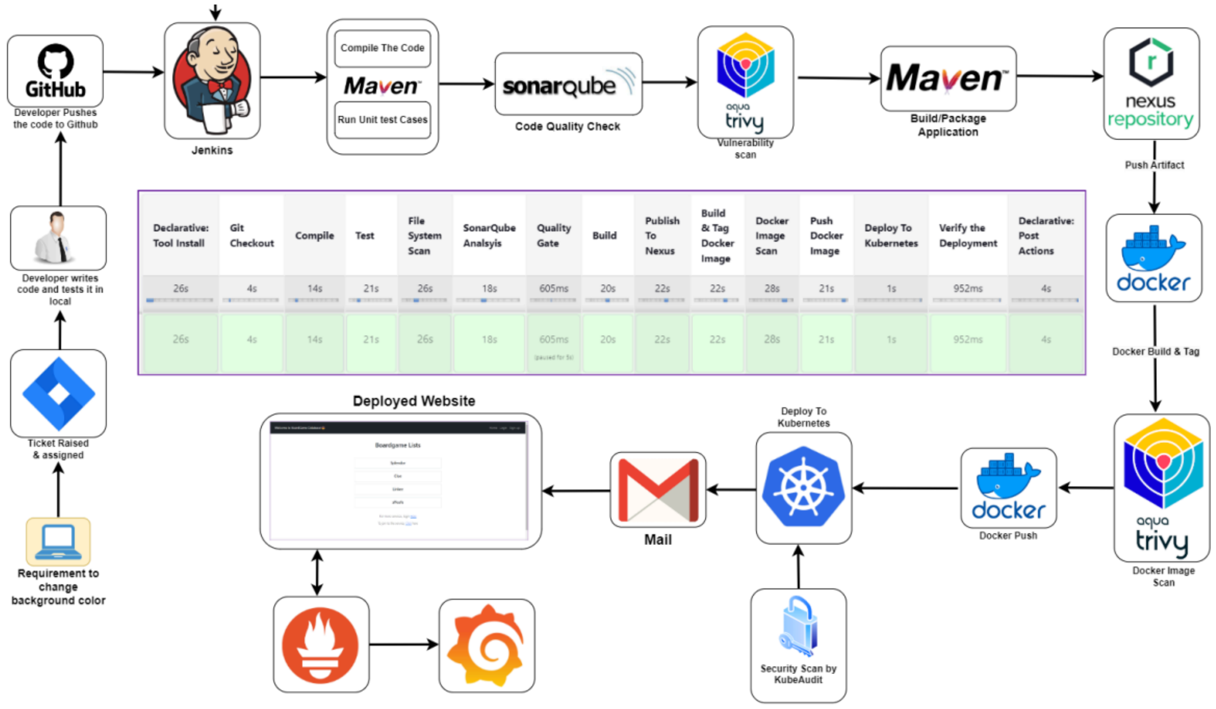
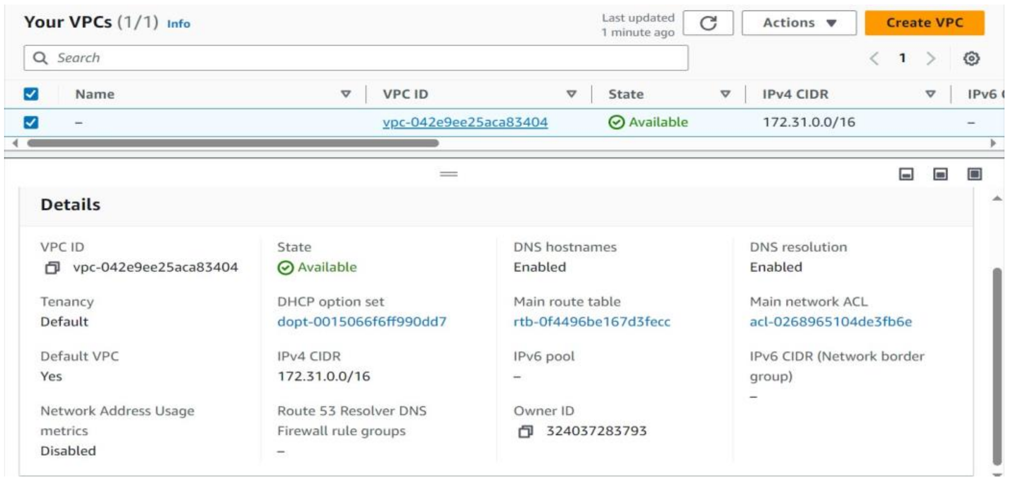
DevOps



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

Setup Environment on AWS:

1. Used the default VPC



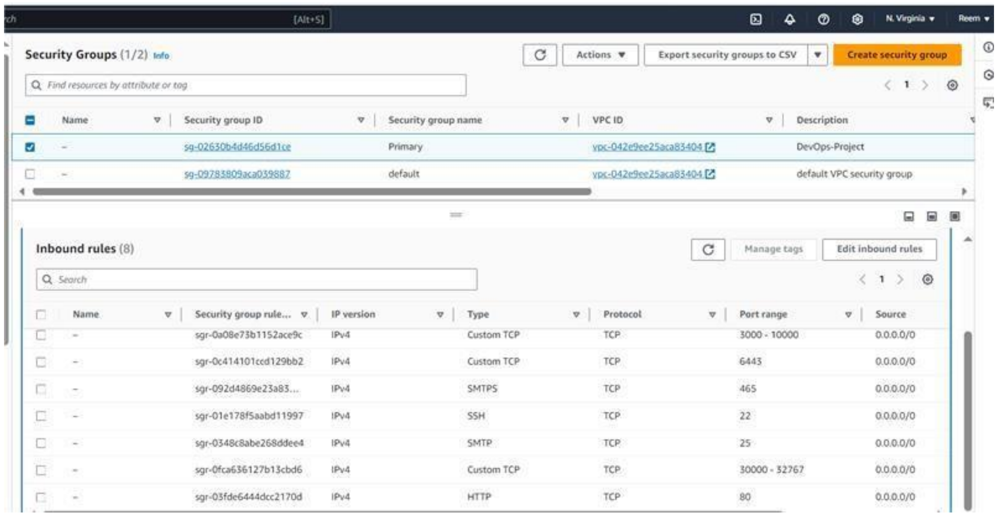
Your VPCs (1/1)

Name	VPC ID	State	IPv4 CIDR	IPv6
–	vpc-042e9ee25aca83404	Available	172.31.0.0/16	–

Details

VPC ID vpc-042e9ee25aca83404	State Available	DNS hostnames Enabled	DNS resolution Enabled
Tenancy Default	DHCP option set dopt-0015066f6ff990dd7	Main route table rtb-0f4496be167d3fecc	Main network ACL acl-0268965104de3fb6e
Default VPC Yes	IPv4 CIDR 172.31.0.0/16	IPv6 pool –	IPv6 CIDR (Network border group) –
Network Address Usage metrics Disabled	Route 53 Resolver DNS Firewall rule groups –	Owner ID 324037283793	

Configured Security group as shown:



Security Groups (1/2)

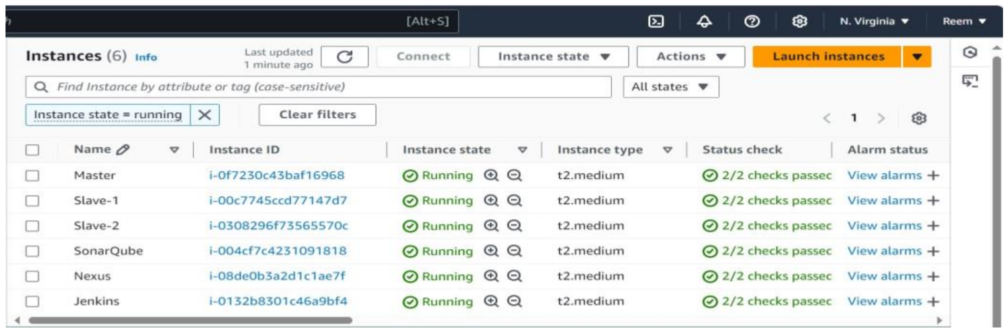
Name	Security group ID	Security group name	VPC ID	Description
–	sg-02630b4d46d56d1ce	Primary	vpc-042e9ee25aca83404	DevOps-Project
–	sg-09783809aca039887	default	vpc-042e9ee25aca83404	default VPC security group

Inbound rules (8)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source
–	sgr-0a08e73b1152ace9c	IPv4	Custom TCP	TCP	3000 - 10000	0.0.0.0/0
–	sgr-0c414101ccd129bb2	IPv4	Custom TCP	TCP	6443	0.0.0.0/0
–	sgr-092d4869e23a83...	IPv4	SMTPS	TCP	465	0.0.0.0/0
–	sgr-01e178f5aabd11997	IPv4	SSH	TCP	22	0.0.0.0/0
–	sgr-0348c8abe268ddee4	IPv4	SMTP	TCP	25	0.0.0.0/0
–	sgr-0fca636127b13cbd6	IPv4	Custom TCP	TCP	30000 - 32767	0.0.0.0/0
–	sgr-03fde6444dcc2170d	IPv4	HTTP	TCP	80	0.0.0.0/0

2. Configure the security group to allow (HTTP, HTTPS, SMTP,SMTPS, SSH) from our IP address.
3. Configure Range 300000-32767 for Deployment of apps, Port 6443 for Set up Kubernetes cluster and Port 465 to send email notifications from Jenkins to our emails

Created 6 Ubuntu EC2 instances in AWS:



Name	Instance ID	Instance state	Instance type	Status check	Alarm status
Master	i-0f7230c43baf16968	Running	t2.medium	2/2 checks passec	View alarms
Slave-1	i-00c7745ccd77147d7	Running	t2.medium	2/2 checks passec	View alarms
Slave-2	i-0308296f73565570c	Running	t2.medium	2/2 checks passec	View alarms
SonarQube	i-004cf7c4231091818	Running	t2.medium	2/2 checks passec	View alarms
Nexus	i-08de0b3a2d1c1ae7f	Running	t2.medium	2/2 checks passec	View alarms
Jenkins	i-0132b8301c46a9bf4	Running	t2.medium	2/2 checks passec	View alarms

Setup K8-Cluster using kubeadm K8 Version-->1.31:

4. One Master Node and 2 Slaves.
5. Update System Packages [On Master & Worker Node]
   a. sudo apt get update
6. Install Docker [On Master & Worker Node]:
   a. sudo apt install docker.io -y
   b. sudo chmod 666 /var/run/docker.sock
7. Install Required Dependencies for Kubernetes [On Master & Worker Node]
   a. sudo apt-get install -y apt-transport-https ca-certificates curl gnupg
   b. sudo mkdir -p -m 755 /etc/apt/keyrings
8. Add Kubernetes Repository and GPG Key [On Master & Worker Node]
   a. curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.31/deb/Release.key |

sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt

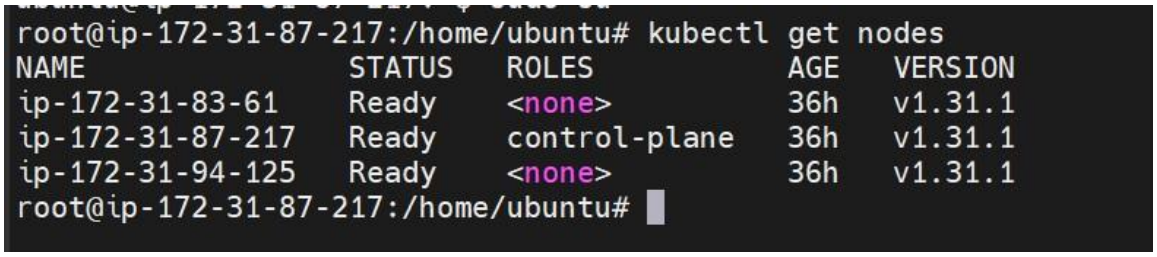
keyring.gpg echo 'deb

[signedby=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]

https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee

/etc/apt/sources.list.d/kubernetes.list

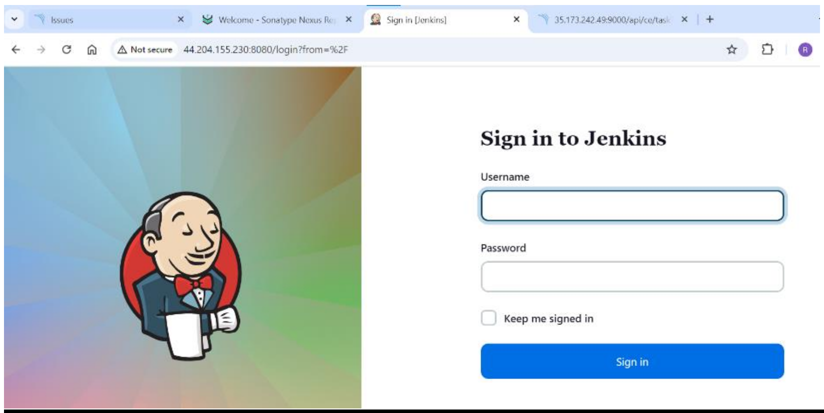
9. Update Package List [On Master & Worker Node]
    a. Sudo apt update
10. Install Kubernetes Components [On Master & Worker Node]
    a. Sudo apt-get install -y kubelet kubeadm kubectl
11. Initialize Kubernetes Master Node [On MasterNode] sudo
    a. kubeadm init --pod-network-cidr=10.244.0.0/16
        i. After running the above command then our vm will acts as master node and it will generate token to connect this with slave node -copy the token and run the command in slave machines 1 & 2
12. Configure Kubernetes Cluster [On MasterNode]
    a. mkdir -p $HOME/.kube
    b. sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config
13. Deploy Networking Solution (Calico) [On MasterNode]
    a. kubectl apply -f https://docs.projectcalico.org/v3.20/manifests/calico.yaml
14. Deploy Ingress Controller (NGINX) [On MasterNode]
    a. kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controllerv0.49.0/deploy/static/provider/baremetal/deploy.yaml

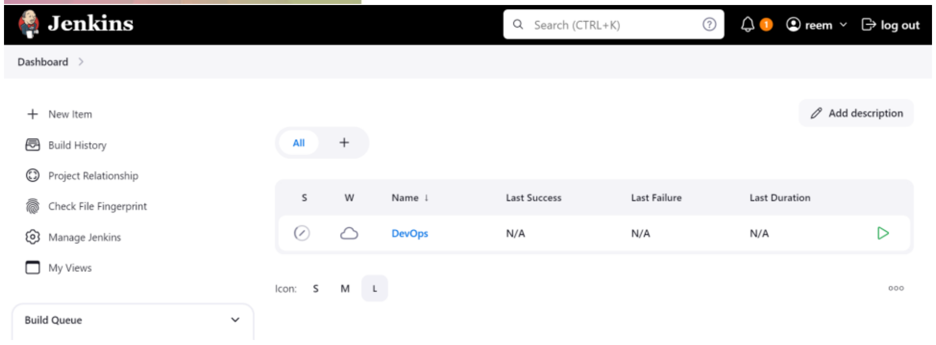


```
root@ip-172-31-87-217:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES           AGE   VERSION
ip-172-31-83-61    Ready    <none>          36h   v1.31.1
ip-172-31-87-217   Ready    control-plane   36h   v1.31.1
ip-172-31-94-125   Ready    <none>          36h   v1.31.1
root@ip-172-31-87-217:/home/ubuntu#
```

Installing Jenkins on Ubuntu:

15. Install OpenJDK 17 JRE Headless
    a. sudo apt install openjdk-17-jre-headless -y
16. Download Jenkins GPG key
    a. sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
       https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
17. Add Jenkins repository to package manager sources
    a. 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
18. Now we can able to access Jenkins: using the public ip address
    http://44.204.155.230:8080/

    a. 
Issues
Welcome - Sonatype Nexus Re
Sign in [Jenkins]
35.173.242.49:9000/api/ce/task
Not secure
44.204.155.230:8080/login?from=%2F
Sign in to Jenkins
Username
Password
Keep me signed in
Sign in

    b. 

