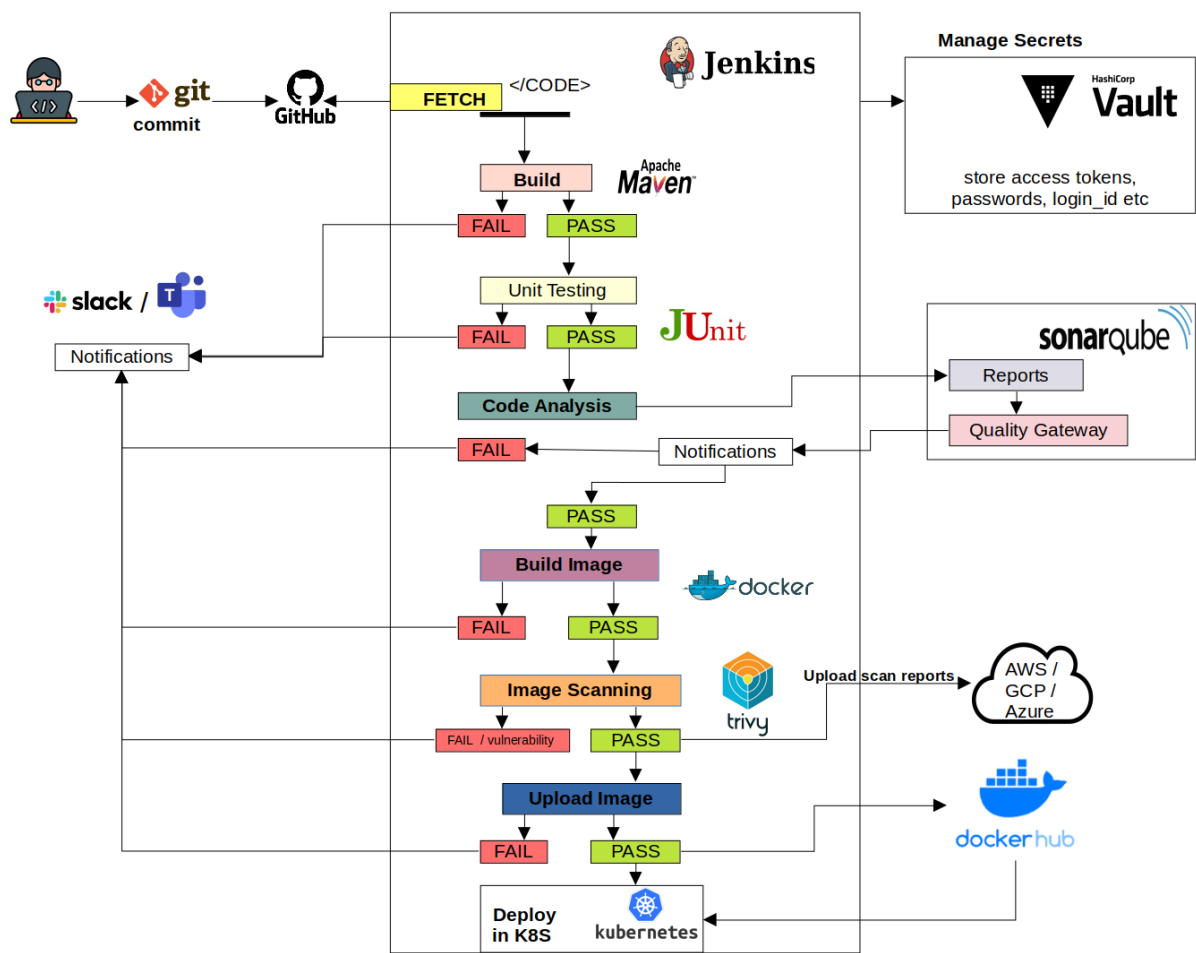
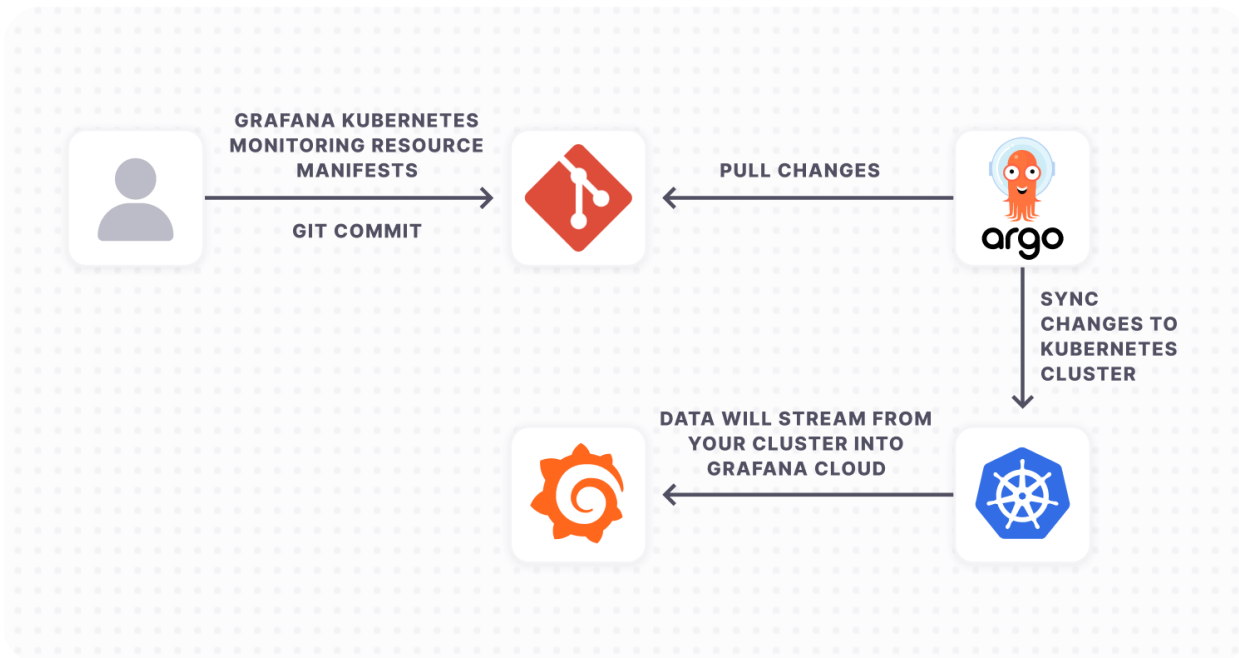


"Comprehensive CI/CD Pipeline with Jenkins, Maven, SonarQube, Nexus, Trivy, ArgoCD, Prometheus, and Grafana"





A CI/CD pipeline is a set of automated processes that are used to build, test, and deploy software applications. The pipeline is designed to streamline the process of software development by automating many of the repetitive tasks that are required to build, test, and deploy applications. A typical CI/CD pipeline includes several stages, including code compilation, testing, and deployment.

Tools Used in the Project

1. Github for version control(Private Reposistory)
2. Docker for containerization
3. SonarQube for code analysis
4. Jenkins for automation
5. Maven for building projects
6. Nexus for artifact management
7. Prometheus and Grafana for monitoring
8. Deploy in kubernetes

Pipeline Flow

1. Jenkins will fetch the code from the remote repo
2. Maven will build the code, if the build fails, the whole pipeline will become a failure and Jenkins will notify the user, If build success then
3. SonarQube scanner will scan the code and will send the report to the SonarQube server, where the report will go through the quality gate and gives the output to the web Dashboard. In the quality gate, we define conditions or rules like how many bugs or vulnerabilities, or code smells should be present in the code. Also, we have to create a webhook to send the status of quality gate status to Jenkins. If the quality gate status becomes a failure, the whole pipeline will become a failure then Jenkins will notify the user that your build fails.
4. After the quality gate passes, Docker will build the docker image. if the docker build fails when the whole pipeline will become a failure and Jenkins will notify the user that your build fails.
5. Trivy will scan the docker image, if it finds any Vulnerability then the whole pipeline will become a failure, and the generated report will be sent to s3 for future review and Jenkins will notify the user that your build fails.
6. After trivy scan docker images will be pushed to the docker hub, if the docker fails to push docker images to the docker hub then the pipeline will become a failure and Jenkins will notify the user that your build fails.
7. After the docker push, Jenkins will create deployment and service in minikube and our application will be deployed into Kubernetes. if Jenkins fails to create deployment and service in Kubernetes, the whole pipeline will become a failure and Jenkins will notify the user that your build fails.
8. ArgoCD will manage the continuous delivery of the application by monitoring the Kubernetes manifests and keeping the application synchronized with the desired state. If ArgoCD fails to sync or deploy the application, the pipeline will become a failure, and Jenkins will notify the user.
9. Prometheus will collect metrics from the application and Kubernetes cluster. Grafana will visualize these metrics on dashboard
10. At each stage the privacy, security is being maintained

STAGE-1:Installation set-up

->Creationof EKS cluster

```
PS C:\WINDOWS\system32> eksctl create cluster --name gclus --version 1.30 --node-type t3.medium --nodes 2 --managed --access
2024-07-16 15:15:17 [E] eksctl version 0.183.0
2024-07-16 15:15:17 [E] using region us-east-1
2024-07-16 15:15:20 [E] skipping us-east-1e from selection because it doesn't support the following instance type(s)
t3.medium
2024-07-16 15:15:20 [E] setting availability zones to [us-east-1b us-east-1c]
2024-07-16 15:15:20 [E] subnets for us-east-1b - public:192.168.0.0/19 private:192.168.64.0/19
2024-07-16 15:15:20 [E] subnets for us-east-1c - public:192.168.32.0/19 private:192.168.96.0/19
2024-07-16 15:15:20 [E] nodegroup "ng-92484b4c" will use "" [AmazonLinux2/1.30]
```

[Alt+S]

N. Virginia

k2 @ 2691-0238-5649

Extended support for Kubernetes versions pricing

New prices for extended support started in the April billing cycle. For more information, see the [blog post](#).

