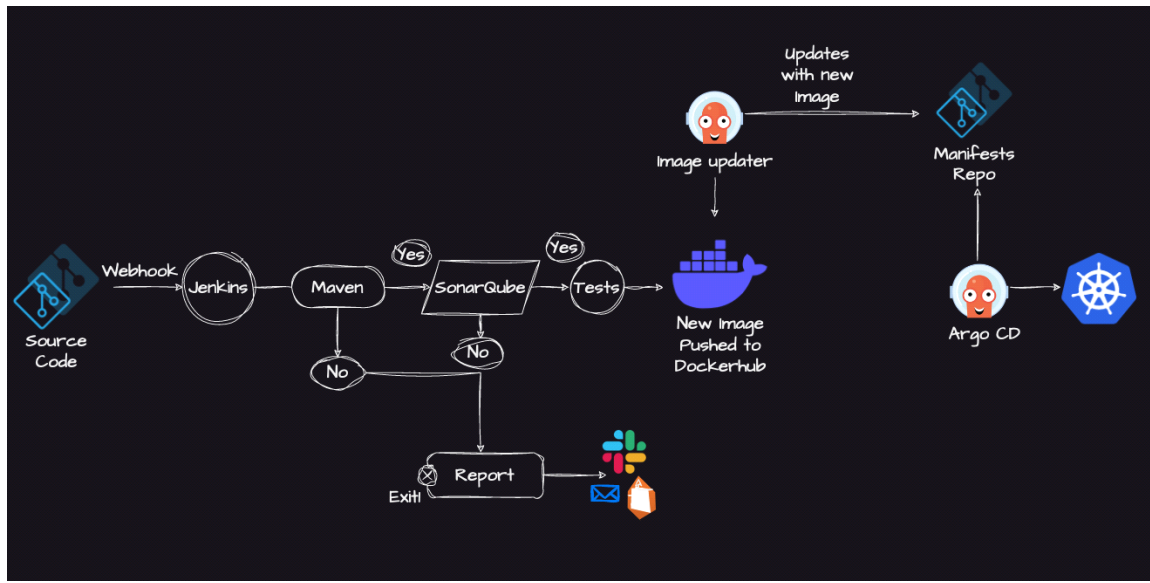


## Jenkins Pipeline for Java based application using Maven, SonarQube, Argo CD, AWS EKS, GIT AND GITHUB.



### Prerequisites:

1. Basic knowledge of Jenkins, Docker, Kubernetes, Maven, SonarQube, Git, GitHub, and ArgoCD, AWS.
2. DockerHub and GitHub accounts are required.
3. Fork my repository <https://github.com/Sanjay6372/java.git>.

code hierarchy

Files

master

Go to file

spring-boot-app-manifests

deployment.yml

service.yml

spring-boot-app

src/main

Dockerfile

JenkinsFile

README.md

pom.xml

.gitignore

README.md

java / spring-boot-app-manifests /

sanjaymavinkurve Update deployment image to version 10

Name	Last commit message
..	
deployment.yml	Update deployment image to version 10
service.yml	first commit

## 1. Create a t2 or t3 medium type of EC2 instance with Ubuntu OS for jenkins and sonarqube.

Go to your aws account---> Type EC2 in search---> Click on Launch instance---> Fill info like name , select AMI to Ubuntu, create new Key for login and select storage to 20GB. Create new security group. Others options take as default.

aws

Services

Q

Search

[Alt+S]

≡

Name

jenkins and sonar

Add additional tags

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q

Search our full catalog including 1000s of application and OS images

Recents

My AMIs

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu®

Windows

Microsoft

Red Hat

Red Hat

SUS

>  
S

Q

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Images / AMIs

## Create key pair



### Key pair name

Key pairs allow you to connect to your instance securely.

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

### Key pair type

☐

RSA

RSA encrypted private and public key pair



ED25519

ED25519 encrypted private and public key pair

### Private key file format



.pem

For use with OpenSSH



.ppk

For use with PuTTY



When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** [Learn](#)

Cancel

Create key pair

aws Services Search [Alt+S]

Instance type info | Get advice

Instance type

t3.medium  
 Family: t3 2 vCPU 4 GiB Memory Current generation: true  
 On-Demand RHEL base pricing: 0.1032 USD per Hour  
 On-Demand Linux base pricing: 0.0432 USD per Hour  
 On-Demand Windows base pricing: 0.0616 USD per Hour  
 On-Demand SUSE base pricing: 0.0995 USD per Hour

☐ All generations

[Compare instance types](#)

Additional costs apply for AMIs with pre-installed software

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

keyy

[Create new key pair](#)

▼ Network settings Info [Edit](#)

Services Search [Alt+S] Stockholm Sanjay

Network Info

vpc-043208d8b4c430725

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

We'll create a new security group called 'launch-wizard-2' with the following rules:

☒ Allow SSH traffic from  
 Helps you connect to your instance Anywhere  
 0.0.0.0/0

☒ Allow HTTPS traffic from the internet  
 To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet  
 To set up an endpoint, for example when creating a web server

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend

▼ Summary

Number of instances Info

1

Software Image (AMI)  
 Canonical, Ubuntu, 24.04 LTS, ...read more  
 ami-0705384c0b33c194c

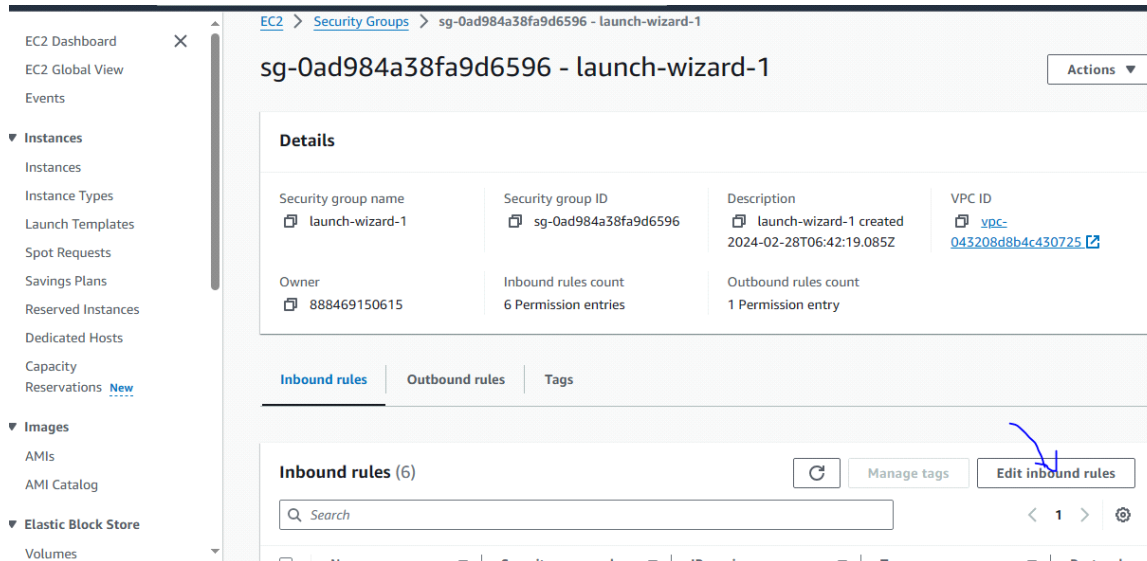
Virtual server type (instance type)  
 t3.medium

Firewall (security group)  
 New security group

Storage (volumes)  
 1 volume(s) - 8 GiB

Cancel [Launch instance](#)  
[Review commands](#)

After launch, wait for sometime then go to instance ---> go to security group and edit inbound rule.



