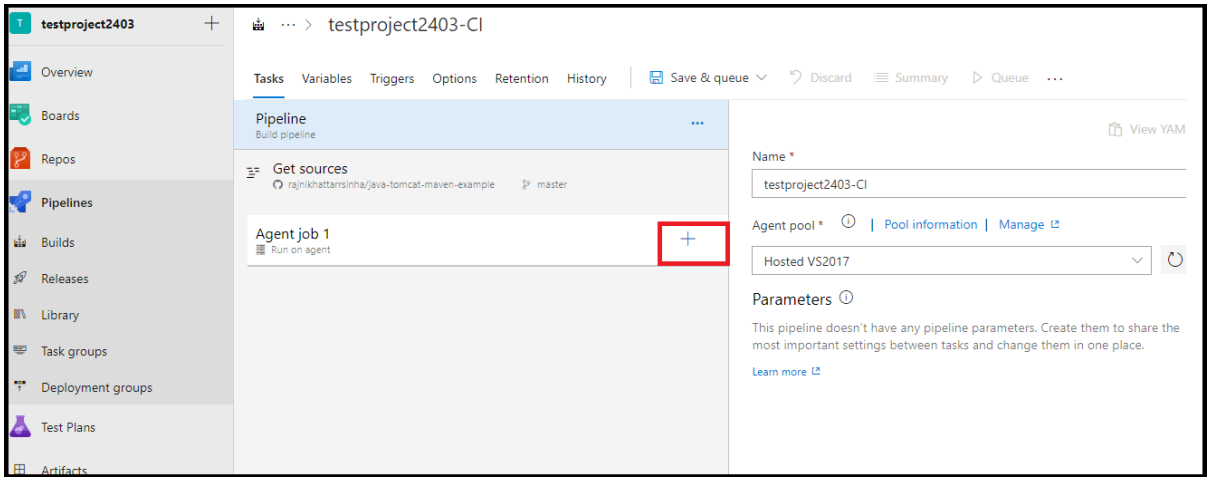
19.)Click on Continue



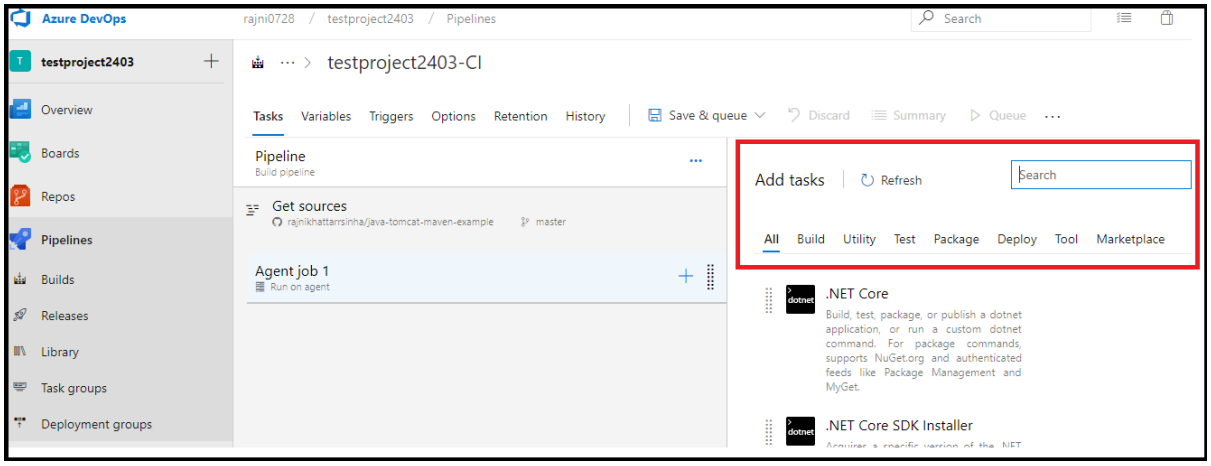
20.) Click on **Empty Job**



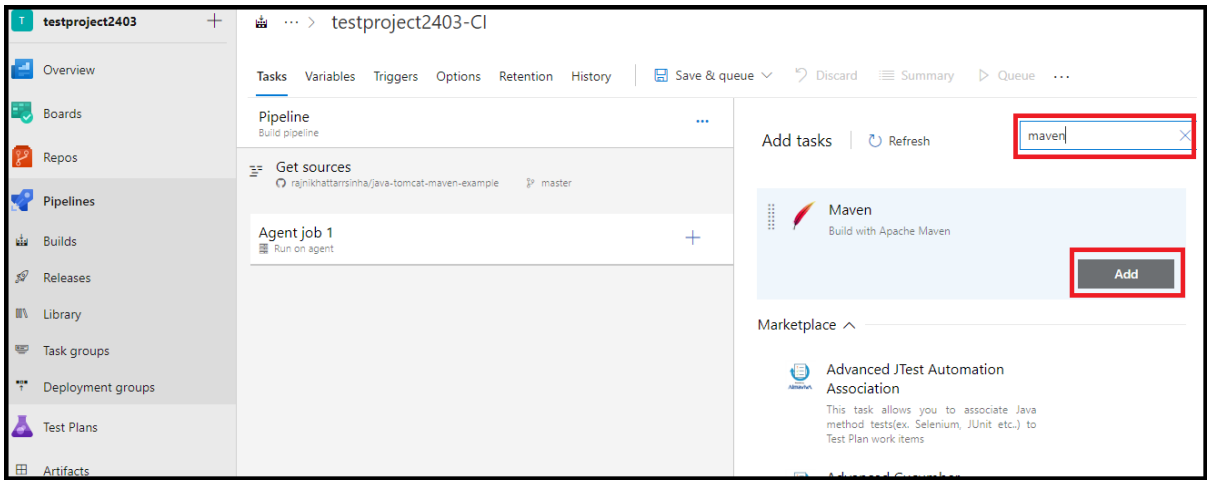
21.)Click on **+** of **Agent Job 1**



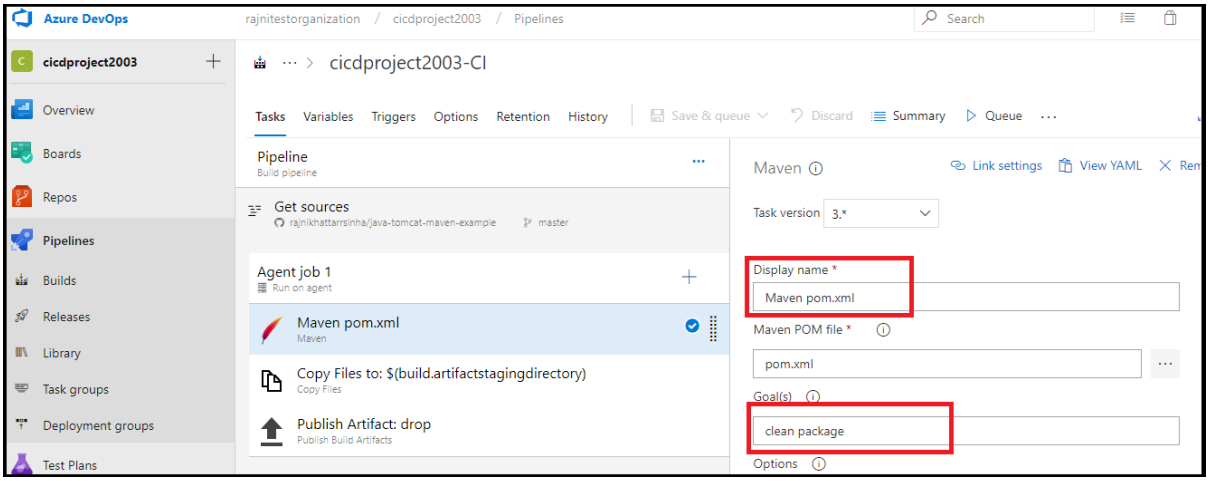
22.) The following window would open, use this search box to search and add the jobs you want to execute for CI pipeline



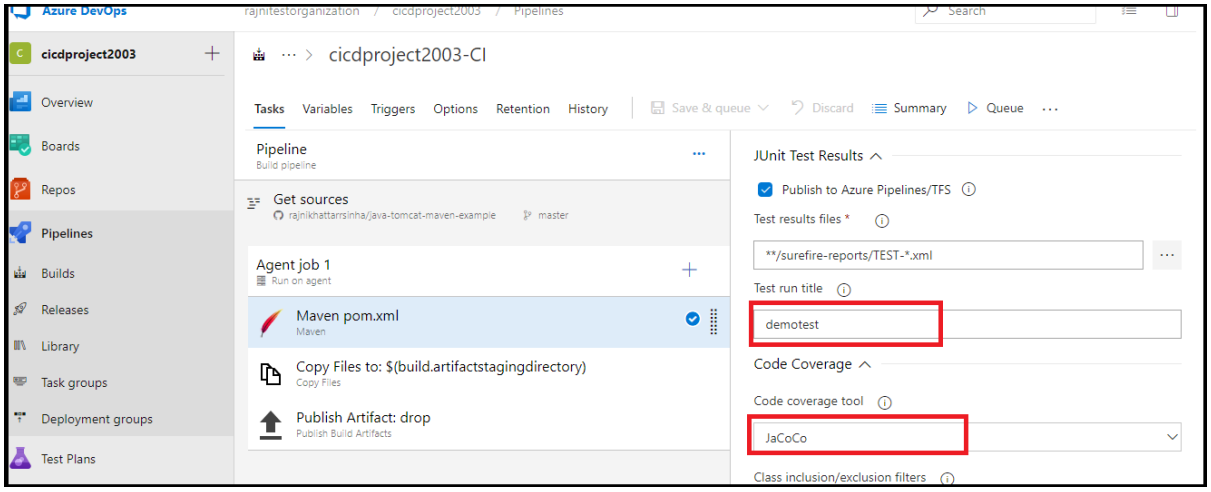
23.)Search **maven**, mouse over on Maven from Search results and click **Add**



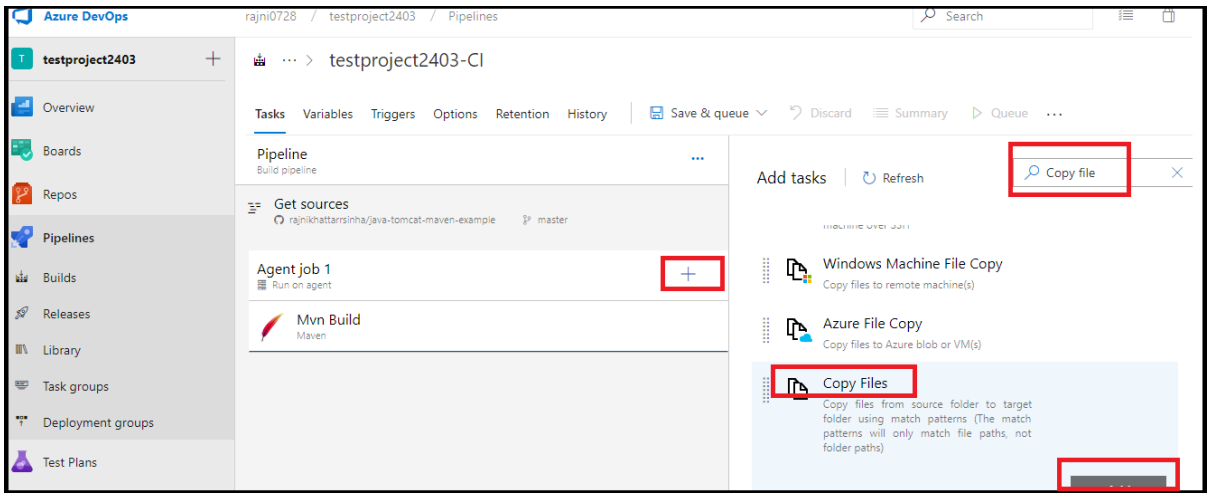
24.) Select **Maven**, Enter **Display name** as Mvn Build, Enter **Goal(s)** as clean package



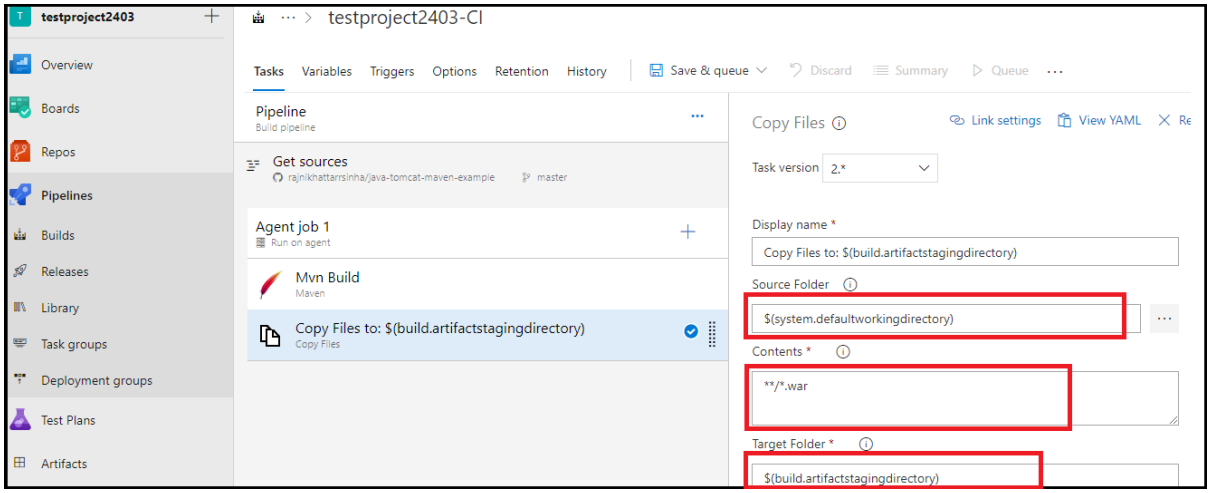
25.)Scroll down, Enter **Test run title** as demotest and Select **Code Coverage Tool** as JaCoCo



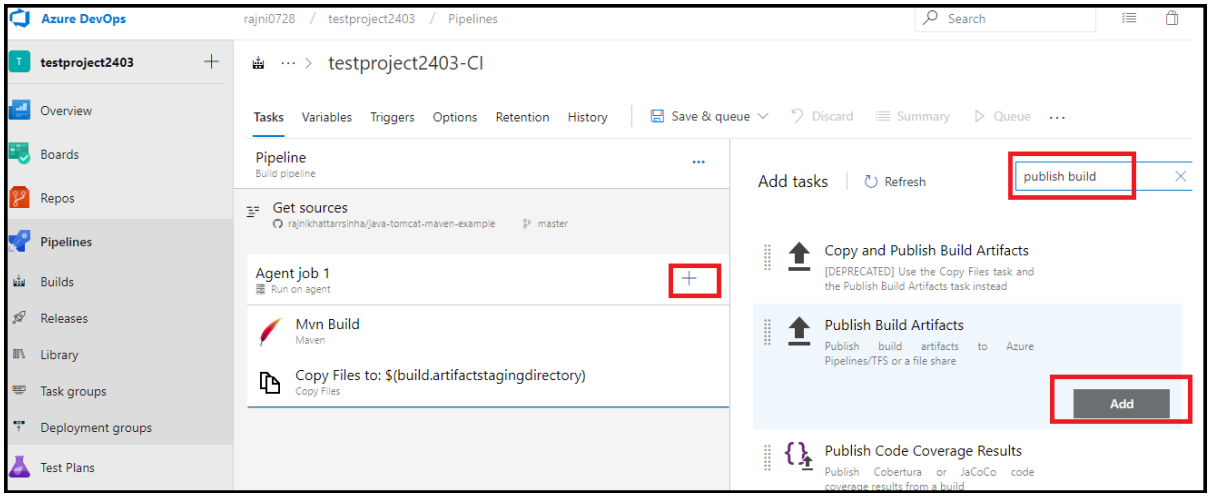
26.) Click on **"+"** again, Search **Copy file**, Mouseover on **Copy Files** and click **Add.**



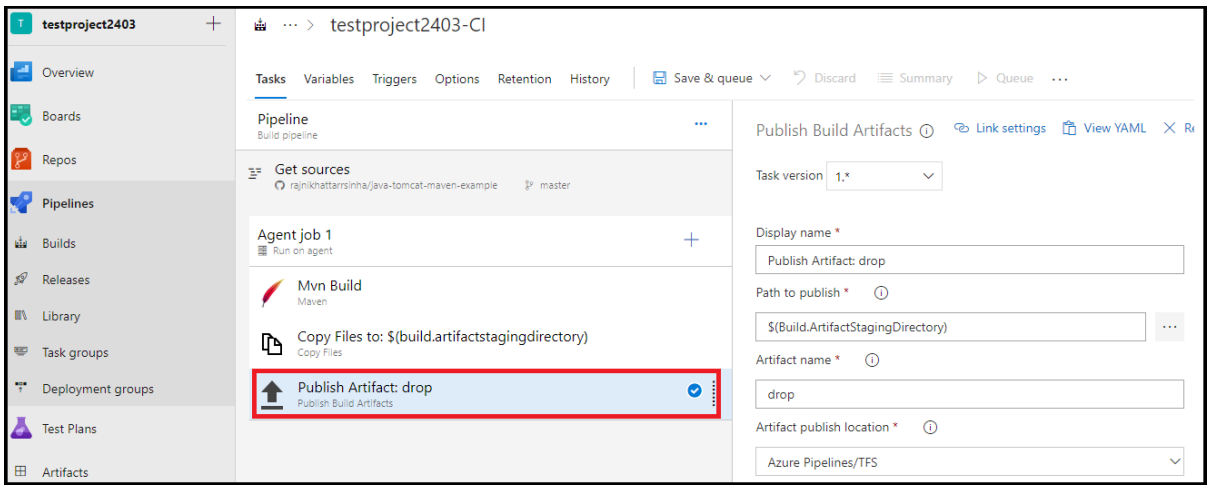
27.) Enter **Source Folder** = $(system.defaultworkingdirectory), Enter **Contents** = **/*.war and Enter **Target =** $(build.artifactstagingdirectory) .