Notifications

0 0 0 4 0

EKS > Clusters

Clusters (1) Info

Filter clusters

	Cluster name	Status	Kubernetes version	Support period	Provider
	gclus	Active	1.30	Standard support until July 28, 2025	EKS

[Alt+S]

N. Virginia

k2 @ 2691-0238-5649

Filter Nodes by property or value

Node name	Instance type	Node group	Created	Status
ip-192-168-17-19.ec2.internal	t3.medium	ng-92484b4c	Created 15 minutes ago	Ready
ip-192-168-50-188.ec2.internal	t3.medium	ng-92484b4c	Created 15 minutes ago	Ready

Node groups (1) Info

Edit

Delete

Add node group

	Group name	Desired size	AMI release version	Launch template	Status
	ng-92484b4c	2	1.30.0-20240703	eksctl-gclus-nodegroup-ng-92484b4c (1)	Active

->Launching the instance using Terraform to keep up the industry standard

```

provider.tf vpc.tf igw.tf subnets.tf X natgw.tf routes.tf eks.tf oldc.tf iam-test.tf
subnets.tf > resource "aws_subnet" "private-us-east-1a" > tags
1 resource "aws_subnet" "private-us-east-1a" {
2   vpc_id = aws_vpc.aws-eks-vpc.id
3   cidr_block = "10.0.0.0/19"
4   availability_zone = "us-east-1a"
5
6   tags = {
7     "Name" = "private-us-east-1a"
8     "kubernetes.io/role/internal-elb" = "1"
9     "kubernetes.io/cluster/demo" = "owned"
10  }
11 }
12
13 resource "aws_subnet" "private-us-east-1b" {
14   vpc_id = aws_vpc.aws-eks-vpc.id
15 }
16 }
17
18 terraform apply
19
20 PS C:\Users\KHUSHI\Desktop\Terraform-project>

```

<input type="checkbox"/>	jen	i-06928064ee3c01e98	Running	t2.medium	2/2 checks passed	View alarms	us-east-1b
<input type="checkbox"/>	sonar	i-098f9bd0f435a1830	Running	t2.medium	2/2 checks passed	View alarms	us-east-1b
<input checked="" type="checkbox"/>	nexus	i-05308f93cf8f893a5	Running	t2.medium	2/2 checks passed	View alarms	us-east-1b

->Installation of sonarqube on the server

```

70770237f1b9d9ad1025a96647d5ab320ad356351d93c609de85569727a9c1b0
ubuntu@ip-172-31-20-46:~$ docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED        STATUS        PORTS                               NAMES
70770237f1b9   sonatype/nexus3    "/opt/sonatype/nexus..." 19 seconds ago Up 14 seconds 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp Nexus
ubuntu@ip-172-31-20-46:~$

ubuntu@ip-172-31-28-115:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
9b857f539cb1: Pull complete
708ff3b02f8b: Pull complete
e1ea69141092: Pull complete
5d590beb7c55: Pull complete
cb850744c992: Pull complete
4bbb66c34e6b: Pull complete
2d55b993c554: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:f51d6604ae94717faf2bf5c63cb90429d445f787ff2ac33ada38c0a36d8f8afe
Status: Downloaded newer image for sonarqube:lts-community
77d5a922c84818f6261d18828af16b263be987a426ea8e2f57c5f8bf57bbd155
ubuntu@ip-172-31-28-115:~$ docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED        STATUS        PORTS                               NAMES
77d5a922c848   sonarqube:lts-community "/opt/sonarqube/dock..." 46 seconds ago Up 40 seconds 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp
ubuntu@ip-172-31-28-115:~$

```

->Installation of Nexus on the server

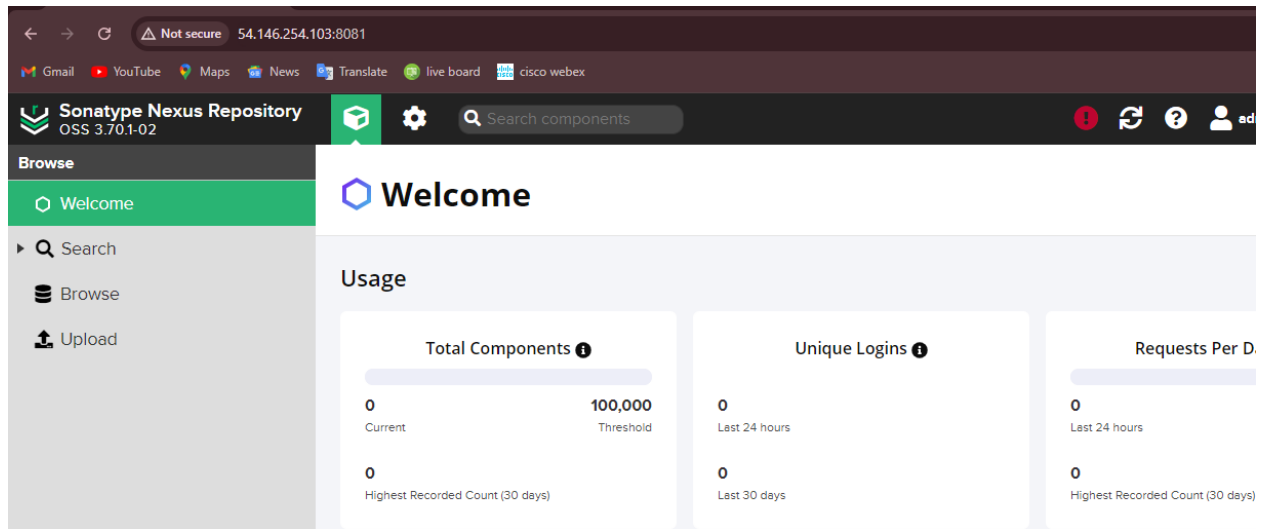
```

ubuntu@ip-172-31-20-46:~$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
70770237f1b9   sonatype/nexus3                    "/opt/sonatype/nexus-..." 19 seconds ago Up 14 seconds 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp Nexus
ubuntu@ip-172-31-20-46:~$ docker exec -it 70770237f1b9 /bin/bash
bash-4.4$ ls
nexus  sonatype-work  start-nexus-repository-manager.sh
bash-4.4$ cd sonatype-work
bash-4.4$ ls
nexus3
bash-4.4$ cd nexus3
admin.password  blobs  cache  db  elasticsearch  etc  generated-bundles  instances  javaprefs  karaf.pid  keystores  lock  log  orient  port  restore-from-backup  tmp
bash-4.4$ cat admin.password
5e6e5b8c-507e-4e45-8987-397ff0b964b9bash-4.4$

```

->UI of Nexus repository

Nexus Repository is a repository manager used to store, organize, and distribute software components and artifacts. It acts as a central hub for managing all kinds of artifacts required in the development process, including binaries, libraries, containers, and other packages.



->UI of sonarqube

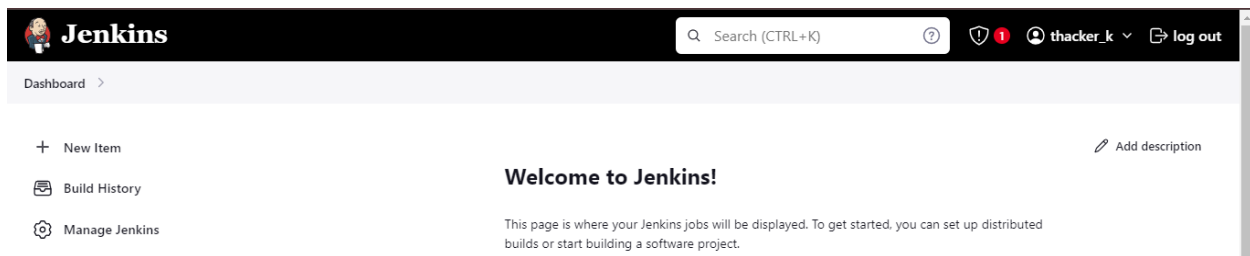
SonarQube is a powerful tool used for continuous inspection of code quality. It helps in ensuring that code meets defined standards and is free of bugs and vulnerabilities.

->Installation of Jenkins on the server

```
1 jen x 2 nexus x 3 sonar x +
ubuntu@ip-172-31-17-141:~$ sudo chmod 666 /var/run/docker.sock
ubuntu@ip-172-31-17-141:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
ubuntu@ip-172-31-17-141:~$ jenkins --version
2.452.3
ubuntu@ip-172-31-17-141:~$
```

->UI of Jenkins

Jenkins is an open-source automation server used extensively for continuous integration and continuous delivery (CI/CD). It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous delivery and integration.



STAGE-2

->Creation of private github reposistory

Private repositories on GitHub is created for several important reasons, particularly to ensure security, control, and confidentiality of code and project artifacts

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * khushi280 / Repository name * Reddit
Checking availability...

Great repository names are short and memorable. Need inspiration? How about [friendly-octo-meme](#)?

Description (optional)

☐ Public
 Anyone on the internet can see this repository. You choose who can commit.

☒ Private
 You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file
 This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
 .gitignore template: None
 Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

->Pushing the source code in it

khushi280 / Reddit

Code Issues Pull requests Actions Projects Security Insights Settings

Reddit Private

Unwatch 1 Fork Star 0

main 1 Branch 0 Tags

Go to file Add file Code

khushi280 Update pom.xml 177d2e0 · 20 hours ago 5 Commits

File	Type	Commit
.github/workflows	final	yesterday
.mvn/wrapper	final	yesterday
src	final	yesterday
.gitignore	final	yesterday
Dockerfile	final	yesterday
Jenkinsfile	final	yesterday
Jenkinsfile123	final	yesterday
README.md	final	yesterday
deployment-service.yaml	Update deployment-service.yaml	yesterday
mvnw	final	yesterday
mvnw.cmd	final	yesterday

About

No description, website, or topics provided.

Readme Activity 0 stars 1 watching 0 forks

Releases

No releases published
[Create a new release](#)

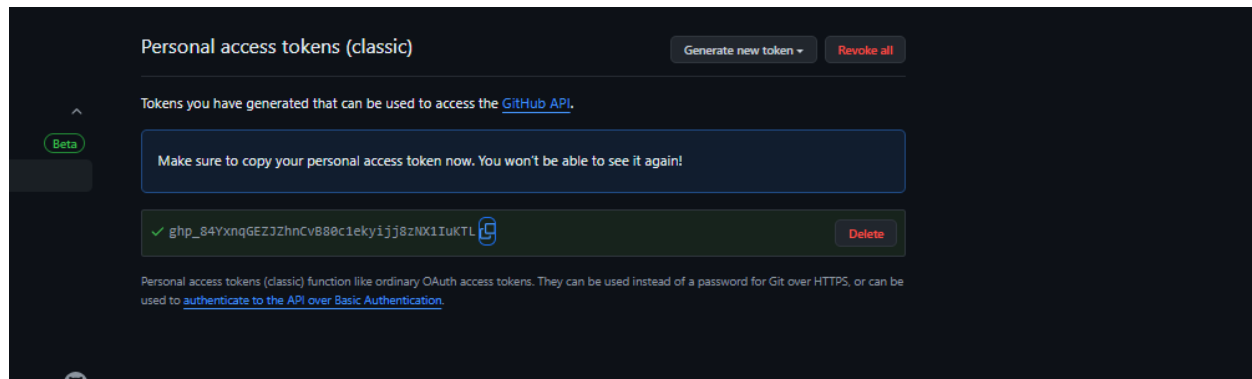
Packages

No packages published
[Publish your first package](#)

