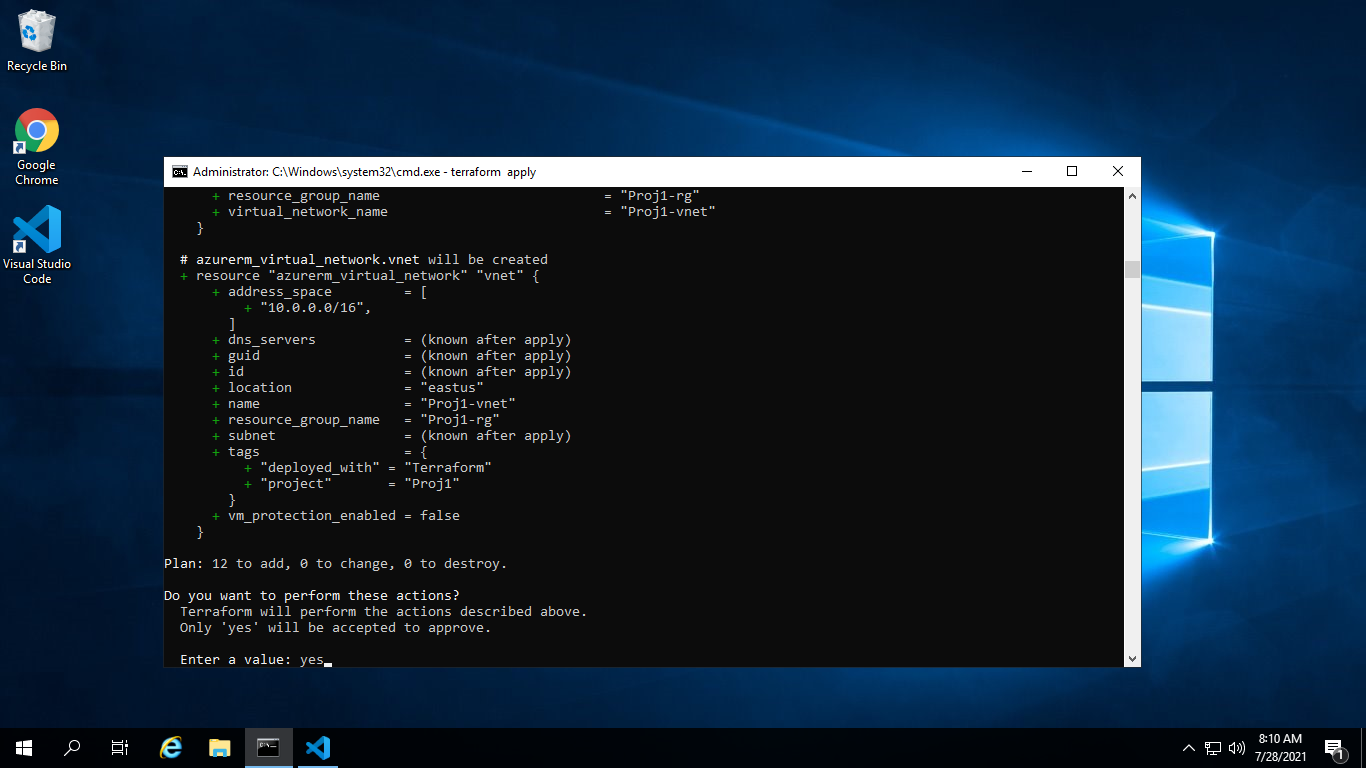
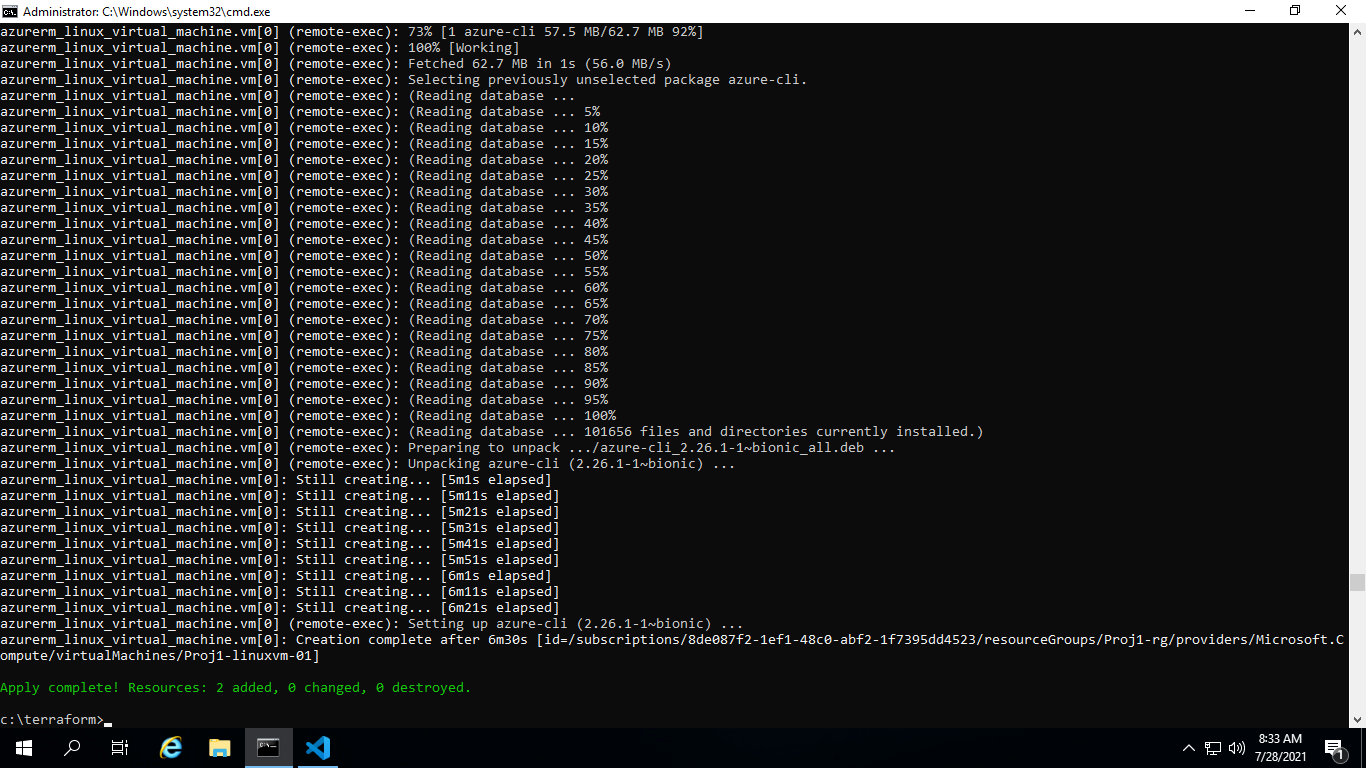
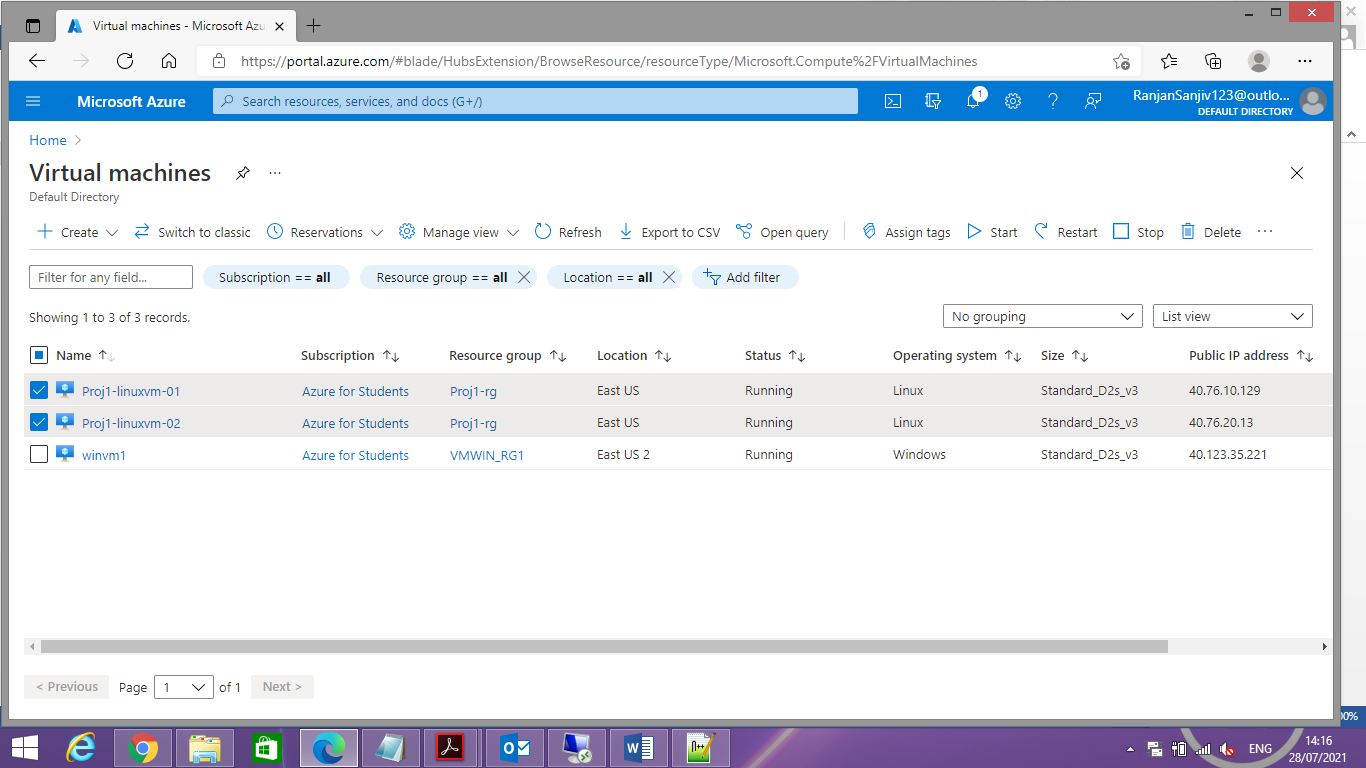
**Project 1 - Documentation**