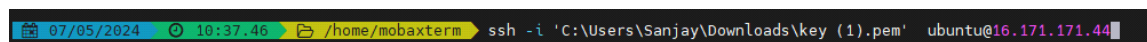
allow all tcp ports and save it.



## 2. Install Mobaxterm for window and ssh to our newly created AWS machine.

open mobaxterm and run command "ssh -i 'private key path' ubuntu@ip-of-aws-machine"

use key that we have generated and put in place of private key path and take public ip of instance from aws console.



## 3. Install Jenkins by run below commands.

- . apt update
- . apt install openjdk-11-jre
- . java -version
- . sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
- https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

```
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \  
<https://pkg.jenkins.io/debian-stable binary/> | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null  
  
. apt update  
  
. apt-get install jenkins  
  
. systemctl enable jenkins  
  
. systemctl start jenkins  
  
. systemctl status jenkins
```

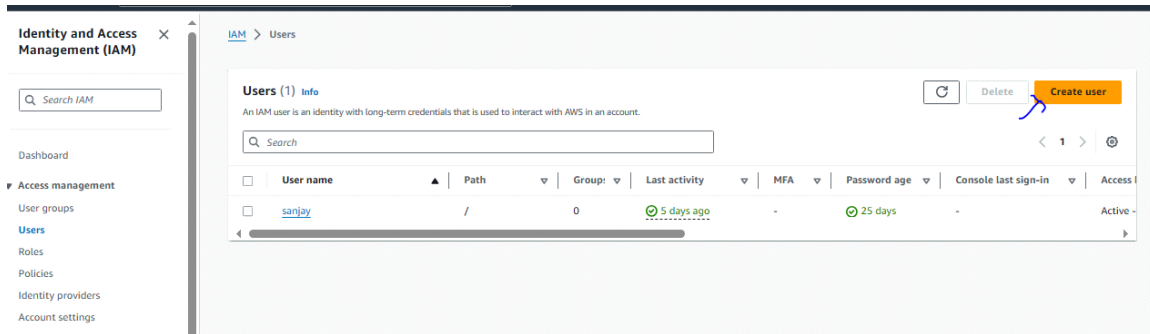
#### **4. Install sonarQube and start by run below commands.**

```
. apt install unzip wget  
  
. sudo apt install default-jdk  
  
. adduser sonarqube  
  
. Usermod -Ga sudo sonarqube  
  
. su - sonarqube  
  
. wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip  
  
. 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/  
  
. ./sonar.sh start
```

### **3. EKS setup on AWS**

**## First Create a user in AWS IAM with name EKS.**

Go to IAM ---> user ---> create user



Fill same info like below

eks

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ \_ - (hyphen)

☒ Provide user access to the AWS Management Console - *optional*  
 If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

Are you providing console access to a person?

User type

☐ Specify a user in Identity Center - Recommended  
 We recommend that you use Identity Center to provide console access to a person. With Identity Center, you can centrally manage access to their AWS accounts and cloud applications.

☒ I want to create an IAM user  
 We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.

Console password

☐ Autogenerated password  
 You can view the password after you create the user.

☒ Custom password  
 Enter a custom password for the user.
 

.....

- Must be at least 8 characters long
- Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols

☐ Show password

☐ Users must create a new password at next sign-in - Recommended  
 Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, create this IAM user. [Learn more](#)

Click on next and Select attach policy option and select below permission and save it.

AmazonEC2FullAccess

AmazonEKS\_CNI\_Policy



AmazonEKSClusterPolicy

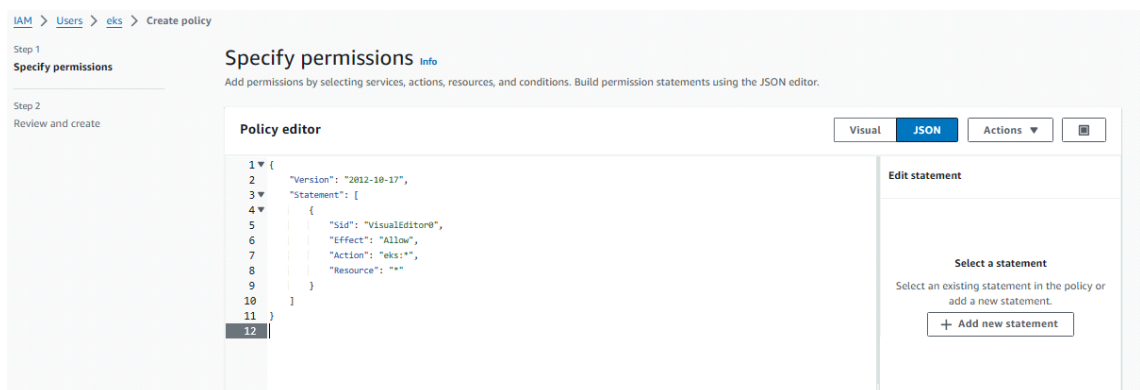
AmazonEKSWorkerNodePolicy

AWSCloudFormationFullAccess

IAMFullAccess

**Then click back to newly created user ---> add policy ---> add inline policy ---> put below code and save it**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": "eks:*",
      "Resource": "*"
    }
  ]
}
```



**# Create one more EC2 instace to control our EKS, use ubuntu AMI.**

**#Login to newly created EC2 instance by mobaxtreme then run below commands**

**#INSTALL AWS CLI**

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

. sudo apt install unzip

. unzip awscliv2.zip

. sudo ./aws/install

. aws configure      # this command will ask access key and secret key so put here your newly created
user access and secret key
```

**#INSTALL KUBECTL**

```
.curl -o kubectl
https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl

. chmod +x ./kubectl

. sudo mv ./kubectl /usr/local/bin

. kubectl version --short --client
```