S	W	Name ↓	Last Success	Last Failure	Last Duration
		DevOps	N/A	N/A	N/A

Install docker and trivy on Jenkins machine:

19. Update package manager repositories
    a. sudo apt-get update
20. Install necessary dependencies
    a. sudo apt-get install -y ca-certificates curl
21. Create directory for Docker GPG key
    a. sudo install -m 0755 -d /etc/apt/keyrings
22. Download Docker's GPG key
    a. sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
23. Change the permissions for the key
    a. sudo chmod 777 /etc/apt/keyrings/docker.asc
24. Add Docker repository to Apt sources
    a. 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

Trivy Installation Steps:

25. sudo apt-get install
26. wget apt-transport-https gnupg lsb-release
27. wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
28. sudo apt-get update
29. sudo apt-get install trivy -y

SetUp Nexus:

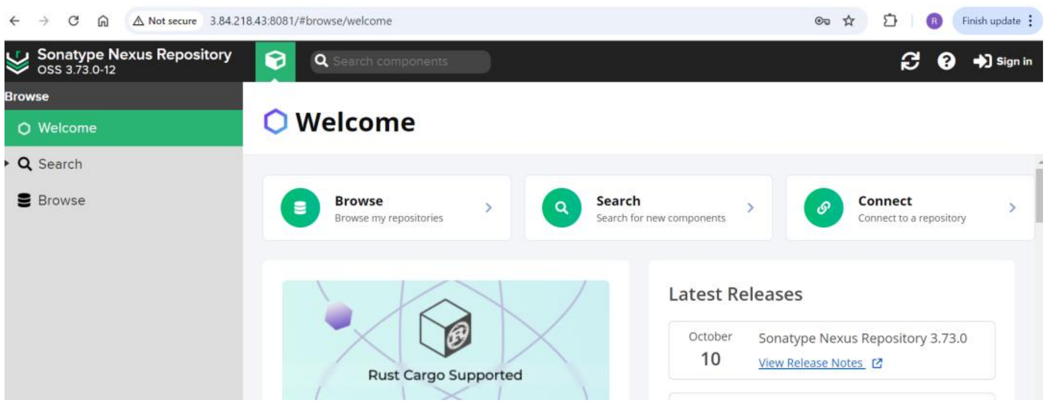
30. Update package manager repositories
    a. sudo apt-get update
31. Install necessary dependencies
    a. sudo apt-get install -y ca-certificates curl
32. Create directory for Docker GPG key

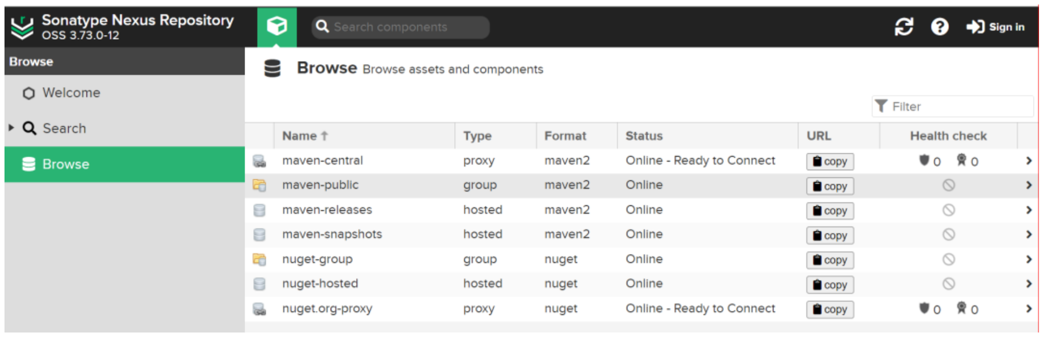
    a. sudo install -m 0755 -d /etc/apt/keyrings # Download Docker's GPG key sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

33. Change permissions for the key
    a. sudo chmod 777 /etc/apt/keyrings/docker.asc
34. Add Docker repository to Apt sources
35. 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
36. Update package manager repositories
    a. sudo apt-get update sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Create Nexus using docker container:

37. Create a Docker container running Nexus and exposing it on port 8081
    a. docker run -d --name nexus -p 8081:8081 sonatype/nexus3:latest
38. Then we can access nexus at http://3.84.218.43:8081



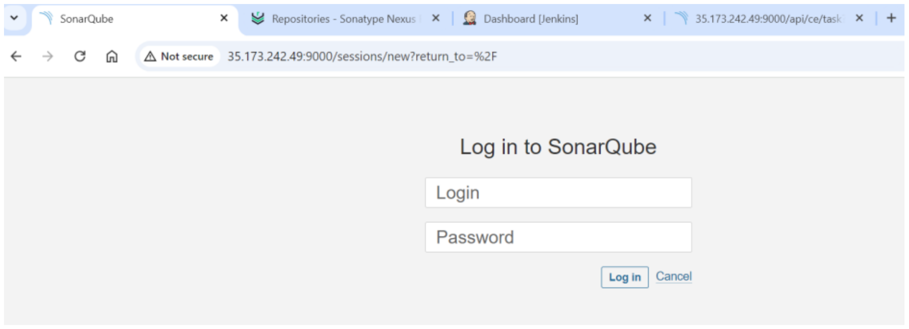


Name	Type	Format	Status	URL	Health check
maven-central	proxy	maven2	Online - Ready to Connect	copy	0 0
maven-public	group	maven2	Online	copy	⊘
maven-releases	hosted	maven2	Online	copy	⊘
maven-snapshots	hosted	maven2	Online	copy	⊘
nuget-group	group	nuget	Online	copy	⊘
nuget-hosted	hosted	nuget	Online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

