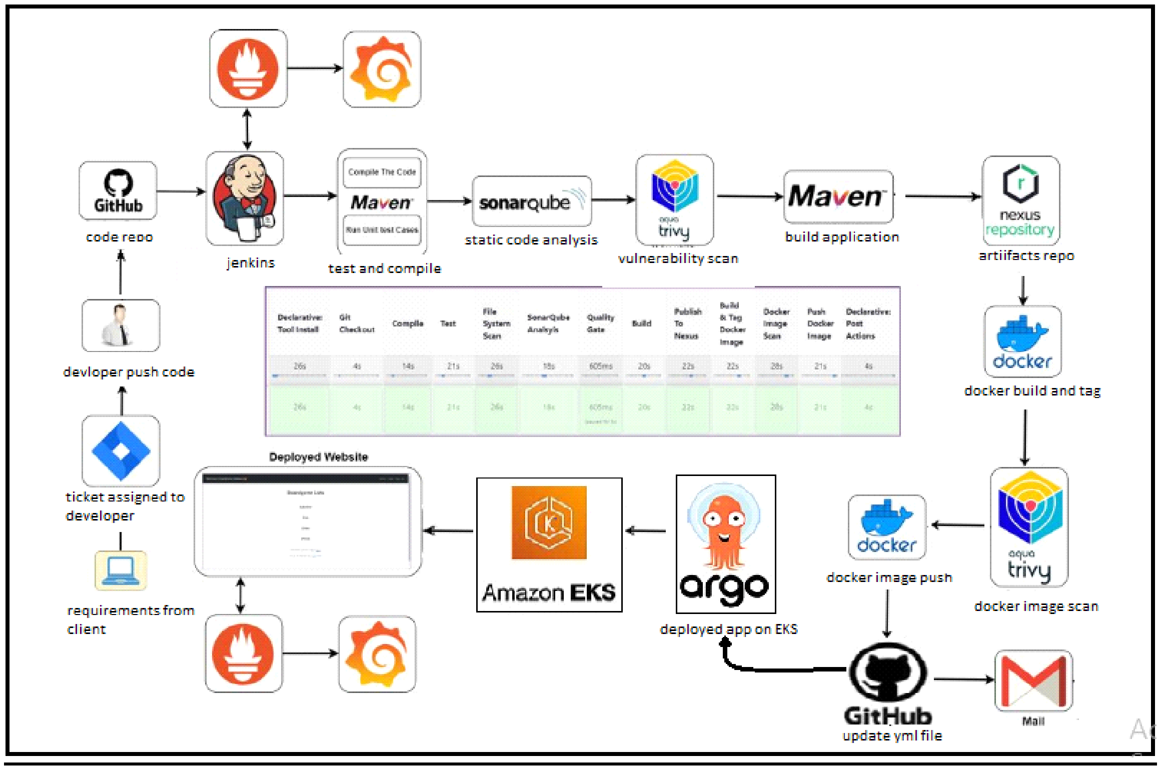
Jenkins Pipeline for Java based application using Maven, SonarQube, Argo CD, AWS EKS, GIT AND GITHUB, Docker, Trivy, Nexus.



Declarative: Tool Install	Git Checkout	Compile	Test	File System Scan	SonarQube Analysis	Quality Gate	Build	Publish To Nexus	Build & Tag Docker Image	Docker Image Scan	Push Docker Image	Declarative: Post Actions
26s	4s	14s	21s	26s	18s	605ms	20s	22s	22s	28s	21s	4s
26s	4s	14s	21s	26s	18s	605ms	20s	22s	22s	28s	21s	4s

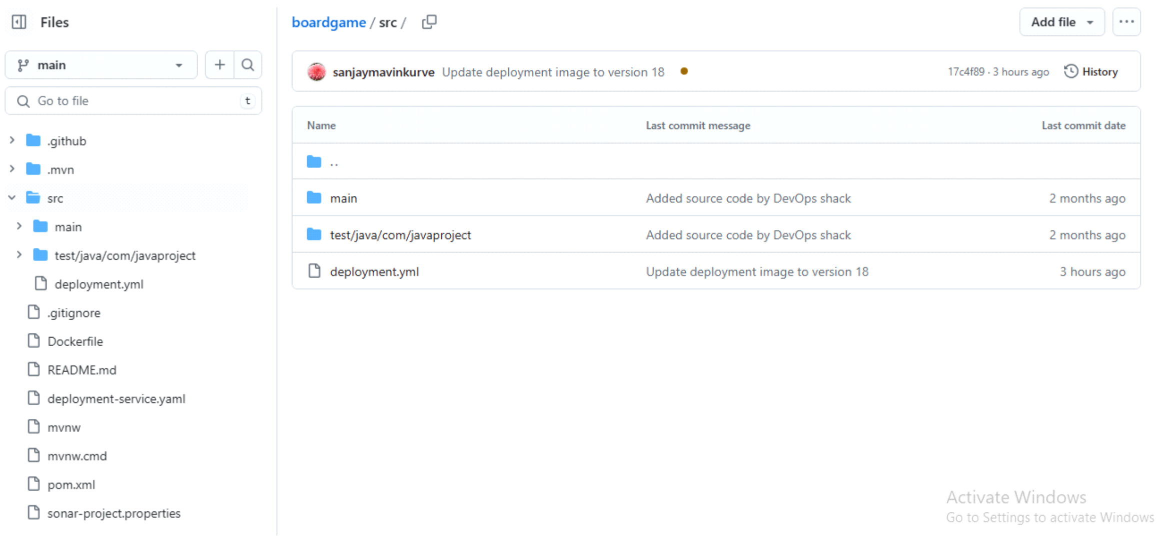
Prerequisites:

1. Basic knowledge of Jenkins, Docker, Kubernetes, Maven, SonarQube, Git, GitHub, and ArgoCD, AWS, Nexus, trivy.

2. DockerHub and GitHub accounts are required.

3. Fork my repository https://github.com/Sanjay6372/boardgame.git.

code hierarchy


Files

main

Go to file

- .github
- .mvn
- src
  - main
  - test/java/com/javaproject
  - deployment.yml
- .gitignore
- Dockerfile
- README.md
- deployment-service.yaml
- mvnw
- mvnw.cmd
- pom.xml
- sonar-project.properties

boardgame / src /

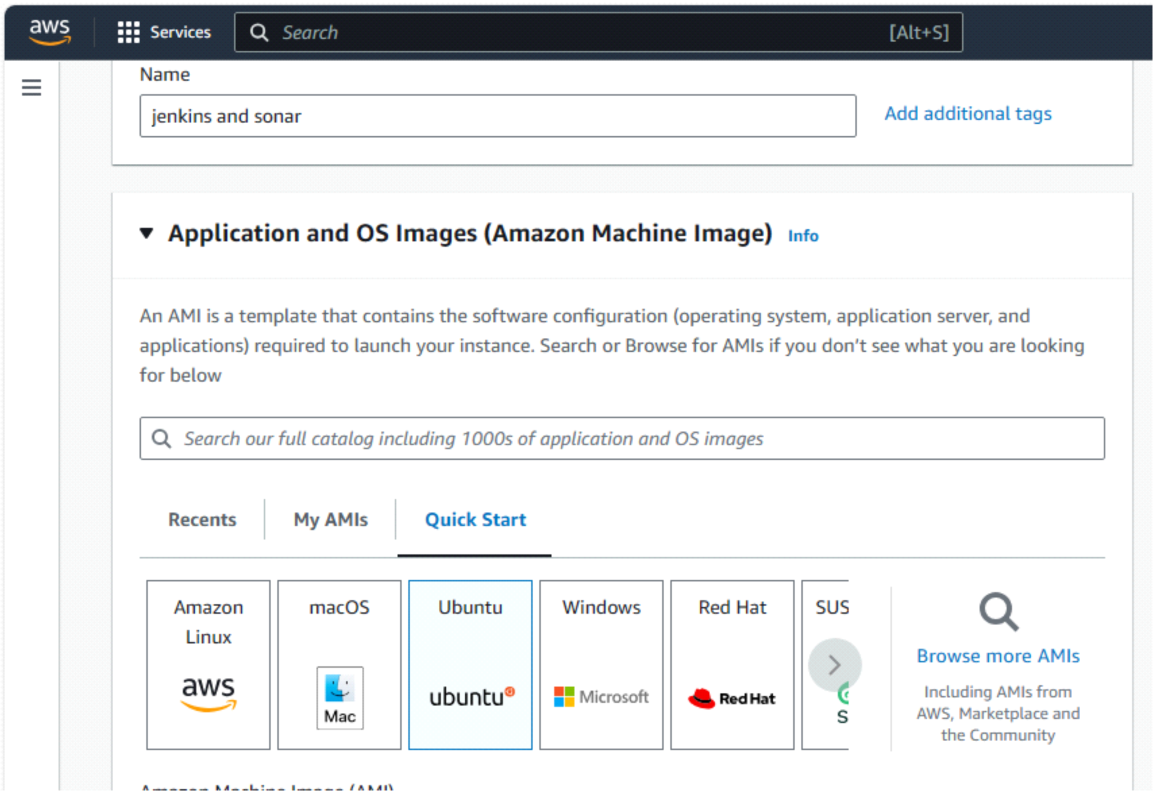
sanjaymavinkurve Update deployment image to version 18 · 17c4f89 · 3 hours ago · History

Name	Last commit message	Last commit date
..		
main	Added source code by DevOps shack	2 months ago
test/java/com/javaproject	Added source code by DevOps shack	2 months ago
deployment.yml	Update deployment image to version 18	3 hours ago

Activate Windows
Go to Settings to activate Windows

1. Create a t2 or t3 medium type of EC2 instance with Ubuntu OS for jekins 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.



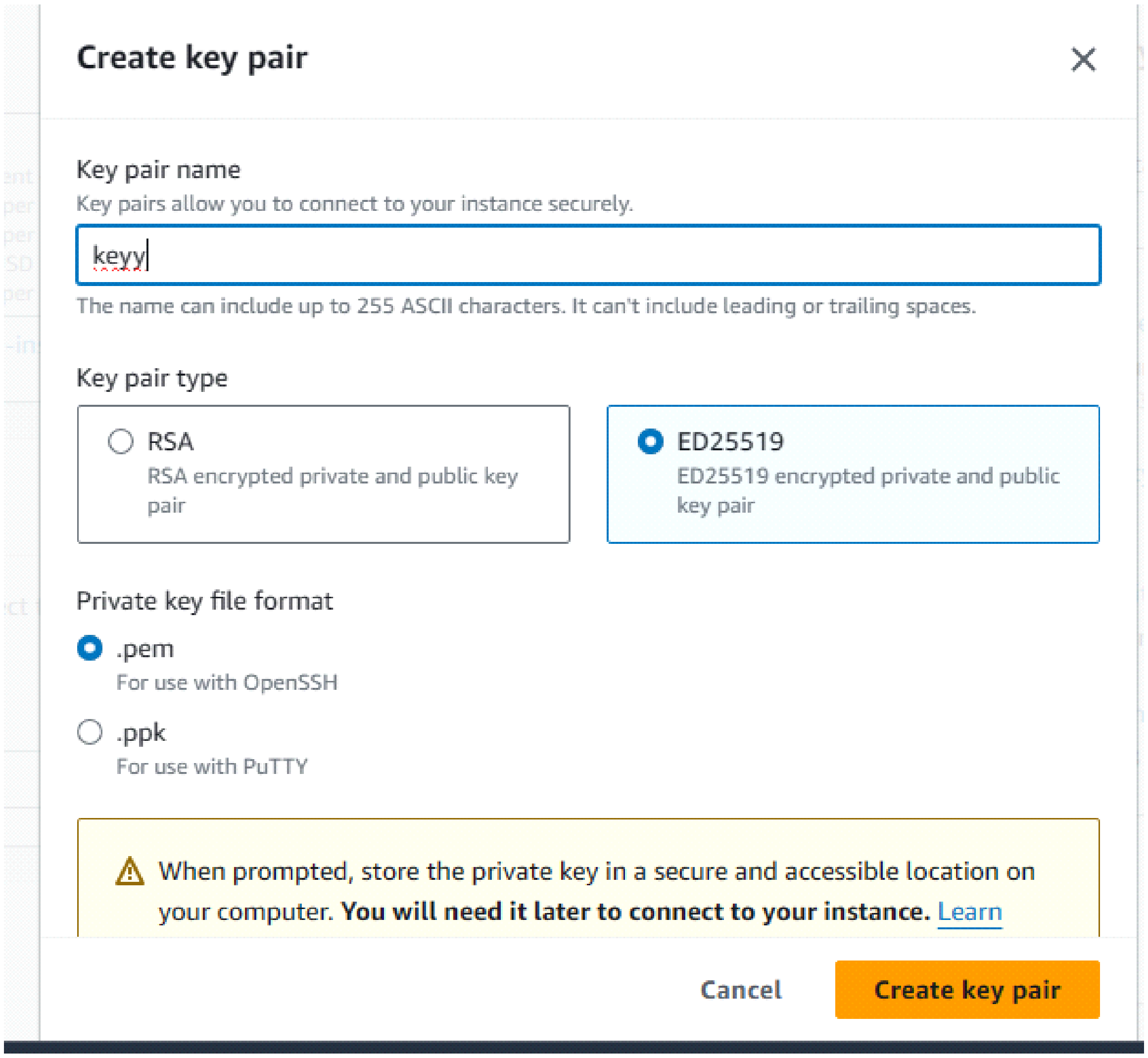
Name

jenkins and sonar

Application and OS Images (Amazon Machine Image)

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

Search our full catalog including 1000s of application and OS images
Recents
My AMIs
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUS
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community



Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

keyy

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

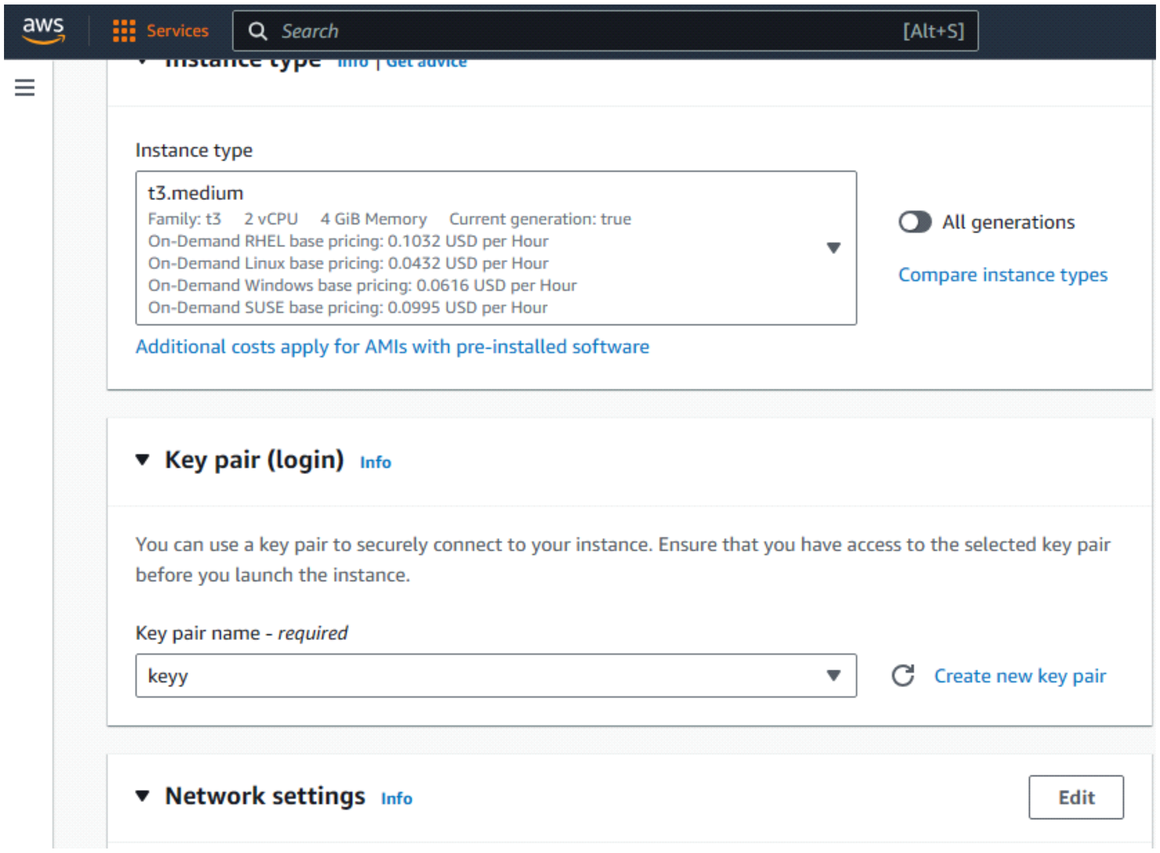
[✓ ED25519]
ED25519 encrypted private and public key pair

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



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

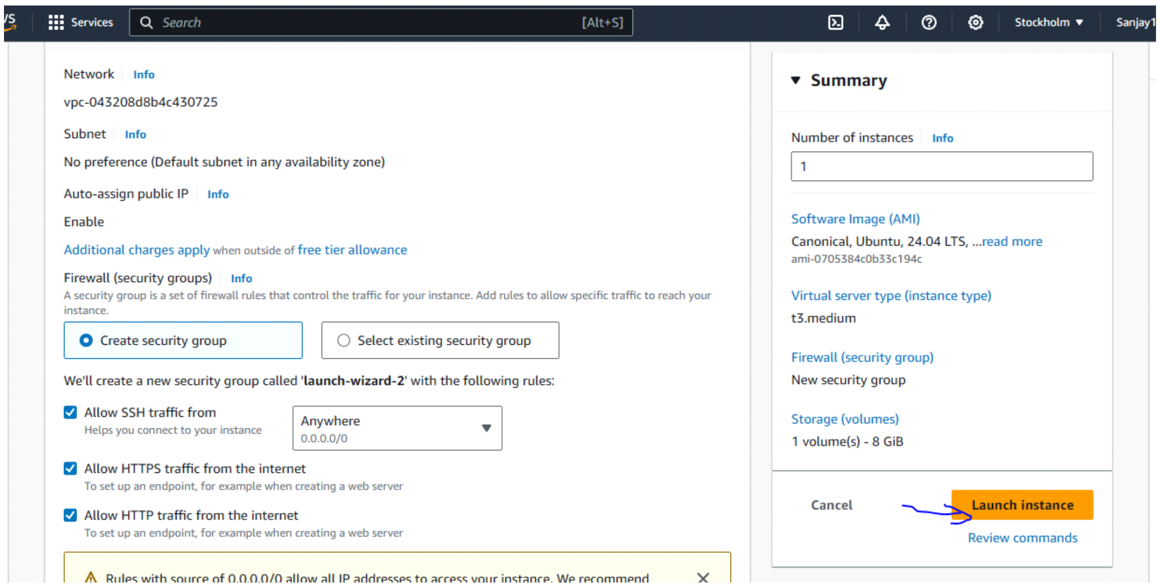
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software

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

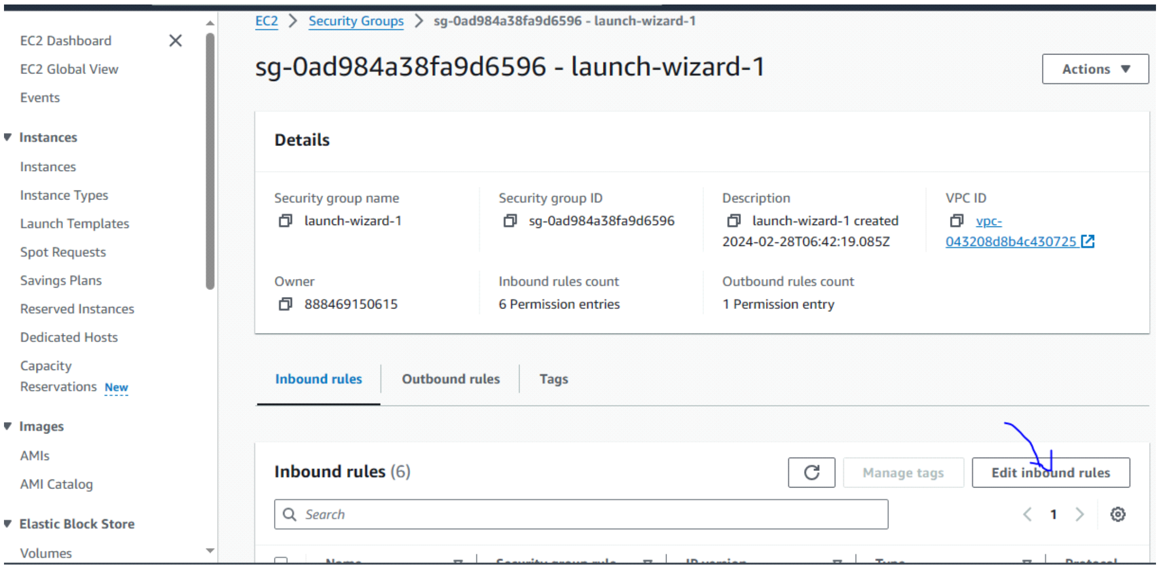
Network settings Info
Edit



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 — Anywhere 0.0.0.0/0
Helps you connect to your instance
[✓] 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.


EC2 > Security Groups > sg-0ad984a38fa9d6596 - launch-wizard-1

sg-0ad984a38fa9d6596 - launch-wizard-1

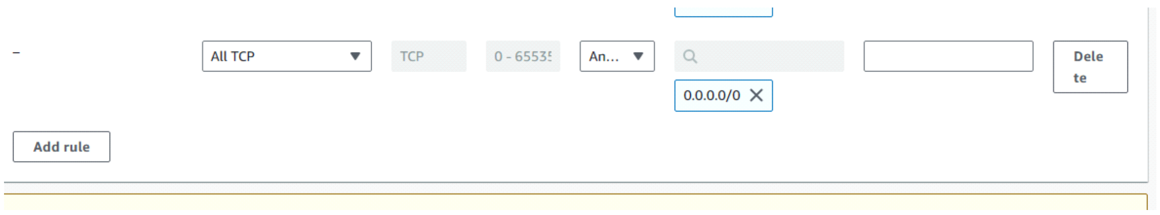
Details

Security group name	Security group ID	Description	VPC ID
launch-wizard-1	sg-0ad984a38fa9d6596	launch-wizard-1 created 2024-02-28T06:42:19.085Z	vpc-043208d8b4c430725

Owner	Inbound rules count	Outbound rules count
888469150615	6 Permission entries	1 Permission entry

Inbound rules (6)

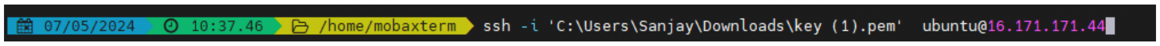
allow all tcp ports and save it.


