**CAPSTONE PROJECT**

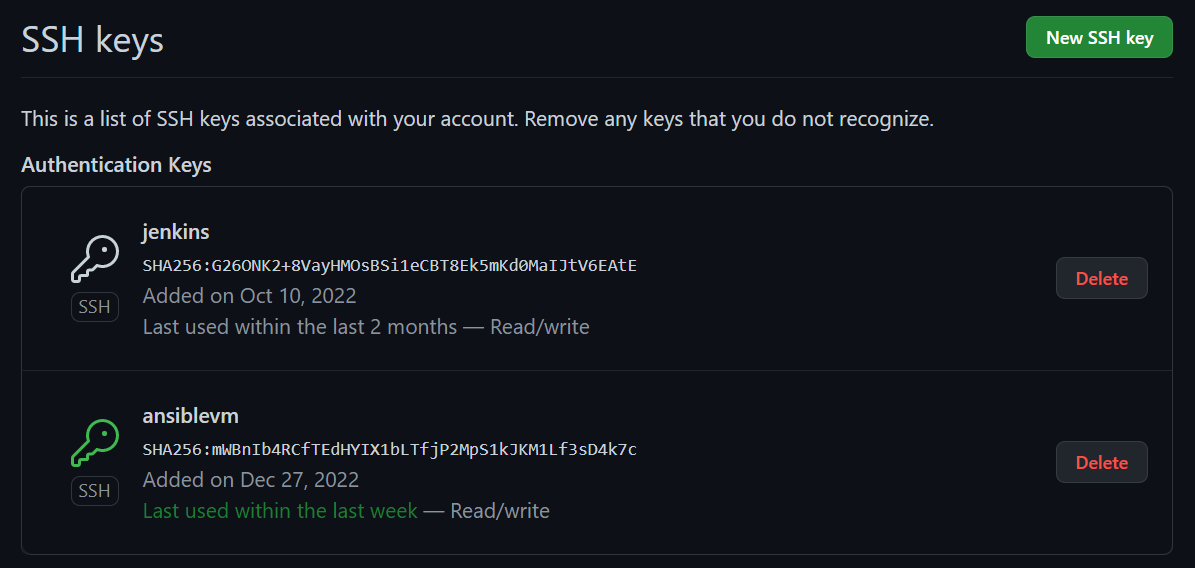
**Building an entire CICD pipeline in Jenkins**

**Jenkins\_CI Pipeline**

1. Generate ssh key upload in github
2. Create a Jenkins pipeline – integrate the private repo with Jenkins that will be first stage
3. Second stage compile the project using maven but exclude test
4. Third stage do code testing using unittest checkstyle codecoverage
5. Add all this stages in the pipeline
6. Then integrate Jenkins with SonarQube. Provide the difference between sonarqube testing and the previous testing framework
7. Finally upload the jar file inside nexus
8. While uploading jar file pipeline syntax generator provide you with static value.
9. Autmatically find out the jar file and upload the same

**Jenkins CI-Pipeline cont…**

1. Convert the application into an container based image
2. Upload the application into nexus repo



**-** Pipeline for above steps

pipeline {

    agent any

    stages {

        stage('poll scm') {

            steps {

                git credentialsId: 'git', url: 'git@github.com:asatyam78/spring-boot-chat-app.git'

            }

        }

        stage('mvn build') {

            steps {

                sh 'mvn -B -DskipTest clean package'

            }

        }

        stage('mvn test') {

            steps {

                sh 'mvn test'

                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('sonar') {

            steps {

                sh 'mvn clean verify sonar:sonar \

                  -Dsonar.projectKey=springboot-chat-app \

                  -Dsonar.host.url=http://20.172.204.197:9000 \

                  -Dsonar.login=sqp\_bf0e65730853209c57c087c5f98a8eb5bfafcba2'

            }

        }

        stage('nexus') {

            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: 'jar']], credentialsId: 'nexusCred', groupId: pom.artifactId, nexusUrl: '172.173.154.5:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-snapshots', version: pom.version

            }

        }

        stage('docker build') {

            steps {

                script {

                        withDockerRegistry(credentialsId: 'nexusCred', url: 'http://172.173.154.5:8085') {

                        sh 'mvn compile jib:build -Djib.allowInsecureRegistries=true -DsendCredentialsOverHttp'