Languages

HTML 50.4% Java 45.9%

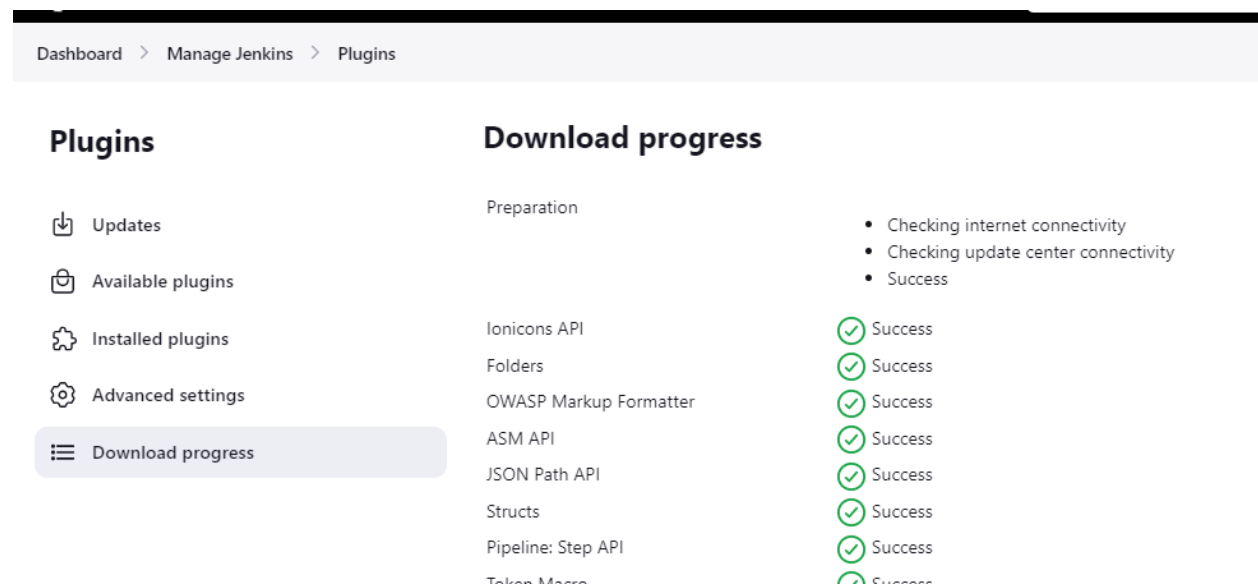
->Access the token



STAGE-3

->Plugins installation

plugins are critical for setting up a robust and efficient CI/CD pipeline. They enable integration with various tools and technologies, automate repetitive tasks, ensure quality and security, and provide monitoring and reporting capabilities. This extensibility and flexibility make Jenkins (and similar CI/CD tools) adaptable to diverse development and deployment workflows. All the necessary plungins have been installed for writing the script.

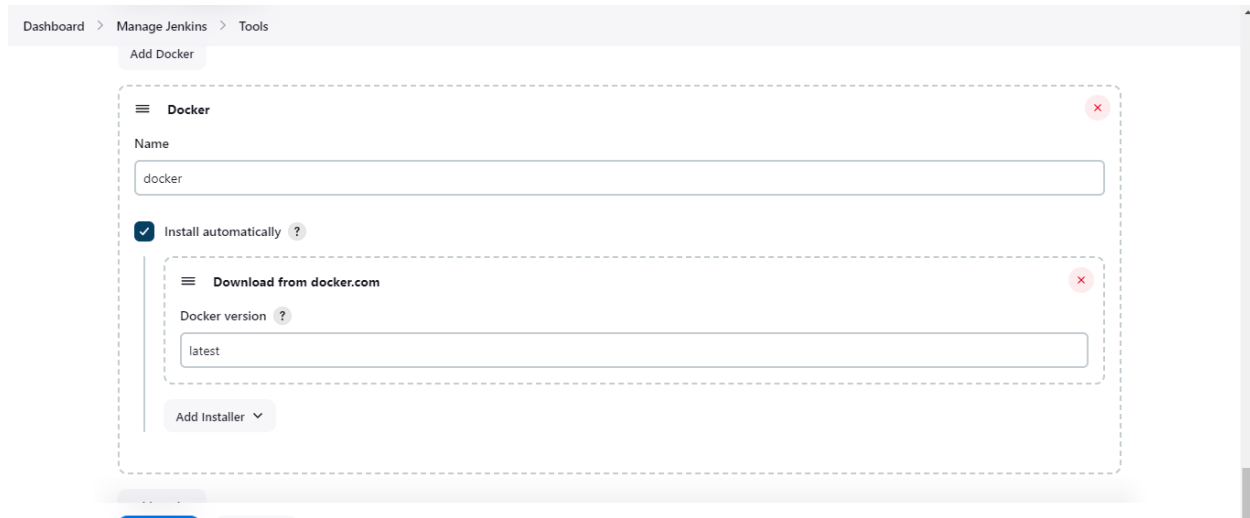


1.Eclipse Temurin Installer:

- This plugin enables Jenkins to automatically install and configure the Eclipse Temurin JDK (formerly known as AdoptOpenJDK).
- 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..
- 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..
- 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.
- 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..
- 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.
- 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.
- 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.

->Configuring tools

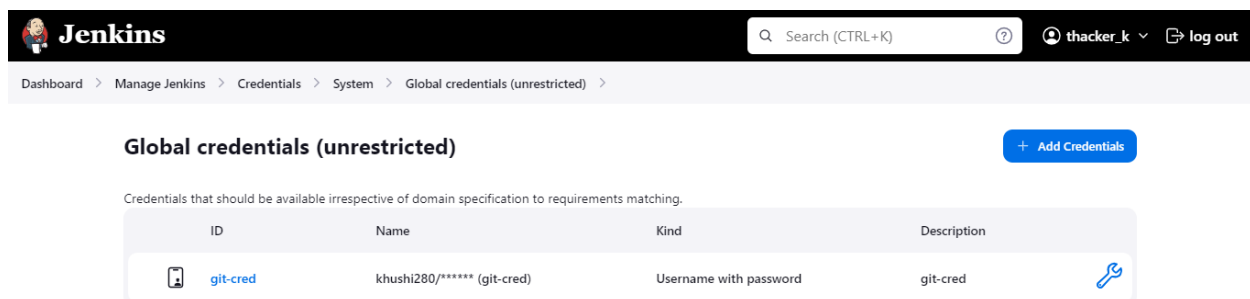
Configuring tools in Jenkins enhances automation, integration, quality assurance, artifact management, containerization, monitoring, environment management, security, scalability, and project management within the CI/CD pipeline. This results in a more efficient, reliable, and scalable software development and deployment process.



-> Creating and managing credentials at each step of a CI/CD pipeline in tools like Jenkins is critical for securely integrating with external systems, adhering to security best practices, automating workflows efficiently, ensuring compliance, and maintaining the overall security posture of the software development and deployment processes. At each step the credentials is going created

```
stages {
    stage('Git Checkout') {
        steps {
            git branch: 'main', credentialsId: 'git-cred', url:
'https://github.com/khushi280/reddit.git'
        }
    }
}
```

This stage is going created for connecting to the github



Each step of the pipeline has been configured using the pipeline syntax

Dashboard > reddit > Pipeline Syntax

khushi280 (git-view)

+ Add ▾

☒ Include in polling? ?

☒ Include in changelog? ?

Generate Pipeline Script

git branch: 'main', credentialsId: 'git-cred', url: 'https://github.com/khushi280/Reddit.git'

-> Trivy is installed in CI/CD pipelines to enhance container security by automating vulnerability scanning, ensuring compliance with security policies, mitigating risks, and supporting DevSecOps practices that prioritize security throughout the software development lifecycle.

```
stage('Trivy scanning') {
    steps {
        sh "trivy fs --format table -o trivy-fs-report.html ."
    }
}
```

This has been created to generate the trivy reports for analysis


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

Generate Tokens

Name Expires in

Enter Token Name 30 days Generate

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

 Copy `squ_bad612eaf261855990db360b30b3d64095f54b98`

Name	Type	Project	Last use	Created	Expiration	
sonar-token	User		Never	July 16, 2024	August 15, 2024	Revoke

The sonarqube token has been generated after that configuration of the webhook has been done.

Create Webhook

All fields marked with * are required

Name *



URL *



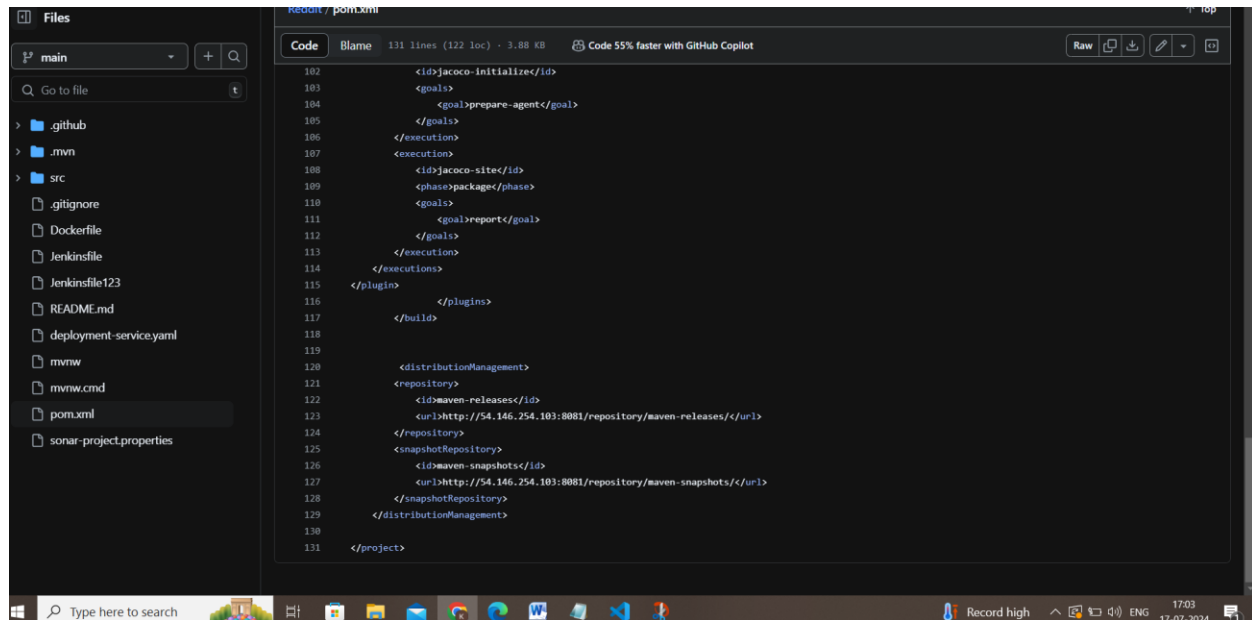
Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret

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

-> Deploys artifacts to Nexus repository using Maven, with global Maven settings configured (global-settings).and configured the pom.xml file

```
stage('Nexus deployment') {
    steps {
        withMaven(globalMavenSettingsConfig: 'global-settings', jdk:
'jdk17', maven: 'maven3', mavenSettingsConfig: '', traceability: true) {
            sh "mvn deploy"
        }
    }
}
```



-> Builds a Docker image (`docker build`), using Docker credentials (`docker-cred`).

-> **Docker Image Scan:**

- Uses Trivy to scan the Docker image for vulnerabilities and generates a report (`trivy-image-report.html`).

