Jenkins Installation: Free Style Pipeline, Maven Pipeline

--------------------------------------------------------------------------

Prerequisite: Java must be installed

Package: java-11-amazon-corretto-headless

Git Install for Source Code management

Package: yum install git -y

<https://www.jenkins.io/doc/book/installing/linux/>

sudo wget -O /etc/yum.repos.d/jenkins.repo \

https://pkg.jenkins.io/redhat-stable/jenkins.repo

sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key

sudo yum install jenkins

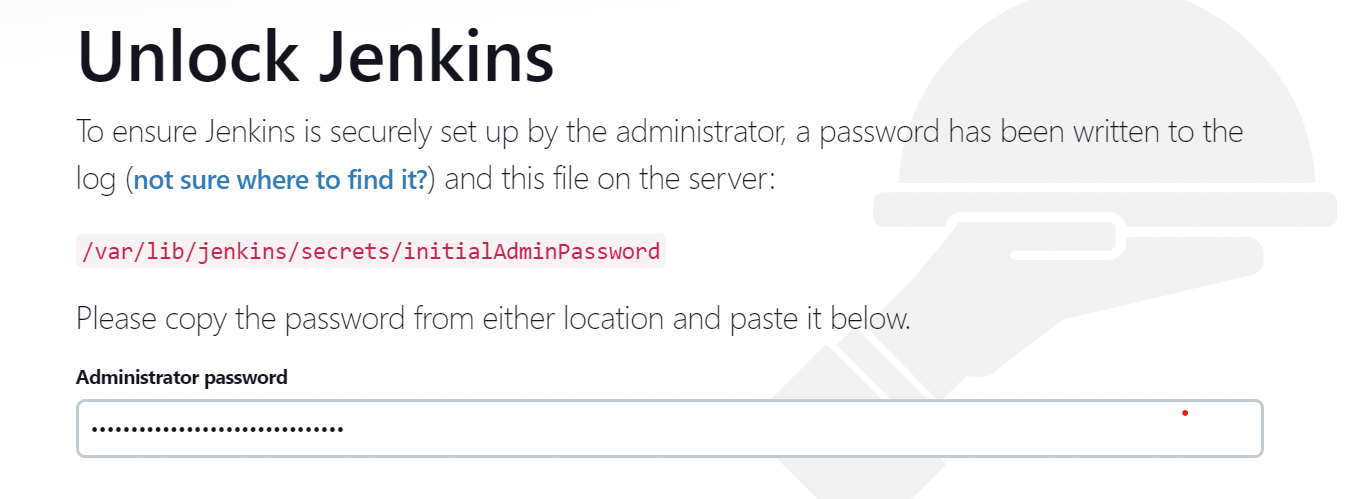
sudo systemctl daemon-reload

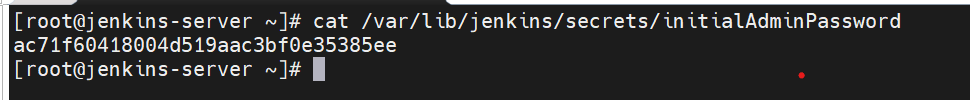
sudo systemctl enable jenkins

sudo systemctl start jenkins

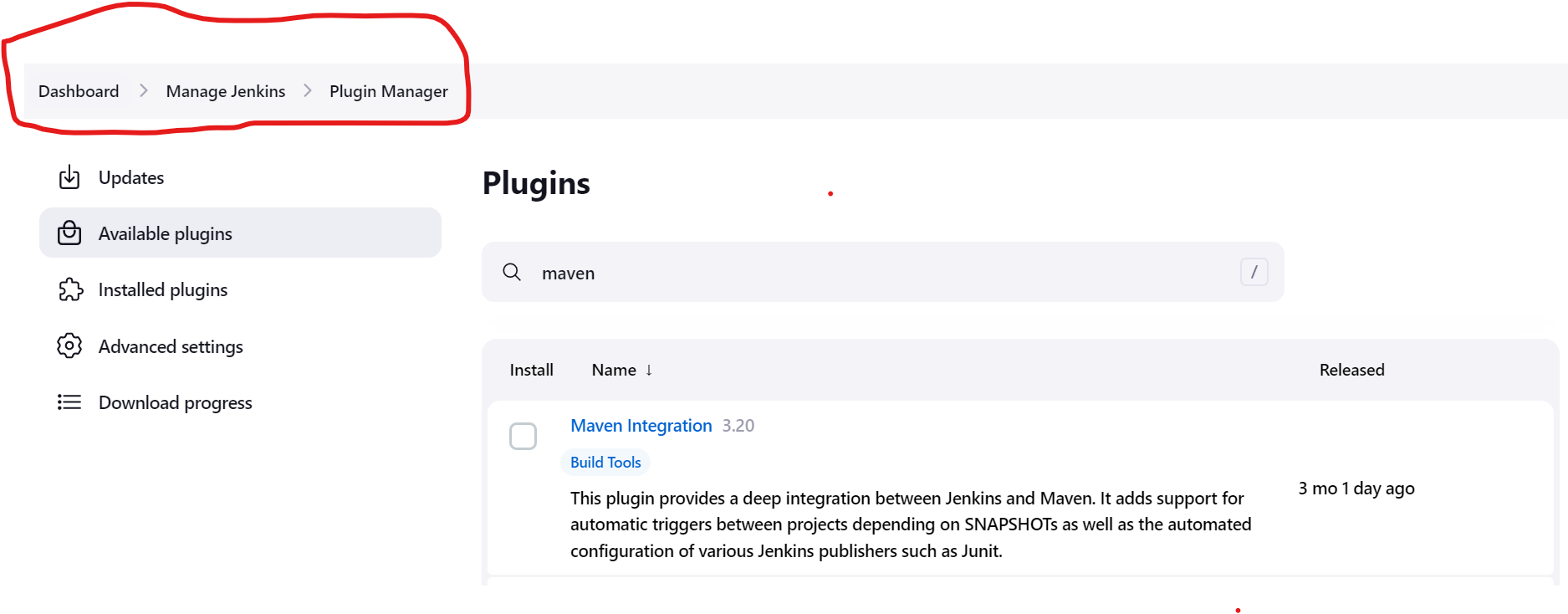
Note: Port 8080 must be open in security group.