                       }

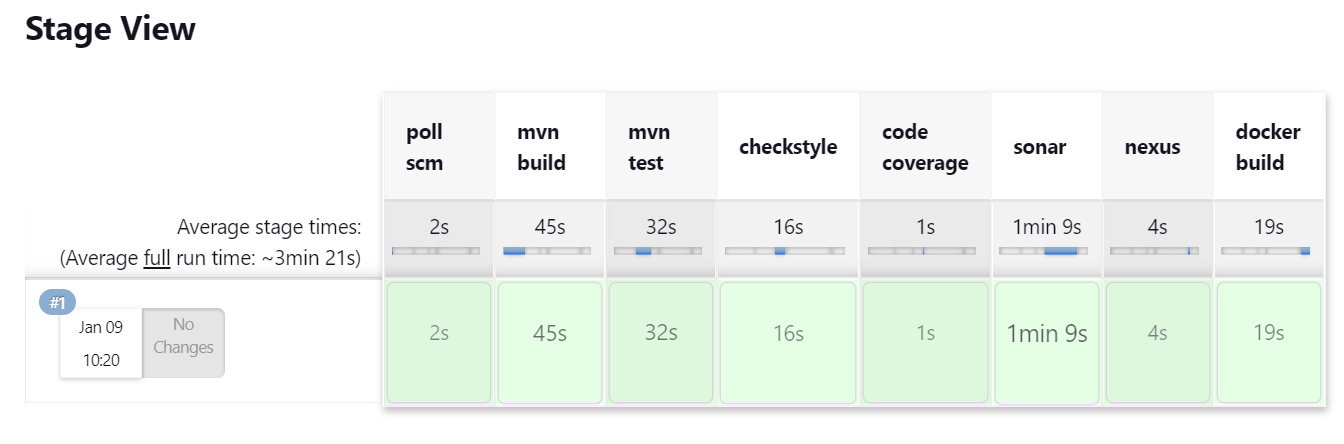
                }

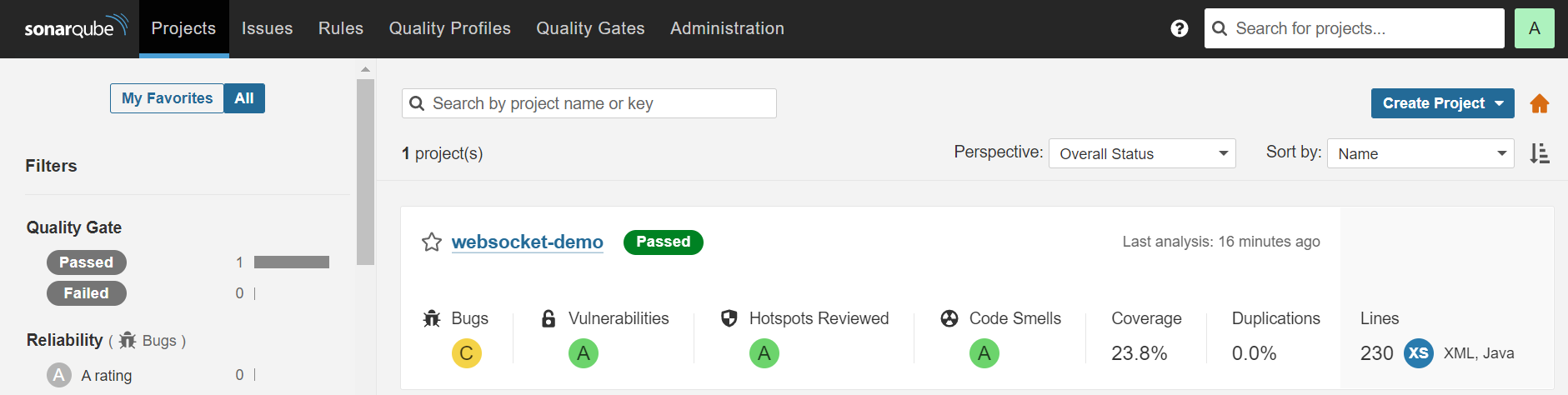
            }

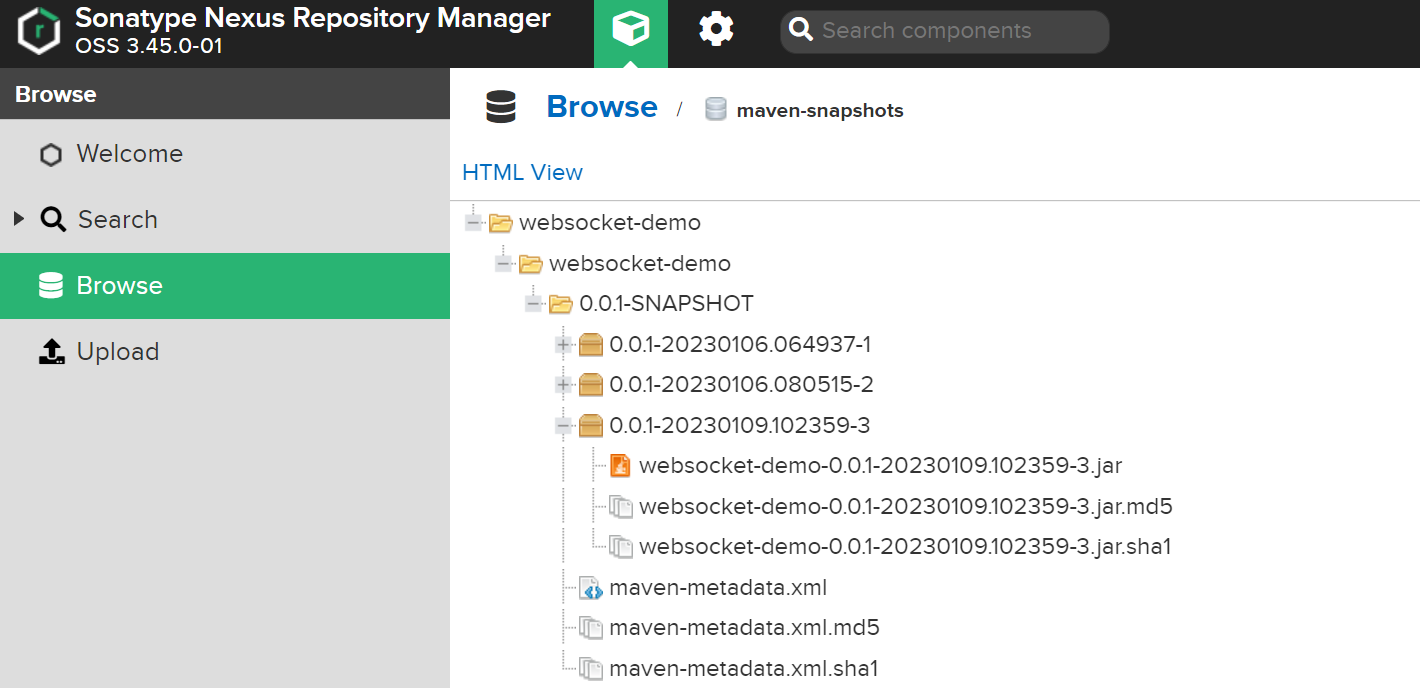
        }

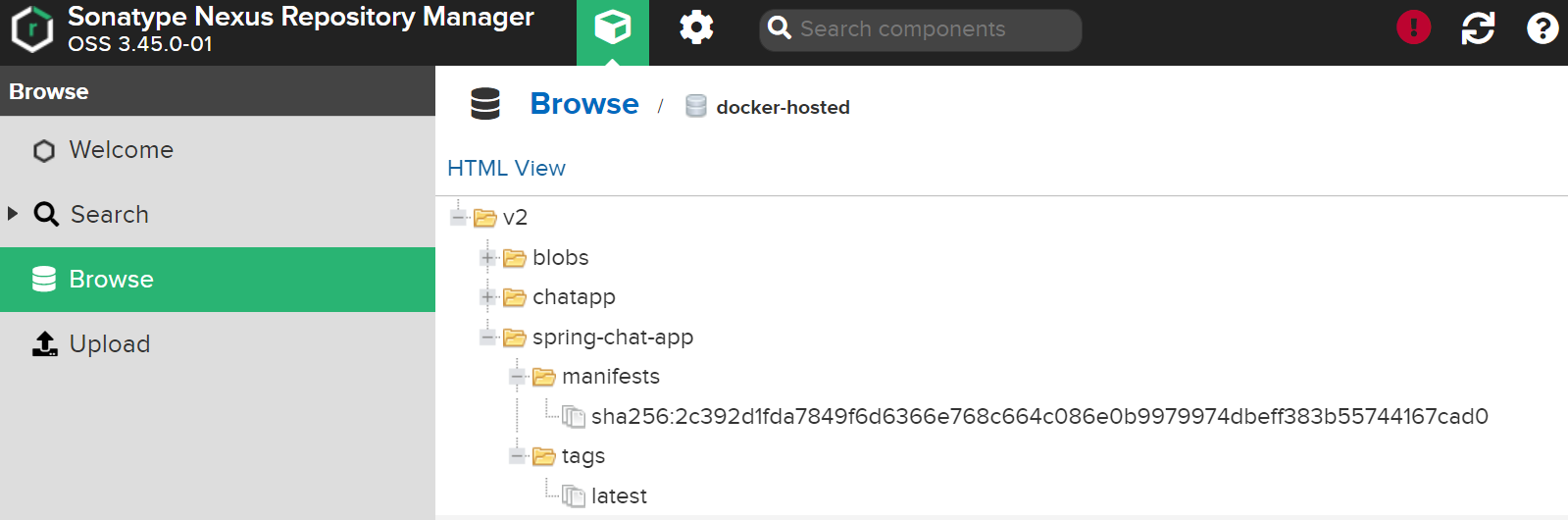
    }

}









**Jenkins CD-pipeline**

1. Create an ansible playbook which install docker in Ubuntu system
2. Create another ansible playbook which will deploy the private image from nexus repo