-> **Push Docker Image:**

- Pushes the Docker image (`docker push`) to a Docker registry using Docker credentials (`docker-cred`).

Dashboard > reddit > Pipeline Syntax

? Examples Reference
? IntelliJ IDEA GDSL

withDockerRegistry: Sets up Docker registry endpoint

withDockerRegistry ?

Docker registry URL ?

Registry credentials

khushi2803/***** (docker-cred) ▼

+ Add ▼

Docker installation

docker ▼

Generate Pipeline Script

// This step should not normally be used in your script. Consult the inline help for details.
withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {
// some block

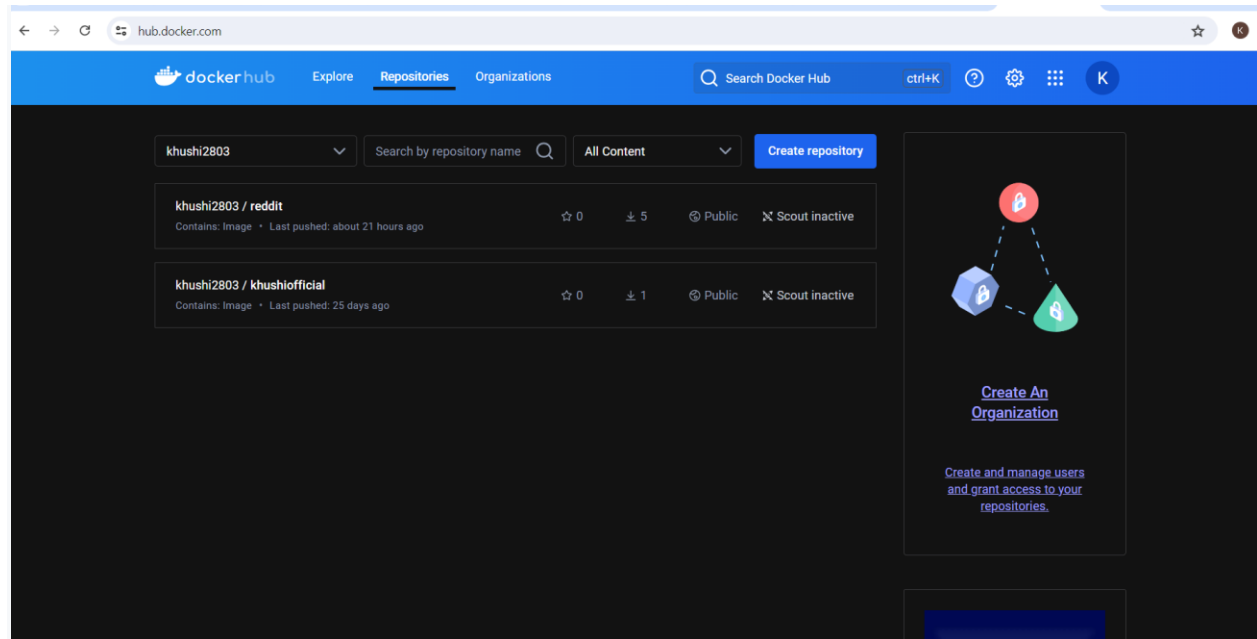
```

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

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

stage('Push Docker Image') {
    steps {
        script {
            withDockerRegistry(credentialsId: 'docker-cred', toolName:
'docker') {
                sh "docker push "
            }
        }
    }
}

```



Docker-hub reposistory


```
! sec.yaml x
! sec.yaml
1  apiVersion: v1
2  kind: Secret
3  type: kubernetes.io/service-account-token
4  metadata:
5    name: mysecretname
6    annotations:
7      kubernetes.io/service-account.name: jenkins
8

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
powershell + v []

spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ kubectl create ns webapps
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (kubectl:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS E:\Devops\Proj_k> kubectl create ns webapps
namespace/webapps created
PS E:\Devops\Proj_k> kubectl apply -f svc.yaml
serviceaccount/jenkins created
PS E:\Devops\Proj_k> kubectl apply -f role.yaml
role.rbac.authorization.k8s.io/app-role created
PS E:\Devops\Proj_k> kubectl apply -f bind.yaml
rolebinding.rbac.authorization.k8s.io/app-rolebinding created
PS E:\Devops\Proj_k> kubectl apply -f sec.yaml -n webapps
The Secret "mysecretname" is invalid: metadata.annotations[kubernetes.io/service-account.name]: Required value
PS E:\Devops\Proj_k> kubectl apply -f sec.yaml -n webapps
secret/mysecretname created
PS E:\Devops\Proj_k> 
```

->Deploys the application to Kubernetes (kubectl apply) in the webapps namespace of the specified Kubernetes cluster (kubernetes), using Kubernetes credentials (k8-cred)

->Verifies the deployment by retrieving and displaying information about pods (kubectl get pods) and services (kubectl get svc) in the webapps namespace.

```
1jen x 2nexus x 3sonar x +
ubuntu@ip-172-31-17-141:~$ trivy --version
Version: 0.53.0
ubuntu@ip-172-31-17-141:~$ vi k.sh
ubuntu@ip-172-31-17-141:~$ sudo chmod +x k.sh
ubuntu@ip-172-31-17-141:~$ ./k.sh
-bash: ./k.sh: No such file or directory
ubuntu@ip-172-31-17-141:~$ ./k.sh
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total     Spent    Left     Speed
100 57.4M 100 57.4M    0     0 19.0M      0  0:00:03  0:00:03 --:--:-- 19.0M
Client Version: v1.19.6-eks-49a6c0
ubuntu@ip-172-31-17-141:~$ 
```

```

stage('Deploy To Kubernetes') {
    steps {
        withKubeConfig(caCertificate: '', clusterName: '',
contextName: '', credentialsId: 'k8-cred', namespace: 'webapps',
restrictKubeConfigAccess: false, serverUrl: 'https://172.31.8.146:6443') {
            sh "kubectl apply -f deployment-service.yaml"
        }
    }
}

```

->The complete pipeline has been set-up with all the crediantials .

Global credentials (unrestricted) [+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
git-cred	khushi280/***** (git-cred)	Username with password	git-cred
sonar-token	sonar-token	Secret text	sonar-token
docker-cred	khushi2803/***** (docker-cred)	Username with password	docker-cred
k8-cred	k8-cred	Secret text	k8-cred

Icon: ☐ S ☐ M ☒ L

->The complete script has been configured

Dashboard > reddit > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Pipeline

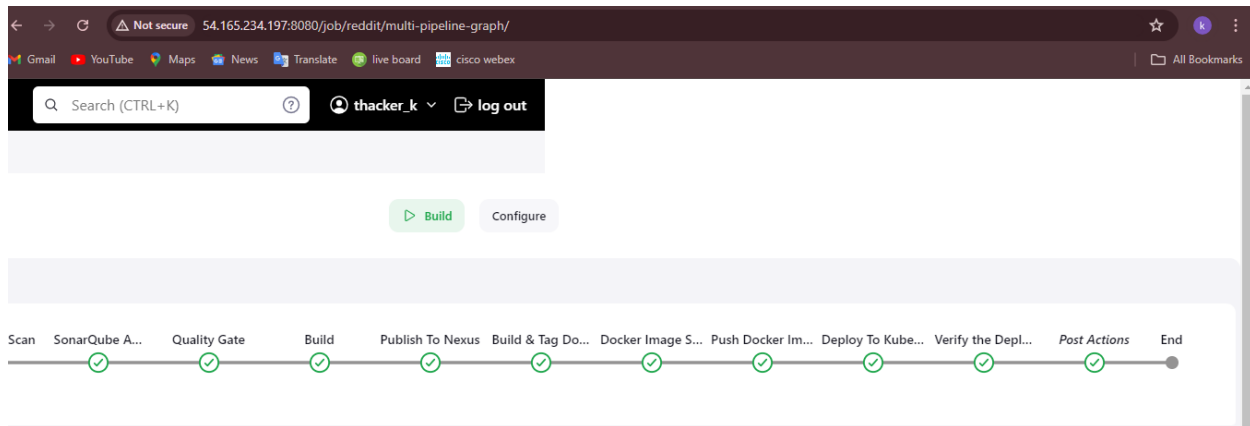
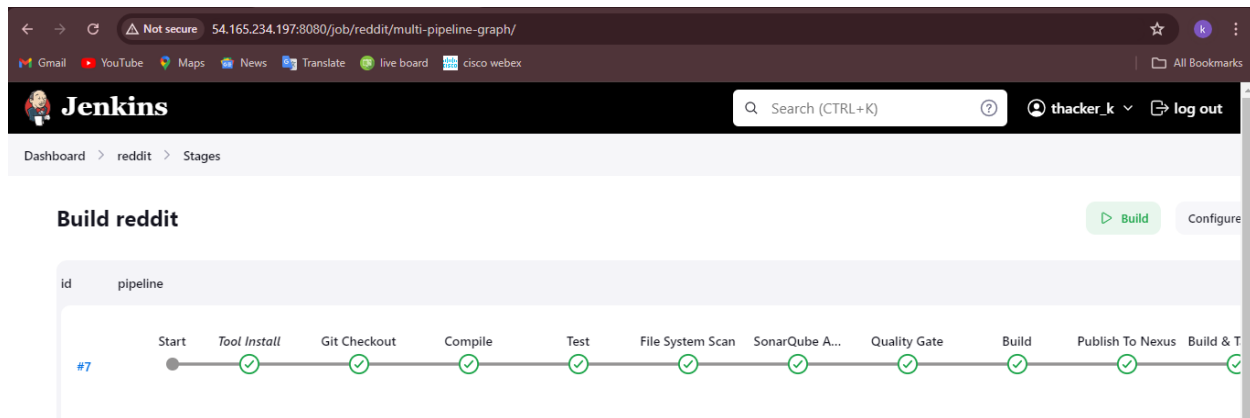
Definition: Pipeline script

```

Script ?
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
    steps {
        script {
            withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {
                sh "docker push khushi2803/reddit:latest"
            }
        }
        stage('Deploy To Kubernetes') {
            steps {
                withKubeConfig(caCertificate: '', clusterName: 'gclus.us-east-1.eksctl.io', contextName: '', credentia
                sh "kubectl apply -f deployment-service.yaml"
            }
        }
    }

