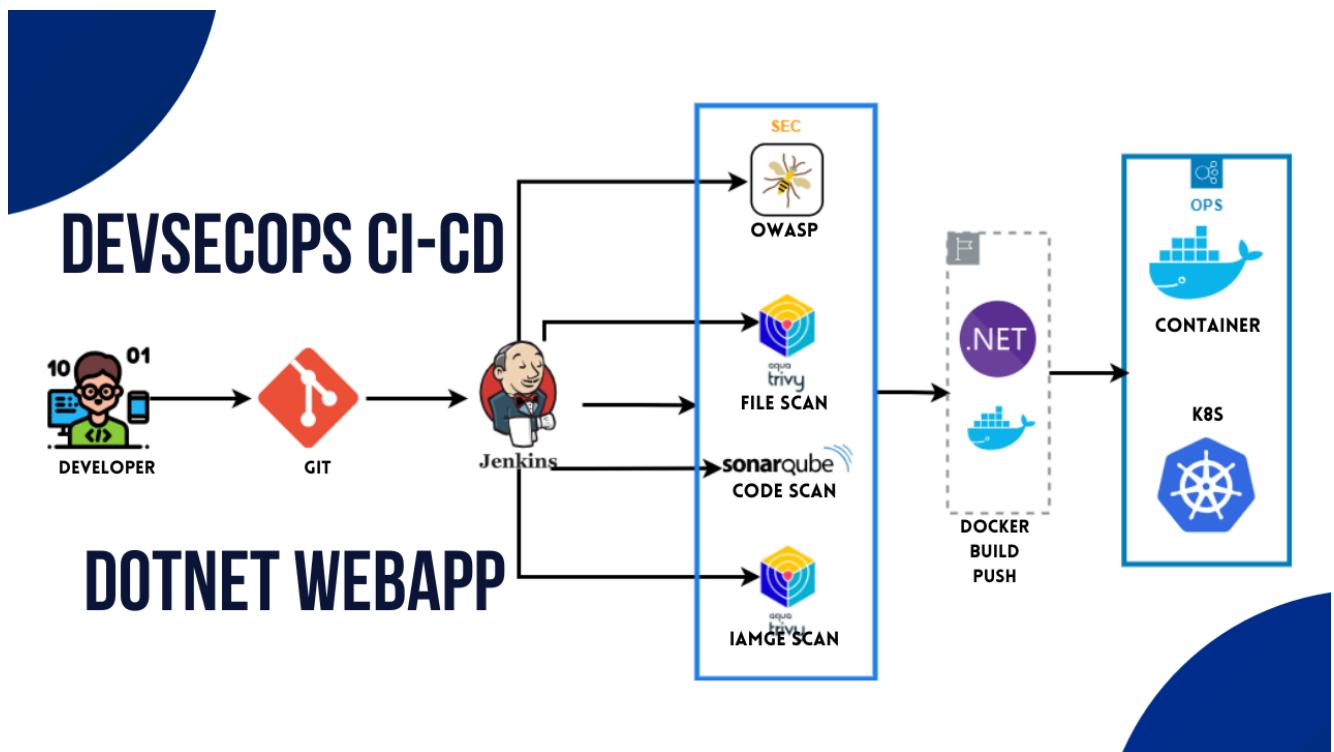


DevOps

Jenkins CI/CD | DOTNET webapp DevSecOps Project



mrcloudbook.com · 8 January 2024



Hello friends, we will be deploying a .Net-based application. This is an everyday use case scenario used by several organizations. We will be using Jenkins as a CICD tool and deploying our application on a Docker Container and Kubernetes cluster. Hope this detailed blog is useful.

Mr Cloud Book

[Step 2 – Install Jenkins, Docker and Trivy](#)

[2A – To Install Jenkins](#)

[2B – Install Docker](#)

[2C – Install Trivy](#)

[Step 3 – Install Plugins like JDK, Sonarqube Scanner, OWASP Dependency Check, Docker.](#)

[3A – Install Plugin](#)

[3B – Configure Java and Maven in Global Tool Configuration](#)

[3C – Create a Job](#)

[Step 4 – Install OWASP Dependency Check Plugins](#)

[Step 5 – Configure Sonar Server in Manage Jenkins](#)

[Step 6 – we have to install make package](#)

[Step 7 – Docker Image Build and Push](#)

[Step 8 – Deploy the image using Docker](#)

[Step 9 – Access the Real World Application](#)

[Step 10 – Kubernetes setup](#)

Steps:-

Step 1 – Create an Ubuntu T2 Large Instance

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

Step 3 – Install Plugins like JDK, Sonarqube Scanner, OWASP Dependency Check,

Step 4 – Create a Pipeline Project in Jenkins using a Declarative Pipeline

Step 5 – Configure Sonar Server in Manage Jenkins

Step 6 – we have to install and make the package

Step 7 – Docker Image Build and Push

Step 8 – Deploy the image using Docker

Step 9 – Access the Real World Application

Step 10 – Kubernetes setup

Step 11 – Terminate the AWS EC2 Instance

Now, let's get started and dig deeper into each of these steps:-

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.

Instances (1) Info												
<input type="checkbox"/>	Name	▲	Instance ID	Instance state	▼	Instance type	▼	Status check	Alarm status	Availability Zone	▼	Public IPv4 D
<input type="checkbox"/>	CI-CD		i-065c10200537a1eee	Running	🕒	t2.large		2/2 checks passed	No alarms	+	ap-south-1a	ec2-52-66-14-

Step 2 – Install Jenkins, Docker and Trivy

2A – To Install Jenkins

Connect to your console, and enter these commands to Install Jenkins



```
sudo vi jenkins.sh
#enter the below code
#!/bin/bash
sudo apt update -y
#sudo apt upgrade -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/api/debian-stable/jenkins.io-2023.key" | sudo tee /etc/apt/sources.list.d/jenkins.list
sudo apt update -y
sudo apt install temurin-17-jdk -y
/usr/bin/java --version
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
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 -y
sudo apt-get install jenkins -y
sudo systemctl start jenkins
sudo systemctl status jenkins
```



sudo chmod 777 jenkins.sh
./jenkins.sh



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 <small>Info</small>						
Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source info	Description - optional <small>Info</small>	
sgr-07f2d1fcab50e40ba	SSH	TCP	22	Custom	<input type="text"/> 0.0.0.0/0 X	<button>Delete</button>
-	Custom TCP	TCP	8080	Anywh...	<input type="text"/> 0.0.0.0/0 X	<button>Delete</button>
-	HTTPS	TCP	443	Anywh...	<input type="text"/> 0.0.0.0/0 X	<button>Delete</button>
-	HTTP	TCP	80	Anywh...	<input type="text"/> 0.0.0.0/0 X	<button>Delete</button>

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.

Getting Started

Unlock Jenkins

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

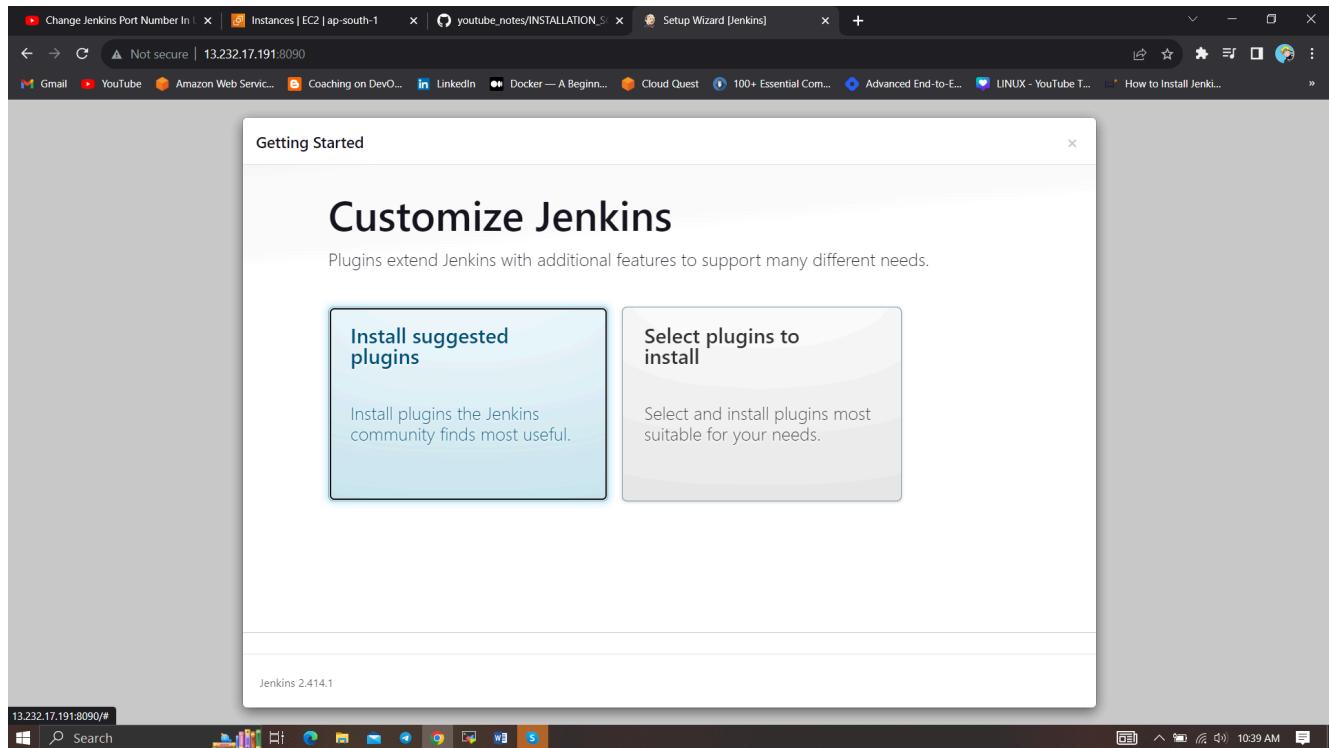
`/var/lib/jenkins/secrets/initialAdminPassword`

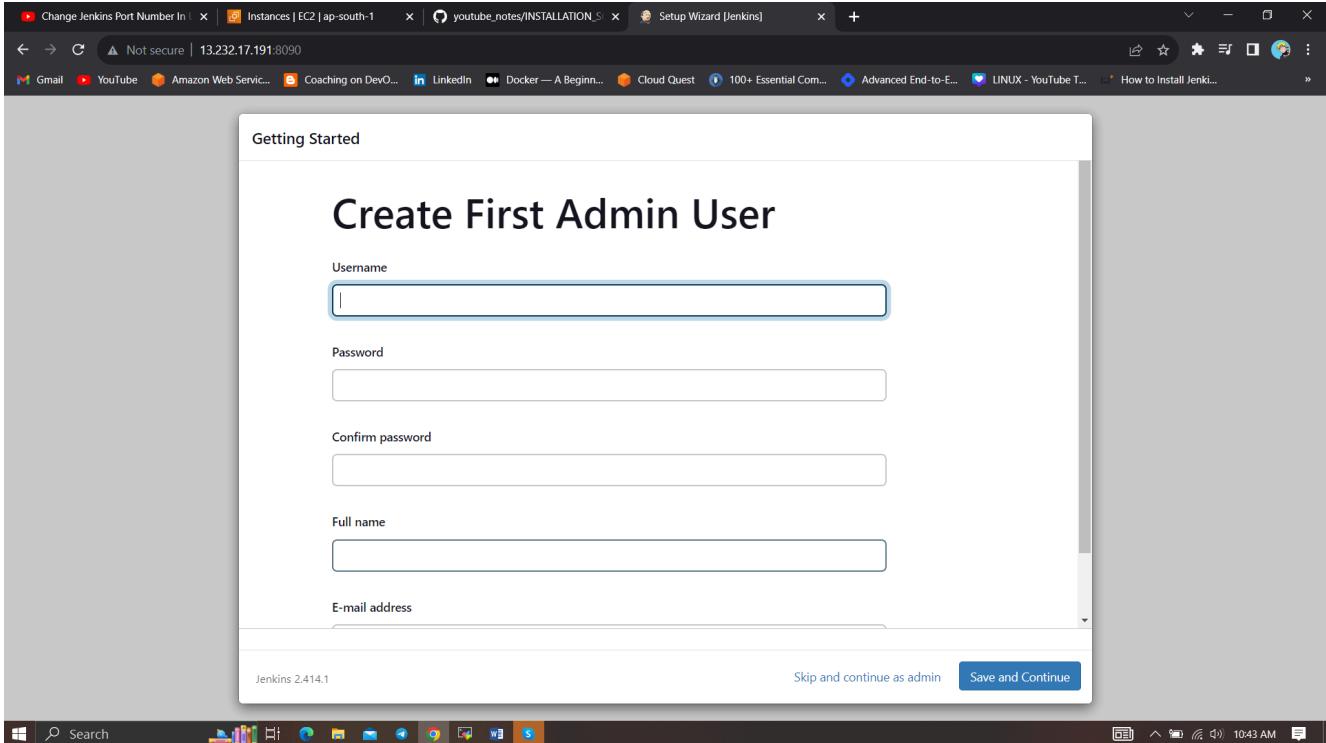
Please copy the password from either location and paste it below.

Administrator password

[Continue](#)

Jenkins will now get installed and install all the libraries.





Jenkins Getting Started Screen

The screenshot shows the Jenkins Dashboard. On the left, there is a sidebar with links: "New Item", "People", "Build History", "Manage Jenkins", and "My Views". The main content area features a "Welcome to Jenkins!" message: "This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project." Below this, there is a "Start building your software project" button and a "Create a job" button. Further down, there is a "Set up a distributed build" section with links to "Set up an agent" and "Configure a cloud", along with a "Learn more about distributed builds" link. The browser's address bar shows the URL: "Not secure | 13.232.17.191:8090". The taskbar at the bottom of the screen includes icons for Search, File Explorer, Task View, Start, Taskbar settings, and system status. The top navigation bar shows the Jenkins logo, search bar, user count (1), admin link, and log out link.

2B – Install Docker

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

```
sudo usermod -aG docker $USER
sudo chmod 777 /var/run/docker.sock
sudo docker ps
```

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

Inbound rules Info					
Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info
sgr-031f64b0253c2469c	Custom TCP	TCP	8080	Custom	<input type="text"/> 0.0.0.0/0
sgr-07f2d1fcab50e40ba	SSH	TCP	22	Custom	<input type="text"/> 0.0.0.0/0
sgr-077b394464a0d49de	HTTPS	TCP	445	Custom	<input type="text"/> 0.0.0.0/0
sgr-076ce119bc21d735d	Custom TCP	TCP	9000	Custom	<input type="text"/> 0.0.0.0/0
sgr-09fc7d687936c1e66	HTTP	TCP	80	Custom	<input type="text"/> 0.0.0.0/0

Add rule Cancel Preview changes Save rules

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

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ 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
44ba2882f0e8: Pull complete
2cabe57ff306: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f5875e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Download newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752fdb04993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4b60c96bf9ad sonarqube:lts-community "/opt/sonarqube/dock..." 9 seconds ago Up 5 seconds 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp sonar
ubuntu@ip-172-31-42-253:~$
```

Now our sonarqube is up and running

Enter username and password, click on login and change password

```
username admin  
password admin
```



Log in to SonarQube

sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration ? Search for projects... A

How do you want to create your project?

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

From Azure DevOps Set up global configuration	From Bitbucket Server Set up global configuration	From Bitbucket Cloud Set up global configuration	From GitHub Set up global configuration	From GitLab Set up global configuration
--	--	---	--	--

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

Manually

2C – Install Trivy

```
sudo apt-get install wget apt-transport-https gnupg lsb-release -y  
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor > /usr/share/keyrings/trivy.gpg  
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity
```

<https://mrcloudbook.com/jenkins-ci-cd-dotnet-webapp-devsecops-project/>

8/36

```
sudo apt-get update
sudo apt-get install trivy -y
```

Next, we will log in to Jenkins and start to configure our Pipeline in Jenkins

Step 3 – Install Plugins like JDK, Sonarqube Scanner, OWASP Dependency Check, Docker.

3A – Install Plugin

Goto Manage Jenkins → Plugins → Available Plugins →

Install below plugins

1 → Install OWASP ((Install without restart)

2 → SonarQube Scanner (Install without restart)

3 → 1 → Eclipse Temurin Installer (Install without restart)

Install	Name	Released
<input checked="" type="checkbox"/>	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net	11 mo ago
<input checked="" type="checkbox"/>	SonarQube Scanner 2.15 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	9 mo 19 days ago

3B – Configure Java and Maven in Global Tool Configuration

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

Add JDK

JDK

Name: jdk17

Install automatically ?

Install from adoptium.net ?

Version: ?
jdk-17.0.8.1+1

Add Installer ▼

3C – Create a Job

Label it as Dotnet CI-CD, click on Pipeline and OK.

Step 4 – Install OWASP Dependency Check Plugins

GotoDashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install it without restart.

Plugins

Install Name ↓ Released

OWASP Dependency-Check 5.4.0

Security DevOps Build Tools Build Reports

This plug-in can independently execute a [Dependency-Check](#) analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.

3 mo 20 days ago

First, we configured the Plugin and next, we had to configure the Tool

Goto Dashboard → Manage Jenkins → Tools →

Dependency-Check installations

[Add Dependency-Check](#)

Dependency-Check

Name

DP-Check

 Install automatically [?](#)

Install from github.com

Version

dependency-check 6.5.1

[Add Installer ▾](#)

Click on Apply and Save here.

Step 5 – Configure Sonar Server in Manage Jenkins

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <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

The screenshot shows the Sonarqube administration interface. At the top, there's a navigation bar with tabs: sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration tab is highlighted with a red box. Below the navigation bar, there's a secondary navigation bar with tabs: Configuration, Security, Projects, System, and Marketplace. Under the Configuration tab, there's a dropdown menu with options: General, Edit global, Find i, and a search bar. The 'Users' option in this dropdown menu is also highlighted with a red box.

Click on Update Token

Administrator admin

< 1 hour ago

sonar-administrators
sonar-users

Tokens

Update Tokens

Create a token with a name and generate

Tokens of Administrator

Generate Tokens

Name	Expires in
<input type="text" value="Enter Token Name"/>	30 days
<input type="button" value="Generate"/>	

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

Copy

Name	Type	Project	Last use	Created	Expiration	
Jenkins	User		Never	September 8, 2023	October 8, 2023	Revoke

Copy this Token

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

New credentials

Kind	<input type="text" value="Secret text"/>
Scope	<input type="text" value="Global (Jenkins, nodes, items, all child items, etc)"/>
Secret	<input type="text" value="POST THE TOKEN HERE"/>
ID	<input type="text" value="Sonar-token"/>
Description	<input type="text" value="Sonar-token"/>
<input type="button" value="Create"/>	

You will this page once you click on create

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

ID	Name	Kind	Description	
Copy Sonar-token	sonar	Secret text	sonar	Edit

Now, go to Dashboard → Manage Jenkins → Configure System

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name	<input type="text" value="sonar-server"/>	X
Server URL	Default is http://localhost:9000 <input type="text" value="http://13.232.17.191:9000"/>	
Server authentication token	SonarQube authentication token. Mandatory when anonymous access is disabled. <input type="text" value="Sonar-token"/>	
<input style="margin-right: 10px;" type="button" value="Add"/> <input style="background-color: #0072bc; color: white; border-radius: 5px; padding: 5px; margin-right: 10px;" type="button" value="Save"/> <input type="button" value="Apply"/>		

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.

SonarQube Scanner installations**SonarQube Scanner****Name**

Install automatically ?

Install from Maven Central**Version**

In the Sonarqube Dashboard add a quality gate also

Administration-> Configuration->Webhooks

The screenshot shows the SonarQube administration interface. The top navigation bar has tabs for Gmail, YouTube, Amazon Web Servic..., Coaching on DevO..., LinkedIn, Docker — A Beginn..., Cloud Quest, 100+ Essential Com..., Advanced End-to-E..., LINUX - YouTube T..., and How to Install Jenki... The main navigation bar includes sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration (which is highlighted with a red box), and a search bar for projects. Below the main navigation, there's a sub-navigation for Administration with tabs for General Settings, Encryption, and Webhooks (also highlighted with a red box). A sidebar on the left lists General Settings, Encryption, and Webhooks. The main content area shows a table of users with one entry: Administrator admin, last connected < 1 hour ago, with groups sonar-administrators and sonar-users, and 1 token. A 'Create User' button is located in the top right corner of the user table.

Click on Create

The screenshot shows the SonarQube configuration page for Webhooks. The top navigation bar is identical to the previous screenshot. The main content area is titled 'Webhooks' and contains a brief description: '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](#)'. Below this, a message says 'No webhook defined.' and features a prominent blue 'Create' button with a black border.

Add details

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

The screenshot shows the 'Create Webhook' dialog box in SonarQube. The 'Name' field contains 'jenkins', the 'URL' field contains 'http://43.204.36.242:8090/sonarqube-webhook/'. There is a note about using an embedded database. At the bottom are 'Create' and 'Cancel' buttons.

Let's go to our Pipeline and add the below code 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/Aj7Ay/DotNetWebapp'
            }
        }
        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, credential:
        }
    }
}

stage("TRIVY File scan"){
    steps{
        sh "trivy fs . > trivy-fs_report.txt"
    }
}

stage("OWASP Dependency Check"){
    steps{
        dependencyCheck additionalArguments: '--scan ./ --format'
        dependencyCheckPublisher pattern: '**/dependency-check-r
    }
}

```

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

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: ~2min 12s)	204ms	375ms	1s	19s	682ms	5s	48s
#1 Sep 14 16:04 No Changes	204ms	375ms	1s	19s	682ms (paused for 6s)	5s	48s

SonarQube Quality Gate

Python-Webapp Passed
 server-side processing: Success



To see the report, you can go to Sonarqube Server and go to Projects.

You can see the report has been generated and the status shows as passed. You can see that there are 522 lines. To see a detailed report, you can go to issues.

Step 6 – we have to install make package



```
sudo apt install make
# to check version install or not
make -v
```



```
ubuntu@ip-172-31-13-201:~$ sudo apt install make
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 125 not upgraded.
Need to get 180 kB of archives.
After this operation, 426 kB of additional disk space will be used.
Get:1 http://ap-northeast-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 make amd64 4.3-4.1build1 [180 kB]
Fetched 180 kB in 0s (8730 kB/s)
Selecting previously unselected package make.
(Reading database ... 66370 files and directories currently installed.)
Preparing to unpack .../make_4.3-4.1build1_amd64.deb ...
```

Step 7 – 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

The screenshot shows the Jenkins 'Manage Jenkins' section under 'Plugins'. A search bar at the top right contains the text 'docker'. Below it, a list of plugins is shown:

- Docker 1.5** (Released 3 days 15 hr ago) - This plugin integrates Jenkins with Docker. It is up for adoption.
- Docker Commons 439.va_3cb_0a_6a_fb_29** (Released 1 mo 29 days ago) - Provides the common shared functionality for various Docker-related plugins.
- Docker Pipeline 572.v950f58993843** (Released 27 days ago) - Build and use Docker containers from pipelines. It is up for adoption.
- Docker API 3.3.1-79.v20b_53427e041** (Released 3 mo 4 days ago) - This plugin provides docker-java API for other plugins.

Now, goto Dashboard → Manage Jenkins → Tools →

The screenshot shows the Jenkins 'Manage Jenkins' section under 'Tools'. Under 'Docker installations', there is a single entry:

- Docker** (Name: docker, Install automatically checked)

Below this, there is a detailed configuration panel for the Docker installation:

- Download from docker.com**
- Docker version** (latest)
- Add Installer**

Add DockerHub Username and Password under Global Credentials

Kind

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
sevenajay

Treat username as secret ?

Password ?
.....

ID ?
docker

Description ?
docker

Create

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

DotNet-monitoring makefile

Code Blame 70 lines (55 loc) · 2.63 KB Raw ⌂ ⌄

```

22     @grep -E '^[a-zA-Z_-]+.*?## .*$' ${MAKEFILE_LIST} | awk 'BEGIN {FS = ".*?## "}; {printf "\033[36m%-20s\033[0m %s\n", $$1, $$2}' |
23     lint: ## 🔍 Lint & format, will not fix but sets exit code on error
24     @dotnet format --help > /dev/null 2> /dev/null || dotnet tool install --global dotnet-format
25     dotnet format --verbosity diag ./src
26
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 to Pipeline Script



```

stage("Docker Build & tag"){
    steps{
        script{
            withDockerRegistry(credentialsId: 'docker', toolName
                sh "make image"
            }
        }
    }
}

```

```

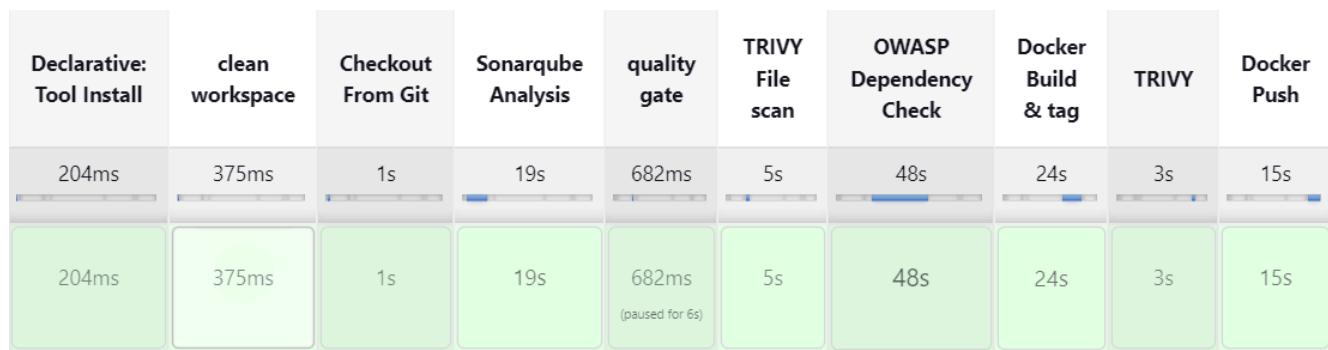
        }
    }
    stage("TRIVY"){
        steps{
            sh "trivy image sevenajay/dotnet-monitoring:latest > trivy.log"
        }
    }
    stage("Docker Push"){
        steps{
            script{
                withDockerRegistry(credentialsId: 'docker', toolName: 'Docker')
                sh "make 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

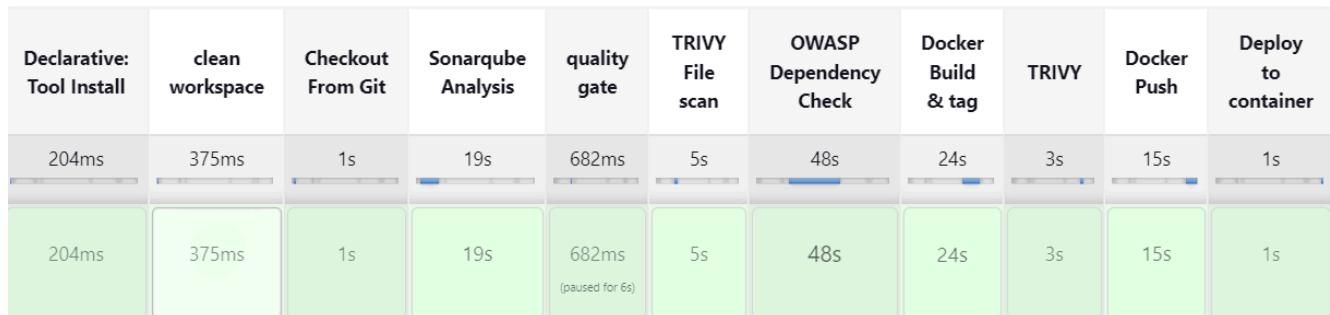


Step 8 – Deploy the image using Docker

Add this stage to your pipeline syntax

```
stage("Deploy to container"){
    steps{
        sh "docker run -d --name dotnet -p 5000:5000 sevenajay/ci-dotnet"
    }
}
```

You will see the Stage View like this,



(Add port 5000 to Security Group)

The screenshot shows the AWS CloudFormation console displaying the configuration for a security group named 'sgr-076ce119bc21d735d'. It lists three rules:

- sgr-076ce119bc21d735d**: A rule allowing Custom TCP traffic from 0.0.0.0/0 to port 9000.
- sgr-09fc7d687936c1e66**: A rule allowing HTTP traffic (port 80) from 0.0.0.0/0.
- : A rule allowing Custom TCP traffic from 0.0.0.0/0 to port 5000.

An 'Add rule' button is visible at the bottom left.

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

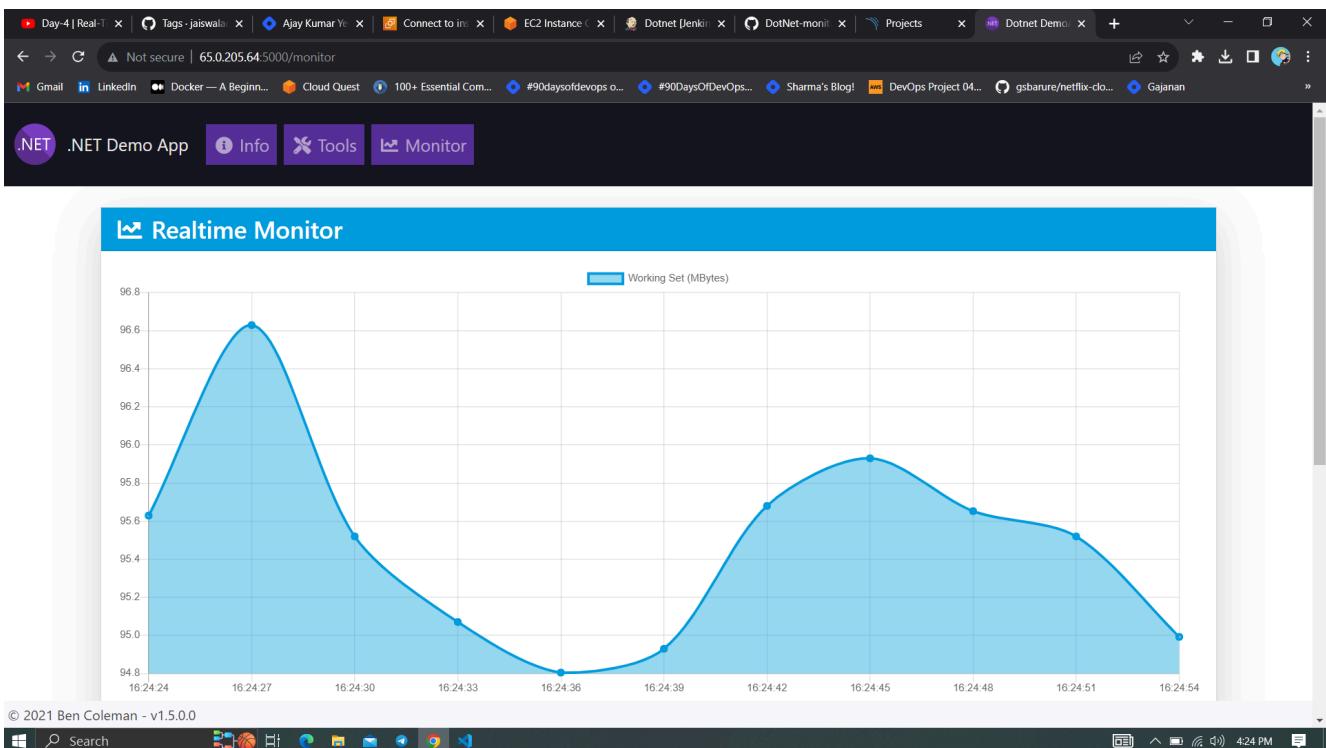
```
<public-ip of jenkins:5000>
```

Step 9 – Access the Real World Application

The screenshot shows a .NET web application interface. At the top, there are tabs for '.NET Demo App', 'Info', 'Tools', and 'Monitor'. Below this is a 'System Information' section with the following details:

Category	Value
Containerized:	Looks like we're running in a Docker container! 😊
Kubernetes:	Not running in Kubernetes 😞
App Insights:	Not enabled
Hostname:	853ba5bdf532
OS Details:	Linux 6.2.0-1011-aws
Architecture:	X64 - 2 cores
Framework:	.NET 6.0.22
Memory:	99 MB / 8,120 MB

Below this is an 'Environment Variables' section, which is currently empty.



Step 10 –Kubernetes setup

Take-Two Ubuntu 20.04 instances one for k8s master and the other one for worker.

Install Kubectl on Jenkins machine also.

Kubectl on Jenkins to be installed



```
sudo apt update
sudo apt install curl
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
kubectl version --client
```



Part 1 ----Master Node-----



```
sudo su
hostname master
bash
clear
```



-----Worker Node-----



```
sudo su
hostname worker
bash
clear
```



Part 2 -----Both Master & Node -----



```
sudo apt-get update
sudo apt-get install -y docker.io
sudo usermod -aG docker Ubuntu
```

```
newgrp docker
sudo chmod 777 /var/run/docker.sock
sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo snap install kube-apiserver
```

Part 3 Master



```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
# in case you're in root exit from it and run below commands
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/
```

◀ ▶

-----Worker Node-----



```
sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --
```

◀ ▶

Copy the config file to Jenkins master or the local file manager and save it

copy it and save it in documents or another folder save it as secret-file.txt

Install Kubernetes Plugin, Once it's installed successfully

Dashboard > Manage Jenkins > Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Plugins

Search: Kuber

Install

Install	Name	Released
<input checked="" type="checkbox"/>	Kubernetes Credentials 0.11 kubernetes credentials Common classes for Kubernetes credentials	9 days 16 hr ago
<input checked="" type="checkbox"/>	Kubernetes Client API 6.8.1-224.vd388fca_4db_3p... kubernetes Library plugins (for use by other plugins) Kubernetes Client API plugin for use by other Jenkins plugins.	9 days 17 hr ago
<input checked="" type="checkbox"/>	Kubernetes 4029.v5712230ccb_f8 Cloud Providers Cluster Management kubernetes Agent Management This plugin integrates Jenkins with Kubernetes	9 days 15 hr ago
<input checked="" type="checkbox"/>	Kubernetes CLI 1.12.1 kubernetes Configure kubectl for Kubernetes	8 days 22 hr ago

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

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: '', cor
                sh 'kubectl apply -f deployment.yaml'
            }
        }
    }
}
```



Before starting a new build remove Old containers.

The screenshot shows the Jenkins Pipeline Dotnet stage view. At the top, there are buttons for 'Add description' and 'Disable Project'. Below this is a 'Stage View' section with a table showing the duration of various stages: Declarative: Tool Install (14ms), clean workspace (45ms), Checkout From Git (974ms), Sonarqube Analysis (17s), quality gate (339ms), TRIVY File scan (2s), OWASP Dependency Check (3min 17s), Docker Build & tag (2s), TRIVY (2s), Docker Push (6s), Deploy to container (547ms), and Deploy to k8s (929ms). A tooltip indicates an average stage time of 14ms and an average full run time of ~3min 59s. A timeline bar shows the progress of the pipeline. Below the table is a 'Permalinks' section with links to the last build, stable build, and successful build.

Output

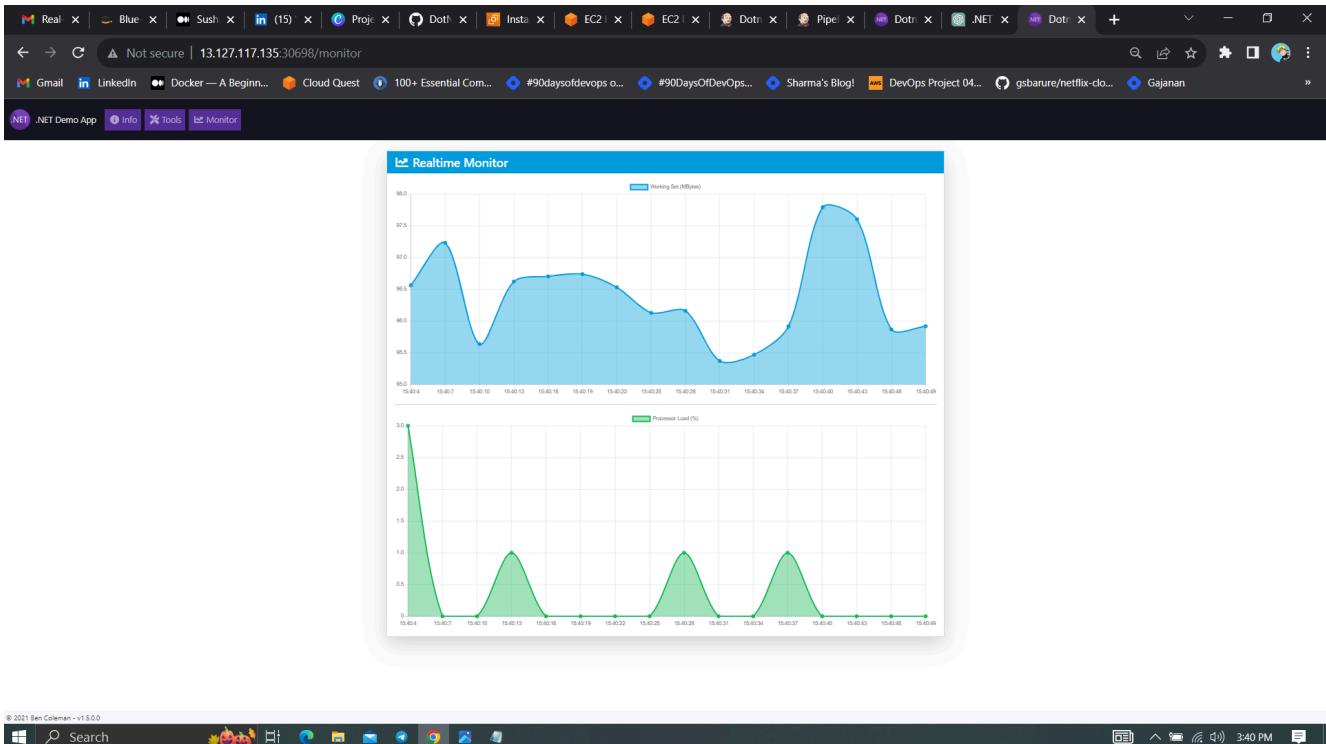
```
kubectl get svc
#copy service port
<worker-ip:svc port>
```

The screenshot shows the .NET Demo App interface. At the top, there are tabs for '.NET', '.NET Demo App', 'Info', 'Tools', and 'Monitor'. Below this is a 'System Information' section with a 'Basic Details' table:

Containerized:	Looks like we're running in a Docker container! 😊
Kubernetes:	We're also running in a Kubernetes pod! 👍
App Insights:	Not enabled
Hostname:	dotnet-app-deployment-5f9bbfb7f4-kfqzk
OS Details:	Linux 5.15.0-1036-aws
Architecture:	X64 - 2 cores
Framework:	.NET 6.0.22
Memory:	97 MB / 4,015 MB

Below this is an 'Environment Variables' section with a table:

Variable	Value
KUBERNETES_PORT	tcp://10.96.0.1:443



Step 11 – Terminate the AWS EC2 Instance

Lastly, do not forget to terminate the AWS EC2 Instance.

complete pipeline



