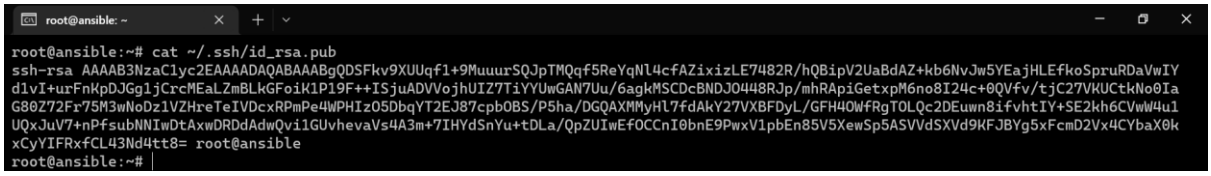
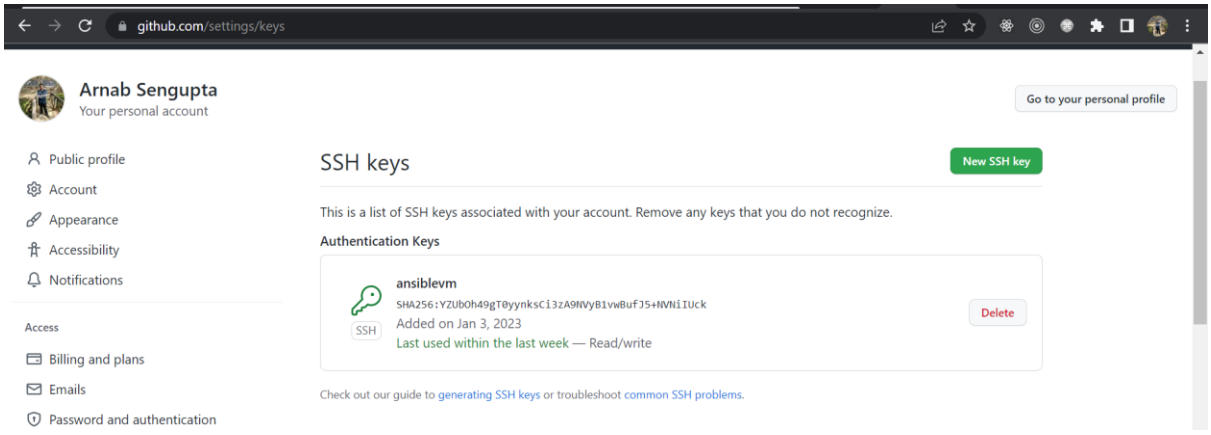
Capstone Project

1. Building an entire CICD pipeline in Jenkins

Jenkins-CI-Pipeline

a. generating SSH key and uploading it to GitHub





b. Create a Jenkins pipeline – integrate the private repo with Jenkins that will be first stage

c. Second stage compile the project using maven but exclude test

d. Third stage do code testing using unittest checkstyle codecoverage

e. Add all this stages in the pipeline

f. Then integrate Jenkins with SonarQube. Provide the difference between sonarqube testing and the previous testing framework

g. Finally upload the jar file inside nexus

h. While uploading jar file pipeline syntax generator provide you with static value.

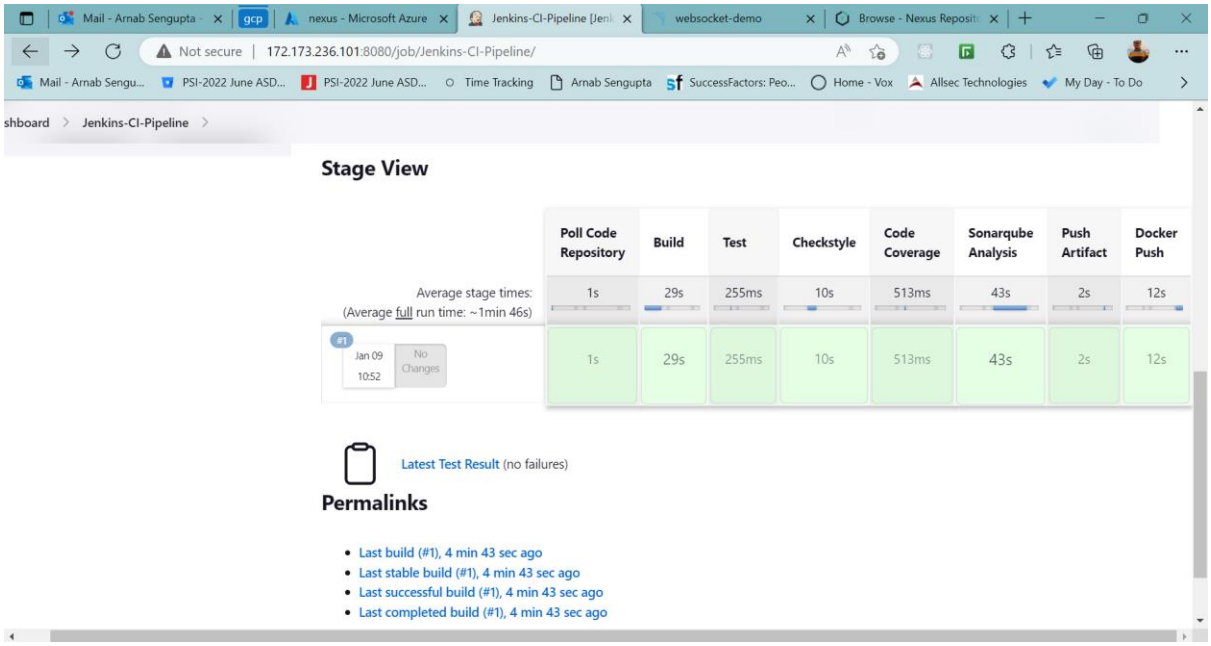
i. Autmatically find out the jar file and upload the same

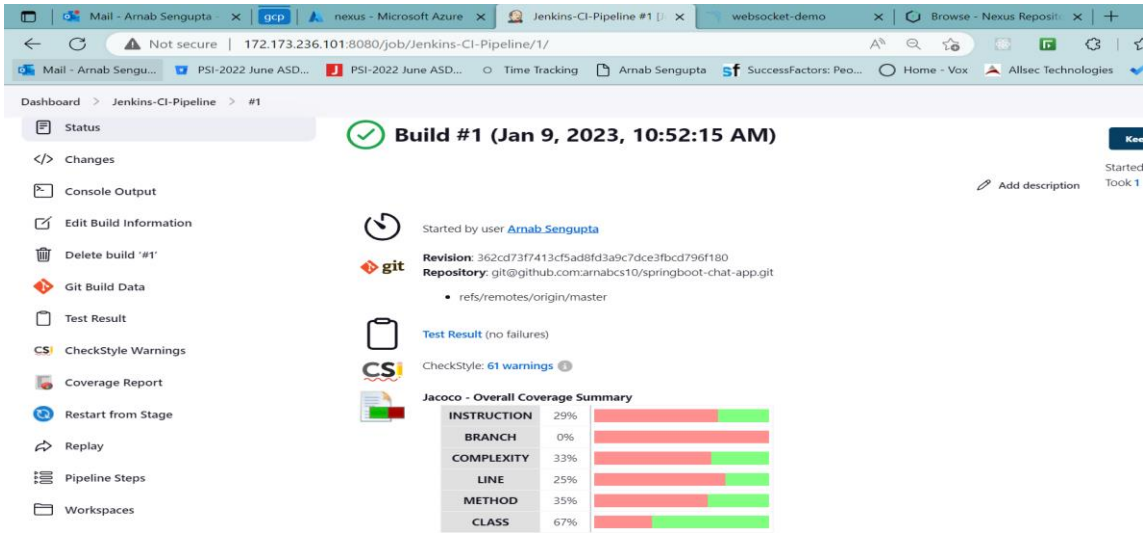
Jenkins-CI-Pipeline

```
pipeline {
    agent any
    stages {
        stage('Poll Code Repository') {
            steps {
                git credentialsId: 'git-key', url:
'git@github.com:arnabcs10/springboot-chat-app.git'
            }
        }
        stage('Build') {
            steps {
                sh 'mvn -B -DskipTest clean package'
            }
        }
        stage('Test') {
            steps {
                junit 'target/surefire-reports/*.xml'
            }
        }
        stage('Checkstyle') {
            steps {
                sh 'mvn checkstyle:checkstyle'
                recordIssues(tools: [checkStyle(pattern: '**/checkstyle-
result.xml')])
            }
        }
        stage('Code Coverage') {
            steps {
                jacoco()
            }
        }
        stage('Sonarqube Analysis'){
            steps {
                sh 'mvn clean verify sonar:sonar \
                  -Dsonar.projectKey=springboot-chat-app \
                  -Dsonar.host.url=http://20.185.62.113:9000 \
```

```
                    -
Dsonar.login=sqp_ea1ed8fedff3c8dc8a5fb91351068cea2769aa91'
            }
        }
        stage('Push Artifact'){
            steps{
                script {
                 pom = readMavenPom file: "pom.xml";
                    filesByGlob = findFiles(glob:
"target/*.${pom.packaging}");
                        echo "${filesByGlob[0].name}
${filesByGlob[0].path} ${filesByGlob[0].directory}
${filesByGlob[0].length} ${filesByGlob[0].lastModified}"
                        artifactPath = filesByGlob[0].path;
                }
                nexusArtifactUploader artifacts: [[artifactId:
pom.artifactId, classifier: '', file: artifactPath, type: pom.packaging,
type: 'war']], credentialsId: 'nexus-cred', groupId: pom.artifactId,
nexusUrl: '40.117.186.85:8081', nexusVersion: 'nexus3', protocol: 'http',
repository: 'maven-snapshots', version: pom.version
            }
        }
        stage('Docker Push') {
            steps {
                script {
                      withDockerRegistry(credentialsId: 'nexus-cred',
url: 'http://40.117.186.85:8085') {
                        // some block
                        sh 'mvn compile jib:build -
Djib.allowInsecureRegistries=true -DsendCredentialsOverHttp'
                      }
                }
            }
        }
    }
}
```

Stages in pipeline





Jacoc Coverage Report

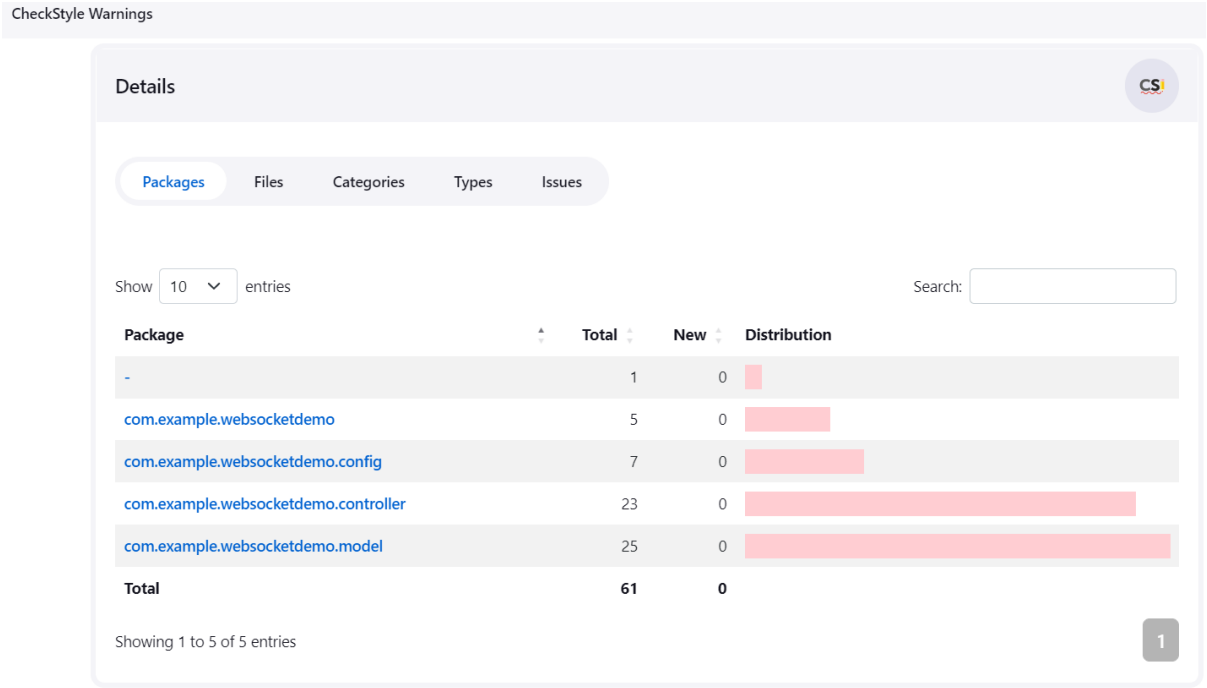
Overall Coverage Summary

name	instruction	branch	complexity	line	method	class
all classes	29% M: 110 C: 46	0% M: 2 C: 0	33% M: 14 C: 7	25% M: 30 C: 10	35% M: 13 C: 7	67% M: 2 C: 4

