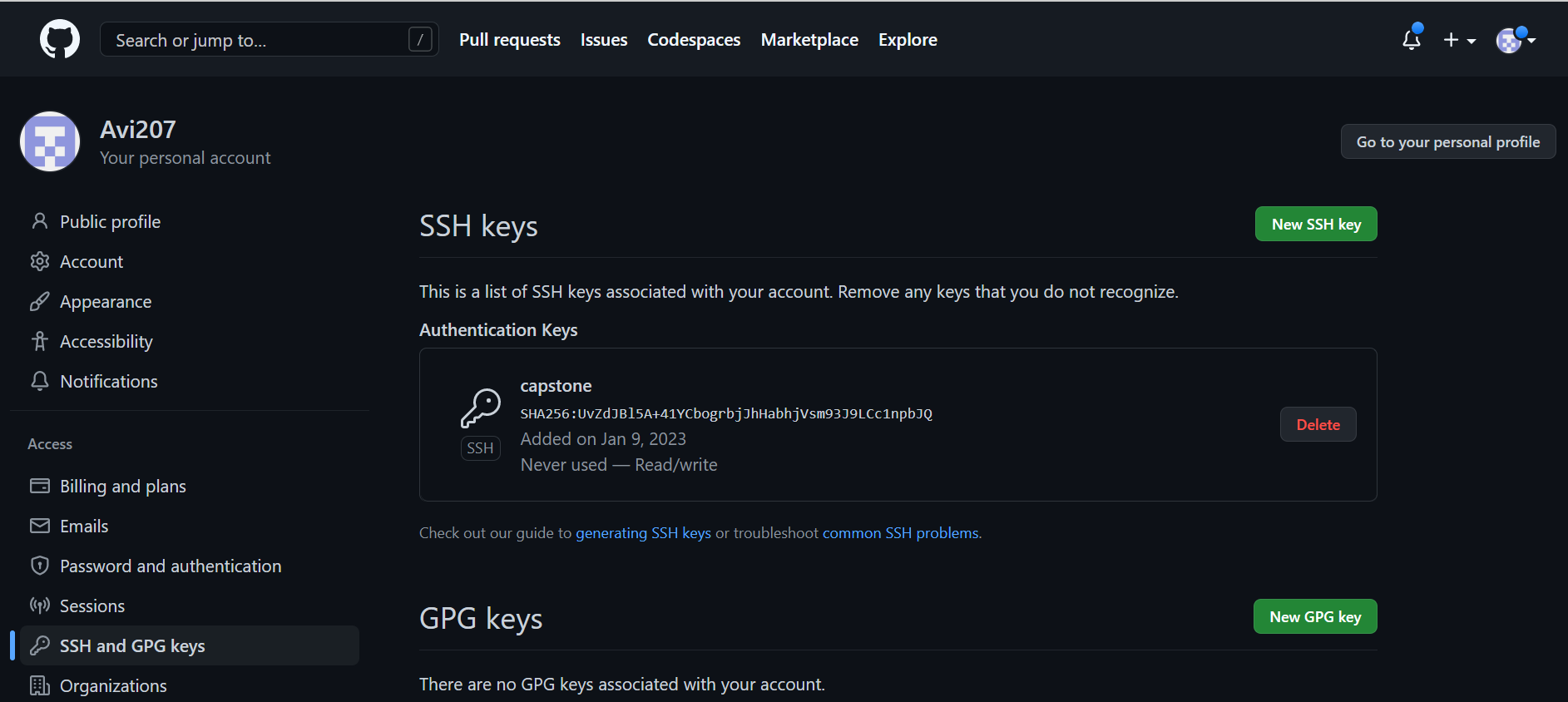
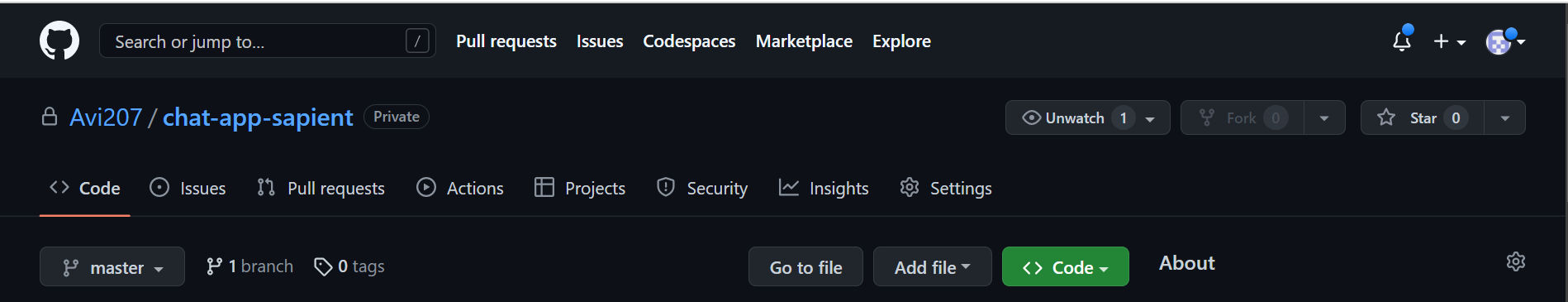
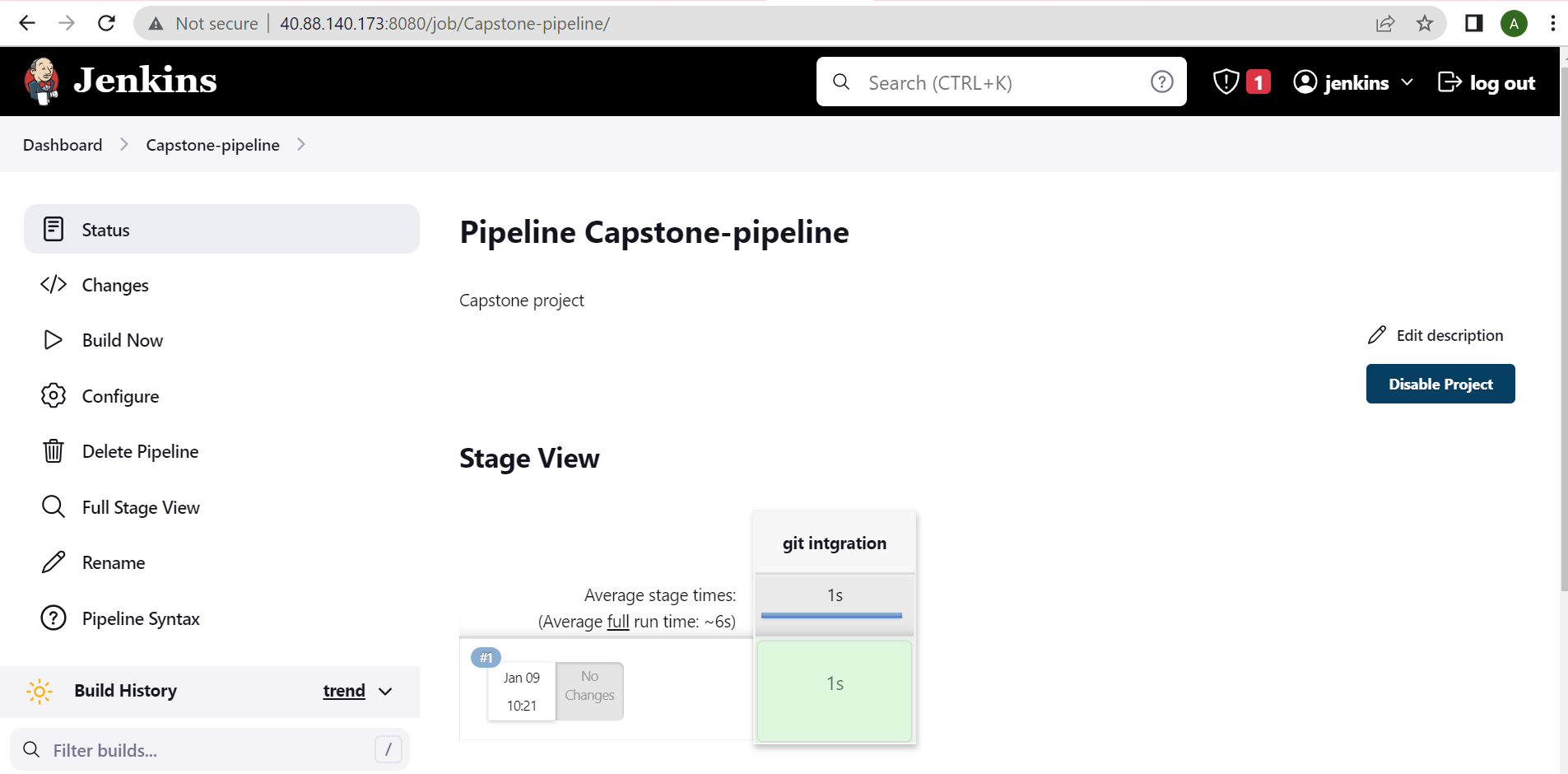
**Capstone Project**

1. **Building an entire CICD pipeline in Jenkins**
2. Generate ssh key upload in github



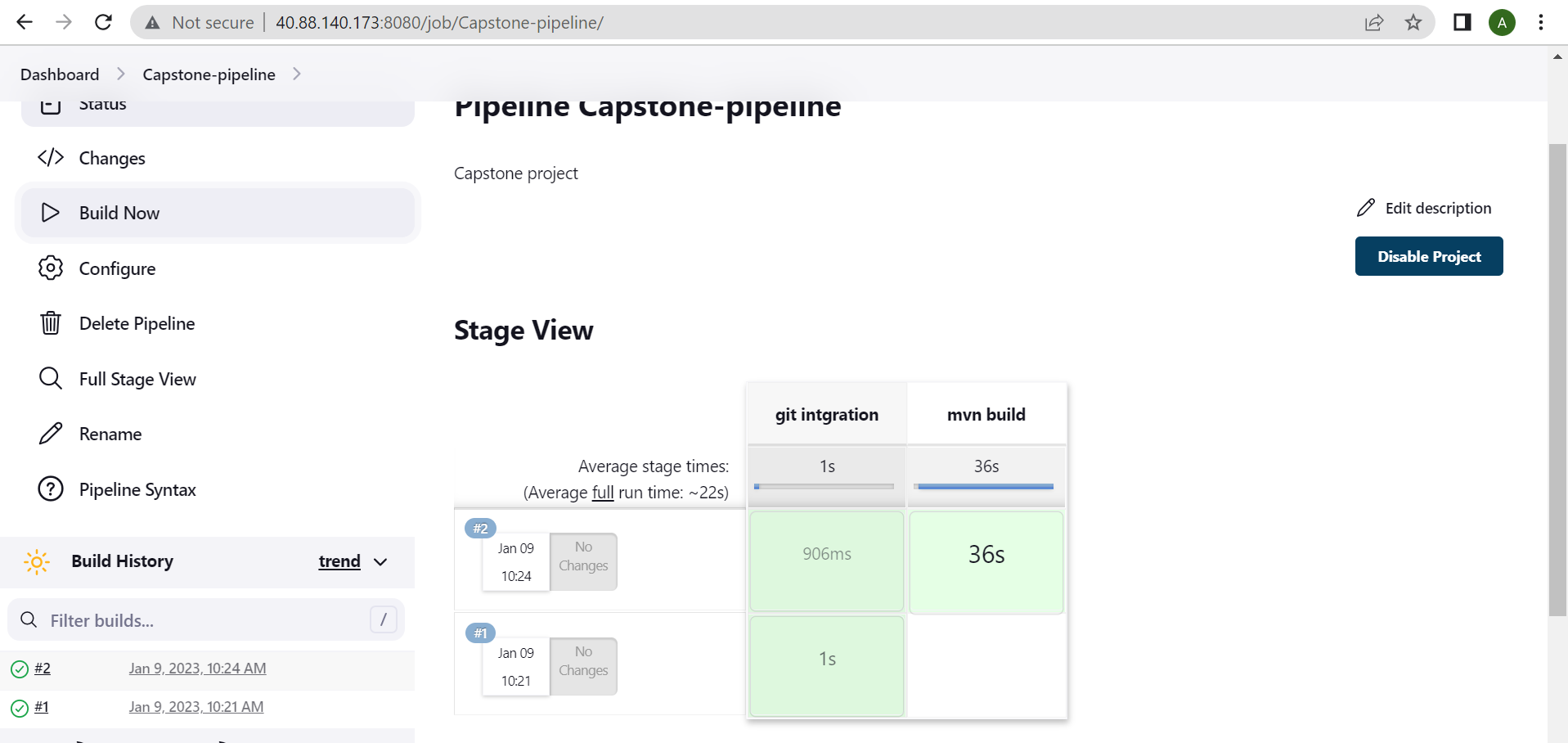
1. **Create a Jenkins pipeline – integrate the private repo with Jenkins that will be first stage**