Type	Protocol	Port range	Source			
All TCP	TCP	0 - 6553!	An...	0.0.0.0/0		Delete

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 consule.


```
07/05/2024  10:37.46  /home/mobaxterm  ssh -i 'C:\Users\Sanjay\Downloads\key (1).pem'  ubuntu@16.171.171.44
```

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

Install docker on jenkins machine

. sudo yum update -y

. sudo yum install -y yum-utils device-mapper-persistent-data lvm2

. sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo

. sudo yum update -y

. sudo yum install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

. sudo systemctl start docker

. sudo systemctl enable docker

. usermod -aG docker jenkins

. usermod -aG docker ubuntu

Install Trivy on jenkins machine by run below commands

. sudo apt-get install wget apt-transport-https gnupg lsb-release

. wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -

. echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | sudo tee -a
/etc/apt/sources.list.d/trivy.list

. sudo apt-get update

. sudo apt-get install trivy

4. Install sonarQube and start by run below commands.

. apt install unzip wget

. sudo apt install default-jdk

. adduser sonarqube

. usermod -aG 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

or

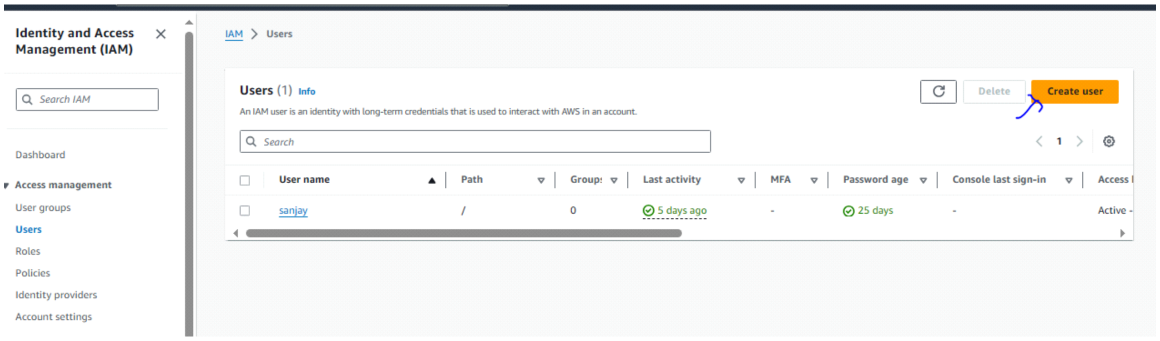
Run below docker command after installing docker on machine

. sudo docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

5. EKS setup on AWS

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

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

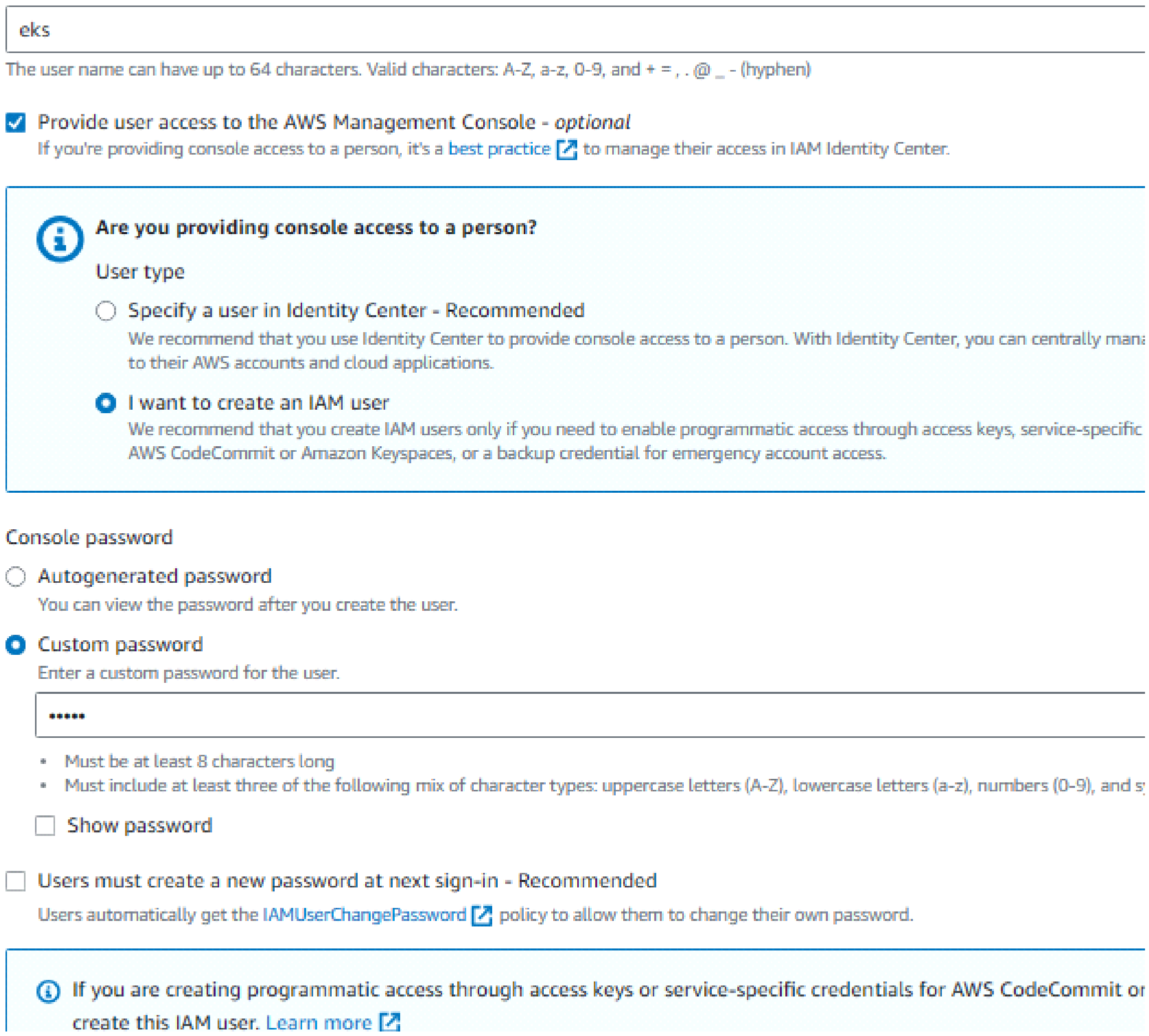


Users (1)

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Group:	Last activity	MFA	Password age	Console last sign-in	Access
sanjay	/	0	5 days ago	-	25 days	-	Active -

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 mana
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 s

[ ] 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
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

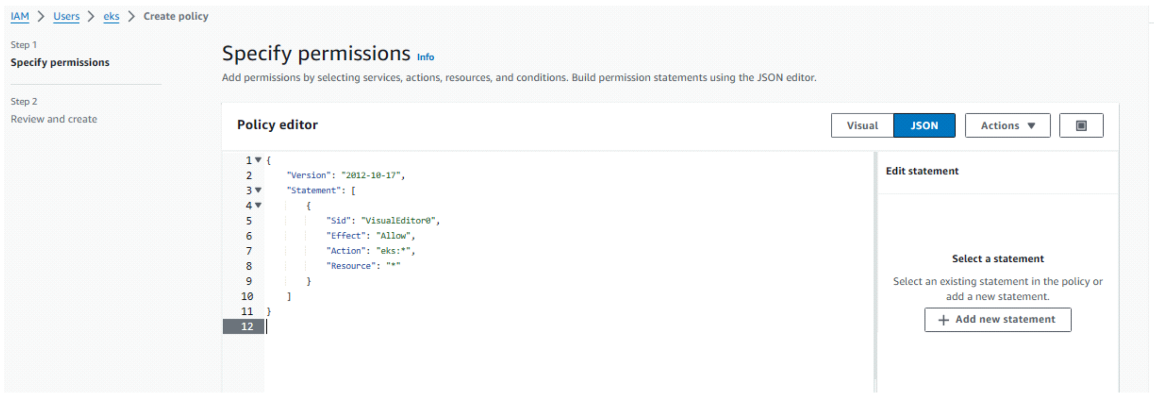
AmazonEKSWorkerNodePolicy

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": "*"
        }
    ]
}
```


IAM > Users > eks > Create policy
Step 1
Specify permissions
Step 2
Review and create
Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.
Policy editor
Visual
JSON
Actions
Edit statement
Select a statement
Select an existing statement in the policy or add a new statement.
Add new statement

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

--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 \   **# change region with your current aws selected region**

--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

6. 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.

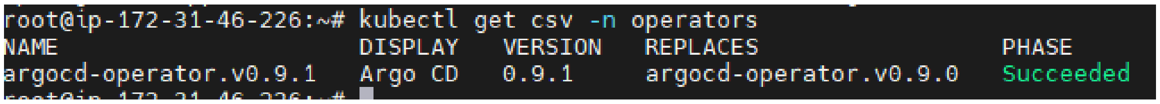
. curl -sL https://github.com/operator-framework/operator-lifecycle-manager/releases/download/v0.27.0/install.sh | bash -s v0.27.0

# 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(please wait for 5mints before run the below 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.

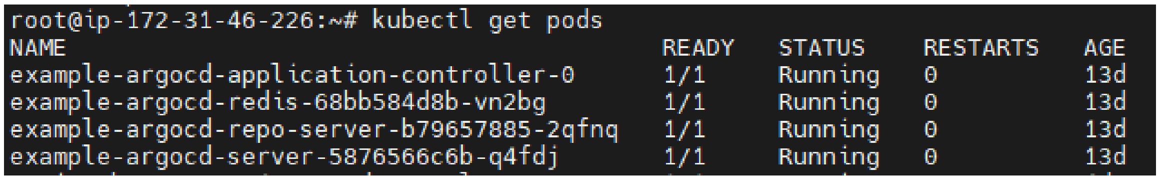
```
apiVersion: argoproj.io/v1alpha1

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

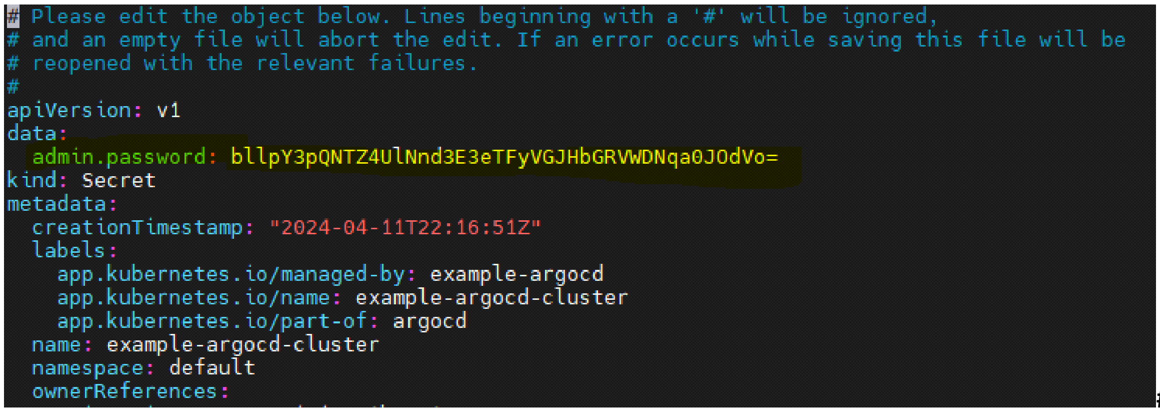


```
root@ip-172-31-46-226:~# kubectl get pods
NAME                                           READY   STATUS    RESTARTS   AGE
example-argocd-application-controller-0        1/1     Running   0          13d
example-argocd-redis-68bb584d8b-vn2bg          1/1     Running   0          13d
example-argocd-repo-server-b79657885-2qfnq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

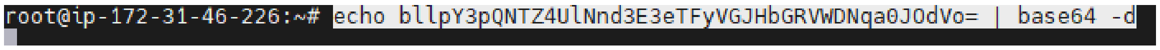
then copy the password , hightlighted below in yellow color



```
# 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: bllpY3pQNTZ4UlNnd3E3eTFyVGJHbGRVWDNqa0JOdVo=
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.

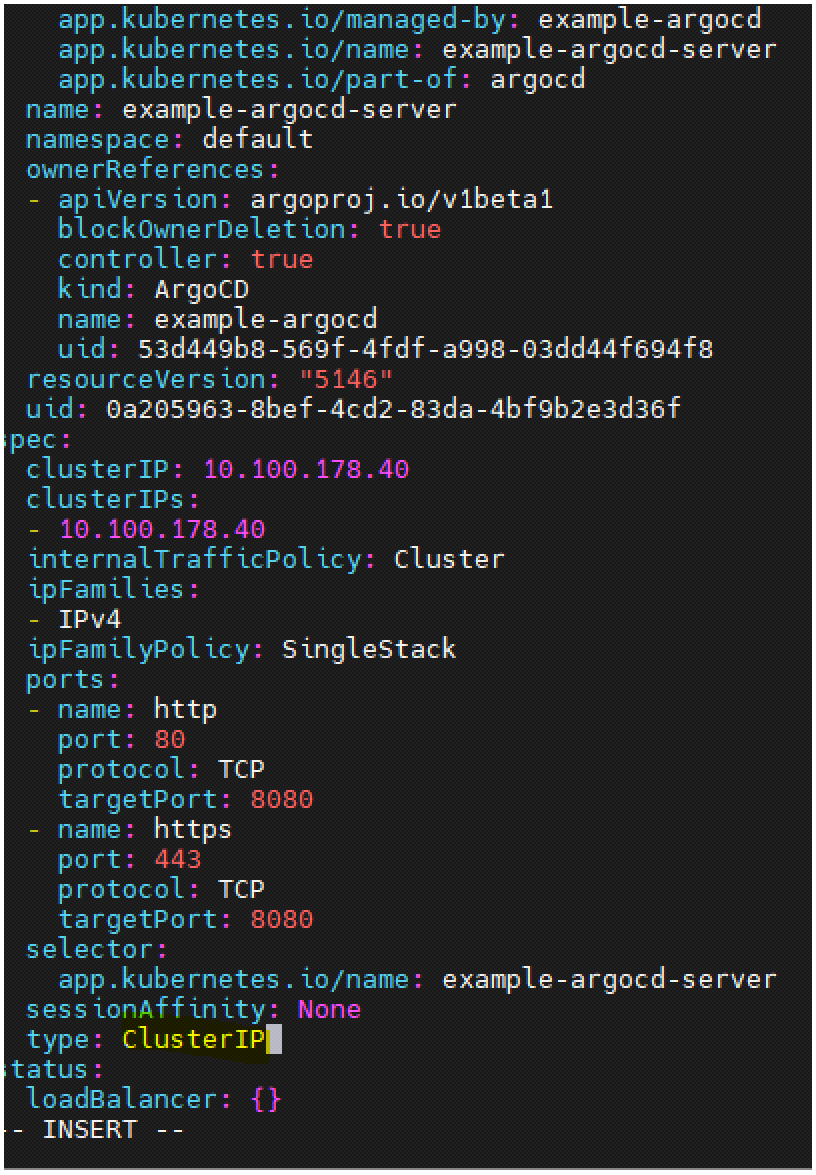
. echo <put password here> | base64 -d



```
root@ip-172-31-46-226:~# echo bllpY3pQNTZ4UlNnd3E3eTFyVGJHbGRVWDNqa0JOdVo= | base64 -d
```

# Run below command

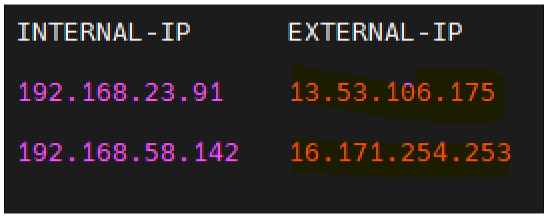
. kubectl edit svc example-argocd-server **# change service : clusterIp to service: NodePort**



```
    app.kubernetes.io/managed-by: example-argocd
    app.kubernetes.io/name: example-argocd-server
    app.kubernetes.io/part-of: argocd
  name: example-argocd-server
  namespace: default
  ownerReferences:
  - apiVersion: argoproj.io/v1beta1
    blockOwnerDeletion: true
    controller: true
    kind: ArgoCD
    name: example-argocd
    uid: 53d449b8-569f-4fdf-a998-03dd44f694f8
  resourceVersion: "5146"
  uid: 0a205963-8bef-4cd2-83da-4bf9b2e3d36f
pec:
  clusterIP: 10.100.178.40
  clusterIPs:
  - 10.100.178.40
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 8080
  - name: https
    port: 443
    protocol: TCP
    targetPort: 8080
  selector:
    app.kubernetes.io/name: example-argocd-server
  sessionAffinity: None
  type: ClusterIP
tatus:
  loadBalancer: {}
- INSERT --
```

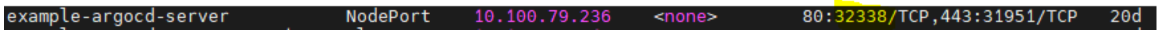
# Run below command to get ip and port no to access our ArgoCd

