**Project**

Part-1--Steps

Use AWS CF Terraform

GenerateAccess Key for Terraform in AWS AWS

Create required Vars.tf to create two VMs in AWS Terraform

Create main.tf to create two Linux VMs Terraform

Use Terraform Provisioner to install JDK and Jenkins in VM1 Terraform

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

Init,Plan and Apply Terraform Script Terraform

Manually Start Jenkins and configure required Plug-ins on Master Jenkins

Command used:

1012 terrform init

1013 terraform init

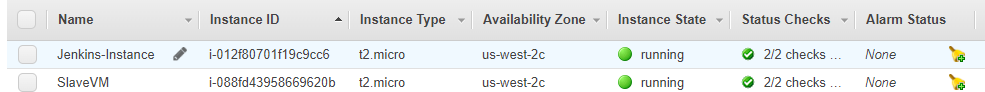
1014 terraform plan

1015 terraform apply

1016 history

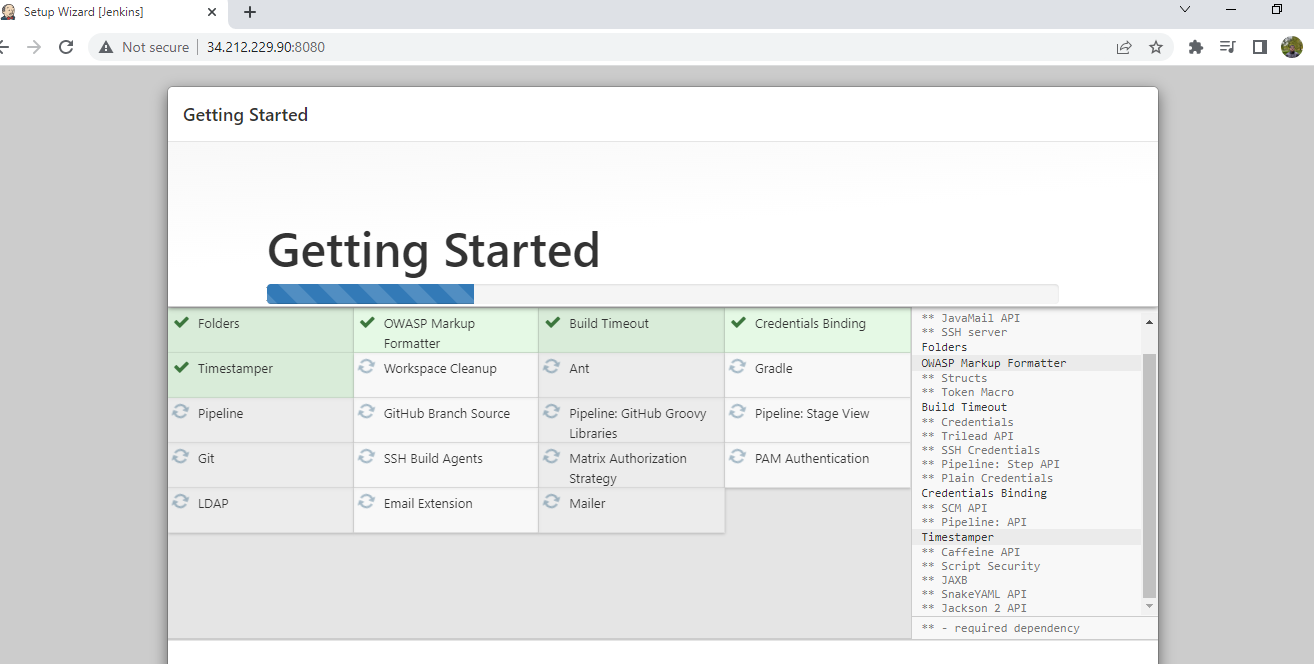
ubuntu@ip-172-31-8-143:~/proj$ ls

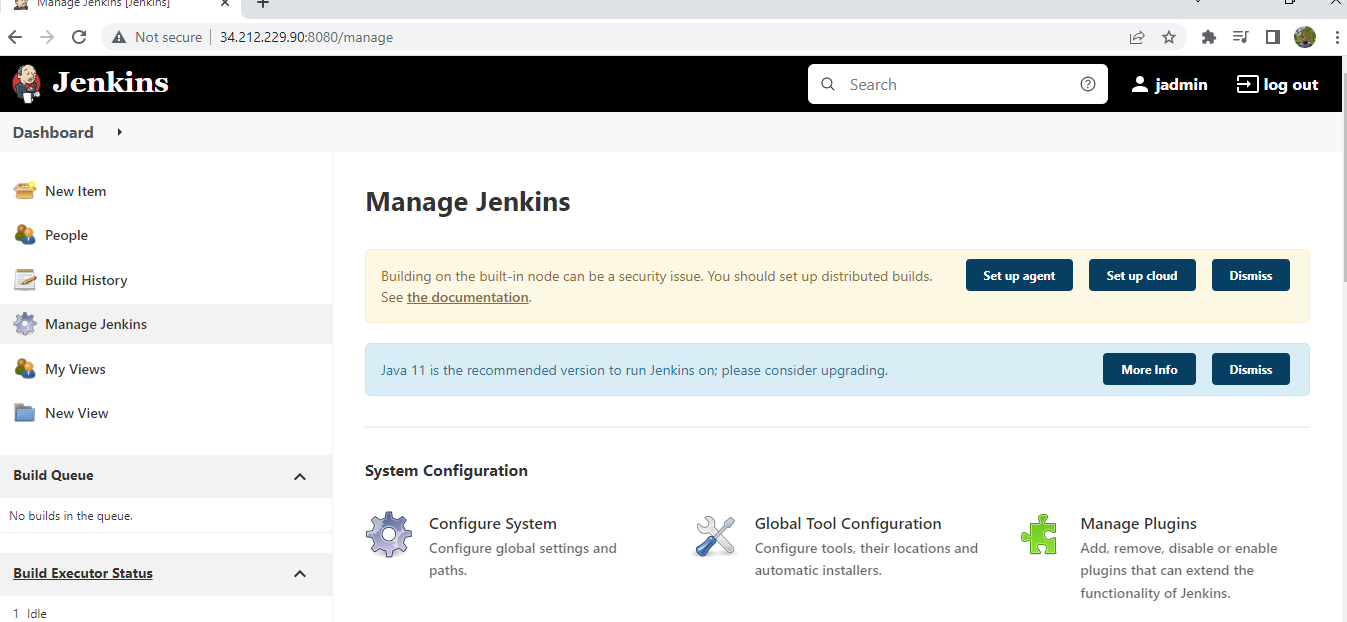
install.sh install\_jenkins.sh main.tf terraform.tfstate terraform.tfstate.backup vars.tf