<http://13.233.92.63:8080>





Install The Plugins In Jenkins-

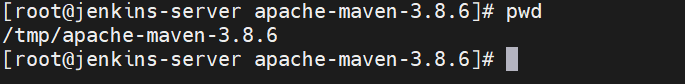


Install the Maven on Ec2 server:

<https://archive.apache.org/dist/maven/maven-3/>

wget <https://archive.apache.org/dist/maven/maven-3/3.8.6/binaries/apache-maven-3.8.6-bin.tar.gz>

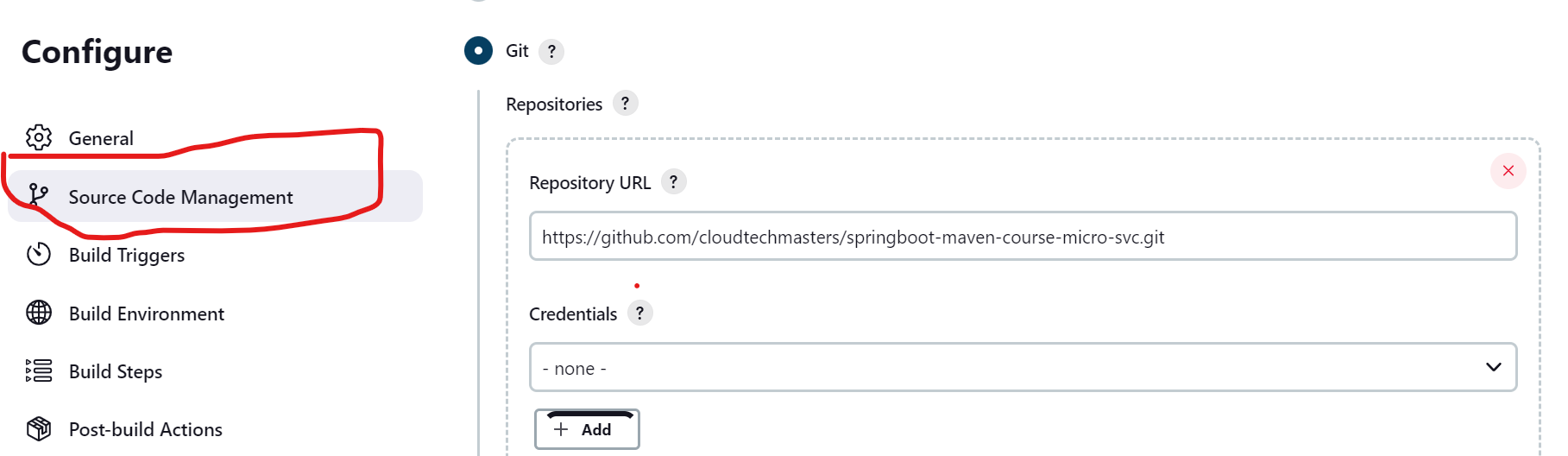
tar -xvf apache-maven-3.8.6-bin.tar.gz

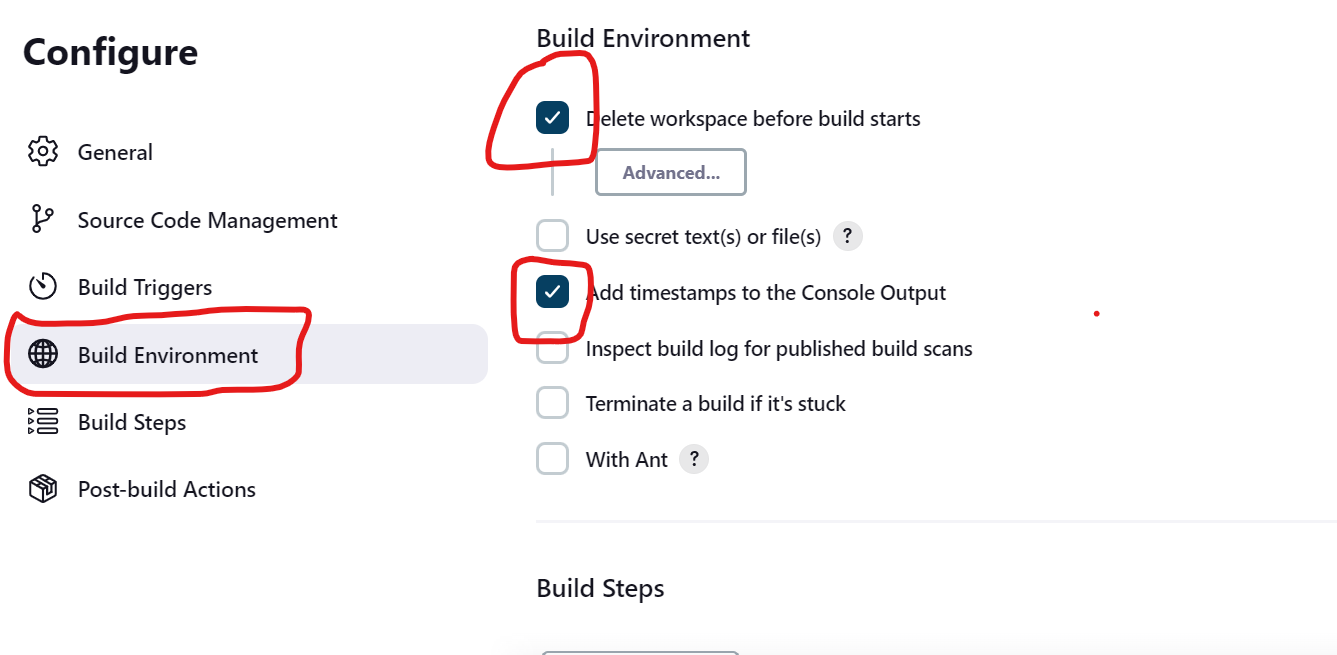


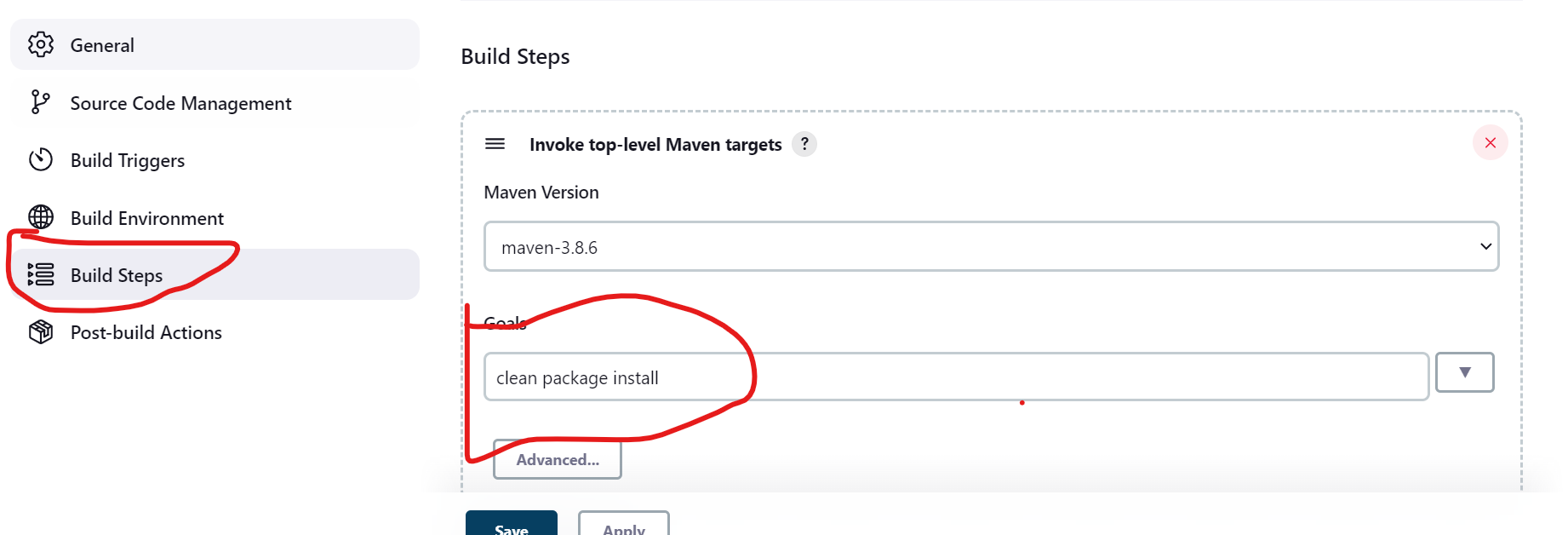
Install the maven tool in jenkins:

[Dashboard](http://13.233.92.63:8080/)🡪 [Manage Jenkins](http://13.233.92.63:8080/manage/)🡪 Global Tool Configuration