**-** ansible playbook for installing docker

---

- name: Install docker

  hosts: client

  become: true

  tasks:

    - name: Update 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: ensure repository key is installed

      apt\_key:

        url: https://download.docker.com/linux/ubuntu/gpg

        state: present

    - name: ensure docker registry is available

      apt\_repository:

        repo: 'deb https://download.docker.com/linux/ubuntu bionic stable'

        state: present

    - name: ensure docker and dependencies are installed

      apt:

        name: docker-ce

        update\_cache: yes

    - name: Install Docker Module for Python

      pip:

        name: docker

    - name: ensure docker can use insecure registries

      copy:

        content: |-

          {

            "insecure-registries" : ["172.173.154.5:8085"]

          }

        dest: /etc/docker/daemon.json

    - name: restart service docker

      service:

        name: docker

        state: restarted

**-** ansible playbook for deploying image from private repository

---

- name: Deploy Docker Image

  hosts: client

  become: true

  vars\_files:

  - nexus-cred.yml

  tasks:

    - name: Log into private registry and force re-authorization

      docker\_login:

        registry: 172.173.154.5:8085

        username: "{{ username }}"

        password: "{{ password }}"

        reauthorize: yes

    - name: Run Docker Conatiner

      docker\_container:

        name: spring-chat-app

        image: 172.173.154.5:8085/spring-chat-app

        state: started

        pull: yes

        ports:

        - "8082:8082"

    - name: Connect to docker container 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: 5

**-** Pipeline for above steps

pipeline {

    agent any

    stages {

        stage('git integration') {

            steps {

                git credentialsId: 'git', url: 'git@github.com:asatyam78/spring-boot-chat-app.git'

            }

        }

        stage ('docker install using ansible'){

            steps {

                ansiblePlaybook credentialsId: 'git', disableHostKeyChecking: true, inventory: 'ansible/dev.inv', playbook: 'ansible/docker.yml'

            }

        }

        stage ('deploy private image using ansible'){

            steps {

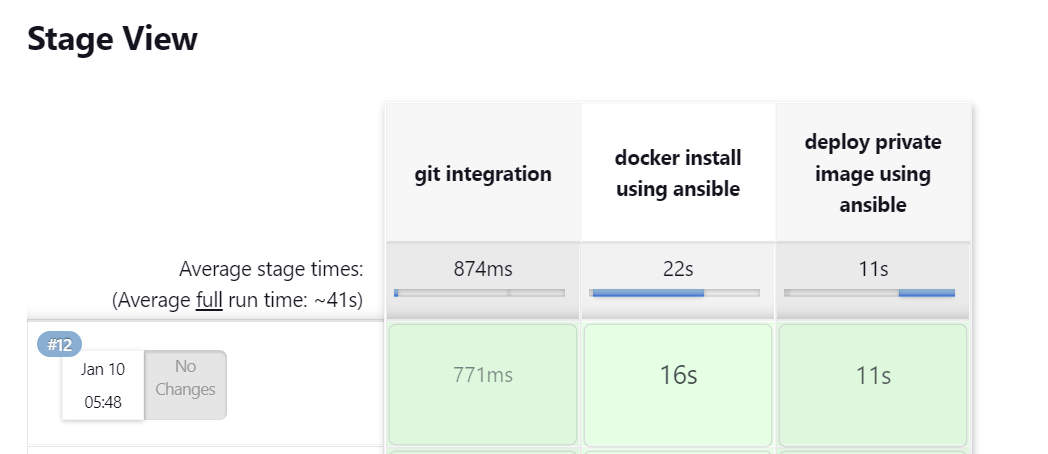
                ansiblePlaybook credentialsId: 'git', disableHostKeyChecking: true, inventory: 'ansible/dev.inv', playbook: 'ansible/deploy-image.yml', vaultCredentialsId: 'ansible-vault'

            }

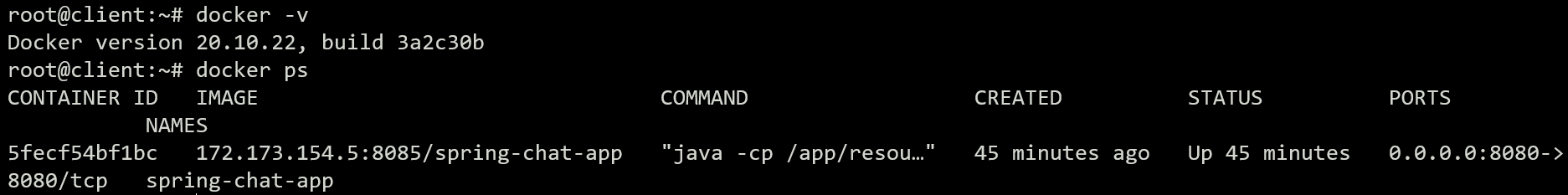
        }

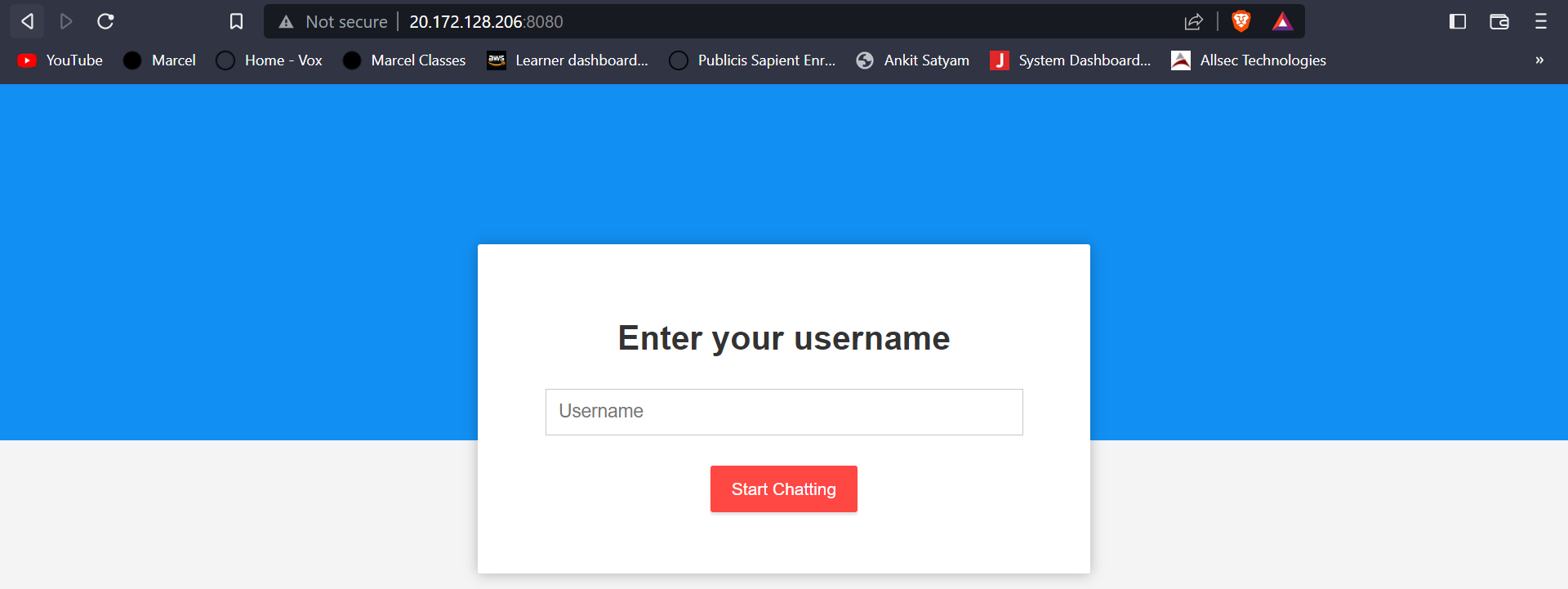
    }

}



**-** Console output on client machine





**Jenkisn CD pipeline with tomcat.**

1. Create an ansible playbook which will install tomcat
2. Deploy the jar file from nexus to tomcat using ansible.

**-** ansible file for installing tomcat

---

- name: Install Tomcat to host

  hosts: client

  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 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

     template:

        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: 5

**-** ansible playbook for deploying war file

---

- name: Deploy war file in tomcat

  hosts: client

  become: true

  vars\_files:

  - nexus-cred.yml

  tasks:

    - name: Install lxml

      pip:

       name: lxml

    - name: "Download war file"

      maven\_artifact:

        group\_id: websocket-demo

        artifact\_id: websocket-demo

        version: 0.0.1-SNAPSHOT

        extension: war

        repository\_url: "http://172.173.154.5:8081/repository/maven-snapshots"

        username: "{{ username }}"

        password: "{{ password }}"

        dest: "/opt/tomcat/webapps/spring-boot-chat-app.war"

        mode: '0644'

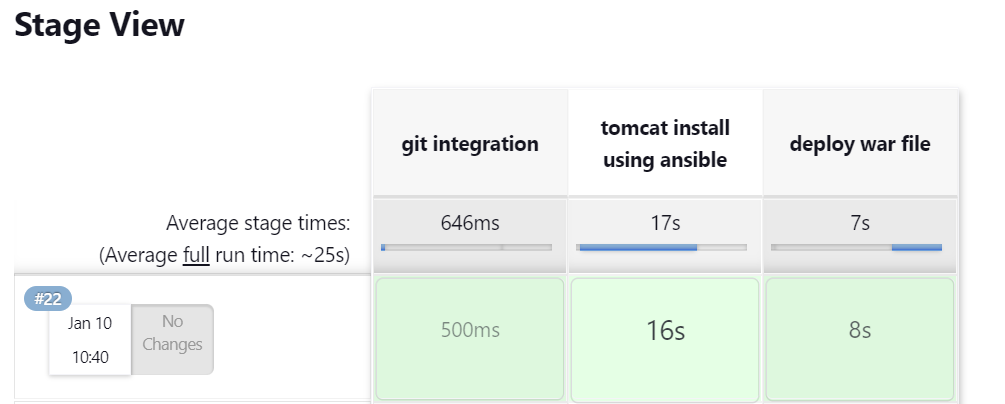
    - name: Enable the tomcat service and start

      systemd:

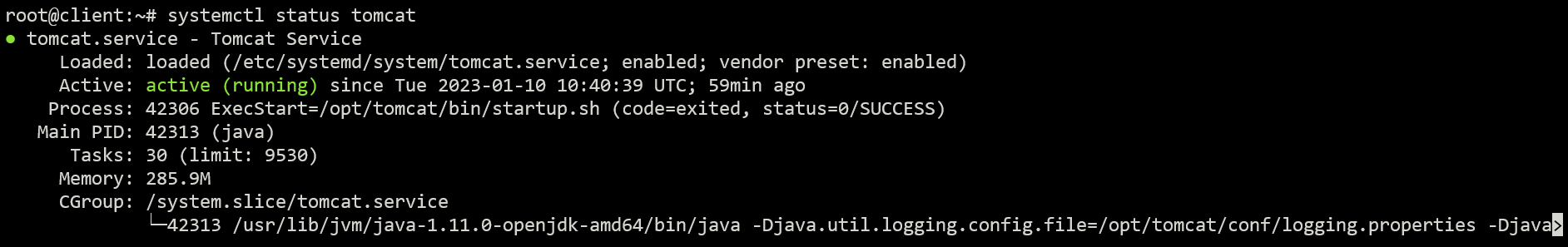
        name: tomcat

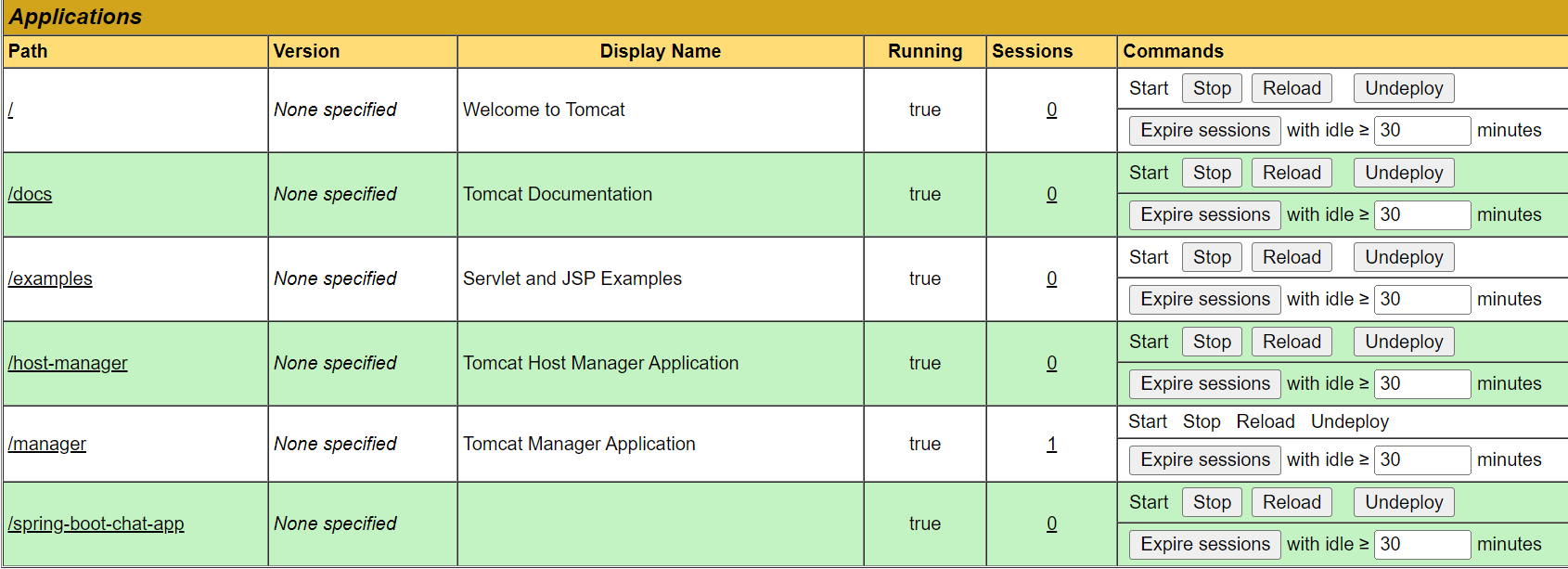
        enabled: yes

        state: restarted



**-** Console output on client machine





Jenkins CD Pipeline terraform

1. Create a service principal in azure.
2. Use the terraform manifest to build a server in azure
3. Once build use the null provisioner to install kind kubernetes cluster.

Doc: for installing kind

<https://kind.sigs.k8s.io/docs/user/quick-start/>

1. Deploy your helm chart using Jenkins
2. Deploy ingress or istio controller and put your application behind the same.

**🡪** All manifest files are in azure-terraform directory.

**-** Pipeline for above steps

pipeline {

    agent any

    tools {

        terraform 'terraform'

    }

    stages {

        stage('Poll Code Repository') {

            steps {

                git credentialsId: 'git', url: 'git@github.com:asatyam78/spring-boot-chat-app.git'

            }

        }

        stage('terraform format') {

            when {

                expression {action == 'apply'}

            }

            steps {

                script {

                    dir('azure-terraform/') {

                        sh 'terraform fmt'

                    }

                }

            }

        }

        stage('terraform initialize') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('azure-terraform/') {

                        sh 'terraform init'

                    }

                }

            }

        }

        stage('terraform validate') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('azure-terraform/') {

                        sh 'terraform validate'

                    }

                }

            }

        }

        stage('terraform plan') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('azure-terraform/') {

                        sh 'terraform plan'

                    }

                }

            }

        }

        stage('terraform apply') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('azure-terraform/') {

                        sh 'terraform apply --auto-approve'

                    }

                }

            }

        }

        stage('Install helm') {

            when {

                expression {action == 'apply'}

            }

            steps{

                sh'''

                    export PUBLIC\_IP=$(az vm show -d -g rg -n websubnet-web-vm --query publicIps -o tsv)

                    ssh -tt -o "StrictHostKeyChecking no" azureuser@$PUBLIC\_IP <<EOT

                    whoami

                    sudo helm repo add bitnami https://charts.bitnami.com/bitnami

                    sudo helm install my-nginx-release bitnami/nginx

                    exit

                '''

            }

        }

        stage("Install Istio"){

            when {

                expression{action == "apply"}

            }

            steps{

                sh '''

                    export PUBLIC\_IP=$(az vm show -d -g rg -n websubnet-web-vm --query publicIps -o tsv)

                    ssh -tt -o "StrictHostKeyChecking no" azureuser@$PUBLIC\_IP <<EOT

                    whoami

                    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

                    sudo kubectl get deployment

                    sudo kubectl get svc

                    exit

                '''

            }

        }

        stage('terraform destroy') {

            when {

                expression {action == 'destroy'}

            }

            steps{

                script {

                    dir('azure-terraform/') {

                        sh 'terraform destroy --auto-approve'

                    }

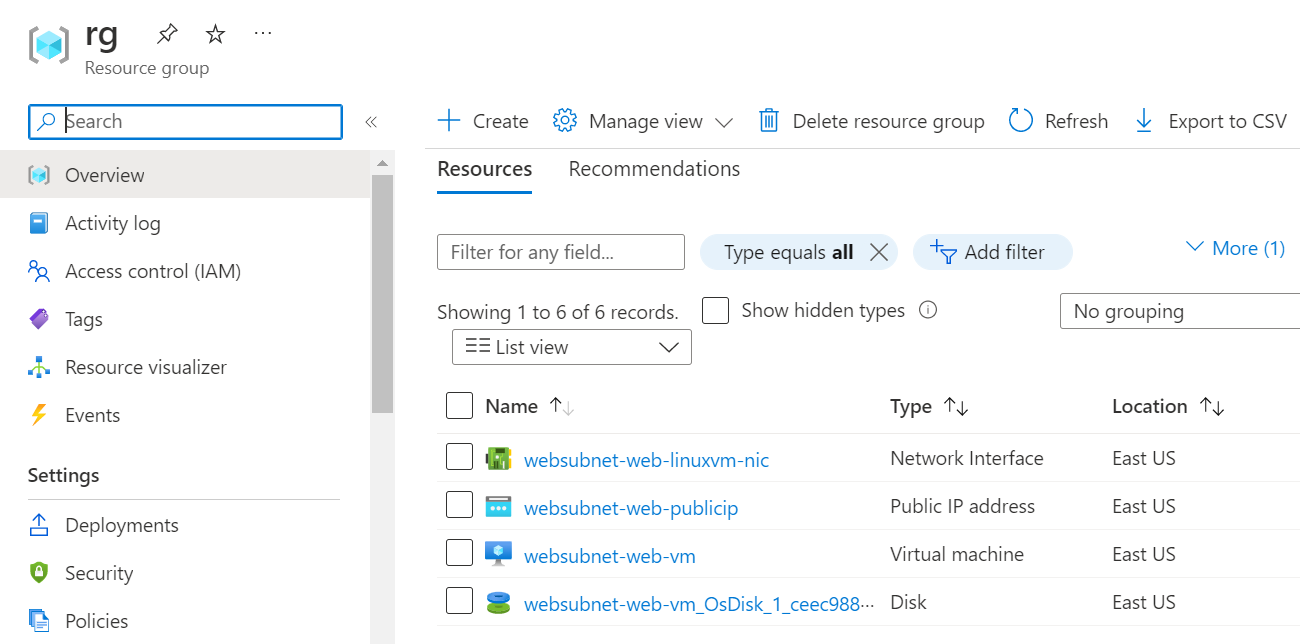
                }

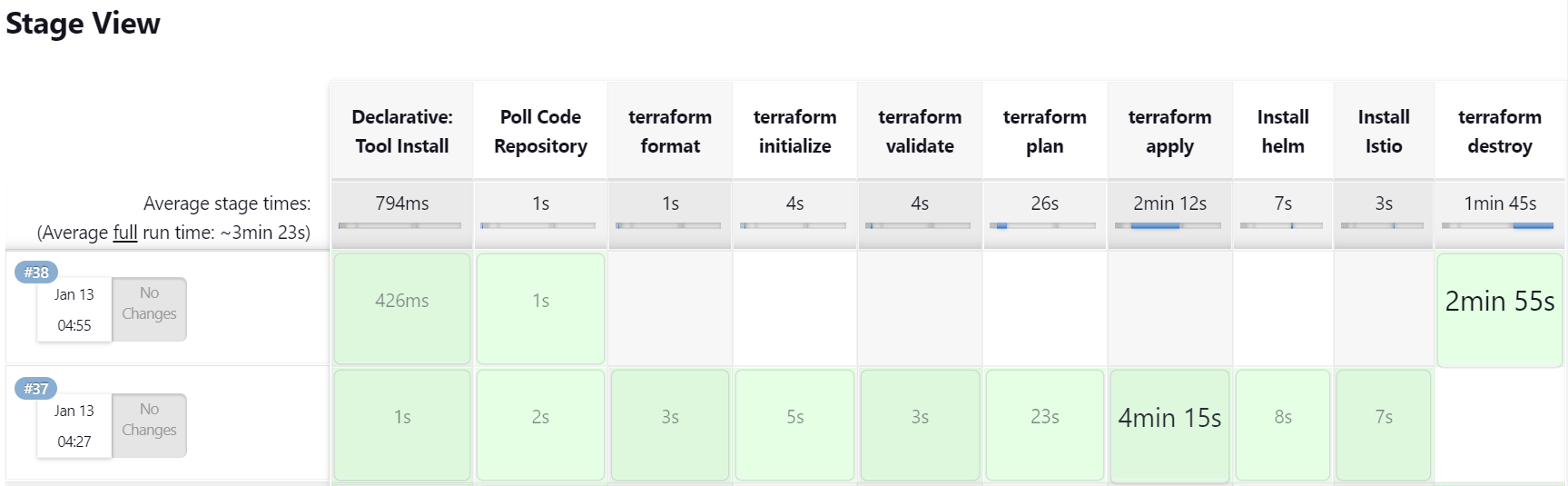
            }

        }

    }

}





* use terraform to provision AKS and associated resources in Azure, then sample created helm created(in previous module/exercise) to deploy AKS cluster, use Jenkins as orchestrator, rest all is fine.

**🡪** All manifest files are in aks-terraform directory.

**-** Pipeline for above steps

pipeline {

    agent any

    tools {

        terraform 'terraform'

    }

    stages {

        stage('Poll Code Repository') {

            steps {

                git credentialsId: 'git', url: 'git@github.com:asatyam78/spring-boot-chat-app.git'

            }

        }

        stage('terraform format') {

            when {

                expression {action == 'apply'}

            }

            steps {

                script {

                    dir('aks-terraform/') {

                        sh 'terraform fmt'

                    }

                }

            }

        }

        stage('terraform initialize') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('aks-terraform/') {

                        sh 'terraform init'

                    }

                }

            }

        }

        stage('terraform validate') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('aks-terraform/') {

                        sh 'terraform validate'

                    }

                }

            }

        }

        stage('terraform plan') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('aks-terraform/') {

                        sh 'terraform plan'

                    }

                }

            }

        }

        stage('terraform apply') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('aks-terraform/') {

                        sh 'terraform apply --auto-approve'

                    }

                }

            }

        }

        stage('configure kubectl') {

            when {

                expression {action == 'apply'}

            }

            steps{

                script {

                    dir('aks-terraform/') {

                        sh 'az aks get-credentials --resource-group $(terraform output -raw resource\_group\_name) --name $(terraform output -raw kubernetes\_cluster\_name)'

                    }

                }

            }

        }

        stage('deploy helm') {

            when {

                expression {action == 'apply'}

            }

            steps{

                sh'''

                    helm repo add bitnami https://charts.bitnami.com/bitnami

                    helm install my-nginx-release bitnami/nginx

                '''

            }

        }

        stage("check deployment"){

            when {

                expression{action == "apply"}

            }

            steps{

                sh '''

                    sleep 1m

                    kubectl get pods

                    kubectl get svc

                    curl $(kubectl get svc my-nginx-release -o yaml | grep -oP '(?<=ip: )[0-9].+')

                '''

            }

        }

        stage('terraform destroy') {

            when {

                expression {action == 'destroy'}

            }

            steps{

                script {

                    dir('aks-terraform/') {

                        sh 'terraform destroy --auto-approve'

                    }

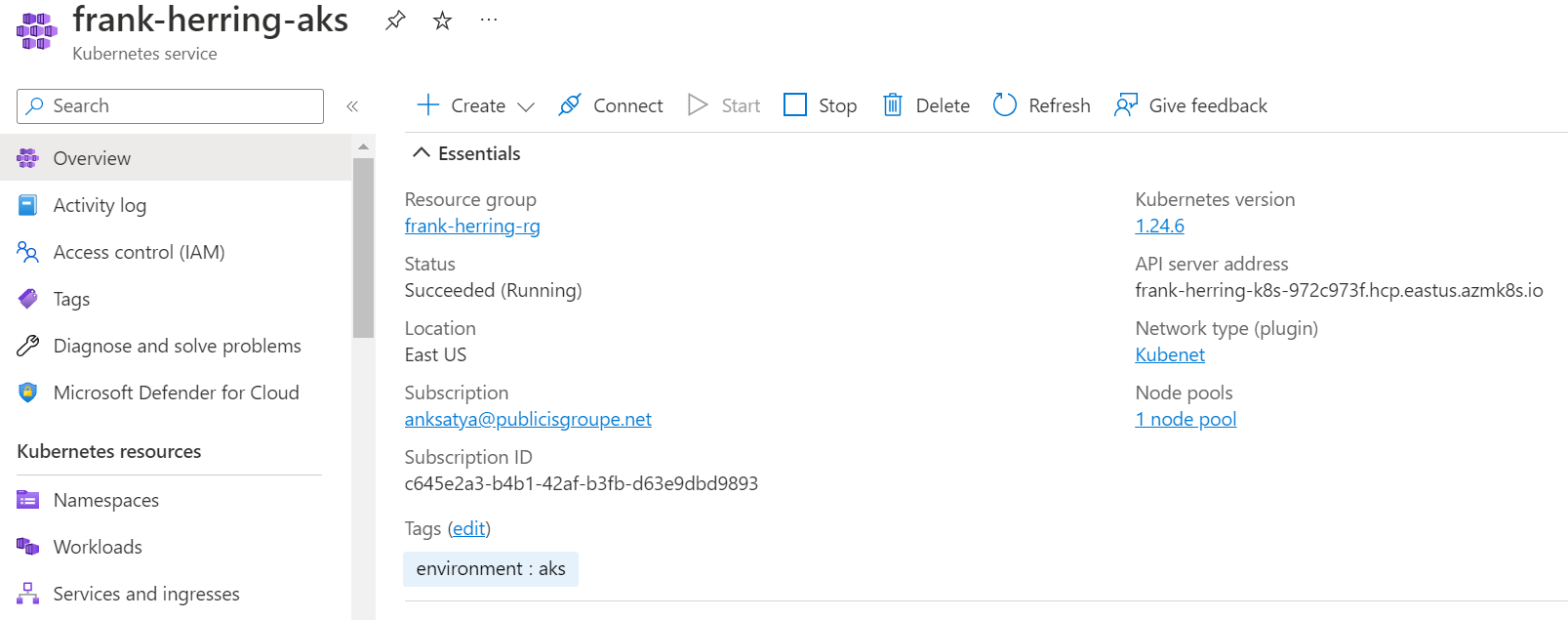
                }

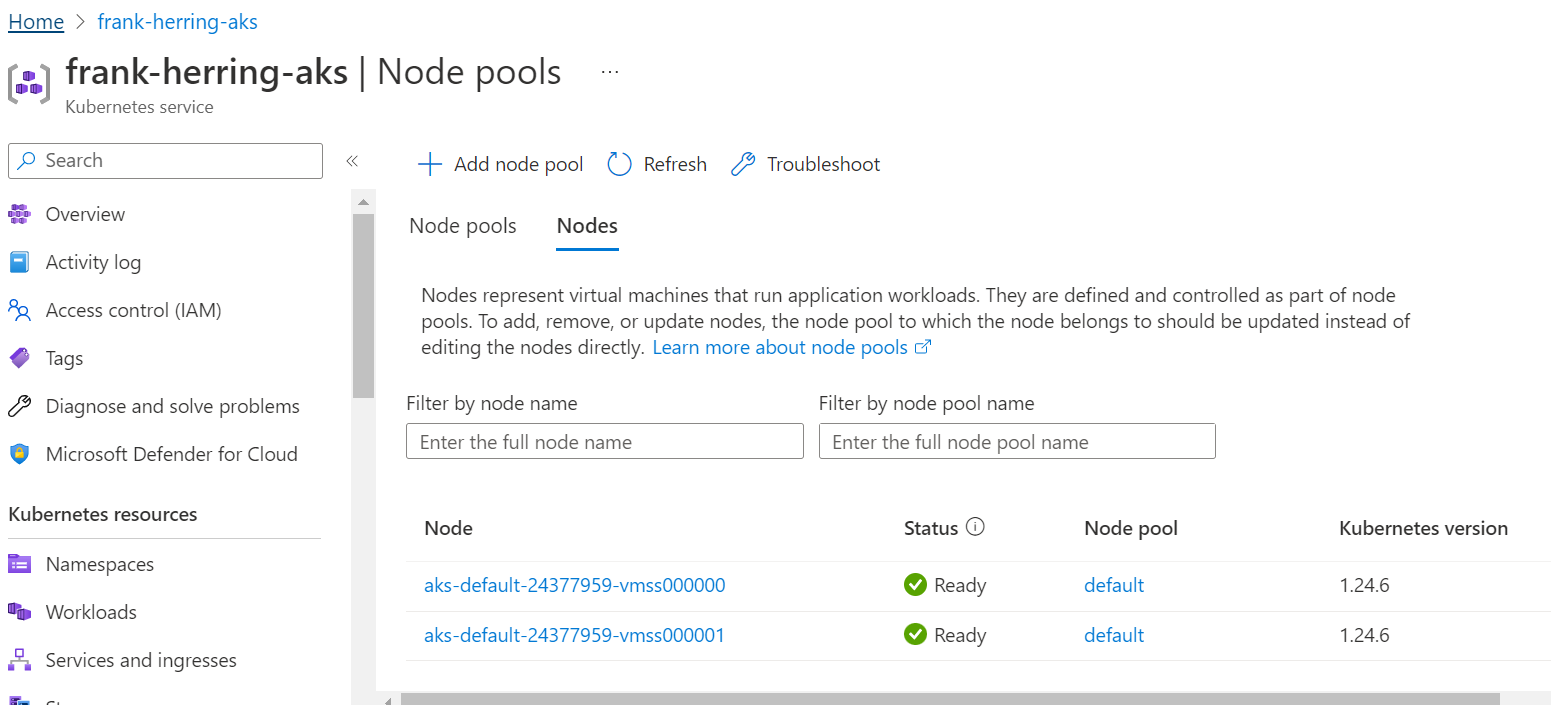
            }

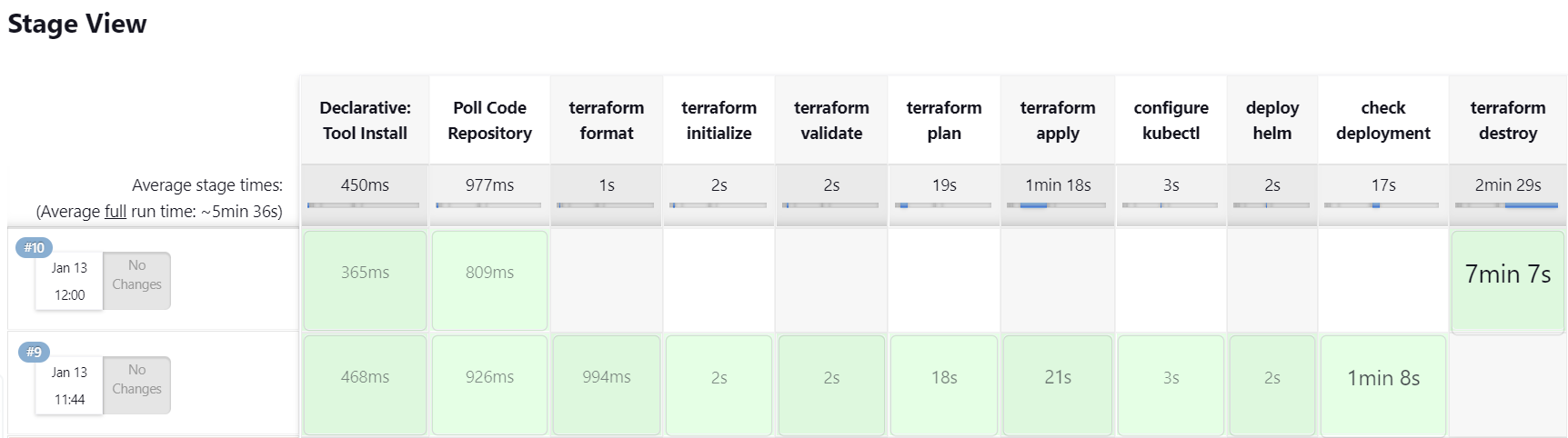
        }

    }

}







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

**-** pipeline for sonarqube quality gate

stage('Sonarqube Analysis'){

            steps {

                withSonarQubeEnv('sonarqube') {

                    sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=springboot-chat-app'

