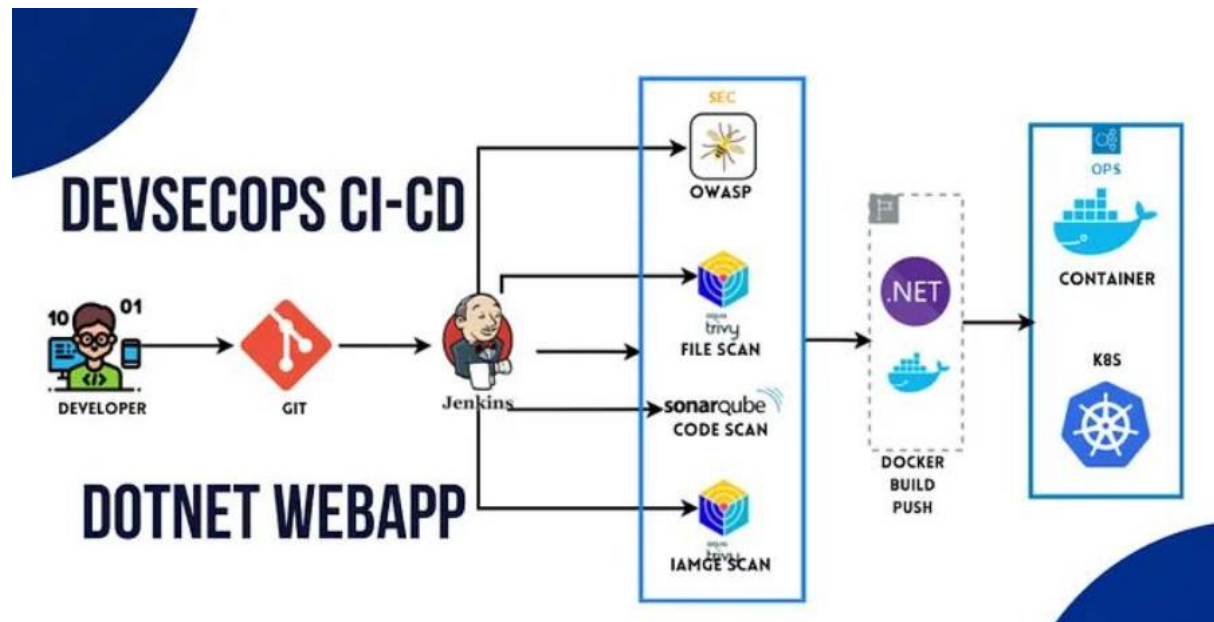# Real-Time DevSecOps Pipeline for a DotNet Web App

Steps:-

Step 1 — Create an Ubuntu T2 Large Instance with 30GB storage

Step 2 — Install Jenkins, Docker and Trivy. Create a Sonarqube Container using Docker.

Step 3 — Install Plugins like JDK, Sonarqube Scanner

Step 4 — Install OWASP Dependency Check Plugins

Step 5 — Configure Sonar Server in Manage Jenkins

Step 6— Create a Pipeline Project in Jenkins using Declarative Pipeline

Step 7 — Install make package

Step 8— Docker Image Build and Push

Step 9 — Deploy the image using Docker

Step 10—Access the Real World Application

Step 11— Kubernetes Set Up

Step 12 — Terminate the AWS EC2 Instance

Step 1 — Launch an AWS T2 Large Instance. Use the image as Ubuntu. You can create a new key pair or use an existing one. Enable HTTP and HTTPS settings in the Security Group.

	Name ✎	▽	Instance ID	Instance state	▽	Instance type	▽	Status check	Alarm status	Availability Zone	▽	Public IPv4 DNS
✓	Dotnet		i-08bd38d38ff9cbebf	⊘ Running ⊕ ⊖		t2.large		⊘ 2/2 checks passed	View alarms ✛	ap-south-1a		ec2-65-0-92-82.a

let's connect to your ec2 via ssh using command ssh -i "laptopkey.pem" ubuntu@ec2-65-0-92-82.ap-south-1.compute.amazonaws.com

run the following commands

a. sudo su

b. apt update

clone the github repo by

git clone https://github.com/sai241194/DotNet.git

Step 2 — Install Jenkins, Docker and Trivy. Create a Sonarqube Container using Docker.

2A — To Install Java and Jenkins

Connect to your console, and enter these commands to Install Java latest version and Jenkins

sudo apt update
sudo apt install openjdk-17-jdk
sudo apt install openjdk-17-jre

```
root@ip-172-31-39-3:~# java --version
openjdk 17.0.10 2024-01-16
OpenJDK Runtime Environment (build 17.0.10+7-Ubuntu-122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.10+7-Ubuntu-122.04.1, mixed mode, sharing
root@ip-172-31-39-3:~#
```

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

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

**echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" **

**https://pkg.jenkins.io/debian-stable binary/ | sudo tee **

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins

```
root@ip-172-31-39-3:~# jenkins --version
2.440.2
root@ip-172-31-39-3:~#
```

sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status Jenkins

```
xecuting: /lib/systemd/systemd-sysv-install enable jenkins
root@ip-172-31-39-3:~# sudo systemctl start jenkins
root@ip-172-31-39-3:~# sudo systemctl status jenkins
 jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor pres
     Active: active (running) since Sat 2024-04-06 18:47:40 UTC; 45s ago
   Main PID: 5849 (java)
      Tasks: 52 (limit: 9498)
     Memory: 2.1G
        CPU: 42.016s
     CGroup: /system.slice/jenkins.service
             └─5849 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/jav

pr 06 18:47:21 ip-172-31-39-3 jenkins[5849]: 486315199b3443c3b56176ad8f524952
pr 06 18:47:21 ip-172-31-39-3 jenkins[5849]: This may also be found at: /var/
pr 06 18:47:21 ip-172-31-39-3 jenkins[5849]: ****************************
pr 06 18:47:21 ip-172-31-39-3 jenkins[5849]: ****************************
pr 06 18:47:21 ip-172-31-39-3 jenkins[5849]: ****************************
```

Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-05a56a1c0627131f8	HTTPS ▼	TCP	443	Custom ▼	🔍 0.0.0.0/0 ✕		Delete
sgr-07fd66983c20d552b	HTTP ▼	TCP	80	Custom ▼	🔍 0.0.0.0/0 ✕		Delete
sgr-01e38e5c11d8929a5	SSH ▼	TCP	22	Custom ▼	🔍 0.0.0.0/0 ✕		Delete
–	Custom TCP ▼	TCP	8080	Anyw... ▼	🔍 0.0.0.0/0 ✕		Delete

Add rule

Now, grab your Public IP Address

```
<EC2 Public IP Address:8080>
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

Unlock Jenkins using an administrative password and install the required plugins.

Jenkins will now get installed and install all the libraries.

Getting Started

Username

admin

Password

•••••

Confirm password

•••••

Full name

Saikarthik

E-mail address

admin@gmail.com

Skip and continue as admin   Save and Continue
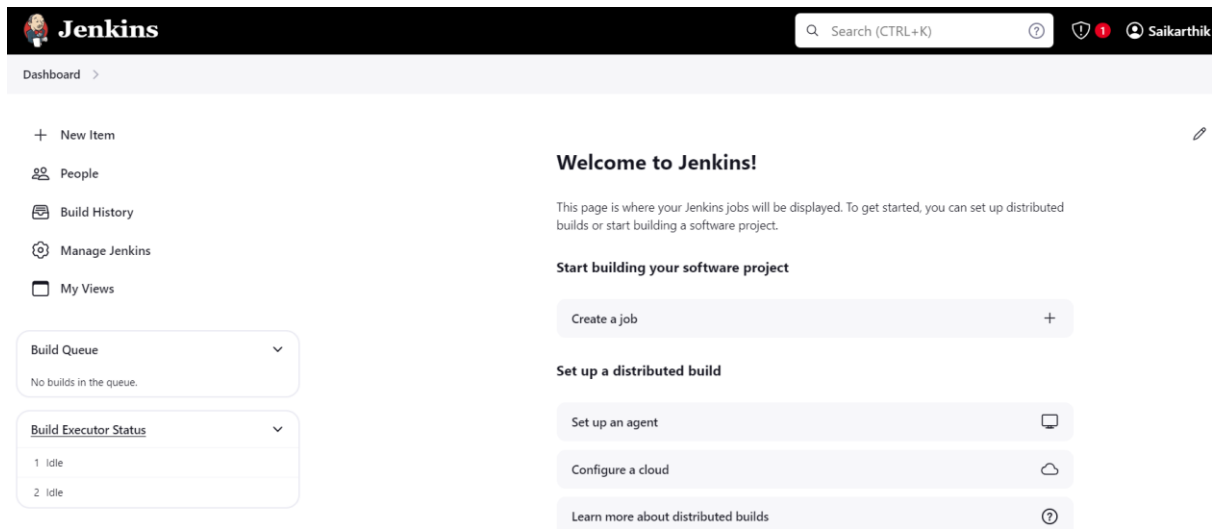
2B — Install Docker

a. **apt-get install docker.io -y**

b. **usermod -aG docker $USER** # Replace with your username e.g 'ubuntu'

c. **newgrp docker**

d. **sudo chmod 777 /var/run/docker.sock**

After the docker installation, we create a sonarqube container (Remember added 9000 port in the security group)

Security group rule ID	Type	Info	Protocol	Info	Port range	Info	Source	Info		Description - optional	Info	
sgr-05a56a1c0627131f8	HTTPS ▼		TCP		443		Custom ▼	Q	0.0.0.0/0 ✕			Delete
sgr-07fd66983c20d552b	HTTP ▼		TCP		80		Custom ▼	Q	0.0.0.0/0 ✕			Delete
sgr-0ee499d4522d45741	Custom TCP ▼		TCP		8080		Custom ▼	Q	0.0.0.0/0 ✕			Delete
sgr-01e38e5c11d8929a5	SSH ▼		TCP		22		Custom ▼	Q	0.0.0.0/0 ✕			Delete
–	Custom TCP ▼		TCP		9000		Anyw... ▼	Q	0.0.0.0/0 ✕			Delete

Add rule

run the following commands to install and run the conatainer of sonarqube on port no. 9000

docker run -itd --name sonar -p 9000:9000 sonarqube:lts-community