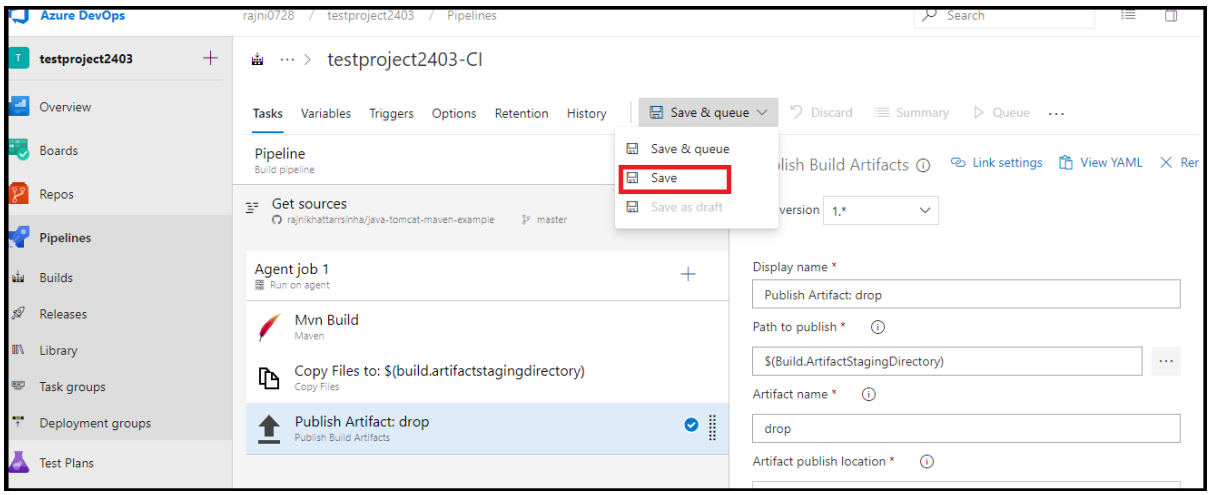
28.) Click on **"+" ,** Search **publish build** , Mouseover on **Publish Build Artifacts** and Click **Add** button .



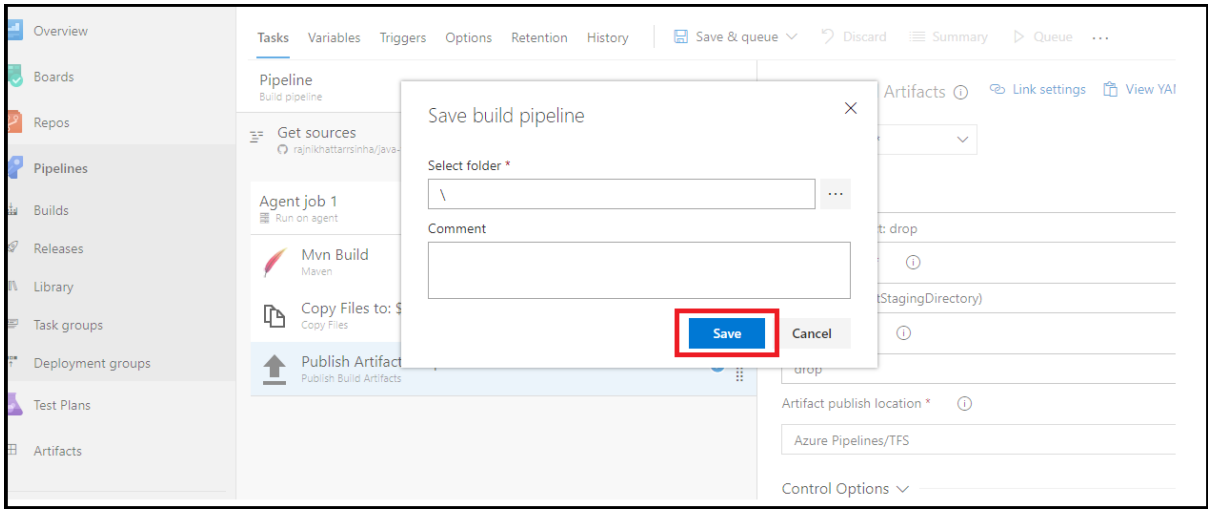
29.) Select **Publish Artifact: drop ,** Keep all default values on Right Panel as displayed.



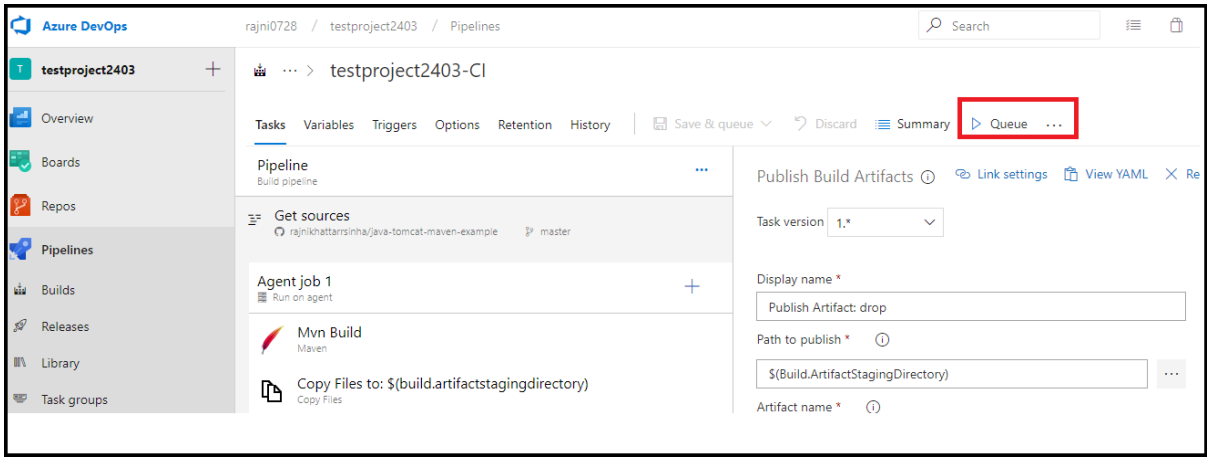
30.) Click on **Save & queue** drop down and Select **Save**



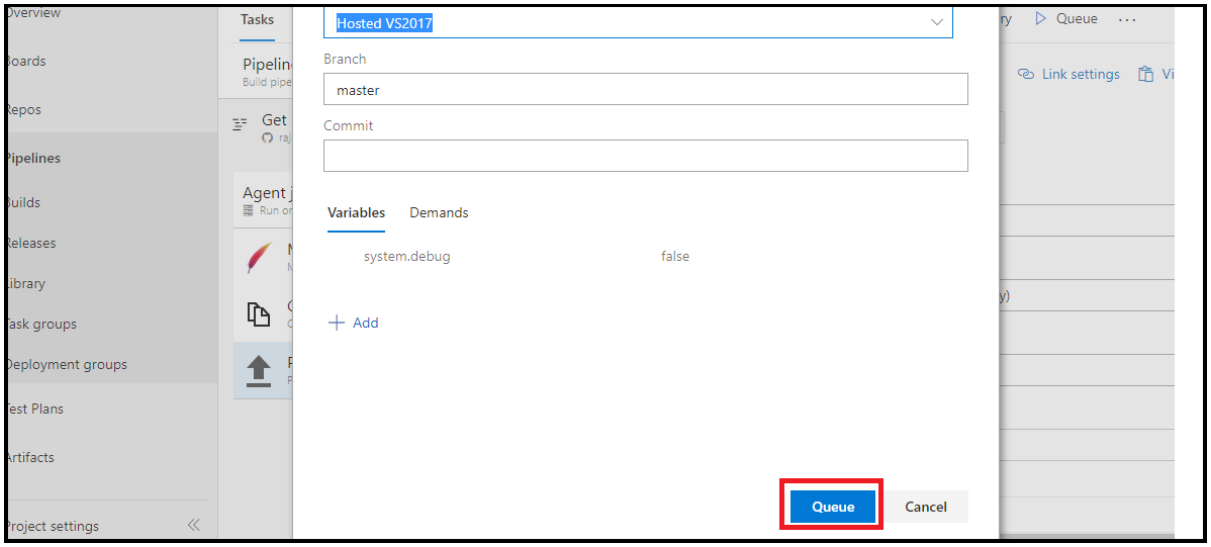
31.) Click on **Save** on Save build pipeline pop up



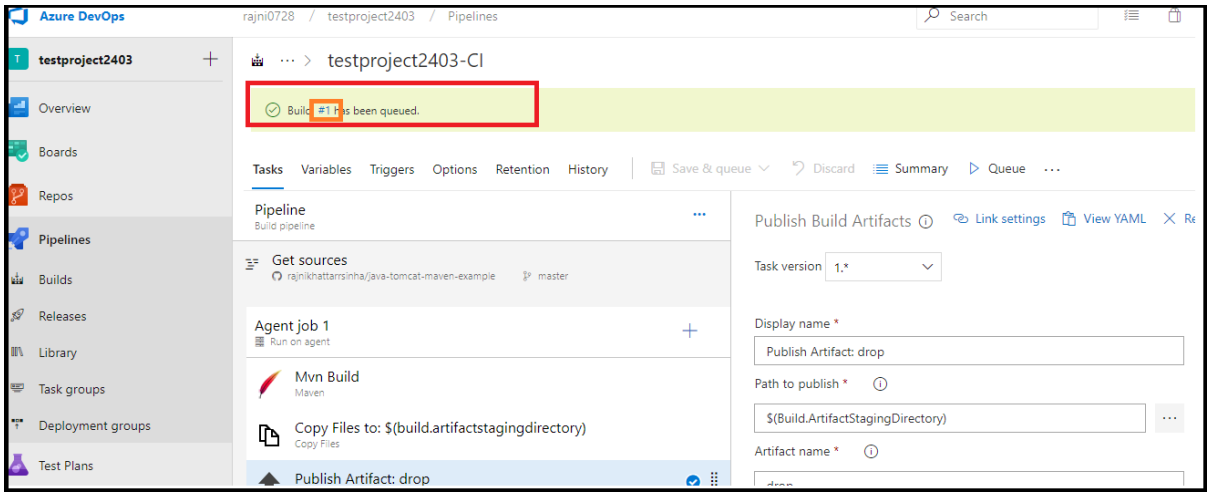
32.) Click on **Queue**



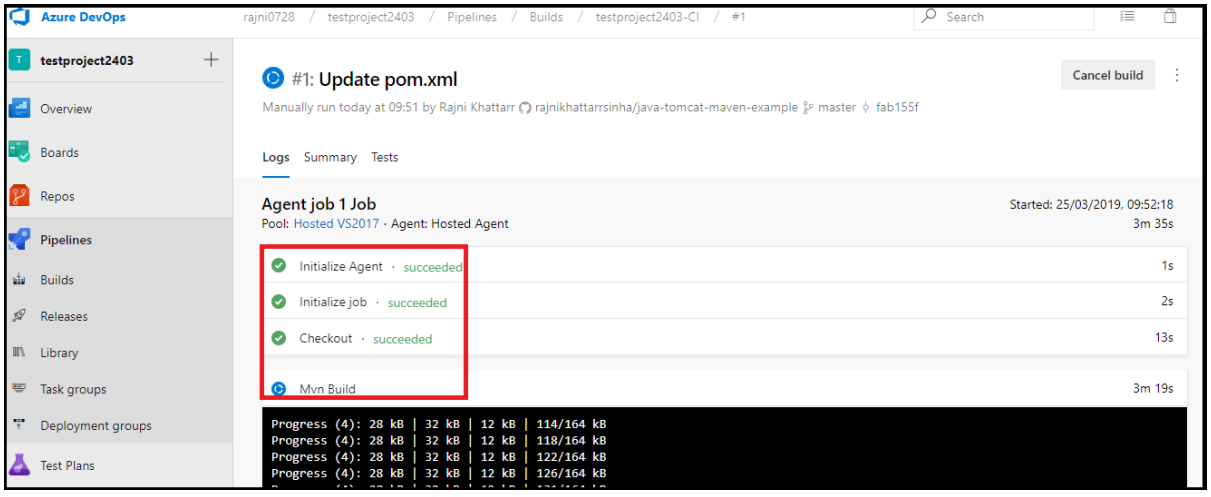
33.)Click on **Queue button** Queue build pop up screen



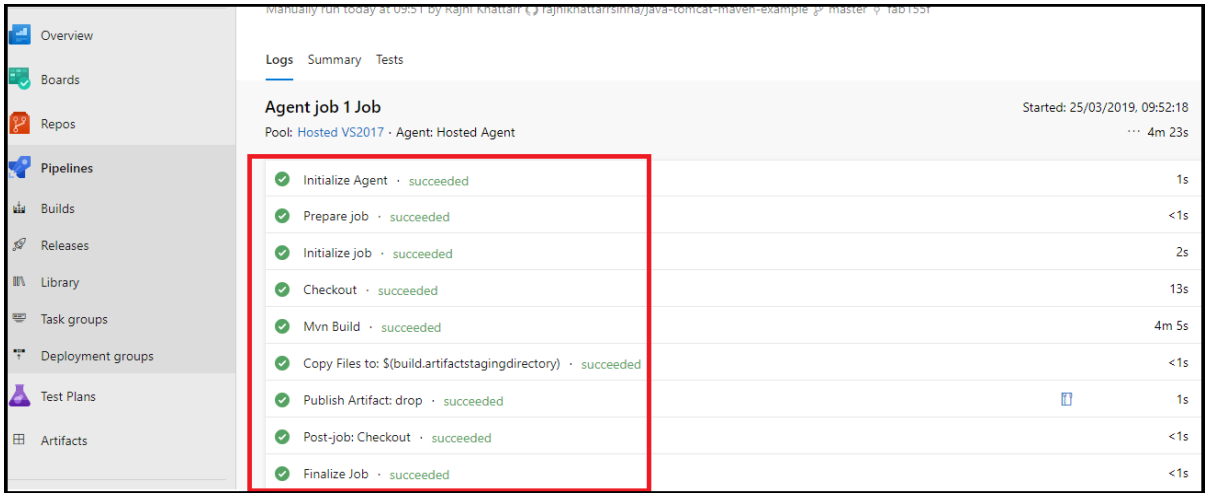
34.)Build gets triggered as shown in below image, Click on **#1** build link



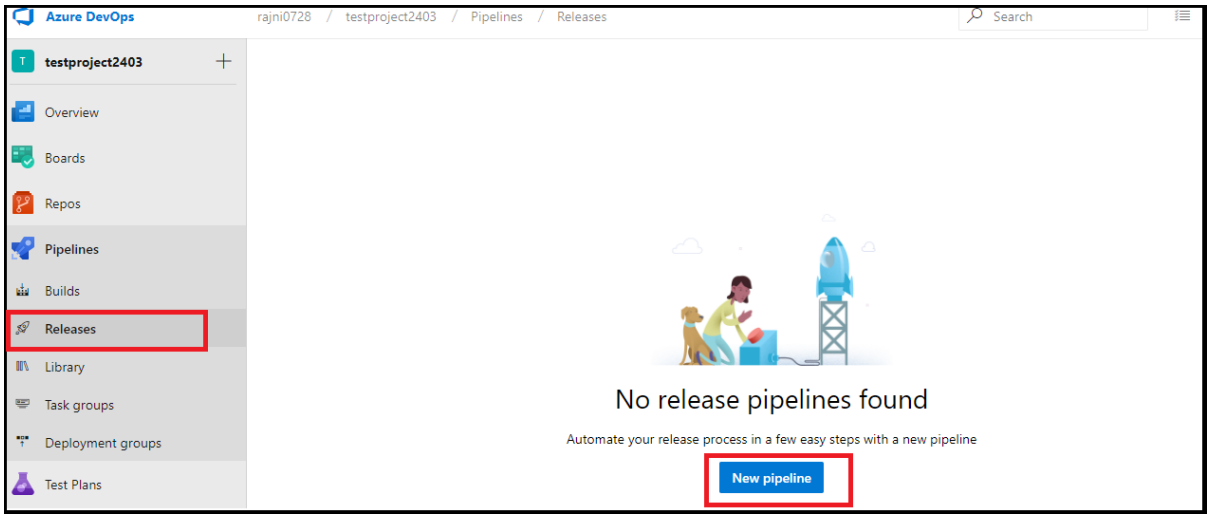
35.) The following screen get displayed and shows CI Pipeline stages building



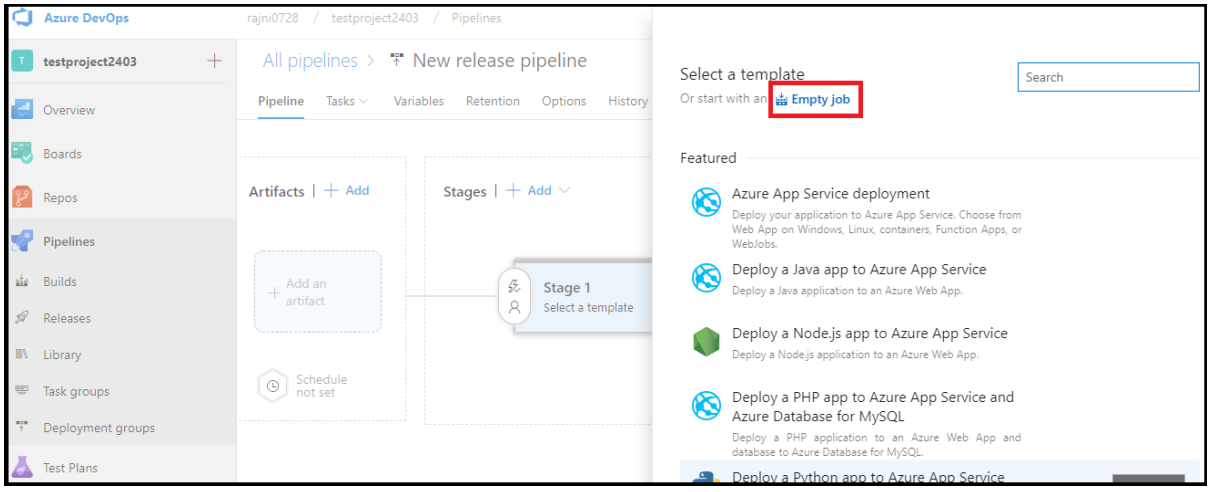
36.) Once the processing of CI Build Pipeline is completed, it shows the status of jobs as shown below