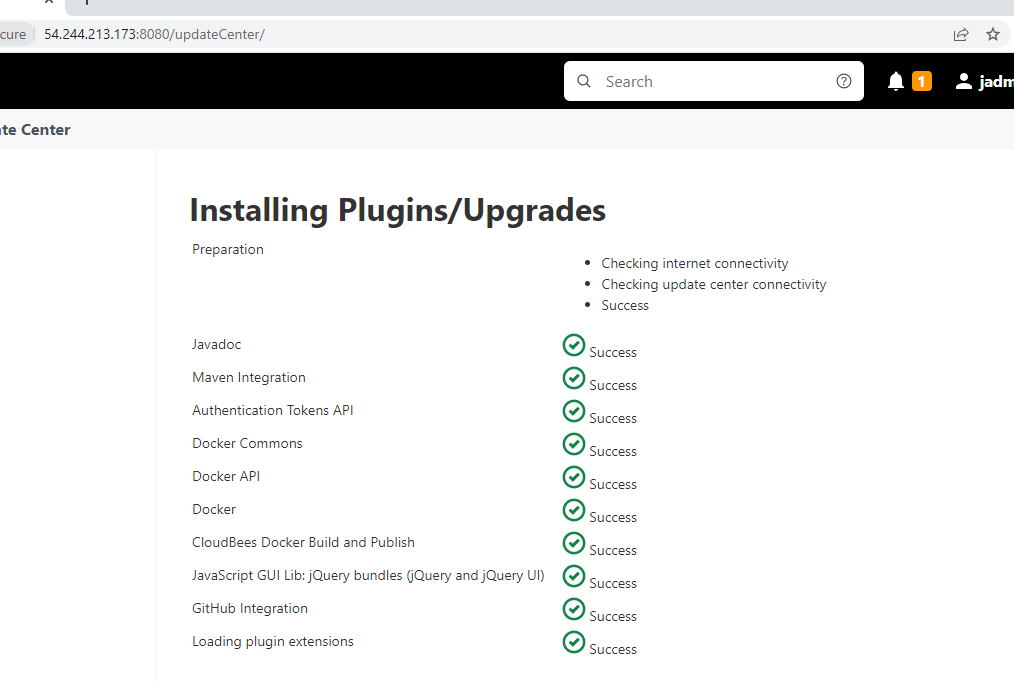


Files Used:









Part 2 - Phase1

Create Maven Project with Archtype as web application in eclipse Eclipse

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

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

Part 2- Phase 2

Gerate Dockerfile under project folder of your app Eclipse

Modify FORM statement to use tomcat as base image Eclipse

Test the Dockerfile by running Docker build and create a container Docker cli

Access the application from container and check it Browser

Part 2- Phase3

Create a github repository and copy repo URL Github Site

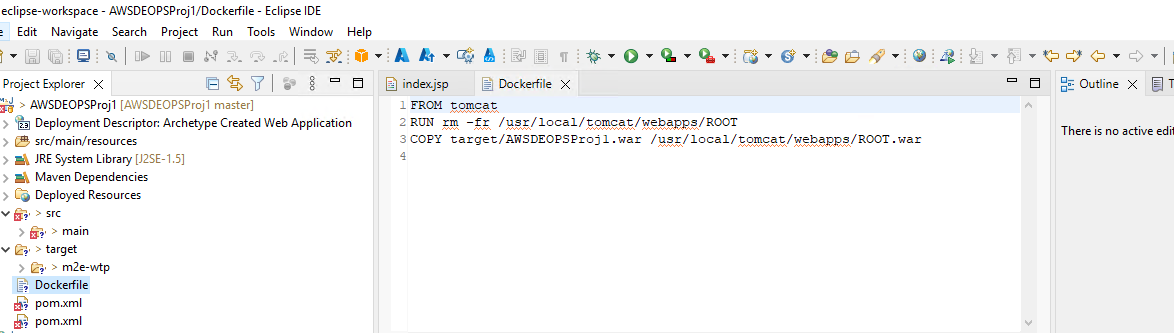
In Eclipse convert the app in to a local repo from Team meanu share Project Option Eclipse

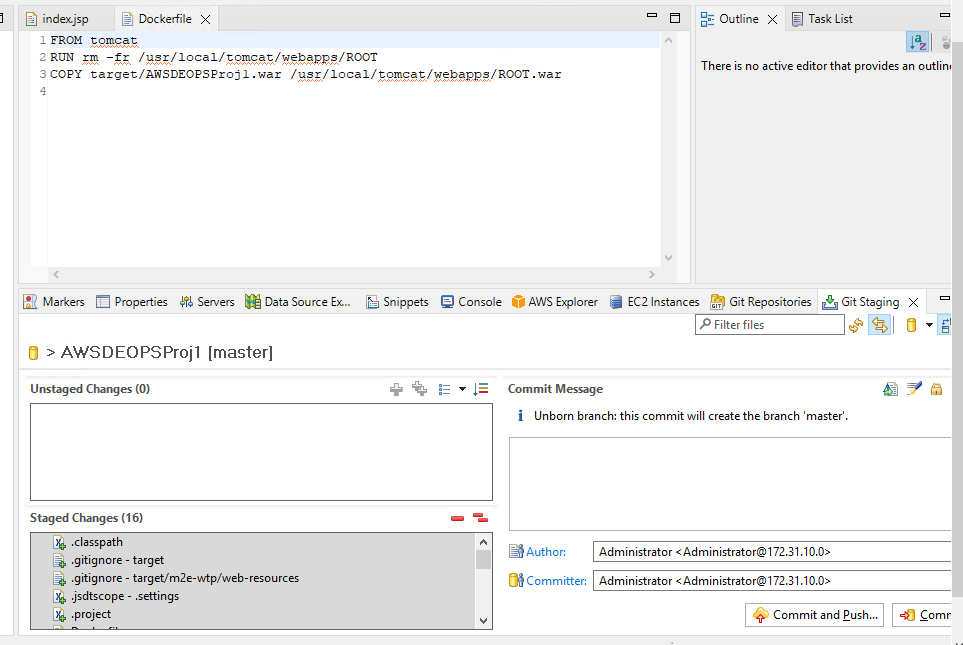
Commit and Push the code to remote repo Eclipse