```
root@ip-172-31-39-3:~# docker run -itd --name sonar -p 9000:9000 sonarqube:lts-c
ommunity
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
23828d760c7b: Pull complete
cfcf356fa6e6: Pull complete
b345ba1396e8: Pull complete
33e0a682e40f: Pull complete
9fa5752204b5: Pull complete
70c5a1f7be5b: Pull complete
c7f4e3bed4fe: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:7380e0ec0ebe276e829df5bfbaaf7e58e44eb5adb1873382905577c8d35a0a2c
Status: Downloaded newer image for sonarqube:lts-community
2b974ac86b1478aae25fc9969b1679e3aba05209451ed29816e0710e9a77ef6d
root@ip-172-31-39-3:~#
```

Now browse **http://your_public_ip:9000** you will see **sonarqube is running**

username =admin
password=admin
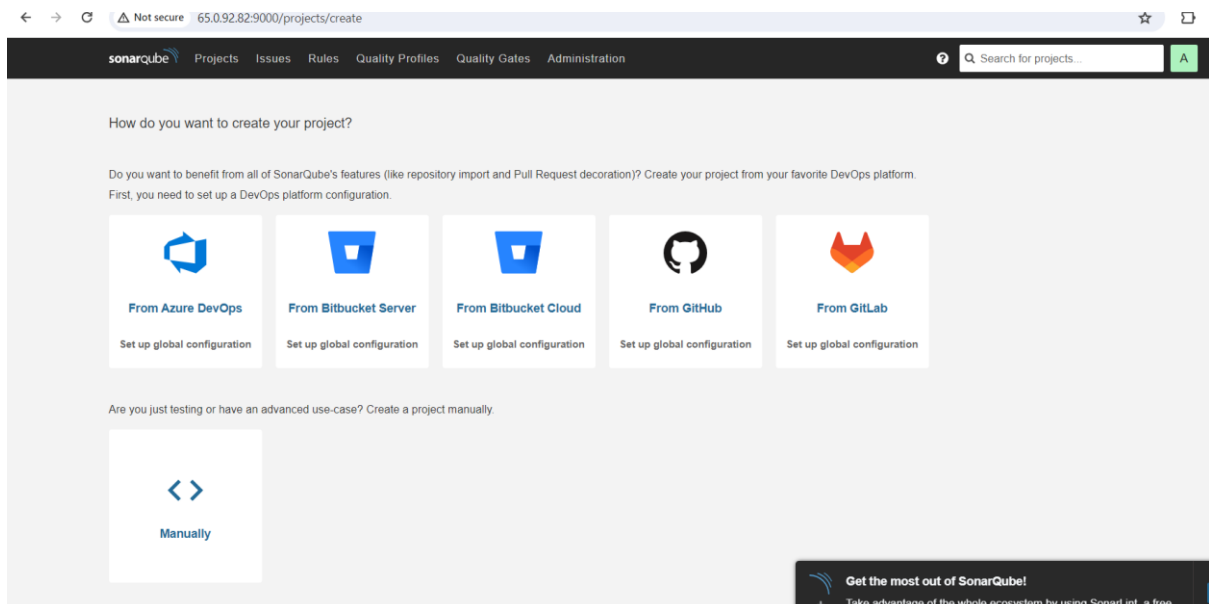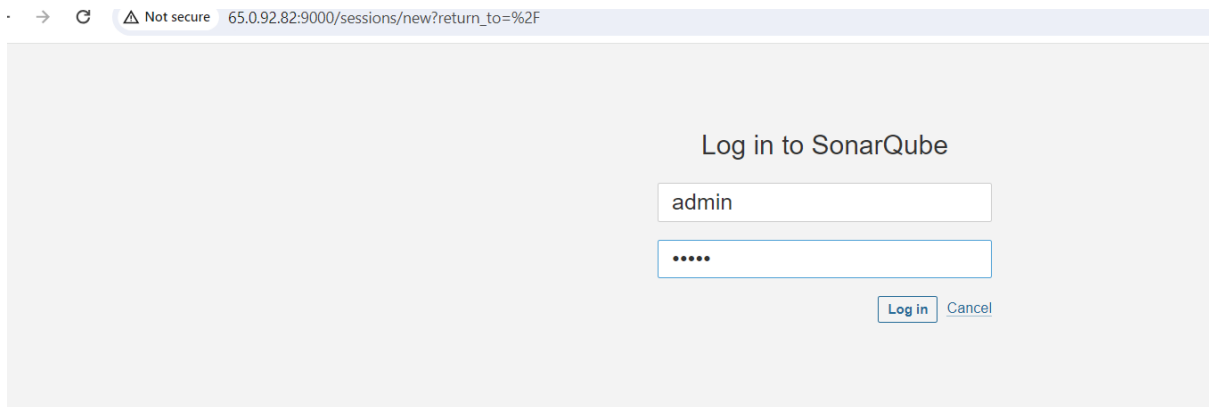
2C — Install Trivy

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

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

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

sudo apt-get update

sudo apt-get install trivy

```
root@ip-172-31-39-3:~# trivy --version
Version: 0.50.1
root@ip-172-31-39-3:~#
```

now your trivy is ready to check and scan your image for any vulnerabilities

to check run the following commands

a. trivy image <image id>