Create Pipeline: Free Style



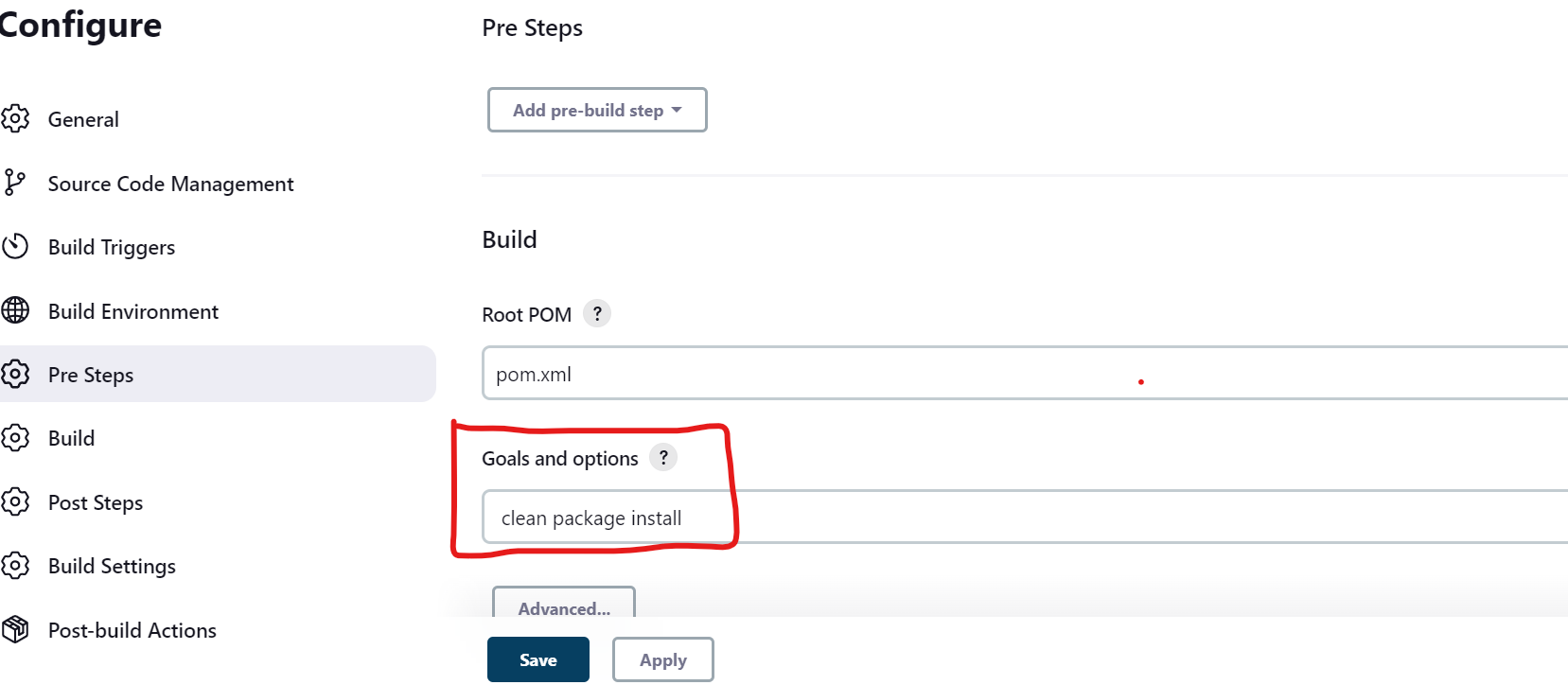




**clean package install -f pom.xml**

**/tmp/apache-maven-3.8.6/bin/mvn clean package install**

**Maven project Free-Style-Maven-Project**

** clean package install**

# 

# Project Gradle-Free-Style

First install the gradle in EC2 server

<https://distfiles.macports.org/gradle/>

wget <https://distfiles.macports.org/gradle/gradle-7.6-bin.zip>

unzip gradle-7.6-bin.zip

**Add the Gradle in global too configuration in Jenkins**

[Dashboard](http://13.233.92.63:8080/)🡪 [Manage Jenkins](http://13.233.92.63:8080/manage/)🡪 Global Tool Configuration