```
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/Aj7Ay/DotNet'
            }
        }
    }
}
```

```
        }
    }
    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: ''
            }
        }
    }
    stage("TRIVY File scan"){
        steps{
            sh "trivy fs . > trivy-fs_report.txt"
        }
    }
    stage("OWASP Dependency Check"){
        steps{
            dependencyCheck additionalArguments: '--scan ./ --format json'
            dependencyCheckPublisher pattern: '**/dependency-check-report.json'
        }
    }
    stage("Docker Build & tag"){
        steps{
            script{
                withDockerRegistry(credentialsId: 'docker', toolName: 'Docker')
                sh "make image"
            }
        }
    }
}
stage("TRIVY"){
    steps{
        sh "trivy image sevenajay/dotnet-monitoring:latest > trivy-report.txt"
    }
}
stage("Docker Push"){
    steps{
        script{
```

```
withDockerRegistry(credentialsId: 'docker', toolName
    sh "make push"
)
}
}
}
stage("Deploy to container"){
    steps{
        sh "docker run -d --name dotnet -p 5000:5000 sevenajay/ci-dotnet:latest"
    }
}
stage('Deploy to k8s'){
    steps{
        dir('K8S') {
            withKubeConfig(caCertificate: '', clusterName: '', contextName: '')
            sh 'kubectl apply -f deployment.yaml'
        }
    }
}
}
}
```



Ajay Kumar Yegireddi is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

Comments

2 responses to “Jenkins CI/CD | DOTNET webapp DevSecOps Project”

**Sudhir Dilip Jadhav**

2 February 2024

Getting the below error when I build the pipeline :

```
*****
Started by user Sudhir Dilip Jadhav
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Pipeline-2
[Pipeline] {
[Pipeline] tool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Tool Install)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (clean workspace)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Checkout From Git)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] git
The recommended git tool is: NONE
```

No credentials specified

Cloning the remote Git repository

Cloning repository <https://github.com/Aj7Ay/DotNet-monitoring.git>

```
> git init /var/lib/jenkins/workspace/Pipeline-2 # timeout=10
```

Fetching upstream changes from <https://github.com/Aj7Ay/DotNet-monitoring.git>

```
> git -version # timeout=10
```

```
> git -version # 'git version 2.34.1'
```

```
> git fetch -tags -force -progress - https://github.com/Aj7Ay/DotNet-monitoring.git
```

```
+refs/heads/*:refs/remotes/origin/* # timeout=10
```

```
> git config remote.origin.url https://github.com/Aj7Ay/DotNet-monitoring.git # timeout=10
```

```
> git config -add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
```

Avoid second fetch

```
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
```

Checking out Revision e2456b125b06eced47096fbf219c1d50b75c153f

(refs/remotes/origin/main)

```
> git config core.sparsecheckout # timeout=10
```

```
> git checkout -f e2456b125b06eced47096fbf219c1d50b75c153f # timeout=10
```

```
> git branch -a -v -no-abbrev # timeout=10
```

```
> git checkout -b main e2456b125b06eced47096fbf219c1d50b75c153f # timeout=10
```

Commit message: "Create deployment.yaml"

First time build. Skipping changelog.

```
[Pipeline] }
```

```
[Pipeline] // withEnv
```

```
[Pipeline] }
```

```
[Pipeline] // stage
```

```
[Pipeline] stage
```

```
[Pipeline] { (Sonarqube Analysis )
```

```
[Pipeline] tool
```

```
[Pipeline] envVarsForTool
```

```
[Pipeline] withEnv
```

```
[Pipeline] {
```

```
[Pipeline] withSonarQubeEnv
```

Injecting SonarQube environment variables using the configuration: sonar-server

```
[Pipeline] {
```

```
[Pipeline] sh
```

```
+ /var/lib/jenkins/tools/hudson.plugins.sonar.SonarRunnerInstallation/sonar-
```

```
scanner/bin/sonar-scanner -Dsonar.projectName=Dotnet-Webapp -
```

```
Dsonar.projectKey=Dotnet-Webapp
```

Error: A JNI error has occurred, please check your installation and try again

Exception in thread "main" java.lang.UnsupportedClassVersionError:

org/sonarsource/scanner/cli/Main has been compiled by a more recent version of
the Java Runtime (class file version 61.0), this version of the Java Runtime only
recognizes class file versions up to 52.0

```
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:756)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:142)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:473)
at java.net.URLClassLoader.access$100(URLClassLoader.java:74)
at java.net.URLClassLoader$1.run(URLClassLoader.java:369)
at java.net.URLClassLoader$1.run(URLClassLoader.java:363)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:362)
at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.java:621)
[Pipeline] }
WARN: Unable to locate 'report-task.txt' in the workspace. Did the SonarScanner succeed?
[Pipeline] // withSonarQubeEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (quality gate)
Stage "quality gate" skipped due to earlier failure(s)
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (TRIVY File scan)
Stage "TRIVY File scan" skipped due to earlier failure(s)
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (OWASP Dependency Check)
Stage "OWASP Dependency Check" skipped due to earlier failure(s)
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

[Reply](#)**test**

28 March 2024

test

[Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot

reCAPTCHA
[Privacy](#) - [Terms](#)

[Post Comment](#)[Uncategorized](#)

How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point

22 July 2025

[AI](#)

How to Run Docker Model Runner on Ubuntu 24.04

11 July 2025

[AI, DevOps](#)

How to Install docker-ai on Ubuntu 24.04

15 June 2025

Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.



Important Links

[Privacy Policy](#)[Terms & Conditions](#)[Contact](#)

Resources

[Blog](#)[YouTube Channel](#)