```

->Successful building of the whole ci-cd pipeline with all the stages

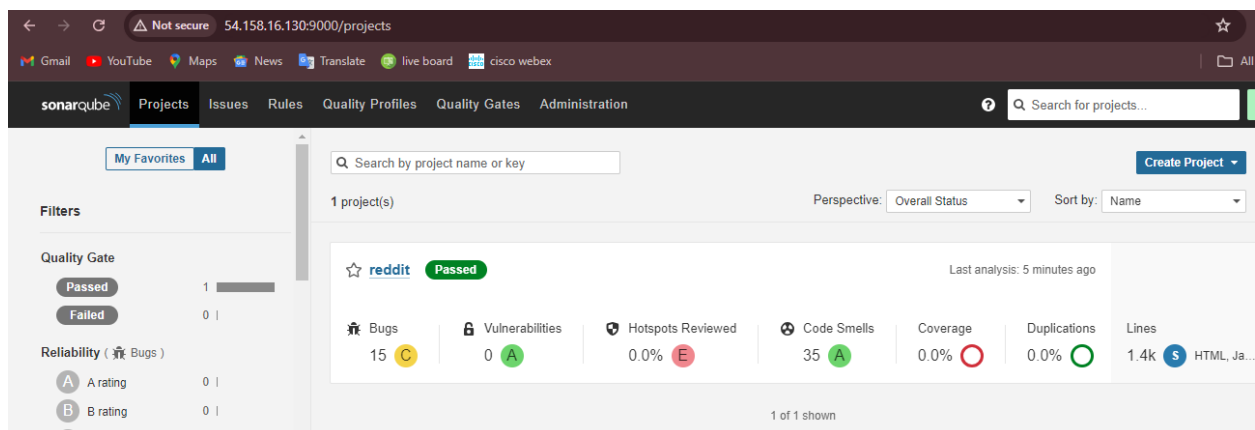


- **Success:** The pipeline successfully executed all stages from code checkout to deployment, ensuring that the application was built, tested, analyzed for security vulnerabilities, packaged into Docker containers, deployed to Kubernetes, and verified for functionality.
 - **Failure:** The pipeline halts if any stage fails (e.g., test failures, security vulnerabilities exceeding thresholds, deployment issues), notifying stakeholders via email and providing details for troubleshooting.
1. **Git Checkout:**
 - **Result:** Successfully fetched the latest code from the Git repository.
 2. **Compile:**
 - **Result:** Compilation completed successfully without errors.
 3. **Test:**
 - **Result:** Unit tests executed and passed, indicating basic functionality is intact.
 4. **File System Scan (Trivy):**

- **Result:** Identified vulnerabilities in local dependencies and filesystem, if any, reported in `trivy-fs-report.html`.
- 5. **SonarQube Analysis:**
 - **Result:** Analyzed code quality metrics and potential issues such as bugs, vulnerabilities, and code smells. Quality Gate status determined the next steps.
- 6. **Quality Gate:**
 - **Result:** Pipeline continued or halted based on SonarQube Quality Gate status. Passed if code met quality criteria; failed otherwise.
- 7. **Build (Maven Package):**
 - **Result:** Packaged the application artifacts (e.g., JAR files) successfully.
- 8. **Publish To Nexus:**
 - **Result:** Deployed artifacts to Nexus repository manager for version control and distribution.
- 9. **Build & Tag Docker Image:**
 - **Result:** Built Docker image containing the application.
- 10. **Docker Image Scan (Trivy):**
 - **Result:** Scanned Docker image for vulnerabilities, producing `trivy-image-report.html` with findings.
- 11. **Push Docker Image:**
 - **Result:** Pushed Docker image to the Docker registry for deployment.
- 12. **Deploy To Kubernetes:**
 - **Result:** Deployed application to Kubernetes cluster in the `webapps` namespace using `deployment-service.yaml`.
- 13. **Verify the Deployment:**
 - **Result:** Confirmed successful deployment by checking pods (`kubectl get pods -n webapps`) and services (`kubectl get svc -n webapps`) in Kubernetes.

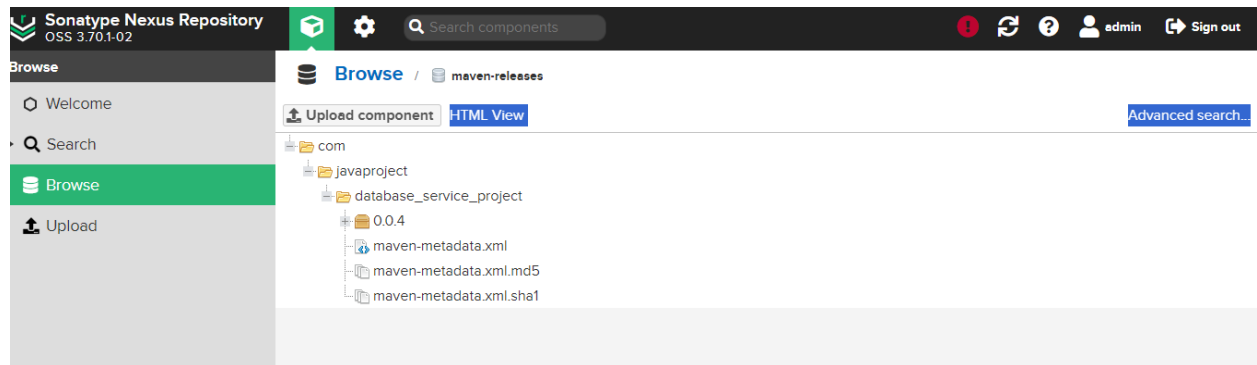
->The results analysis of the whole pipeline

Sonarqube quality test has been passed



->The artifacts has been formed

->The nexus repository has been configured



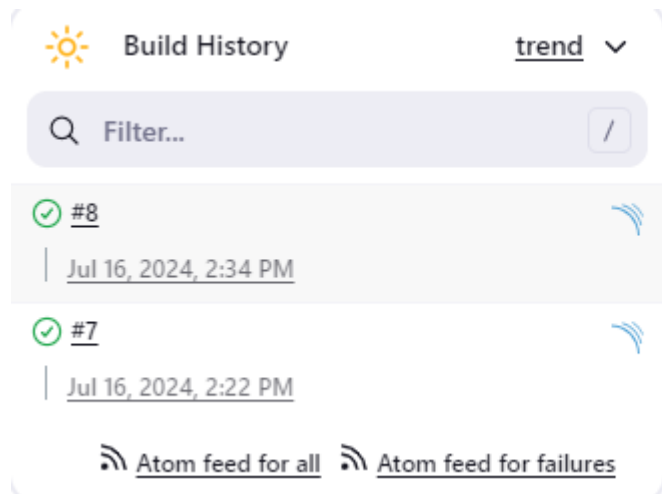
->The Trivy report has been generated

18.159.41.141:8080/job/tetrisgame/3/execution/node/3/ws/trivy-fs-report.html

pom.xml (pom) ===== Total: 49 (UNKNOWN: 0, LOW: 0, MEDIUM: 17, HIGH: 24, CRITICAL: 8)

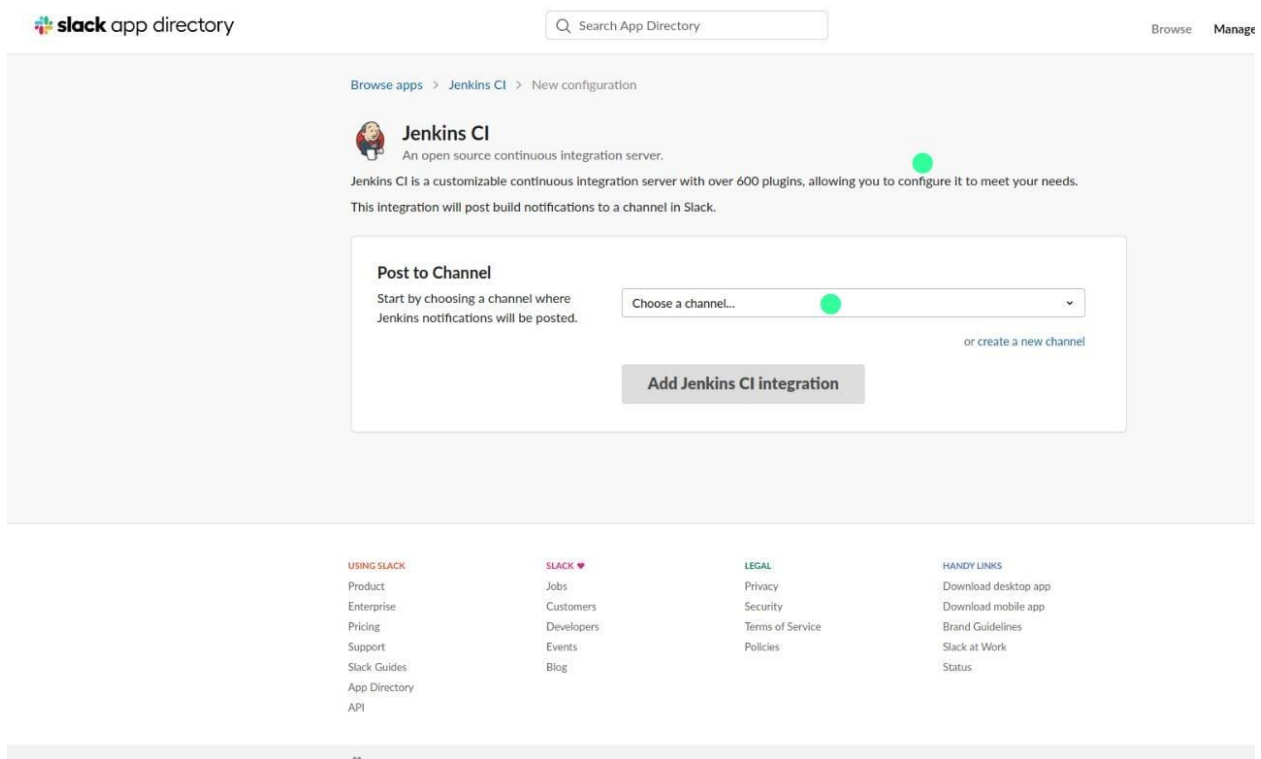
Library	Vulnerability	Severity	Status	Installed Version	Fixed Version	Title
ch.qos.logback:logback-classic	CVE-2023-6378	HIGH	fixed	1.2.6	1.3.12, 1.4.12, 1.2.13	logback: serialization vulnerability in logback receiver
https://avd.aquasec.com/nvd/cve-2023-6378						ch.qos.logback:logback-core
MEDIUM	1.2.9	logback: remote code execution through JNDI call from within			its configuration file...	https://avd.aquasec.com/nvd/cve-2021-42550
com.fasterxml.jackson.core:jackson-databind	CVE-2020-36518	HIGH		2.12.5	2.13.2.1, 2.12.6.1	jackson-databind: denial of service via a large depth of nested objects
https://avd.aquasec.com/nvd/cve-2020-36518						CVE-2021-46877
2.12.6, 2.13.1	jackson-databind: Possible DoS if using JDK serialization to				serialize JsonNode	https://avd.aquasec.com/nvd/cve-2021-46877
2.12.7.1, 2.13.4.2	jackson-databind: deep wrapper array nesting wrt				UNWRAP_SINGLE_VALUE_ARRAYS	https://avd.aquasec.com/nvd/cve-2022-42003
2.12.7.1, 2.13.4	jackson-databind: use of deeply nested arrays					https://avd.aquasec.com/nvd/cve-2022-42004
com.h2database:h2	CVE-2021-42392	CRITICAL		1.4.200	2.0.206	h2: Remote Code Execution in Console
2.1.210	h2: Loading of custom classes from remote servers through				JNDI	https://avd.aquasec.com/nvd/cve-2022-23221
2.0.202	h2database: XXE injection vulnerability					https://avd.aquasec.com/nvd/cve-2021-23463
2.2.220	The web-based admin console in H2 Database Engine before				2.2.220 can b...	https://avd.aquasec.com/nvd/cve-2022-45868

