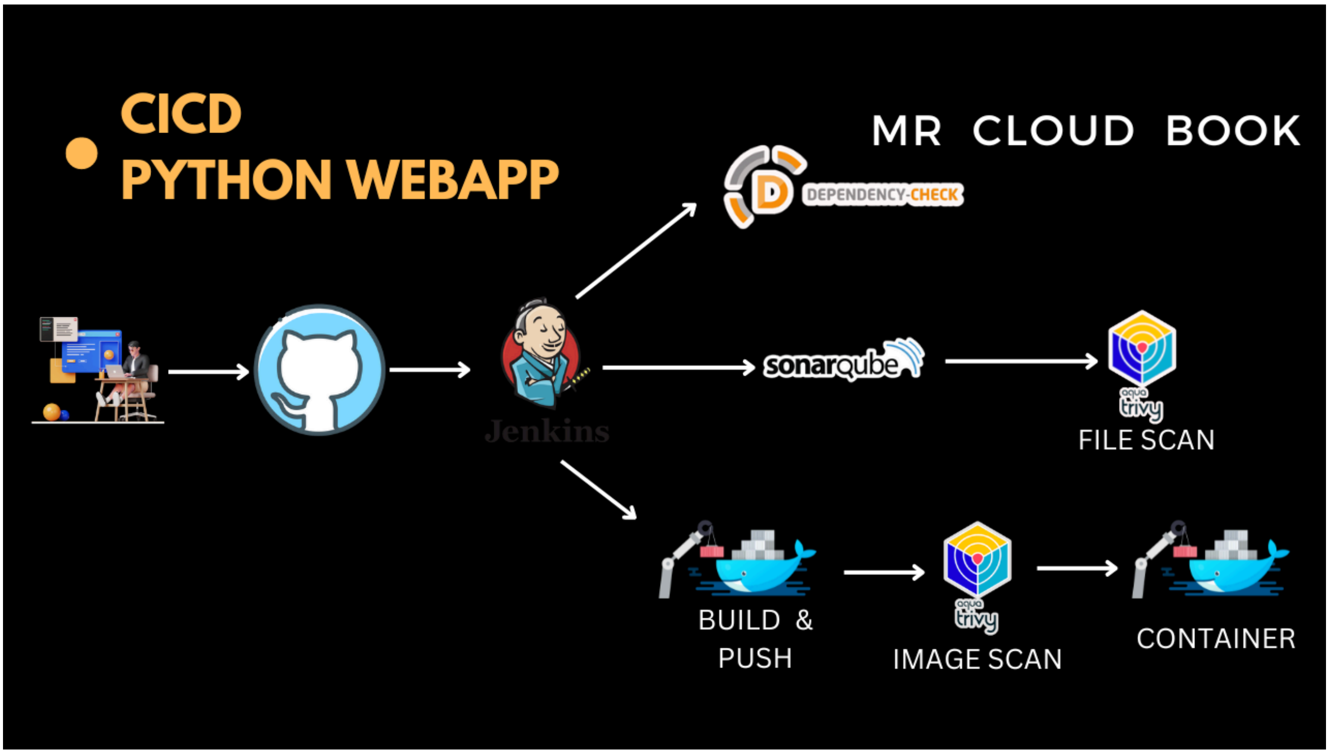
DevOps

CI-CD DevSecOps project with Jenkins | Python webapp

mrcloudbook.com · 8 January 2024



Hello friends, we will be deploying a python 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. Hope this detailed blog is useful.

```
Github: https://github.com/Aj7Ay/Python-System-Monitoring.git
```

Contents [hide]

Steps:-

Step 1 – Launch an AWS T2 Large Instance.

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 – Terminate the AWS EC2 Instance

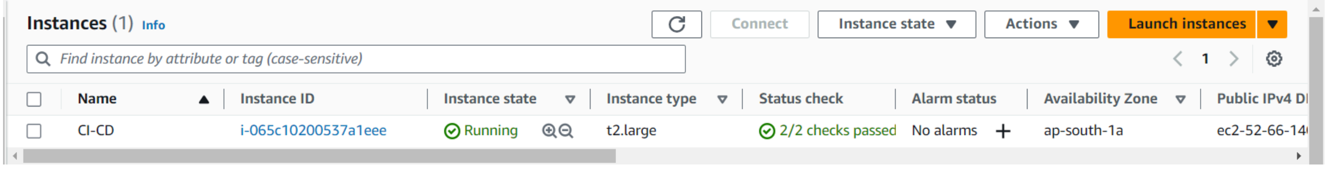
Steps:-

```
 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 — 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.



□	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DI
□	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

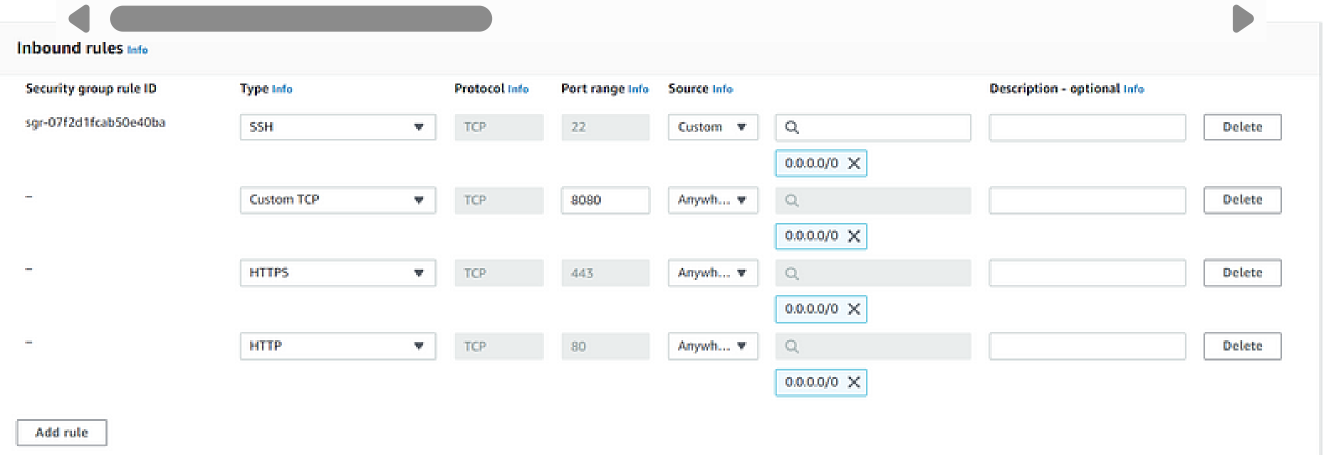
Connect to your console, and enter these commands to Install Jenkins

```
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.a
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 | su
                  /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 te
```

```
                              /etc/apt/sources.list.d/jenkins.list > /de
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.



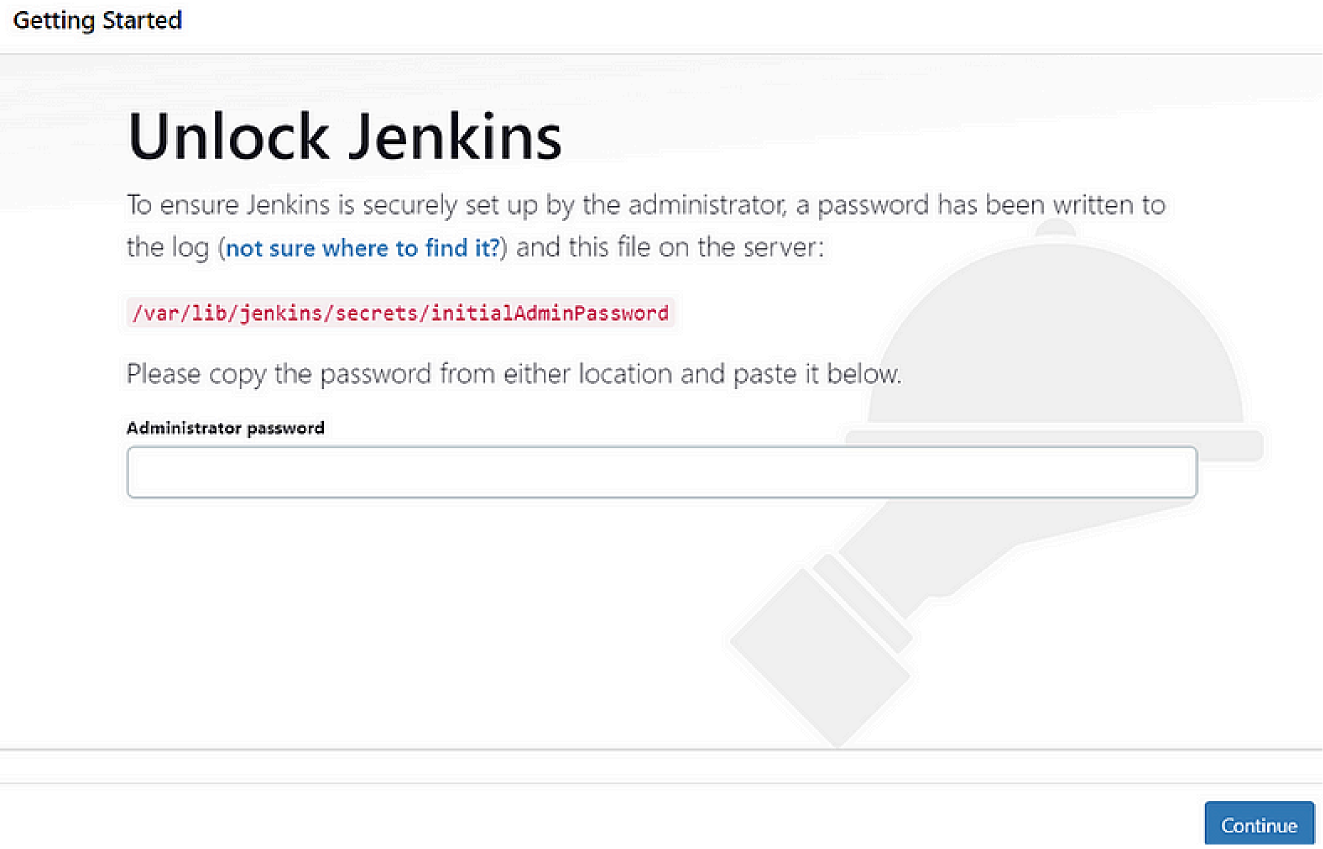
Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-07f2d1fcab50e40ba	SSH	TCP	22	Custom	0.0.0.0/0		Delete
-	Custom TCP	TCP	8080	Anywh...	0.0.0.0/0		Delete
-	HTTPS	TCP	443	Anywh...	0.0.0.0/0		Delete
-	HTTP	TCP	80	Anywh...	0.0.0.0/0		Delete

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

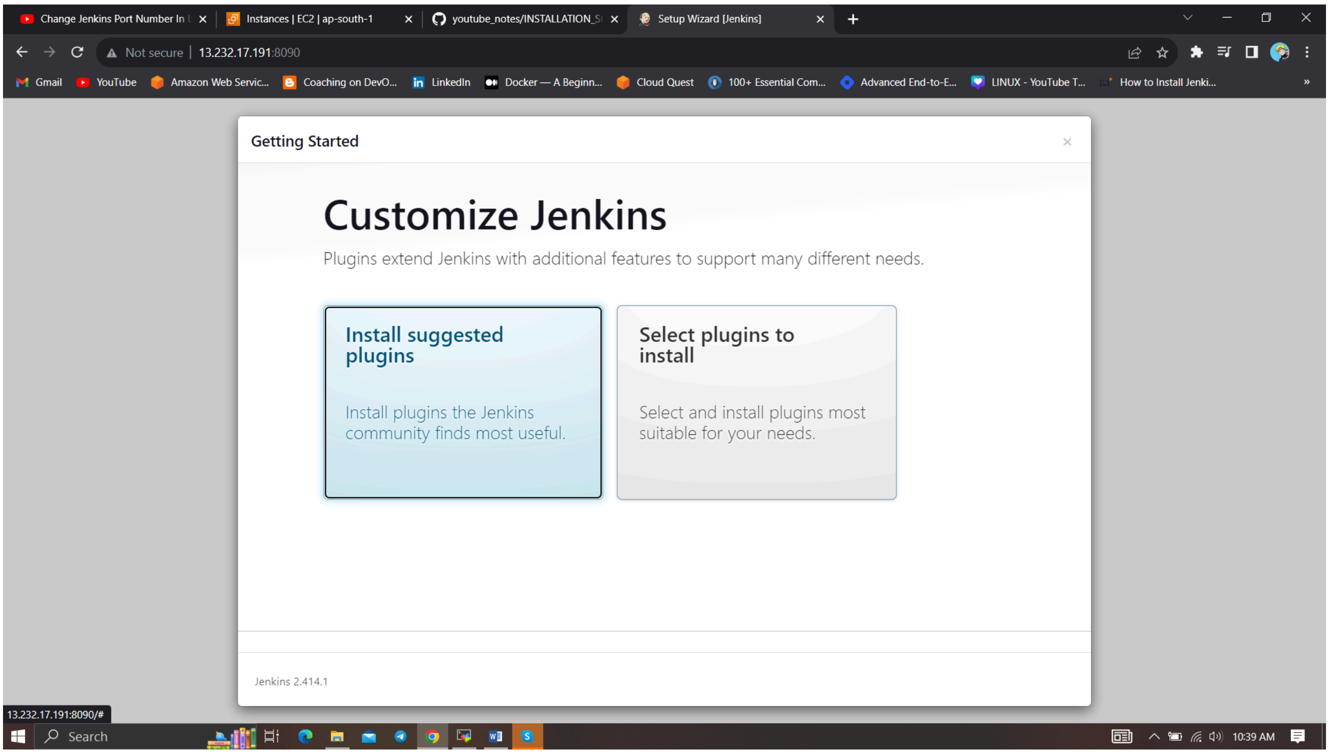
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

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

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

Administrator password

Jenkins will now get installed and install all the libraries.



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

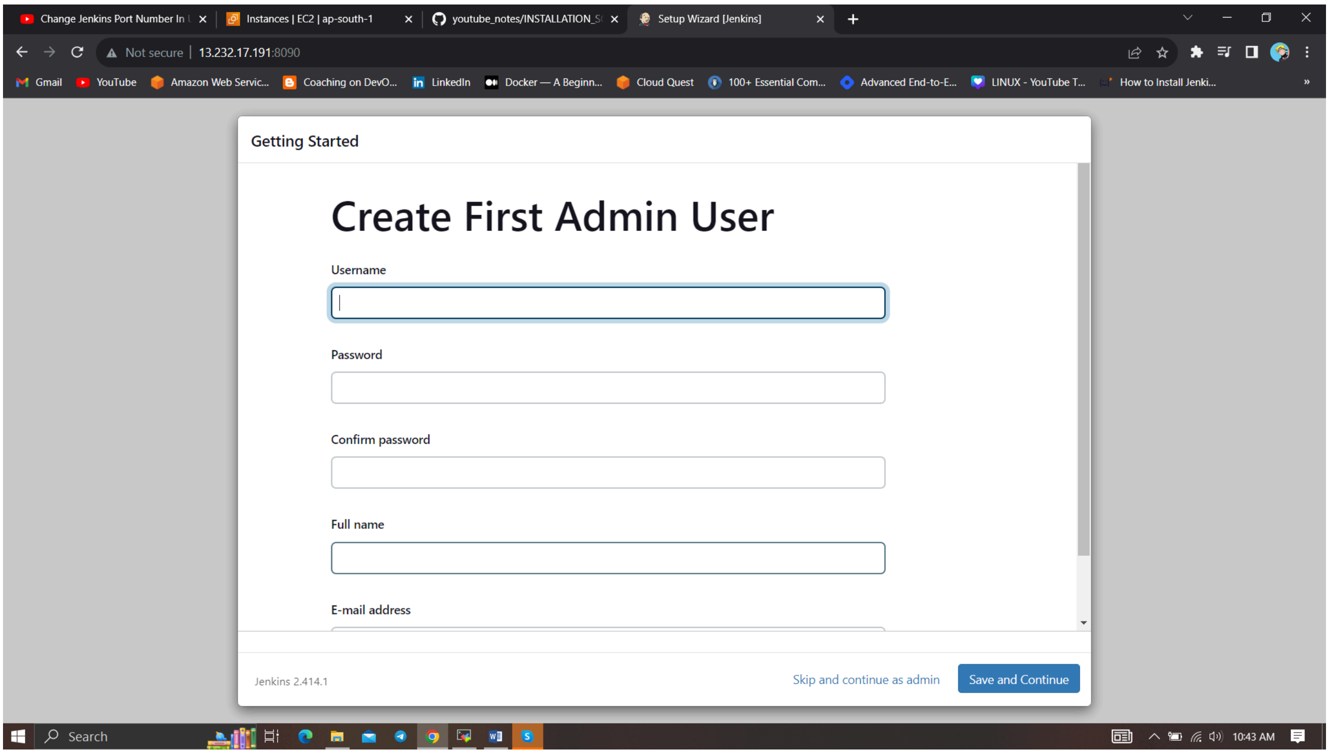
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.414.1



Getting Started

Create First Admin User

Username

Password

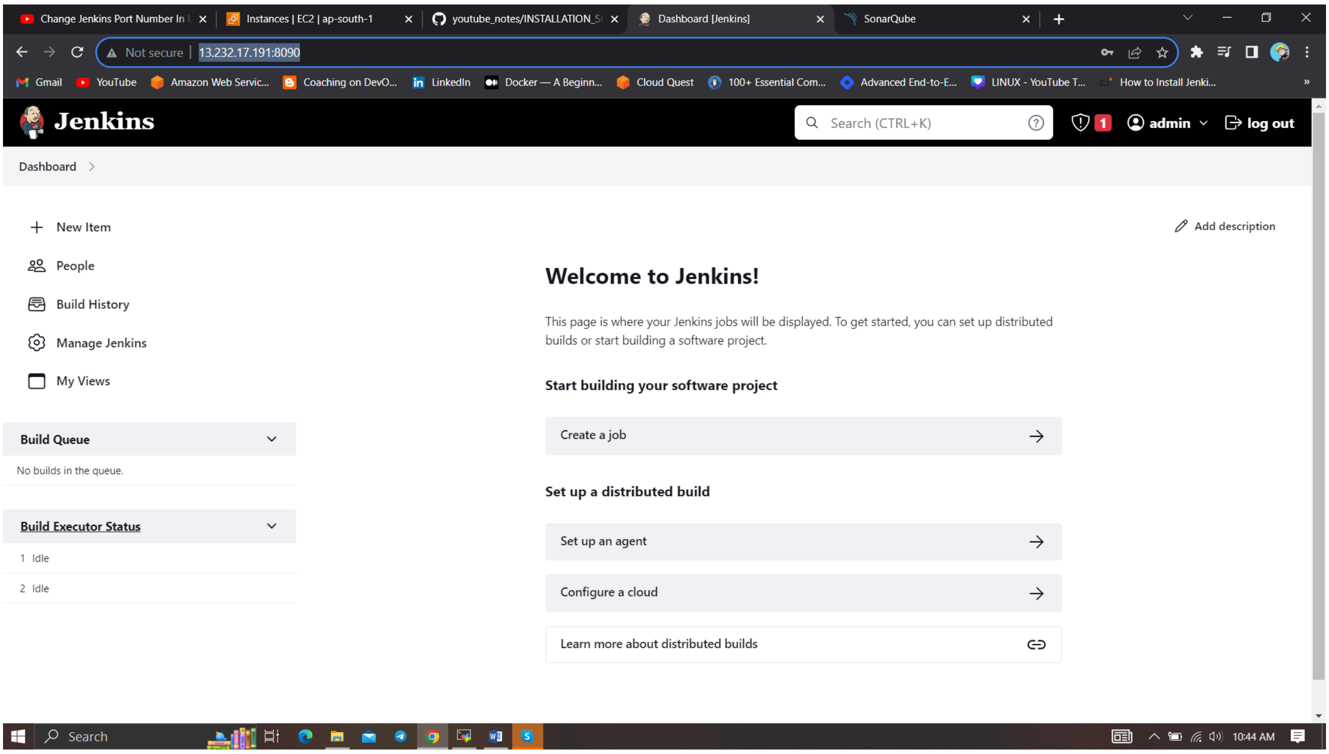
Confirm password

Full name

E-mail address

Jenkins 2.414.1

Jenkins Getting Started Screen



Welcome to Jenkins!

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

Start building your software project

Create a job

Set up a distributed build

Set up an agent

Configure a cloud

Learn more about distributed builds

Build Queue

No builds in the queue.

Build Executor Status

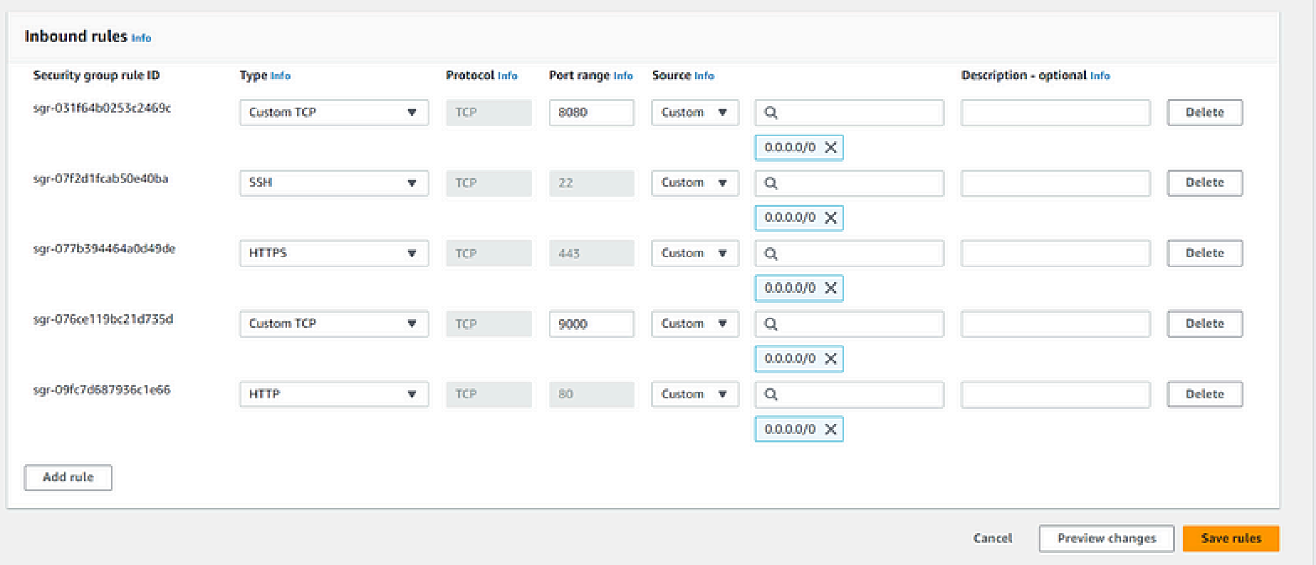
1 Idle

2 Idle

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
```

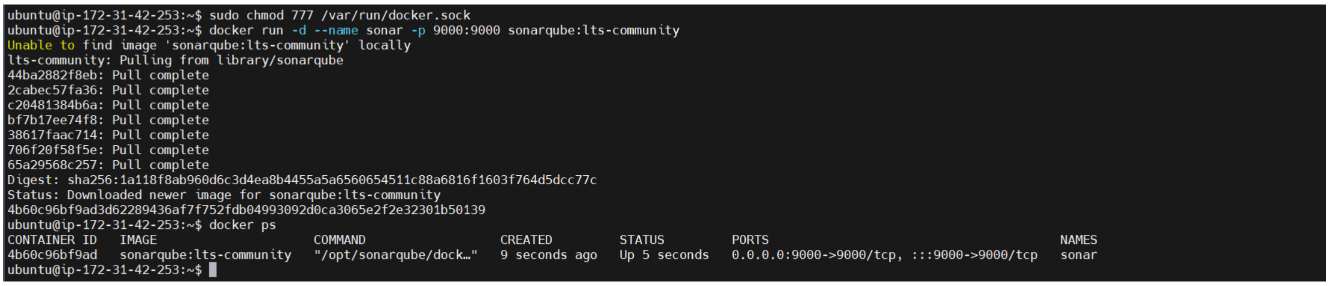
After the docker installation, we create a sonarqube container (Remember added 9000 port in the security group)



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	0.0.0.0/0		Delete
sgr-07f2d1fcab50e40ba	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-077b394464a0d49de	HTTPS	TCP	443	Custom	0.0.0.0/0		Delete
sgr-076ce119bc21d735d	Custom TCP	TCP	9000	Custom	0.0.0.0/0		Delete
sgr-09fc7d687936c1e66	HTTP	TCP	80	Custom	0.0.0.0/0		Delete

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

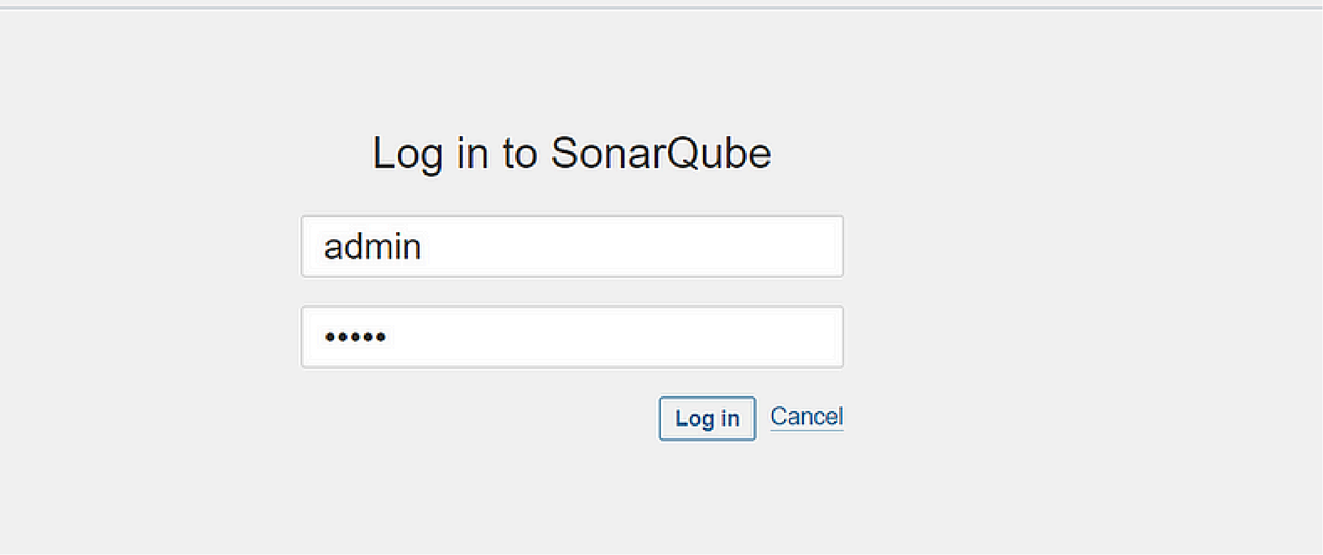


```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded 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

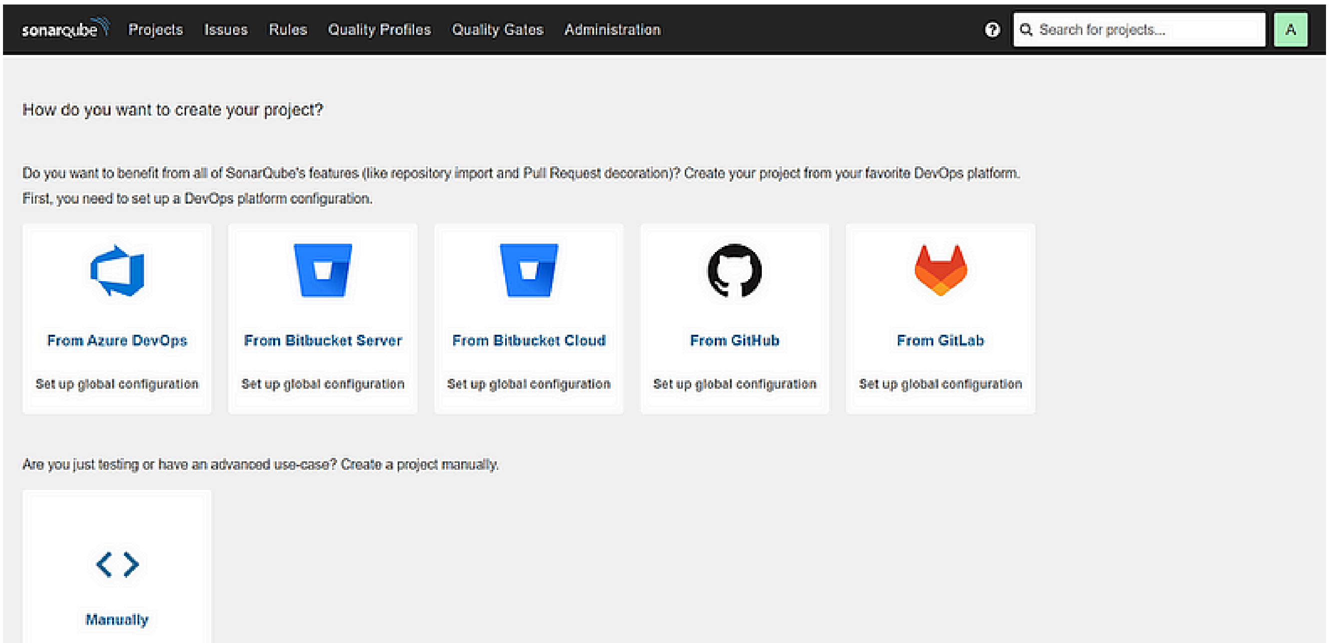
```
username admin
password admin
```



Log in to SonarQube

admin

•••••



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.

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 | gp
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity
```

```
    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

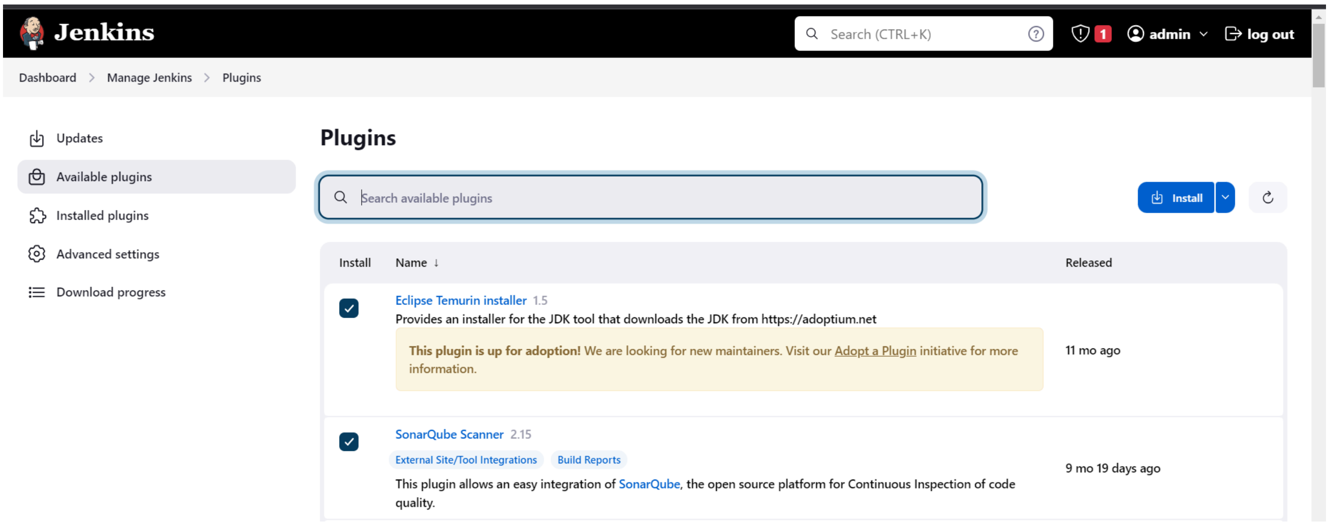
Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Install OWASP ((Install without restart)

2 → SonarQube Scanner (Install without restart)

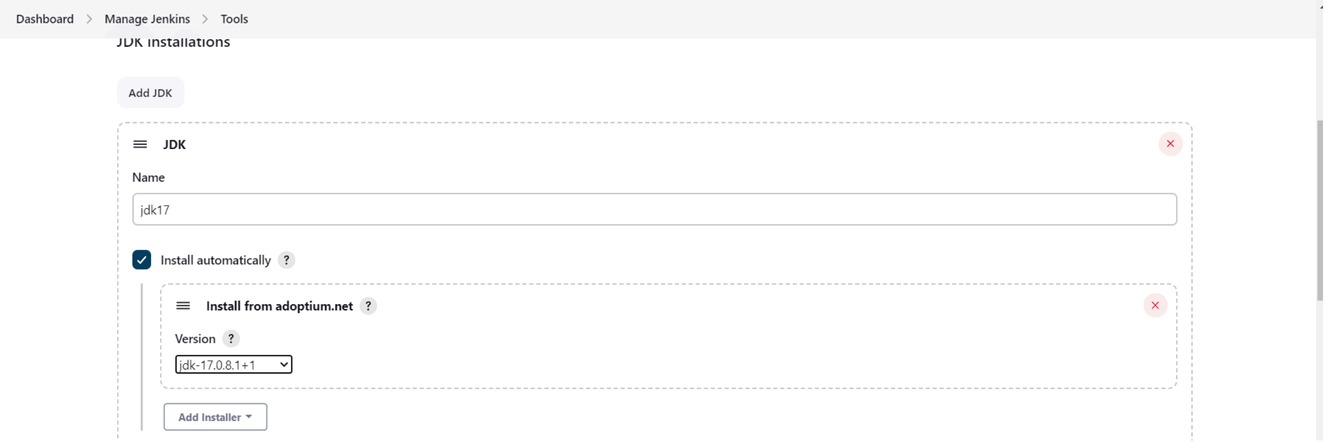
3 → 1 → Eclipse Temurin Installer (Install without restart)



Install	Name ↓	Released
☑	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	11 mo ago
☑	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



Dashboard > Manage Jenkins > Tools

JDK installations

Add JDK

JDK

Name: jdk17

☑ Install automatically

Install from adoptium.net

Version: jdk-17.0.8.1+1

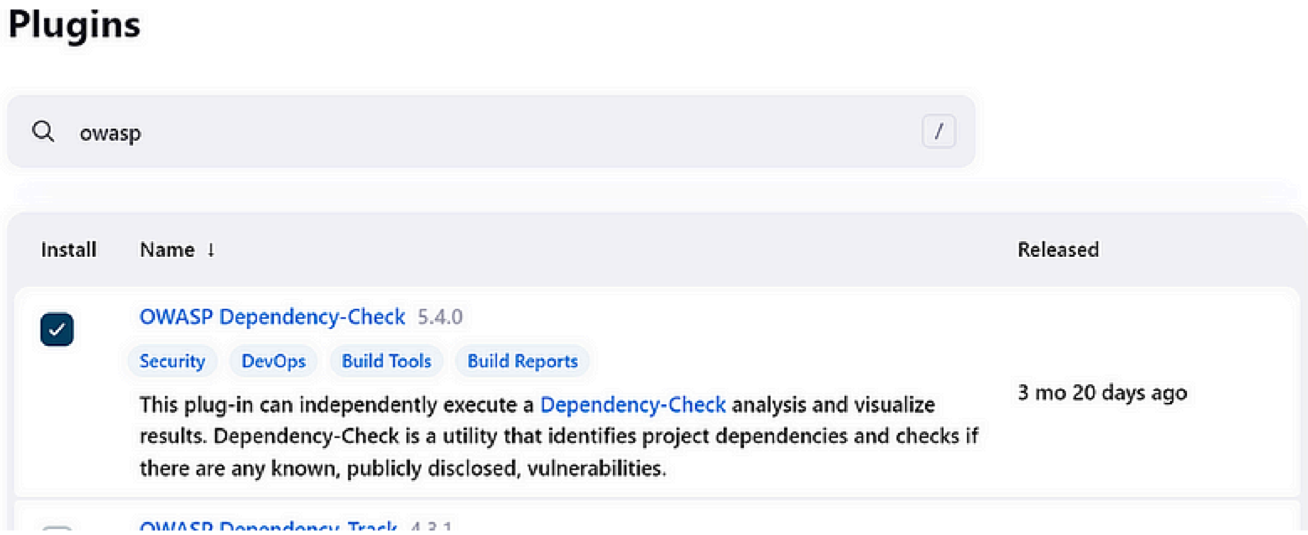
Add Installer

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.



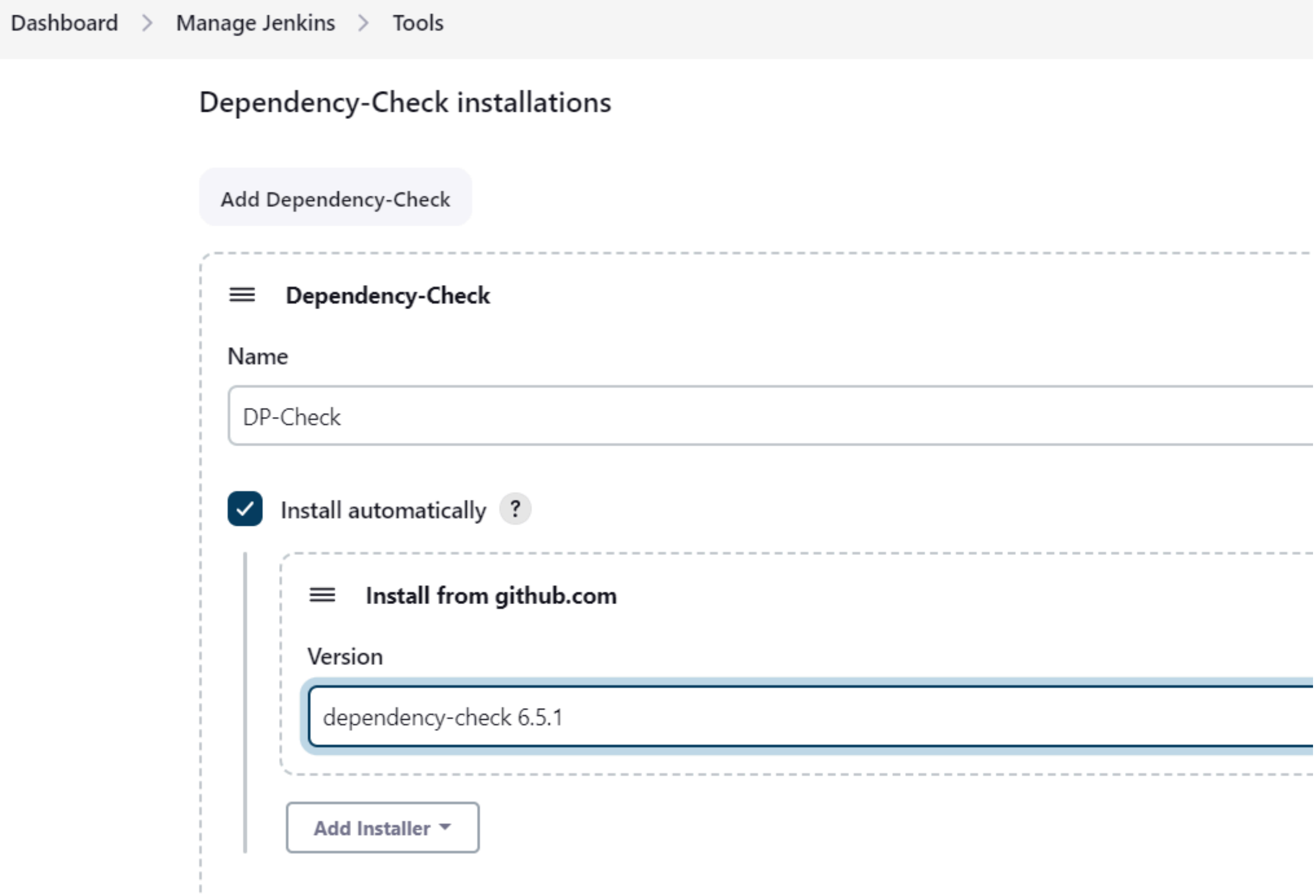
Plugins

owasp

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
□	OWASP Dependency-Track 4.3.1	

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

Goto Dashboard → Manage Jenkins → Tools →



Dependency-Check installations

Dependency-Check

Name: DP-Check

☑ Install automatically

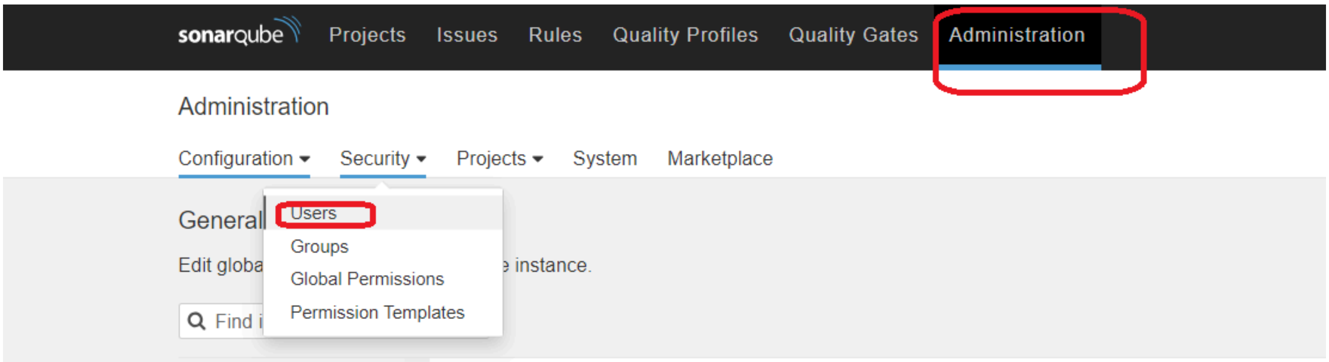
Install from github.com

Version: dependency-check 6.5.1

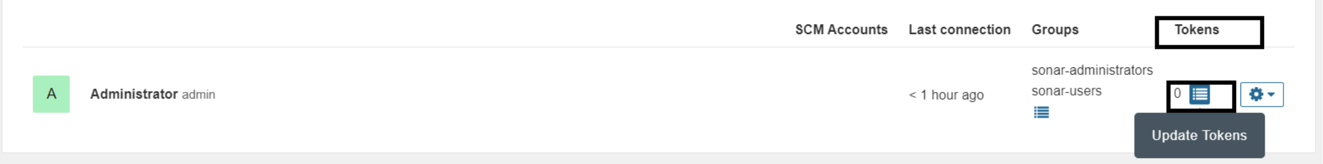
Click on Apply and Save here.

Step 5 – Configure Sonar Server in Manage Jenkins

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

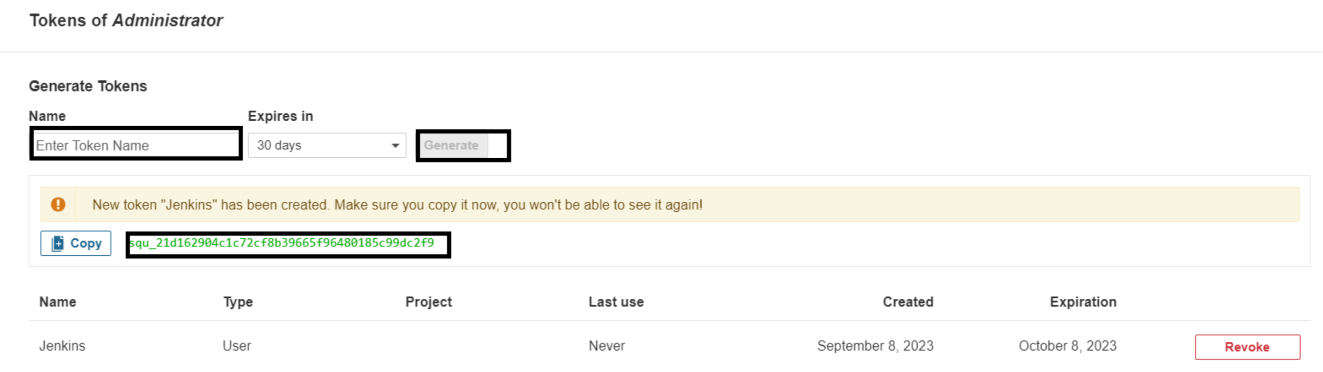


Click on Update Token



	SCM Accounts	Last connection	Groups	Tokens
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	0

Create a token with a name and generate



Tokens of *Administrator*

Generate Tokens

Name: Enter Token Name

Expires in: 30 days

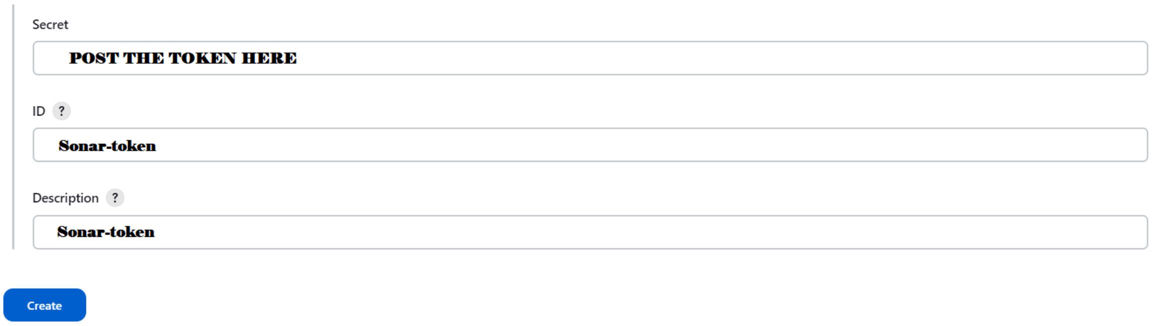
New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

`squ_21d162904c1c72cf8b39665f96480185c99dc2f9`

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

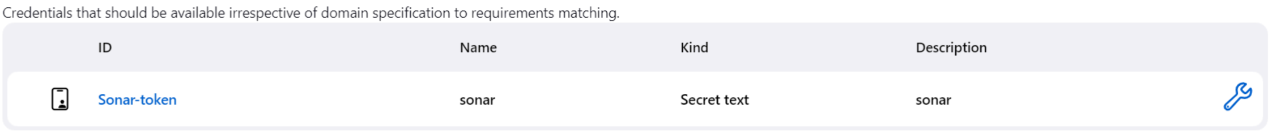


Secret: **POST THE TOKEN HERE**

ID ?: **Sonar-token**

Description ?: **Sonar-token**

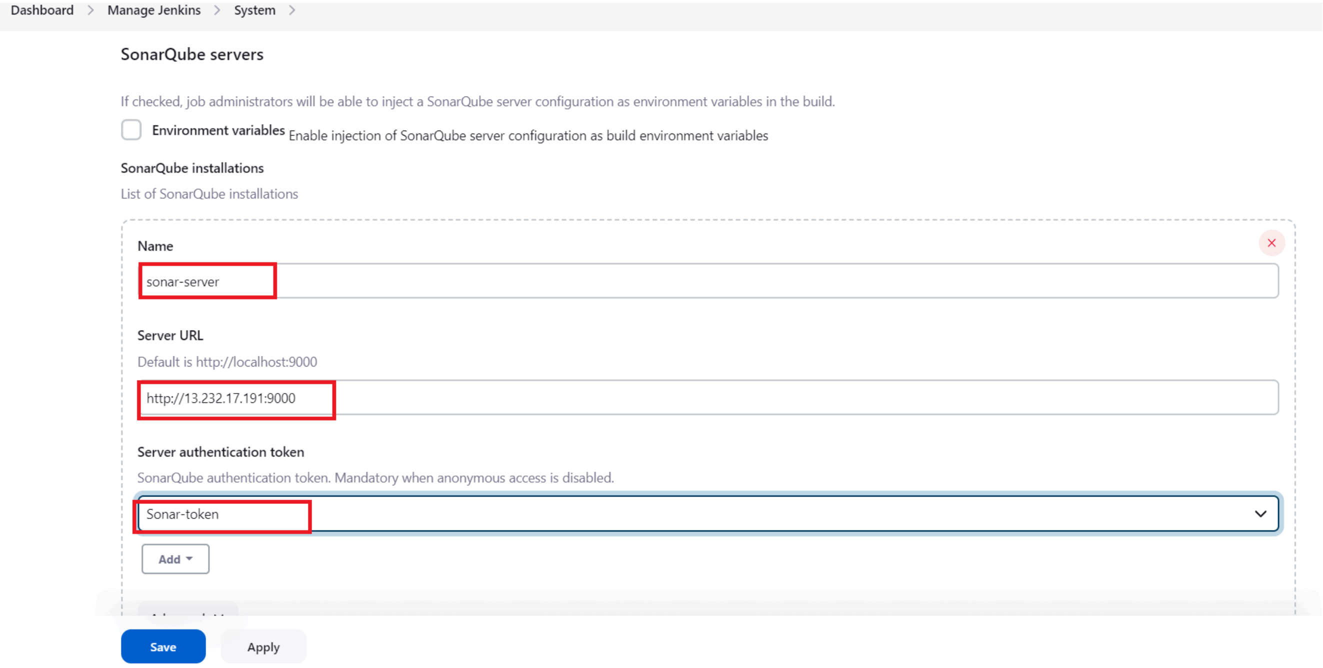
You will this page once you click on create



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

ID	Name	Kind	Description
Sonar-token	sonar	Secret text	sonar

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: sonar-server

Server URL

Default is http://localhost:9000

http://13.232.17.191:9000

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

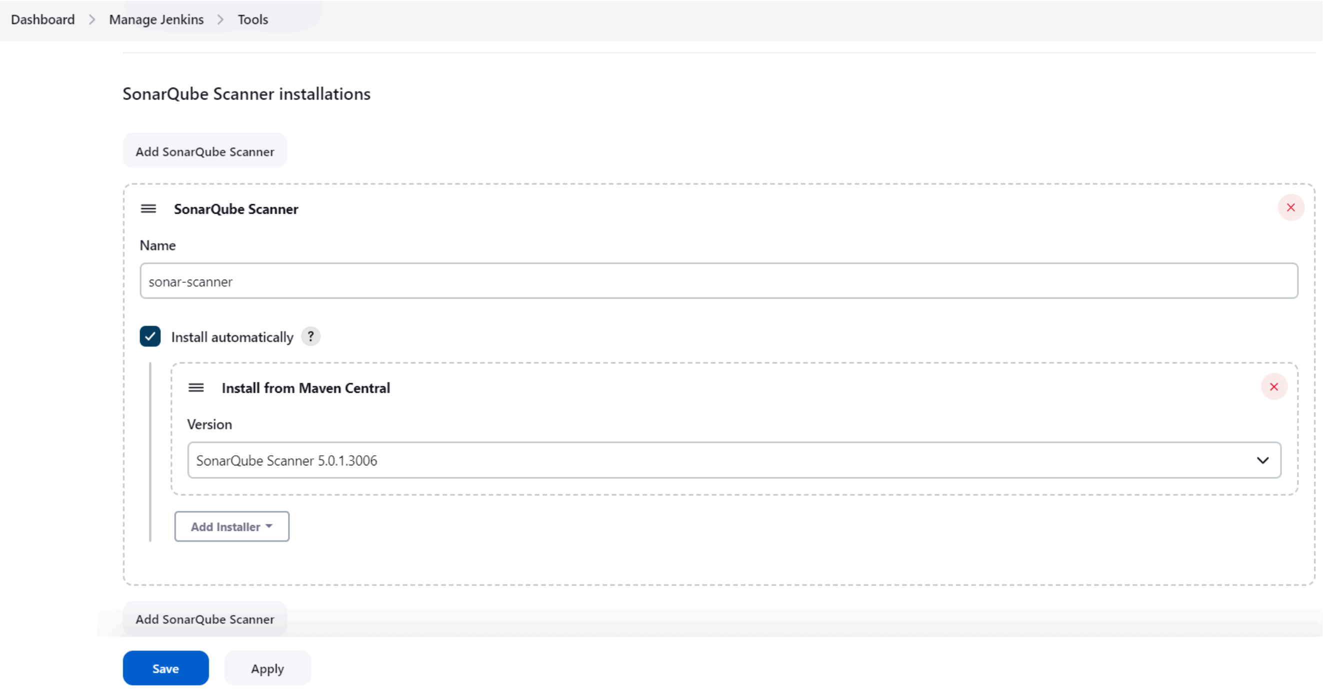
Sonar-token

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.



SonarQube Scanner installations

SonarQube Scanner

Name: sonar-scanner

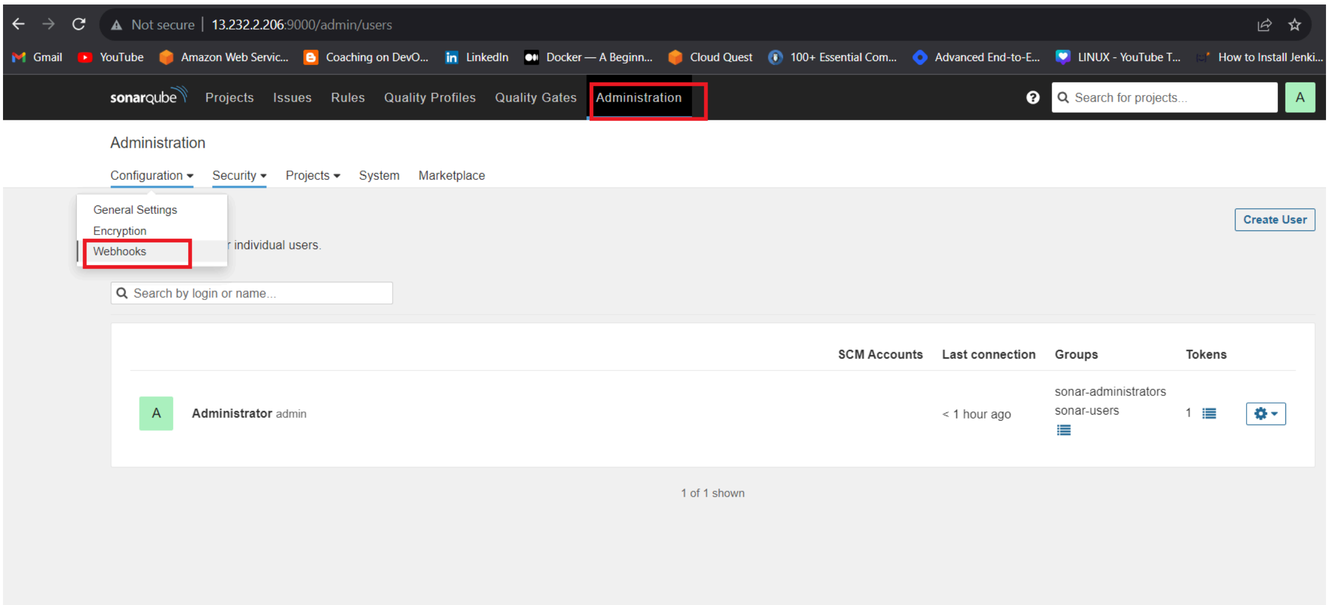
☑ Install automatically

Install from Maven Central

Version: SonarQube Scanner 5.0.1.3006

In the Sonarqube Dashboard add a quality gate also

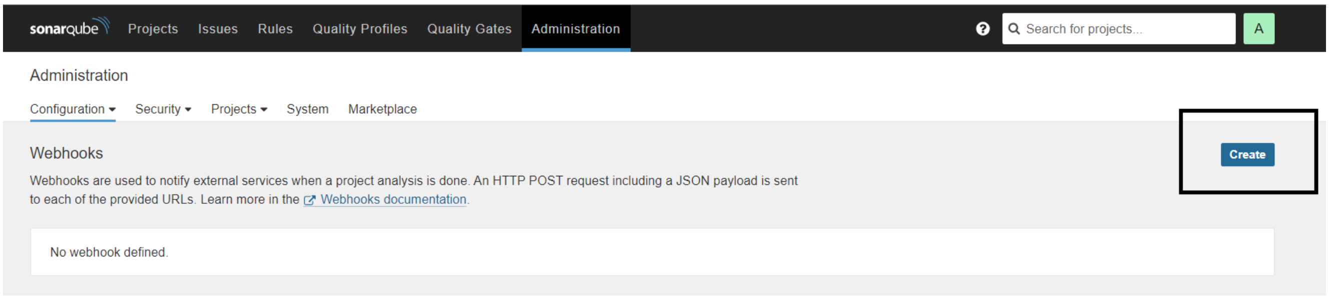
Administration–> Configuration–>Webhooks



	SCM Accounts	Last connection	Groups	Tokens
Administrator admin		< 1 hour ago	sonar-administrators sonar-users	1

1 of 1 shown

Click on Create



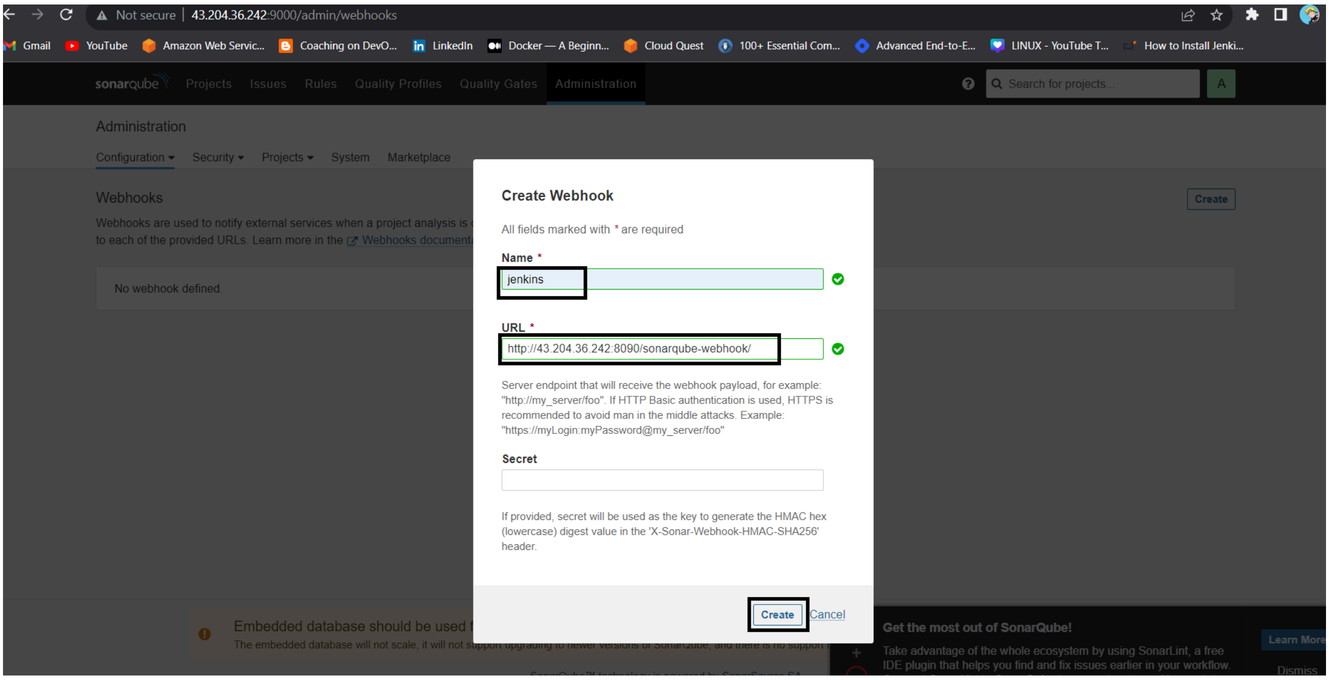
Webhooks

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.

No webhook defined.

Add details

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



Webhooks

Webhooks are used to notify external services when a project analysis is
to each of the provided URLs. Learn more in the Webhooks document

No webhook defined.

Create Webhook

All fields marked with * are required

Name *
jenkins

URL *
http://43.204.36.242:8090/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.

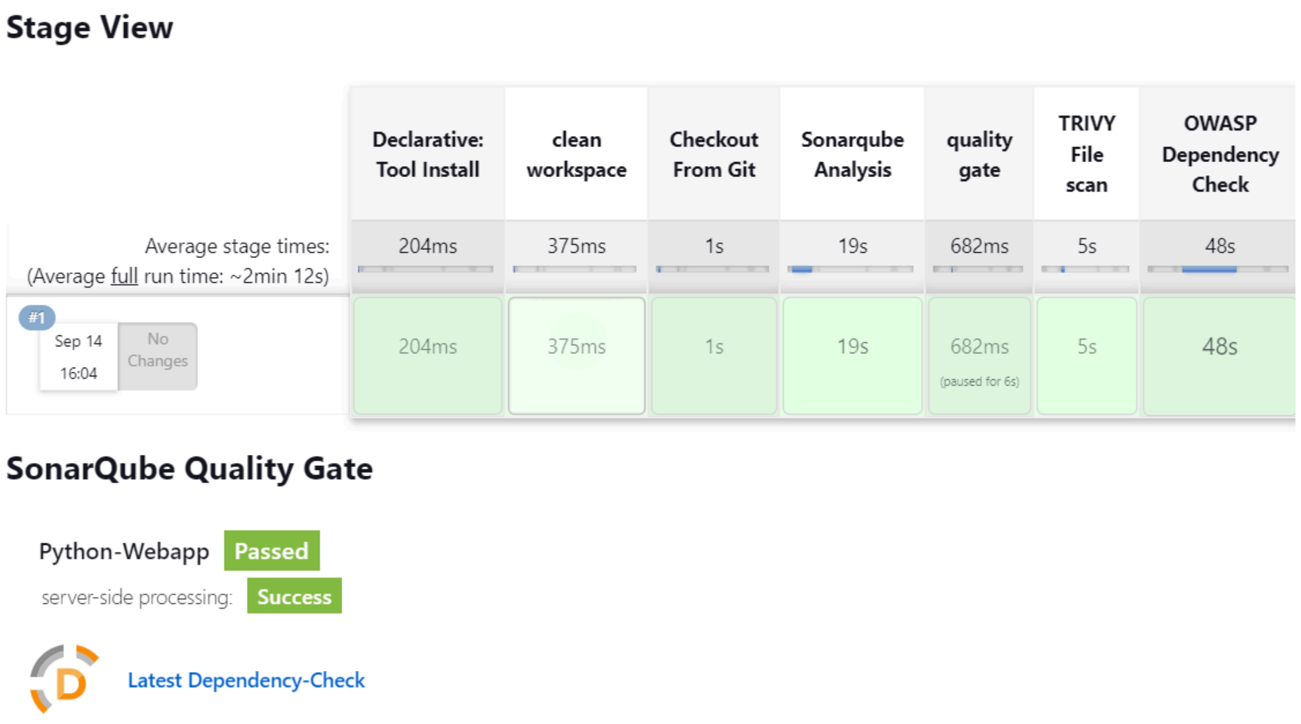
Embedded database should be used f
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support

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

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/Pytho
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.proje
                    -Dsonar.projectKey=Python-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 ./ --forma
                dependencyCheckPublisher pattern: '**/dependency-check-
            }
        }
    }
}
```

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

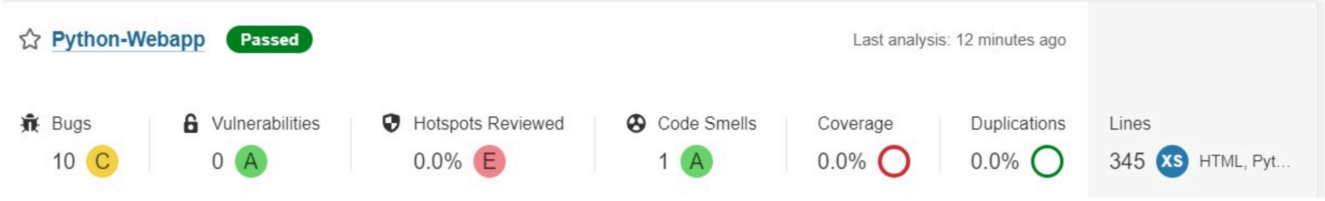
SonarQube Quality Gate

Python-Webapp Passed

server-side processing: Success

Latest Dependency-Check

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



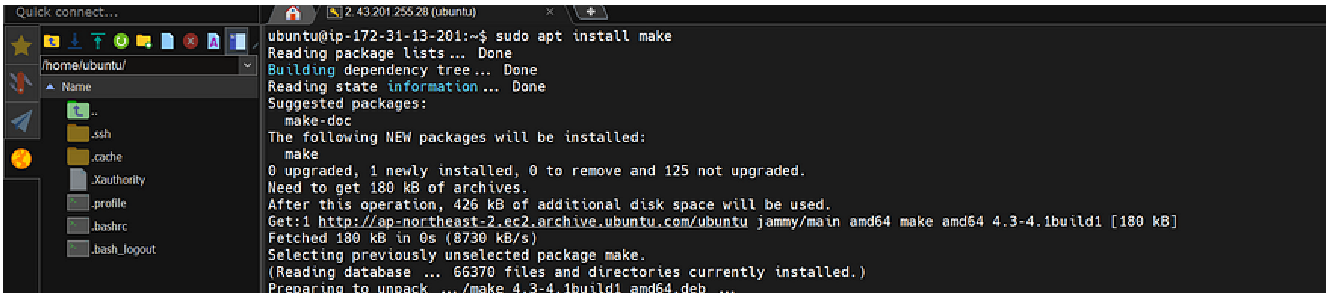
Python-Webapp Passed Last analysis: 12 minutes ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
10 C	0 A	0.0% E	1 A	0.0%	0.0%	345 XS HTML, Pyt...

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
```