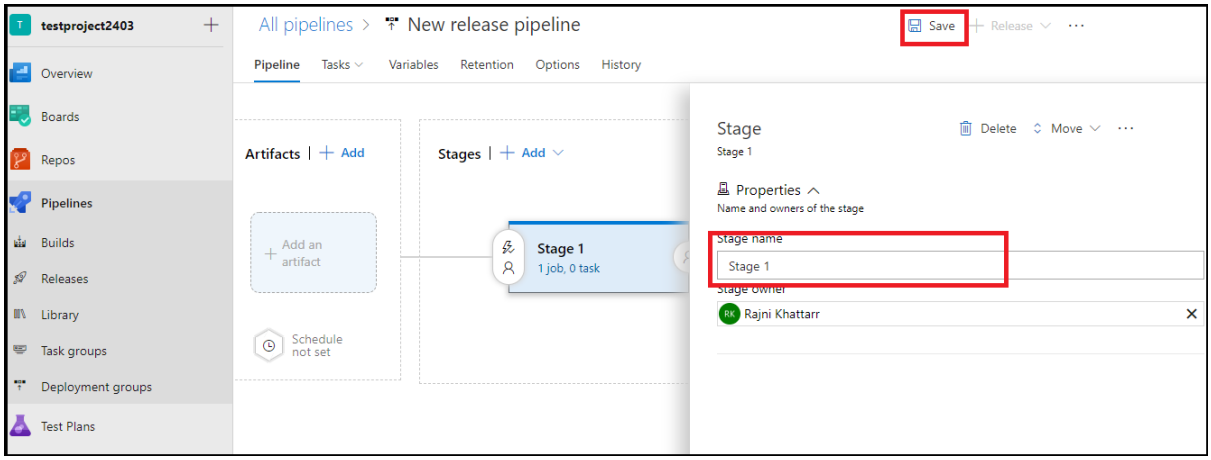
37.) Click on **Pipelines ->Releases** and click on **New pipeline**



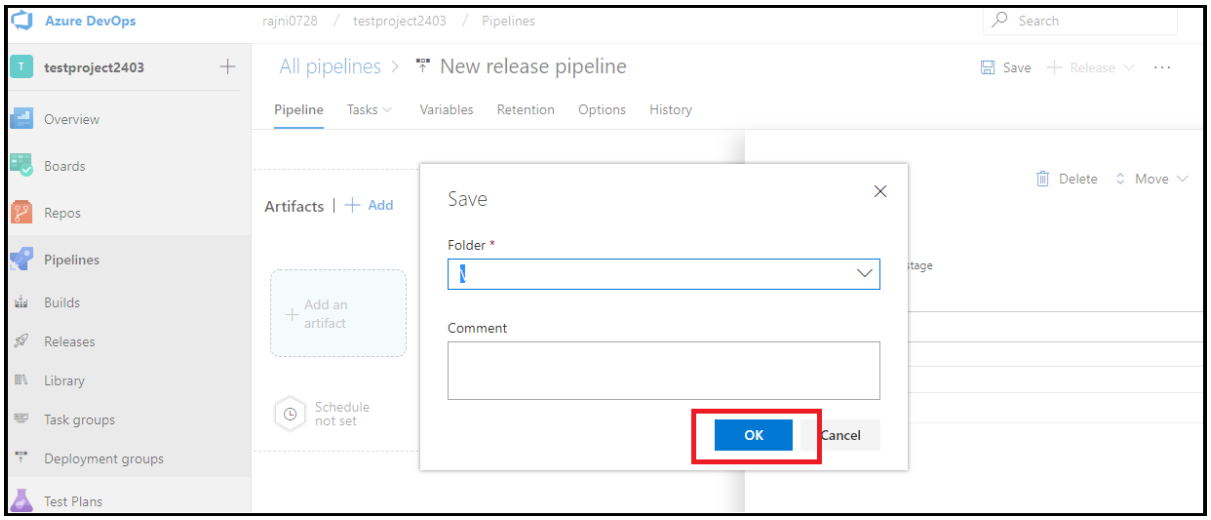
38.) Click on **Empty job**



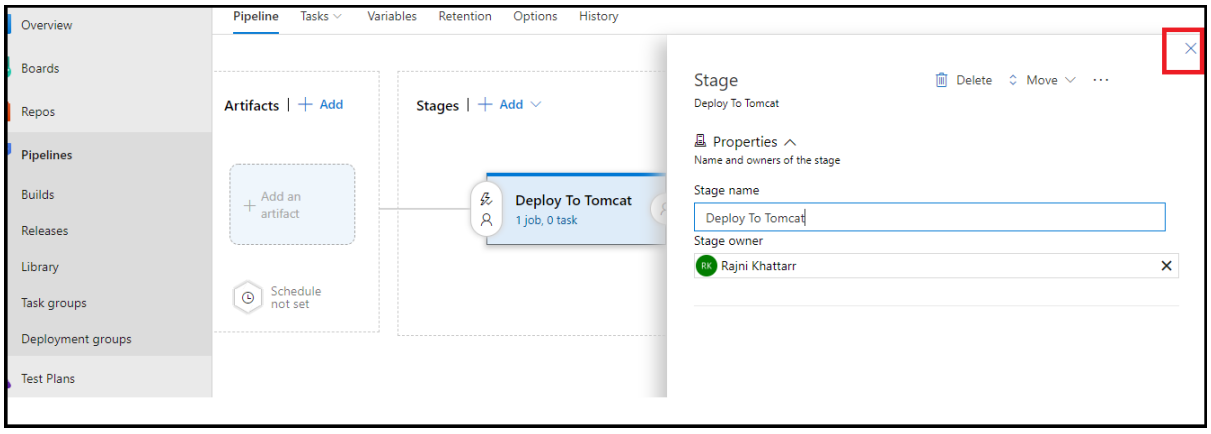
39.) Enter **Stage name** = Deploy To Tomcat and Click **Save.**



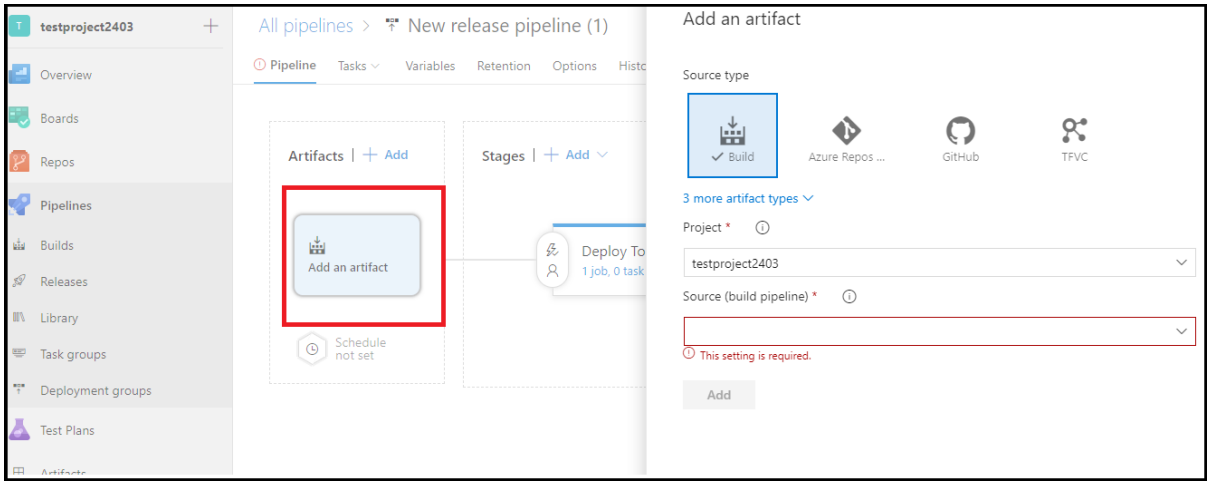
40.)Click **OK** on the **Save** pop up.



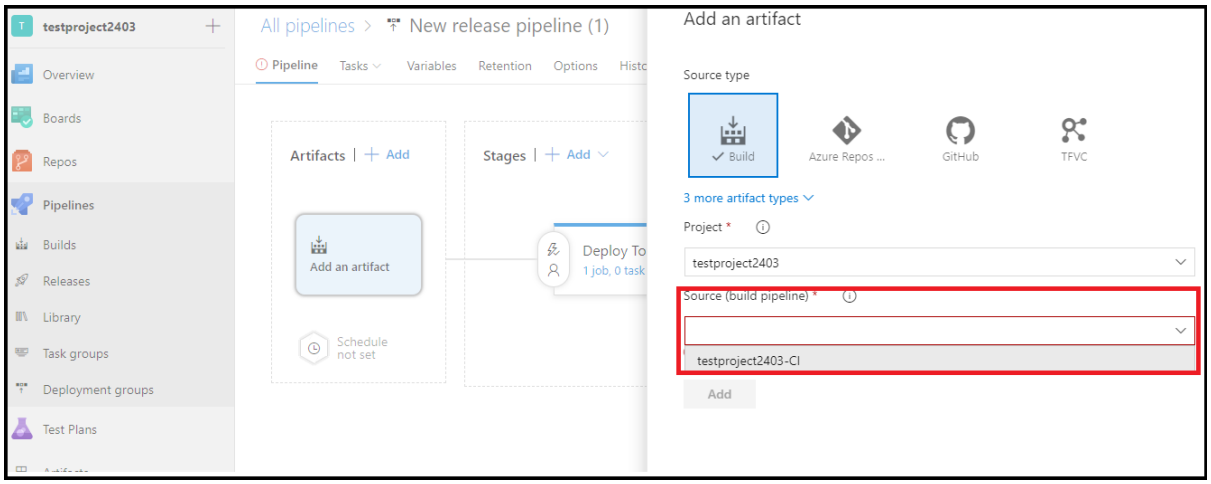
41.) Click on "X" of Stage panel.



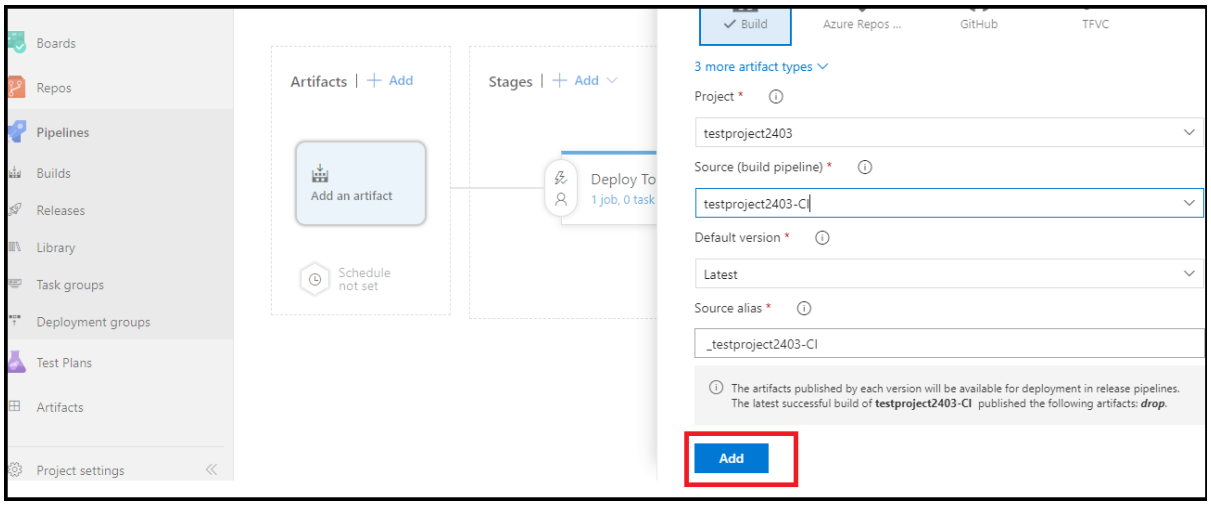
42.)Click on **Add an artifact** block



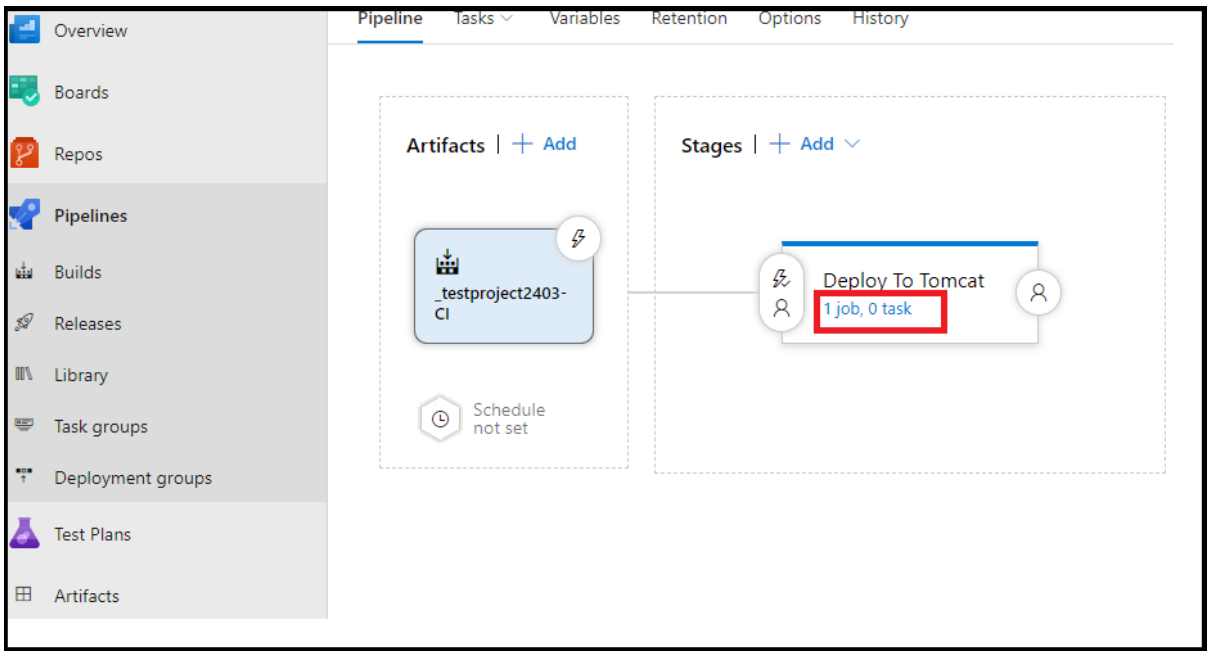
43.) Select **Source (Build Pipeline)** as the build pipeline created above.