->The complete build history of the pipeline

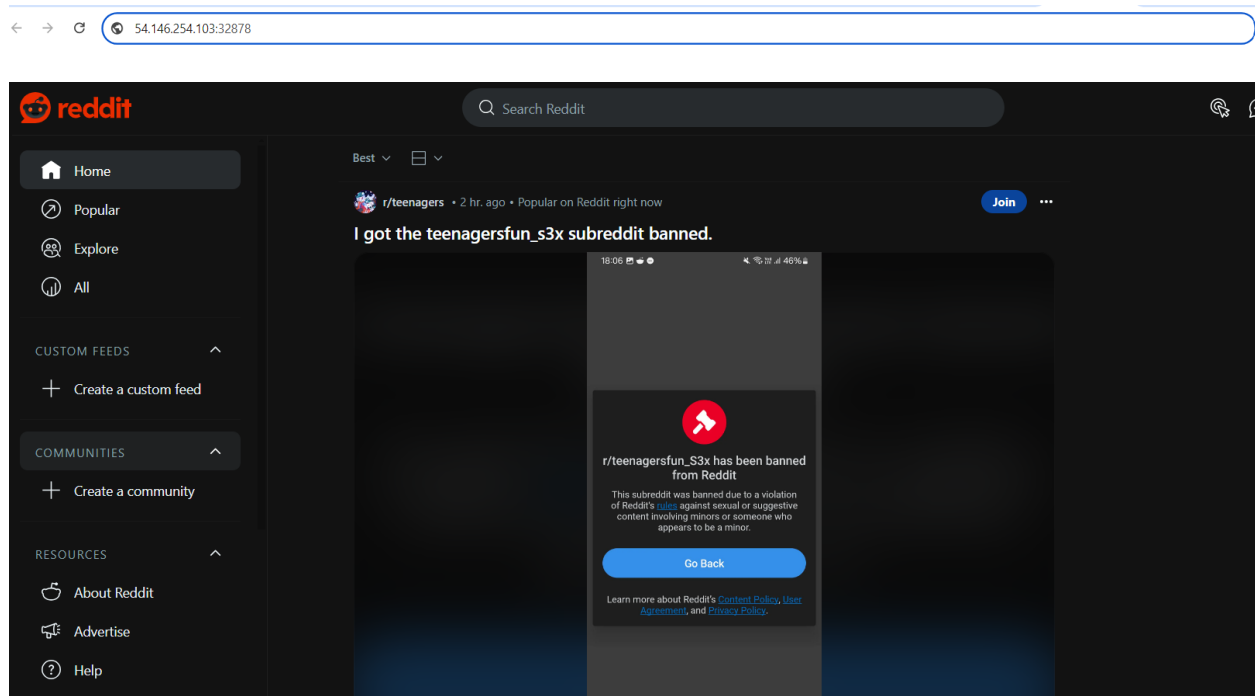


- Last build (#8), 7
- Last stable build
- Last successful b
- Last completed I

->Slack notification



->The application has been deployed and the pods are running



ArgoCD streamlines and automates Kubernetes deployments through GitOps principles, providing robust capabilities for managing application lifecycles, ensuring consistency, scalability, security, and visibility across Kubernetes clusters. This makes it a preferred choice for organizations adopting Kubernetes for containerized application orchestration and deployment. Configured the whole argocd pipeline

```

Handling connection for 8080
PS E:\Devops\Proj_k> kubectl get po -n argocd
NAME                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0     1/1     Running   0           16m
argocd-applicationset-controller-8485455fd5-pljzg  1/1     Running   0           16m
argocd-dex-server-66779d96df-1l29l    1/1     Running   0           16m
argocd-notifications-controller-c4b69fb67-pbp5g  1/1     Running   0           16m
argocd-redis-7bf7cb9748-v25v4         1/1     Running   0           16m
argocd-repo-server-795d79dfb6-wszbj    1/1     Running   0           16m
argocd-server-544b7f897d-9jnlm        1/1     Running   0           16m
PS E:\Devops\Proj_k> kubectl port-forward svc/argocd-server -n argocd 8080:443
Forwarding from [::1]:8080 -> 8080
PS E:\Devops\Proj_k> kubectl port-forward -n argocd argocd-server-544b7f897d-9jnlm 0 8080:8080
error: remote port must be > 0
PS E:\Devops\Proj_k> kubectl port-forward -n argocd argocd-server-544b7f897d-9jnlm 8080:8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
Handling connection for 8080

```

->Installation of the argocd and configuring the argocd application

```

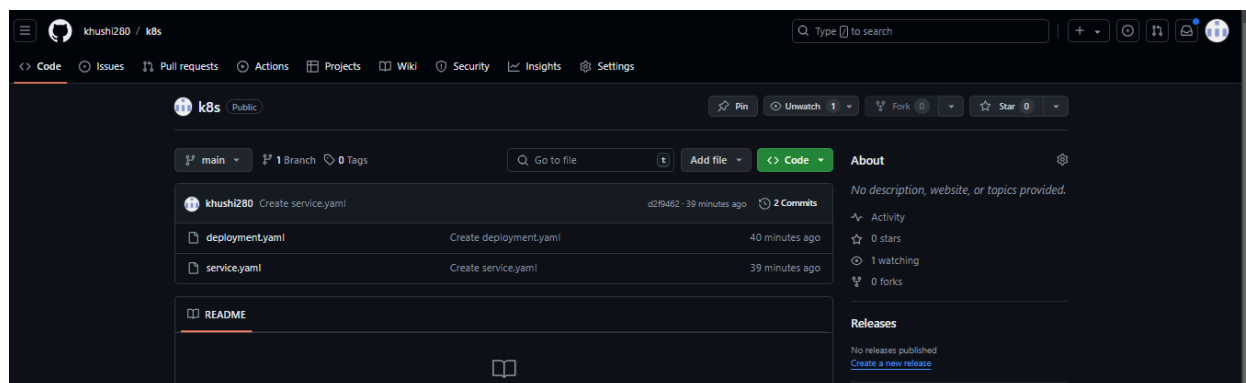
PS E:\Devops\Proj_k> kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}"
+ FullyQualifiedErrorId : CommandNotFoundException

PS E:\Devops\Proj_k> aws configure
AWS Access Key ID [*****XPBV]:
AWS Secret Access Key [*****dU0w]:
Default region name [us-east-1]:
Default output format [table]:
PS E:\Devops\Proj_k> kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d
base64 : The term 'base64' is not recognized as the name of a cmdlet, function, script file, or
is correct and try again.
At line:1 char:91
+ ... rgocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d
+
+ CategoryInfo          : ObjectNotFound: (base64:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

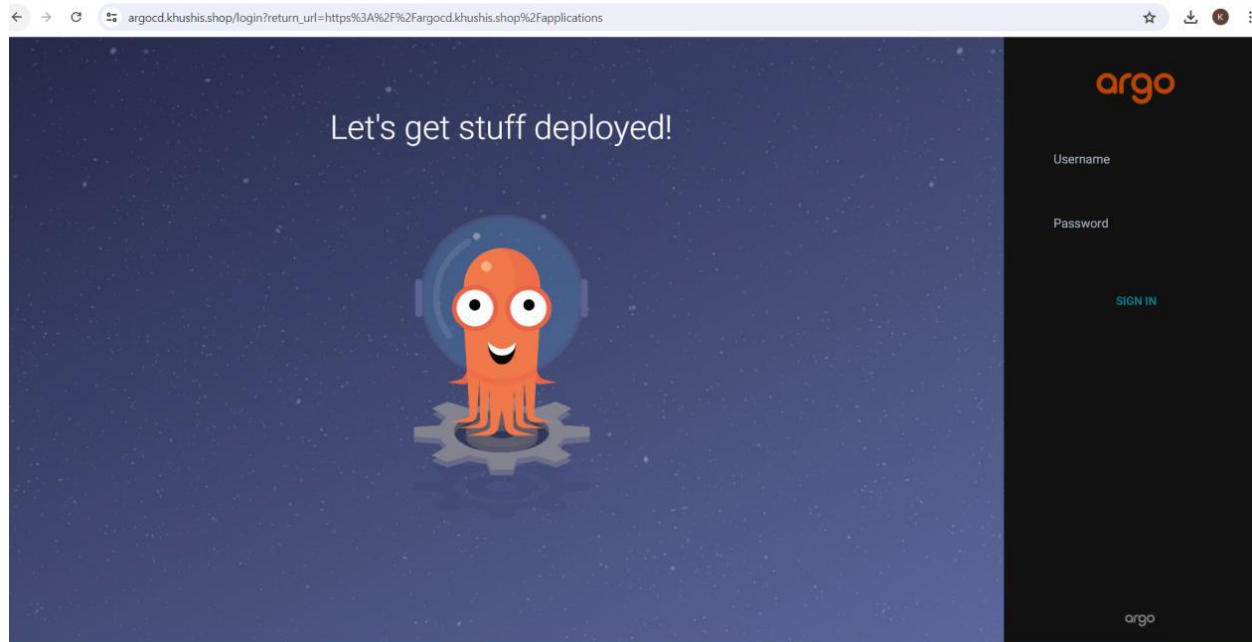
PS E:\Devops\Proj_k> $secret = kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}"
PS E:\Devops\Proj_k> $decodedSecret = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($secret))
PS E:\Devops\Proj_k> Write-Output $decodedSecret
mSumtH0XI4tVByDy
PS E:\Devops\Proj_k>
PS E:\Devops\Proj_k>

```

->Creation of the argocd repository for the syncing and managing the changes



->The argocd syncing of the application



Configuring the argo-cd cli

```
ShimGen has successfully created a shim for argocd.exe
The install of argocd-cli was successful.
Software install location not explicitly set, it could be in package or
default install location of installer.

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
```

->Successful configuration of the argocd pipeline

→

Applications

+ NEW APP

↻ SYNC APPS

↻ REFRESH APPS

🔍 Search applications...

reddit-app

Project: default

Labels:

Status: Healthy Unknown

Reposito... https://github.com/khushi280/Reddit.git

Target R... HEAD

Path: k8s

Destinati... in-cluster

Namesp... webapps

Created ... 07/16/2024 21:24:36 (13 minutes ago)

↻ SYNC

↻

✕

Healthy

Synced to HEAD (d742ec0)

Sync OK to d742ec0

Auto sync is enabled.