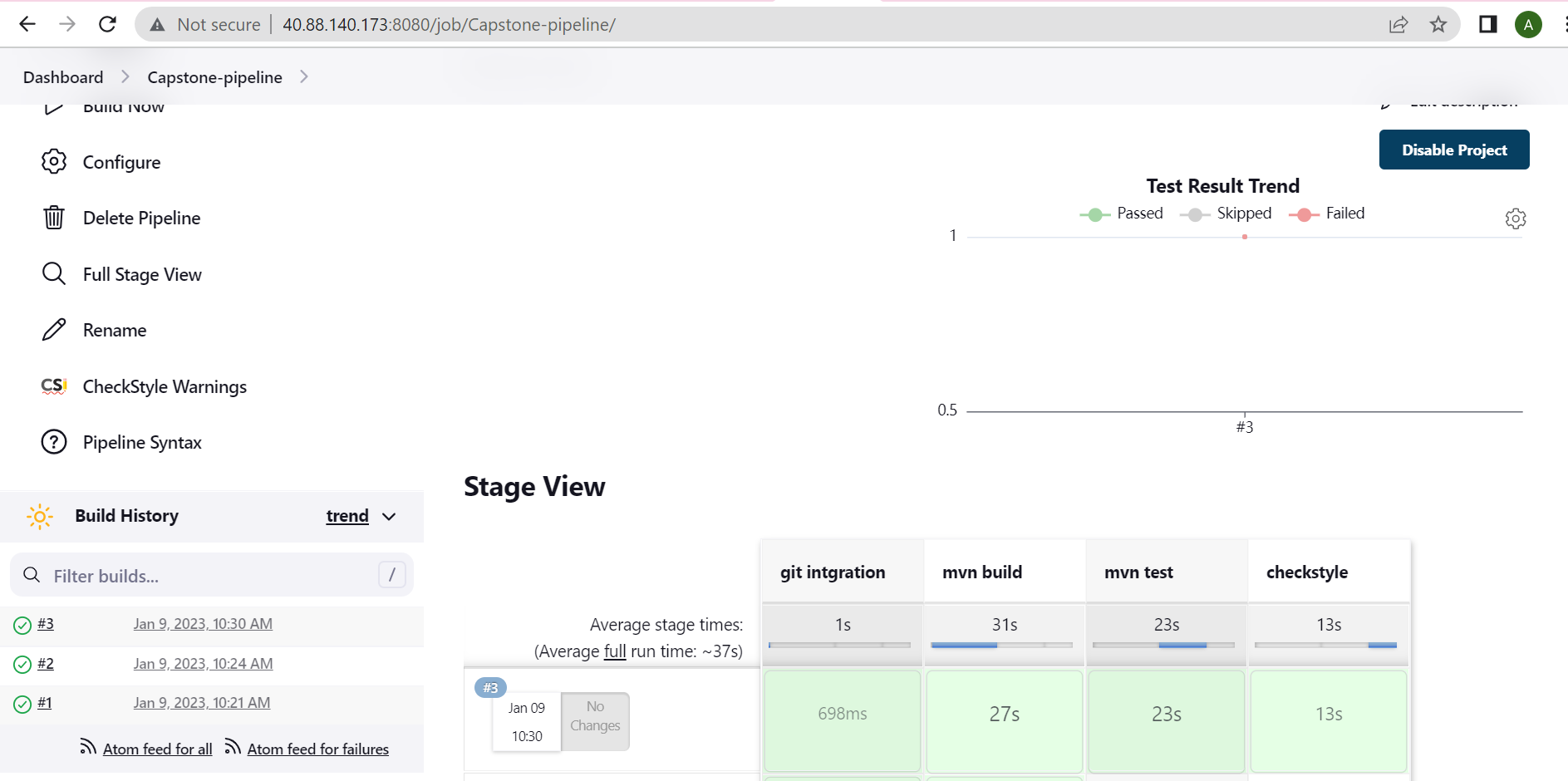


1. **Second stage compile the project using maven but exclude test**





1. **Third stage do code testing using unittest checkstyle codecoverage**

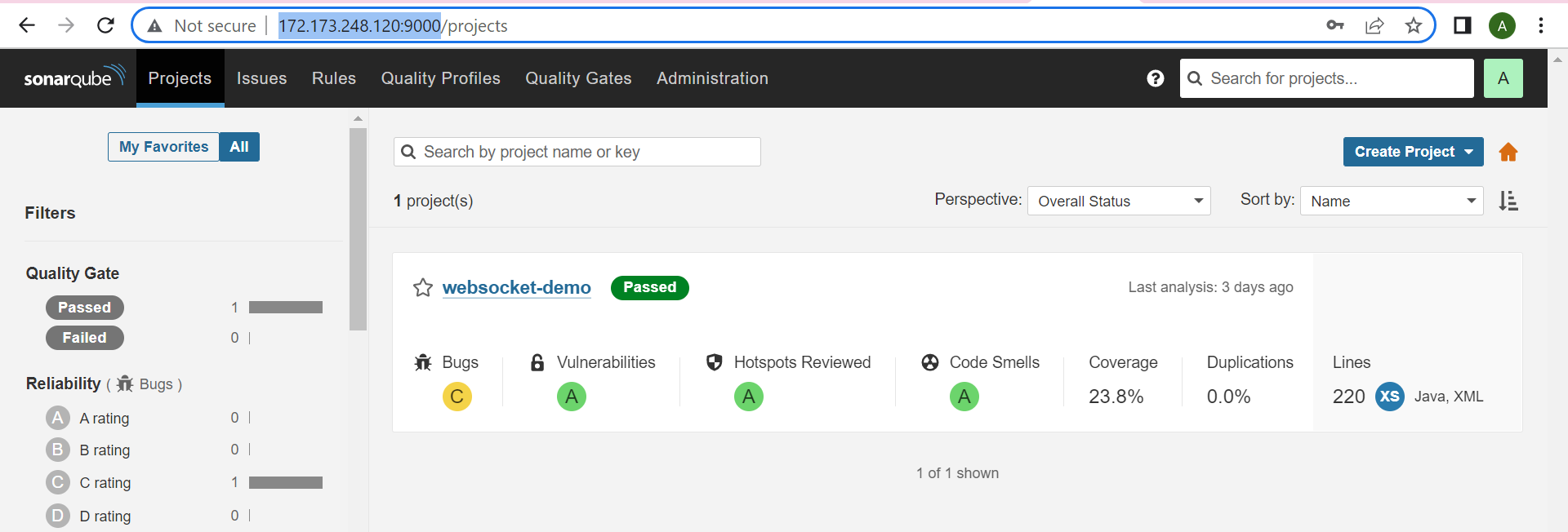


1. **Add all this stages in the pipeline**



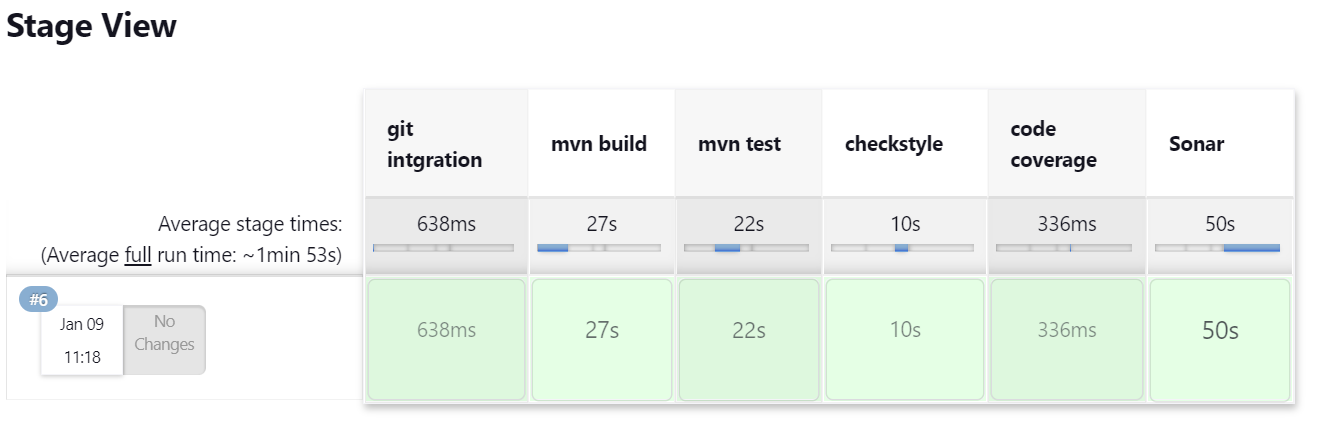
1. **Then integrate Jenkins with SonarQube. Provide the difference between sonarqube testing and the previous testing framework.**

Sonarqube dashboard:



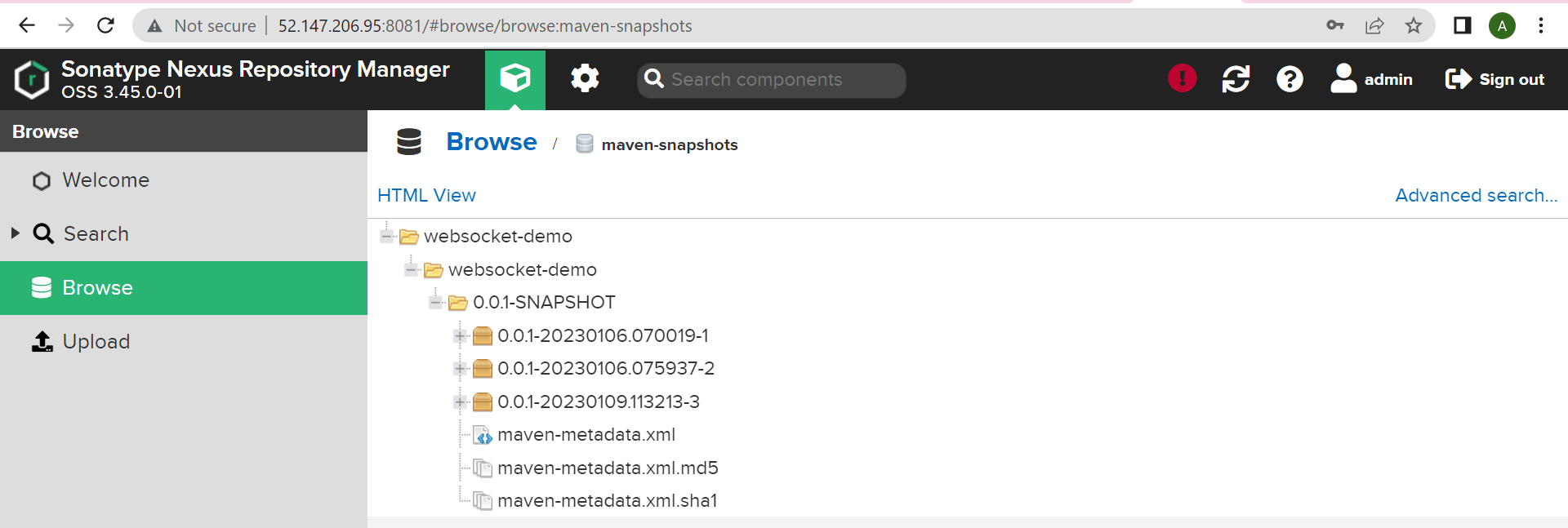
Jenkins pipeline:

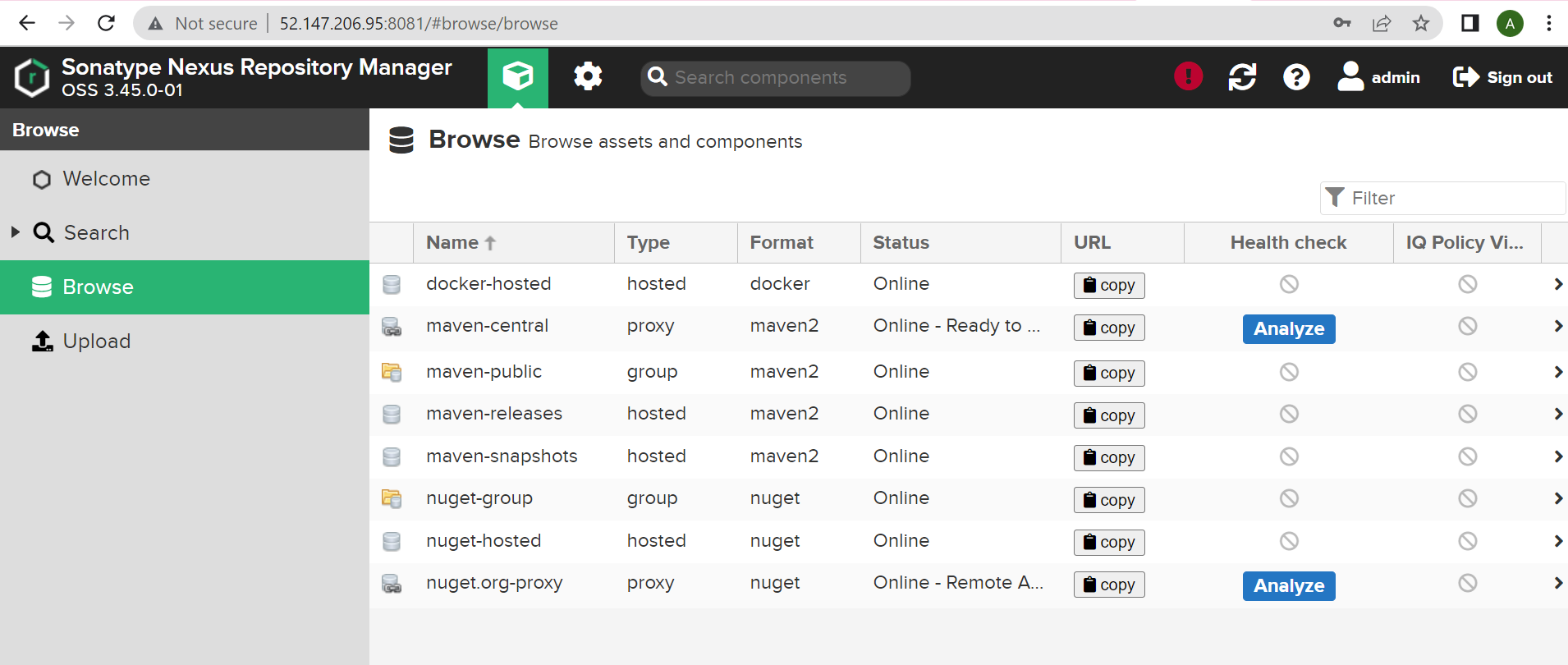




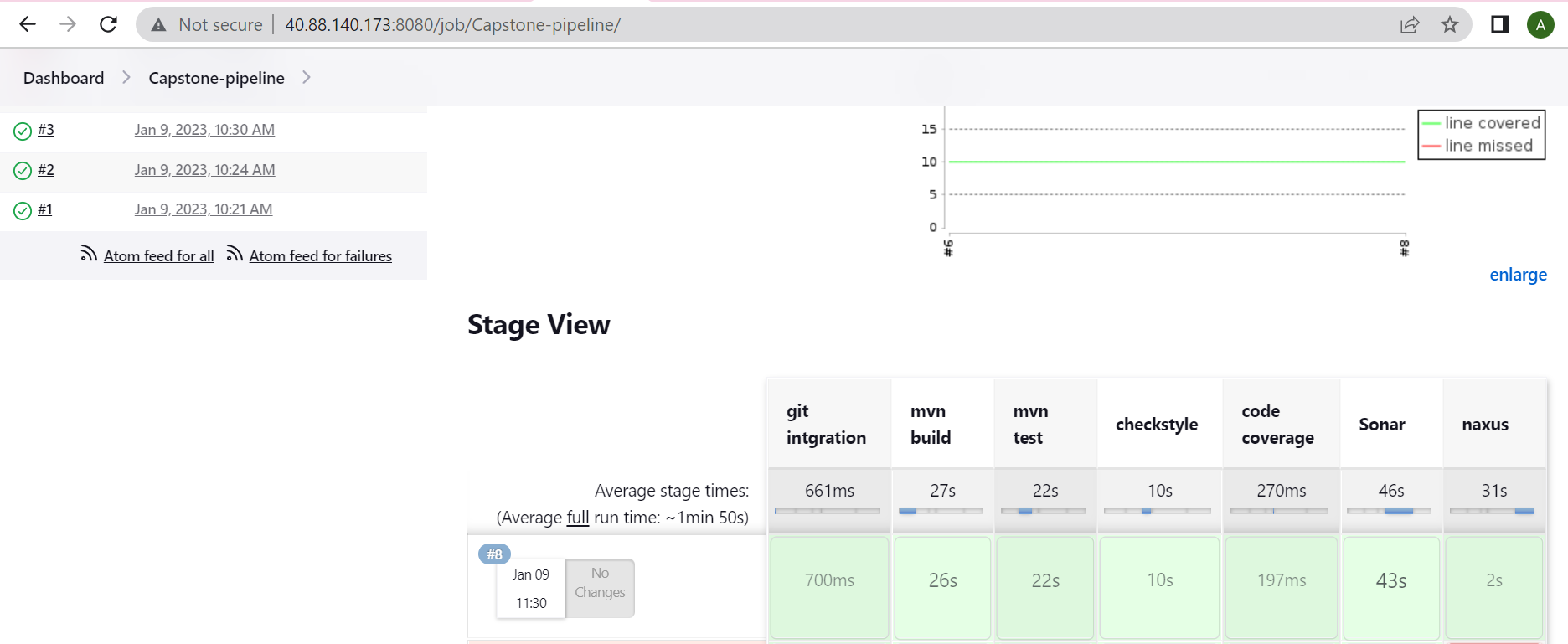
1. **Finally upload the jar file inside nexus**
2. **While uploading jar file pipeline syntax generator provide you with static value**

Naxus dashboard:





Jenkins pipeline:



1. **Autmatically find out the jar file and upload the same**

stage('naxus'){

    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;

    }

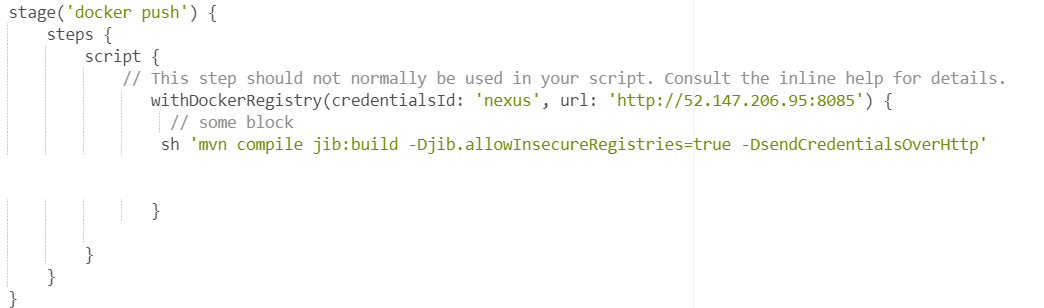
        nexusArtifactUploader artifacts: [[artifactId: pom.artifactId, classifier: '', file: artifactPath, type: pom.packaging, type: 'jar']], credentialsId: 'nexus', groupId: pom.artifactId, nexusUrl: '52.147.206.95:8081', nexusVersion: 'nexus3', protocol: 'http', repository: 'maven-snapshots', version: pom.version

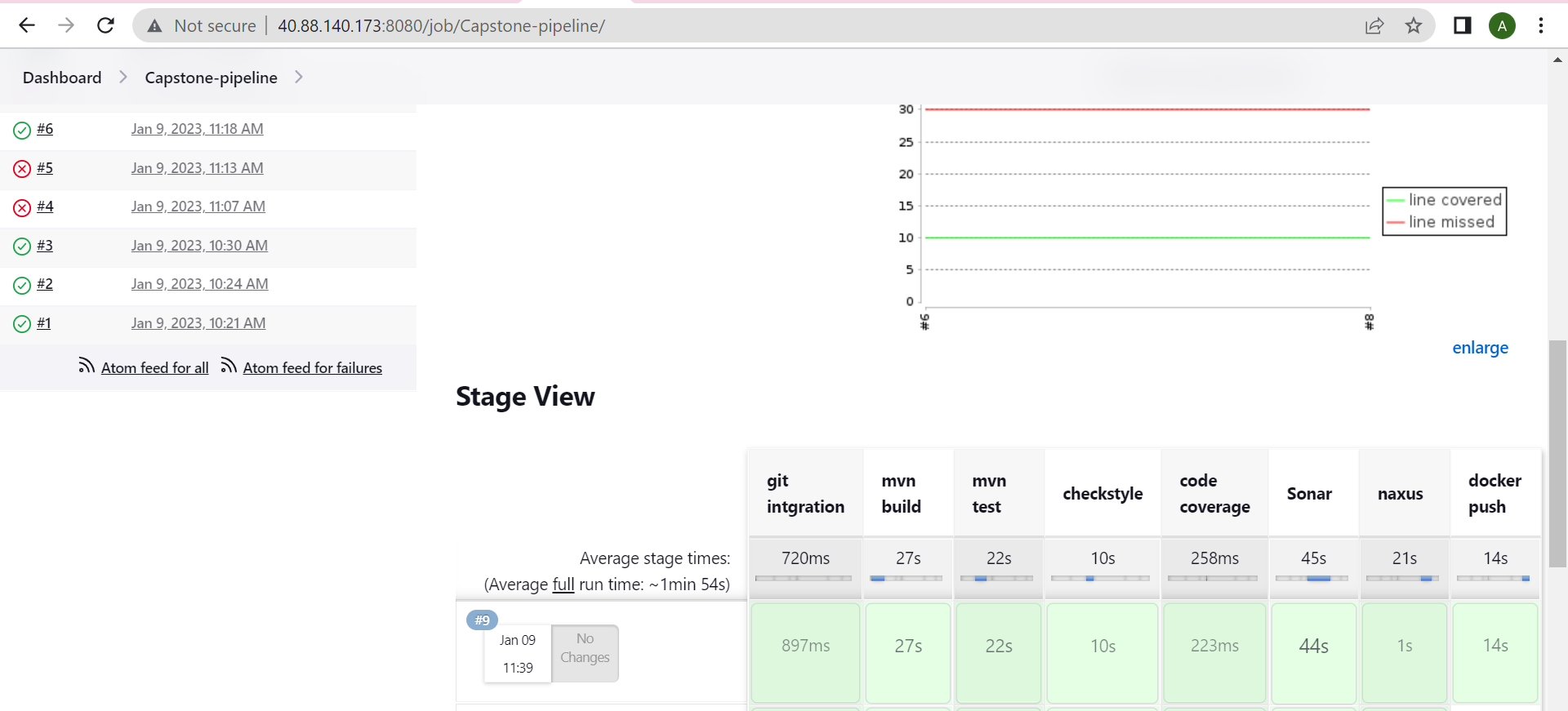
    }

}

**Jenkins CI-Pipeline cont…**

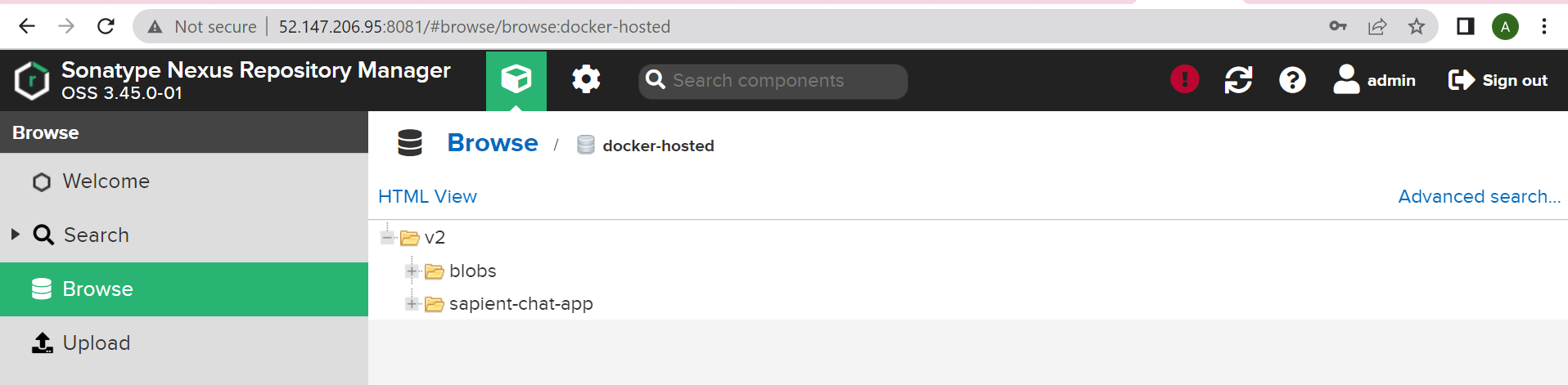
1. **Convert the application into an container based image**





1. **Upload the application into nexus repo**

Nexus dashboard



**Jenkins CD-pipeline**

1. **Create an ansible playbook which install docker in Ubuntu system**

---

- hosts: all

  become: true

  tasks:

    - name: Update and upgrade apt packages

      apt:

      upgrade: yes

      update\_cache: yes

    - 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

    # Uncomment the following to enable insecure registries with Docker

    - name: ensure docker can use insecure registries in 10.11.0.0/16

      lineinfile: "dest=/etc/default/docker regexp=^DOCKER\_OPTS line=DOCKER\_OPTS='--insecure-registry 10.0.0.0/24'"

    - name: restart service docker

      service:

        name: docker

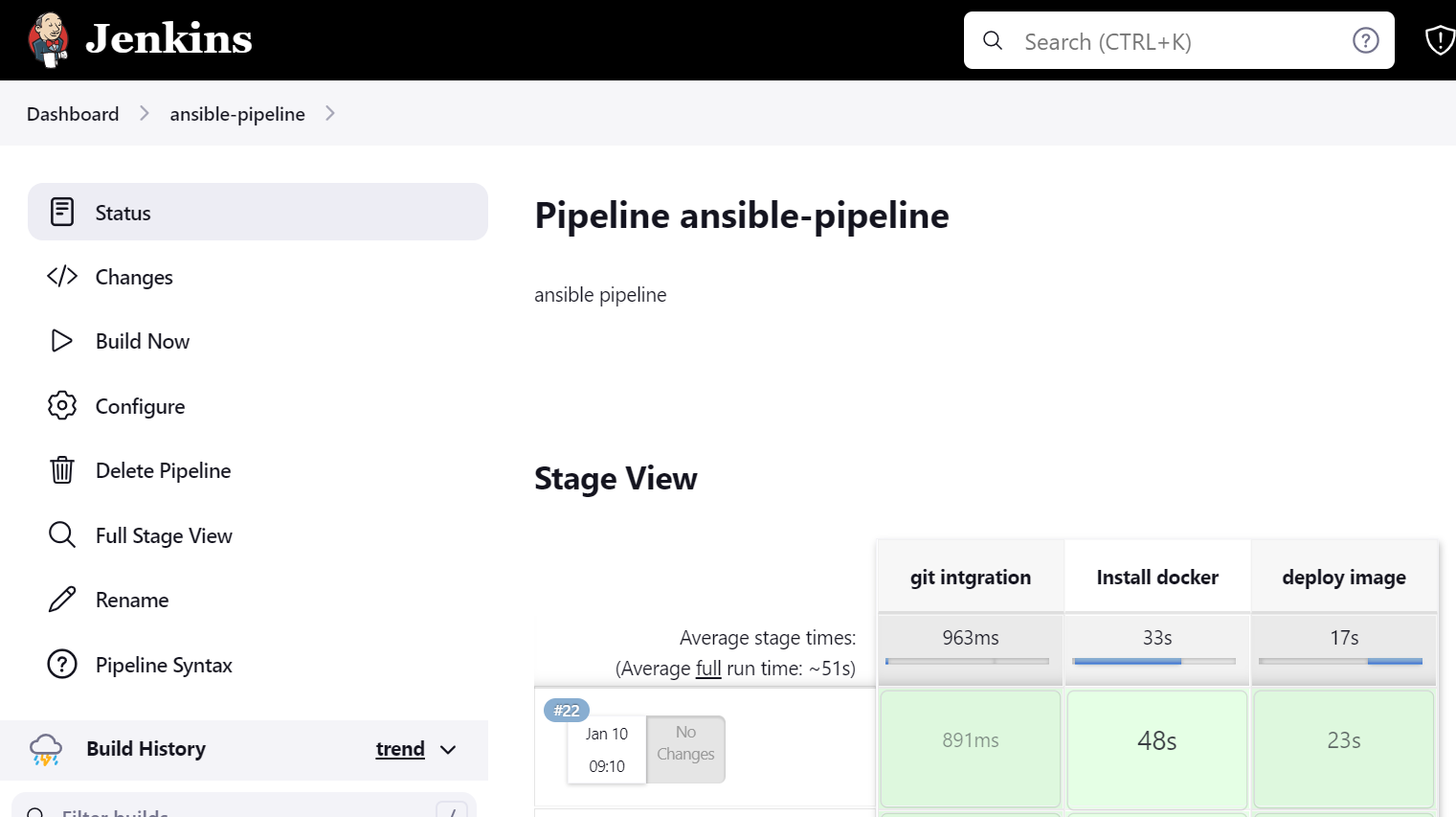
        enabled: yes

        state: started

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

docker-image.yml





**Jenkisn CD pipeline with tomcat.**

1. **Create an ansible playbook which will install tomcat**

---

- 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:

      content: |-

        [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.pid"