                }

            }

        }

        stage("SonarQube Quality Check"){

            steps{

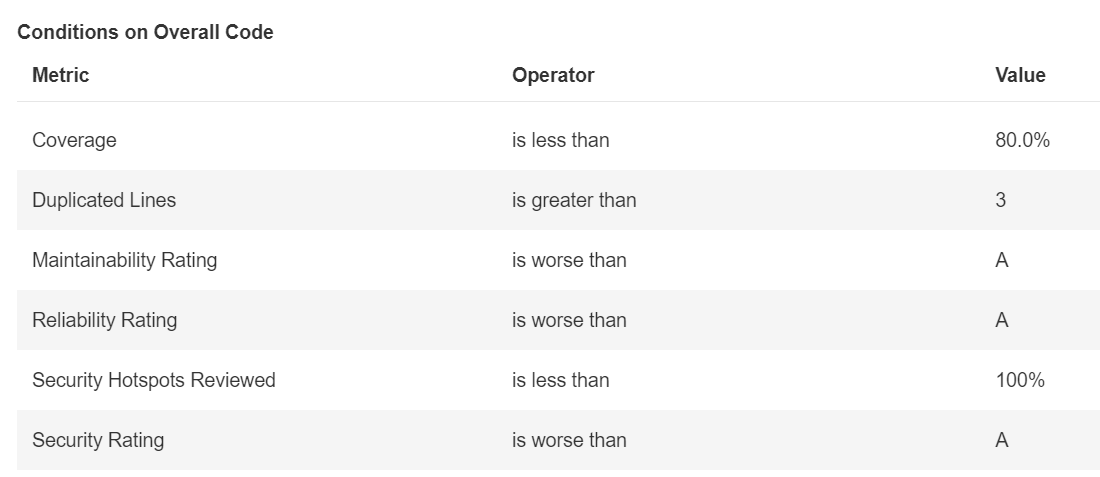
                timeout(time:1, unit:'MINUTES'){

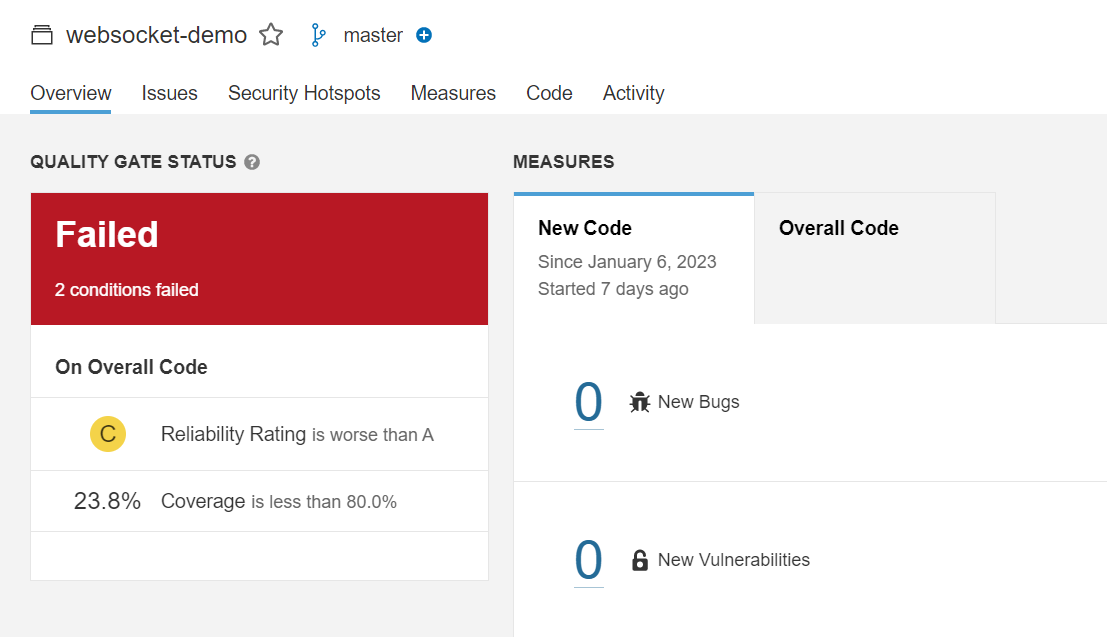
                    waitForQualityGate abortPipeline: true, credentialsId: 'sonar'

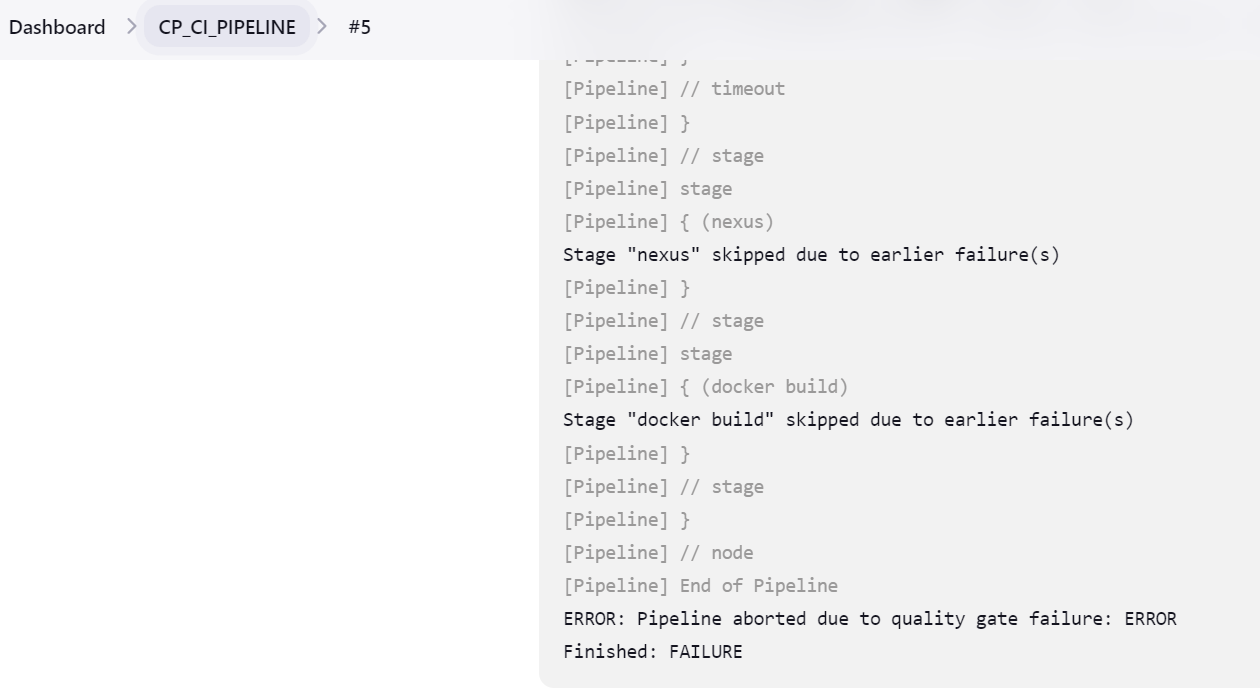
                }

            }

        }







**🡪** Above build failed due to quality gate failure.

**🡪** Changed quality gates

