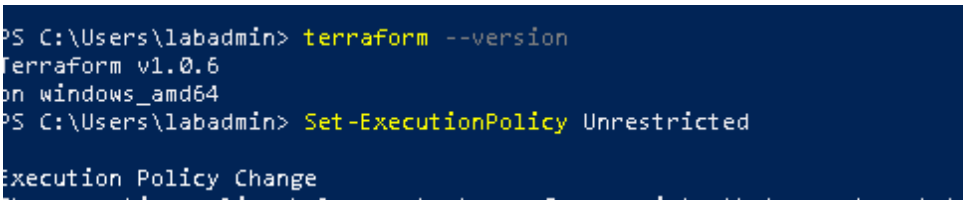
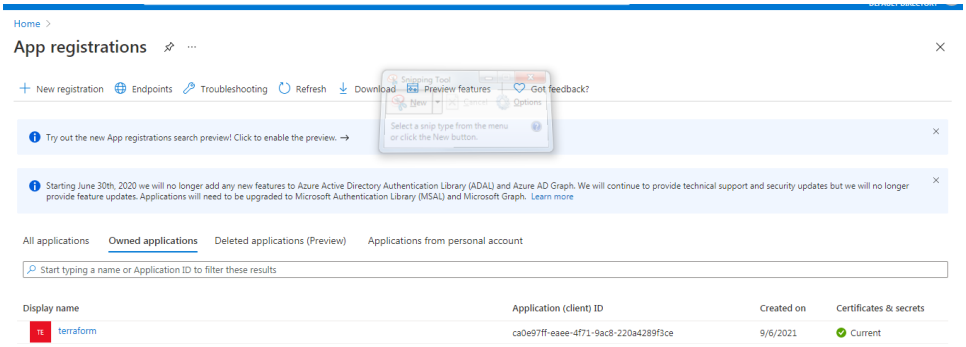
Project 1:

Task1:



Terraform Version.

Generate Service Principle for Terraform in Azure:



Create required Vars.tf to create two VMs in Azure:

C: > project1 > variable.tf

```
#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*

# Service Principal Variables


# Prefix and Tags

variable "prefix" {
    description =   "Prefix to append to all resource names"
    type        =   string
    default     =   "jenkins"
}

variable "tags" {
    description =   "Resouce tags"
    type        =   map(string)
    default     =   {
        "project"       =   "jenkins"
        "deployed_with" =   "Terraform"
    }
}

# Resource Group

variable "location" {
    description =   "Location of the resource group"
    type        =   string
    default     =   "West US"
}

# Vnet and Subnet

variable "vnet_address_range" {
```

Newone
Newone

Create main.tf to create two Linux VMs:

C: > project1 > linuxvm.tf

```
    source_image_reference {
        publisher                    =   var.publisher
        offer                        =   var.offer
        sku                          =   var.sku
        version                      =   var.vm_image_version
    }

    connection {
    type = "ssh"
    host = "${azurerm_public_ip.pip.1.ip_address}"
    user = "vmadmin"
    password = "Newone@13579"
    }



   provisioner "file" {
     source      = "slave.sh"
     destination = "/home/vmadmin/slave.sh"
    }

    provisioner "remote-exec" {
     inline = [
      "sudo apt-get update -y" ,
      "sudo apt-get install dos2unix",
      "dos2unix slave.sh",
      "chmod +x /home/vmadmin/slave.sh",
      "/home/vmadmin/slave.sh ",
     ]
    }
```

Use Terraform Provisioner to install JDK and Jenkins in VM1:

C: > project1 > master.sh

```
#!/bin/sh
sudo apt-get update -y
sudo apt-get install openjdk-8-jdk openjdk-8-jre -y
sudo chmod 777 /etc/environment
sudo echo 'JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64' >> /etc/environment
sudo echo 'JRE_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' >> /etc/environment
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins -y
```

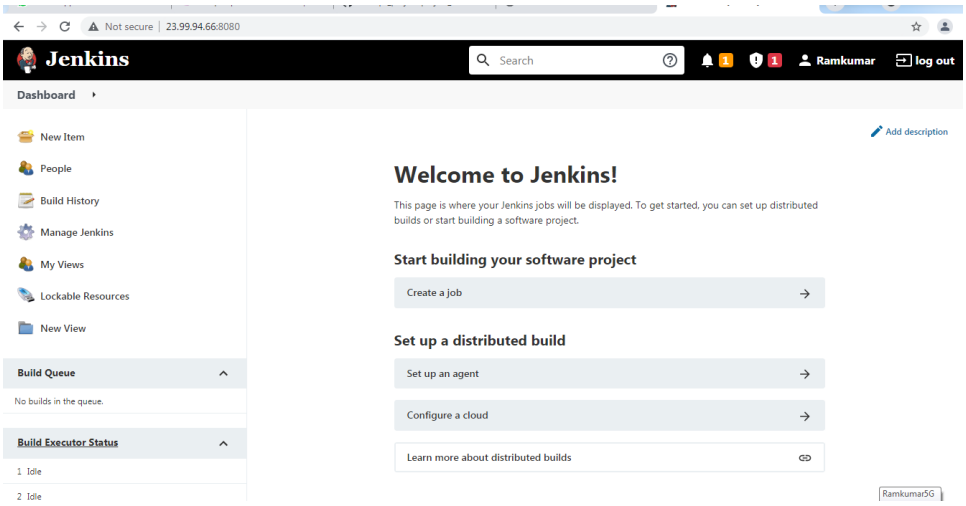
Use Terraform Provisioner to install JDK, Maven,Ansible,Docker,AzureCli and Git:

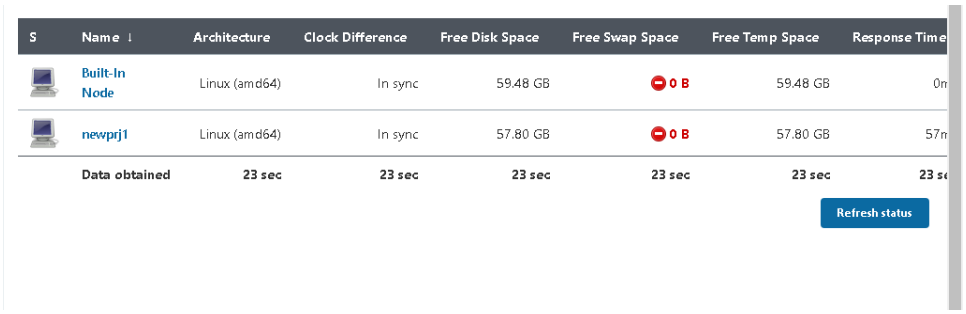
C: > project1 > slave.sh

```
#!/bin/sh
sudo apt-get update -y
sudo apt-get install openjdk-8-jdk openjdk-8-jre -y
sudo chmod 777 /etc/environment
sudo echo 'JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64' >> /etc/environment
sudo echo 'JRE_HOME=/usr/lib/jvm/java-8-openjdk-amd64/jre' >> /etc/environment
sudo apt-get install -y docker*
sudo service docker start
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
sudo apt install python3-pip -y
sudo apt-get install software-properties-common -y
sudo apt-add-repository ppa:ansible/ansible -y
sudo apt-get update -y
sudo apt-get install ansible -y
sudo pip install ansible[azure]
sudo apt install maven -y
```

Init,Plan and Apply Terraform Script:

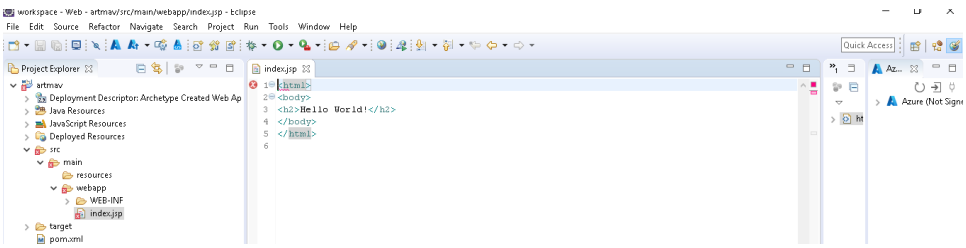
Manually Start Jenkins and configure required Plug-ins and Master Slave Configuration:



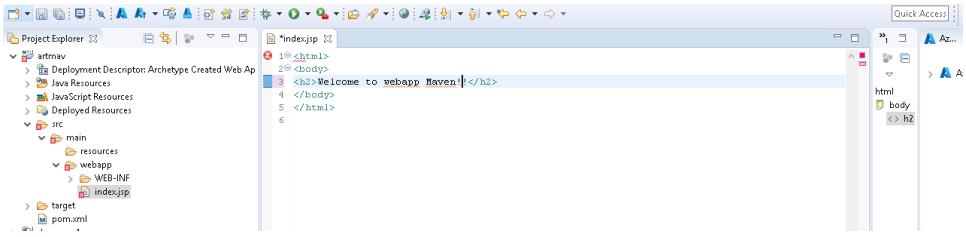


Part 2 – Phase1:

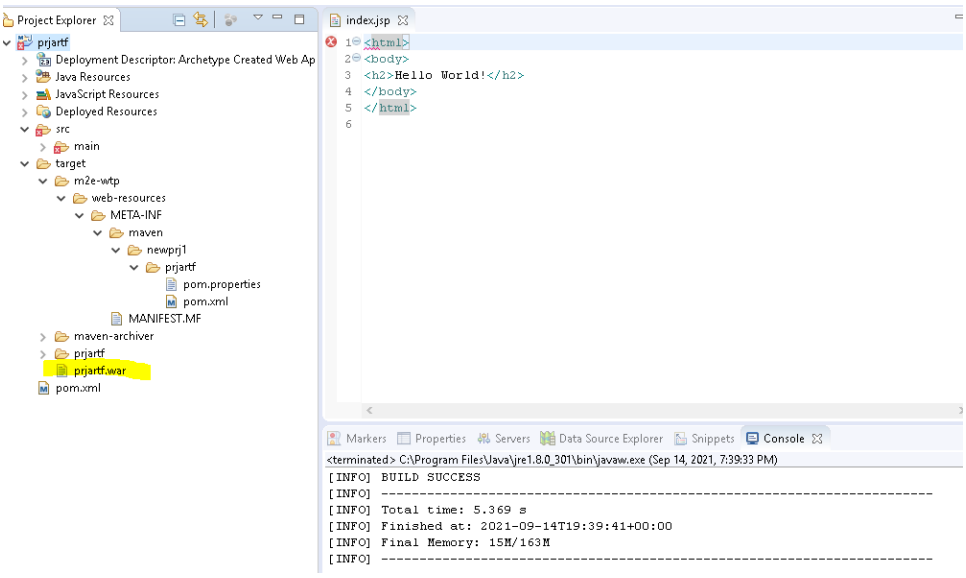
Create Maven Project with Archtype as web application in eclipse:



Modify Index.jsp under src/main/webcontent to display a custom message

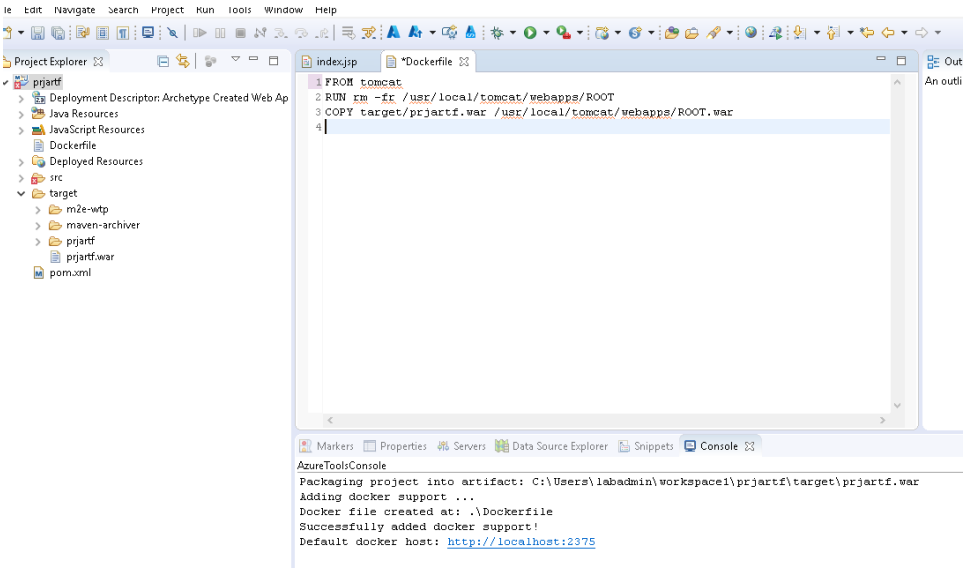


Run Maven clean install in eclipse to check the build and ckeck for .war file in target folder



Part 2- Phase 2

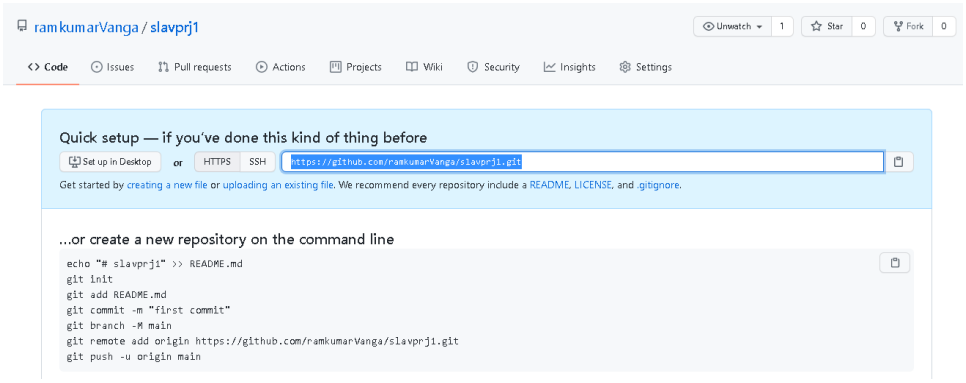
Gerate Dockerfile under project folder of your app & Modify FORM statement to use tomcat as base image



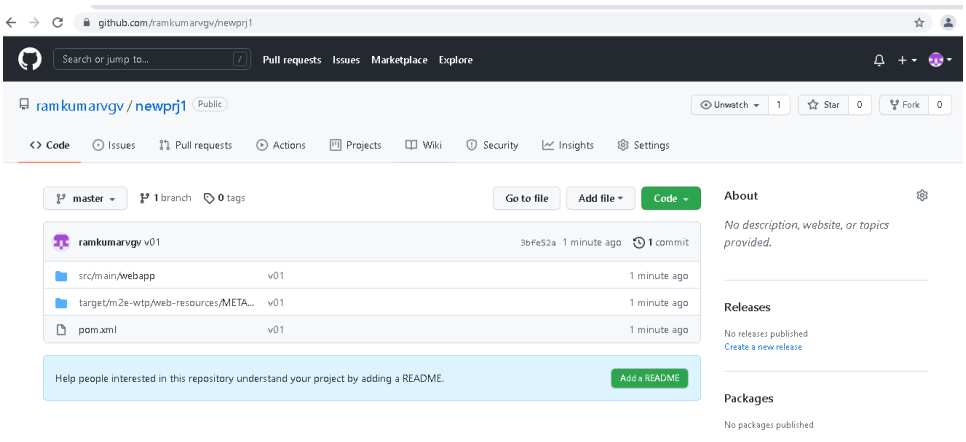
Part 2- Phase 3

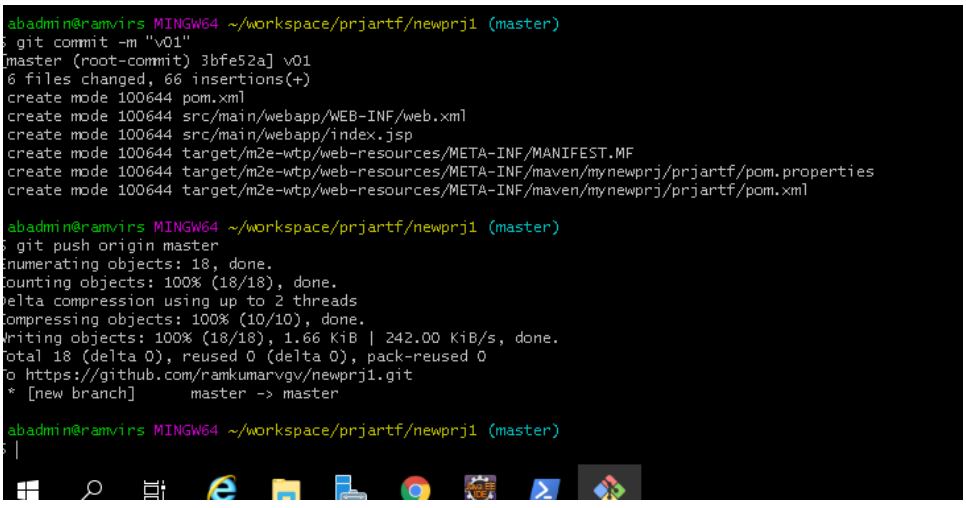
Create a github repository and copy repo URL:

https://github.com/ramkumarvgv/newprj1.git - Repo URL



Commit and Push the code to remote repo





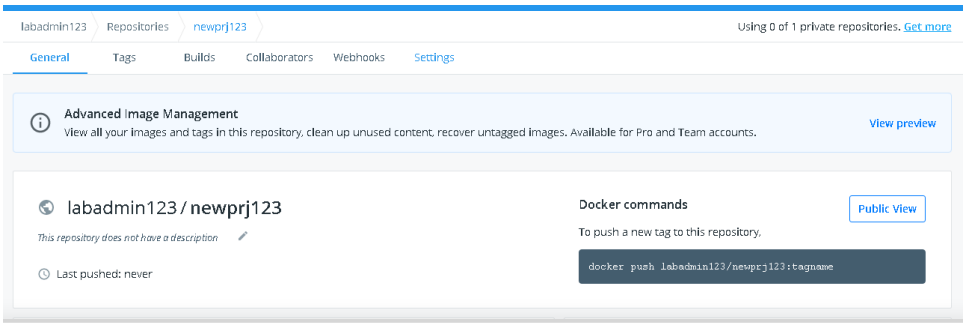
Modify ansible.cfg to use hosts file as inventory:

```
# config file for ansible -- https://ansible.com/
# ===============================================

# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
#forks          = 5
#poll_interval  = 15
#sudo_user      = root
#ask_sudo_pass = True
#ask_pass      = True
#transport      = smart
#remote_port    = 22
```

install python-pip in ansible server

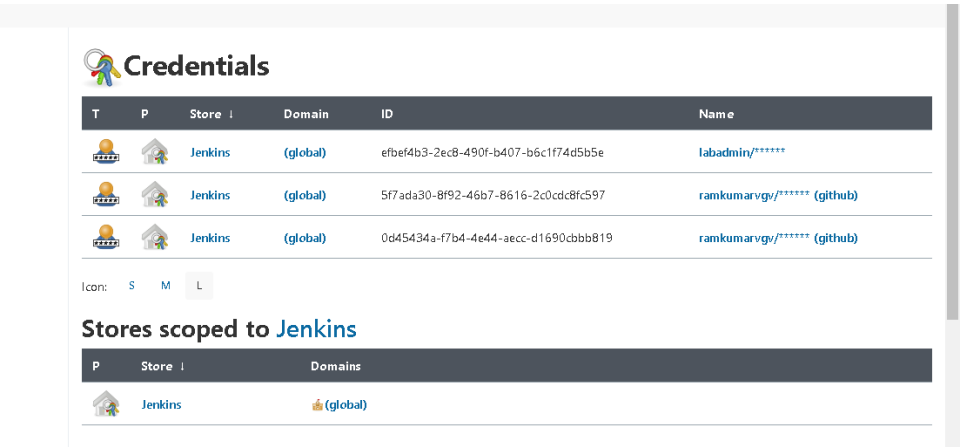
```
vmadmin@Jenkinsbuild:/etc/ansible$ sudo apt-get install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-pip is already the newest version (20.0.2-5ubuntu1.6).
0 upgraded, 0 newly installed, 0 to remove and 160 not upgraded.
vmadmin@Jenkinsbuild:/etc/ansible$ python3-pip --version
python3-pip: command not found
vmadmin@Jenkinsbuild:/etc/ansible$
```

Create Repo in hub.docker.com:

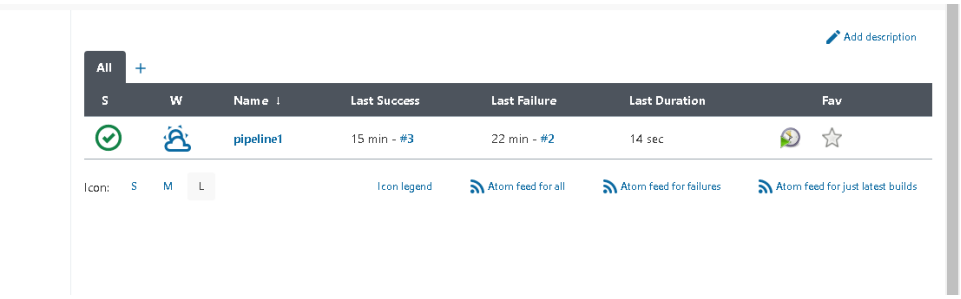


Configure Global tool configurations in Jenkins to use JDK,Maven and Git

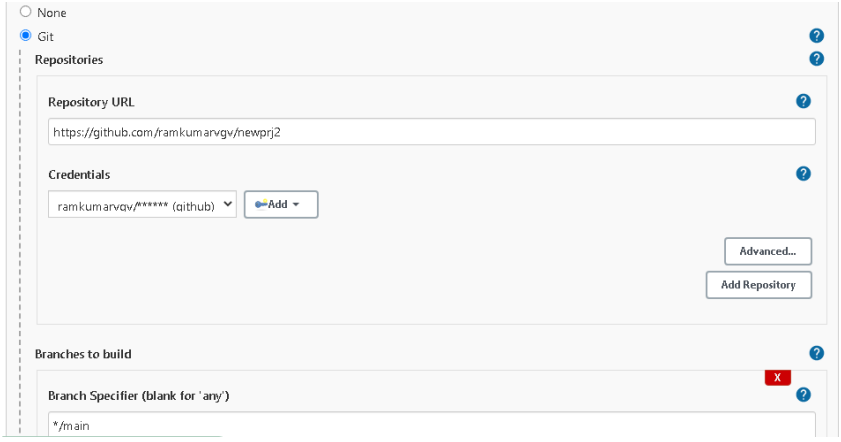
Configure Git credentials in Jenkins Vault