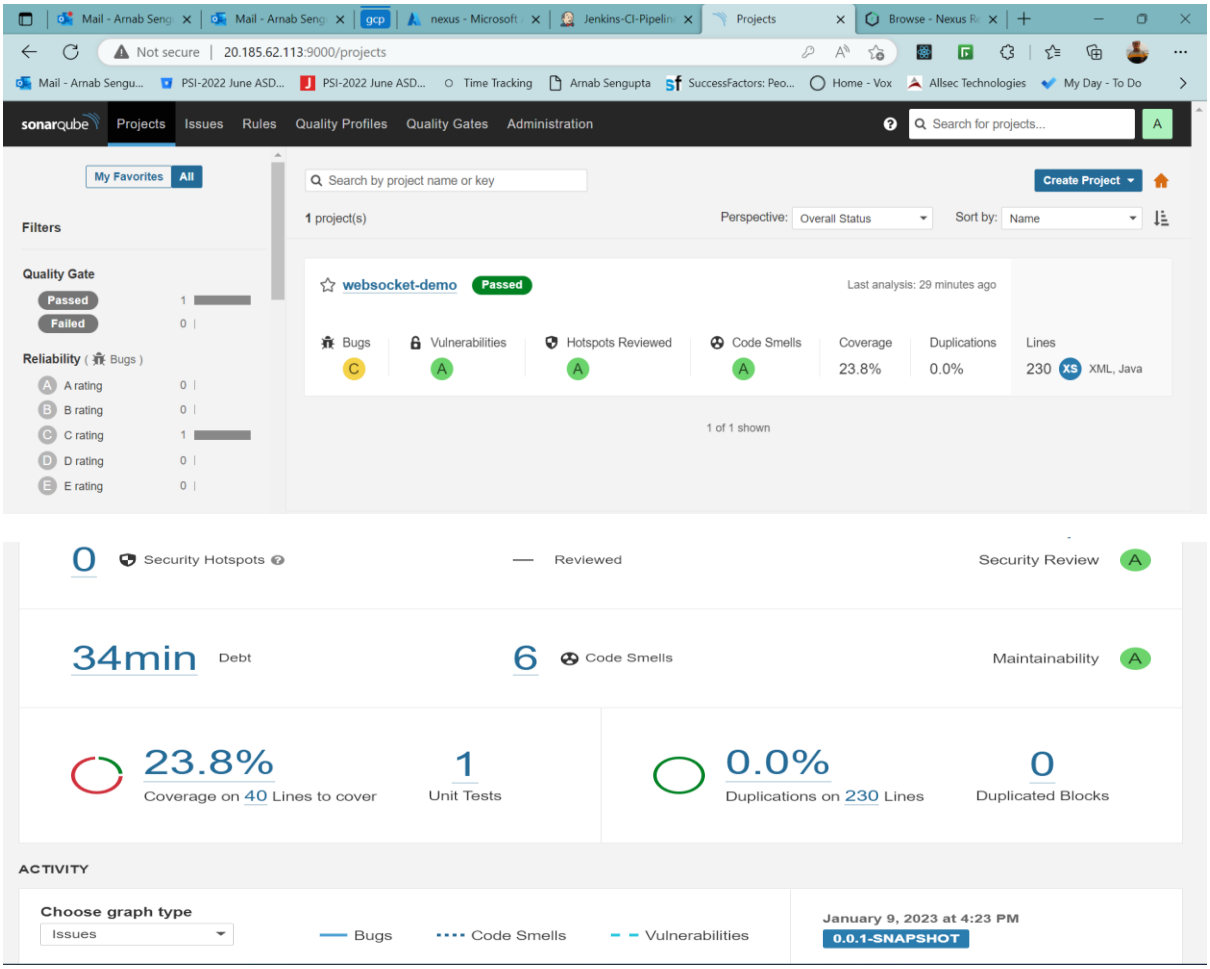
Coverage Breakdown by Package

name	instruction	branch	complexity	line	method	class
com.example.websocketdemo	M: 5 C: 3 38%	M: 0 C: 0 100%	M: 1 C: 1 50%	M: 2 C: 1 33%	M: 1 C: 1 50%	M: 0 C: 1 100%
com.example.websocketdemo.config	M: 0 C: 33 100%	M: 0 C: 0 100%	M: 0 C: 3 100%	M: 0 C: 6 100%	M: 0 C: 3 100%	M: 0 C: 1 100%
com.example.websocketdemo.controller	M: 47 C: 10 18%	M: 2 C: 0 0%	M: 5 C: 3 38%	M: 14 C: 3 18%	M: 4 C: 3 43%	M: 0 C: 2 100%
com.example.websocketdemo.model	M: 58 C: 0 0%	M: 0 C: 0 100%	M: 8 C: 0 0%	M: 14 C: 0 0%	M: 8 C: 0 0%	M: 2 C: 0 0%

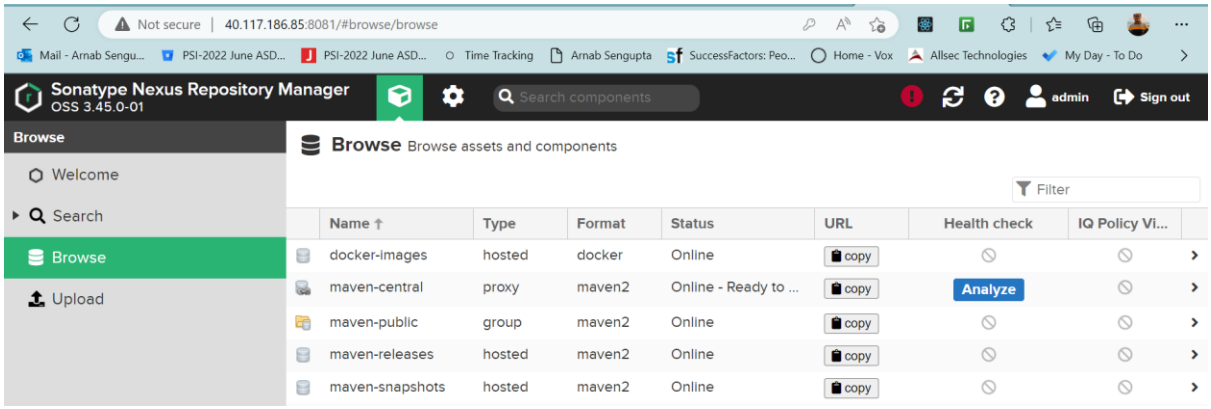
CheckStyle Report

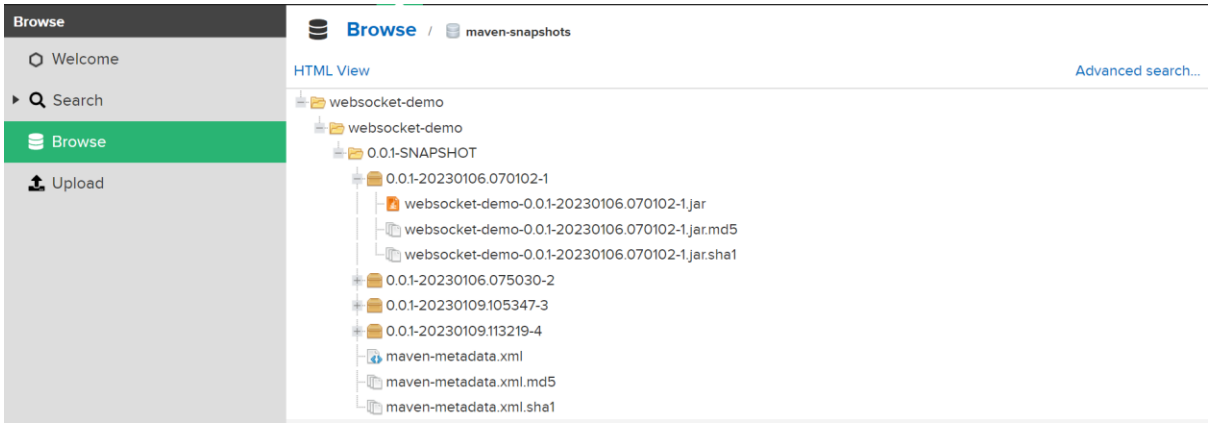


SonarQube Report

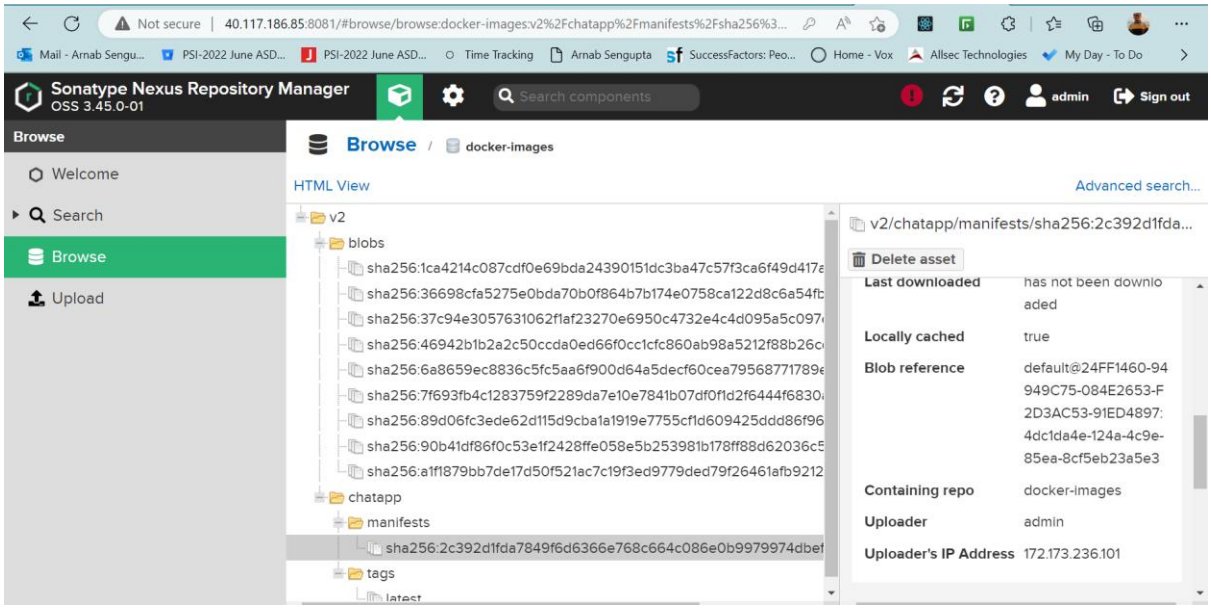


Nexus Artifact Registry





Docker Image



Jenkins CD-pipeline

a. Create an ansible playbook which install docker in Ubuntu system

Playbook for installing Docker in Ubuntu system

```
---
- name: Install Docker in Ubuntu system
  hosts: all
  become: true
  tasks:
    - name: Update and upgrade apt packages
      apt:
        upgrade: yes
        update_cache: yes
    - name: Install required system packages
      apt:
        pkg:
          - apt-transport-https
          - ca-certificates
          - curl
          - software-properties-common
          - python3-pip
          - virtualenv
          - python3-setuptools
        state: latest
        update_cache: true

    - name: Add Docker GPG apt Key
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present

    - name: Add Docker Repository
      apt_repository:
        repo: 'deb https://download.docker.com/linux/ubuntu
bionic stable'
        state: present
```

```
    - name: Update apt and install docker-ce
      apt:
        name: docker-ce
        state: latest
        update_cache: yes

    - name: Install Docker Module for Python
      pip:
        name: docker

    - name: Allow insecure registries
      copy:
        dest: /etc/docker/daemon.json
        content: |-
          {
            "insecure-registries" : ["40.117.186.85:8085"]
          }

    - name: Enable the docker service and start
      service:
          name: docker
          enabled: yes
          state: restarted

    - name: Check if docker is Installed
      shell:
        docker -v
```

b. Create another ansible playbook which will deploy the private image from nexus repo