        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

      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

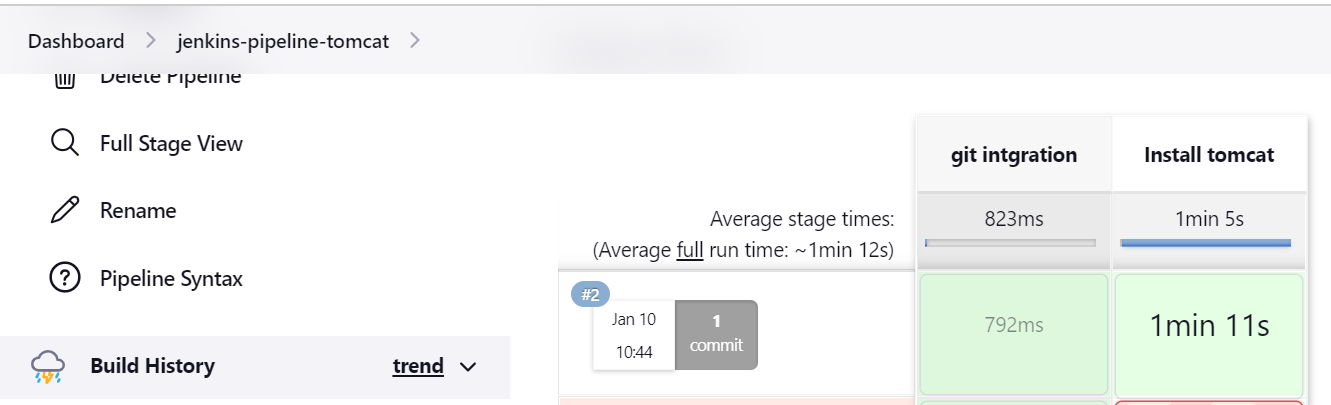
     register: result

     until: "result.status == 200"

     retries: 5

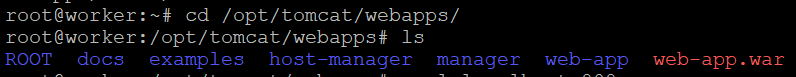
     delay: 10

Jenkins pipeline for install tomcat:

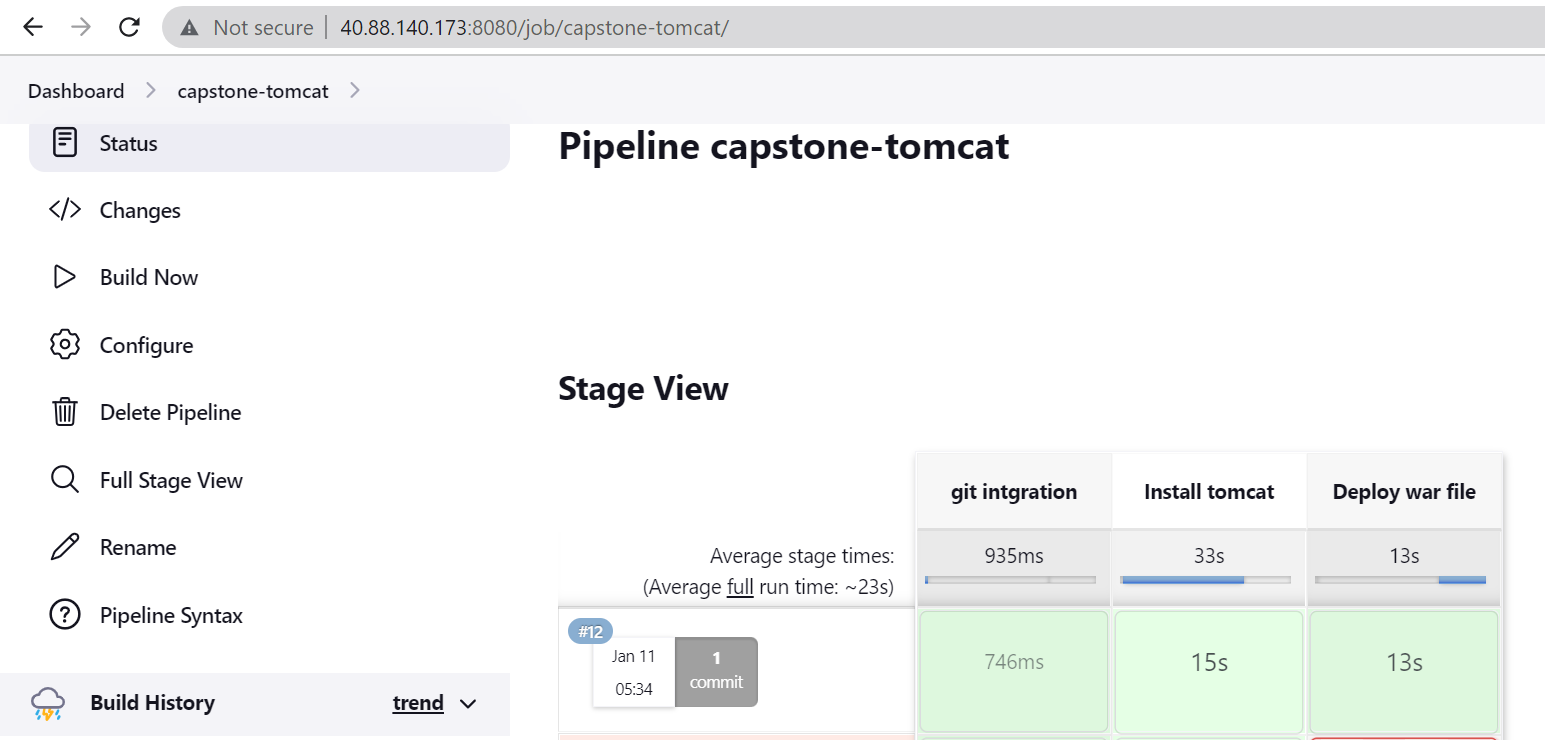


1. **Deploy the jar file from nexus to tomcat using ansible.**

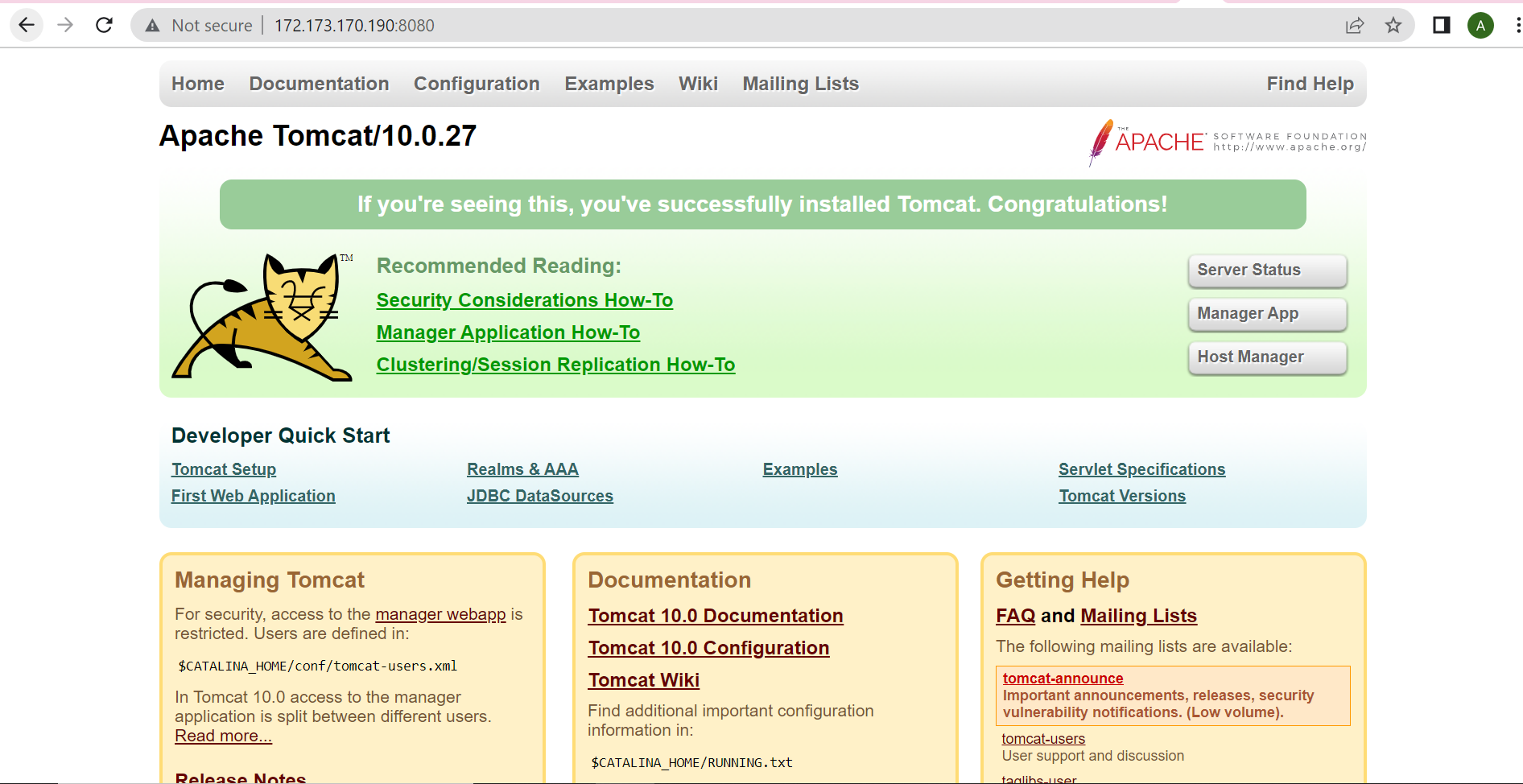
web-app.war file created:



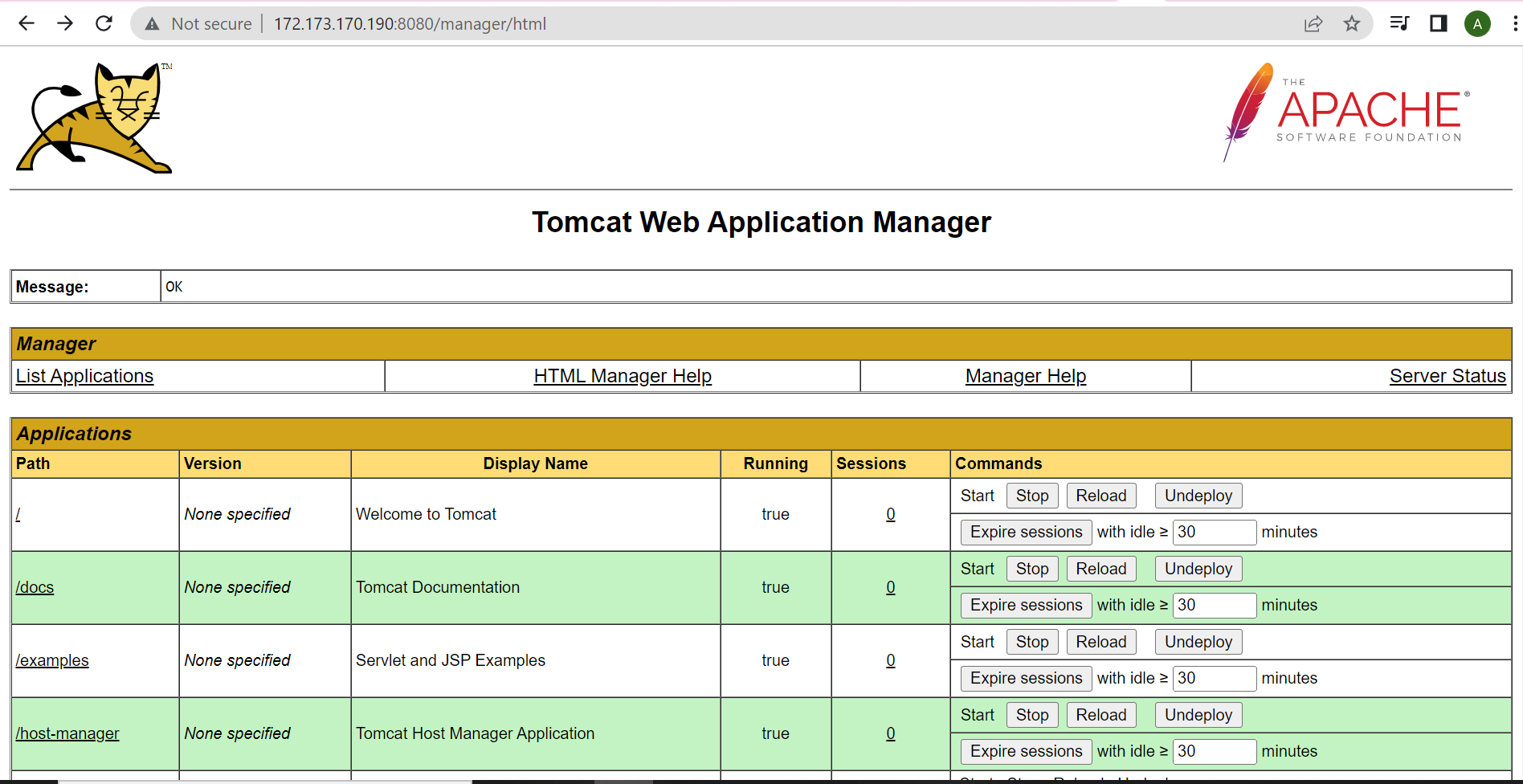
Pipeline for install tomcat & deploy war file