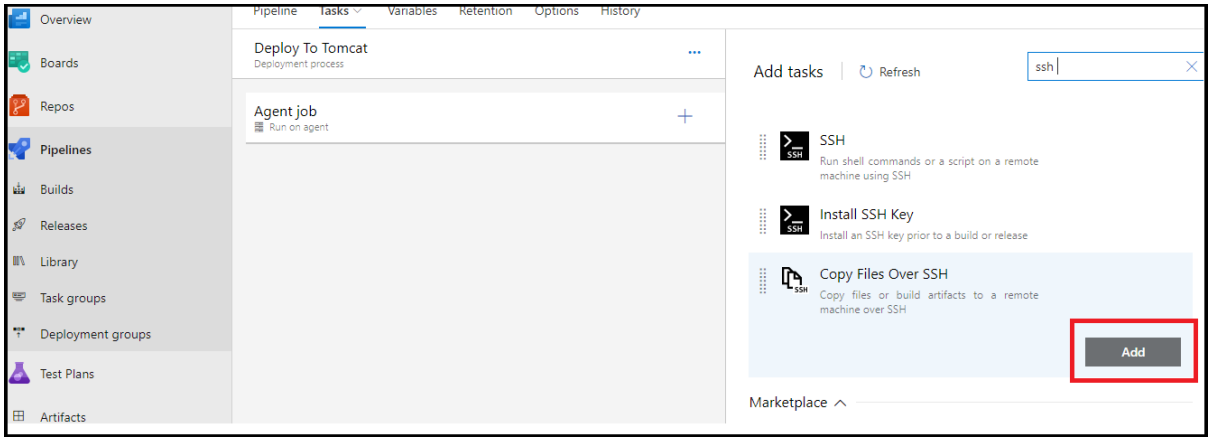
44.)Click on **Add**



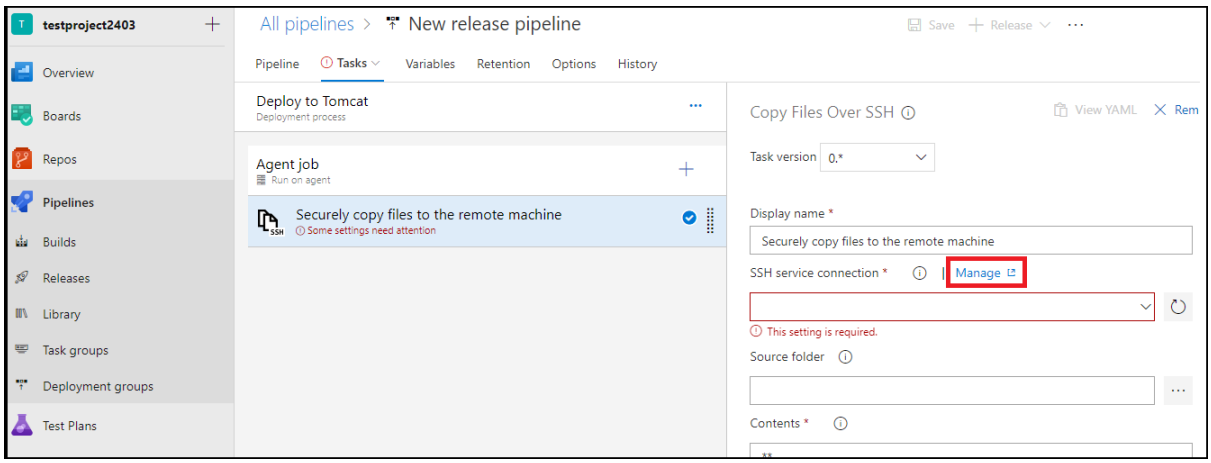
45.)Click on **1 job, 0 tasks** link



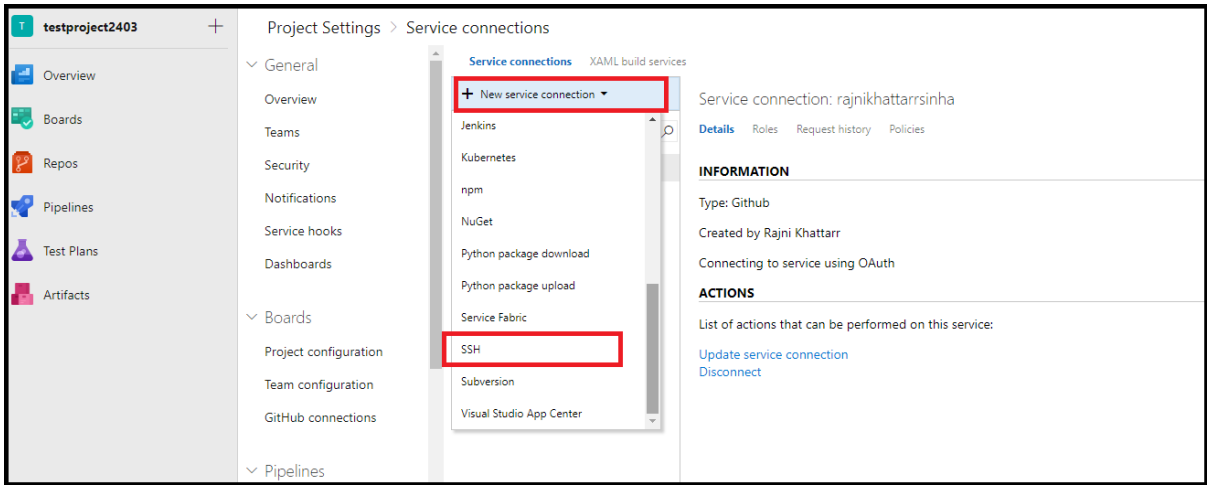
46.)Click on "+", Search **ssh,** Mouseover on **Copy Files Over SSH** and Click **Add**



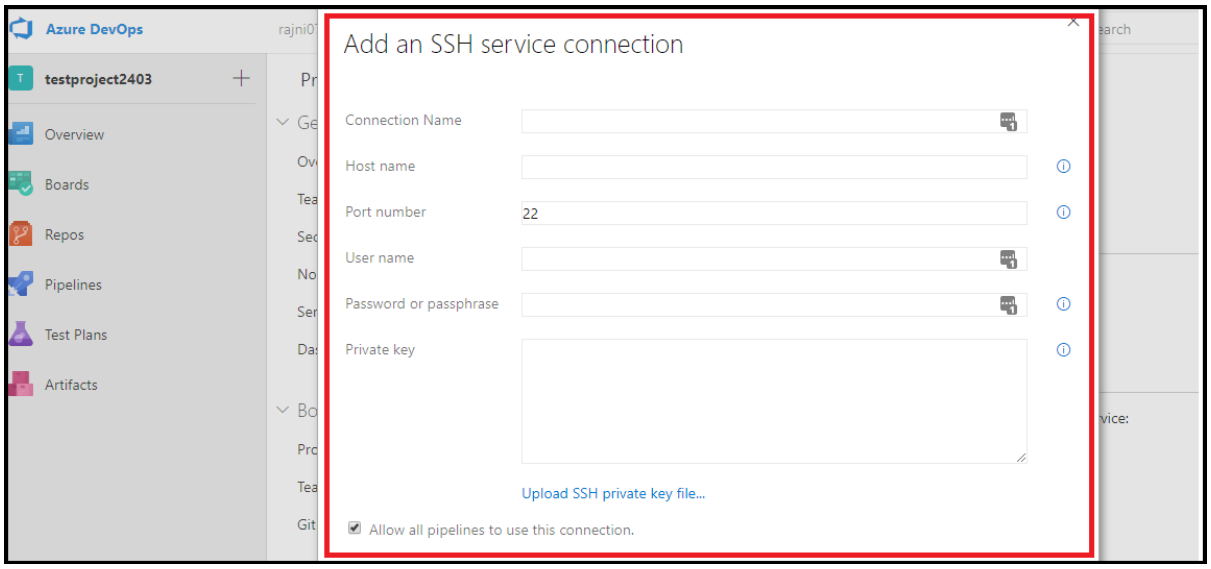
47.) Select the Agent Job "**Securely copy files to the remote machine**" , On Right panel , R-Click on **Manage** link and Open in new Tab.



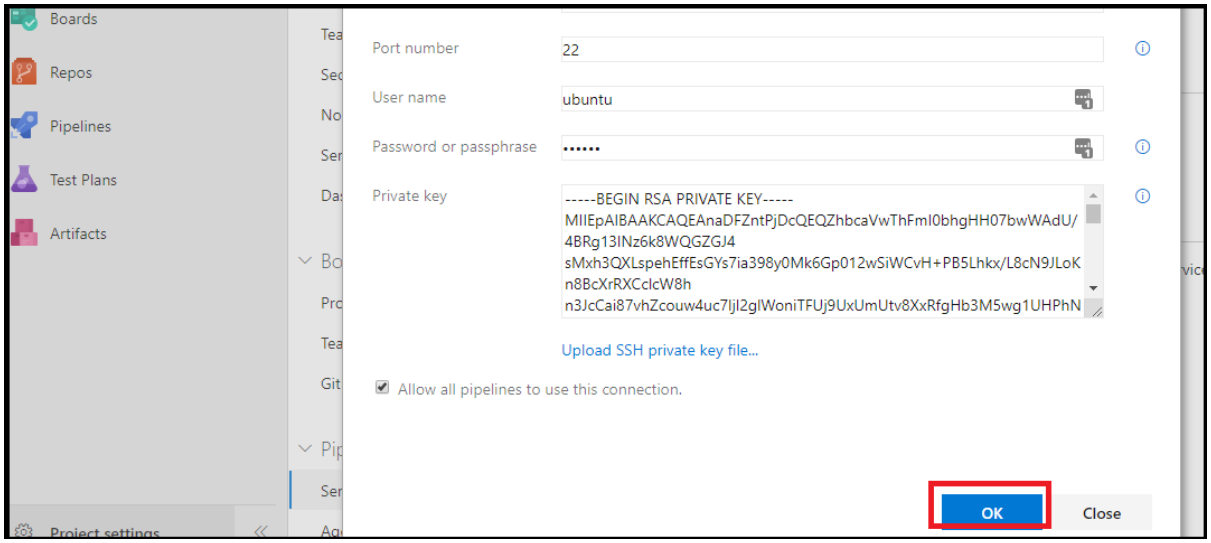
48.) Click on **New Service Connection,** Scroll down the list and Select **SSH**



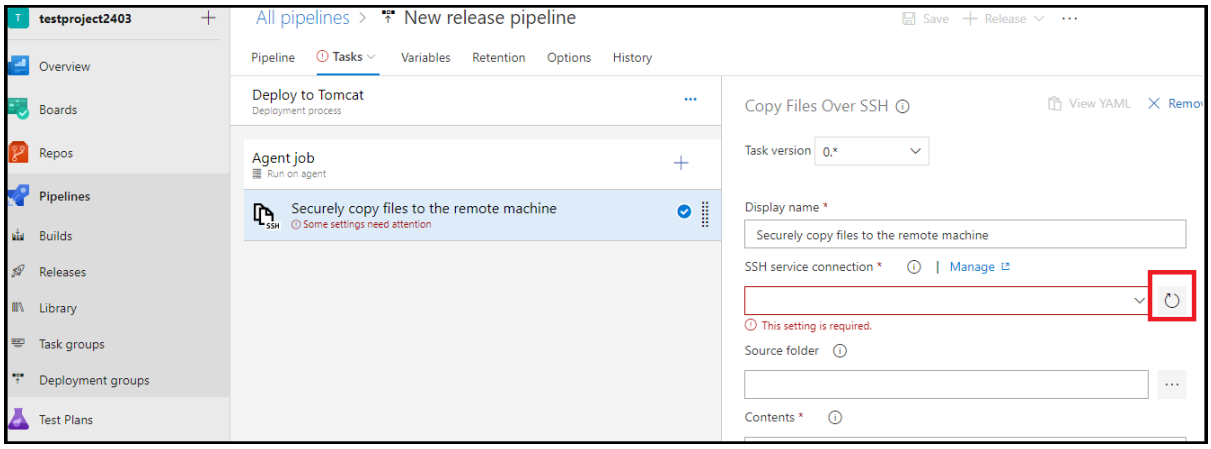
49.) Enter **Connection Name =** Tomcat Server, Enter **Host name** = IP of the Tomcat Server, Enter **User name =** ubuntu, Enter **Password or passphrase =** ubuntu**,** Enter **Private key** (Take from Trainer)



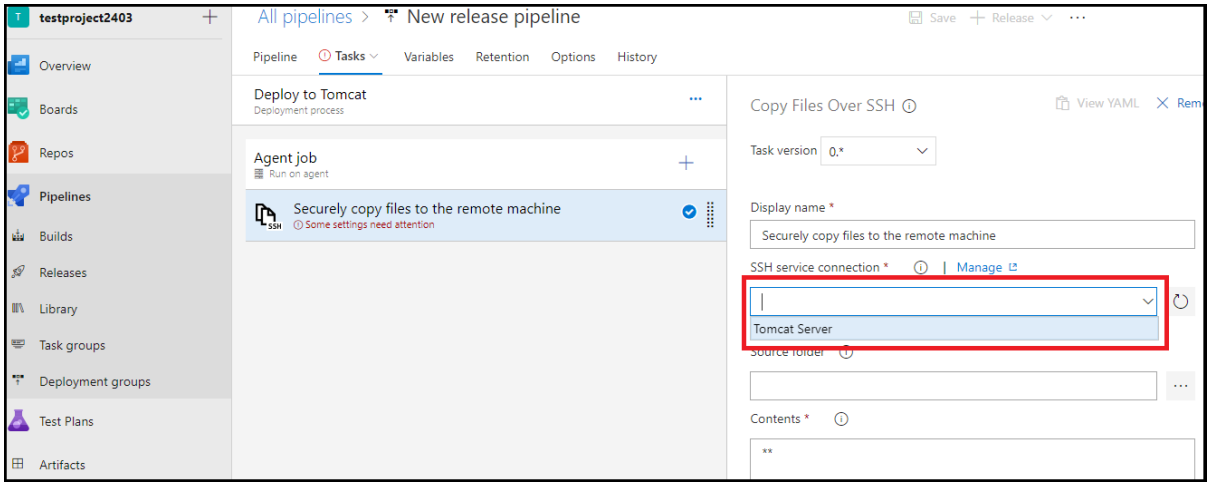
50.)Click on **OK** button



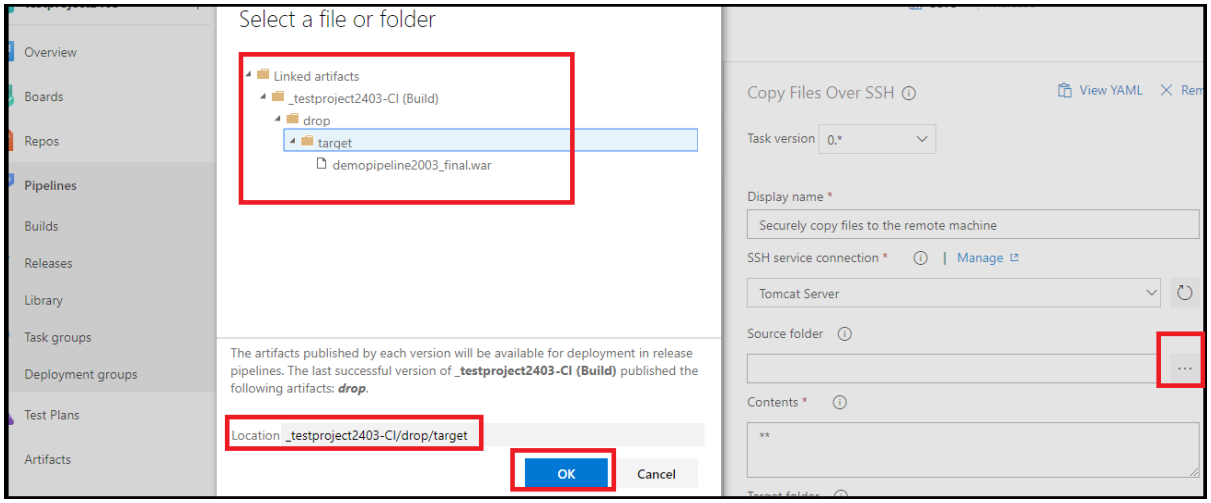
51.) Navigate back to Release pipeline opened in different Tab and Click on Refresh icon of **SSH Service Connection**



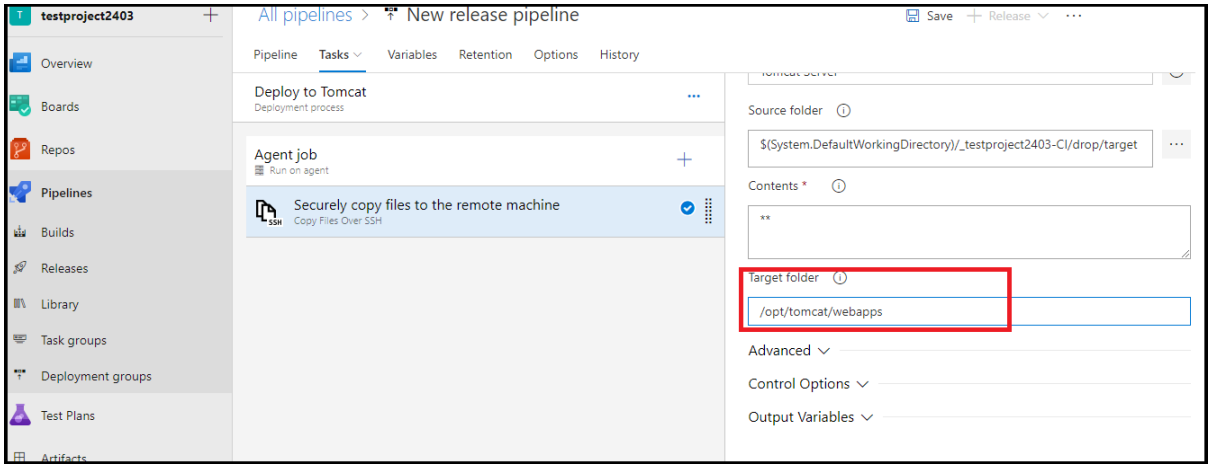
52.) Select **SSH service connection** =Tomcat Server



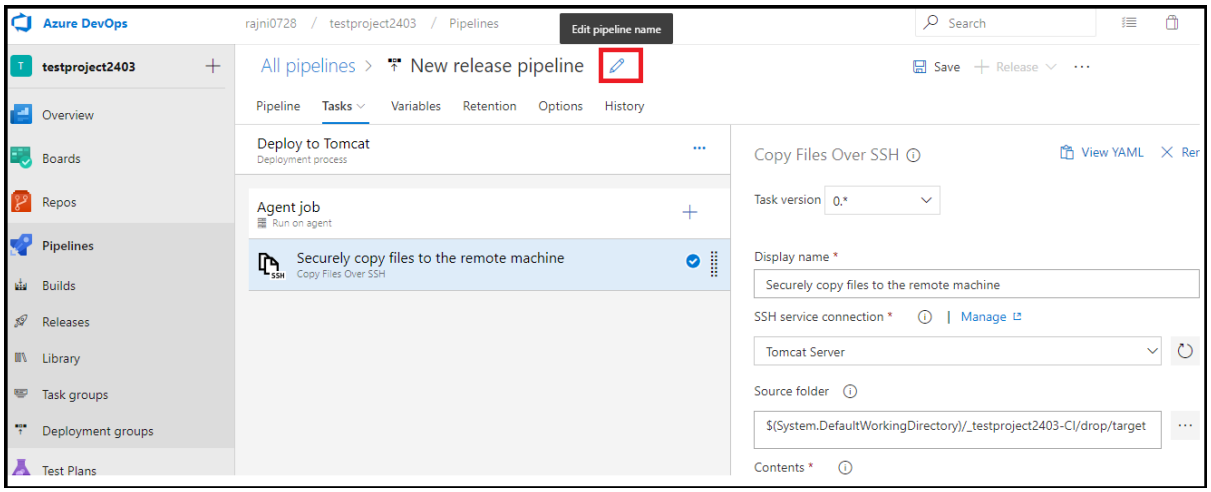
53.)Click on … of **Source folder** field and Select the folder till target which should display the selected path in **Location** field as shown below and then click OK