```
root@ip-172-31-39-3:~# trivy image 1e64814986e1
2024-04-06T19:11:45.398Z      INFO    Need to update DB
2024-04-06T19:11:45.398Z      INFO    DB Repository: ghcr.io/aquasecurity/trivy-db:2
2024-04-06T19:11:45.398Z      INFO    Downloading DB...
44.86 MiB / 44.86 MiB [------------------------------------------------------------------------------------
2024-04-06T19:11:49.447Z      INFO    Vulnerability scanning is enabled
2024-04-06T19:11:49.447Z      INFO    Secret scanning is enabled
2024-04-06T19:11:49.447Z      INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2024-04-06T19:11:49.447Z      INFO    Please see also https://aquasecurity.github.io/trivy/v0.50/docs/scanner/secret/#recommendation for faster
2024-04-06T19:11:57.791Z      INFO    Java DB Repository: ghcr.io/aquasecurity/trivy-java-db:1
2024-04-06T19:11:57.791Z      INFO    Downloading the Java DB...
519.32 MiB / 519.32 MiB [----------------------------------------------------------------------------------
2024-04-06T19:12:35.372Z      INFO    The Java DB is cached for 3 days. If you want to update the database more frequently, the '--reset' flag
2024-04-06T19:12:35.988Z      INFO    Detected OS: ubuntu
2024-04-06T19:12:35.988Z      INFO    Detecting Ubuntu vulnerabilities...
2024-04-06T19:12:35.995Z      INFO    Number of language-specific files: 1
2024-04-06T19:12:35.995Z      INFO    Detecting jar vulnerabilities...
```

Step 3 — Install some suggested plugins for pipeline to run without errors

Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)

3 → OWASP Dependency Check (Install without restart)
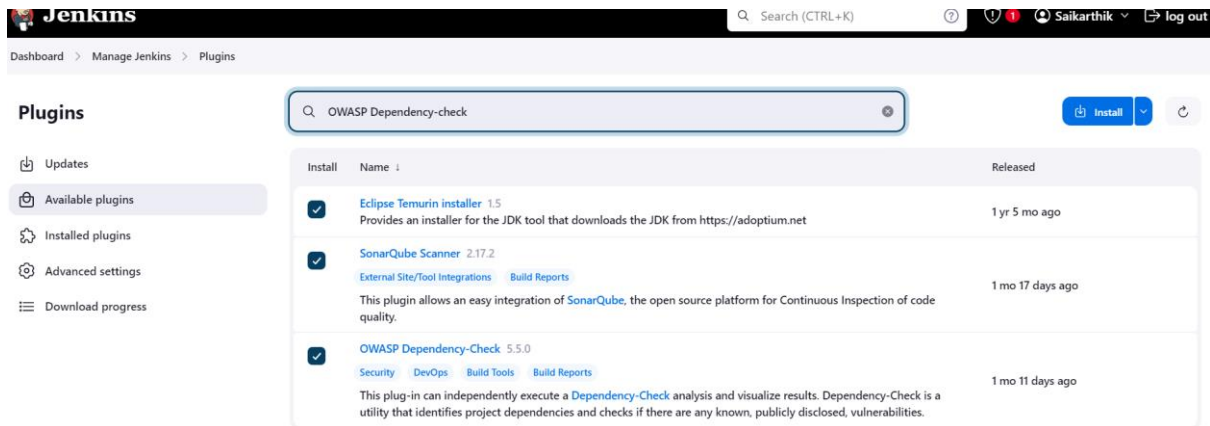
3A — Configure Java in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK17→ Click on Apply and Save

1. click on add jdk and select installer adoptium.net

2. choose jdk-17.0.10+7 version and in name section enter jdk 17
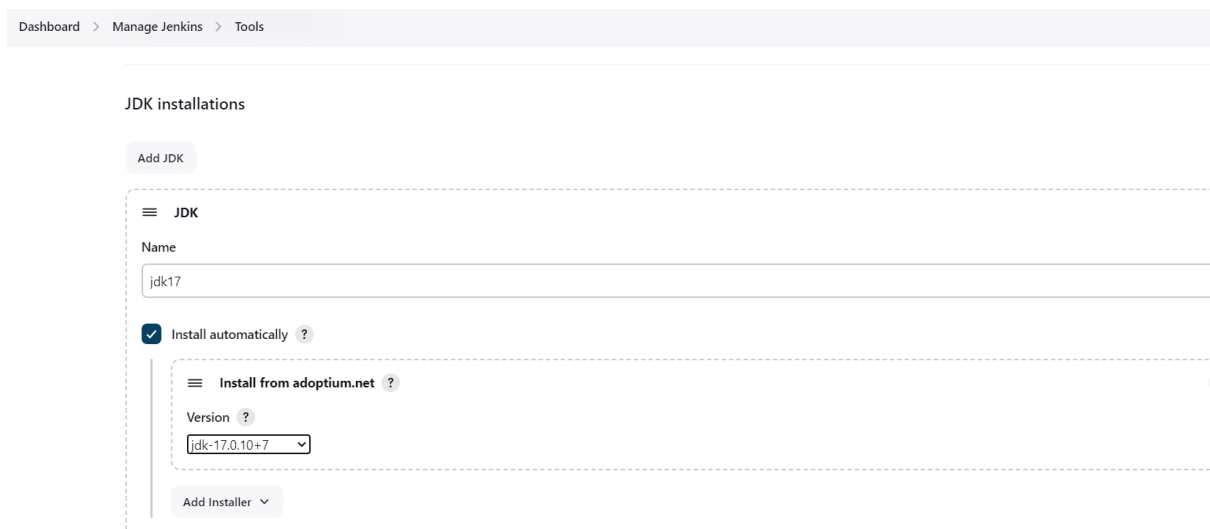
Next is to add dependency check

1. add dependency check

2. name = DP-Check

3. from add installer select install from github.com

Click on apply and Save here.

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token
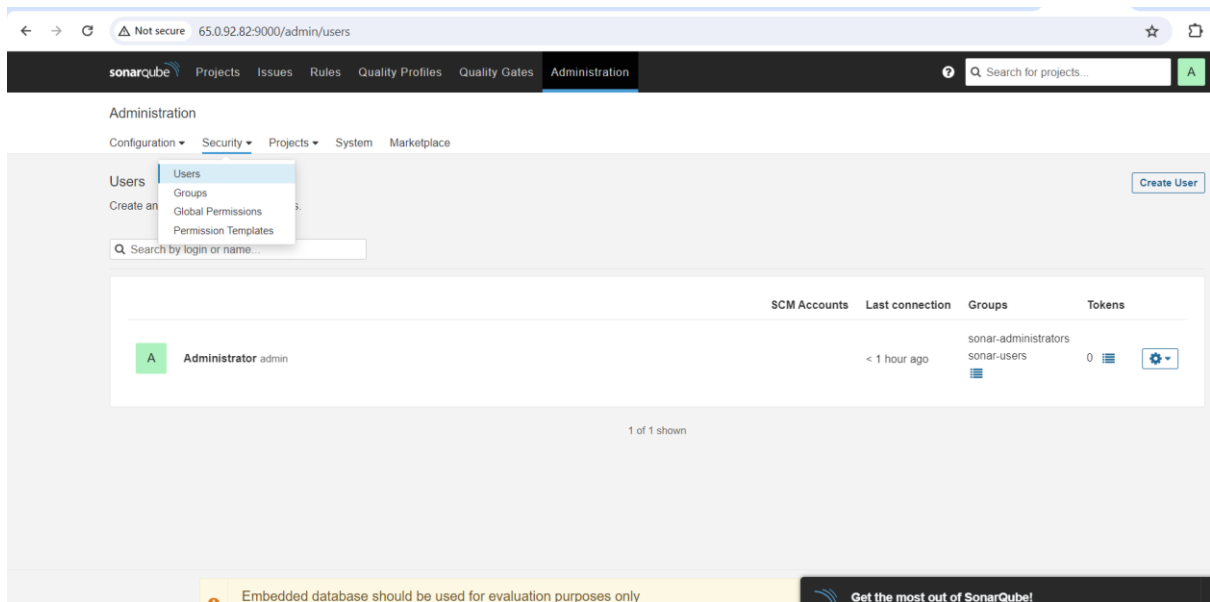
Click on Update Token
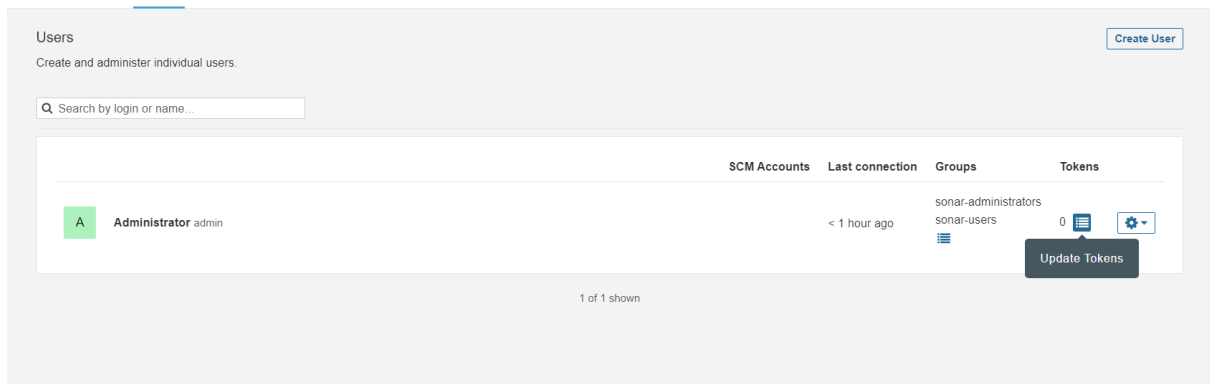
Create a token with a name and generate

Copy this Token

Goto Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this

Enter the Token in secret field

You will this page once you click on create
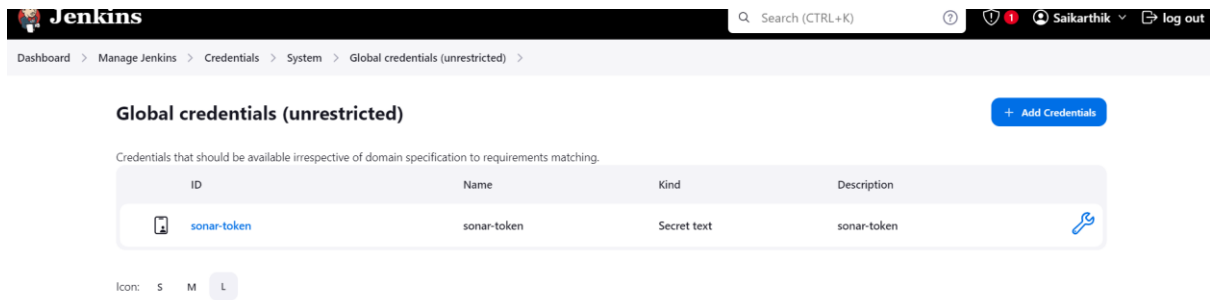
Step 5 — Configure Sonar Server in Manage Jenkins

Now, go to Dashboard → Manage Jenkins → Configure System→ add sonarqube servers

1. name =sonar-server

2. server_url=http://public_ip:9000

3. server authentication token = sonar-token
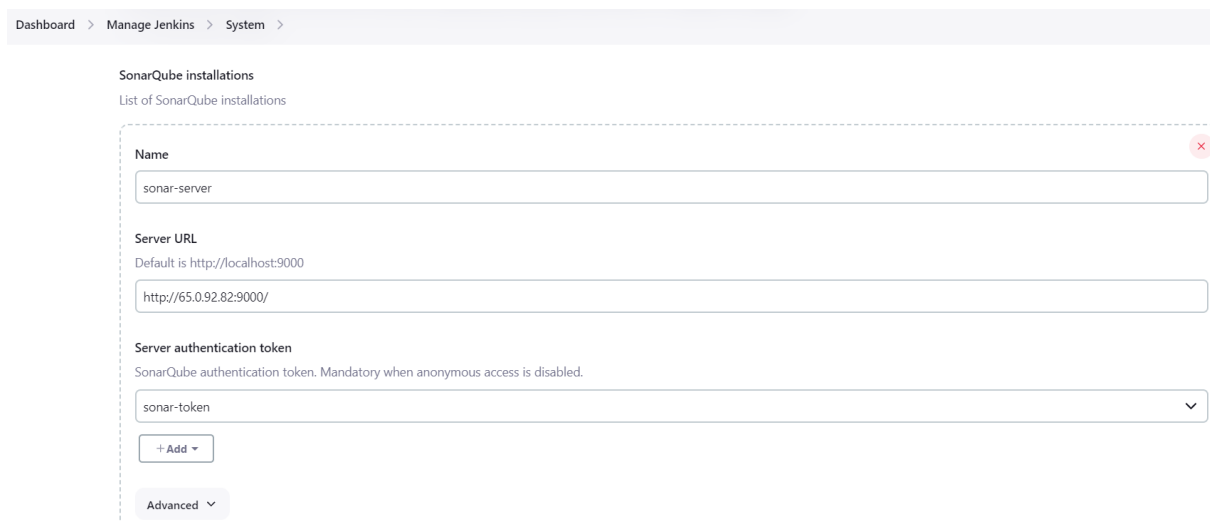
Click on Apply and Save

The Configure System option is used in Jenkins to configure different server

Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

Go To Dashboard → Manage Jenkins → Tools

1. add sonar scanner

2. name =sonar-scanner

SonarQube Scanner installations

SonarQube Scanner installations ^   ✎ Edited

Add SonarQube Scanner

≡   SonarQube Scanner
Name

sonar-scanner

☑ Install automatically ?

≡   Install from Maven Central
Version

SonarQube Scanner 5.0.1.3006

Add Installer ∨

In the Sonarqube Dashboard add a quality gate also

Administration → Configuration →Webhooks
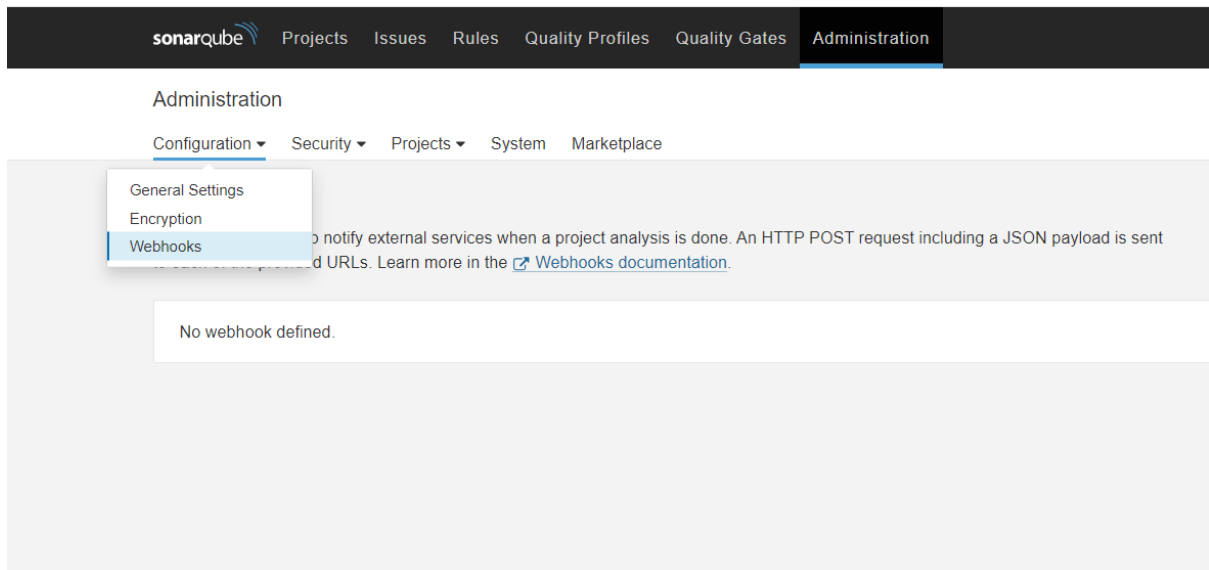
Click on create

#in url section of quality gate
<http://jenkins-public-ip:8080>/sonarqube-webhook/

Create Webhook

All fields marked with * are required

Name *

jenkins

URL *

http://65.0.92.82:8080/sonarqube-webhook/

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.

Create   Cancel

Administration

Configuration ▾   Security ▾   Projects ▾   System   Marketplace

Webhooks   Create

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent
to each of the provided URLs. Learn more in the ⧉ Webhooks documentation.

Name	URL	Has secret?	Last delivery	Actions
jenkins	http://65.0.92.82:8080/sonarqube-webhook/	No	Never	⚙▾

Click on Apply and Save here.

Step 6 — Create a Pipeline Project in Jenkins using Declarative Pipeline

1. **go to new item** →**select pipeline** →in the name section type **dotnet**

2. scroll down to the pipeline script and copy paste the following code