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



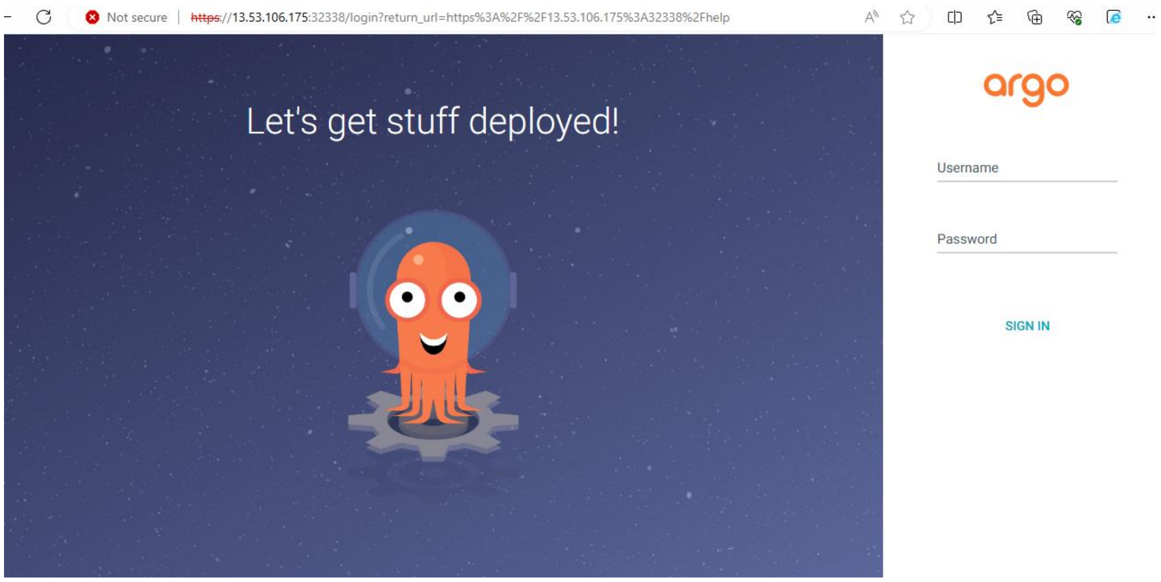
```
INTERNAL-IP       EXTERNAL-IP

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 hightlighted 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....(before login to argo, allow above port in you EKS machines security group)



Not secure | ~~https~~://13.53.106.175:32338/login?return_url=https%3A%2F%2F13.53.106.175%3A32338%2Fhelp

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

7. Install nexus

# First install docker by run below command

. sudo yum update -y

. sudo yum install -y yum-utils device-mapper-persistent-data lvm2

. sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo

. sudo yum update -y

. sudo yum install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

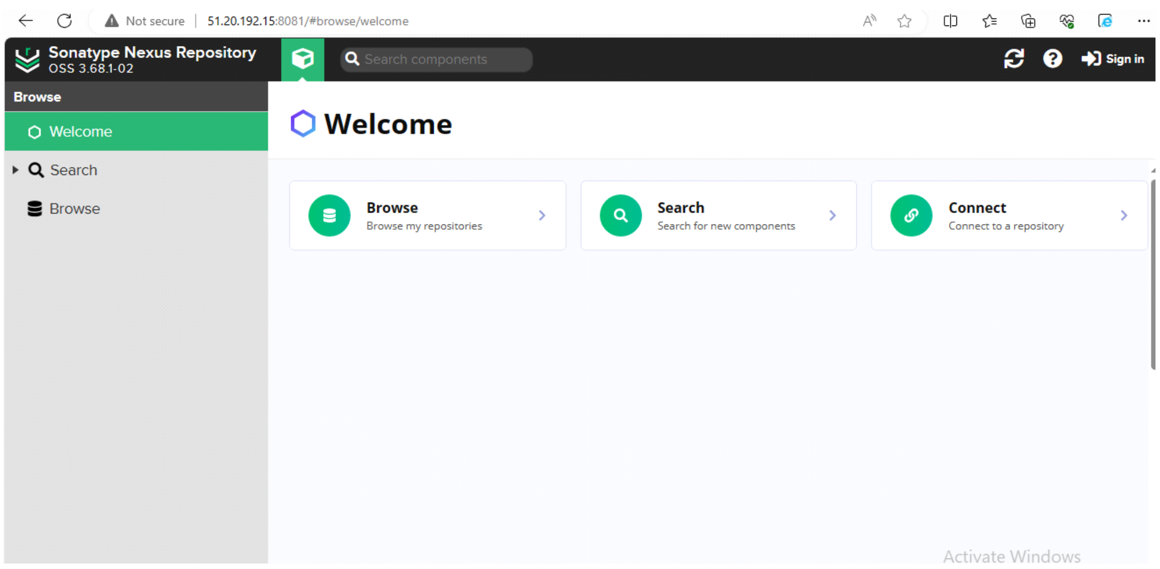
. sudo systemctl start docker

. sudo systemctl enable docker

# Now install nexus continer by run below command

. docker run -d --name nexus -p 8081:8081 sonatype/nexus3:latest

# Access your Nexus by <serverip:8081>



use "admin" as user and forpassword follow below steps.

. Login to you nexus server

. run "docker ps" and get the nexus conatiner id

. run "docker exec -it <neuxus.container_ID> /bin/bash" for go inside the nexus container.

. run "cd sonatype-work/nexus3"

. run "cat admin.password" copy the password and exit

Our infrastructure is ready now.

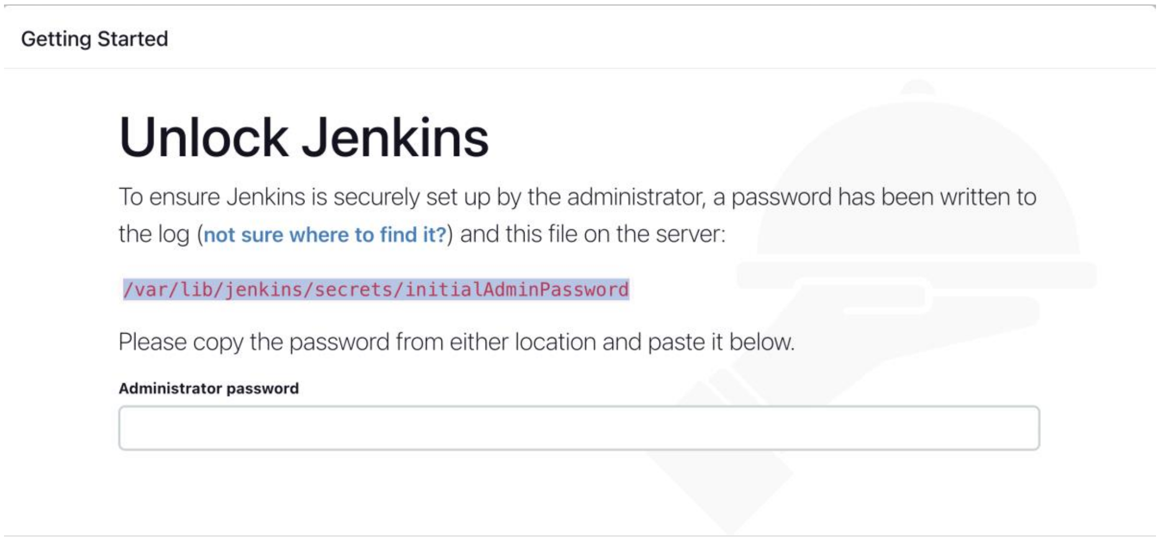
Lets' login on each tool that we have installed

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 jenkin'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


Getting Started

Unlock Jenkins

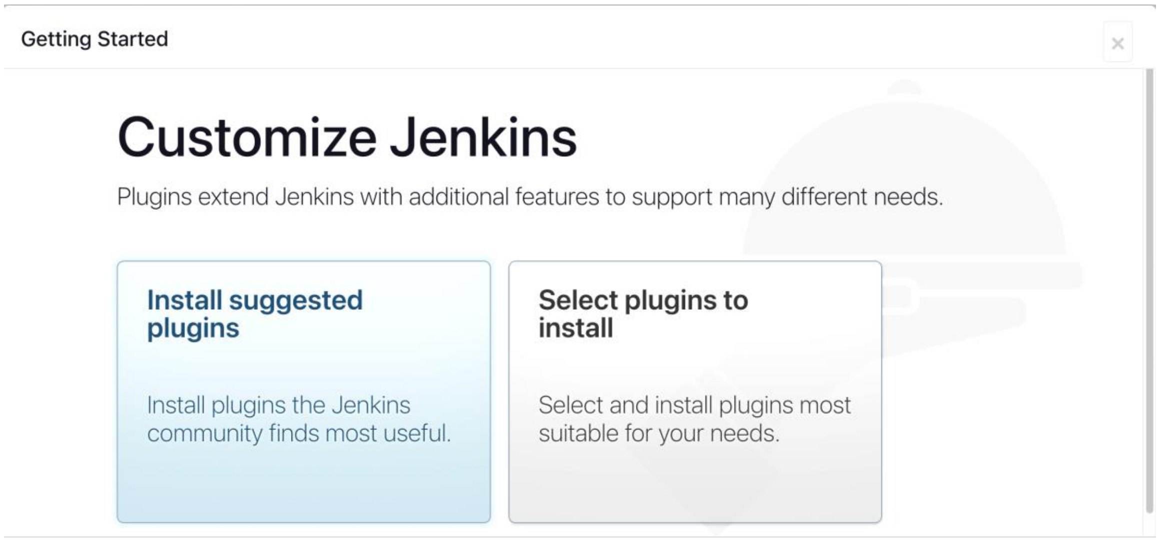
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

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

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

Administrator password

. Click on install suggested plugins

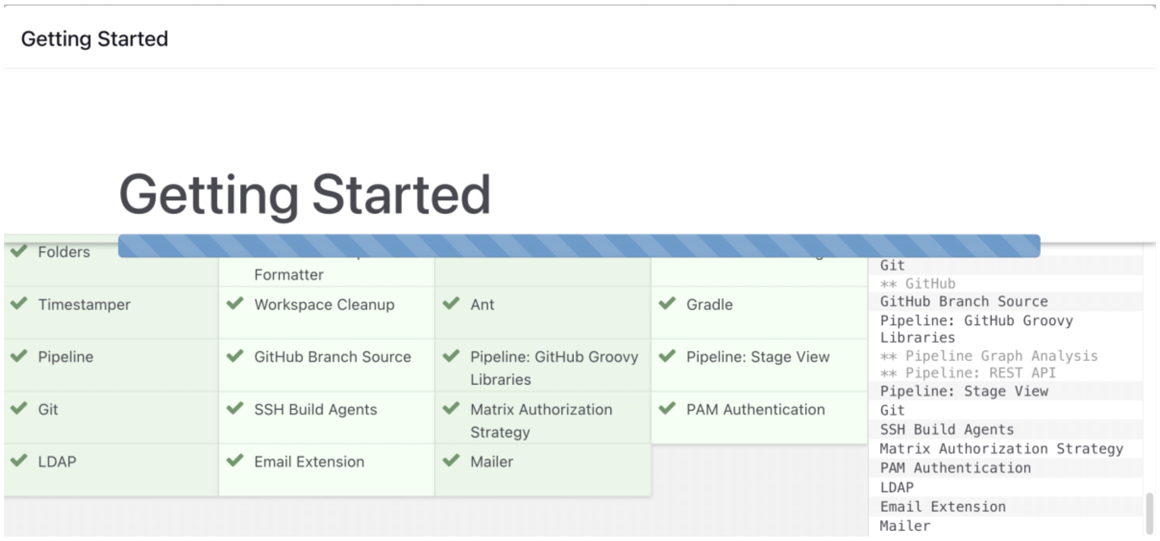

Getting Started

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



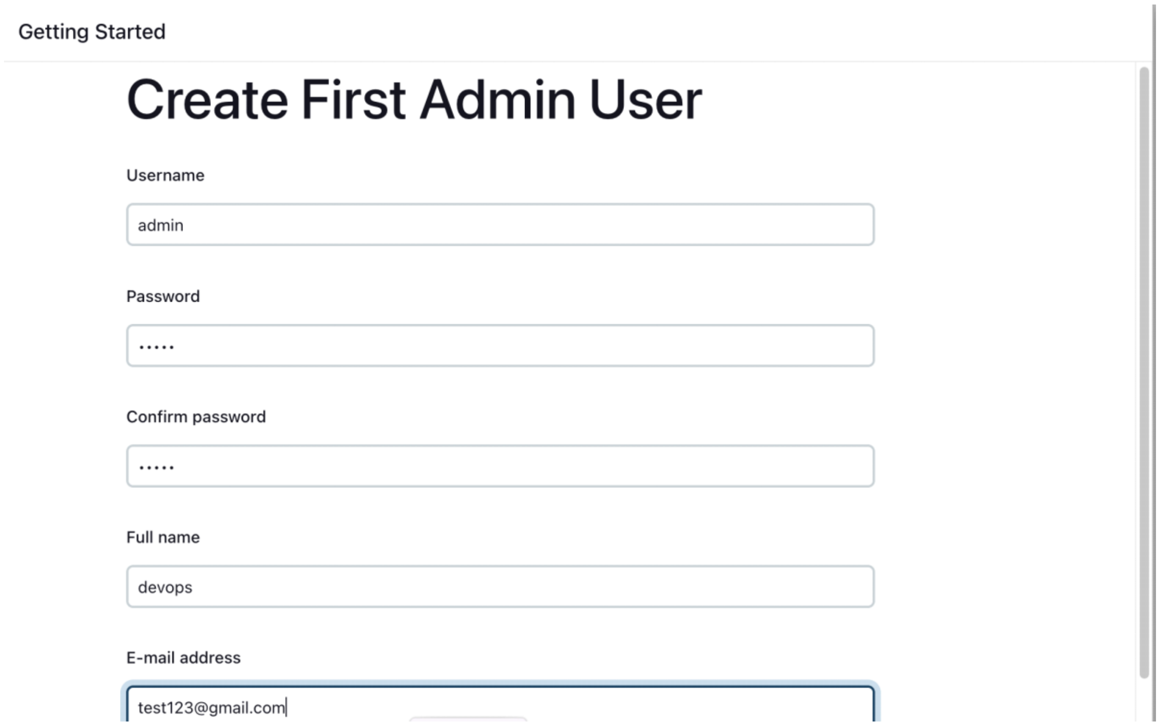
Getting Started

Getting Started

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

```
Git
** GitHub
GitHub Branch Source
Pipeline: GitHub Groovy
Libraries
** 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]

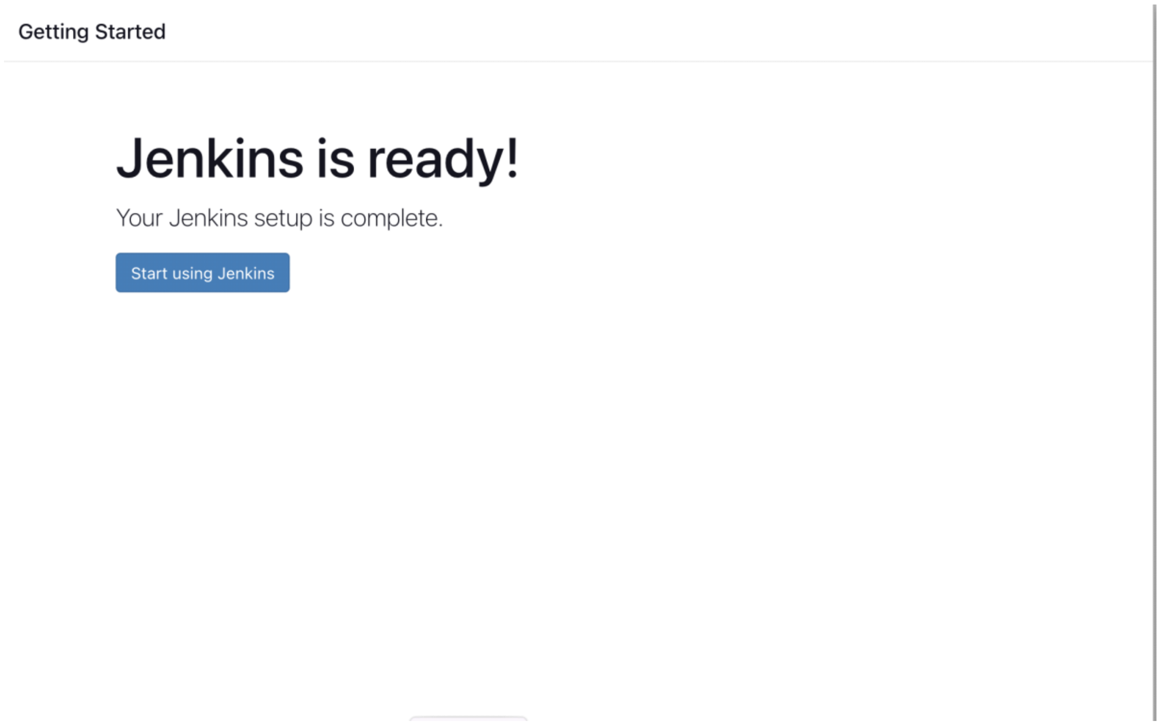


Getting Started

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


Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

# let's follow some more steps.

Go to Manage Jenkins > Manage Plugins.

In the Available tab, search for below plugins and install it.

1. Docker Pipeline and docker plugin,
2. SonarQube Scanner
3. Config file provider plugin
4. Pipeline maven integration plugin
5. Eclipse Temurin installer Plugin

Restart Jenkins after the plugin is installed.

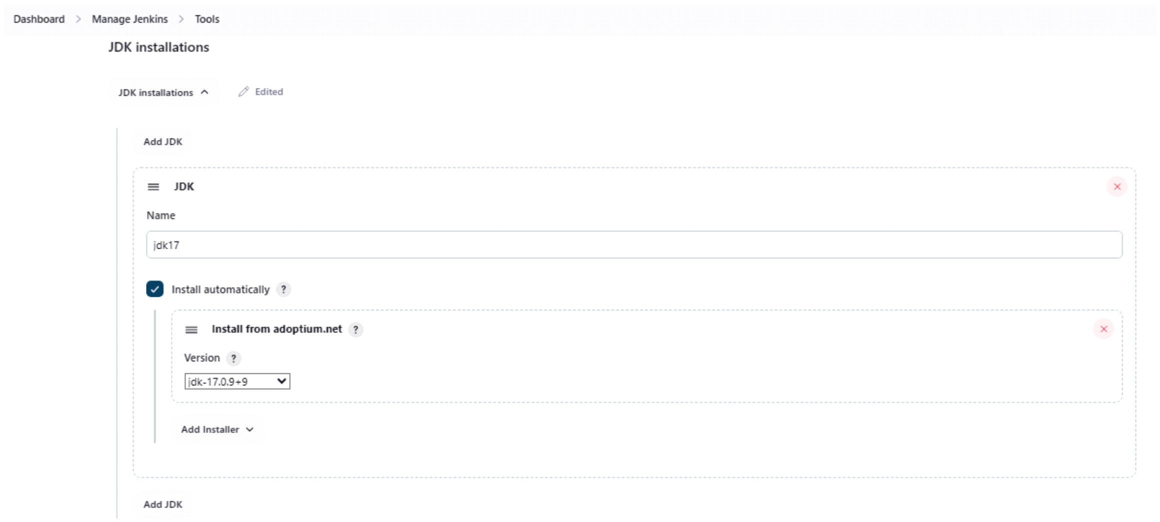
Then go to tools and do below configuration

go down and under the JDK section put below details

name as "jdk17"

version as "jdk 17.0.9+9"

installer as "Install from adoptium.net" and click on Install automatically


Dashboard > Manage Jenkins > Tools

JDK installations

JDK installations — Edited

Add JDK

JDK

Name: jdk17

Install automatically: ✓

Install from adoptium.net

Version: jdk-17.0.9+9

Add Installer

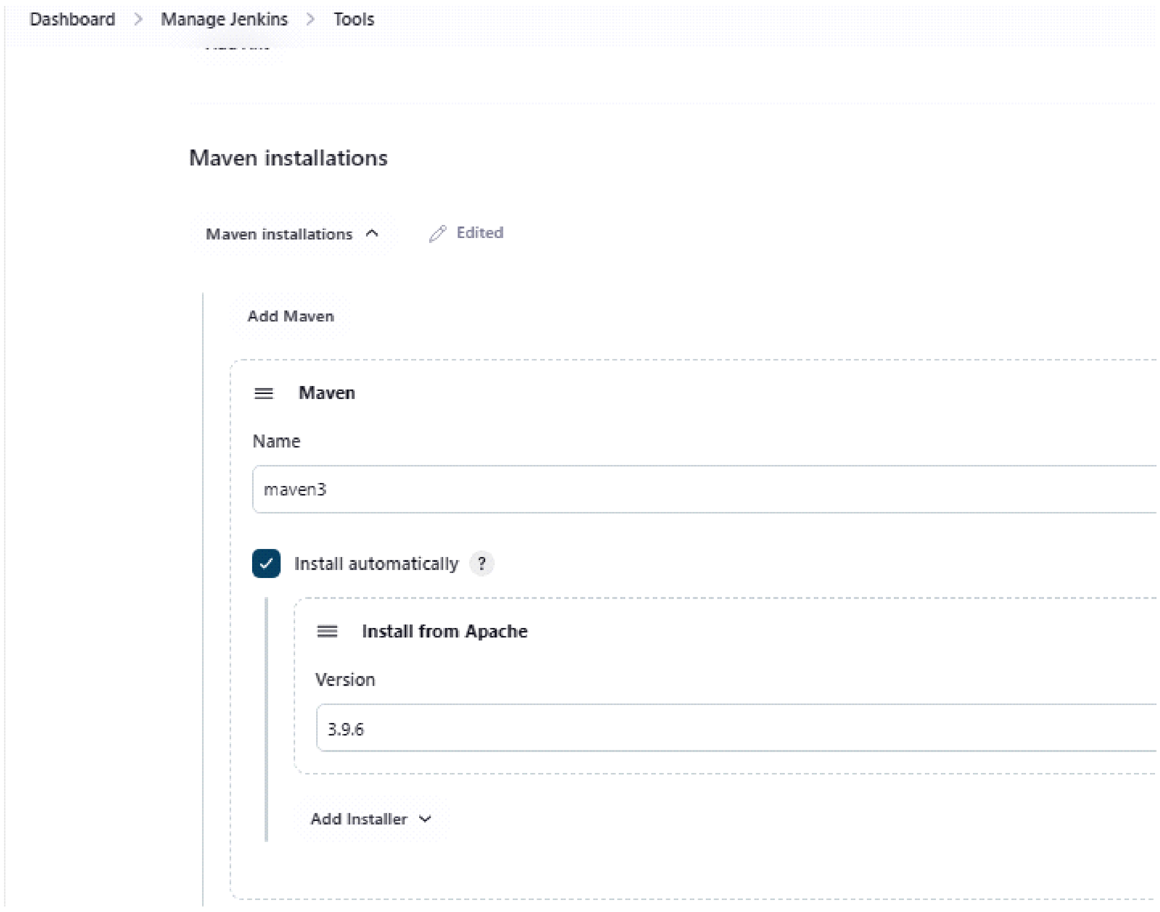
Add JDK

go down and under the Maven section put below details

name as "maven13"

version as "3.9.6"

click on Install automatically


Dashboard > Manage Jenkins > Tools

Maven installations

Maven installations — Edited

Add Maven

Maven

Name: maven3

Install automatically: ✓

Install from Apache

Version: 3.9.6

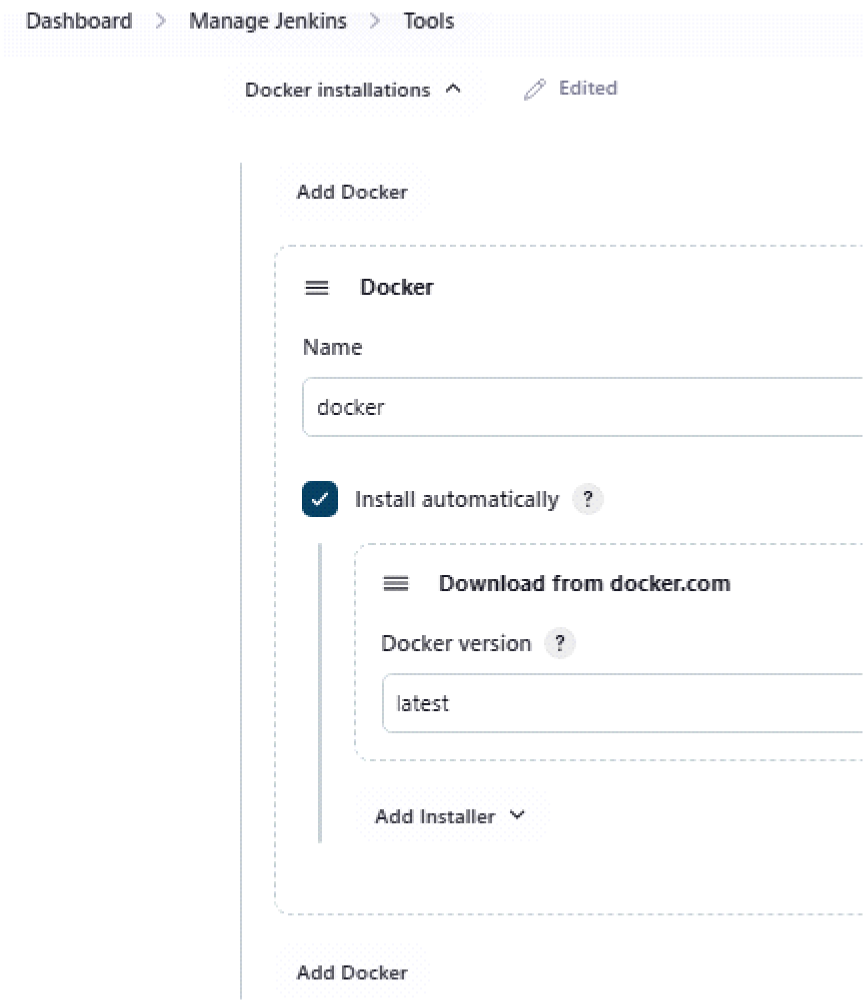
Add Installer

go down and under the docker section put below details

name as "docker"

version as "latest"

click on Install automatically and installer as "docker.com"


Dashboard > Manage Jenkins > Tools

Docker installations ^ Edited

Add Docker

Docker

Name

docker

✓ Install automatically ?

Download from docker.com

Docker version ?