Playbook for deploying docker image, pulling from nexus private remository

```
---
- name: Deploy Docker Image
  hosts: all
  become: true
  vars_files:
```

```
    - cred.yaml
  tasks:
    - name: Log into private registry and force re-
authorization
      docker_login:
        registry: 40.117.186.85:8085
        username: "{{ username }}"
        password: "{{ password }}"
        reauthorize: yes

    - name: Running Docker Conatiner
      docker_container:
        name: chatapp
        image: 40.117.186.85:8085/chatapp
        state: started
        pull: yes
        ports:
        - "8080:8080"
    - name: Connect to app server on port 8080 and check
status 200 - Try 5 times
      tags: test
      uri:
        url: http://localhost:8080
      register: result
      until: "result.status == 200"
      retries: 5
      delay: 10
```

Encrypt nexus credentials in cred.yaml file, which will be used as authentication while pulling docker image from private nexus registry

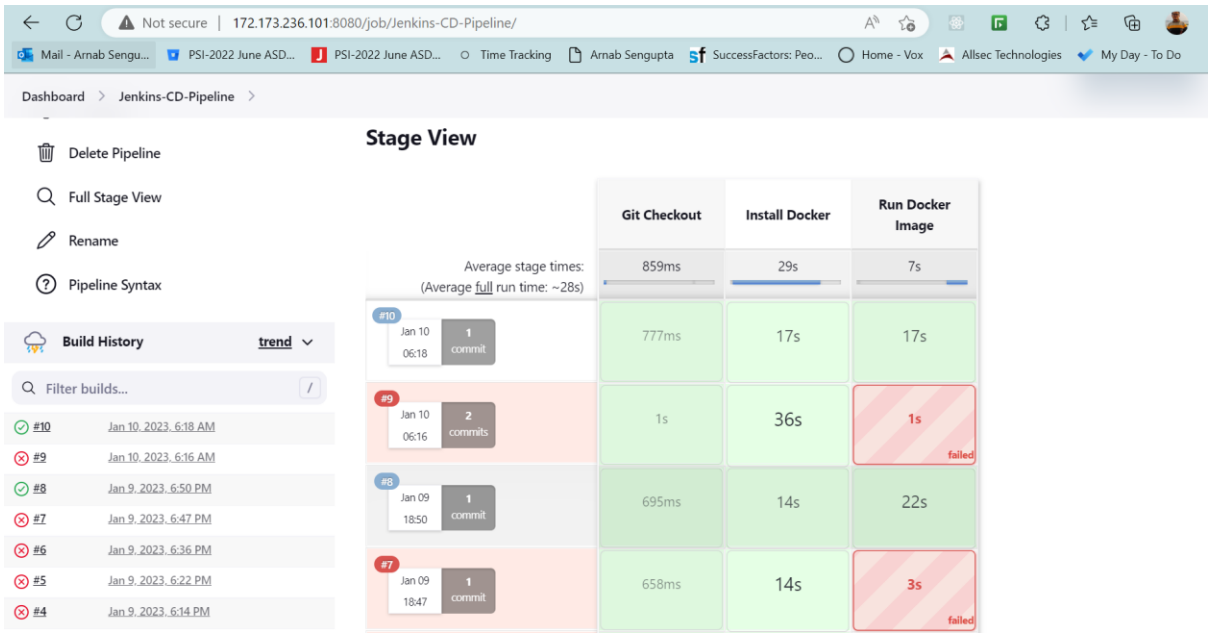
```
root@ansible:~/springboot-chat-app/ansible# ansible-vault create cred.yaml
New Vault password:
Confirm New Vault password:
root@ansible:~/springboot-chat-app/ansible# cat cred.yaml
$ANSIBLE_VAULT;1.1;AES256
33386261653866646361643931643730653037386561393338343236316230356565326434313031
35663664623166373538306633643530656532343333339390a373066623164353637623738616634
36306232653261316663313231346661333432633537383665616438613162373237343863313832
38656436313432333310a35646333393363356466626333613337316431303664653630323361383 2
30393132343331383632623933353335363461343930383064643438376534333863326535326663
38363734323235376631363964303261386163383734346462
root@ansible:~/springboot-chat-app/ansible# git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        cred.yaml

nothing added to commit but untracked files present (use "git add" to track)
root@ansible:~/springboot-chat-app/ansible# git add .
root@ansible:~/springboot-chat-app/ansible# git commit -m "added encrypted files for nexus auth"
[master a678068] added encrypted files for nexus auth
```

Final Jenkins-CD-Pipeline

```
pipeline {
    agent any
    stages {
        stage('Git Checkout') {
            steps {
                git credentialsId: 'git-key', url:
'git@github.com:arnabcs10/springboot-chat-app.git'
            }
        }

        stage('Install Docker'){
            steps {
                ansiblePlaybook colorized: true,
credentialsId: 'git-key', disableHostKeyChecking: true,
inventory: 'ansible/dev.inv', playbook: 'ansible/docker-
install.yaml'
            }
        }
        stage('Run Docker Image'){
            steps {
                ansiblePlaybook colorized: true,
credentialsId: 'git-key', disableHostKeyChecking: true,
inventory: 'ansible/dev.inv', playbook: 'ansible/deploy-
docker-image.yaml', vaultCredentialsId: 'vault-pass'
            }
        }
    }
```

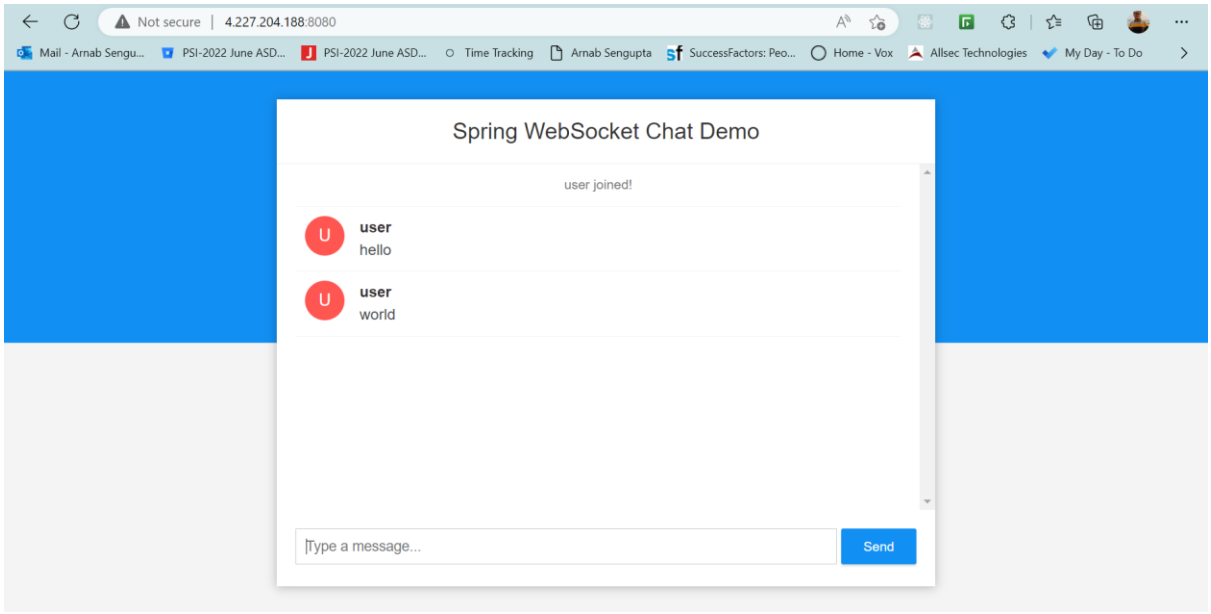
```
}
```

Jenkins pipeline stages



Docker installed in Client machine and container running on port 8080:

```
root@client:~#
root@client:~#
root@client:~# docker -v
Docker version 20.10.22, build 3a2c30b
root@client:~#
root@client:~#
root@client:~# docker images
REPOSITORY                    TAG       IMAGE ID       CREATED         SIZE
40.117.186.85:8085/chatapp    latest    90b41df86f0c   53 years ago    237MB
root@client:~#
root@client:~#
root@client:~# docker ps
CONTAINER ID   IMAGE                         COMMAND                  CREATED        STATUS         PORTS                    NAMES
fb4a97a40470   40.117.186.85:8085/chatapp    "java -cp /app/resou…"   12 hours ago   Up 8 minutes   0.0.0.0:8080->8080/tcp   chatapp
root@client:~#
root@client:~#
root@client:~#
```



Jenkisn CD pipeline with tomcat

Create an ansible playbook which will install tomcat

Ansible playbook for installing tomcat to remote server client

```
---
- name: Install Tomcat to remote server
  hosts: all
  become: yes
  vars:
    download_url: https://dlcdn.apache.org/tomcat/tomcat-
10/v10.0.27/bin/apache-tomcat-10.0.27.tar.gz
  tasks:
   - name: Update and upgrade apt packages
     apt:
      upgrade: yes
      update_cache: yes

   - name: Download Open JDK
     apt:
      name: default-jdk
      update_cache: yes
      state: present
```

```
- name: Check if Java is Installed
  shell:
   java -version

- name: Create Group
  group:
   name: tomcat
   state: present

- name: Create User
  user:
     name: tomcat
     state: present

- name: Create a Directory /opt/tomcat
  file:
   path: /opt/tomcat
   state: directory
   mode: 0755
   owner: tomcat
   group: tomcat

- name: Download Tomcat using unarchive
  unarchive:
   src: "{{download_url}}"
   dest: /opt/tomcat
   remote_src: yes
   extra_opts: [--strip-components=1]
- name: Change ownership of tomcat directory
  file:
   path: /opt/tomcat
   owner: tomcat
   group: tomcat
   mode: "u+rwx,g+rx,o=rx"
   recurse: yes
   state: directory

- name: Creating a service file
```

```
    copy:
     src: templates/tomcat.service.j2
     dest: /etc/systemd/system/tomcat.service

  - name: Reload the SystemD to re-read configurations
    systemd:
       daemon-reload: yes

  - name: Enable the tomcat service and start
    systemd:
       name: tomcat
       enabled: yes
       state: started

  - name: Connect to Tomcat server on port 8080 and check
status 200 - Try 5 times
    tags: test
    uri:
      url: http://localhost:8080
    register: result
    until: "result.status == 200"
    retries: 5
    delay: 10
```

tomcat.service.j2

```
[Unit]
        Description=Tomcat Service
        Requires=network.target
        After=network.target

        [Service]
        Type=forking
        User=tomcat
        Group=tomcat

        Environment="JAVA_HOME=/usr/lib/jvm/java-1.11.0-
openjdk-amd64"
```

```
        Environment="JAVA_OPTS=-
Djava.security.egd=file:///dev/urandom"
        Environment="CATALINA_PID=/opt/tomcat/logs/tomcat.pi
d"
        Environment="CATALINA_BASE=/opt/tomcat"
        Environment="CATALINA_HOME=/opt/tomcat"
        Environment="CATALINA_OPTS=-Xms512M -Xmx1024M -
server -XX:+UseParallelGC"

        ExecStart=/opt/tomcat/bin/startup.sh
        ExecStop=/opt/tomcat/bin/shutdown.sh
        Restart=always
        RestartSec=10

        [Install]
        WantedBy=multi-user.target
```