```
pipeline {

  agent any

  tools {

    jdk 'jdk17'

  }

  environment {

    SCANNER_HOME = tool 'sonar-scanner'

  }

  stages {

    stage('clean workspace') {

      steps {
```

```
            cleanWs()

        }

    }

    stage('Checkout From Git') {

        steps {

            git branch: 'main', url:
'https://github.com/sai241194/DotNet.git'

        }

    }

    stage("Sonarqube Analysis ") {

        steps {

            withSonarQubeEnv('sonar-server') {

                sh """$SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Dotnet-Webapp \

                    -Dsonar.projectKey=Dotnet-Webapp"""

            }

        }

    }

    stage("quality gate") {
```

```
        steps {

            script {

                waitForQualityGate abortPipeline: false,
credentialsId: 'Sonar-token'

            }

        }

    }

    stage("TRIVY File scan") {

        steps {

            sh "trivy fs . > trivy-fs_report.txt"

        }

    }

    stage("OWASP Dependency Check") {

        steps {

            dependencyCheck additionalArguments: '--scan ./ -
-format XML ', odcInstallation: 'DP-Check'

            dependencyCheckPublisher pattern:
'**/dependency-check-report.xml'

        }
```

```
        }

      }

  }
```

Click on Build now, you will see the stage view like this

Dotnet - Stage View

	Declarative: Tool Install	clean workspace	Checkout From Git	Sonarqube Analysis	quality gate	TRIVY File scan	OWASP Dependency Check
Average stage times: (Average full run time: ~3min 15s)	13s	338ms	2s	22s	499ms	4s	2min 25s
#4 Apr 07 01:54 No Changes	114ms	323ms	1s	26s	544ms (paused for 2s)	3s	1min 9s

Dashboard › Dotnet ›

📄 Status	
</> Changes	
▷ Build Now	
⚙ Configure	
🗑 Delete Pipeline	
🔍 Full Stage View	
📶 SonarQube	
✎ Rename	
⑦ Pipeline Syntax	

✓ **Dotnet**

SonarQube Quality Gate

Dotnet-Webapp Passed

server-side processing: Success

🌀 Latest Dependency-Check

Permalinks

- Last build (#28), 38 min ago
- Last stable build (#28), 38 min ago
- Last successful build (#28), 38 min ago
- Last failed build (#27), 40 min ago
- Last unsuccessful build (#27), 40 min ago
- Last completed build (#28), 38 min ago

Dependency-Check Trend

Unassigned ● Low ● Medium ● Hi

Build History	trend ⌄
🔍 Filter...	/
✓ #28	📶
Apr 7, 2024, 8:56 AM	

To see the report, you can go to Sonarqube Server and go to Projects.
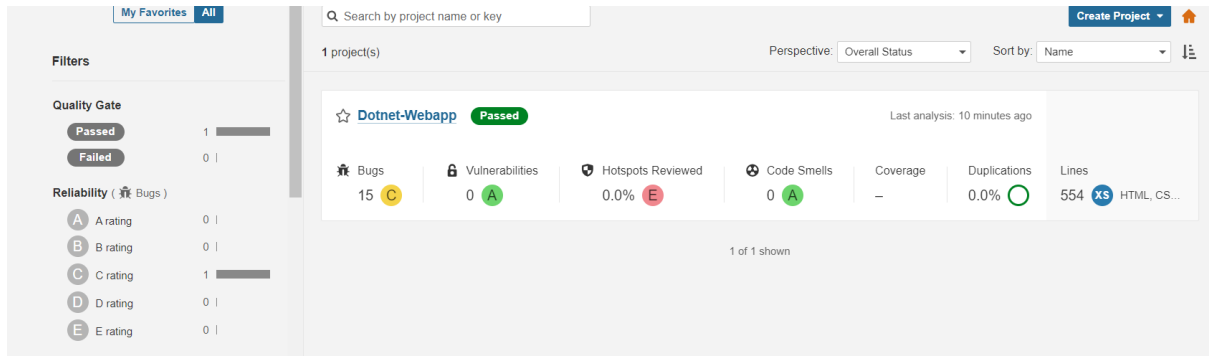
Step 7 — Install make package

sudo apt install make

to check version install or not

make -v

Step 8 — Docker Image Build and Push

We need to install the Docker tool in our system, Goto Dashboard →
Manage Plugins → Available plugins → Search for Docker and
install these plugins

- `Docker`

- `Docker Commons`

- `Docker Pipeline`

- `Docker API`

- `docker-build-step`

and click on install without restart

Now, goto Dashboard → Manage Jenkins → Tools

1. click on add docker

2. name=docker

3. add installer =download from docker.com

Docker installations

Add Docker

≡ **Docker**

Name

docker

☑ Install automatically ?

≡ **Download from docker.com**

Docker version ?

latest

Add Installer ⌄

Click on apply and save

Now go to Dashboard → Manage Jenkins → Credentials

Add DockerHub Username and Password under Global Credentials

Global credentials (unrestricted)

+ Add Credentials

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

	ID	Name	Kind	Description	
	sonar-token	sonar-token	Secret text	sonar-token	🔧
	docker	sai2411/****** (docker)	Username with password	docker	🔧

Icon: S M L

In the makefile, we already defined some conditions to build, tag and push images to dockerhub.

DotNet / makefile ↑ Top

Code | Blame | 70 lines (55 loc) · 2.63 KB | Code 55% faster with GitHub Copilot | Raw ⟨⟩