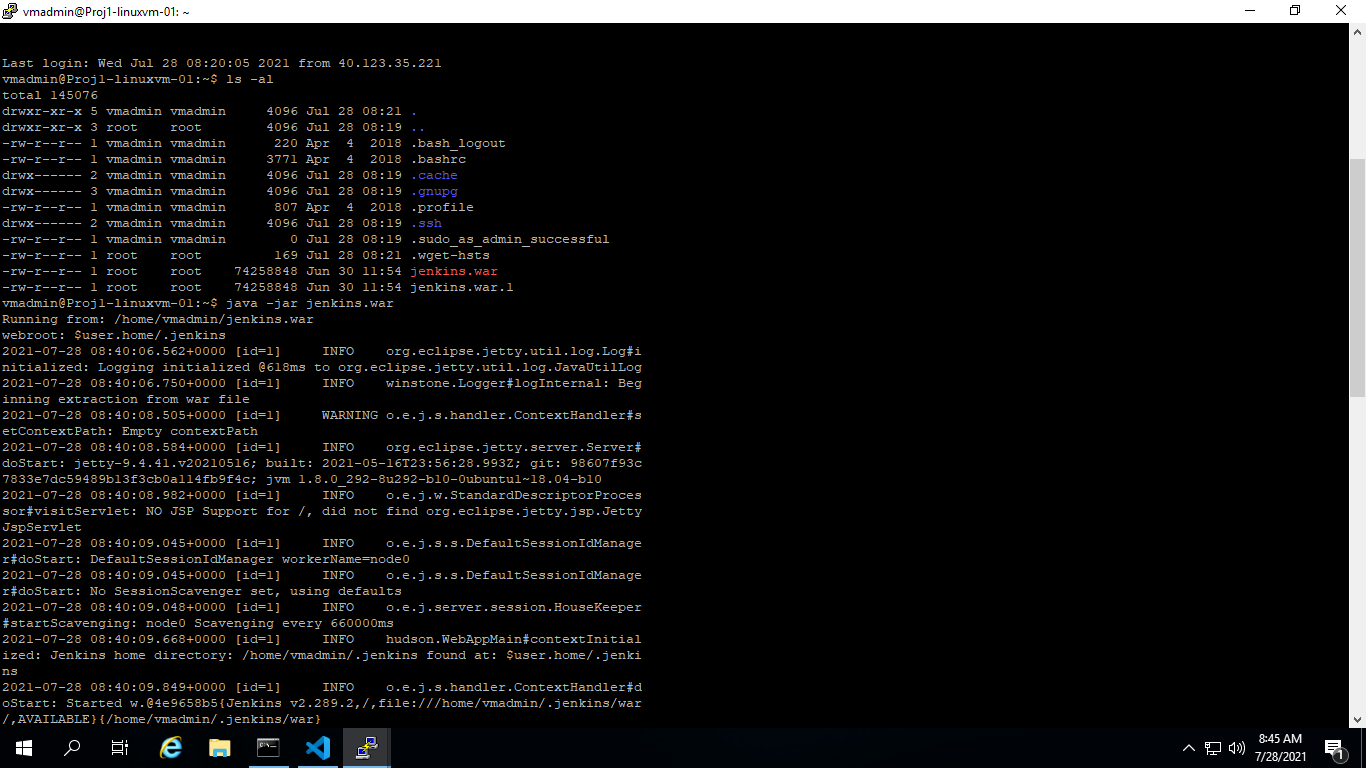
Phase 1 –   
  
1. Terraform scripts for 2VMs

Apply completed

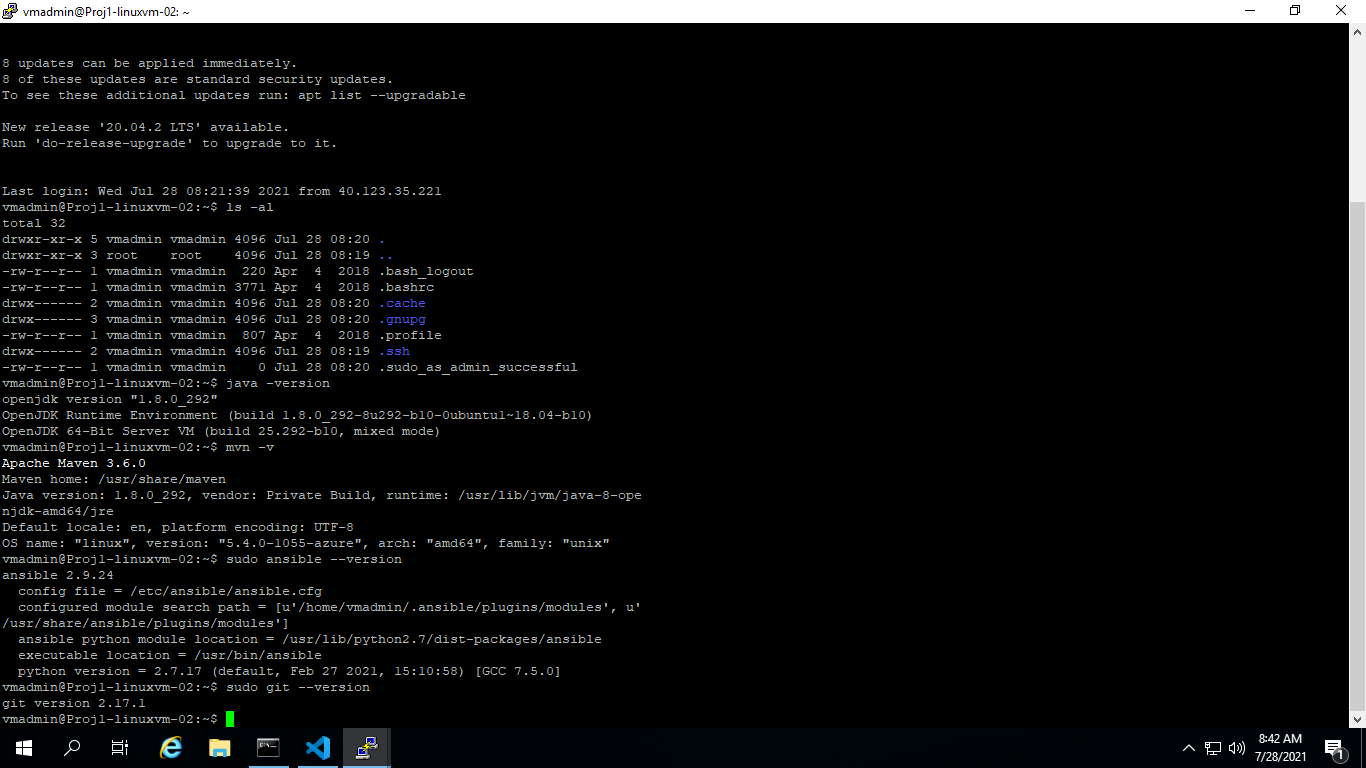


After successful Apply, 2 VMs got created and running on Azure portal:

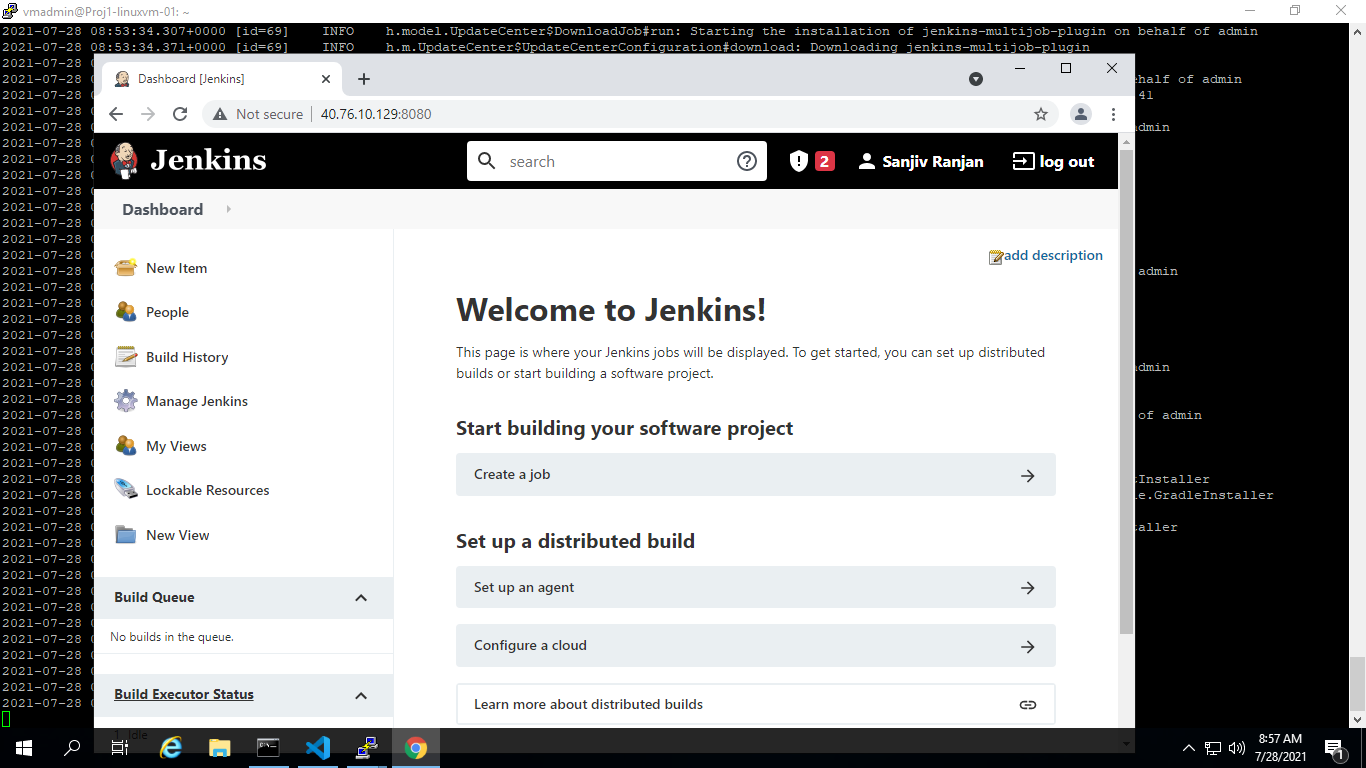


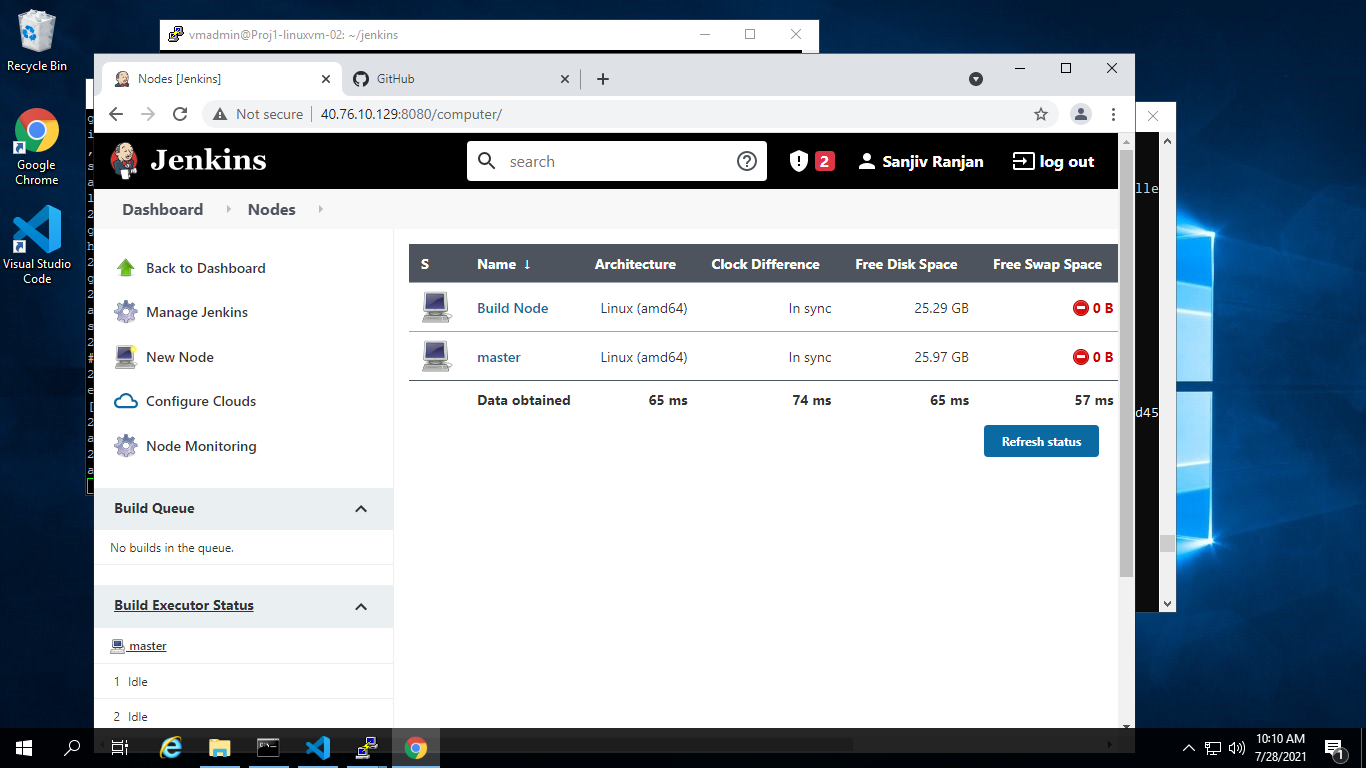
On Build node ([Proj1-linuxvm-02](https://portal.azure.com/#@RanjanSanjiv123outlook.onmicrosoft.com/resource/subscriptions/8de087f2-1ef1-48c0-abf2-1f7395dd4523/resourceGroups/Proj1-rg/providers/Microsoft.Compute/virtualMachines/Proj1-linuxvm-02))–  
- Java, Jenkins war got installed  
  


On master node ([Proj1-linuxvm-01](https://portal.azure.com/#@RanjanSanjiv123outlook.onmicrosoft.com/resource/subscriptions/8de087f2-1ef1-48c0-abf2-1f7395dd4523/resourceGroups/Proj1-rg/providers/Microsoft.Compute/virtualMachines/Proj1-linuxvm-01))



Jenkins and configure required Plug-ins and Master Slave Configuration:

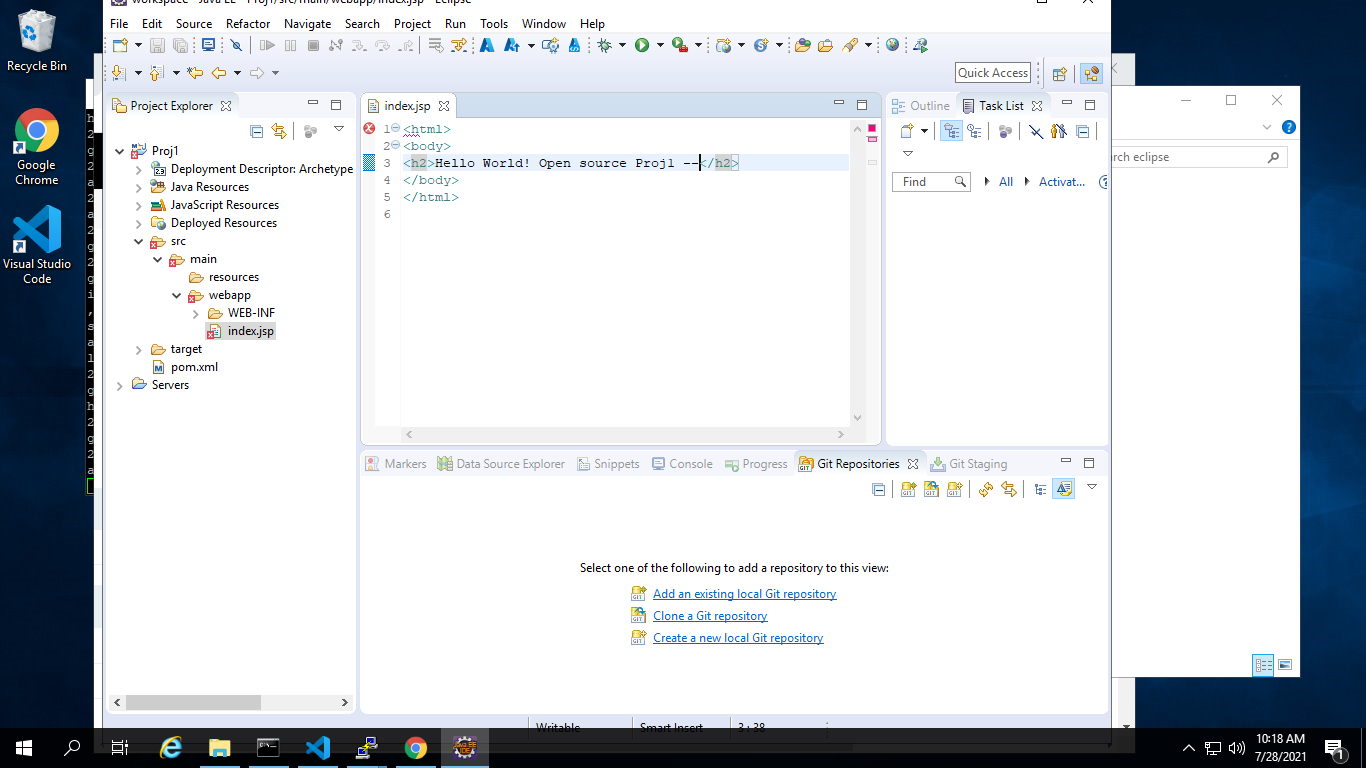




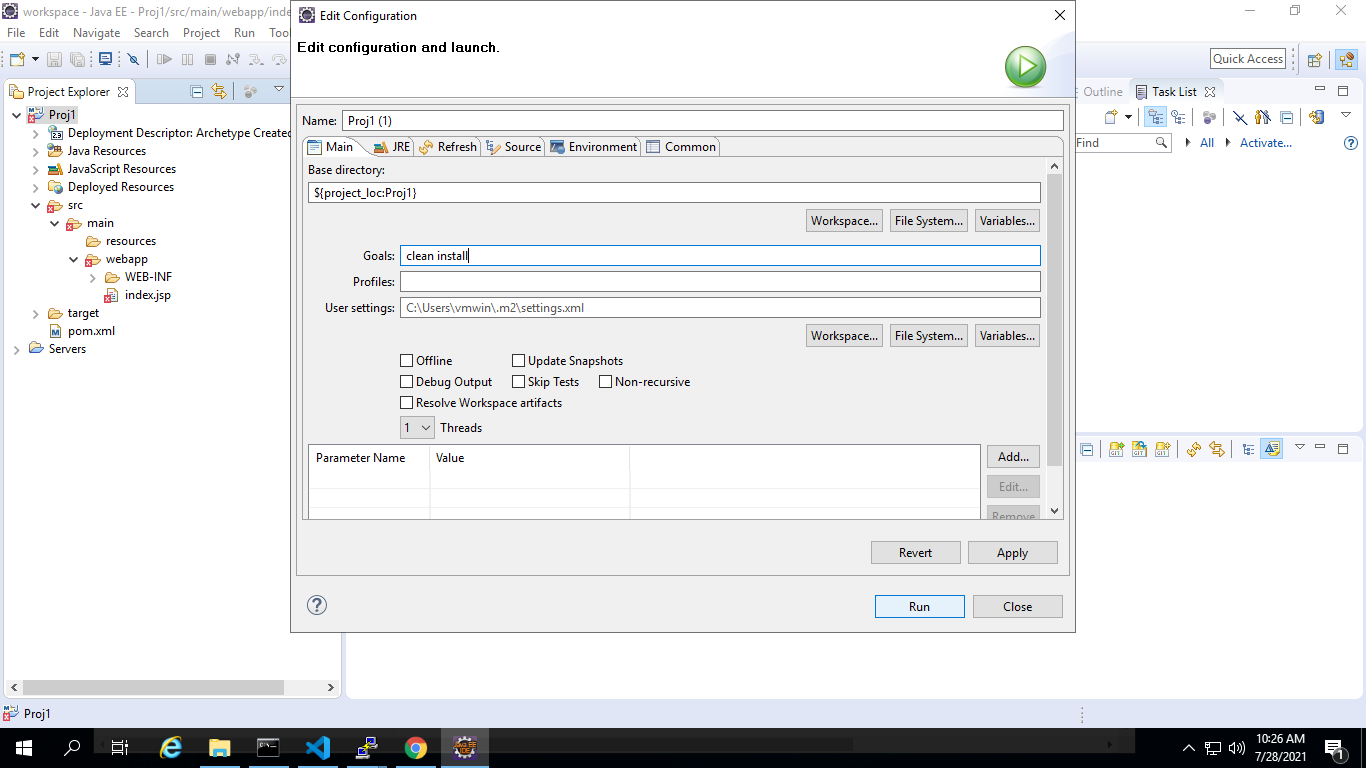
Part 2 –

**Phase 1 (– Eclipse)**

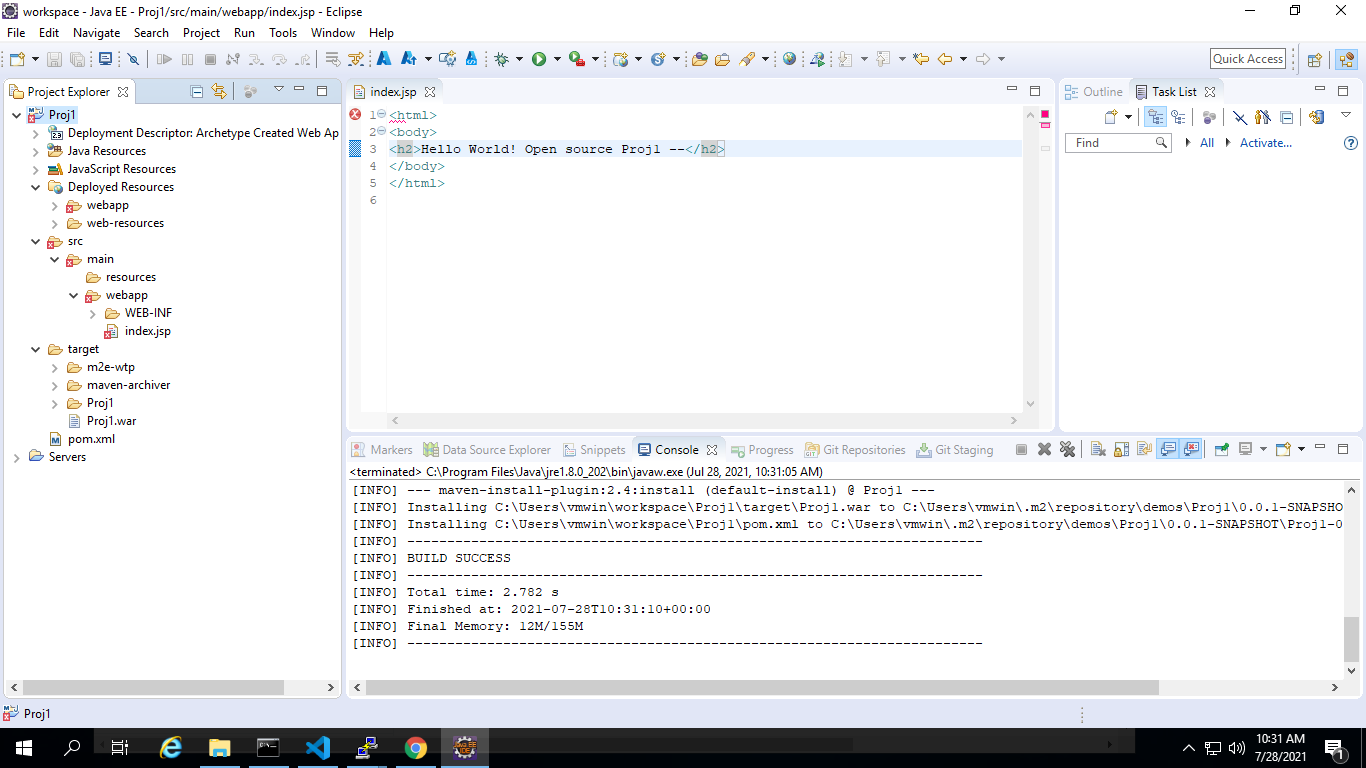
1. Create Maven Project with Archtype as web application in eclipse  
   Modify Index.jsp under src/main/webcontent to display a custom message



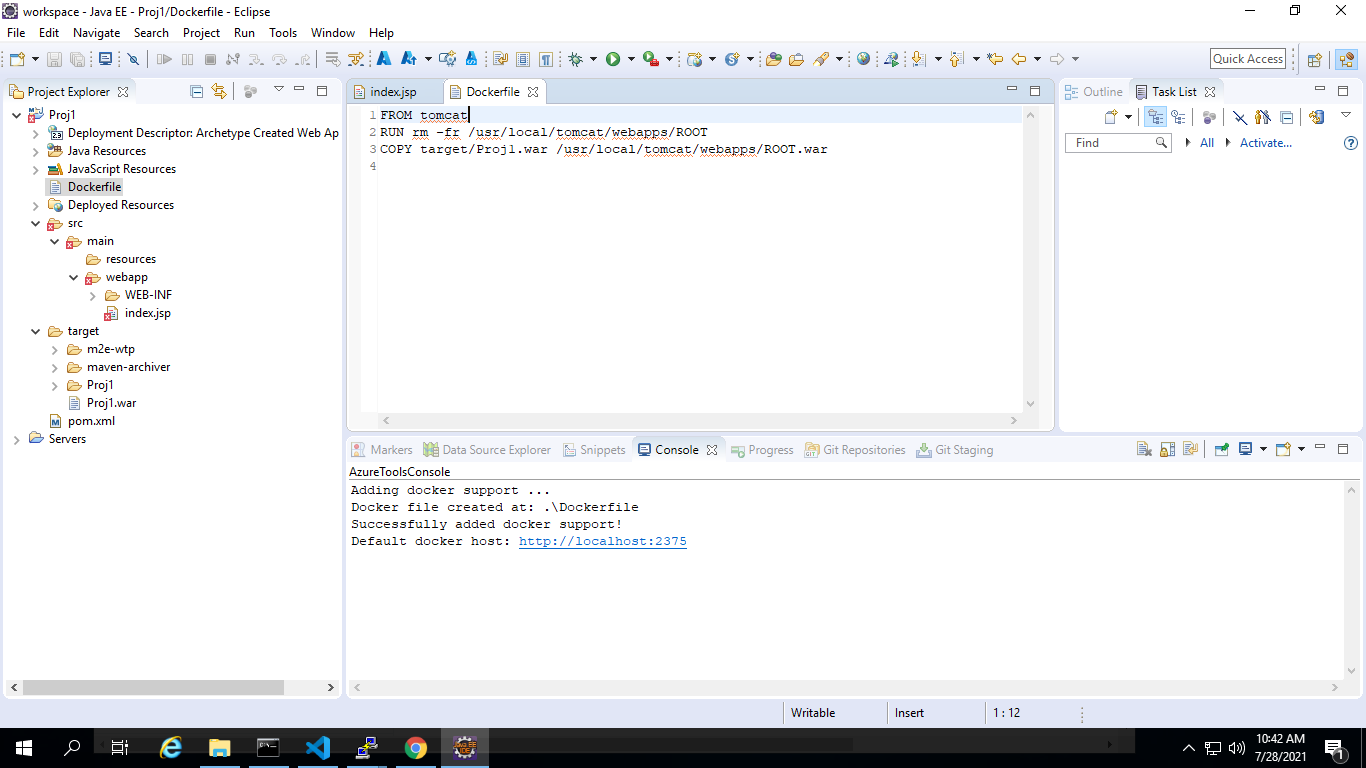
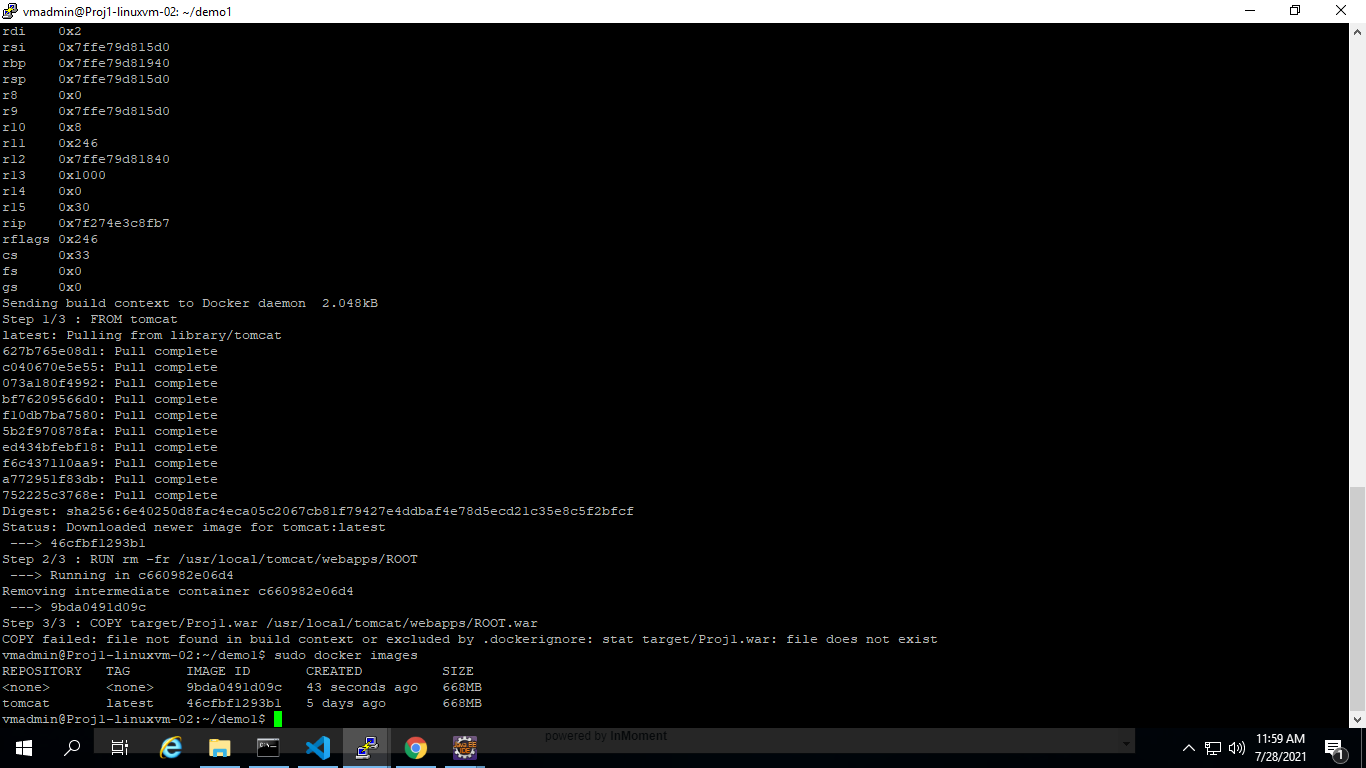
1. Run Maven clean install in eclipse to check the build and ckeck for .war file in target folder