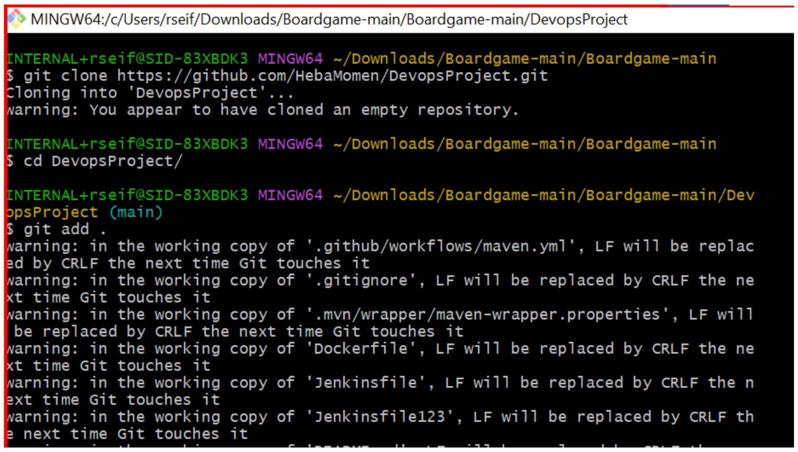
SetUp SonarQube:

39. Update package manager repositories
    a. sudo apt-get update
40. Install necessary dependencies
    a. sudo apt-get install -y ca-certificates curl
41. Create directory for Docker GPG key
    a. sudo install -m 0755 -d /etc/apt/keyrings
42. Download Docker's GPG key
    a. sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
43. Change permissions for the key
    a. sudo chmod 777 /etc/apt/keyrings/docker.asc
44. 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
45. Update package manager repositories
    a. sudo apt-get update
    b. sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Create Sonarqube Docker container:

46. Download the sonarqube:lts-community Docker image from Docker and 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).
    a. docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
47. Access SonarQube through http://35.173.242.49:9000
    a. 
SonarQube
Repositories - Sonatype Nexus
Dashboard [Jenkins]
35.173.242.49:9000/api/ce/task
Not secure
35.173.242.49:9000/sessions/new?return_to=%2F
Log in to SonarQube
Login
Password
Log in
Cancel

Create a private Git repository:

48. generate a personal access token
49. Create a Private Git Repository
50. Generate a Personal Access Token.
51. Clone the Repository Locally using the following command,
    a. git clone https://github.com/HebaMomen/DevopsProject.git
    b. 

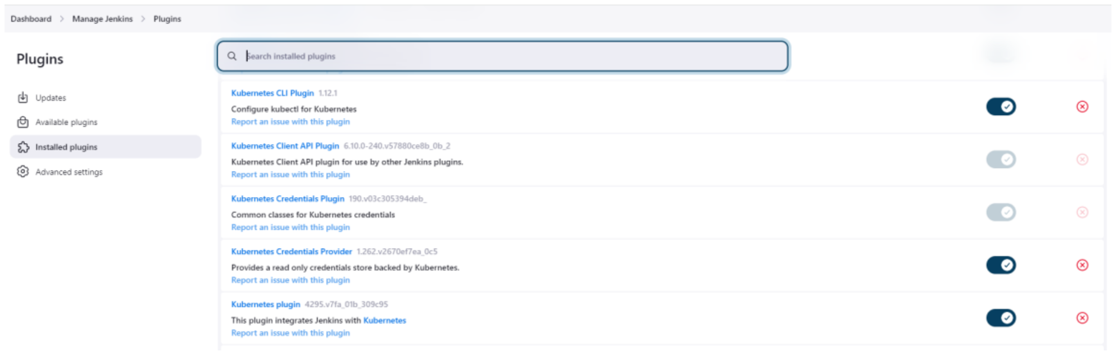
```
MINGW64:/c/Users/rseif/Downloads/Boardgame-main/Boardgame-main/DevopsProject

INTERNAL+rseif@SID-83XBDK3 MINGW64 ~/Downloads/Boardgame-main/Boardgame-main
$ git clone https://github.com/HebaMomen/DevopsProject.git
Cloning into 'DevopsProject'...
warning: You appear to have cloned an empty repository.

INTERNAL+rseif@SID-83XBDK3 MINGW64 ~/Downloads/Boardgame-main/Boardgame-main
$ cd DevopsProject/

INTERNAL+rseif@SID-83XBDK3 MINGW64 ~/Downloads/Boardgame-main/Boardgame-main/Dev
opsProject (main)
$ git add .
warning: in the working copy of '.github/workflows/maven.yml', LF will be replac
ed by CRLF the next time Git touches it
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the ne
xt time Git touches it
warning: in the working copy of '.mvn/wrapper/maven-wrapper.properties', LF will
 be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the ne
xt time Git touches it
warning: in the working copy of 'Jenkinsfile', LF will be replaced by CRLF the n
ext time Git touches it
warning: in the working copy of 'Jenkinsfile123', LF will be replaced by CRLF th
e next time Git touches it
```

Installed the below Plugins in Jenkin:

a. Pipeline Maven Integration/Config File Provider/SonarQube Scanner/Kubernetes CLI/Kubernetes/Docker/Docker Pipeline Step
    a. 