```
27
28    image: ## 🔨 Build container image from Dockerfile
29          docker build . --file build/Dockerfile \
30            --tag $(IMAGE_REG)/$(IMAGE_REPO):$(IMAGE_TAG)
31
32    push: ## 📤 Push container image to registry
33          docker push $(IMAGE_REG)/$(IMAGE_REPO):$(IMAGE_TAG)
```

that's why we are using make image and make a push in the place of docker build -t and docker push

Add this stage which is highlighted in bold to Pipeline Script

pipeline {

   agent any

   tools {

      jdk 'jdk17'

```
    }

    environment {

        SCANNER_HOME = tool 'sonar-scanner'

    }

    stages {

        stage('clean workspace') {

            steps {

                cleanWs()

            }

        }

        stage('Checkout From Git') {

            steps {

                git branch: 'main', url: 'https://github.com/sai241194/DotNet.git'

            }

        }

        stage("Sonarqube Analysis ") {

            steps {

                withSonarQubeEnv('sonar-server') {
```

```
        sh """$SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Dotnet-Webapp \

            -Dsonar.projectKey=Dotnet-Webapp"""

    }

  }

}

stage("quality gate") {

  steps {

    script {

      waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

    }

  }

}

stage("TRIVY File scan") {

  steps {

    sh "trivy fs . > trivy-fs_report.txt"

  }

}

stage("OWASP Dependency Check") {
```

```
steps {

    dependencyCheck additionalArguments: '--scan ./ --format XML ',
odcInstallation: 'DP-Check'

    dependencyCheckPublisher pattern: '**/dependency-check-report.xml'

}

}

stage("Docker Build & tag") {

    steps {

        script {

            withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){

                sh "make image"

            }

        }

    }

}

stage("TRIVY") {

    steps {

        sh "trivy image sai2411/dotnet-monitoring:latest > trivy.txt"

    }
```

```
        }

    stage("Docker Push") {

        steps {

            script {

                withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){

                    sh "make push"

                }

            }

        }

    }

  }

}
```

When all stages in docker are successfully created then you will see
the result You log in to Dockerhub, and you will see a new image is
created

Stage View

	Declarative: Tool Install	clean workspace	Checkout From Git	Sonarqube Analysis	quality gate	TRIVY File scan	OWASP Dependency Check	Docker Build & tag	TRIVY	Docker Push
Average stage times: (Average full run time: ~1min 19s)	96ms	240ms	1s	13s	335ms	2s	15s	10s	3s	6s
#13 Apr 07 02:54 1 commit	84ms	208ms	1s	11s	314ms (paused for 2s)	2s	7s	3s	2s	21s

Step 9 — Deploy the image using Docker

Add this stage to your above pipeline syntax

stage("Deploy to container"){

   steps{

      sh "docker run -itd --name dotnet -p 5000:5000 sai2411/dotnet-monitoring:latest"

   }

}

(Add port 5000 to Security Group)

You will see the Stage View like this,

And you can access your application on Port 5000. This is a Real World Application that has all Functional Tabs.

Step 10 — Access the Real World Application

← → C ⚠ Not secure 65.0.92.82:5000 ☆

.NET .NET Demo App ⓘ Info ✕ Tools ⬚ Monitor
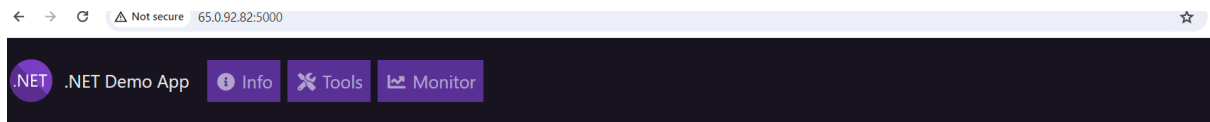
.NET .NET Demo Web App

This is a modern .NET web app using the new minimal hosting model, and Razor pages. It has been designed with cloud demos & containers in mind, and for ease of deployment into Azure or Kubernetes. A simple app to use anytime you want something lightweight to run & deploy.

Basic features:

- System status / information view
- Support for user sign-in with Azure AD and Graph API
- App Insights support

← → C ⚠ Not secure 65.0.92.82:5000/info ☆ ⟲ ◻

.NET .NET Demo App ⓘ Info ✕ Tools ⬚ Monitor

ⓘ System Information

Basic Details

📦 Containerized:	Looks like we're running in a Docker container! 😄
⸬ Kubernetes:	Not running in Kubernetes 😳
⬚ App Insights:	Not enabled
🏠 Hostname:	80fcc65b284f
⚙ OS Details:	Linux 6.5.0-1014-aws
▮ Architecture:	X64 - 2 cores
◈ Framework:	.NET 6.0.28
▦ Memory:	96 MB / 8,120 MB

Environment Variables

DOTNET_VERSION   6.0.28
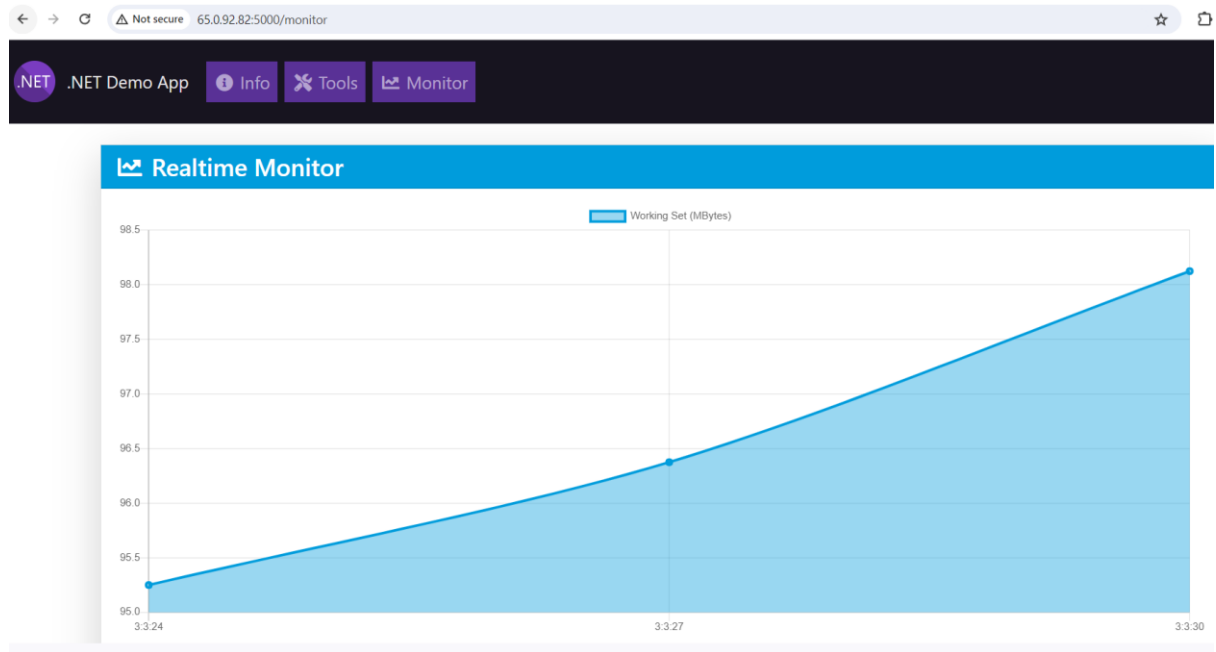
Step 11 — Kubernetes Set Up

Install Kops and Kubectl on Jenkins machine as well.

