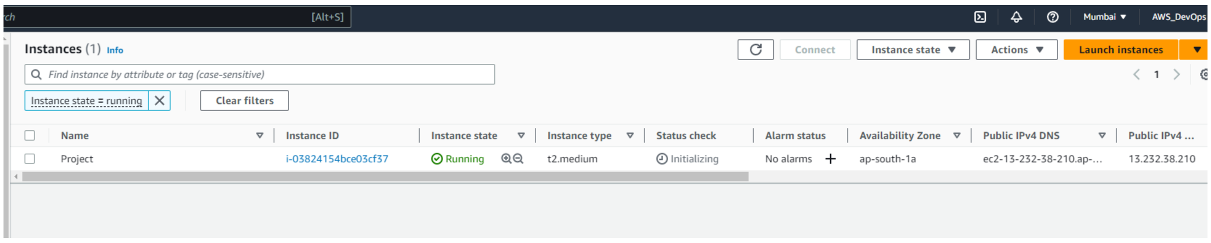
Pre requits for project:

T2. Large Ubuntu server 22.04

Install Git, Jenkins, Docker, EKS.

Step 1: launch Ubuntu server Open all security, memory 30 GB



Instances (1) Info

Find instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
Project	i-03824154bce03cf37	Running	t2.medium	Initializing	No alarms	ap-south-1a	ec2-13-232-38-210.ap-...	13.232.38.210

Attach EC2 admin role to the server.

Step 2: Installation of GIT, Docker, Jenkins

sudo -i

sudo apt-get update -y

sudo apt install git -y

git –version

sudo apt install docker.io -y

docker info

Install Jenkins: https://pkg.jenkins.io/debian-stable/

Java -version

ps -ef | grep jenkins

systemctl status Jenkins

Add Jenkins as admin user:

#usermod -aG sudo jenkins

Step 3: Git Repo setup

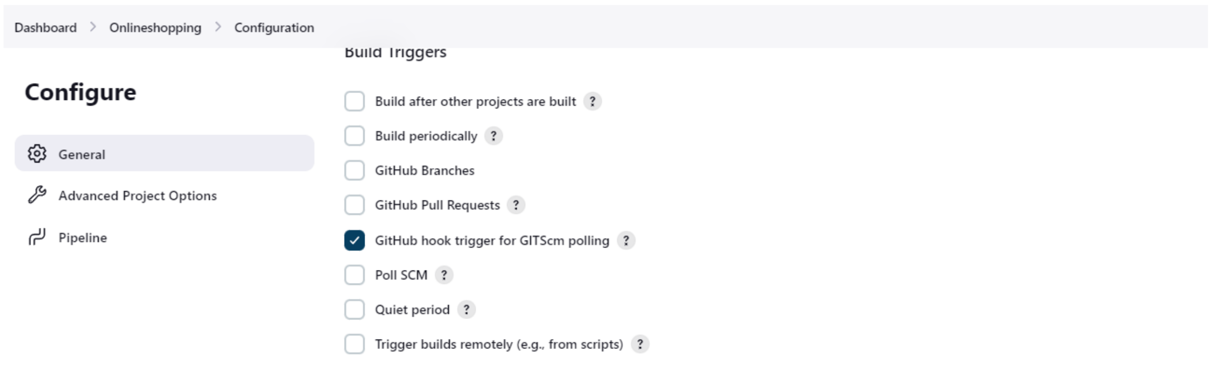
Get the URL https://github.com/Venkateshd279/my-project.git

Step 3.1: Plugins installation for Jenkins #note down default added plugins

Blue Ocean,

Configure webhook:

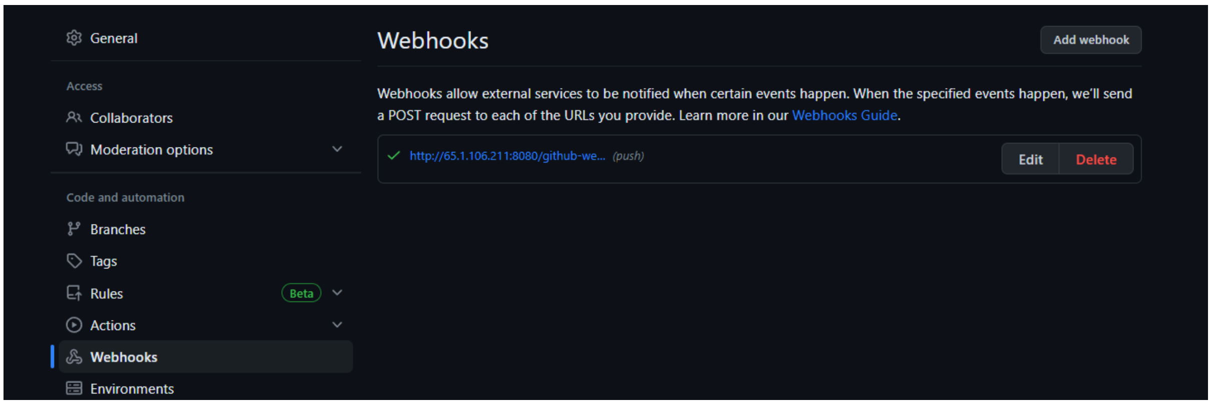
In pipeline job enable below



Build Triggers

- [ ] Build after other projects are built
- [ ] Build periodically
- [ ] GitHub Branches
- [ ] GitHub Pull Requests
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM
- [ ] Quiet period
- [ ] Trigger builds remotely (e.g., from scripts)

In Github repo go to settings -> webhooks -> add below



Webhooks

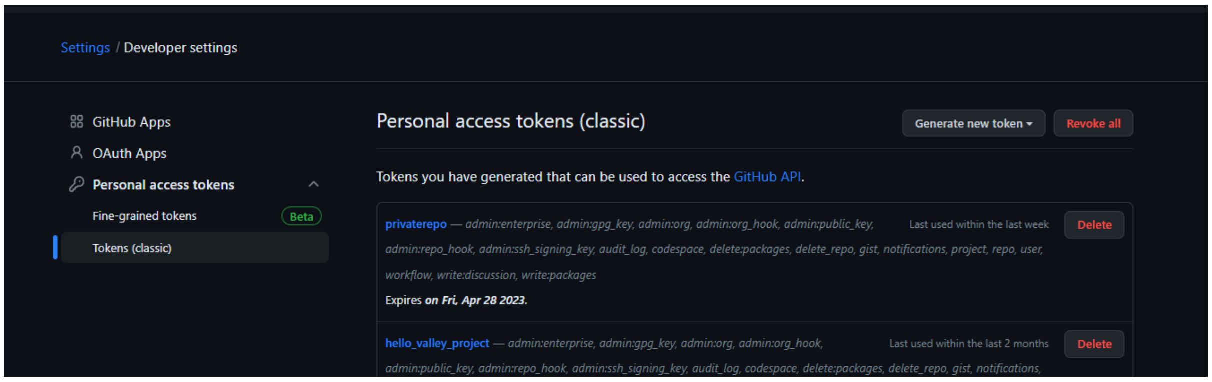
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ http://65.1.106.211:8080/github-we... (push)

Step 4: Setup Jenkins pipeline for project

Stage 1 SCM checkout

Go to syntax



Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

privaterepo — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, delete:packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:packages — Last used within the last week

Expires on Fri, Apr 28 2023.

hello_valley_project — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, delete:packages, delete_repo, gist, notifications, — Last used within the last 2 months

Generate token for private repository.

Settings -> Developer setting -> Personal access token -> Token (classic) -> Generate new token

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

pipeline syntax -> checkout git

Provide URL -> Username, Authentication password -> mention branch -> generate

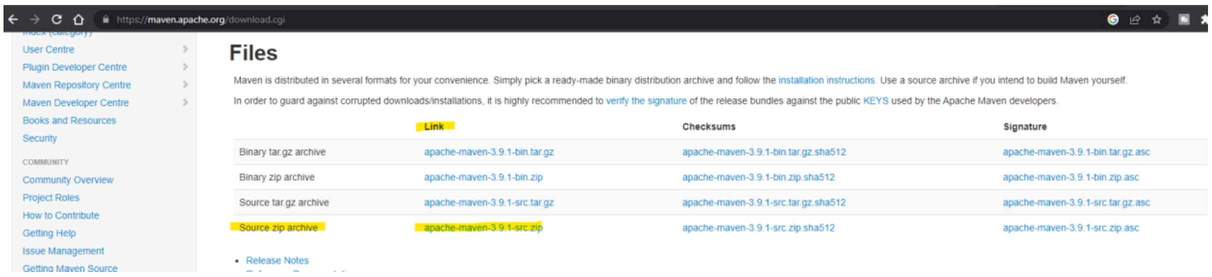
checkout scmGit(branches: [[name: '*/master']], extensions: [], userRemoteConfigs: [[credentialsId: 'Gitrepo_pass', url: 'https://github.com/Venkateshd279/my-project.git']])

```
stage('SCM_Checkout') {
        steps {
            checkout scmGit(branches: [[name: '*/master']], extensions: [], userRemoteConfigs:
[[credentialsId: 'Gitrepo_pass', url: 'https://github.com/Venkateshd279/my-project.git']])
        }
    }
```

Maven installation:

Download Maven: https://maven.apache.org/download.cgi

Download zip format



https://maven.apache.org/download.cgi

Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the installation instructions. Use a source archive if you intend to build Maven yourself.

In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public KEYS used by the Apache Maven developers.

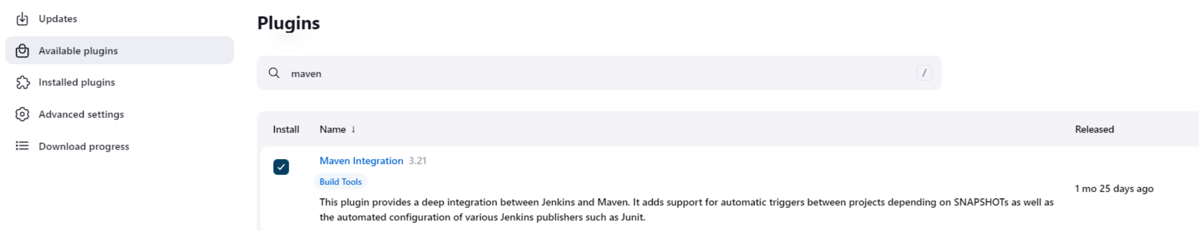
	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.9.1-bin.tar.gz	apache-maven-3.9.1-bin.tar.gz.sha512	apache-maven-3.9.1-bin.tar.gz.asc
Binary zip archive	apache-maven-3.9.1-bin.zip	apache-maven-3.9.1-bin.zip.sha512	apache-maven-3.9.1-bin.zip.asc
Source tar.gz archive	apache-maven-3.9.1-src.tar.gz	apache-maven-3.9.1-src.tar.gz.sha512	apache-maven-3.9.1-src.tar.gz.asc
Source zip archive	apache-maven-3.9.1-src.zip	apache-maven-3.9.1-src.zip.sha512	apache-maven-3.9.1-src.zip.asc

- Release Notes

Unzip maven

Move to to opt directory and name as maven

Install maven integration plugin #version 3.21 #restart Jenkins



Plugins

maven

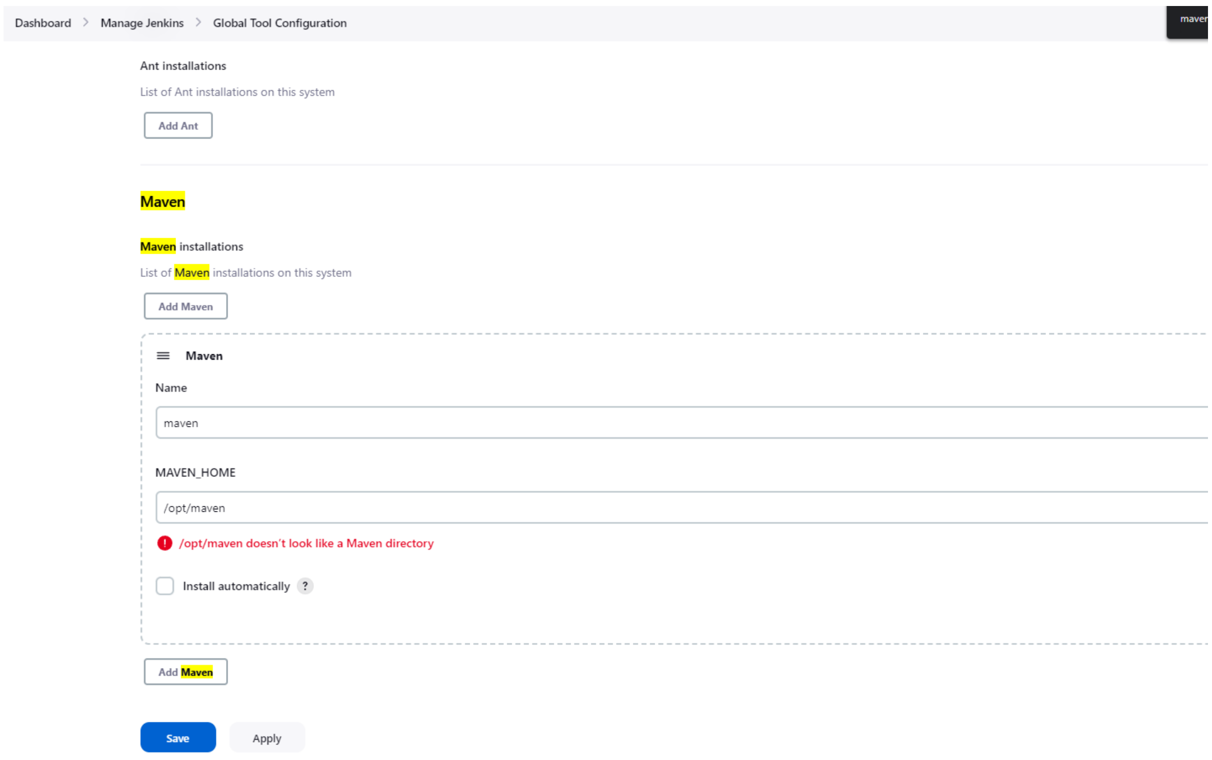
Install	Name ↓	Released
✓	Maven Integration 3.21 Build Tools This plugin provides a deep integration between Jenkins and Maven. It adds support for automatic triggers between projects depending on SNAPSHOTs as well as the automated configuration of various Jenkins publishers such as Junit.	1 mo 25 days ago

To configure maven home directory in Jenkins

Manage Jenkins -> add maven

Name: “maven”

Maven Home: “/opt/maven”


Dashboard > Manage Jenkins > Global Tool Configuration

Ant installations

List of Ant installations on this system

Add Ant

Maven

Maven installations

List of Maven installations on this system

Add Maven

Maven

Name

maven

MAVEN_HOME

/opt/maven

/opt/maven doesn't look like a Maven directory

Install automatically

Add Maven

Save Apply

Need to install maven: #sudo apt install maven -y

#mvn --version

Go to maven home directory and execute script manually.

#cd /opt/maven

#mvn clean verify sonar:sonar   **#check goal gets success or not as a root user**

Stage 2 SonarQube analyse

2.1 Install SonarQube

adduser sonarqube   #pass: sonar@123

wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip

sudo apt install unzip

chmod -R 755 /home/sonarqube/sonarqube-9.4.0.54424

chown -R sonarqube:sonarqube /home/sonarqube/sonarqube-9.4.0.54424

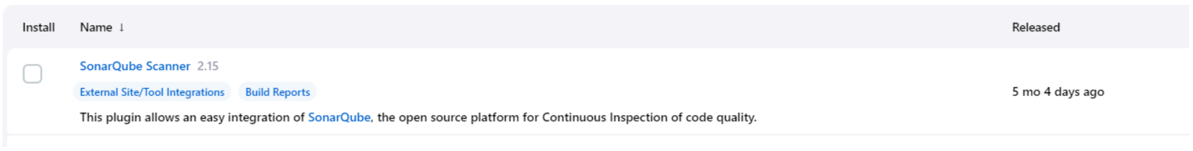
cd sonarqube-9.4.0.54424/bin/linux-x86-64/

should not run as root user:

./sonar.sh start

- Change sonarqube default password
- Add sonarscanner plugin # sonarqube scanner 2.15

- Restart Jenkins



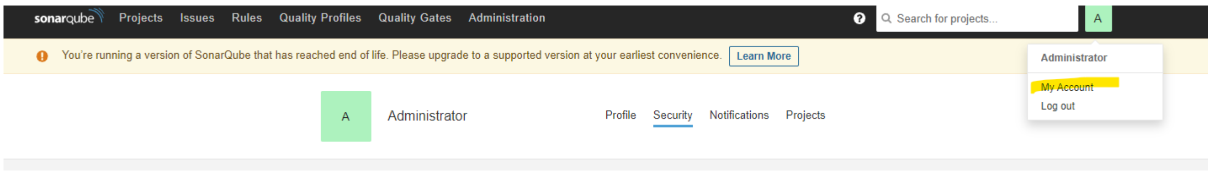
Install	Name ↓	Released
☐	SonarQube Scanner 2.15 External Site/Tool Integrations · Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	5 mo 4 days ago

-

2.2 SonarQube with Jenkins Integration

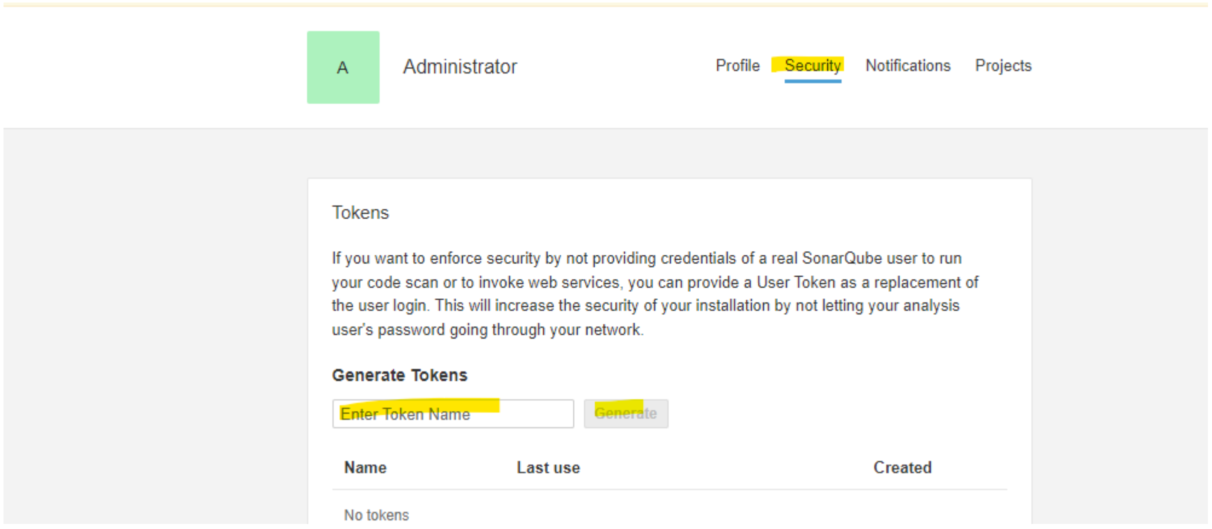
Need to generate authentication token from SonarQube.

Admin -> my account



You're running a version of SonarQube that has reached end of life. Please upgrade to a supported version at your earliest convenience. Learn More

Security -> generate



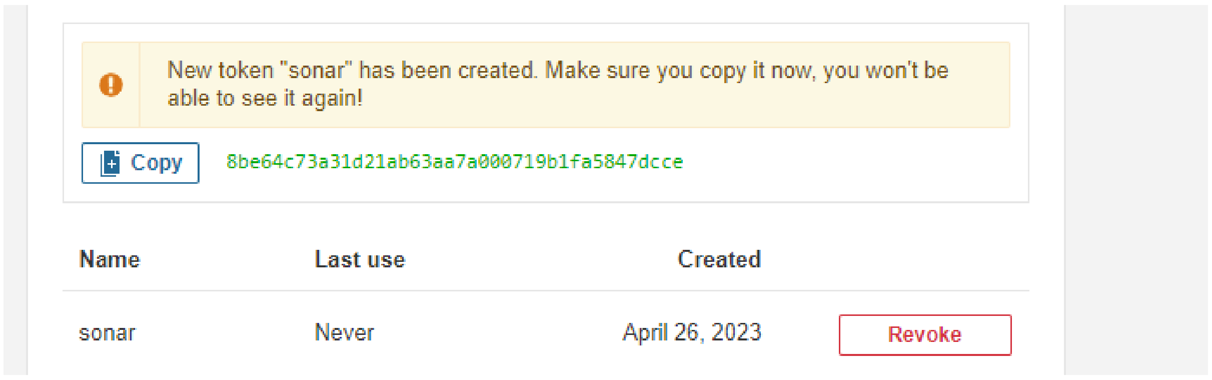
Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

Enter Token Name | Generate

Name	Last use	Created
No tokens		



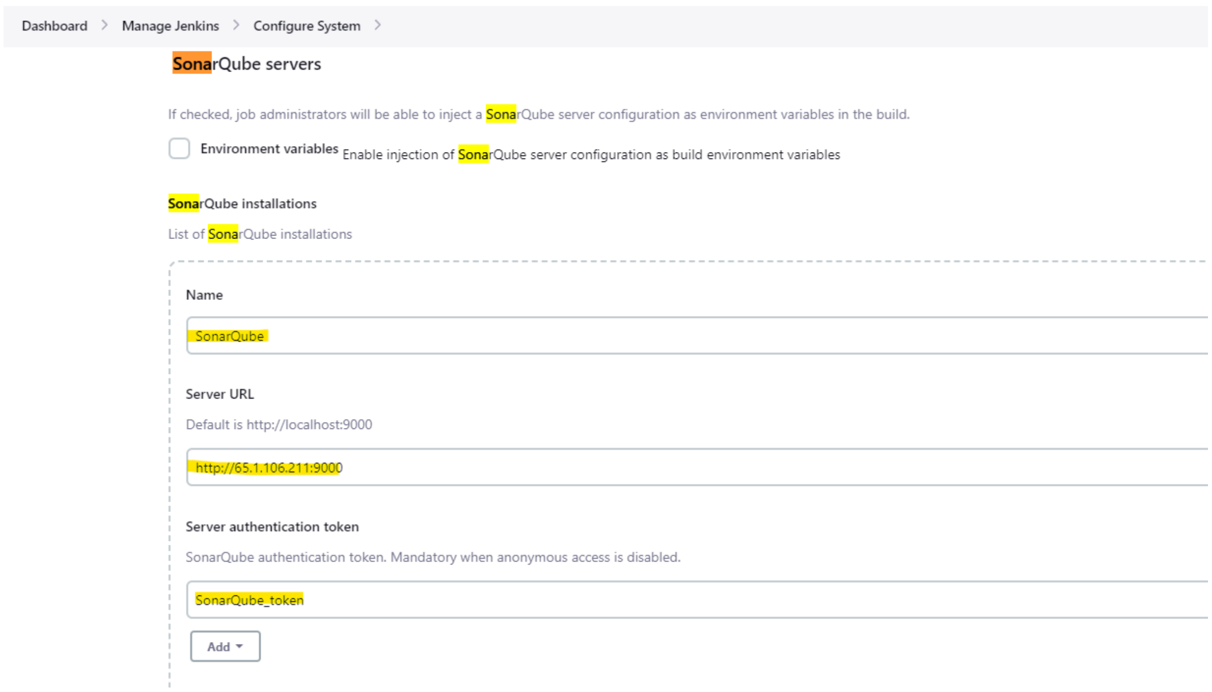
New token "sonar" has been created. Make sure you copy it now, you won't be able to see it again!

Copy `8be64c73a31d21ab63aa7a000719b1fa5847dcce`

Name	Last use	Created	
sonar	Never	April 26, 2023	Revoke

Add this token in Jenkins as secret text.

Go to Manage Jenkins -> Configure system -> add sonarqube


Dashboard > Manage Jenkins > Configure System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name

SonarQube

Server URL

Default is http://localhost:9000

http://65.1.106.211:9000

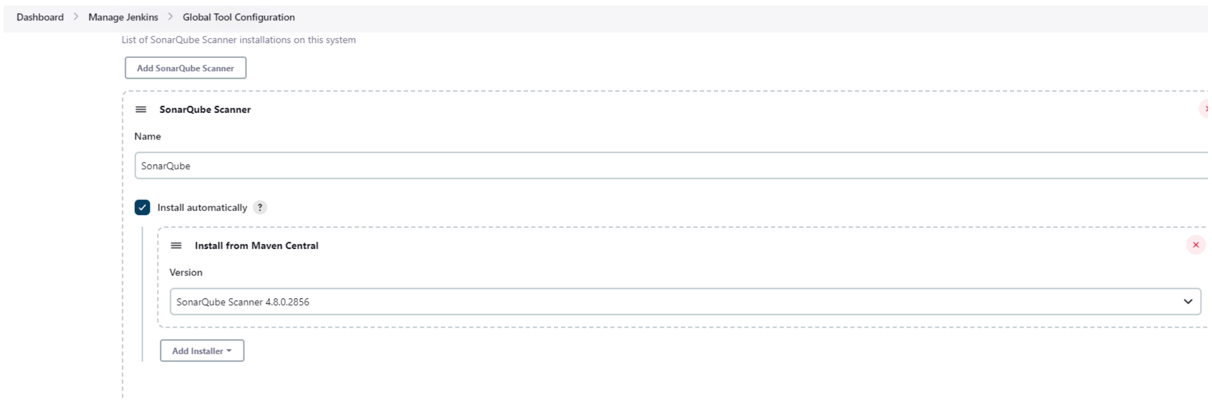
Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

SonarQube_token

Add

Add sonarscanner in Global Tool configuration


Dashboard > Manage Jenkins > Global Tool Configuration

List of SonarQube Scanner installations on this system

Add SonarQube Scanner

SonarQube Scanner

Name

SonarQube

☑ Install automatically

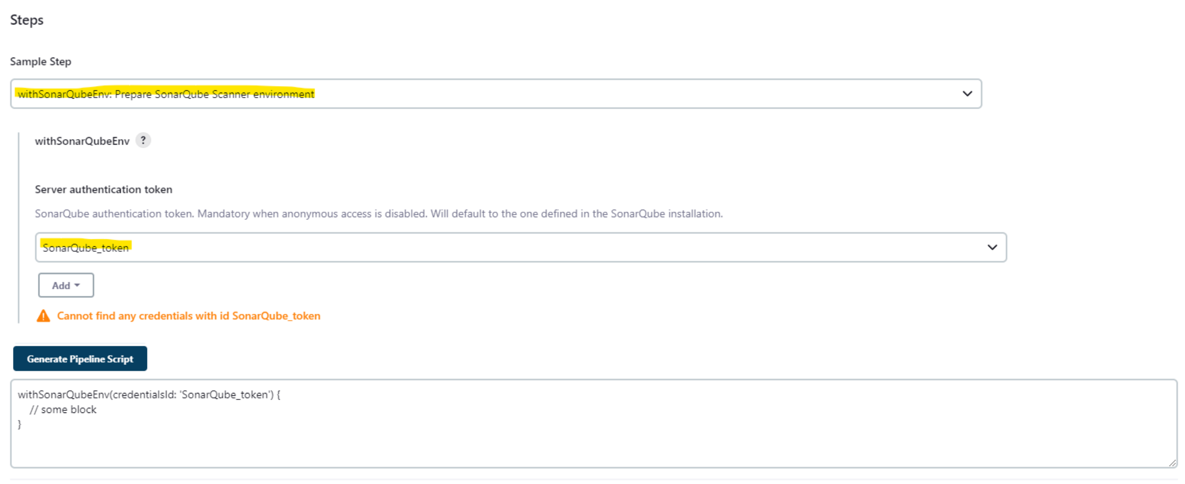
Install from Maven Central

Version

SonarQube Scanner 4.8.0.2856

Add Installer

To generate script:


Steps

Sample Step

withSonarQubeEnv: Prepare SonarQube Scanner environment

withSonarQubeEnv

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled. Will default to the one defined in the SonarQube installation.

SonarQube_token

Add

⚠ Cannot find any credentials with id SonarQube_token

Generate Pipeline Script

```
withSonarQubeEnv(credentialsId: 'SonarQube_token') {
    // some block
}
```

In authentication token give that secret password. And generate script.

Manually run as root user at /var/lib/Jenkins/workspace/<yourjob>

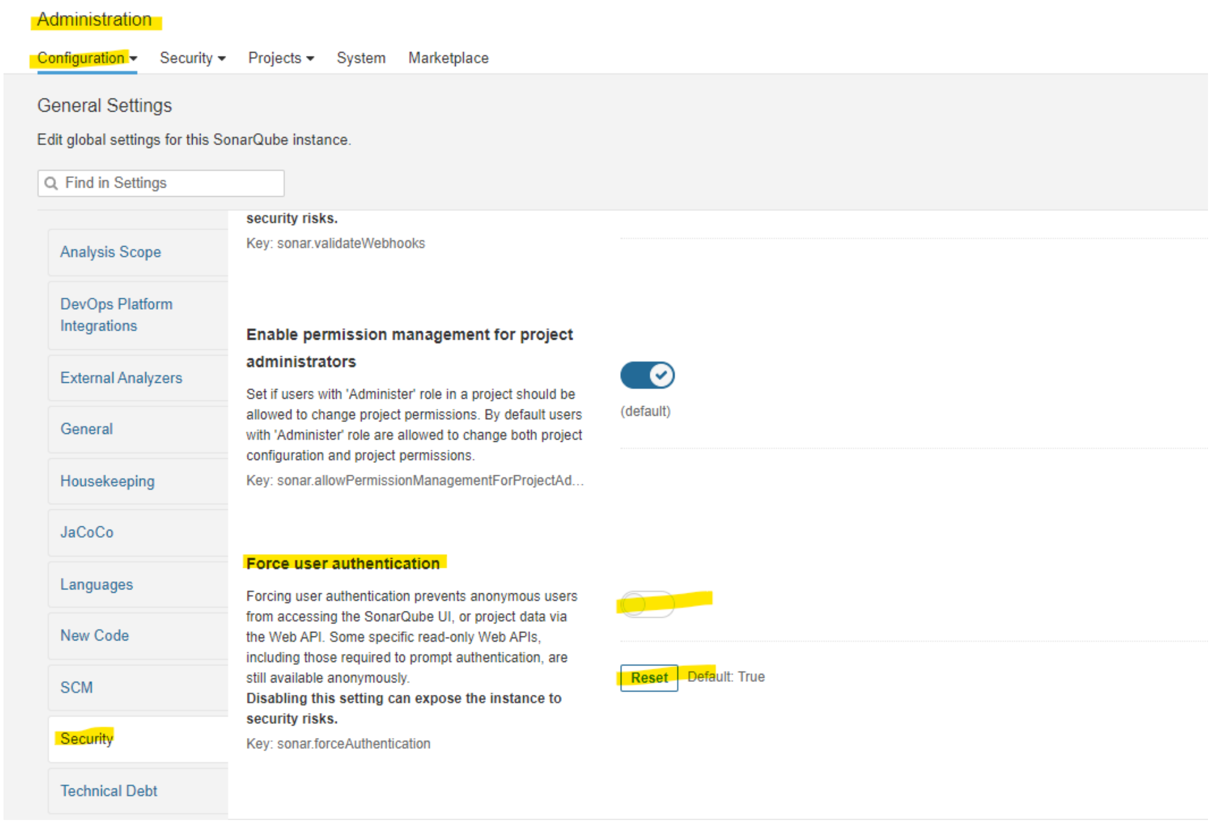
#mvn clean verify sonar:sonar

#mvn clean install

Need to give permission for Jenkins at target. Change ownership for Jenkins.

#chown -R Jenkins:Jenkins target

Disable authentication option in SonarQube



General Settings

Edit global settings for this SonarQube instance.

security risks.

Key: sonar.validateWebhooks

Enable permission management for project administrators

Set if users with 'Administer' role in a project should be allowed to change project permissions. By default users with 'Administer' role are allowed to change both project configuration and project permissions.

Key: sonar.allowPermissionManagementForProjectAd...

(default)

Force user authentication

Forcing user authentication prevents anonymous users from accessing the SonarQube UI, or project data via the Web API. Some specific read-only Web APIs, including those required to prompt authentication, are still available anonymously.
Disabling this setting can expose the instance to security risks.

Key: sonar.forceAuthentication

Default: True

```
stage('SonarQube Analysis') {
        steps {
          script {
         withSonarQubeEnv(credentialsId: 'SONAR_NEW_TOKEN') {
          sh 'mvn clean verify sonar:sonar'
         }
        }
        }
    }
```

Stage 4: Quality gate for SonarQube

```
    stage("Quality Gate") {
      steps {
        sleep(60)
       timeout(time: 1, unit: 'HOURS') {
        waitForQualityGate abortPipeline: true, credentialsId: 'SonarQube_token'
       }
      }
      post {

    failure {
      echo 'sending email notification from jenkins'

          step([$class: 'Mailer',
          notifyEveryUnstableBuild: true,
          recipients: emailextrecipients([[$class: 'CulpritsRecipientProvider'],
                      [$class: 'RequesterRecipientProvider']])])



        }
      }
     }
```

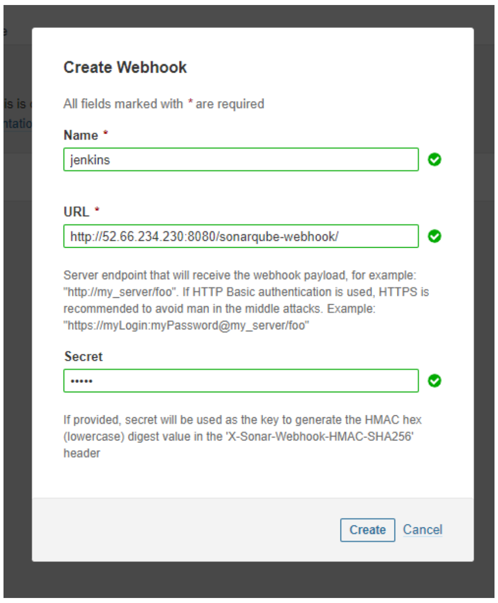
Configure webhook at SonarQube.

Administration -> configuration -> webhooks -> create

Name – any name

URL: http://jenkins-server-ip:8080/sonarqube-webhook/

Secret: sonar #our wish


Create Webhook

All fields marked with * are required

Name *
jenkins

URL *
http://52.66.234.230:8080/sonarqube-webhook/

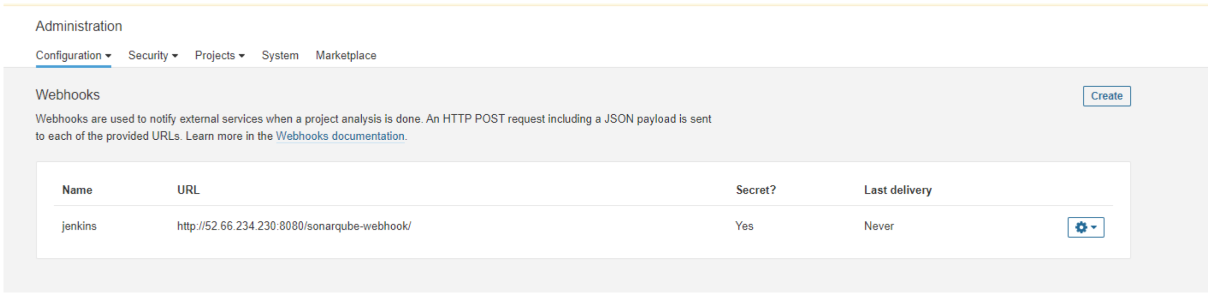
Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret
•••••

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header

Configure the secret word as secret text in Jenkins credentials.

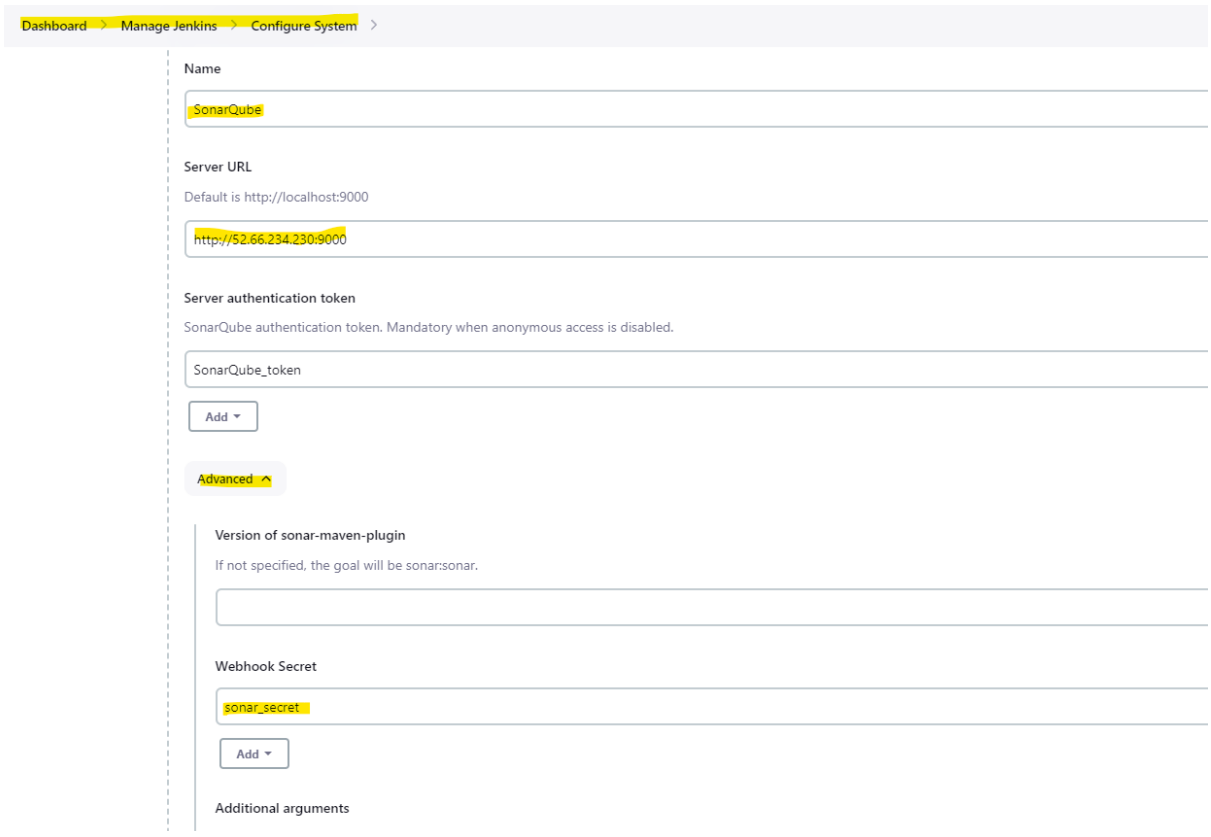
Add that secret in configure system under advanced settings



Webhooks

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the Webhooks documentation.

Name	URL	Secret?	Last delivery
jenkins	http://52.66.234.230:8080/sonarqube-webhook/	Yes	Never


Dashboard > Manage Jenkins > Configure System >

Name

SonarQube

Server URL

Default is http://localhost:9000

http://52.66.234.230:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

SonarQube_token

Add

Advanced

Version of sonar-maven-plugin

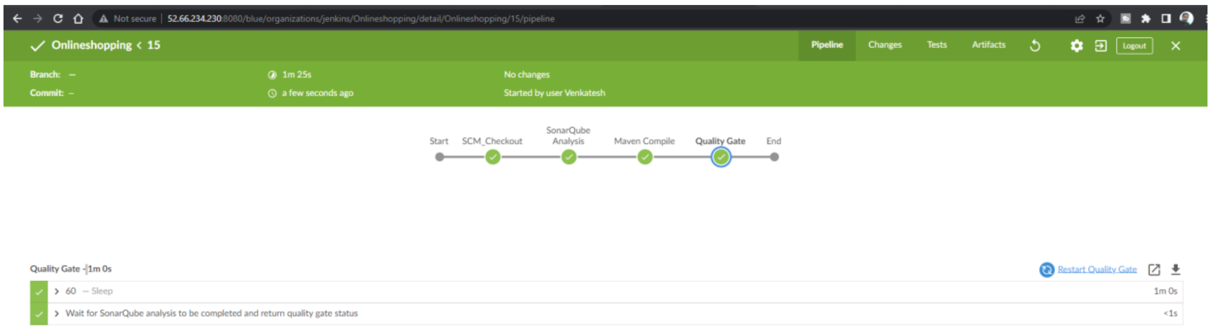
If not specified, the goal will be sonar:sonar.

Webhook Secret

sonar_secret

Add

Additional arguments


Not secure | 52.66.234.230:8080/blue/organizations/jenkins/Onlineshopping/detail/Onlineshopping/15/pipeline

✓ Onlineshopping < 15

Pipeline Changes Tests Artifacts Logout

Branch: –
Commit: –
1m 25s
a few seconds ago
No changes
Started by user Venkatesh

Start — SCM_Checkout — SonarQube Analysis — Maven Compile — Quality Gate — End

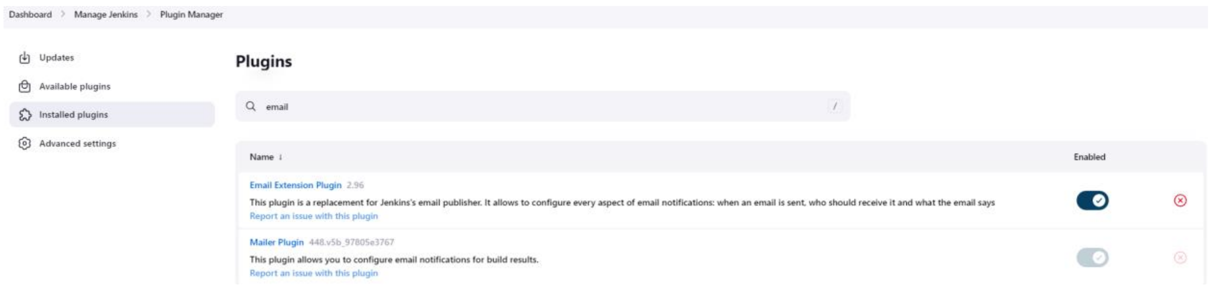
Quality Gate - 1m 0s   Restart Quality Gate

```
✓  > 60 — Sleep                                                                1m 0s
✓  > Wait for SonarQube analysis to be completed and return quality gate status  <1s
```

Stage: Email- Notification

Plugin: email extension

By default, plugin will be added here.


Dashboard > Manage Jenkins > Plugin Manager

Updates
Available plugins
Installed plugins
Advanced settings

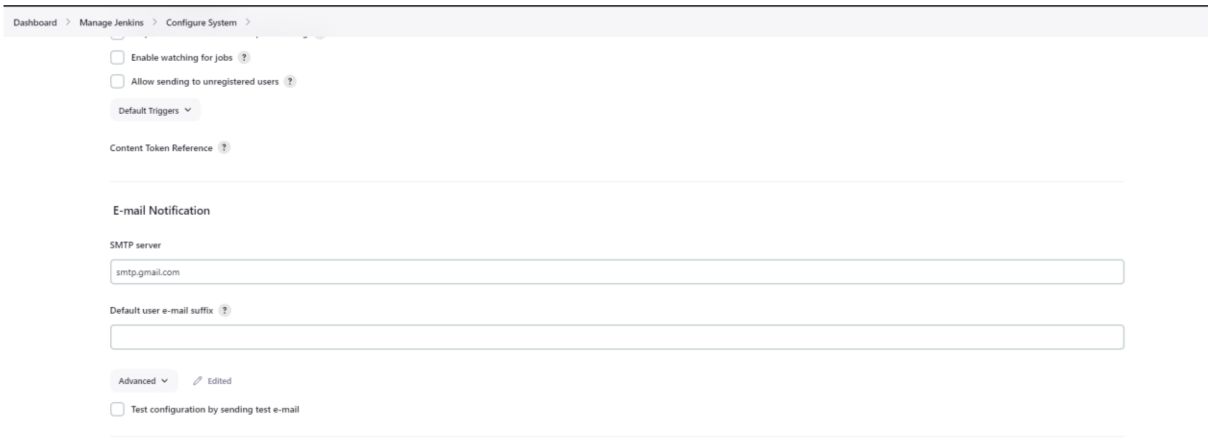
Plugins

email

Name ↓	Enabled
Email Extension Plugin 2.96 This plugin is a replacement for Jenkins's email publisher. It allows to configure every aspect of email notifications: when an email is sent, who should receive it and what the email says Report an issue with this plugin	✓
Mailer Plugin 448.v5b_97805e3767 This plugin allows you to configure email notifications for build results. Report an issue with this plugin	✓

To configure email notification

Manage Jenkins -> configure system -> email notification


Dashboard > Manage Jenkins > Configure System >

[ ] Enable watching for jobs ?

[ ] Allow sending to unregistered users ?

Default Triggers

Content Token Reference ?

E-mail Notification

SMTP server: smtp.gmail.com

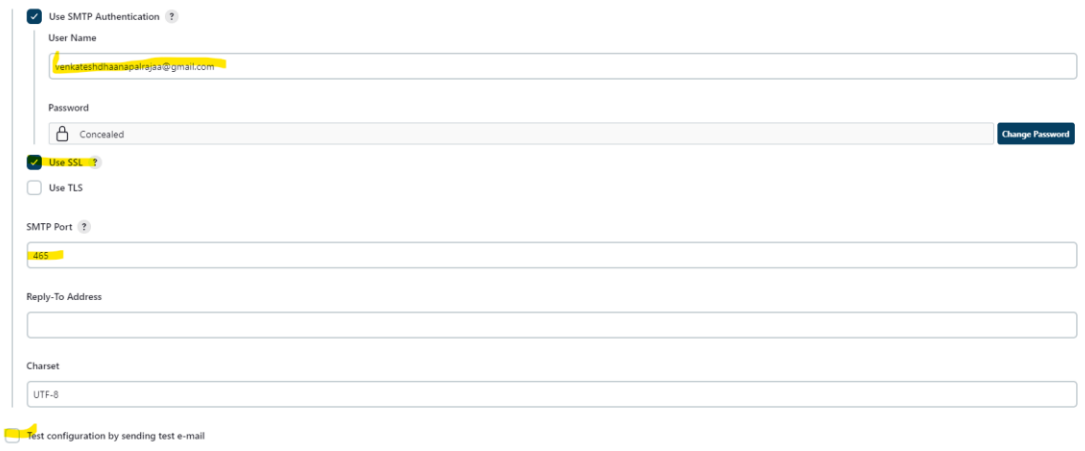
Default user e-mail suffix ?

Advanced — Edited

[ ] Test configuration by sending test e-mail

SMTP server name: smtp.gmail.com

Click advance


[✓] Use SMTP Authentication ?

User Name: venkateshdhaanapalrajaa@gmail.com

Password: Concealed — Change Password

[✓] Use SSL ?

[ ] Use TLS

SMTP Port ?: 465

Reply-To Address:

Charset: UTF-8

[ ] Test configuration by sending test e-mail

Configure in Gmail.

Manage google account -> security -> 2 step verification -> app permission -> add name -> generate

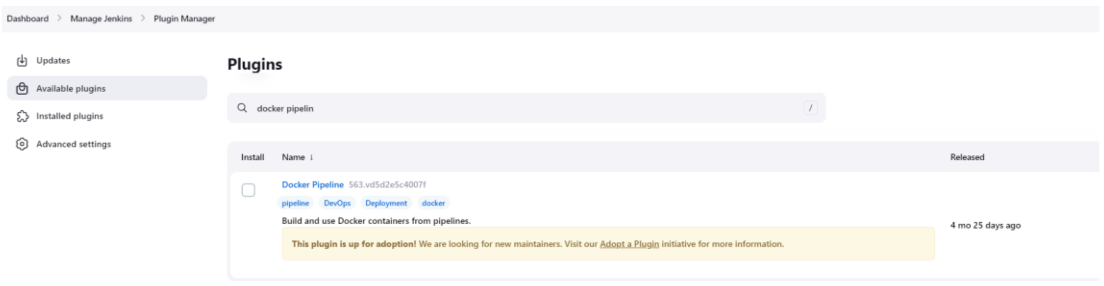
Use that generated key in password.

Test configuration by sending email.

Stage: Docker build

Plugin: docker pipeline plugin

Permission: #chmod 777 /var/run/docker.sock



Plugins

docker pipelin

Install	Name ↓	Released
☐	Docker Pipeline 563.vd5d2e5c4007f pipeline DevOps Deployment docker Build and use Docker containers from pipelines. **This plugin is up for adoption!** We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	4 mo 25 days ago

```
stage('Build Docker Image') {
          steps {
             script {
                def dockerImage = docker.build("venkateshdhanap/myweb:0.0.2", "--file Dockerfile .")
             }
          }
       }
```

Stage CD Approvals

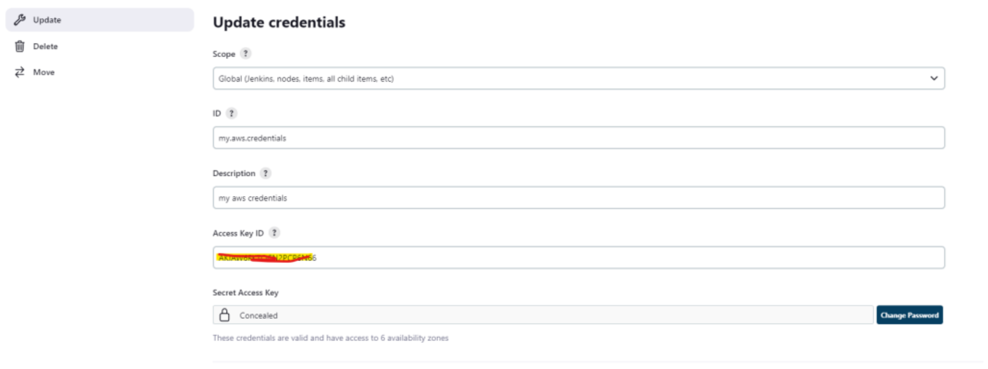
```
Stage {
  Steps {
     Input ‘Approve for CD’
}
}
```

Stage: Push docker image to ECR.

Plugin: Amazon ECR, CloudBees Docker Build and Publish

Create access key and secret key for aws account. Store in credentials.

AWS credentials .



Update credentials

Scope: Global (Jenkins, nodes, items, all child items, etc)

ID: my.aws.credentials

Description: my aws credentials

Access Key ID: [redacted]

Secret Access Key: Concealed

These credentials are valid and have access to 6 availability zones

Stage: Deploy on EKS

Install aws cli: #apt install awscli

Configure aws credentials: #aws configure

Refer the creation document.

cat /var/lib/jenkins/.kube/config

Create EKS Cluster

Pre-requisites:

Jenkins EC2 instance needs to have following configured:

Install AWS CLI:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

sudo apt install unzip

sudo unzip awscliv2.zip

sudo ./aws/install

aws –version

Install eksctl – A command line tool for working with EKS clusters that automates many individual tasks.

eksctl is a command line tool for working with EKS clusters that automates many individual tasks.

The ***eksctl*** tool uses CloudFormation under the hood, creating one stack for the EKS master control plane and another stack for the worker nodes.

Download and extract the latest release of `eksctl` with the following command.

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

Move the extracted binary to /usr/local/bin.

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

Install kubectl – A command line tool for working with Kubernetes clusters.

sudo curl --silent --location -o /usr/local/bin/kubectl   https://s3.us-west-2.amazonaws.com/amazon-eks/1.22.6/2022-03-09/bin/linux/amd64/kubectl

sudo chmod +x /usr/local/bin/kubectl

Verify if kubectl got installed

kubectl version --short --client

Create IAM Role with Administrator Access

Assign the role to EC2 instance

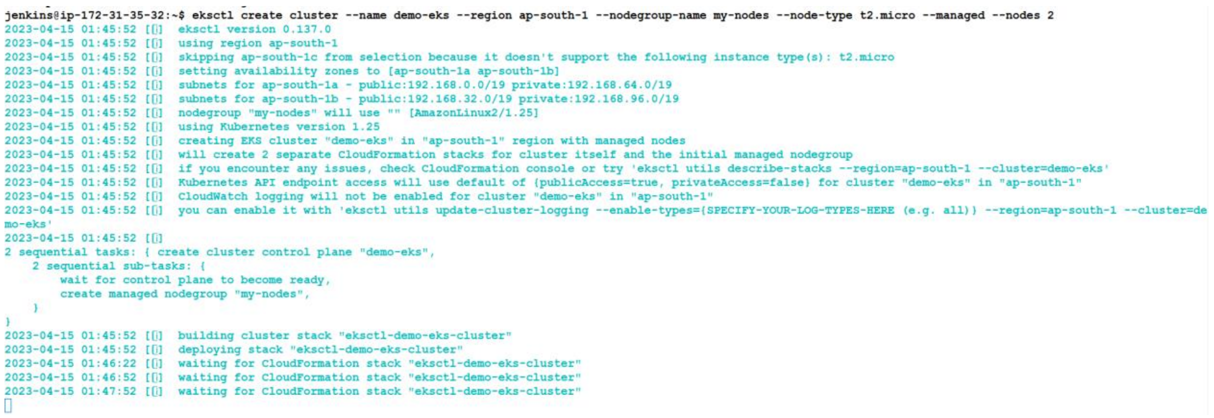
Switch to Jenkins user

sudo su - jenkins

Create EKS Cluster with two worker nodes using eksctl

eksctl create cluster --name demo-eks --region ap-south-1 --nodegroup-name my-nodes --node-type t2.micro --managed --nodes 2

the above command should create a EKS cluster in AWS, it might take 15 to 20 mins. The eksctl tool uses CloudFormation under the hood, creating one stack for the EKS master control plane and another stack for the worker nodes.



```
jenkins@ip-172-31-35-32:~$ eksctl create cluster --name demo-eks --region ap-south-1 --nodegroup-name my-nodes --node-type t2.micro --managed --nodes 2
2023-04-15 01:45:52 [ℹ]  eksctl version 0.137.0
2023-04-15 01:45:52 [ℹ]  using region ap-south-1
2023-04-15 01:45:52 [ℹ]  skipping ap-south-1c from selection because it doesn't support the following instance type(s): t2.micro
2023-04-15 01:45:52 [ℹ]  setting availability zones to [ap-south-1a ap-south-1b]
2023-04-15 01:45:52 [ℹ]  subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.64.0/19
2023-04-15 01:45:52 [ℹ]  subnets for ap-south-1b - public:192.168.32.0/19 private:192.168.96.0/19
2023-04-15 01:45:52 [ℹ]  nodegroup "my-nodes" will use "" [AmazonLinux2/1.25]
2023-04-15 01:45:52 [ℹ]  using Kubernetes version 1.25
2023-04-15 01:45:52 [ℹ]  creating EKS cluster "demo-eks" in "ap-south-1" region with managed nodes
2023-04-15 01:45:52 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-04-15 01:45:52 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=demo-eks'
2023-04-15 01:45:52 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "demo-eks" in "ap-south-1"
2023-04-15 01:45:52 [ℹ]  CloudWatch logging will not be enabled for cluster "demo-eks" in "ap-south-1"
2023-04-15 01:45:52 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=ap-south-1 --cluster=de
mo-eks'
2023-04-15 01:45:52 [ℹ]
2 sequential tasks: { create cluster control plane "demo-eks",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "my-nodes",
    }
}
2023-04-15 01:45:52 [ℹ]  building cluster stack "eksctl-demo-eks-cluster"
2023-04-15 01:45:52 [ℹ]  deploying stack "eksctl-demo-eks-cluster"
2023-04-15 01:46:22 [ℹ]  waiting for CloudFormation stack "eksctl-demo-eks-cluster"
2023-04-15 01:46:52 [ℹ]  waiting for CloudFormation stack "eksctl-demo-eks-cluster"
2023-04-15 01:47:52 [ℹ]  waiting for CloudFormation stack "eksctl-demo-eks-cluster"
```

eksctl get cluster --name demo-eks --region ap-south-1

This should confirm that EKS cluster is up and running.

Update Kube config by entering below command:

aws eks update-kubeconfig --name demo-eks --region ap-south-1

kubeconfig file be updated under /var/lib/jenkins/.kube folder.

you can view the kubeconfig file by entering the below command:

cat /var/lib/jenkins/.kube/config

Connect to EKS cluster using kubectl commands

To view the list of worker nodes as part of EKS cluster.

kubectl get nodes

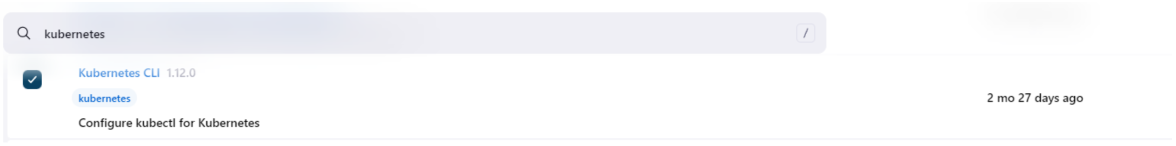
kubectl get ns

Delete EKS Cluster using eksctl

eksctl delete cluster --name demo-eks --region ap-south-1

cat /var/lib/jenkins/.kube/config - need to save as file and that file we use as secret file.

Plugin: need to install Kubernetes CLI



kubernetes

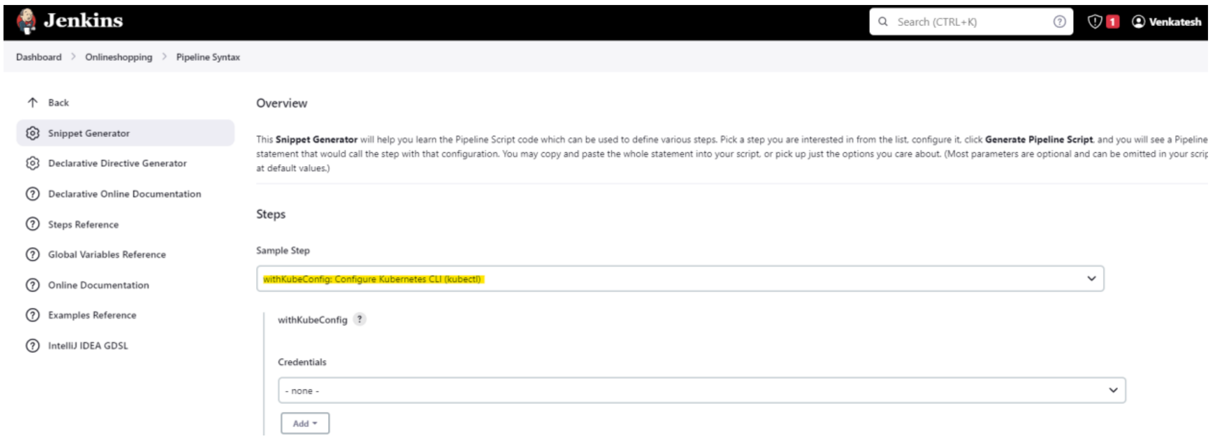
Kubernetes CLI 1.12.0

kubernetes

Configure kubectl for Kubernetes

2 mo 27 days ago

You will get below snippet generator



Jenkins
Search (CTRL+K)
Venkatesh
Dashboard > Onlineshopping > Pipeline Syntax
Back
Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL

Overview

This **Snippet Generator** will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click **Generate Pipeline Script**, and you will see a Pipeline statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your scrip at default values.)

Steps

Sample Step: withKubeConfig: Configure Kubernetes CLI (kubectl)

withKubeConfig

Credentials: - none -

Add

stage('k8s') {

steps {

echo 'Deploying on EKS'

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

```
                sh 'kubectl apply -f /var/lib/jenkins/mainservice.yaml'
            }
           }
        }



     apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app:  myapp
spec:
  containers:
  - name: nginx
    image: 477099163803.dkr.ecr.ap-south-1.amazonaws.com/jenkins_project
    ports:
    - containerPort: 8080

---


kind: Service
apiVersion: v1
metadata:
  name:  mynodeport
spec:
  selector:
    app:  myapp
  type:  NodePort
```

```
 ports:
 - name:  httpd
   port:  80
   targetPort:  8080
   protocol: TCP
```

----

example http://13.233.178.217:32715/newapp/

```
kubectl get services
```

```
kubectl get nodes -o wide
```

http://<node-ip>:<service-port>/<webapp-name>

Usually after restart need to follow below steps:

Need start sonarqube as non root user

Need to add sonarqube url in Jenkins configuration

Need to add dameon permission for docker #chmod 777 var/run/docker.sock