Dashboard > Manage Jenkins > Plugins

Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings

Search installed plugins

Kubernetes CLI Plugin 1.12.1
Configure kubectl for Kubernetes
Report an issue with this plugin

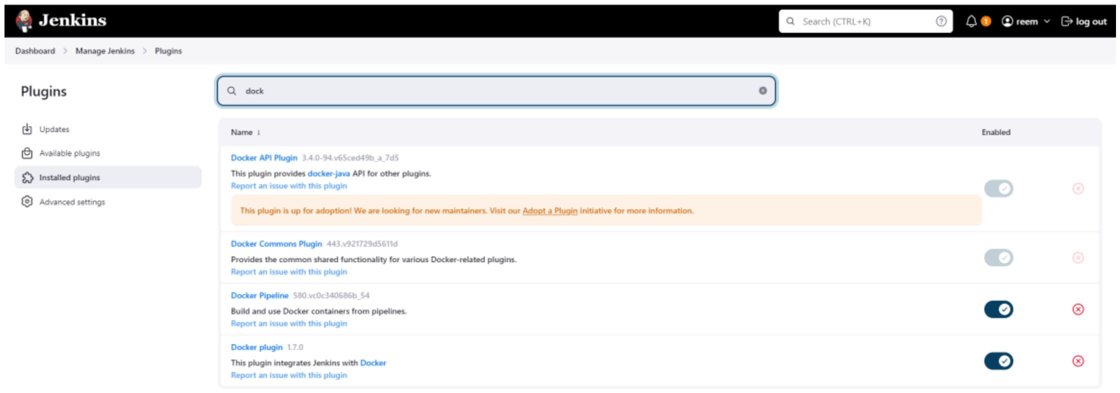
Kubernetes Client API Plugin 6.10.0-240.v57880ce8b_0b_2
Kubernetes Client API plugin for use by other Jenkins plugins.
Report an issue with this plugin

Kubernetes Credentials Plugin 190.v03c305394deb_
Common classes for Kubernetes credentials
Report an issue with this plugin

Kubernetes Credentials Provider 1.262.v2670ef7ea_0c5
Provides a read only credentials store backed by Kubernetes.
Report an issue with this plugin

Kubernetes plugin 4295.v7fa_01b_309c95
This plugin integrates Jenkins with Kubernetes
Report an issue with this plugin

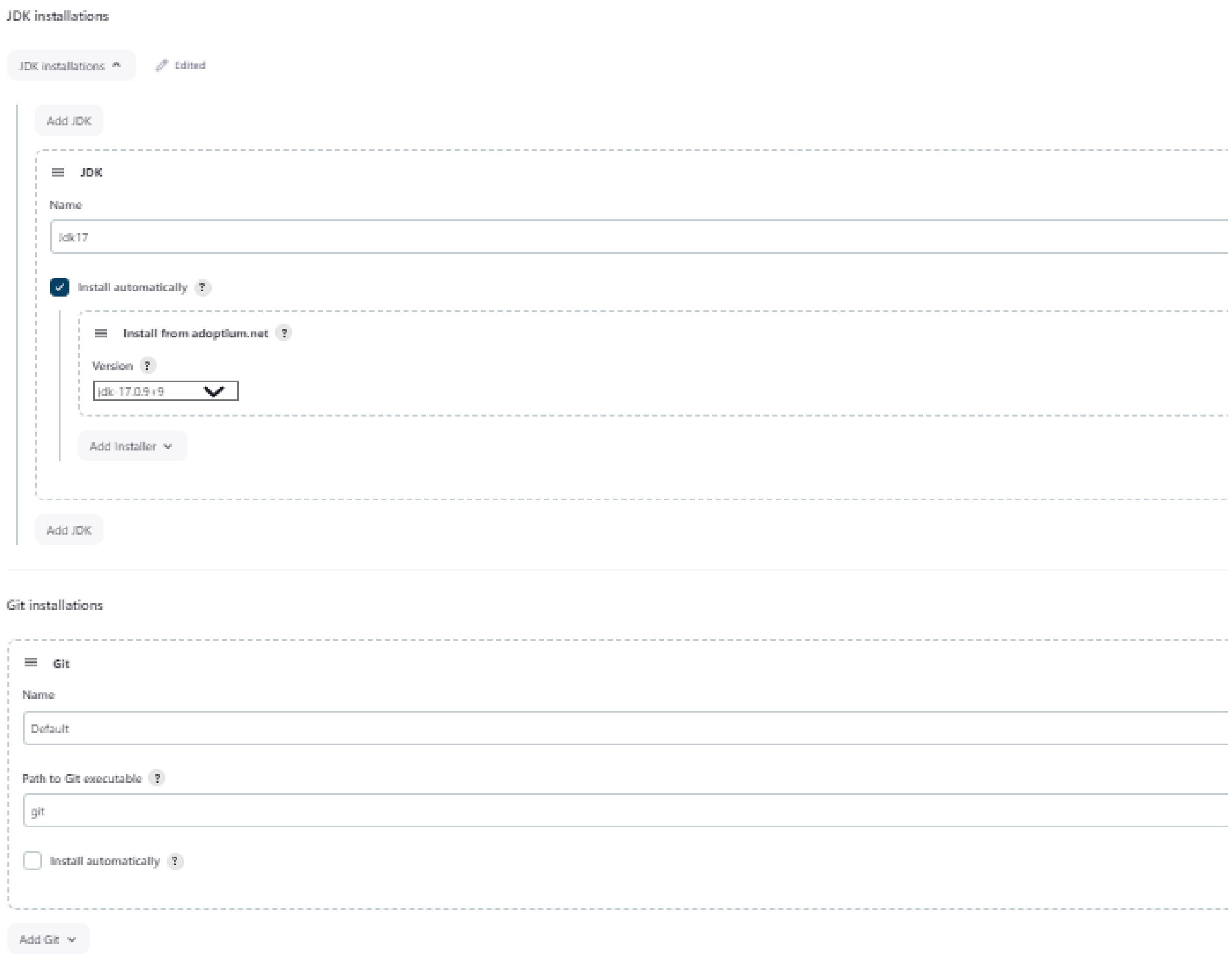
b.



dock

Name ↓	Enabled
Docker API Plugin 3.4.0-94.v65ced49b_a_7d5 This plugin provides docker-java API for other plugins. Report an issue with this plugin This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	✓
Docker Commons Plugin 443.v921729d5611d Provides the common shared functionality for various Docker-related plugins. Report an issue with this plugin	✓
Docker Pipeline 580.vc0c340686b_54 Build and use Docker containers from pipelines. Report an issue with this plugin	✓
Docker plugin 1.7.0 This plugin integrates Jenkins with Docker Report an issue with this plugin	✓

Configure Above Plugins in Jenkins Pipeline:



JDK installations

JDK

Name: jdk17

Install automatically ✓

Install from adoptium.net

Version: jdk-17.0.9+9

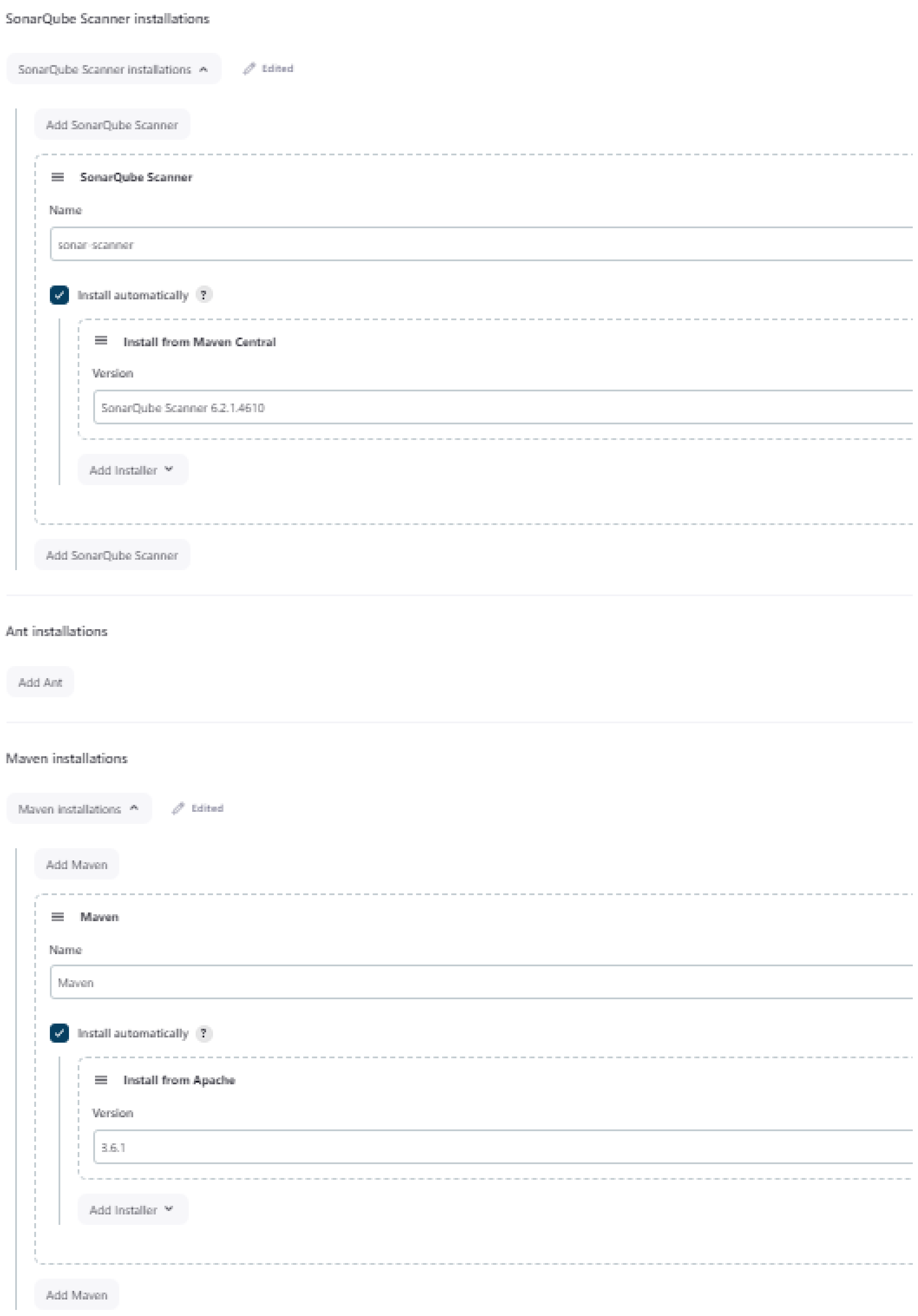
Git

Name: Default

Path to Git executable: git

Install automatically ☐

a.


SonarQube Scanner installations

SonarQube Scanner

Name: sonar-scanner

Install automatically ✓

Install from Maven Central

Version: SonarQube Scanner 6.2.1.4610

Ant installations

Maven installations

Maven

Name: Maven

Install automatically ✓

Install from Apache

Version: 3.6.1

b.

Java CI Pipeline with GitHub Actions:

The pipeline includes steps for building the project, running security scans, performing code quality analysis with SonarQube, building and scanning Docker images, and deploying to Kubernetes. Secrets are used to securely store sensitive information such as authentication tokens and configuration files.

Pipeline Overview:

1. **Java Build and Package:**
    a. Sets up JDK 17 using Temurin distribution.
    b. Builds the Java project using Maven.
    c. Uploads the generated JAR artifact as a GitHub Action artifact.
2. **Security Scans:**
    a. Performs file system scan using Trivy.
    b. Runs SonarQube scan for code quality analysis.
3. **Docker Build and Scan:**
    a. Sets up QEMU and Docker Buildx.
    b. Builds Docker image for the Java application.
    c. Scans Docker image using Trivy.
    d. Logs in to Docker Hub using provided credentials.
    e. Pushes the Docker image to Docker Hub.
4. **Kubernetes Deployment:**
    a. Uses Kubectl
    b. Action to interact with Kubernetes cluster.
    c. Applies deployment and service configuration from deployment-service.yaml file to deploy the application to Kubernetes namespace webapps.

Pipeline Configuration:

```
pipeline {
    agent any

    tools {
        jdk 'Jdk17'
        maven 'Maven'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }

    stages {
        stage('Get Checkout') {
            steps {
                git branch: 'main', credentialsId: 'git-cred', url:
'https://github.com/HebaMomen/DevopsProject.git'
            }
        }

        stage('Compile') {
            steps {
                sh "mvn compile"
            }
        }
        stage('Test') {
            steps {
                sh "mvn test"
            }
        }

        stage('File System Scan') {
            steps {
                sh "trivy fs --format table -o trivy-fs-report.html ."
            }
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonar') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=BoardGame
-Dsonar.projectKey=BoardGame \
                        -Dsonar.java.binaries=. '''
                }
            }
        }

        stage('Quality Gate') {
            steps {
                script {
```

```
                waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'
            }
        }
    }

    stage('Build') {
        steps {
            sh "mvn package"
        }
    }

    stage('Publish to Nexus') {
        steps {
            withMaven(globalMavenSettingsConfig: 'global-settings', jdk: 'Jdk17', maven:
'Maven', mavenSettingsConfig: '', traceability: true) {
                sh "mvn deploy"
            }
        }
    }

    stage('Build & Tag Docker Image') {
        steps {
            script {
               withDockerRegistry(credentialsId: 'docker-token', toolName: 'docker') {
                   sh "docker build -t reeemseif/boardgame:latest ."
                 }
            }
          }
        }

        stage('Docker Image Scan') {
          steps {
            sh "trivy image --format table -o trivy-image-report.html
reeemseif/boardgame:latest"
          }
        }

        stage('Push Docker Image') {
          steps {
            script {
              withDockerRegistry(credentialsId: 'docker-token', toolName: 'docker') {
                sh "docker push reeemseif/boardgame:latest"
              }
            }
          }
        }
        stage('Deploy To Kubernetes') {
          steps {
              withKubeConfig(caCertificate: '', clusterName: 'kubernetes', contextName: '',
credentialsId: 'K8-cred', namespace: 'webapps', restrictKubeConfigAccess: false,
serverUrl: 'https://172.31.87.217:6443') {
                sh "kubectl apply -f deployment-service.yaml"
            }
          }
        }

        stage('Verify the Deployment') {
          steps {
            withKubeConfig(caCertificate: '', clusterName: 'kubernetes', contextName: '',
credentialsId: 'K8-cred', namespace: 'webapps', restrictKubeConfigAccess: false,
serverUrl: 'https://172.31.87.217:6443') {
              sh "kubectl get pods -n webapps"
              sh "kubectl get svc -n webapps"
            }
          }
        }
      }

      post {
        always {
          script {
            def jobName = env.JOB_NAME
            def buildNumber = env.BUILD_NUMBER
            def pipelineStatus = currentBuild.result ?: 'UNKNOWN'
            def bannerColor = pipelineStatus.toUpperCase() == 'SUCCESS' ? 'green' : 'red'

            def body = """
            <html>
            <body>
            <div style="border: 4px solid ${bannerColor}; padding: 10px;">
               <h2>${jobName} - Build ${buildNumber}</h2>
               <div style="background-color: ${bannerColor}; padding: 10px;">
                  <h3 style="color: white;">Pipeline Status:
${pipelineStatus.toUpperCase()}</h3>
               </div>
               <p>Check the <a href="${BUILD_URL}">console output</a>.</p>
            </div>
            </body>
            </html>
            """

            emailext(
               subject: "${jobName} - Build ${buildNumber} -
${pipelineStatus.toUpperCase()}",
               body: body,
               to: 'reeemseif@gmail.com',
               from: 'jenkins@example.com',
               replyTo: 'jenkins@example.com',
               mimeType: 'text/html',
               attachmentsPattern: 'trivy-image-report.html'
            )
         }
      }
```

```
    }
}
```