Author: khushi280 <khushihacker2003@gmail.com> -

Comment: Done by Jenkins Job changemanifest: 4

Succeeded 5 minutes ago (Sat Jul 06 2024 19:42:18 GMT+0530)

Author: khushi280 <khushihacker2003@gmail.com> -

Comment: Done by Jenkins Job changemanifest: 4

100%

Settings / Repositories

REPOSITORIES

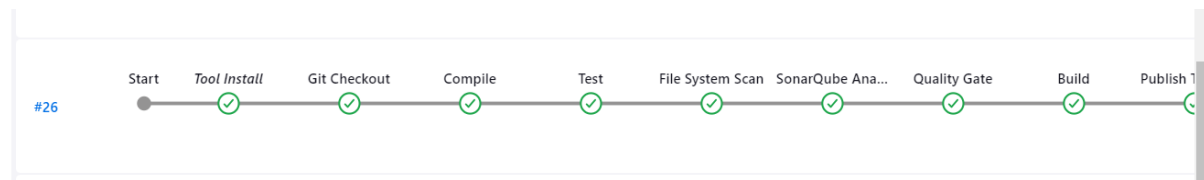
+ CONNECT REPO

↻ REFRESH LIST

Log out

	TYPE	NAME	REPOSITORY	CONNECTION STATUS	
	git		https://github.com/khushi280/Reddit.git	Successful	
	git		https://github.com/khushi280/k8s.git	Successful	

26



Prometheus is an open-source monitoring and alerting toolkit designed for reliability and scalability in dynamic environments like cloud-native applications and microservices.

- **Metrics Collection:** Prometheus collects metrics from monitored targets by scraping HTTP endpoints at regular intervals.
- **Data Model:** It stores time-series data in a custom, efficient format, allowing flexible querying for monitoring and alerting.
- **Service Discovery:** Supports dynamic service discovery through integrations with Kubernetes, Consul, and other service discovery mechanisms.
- **Alerting:** Provides flexible alerting rules based on Prometheus's query language (PromQL), enabling real-time alerting on metrics anomalies.
- **Visualization:** While Prometheus itself focuses on data collection and alerting, it integrates with visualization tools like Grafana for more advanced dashboarding and analysis.

Grafana

Purpose: Grafana is an open-source platform for monitoring and observability, allowing users to query, visualize, alert on, and understand metrics from multiple sources.

- **Visualization:** Grafana offers powerful visualization capabilities, including graphs, charts, and histograms, to create comprehensive dashboards for monitoring and analysis.
- **Data Source Integration:** Integrates with various data sources, including Prometheus, Elasticsearch, InfluxDB, MySQL, and more, enabling unified visualization and analysis across multiple systems.
- **Alerting and Notifications:** Provides alerting features to notify users of important changes or incidents based on predefined thresholds and conditions.
- **Dashboard Templating:** Supports dashboard templating, allowing users to create dynamic, reusable dashboards that adapt to different environments or use cases.

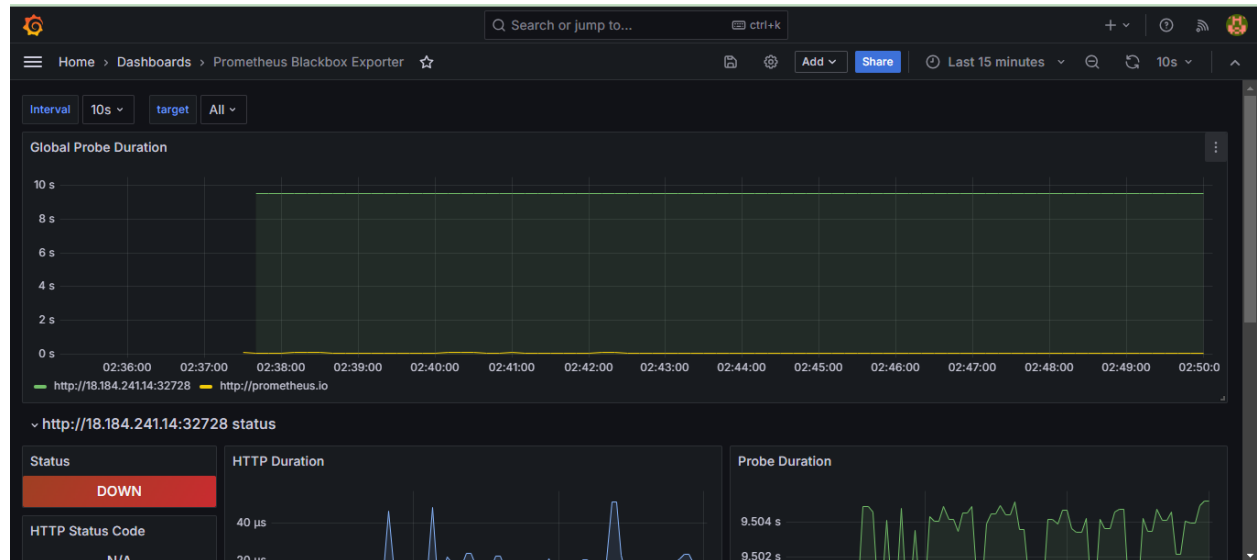
->Installation of the grafana and prometheus and configuring its dashboard

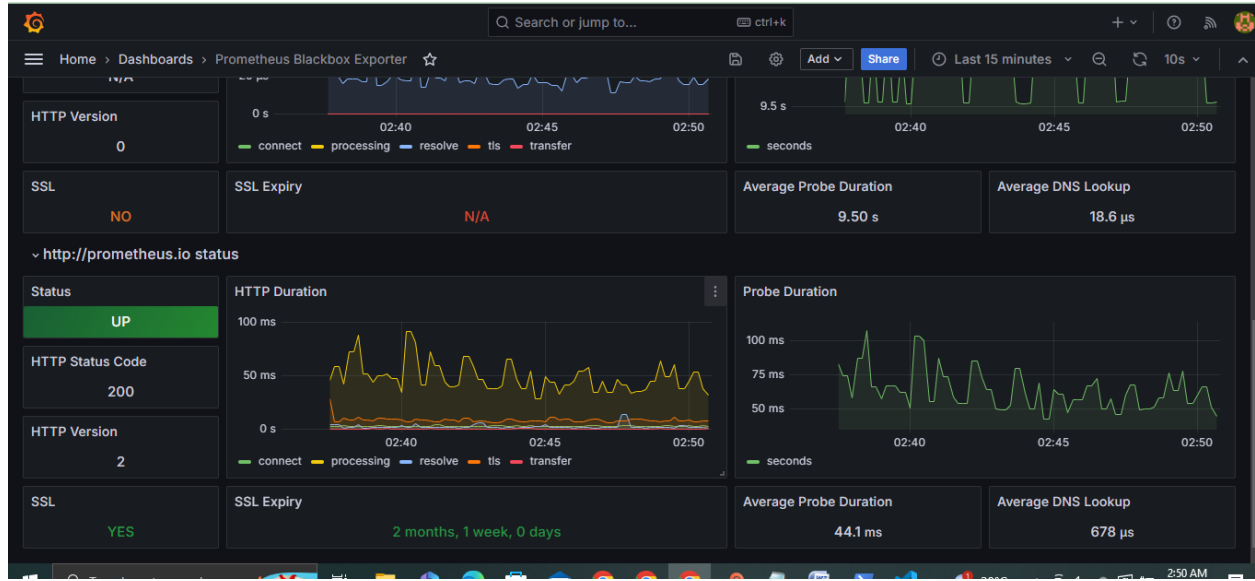
Monitoring of the complete infrastructure of prometheus,application and Jenkins

```
1 JENKINS 2 SONAR-QUBE 3 NEXUS +
Saving to: 'prometheus-2.53.1.linux-amd64.tar.gz'
prometheus-2.53.1.linux-amd64.tar.gz 100%[=====] 99.36M 130MB/s in 0.8s
2024-07-15 20:24:36 (130 MB/s) - 'prometheus-2.53.1.linux-amd64.tar.gz' saved [104191695/104191695]

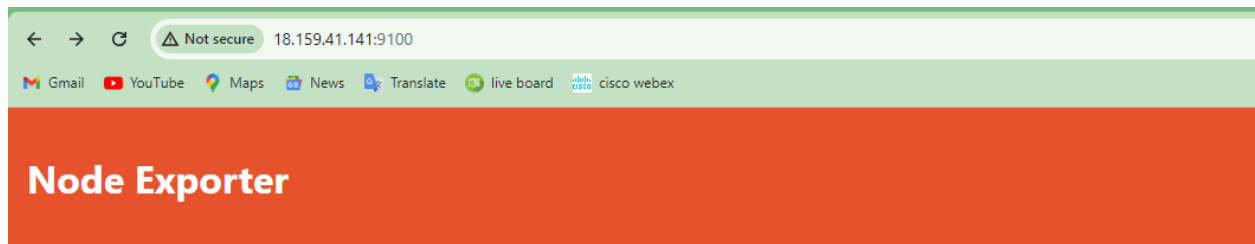
ubuntu@ip-172-31-28-5:~$ ls
prometheus-2.53.1.linux-amd64.tar.gz
ubuntu@ip-172-31-28-5:~$ tar -xvf prometheus-2.53.1.linux-amd64.tar.gz
prometheus-2.53.1.linux-amd64/
prometheus-2.53.1.linux-amd64/prometheus.yml
prometheus-2.53.1.linux-amd64/prometheus
prometheus-2.53.1.linux-amd64/consoles/
prometheus-2.53.1.linux-amd64/consoles/node-disk.html
prometheus-2.53.1.linux-amd64/consoles/node-overview.html
prometheus-2.53.1.linux-amd64/consoles/node-cpu.html
prometheus-2.53.1.linux-amd64/consoles/node.html
prometheus-2.53.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.53.1.linux-amd64/consoles/index.html.example
prometheus-2.53.1.linux-amd64/consoles/prometheus.html
prometheus-2.53.1.linux-amd64/LICENSE
prometheus-2.53.1.linux-amd64/promtool
prometheus-2.53.1.linux-amd64/console_libraries/
prometheus-2.53.1.linux-amd64/console_libraries/menu.lib
prometheus-2.53.1.linux-amd64/console_libraries/prom.lib
prometheus-2.53.1.linux-amd64/NOTICE
ubuntu@ip-172-31-28-5:~$ rm -rf tar -xvf prometheus-2.53.1.linux-amd64.tar.gz
rm: invalid option -- 'x'
Try 'rm --help' for more information.
ubuntu@ip-172-31-28-5:~$ rm -rf tar prometheus-2.53.1.linux-amd64.tar.gz
ubuntu@ip-172-31-28-5:~$ ls
prometheus-2.53.1.linux-amd64
ubuntu@ip-172-31-28-5:~$ cd prometheus-2.53.1.linux-amd64
ubuntu@ip-172-31-28-5:~/prometheus-2.53.1.linux-amd64$ ls
LICENSE NOTICE console_libraries consoles prometheus prometheus.yml promtool
ubuntu@ip-172-31-28-5:~/prometheus-2.53.1.linux-amd64$
```

```
1 JENKINS 2 SONAR-QUBE 3 ec2-3-79-184-139.eu-cent-... 4 NEXUS +
ubuntu@ip-172-31-28-5:~$ sudo /bin/systemctl start grafana-server
ubuntu@ip-172-31-28-5:~$
```