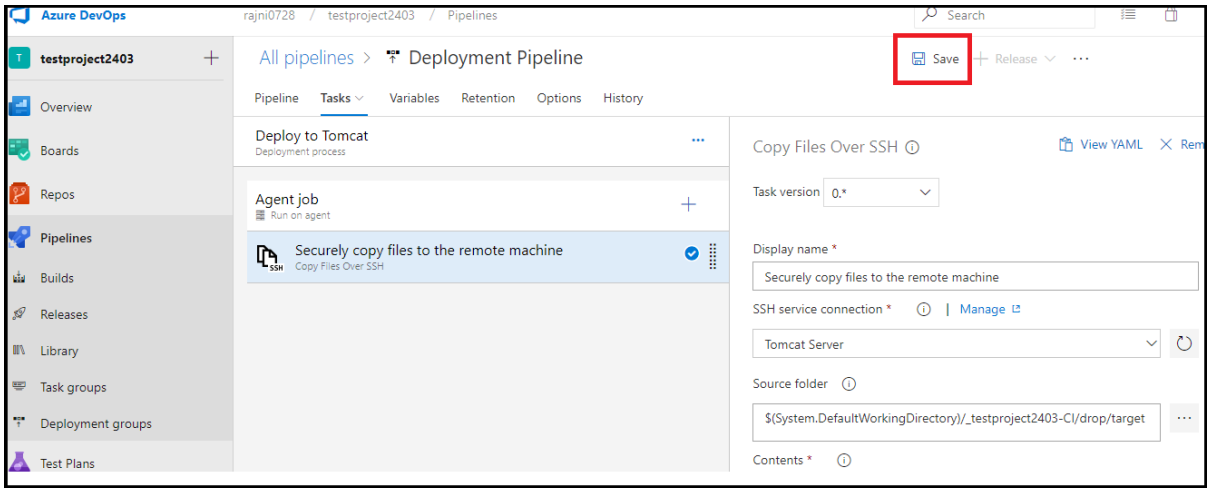
54.) Scroll down and Enter **Target folder** =/opt/tomcat/webapps



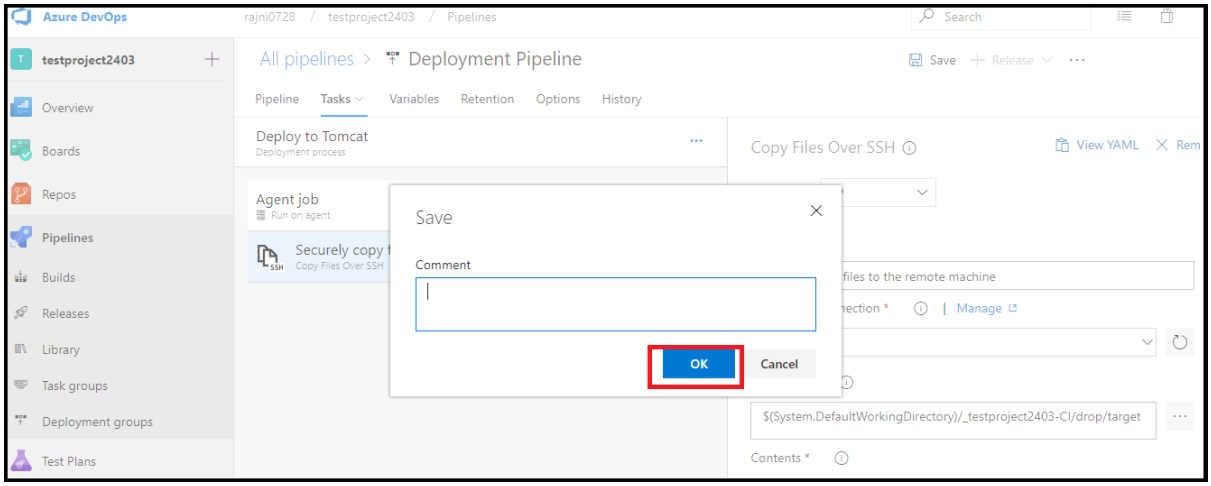
55.) Scroll up and Click on Edit icon of Release Pipeline name and Enter name as Deployment pipeline



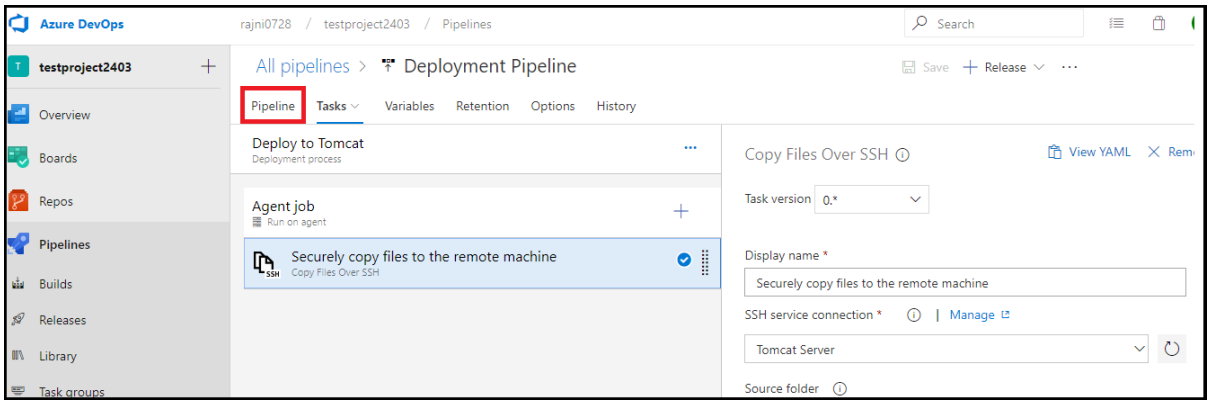
56.) Click on **Save**



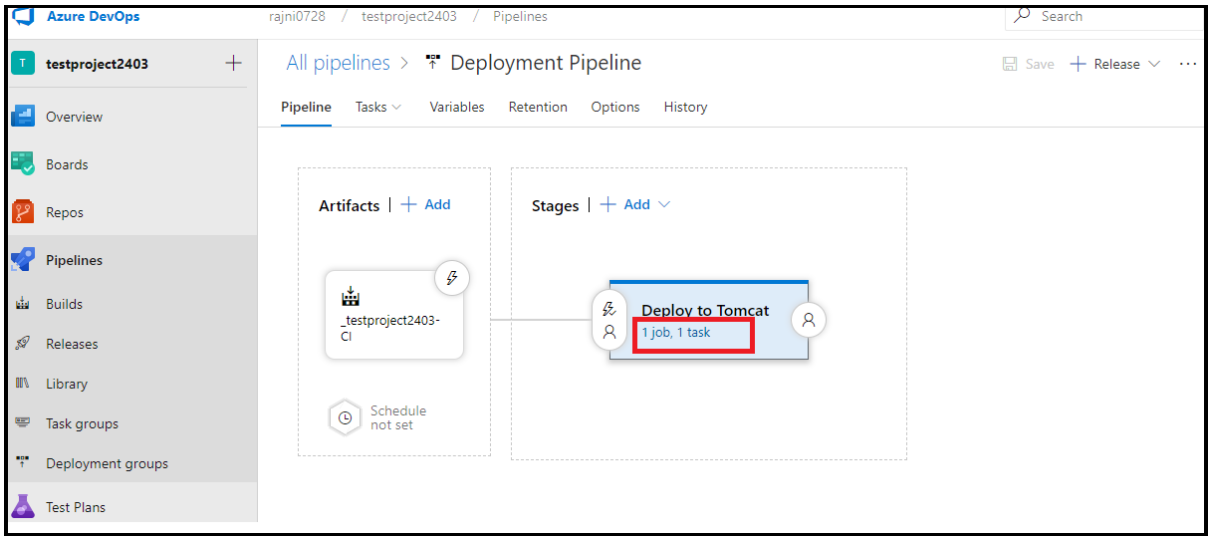
57.) Click on **OK**



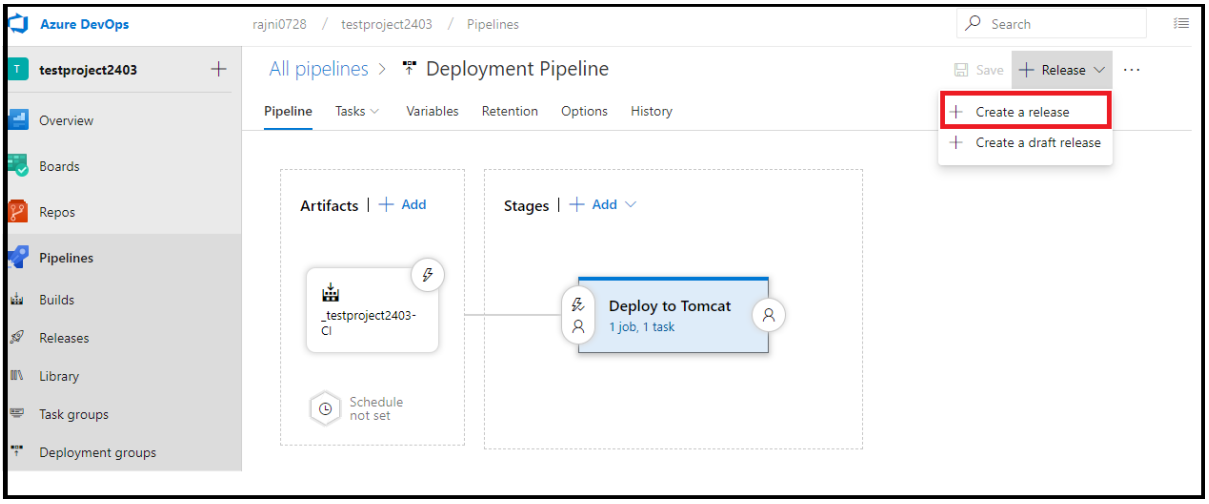
58.)Click on the **Pipeline** Tab



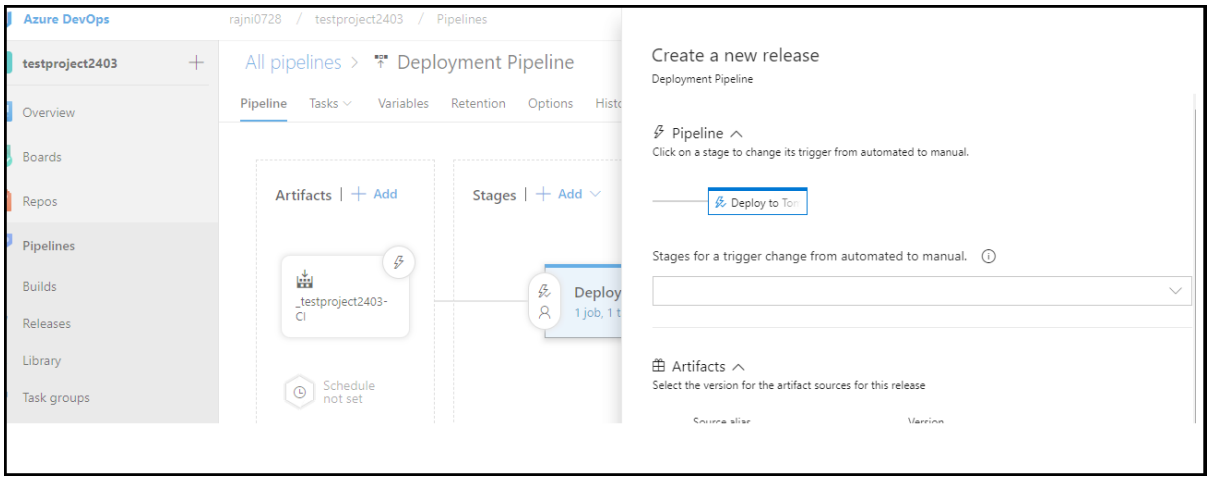
59.) Now Stage shows 1 job, 1 task



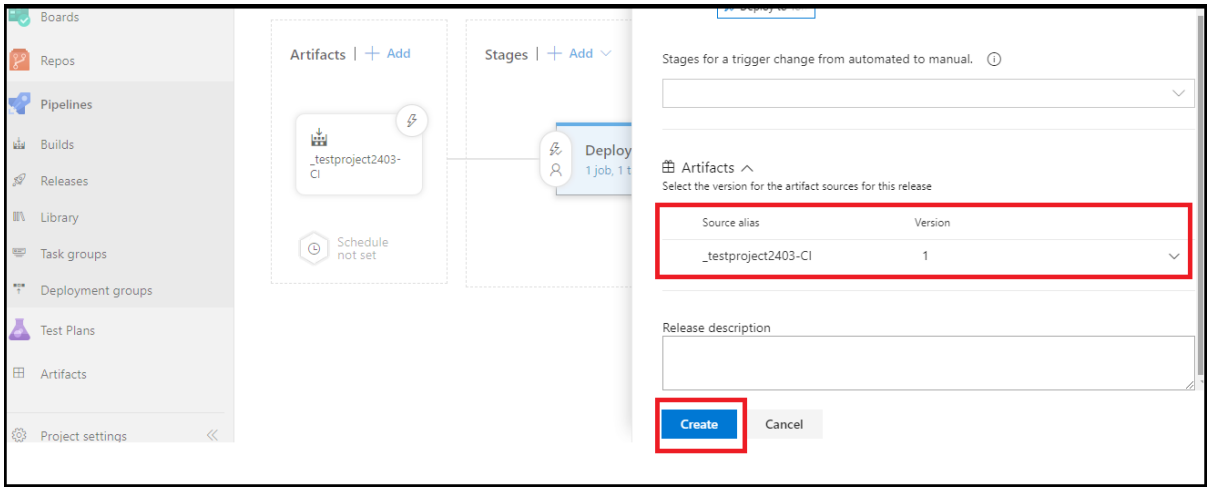
60.)Now click on **Release** dropdown and Select **Create a release**



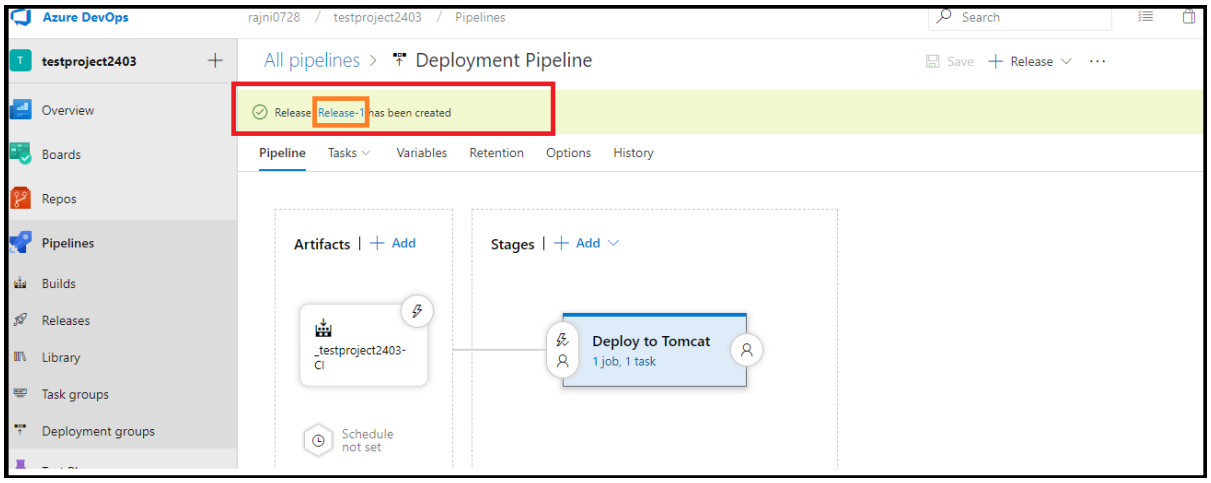
61.)The following screen is displayed showing pipeline Deploy to Tomcat stage



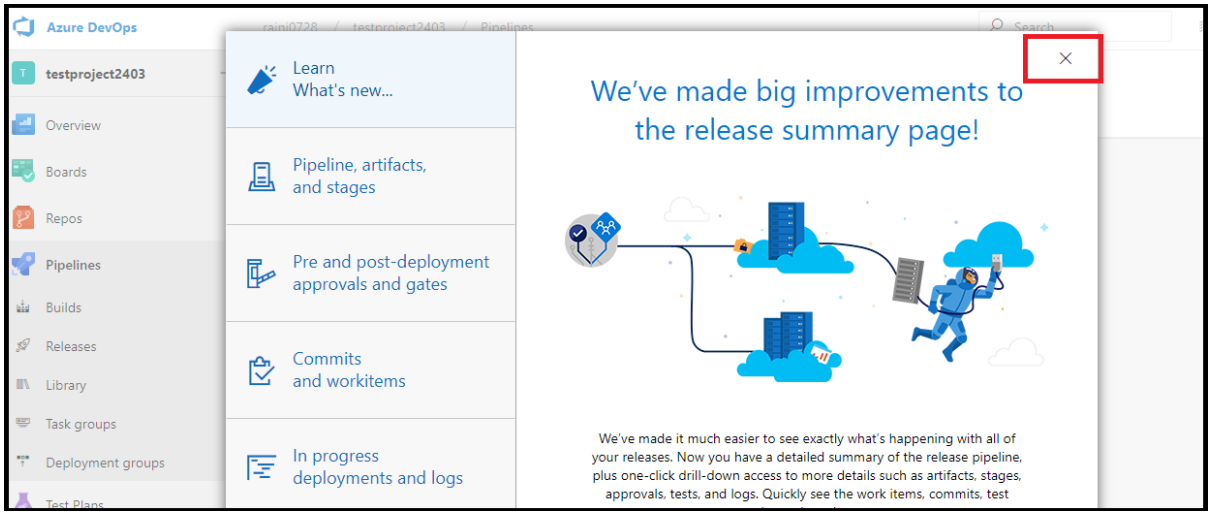
62.)Scroll down and View the **Artifacts** section showing Source alias and Version. Click on **Create**