**#INSTALL EKS CTL**

```
. curl --silent --location
"<https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_${uname -s}_amd64.tar.gz>"
| tar xz -C /tmp

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

. eksctl version
```

**## Create EKS CLUSTER by run below commands**

```
. eksctl create cluster --name=EKS-1 \
```

```
region --region=ap-south-1 \    # change region with your current aws selected

--zones=ap-south-1a,ap-south-1b \    # put zones of same region you have
selected above

--without-nodegroup
```

```
. eksctl utils associate-iam-oidc-provider \

--region ap-south-1 \

--cluster EKS-1 \

--approve
```

```
. eksctl create nodegroup --cluster=EKS-1 \

--region=ap-south-1 \

--name=node2 \

--node-type=t3.medium \

--nodes=3 \

--nodes-min=2 \

--nodes-max=2 \

--node-volume-size=20 \

--ssh-access \

--ssh-public-key=DevOps \    # change Devops name    with key name
that u are using for login on aws machines

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \
```

--alb-ingress-access

remove all line that are in bold before run above command

#### 4. Follow below steps to Install ArgoCD on same machine where you have ran above EKS commands.

# Install Operator Lifecycle Manager (OLM), a tool to help manage the Operators running on your cluster by following command.

```
. curl -sL  
https://github.com/operator-framework/operator-lifecycle-manager/releases/download/v0.27.0/install  
.sh | bash -s v0.27.0
```

# Install the operator by running the following command.

```
. kubectl create -f https://operatorhub.io/install/argocd-operator.yaml
```

#After install, watch your operator come up using next command.

```
. kubectl get csv -n operators
```

```
root@ip-172-31-46-226:~# kubectl get csv -n operators  
NAME              DISPLAY   VERSION   REPLACES              PHASE  
argocd-operator.v0.9.1  Argo CD  0.9.1     argocd-operator.v0.9.0 Succeeded
```

# create file with name argo.yml and put below code in it.

```
apiVersion: argoproj.io/v1alpha1
```

```
kind: ArgoCD
```

```
metadata:
```

```
  name: example-argocd
```

```
  labels:
```

```
    example: basic
```

```
spec: {}
```

# save above file and run below commad

```
. kubectl apply -f argo.yml
```

# Run below command to see your Argocd running pods

. kubectl get pods

```
root@ip-172-31-46-226:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
example-argocd-application-controller-0  1/1     Running   0           13d
example-argocd-redis-68bb584d8b-vn2bg    1/1     Running   0           13d
example-argocd-repo-server-b79657885-2qfnq 1/1     Running   0           13d
example-argocd-server-5876566c6b-q4fdj    1/1     Running   0           13d
```

# Run below command to get password for login on Argocd

. kubectl edit secret example-argocd-cluster

then copy the password , highlighted below in yellow color

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  admin.password: bllpY3pQNTZ4UlnNd3E3eTFyVGJHbGRVWDNqa0J0dVo=
kind: Secret
metadata:
  creationTimestamp: "2024-04-11T22:16:51Z"
  labels:
    app.kubernetes.io/managed-by: example-argocd
    app.kubernetes.io/name: example-argocd-cluster
    app.kubernetes.io/part-of: argocd
  name: example-argocd-cluster
  namespace: default
  ownerReferences:
```

# run

below command, change <put password here> with password you have just copied.

. echo <put password here> | base64 -d

```
root@ip-172-31-46-226:~# echo bllpY3pQNTZ4UlnNd3E3eTFyVGJHbGRVWDNqa0J0dVo= | base64 -d
```

# Run below command

. kubectl edit svc example-argocd-server # change service : clusterIp to service: nodeport

# Run below command to get ip and port no to access our ArgoCd

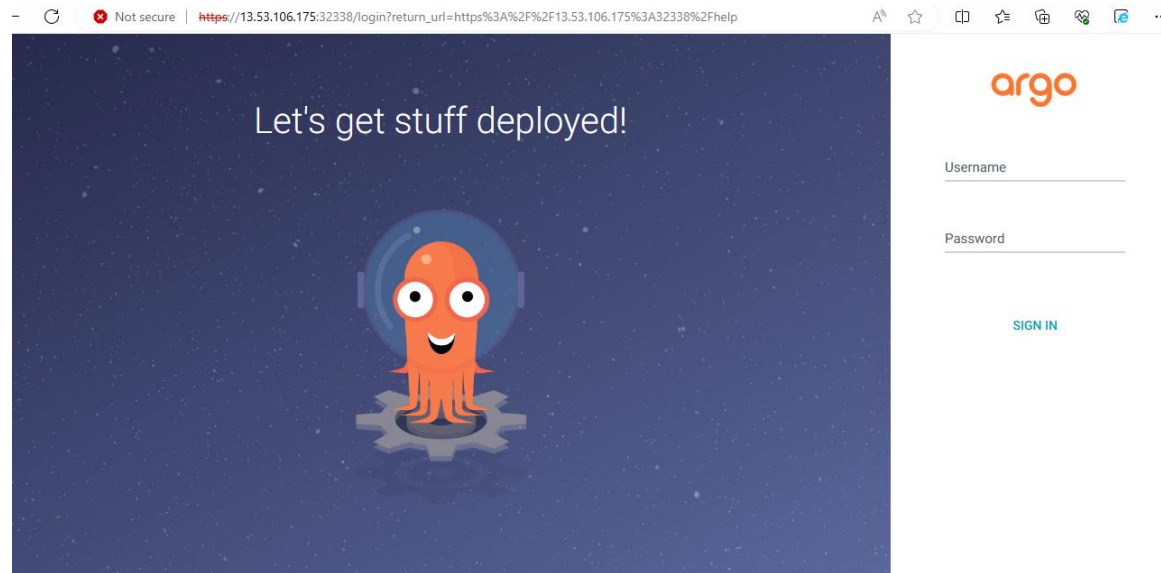
. kubectl get nodes -o wide # copy any one external ip

INTERNAL-IP	EXTERNAL-IP
192.168.23.91	13.53.106.175
192.168.58.142	16.171.254.253

. kubectl get svc      # copy the port no of example-argocd-server like highlighted below