****

# Project DotNet Free Style

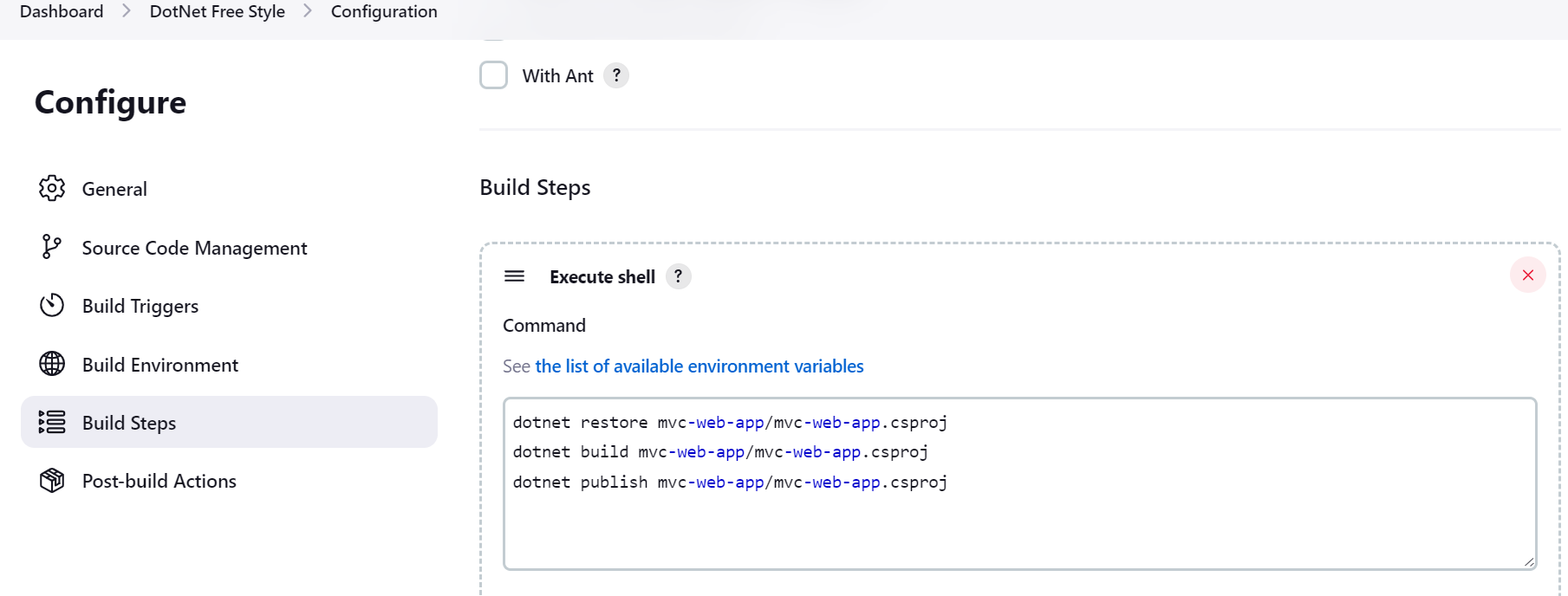
Install the dot in ec2 instance:

<https://docs.servicestack.net/deploy-netcore-to-amazon-linux-2-ami>

sudo rpm -Uvh <https://packages.microsoft.com/config/centos/7/packages-microsoft-prod.rpm>

sudo yum install dotnet-sdk-6.0

git Repo: <https://github.com/cloudtechmasters/dotnet-project-repo.git>



**dotnet restore mvc-web-app/mvc-web-app.csproj**

**dotnet build mvc-web-app/mvc-web-app.csproj**

**dotnet publish mvc-web-app/mvc-web-app.csproj**

NodeJs Free Style Project

Install the nodejs on ec2 server

<https://techviewleo.com/how-to-install-nodejs-on-amazon-linux/>

# Pipeline Declarative-Pipeline-maven:

https://www.jenkins.io/blog/2017/02/07/declarative-maven-project/

*Jenkinsfile (Declarative Pipeline)*

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'This is a minimal pipeline.'

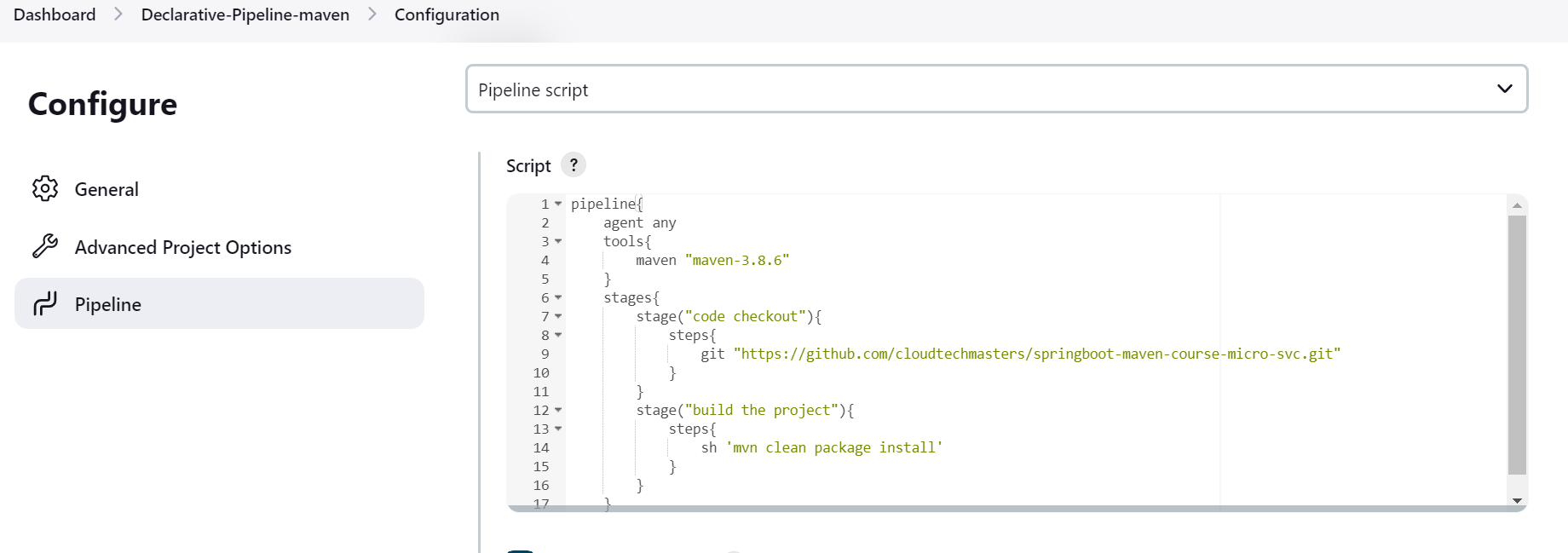
}

}

}

}

* All Declarative Pipelines start with a pipeline section.
* Select where to run this Pipeline, in this case "any" agent, regardless of label.
* Declarative automatically performs a checkout of source code on the agent, whereas Scripted Pipeline users must explicitly call checkout scm,
* A Declarative Pipeline is defined as a series of stages.
* Run the "Build" stage.
* Each stage in a Declarative Pipeline runs a series of steps.
* Run the echo step to print a message in the Console Output.