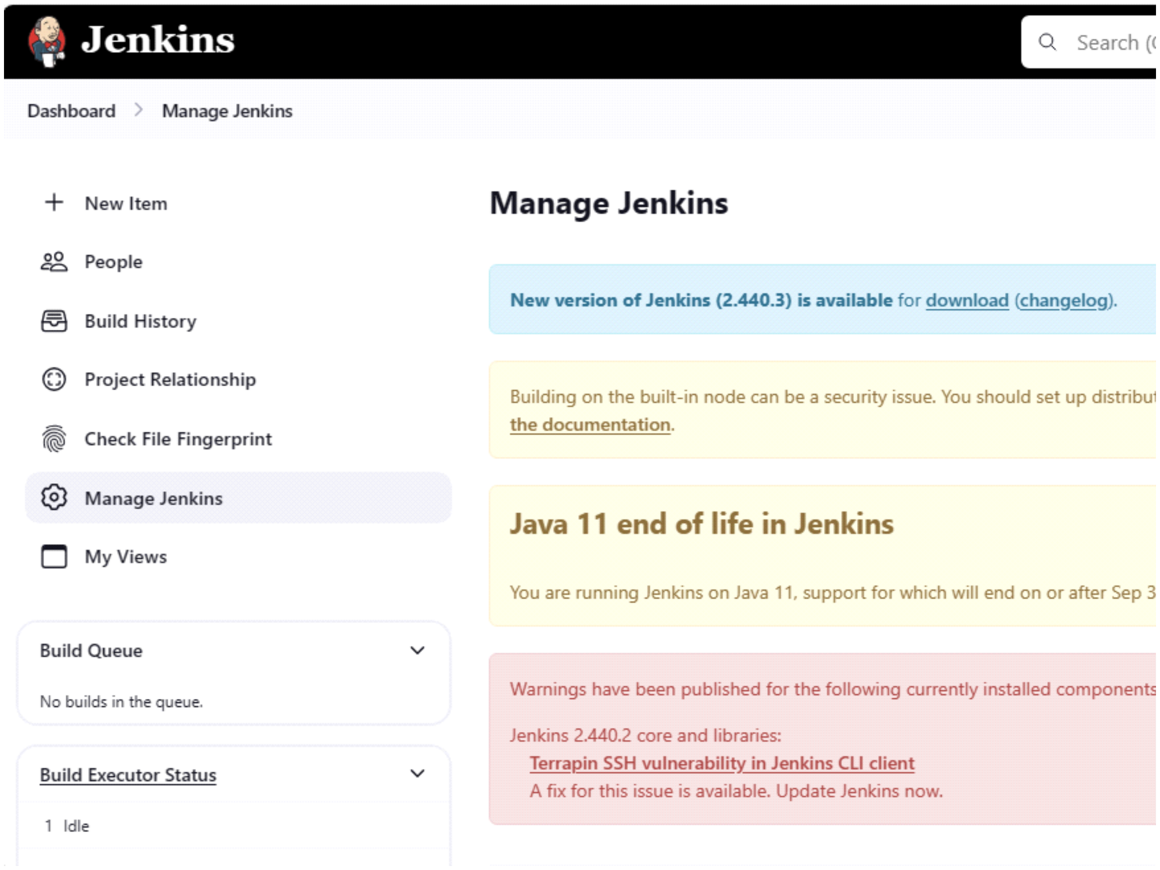
latest

Add Installer

Add Docker

# Go to jekins GUI again....

. Click on manage Jenkins



Manage Jenkins

New version of Jenkins (2.440.3) is available for download (changelog).

Building on the built-in node can be a security issue. You should set up distribut
the documentation.

Java 11 end of life in Jenkins

You are running Jenkins on Java 11, support for which will end on or after Sep 3

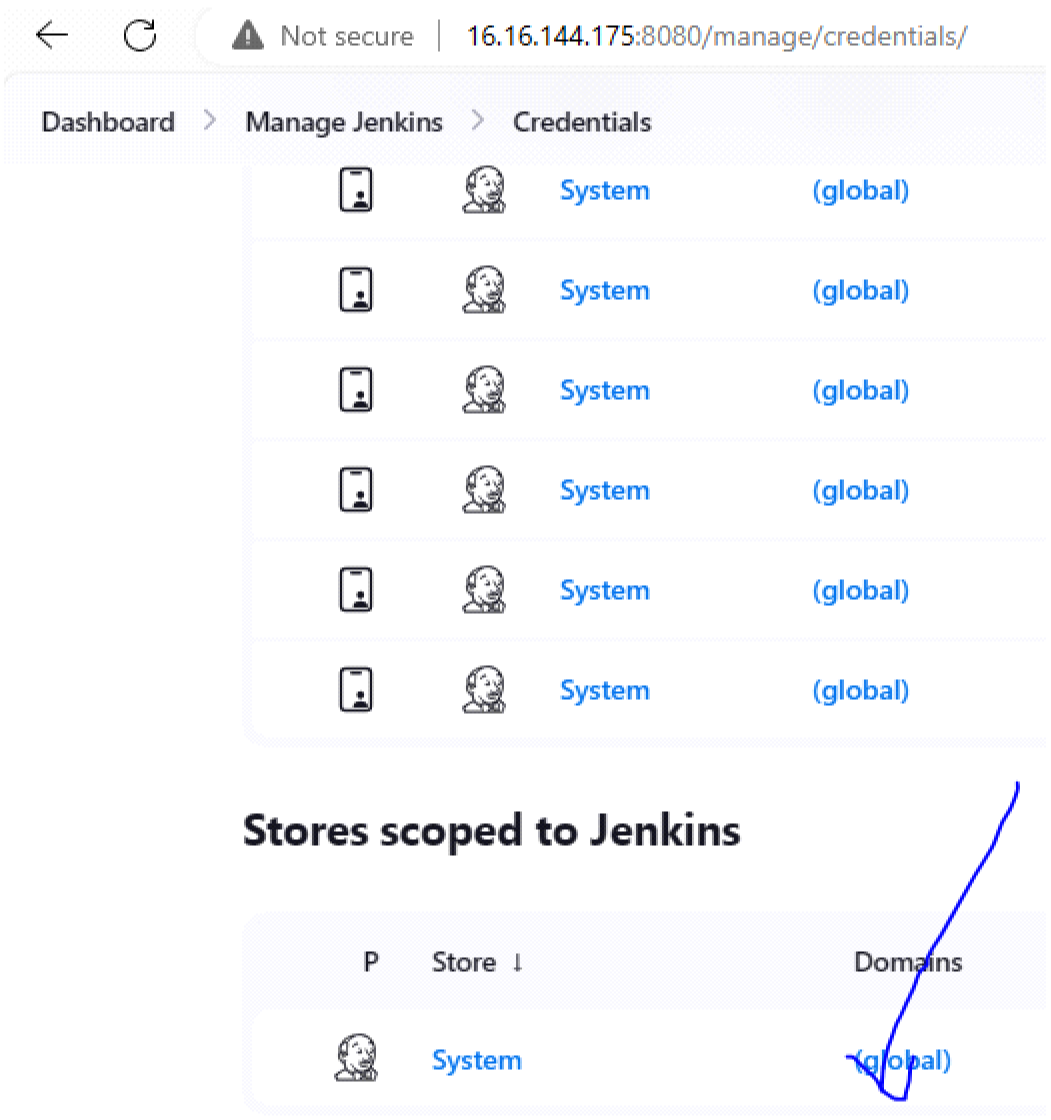
Warnings have been published for the following currently installed components

Jenkins 2.440.2 core and libraries:
Terrapin SSH vulnerability in Jenkins CLI client
A fix for this issue is available. Update Jenkins now.

No builds in the queue.

1 Idle

. Scroll down then click on crendtails then click on global

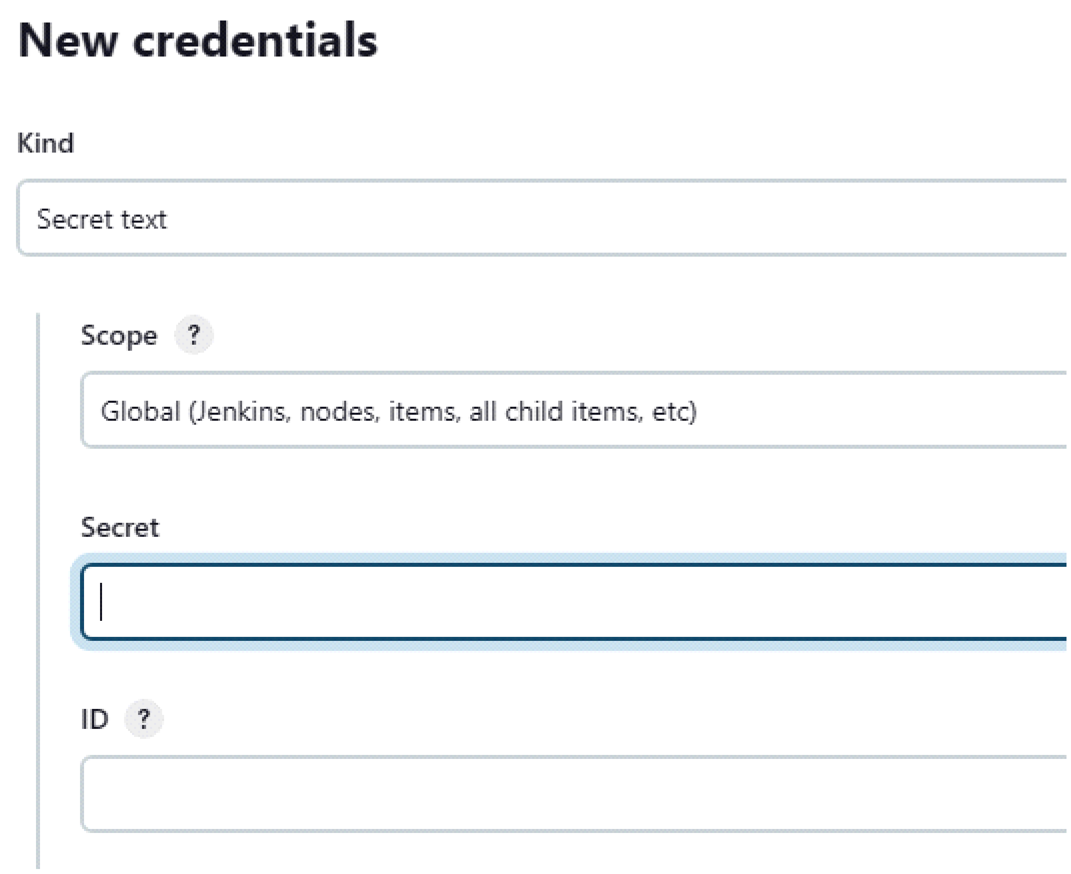


		System	(global)
		System	(global)
		System	(global)
		System	(global)
		System	(global)
		System	(global)

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

. Click on ADD crendentils 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'

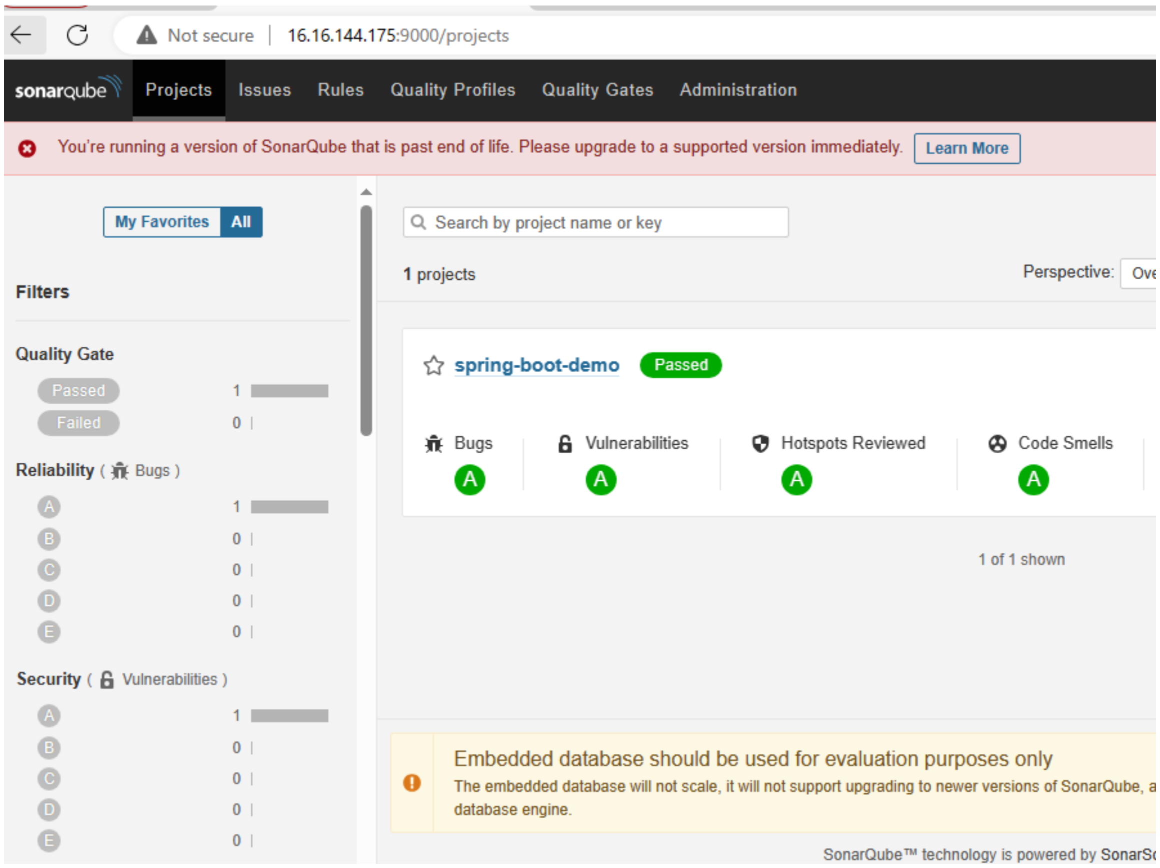
In secret need to put sonarqube secret for this follow below steps

. login to sonar qube

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

example: 12.3.4.2:9000

use user 'admin', password as 'admin'



You're running a version of SonarQube that is past end of life. Please upgrade to a supported version immediately.

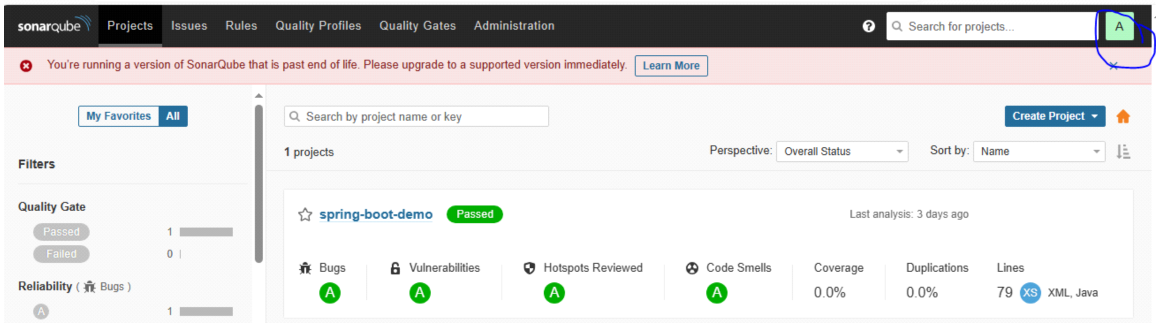
1 projects

Embedded database should be used for evaluation purposes only

The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, a database engine.

SonarQube™ technology is powered by SonarS

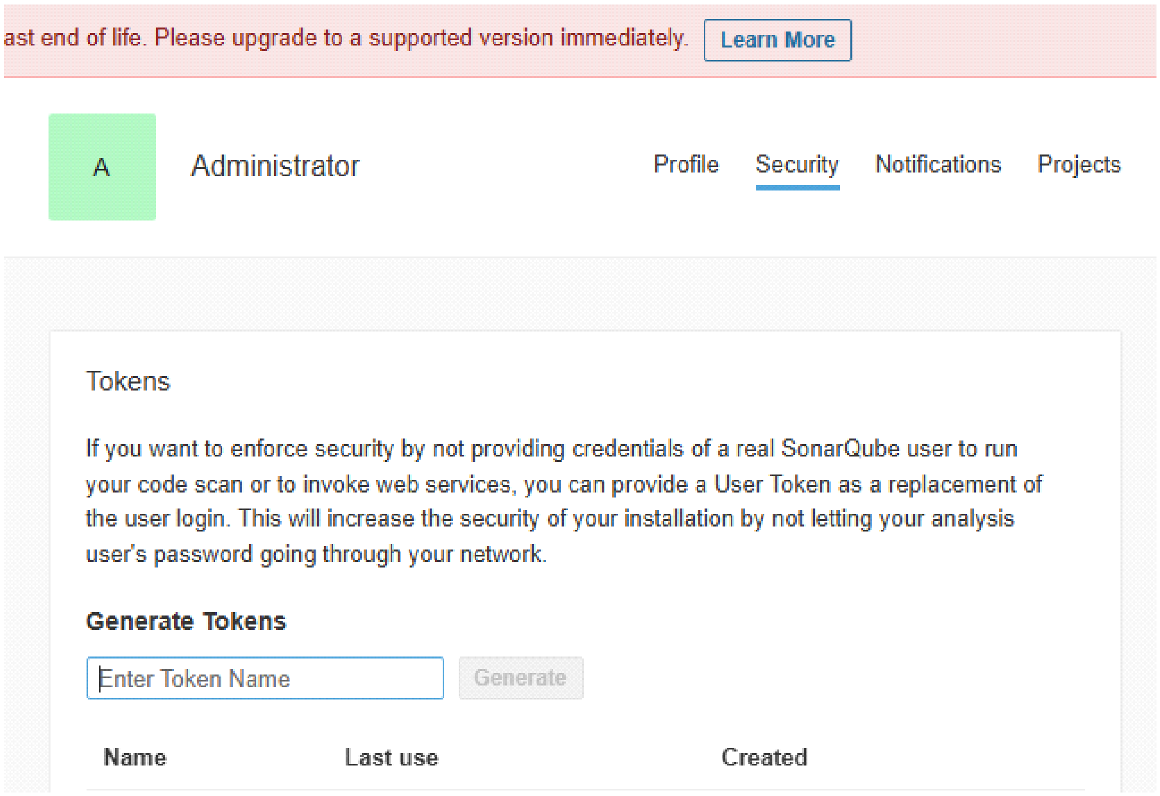
. After login to sonar GUI, Click on 'A' icon present on top right corner on Sonar GUI



You're running a version of SonarQube that is past end of life. Please upgrade to a supported version immediately.

1 projects

. 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

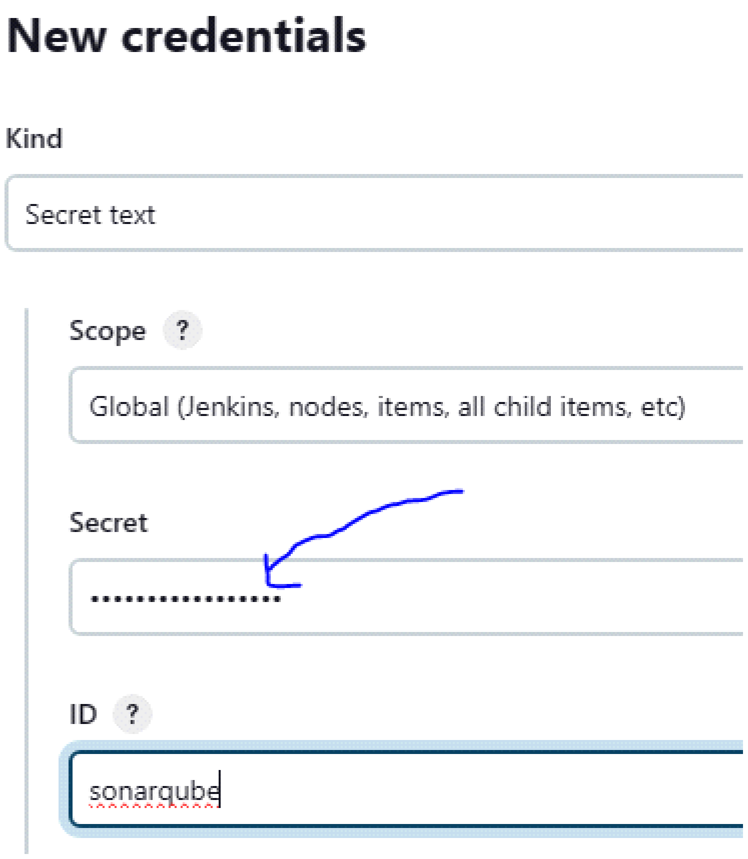
If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going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

Generate Tokens

Enter Token Name | 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

sonarqube

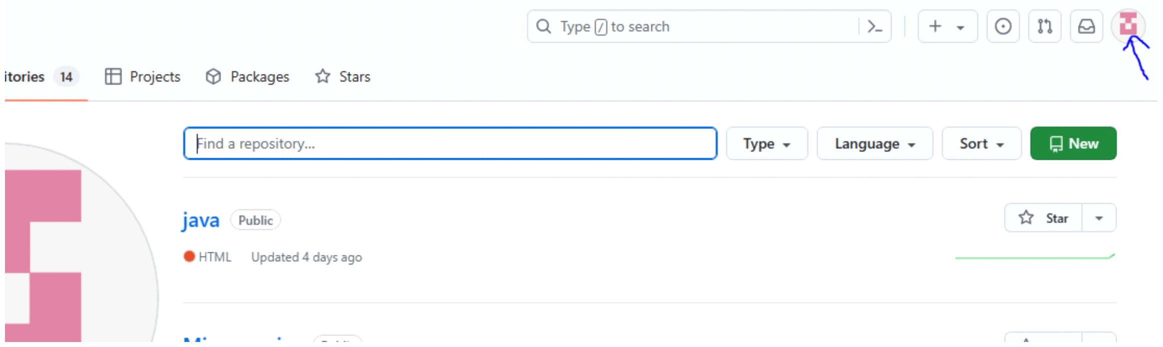
2. let's add Git crendtials, click on add button again.