```
example-argocd-server      NodePort      10.100.79.236      <none>      80:32338/TCP,443:31951/TCP      20d
```

Then put Externalip:port no on browser like 13.53.106.175:32338, you will see below login page....



put id as "admin " and put pass that you have got by ran above command "echo | base64"

**Our**

**infrastructure is ready now.**

**Lets' login on each tool that we have installed**

Before login, go to your AWS account click on your Jenkins machine and enable all ports and do same for others machine as well.

### **1. Login to jenkins's**

. Go to your AWS account click on your Jenkins machine and copy the pulic ip. Paste the public with colon 8080. Ex 2.3.5.6:8080

. After that login to your jenkins machine, - Run the command to copy the Jenkins Admin Password -  
sudo cat /var/lib/jenkins/secrets/initialAdminPassword - then Enter the Administrator password in Jenkins appliaction

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/var/lib/jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

. Click on install suggested plugins



## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

. Wait for the Jenkins to Install suggested plugins

# Getting Started

✓ Folders	Formatter			Git
✓ Timestampers	✓ Workspace Cleanup	✓ Ant	✓ Gradle	** GitHub
✓ Pipeline	✓ GitHub Branch Source	✓ Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View	GitHub Branch Source
✓ Git	✓ SSH Build Agents	✓ Matrix Authorization Strategy	✓ PAM Authentication	Pipeline: GitHub Groovy Libraries
✓ LDAP	✓ Email Extension	✓ Mailer		** Pipeline Graph Analysis
				** Pipeline: REST API
				Pipeline: Stage View
				Git
				SSH Build Agents
				Matrix Authorization Strategy
				PAM Authentication
				LDAP
				Email Extension
				Mailer

. Create First Admin User or Skip the step [If you want to use this Jenkins instance for future use-cases as well, better to create admin user]

## Create First Admin User

Username

admin

Password

.....

Confirm password

.....

Full name

devops

E-mail address

test123@gmail.com

. Jenkins Installation is Successful. You can now use the Jenkins



# Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

## # let's follow some more steps.

Go to Manage Jenkins > Manage Plugins.

In the Available tab, search for "Docker Pipeline, SonarQube Scanner for Jenkins".

Select the plugin and click the Install button.

Restart Jenkins after the plugin is installed.

## # Then go to your jenkins machine CLI and run below commands.

```
. sudo apt update
```

```
. sudo apt install docker.io
```

```
. sudo -i
```

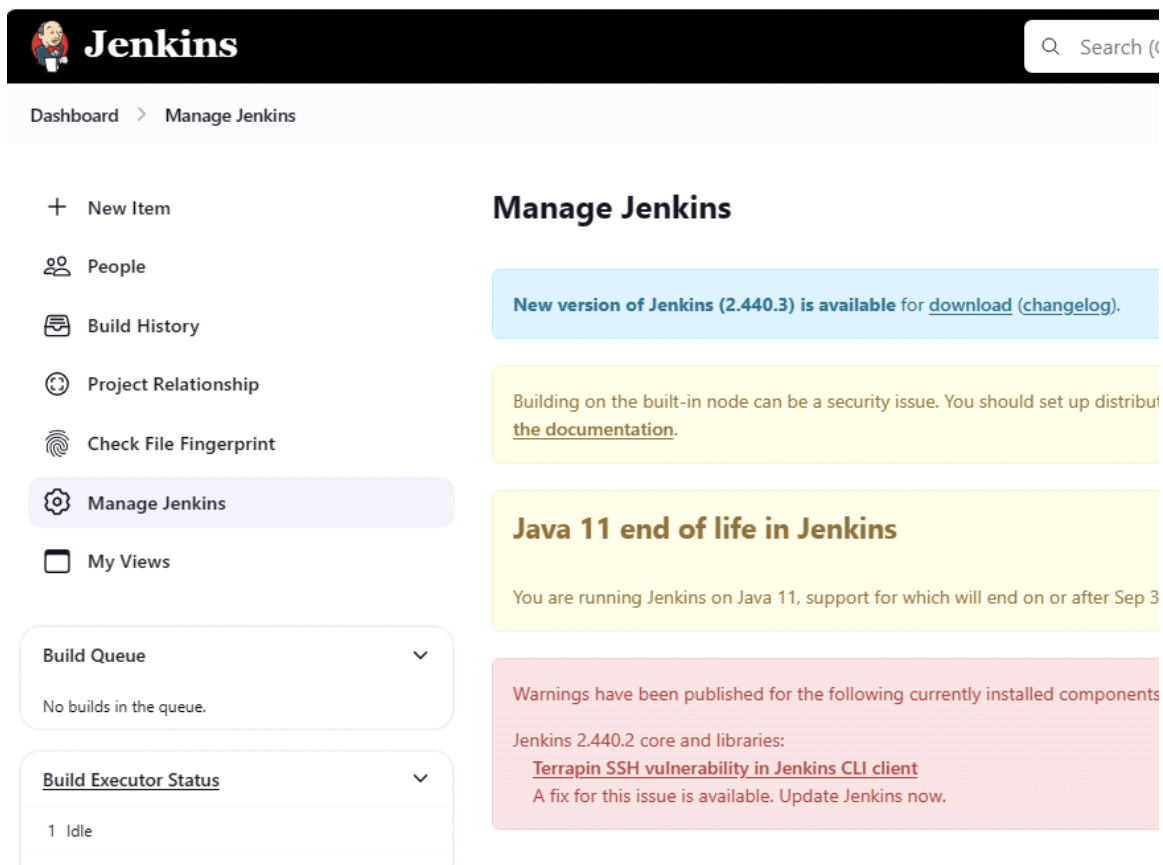
```
. usermod -aG docker jenkins
```