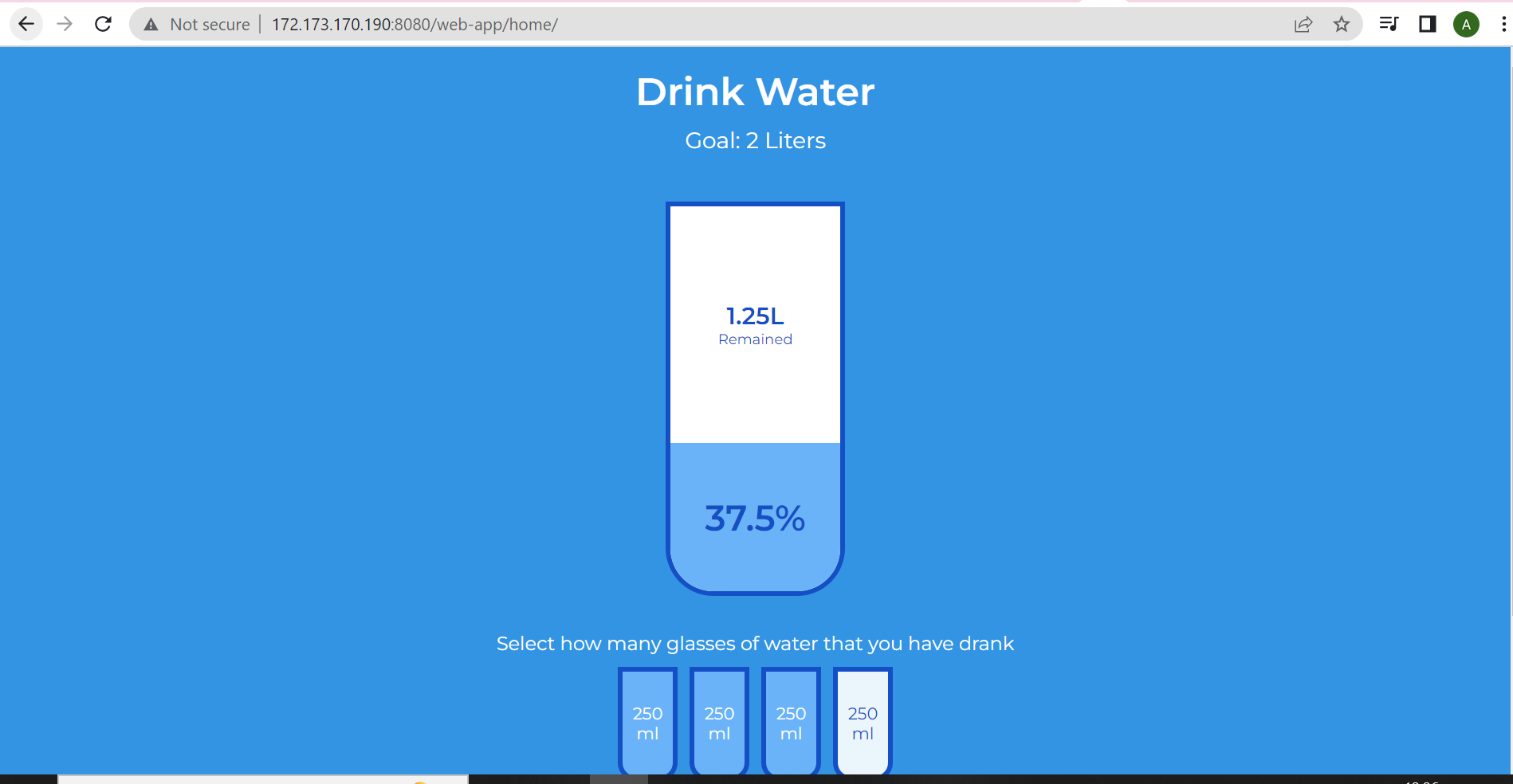
Tomcat is running on port <ip>:8080



running on port <ip>:8080/manager/html



Web app deployed:



**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

Jenkinsfile:

pipeline {

    agent any

    environment {

        RESOURCE\_GROUP = 'hr-dev-capstone'

        HOST\_VM = 'websubnet-web-linuxvm'

        USERNAME = 'azureuser'

    }

    tools{

        terraform 'terraform'

    }

    stages {

        stage('Poll SCM') {

            steps {

                git credentialsId: 'git-key', url: 'git@github.com:avi207/chat-app-sapient.git'

            }

        }

        stage('terraform format') {

            when {

                expression {stage == 'plan'}

            }

            steps {

                script {

                    dir('terraform/') {

                        sh 'terraform fmt'

                    }

                }

            }

        }

        stage('terraform initialize') {

            when {

                expression {stage == 'plan'}

            }

            steps{

                script {

                    dir('terraform/') {

                        sh 'terraform init'

                    }

                }

            }

        }

        stage('terraform validate') {

            when {

                expression {stage == 'plan'}

            }

            steps{

                script {

                    dir('terraform/') {

                        sh 'terraform validate'

                    }

                }

            }

        }

        stage('terraform plan') {

            when {

                expression {stage == 'plan'}

            }

            steps{

                script {

                    dir('terraform/') {

                        sh 'terraform plan'

                    }

                }

            }

        }

        stage('terraform apply') {

            when {

                expression {stage == 'plan'}

            }

            steps{

                script {

                    dir('terraform/') {

                        sh 'terraform apply --auto-approve'

                    }

                }

            }

        }

        stage('vm public key') {

            when {

                expression {stage == 'plan'}

            }

            steps{

                script {

                    env.vm\_ip = sh( script: "az vm show -d -g hr-dev-capstone -n websubnet-web-linuxvm --query publicIps -o tsv",

                                    returnStdout: true).trim()

                    echo "MYVAR: ${env.vm\_ip}"

                }

            }

        }

        stage('SSH Connect to VM') {

            when {

                expression {stage == 'plan'}

            }

            steps{

                sh'''

                export PUBLIC\_IP=$(az vm show -d -g $RESOURCE\_GROUP -n $HOST\_VM --query publicIps -o tsv)

                ssh -o "StrictHostKeyChecking no" $USERNAME@$PUBLIC\_IP

                '''

            }

        }

        stage("Install Istio"){

            when {

                expression {stage == 'plan'}

            }

            steps{

                sh '''

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

                    ssh -tt -o "StrictHostKeyChecking no" azureuser@$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('Install Helm Chart') {

            when {

                expression {stage == 'plan'}

            }

            steps{

                sh'''

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

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

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

                sudo helm install my-nginx-release bitnami/nginx

                exit

                '''

            }

        }

        stage('terraform destroy') {

            when {

                expression {stage == 'destroy'}

            }

            steps{

                script {

                    dir('terraform/') {

                        sh 'terraform destroy --auto-approve'