```
curl -Lo kops https://github.com/kubernetes/kops/releases/download/$(curl -s
https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name |
cut -d '"' -f 4)/kops-linux-amd64
chmod +x kops
sudo mv kops /usr/local/bin/kops
```

```
curl -LO https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
chmod +x kubectl
sudo mv kops /usr/local/bin/kubectl
```

Create one IAM user and give full admin access

Login with AWS CLI and install AWS CLI using apt install awscli -y

Give aws configure and give details (AWS Access Key ID,Secret Access Key,region and output format)

Now we need to export our access key and secret key

export AWS_ACCESS_KEY_ID=$(aws configure get aws_access_key_id)
export AWS_SECRET_ACCESS_KEY=$(aws configure get aws_secret_access_key)

Next create S3 bucket and enable versionong

aws s3 mb s3://apr07bucket --region ap-south-1

mb=make bucket

aws s3api put-bucket-versioning --bucket sep16bucket --versioning-configuration Status=Enabled

```
root@ip-172-31-39-3:~# aws s3 mb s3://apr07bucket --region ap-south-1
make_bucket: apr07bucket
root@ip-172-31-39-3:~# aws s3api put-bucket-versioning --bucket apr07bucket --versioning-configuration Status=Enabled
root@ip-172-31-39-3:~#
```

Next create sshkey for machines

ssh-keygen

id_rsa=private key

id_rsa.pub=public Key

We are now going to give name for cluster by using DNS or gossip-based cluster

DNS:

```
 # export NAME=aws.sai.com
  # export KOPS_STATE_STORE=s3://apr07bucket
Or
```

Gossip-based: free domain shared by K8'S

For a gossip-based cluster, make sure the name ends with k8s.local. For example:
  # export NAME=sai.k8s.local
  # export KOPS_STATE_STORE=s3://apr07bucket

Creating Cluster now

kops create cluster --zones ap-south-1b ${NAME}

Control Plane-master
Node-worker

To edit node:

kops edit ig --name=sai.k8s.local nodes-ap-south-1b

ig=instance group

To edit control plane:

kops edit ig --name=sai.k8s.local control-plane-ap-south-1b

Update cluster:

kops update cluster --name sai.k8s.local --yes –admin

List clusters: **kops get cluster**

```
root@ip-172-31-39-3:~# kops get cluster
NAME                CLOUD   ZONES
sai.k8s.local       aws     ap-south-1b
root@ip-172-31-39-3:~#
```

Validate Cluster:**kops validate cluster**

```
root@ip-172-31-39-3:~# kops validate cluster
Using cluster from kubectl context: sai.k8s.local

Validating cluster sai.k8s.local

INSTANCE GROUPS
NAME                       ROLE           MACHINETYPE   MIN   MAX   SUBNETS
control-plane-ap-south-1b  ControlPlane   t3.medium     1     1     ap-south-1b
nodes-ap-south-1b          Node           t3.medium     1     1     ap-south-1b

NODE STATUS
NAME                ROLE           READY
i-03aa42b2c6cd3263f control-plane   True
i-0fc0ce824b0bb8b7c node            True

Your cluster sai.k8s.local is ready
root@ip-172-31-39-3:~#
```

List Nodes: **kubectl get nodes --show-labels**

root@ip-172-31-39-3:~# kubectl get nodes --show-labels
NAME   STATUS   ROLES   AGE   VERSION   LABELS
i-03 aa42b2c6cd3263f   Ready   control-plane   156m   v1.28.6   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=t3.medium,beta.kubernetes.io/os=linux,failure-domain.beta.kub
rnetes.io/region=ap-south-1,failure-domain.beta.kubernetes.io/zone=ap-south-1b,kops.k8s.io/kops-controller-pki=,kubernetes.io/arch=amd64,kubernetes.io/hostname=i-03aa42b2c6cd3263f,kubernet
s.io/os=linux,node-role.kubernetes.io/control-plane=,node.kubernetes.io/exclude-from-external-load-balancers=,node.kubernetes.io/instance-type=t3.medium,topology.ebs.csi.aws.com/zone=ap-so
th-1b,topology.kubernetes.io/region=ap-south-1,topology.kubernetes.io/zone=ap-south-1b
i-0fc0ce824b0bb8b7c   Ready   node   154m   v1.28.6   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=t3.medium,beta.kubernetes.io/os=linux,failure-domain.beta.kub
rnetes.io/region=ap-south-1,failure-domain.beta.kubernetes.io/zone=ap-south-1b,kubernetes.io/arch=amd64,kubernetes.io/hostname=i-0fc0ce824b0bb8b7c,kubernetes.io/os=linux,node-role.kubernet
s.io/node=,node.kubernetes.io/instance-type=t3.medium,topology.ebs.csi.aws.com/zone=ap-south-1b,topology.kubernetes.io/region=ap-south-1,topology.kubernetes.io/zone=ap-south-1b
root@ip-172-31-39-3:~#

Now Give this command in CLI

cat /root/.kube/config

root@ip-172-31-39-3:~# cat /root/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMrRENDQWVDZ0F3SUJBZ01NRjhQcjBYcGF4QVdSY00wc01BMEdDU3FHU01iM0RRRUJDd1VBTUJneEZqQVUKQmdOVkJBTVREV3QxWW1WeWJtVjBaWE1
TJFdU0hoY05Na1F3TkRBMU1EWXpNa1E0V2hjTk16UXdOREExTURZegpNa1E0V2pBWU1SWXdGQV1EV1FRREV3MXJkV0psY201bGRHVnpMV05oTU1JQklgQU5CZ2txaGtpRz13MEJBUUVGCkFBT0NBT3BUUlJQkNnS0NBUUVBdj1RVV1hNY1U0QnFxM1J2
o4NzV1UnZWcW1hQTd2T2pJNjJBZDRES0RMSUsKbUVkbnpmQmZZR2g1emRuUnFHd2FyVjBTYmtsU01Va0xORk1NQzFrZGtGSzN5bStxam1vUUJyUkVKOFVQQWJNVApr ejdRNkpLd1hERE5VV1dObWNRL2hnaW4zNU14K1Zac1prdU5oQz24xYkgyS
NW01dgpnSHhBMkhxcy9vSndZNUpiWE5EE21LaG8zYnN1dDczMFFWYW43K3FTRXdJREFQUJvMEl3UUBT0JnT1zIUThCCkFmOEVCQU1DQVFEd0R3WURWUjBUQVFIL0JBVKdBd0VCL3pBZEJnT1zIUTRFRmdRV083MEJkYTVCQ00wSHVFNVAKUFhzdHk
GovQmt3RFFZSktvWklodmNOQVFFTEJRQURnZ0VCQU1ER1J2NEJvVXd5M200ZT13WFFSR3hobFFhawpOeXY4UWF0l3FFNzhlYjU1WXdwMDBIUzNnd2drNHIZO0ZSVDN4N2RKRC91R21aQy85N0h1YTRDZXMzRmJSUDFJC1loZUpHM1FLVFVzdFN6VXqy
1IRDAlQkpIcmgzT1BUNHpvS1RScFB1V0w4akFla31LbmsxTjR3ek2ua3todFMKOGpkVFN3QXIIenNZcWtHb3gwSU5PYW1hSTBOdTVUcit1Tlo3EG15MnlRakpsDF6SGRuUDdML2hTMXRlaWtLwpPcWNpaWxoeEd2OFJHNkhXSDJTZXh5NVVKNEp6d
EWldVV1pwS1hVcTF0bHEzaFR5TUN6MXArZDJIaU50enA2CjFyblZnenZ0Rldsb3JVMkYwbTVSQjkzZklJVXhUVTM0bmxlQWttaXR4TlV4bXk0N0ZWUFFtLzRrenGpNFQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://api-sai-k8s-local-9k179a-4cafcbfd6b09b020.elb.ap-south-1.amazonaws.com
    tls-server-name: api.internal.sai.k8s.local
  name: sai.k8s.local
contexts:
- context:
    cluster: sai.k8s.local
    user: sai.k8s.local
  name: sai.k8s.local
current-context: sai.k8s.local
kind: Config
preferences: {}
users:
- name: sai.k8s.local
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURLVENDQWhZ0F3SUJBZ01SQUpzVVdkOVBxTStsL3g5NDRxWjJaTjR3RFFZSktvWklodmNOQVFFTEJRQXcKR0RFV01CUUdBMVVFQXhNTmEzVmlaWEp1W1
bGN5MWpZVEF1RncweU5EQTBNRFV3TmpNek1EUmFGdzB5TkRBMApNRGd3TURNek1EUmFNREF4RnpBVkJnT1ZCQW9URG5ONWMzUmxiVHB0WVhOMFpYSnpNU1V3RXdZRFZRUURFd3hyCmRXSmxZM1puFfhKdmIzUXdnZ0VpTUEwR0NTcUdTSWIzRFFFQkF
UFBNElCRHdBd2dnRUtBb01CQVFDMHRCZWMKS2pZRjdRTHA2R1JLWUF3Z2ZqTGEvSVdCWFVZdnhxL2JJcWZmK29WEndsTkoyZ0TExdDEGQVpCRmh3TGJtdlRkWQp5bnRzOTNZU1NycmdyR2FSa3pVMFBFcFB0dFVnN2l3dndYEzFEaDBUdENWYU1BZyt
h4RHV0dWFZczNKWGdvCjVzWDRURm1CVmFFUW1oUmY1bW9vclR5ejJsaSti2WNvUnRVbG1tTEd1VTNpaD1EYzlKc2JXdWIvWDhCaE5VLJcKbWkrV0ZhbGc4c1Z2YTc2WF1KRm1GMU9NS1NvK0QyaFNSkNvZXRBdFcrNnhhN1AwWUswwWNNb2hZZDBId
3VApydWhzOTdDOUtNNnVXRTVwMXBmQUtIM0pFQmxOZEt10WtQdFRQTVYwaGE0Tl8SZS9XNnFHNzkvcmtgaGVFNVStCi9DM3B6NDFnQ210cE43WnBBZ01CQUFHalZqQlVNQTRHQTFVZER3RUIvdlFFQXdJSGdEQVRCE05WSFNVRUREQUsKQmdncka
UWNEQWpBTUJnT1zIUk1CQWY4RUFqQUFNQjhHQTFVZE13UU1NQmFBRkR10UFYV3VRUWpOQjdoTwpUejE3TGN0aDQvd1pNQTBHQ1NxR1NJY1NEUUVCQ3dVQUE0SUJBUUF3cmdmZWczTmg3cXhYRjBRQXhsbWhIdlR2C1Y4NVd1ejQ4M3BhcmhXUzJPEW1
FVvejFldjBOM29QQkdmVW1yTmZCQjdLeEd4cj11SFA4QWtvWFpCUU5KdG1KYk5QeVNhQTN6QUEr2FJkdzB6YUtlV1dFYzdqcDhPW1ZhVFVSSXAzZmFCOXNQMnYvSn1HbWRnRWdZWEJDdG12ZApHdlhDWkJjTWVGTVBzd3V5UTM0Tl0yTG1DSVFyODI1
JWdnYvWFc3eG53UkVIWS8vVE5GT1dWbVVpOWxWSmZNCnRCS3V4Y2xISmtQOXdCYnpXb3hkZVMxOElXblk3eWYxQTlfYWt5cjRsdjBDVjY0Ym14TThMOWIvEisvaVVGYjEKbC05RINsRENLblBLVk14T3BnRnBHaU1uL11YV3RWNkxncUNjMy9pSV1Vc
GVm5pc1laVFU4UGNHV1JwCi0tLS0tRU5EIENFU1JRk1DQVRFLS0tLS0K
    client-key-data: LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSU1Fb2dJQkFBS0NBUUVBdExrRWG5DbzJCETBDNmVoa1NtQU1J3DR5MnZ5RmdWMU0MOGF2MjhLbjMvcUZYOEpUC1Nkbm1xb2VoUU dRU11jQzI1cjAzV01wN2JQZD
a3E2NEt4bWtaTTF0RHpxVDdiVk1PNXNMOEY0T1E0SEU3UWwRV2pBSVBgdHNjUTdyYm1tTE55VjRLfZJGK0V4WmdWV2hFSm9VWCtacUtLMDhzOXBSdm0zbktFYlEKWXBpeHJsTgo0b2SRM1BTYkcxcm0vMS9BWVRWUDhKbJSsaFdwWVBLMWlydStsMkN
mhkVGpDa3FQ2zlvVWtYd3FIcVFMVnI1CnNKdXo5R0N0TW5ES01XSGRCOEV1MDY3b2JQ2Xd2U2pPcmxoT2FkYVh3Q2g5eVJBW1RYU252WkQ3VXp6RmRJV3UKRFQwWHYxdXFod89mMjVJNFhoTlSadnd0NmMrTl1BcGphVGUyYVFJREFRQUJBb01CQUJD
xFSkJORj1NN1NTUwptb3Fka045MjFySjJWc2VPeWIvZkxRd3ZMZnlKWXE5dEJ2cVR6NVM2aVJxQ18YczFGMVhONzRQdWF6Ql4NzB3CklVSXBjZTFScDZQWkorWVRzak9vMERQKlBrdl10S0MzZkdwd1dJRm1NVW1jQmcrSUR5a0vwcUU2dHRoYU1pb
KdEFmekxEVzBnUENsSHhxS0pGK3EkWGFrTWpPVS9tYWJ1MXNrd3JPTjIzZEpjcmxRNEpXSzVTZEswc0lveGVycgpvT25tYzlGdzJmeGtwRnZPTnZ4U3RRdGIrTmRBcWxPa3p6N3ROSkhqVGJpb2ZsTj1QUzAzSDJydHJxcXQraD16Cm5WV1hXMHpLdn
d2QwTUEzcHRQaXRpd0EyOWhHb0RpVm90dzB5bUpaNGRYRlBYNG1mUEM3TjFqbHMwaHdFbXkKckxJGzdVVUNnWUVBN1ZJTExLMG1wSm8yanRrSUFRTXpaZExvUVhvelBpcEZCNGhod2gydWFrRFBxbUhzQU1CMApwSU00R3FkWFdka3hKQ2FndWExR2c
FBQVfdNVTM0Ml1TRVN1SDJ5bTRVd1N5TXBIVVBrOEw0VzVaT1V5bjQyC1BVeXdvMlh0ZW1hSFNzUk1PVFdTSW9KZ1zWG1wVGZFdUQzcjVtcn16MZ9KZEJaL3M4YU9SdE1DZ11FQXhrUzUKbm1KdVU0Unp6ZFolR3JXdytX8TczQW18OUklTUt4emZQ
ppWm10bFVTaVAzdGFWFBLaHpxb3FXNTVXVk5OVQphNTAwL212UWJCNG5oRXVnNGdqM1AzVzRmNUswZGpnR3VFVXFzNlo0bnZwEENWrZ8QklJEDR3bGhUTW5LdDlZCmFlcmlYckhVZitIVzRwWkxXdXpEdDA5b1hjU0xjODVxTk9jNEpGTUNnWUFEc
HeXdNQ2RWRG1NSTJ4MjU2QUYKKlQvazJKSzZabHhVZEJobXdhVWcxQWp5d1R6Ri9rTlkzU0VqZW93c1RIcjBBUkFFSGZIQTNla21KQlFrdFJjUwpqVDBIWGFSVnc5SXRNMDdRd1dONVVPWjNsTTZDSzErYnMvd05kR3c1UX1aY1RhRitGW1JLcXhmd
YnShR1BZCkxGSHVmcjE1L01gdHBmVTUxdV1rT1FLQmdCTDNM1jVOa2hvaEdHY39ObFJzdnVJTW56YThINThTSVkvYU5PeWQKDE9hTERpVUdTVVRhcWI1QlkyS1Ct1eRELEzUyWG5ES3h4Z2owVFRxYk5Tk8zTFBpZ1pfdWNoaVo4dk5BQnJ5WApCMCt
SszSkt50UExRk5MSGs0NU5MbWFlQXdJMDA5R2JhTDdSY1YlMS9GU31lTQ01kMnhJaDBZNDc5SURFN3U5Cnhtd0xBb0dBSXkyV21VaGVyT1VNV1NDMk8rK003d2I2L2d2SHdHMDgJOXpNUmZFUnYraG1VaURnbnRzTONSc0QKdmZDU3ZDQURJWmRoWEdo
Z1d1VVys1QnIvSDE0dHBQU05PZ2pPM1dsMmMwOEtwekkzeEZGeENrNE1MWEFqOGp2RTZneFZIbEt4RnMzNUVWc2pxaC8wRndnTERnUkRPeGZQV21UNHN1RT1wOUVRREFXMW9Ci0tLS0tRU5EIFJTQSBQUk1WQVRFIEtFWS0tLS0tCg==

Copy the config file to Jenkins master or the local file manager and save it as secret-file.txt

Now, goto the Master Node

Install Kubernetes Plugin, Once it's installed successfully

goto manage Jenkins → manage credentials → Click on Jenkins global → add credentials

New credentials

Kind

```
Secret file
```

Scope ?

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

File

⬆ Choose File   secret-file.txt

ID ?

```
k8s
```

Description ?