```
. usermod -aG docker ubuntu
```

```
. systemctl restart docker
```

## # Go to jenkins GUI again....

. Click on manage Jenkins















The screenshot shows the Jenkins web interface. At the top is a black header with the Jenkins logo and a search bar. Below the header is a breadcrumb trail: "Dashboard > Manage Jenkins". On the left is a sidebar with a list of navigation items: "New Item", "People", "Build History", "Project Relationship", "Check File Fingerprint", "Manage Jenkins" (which is highlighted with a light purple background), and "My Views". Below the sidebar are two expandable sections: "Build Queue" (showing "No builds in the queue.") and "Build Executor Status" (showing "1 Idle"). The main content area is titled "Manage Jenkins". It contains three informational boxes: a blue box announcing "New version of Jenkins (2.440.3) is available for [download](#) ([changelog](#)).", a yellow box warning that "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).", and another yellow box titled "Java 11 end of life in Jenkins" stating "You are running Jenkins on Java 11, support for which will end on or after Sep 30, 2019". At the bottom is a red box with warnings: "Warnings have been published for the following currently installed components: Jenkins 2.440.2 core and libraries: Terrapin SSH vulnerability in Jenkins CLI client. A fix for this issue is available. Update Jenkins now."


. Scroll down then click on crendtails then click on global

← ↻ ⚠ Not secure | 16.16.144.175:8080/manage/credentials/

Dashboard > Manage Jenkins > Credentials

		System	(global)
		System	(global)
		System	(global)
		System	(global)
		System	(global)
		System	(global)

### Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

. Click on ADD credentils then add below credentials one by one

# New credentials

Kind

Secret text

Scope ?

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

Secret

|

ID ?

## # Credential for sonar login

select kind as 'secret text'.

Id as 'sonarqube'

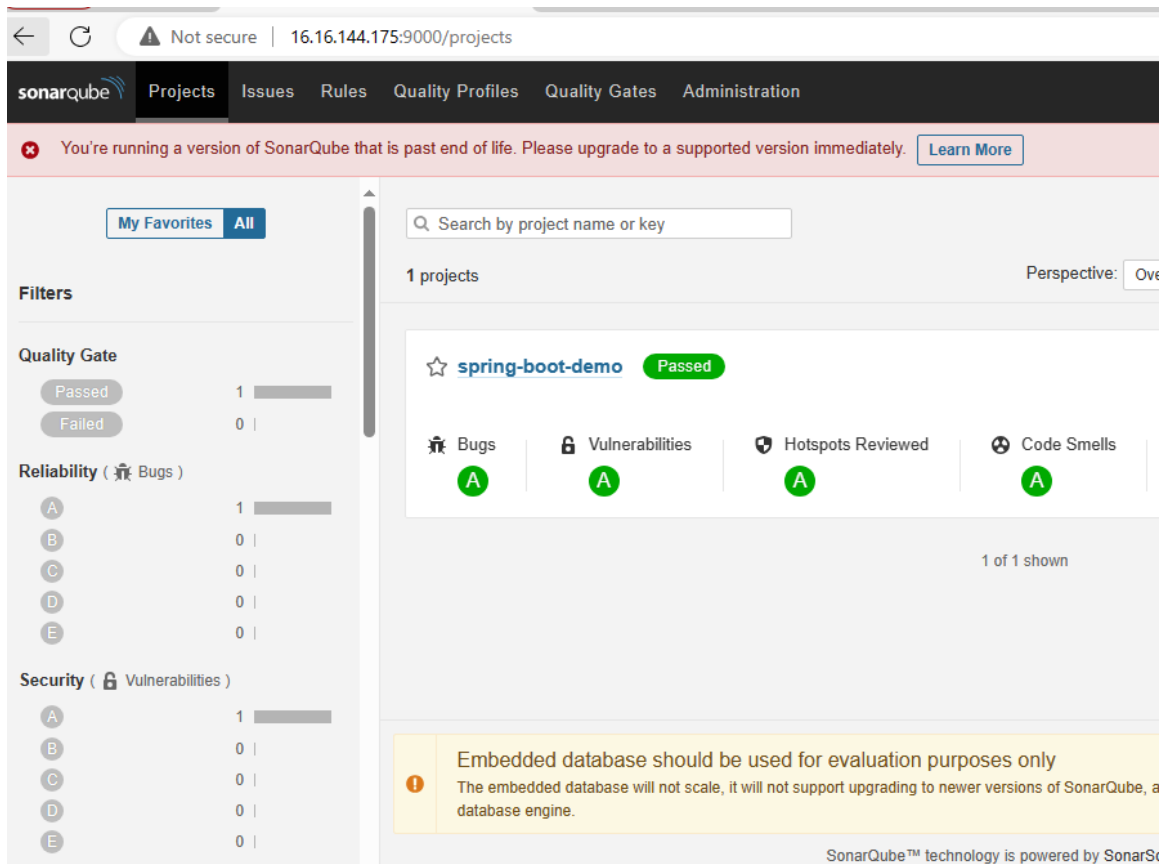
In secret need to put sonarqube secret for this follow below steps

. login to sonar qube

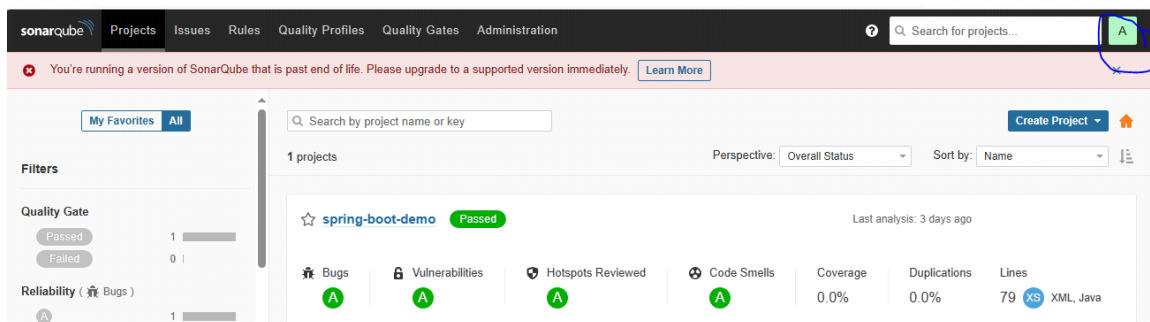
To login, take ip of your jenkins machine and paste it to browser with :9000

example: 12.3.4.2:9000

use user 'admin', password as 'admin'



. After login to sonar GUI, Click on 'A' icon present on top right corner on Sonar GUI



. Click on my account ----> security -----> put the name under the generate token section ----> Generate.

ast end of life. Please upgrade to a supported version immediately.

[Learn More](#)

A

Administrator

Profile

Security

Notifications

Projects

## 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

Generate

Name

Last use

Created

. Copy the token and put in secret tab of jenkins

# New credentials

Kind

Secret text

Scope ?

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

Secret

.....

ID ?

sonargube

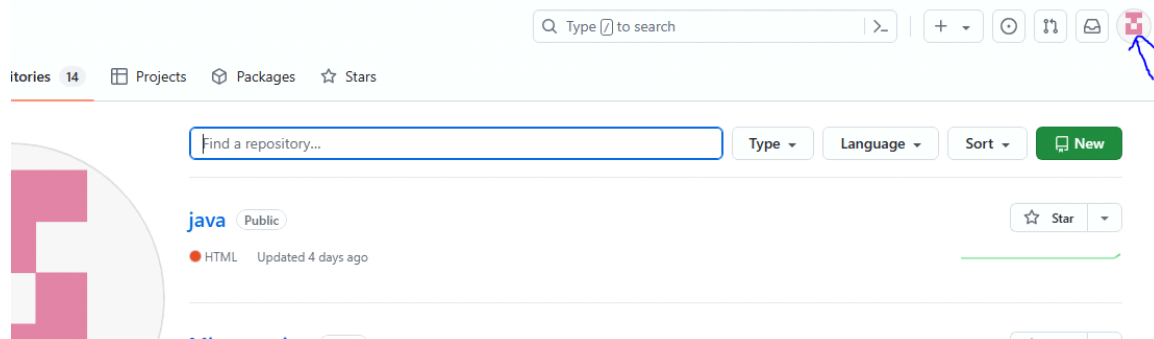
## 2. let's add Git credentials, click on add button again.

select kind as 'secret text'.

Id as 'sanjugithub'

In secret need to put git secret, for this follow below steps

. login to your git account ---> click on settings ---> at right side button click on developer option  
---> personal token ---> token classic ---> generate new token ---> give any name, select all permissions,  
generate it, copy it and put in secret box of jenkins.





😊 Set status

👤 Your profile

👤+ Add account

📁 Your repositories

📁 Your projects

🤖 Your Copilot

🏢 Your organizations

🌐 Your enterprises

★ Your stars

❤️ Your sponsors

📄 Your gists

⬆️ Upgrade

🌐 Try Enterprise

Free

🔧 Feature preview

⚙️ Settings



moderation

## Code, planning, and automation

Repositories