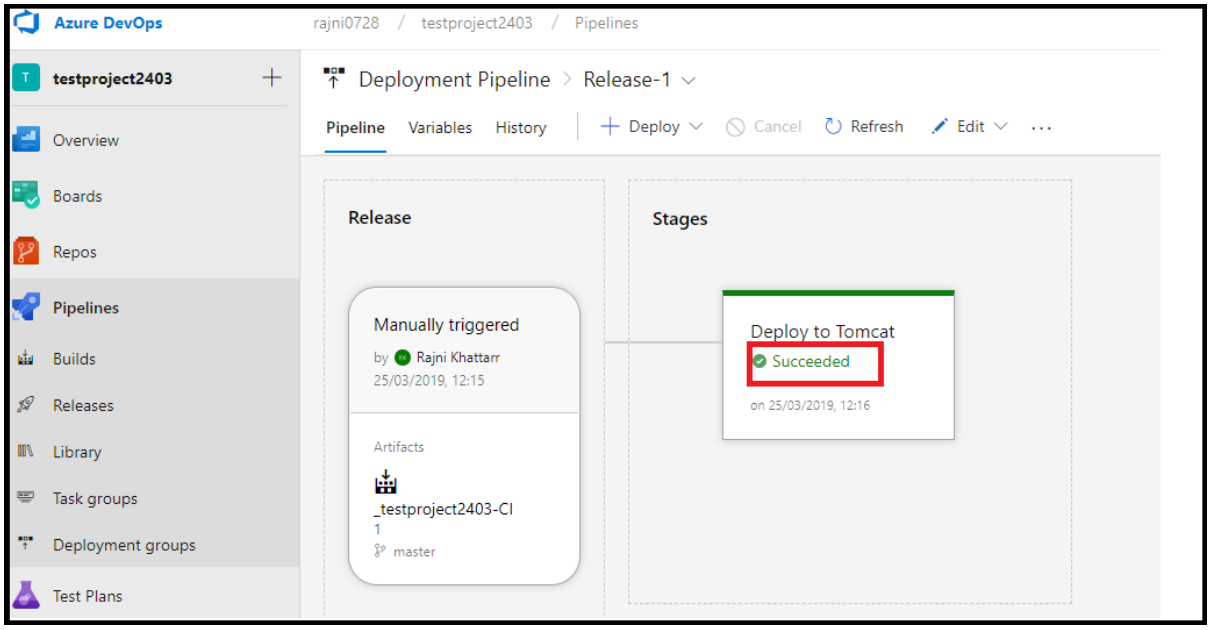
63.)Release -1 will be created and displayed on top. Click on **Release-1** link



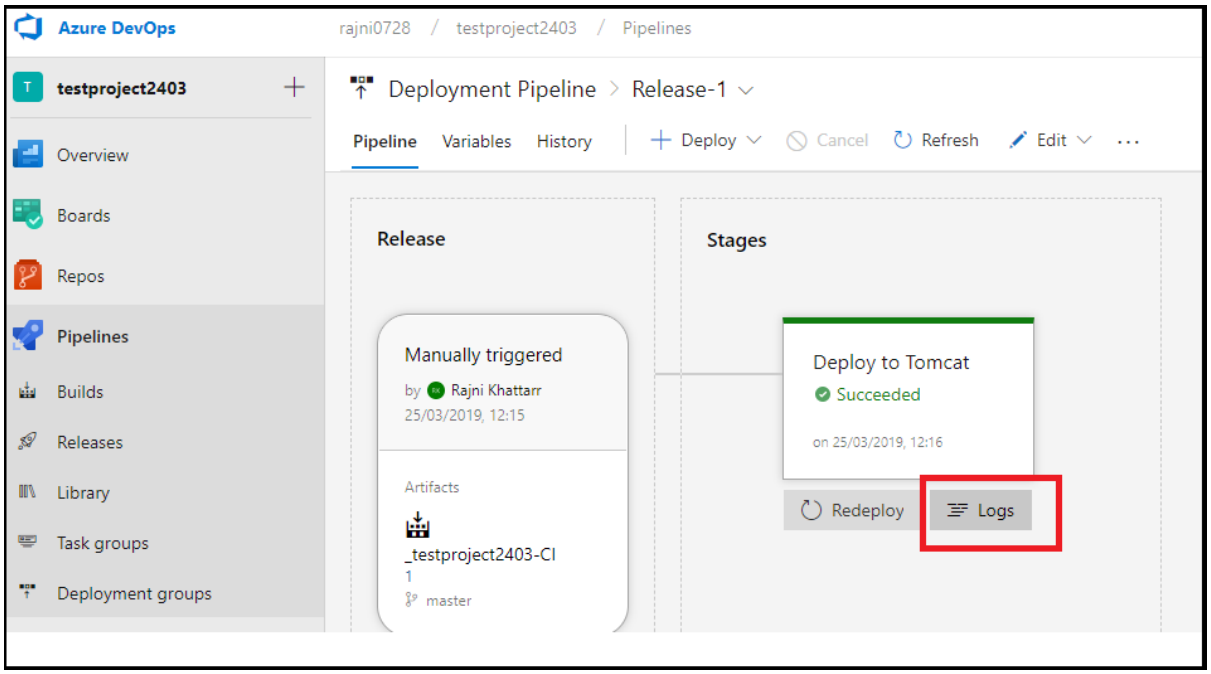
64.)You may get this screen as we are using this Release pipeline for the first time, Click on **"X"** .



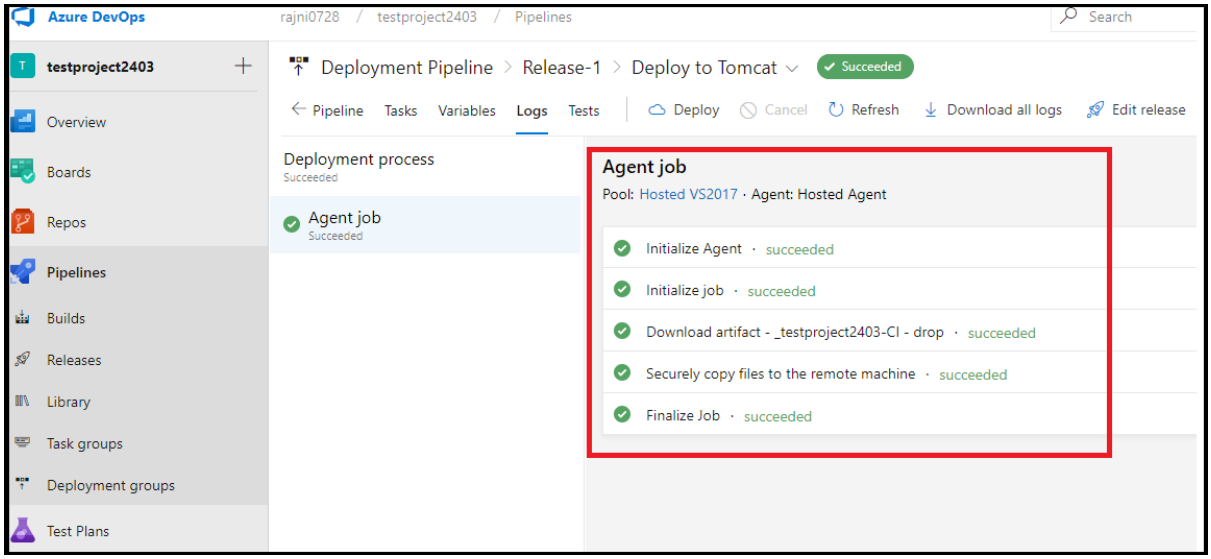
65.) View the Screen showing Deployment succeeded.



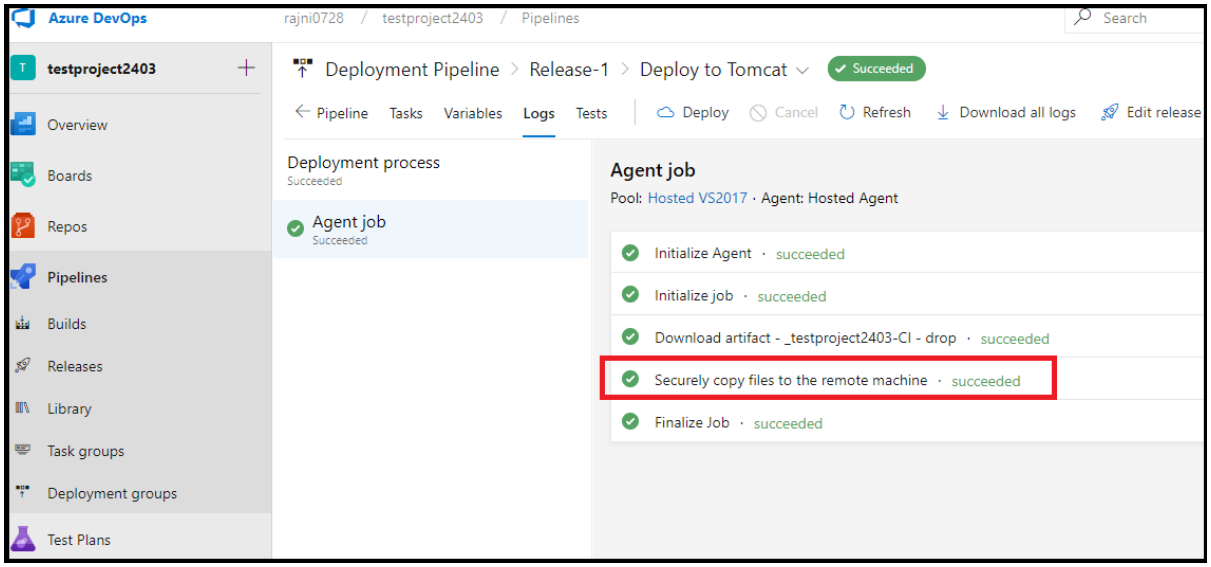
66.)Click on Logs



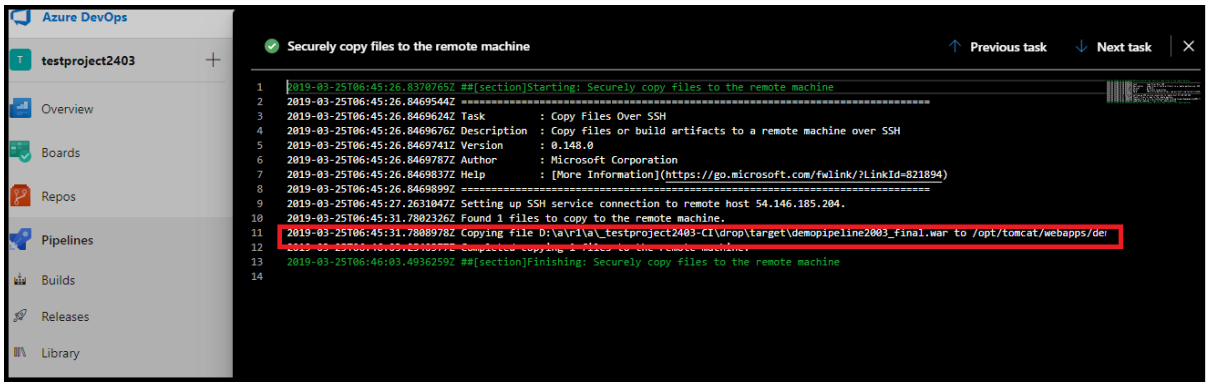
67.) View the logs showing all stages as succeeded.



68.)Click on **Securely copy files to the remote machine**



69.)Following logs are displayed



70.) Open any browser, Enter the URL in the following format

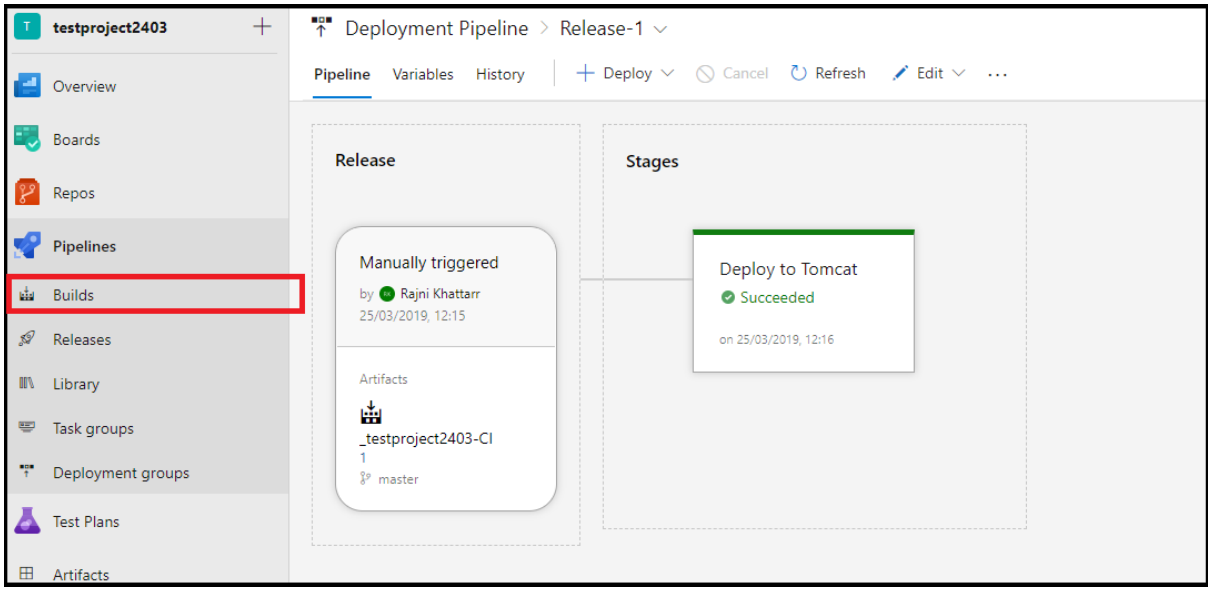
http://<tomcat server ip>:8080/<name of the war file>

Note: Tomcat Server IP and name of war to be taken as mentioned in Pre-requisites

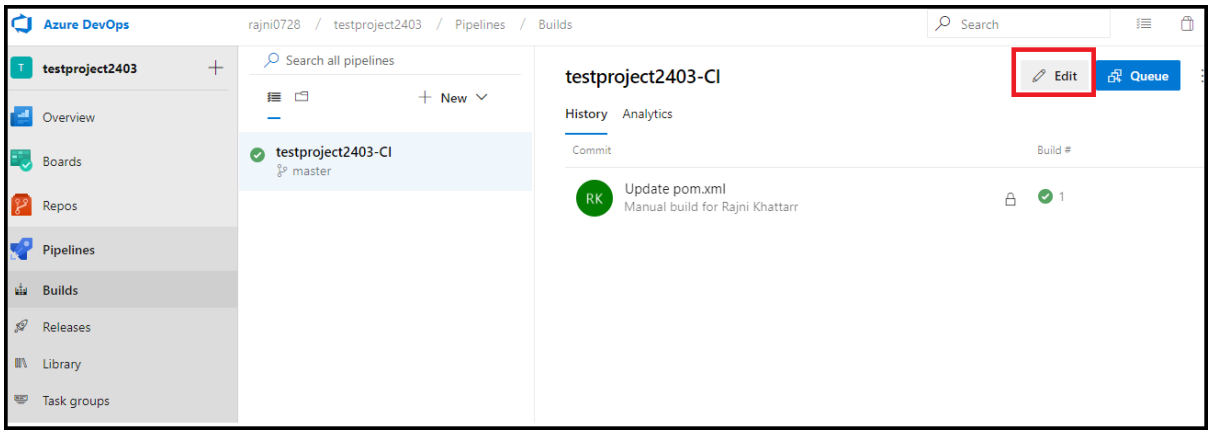
Result: It will display the java web application deployed on tomcat server.