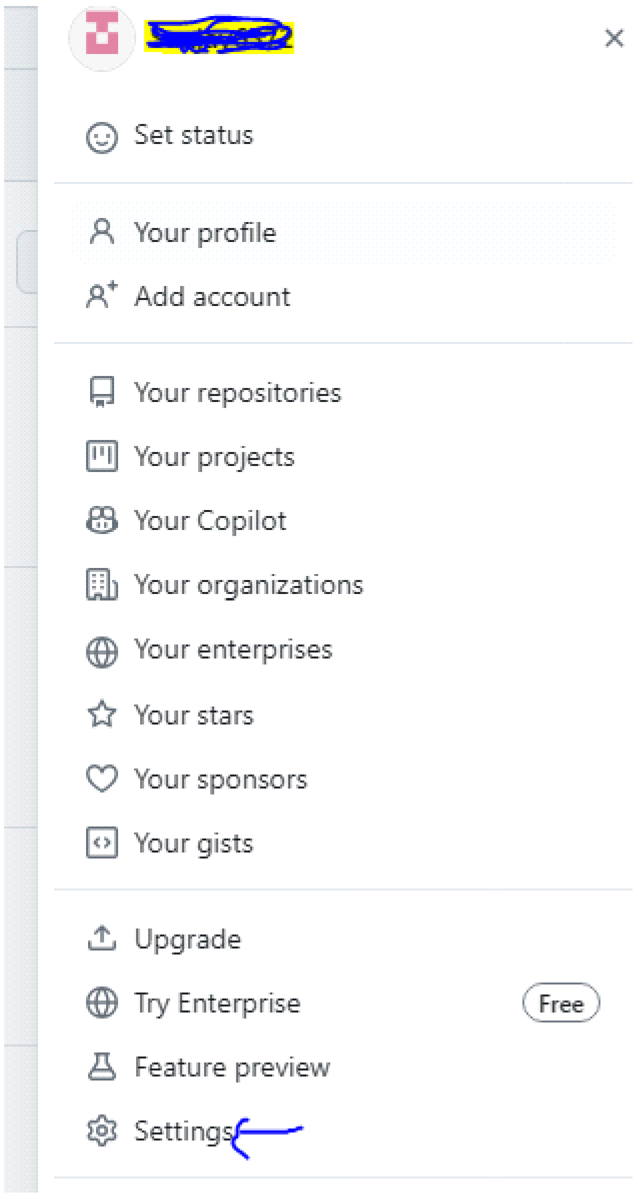
select kind as 'secret text'.

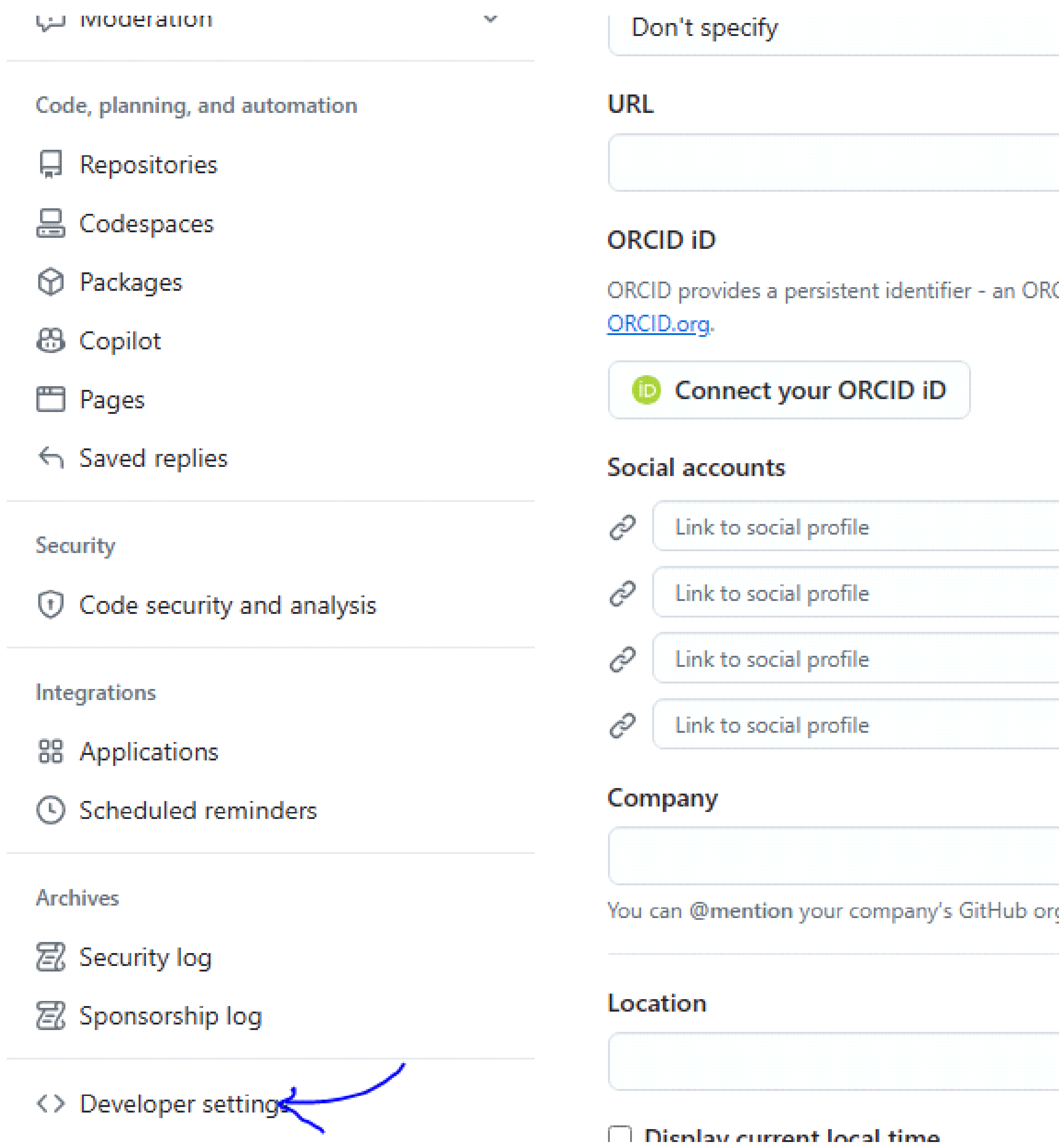
Id as 'sanjugithub'

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 devloper option ---> personal token ---> token classic ---> generate new token ---> give any name, select all permisssions, generate it, copy it and put in secret box of jenkins.




[redacted]
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 ORC
ORCID.org.

Connect your ORCID iD

Social accounts

Link to social profile

Link to social profile

Link to social profile

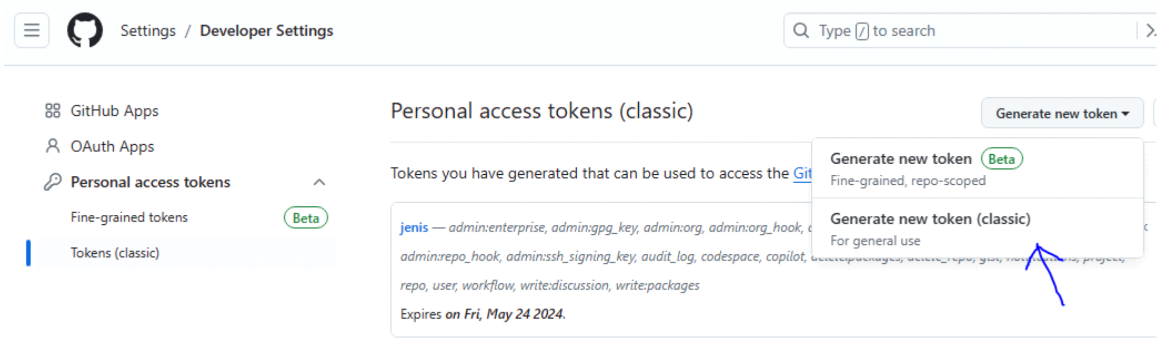
Link to social profile

Company

You can **@mention** your company's GitHub org

Location

Display current local time



Settings / **Developer Settings**

Type / to search

GitHub Apps

OAuth Apps

Personal access tokens

Fine-grained tokens Beta

Tokens (classic)

Personal access tokens (classic)

Generate new token ▾

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

Generate new token Beta
Fine-grained, repo-scoped

Generate new token (classic)
For general use

jenis — *admin:enterprise, admin:gpg_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.*

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

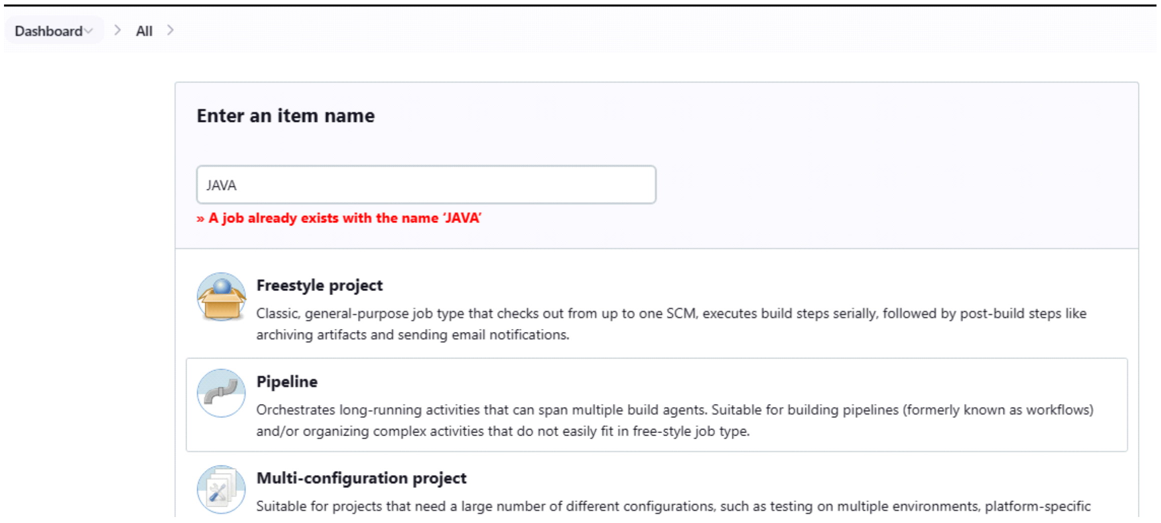
select kind as 'username and pass'. Give you user id and pass of dockerhub

Id as 'docker-cred' then save it.

Use the ids name same as i used

Let's start building CI pipline

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



Enter an item name

JAVA

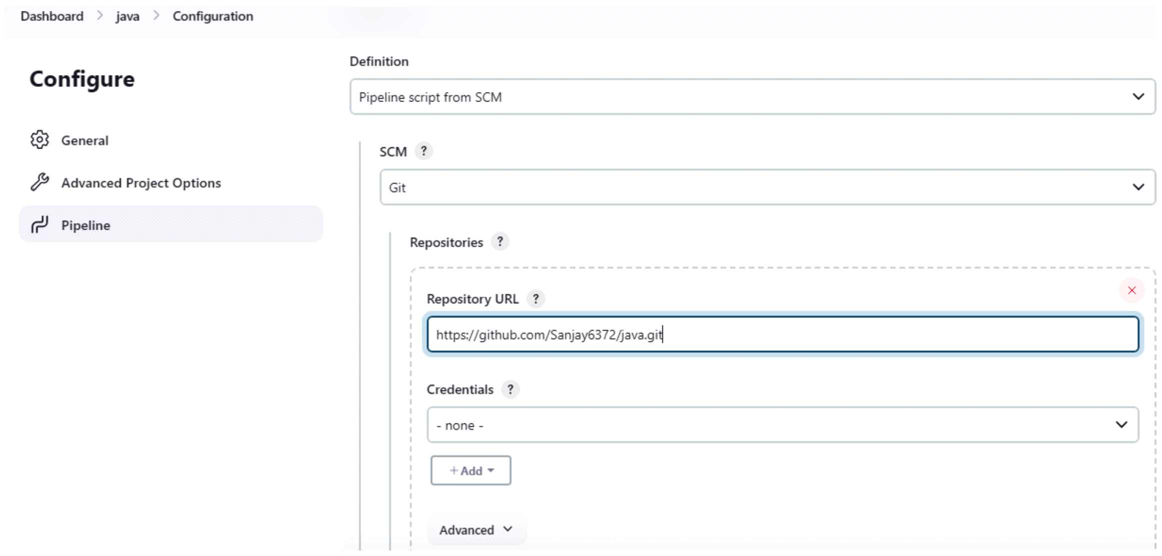
» A job already exists with the name 'JAVA'

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 jenkin 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: https://github.com/Sanjay6372/java.git

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

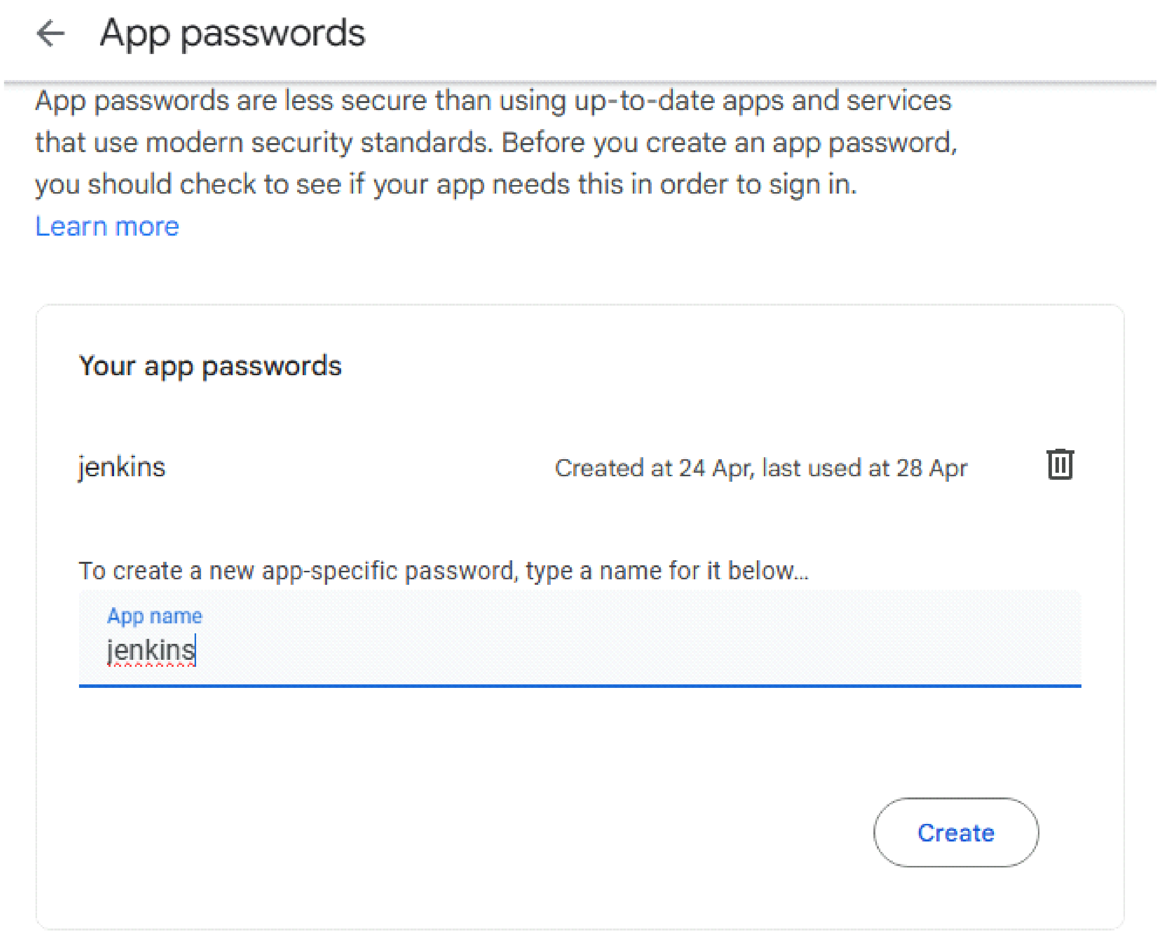
In E-mail Notification

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

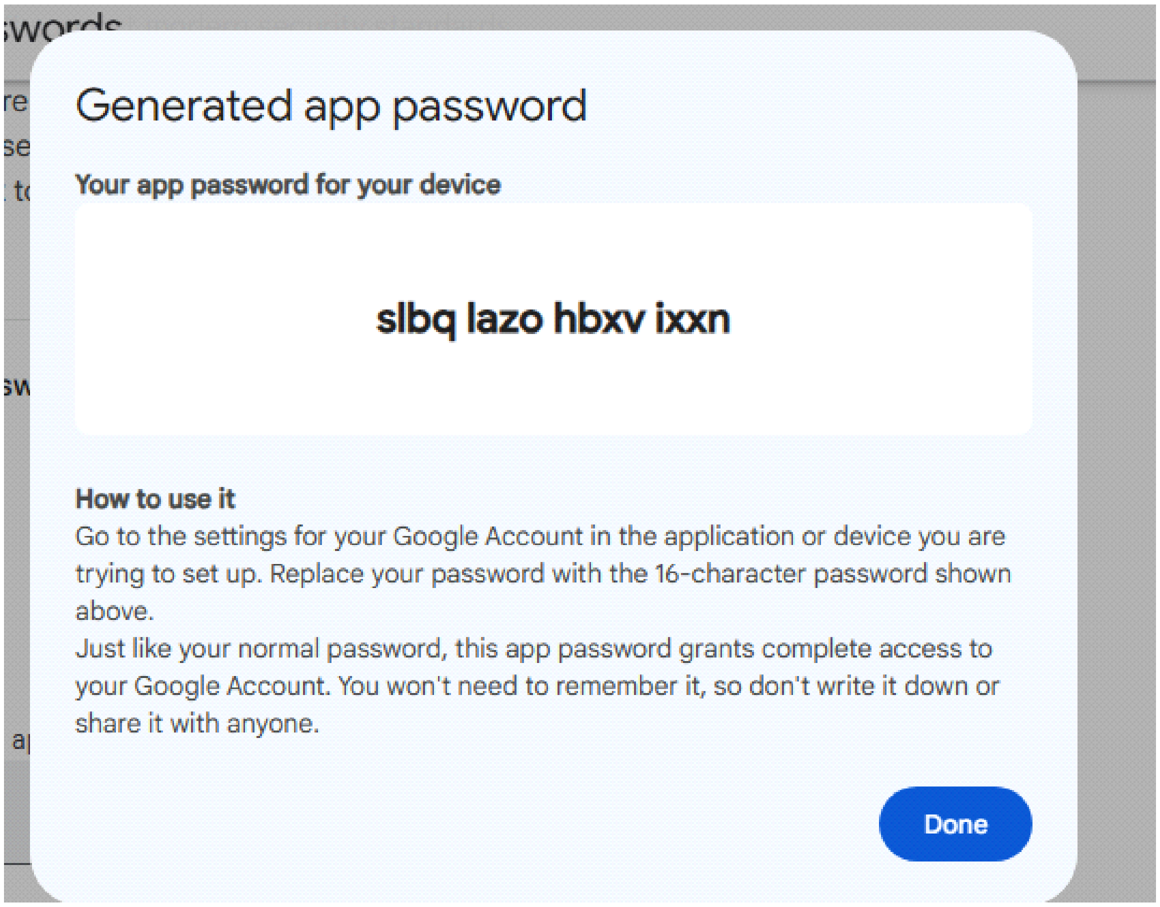
App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.
Learn more

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



Generated app password

Your app password for your device

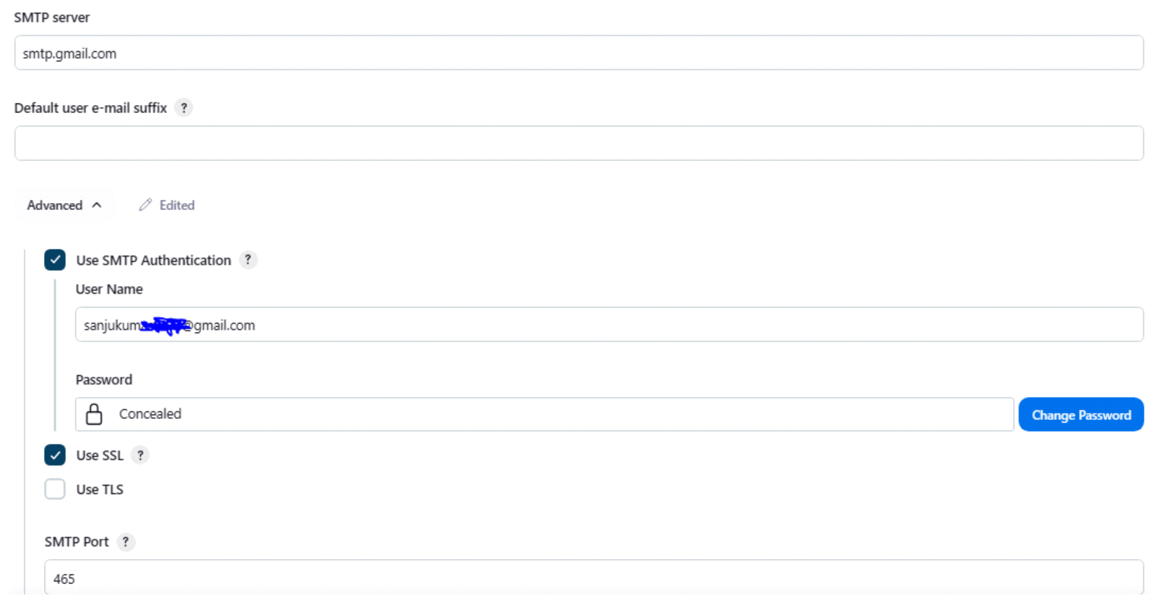
slbq lazo hbxv ixxn

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above.

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

Put info like below



SMTP server: smtp.gmail.com

Default user e-mail suffix:

[✓] Use SMTP Authentication

User Name: sanjukum[redacted]@gmail.com

Password: Concealed

[✓] Use SSL

[ ] Use TLS

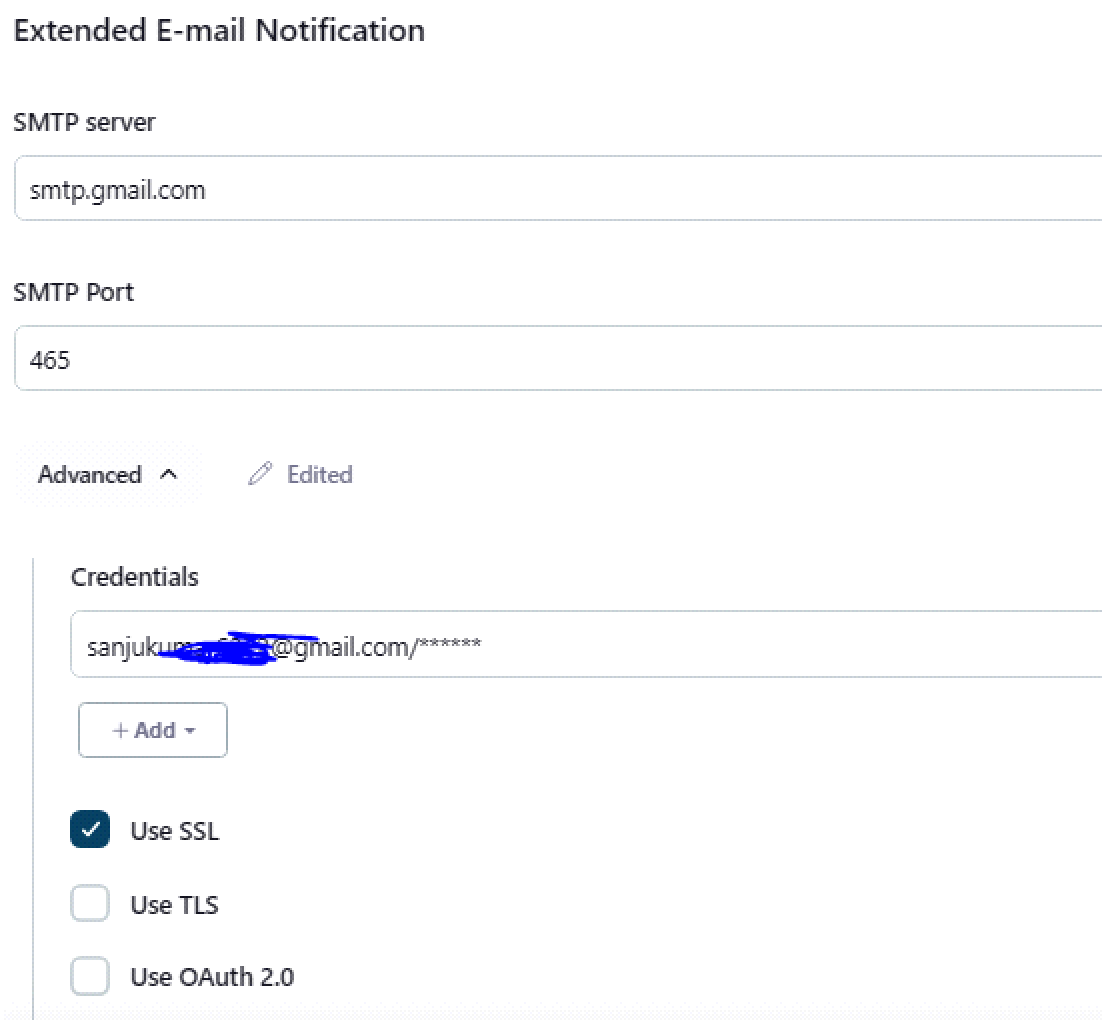
SMTP Port: 465

Extended E-mail Notification section

SMTP server = smtp.gmail.com

SMTP Port = 465

In credentails click on add then create crendentails 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

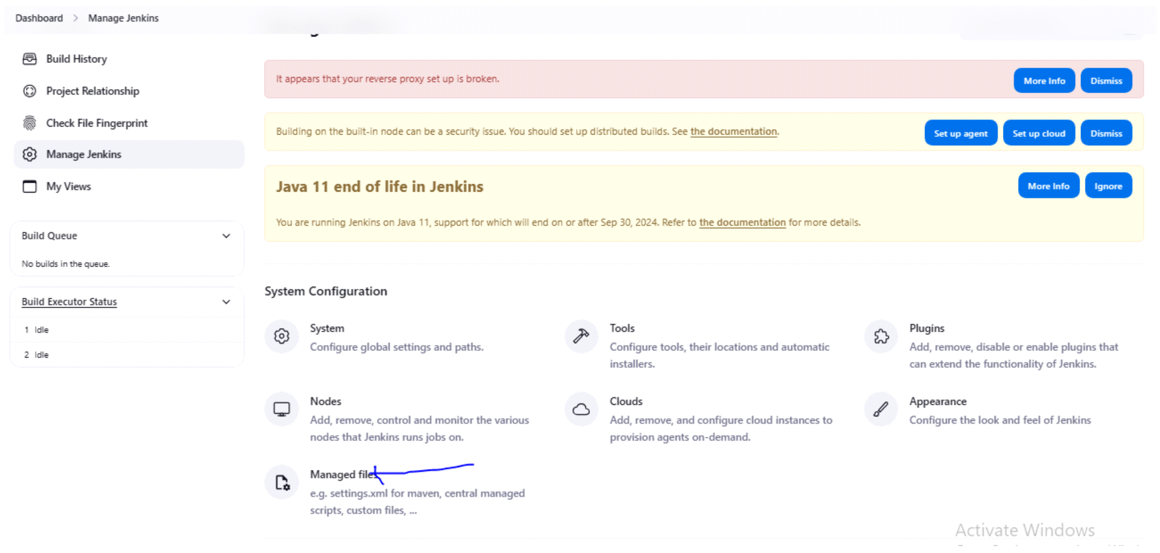
Credentials: sanjuku[redacted]@gmail.com/******

- [✓] Use SSL
- [ ] Use TLS
- [ ] Use OAuth 2.0

After that click on save button

Let's connect Nexus with jenkins

Go back to dashboard then go to managed files and click on create new config


Dashboard > Manage Jenkins

No builds in the queue.

1 Idle

2 Idle

It appears that your reverse proxy set up is broken.

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

Java 11 end of life in Jenkins

You are running Jenkins on Java 11, support for which will end on or after Sep 30, 2024. Refer to the documentation for more details.

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

Appearance
Configure the look and feel of Jenkins

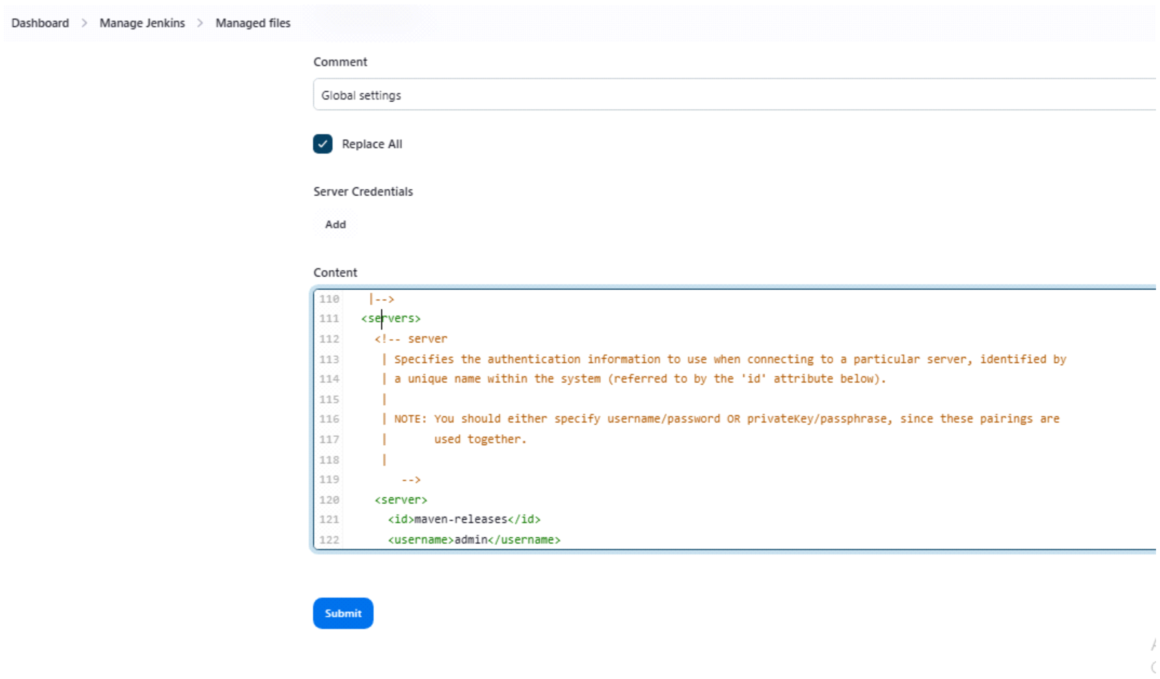
Managed files
e.g. settings.xml for maven, central managed scripts, custom files, ...

Activate Windows

Select the Global Maven settings.xml, give id as "global-settings" and click on next

Then in the context section go down and under the server section put below code

```
<server>
      <id>maven-releases</id>
      <username>admin</username>      # change admin with your nexus user name
      <password>sanjay</password>       # change sanjay with your nexus password
    </server>
    <server>
      <id>maven-snapshots</id>
      <username>admin</username>     # change admin with your nexus user name
      <password>sanjay</password>      # change sanjay with your nexus password
    </server>
```



Comment

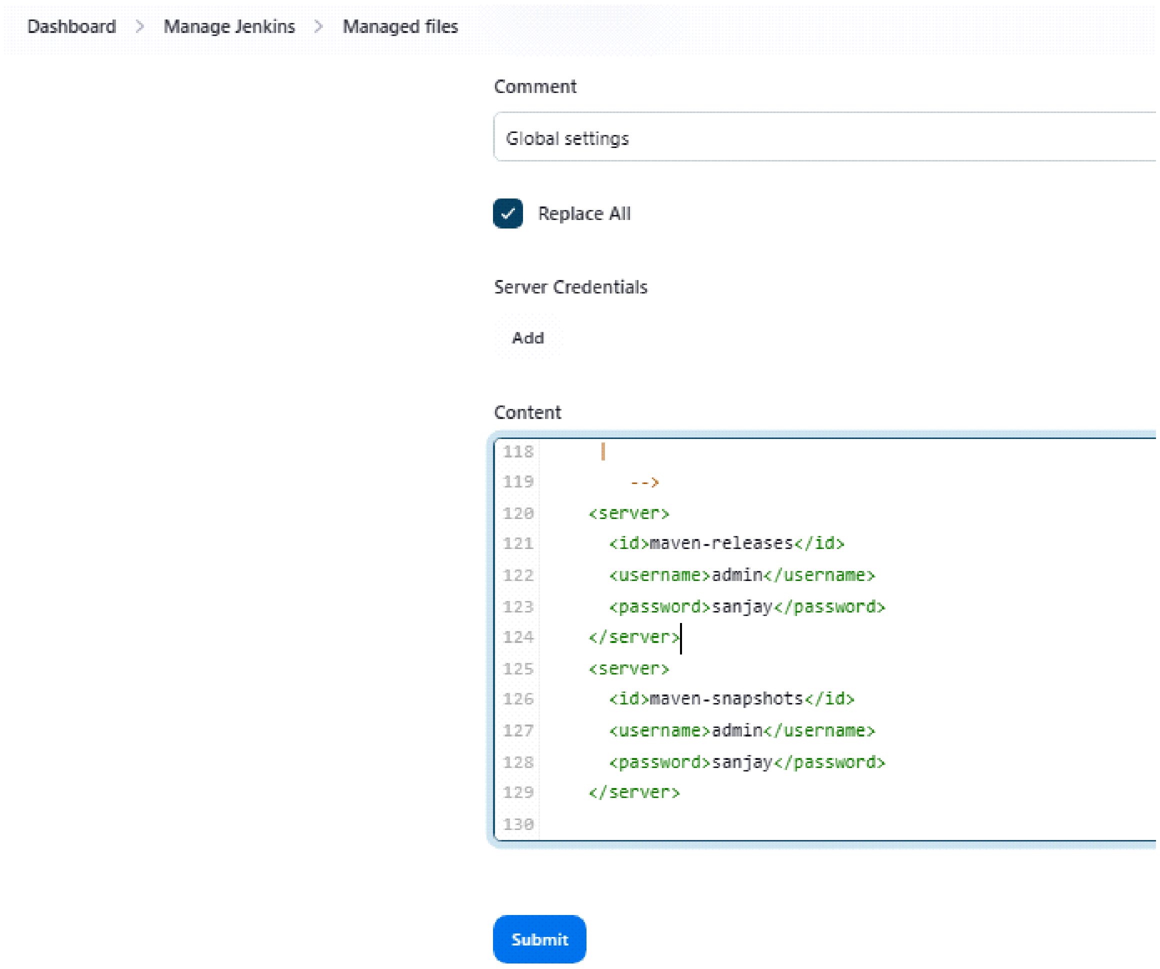
Global settings

✓ Replace All

Server Credentials

Content

```
  |-->
<servers>
  <!-- server
   | Specifies the authentication information to use when connecting to a particular server, identified by
   | a unique name within the system (referred to by the 'id' attribute below).
   |
   | NOTE: You should either specify username/password OR privateKey/passphrase, since these pairings are
   |       used together.
   |
    -->
  <server>
    <id>maven-releases</id>
    <username>admin</username>
```



Comment: Global settings

Replace All [✓]

Server Credentials

Content

```
        -->
    <server>
      <id>maven-releases</id>
      <username>admin</username>
      <password>sanjay</password>
    </server>
    <server>
      <id>maven-snapshots</id>
      <username>admin</username>
      <password>sanjay</password>
    </server>
```

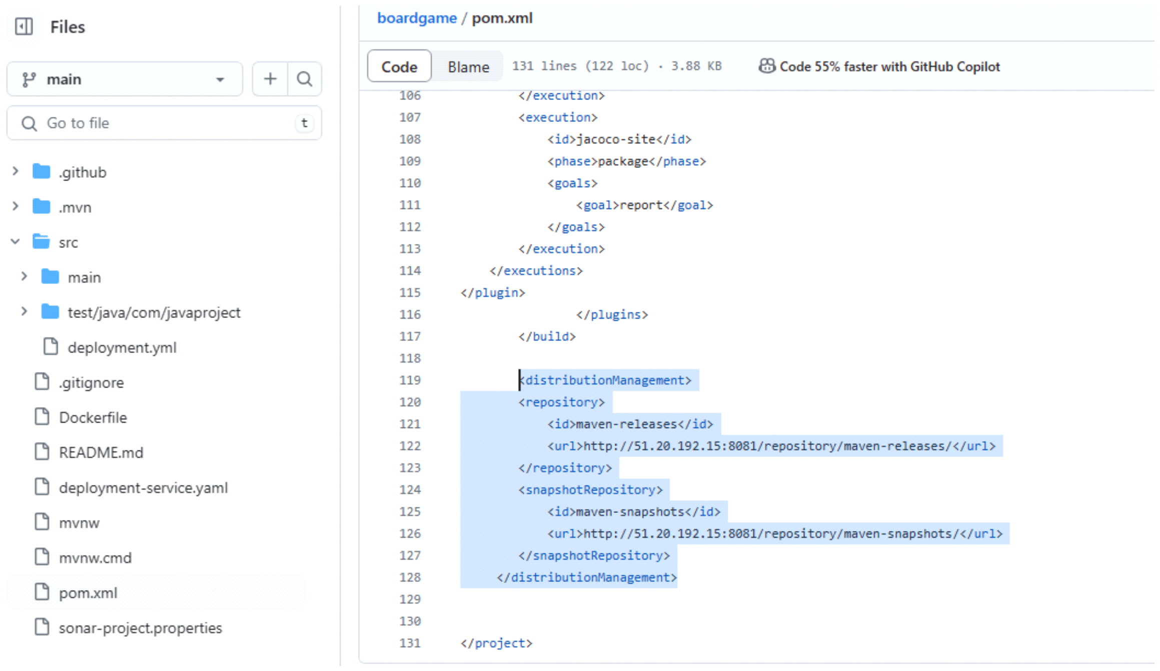
then click on submit

Now go to git repo then go to pom.yml file and add below code

```
<distributionManagement>
            <repository>
                  <id>maven-releases</id>
                  <url>http://51.20.192.15:8081/repository/maven-releases/</url>
```
# change ip with nexus server ip

```
            </repository>
            <snapshotRepository>
                  <id>maven-snapshots</id>
```

```
                    <url>http://51.20.192.15:8081/repository/maven-snapshots/</url>     # change ip with
nexus server ip
                </snapshotRepository>
            </distributionManagement>
```



boardgame / pom.xml

131 lines (122 loc) · 3.88 KB

```
            </execution>
            <execution>
                <id>jacoco-site</id>
                <phase>package</phase>
                <goals>
                    <goal>report</goal>
                </goals>
            </execution>
        </executions>
    </plugin>
                </plugins>
            </build>

            <distributionManagement>
            <repository>
                <id>maven-releases</id>
                <url>http://51.20.192.15:8081/repository/maven-releases/</url>
            </repository>
            <snapshotRepository>
                <id>maven-snapshots</id>
                <url>http://51.20.192.15:8081/repository/maven-snapshots/</url>
            </snapshotRepository>
        </distributionManagement>


    </project>
```

Save the git file...Our nexus setup is completed

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

```
pipeline {
    agent any

    tools {
        jdk 'jdk17'
        maven 'maven3'
    }
    environment {
        DOCKER_IMAGE = "sanjay9888/boardgame:${BUILD_NUMBER}"     # change sanjay9888 with
```

your docker hub user name

```
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: 'main', url: 'https://github.com/Sanjay6372/boardgame.git'     # change
link with your repolink
            }
        }
        stage('Compile and Test with Maven') {
            steps {
                sh 'mvn compile && mvn test'
            }
        }
        stage('File System Scan with Trivy') {
            steps {
                sh "trivy fs --format table -o trivy-fs-report.html ."
            }
        }
        stage('Static Code Analysis with SonarQube') {
            environment {
                SONAR_URL = "http://51.20.187.155:9000"       #change ip with you sonar ip
            }
            steps {
                withCredentials([string(credentialsId: 'sonarqube', variable:
'SONAR_AUTH_TOKEN')]) {
                    sh 'mvn sonar:sonar -Dsonar.login=$SONAR_AUTH_TOKEN
-Dsonar.host.url=${SONAR_URL}'
```

```
                }
            }
        }
        stage('Build artifacts with Maven') {
            steps {
                sh "mvn package"
            }
        }
        stage('Publish artifacts To Nexus') {
            steps {
                withMaven(globalMavenSettingsConfig: 'global-settings', jdk: 'jdk17', maven:
'maven3', mavenSettingsConfig: '', traceability: true) {
                    sh 'mvn clean deploy -X'
                }
            }
        }
        stage('Build & Tag Docker Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred') {
                        sh "docker build -t ${DOCKER_IMAGE} ."
                    }
                }
            }
        }
        stage('Docker Image Scan') {
            steps {
                sh "trivy image --format table -o trivy-image-report.html ${DOCKER_IMAGE}"
            }
        }
        stage('Push Docker Image') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker-cred') {
                        sh "docker push ${DOCKER_IMAGE}"
                    }
                }
            }
        }
        stage('Update Deployment File') {
            environment {
                GIT_REPO_NAME = "boardgame"     # change java with your git repo name where
code is it.
                GIT_USER_NAME = "Sanjay6372"     # change sanjay6372 with your git repo usr
name.
            }
            steps {
                withCredentials([string(credentialsId: 'sanjugithub', variable: 'GITHUB_TOKEN')]) {
                    sh '''
                        git config --global user.email "sanjay@gmail.com"
                        git config --global user.name "sanjay"
                        sed -i "s/18/${BUILD_NUMBER}/g" src/deployment.yml
                        git add src/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:main
                    '''
                }
            }
        }

    }
    post {
        always {
            emailext (
                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: 'sanjukumar6372@gmail.com',       # change sanjukumar62@gmail.com with
your email id
                from: 'jenkins@example.com',
                replyTo: 'jenkins@example.com',
                mimeType: 'text/html',
                attachmentsPattern: 'trivy-image-report.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 ann follow the below step.

1. click on application

2. click on new app

 enter name as 'java'

 project as 'default'

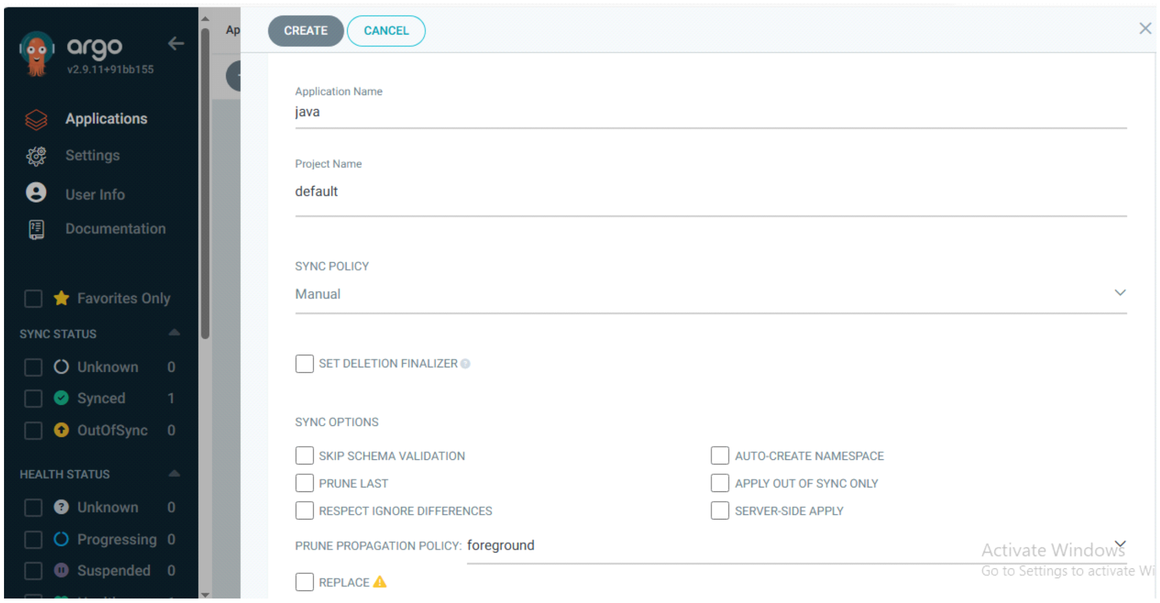
 in source put your code repo link

 in path put 'src'

 in destination section

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

 namespace as 'default' and click on create



Application Name
java

Project Name
default

SYNC POLICY
Manual

SET DELETION FINALIZER

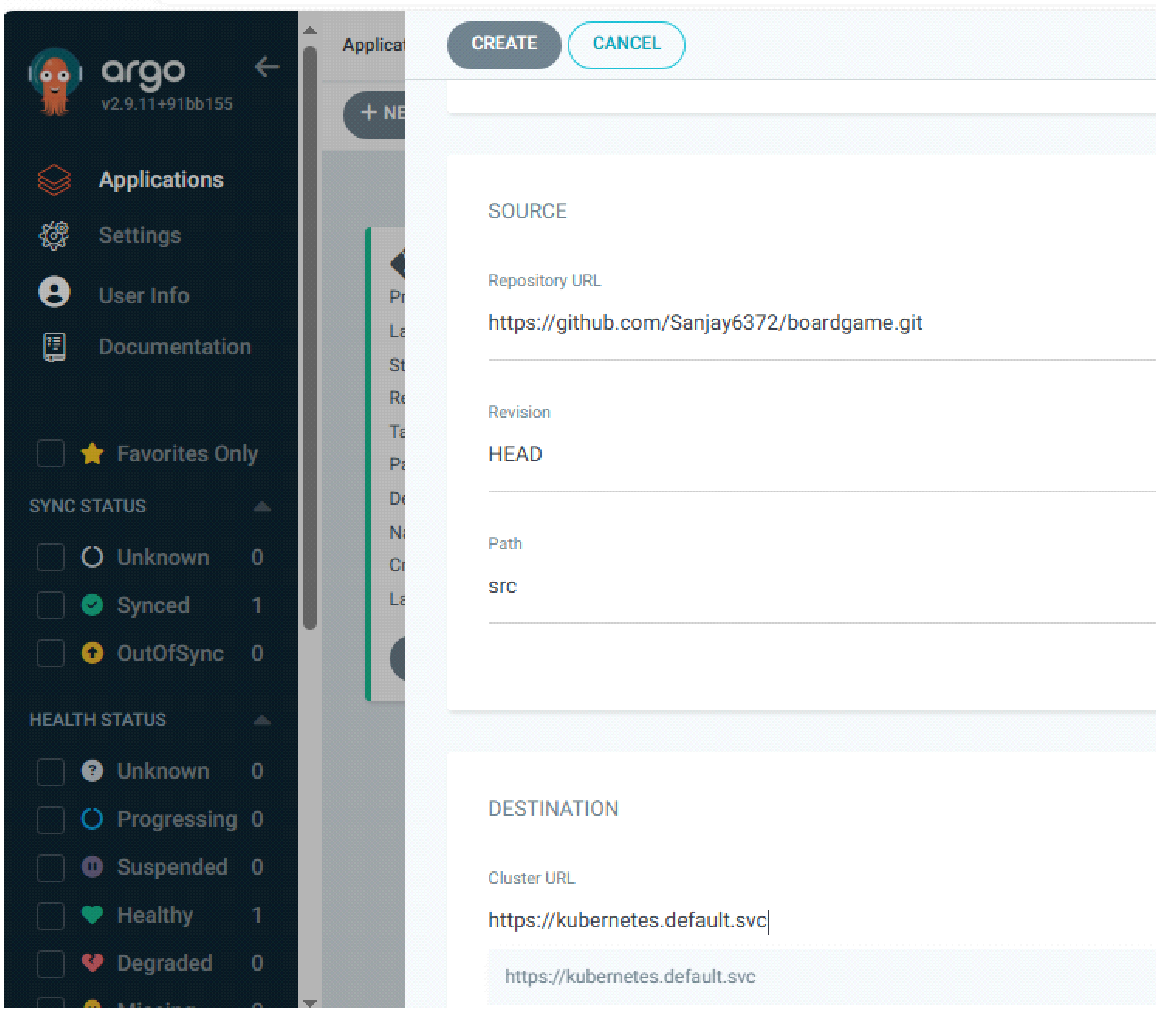
SYNC OPTIONS

SKIP SCHEMA VALIDATION	AUTO-CREATE NAMESPACE
PRUNE LAST	APPLY OUT OF SYNC ONLY
RESPECT IGNORE DIFFERENCES	SERVER-SIDE APPLY

PRUNE PROPAGATION POLICY: foreground

REPLACE

Activate Windows
Go to Settings to activate Wi



SOURCE

Repository URL
https://github.com/Sanjay6372/boardgame.git

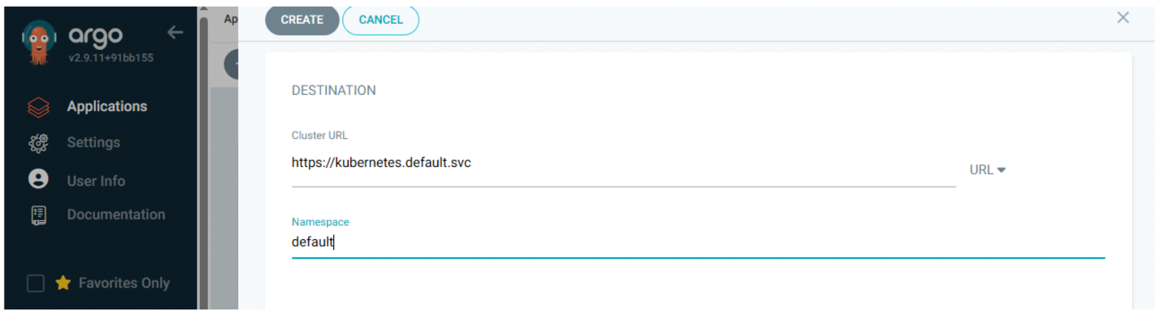
Revision
HEAD

Path
src

DESTINATION

Cluster URL
https://kubernetes.default.svc

https://kubernetes.default.svc



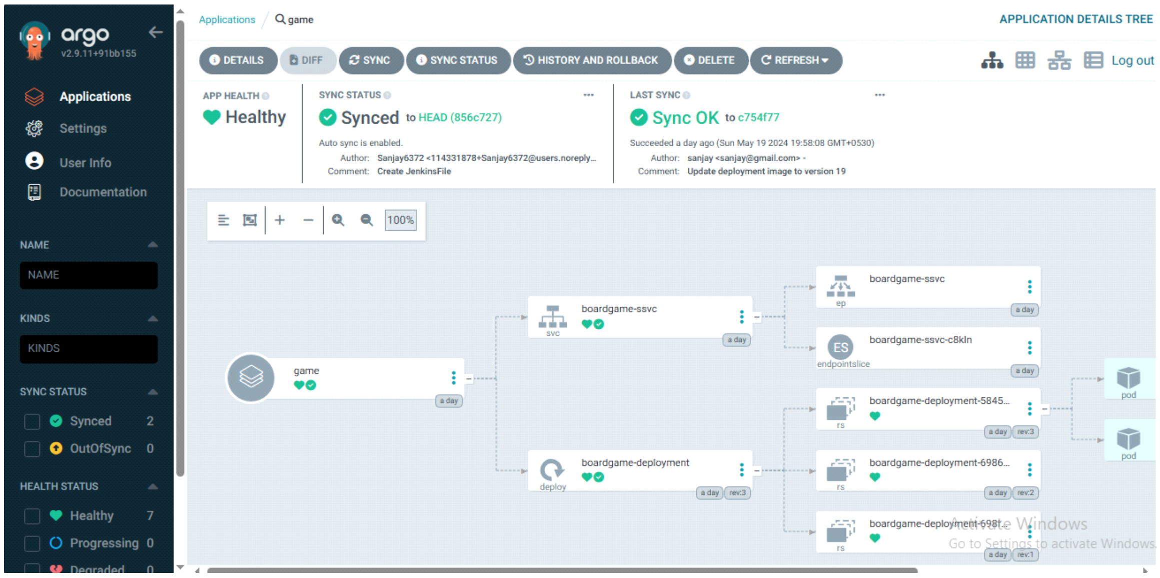
DESTINATION

Cluster URL
https://kubernetes.default.svc

Namespace
default

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

After complition it will look like below

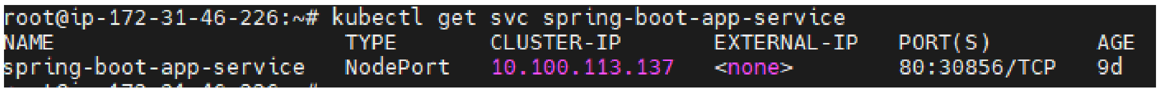


Horrey! you have successfuly deployed your application on aws EKS, let's access our application

. 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

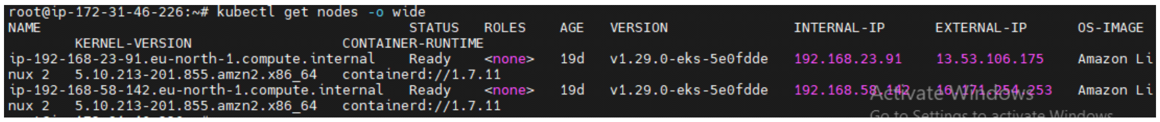


```
root@ip-172-31-46-226:~# kubectl get svc spring-boot-app-service
NAME                      TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
spring-boot-app-service   NodePort   10.100.113.137   <none>        80:30856/TCP   9d
```

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
         KERNEL-VERSION                  CONTAINER-RUNTIME
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 Li
nux 2    5.10.213-201.855.amzn2.x86_64   containerd://1.7.11
ip-192-168-58-142.eu-north-1.compute.internal   Ready    <none>   19d   v1.29.0-eks-5e0fdde   192.168.58.142  16.171.254.253  Amazon Li
nux 2    5.10.213-201.855.amzn2.x86_64   containerd://1.7.11
```

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

Example: 13.53.106.175:30856(also enable app port in our EKS Machines)

let's enable monitoring

Create EC2 instance with ubuntu OS and type should atleast t2.medium.

Login to newly created machine and run below commands to install prometheus.

. apt update

. wget https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz

. tar -xvf prometheus-2.52.0.linux-amd64.tar.gz

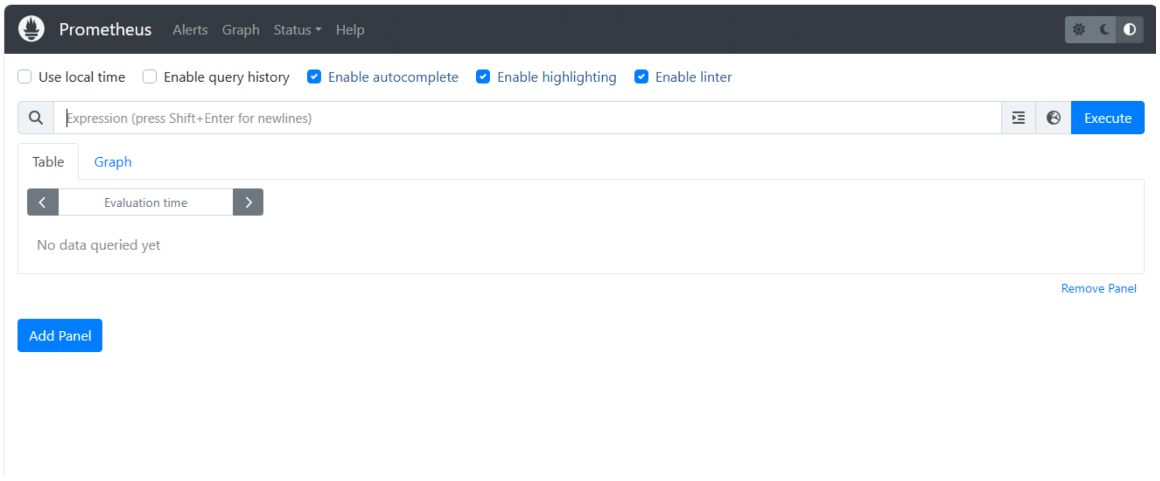
. rm -rf prometheus-2.52.0.linux-amd64.tar.gz

. cd prometheus-2.52.0.linux-amd64

. ./prometheus &

click enter

Access your prometheus GUI by ip-of-server:9090



Install Grafana.

. sudo apt-get install -y adduser libfontconfig1 musl

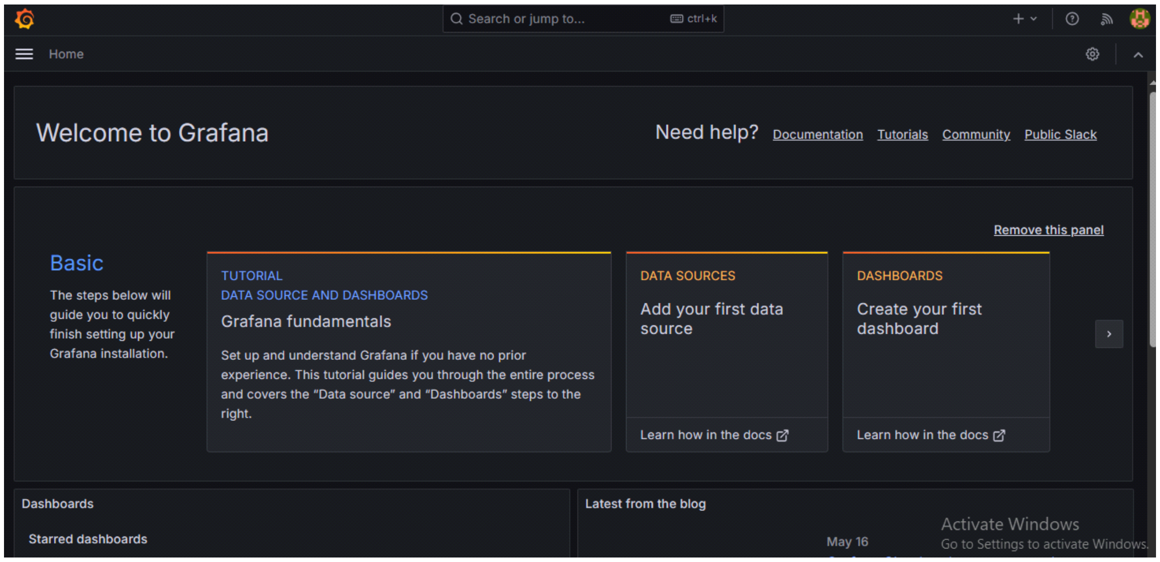
. wget https://dl.grafana.com/enterprise/release/grafana-enterprise_11.0.0_amd64.deb

. sudo dpkg -i grafana-enterprise_11.0.0_amd64.deb

. sudo /bin/systemctl start grafana-server

Access your Grafana GUI by ip-of-server:3000

Use user as ''admin" and pass as "admin" fro login



The steps below will guide you to quickly finish setting up your Grafana installation.

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

Install Blackbox exporter on our machine.

. wget https://github.com/prometheus/blackbox_exporter/releases/download/v0.25.0/blackbox_exporter-0.25.0.linux-amd64.tar.gz

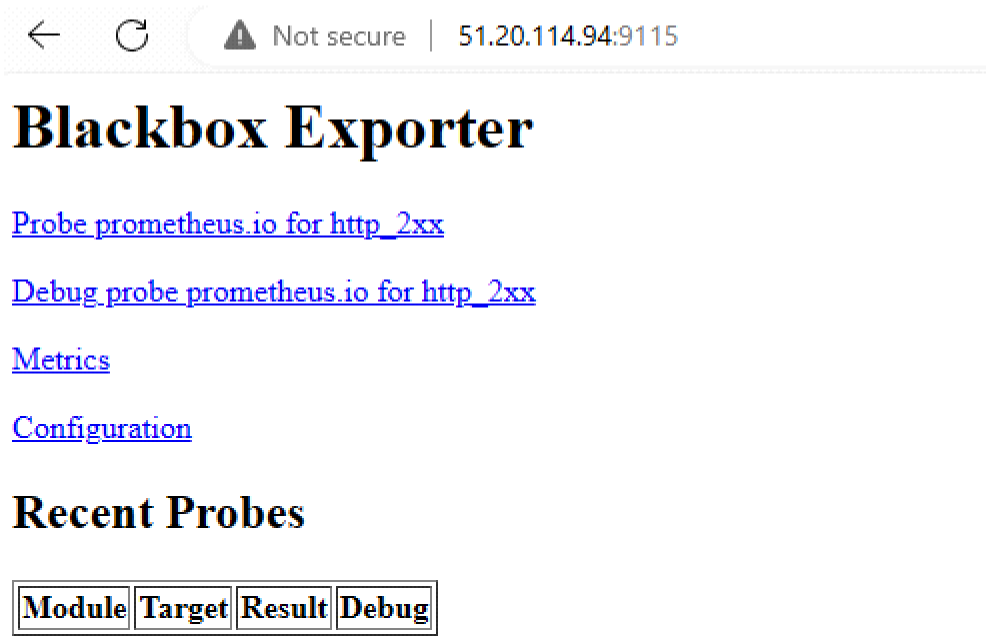
. tar -xvf blackbox_exporter-0.25.0.linux-amd64.tar.gz

. blackbox_exporter-0.25.0.linux-amd64

. ./blackbox_exporter

press enter

Access Blacbox by ip-of-server:9115



Blackbox Exporter

Probe prometheus.io for http_2xx

Debug probe prometheus.io for http_2xx

Metrics

Configuration

Recent Probes

Module	Target	Result	Debug

Let's configure monitoring for our application

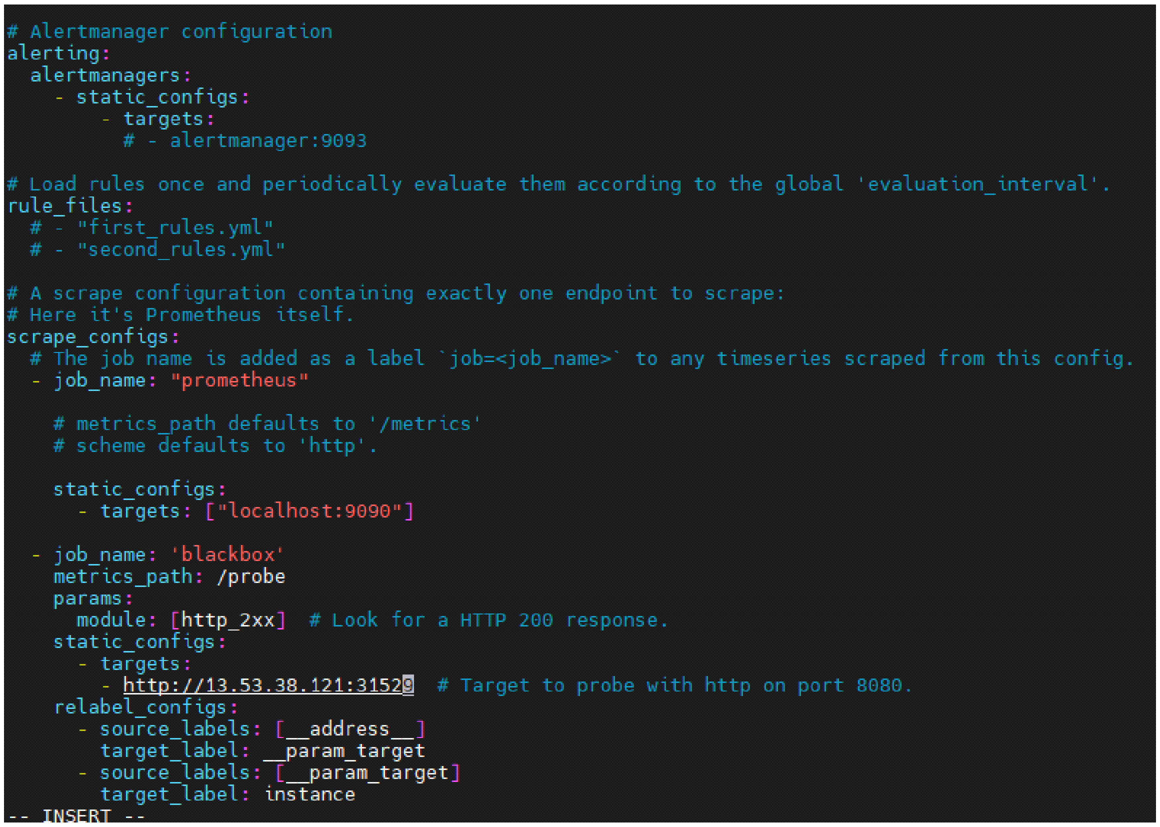
run below commands

. cd prometheus-2.52.0.linux-amd64

. vi prometheus.yml

Copy the below code and add in above file...

```
- job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]    # Look for a HTTP 200 response.
    static_configs:
      - targets:
                 - http://example.com:8080 # change url with your website url
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 127.0.0.1:9115    # change ip with your blackbox exporter server ip
```



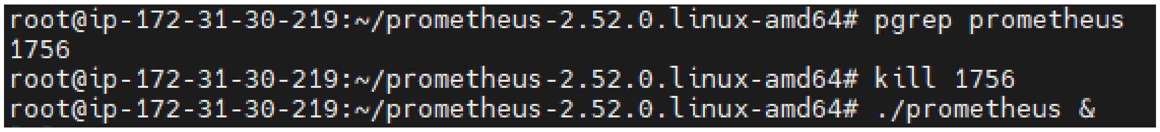
```
# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]  # Look for a HTTP 200 response.
    static_configs:
      - targets:
        - http://13.53.38.121:31520  # Target to probe with http on port 8080.
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
-- INSERT --
```

save the file and restart your promethues service by run below command

. pgrep prometheus **grep id and kill it**

. kill <id>

. ./prometheus & **then start prometheus again**



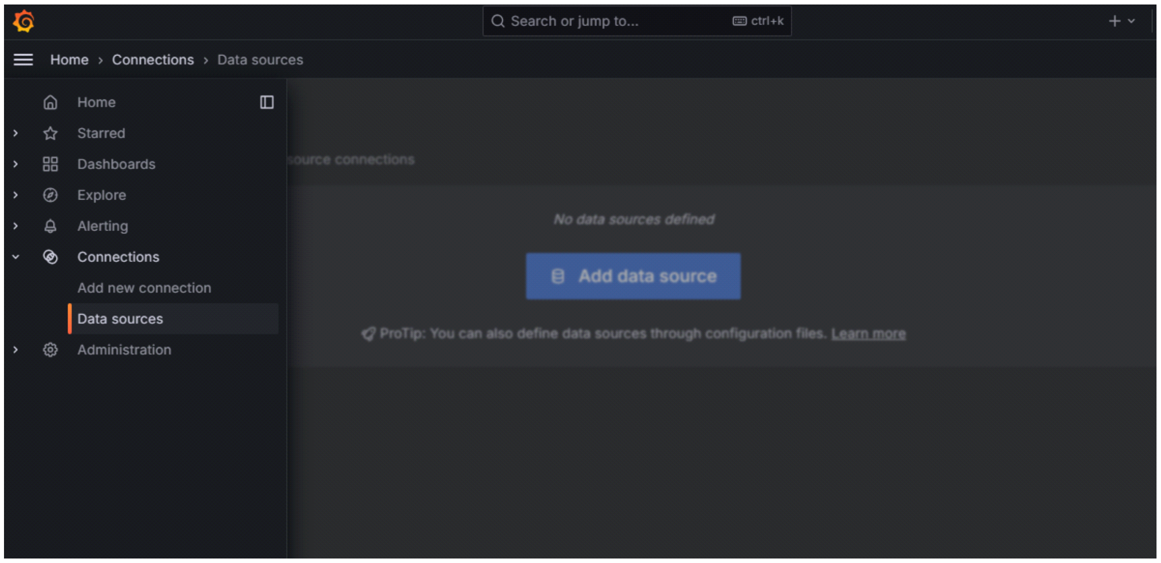
```
root@ip-172-31-30-219:~/prometheus-2.52.0.linux-amd64# pgrep prometheus
1756
root@ip-172-31-30-219:~/prometheus-2.52.0.linux-amd64# kill 1756
root@ip-172-31-30-219:~/prometheus-2.52.0.linux-amd64# ./prometheus &
```

Now let's make dashboard for this in grafana

. go back to your grafana GUI

. Click on 3 lines, present on top right side

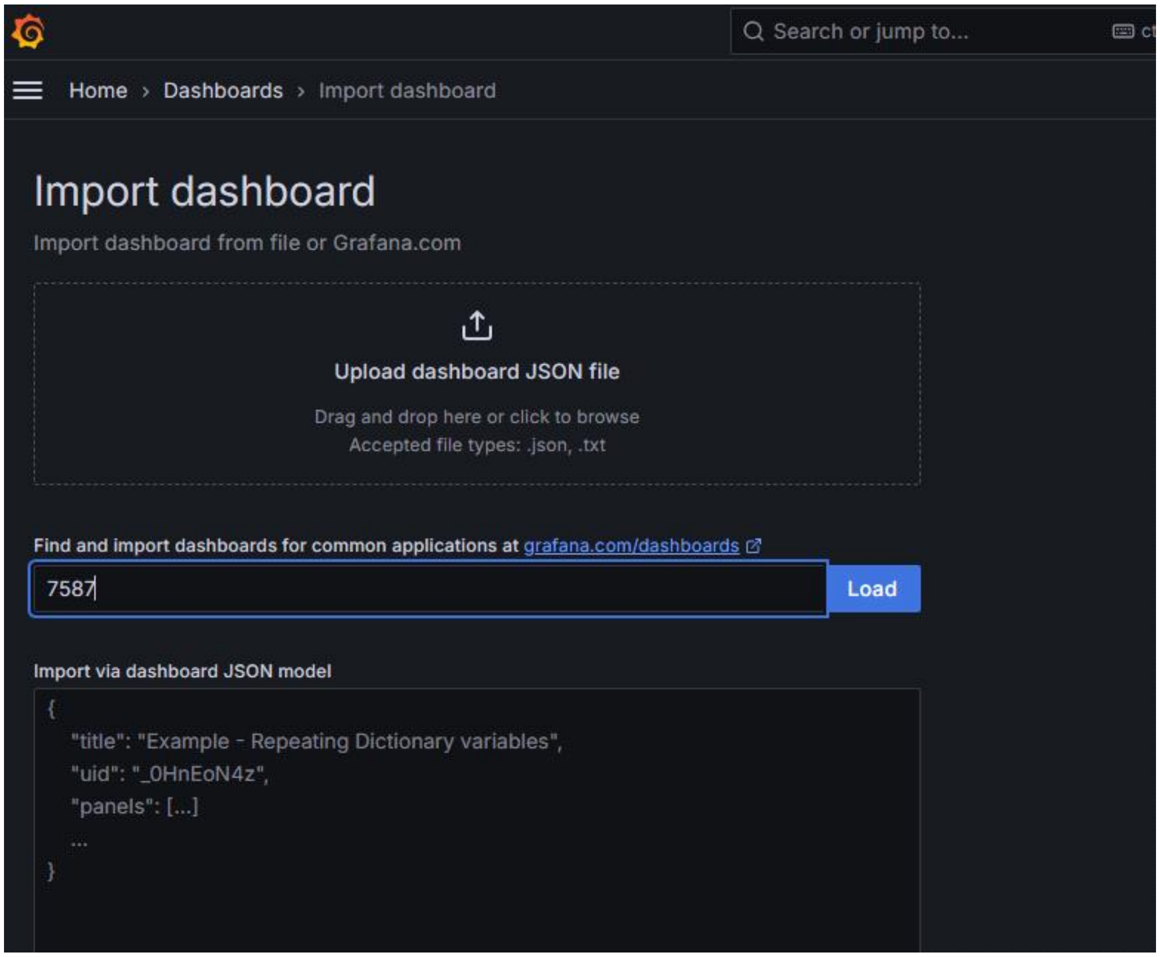
. expand connections then click on data source and click on add data source



. select promethous > in connections provide promethus url > at the end click on save..

. Now left side click on dashboards and click on import.

. Go to site https://grafana.com/grafana/dashboards/7587-prometheus-blackbox-exporter/ and copy the dashboard id and put in search box then click on load..go down slelect datasource as promethus and click on import


Search or jump to...
Home › Dashboards › Import dashboard

Import dashboard

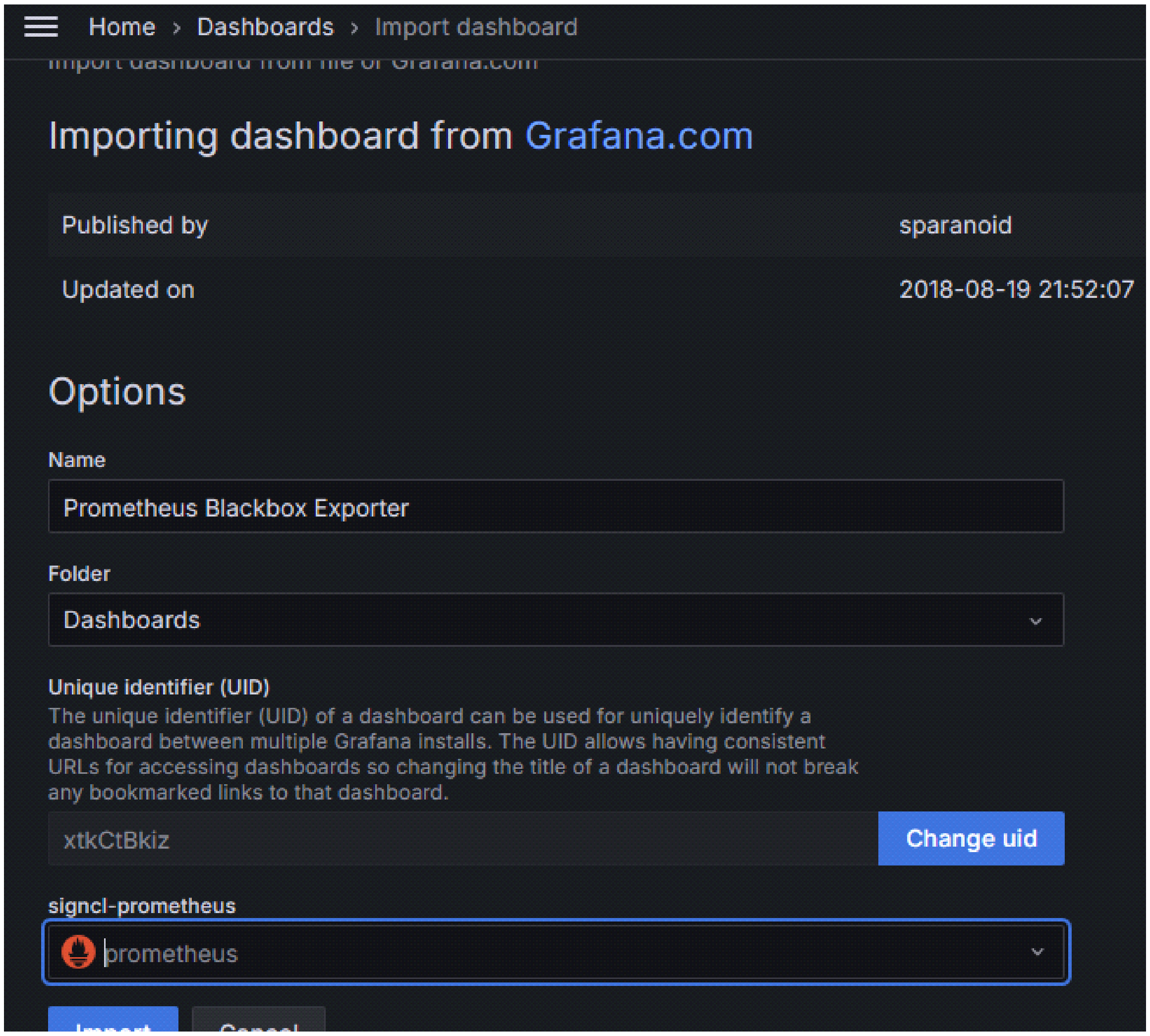
Import dashboard from file or Grafana.com

Upload dashboard JSON file
Drag and drop here or click to browse
Accepted file types: .json, .txt

Find and import dashboards for common applications at grafana.com/dashboards

Import via dashboard JSON model

```
{
  "title": "Example - Repeating Dictionary variables",
  "uid": "_0HnEoN4z",
  "panels": [...]
  ...
}
```



Importing dashboard from Grafana.com

Published by	sparanoid
Updated on	2018-08-19 21:52:07

Options

Name: Prometheus Blackbox Exporter

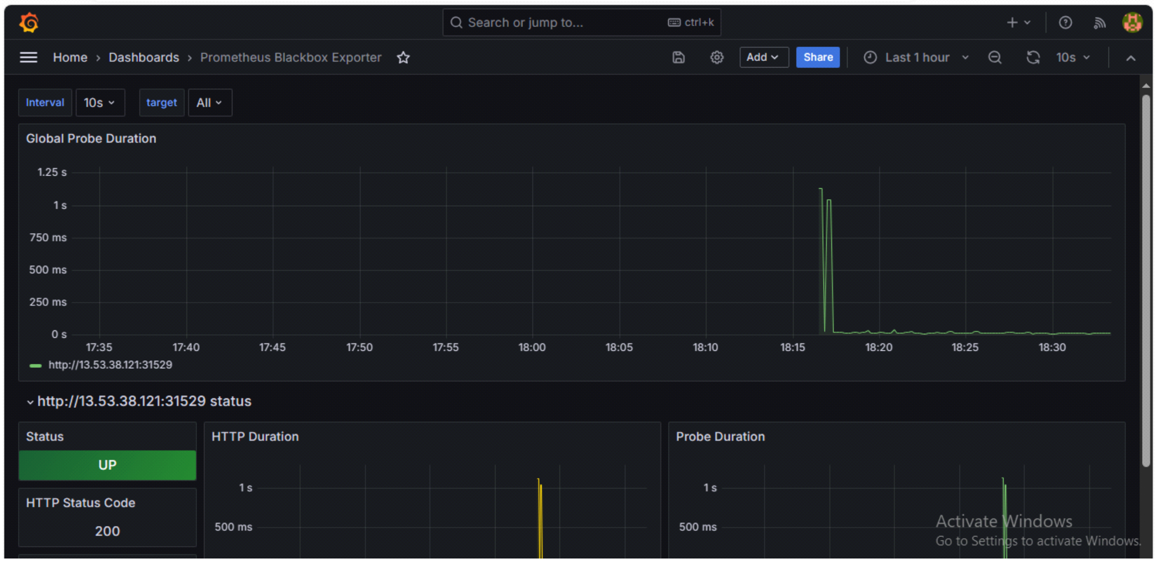
Folder: Dashboards

Unique identifier (UID)
The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

xtkCtBkiz

signcl-prometheus: prometheus

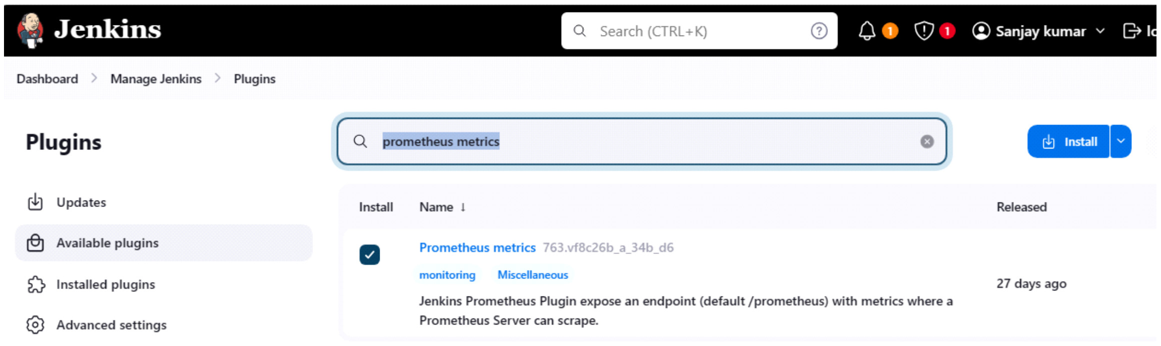
Our Dashboard is ready now


Home › Dashboards › Prometheus Blackbox Exporter
Interval 10s target All
Global Probe Duration
http://13.53.38.121:31529
http://13.53.38.121:31529 status
Status
UP
HTTP Status Code
200
HTTP Duration
Probe Duration

Let's enable monitoring for our jenkins

Go to our jenkins machine and install below plugin.

prometheus metrics


Dashboard > Manage Jenkins > Plugins

Plugins

prometheus metrics

Install	Name ↓	Released
✓	Prometheus metrics 763.vf8c26b_a_34b_d6 monitoring Miscellaneous Jenkins Prometheus Plugin expose an endpoint (default /prometheus) with metrics where a Prometheus Server can scrape.	27 days ago

After install, in system settings we will be avaible to select what we want to send to prometheus

Install Node exporter on your jenkins machine

. wget https://github.com/prometheus/node_exporter/releases/download/v1.8.0/node_exporter-1.8.0.linux-amd64.tar.gz

. tar -xvf   node_exporter-1.8.0.linux-amd64.tar.gz

. cd node_exporter-1.8.0.linux-amd64

. ./node_exporter &

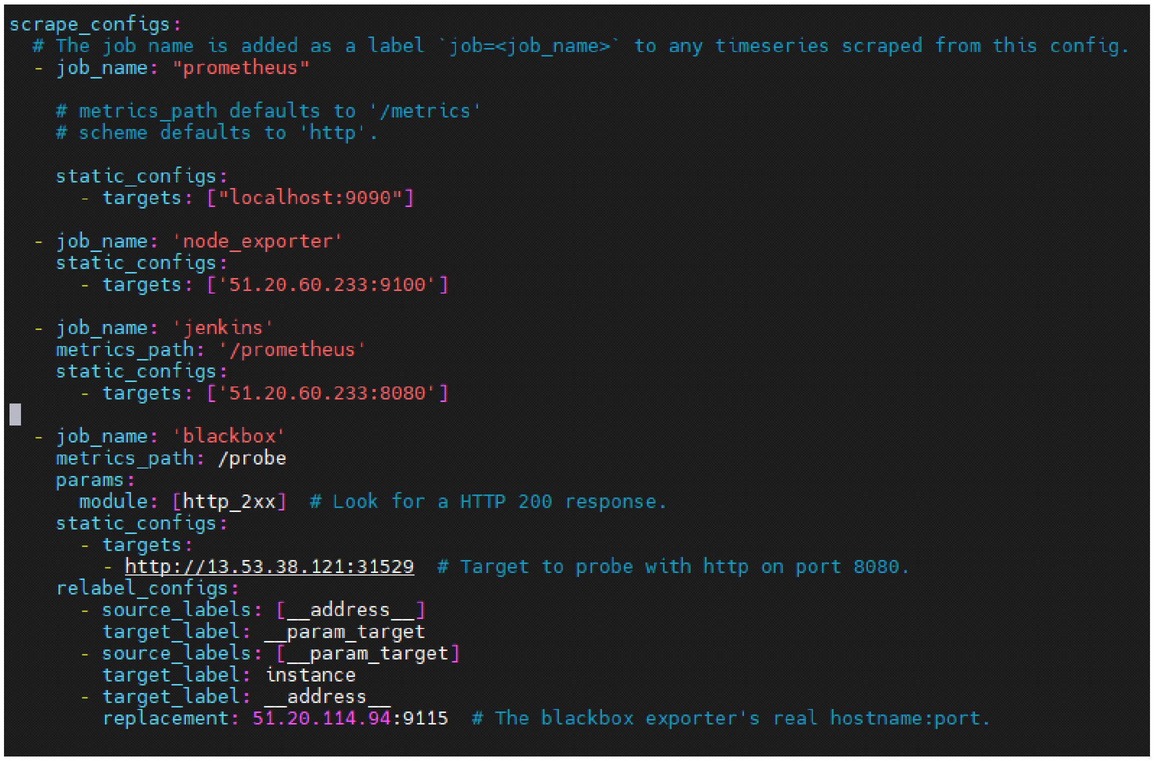
Acces your node exporter ip:9100

Now go back to your prometheus machine.

. open again prometheus.yml file and ad below code

. add below code

```
- job_name: 'node_exporter'
    static_configs:
      - targets: ['51.20.60.233:9100']      # change ip with your jenkins machine ip


  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['51.20.60.233:8080']      #change ip with your jenkins machine ip
```



```
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: 'node_exporter'
    static_configs:
      - targets: ['51.20.60.233:9100']

  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['51.20.60.233:8080']

  - job_name: 'blackbox'
    metrics_path: /probe
    params:
      module: [http_2xx]  # Look for a HTTP 200 response.
    static_configs:
      - targets:
        - http://13.53.38.121:31529  # Target to probe with http on port 8080.
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 51.20.114.94:9115  # The blackbox exporter's real hostname:port.
```

save the file and restart your promethues service by run below command

. pgrep prometheus   **grep id and kill it**

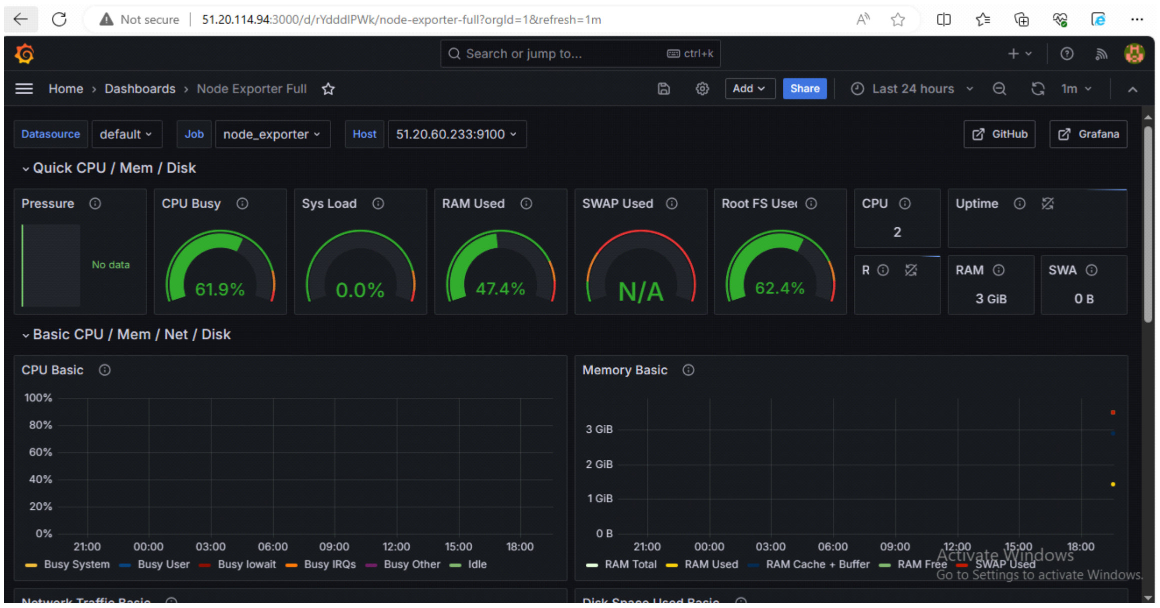
. kill <id>

. ./prometheus &   **then start prometheus again**

Now Let's create dashboard for jenkins server monitoring on Grafana

. Go to site https://grafana.com/grafana/dashboards/1860-node-exporter-full/ and copy the dashboard id.

. Go to Grafana and import new dashboard.


Not secure | 51.20.114.94:3000/d/rYdddlPWk/node-exporter-full?orgId=1&refresh=1m

Home › Dashboards › Node Exporter Full

Datasource default | Job node_exporter | Host 51.20.60.233:9100

Last 24 hours | 1m

<u>Our project is completed here</u>

<u>Thanks</u>