Codespaces

Packages

Copilot

Pages

← Saved replies

## Security

Code security and analysis

## Integrations

Applications

Scheduled reminders

## Archives

Security log

Sponsorship log

<> Developer settings

Don't specify

## URL

## ORCID iD

ORCID provides a persistent identifier - an ORCID [ORCID.org](https://orcid.org).

id Connect your ORCID iD

## Social accounts

Link to social profile

Link to social profile

Link to social profile

Link to social profile

## Company

You can @mention your company's GitHub org

## Location

☐ Display current local time

Settings / Developer Settings

Q Type to search

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens

Tokens (classic)

## Personal access tokens (classic)

Generate new token

Tokens you have generated that can be used to access the Git

jenis — admin:enterprise, admin:pgp\_key, admin:org, admin:org\_hook, admin:repo\_hook, admin:ssh\_signing\_key, audit\_log, codespace, copilot, repo, user, workflow, write:discussion, write:packages  
Expires on Fri, May 24 2024.

Generate new token (Beta)  
Fine-grained, repo-scoped

Generate new token (classic)  
For general use

3. let's add Dockerhub credentials, click on add button again.

select kind as 'username and pass'. Give you user id and pass of dockerhub

Id as 'docker-cred' then save it.

**Use the ids name same as i used**

**Let's start**

## building CI pipeline

1. Login to jenkins GUI ---> click on new item ---> click on pipeline, give name and click on OK


---

Dashboard ▾ > All >


**Enter an item name**

JAVA


» A job already exists with the name 'JAVA'



**Freestyle project**  
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific

2. go to botom ---> under pipeline section select defination as SCM ---> SCM as git ---> put link your repo where the jenkins file is present ---> click on apply an save.

Dashboard > java > Configuration

## Configure

- General
- Advanced Project Options
- Pipeline

Definition  
Pipeline script from SCM

SCM ?  
Git

Repositories ?

Repository URL ?

Credentials ?  
- none -

+ Add

Advanced

### step to copy repo link

Go to your git account ---> click on repo ---> click on code ---> then copy the url

3. Go to your jenkins and follow below step to configure email notificaiton

go to jenkins ---> Manage jenkins ---> systems ---> fill the below info in email notification and in Extended E-mail Notification section

### In E-mail Notification

SMTP server = smtp.gmail.com

SMTP Port = 465

In advance put your email id and get password by following below steps

go to your manage google account settings ---> search app password ---> create app password and copy it.

## ← App passwords

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

[Learn more](#)

### Your app passwords

jenkins

Created at 24 Apr, last used at 28 Apr

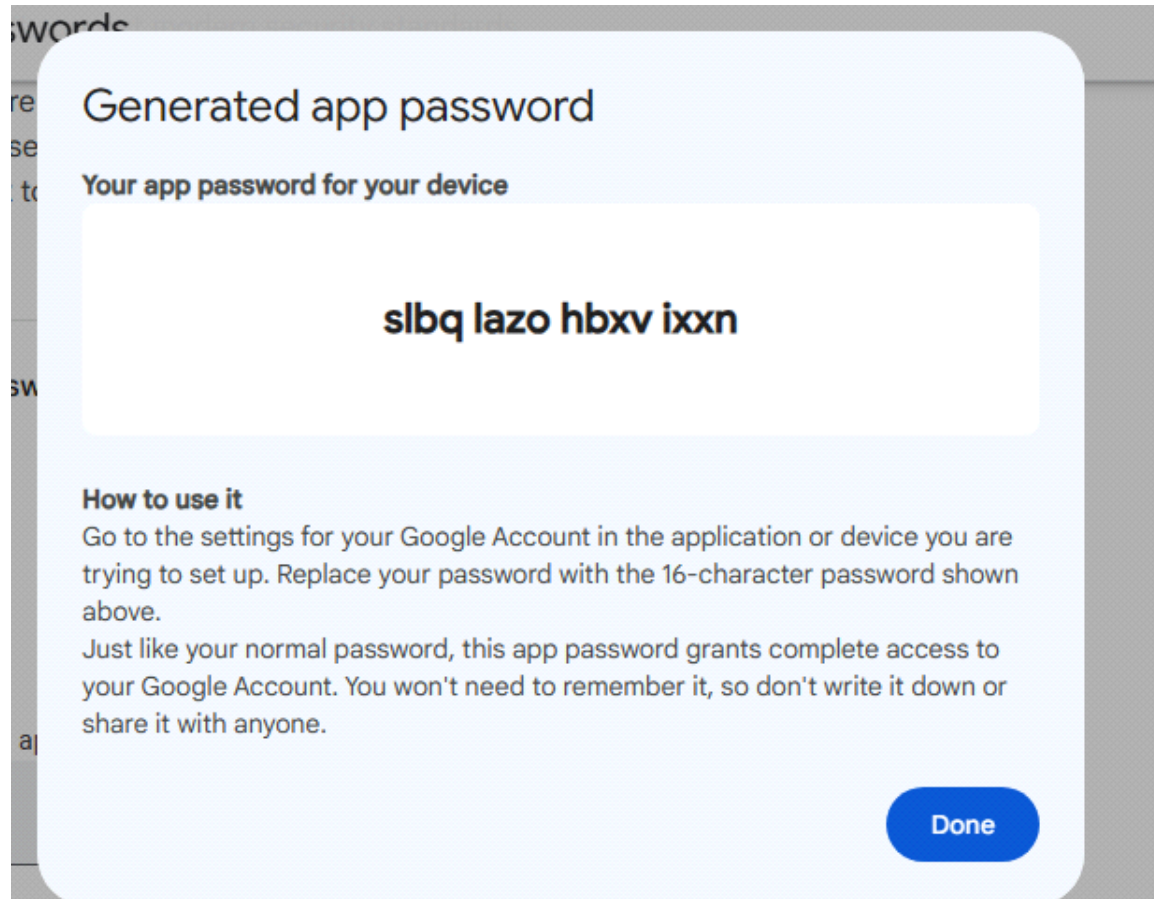


To create a new app-specific password, type a name for it below...

App name

jenkins

Create



Put info like below

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

Advanced ^ Edited

☒ Use SMTP Authentication ?

User Name

sanjukum@gmail.com

Password

Concealed [Change Password](#)

☒ Use SSL ?

☐ Use TLS

SMTP Port ?

465

### Extended E-mail Notification section

SMTP server = smtp.gmail.com

SMTP Port = 465

In credentials click on add then create credentials with username as 'your email id' and use same pass that we have used above.

### Extended E-mail Notification

SMTP server

smtp.gmail.com

SMTP Port

465

Advanced ^

 Edited

Credentials

sanjukumar@gmail.com/\*\*\*\*\*

+ Add ▾

☒ Use SSL

☐ Use TLS

☐ Use OAuth 2.0

After that click on save button

**Open jenkins file from github and make some below changes then save it**