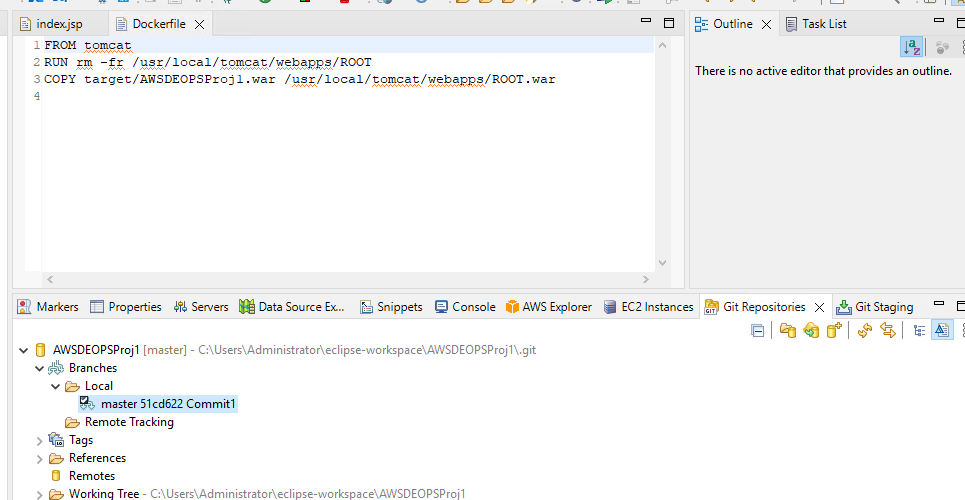
Part 2- Phase 4

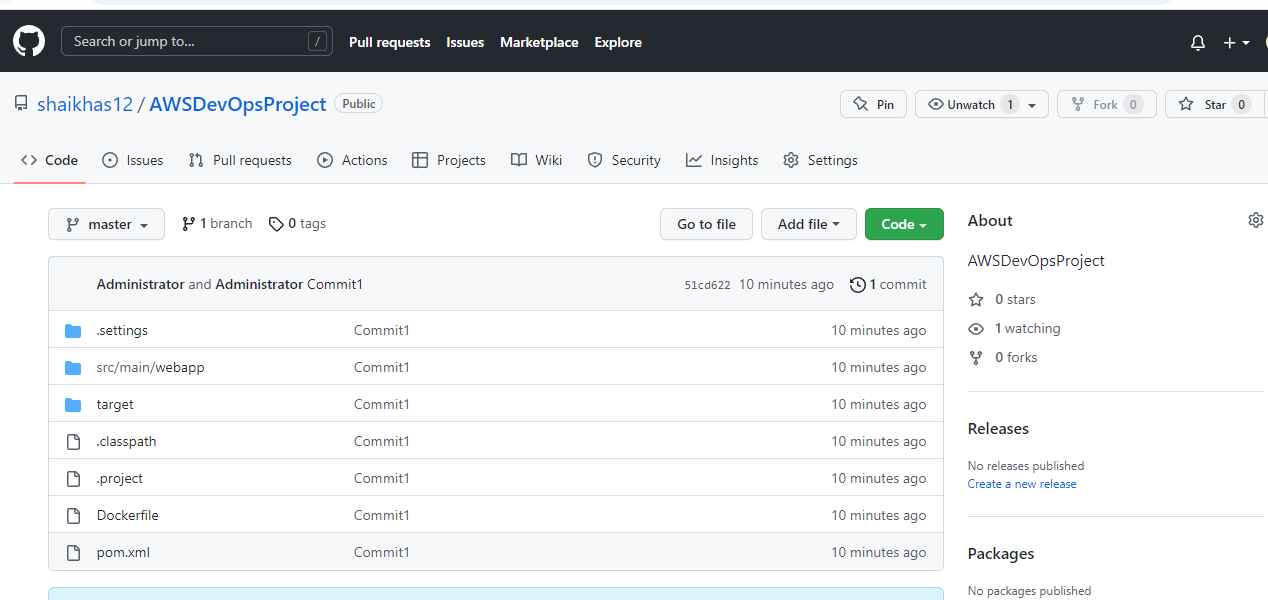
Modify Project index.jsp, Commit and Push to remote repo Eclipse

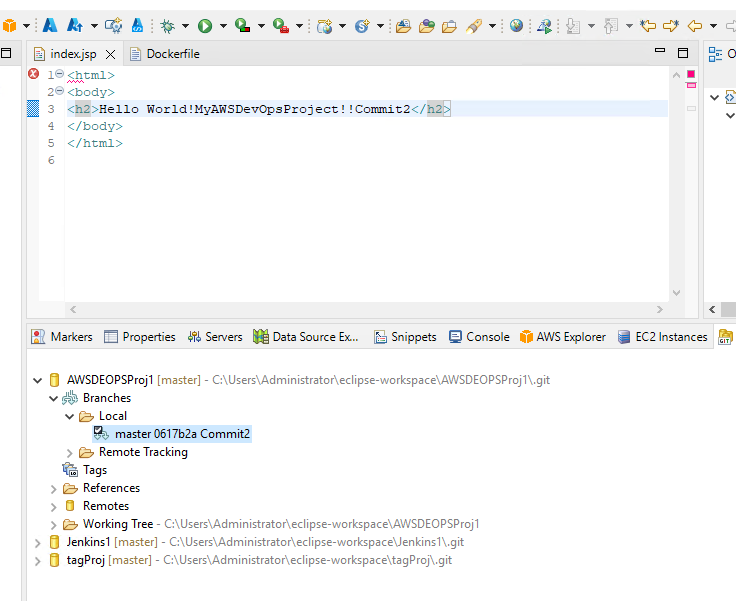
Check for the change in remote Repo Github Site

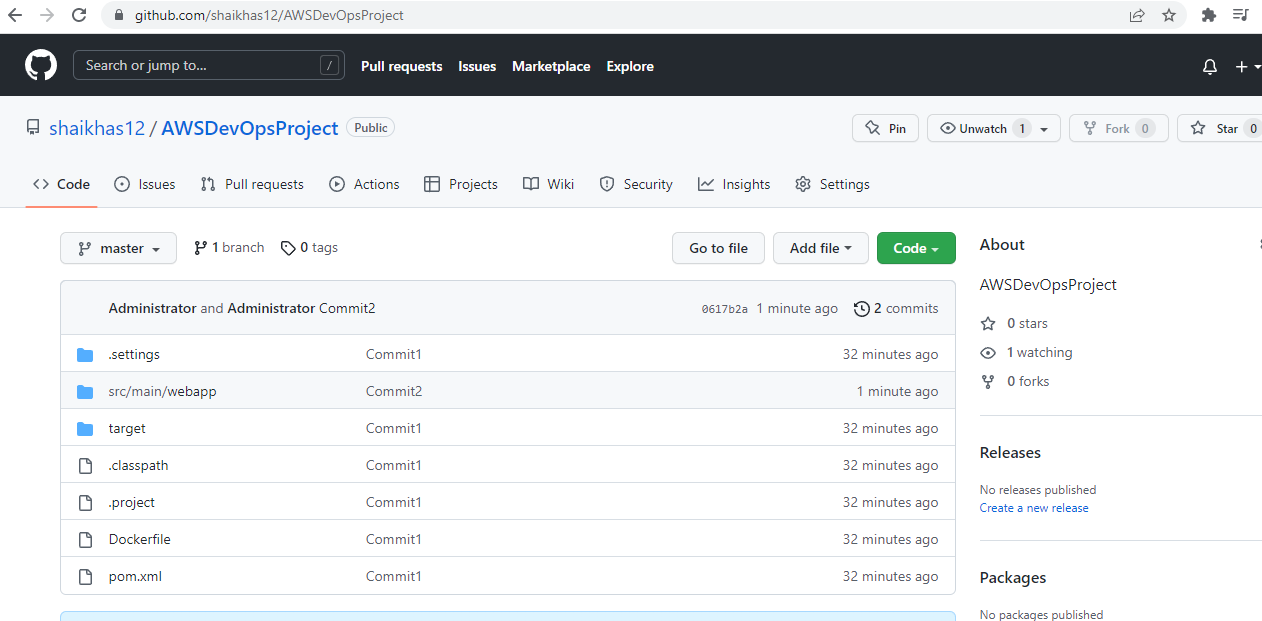












**Docker container:**

ubuntu@ip-172-31-0-22:~/docker$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

shaikhas12/awsdevopsproj1 latest e435a0ba107b 23 minutes ago 685MB

tomcat latest 5eb506608219 5 hours ago 685MB

ubuntu@ip-172-31-0-22:~/docker$ docker run -d -p 9898:80 --name con1 e435a0ba107b

3779c1960c3099d8fd89ec4fe577bd100684cff6f0a10dda4d9de1e5071e1010

ubuntu@ip-172-31-0-22:~/docker$ docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

3779c1960c30 e435a0ba107b "catalina.sh run" 5 seconds ago Up 4 seconds 8080/tcp, 0.0.0.0:9080->80/tcp, :::9080->80/tcp con1

Part 2- Phase 5

In jenkins server configure Ansible manually Shell

Modify ansible.cfg to use hosts file as inventory Shell

install python-pip in ansible server apt-get

Using PIP install aws modules in Ansible server PIP

Use the same access key created for terraform, use in ansible to get authenticated to AWS AWS Console

Create a playbook1 to create Eks cluster in AWS and test vi

**ubuntu@ip-172-31-9-242:~$ ansible --list-host all**

hosts (1):

172.31.15.235

**ubuntu@ip-172-31-9-242:~$ ansible all -m ping**

172.31.15.235 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"changed": false,

"ping": "pong"

}

**Adding the server also in the inventory**

ubuntu@ip-172-31-9-242:~$ ansible all -m ping

[DEPRECATION WARNING]: Distribution Ubuntu 18.04 on host 172.31.9.242 should use /usr/bin/python3, but is using

/usr/bin/python for backward compatibility with prior Ansible releases. A future Ansible release will default to using

the discovered platform python for this host. See

https://docs.ansible.com/ansible/2.9/reference\_appendices/interpreter\_discovery.html for more information. This feature

will be removed in version 2.12. Deprecation warnings can be disabled by setting deprecation\_warnings=False in

ansible.cfg.

172.31.9.242 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python"

},

"changed": false,

"ping": "pong"

}

172.31.15.235 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"changed": false,

"ping": "pong"