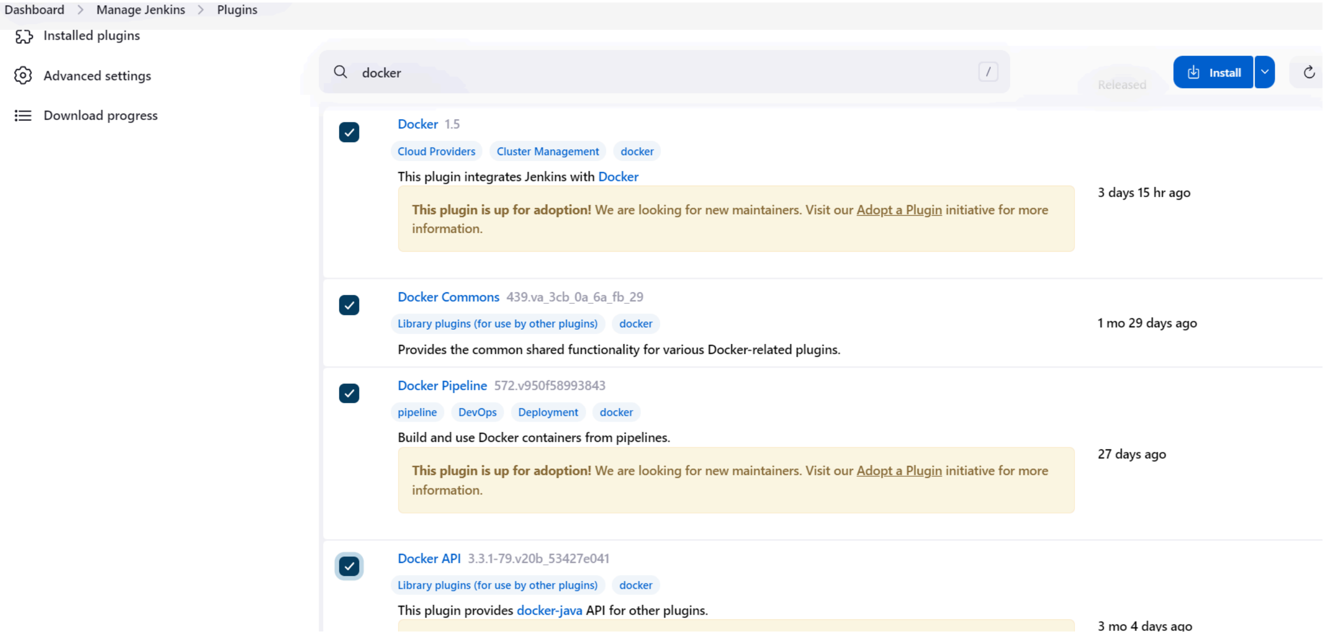
```
ubuntu@ip-172-31-13-201:~$ sudo apt install make
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 125 not upgraded.
Need 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



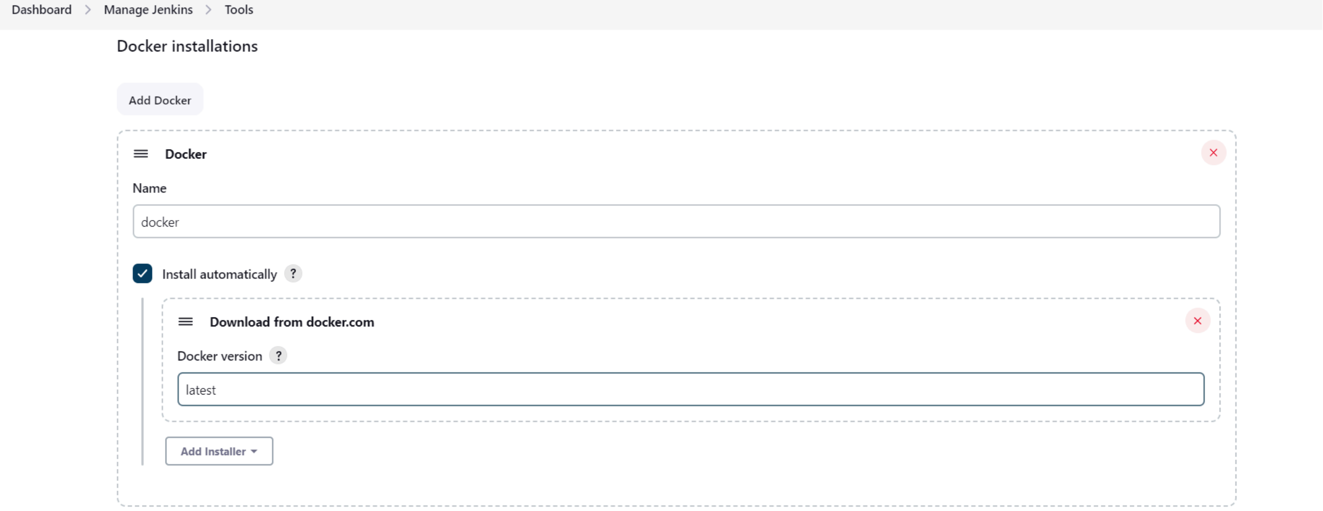
☑ Docker 1.5
Cloud Providers · Cluster Management · docker
This plugin integrates Jenkins with Docker
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.
3 days 15 hr ago

☑ Docker Commons 439.va_3cb_0a_6a_fb_29
Library plugins (for use by other plugins) · docker
Provides the common shared functionality for various Docker-related plugins.
1 mo 29 days ago

☑ Docker Pipeline 572.v950f58993843
pipeline · DevOps · Deployment · docker
Build and use Docker containers from pipelines.
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.
27 days ago

☑ Docker API 3.3.1-79.v20b_53427e041
Library plugins (for use by other plugins) · docker
This plugin provides docker-java API for other plugins.
3 mo 4 days ago

Now, goto Dashboard → Manage Jenkins → Tools →



Dashboard > Manage Jenkins > Tools

Docker installations

Add Docker

Docker

Name

docker

☑ Install automatically ?

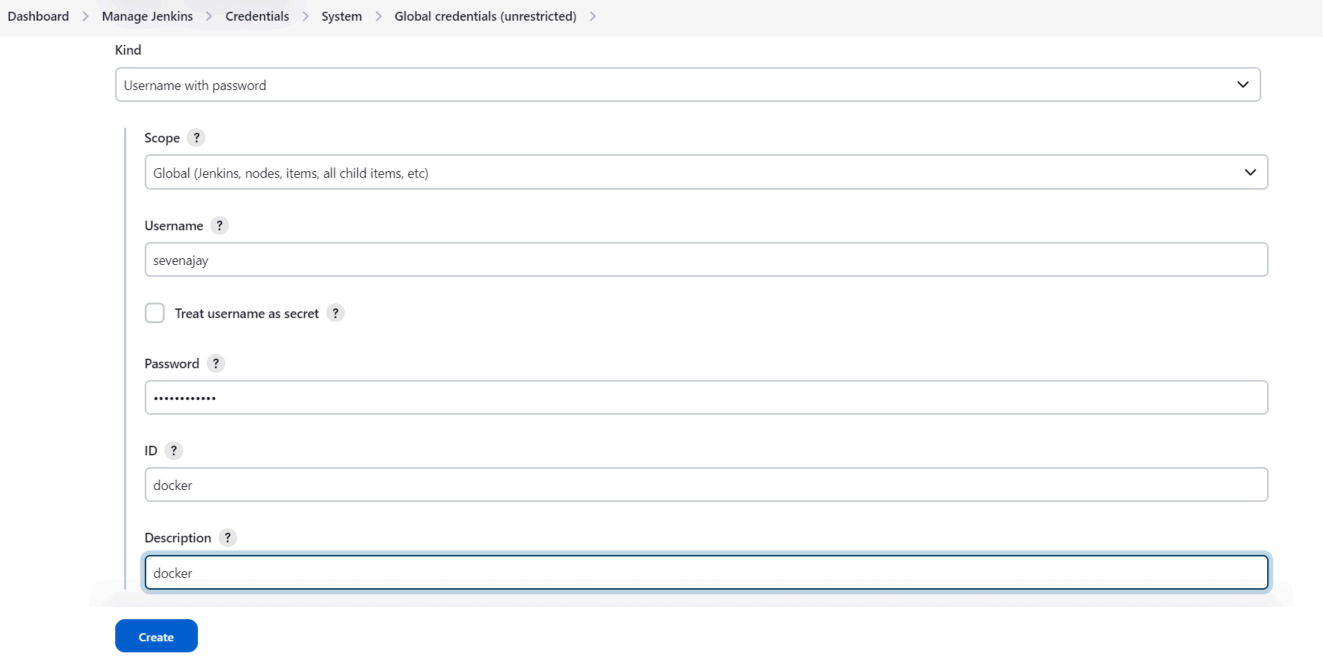
Download from docker.com

Docker version ?

latest

Add Installer ▾

Add DockerHub Username and Password under Global Credentials



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Kind

Username with password

Scope ?

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

Username ?

sevenajay

□ Treat username as secret ?

Password ?

••••••••••••

ID ?

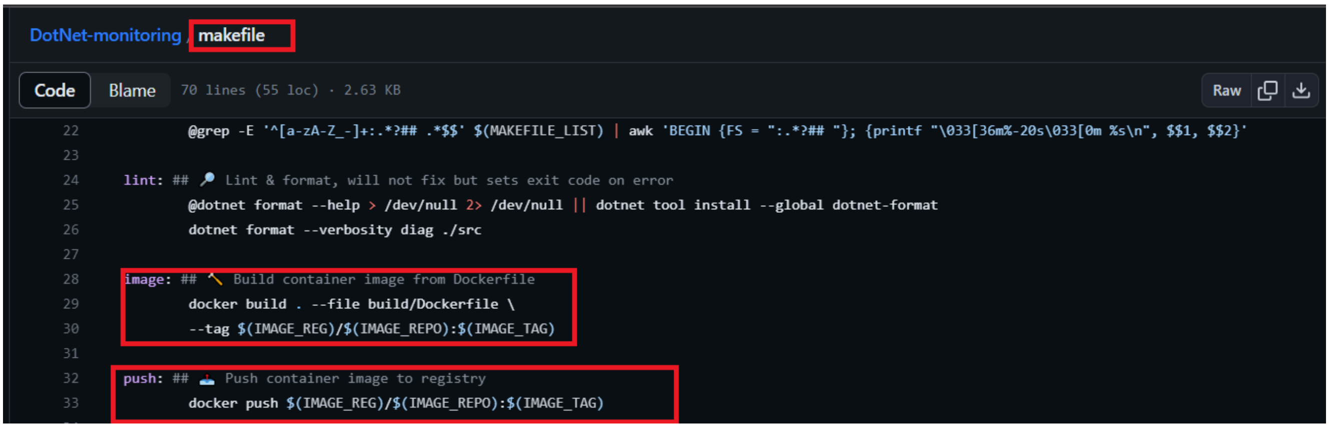
docker

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

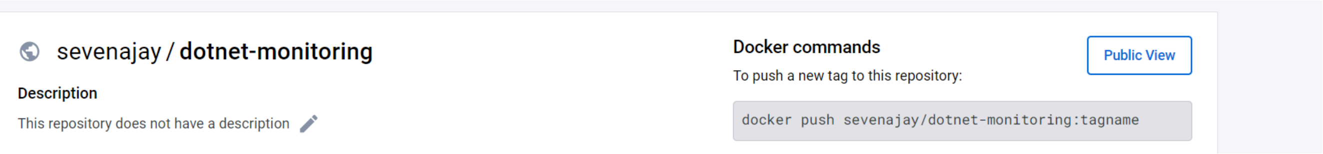
```
	@grep -E '^[a-zA-Z_-]+:.*?## .*$$' $(MAKEFILE_LIST) | awk 'BEGIN {FS = ":.*?## "}; {printf "\033[36m%-20s\033[0m %s\n", $$1, $$2}'

lint: ## 🔎 Lint & format, will not fix but sets exit code on error
	@dotnet format --help > /dev/null 2> /dev/null || dotnet tool install --global dotnet-format
	dotnet format --verbosity diag ./src

image: ## 🔨 Build container image from Dockerfile
	docker build . --file build/Dockerfile \
	--tag $(IMAGE_REG)/$(IMAGE_REPO):$(IMAGE_TAG)

push: ## 📤 Push container image to registry
	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/python-system-monitoring:lates
            }
        }
        stage("Docker Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName
                       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



sevenajay / **dotnet-monitoring**

Description

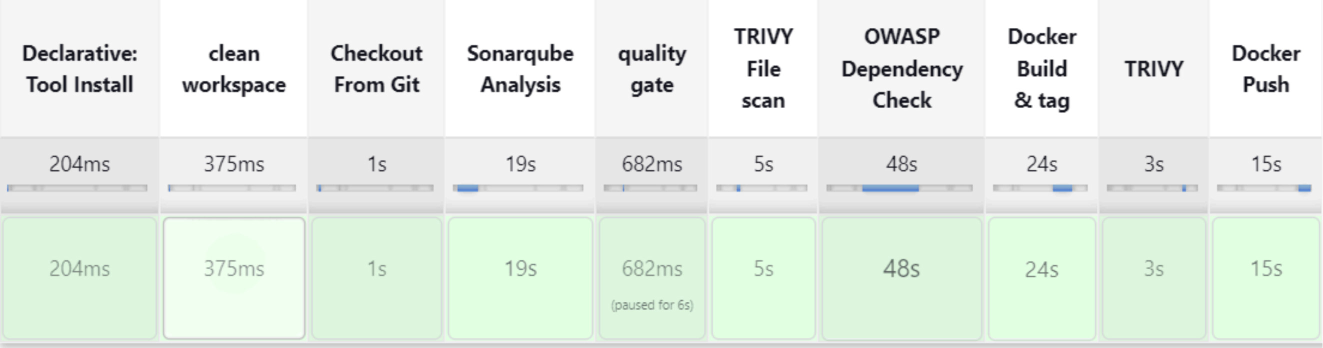
This repository does not have a description