```
pipeline {  
    agent {
```

```

docker {
    image 'abhishekf5/maven-abhishek-docker-agent:v1'
    args '--user root -v /var/run/docker.sock:/var/run/docker.sock'
}
}
stages {
    stage('Checkout') {
        steps {
            git branch: 'master', url: 'https://github.com/Sanjay6372/java.git' # change link
with your git repo link
        }
    }
    stage('Build and Test with Maven') {
        steps {
            sh 'cd spring-boot-app && mvn clean package'
        }
    }
    stage('Static Code Analysis with SonarQube') {
        environment {
            SONAR_URL = "http://16.16.198.24:9000" # only change ip with your
Sonarqube machine IP
        }
        steps {
            withCredentials([string(credentialsId: 'sonarqube', variable:
'SONAR_AUTH_TOKEN')]) {
                sh 'cd spring-boot-app && mvn sonar:sonar
-Dsonar.login=${SONAR_AUTH_TOKEN} -Dsonar.host.url=${SONAR_URL}'
            }
        }
    }
}

```

```

    }

    stage('Build and Push Docker Image') {

        environment {

            DOCKER_IMAGE = "sanjay9888/javagapp:${BUILD_NUMBER}"    # change
sanjay9888 with your docker hub user name

            REGISTRY_CREDENTIALS = credentials('docker-cred')

        }

        steps {

            script {

                sh 'cd spring-boot-app && docker build -t ${DOCKER_IMAGE} .'

                def dockerImage = docker.image("${DOCKER_IMAGE}")

                docker.withRegistry('https://index.docker.io/v1/', "docker-cred") {

                    dockerImage.push()

                }

            }

        }

    }

    stage('Update Deployment File') {

        environment {

            GIT_REPO_NAME = "java"    # change java with your git repo name where code
is it.

            GIT_USER_NAME = "Sanjay6372" # change sanjay6372 with your git repo usr
name.

        }

        steps {

            withCredentials([string(credentialsId: 'sanjugithub', variable: 'GITHUB_TOKEN')]) {

                sh '''

                    rm -rf spring-boot-app/target/

```



```

        git config --global user.email "sanjay@gmail.com"

        git config --global user.name "sanjay"

        sed -i "s/5/${BUILD_NUMBER}/g"
spring-boot-app-manifests/deployment.yml

        git add spring-boot-app-manifests/deployment.yml

        git commit -m "Update deployment image to version ${BUILD_NUMBER}"

        git push
https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME} HEAD:master
    ""
    }
    }
    }
}

post {
    always {
        emailx (
            subject: "Pipeline Status: ${BUILD_NUMBER}",
            body: ""
                <html>
                    <body>
                        <p>Build Status: ${BUILD_STATUS}</p>
                        <p>Build Number: ${BUILD_NUMBER}</p>
                        <p>Check the <a href="${BUILD_URL}">console output</a>.</p>
                    </body>
                </html>
            "",
            to: 'sanjukumar62@gmail.com', # change sanjukumar62@gmail.com with your
email id
            from: 'jenkins@example.com',

```

```

        replyTo: 'jenkins@example.com',

        mimeType: 'text/html'

    )