Maven Declarative Pipe line

**pipeline{**

**agent any**

**tools{**

**maven "maven-3.8.6"**

**}**

**stages{**

**stage("code checkout"){**

**steps{**

**git "https://github.com/cloudtechmasters/springboot-maven-course-micro-svc.git"**

**}**

**}**

**stage("build the project"){**

**steps{**

**sh 'mvn clean package install'**

**}**

**}**

**}**

**}**

# Pipeline Gradle Declarative Pipeline

****

**pipeline{**

**agent any**

**tools{**

**gradle "gradle-7.6"**

**}**

**stages{**

**stage("git clone"){**

**steps{**

**git "https://github.com/cloudtechmasters/nov13-gradle-demo.git"**

**}**

**}**

**stage("project build"){**

**steps{**

**sh 'gradle clean build'**

**}**

**}**

**}**

**}**

**Integrate Jenkins with SonarQube for code quality Test:**

**-------------------------------------------------------------------------------**

**Sonar Server:**

Java must be installed to run the sonar

sudo yum install java-11-amazon-corretto-headless -y

sudo hostnamectl set-hostname Sonar-Server

sudo -i

google: download sonarqube

wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.8.0.63668.zip>

Imp:

1. Its not recommended to run the sonar with root user, we need to change the ownwership of sonar folder to normal user like ec2-user or we can create the dedicated user name like sonar

Let download the sonar in /opt folder

Cd /opt

wget <https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.8.0.63668.zip>

unzip sonarqube-9.8.0.63668

/tmp/sonarqube-9.8.0.63668/bin/linux-x86-64/sonar.sh start

Sonar port 9000 must be open in security group

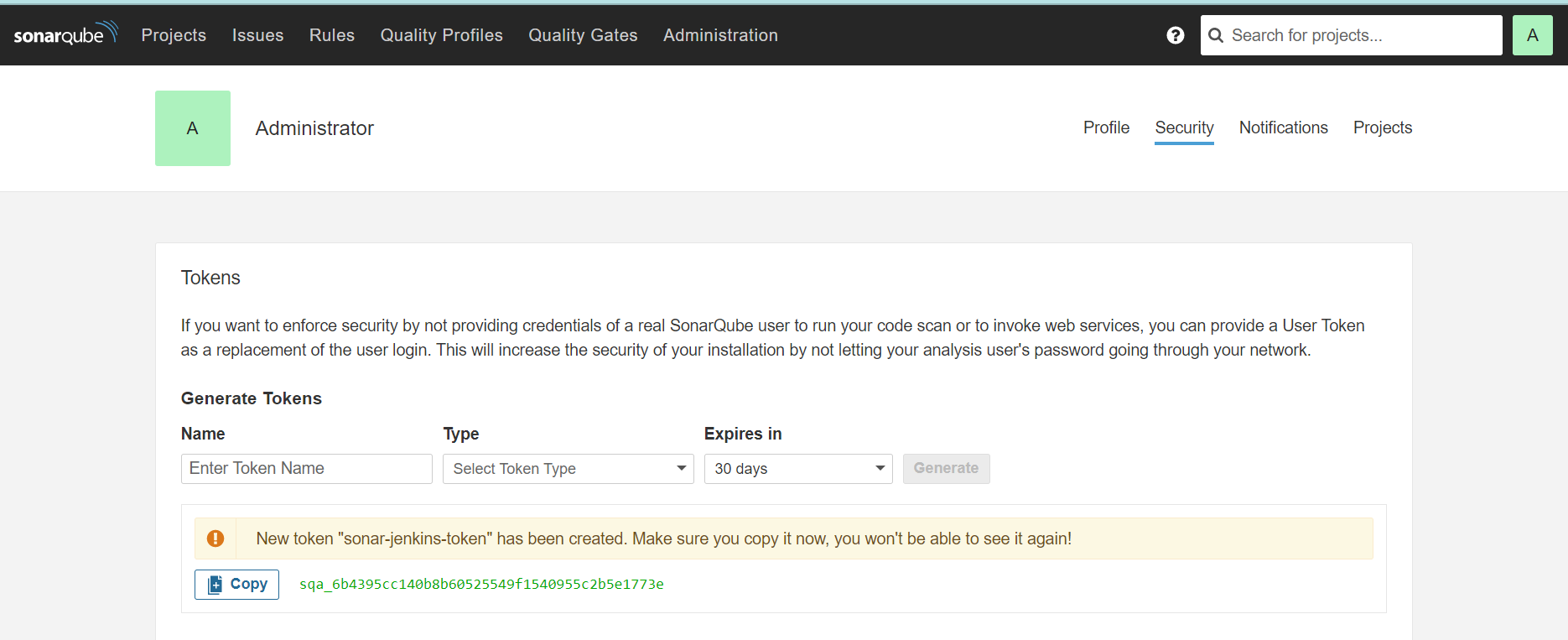
Access the Sonar UI through browser using url: <http://13.126.255.253:9000/>

Default user and password is admin, admin

**Step to integrate the SonarQube with Jenkins-**

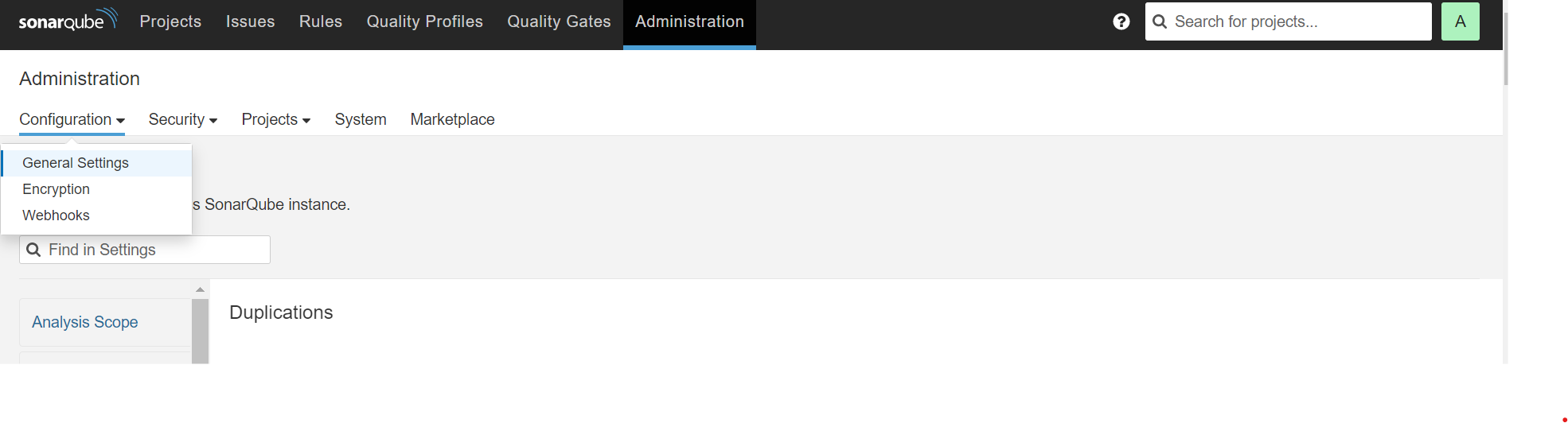
At Sonar Side:

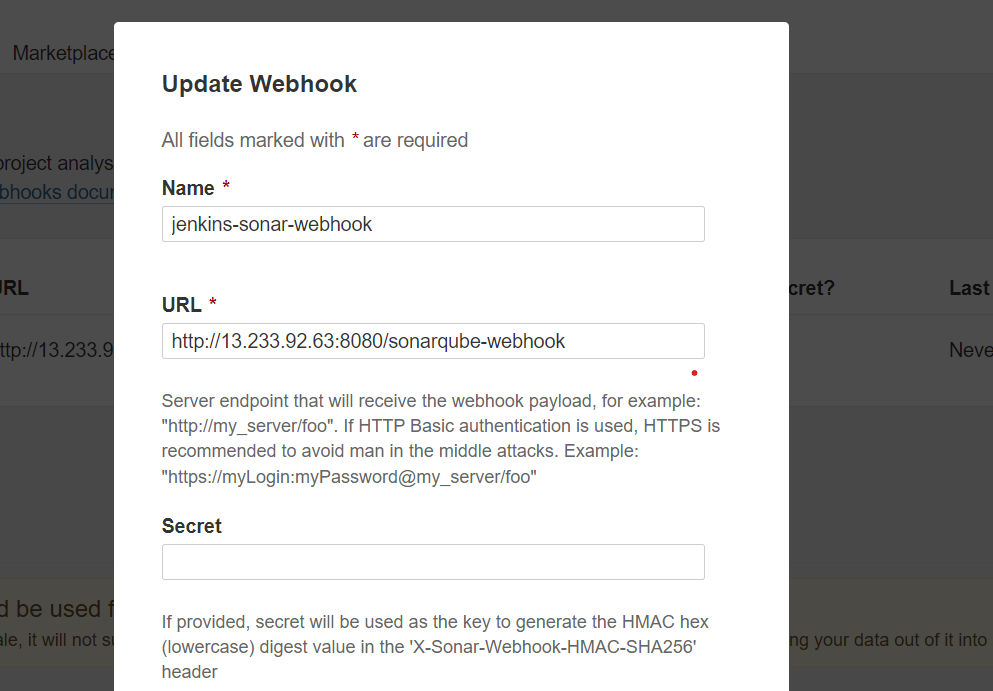
Generate the token



sqa\_6b4395cc140b8b60525549f1540955c2b5e1773e

Generate the webhook: **Administrator🡪 Configuration 🡪 Webhooks**

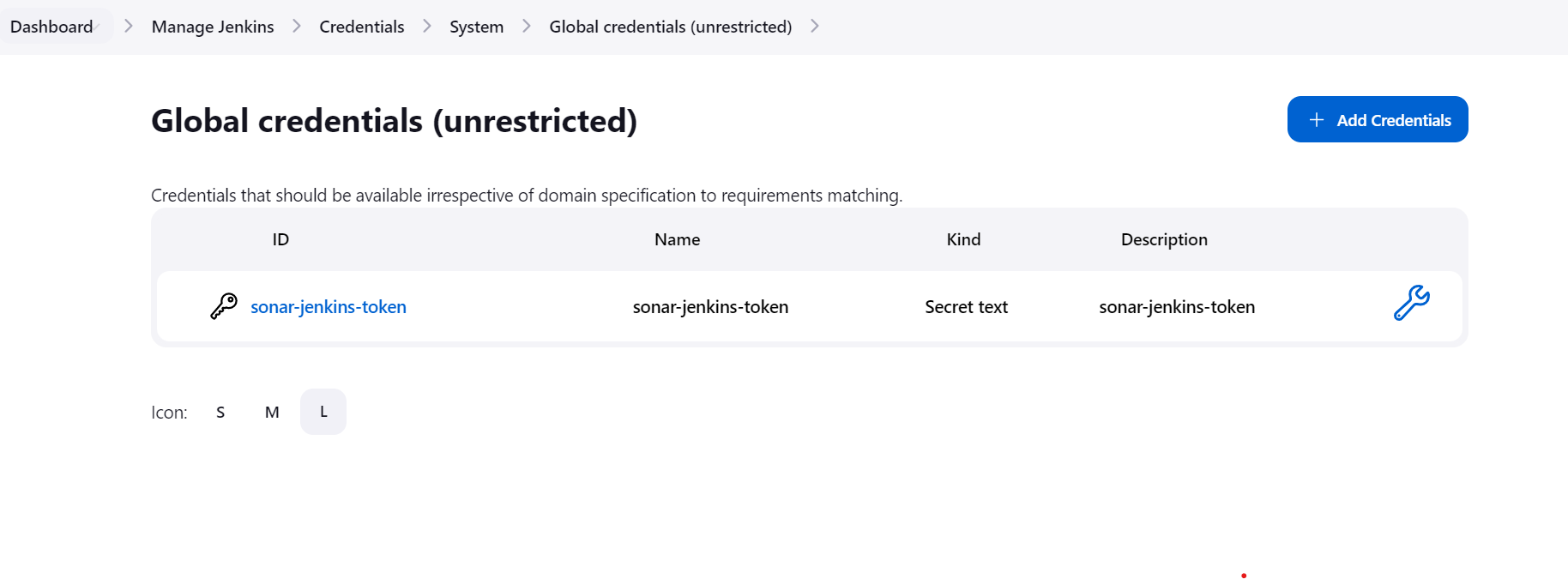




At Jenkins Side:

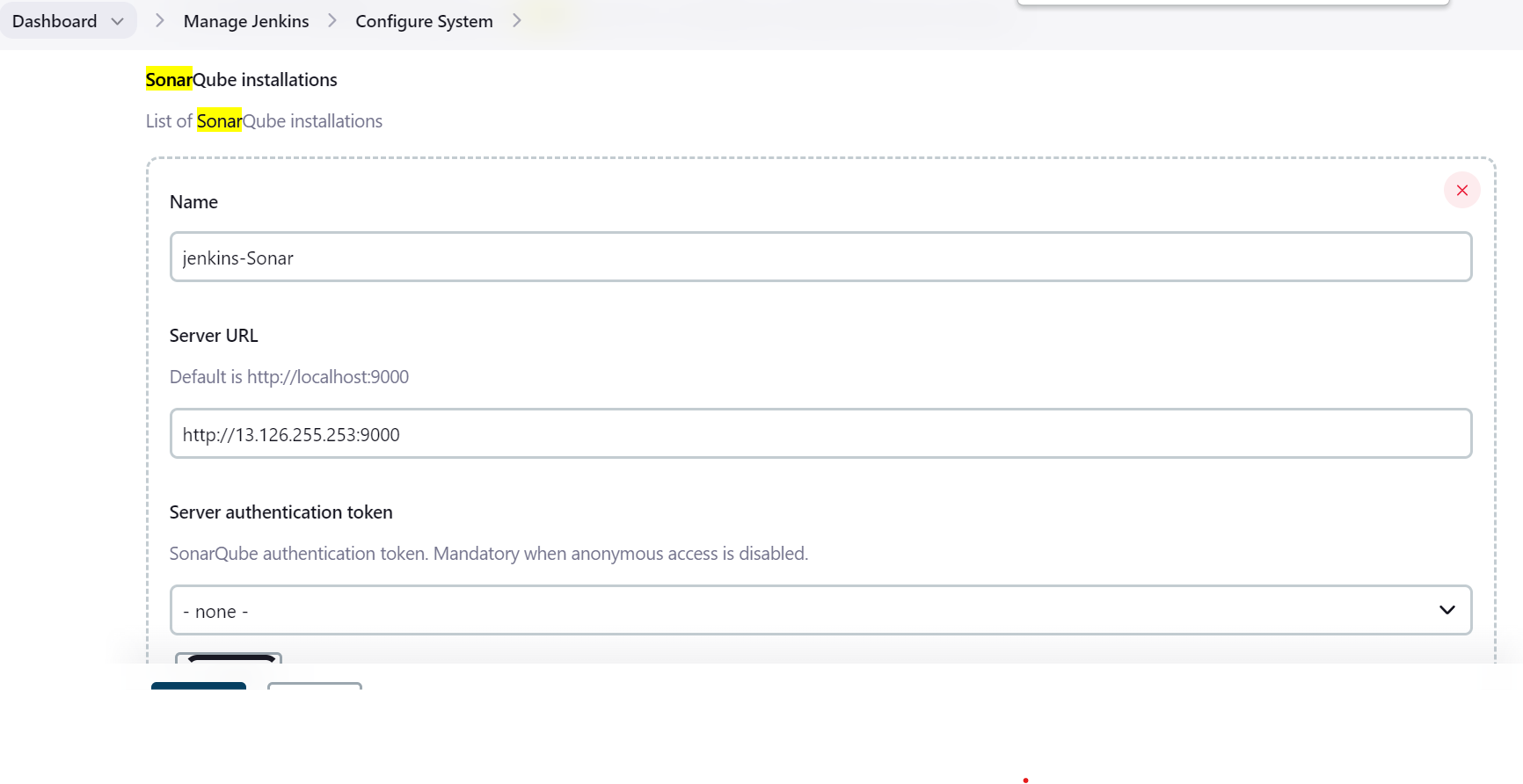
First add the SonarQube plugin

Add the SonarQube token in Credential setion-



Now tell Jenkins where the Sonar is running under configuration system

[Dashboard](http://13.233.92.63:8080/)🡪 [Manage Jenkins](http://13.233.92.63:8080/manage/)🡪 Configure System



Sonar scanner will not be available by default we need to add sonar scanner plugins under manage plugins

Just Search in google- sonar Jenkins pipeline

<https://www.jenkins.io/doc/pipeline/steps/sonar/>

pipeline {

agent none

**stages {**

**stage("build & SonarQube analysis") {**

**agent any**

**steps {**

**withSonarQubeEnv('My SonarQube Server') {**

**sh 'mvn clean package sonar:sonar'**

**}**

**}**

**}**

**stage("Quality Gate") {**

**steps {**

**timeout(time: 1, unit: 'HOURS') {**

**waitForQualityGate abortPipeline: true**

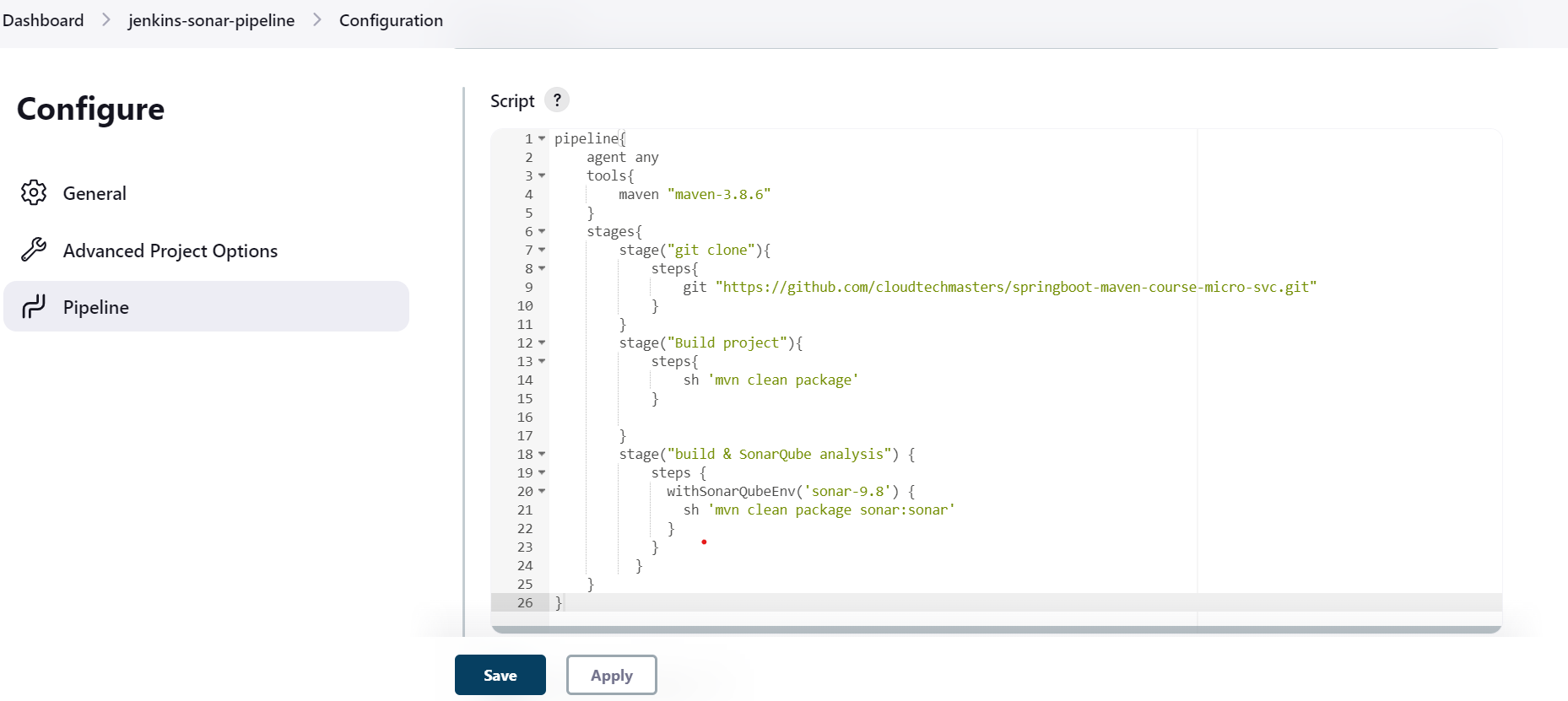
}

}

}

}

}

****

**pipeline{**

**agent any**

**tools{**

**maven "maven-3.8.6"**

**}**

**stages{**

**stage("git clone"){**

**steps{**

**git "https://github.com/cloudtechmasters/springboot-maven-course-micro-svc.git"**

**}**

**}**

**stage("Build project"){**

**steps{**

**sh 'mvn clean package'**

**}**

**}**

**stage("build & SonarQube analysis") {**

**steps {**

**withSonarQubeEnv('sonar-9.8') {**

**sh 'mvn clean package sonar:sonar'**

**}**

**}**

**}**

**}**

**}**

**IMP:** For all project we can use the generic step for sonar scan using sonar scanner,

We need to install the sonar scanner on Jenkins machine that means sonar scanner always run on Jenkins machine.

We need to mention the sonar url and token in[**project.properties**](https://github.com/cloudtechmasters/springboot-maven-course-micro-svc/blob/master/sonar-project.properties) for every project.

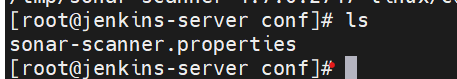
Google- sonar maven scanner

On Jenkins Machine-

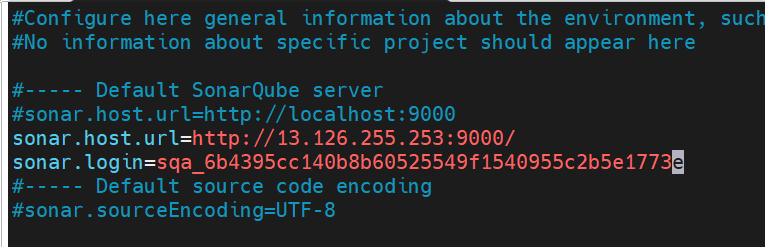
wget <https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.7.0.2747-linux.zip>

once sonar scanner downloaded we need to inform sonar scanner where our sonar running and what is the token for sonar-

/tmp/sonar-scanner-4.7.0.2747-linux/conf



Make the entry into sonar-scanner.properties file add the sonar url and token in properties file



**Now we can change our pipeline, we need to run the scanner for sonar scan**

**Add the sonar-scanner full path in sh command**

**stage("analysis using scanner") {**

**steps {**

**sh '/tmp/sonar-scanner-4.7.0.2747-linux/bin/sonar-scanner'**

**}**

**}**

**Note: sonar scanner always see the sonar-scanner.properties file under root folder of every project**

****

**pipeline{**

**agent any**

**tools{**

**maven "maven-3.8.6"**

**}**

**stages{**

**stage("git clone"){**

**steps{**

**git "https://github.com/cloudtechmasters/springboot-maven-course-micro-svc.git"**

**}**

**}**

**stage("Build project"){**

**steps{**

**sh 'mvn clean package'**

**}**

**}**

**stage("sonar-9.8") {**

**steps {**

**sh '/tmp/sonar-scanner-4.7.0.2747-linux/bin/sonar-scanner'**

**}**

**}**

**}**

**}**

**Now add the one more stage called quality gate in which we can mention the time, means if the sonar scan takes time so in that case pipeline will be stop after mentioned time.**

****

**pipeline{**

**agent any**

**tools{**

**maven "maven-3.8.6"**

**}**

**stages{**

**stage("git clone"){**

**steps{**

**git "https://github.com/cloudtechmasters/springboot-maven-course-micro-svc.git"**

**}**

**}**

**stage("Build project"){**

**steps{**

**sh 'mvn clean package'**

**}**

**}**

**stage("analysis using scanner") {**

**steps {**

**withSonarQubeEnv(sonar-9.8') {**

**sh '/tmp/sonar-scanner-4.7.0.2747-linux/bin/sonar-scanner'**

**}**

**}**

**}**

**stage("Quality Gate") {**

**steps {**

**timeout(time: 1, unit: 'HOURS') {**

**waitForQualityGate abortPipeline: true**

**}**

**}**

**}**

**}**

**}**