Create Pipeline1 using Freestyle project in Jenkins

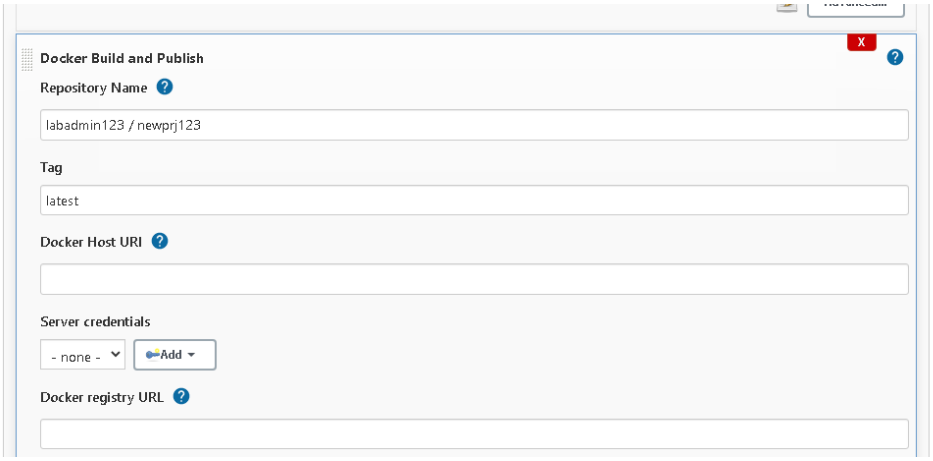


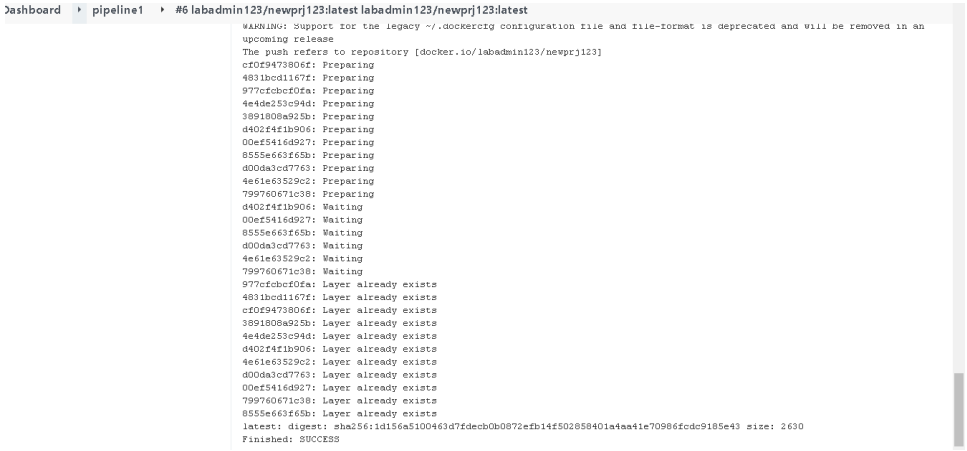
In SCM stage Pull code form Remote Repo

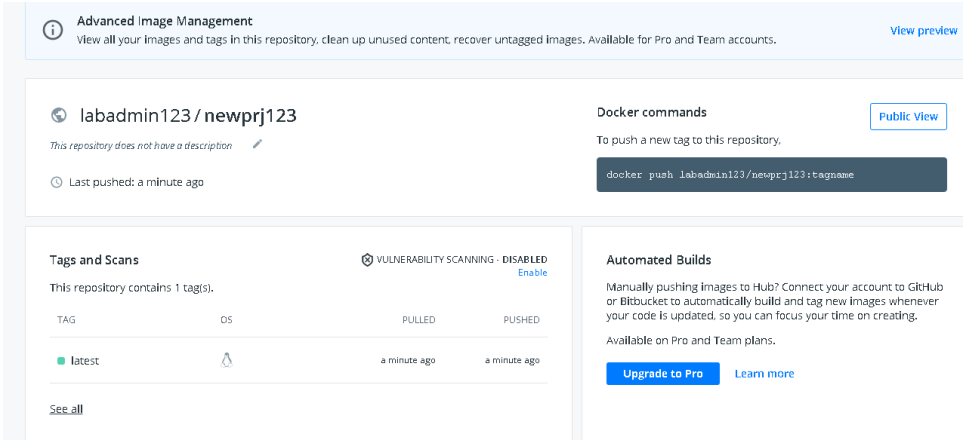


In Build Stage, Step1 : use maven top level target to build

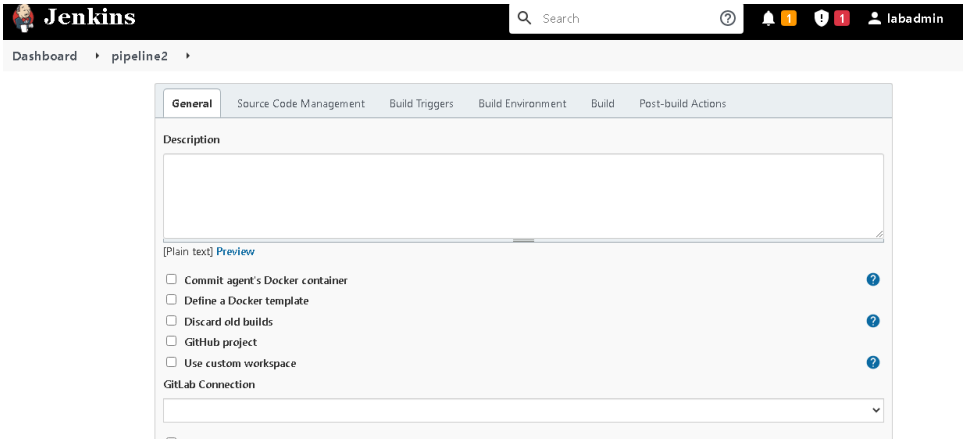
In Build Stage Step 2: User Docker build and Push to create image whih contains your app and push to Docker Hub



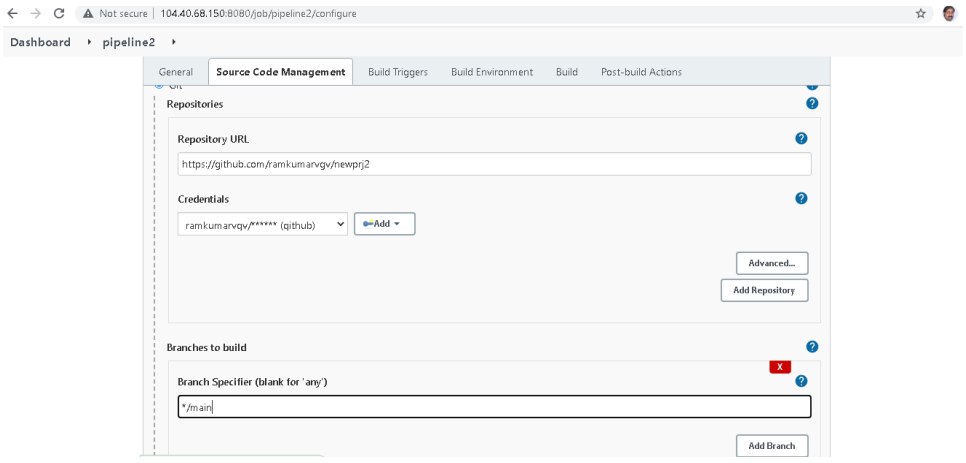




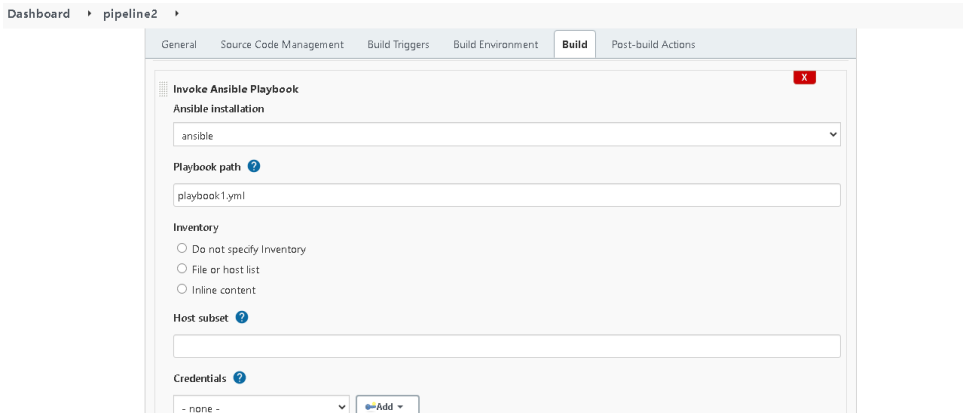
Create Pipeline2 using Freestyle project in Jenkins



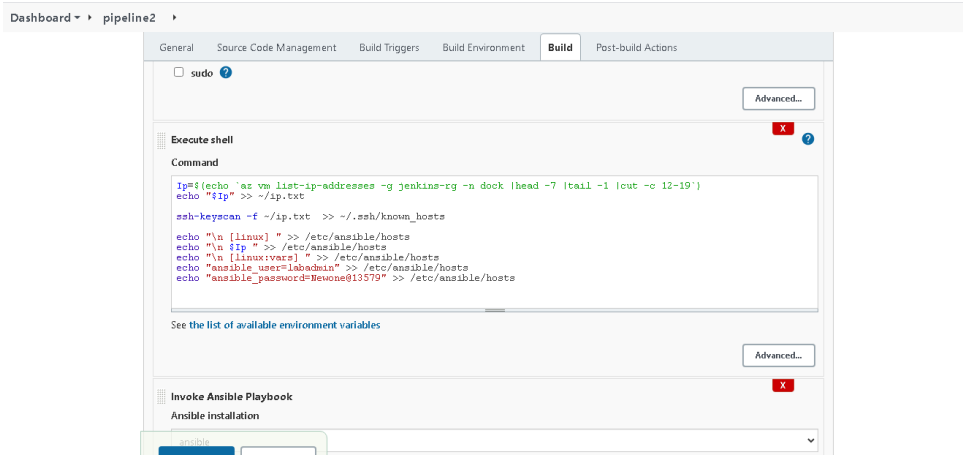
In SCM stage pull code form Remote Repo



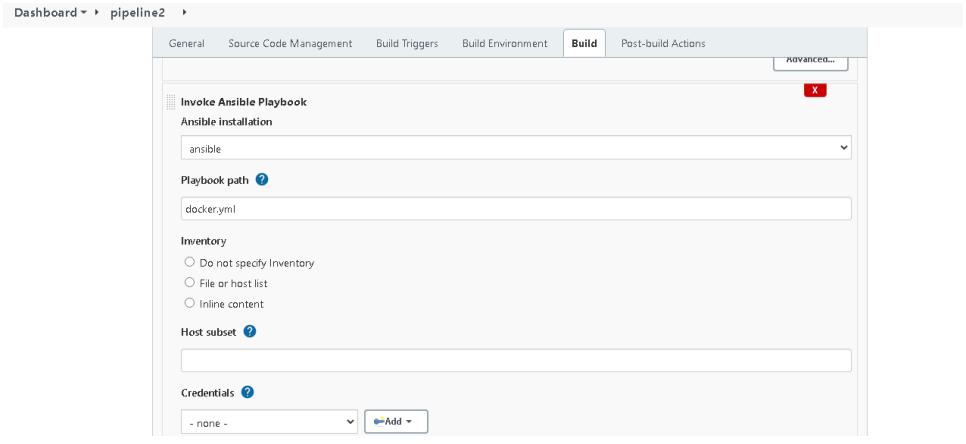
In Build Stage Step 1: Run ansible Playbook1



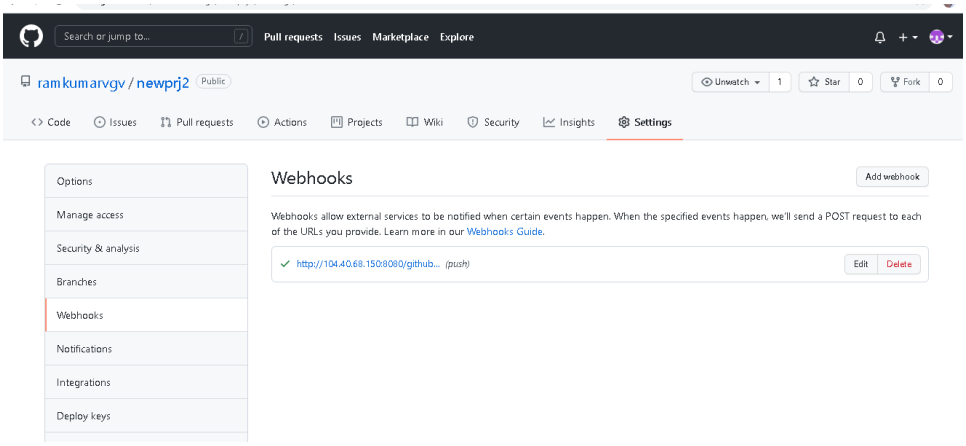
In Build Stage Step 2: Call Shell Script



In Build Stage Step 3: Run Ansible Playbook3(docker.yml)



In Github repo configure webhook for Jenkins



For bothe the Pipelines configure Build trigger to use Github webhook for continuous integration