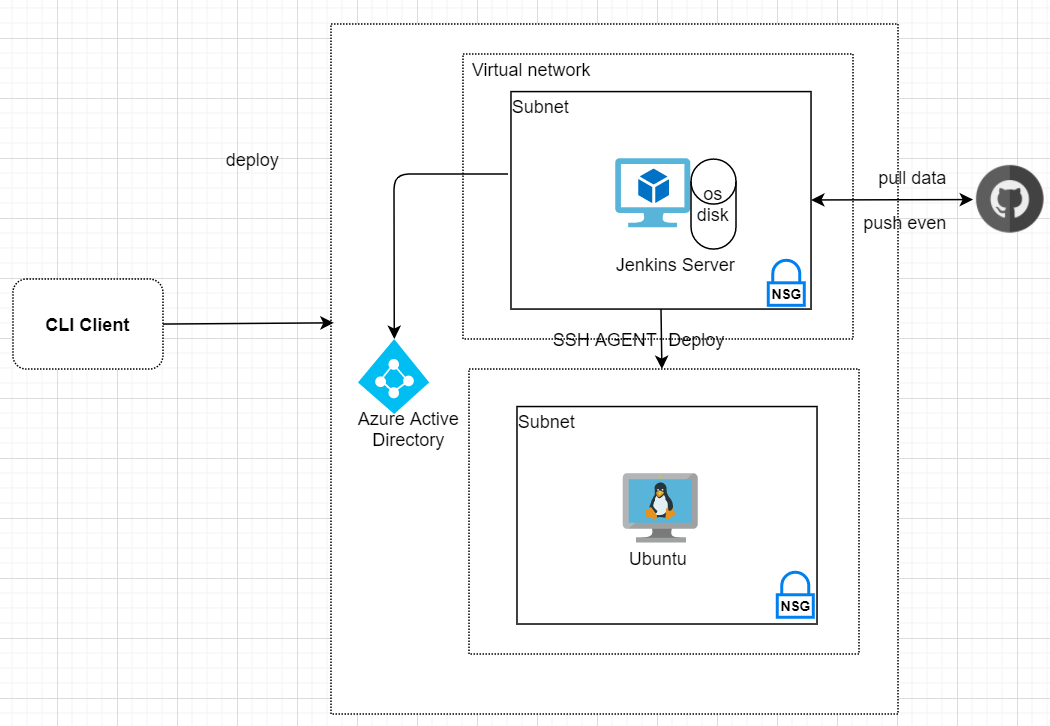
**Install Jenkins on Azure**

**Deployment Architecture**

  
  
In this exercise, we will deploy a Jenkins server on azure cloud using azure cli api.  
  
**Step 1**: Login into Azure

az login –username=youruser  
Password prompt next

**Step 2**: Create resource group to manage Jenkins server and all related resources which are created later.  
  
  
  
**Step 3**: Create ubuntu VM and Initialize the server with shell script which executes:  
- install open JDK 8  
- install Jenkins server  
- install docker

az group create --name jenkins-rg --location eastus

az vm create --resource-group jenkins-rg --name jenkins --image Canonical:UbuntuServer:16.04-LTS:latest --admin-username azureuser --generate-ssh-keys --custom-data cloud-init-jenkins.txt

**cloud-init-jenkins.txt:**

#cloud-config

runcmd:

- sudo apt-get update

- sudo apt install openjdk-8-jre-headless -y

- sudo apt install openjdk-8-jdk-headless -y

- wget -qO - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -

- sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

- sudo apt-get update && sudo apt-get install jenkins -y

- sudo curl -sSL https://get.docker.com/ | sh

- sudo usermod -aG docker azureuser

- sudo usermod -aG docker jenkins

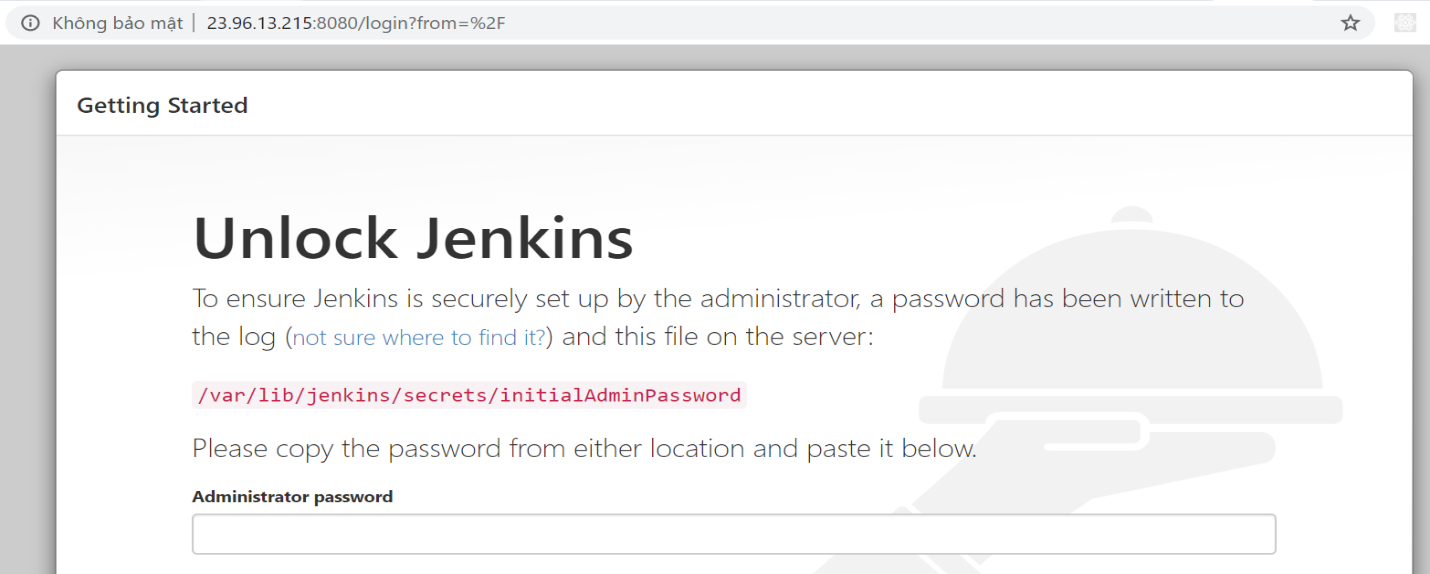
- sudo service jenkins restart

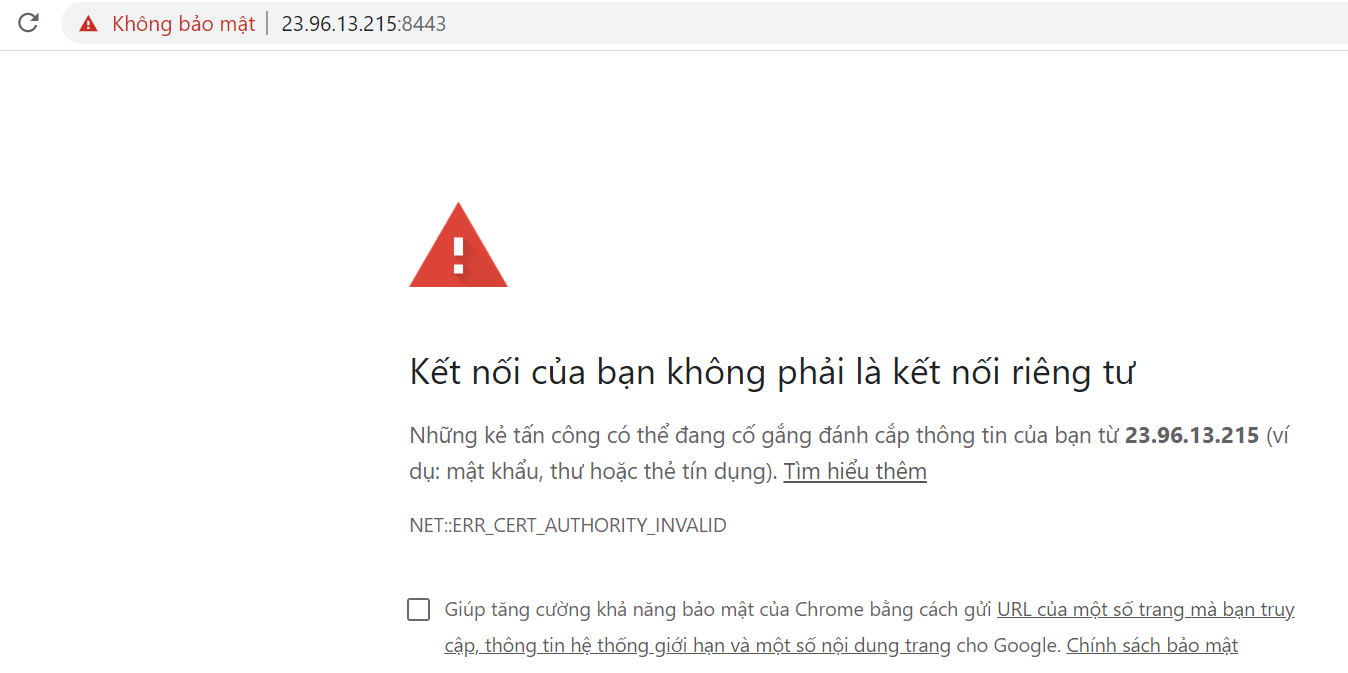
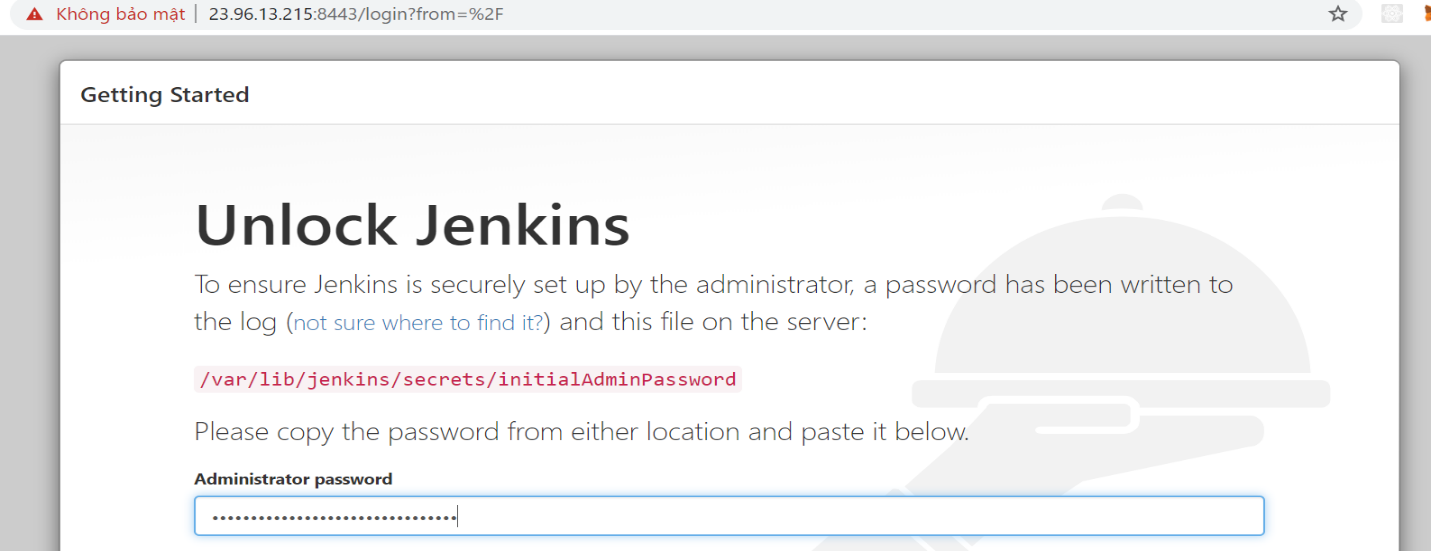
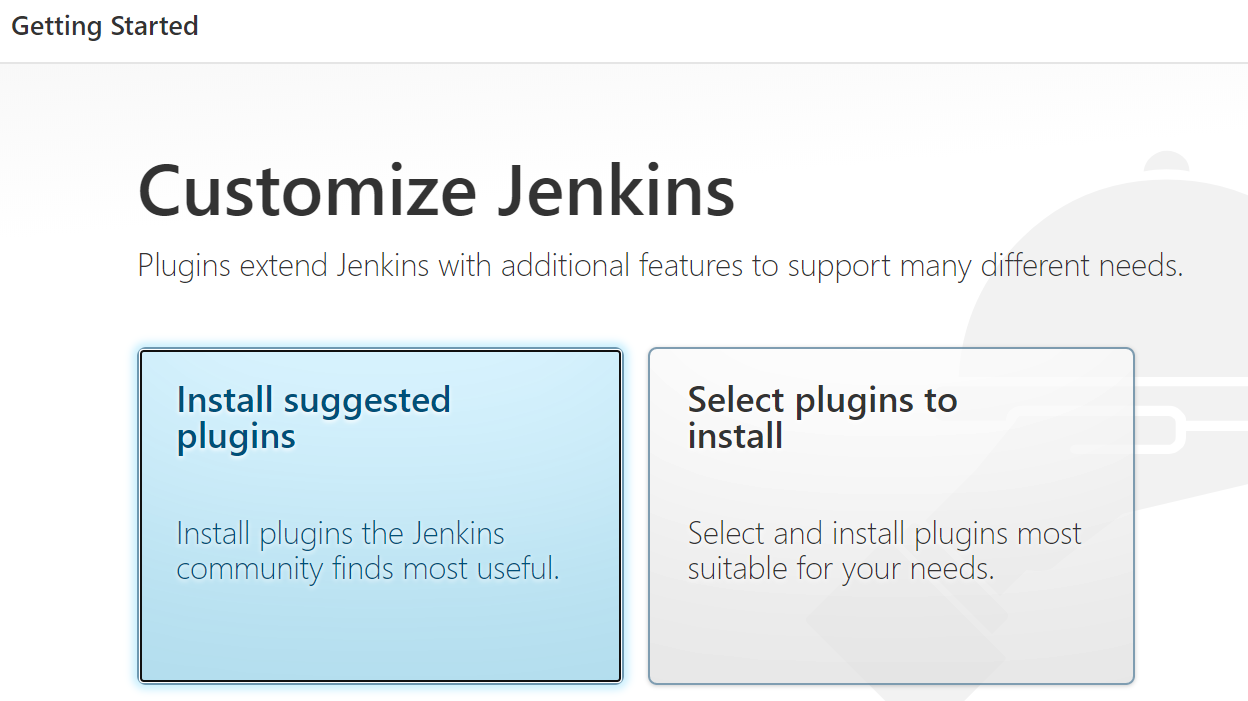
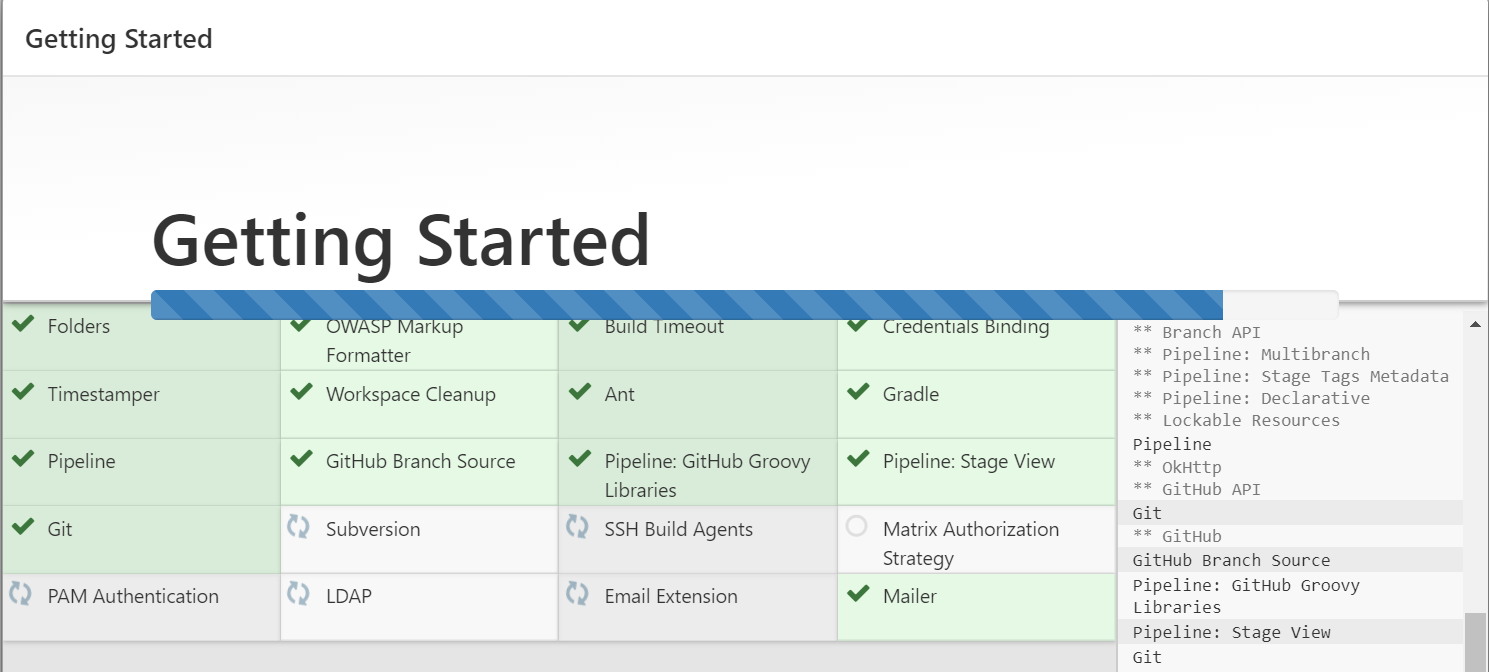
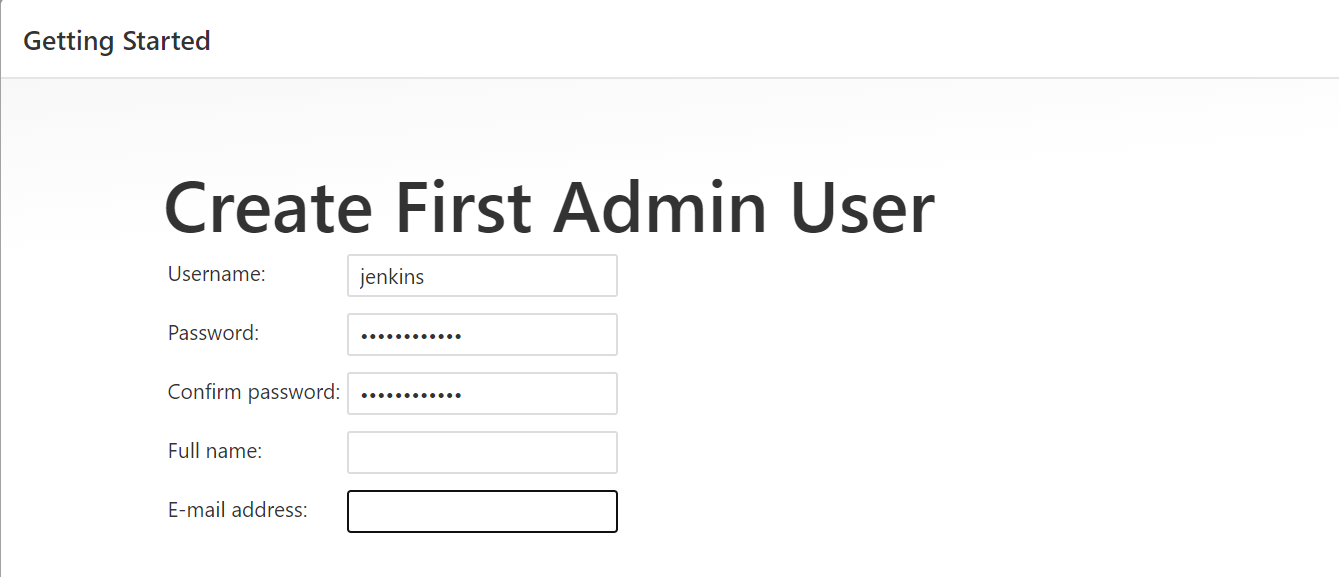
**Step 4**: Add inbound rules for port 8080 (http) and 8443 (https)  
  
  
  
  
  
**Step 5**: Check checking is working?  
- Get IP of Jenkins server:  
   
  
  
- Open browser and go to the http://ip\_address:8080

az vm show --resource-group jenkins-rg --name jenkins -d --query [publicIps] --o tsv

az vm open-port --resource-group jenkins-rg --name jenkins --port 8080 --priority 1001

az vm open-port --resource-group jenkins-rg --name jenkins --port 8443 --priority 1002

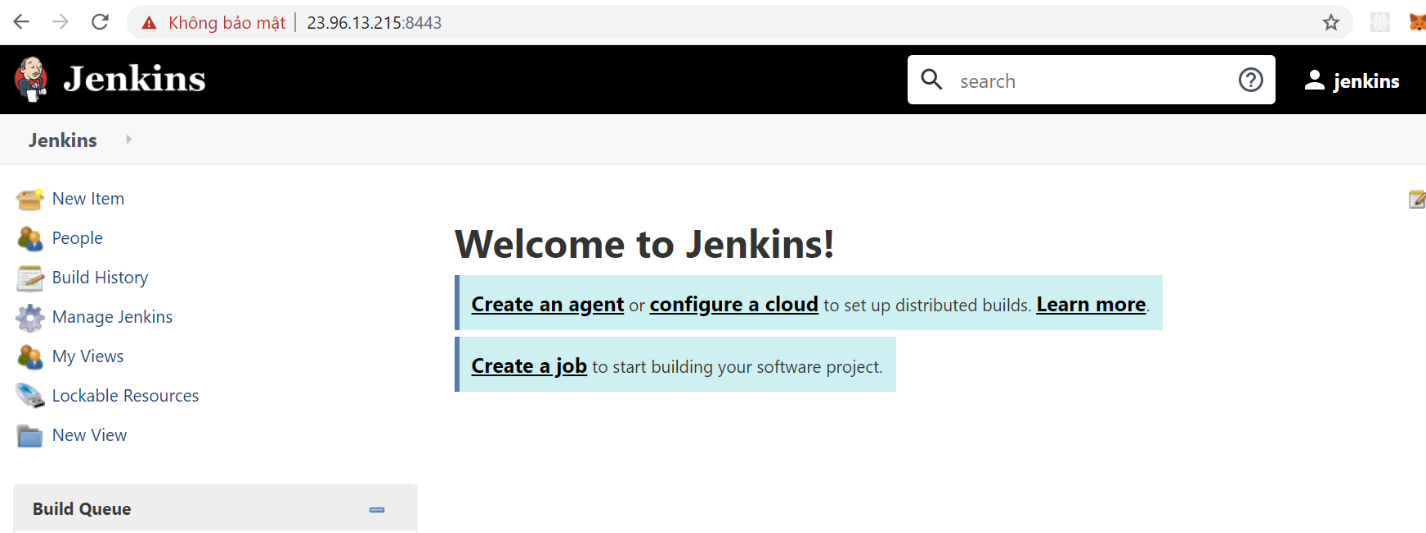
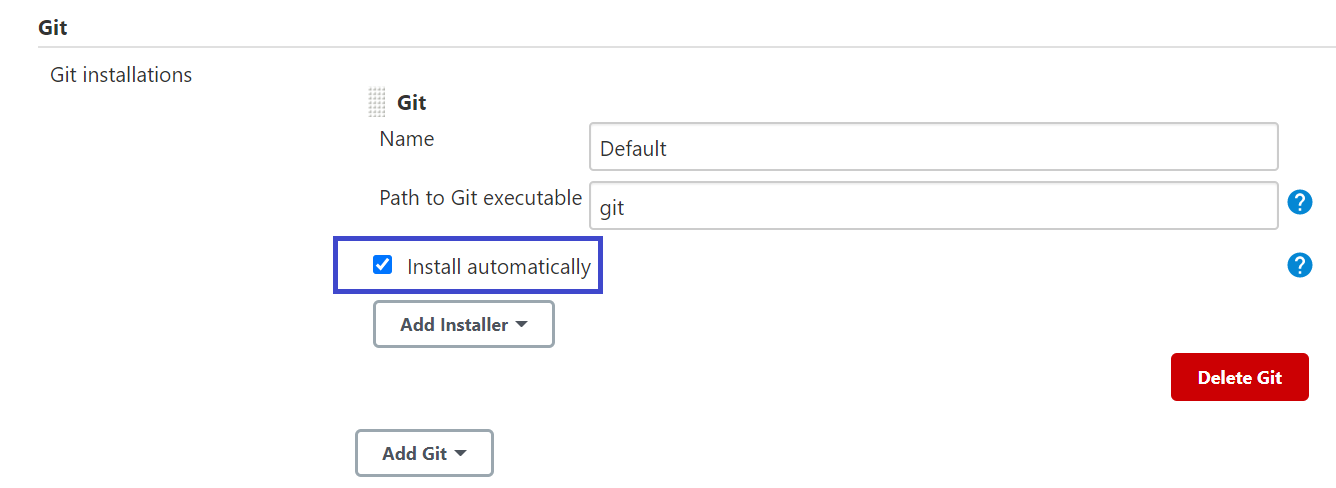
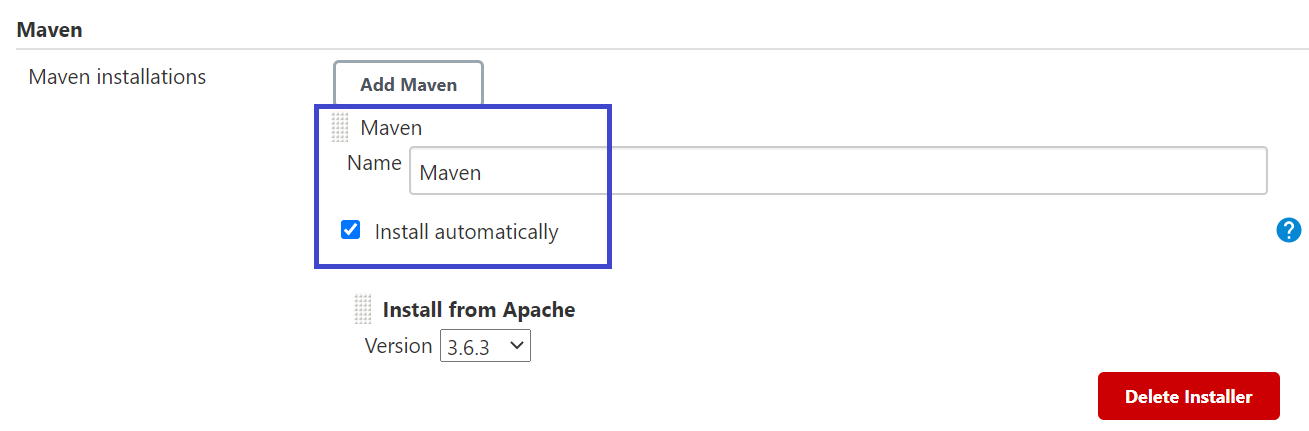
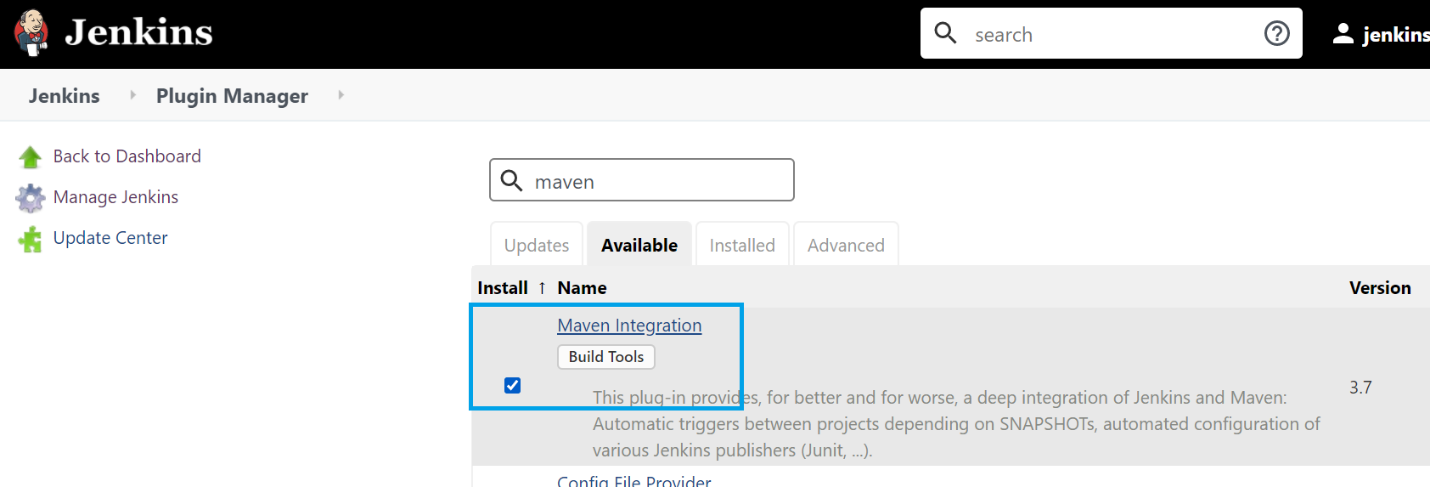


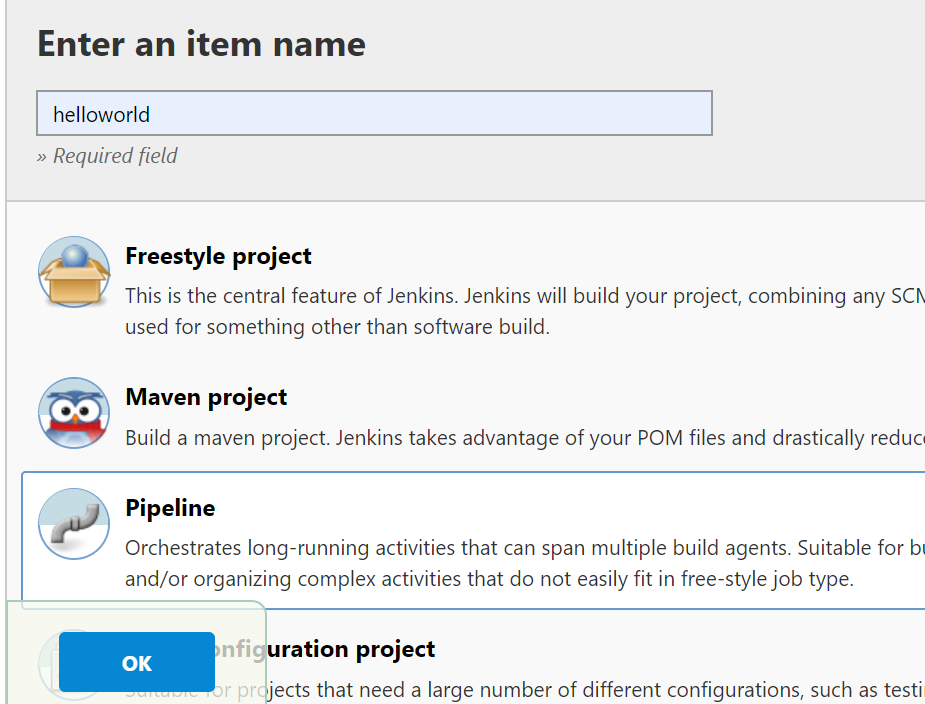
**Step 5:** Config HTTPS:  
- SSH to the Jenkins server and generate cerfiticate  
  
  
  
  
  
- Update Jenkins config   
  
  
  
  
  
- Restart Jenkins server  
  
  
  
- Open browser and check <https://ip_address:8443>  
  
  
**Step 7:** Config Jenkins server  
- Follow the Jenkins and add admin user  
  
- Install suggested plugins  
  
  
  
- Add admin user  


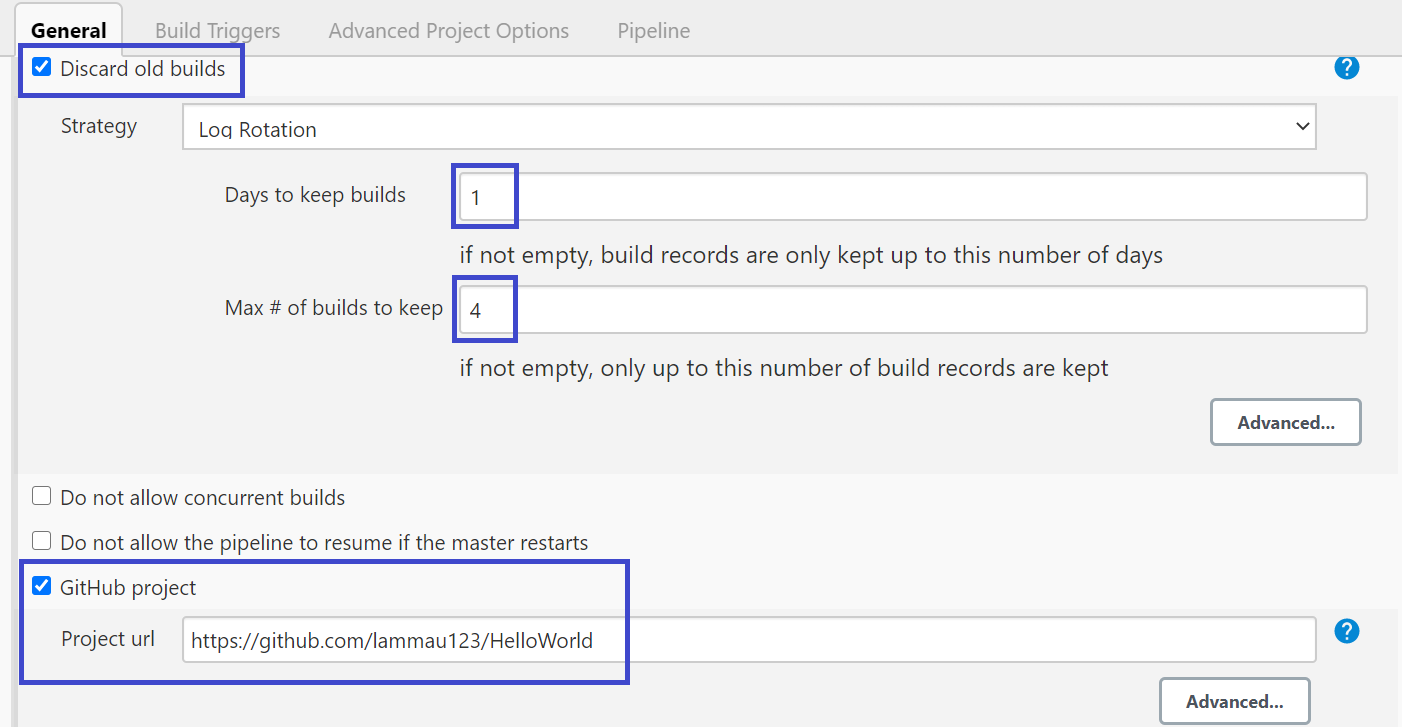
Sudo service Jenkins restart

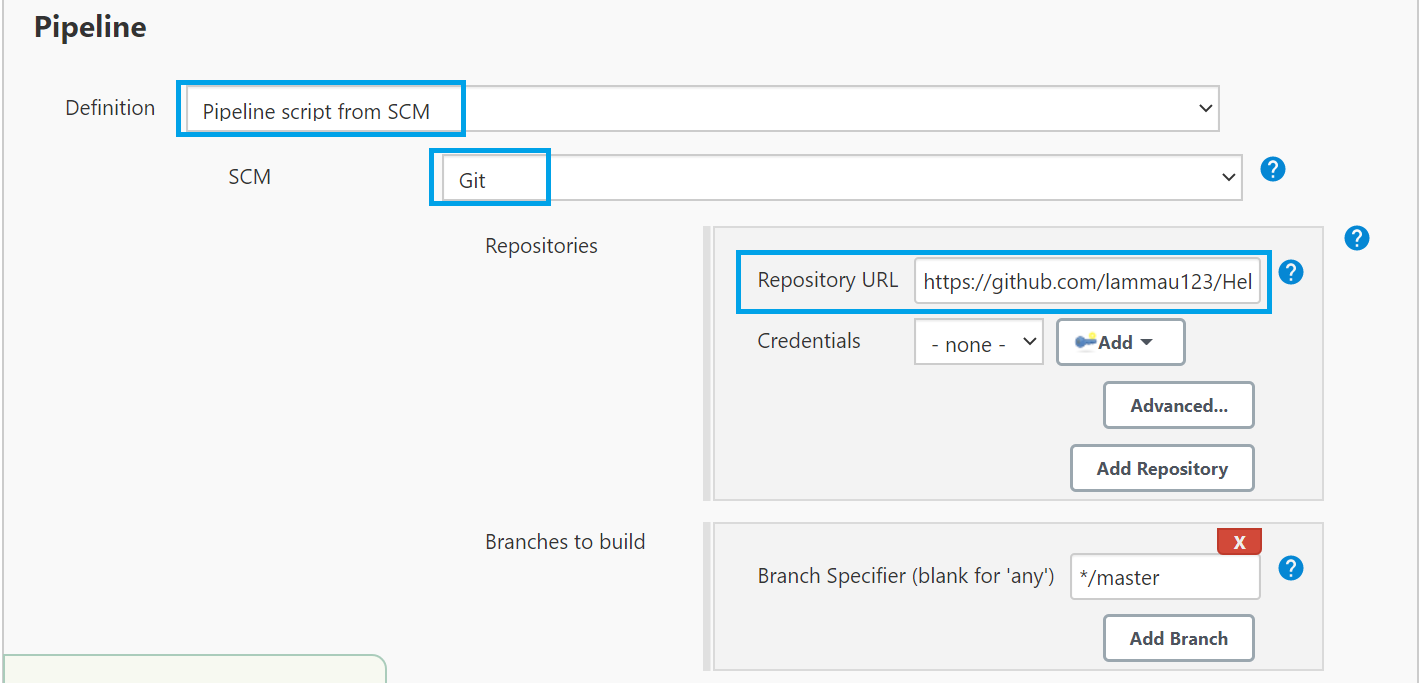
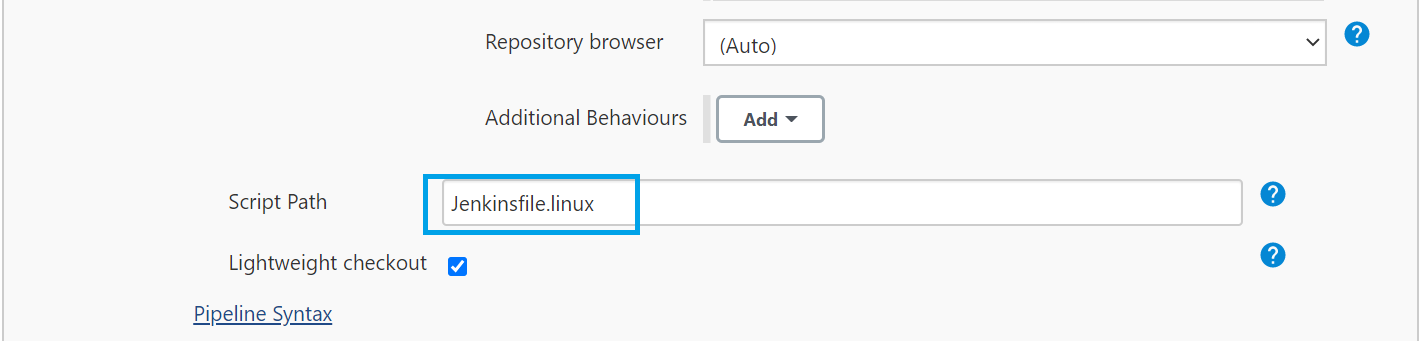
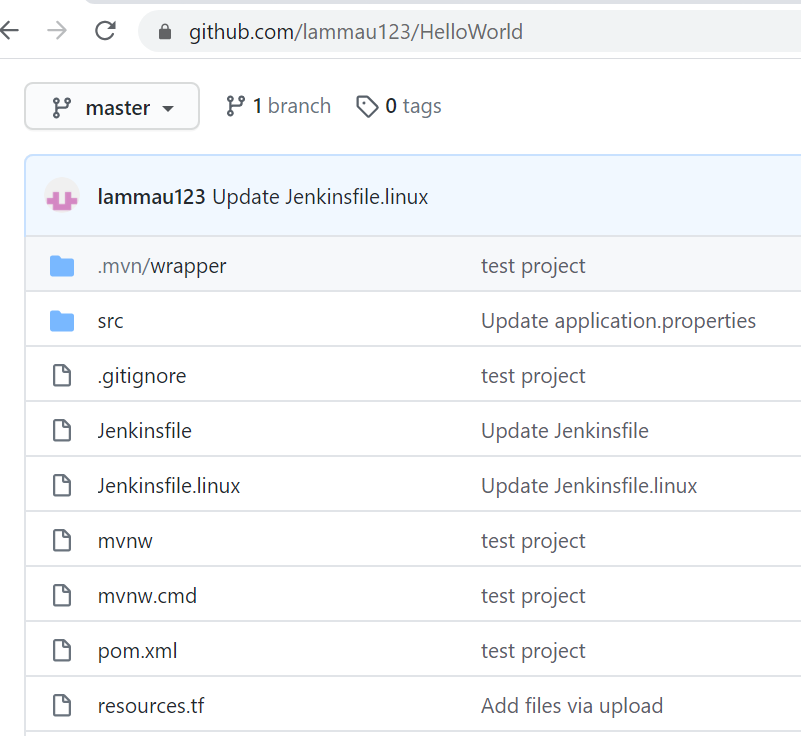
sudo vi /etc/default/jenkins

ssh azueruser@ip\_address  
keytool -genkey -v -keystore jenkins.keystore -alias alias\_name -keyalg RSA -keysize 2048 -validity 10000

  
  
- go to Manage Jenkins -> Global Tool Configuration: add git and maven  
  
  
  
- go to Manage Jenkins -> Manage Plugins : install maven integration  
  


**Step 8**: Create Job: go to Jenkins   
- Create job  
  
  
- Config build and github project



  
  
  
  
- Source git structure:  
  
  
**Step 9**: Creating ubuntu server to host the helloworld spring boot app.  
- Create ssh key pair

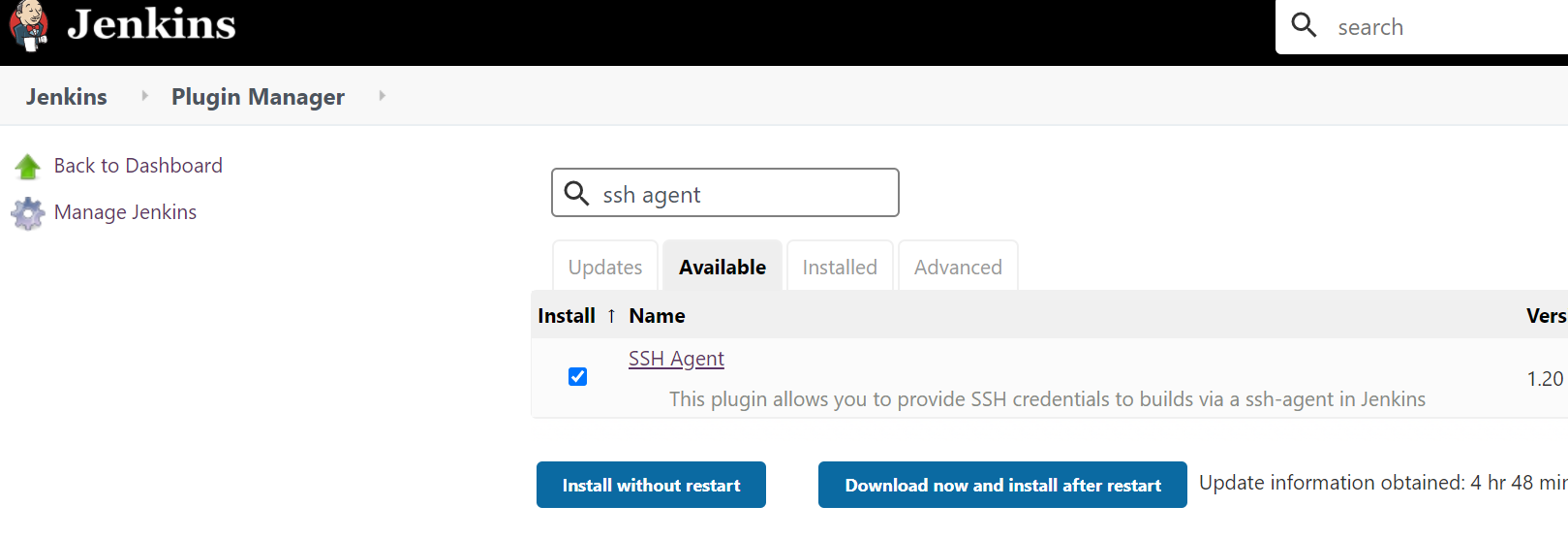
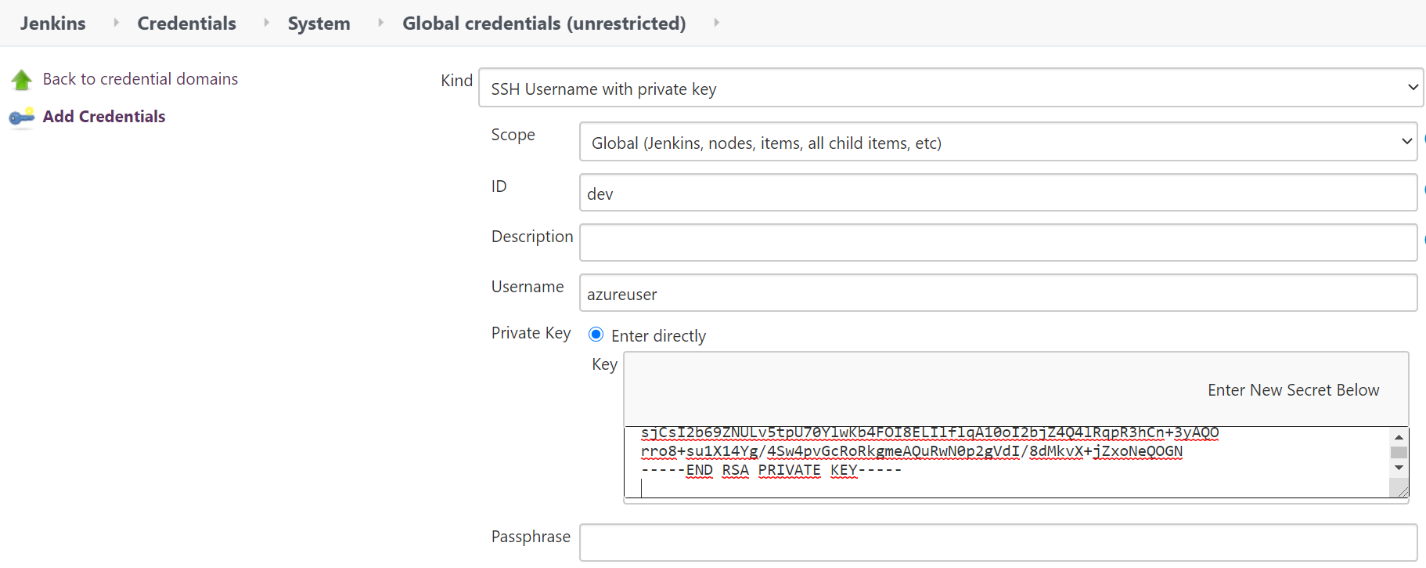
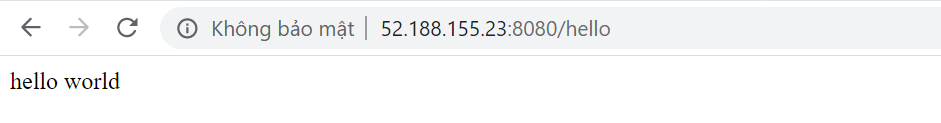
ssh-keygen -m PEM -t rsa -b 4096

- Create ubuntu server

az group create --name servers-rg --location eastus

az vm create --resource-group servers-rg --name rest\_servers --image Canonical:UbuntuServer:16.04-LTS:latest --admin-username azureuser --ssh-key-value rest\_server.pub --custom-data cloud-init-server.txt

az vm open-port --resource-group servers-rg --name rest\_servers --port 8080 --priority 1001

**cloud-init-server.txt**  
  
  
  
  
  
  
  
  
- Create dev folder on ubuntu server to deploy the app  
  
  
  
  
  
**Step 10:** Create Jenkins file  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
- start.sh  
  
  
  
  
  
**Step 11:** When Jenkins server build successfully and it will copy the jar file to ubuntu server and start the application server. Jenkins use ssh agent plugin to copy to ubuntu server.  
- Goto Jenkins -> Manage Jenkins -> Manage Plugins and install ssh agent  
  
- Go to Jenkins -> Manage Jenkins -> Manage Credential -> Add Credential and enter the private ssk key pair so Jenkins server can copy file to ubuntu server.  
  
  
**Step 12:** Click Build to check the result. After the build finishing, open browser and check <http://ip_address:8080/hello>  


#cloud-config

runcmd:

- sudo apt-get update

- sudo apt install openjdk-8-jre-headless -y

#cloud-config

runcmd:

- sudo apt-get update

- sudo apt install openjdk-8-jre-headless -y

az vm show --resource-group servers-rg --name rest\_servers -d --query [publicIps] --o tsv  
ssh -i rest\_server azureuser@ip\_address  
mkdir dev

pipeline {

agent any

tools {

maven 'Maven'

}

stages {

stage('Initialize') {

steps {

sh 'echo Initialzie'

}

}

stage ('Build') {

steps {

sh 'mvn clean package'

}

}

stage ('Deploy to Dev') {

steps {

sshagent (['dev']) {

sh 'scp -o StrictHostkeyChecking=no target/\*.jar azureuser@52.188.155.23:/home/azureuser/dev/'

sh 'ssh -tt azureuser@52.188.155.23 < start.sh'

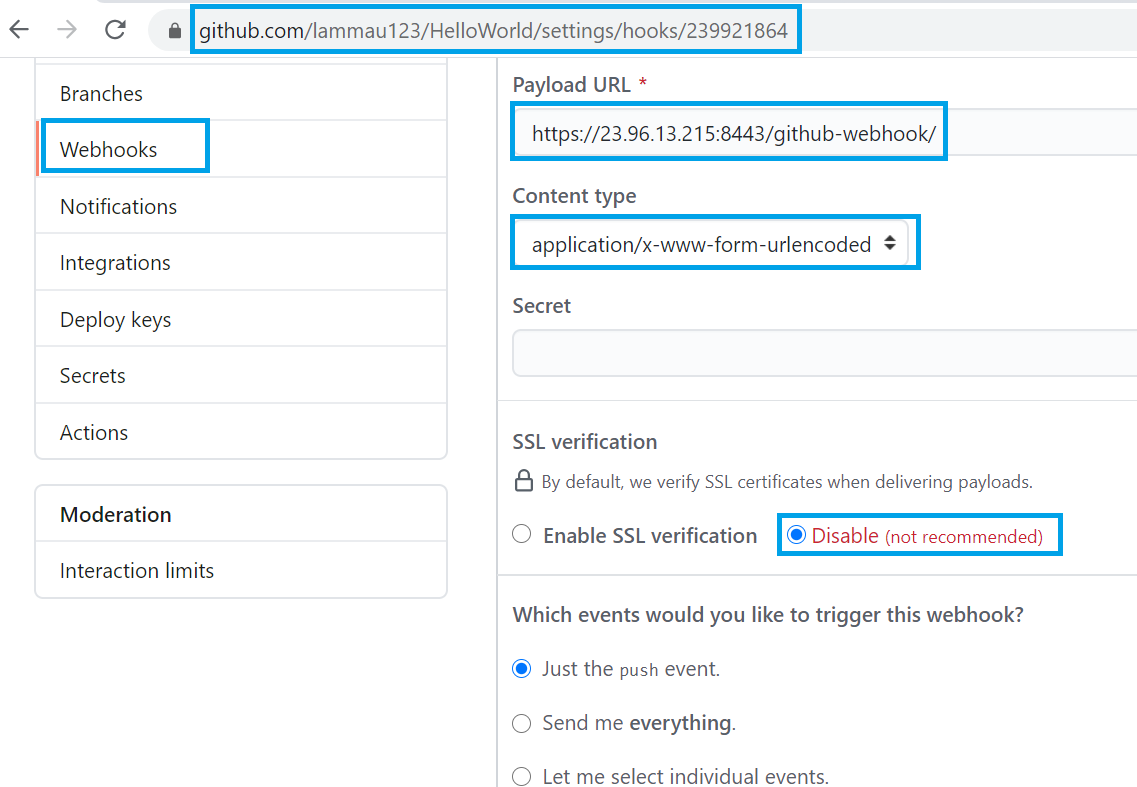
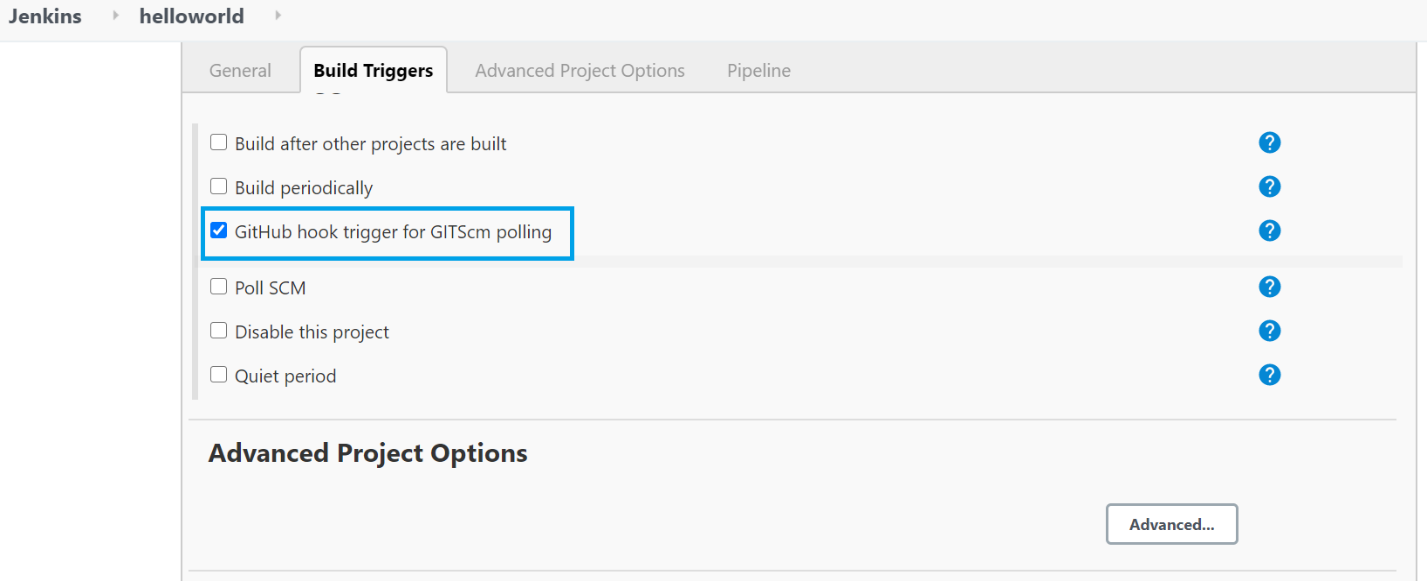
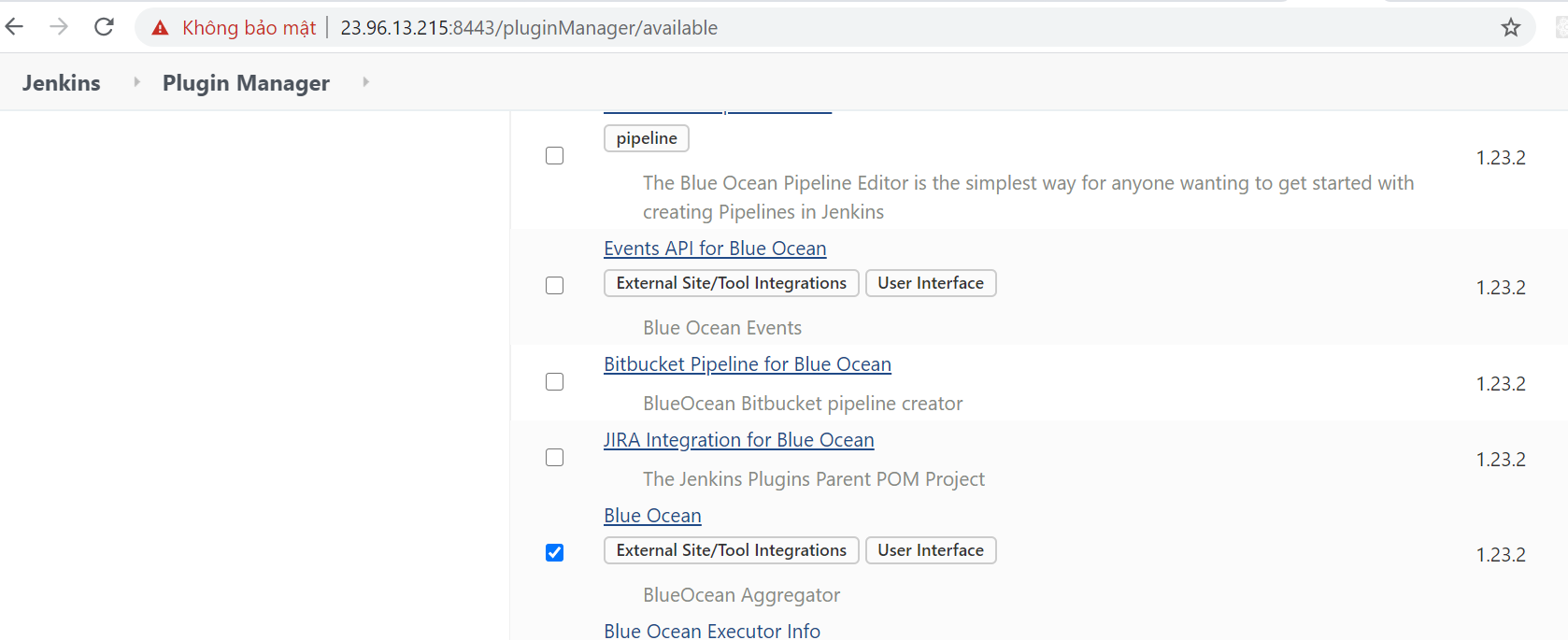
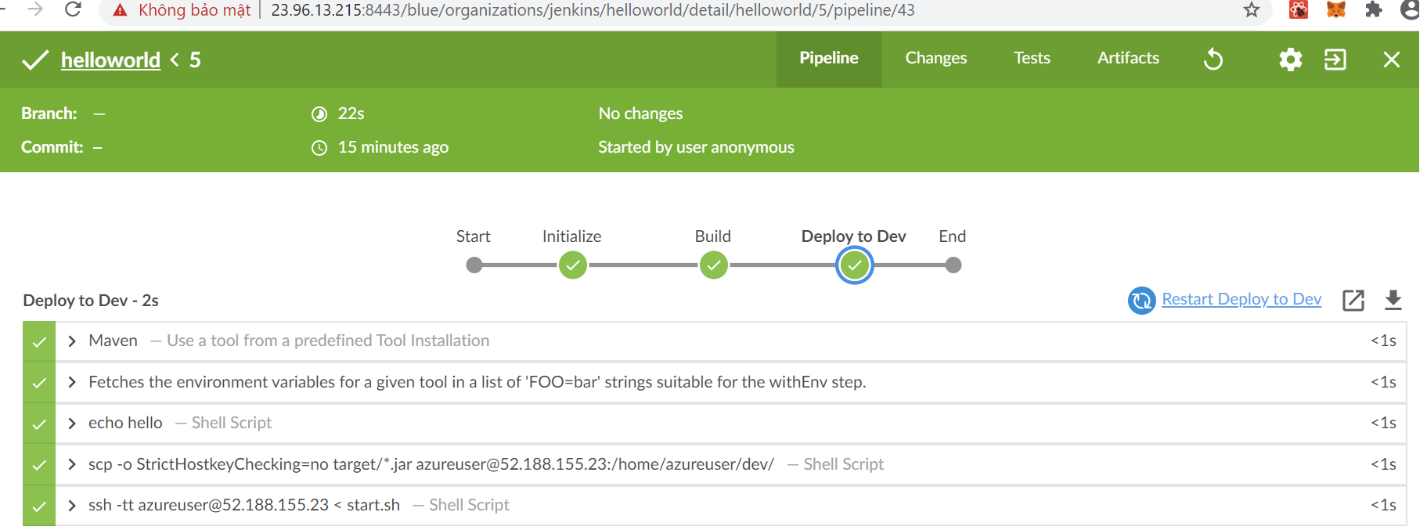
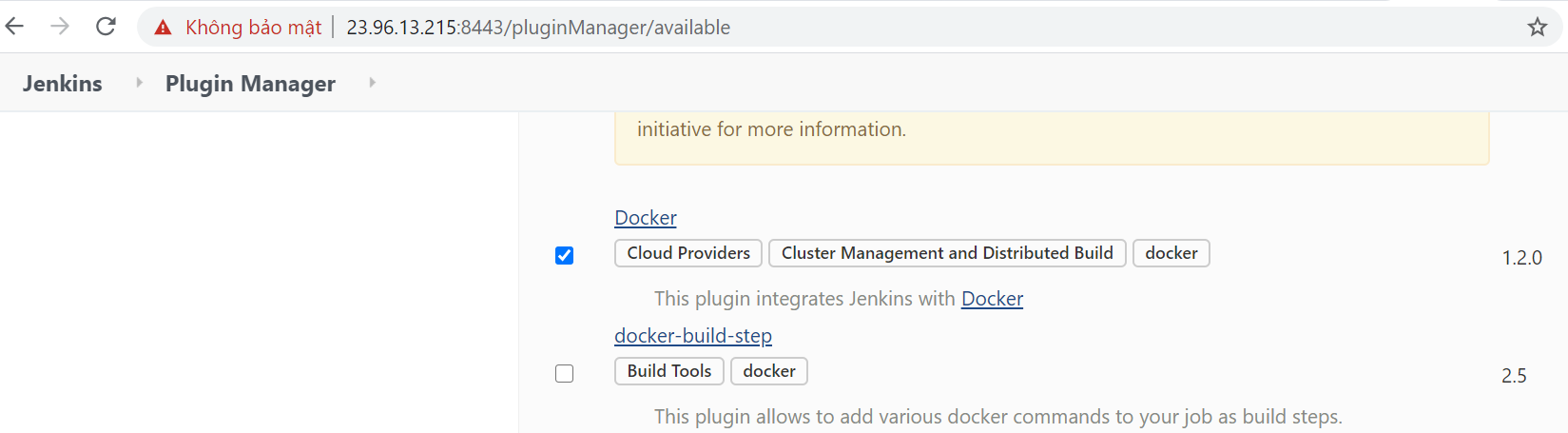
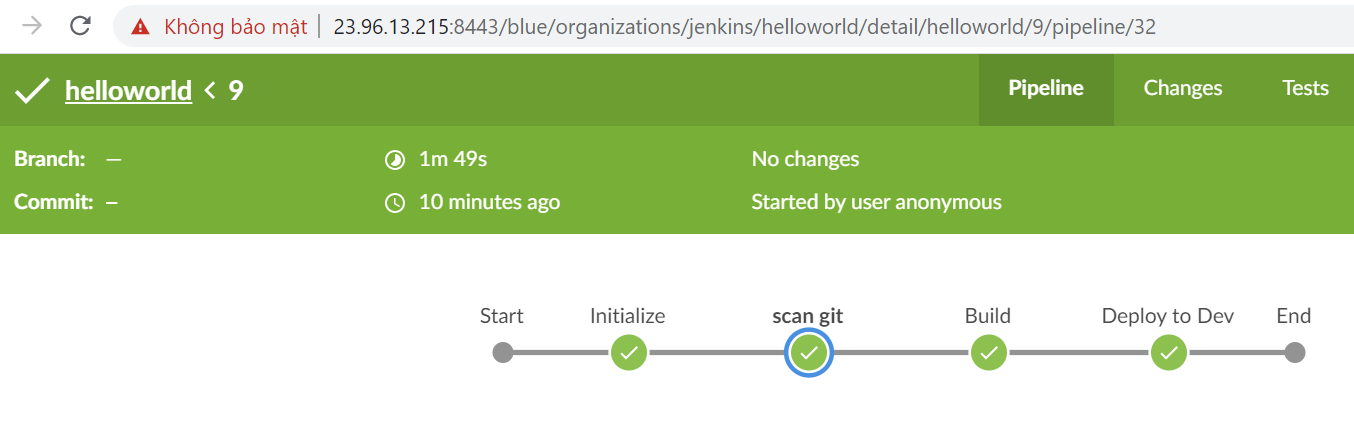
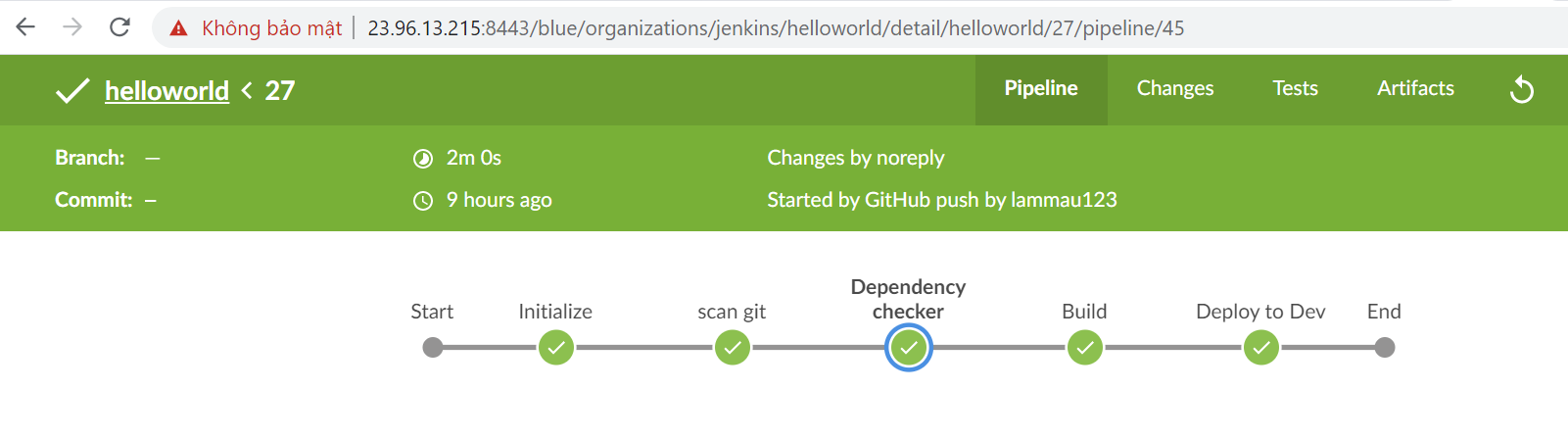
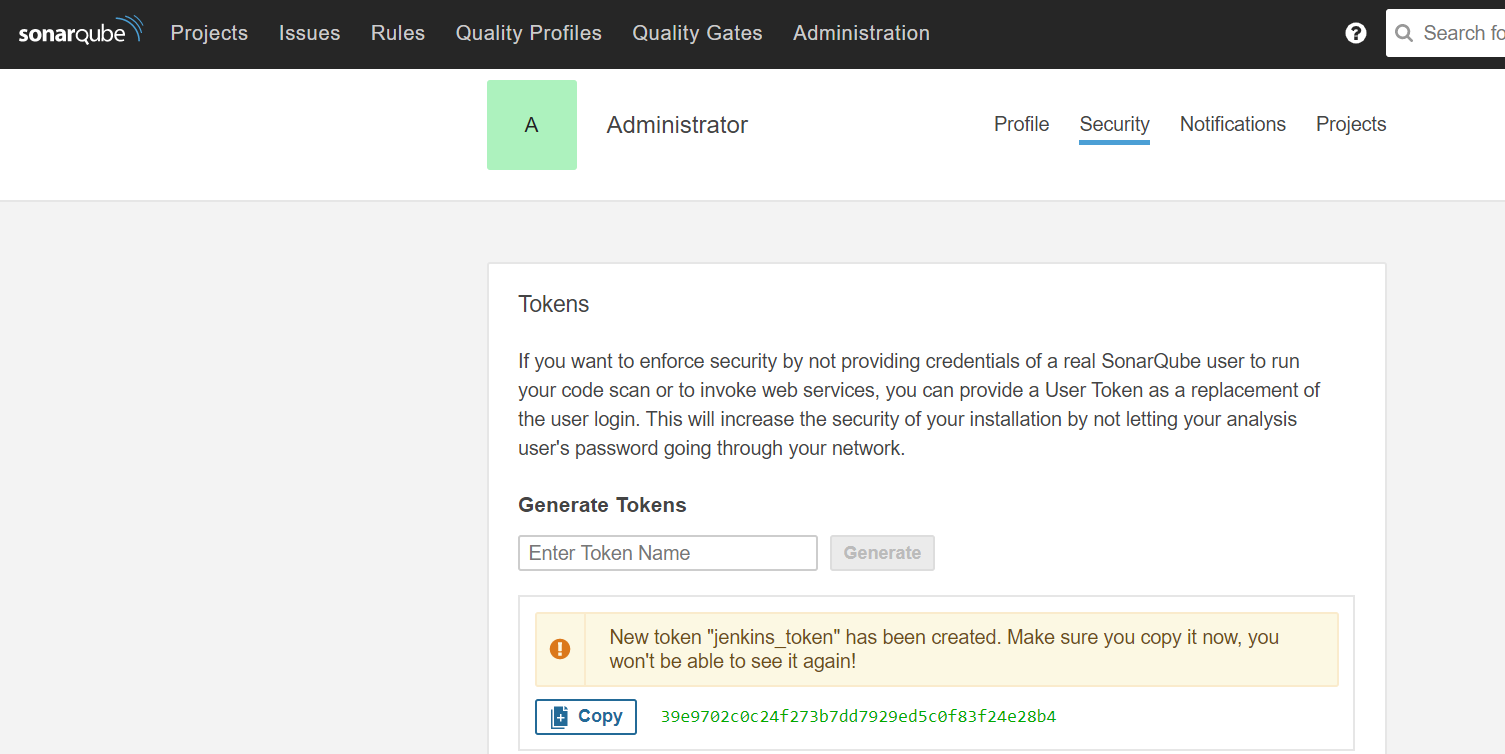
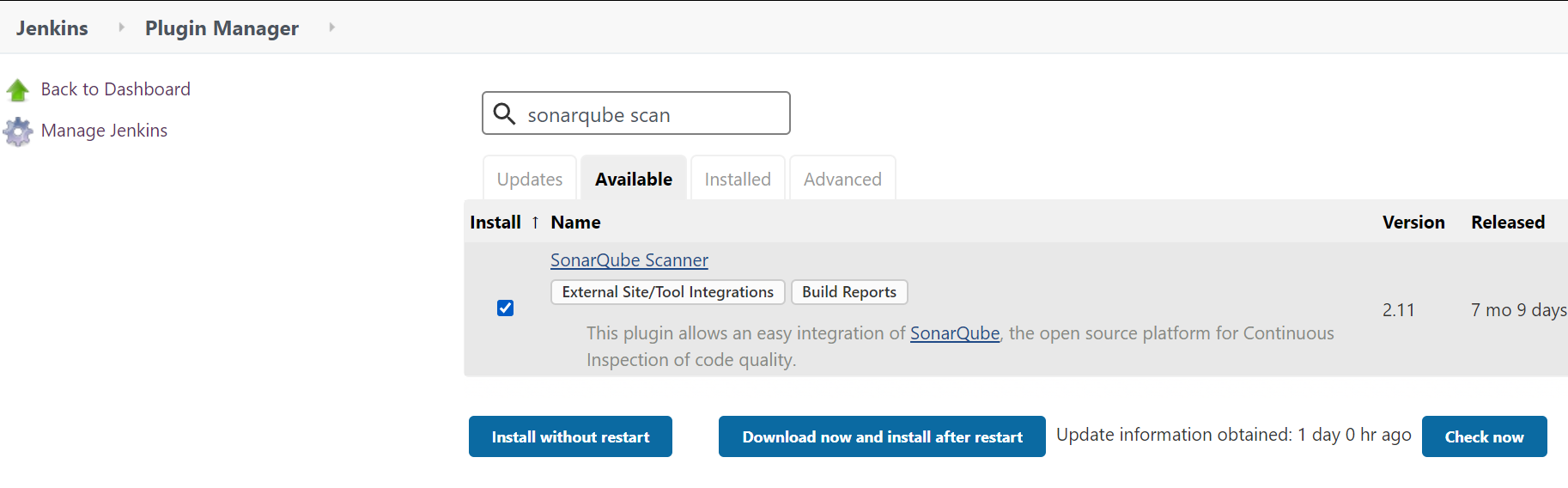
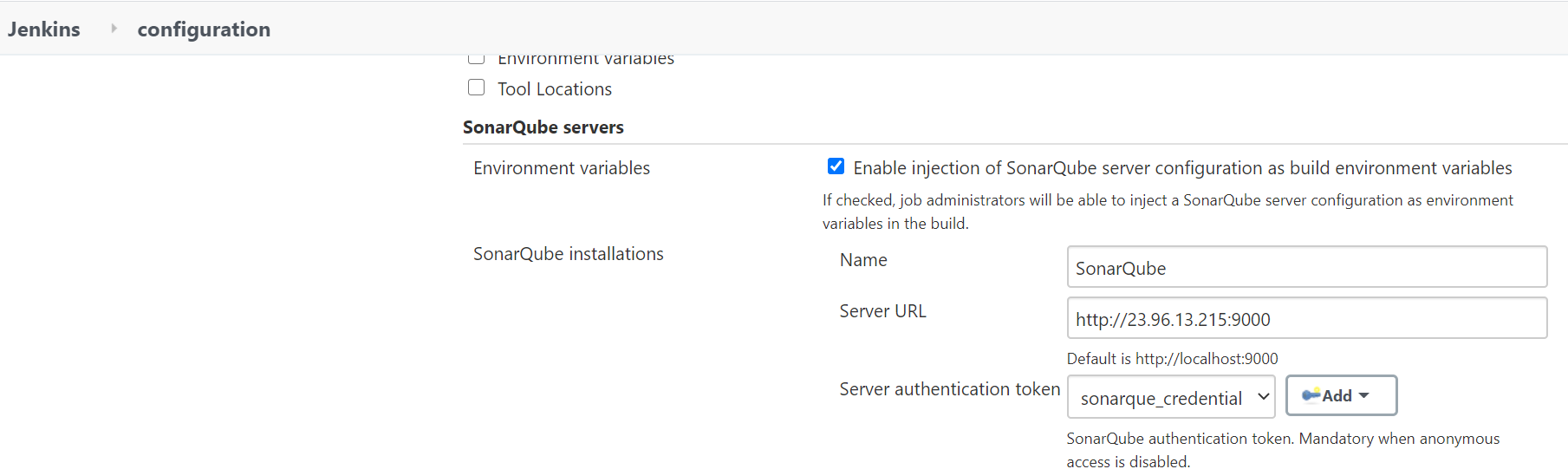
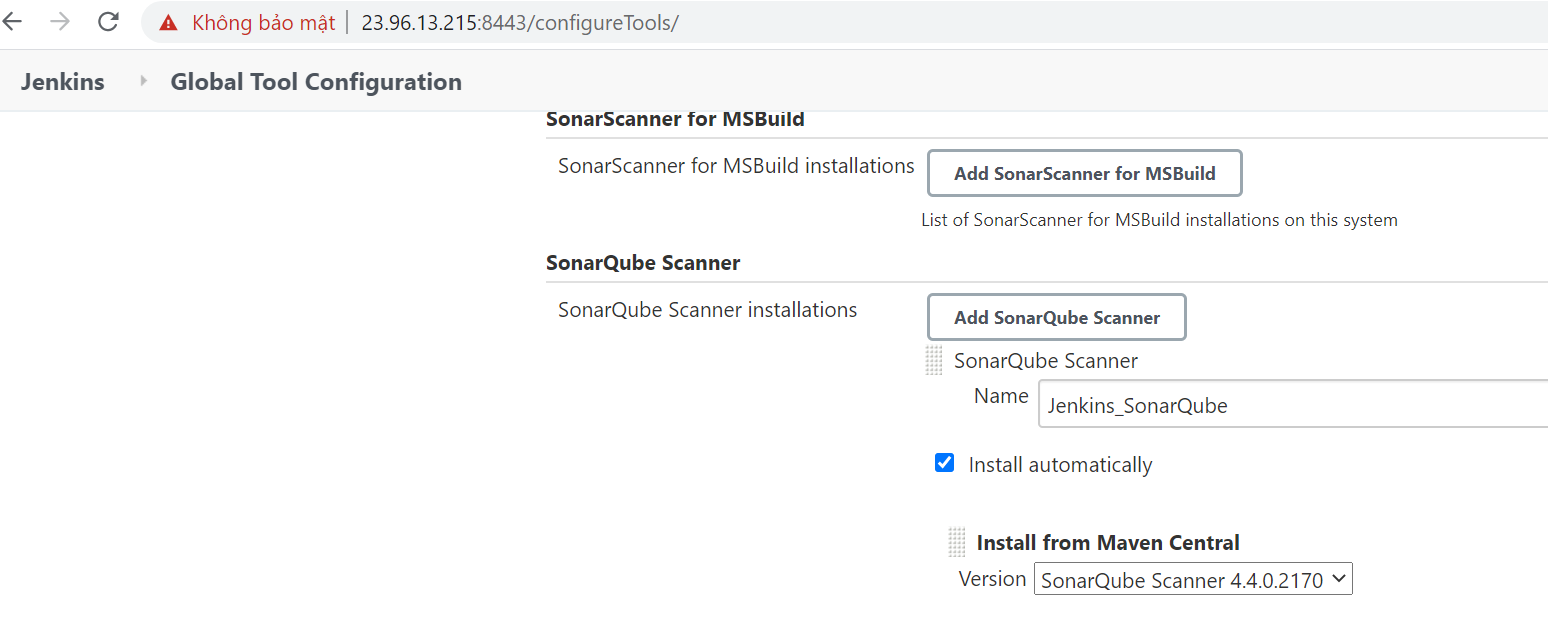
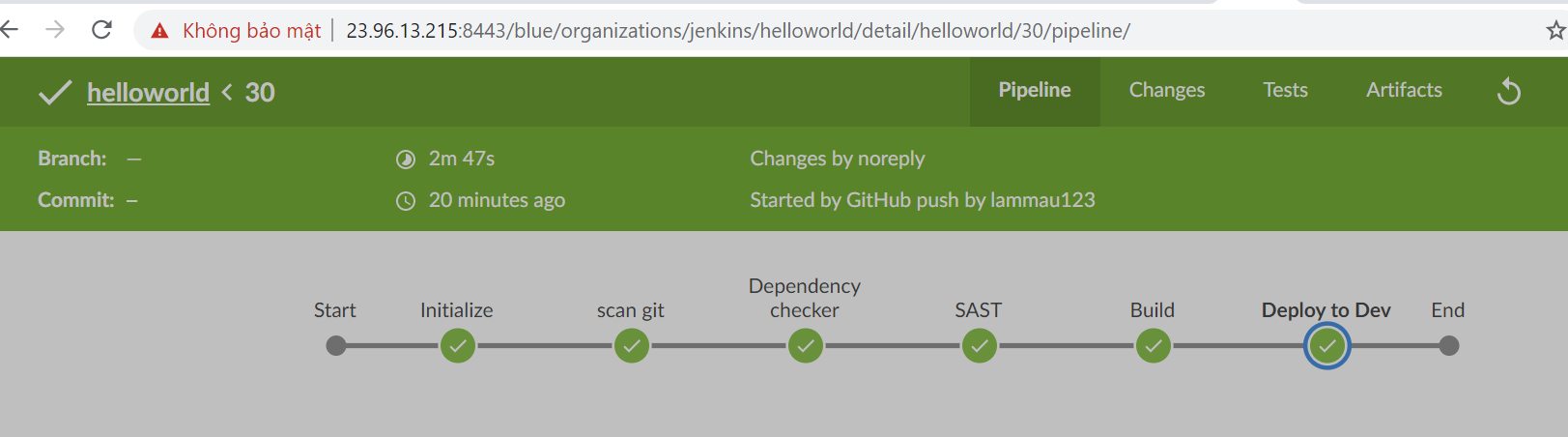
}

}

} }}

sudo nohup java -jar ./dev/\*.jar com.helloworld.test.HelloworldApplication > log.txt &

exit

Step 13: Config git webhook to push notification to Jenkins server when something change on git repo.  
- Login to git repos and add webhook point to Jenkins server:  
  
  
  
- Goto Jenkins -> helloworld -> Configure to config Jenkins trigger the build when receiving event from git  
  
  
 **Step 14:** Install Blue Ocean help manage Jenkins easier  
  
  
  
  
**- Step 14:** Check git push event:  
- Commit a change to git and see Jenkins will kick a build automatically  
- After the build finishing click on the build and click on the Blue Ocean   
  
  
Step 15: Applying Secrutity Scan for git repos  
- Go to Jenkins -> Manage Plugins and Install docker agent  
  
  
  
- Go to github and edit the Jenkins.linux file. Add the below step after initial step  
  
  
  
  
  
  
  
  
  
  
  
  
  
- After the build finished and check the result  
**Step 16:** Applying OWASP Dependency Checking  
- Dependency check script: add dependency\_check.sh below to repos  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
- Adding dependency check step to Jenkins.linux file before step build  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
- Check the build result in Blue Ocean  
  
**Step 17:** Static Application Security Testing (SAST) with SonarQube  
- SSH to the jenkins server and run SonarQube docker on Jenkins server.  
  
  
  
- Open port 9000  
  
  
- Login to sonarqube at http://ip\_address:9000 with System Administrator credentials (login=admin, password=admin). and generate token at My Account -> Security -> Generate Token  
  
  
- Install sonarqube scanner plugins: goto Jenkins -> Manage Jenkins -> Manage Plugins and search Sonar Scaner  
  
  
  
- Adding SonarQube: go to Jenkins -> Manager Jenkins -> Configure System  
  
  
- Configure SonarQube Scanner: Go to Jenkins -> Manager Jenkins -> Global Tool Configuration  
  
- Adding SonaQube to Jenkins.linux  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
- Check build result:  


stage ('scan git'){

steps {

sh 'rm trufflehog || true'

sh 'docker run gesellix/trufflehog --json https://github.com/lammau123/HelloWorld.git > trufflehog'

sh 'cat trufflehog'

}

}

#!/bin/sh

DC\_VERSION="latest"

DC\_DIRECTORY=$HOME/OWASP-Dependency-Check

DC\_PROJECT="dependency-check scan: $(pwd)"

DATA\_DIRECTORY="$DC\_DIRECTORY/data"

CACHE\_DIRECTORY="$DC\_DIRECTORY/data/cache"

if [ ! -d "$DATA\_DIRECTORY" ]; then

echo "Initially creating persistent directory: $DATA\_DIRECTORY"

mkdir -p "$DATA\_DIRECTORY"

fi

# Make sure we are using the latest version

docker pull owasp/dependency-check:$DC\_VERSION

docker run --rm \

-e user=$USER \

-u $(id -u ${USER}):$(id -g ${USER}) \

--volume $(pwd):/src:z \

--volume "$DATA\_DIRECTORY":/usr/share/dependency-check/data:z \

--volume $(pwd)/odc-reports:/report:z \

owasp/dependency-check:$DC\_VERSION \

--scan /src \

--format "ALL" \

--project "$DC\_PROJECT" \

--out /report

# Use suppression like this: (where /src == $pwd)

# --suppression "/src/security/dependency-check-suppression.xml"

stage ('Dependency checker') {

steps {

sh 'mkdir odc-reports || true'

sh 'rm ./odc-reports/\* || true'

sh 'chmod +x dependency\_check.sh'

sh 'bash dependency\_check.sh'

sh 'cat ./odc-reports/dependency-check-report.xml'

}

}

docker run -d --name sonarqube -p 9000:9000 <image\_name>

az vm open-port --resource-group jenkins-rg --name jenkins --port 9000 --priority 1003

stage('SAST') {

steps {

withSonarQubeEnv('SonarQube') {

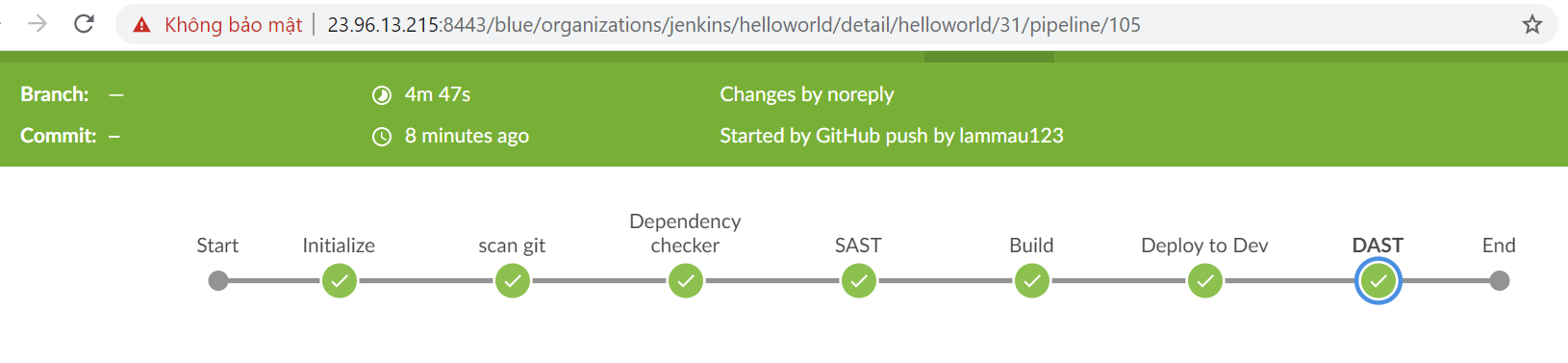
sh 'mvn sonar:sonar'

sh 'cat target/sonar/report-task.txt'

}

}

}

**Step 18:** Dynamic Application Security Testing (SAST) with ZAP **-** Adding ZAP step to Jenkins.linux   
  
  
  
  
  
  
  
  
  
- Check build result in Blue Ocean  
****

stage ('DAST') {

steps {

sh 'docker run -t owasp/zap2docker-stable zap-baseline.py -t http://52.188.155.23:8080/ || true'

}

}