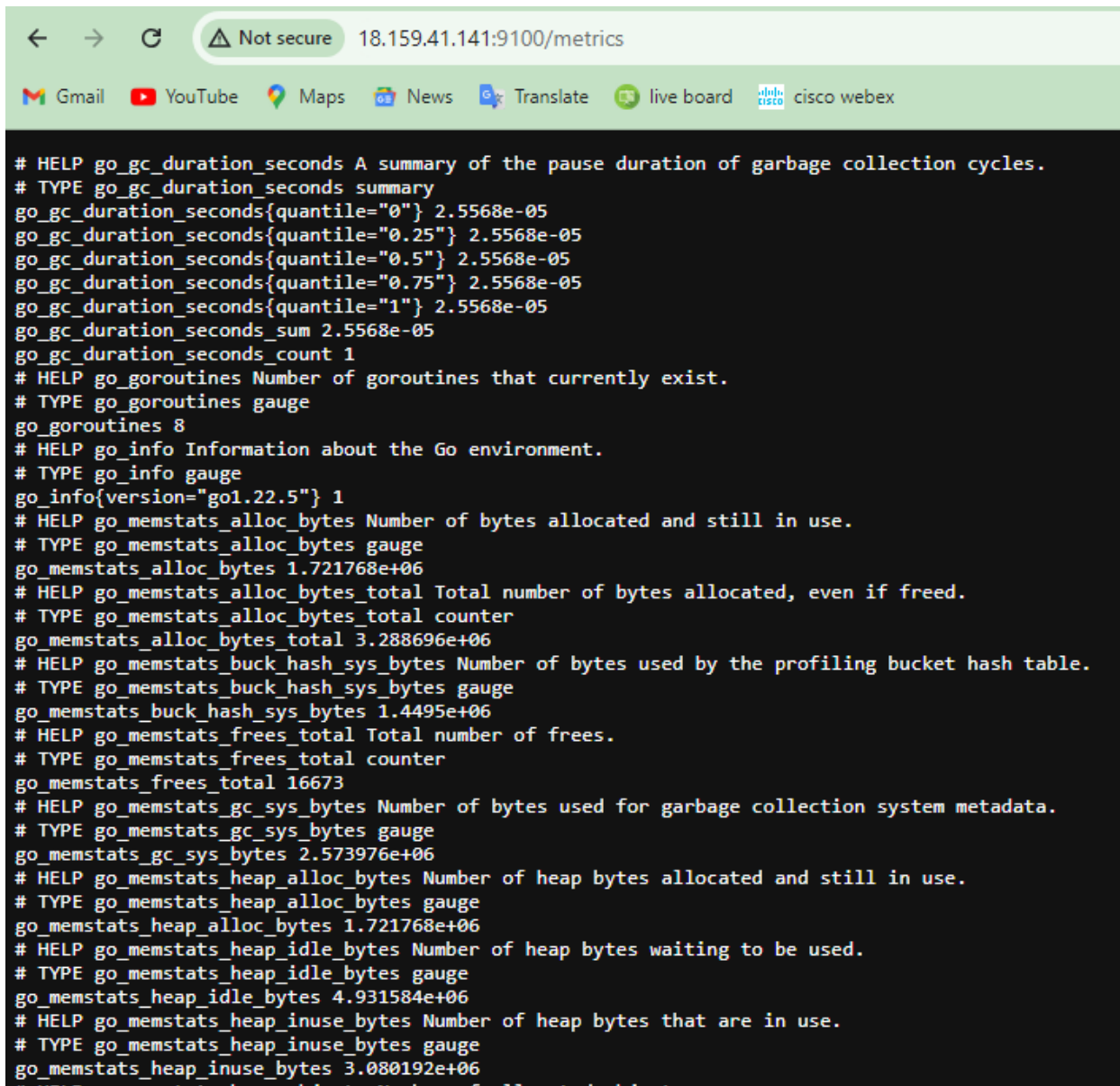
->Node exporter configuration for the view of the metrics



Prometheus Node Exporter

Version: (version=1.8.2, branch=HEAD, revision=f1e0e8360aa60b6cb5e5cc1560bed348fc2c1895)

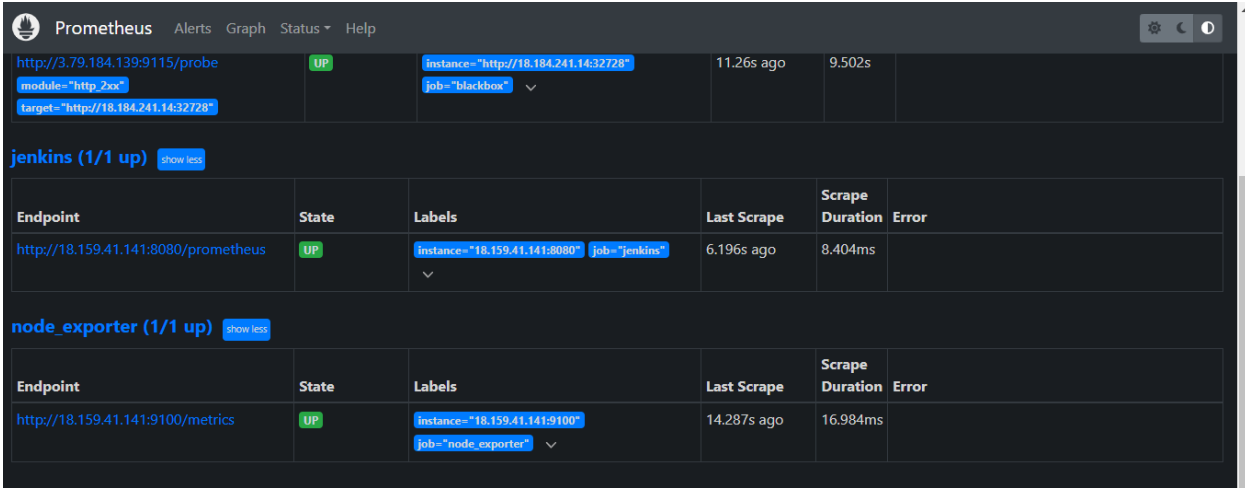
- [Metrics](#)



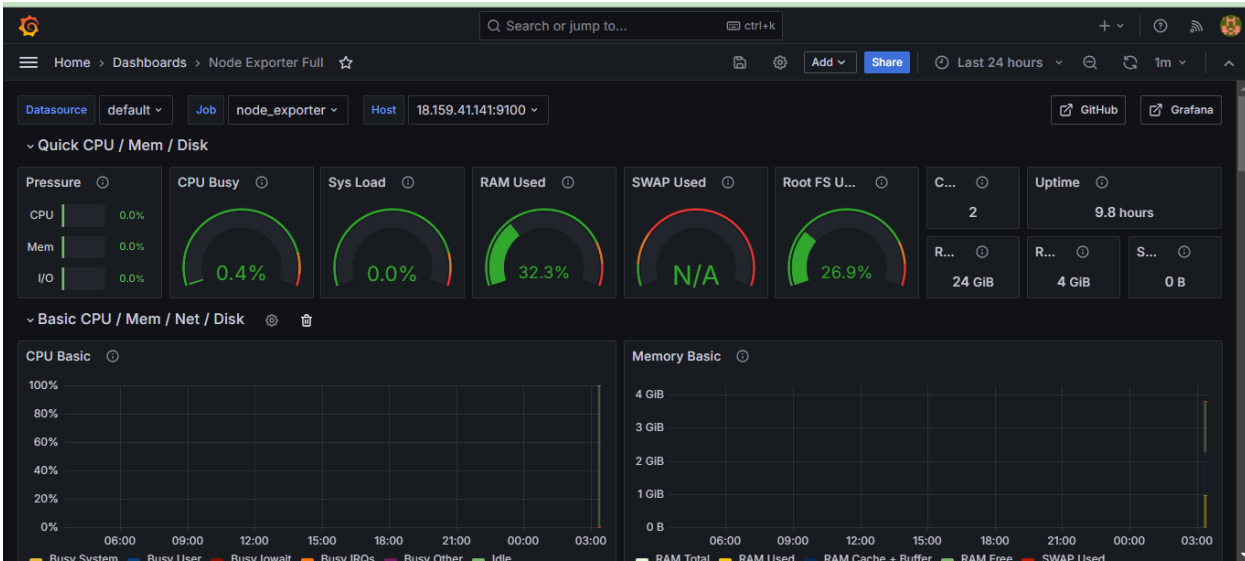
```

# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.5568e-05
go_gc_duration_seconds{quantile="0.25"} 2.5568e-05
go_gc_duration_seconds{quantile="0.5"} 2.5568e-05
go_gc_duration_seconds{quantile="0.75"} 2.5568e-05
go_gc_duration_seconds{quantile="1"} 2.5568e-05
go_gc_duration_seconds_sum 2.5568e-05
go_gc_duration_seconds_count 1
# 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="go1.22.5"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.721768e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 3.288696e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.4495e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 16673
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 2.573976e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.721768e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 4.931584e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 3.080192e+06
# HELP go_memstats_heap_objects Number of objects in use and freed.
# TYPE go_memstats_heap_objects counter
go_memstats_heap_objects 1267
# HELP go_memstats_mcache_bytes Number of bytes used for mcache system metadata.
# TYPE go_memstats_mcache_bytes gauge
go_memstats_mcache_bytes 1.048e+06
# HELP go_memstats_mcache_sys_bytes Number of bytes used for mcache system metadata.
# TYPE go_memstats_mcache_sys_bytes gauge
go_memstats_mcache_sys_bytes 1.048e+06
# HELP go_memstats_next_gc_bytes Number of bytes available for the next garbage collection.
# TYPE go_memstats_next_gc_bytes gauge
go_memstats_next_gc_bytes 1.048e+06
# HELP go_memstats_stack_bytes Number of bytes used for stack system metadata.
# TYPE go_memstats_stack_bytes gauge
go_memstats_stack_bytes 1.048e+06
# HELP go_memstats_stack_sys_bytes Number of bytes used for stack system metadata.
# TYPE go_memstats_stack_sys_bytes gauge
go_memstats_stack_sys_bytes 1.048e+06
# HELP go_memstats_sys_bytes Number of bytes used for system metadata.
# TYPE go_memstats_sys_bytes gauge
go_memstats_sys_bytes 1.048e+06

```



->Monitoring of the jenkins



Conclusion

In conclusion, this CI/CD pipeline establishes a structured approach to software delivery, integrating automated testing, security checks, deployment orchestration, and monitoring. By leveraging Jenkins for build automation, ArgoCD for GitOps-driven deployments, and tools like SonarQube, Nexus, Trivy, Prometheus, and Grafana for quality assurance, artifact management, security scanning, and monitoring, the pipeline ensures continuous improvement in software quality, reliability, and performance across the development lifecycle.

KHUSHI THACKER -21bctc91