                    }

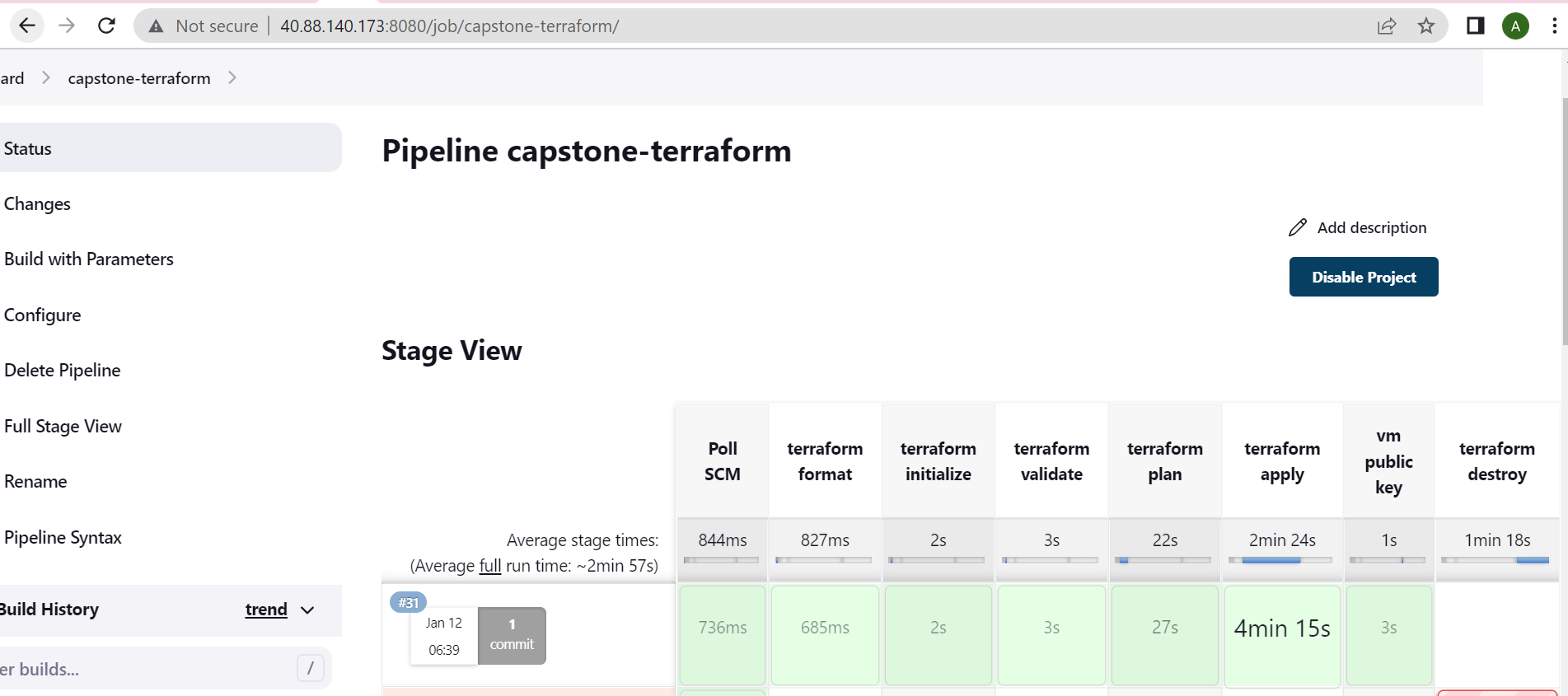
                }

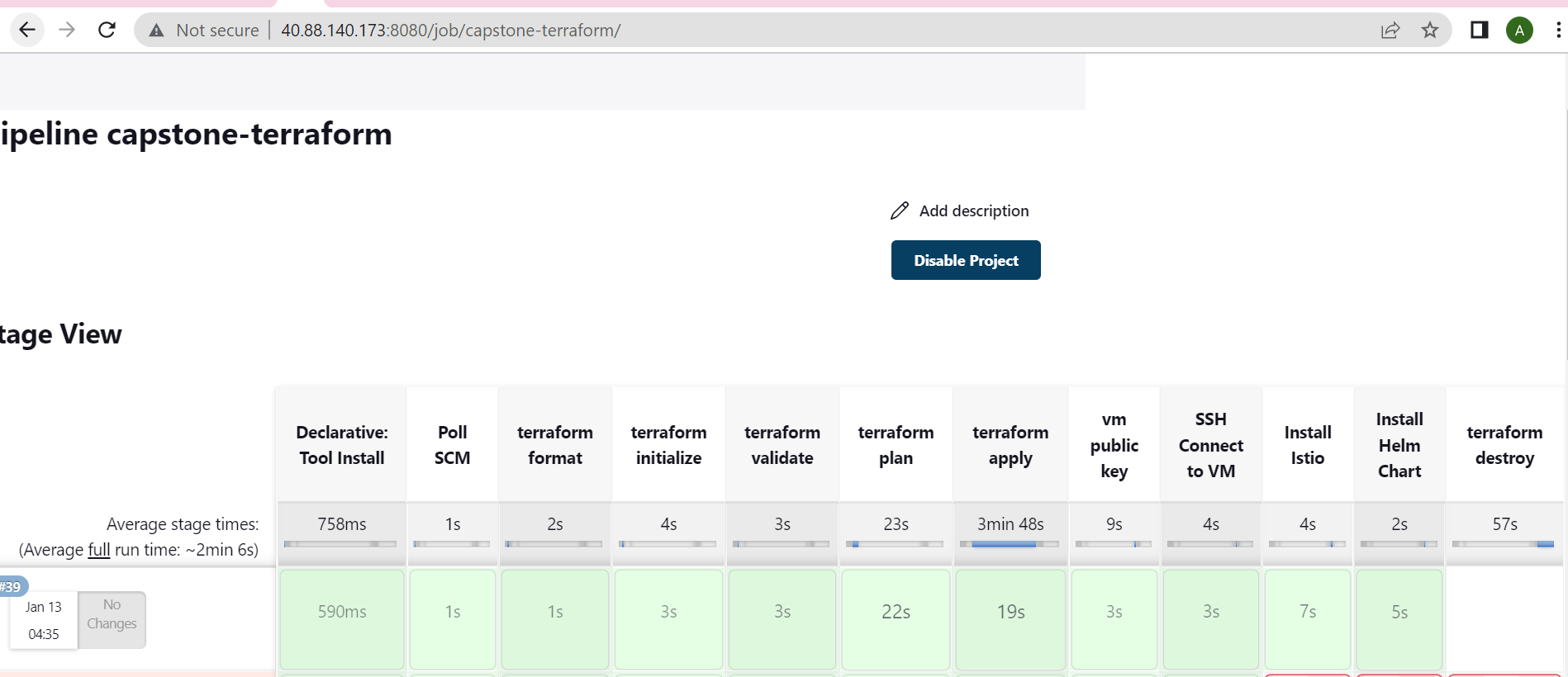
            }

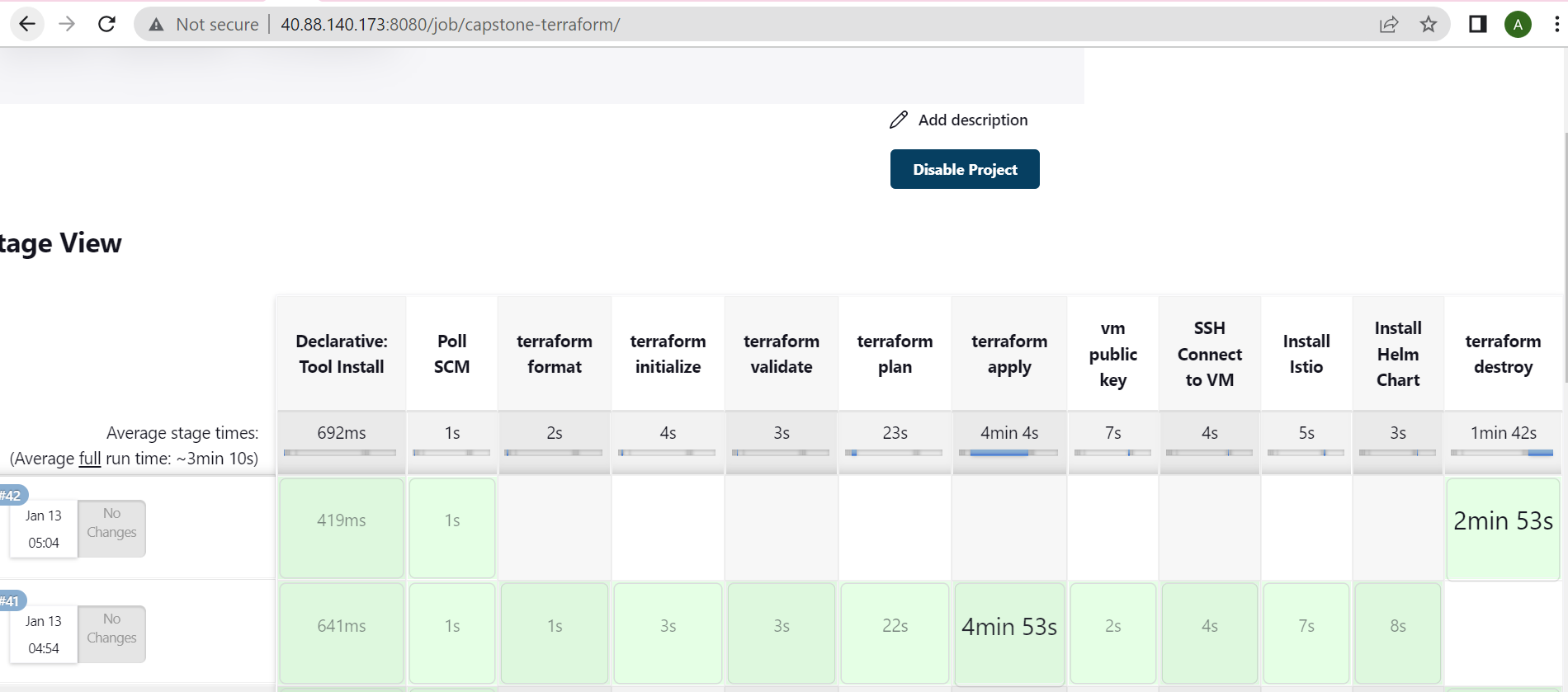
        }

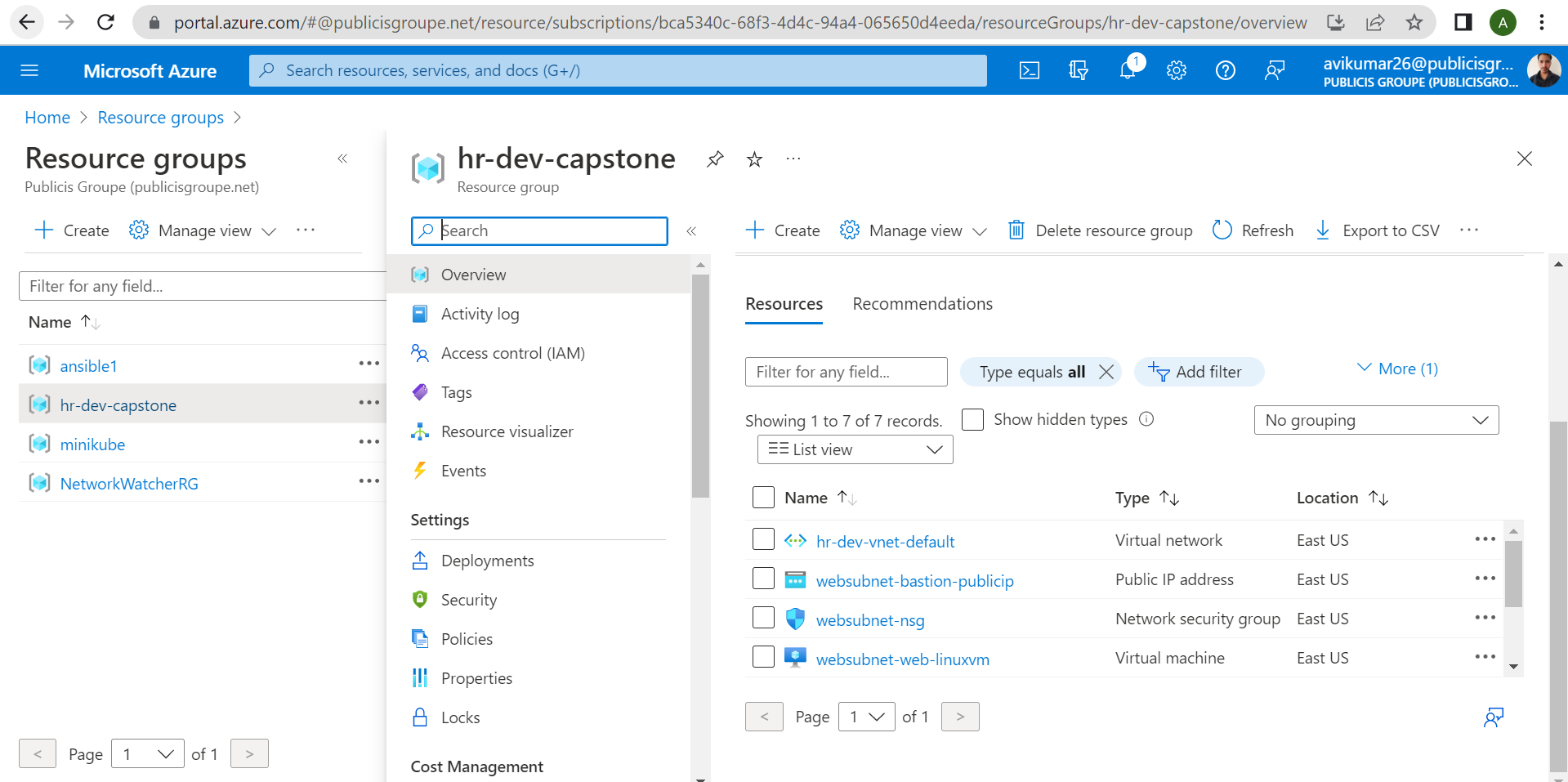
    }

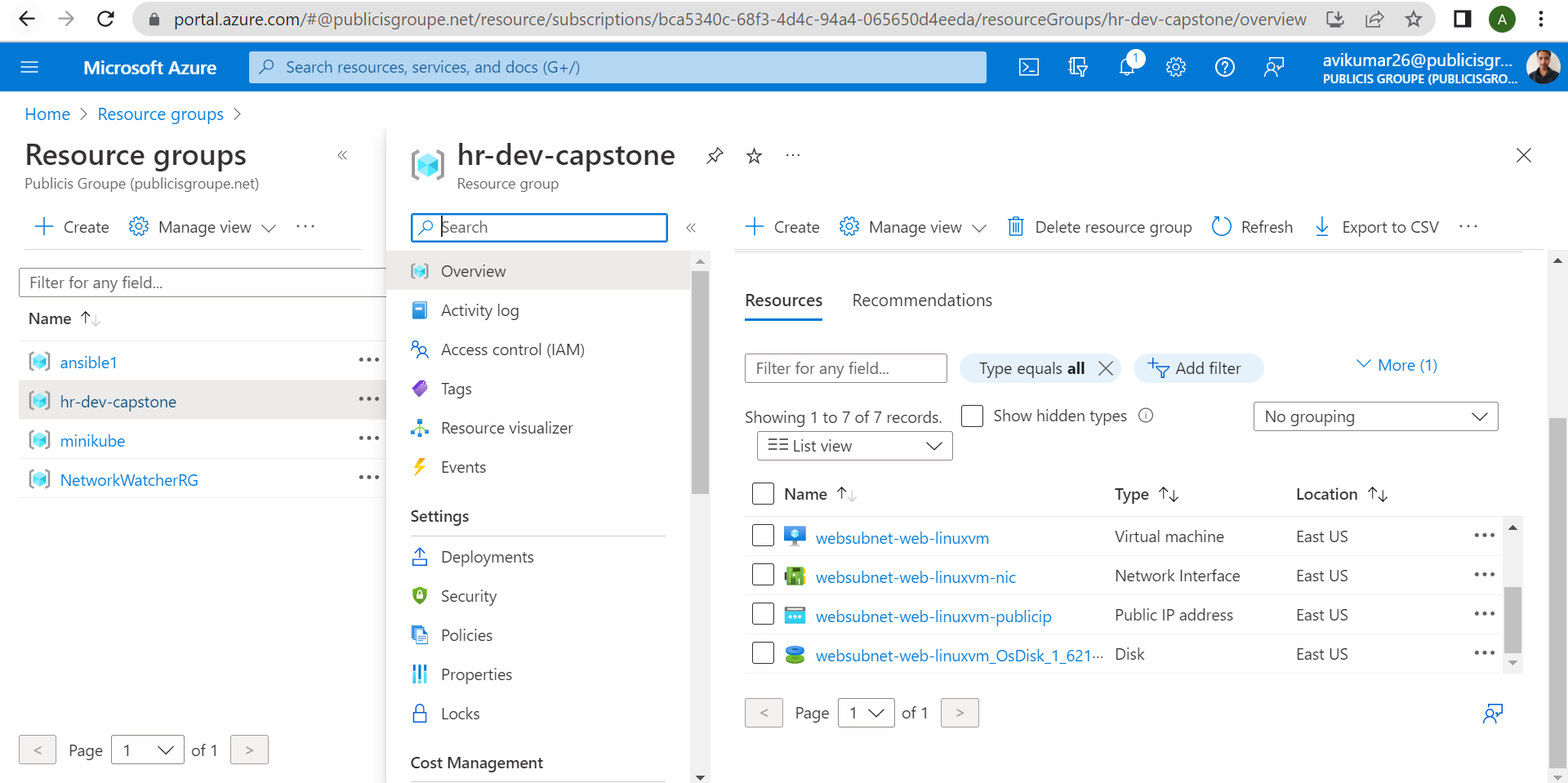
}

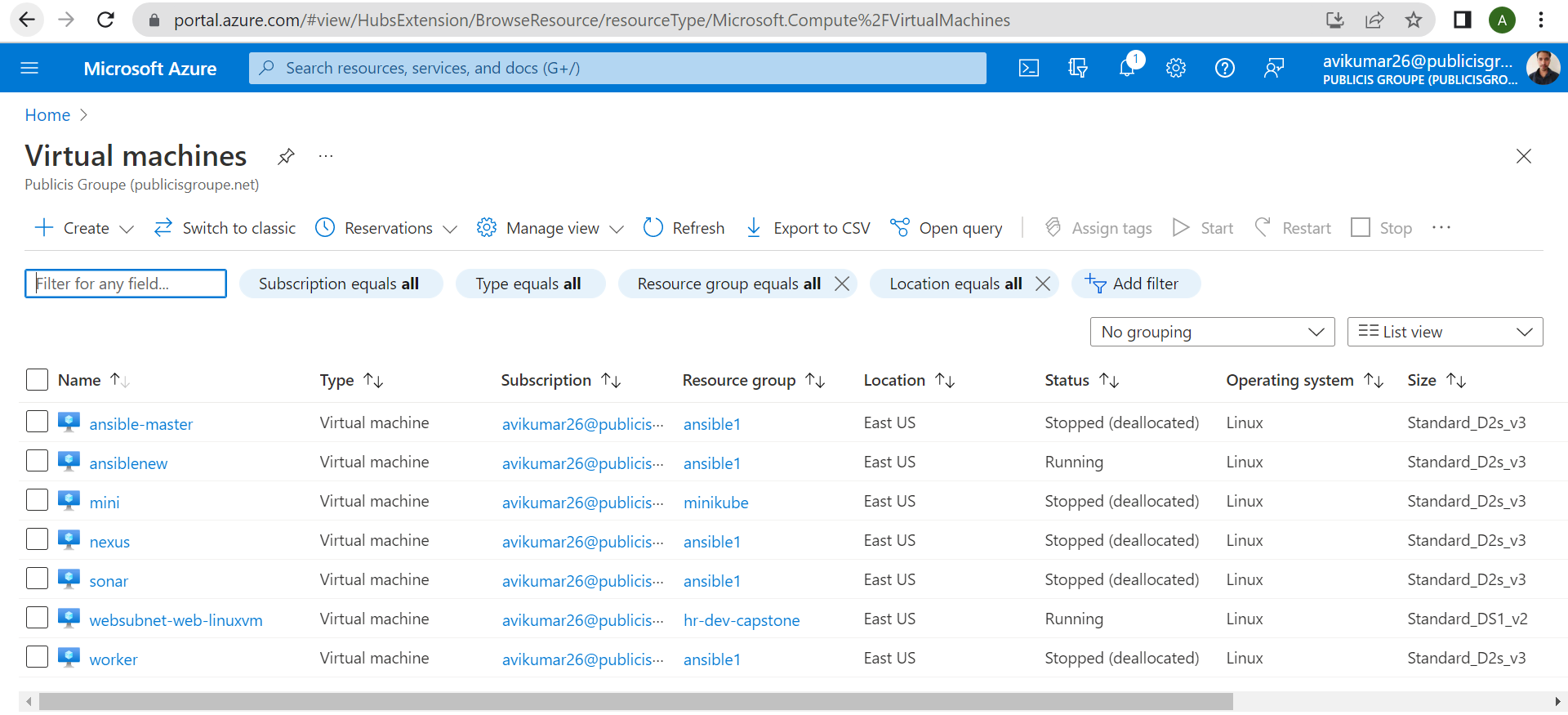




Pipeline after Terraform destroy:

Azure dashboard for resource group:



Azure dashboard for virtual machine:

**include SonarQube quality gate enforcement in Jenkins pipeline to be added**

pipeline stages for quality gate:

stage('Sonarqube Analysis'){

            steps {