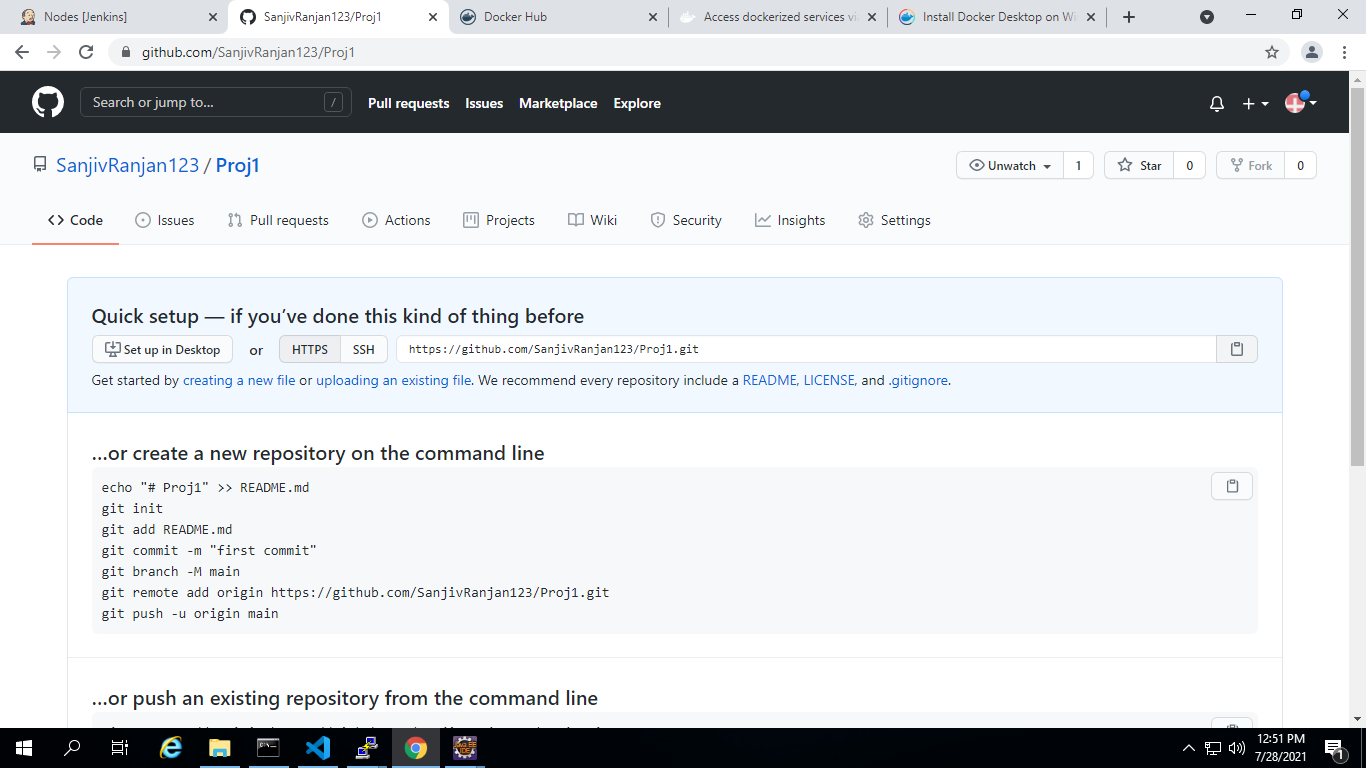
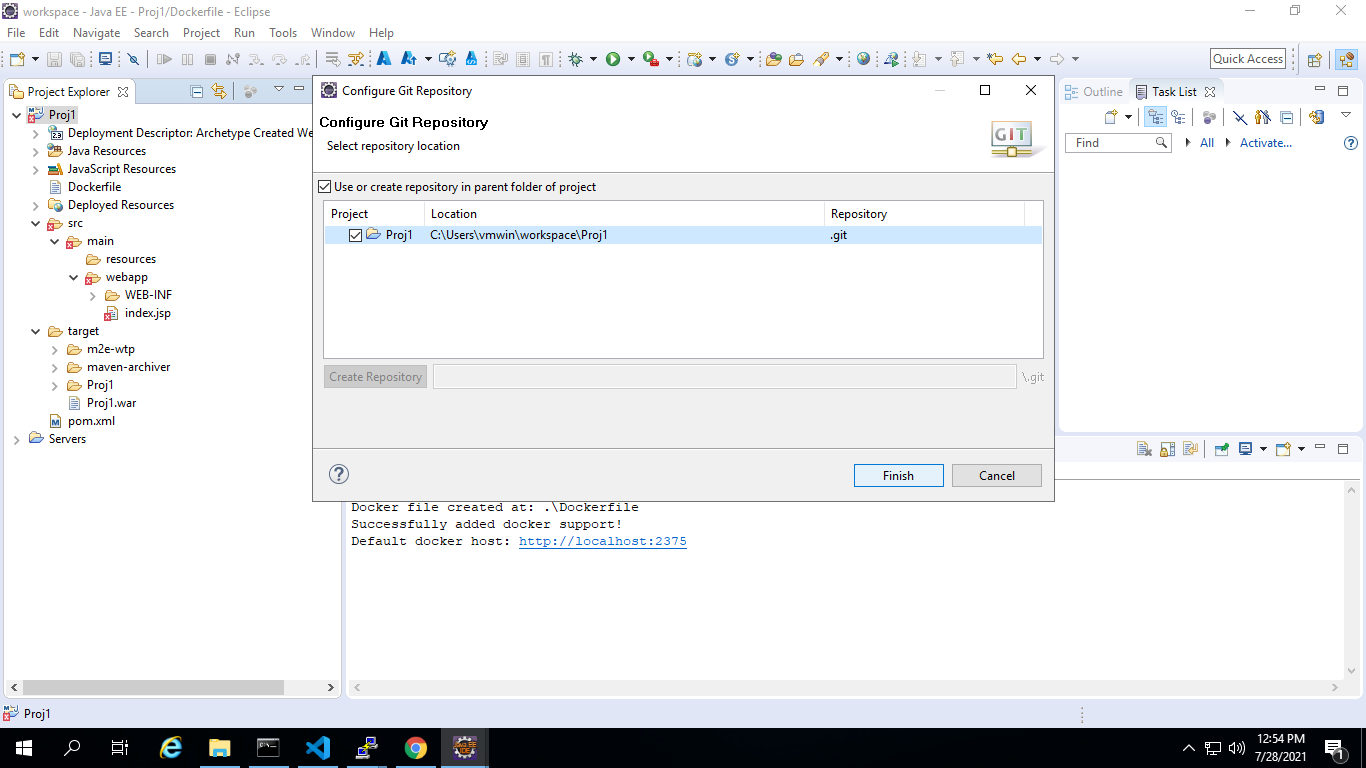
War file in Target folder



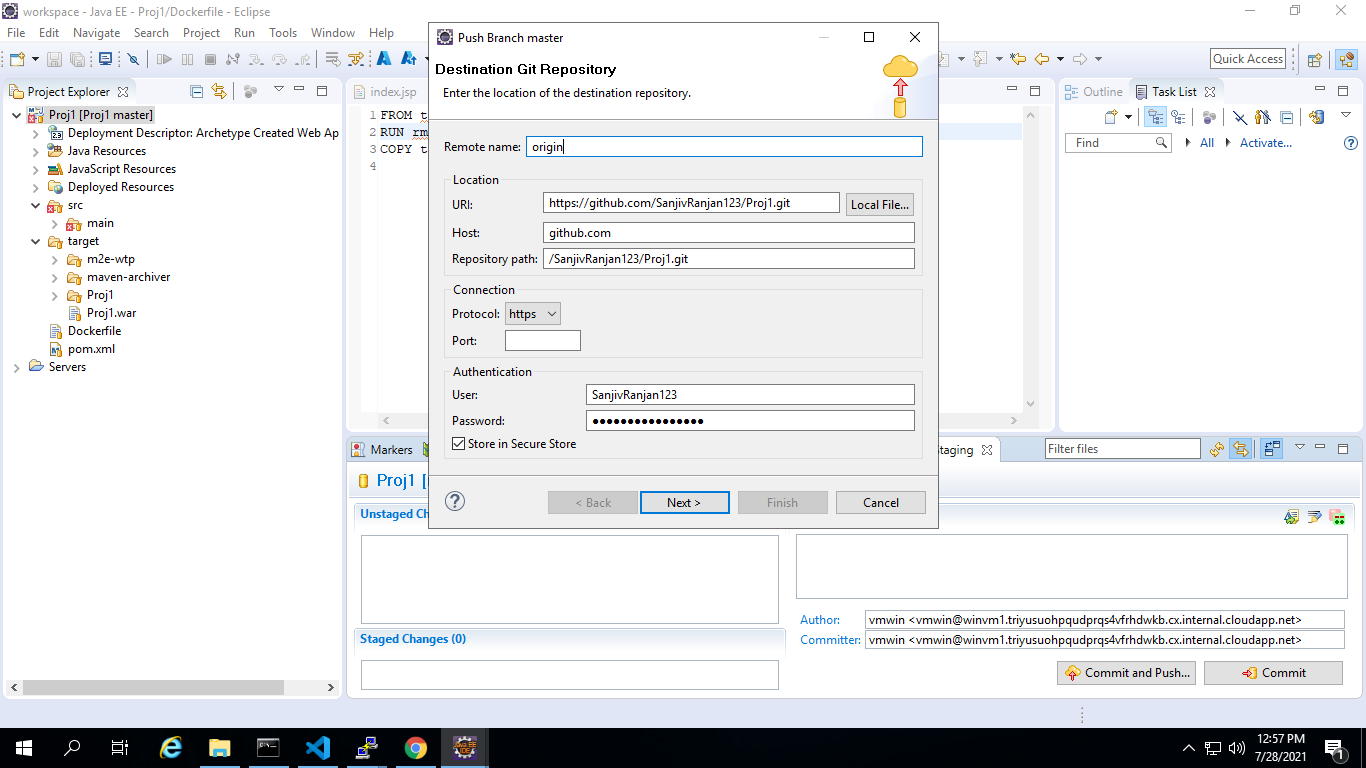
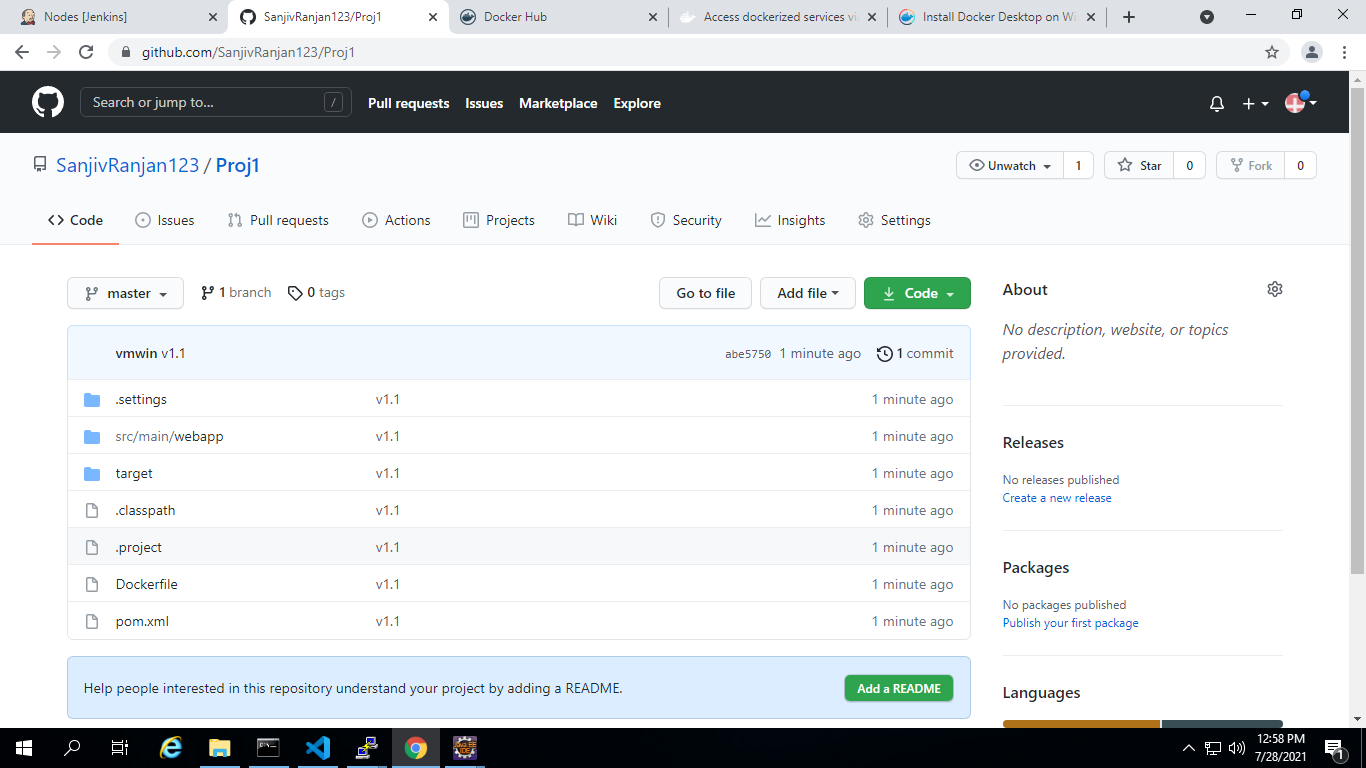
**Part 2 – Phase 2**

1. Gerate Dockerfile under project folder of your app – Eclipse  
   Modify FORM statement to use tomcat as base image – Eclipse  
     
   
2. Test the Dockerfile by running Docker build and create a container  
   

**Part 2- Phase 3**

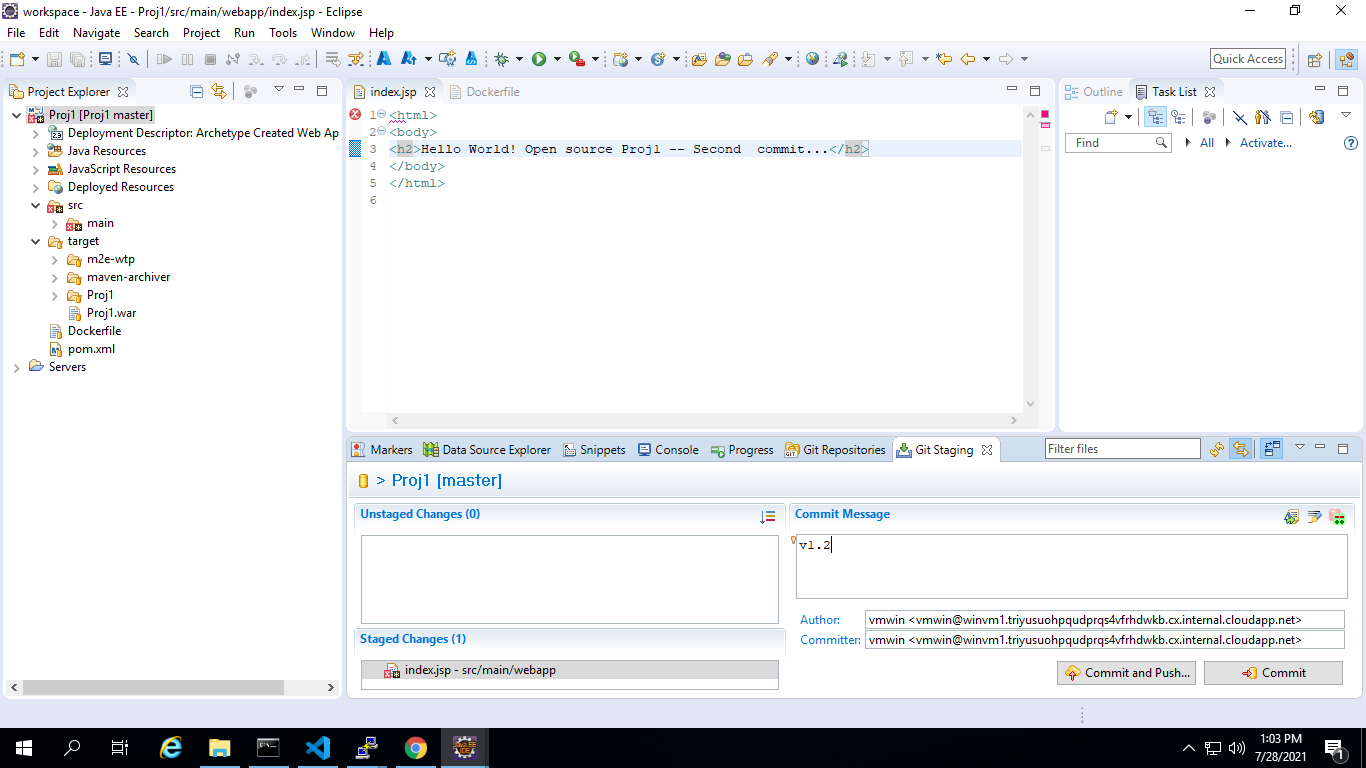
1. Create a github repository and copy repo URL -Github Site  
   
2. In Eclipse convert the app in to a local repo from Team meanu share Project Option – Eclipse  
   

1. Commit and Push the code to remote repo - Eclipse

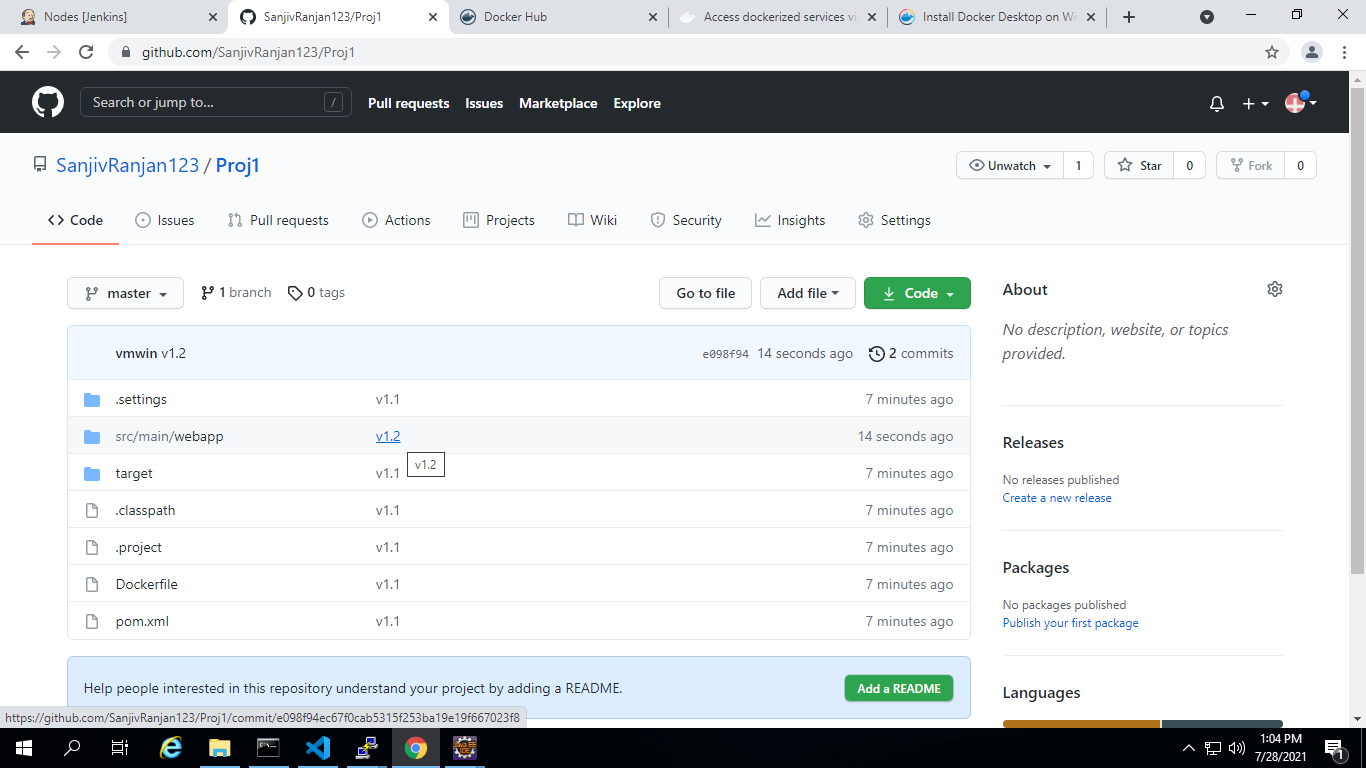
  
  
Refresh GITHub to see the push:  
  


**Part 2- Phase 4**

Modify Project index.jsp, Commit and Push to remote repo - Eclipse

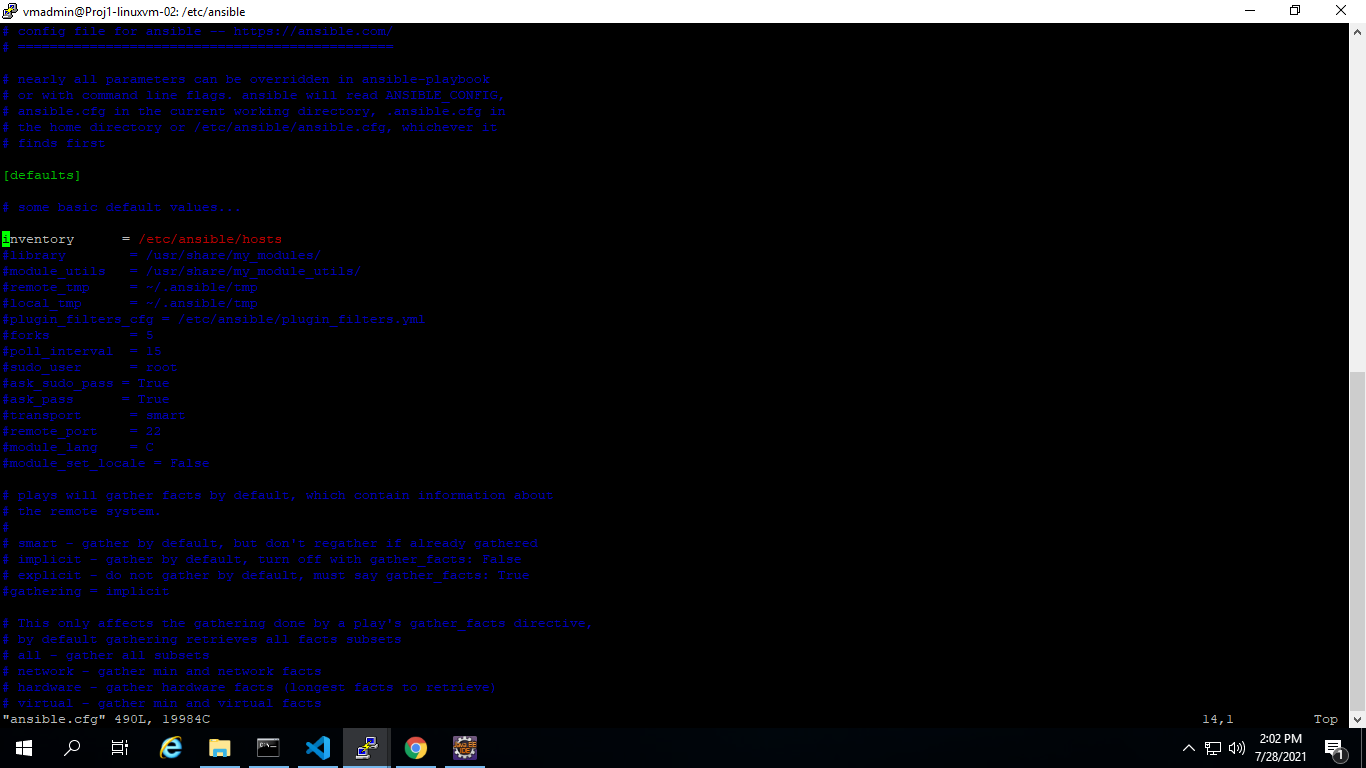


Check for the change in remote Repo - Github Site

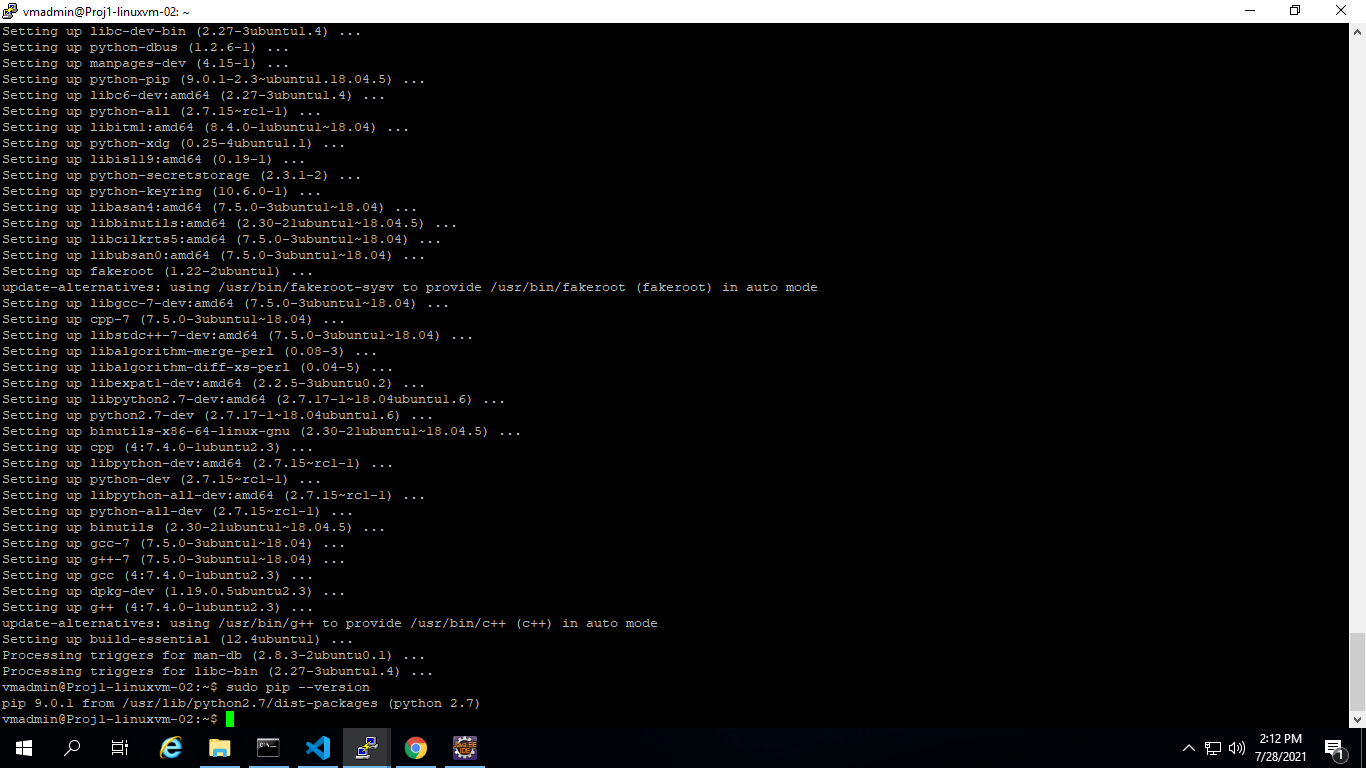


**Part 2- Phase 5**

1. In build server configure Ansible manually - Shell
2. Modify ansible.cfg to use hosts file as inventory - Shell

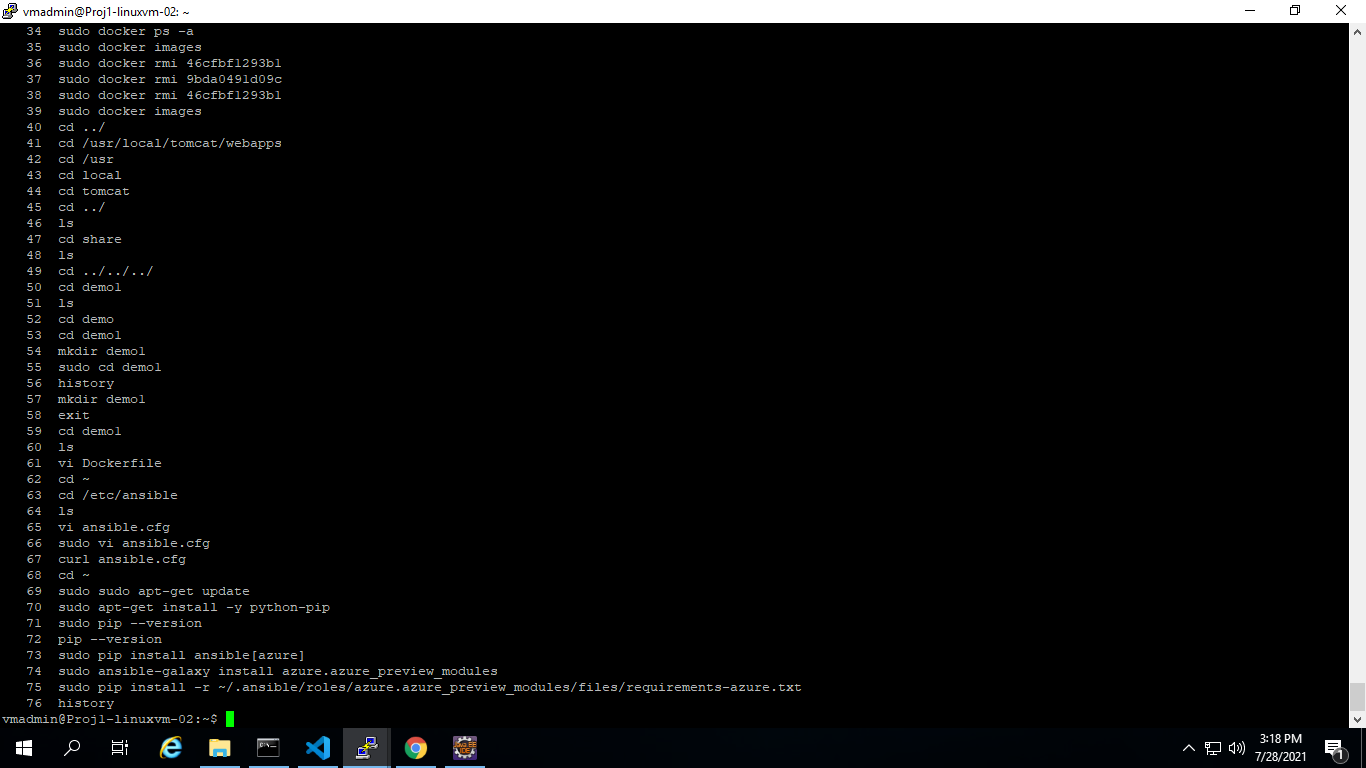


1. Install python-pip in ansible server - apt-get:  
     
   $ sudo apt-get install -y python-pip  
   $ pip --version



1. Using PIP install azure modules in Ansible server – PIP:  
     
   $sudo pip install ansible[azure]

$sudo ansible-galaxy install azure.azure\_preview\_modules

$sudo pip install -r ~/.ansible/roles/azure.azure\_preview\_modules/files/requirements-azure.txt  


1. Use the same service Principle created for terraform for ansible to get authenticated to Azure

mkdir ~/.azure

vi ~/.azure/credentials

and add as below:

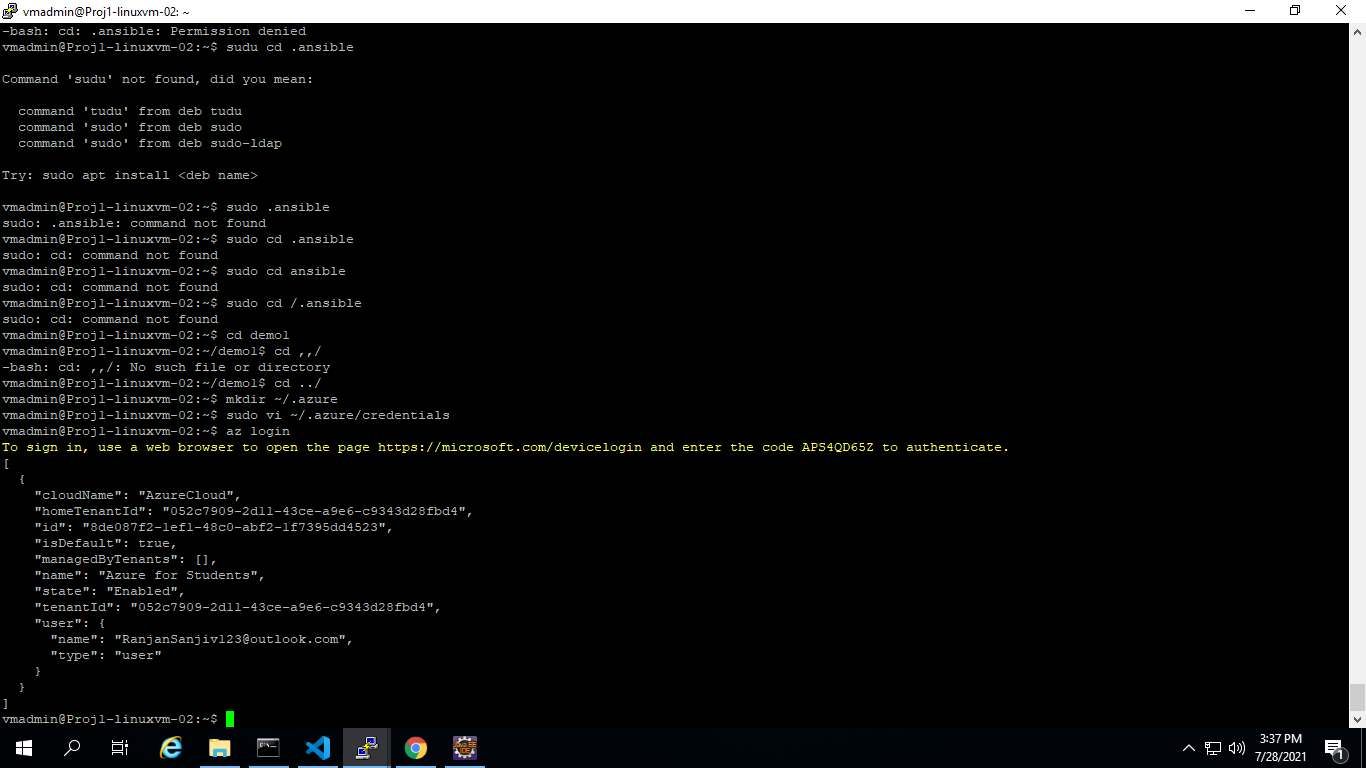
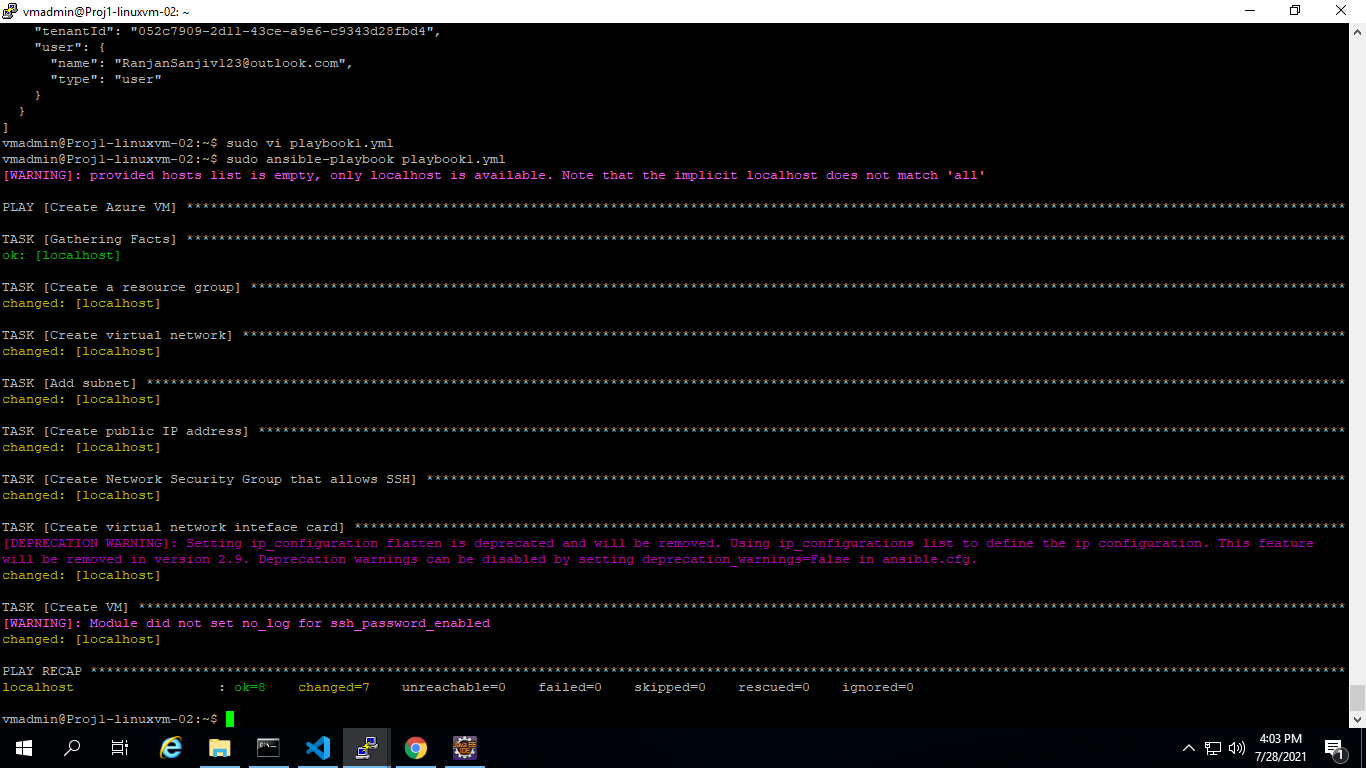
[default]

subscription\_id=<your-subscription\_id>

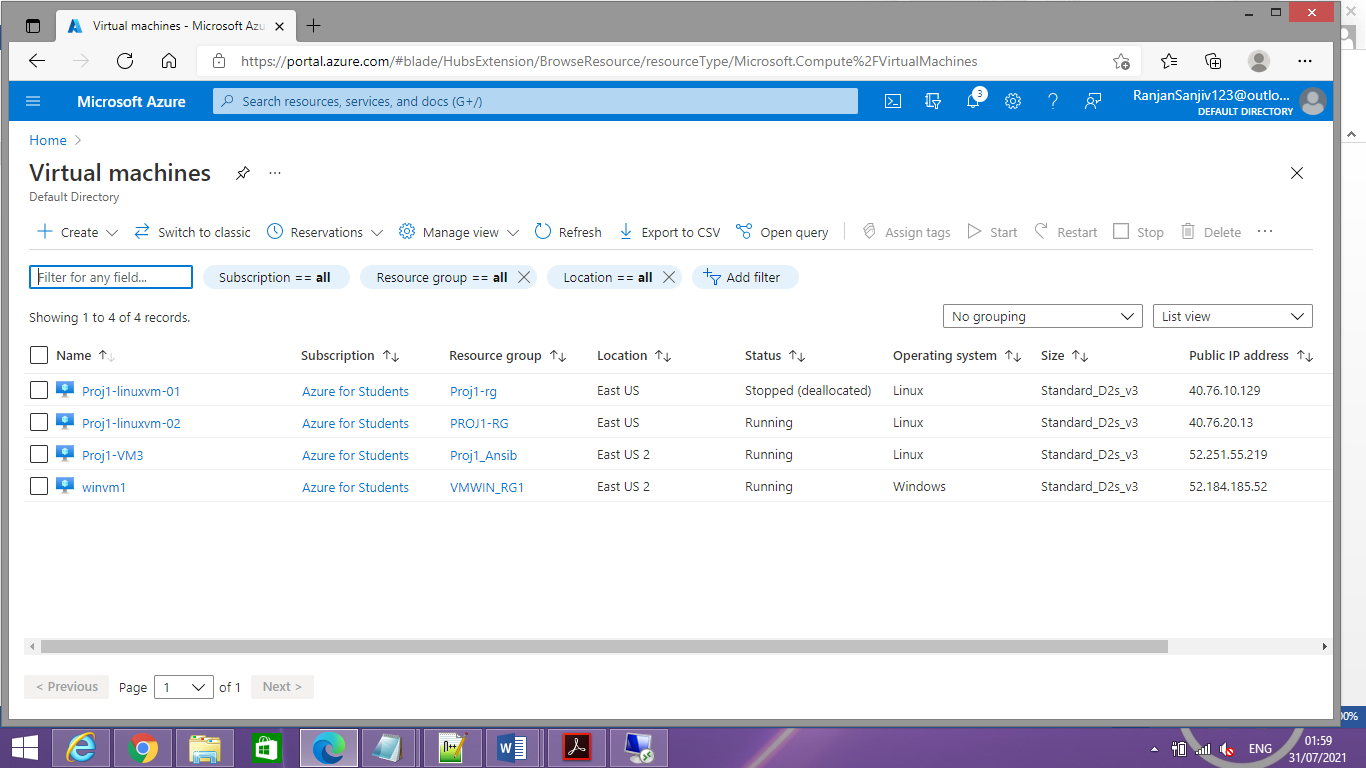
client\_id=<security-principal-appid>

secret=<security-principal-password>

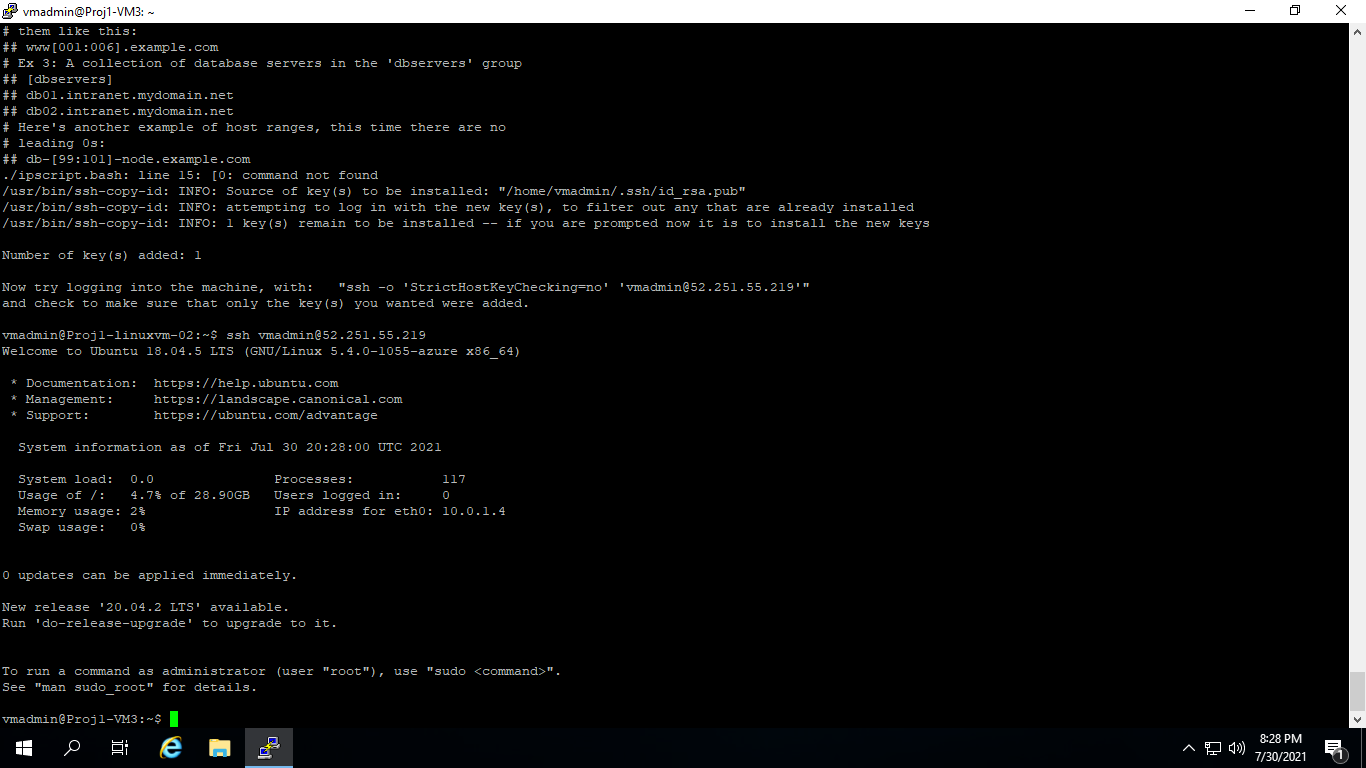
tenant=<security-principal-tenant>

  
  
6) Create a playbook1.yml to create a vm in azure - vi  
  


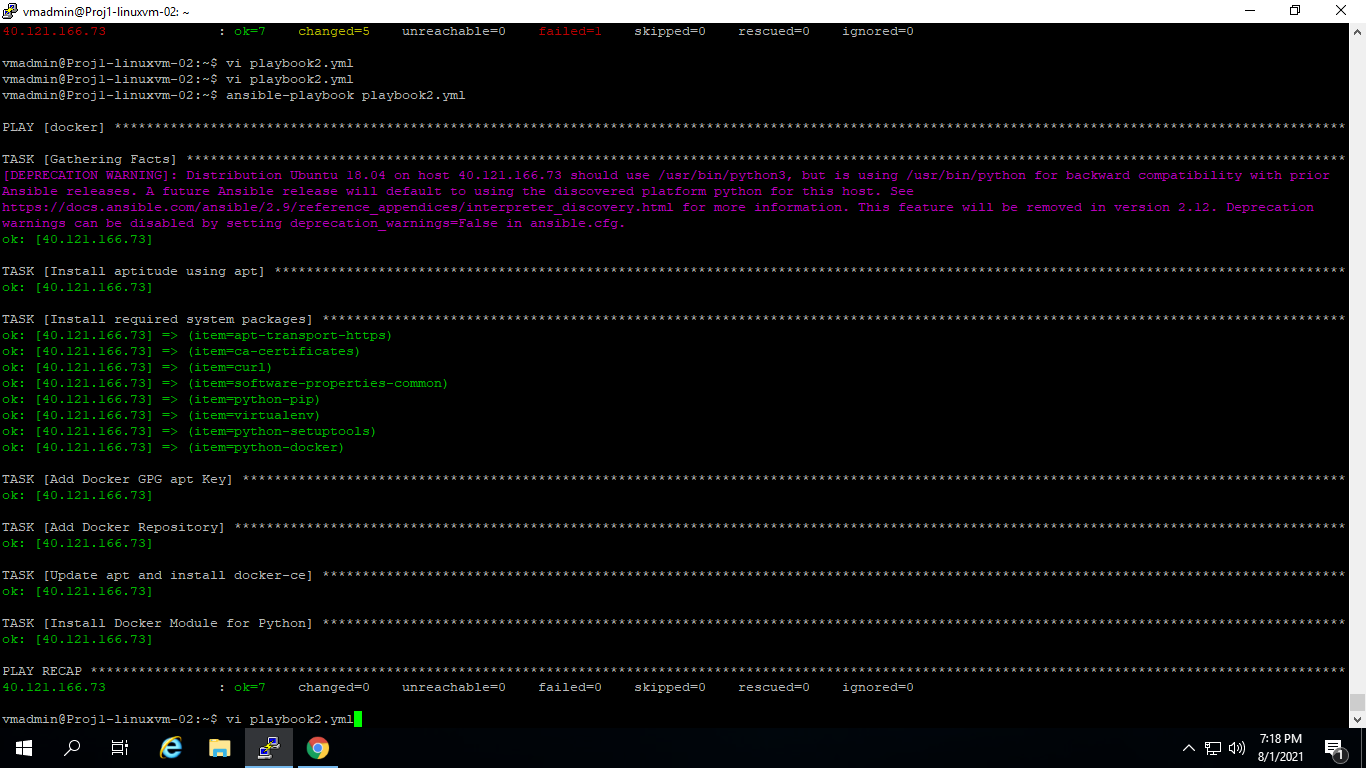
Portal view:



Scipt execution, key generation and ssh connection with passwor



From Build server run the playbook1, playbook2 and shell script to test for the required result



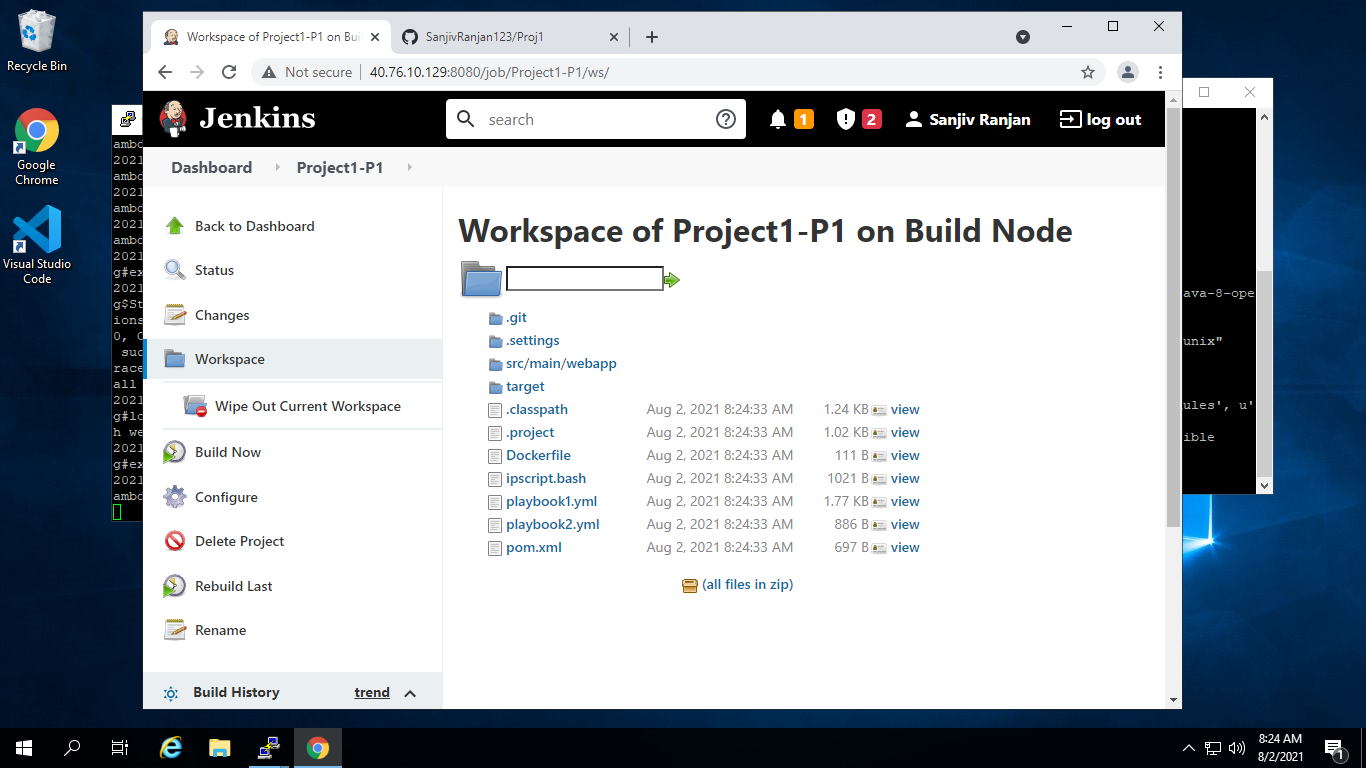
Pushed file to git hub



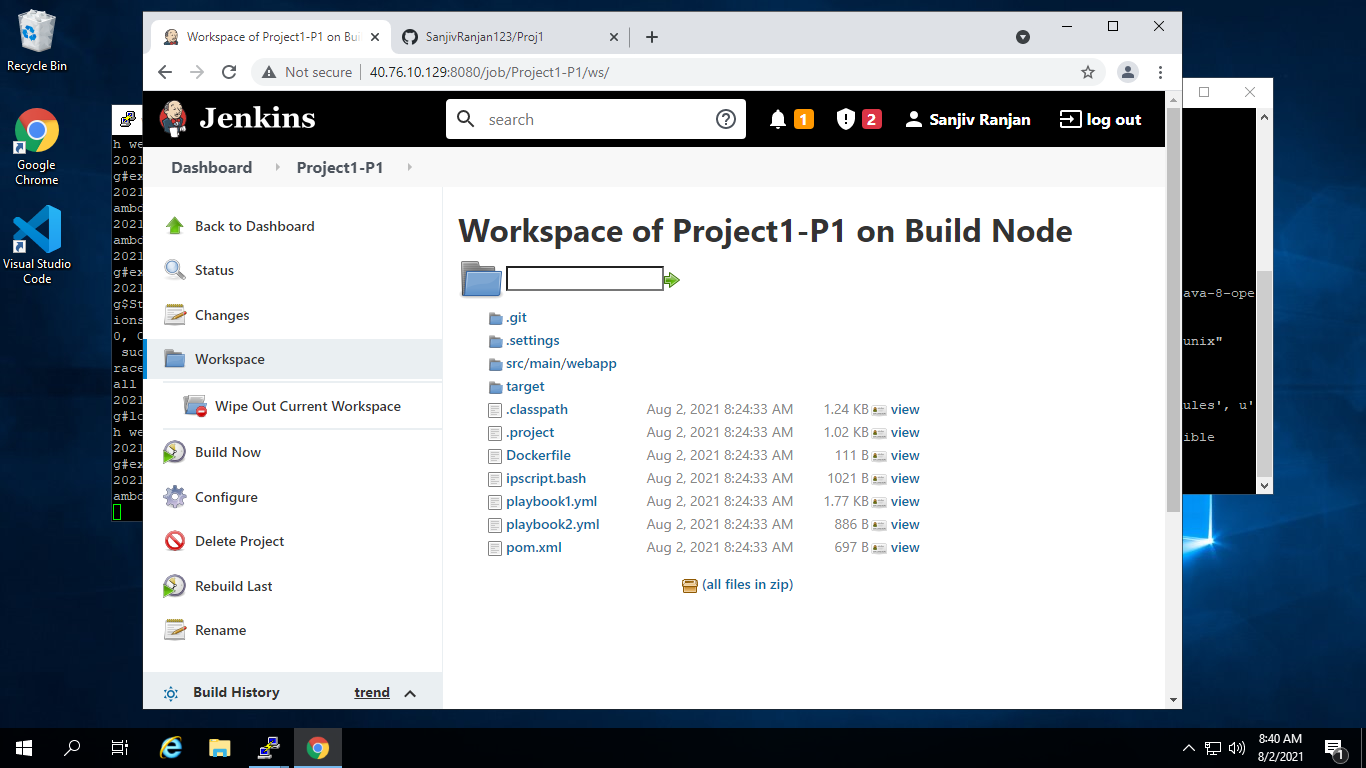
Configure Git credentials in Jenkins Vault

Create Pipeline1 using Freestyle project in Jenkins

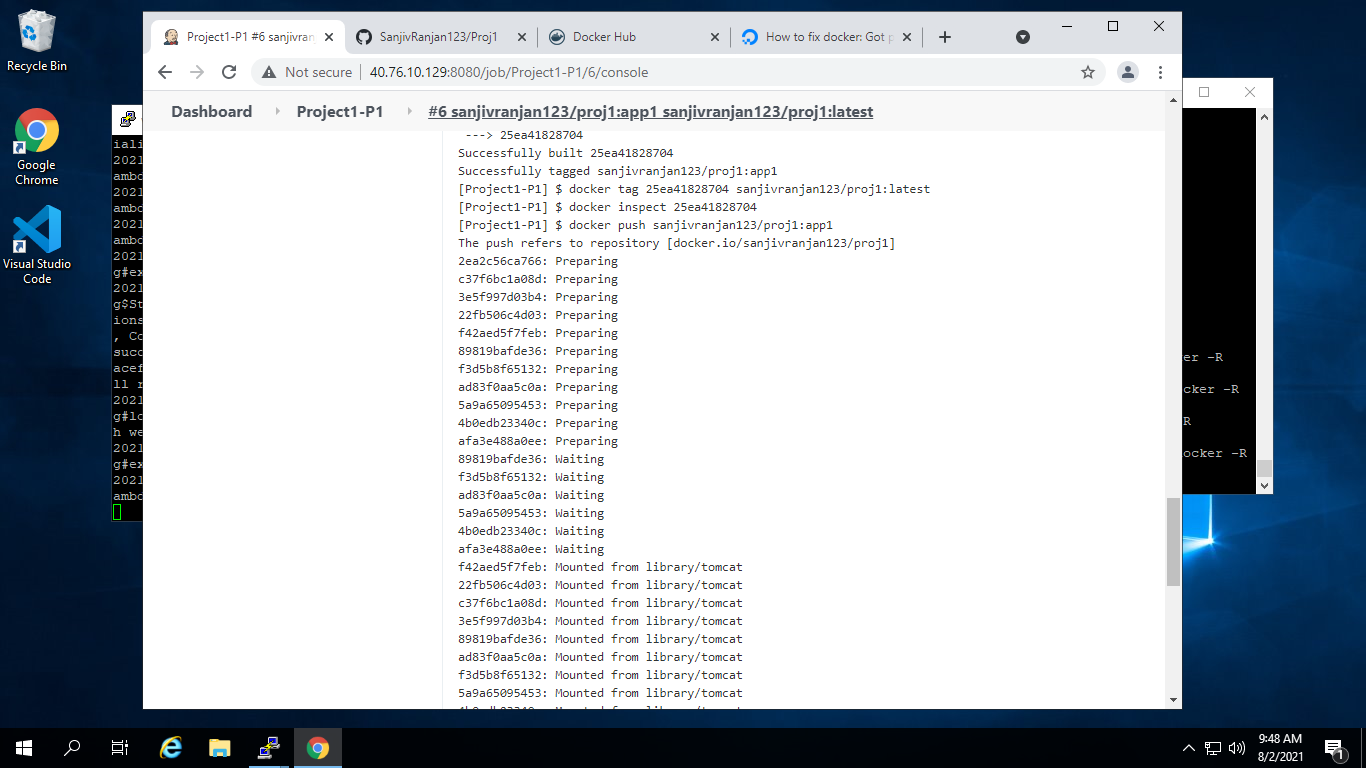
In SCM stage Pull code form Remote Repo

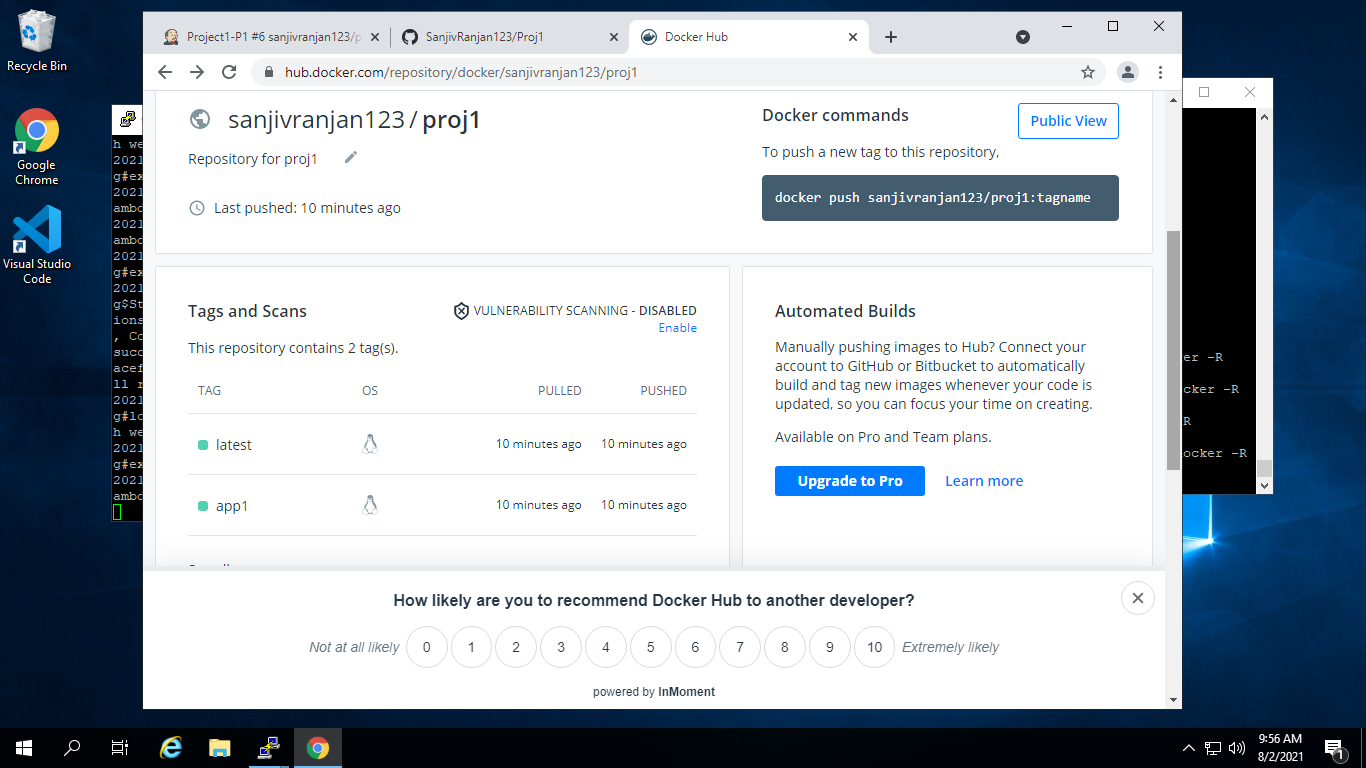


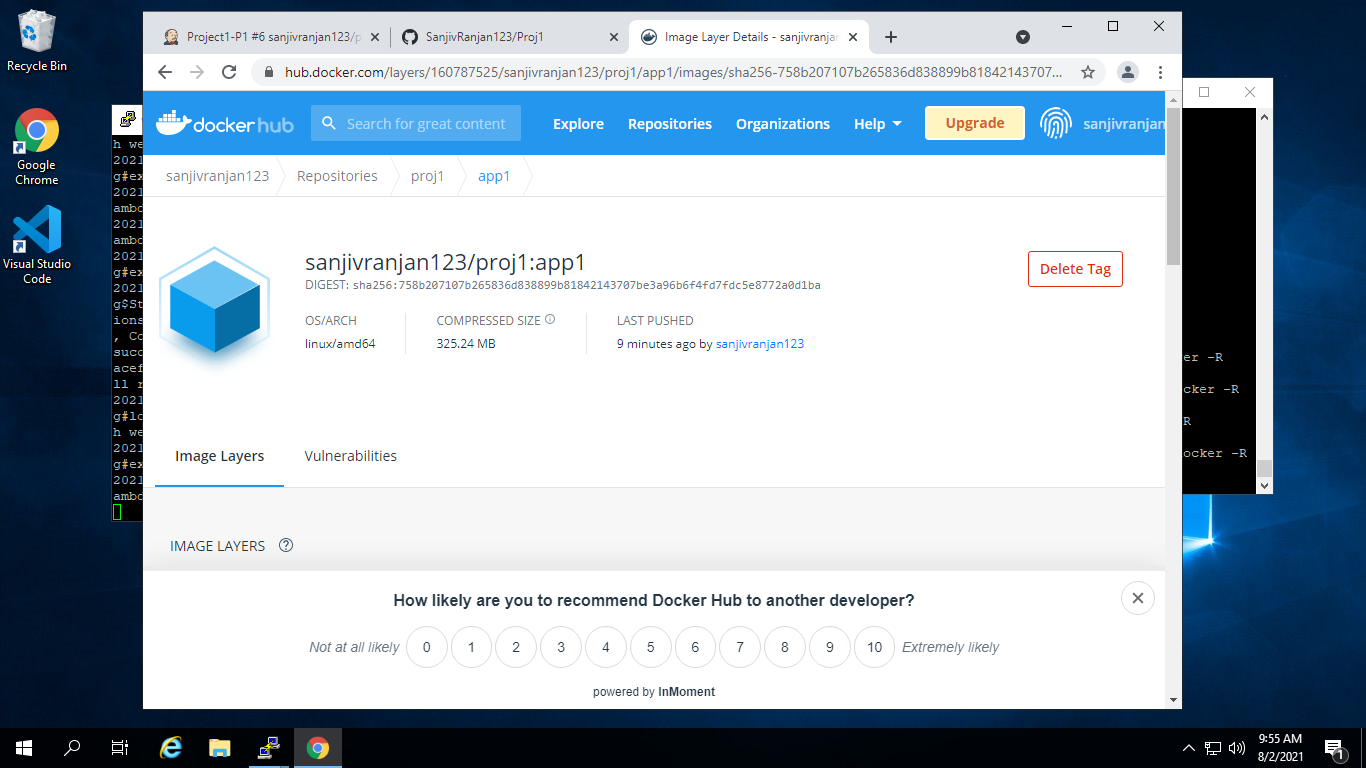
In Build Stage, Step1 : use maven top level target to build



In Build Stage Step 2: User Docker build and Push to create image which contains your app and push to Docker Hub





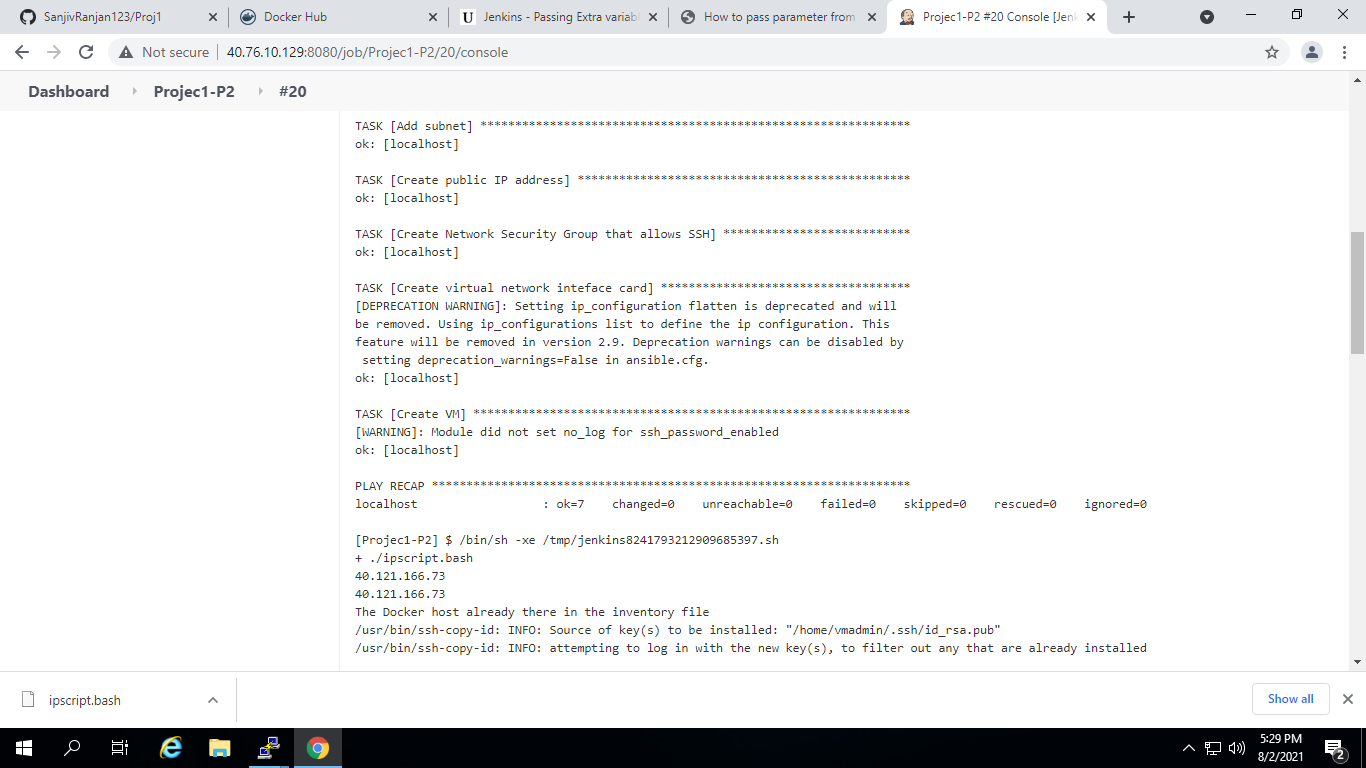


Create Pipeline2 using Freestyle project in Jenkins

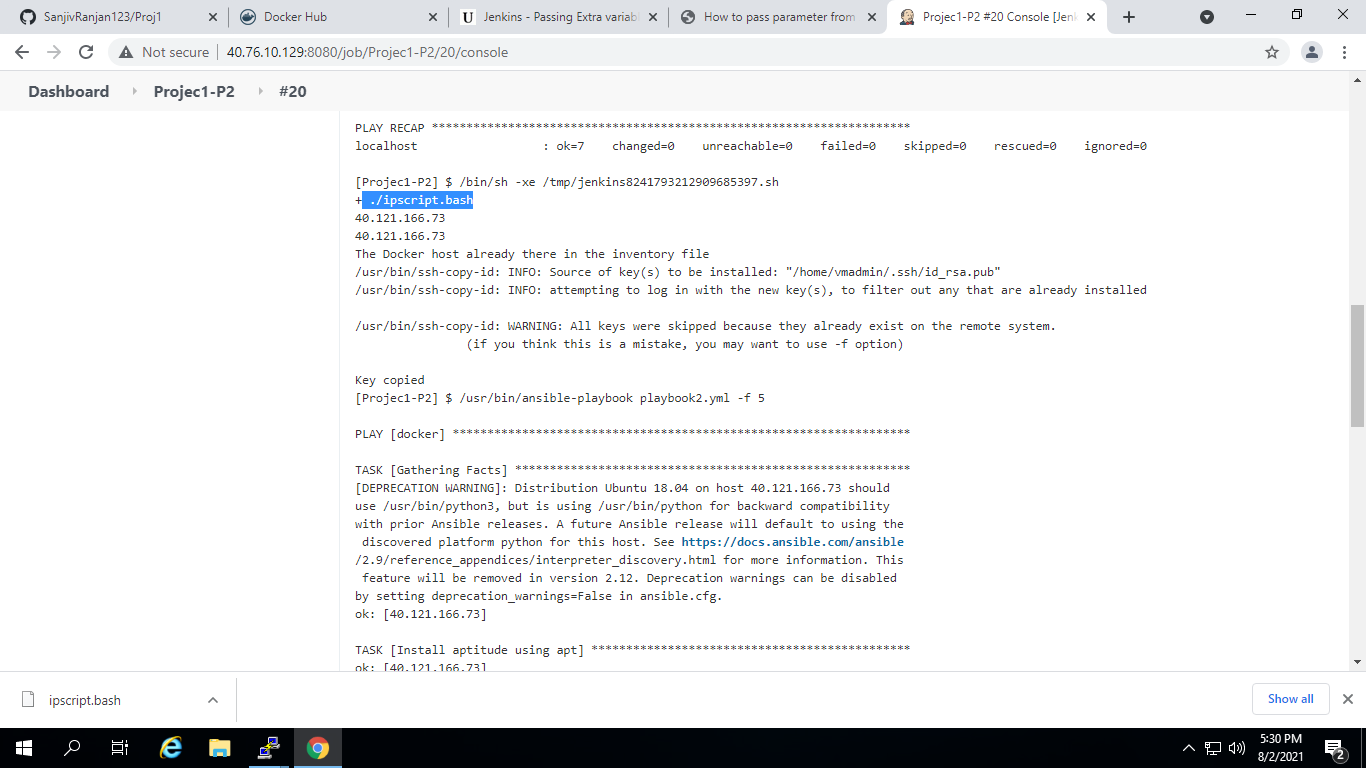
In SCM stage pull code form Remote Repo

Pipeline 2 will be downstream pipeline of Pipeline 1

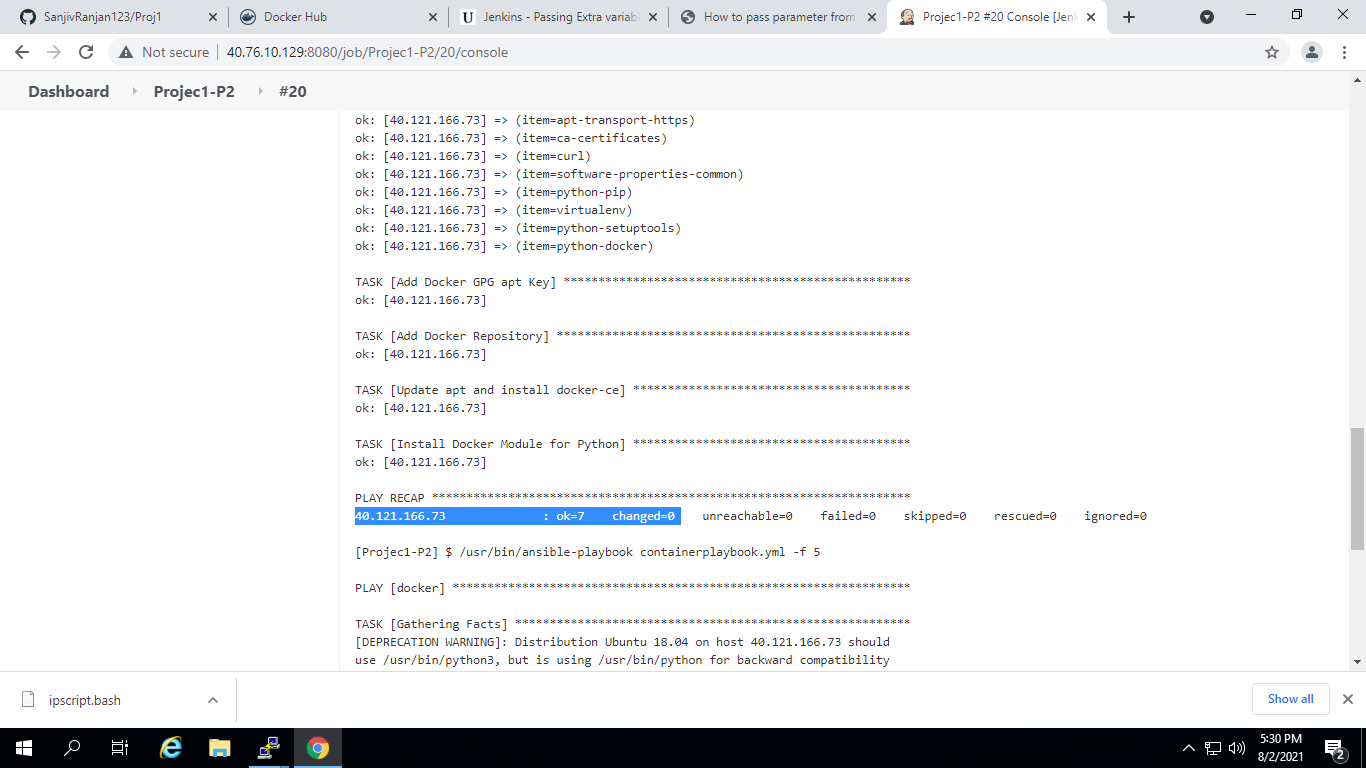
In Build Stage Step 1: Run ansible Playbook1



In Build Stage Step 2: Call Shell Script



In Build Stage Step 3: Run Ansible Playbook2



In Post Build Stage: Deploy Docker Container on Docker VM using image created in Docker HUB