}

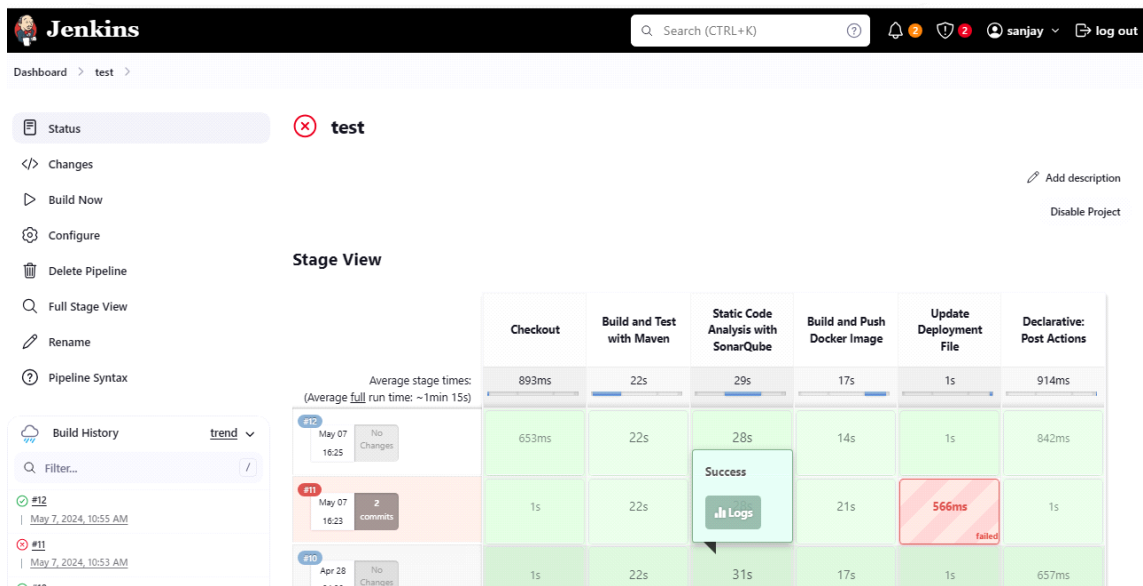
}

}

```

Remove all the lines that are in bold before run above command

Now click on your job then click on Build now



Horrey!! your CI part is successfully completed, let's move onto deployment part

Login to ArgoCD and follow the below step.

1. click on application

2. click on new app

enter name as 'java'

project as 'default'

in source put your code repo link

in path put 'spring-boot-app-manifests'

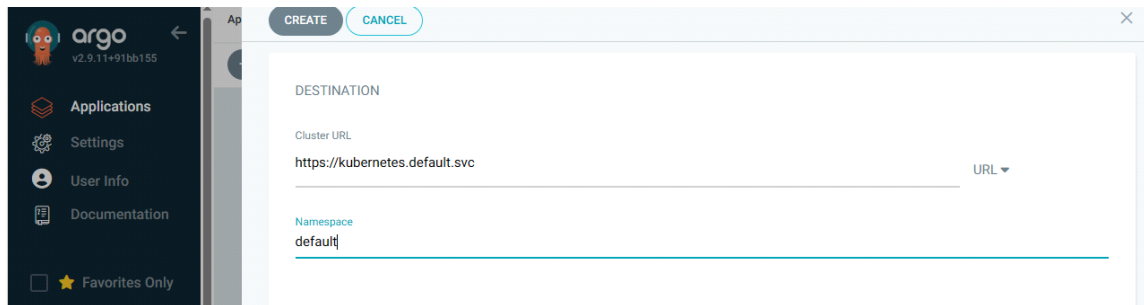
### in destination section

select path as 'https://kubernetes.default.svc'

namespace as 'default' and click on create

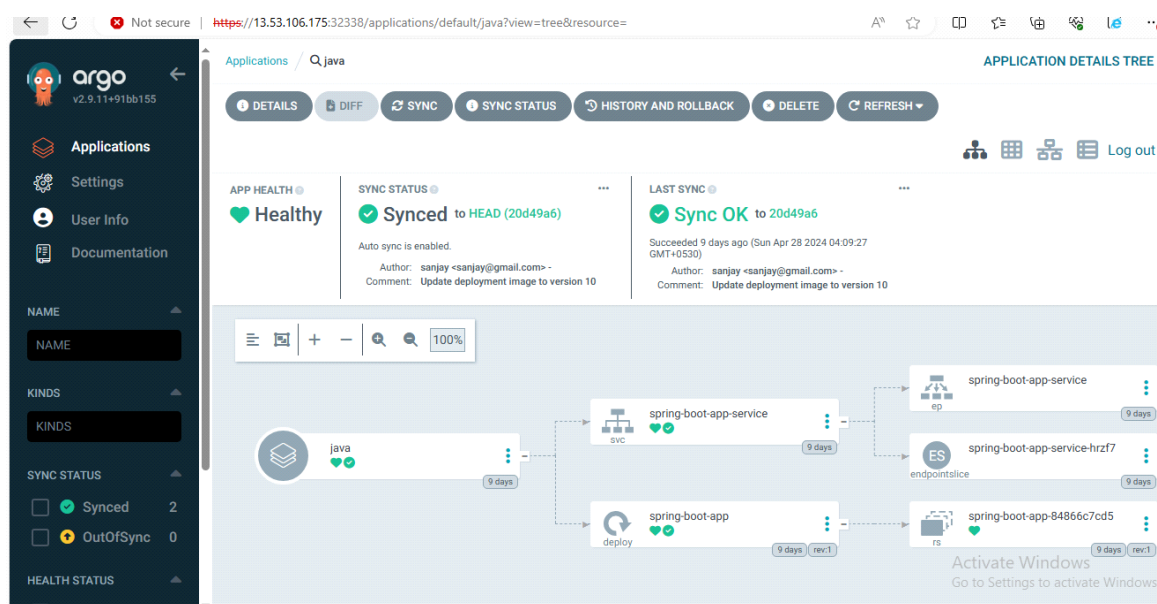
The screenshot shows the top portion of the Argo CD 'CREATE' form. On the left is a sidebar with the Argo logo, version 'v2.9.11+91bb155', and navigation links: Applications, Settings, User Info, and Documentation. Below these are filters for 'Favorites Only' and 'SYNC STATUS' (Unknown: 0, Synced: 1, OutOfSync: 0) and 'HEALTH STATUS' (Unknown: 0, Progressing: 0, Suspended: 0). The main form area has a 'CREATE' button and a 'CANCEL' button. The form fields are: 'Application Name' with the value 'java', 'Project Name' with the value 'default', and 'SYNC POLICY' set to 'Manual'. There are several checkboxes: 'SET DELETION FINALIZER' (unchecked), 'SKIP SCHEMA VALIDATION' (unchecked), 'PRUNE LAST' (unchecked), 'RESPECT IGNORE DIFFERENCES' (unchecked), 'AUTO-CREATE NAMESPACE' (unchecked), 'APPLY OUT OF SYNC ONLY' (unchecked), and 'SERVER-SIDE APPLY' (unchecked). The 'PRUNE PROPAGATION POLICY' is set to 'foreground'. At the bottom left of the form is a 'REPLACE' checkbox with a warning icon. A Windows watermark is visible in the bottom right corner.

The screenshot shows the bottom portion of the Argo CD 'CREATE' form. The form fields are: 'Repository URL' with the value 'https://github.com/Sanjay6372/java.git' and a 'GIT' dropdown menu, 'Revision' with the value 'HEAD' and a 'Branches' dropdown menu, and 'Path' with the value 'spring-boot-app-manifests'. Below these is the 'DESTINATION' section, which includes 'Cluster URL' with the value 'https://kubernetes.default.svc' and a 'URL' dropdown menu. A Windows watermark is visible in the bottom right corner.



After click on create it will take few minutes to complete the process

After completion it will look like below



Horrey! you have successfully deployed your application on aws EKS, let's access our application

. Login to your EKS machine through mobaxterm

. Run below command to see port no of your application and copy it

kubectl get svc spring-boot-app-service

```
root@ip-172-31-46-226:~# kubectl get svc spring-boot-app-service
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
spring-boot-app-service            NodePort    10.100.113.137  <none>       80:30856/TCP     9d
```

Copy the port no, here port no is 30856

. Run below command to get external ip to access our app

kubectl get nodes -o wide

```
root@ip-172-31-46-226:~# kubectl get nodes -o wide
NAME                                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE
ip-192-168-23-91.eu-north-1.compute.internal Ready    <none>   19d   v1.29.0-eks-5e0fdde 192.168.23.91  13.53.106.175  Amazon Linux 2
ip-192-168-58-142.eu-north-1.compute.internal Ready    <none>   19d   v1.29.0-eks-5e0fdde 192.168.58.142 13.53.106.175  Amazon Linux 2
```

Copy the any node external ip and paste it on browser along with our app port no

Example: 13.53.106.175:30856



I have successfully built a sprint boot application using  
Maven

This application is deployed on to Kubernetes using Argo CD

Our project is  
finished here

Thanks