                withSonarQubeEnv('sonar-qualitygate') {

                    sh 'mvn clean verify sonar:sonar -Dsonar.projectKey=sapient-chatapp'

                }

            }

        }

        stage("SonarQube Quality Check"){

            steps{

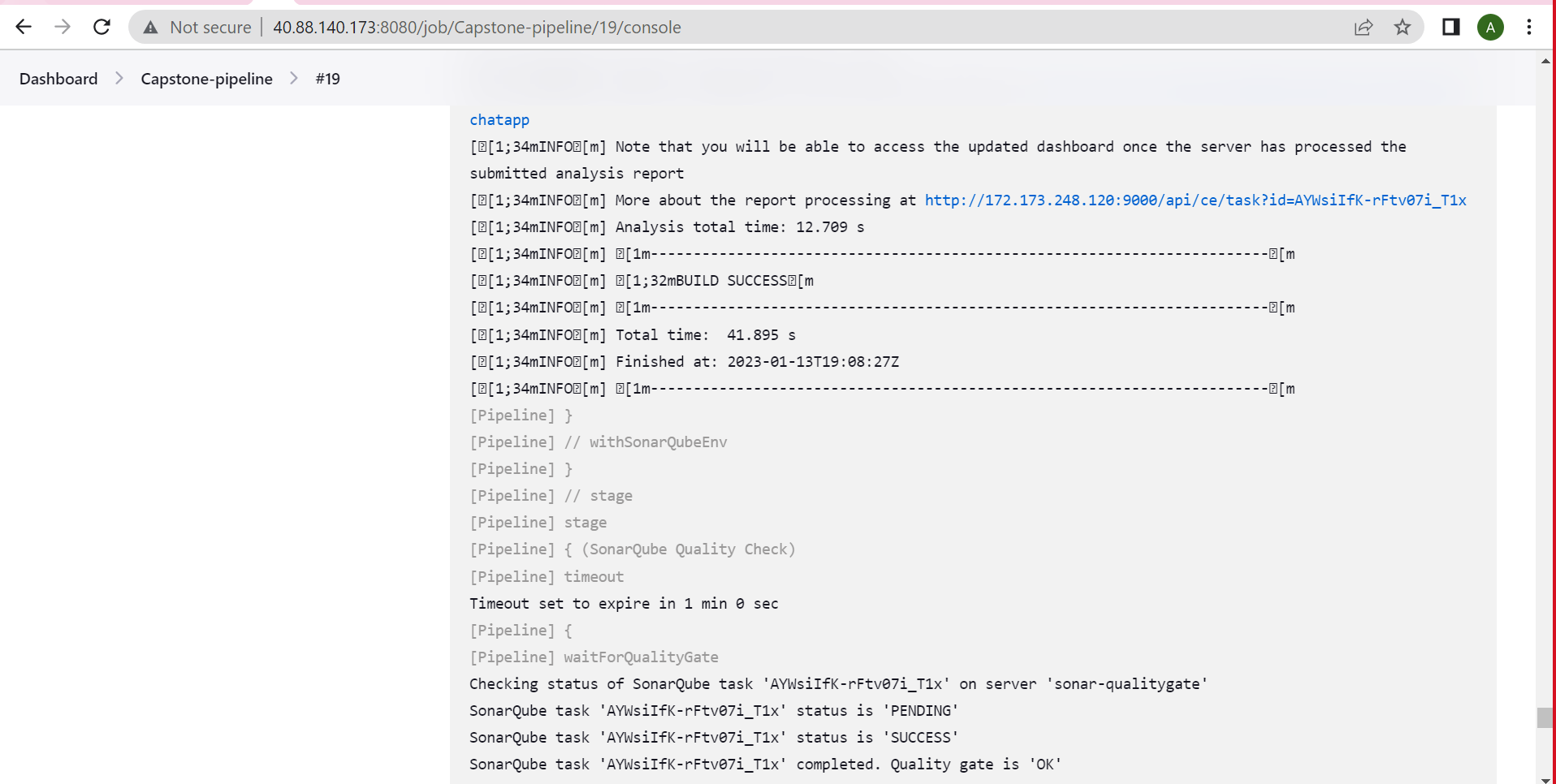
                timeout(time:1, unit:'MINUTES'){

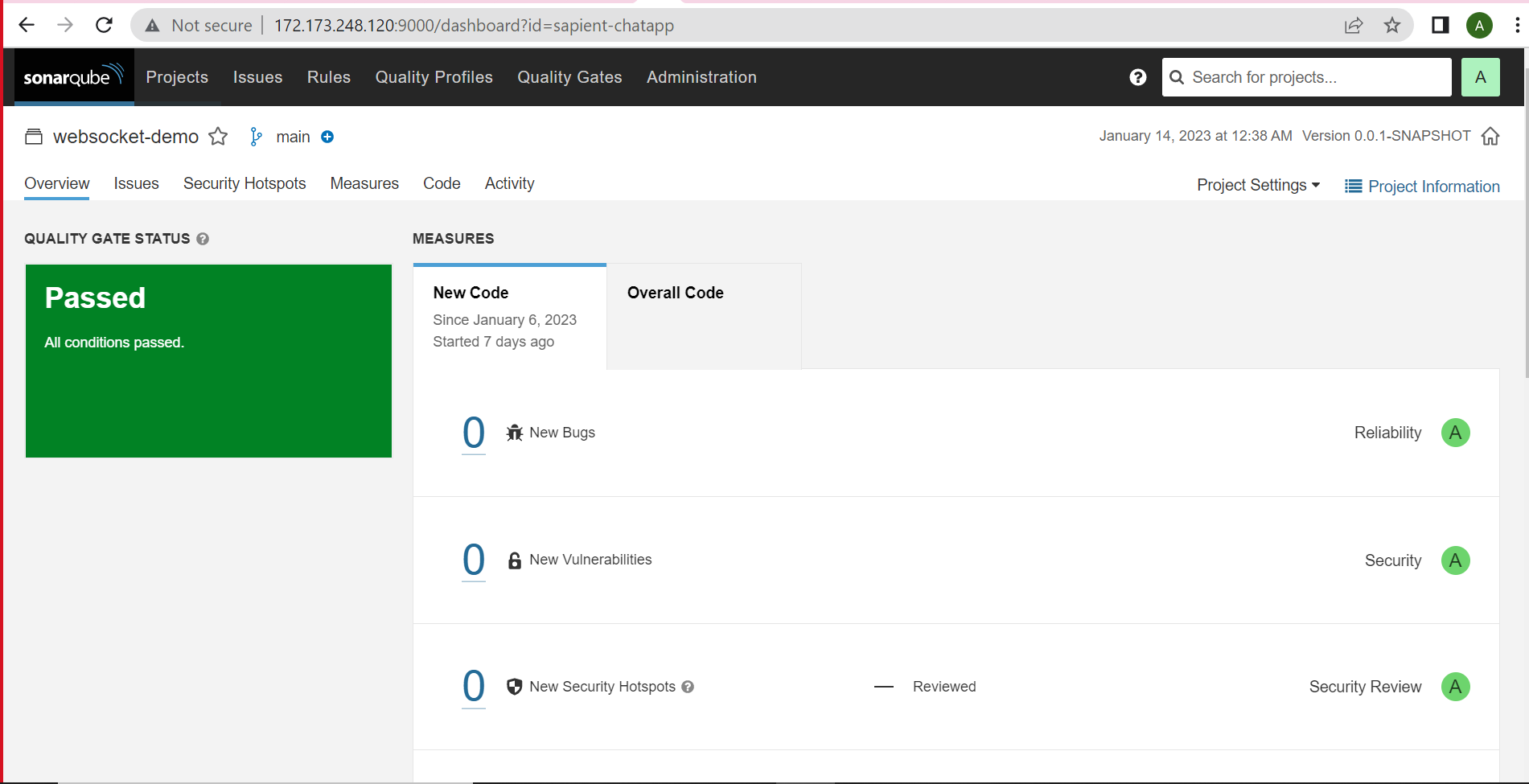
                    waitForQualityGate abortPipeline: true, credentialsId: 'sonar-qualitygate'

                }

            }

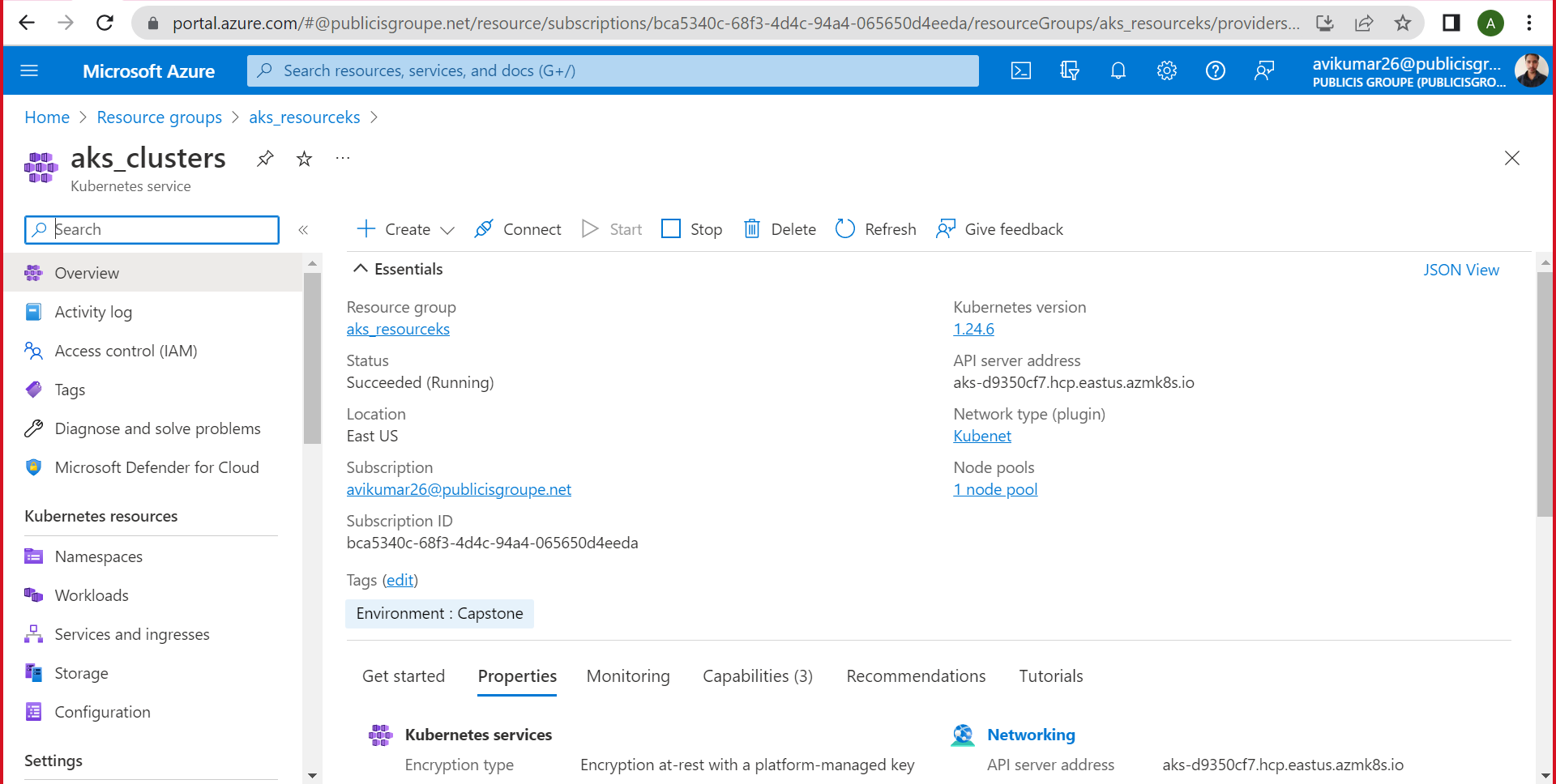
        }

Sonar dashboard:

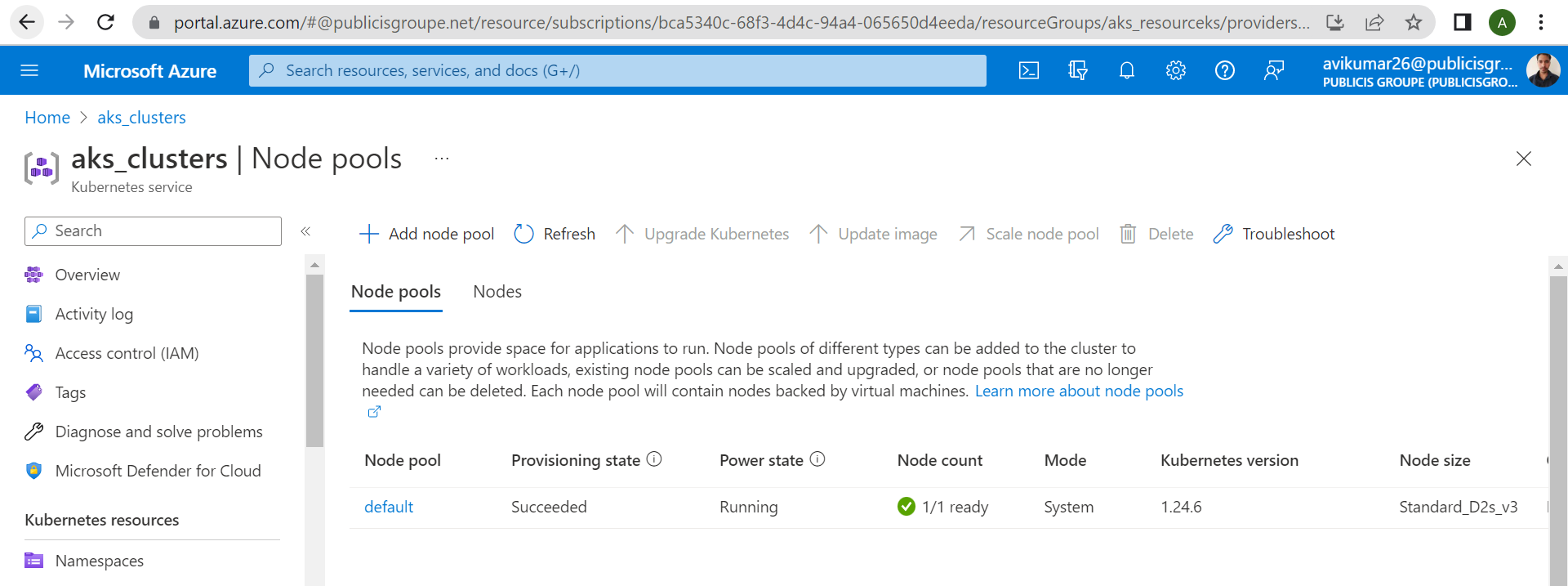


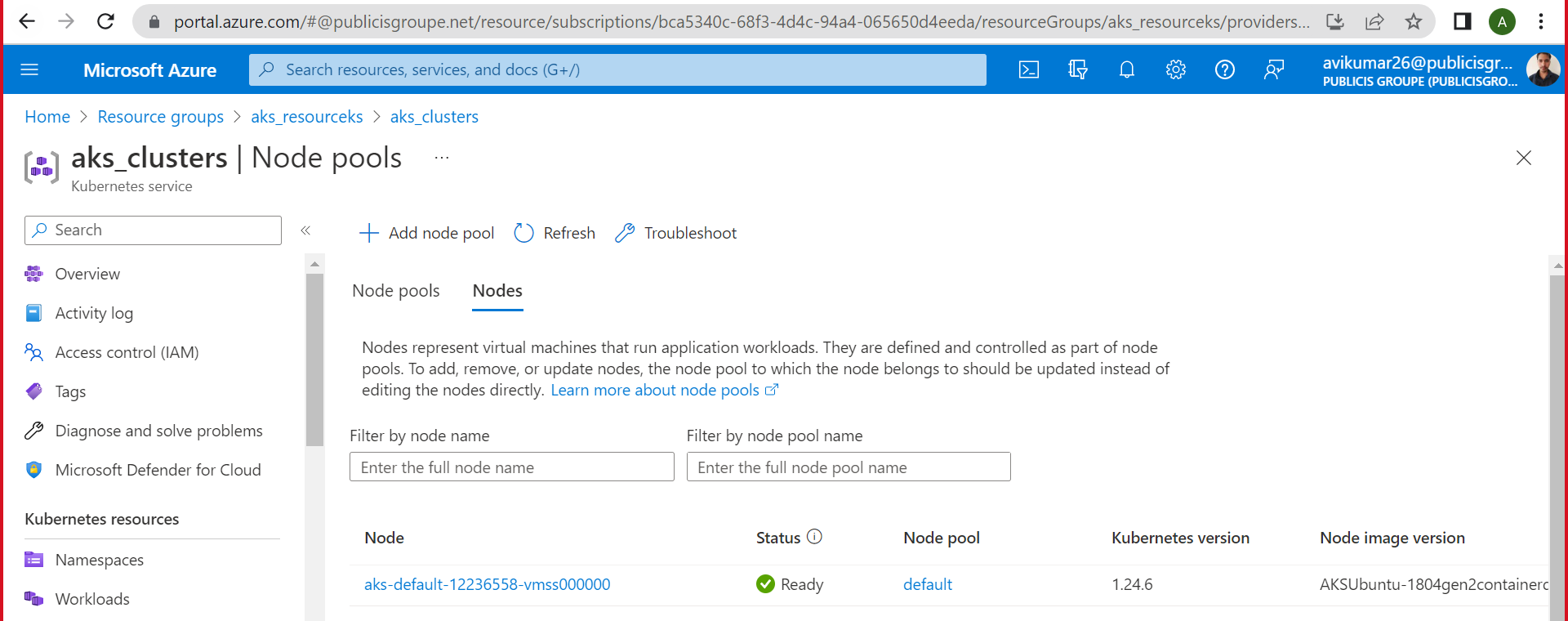
**Use terraform to provision AKS and associated resources in Azure, then sample created helm created(in previous module/exercise) to deploy AKS cluster, use Jenkins as orchestrator.**

Aks cluster is created:

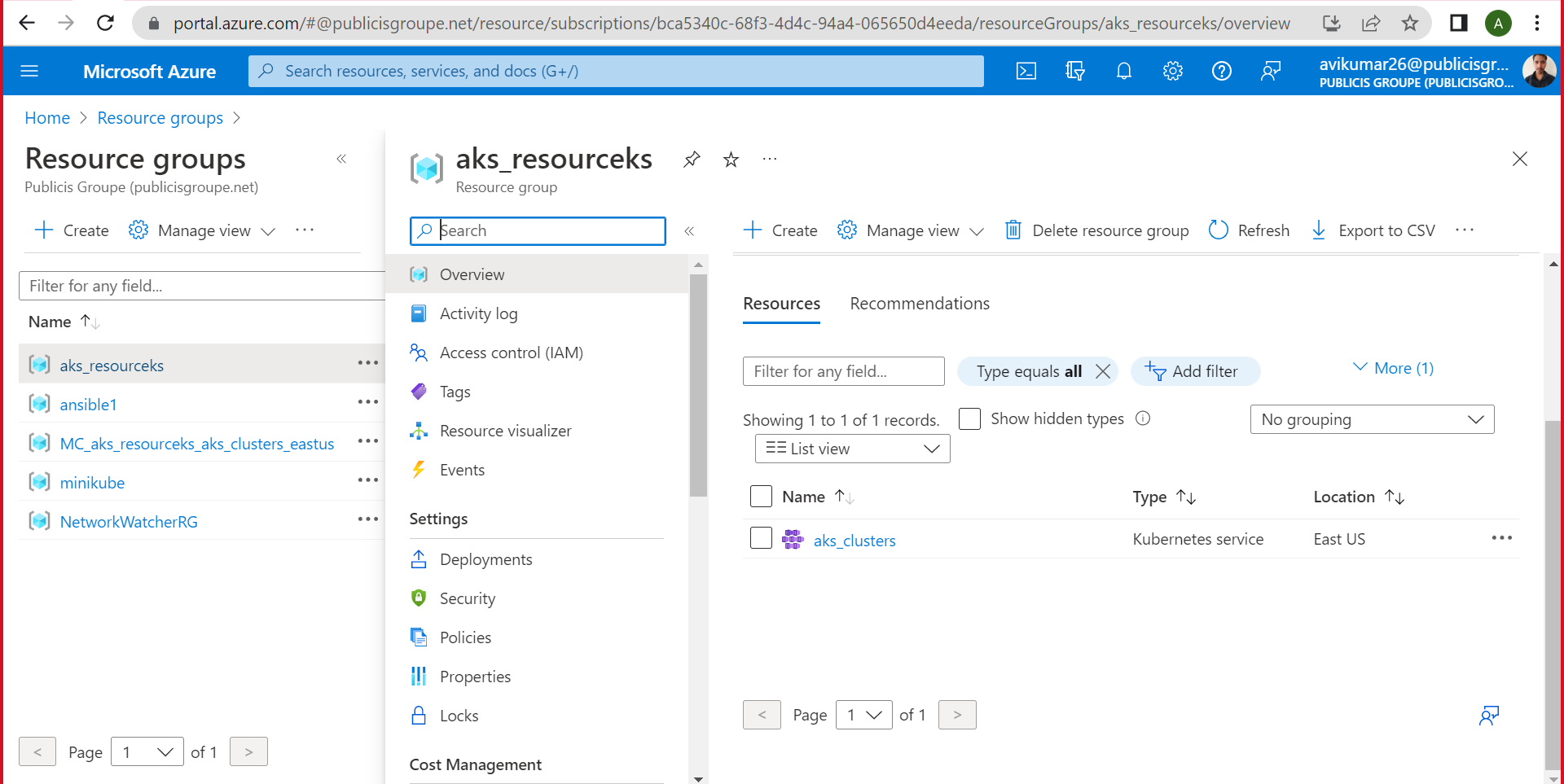


Node is created:

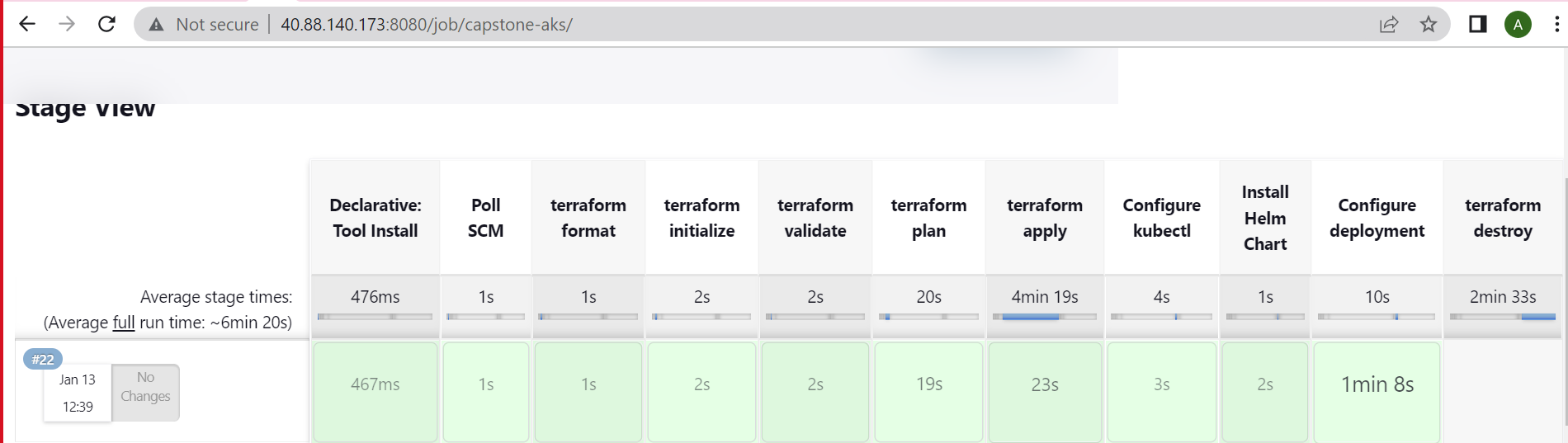




Resource group dashboard:



Jenkins pipeline:



Terraform destroyed: