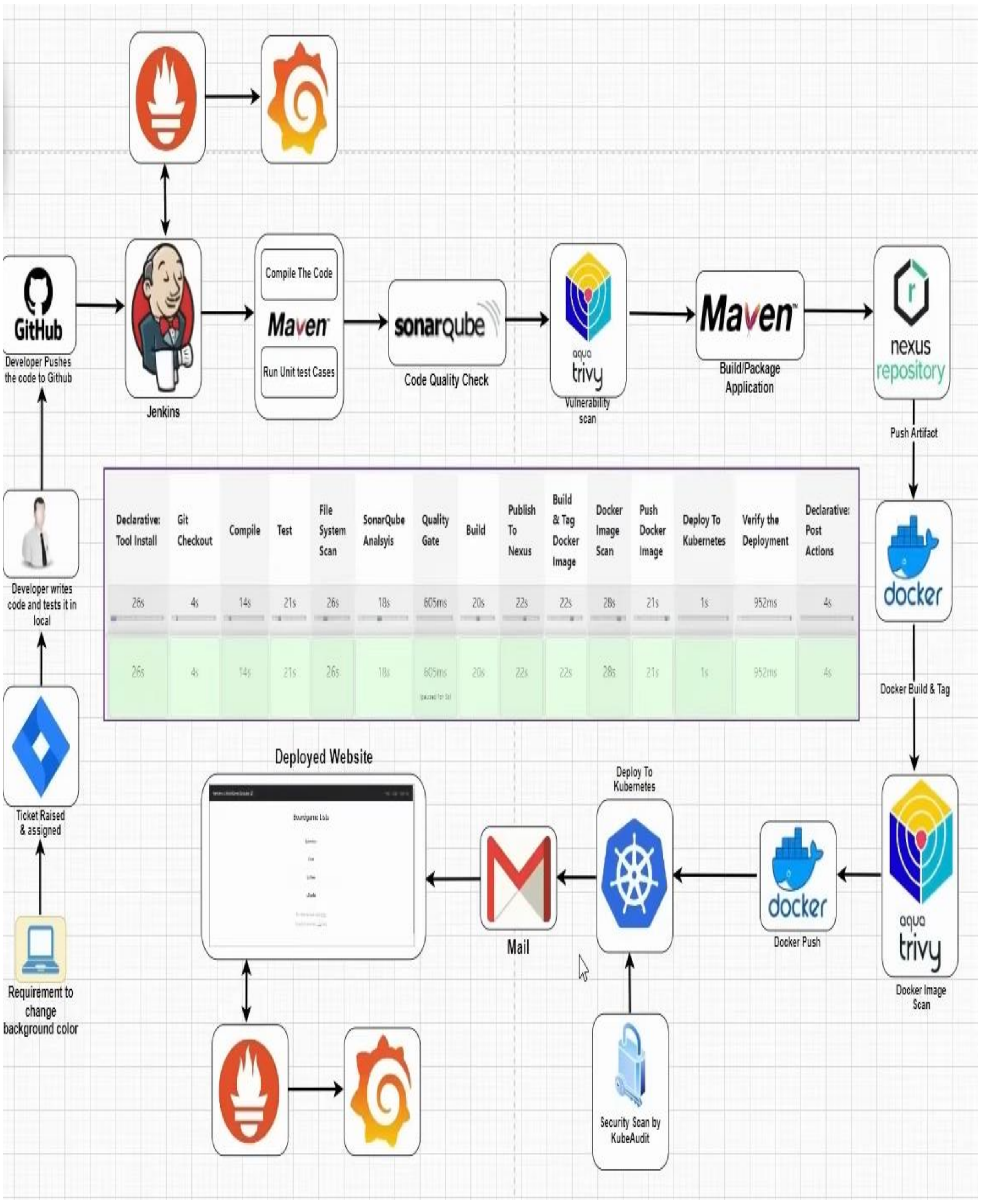
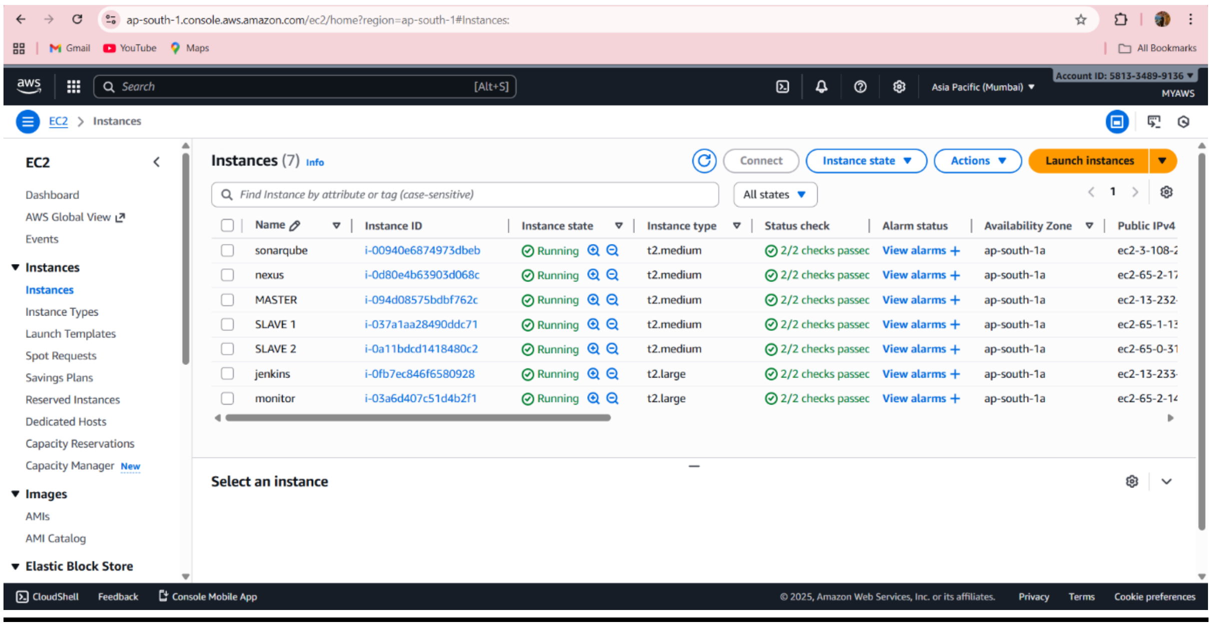
The end – end CICD DevOps Pipeline Project:



Declarative: Tool Install	Git Checkout	Compile	Test	File System Scan	SonarQube Analsyis	Quality Gate	Build	Publish To Nexus	Build & Tag Docker Image	Docker Image Scan	Push Docker Image	Deploy To Kubernetes	Verify the Deployment	Declarative: Post Actions
26s	4s	14s	21s	26s	18s	605ms	20s	22s	22s	28s	21s	1s	952ms	4s
26s	4s	14s	21s	26s	18s	605ms (paused for 5s)	20s	22s	22s	28s	21s	1s	952ms	4s

PHASE 1:

Setup Infrastructure



Instances (7)

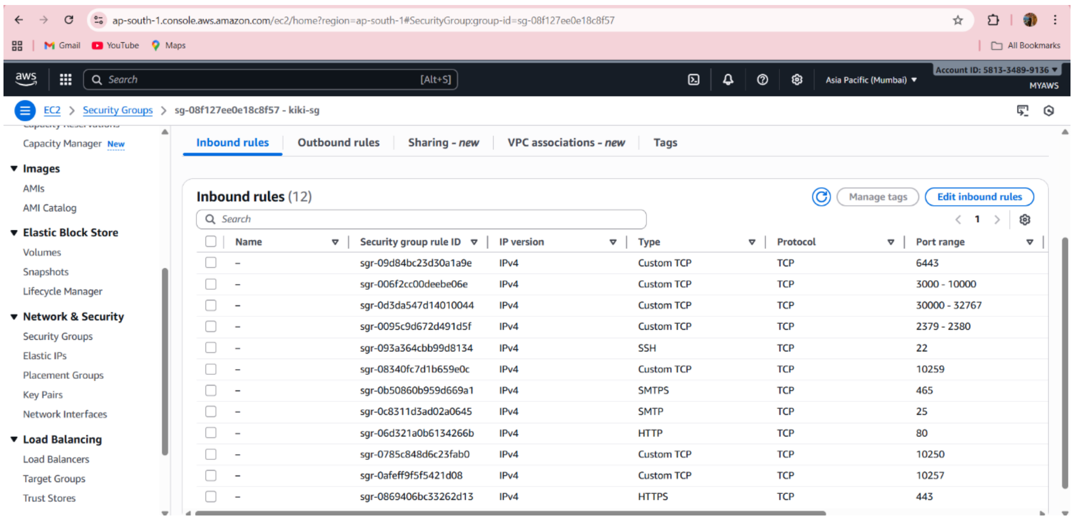
Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
sonarqube	i-00940e6874973dbeb	Running	t2.medium	2/2 checks passec	View alarms +	ap-south-1a	ec2-3-108-
nexus	i-0d80e4b63903d068c	Running	t2.medium	2/2 checks passec	View alarms +	ap-south-1a	ec2-65-2-1
MASTER	i-094d08575bdbf762c	Running	t2.medium	2/2 checks passec	View alarms +	ap-south-1a	ec2-13-232-
SLAVE 1	i-037a1aa28490ddc71	Running	t2.medium	2/2 checks passec	View alarms +	ap-south-1a	ec2-65-1-1
SLAVE 2	i-0a11bdcd1418480c2	Running	t2.medium	2/2 checks passec	View alarms +	ap-south-1a	ec2-65-0-31
jenkins	i-0fb7ec846f6580928	Running	t2.large	2/2 checks passec	View alarms +	ap-south-1a	ec2-13-233-
monitor	i-03a6d407c51d4b2f1	Running	t2.large	2/2 checks passec	View alarms +	ap-south-1a	ec2-65-2-1

To create an Ubuntu EC2 instance in AWS, follow these steps:

1. **Sign in to the AWS Management Console**:
   - Go to the AWS Management Console at https://aws.amazon.com/console/.
   - Sign in with your AWS account credentials.
2. **Navigate to EC2**:
   - Once logged in, navigate to the EC2 dashboard by typing "EC2" in the search bar at the top or by selecting "Services" and then "EC2" under the "Compute" section.
3. **Launch Instance**:
   - Click on the "Instances" link in the EC2 dashboard sidebar.
   - Click the "Launch Instance" button.
4. **Choose an Amazon Machine Image (AMI)**:
   - In the "Step 1: Choose an Amazon Machine Image (AMI)" section, select "Ubuntu" from the list of available AMIs.
   - Choose the Ubuntu version you want to use. For example, "Ubuntu Server 20.04 LTS".
   - Click "Select".
5. **Choose an Instance Type**:

- In the "Step 2: Choose an Instance Type" section, select the instance type that fits your requirements. The default option (usually a t2.micro instance) is suitable for testing and small workloads.
- Click "Next: Configure Instance Details".

6. **Configure Instance Details**:
   - Optionally, configure instance details such as network settings, subnets, IAM role, etc. You can leave these settings as default for now.
   - Click "Next: Add Storage".
7. **Add Storage**:
   - Specify the size of the root volume (default is usually fine for testing purposes).
   - Click "Next: Add Tags".
8. **Add Tags**:
   - Optionally, add tags to your instance for better organization and management.
   - Click "Next: Configure Security Group".
9. **Configure Security Group**:
   - In the "Step 6: Configure Security Group" section, configure the security group to allow SSH access (port 22) from your IP address.
   - You may also want to allow other ports based on your requirements (e.g., HTTP, HTTPS) as in this pic
   - Click "Review and Launch".


ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#SecurityGroup:group-id=sg-08f127ee0e18c8f57
Account ID: 5813-3489-9136
Asia Pacific (Mumbai)
EC2 > Security Groups > sg-08f127ee0e18c8f57 - kiki-sg
Inbound rules
Outbound rules
Sharing - new
VPC associations - new
Tags
Manage tags
Edit inbound rules

Inbound rules (12)

Name	Security group rule ID	IP version	Type	Protocol	Port range
–	sgr-09d84bc23d30a1a9e	IPv4	Custom TCP	TCP	6443
–	sgr-006f2cc00deebe06e	IPv4	Custom TCP	TCP	3000 - 10000
–	sgr-0d3da547d14010044	IPv4	Custom TCP	TCP	30000 - 32767
–	sgr-0095c9d672d491d5f	IPv4	Custom TCP	TCP	2379 - 2380
–	sgr-093a364cbb99d8134	IPv4	SSH	TCP	22
–	sgr-08340fc7d1b659e0c	IPv4	Custom TCP	TCP	10259
–	sgr-0b50860b959d669a1	IPv4	SMTPS	TCP	465
–	sgr-0c8311d3ad02a0645	IPv4	SMTP	TCP	25
–	sgr-06d321a0b6134266b	IPv4	HTTP	TCP	80
–	sgr-0785c848d6c23fab0	IPv4	Custom TCP	TCP	10250
–	sgr-0afeff9f5f5421d08	IPv4	Custom TCP	TCP	10257
–	sgr-0869406bc33262d13	IPv4	HTTPS	TCP	443

10. **Review and Launch**:
    - Review the configuration of your instance.
    - Click "Launch".

11. **Select Key Pair**:
    - In the pop-up window, select an existing key pair or create a new one.
    - Check the acknowledgment box.
    - Click "Launch Instances".
12. **Access Your Instance**:
    - Use Mobaxterm

Setup K8-Cluster using kubeadm [K8 Version-->1.28.1]

1. Update System Packages [On Master & Worker Node]

```
sudo apt-get update
```

2. Install Docker[On Master & Worker Node]

```
sudo apt install docker.io -y
```

```
sudo chmod 666 /var/run/docker.sock
```

3. Install Required Dependencies for Kubernetes[On Master & Worker Node]

```
sudo apt-get install -y apt-transport-https ca-certificates curl gnupg
```

```
sudo mkdir -p -m 755 /etc/apt/keyrings
```

4. Add Kubernetes Repository and GPG Key[On Master & Worker Node]

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

5. Update Package List[On Master & Worker Node]

```
sudo apt update
```

6. Install Kubernetes Components[On Master & Worker Node]

```
sudo apt install -y kubeadm=1.28.1-1.1 kubelet=1.28.1-1.1 kubectl=1.28.1-1.1
```

7. Initialize Kubernetes Master Node [On MasterNode]

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

8. Configure Kubernetes Cluster [On MasterNode]

```
mkdir -p $HOME/.kube
```

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

9. Deploy Networking Solution (Calico) [On MasterNode]

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

10. Deploy Ingress Controller (NGINX) [On MasterNode]

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.49.0/deploy/static/provider/baremetal/deploy.yaml
```

Installing Jenkins on Ubuntu

```
#!/bin/bash


# Install OpenJDK 17 JRE Headless

sudo apt install openjdk-17-jre-headless -y


# Download Jenkins GPG key

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key


# Add Jenkins repository to package manager sources

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


# Update package manager repositories

sudo apt-get update


# Install Jenkins

sudo apt-get install jenkins -y
```

Save this script in a file, for example, install_jenkins.sh, and make it executable using:

```
chmod +x install_jenkins.sh
```

Then, you can run the script using:

```
./install_jenkins.sh
```

This script will automate the installation process of OpenJDK 17 JRE Headless and Jenkins.

Install docker for future use

```
#!/bin/bash


# Update package manager repositories

sudo apt-get update


# Install necessary dependencies

sudo apt-get install -y ca-certificates curl


# Create directory for Docker GPG key

sudo install -m 0755 -d /etc/apt/keyrings


# Download Docker's GPG key

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc


# Ensure proper permissions for the key

sudo chmod a+r /etc/apt/keyrings/docker.asc


# Add Docker repository to Apt sources

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null


# Update package manager repositories

sudo apt-get update
```

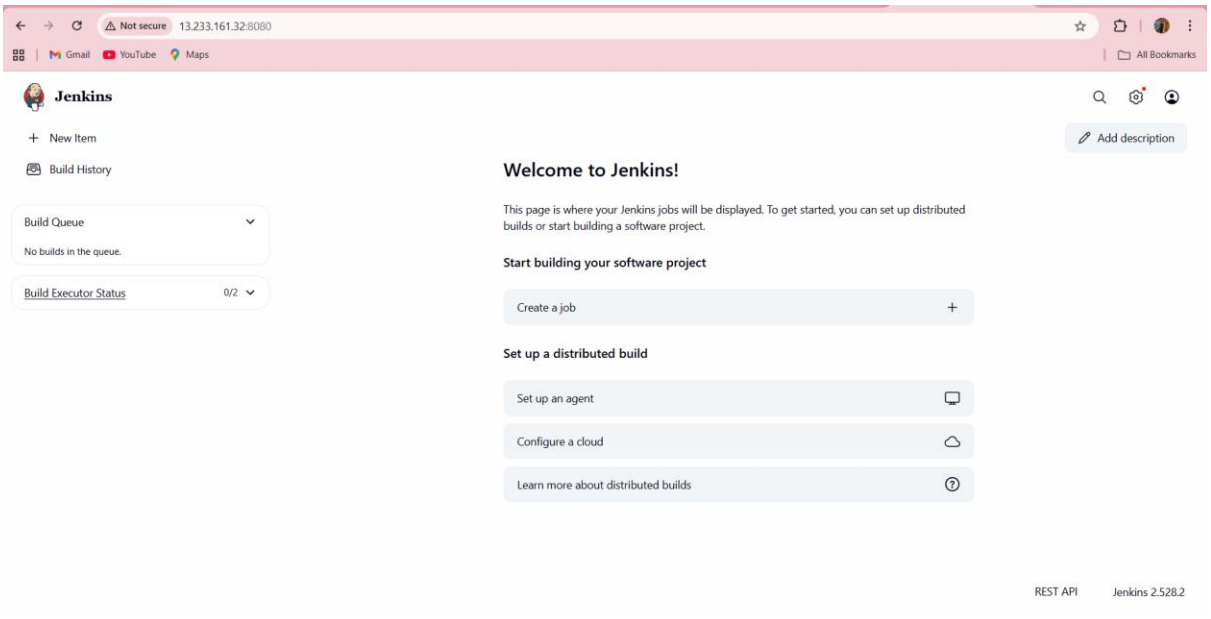
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Save this script in a file, for example, install_docker.sh, and make it executable using:

chmod +x install_docker.sh

Then, you can run the script using:

./install_docker.sh



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Set up a distributed build

No builds in the queue.

SetUp Nexus

#!/bin/bash

# Update package manager repositories

sudo apt-get update

# Install necessary dependencies

sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key

sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc


# Ensure proper permissions for the key

sudo chmod a+r /etc/apt/keyrings/docker.asc


# Add Docker repository to Apt sources

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null


# Update package manager repositories

sudo apt-get update


sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Save this script in a file, for example, install_docker.sh, and make it executable using:

```
chmod +x install_docker.sh
```

Then, you can run the script using:

```
./install_docker.sh
```

Create Nexus using docker container

To create a Docker container running Nexus 3 and exposing it on port 8081, you can use the following command:

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

This command does the following:

- -d: Detaches the container and runs it in the background.
- --name nexus: Specifies the name of the container as "nexus".
- -p 8081:8081: Maps port 8081 on the host to port 8081 on the container, allowing access to Nexus through port 8081.
- sonatype/nexus3:latest: Specifies the Docker image to use for the container, in this case, the latest version of Nexus 3 from the Sonatype repository.

After running this command, Nexus will be accessible on your host machine at http://IP:8081.

Get Nexus initial password

Your provided commands are correct for accessing the Nexus password stored in the container. Here's a breakdown of the steps:

1. **Get Container ID**: You need to find out the ID of the Nexus container. You can do this by running:

```
docker ps
```

This command lists all running containers along with their IDs, among other information.

2. **Access Container's Bash Shell**: Once you have the container ID, you can execute the docker exec command to access the container's bash shell:

```
docker exec -it <container_ID> /bin/bash
```

Replace <container_ID> with the actual ID of the Nexus container.

3. **Navigate to Nexus Directory**: Inside the container's bash shell, navigate to the directory where Nexus stores its configuration:

```
cd sonatype-work/nexus3
```

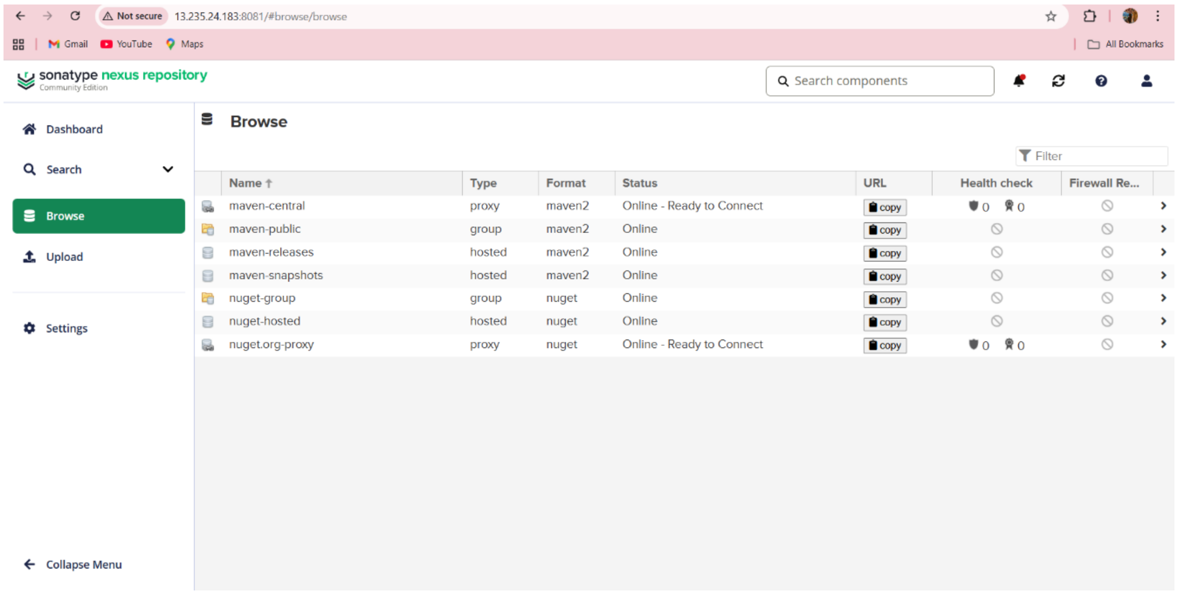
4. **View Admin Password**: Finally, you can view the admin password by displaying the contents of the admin.password file:

```
cat admin.password
```

5. **Exit the Container Shell**: Once you have retrieved the password, you can exit the container's bash shell:

```
exit
```

This process allows you to access the Nexus admin password stored within the container. Make sure to keep this password secure, as it grants administrative access to your Nexus instance.



13.235.24.183:8081/#browse/browse

Browse

Name ↑	Type	Format	Status	URL	Health check	Firewall Re...
maven-central	proxy	maven2	Online - Ready to Connect	copy	0 0	⃠
maven-public	group	maven2	Online	copy	⃠	⃠
maven-releases	hosted	maven2	Online	copy	⃠	⃠
maven-snapshots	hosted	maven2	Online	copy	⃠	⃠
nuget-group	group	nuget	Online	copy	⃠	⃠
nuget-hosted	hosted	nuget	Online	copy	⃠	⃠
nuget.org-proxy	proxy	nuget	Online - Ready to Connect	copy	0 0	⃠

SetUp SonarQube

```
#!/bin/bash

# Update package manager repositories
sudo apt-get update

# Install necessary dependencies
sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key
sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker repository to Apt sources
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update package manager repositories
sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Save this script in a file, for example, install_docker.sh, and make it executable using:

```
chmod +x install_docker.sh
```

Then, you can run the script using:

```
./install_docker.sh
```

Create Sonarqube Docker container

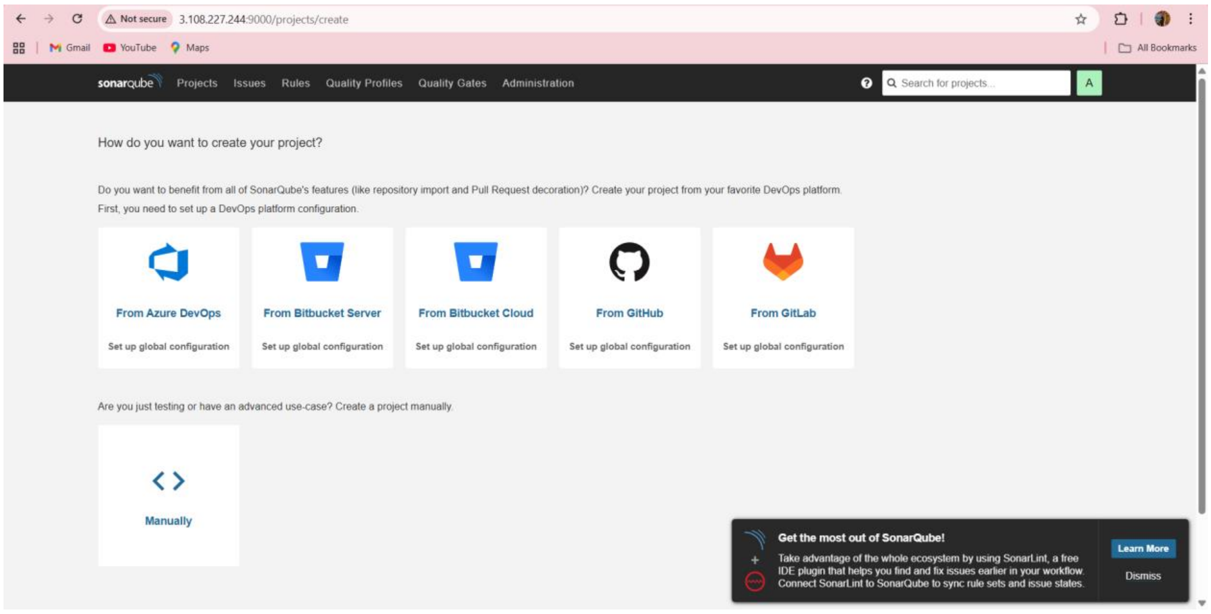
To run SonarQube in a Docker container with the provided command, you can follow these steps:

1. Open your terminal or command prompt.
2. Run the following command:

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

This command will download the sonarqube:lts-community Docker image from Docker Hub if it's not already available locally. Then, it will create a container named "sonar" from this image, running it in detached mode (-d flag) and mapping port 9000 on the host machine to port 9000 in the container (-p 9000:9000 flag).

3. Access SonarQube by opening a web browser and navigating to http://VmIP:9000.



How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.
First, you need to set up a DevOps platform configuration.

Are you just testing or have an advanced use-case? Create a project manually.

Get the most out of SonarQube!
Take advantage of the whole ecosystem by using SonarLint, a free IDE plugin that helps you find and fix issues earlier in your workflow. Connect SonarLint to SonarQube to sync rule sets and issue states.

This will start the SonarQube server, and you should be able to access it using the provided URL. If you're running Docker on a remote server or a different port, replace localhost with the appropriate hostname or IP address and adjust the port accordingly.

PHASE 2:

Steps to create a private Git repository, generate a personal access token, connect to the repository, and push code to it:

1. **Create a Private Git Repository**:
   - Go to your preferred Git hosting platform (e.g., GitHub, GitLab, Bitbucket).
   - Log in to your account or sign up if you don't have one.
   - Create a new repository and set it as private.
2. **Generate a Personal Access Token**:
   - Navigate to your account settings or profile settings.
   - Look for the "Developer settings" or "Personal access tokens" section.
   - Generate a new token, providing it with the necessary permissions (e.g., repo access).
3. **Clone the Repository Locally**:
   - Open Git Bash or your terminal.
   - Navigate to the directory where you want to clone the repository.
   - Use the git clone command followed by the repository's URL. For example:

```
git clone <repository_URL>
```

4. Replace <repository_URL> with the URL of your private repository.
5. **Add Your Source Code Files**:
   - Navigate into the cloned repository directory.
   - Paste your source code files or create new ones inside this directory.
6. **Stage and Commit Changes**:
   - Use the git add command to stage the changes:

```
git add .
```

   - Use the git commit command to commit the staged changes along with a meaningful message:

```
git commit -m "Your commit message here"
```

7. **Push Changes to the Repository**:

- Use the git push command to push your committed changes to the remote repository:

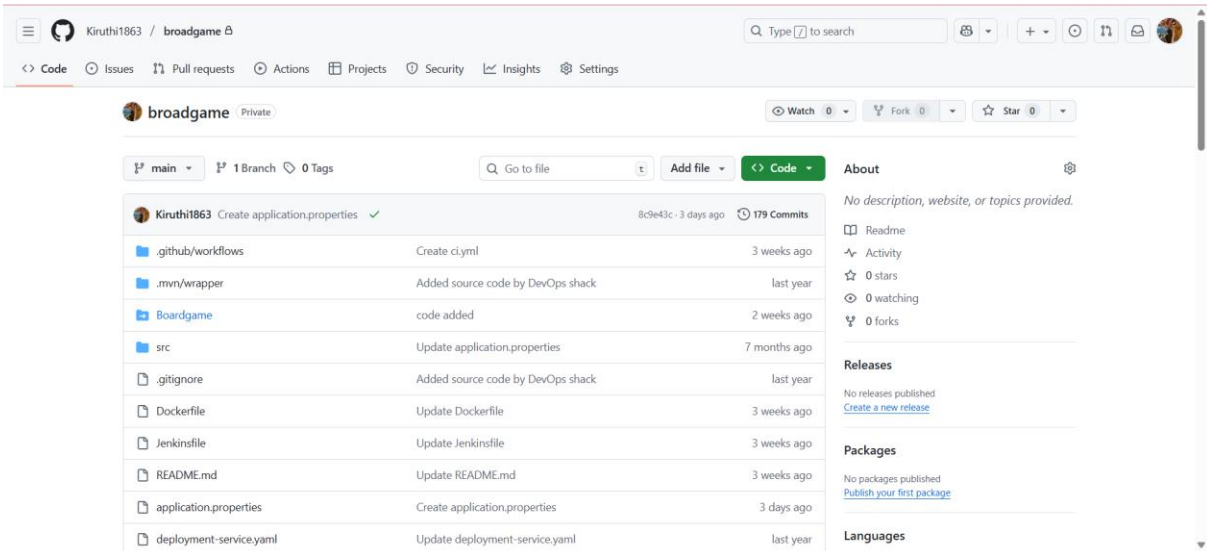
```
git push
```

- If it's your first time pushing to this repository, you might need to specify the remote and branch:

```
git push -u origin master
```

8. Replace master with the branch name if you're pushing to a different branch.
9. **Enter Personal Access Token as Authentication**:
   - When prompted for credentials during the push, enter your username (usually your email) and use your personal access token as the password.

By following these steps, you'll be able to create a private Git repository, connect to it using Git Bash, and push your code changes securely using a personal access token for authentication.



Kiruthi1863 Create application.properties ✓ 8c9e43c · 3 days ago 179 Commits

.github/workflows	Create ci.yml	3 weeks ago
.mvn/wrapper	Added source code by DevOps shack	last year
Boardgame	code added	2 weeks ago
src	Update application.properties	7 months ago
.gitignore	Added source code by DevOps shack	last year
Dockerfile	Update Dockerfile	3 weeks ago
Jenkinsfile	Update Jenkinsfile	3 weeks ago
README.md	Update README.md	3 weeks ago
application.properties	Create application.properties	3 days ago
deployment-service.yaml	Update deployment-service.yaml	last year

About

No description, website, or topics provided.

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

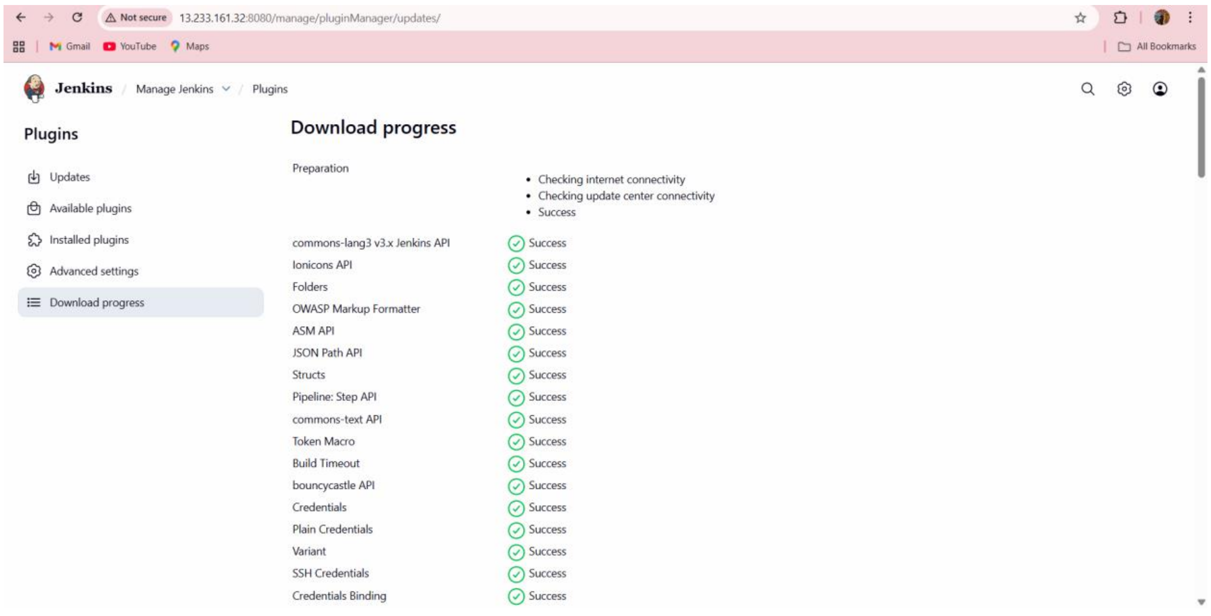
Packages

No packages published
Publish your first package

Languages

PHASE 3:

Install Plugins in Jenkins



Download progress

Preparation	• Checking internet connectivity • Checking update center connectivity • Success
commons-lang3 v3.x Jenkins API	✓ Success
Ionicons API	✓ Success
Folders	✓ Success
OWASP Markup Formatter	✓ Success
ASM API	✓ Success
JSON Path API	✓ Success
Structs	✓ Success
Pipeline: Step API	✓ Success
commons-text API	✓ Success
Token Macro	✓ Success
Build Timeout	✓ Success
bouncycastle API	✓ Success
Credentials	✓ Success
Plain Credentials	✓ Success
Variant	✓ Success
SSH Credentials	✓ Success
Credentials Binding	✓ Success

1. **Eclipse Temurin Installer**:
   - This plugin enables Jenkins to automatically install and configure the Eclipse Temurin JDK (formerly known as AdoptOpenJDK).
   - To install, go to Jenkins dashboard -> Manage Jenkins -> Manage Plugins -> Available tab.
   - Search for "Eclipse Temurin Installer" and select it.
   - Click on the "Install without restart" button.
2. **Pipeline Maven Integration**:
   - This plugin provides Maven support for Jenkins Pipeline.
   - It allows you to use Maven commands directly within your Jenkins Pipeline scripts.
   - To install, follow the same steps as above, but search for "Pipeline Maven Integration" instead.
3. **Config File Provider**:

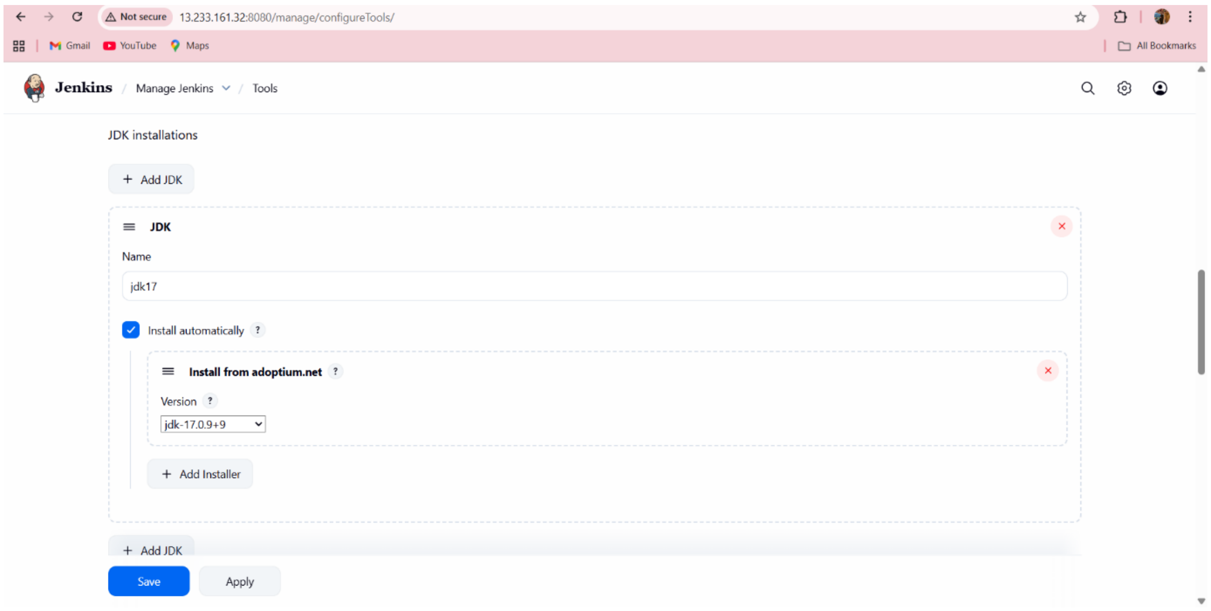
- This plugin allows you to define configuration files (e.g., properties, XML, JSON) centrally in Jenkins.
- These configurations can then be referenced and used by your Jenkins jobs.
- Install it using the same procedure as mentioned earlier.

4. **SonarQube Scanner**:
    - SonarQube is a code quality and security analysis tool.
    - This plugin integrates Jenkins with SonarQube by providing a scanner that analyzes code during builds.
    - You can install it from the Jenkins plugin manager as described above.
5. **Kubernetes CLI**:
    - This plugin allows Jenkins to interact with Kubernetes clusters using the Kubernetes command-line tool (kubectl).
    - It's useful for tasks like deploying applications to Kubernetes from Jenkins jobs.
    - Install it through the plugin manager.
6. **Kubernetes**:
    - This plugin integrates Jenkins with Kubernetes by allowing Jenkins agents to run as pods within a Kubernetes cluster.
    - It provides dynamic scaling and resource optimization capabilities for Jenkins builds.
    - Install it from the Jenkins plugin manager.
7. **Docker**:
    - This plugin allows Jenkins to interact with Docker, enabling Docker builds and integration with Docker registries.
    - You can use it to build Docker images, run Docker containers, and push/pull images from Docker registries.
    - Install it from the plugin manager.
8. **Docker Pipeline Step**:
    - This plugin extends Jenkins Pipeline with steps to build, publish, and run Docker containers as part of your Pipeline scripts.
    - It provides a convenient way to manage Docker containers directly from Jenkins Pipelines.
    - Install it through the plugin manager like the others.

After installing these plugins, you may need to configure them according to your specific environment and requirements. This typically involves setting up credentials, configuring paths, and specifying options in Jenkins global configuration or individual job configurations. Each plugin usually comes with its own set of documentation to guide you through the configuration process.

Configure Above Plugins in Jenkins

Jdk configure


13.233.161.32:8080/manage/configureTools/

Jenkins / Manage Jenkins / Tools

JDK installations

Add JDK

JDK

Name: jdk17

✓ Install automatically

Install from adoptium.net

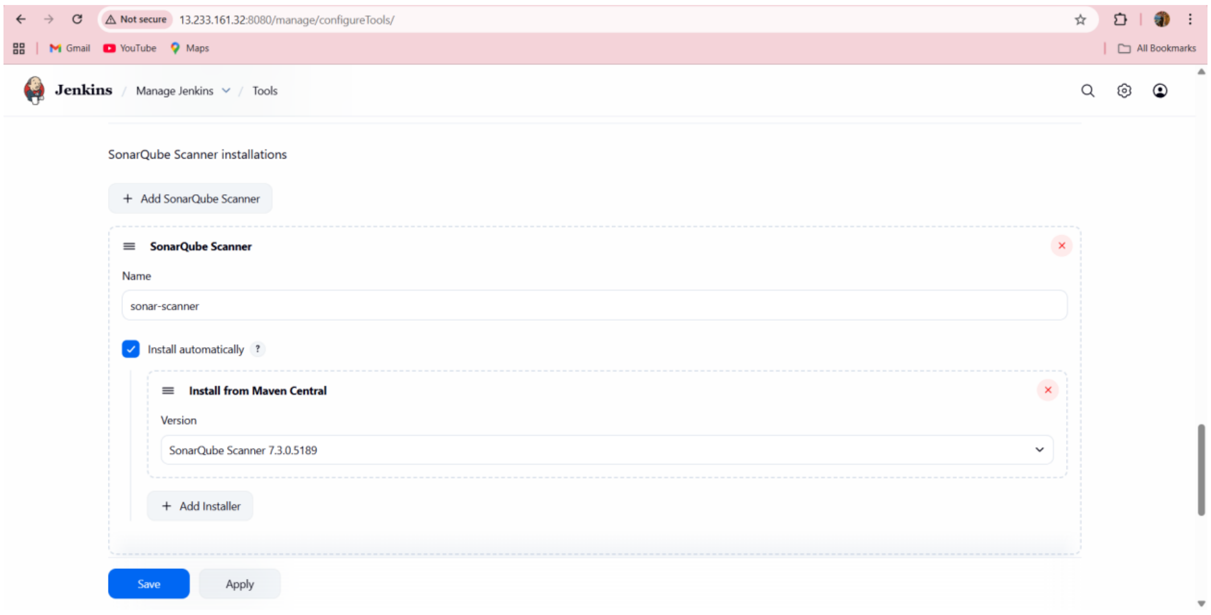
Version: jdk-17.0.9+9

Add Installer

Add JDK

Save Apply

Sonarqube configure


13.233.161.32:8080/manage/configureTools/

Jenkins / Manage Jenkins / Tools

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name: sonar-scanner

✓ Install automatically

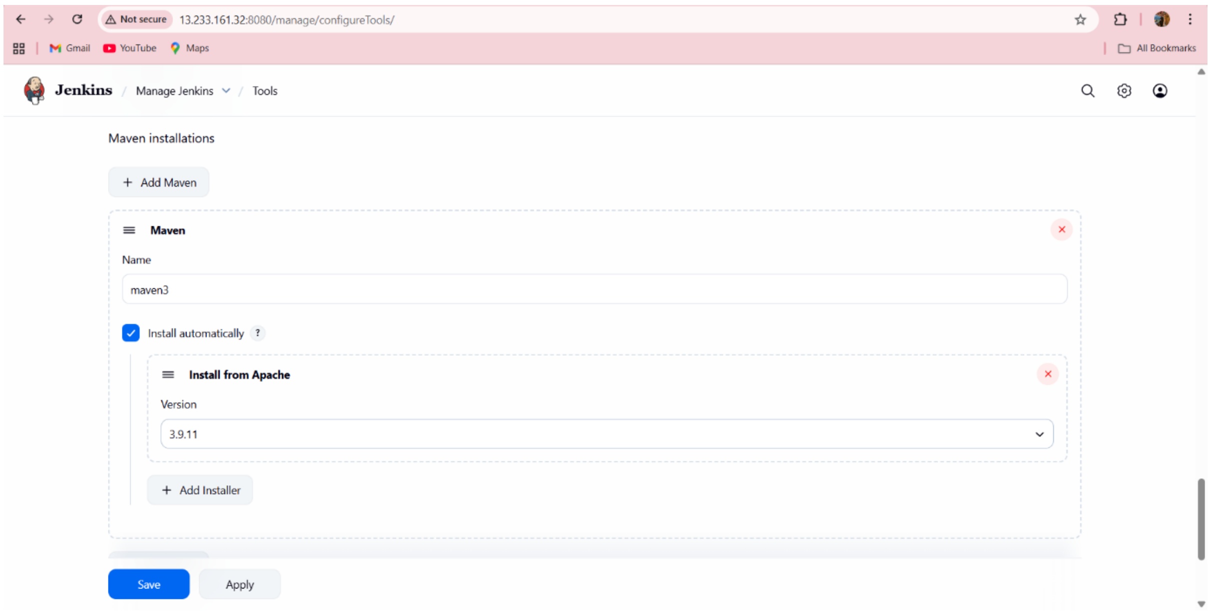
Install from Maven Central

Version: SonarQube Scanner 7.3.0.5189

Add Installer

Save Apply

Maven configure



13.233.161.32:8080/manage/configureTools/

Jenkins / Manage Jenkins / Tools

Maven installations

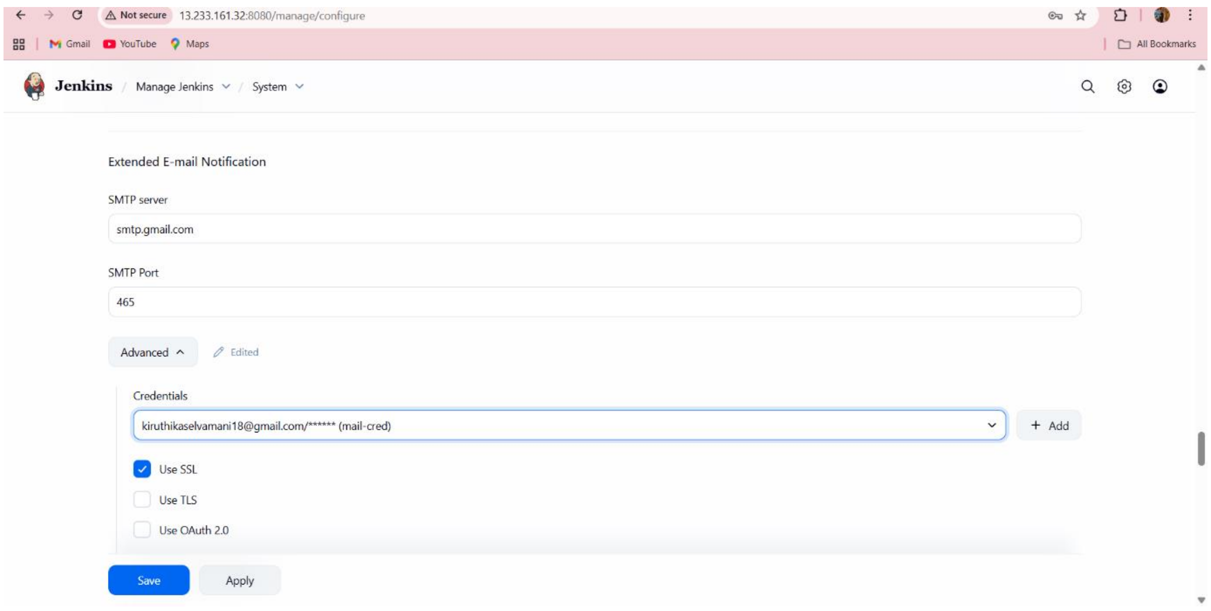
Maven

Name: maven3

Install automatically ✓

Install from Apache

Version: 3.9.11



13.233.161.32:8080/manage/configure

Jenkins / Manage Jenkins / System

Extended E-mail Notification

SMTP server: smtp.gmail.com

SMTP Port: 465

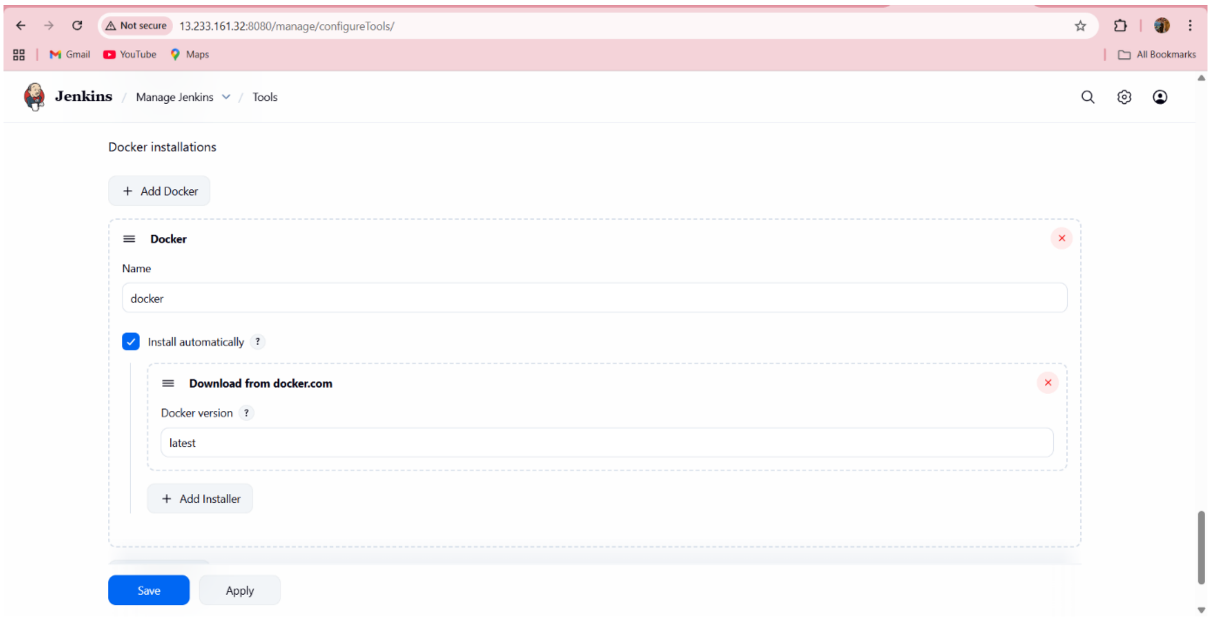
Credentials: kiruthikaselvamani18@gmail.com/****** (mail-cred)

Use SSL ✓

Use TLS

Use OAuth 2.0

Docker configure


Not secure 13.233.161.32:8080/manage/configureTools/

Jenkins / Manage Jenkins / Tools

Docker installations

+ Add Docker

Docker

Name

docker

✓ Install automatically ?

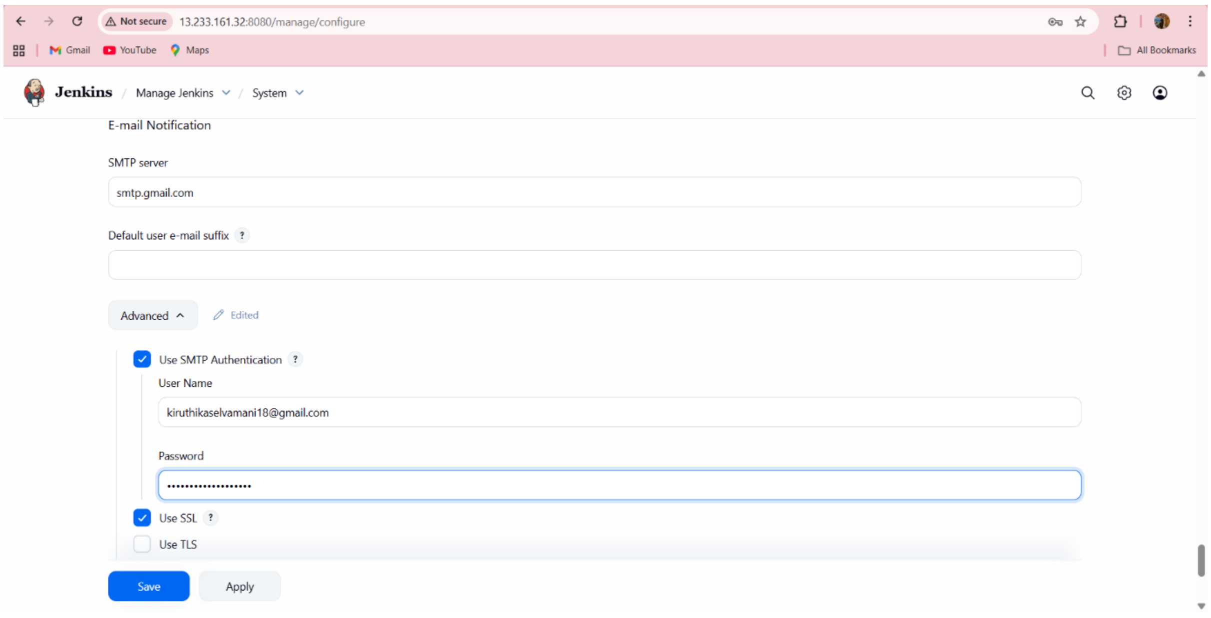
Download from docker.com

Docker version ?

latest

+ Add Installer

Save Apply


Not secure 13.233.161.32:8080/manage/configure

Jenkins / Manage Jenkins / System

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

Advanced Edited

✓ Use SMTP Authentication ?

User Name

kiruthikaselvamani18@gmail.com

Password

••••••••••••••••••••

✓ Use SSL ?

Use TLS

Save Apply

Pipeline:

```
pipeline {
    agent any

    tools {
        jdk 'jdk17'
        maven 'maven3'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
        DOCKER_IMAGE = "kiruthikaselvamani/board_game:${env.BUILD_NUMBER}"
        DOCKER_IMAGE_LATEST = "kiruthikaselvamani/board_game:latest"
        NEXUS_REPO_URL = 'http://65.2.171.132:8081'
    }

    stages {
        // ... your existing stages (Git Checkout, Compile, SonarQube, etc.)

        stage('Fix Kubernetes RBAC') {
            steps {
                script {
                    withCredentials([string(credentialsId: 'k8-cred', variable: 'K8S_TOKEN')]) {
                        sh """
                        echo "🔐 Fixing Kubernetes RBAC permissions..."

                        # Create temporary kubeconfig
                        cat > /tmp/rbac-fix-\$\$ << EOF
apiVersion: v1
```

```
kind: Config
clusters:
- name: kubernetes
  cluster:
    server: https://172.31.45.168:6443
    insecure-skip-tls-verify: true
users:
- name: jenkins
  user:
    token: $K8S_TOKEN
contexts:
- name: jenkins-context
  context:
    cluster: kubernetes
    user: jenkins
    namespace: webapps
current-context: jenkins-context
EOF

                        export KUBECONFIG=/tmp/rbac-fix-\$\$

                        # Apply RBAC permissions for logs
                        kubectl apply -f - << RBACEOF
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: webapps
  name: pod-logs-reader
rules:
- apiGroups: [""]
  resources: ["pods", "pods/log"]
  verbs: ["get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-logs
  namespace: webapps
subjects:
- kind: ServiceAccount
  name: jenkins
  namespace: webapps
roleRef:
  kind: Role
  name: pod-logs-reader
  apiGroup: rbac.authorization.k8s.io
RBACEOF

                        echo "✅ RBAC permissions applied!"
                        rm -f /tmp/rbac-fix-\$\$
                        """
                    }
                }
            }
        }

        stage('Fix Service Type') {
            steps {
                script {
                    withCredentials([string(credentialsId: 'k8-cred', variable: 'K8S_TOKEN')]) {
                        sh """
                        echo "🌐 Fixing Service configuration..."
                    # Create temporary kubeconfig
                    cat > /tmp/service-fix-\$\$ << EOF
apiVersion: v1
kind: Config
clusters:
- name: kubernetes
  cluster:
    server: https://172.31.45.168:6443
    insecure-skip-tls-verify: true
users:
- name: jenkins
  user:
    token: $K8S_TOKEN
contexts:
- name: jenkins-context
  context:
    cluster: kubernetes
    user: jenkins
    namespace: webapps
current-context: jenkins-context
EOF

                    export KUBECONFIG=/tmp/service-fix-\$\$

                    # Change to NodePort if LoadBalancer is pending
                    echo "🔄 Changing service to NodePort..."
                    kubectl patch svc boardgame-service -n webapps -p '{"spec":{"type":"NodePort"}}' ||
echo "Service patch failed or not needed"

                    # Wait for service to be ready
                    sleep 10

                    echo "✅ Service configuration updated!"
                    rm -f /tmp/service-fix-\$\$
                    """
                }
            }
        }
    }

    // ... your existing Deploy To Kubernetes stage

    stage('Verify Deployment with Fixed Permissions') {
        steps {
            script {
                withCredentials([string(credentialsId: 'k8-cred', variable: 'K8S_TOKEN')]) {
                    sh """
                    echo "🔍 Verifying deployment with fixed permissions..."

                    # Create temporary kubeconfig
                    cat > /tmp/verify-fixed-\$\$ << EOF
apiVersion: v1
kind: Config
clusters:
- name: kubernetes
  cluster:
    server: https://172.31.45.168:6443
    insecure-skip-tls-verify: true
users:
- name: jenkins
  user:
    token: $K8S_TOKEN
contexts:
- name: jenkins-context
  context:
    cluster: kubernetes
    user: jenkins
    namespace: webapps
current-context: jenkins-context
EOF

                        export KUBECONFIG=/tmp/verify-fixed-\$\$

                        echo "=== ✅ Enhanced Deployment Verification ==="
                        echo "=== Pods Status ==="
                        kubectl get pods -n webapps -o wide

                        echo "=== Services Status ==="
                        kubectl get svc -n webapps

                        echo "=== Application Logs (Now with proper permissions) ==="
                        kubectl logs -n webapps deployment/boardgame-deployment --tail=15 --prefix=true ||
echo "Logs might still be initializing"

                        echo "=== Application URLs ==="
                        kubectl get svc -n webapps -o wide

                        echo "=== Testing Application Health ==="
                        # Get pod name and test connectivity
                        POD_NAME=\$(kubectl get pods -n webapps -l app=boardgame -o
jsonpath='{.items[0].metadata.name}')
                        echo "Testing pod: \$POD_NAME"
```

```
            # Test if application is responding
            kubectl exec -n webapps \$POD_NAME -- wget -q -O- http://localhost:80 > /dev/null && echo "✅ Application is responding" || echo "⚠️ Application might be starting"

            # Get NodePort for access
            NODE_PORT=\$(kubectl get svc boardgame-service -n webapps -o jsonpath='{.spec.ports[0].nodePort}')
            echo "📱 Access your application at: http://<any-node-ip>:\$NODE_PORT"

            # Get node IPs
            echo "🌐 Available node IPs:"
            kubectl get nodes -o wide | awk '{print \$1, \$6, \$7}' | grep -v NAME

            rm -f /tmp/verify-fixed-\$\$
            """
          }
        }
      }
    }

    stage('Application Health Check') {
      steps {
        script {
          withCredentials([string(credentialsId: 'k8-cred', variable: 'K8S_TOKEN')]) {
            sh """
            echo "🏥 Performing application health check..."

            cat > /tmp/health-check-\$\$ << EOF
apiVersion: v1
kind: Config
clusters:
```

```
- name: kubernetes
  cluster:
    server: https://172.31.45.168:6443
    insecure-skip-tls-verify: true
users:
- name: jenkins
  user:
    token: $K8S_TOKEN
contexts:
- name: jenkins-context
  context:
    cluster: kubernetes
    user: jenkins
    namespace: webapps
current-context: jenkins-context
EOF

                export KUBECONFIG=/tmp/health-check-\$\$

                # Wait for application to be ready
                echo "⏳ Waiting for application to be ready..."
                kubectl rollout status deployment/boardgame-deployment -n webapps --timeout=300s

                # Test application endpoint
                echo "🔍 Testing application endpoint..."
                kubectl port-forward svc/boardgame-service -n webapps 8080:80 &
                PORT_FORWARD_PID=\$!
                sleep 5

                # Test the application
                if curl -f http://localhost:8080 > /dev/null 2>&1; then
```

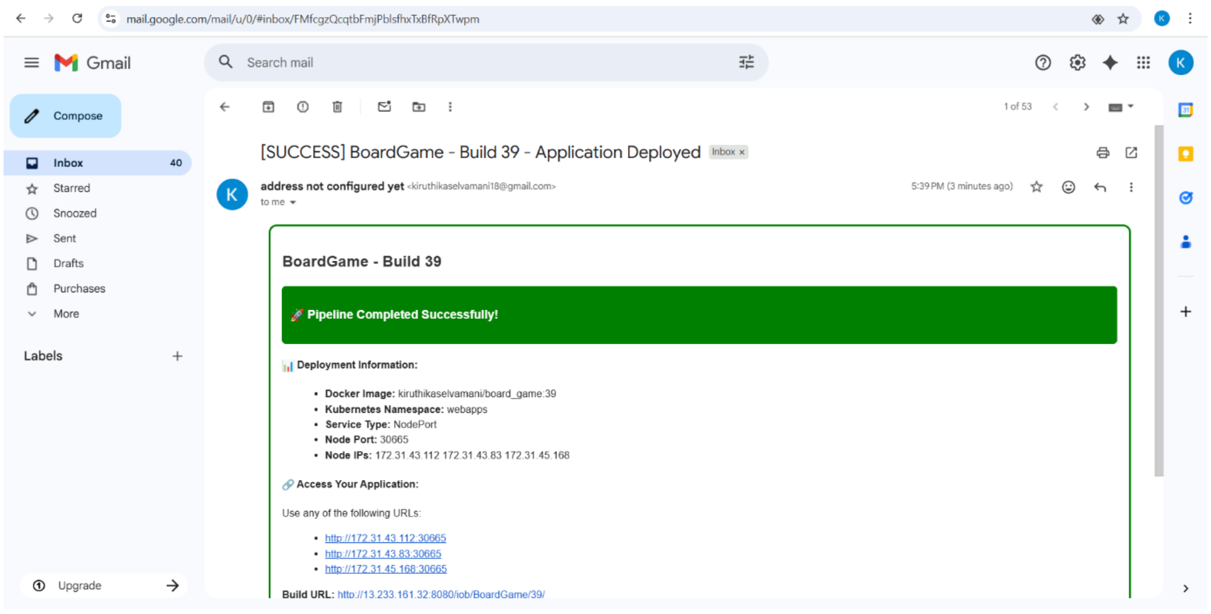
```
                    echo "✅ Application is healthy and responding!"
                  else
                    echo "⚠️ Application might be starting up..."
                  fi

                  # Kill port-forward
                  kill \$PORT_FORWARD_PID 2>/dev/null || true

                  rm -f /tmp/health-check-\$\$
                  """
                }
              }
            }
          }
        }

        post {
          always {
            script {
              // Enhanced email with access information
              withCredentials([string(credentialsId: 'k8-cred', variable: 'K8S_TOKEN')]) {
                sh """
                cat > /tmp/email-info-\$\$ << EOF
apiVersion: v1
kind: Config
clusters:
- name: kubernetes
  cluster:
    server: https://172.31.45.168:6443
    insecure-skip-tls-verify: true
users:
```

```
- name: jenkins
  user:
    token: $K8S_TOKEN
contexts:
- name: jenkins-context
  context:
    cluster: kubernetes
    user: jenkins
    namespace: webapps
current-context: jenkins-context
EOF

                    export KUBECONFIG=/tmp/email-info-\$\$

                    # Get application access info
                    NODE_PORT=\$(kubectl get svc boardgame-service -n webapps -o
jsonpath='{.spec.ports[0].nodePort}' 2>/dev/null || echo "unknown")
                    NODE_IPS=\$(kubectl get nodes -o
jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}' 2>/dev/null || echo
"unknown")

                    echo "NODE_PORT=\$NODE_PORT" > /tmp/app-info.txt
                    echo "NODE_IPS=\$NODE_IPS" >> /tmp/app-info.txt
                    """
                }

                def appInfo = readFile('/tmp/app-info.txt')
                def nodePort = appInfo.split('\n').find { it.startsWith('NODE_PORT=') }?.split('=')?.last() ?:
'unknown'
                def nodeIps = appInfo.split('\n').find { it.startsWith('NODE_IPS=') }?.split('=')?.last() ?:
'unknown'

                archiveArtifacts artifacts: 'target/*.jar, trivy-*.html', fingerprint: true
```

```
def body = """
<html>
<body style="font-family: Arial, sans-serif;">
<div style="border: 3px solid green; border-radius: 10px; padding: 15px; margin: 10px;">
  <h2 style="color: #333;">${env.JOB_NAME} - Build ${env.BUILD_NUMBER}</h2>
  <div style="background-color: green; color: white; padding: 10px; border-radius: 5px;">
    <h3>🚀 Pipeline Completed Successfully!</h3>
  </div>

  <h4>📊 Deployment Information:</h4>
  <ul>
    <li><strong>Docker Image:</strong> ${env.DOCKER_IMAGE}</li>
    <li><strong>Kubernetes Namespace:</strong> webapps</li>
    <li><strong>Service Type:</strong> NodePort</li>
    <li><strong>Node Port:</strong> ${nodePort}</li>
    <li><strong>Node IPs:</strong> ${nodeIps}</li>
  </ul>

  <h4>🔗 Access Your Application:</h4>
  <p>Use any of the following URLs:</p>
  <ul>
    ${nodeIps.split(' ').collect { ip -> "<li>http://${ip}:${nodePort}</li>" }.join('')}
  </ul>

  <p><strong>Build URL:</strong> <a href="${env.BUILD_URL}">${env.BUILD_URL}</a></p>
</div>
</body>
</html>
"""
```

```
            emailext (
                subject: "[SUCCESS] ${env.JOB_NAME} - Build ${env.BUILD_NUMBER} - Application Deployed",
                body: body,
                to: 'kiruthikaselvamani18@gmail.com',
                mimeType: 'text/html'
            )

            sh 'rm -f /tmp/email-info-* /tmp/app-info.txt'
        }
      }
    }
}
```

OUTPUT:



[SUCCESS] BoardGame - Build 39 - Application Deployed

address not configured yet <kiruthikaselvamani18@gmail.com>
to me

5:39 PM (3 minutes ago)

BoardGame - Build 39

🚀 **Pipeline Completed Successfully!**

📊 **Deployment Information:**

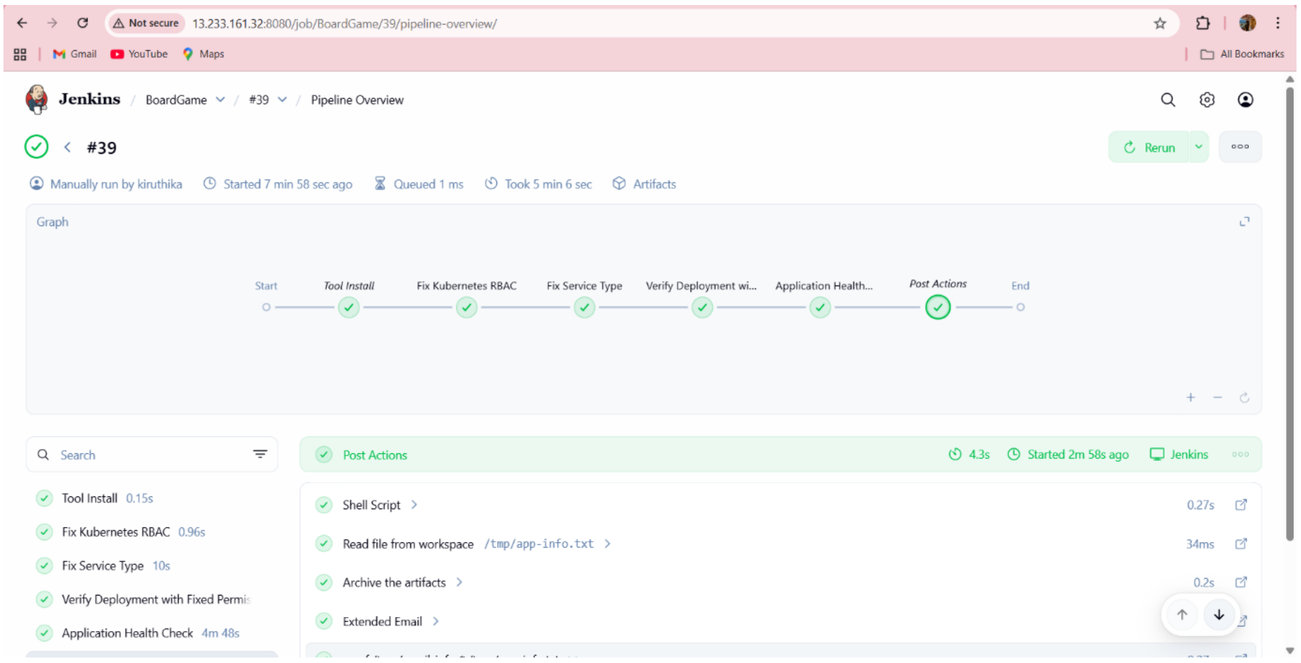
- **Docker Image:** kiruthikaselvamani/board_game:39
- **Kubernetes Namespace:** webapps
- **Service Type:** NodePort
- **Node Port:** 30665
- **Node IPs:** 172.31.43.112 172.31.43.83 172.31.45.168

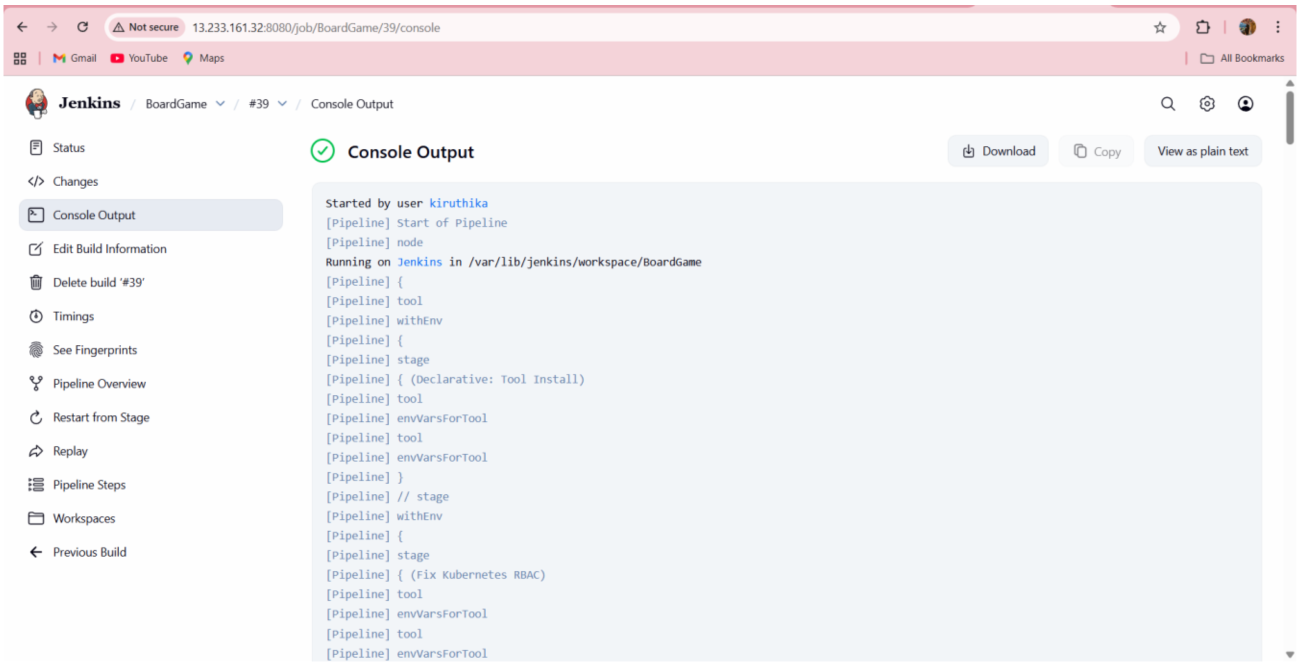
🔗 **Access Your Application:**

Use any of the following URLs:

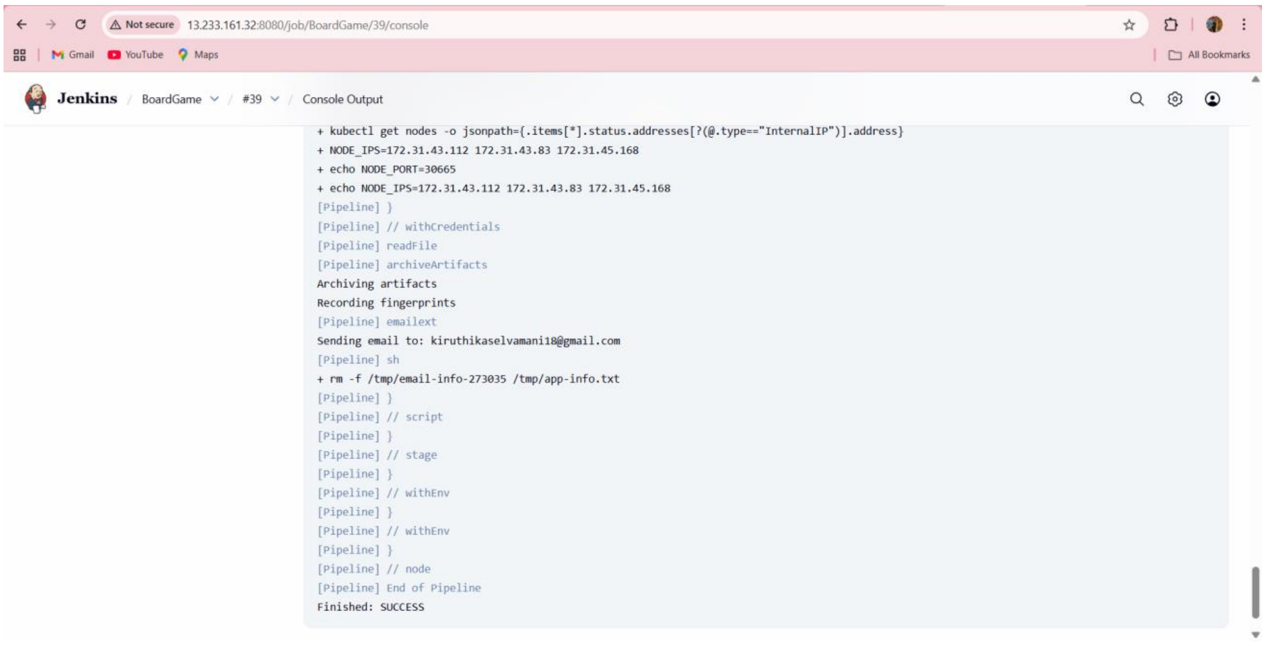
- http://172.31.43.112:30665
- http://172.31.43.83:30665
- http://172.31.45.168:30665

Build URL: http://13.233.161.32:8080/job/BoardGame/39/

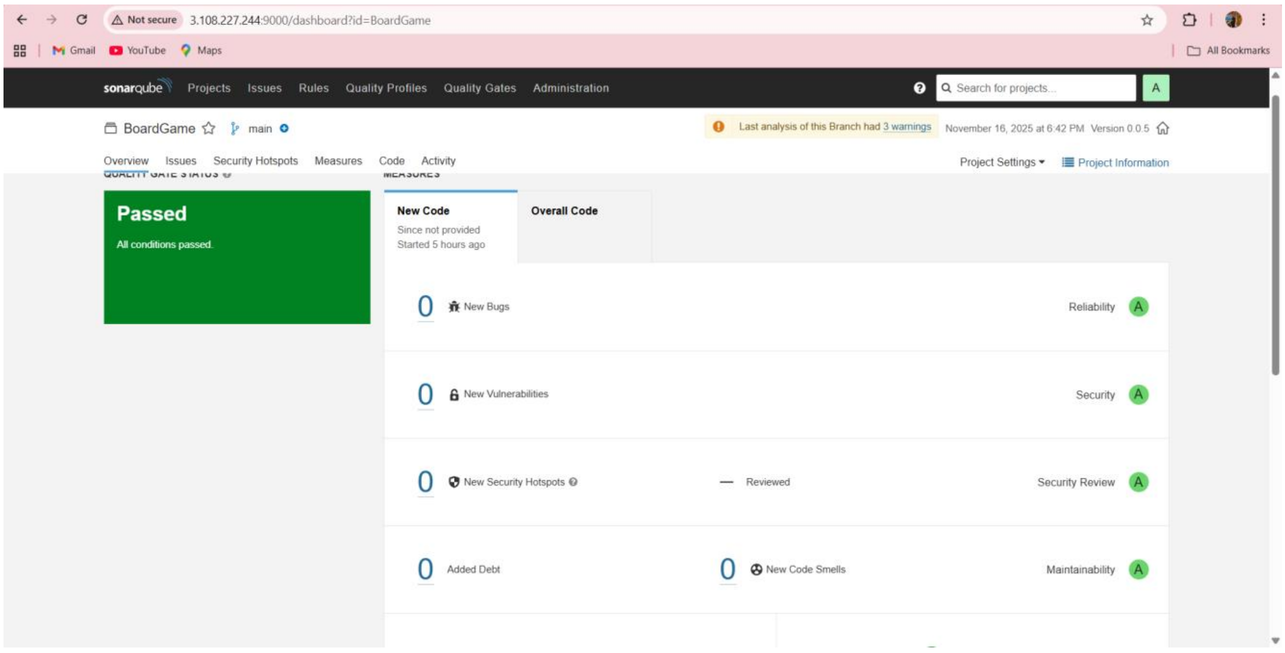




```
Started by user kiruthika
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/BoardGame
[Pipeline] {
[Pipeline] tool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Fix Kubernetes RBAC)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] tool
[Pipeline] envVarsForTool
```


Not secure 13.233.161.32:8080/job/BoardGame/39/console
Gmail
YouTube
Maps
All Bookmarks
Jenkins / BoardGame / #39 / Console Output

```
+ kubectl get nodes -o jsonpath={.items[*].status.addresses[?(@.type=="InternalIP")].address}
+ NODE_IPS=172.31.43.112 172.31.43.83 172.31.45.168
+ echo NODE_PORT=30665
+ echo NODE_IPS=172.31.43.112 172.31.43.83 172.31.45.168
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] readFile
[Pipeline] archiveArtifacts
Archiving artifacts
Recording fingerprints
[Pipeline] emailext
Sending email to: kiruthikaselvamani18@gmail.com
[Pipeline] sh
+ rm -f /tmp/email-info-273035 /tmp/app-info.txt
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Last analysis of this Branch had 3 warnings November 16, 2025 at 6:42 PM Version 0.0.5

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

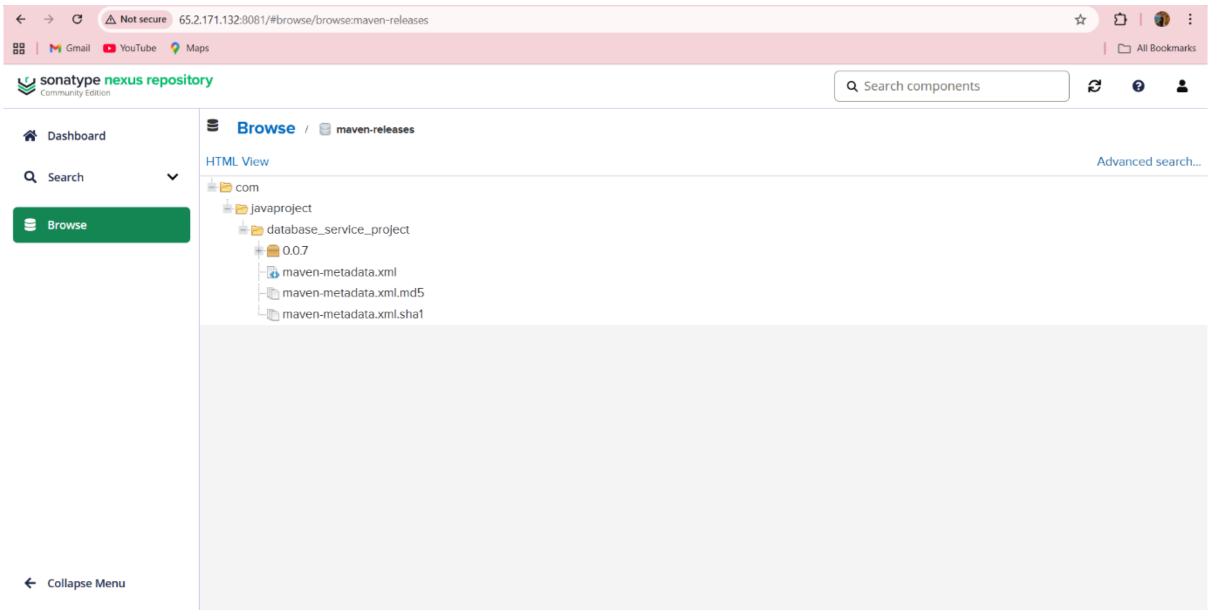
New Code Since not provided Started 5 hours ago | **Overall Code**

0 New Bugs — Reliability A

0 New Vulnerabilities — Security A

0 New Security Hotspots — Reviewed — Security Review A

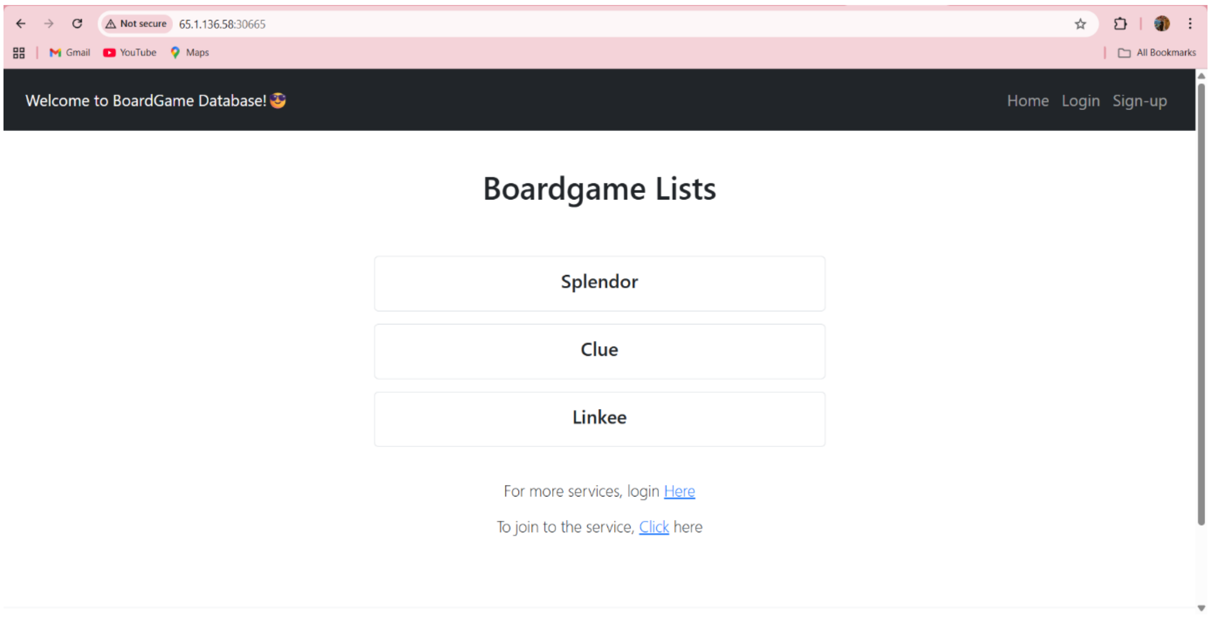
0 Added Debt — 0 New Code Smells — Maintainability A



```
com
  javaproject
    database_service_project
      0.0.7
      maven-metadata.xml
      maven-metadata.xml.md5
      maven-metadata.xml.sha1
```

Now we can able to access the application that are running in salve machine 1 & 2 check the ip address of both slave 1 & 2

Slave 1:



Boardgame Lists

Splendor

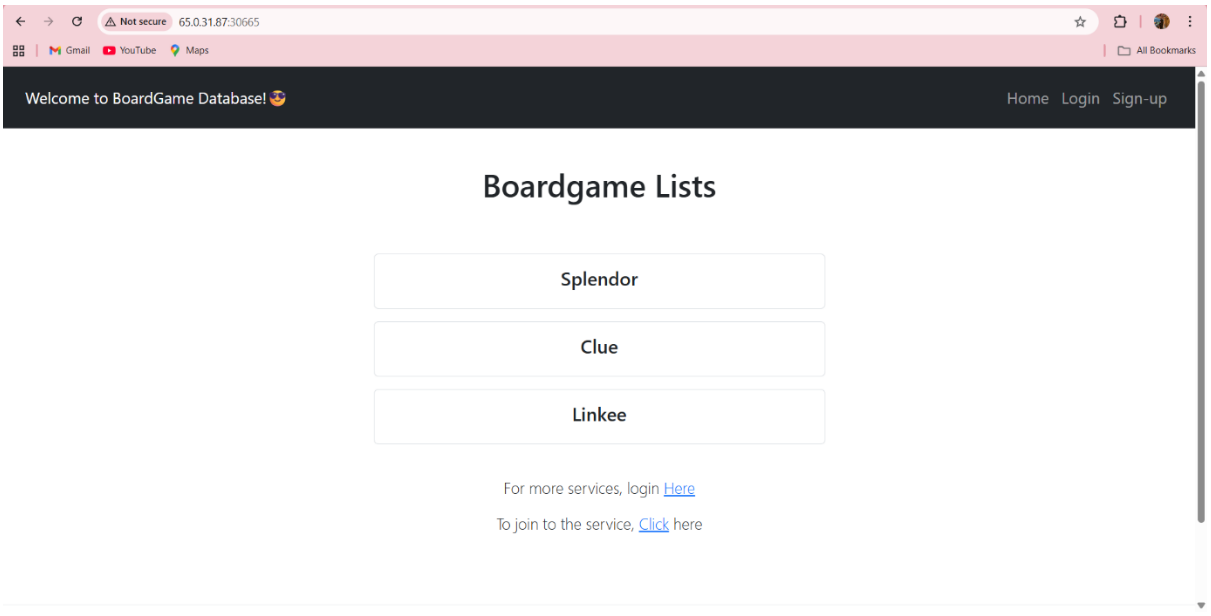
Clue

Linkee

For more services, login Here

To join to the service, Click here

Slave 2:



Welcome to BoardGame Database! 😎

Boardgame Lists

Splendor

Clue

Linkee

For more services, login Here

To join to the service, Click here

PHASE 4:

Monitoring

Launch an EC2 instance with the t2 medium and start setup installation Prometheus and Grafana

Before starting installation update the package

Sudo apt update

1. Links to download Prometheus, Node_ Exporter & black Box exporter

https://prometheus.io/download/

Download the latest prometheus

Copy the link from the above official document website and download in your local machine
wget
https://github.com/prometheus/prometheus/releases/download/v3.7.3/prometheus-3.7.3.linux-amd64.tar.gz

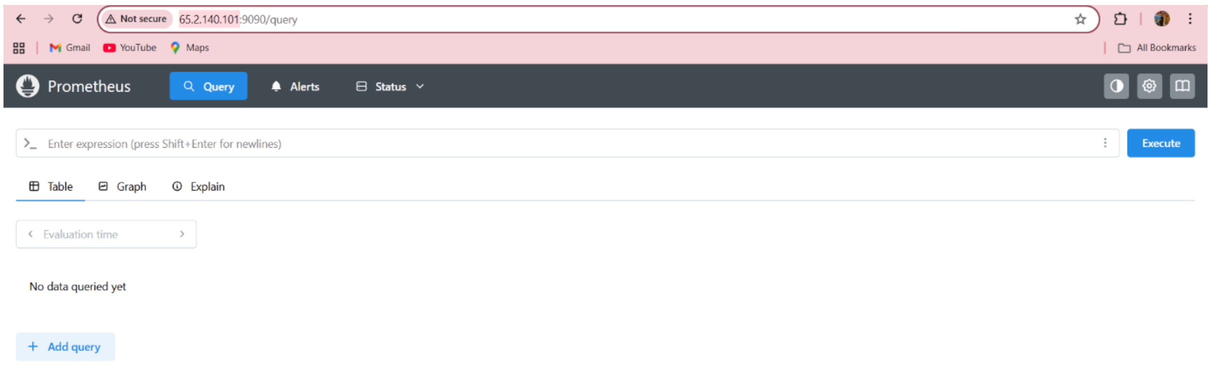
Extract the downloaded archive

tar -xvf prometheus-2.5.4.linux-
amd64.tar.gz(cd prometheus-2.5.4.linux-)
amd64

Now given we be can able to ver the executable file i the name of prometheus
To start prometheus run the executable script in the name prometheus / prometheus.sh

&

Now we can able to access the prometheus...



2. **Links to download Grafana** https://grafana.com/grafana/download

Refer the official documentation and install the latest version

sudo apt-get install -y adduser libfontconfig1 musl

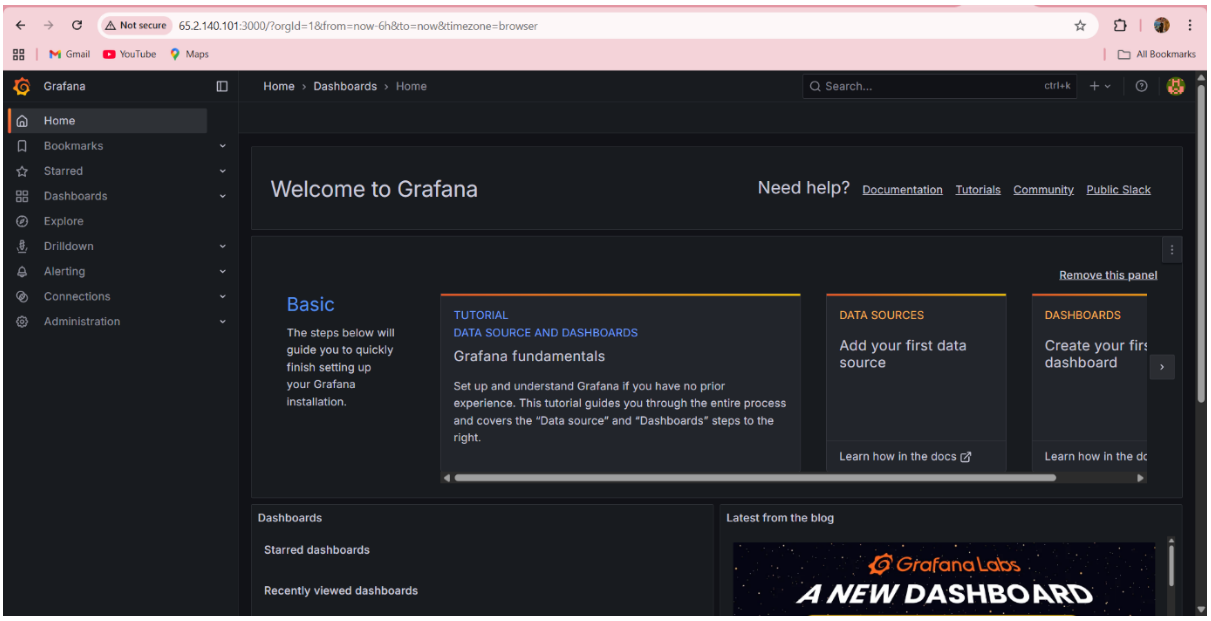
wget https://grafana.com/enterprise/release/grafana-enterprise_11.1.4_amd64.deb

sudo dpkg -i grafana-enterprise_11.1.4_amd64.deb

You can start grafana-server by executing

sudo /bin/systemctl start grafana-server

Grafana is running now, and we can connect to it at http://localhost:3000
password is admin / admin



Not secure 65.2.140.101:3000/?orgId=1&from=now-6h&to=now&timezone=browser
Gmail
YouTube
Maps
All Bookmarks
Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
Alerting
Connections
Administration
Home › Dashboards › Home
Search...
ctrl+k
Welcome to Grafana
Need help?
Documentation
Tutorials
Community
Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
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.
DATA SOURCES
Add your first data source
Learn how in the docs
DASHBOARDS
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Recently viewed dashboards
Latest from the blog
Grafana Labs
A NEW DASHBOARD

3. https://github.com/prometheus/blackbox_exporter

download the black box exporter

wget

https://github.com/prometheus/blackbox_exporter/releases/download/v0.27.0/blackbox_exporter-0.27.0.linux-amd64.tar.gz

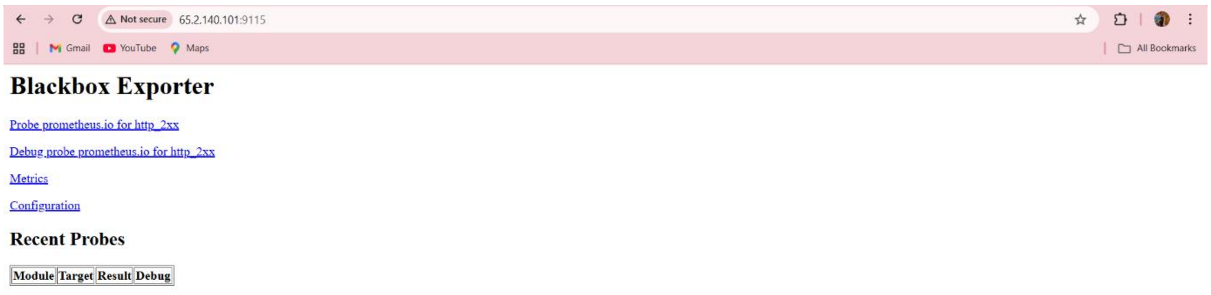
Now Extract the tar file of blackbox exporter

To start the black box exporter

cd bischlau_exporter@2.5.8.linux-amd64/

Now give it we can able to see the executable file in the name of blackbox.To start bischlau run the executable script in the name blackbox:

./blackbox_exporter &



Blackbox Exporter

Probe prometheus.io for http_2xx

Debug probe prometheus.io for http_2xx

Metrics

Configuration

Recent Probes

Module	Target	Result	Debug

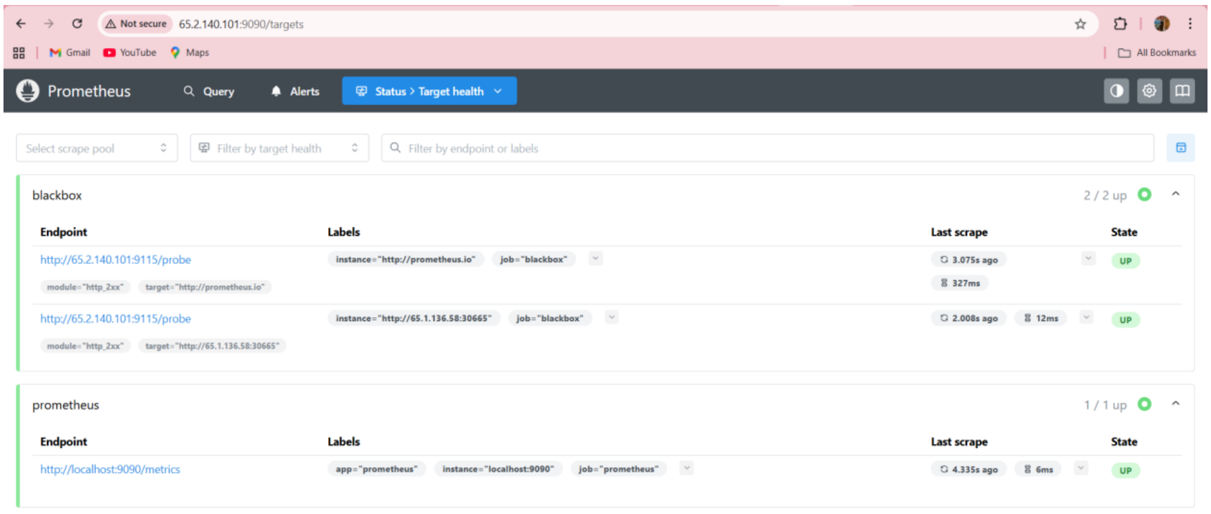
Prometheus configurations

Vi Prometheus.yml

```
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"]
        # The label name is added as a label `label_name=<label_value>` to any timeseries scraped from
this config.
        labels:
          app: "prometheus"

  - job_name: 'blackbox'
    metrics_path: /probe
    params:
```

```
    module: [http_2xx]
  static_configs:
    - targets:
      - http://prometheus.io
      - http://65.1.136.58:30665
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      replacement: 65.2.140.101:9115
```

now need to restart the Prometheus



blackbox 2 / 2 up

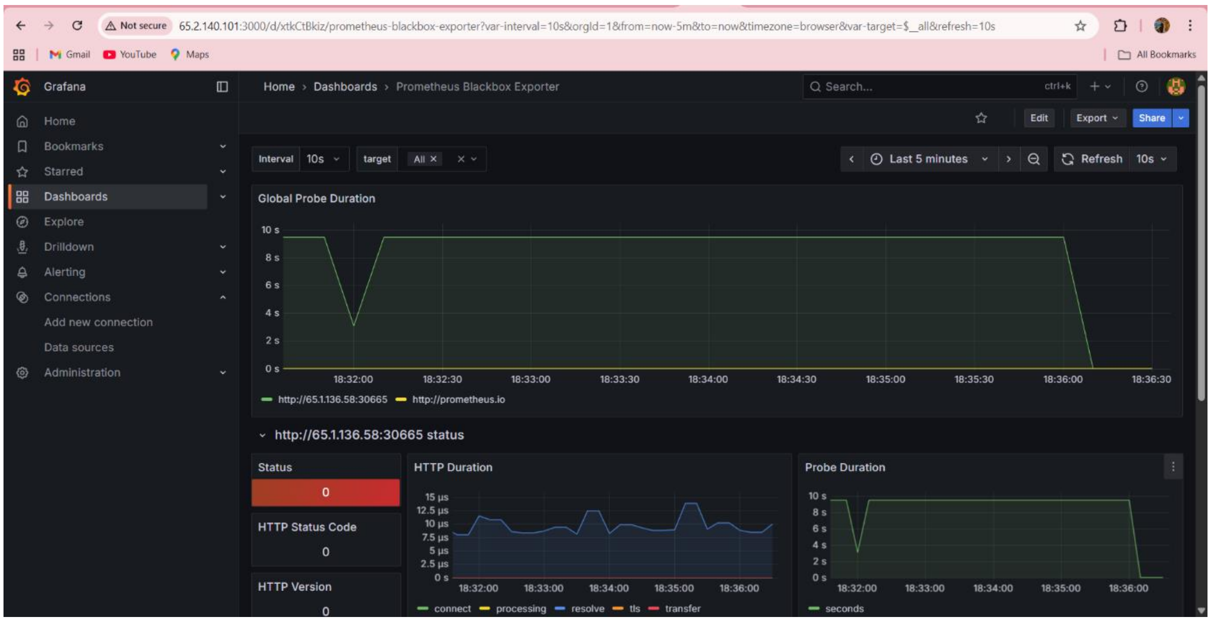
Endpoint	Labels	Last scrape	State
http://65.2.140.101:9115/probe module="http_2xx" target="http://prometheus.io"	instance="http://prometheus.io" job="blackbox"	3.075s ago 327ms	UP
http://65.2.140.101:9115/probe module="http_2xx" target="http://65.1.136.58:30665"	instance="http://65.1.136.58:30665" job="blackbox"	2.008s ago 12ms	UP

prometheus 1 / 1 up

Endpoint	Labels	Last scrape	State
http://localhost:9090/metrics	app="prometheus" instance="localhost:9090" job="prometheus"	4.335s ago 6ms	UP

Now need to add promethus as our datasource in Grafana

Then import dashboard give id 7587 to create black box exporter dashboard



Let us monitor the Jenkins machine metrics by node exporter install Prometheus metrics plugins in Jenkins

Let us download the node exporter in Jenkins machine

Wget

https://github.com/prometheus/node_exporter/releases/download/v1.10.2/node_exporter-1.10.2.linux-amd64.tar.gz

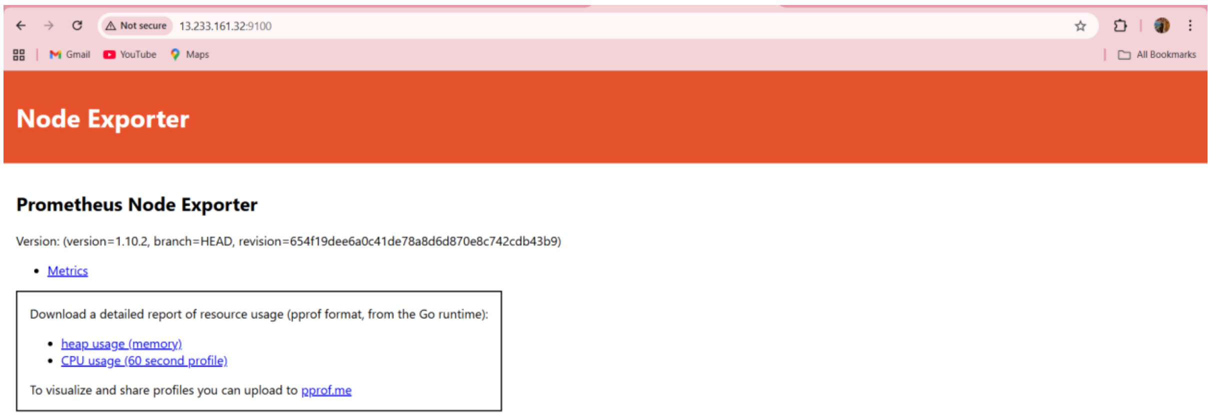
Extract the tar file

tar -xzf node_exporter-1.7.0.linux-amd64.tar.gz

Move to appropriate directory

sudo mv node_exporter-1.7.0.linux-amd64/node_exporter /usr/local/bin/

./node_exporter &


13.233.161.32:9100

Node Exporter

Prometheus Node Exporter

Version: (version=1.10.2, branch=HEAD, revision=654f19dee6a0c41de78a8d6d870e8c742cdb43b9)

- Metrics

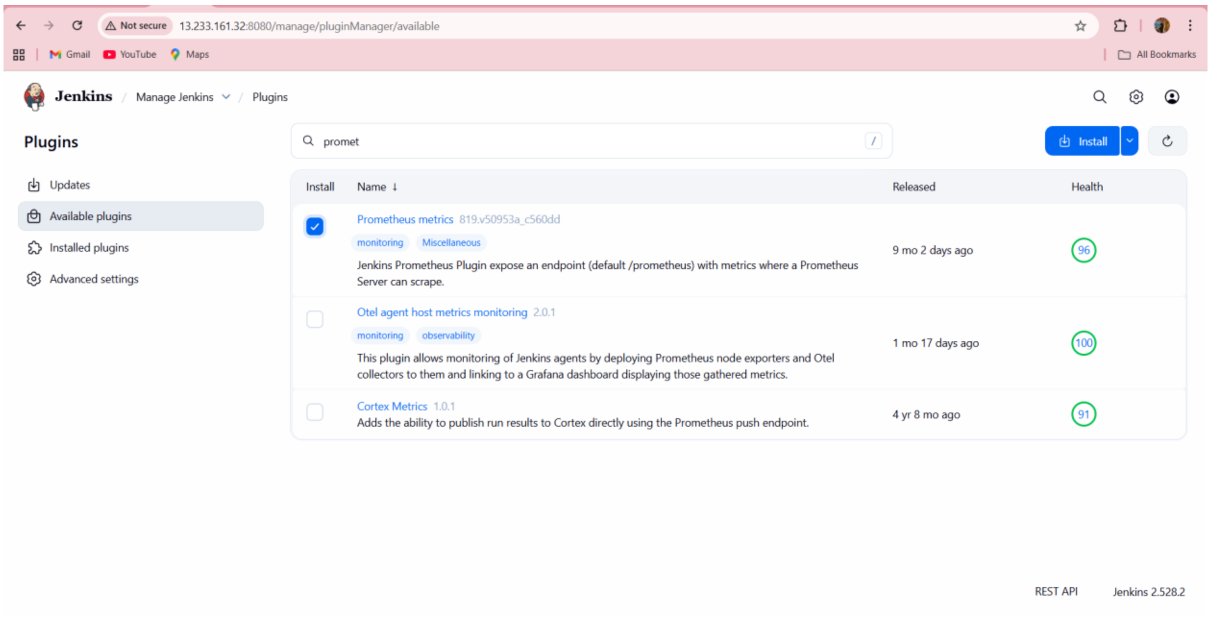
Download a detailed report of resource usage (pprof format, from the Go runtime):

- heap usage (memory)
- CPU usage (60 second profile)

To visualize and share profiles you can upload to pprof.me

Now nodeExporter can be accessible through the brower – port :9100

Make sure the below one was configured in manage Jenkins---→ system


13.233.161.32:8080/manage/pluginManager/available

Jenkins / Manage Jenkins / Plugins

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings

promet

Install	Name ↓	Released	Health
✓	Prometheus metrics 819.v50953a_c560dd monitoring Miscellaneous Jenkins Prometheus Plugin expose an endpoint (default /prometheus) with metrics where a Prometheus Server can scrape.	9 mo 2 days ago	96
	Otel agent host metrics monitoring 2.0.1 monitoring observability This plugin allows monitoring of Jenkins agents by deploying Prometheus node exporters and Otel collectors to them and linking to a Grafana dashboard displaying those gathered metrics.	1 mo 17 days ago	100
	Cortex Metrics 1.0.1 Adds the ability to publish run results to Cortex directly using the Prometheus push endpoint.	4 yr 8 mo ago	91

REST API Jenkins 2.528.2

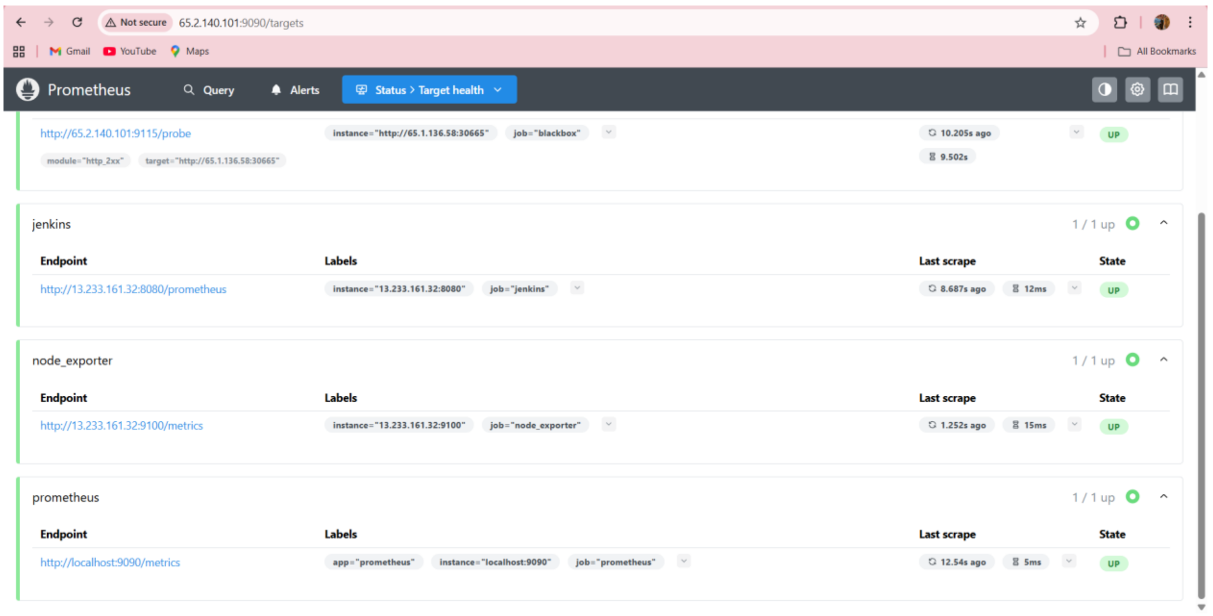
Now need to edit the yaml file ofprometheus

Vi Prometheus.yaml

```
scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
         - targets: ['13.233.161.32:9100']

  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
         - targets: ['13.233.161.32:8080']
```

After update restart Prometheus:

```
pgrep Prometheus
kill id
./Prometheus &
```



Endpoint	Labels	Last scrape	State
http://65.2.140.101:9115/probe	instance="http://65.1.136.58:30665" job="blackbox" module="http_2xx" target="http://65.1.136.58:30665"	10.205s ago 9.502s	UP

jenkins 1 / 1 up

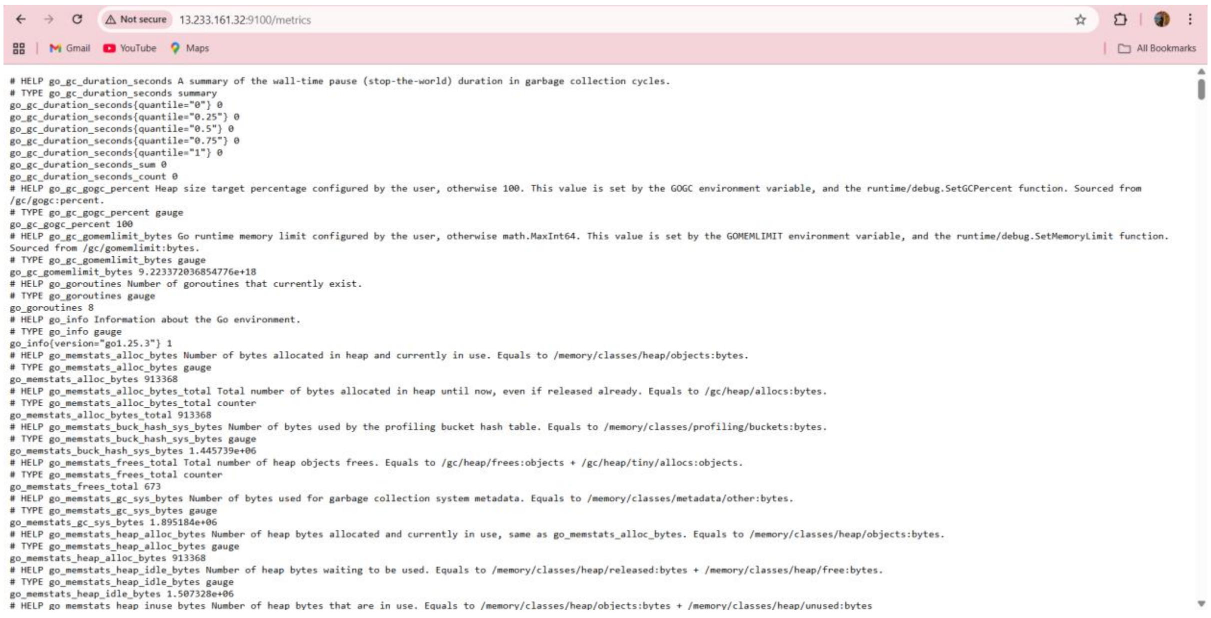
Endpoint	Labels	Last scrape	State
http://13.233.161.32:8080/prometheus	instance="13.233.161.32:8080" job="jenkins"	8.687s ago 12ms	UP

node_exporter 1 / 1 up

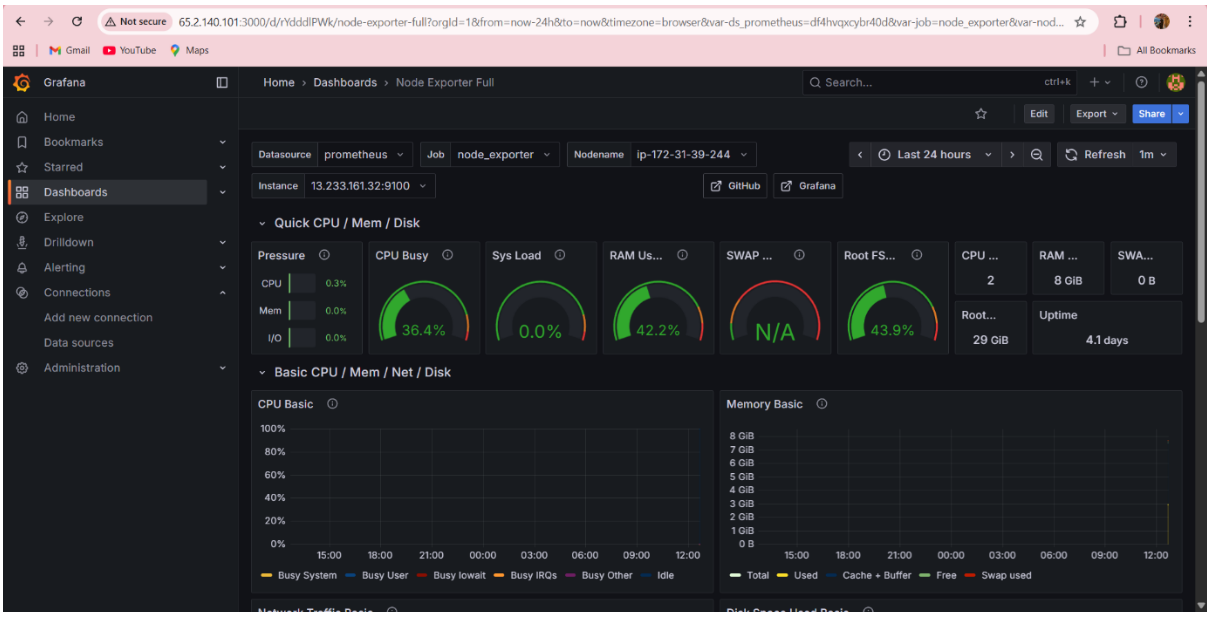
Endpoint	Labels	Last scrape	State
http://13.233.161.32:9100/metrics	instance="13.233.161.32:9100" job="node_exporter"	1.252s ago 15ms	UP

prometheus 1 / 1 up

Endpoint	Labels	Last scrape	State
http://localhost:9090/metrics	app="prometheus" instance="localhost:9090" job="prometheus"	12.54s ago 5ms	UP



```
# HELP go_gc_duration_seconds A summary of the wall-time pause (stop-the-world) duration in garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_gc_gogc_percent Heap size target percentage configured by the user, otherwise 100. This value is set by the GOGC environment variable, and the runtime/debug.SetGCPercent function. Sourced from
/gc/gogc:percent.
# TYPE go_gc_gogc_percent gauge
go_gc_gogc_percent 100
# HELP go_gc_gomemlimit_bytes Go runtime memory limit configured by the user, otherwise math.MaxInt64. This value is set by the GOMEMLIMIT environment variable, and the runtime/debug.SetMemoryLimit function.
Sourced from /gc/gomemlimit:bytes.
# TYPE go_gc_gomemlimit_bytes gauge
go_gc_gomemlimit_bytes 9.223372036854776e+18
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 8
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.25.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated in heap and currently in use. Equals to /memory/classes/heap/objects:bytes.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 913368
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated in heap until now, even if released already. Equals to /gc/heap/allocs:bytes.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 913368
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table. Equals to /memory/classes/profiling/buckets:bytes.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.445739e+06
# HELP go_memstats_frees_total Total number of heap objects frees. Equals to /gc/heap/frees:objects + /gc/heap/tiny/allocs:objects.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 673
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata. Equals to /memory/classes/metadata/other:bytes.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 1.895184e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and currently in use, same as go_memstats_alloc_bytes. Equals to /memory/classes/heap/objects:bytes.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 913368
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used. Equals to /memory/classes/heap/released:bytes + /memory/classes/heap/free:bytes.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 1.507328e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use. Equals to /memory/classes/heap/objects:bytes + /memory/classes/heap/unused:bytes
```



Not secure
65.2.140.101:3000/d/rYdddlPWk/node-exporter-full?orgId=1&from=now-24h&to=now&timezone=browser&var-ds_prometheus=df4hvqxcybr40d&var-job=node_exporter&var-nod...
Gmail
YouTube
Maps
All Bookmarks
Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
Alerting
Connections
Add new connection
Data sources
Administration
Home › Dashboards › Node Exporter Full
Search...
ctrl+k
Edit
Export
Share
Datasource prometheus
Job node_exporter
Nodename ip-172-31-39-244
Instance 13.233.161.32:9100
Last 24 hours
Refresh 1m
GitHub
Grafana
Quick CPU / Mem / Disk
Pressure
CPU 0.3%
Mem 0.0%
I/O 0.0%
CPU Busy 36.4%
Sys Load 0.0%
RAM Us... 42.2%
SWAP ... N/A
Root FS... 43.9%
CPU ... 2
RAM ... 8 GiB
SWA... 0 B
Root... 29 GiB
Uptime 4.1 days
Basic CPU / Mem / Net / Disk
CPU Basic
100%
80%
60%
40%
20%
0%
15:00
18:00
21:00
00:00
03:00
06:00
09:00
12:00
Busy System
Busy User
Busy Iowait
Busy IRQs
Busy Other
Idle
Memory Basic
8 GiB
7 GiB
6 GiB
5 GiB
4 GiB
3 GiB
2 GiB
1 GiB
0 B
15:00
18:00
21:00
00:00
03:00
06:00
09:00
12:00
Total
Used
Cache + Buffer
Free
Swap used