Ansible playbook for deploying WAR file to Tomcat server in Client

```
---
- name: Install Tomcat to remote server
  hosts: all
  become: yes
  vars_files:
    - cred.yaml
  tasks:
   - name: Update and upgrade apt packages
     apt:
      upgrade: yes
      update_cache: yes

   - name: Install lxml
     pip:
      name: lxml

   - name: Download artifact from nexus private repository
     maven_artifact:
```

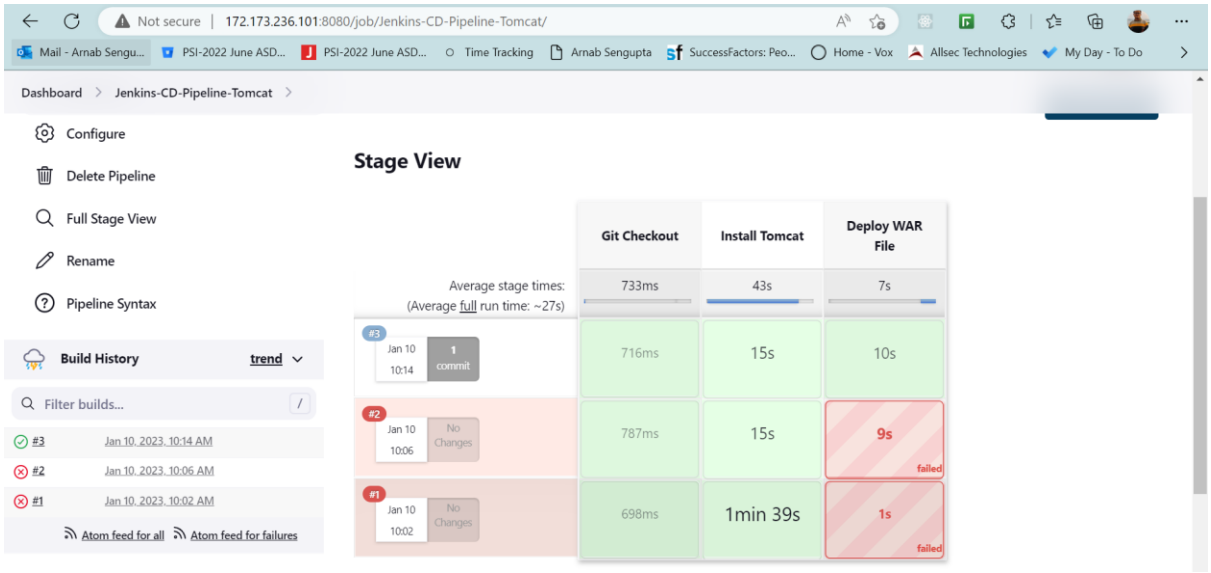
```
      group_id: websocket-demo
      artifact_id: websocket-demo
      version: 0.0.1-SNAPSHOT
      repository_url:
'http://40.117.186.85:8081/repository/maven-snapshots'
      username: "{{ username }}"
      password: "{{ password }}"
      dest: /opt/tomcat/webapps/chatapp.war
      mode: "0644"

   - name: Restart Tomcat
     systemd:
      name: tomcat
      enabled: yes
      state: restarted
```

Jenkins-CD-Pipeline-Tomcat

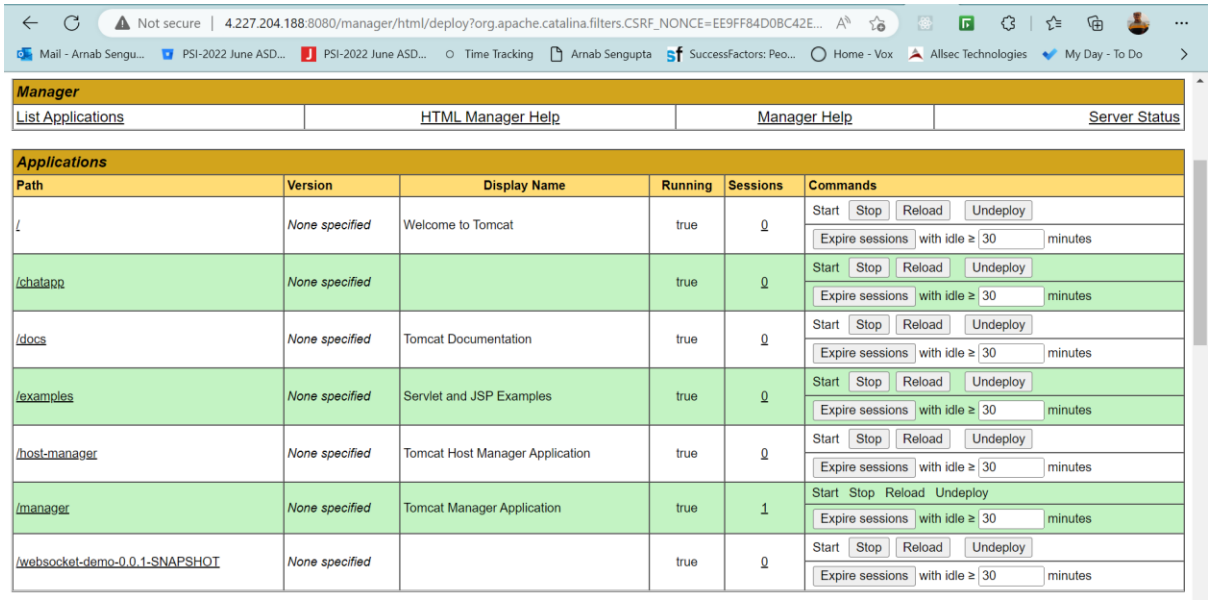
Jenkins pipeline for installing tomcat and deploying war file on tomcat server in client

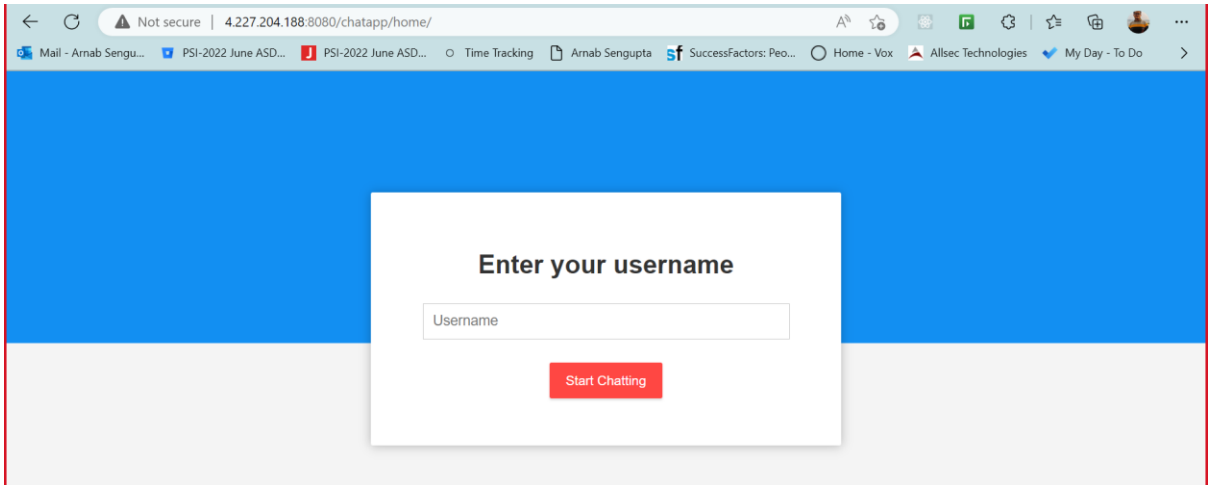
```
pipeline {
    agent any
    stages {
        stage('Git Checkout') {
            steps {
                git credentialsId: 'git-key', url:
'git@github.com:arnabcs10/springboot-chat-app.git'
            }
        }

        stage('Install Tomcat'){
            steps {
                ansiblePlaybook colorized: true,
credentialsId: 'git-key', disableHostKeyChecking: true,
inventory: 'ansible/dev.inv', playbook:
'ansible/tomcat.yaml'
            }
        }
```

```
        stage('Deploy WAR File'){
            steps {
                ansiblePlaybook colorized: true,
credentialsId: 'git-key', disableHostKeyChecking: true,
inventory: 'ansible/dev.inv', playbook: 'ansible/deploy-war-
file.yaml', vaultCredentialsId: 'vault-pass'
            }
        }

    }
}
```




```
root@client:/opt/tomcat/webapps#
root@client:/opt/tomcat/webapps# systemctl status tomcat
● tomcat.service - Tomcat Service
     Loaded: loaded (/etc/systemd/system/tomcat.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2023-01-10 10:23:45 UTC; 1h 22min ago
    Process: 15201 ExecStart=/opt/tomcat/bin/startup.sh (code=exited, status=0/SUCCESS)
   Main PID: 15208 (java)
      Tasks: 30 (limit: 9530)
     Memory: 431.3M
     CGroup: /system.slice/tomcat.service
             └─15208 /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -Djava.util.logging.config.file=/opt/tomcat/conf/logging.proper>

Jan 10 10:23:45 client systemd[1]: Starting Tomcat Service...
Jan 10 10:23:45 client startup.sh[15201]: Tomcat started.
Jan 10 10:23:45 client systemd[1]: Started Tomcat Service.
lines 1-13/13 (END)
```


Not secure | 4.227.204.188:8080/manager/html/deploy?org.apache.catalina.filters.CSRF_NONCE=EE9FF84D0BC42E...
Mail - Arnab Sengu...
PSI-2022 June ASD...
PSI-2022 June ASD...
Time Tracking
Arnab Sengupta
SuccessFactors: Peo...
Home - Vox
Allsec Technologies
My Day - To Do
Manager
List Applications
HTML Manager Help
Manager Help
Server Status
Applications
Path
Version
Display Name
Running
Sessions
Commands
/
None specified
Welcome to Tomcat
true
0
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes
/chatapp
None specified
true
0
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes
/docs
None specified
Tomcat Documentation
true
0
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes
/examples
None specified
Servlet and JSP Examples
true
0
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes
/host-manager
None specified
Tomcat Host Manager Application
true
0
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes
/manager
None specified
Tomcat Manager Application
true
1
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes
/websocket-demo-0.0.1-SNAPSHOT
None specified
true
0
Start Stop Reload Undeploy
Expire sessions with idle ≥ 30 minutes


Not secure | 4.227.204.188:8080/chatapp/home/
Mail - Arnab Sengu...
PSI-2022 June ASD...
PSI-2022 June ASD...
Time Tracking
Arnab Sengupta
SuccessFactors: Peo...
Home - Vox
Allsec Technologies
My Day - To Do
Enter your username
Username
Start Chatting

Jenkins CD Pipeline terraform

1. Authenticate Jenkins server to Azure CLI

2. Create terraform manifest file for provisioning resources.

```
root@ansible:~/new# cd springboot-chat-app/
root@ansible:~/new/springboot-chat-app# ls
Readme.md  azure-pipelines.yml  deploy.yml     infrastructure       mvnw      pom.xml         src
ansible    build.sh             developer.jpg  k8s-deployment.yaml  mvnw.cmd  screenshot.png  start.sh
root@ansible:~/new/springboot-chat-app# cd infrastructure/
root@ansible:~/new/springboot-chat-app/infrastructure# tree
.
├── input-variables.tf
├── locals.tf
├── network-interface.tf
├── null-provisioner.tf
├── output.tf
├── provider.tf
├── resource-group.tf
├── script.sh
├── virtual-network.tf
├── vm.tf
├── vnet-input-varaibles.tf
└── web-subnet.tf

0 directories, 12 files
root@ansible:~/new/springboot-chat-app/infrastructure#
```

3. Create Null provisioner to transfer ssh public key to remote host and installing required packages and dependencies

```
resource "null_resource" "null_copy_ssh_key_to_vm" {
  depends_on = [
    azurerm_linux_virtual_machine.webserver
  ]
  connection {
    type        = "ssh"
    host        =
azurerm_linux_virtual_machine.webserver.public_ip_address
    user        =
azurerm_linux_virtual_machine.webserver.admin_username
    private_key = file("~/.ssh/id_rsa")
  }
  #file provisioiner which will upload my key
  provisioner "file" {
    source      = "~/.ssh/id_rsa"
    destination = "/tmp/id_rsa"
  }
  provisioner "file" {
    source      = "script.sh"
```

```
    destination = "/tmp/script.sh"
  }
  provisioner "remote-exec" {
    inline = [
      "sudo chmod 400 /tmp/id_rsa",
      "chmod +x /tmp/script.sh",
      "/tmp/script.sh"
    ]
  }
}
```

Script.sh

```
#!/bin/bash

# Update apt repository
echo Update apt repository
sudo apt-get update -y

# Install Docker
echo Install Docker
sudo apt-get install -y ca-certificates  curl  gnupg  lsb-
release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-compose-plugin -y

# Install kind
echo Install kind
```

```
curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.17.0/kind-linux-amd64
chmod +x ./kind
sudo mv ./kind /usr/local/bin/kind
kind --version

# Install kubectl
echo Install kubectl
sudo apt-get install -y apt-transport-https
sudo curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main' | sudo tee /etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
kubectl version --short

# Create kind cluster
echo Create kind cluster
sudo kind create cluster
sudo kubectl get nodes
sudo kubectl get pods

# Install helm
echo Install helm
wget https://get.helm.sh/helm-v3.10.3-linux-amd64.tar.gz
tar zxvf helm-v3.10.3-linux-amd64.tar.gz
cd linux-amd64/
sudo cp helm /usr/bin
helm version
```

4. Create Jenkins pipeline

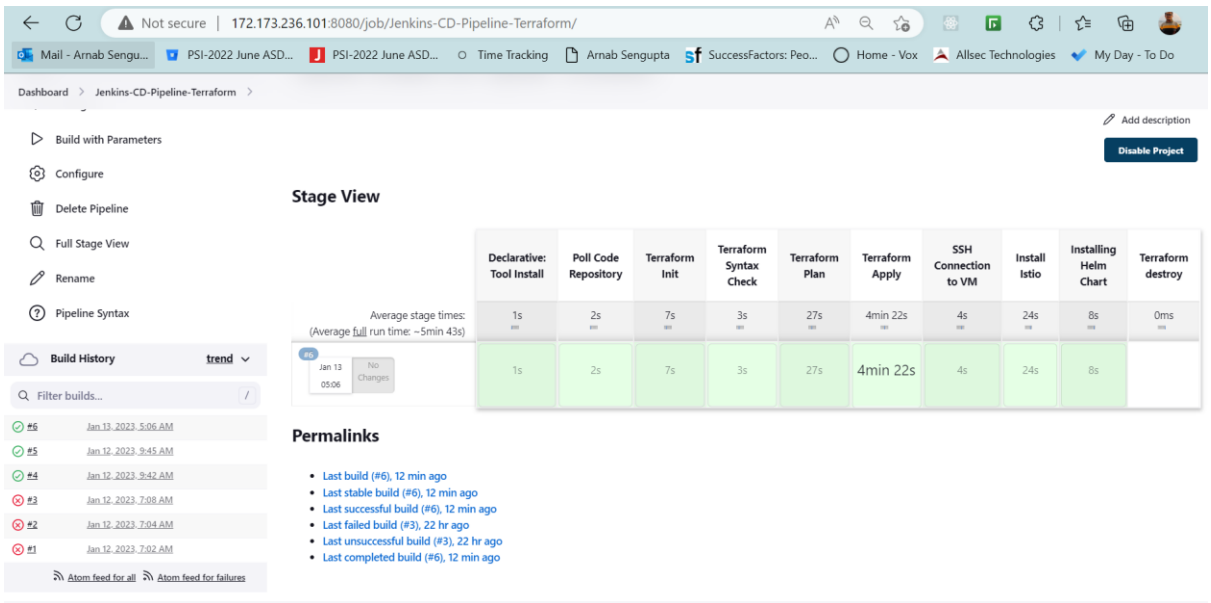
```
pipeline {
    agent any
```

```
    tools {
       terraform 'terraform'
    }
    environment {
        RESOURCE_GROUP = 'rg-default'
        HOST_VM = 'web-linuxvm'
        USERNAME = 'azureuser'
    }
    stages {
        stage('Poll Code Repository') {
            steps {
                git credentialsId: 'git-key', url:
'git@github.com:arnabcs10/springboot-chat-app.git'
            }
        }
        stage('Terraform Init') {
            when {
                expression{action == "apply"}
            }
            steps{
               script {
                    dir('infrastructure/') {
                        sh 'terraform init'
                    }
                }
            }
        }
        stage('Terraform Syntax Check') {
            when {
                expression{action == "apply"}
            }
            steps{
               script {
                    dir('infrastructure/') {
                        sh 'terraform validate'
                    }
                }
            }
```

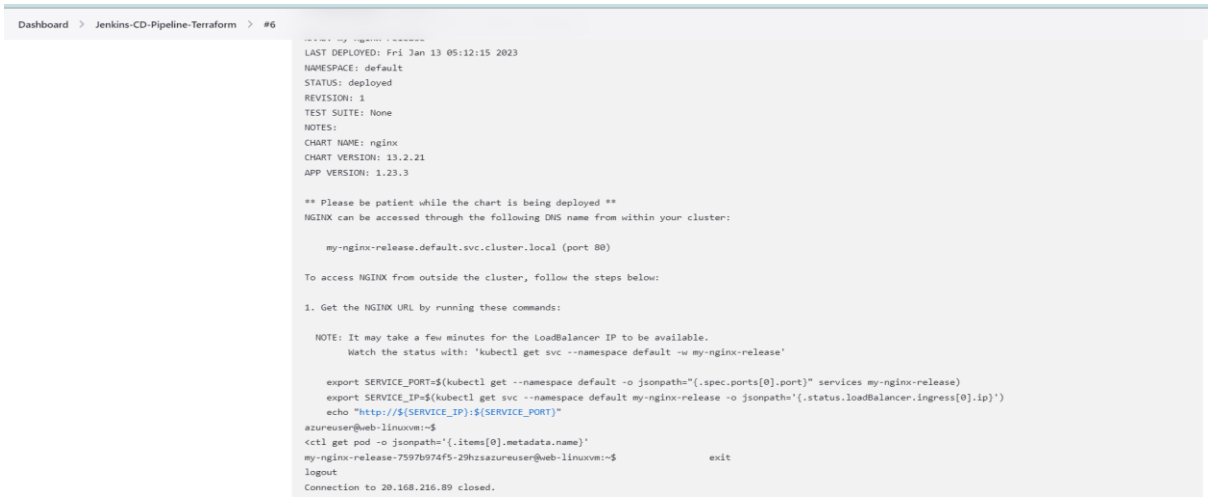
```
        }

        stage('Terraform Plan') {
            when {
                expression{action == "apply"}
            }
            steps{
               script {
                    dir('infrastructure/') {
                        sh 'terraform plan'
                    }
                }
            }
        }
        stage('Terraform Apply') {
            when {
                expression{action == "apply"}
            }
            steps{
               script {
                    dir('infrastructure/') {
                        sh 'terraform apply --auto-approve'
                    }
                }
            }
        }
        stage('SSH Connection to VM') {
            when {
                expression{action == "apply"}
            }
            steps{
                sh'''
                PUBLIC_IP=$(az vm show -d -g $RESOURCE_GROUP
-n $HOST_VM --query publicIps -o tsv)
                ssh -tt -o "StrictHostKeyChecking no"
$USERNAME@$PUBLIC_IP <<'EOT'
                echo "They are executed by: $( whoami )"
                exit
```

```
                '''
            }
        }
        stage("Install Istio"){

            when {
                expression{action == "apply"}
            }
            steps{
                sh '''
                PUBLIC_IP=$(az vm show -d -g $RESOURCE_GROUP
-n $HOST_VM --query publicIps -o tsv)
                ssh -tt -o "StrictHostKeyChecking no"
$USERNAME@$PUBLIC_IP <<'EOT'
                curl -L https://istio.io/downloadIstio | sh
-
                cd istio-1.16.1
                export PATH=$PWD/bin:$PATH
                sudo istioctl install --set profile=demo -y
                sudo kubectl label namespace default istio-
injection=enabled
                exit
                '''
            }
        }
        stage('Installing Helm Chart') {
            when {
                expression{action == "apply"}
            }
            steps{
                sh'''
                PUBLIC_IP=$(az vm show -d -g $RESOURCE_GROUP
-n $HOST_VM --query publicIps -o tsv)
                ssh -tt -o "StrictHostKeyChecking no"
$USERNAME@$PUBLIC_IP <<'EOT'
                sudo helm repo add bitnami
https://charts.bitnami.com/bitnami
```

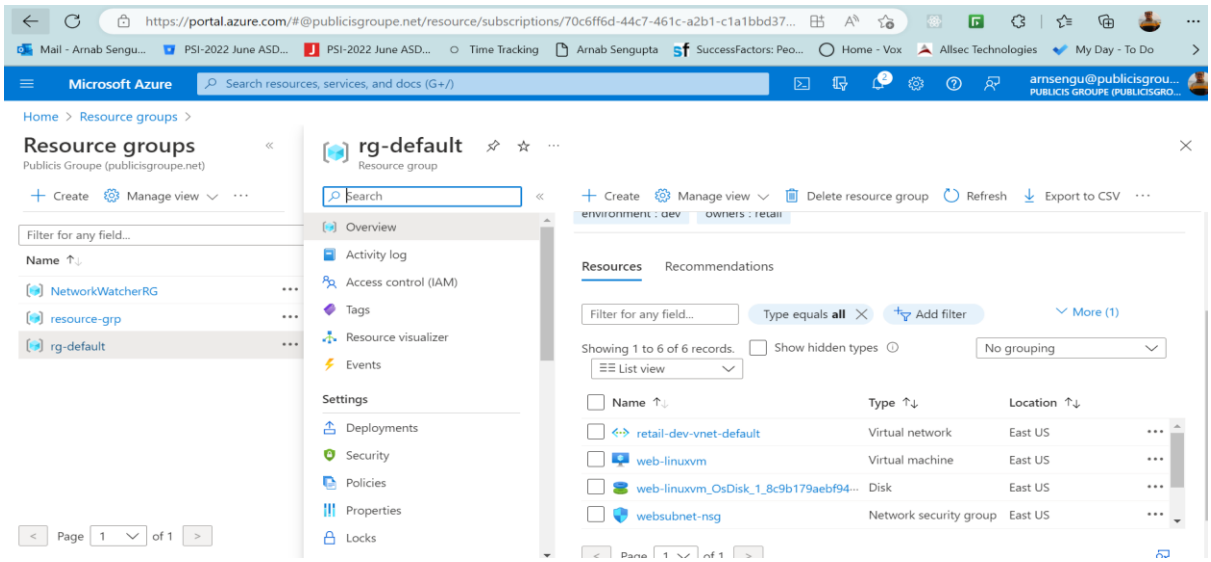
```
                sudo helm install my-nginx-release
bitnami/nginx
                sudo kubectl get pod -o
jsonpath='{.items[0].metadata.name}'
                exit
                '''
            }
        }

        stage("Terraform destroy"){
            when {
                expression{action == "destroy"}
            }
            steps{
                sh 'terraform destroy -auto-approve'
            }
        }

    }


}
```



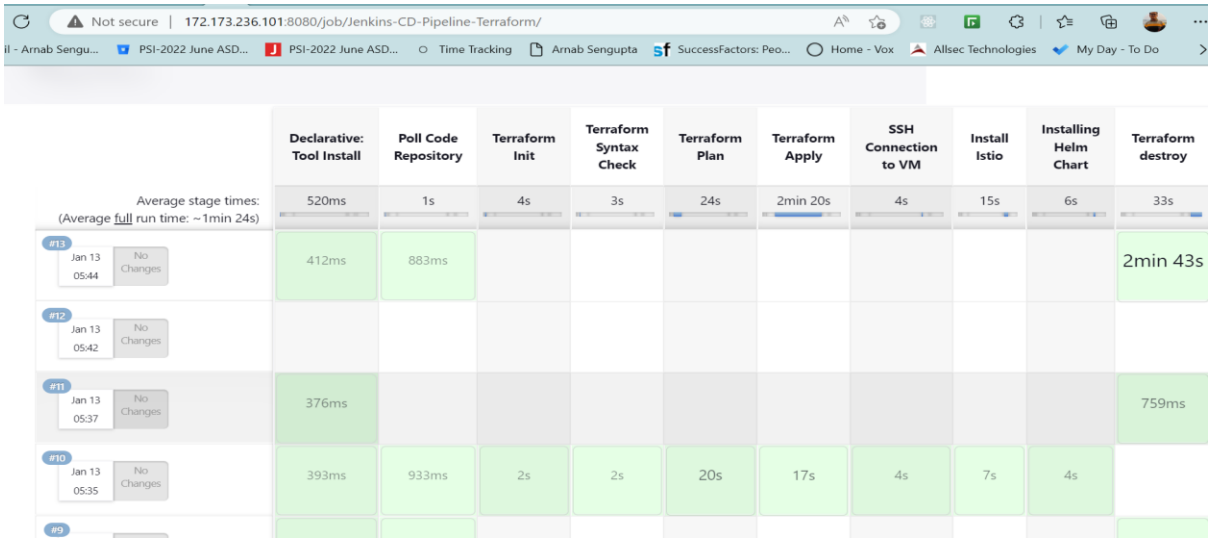
Helm chart deployed on remote server



Created Resource:



Destroying the Infrustructure:



Jenkins CD Pipeline terraform AKS

1. Authenticate Jenkins server to Azure CLI

2. Create terraform manifest file for provisioning resources AKS Cluster

```
root@ansible:~/new/springboot-chat-app# cd aks-cluster-infrastructure/
root@ansible:~/new/springboot-chat-app/aks-cluster-infrastructure# ls
aks-cluster.tf  outputs.tf  provider.tf
root@ansible:~/new/springboot-chat-app/aks-cluster-infrastructure# tree
.
├── aks-cluster.tf
├── outputs.tf
└── provider.tf

0 directories, 3 files
root@ansible:~/new/springboot-chat-app/aks-cluster-infrastructure#
```

3. Create Jenkins pipeline

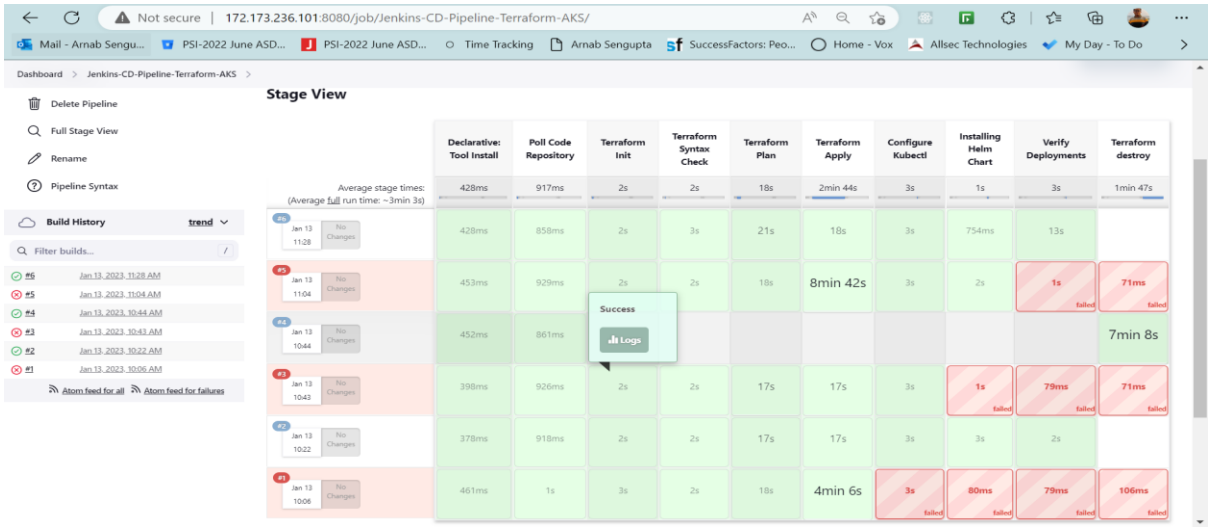
```
pipeline {
    agent any
    tools {
       terraform 'terraform'
    }
    environment {
        CHART_NAME = 'nginx-service'
    }
    stages {
        stage('Poll Code Repository') {
            steps {
                git credentialsId: 'git-key', url:
'git@github.com:arnabcs10/springboot-chat-app.git'
            }
        }
        stage('Terraform Init') {
            when {
                expression{action == "apply"}
            }
            steps{
               script {
                    dir('aks-cluster-infrastructure/') {
                        sh 'terraform init'
                    }
                }
```

```
        }
    }
    stage('Terraform Syntax Check') {
        when {
            expression{action == "apply"}
        }
        steps{
           script {
                dir('aks-cluster-infrastructure/') {
                    sh 'terraform validate'
                }
            }
        }
    }

    stage('Terraform Plan') {
        when {
            expression{action == "apply"}
        }
        steps{
           script {
                dir('aks-cluster-infrastructure/') {
                    sh 'terraform plan'
                }
            }
        }
    }
    stage('Terraform Apply') {
        when {
            expression{action == "apply"}
        }
        steps{
           script {
                dir('aks-cluster-infrastructure/') {
                    sh 'terraform apply --auto-approve'
                }
            }
        }
```

```
        }
        stage('Configure Kubectl') {
            when {
                expression{action == "apply"}
            }
            steps {
                script {
                    dir('aks-cluster-infrastructure/') {
                        sh'''
                        az aks get-credentials --resource-
group $(terraform output -raw resource_group_name) --name
$(terraform output -raw kubernetes_cluster_name)
                        kubectl get nodes
                        '''
                    }
                }
            }

        }
        stage('Installing Helm Chart') {
            when {
                expression{action == "apply"}
            }
            steps{
                sh'''
                helm list
                # helm repo add bitnami
https://charts.bitnami.com/bitnami
                # helm install $CHART_NAME
bitnami/nginx
                '''
            }
        }
        stage('Verify Deployments') {
            when {
                expression{action == "apply"}
            }
            steps{
```

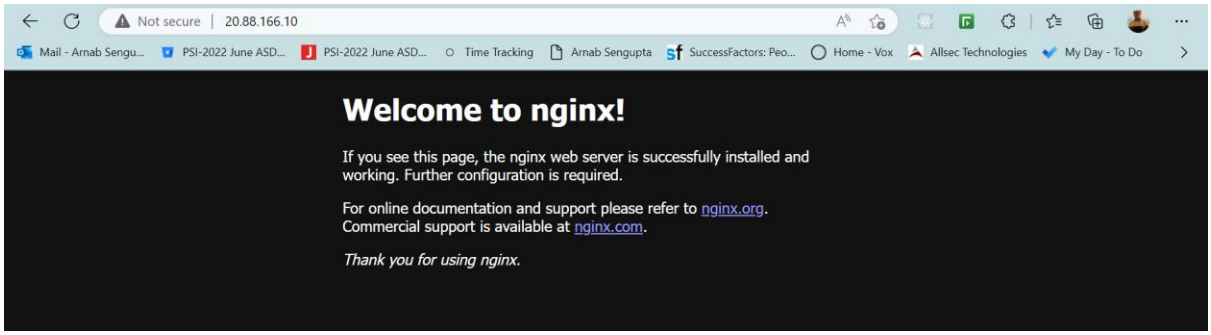
```
                sh'''
                kubectl get pods
                kubectl get deployments
                kubectl get services

                echo "Waiting for end point..."
                sleep 10
                EXTERNAL_IP=$(kubectl get svc $CHART_NAME -o
yaml | grep -oP '(?<=ip: )[0-9].+')
                echo 'End point ready:' && echo $EXTERNAL_IP
                echo "URL: http://$EXTERNAL_IP"
                '''
            }
        }

        stage("Terraform destroy"){
            when {
                expression{action == "destroy"}
            }
            steps {
                script {
                    dir('aks-cluster-infrastructure/') {
                        sh 'terraform destroy --auto-
approve'
                    }
                }
            }
        }
    }
}
```

Build Pipeline:

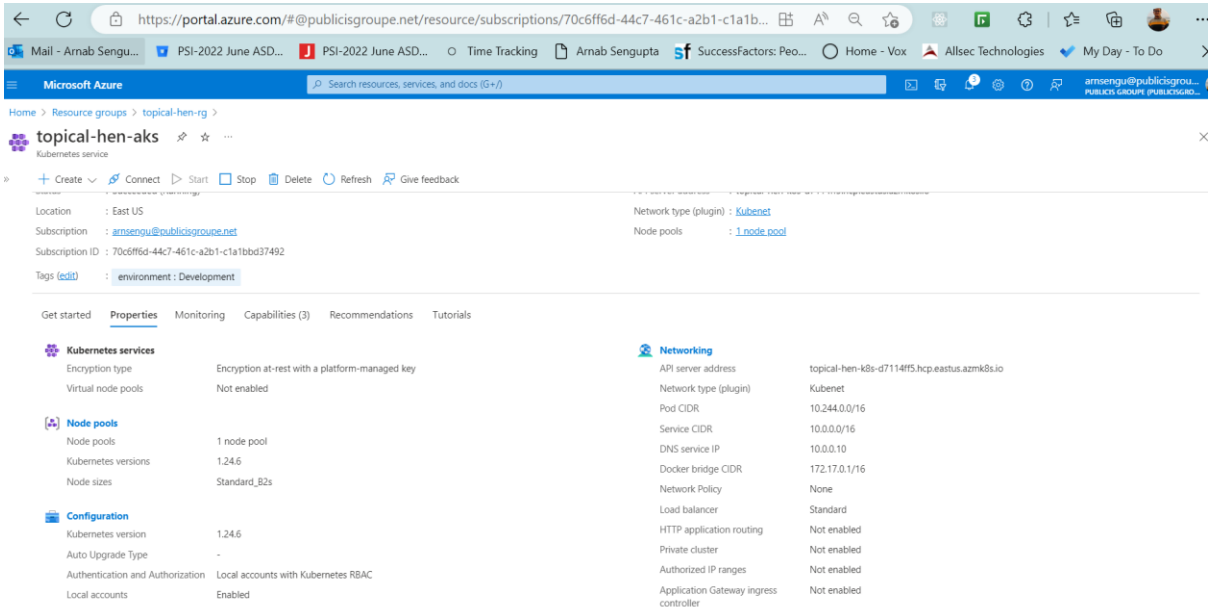


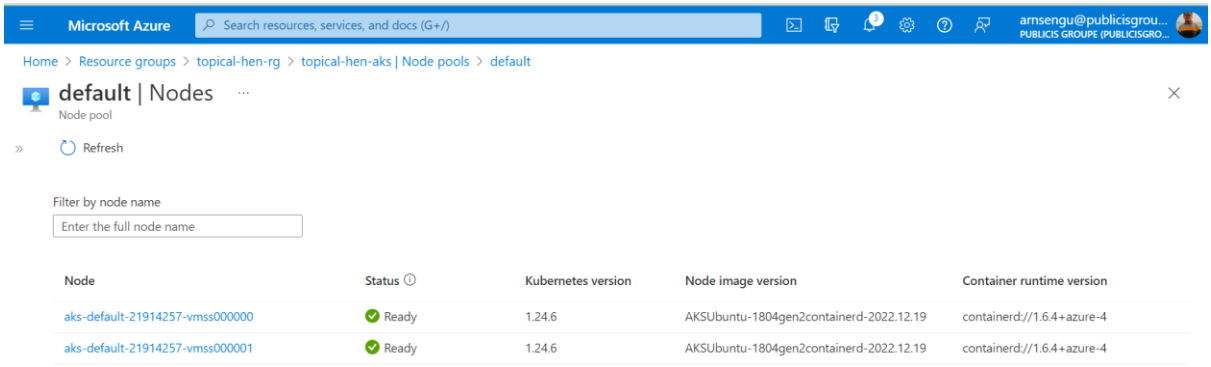

```
[Pipeline] sh
+ terraform output -raw resource_group_name
+ terraform output -raw kubernetes_cluster_name
+ az aks get-credentials --resource-group topical-hen-rg --name topical-hen-aks
WARNING: Merged "topical-hen-aks" as current context in /var/lib/jenkins/.kube/config
+ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-default-21914257-vmss000000   Ready    agent   18m   v1.24.6
aks-default-21914257-vmss000001   Ready    agent   21m   v1.24.6
```

```
+ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
nginx-service-75647c85d-j8p5z   1/1     Running   0          16m
+ kubectl get deployments
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
nginx-service   1/1     1            1           16m
+ kubectl get services
NAME            TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
kubernetes      ClusterIP      10.0.0.1      <none>         443/TCP        22m
nginx-service   LoadBalancer   10.0.4.249    20.88.166.10   80:31085/TCP   16m
+ echo Waiting for end point...
Waiting for end point...
+ sleep 10
+ kubectl get svc nginx-service -o yaml
+ grep -oP (?<=ip: )[0-9].+
+ EXTERNAL_IP=20.88.166.10
+ echo End point ready:
End point ready:
+ echo 20.88.166.10
20.88.166.10
+ echo URL: http://20.88.166.10
URL: http://20.88.166.10
```



Created AKS Cluster:



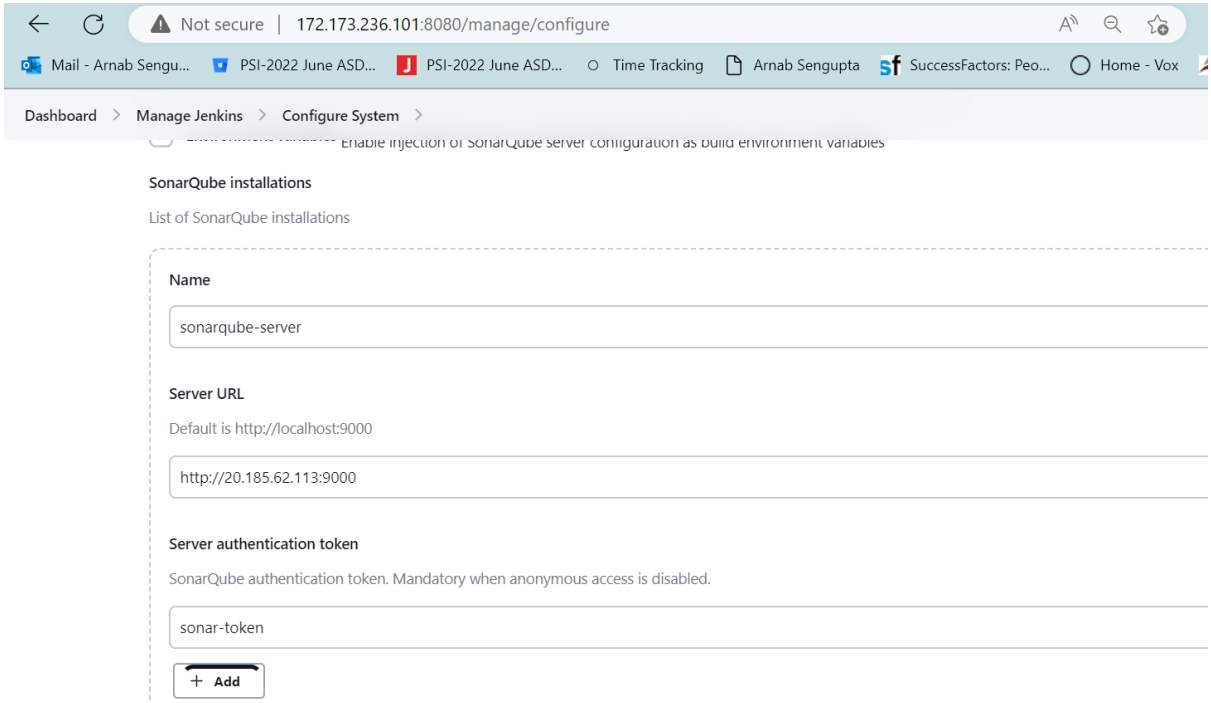


Destroying AKS Cluster:

Jenkins: sonar pipeline. Include SonarQube quality gate enforcement in Jenkins pipeline to be added.

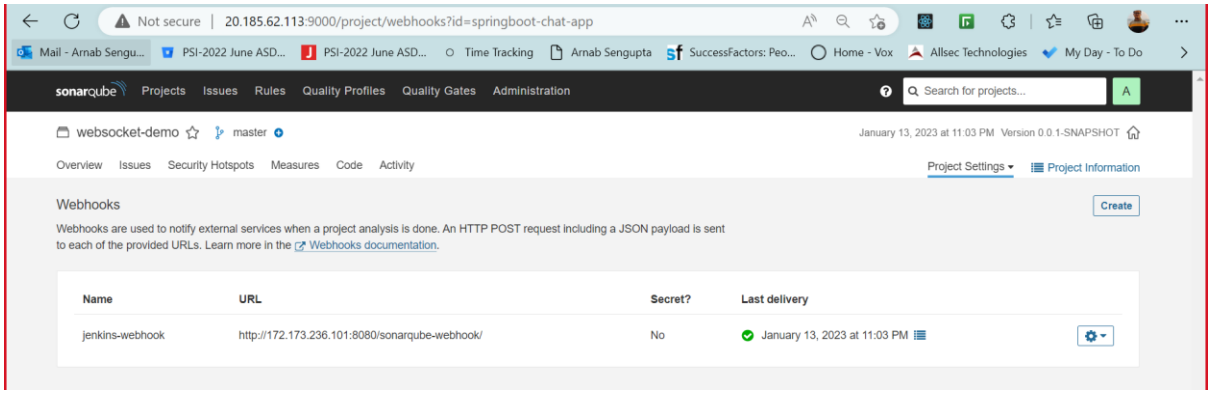
1. Configured Jenkins setting for SonarQube quality gate

Added sonar server URL and authentication token

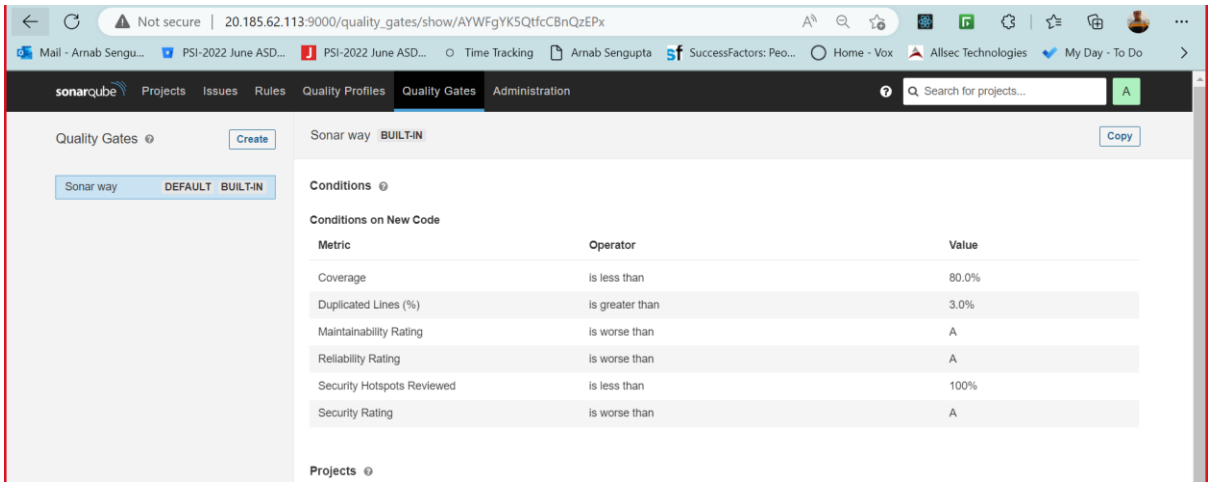


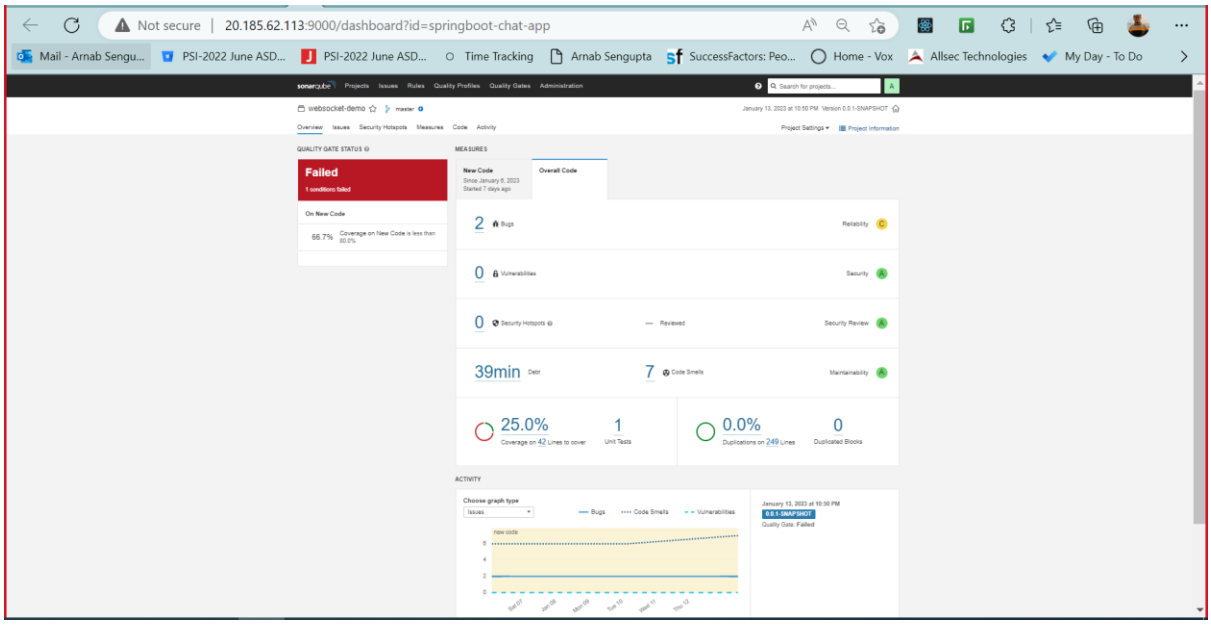
Modified Jenkins pipeline stages for quality gate:

2. Added SonarQube webhook for Jenkins

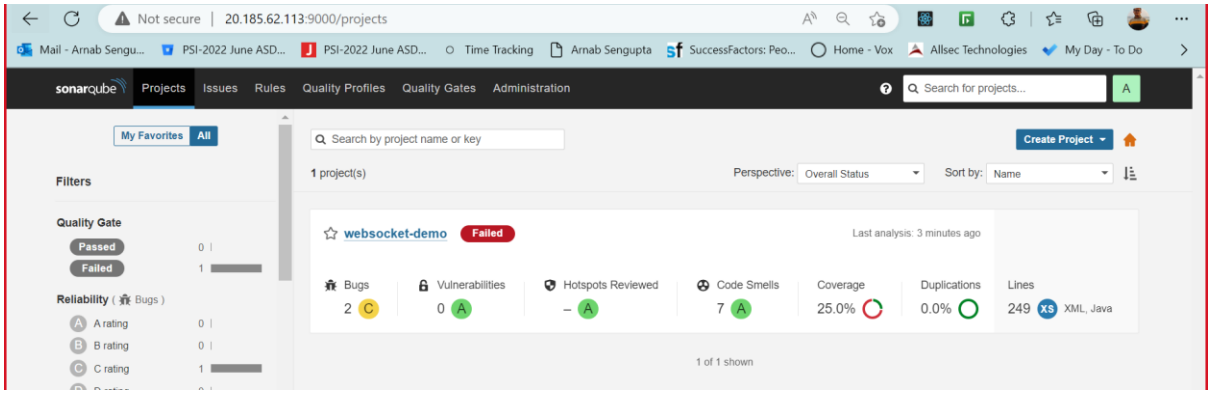


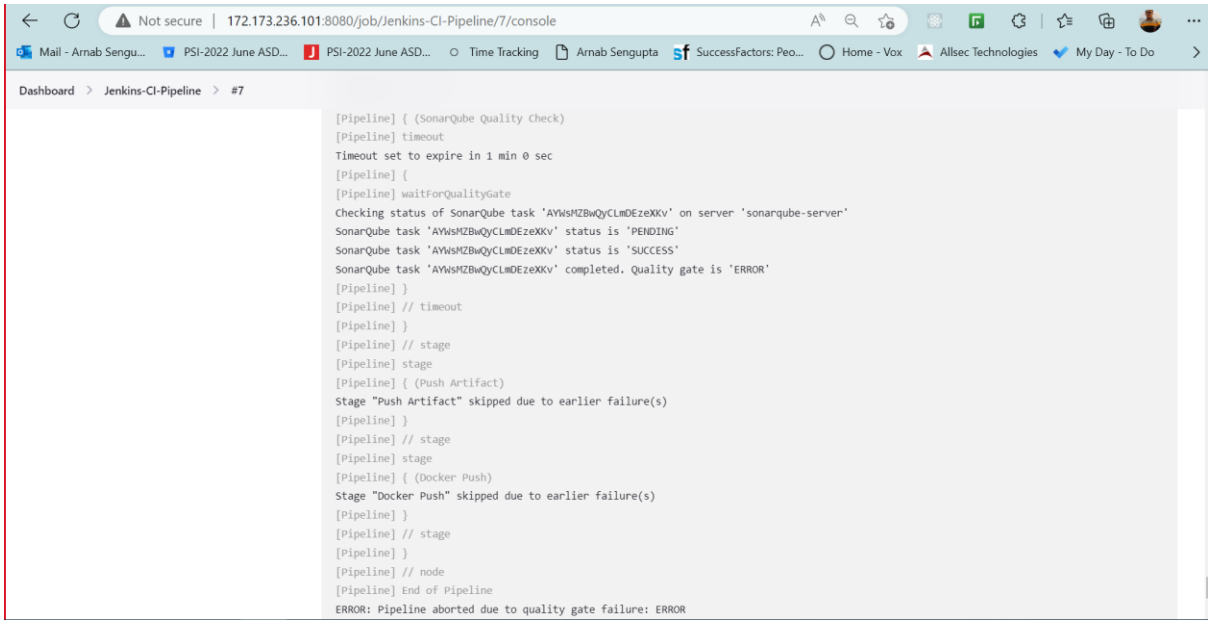
Quality gate: Built In





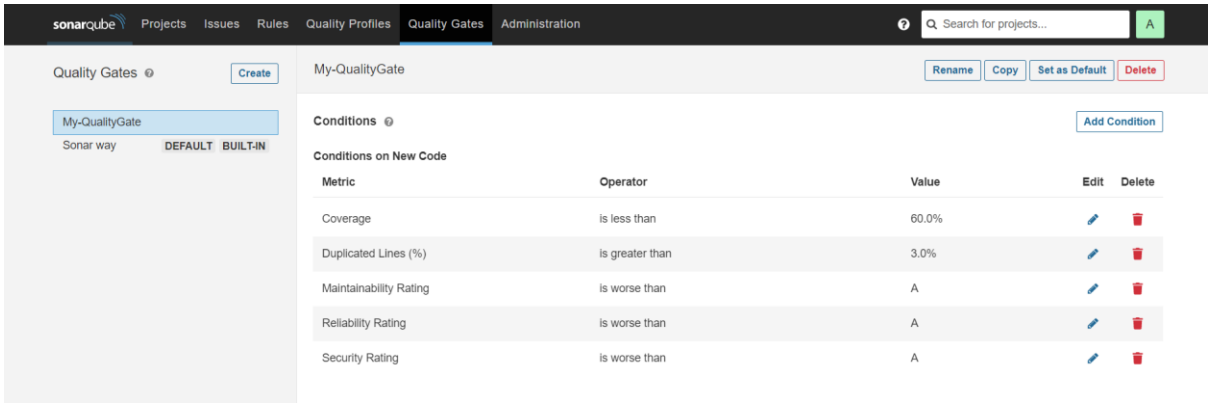
Quality Check failed



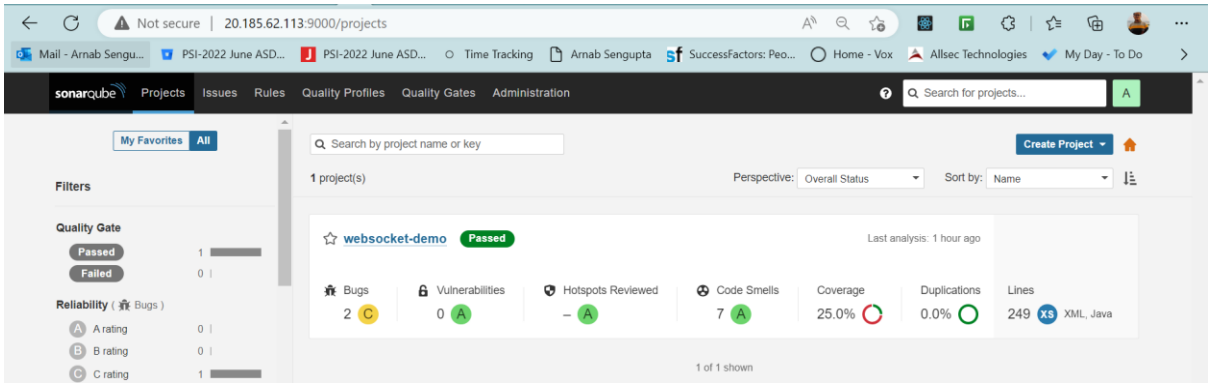


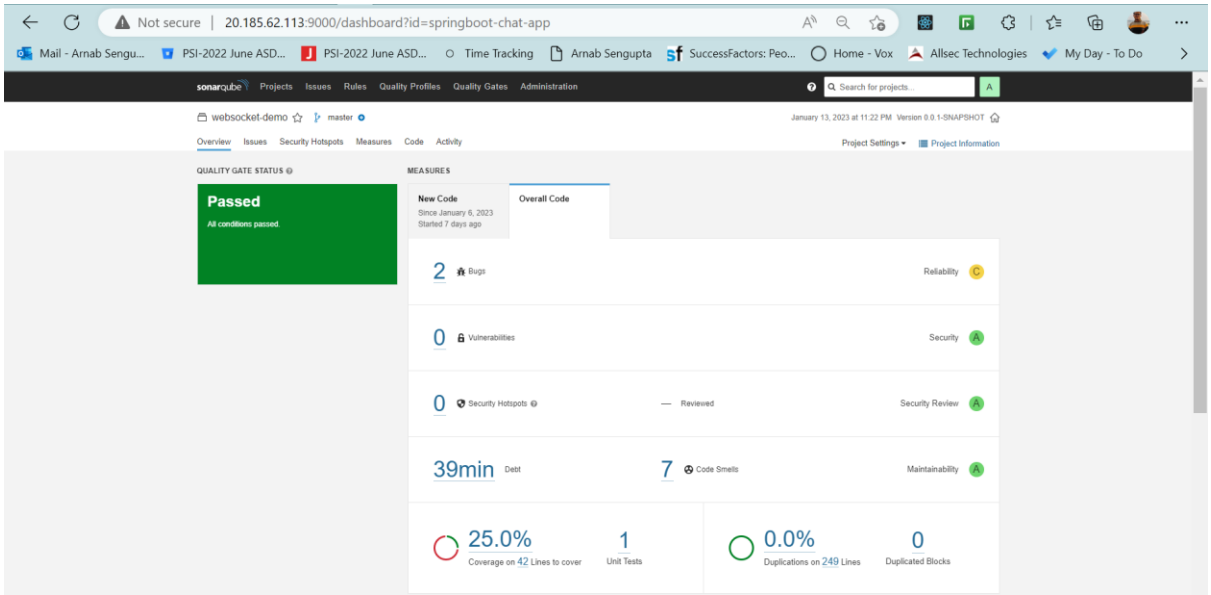

```
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (SonarQube Quality Check)
[Pipeline] timeout
Timeout set to expire in 1 min 0 sec
[Pipeline] {
[Pipeline] waitForQualityGate
Checking status of SonarQube task 'AYWsQ25IQyCLmDEzeXK7' on server 'sonarqube-server'
SonarQube task 'AYWsQ25IQyCLmDEzeXK7' status is 'IN_PROGRESS'
SonarQube task 'AYWsQ25IQyCLmDEzeXK7' status is 'SUCCESS'
SonarQube task 'AYWsQ25IQyCLmDEzeXK7' completed. Quality gate is 'OK'
[Pipeline] }
[Pipeline] // timeout
```

3. Added new quality gate to pass build pipeline:



Build Passed:





Jenkins Output: Build Success

```
[Pipeline] { (SonarQube Quality Check)
[Pipeline] timeout
Timeout set to expire in 1 min 0 sec
[Pipeline] {
[Pipeline] waitForQualityGate
Checking status of SonarQube task 'AYWsQ25IQyCLmDEzeXK7' on server 'sonarqube-server'
SonarQube task 'AYWsQ25IQyCLmDEzeXK7' status is 'IN_PROGRESS'
SonarQube task 'AYWsQ25IQyCLmDEzeXK7' status is 'SUCCESS'
SonarQube task 'AYWsQ25IQyCLmDEzeXK7' completed. Quality gate is 'OK'
[Pipeline] }
```