Lab 2: Automated CICD Pipeline

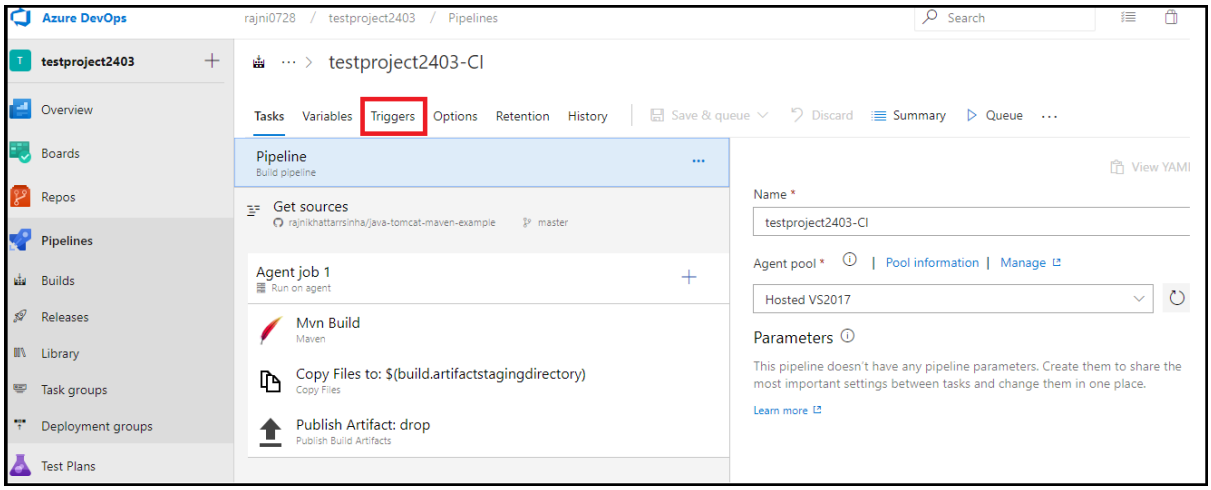
71.) Click on the **Builds** in Left Panel



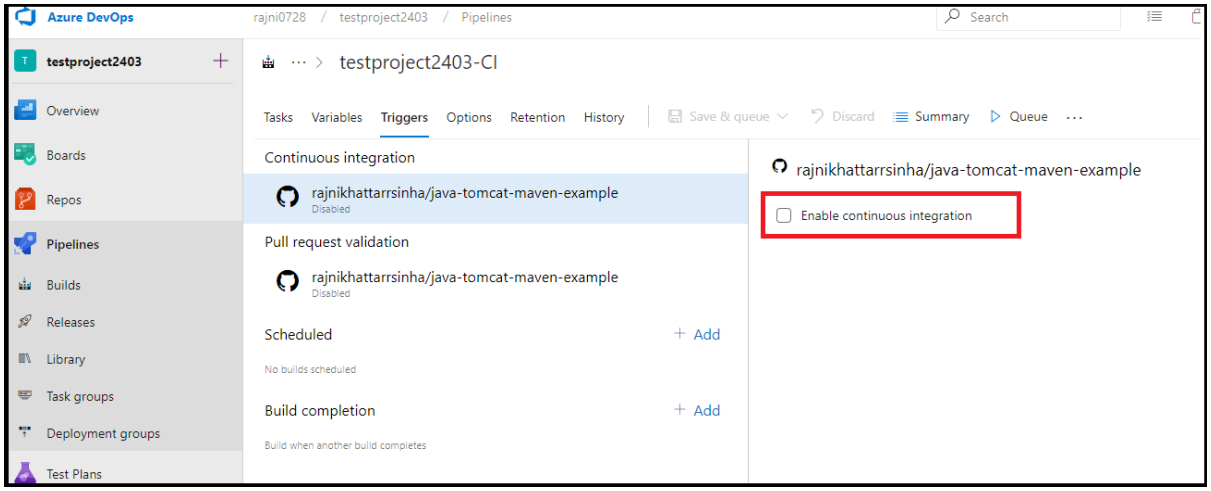
72.)Click on **Edit**



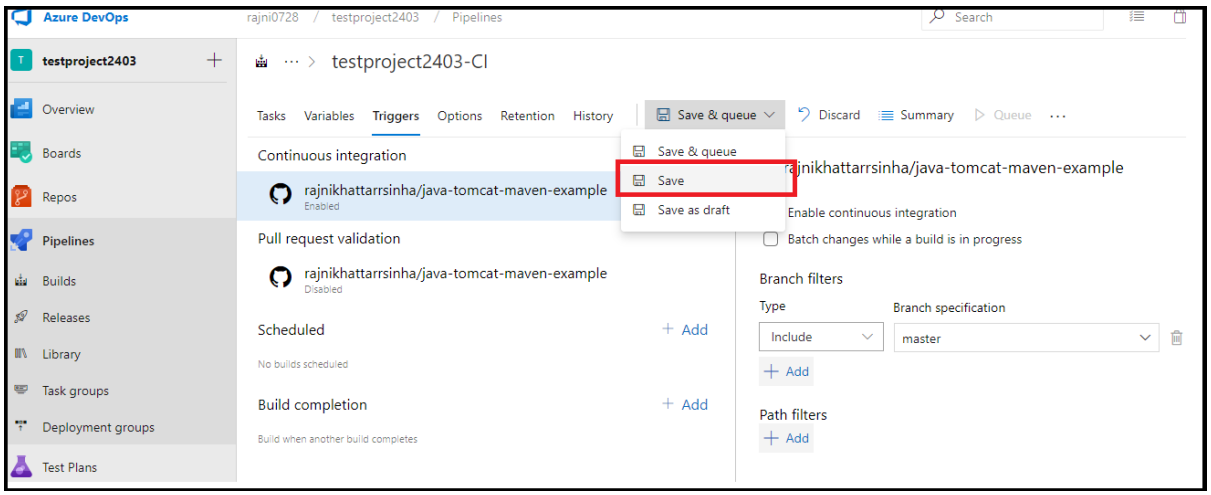
73.) Click on **Triggers** tab



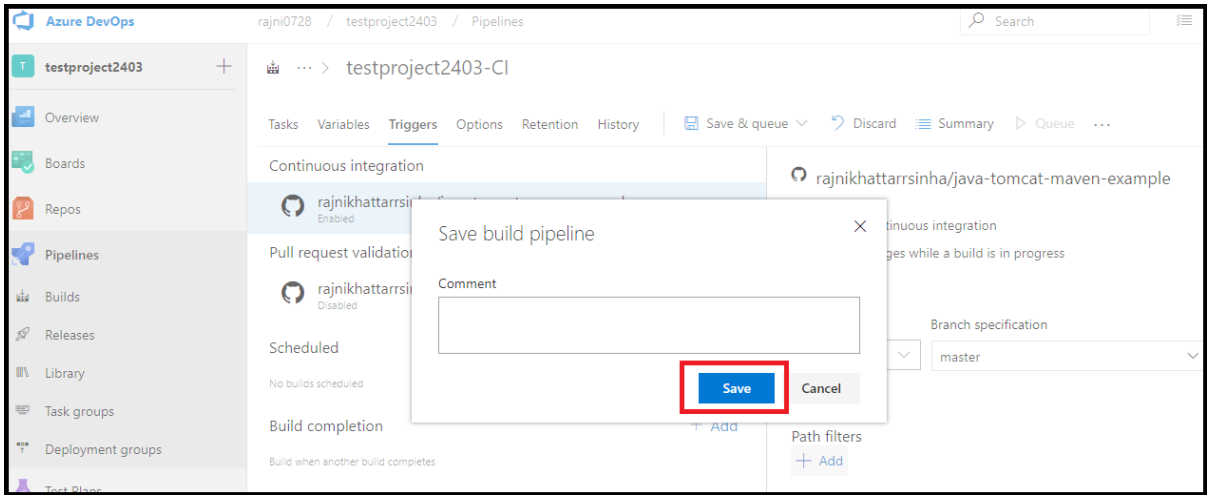
74.)Select **Enable continuous integration** checkbox



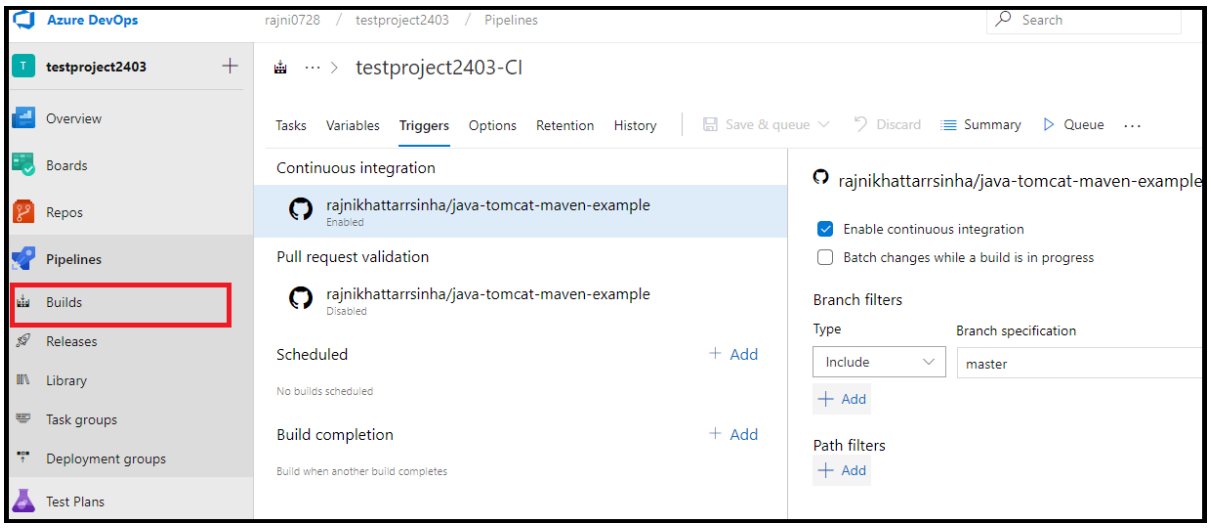
75.)Click on **Save & queue** dropdown and Click **Save**



76.)Click on **Save** button on **Save build pipeline** pop up.

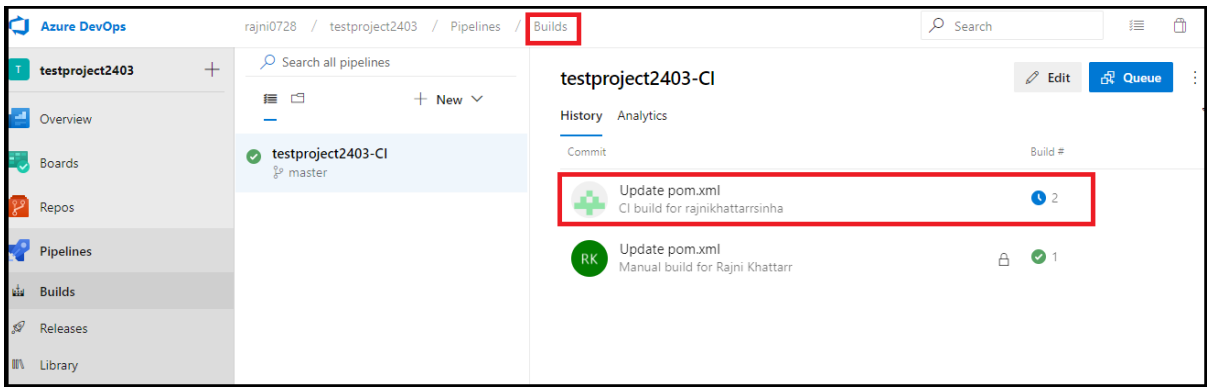


77.)Click on **Builds** from Left panel

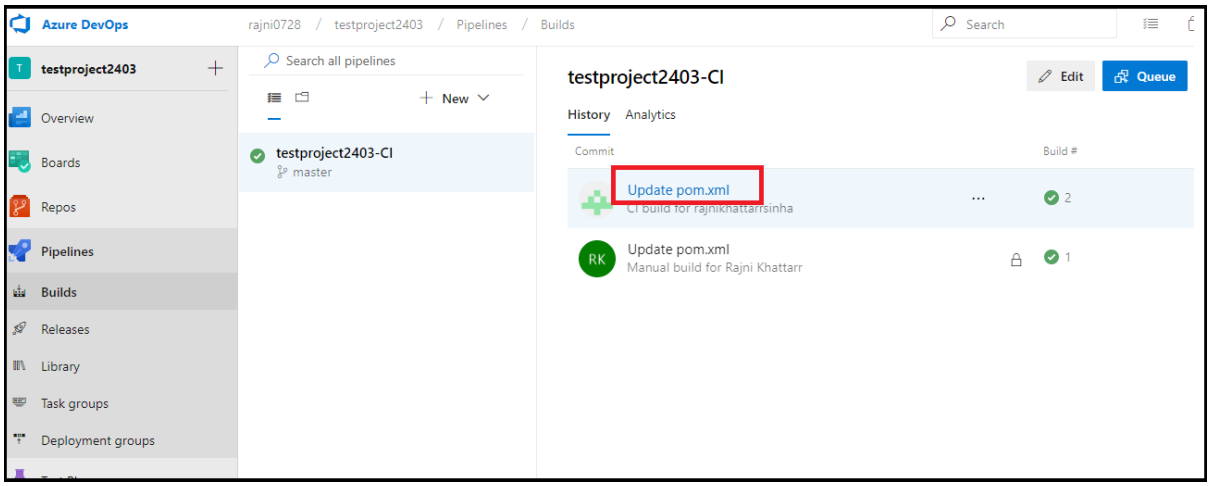


78.) Navigate to Github, open your repo forked for this lab. Open pom.xml in edit mode and change the name of war file as

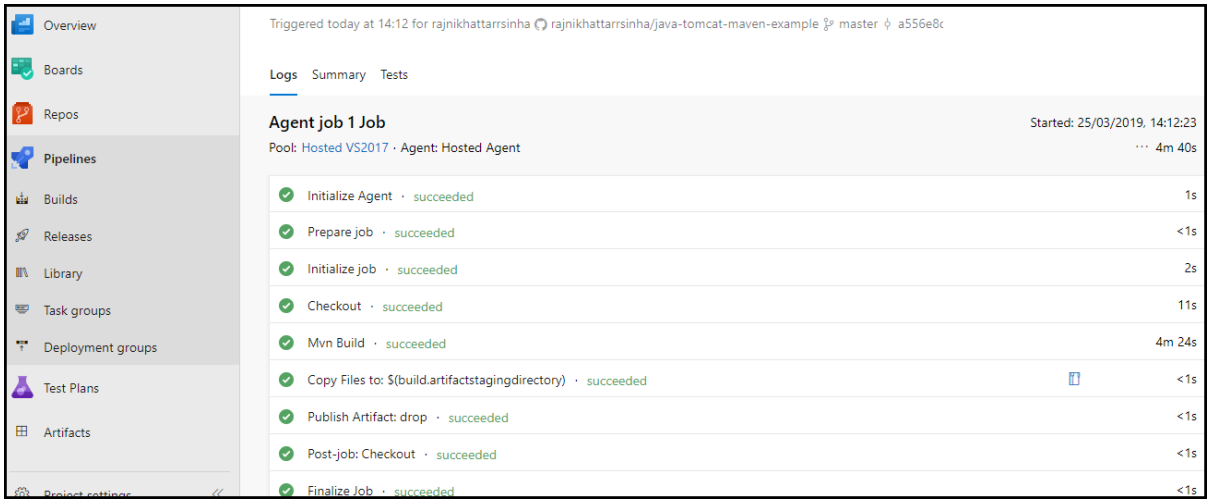
79.) As soon as the change in pom.xml is committed, click on the **Builds** link on the top to refresh the page and view the build # 2 is triggered.



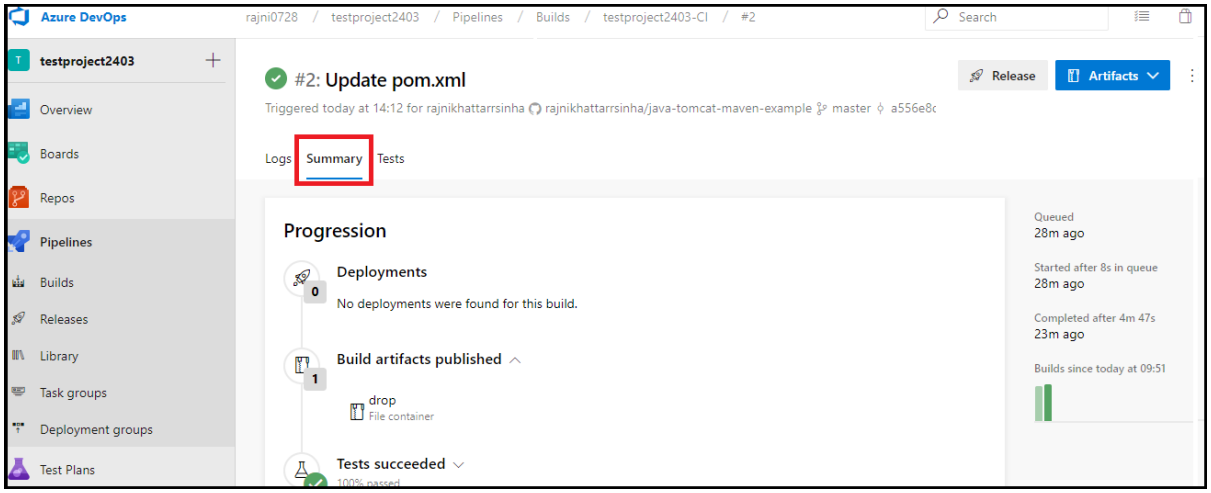
80.) Wait for some time and Click on **Update pom.xml** link once Build # 2 shows green tick mark.



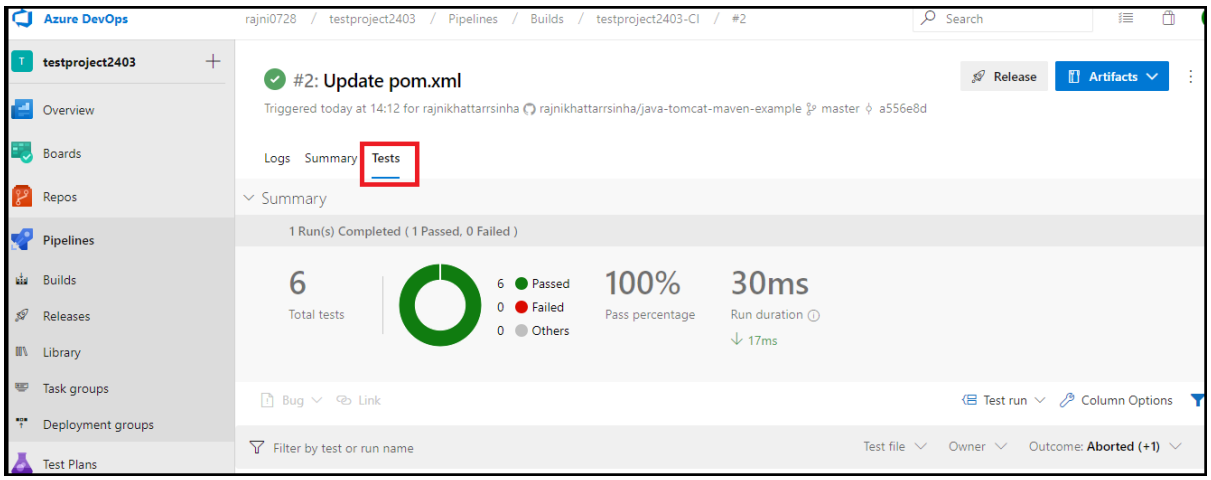
81.) View the **Logs** as shown below



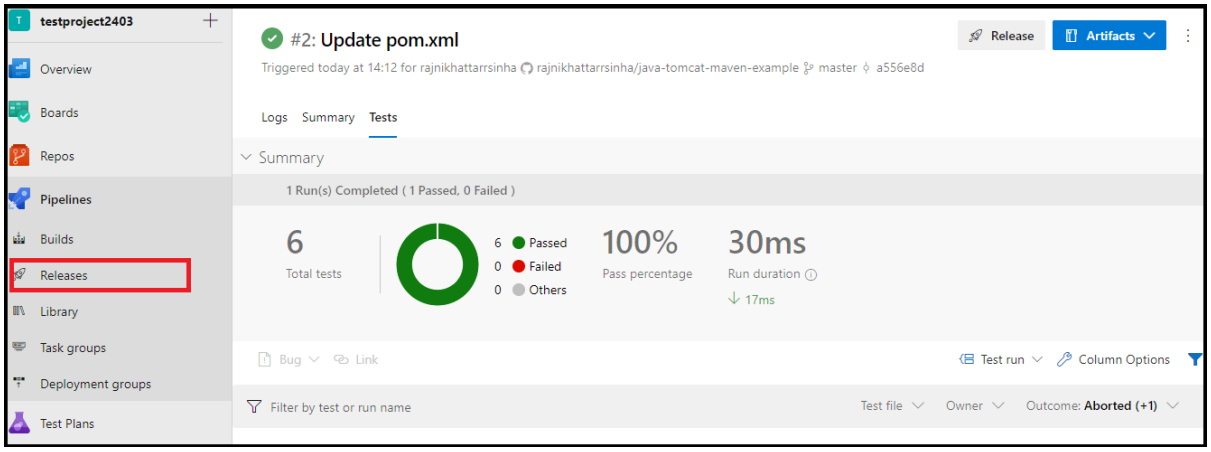
82.) Navigate to **Summary** tab and view and explore the screen



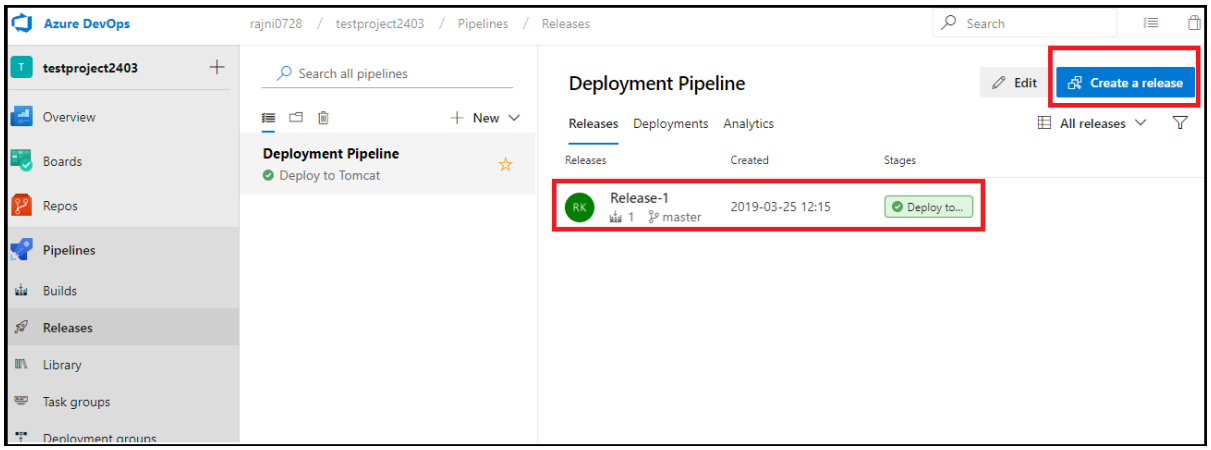
83.) Navigate to **Tests** tab and View the Test Results



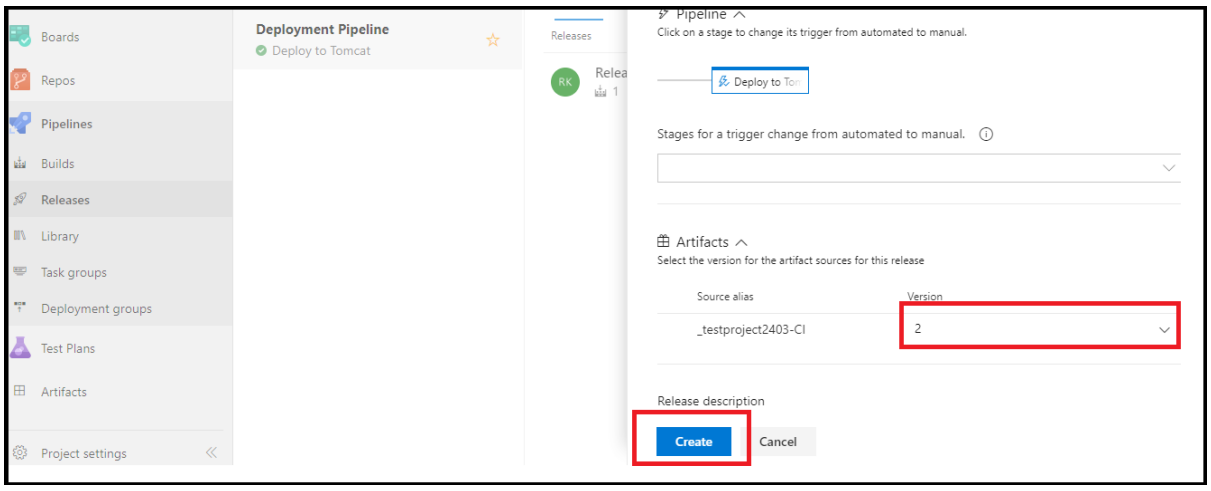
84.) Click on **Releases** from Left Panel.



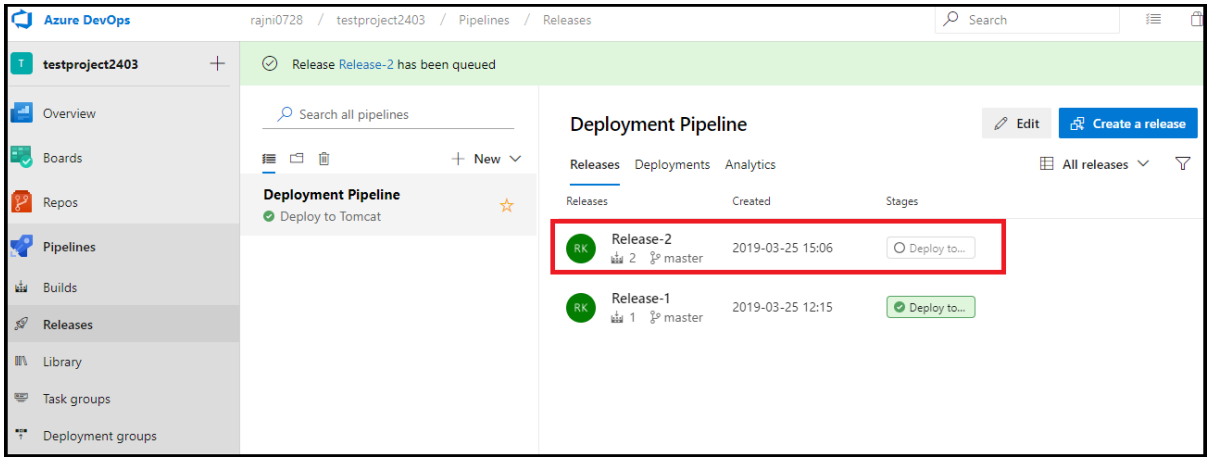
85.) The screen shows that Release-1 is done on version # 1 of artifacts. Click on **Create a release** button.



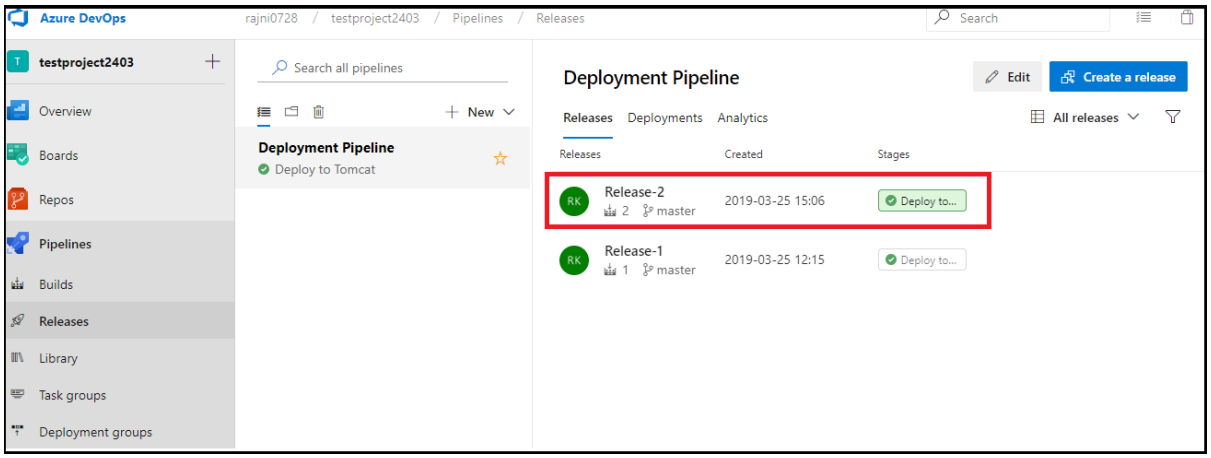
86.)View the **Artifacts** section now showing version # 2 of the Artifacts



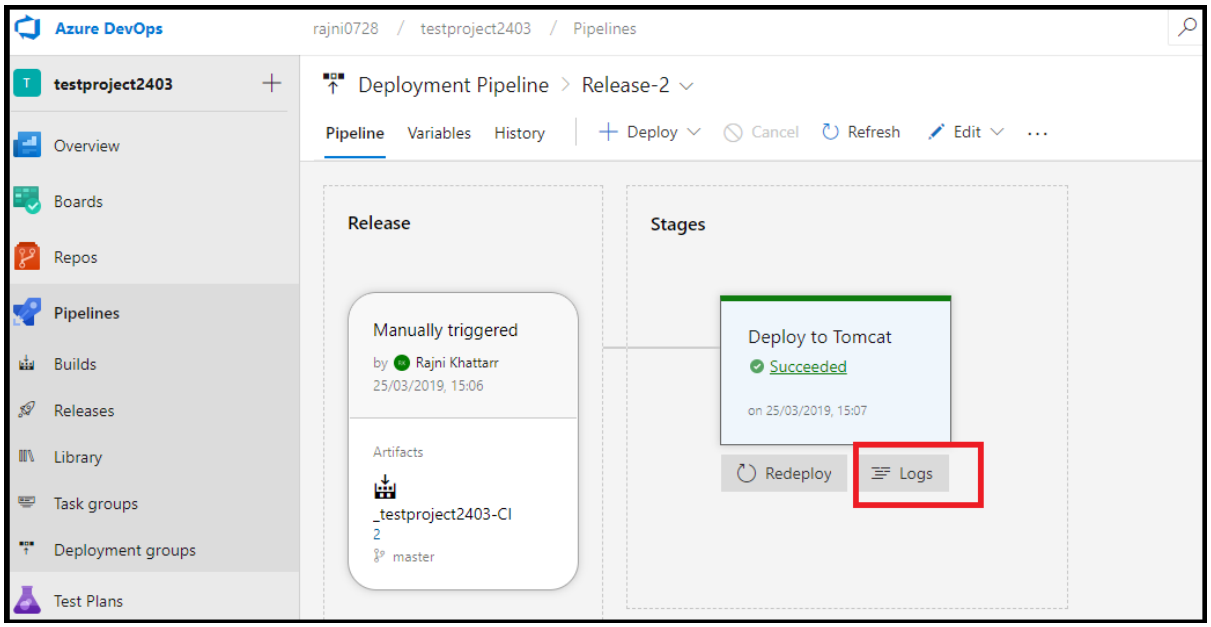
87.) View the triggered release



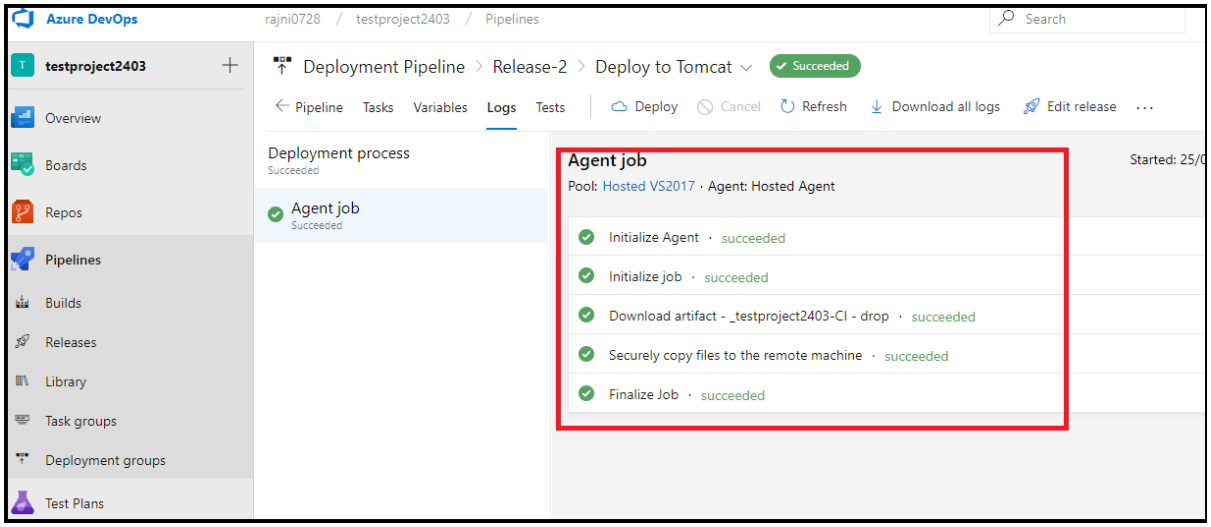
88.)Refresh the page and view the Release is successfully completed



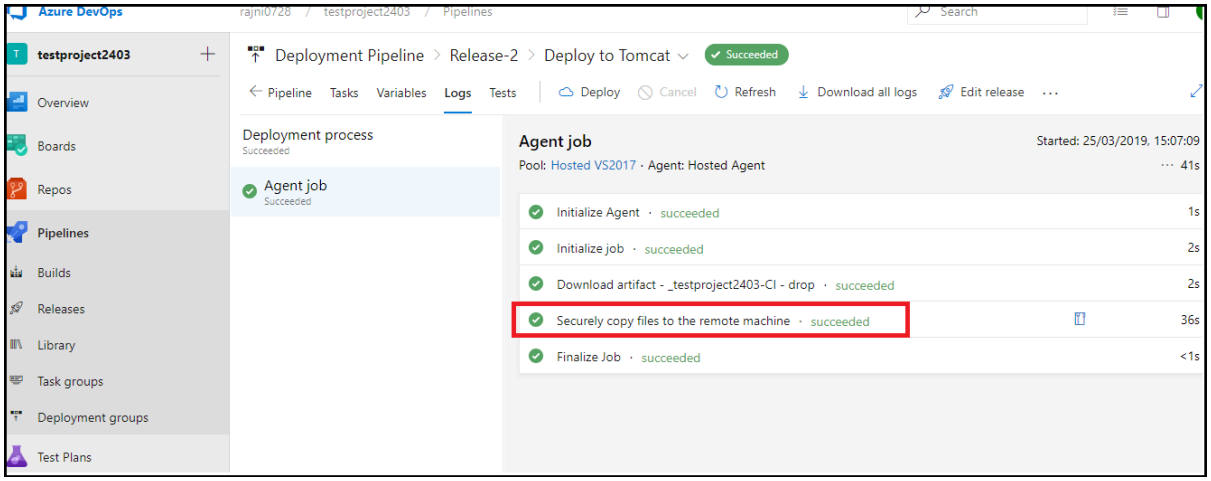
89.)Click on **Release-2** link to view the details.



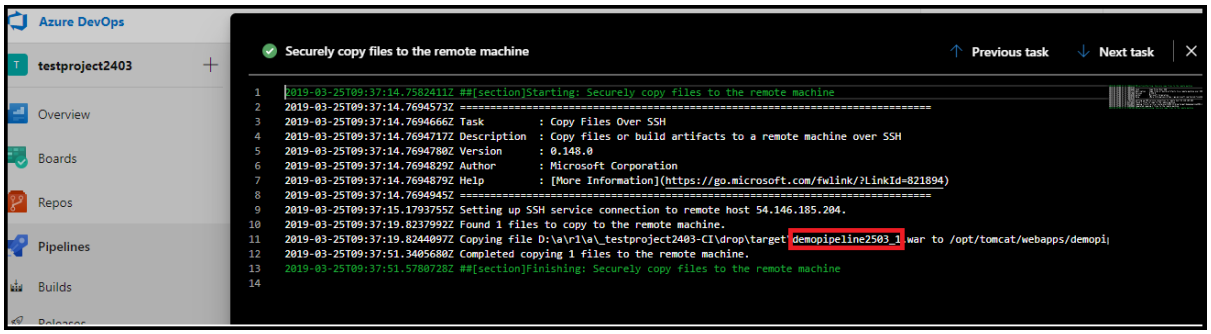
90.)View the logs



91.)Click on **Securely copy files to the remote machine**



92.)View the logs of **Securely copy files to the remote machine**



93.)Copy the name of the war file and Open the browser where the application is opened. Replace war file name with new one.