}

Part 2- Phase 8

Create Repo in hub.docker.com or acr Docker hub Site

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

Configure Git credentials in Jenkins Vault Jenkins Web UI

Create Pipeline1 using Freestyle project in Jenkins Jenkins Web UI

In SCM stage Pull code form Remote Repo Jenkins Web UI

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

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

Hub or acr

Jenkins Web UI

Create release pipeline in AWS codePipeline AWS

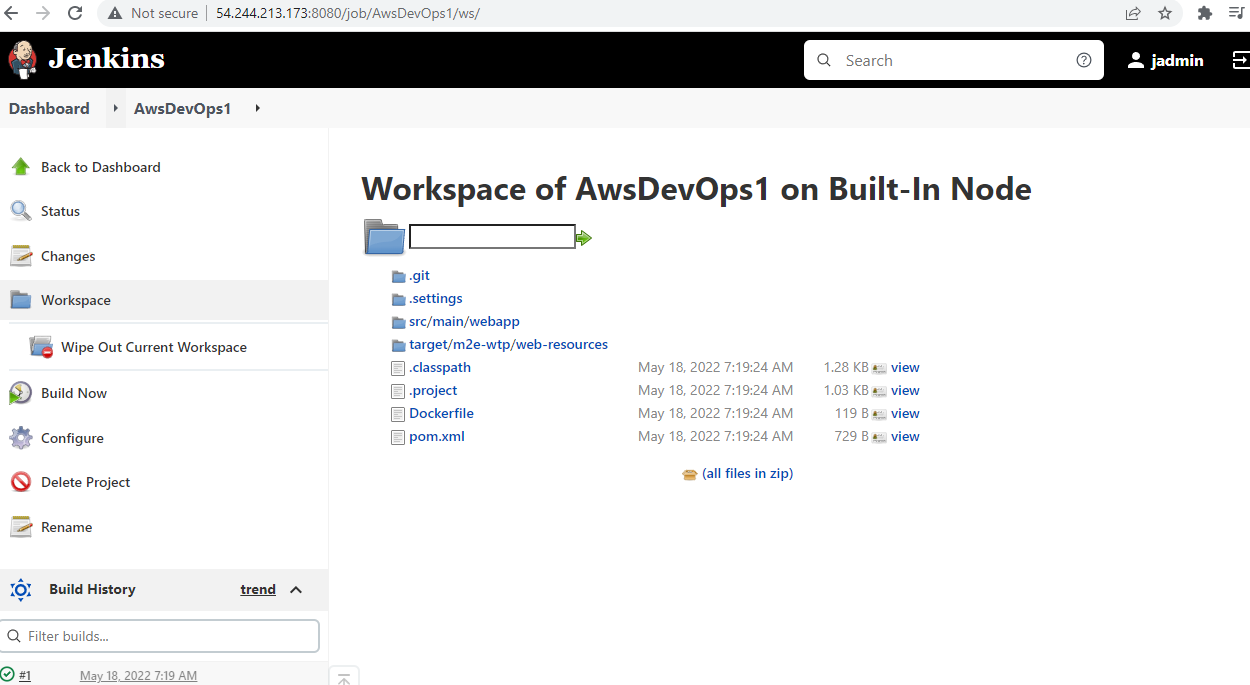
In Artifacts pull plybook and k8s application manifests (YAMLs) form Remote Repo AWS

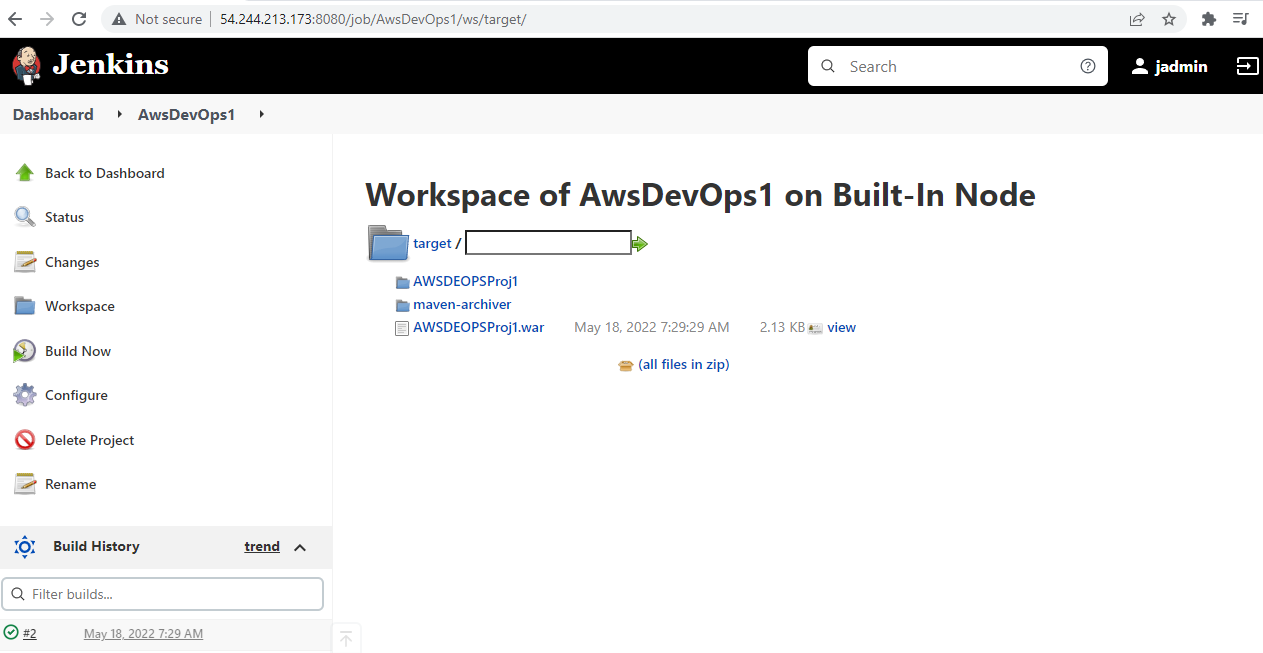
In Build Stage Step 1: Run ansible Playbook1 AWS