```
k8s
```

Create

Install Kubernetes Plugin, Once it's installed successfully

Plugins

| | Q kuber | / | Install ⌄ ↻ |

Updates
Available plugins
Installed plugins
Advanced settings

Install	Name ↓	Released
☑	**Kubernetes Client API** 6.10.0-240.v57880ce8b_0b_2 kubernetes Library plugins (for use by other plugins) Kubernetes Client API plugin for use by other Jenkins plugins.	2 mo 11 days ago
☑	**Kubernetes Credentials** 0.11 kubernetes credentials Common classes for Kubernetes credentials	7 mo 11 days ago
☑	**Kubernetes** 4203.v1dd44f5b_1cf9 Cloud Providers Cluster Management kubernetes Agent Management This plugin integrates Jenkins with Kubernetes	18 days ago
☑	**Kubernetes CLI** 1.12.1 kubernetes Configure kubectl for Kubernetes	7 mo 10 days ago
☑	**Kubernetes Credentials Provider** 1.262.v2670ef7ea_0c5 kubernetes credentials Provides a read only credentials store backed by Kubernetes.	1 mo 7 days ago

The final step to deploy on the Kubernetes cluster, add this stage to the pipeline.

stage('Deploy to k8s'){

```
steps{

    dir('K8S') {

        withKubeConfig(caCertificate: '', clusterName: '',
contextName: '', credentialsId: 'k8s', namespace: '',
restrictKubeConfigAccess: false, serverUrl: '') {

            sh 'kubectl apply -f deployment.yaml'

        }

    }

}
```

Before starting a new build remove Old containers.

kubectl get svc
#copy service port
<worker-ip:svc port>

< http://13.235.69.18:31283/>
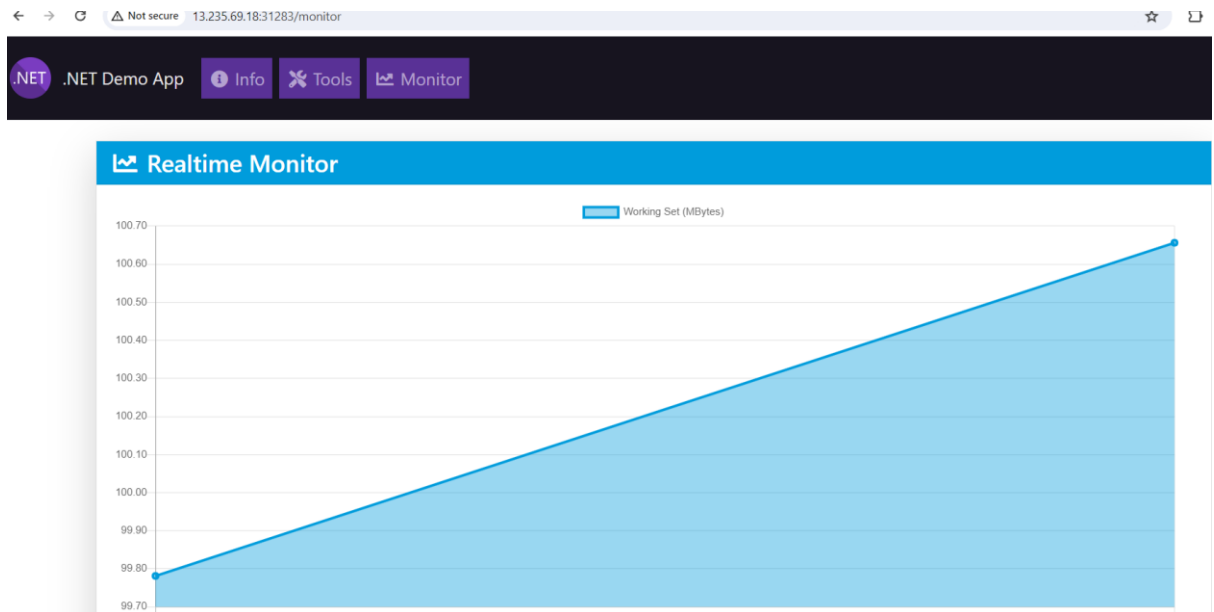
```
root@ip-172-31-39-3:~# kubectl get svc
NAME                TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
dotnet-app-service  NodePort    100.65.8.187   <none>        80:31283/TCP   18m
kubernetes          ClusterIP   100.64.0.1     <none>        443/TCP        159m
root@ip-172-31-39-3:~#
```
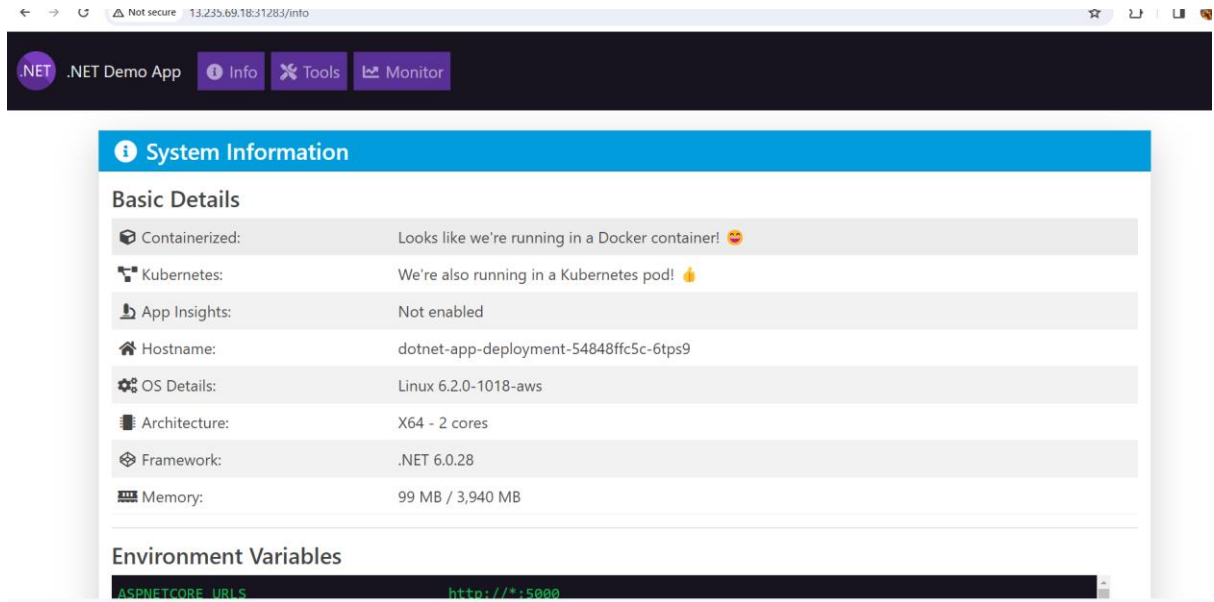
Step 12 — Terminate the AWS EC2 Instance

The complete pipeline script

Dotnet - Stage View

	Declarative: Tool Install	clean workspace	Checkout From Git	Sonarqube Analysis	quality gate	TRIVY File scan	OWASP Dependency Check	Docker Build & tag	TRIVY	Docker Push	Deploy to container	Deploy to k8s
Average stage times: (Average full run time: ~37s)	100ms	210ms	1s	10s	283ms	1s	6s	3s	2s	6s	668ms	1s
#29 Apr 07 15:17 No Changes	100ms	210ms	1s	10s	283ms (paused for 2s)	1s	6s	3s	2s	6s	668ms	1s

```
pipeline{

  agent any

  tools{

    jdk 'jdk17'

  }

  environment {

    SCANNER_HOME=tool 'sonar-scanner'

  }

  stages {

    stage('clean workspace'){

      steps{

        cleanWs()
```

```
        }

    }

    stage('Checkout From Git'){

        steps{

            git branch: 'main', url:
'https://github.com/sai241194/DotNet.git'

        }

    }

    stage("Sonarqube Analysis "){

        steps{

            withSonarQubeEnv('sonar-server') {

                sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=Dotnet-Webapp \

                    -Dsonar.projectKey=Dotnet-Webapp '''

            }

        }

    }

    stage("quality gate"){

        steps {
```

```
        script {

            waitForQualityGate abortPipeline: false,
credentialsId: 'Sonar-token'

        }

    }

}

stage("TRIVY File scan"){

    steps{

        sh "trivy fs . > trivy-fs_report.txt"

    }

}

stage("OWASP Dependency Check"){

    steps{

        dependencyCheck additionalArguments: '--scan ./ -
-format XML ', odcInstallation: 'DP-Check'

        dependencyCheckPublisher pattern:
'**/dependency-check-report.xml'

    }

}
```

```
stage("Docker Build & tag"){

    steps{

        script{

            withDockerRegistry(credentialsId: 'docker',
toolName: 'docker'){

                sh "make image"

            }

        }

    }

}

stage("TRIVY"){

    steps{

        sh "trivy image sai2411/dotnet-monitoring:latest >
trivy.txt"

    }

}

stage("Docker Push"){

    steps{

        script{
```

```groovy
                withDockerRegistry(credentialsId: 'docker',
toolName: 'docker'){

                    sh "make push"

                }

            }

        }

    }

    stage("Deploy to container"){

        steps{

            sh "docker run -d --name dotnet -p 5000:5000
sai2411/dotnet-monitoring:latest"

        }

    }

    stage('Deploy to k8s'){

        steps{

            dir('K8S') {

                withKubeConfig(caCertificate: '', clusterName: '',
contextName: '', credentialsId: 'k8s', namespace: '',
restrictKubeConfigAccess: false, serverUrl: '') {

                    sh 'kubectl apply -f deployment.yaml'
```

```
                }

            }

         }

      }

   }

}
```

Here is the GitHub for this project

https://github.com/sai241194/DotNet