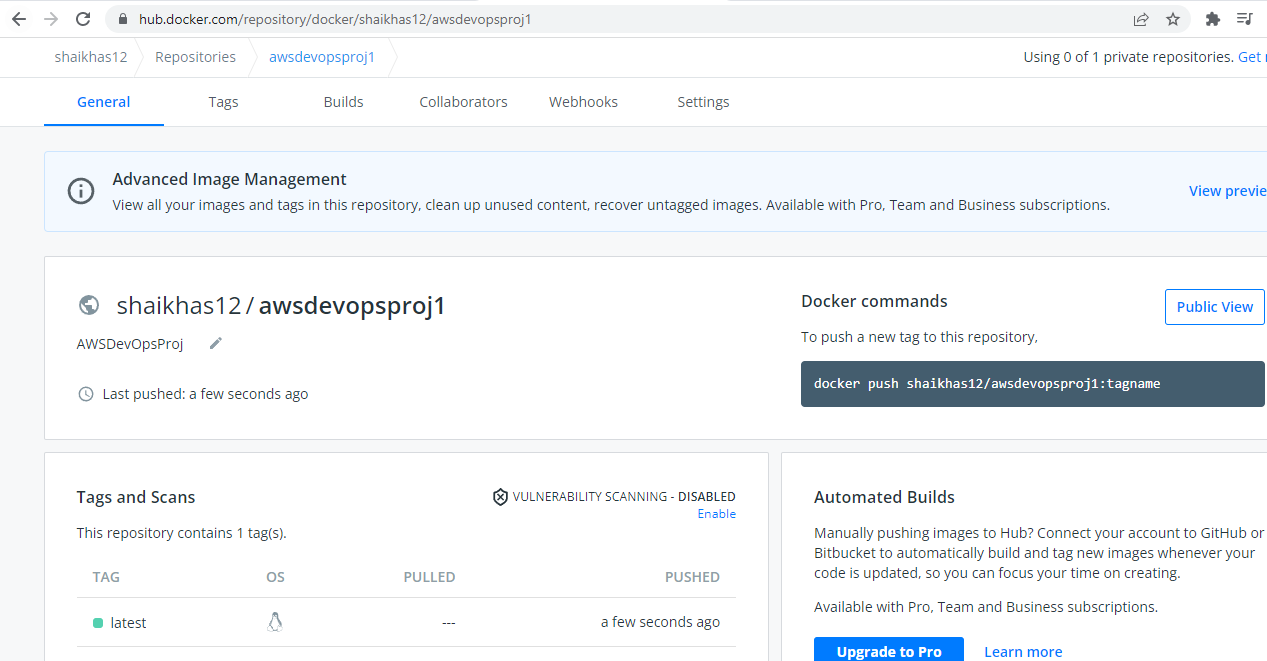
In Build Stage Step2 : Deploy the pods to AKS Cluster using image created in Docker HUB or ACR AWS

In Github repo configure webhook for Jenkins Github Site

For jenkins Pipeline configure Build trigger to use Github webhook for continuous integration Jenkins Web UI







Git Hub URL

https://github.com/shaikhas12/AWSDevOpsProject.git

Docker URL:

https://hub.docker.com/repository/docker/shaikhas12/awsdevopsproj1