Docker commands

To push a new tag to this repository:

```
docker push sevenajay/dotnet-monitoring:tagname
```

stage view



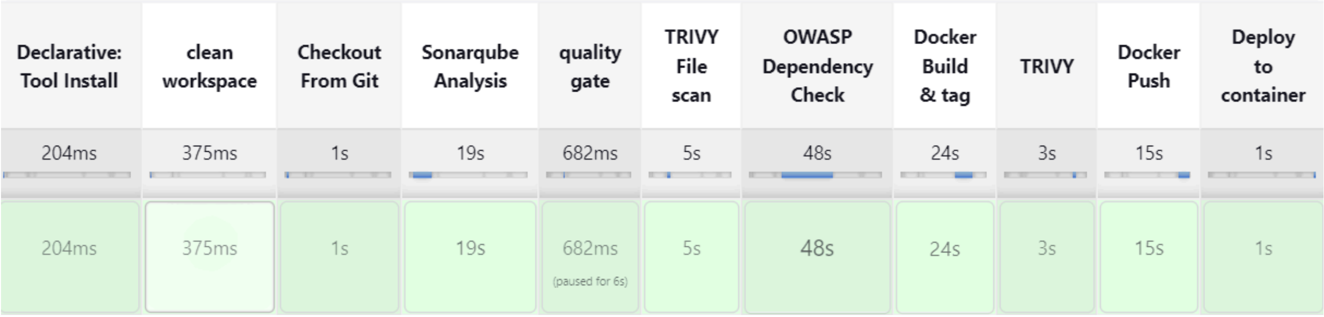
Declarative: Tool Install	clean workspace	Checkout From Git	Sonarqube Analysis	quality gate	TRIVY File scan	OWASP Dependency Check	Docker Build & tag	TRIVY	Docker Push
204ms	375ms	1s	19s	682ms	5s	48s	24s	3s	15s
204ms	375ms	1s	19s	682ms (paused for 6s)	5s	48s	24s	3s	15s

Step 8 – Deploy the image using Docker

Add this stage to your pipeline syntax

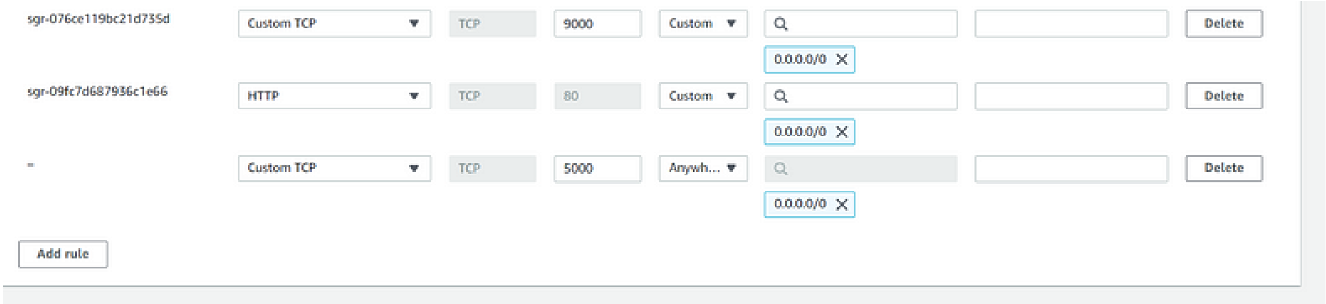
```
stage("Deploy to container"){
            steps{
                sh "docker run -d --name python1 -p 5000:5000 sevenajay,
            }
        }
```

You will see the Stage View like this,



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
204ms	375ms	1s	19s	682ms	5s	48s	24s	3s	15s	1s
204ms	375ms	1s	19s	682ms (paused for 6s)	5s	48s	24s	3s	15s	1s

(Add port 5000 to Security Group)



sgr-076ce119bc21d735d	Custom TCP	TCP	9000	Custom	0.0.0.0/0		Delete
sgr-09fc7d687936c1e66	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
-	Custom TCP	TCP	5000	Anywh...	0.0.0.0/0		Delete

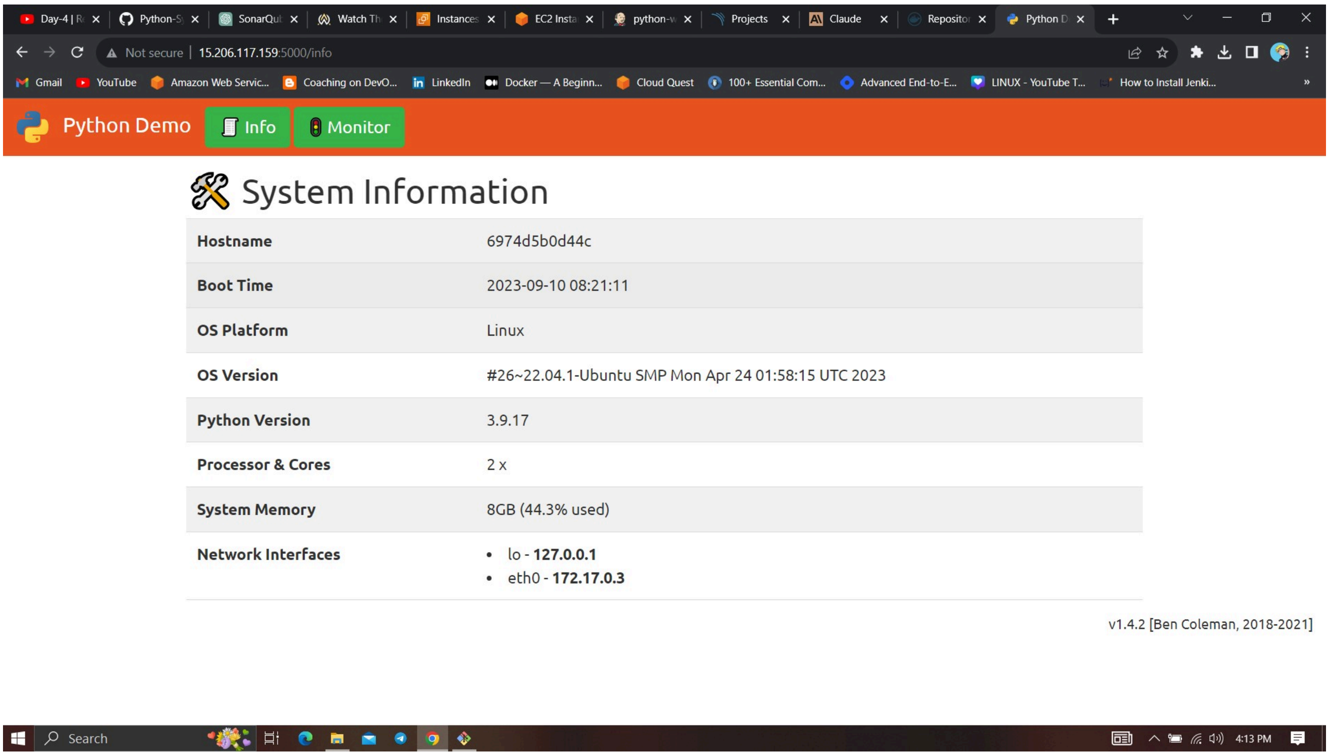
The final script looks like this,

```
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/Pytho
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.proje
                    -Dsonar.projectKey=Python-Webapp '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentials
                }
            }
        }
        stage("TRIVY File scan"){
            steps{
                sh "trivy fs . > trivy-fs_report.txt"
            }
        }
        stage("OWASP Dependency Check"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ --forma
                dependencyCheckPublisher pattern: '**/dependency-check-
            }
        }
        stage("Docker Build & tag"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName
                       sh "make image"
                    }
                }
            }
        }
        stage("TRIVY"){
            steps{
                sh "trivy image sevenajay/python-system-monitoring:lates
            }
        }
        stage("Docker Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName
                       sh "make push"
                    }
                }
            }
        }
        stage("Deploy to container"){
            steps{
                sh "docker run -d --name python1 -p 5000:5000 sevenajay/
            }
        }
    }
}
```

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
```

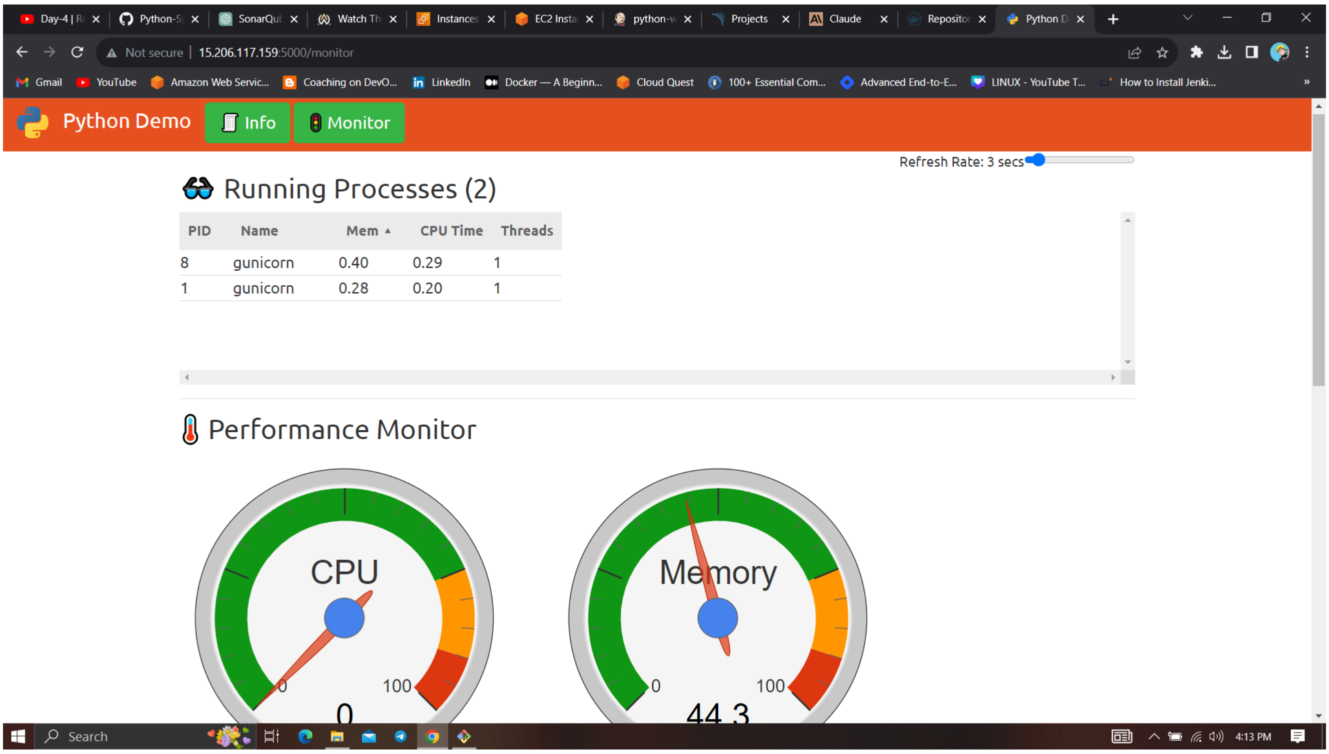
Step 9 — Access the Real World Application



System Information

Hostname	6974d5b0d44c
Boot Time	2023-09-10 08:21:11
OS Platform	Linux
OS Version	#26~22.04.1-Ubuntu SMP Mon Apr 24 01:58:15 UTC 2023
Python Version	3.9.17
Processor & Cores	2 x
System Memory	8GB (44.3% used)
Network Interfaces	• lo - **127.0.0.1** • eth0 - **172.17.0.3**

v1.4.2 [Ben Coleman, 2018-2021]



Running Processes (2)

PID	Name	Mem ▲	CPU Time	Threads
8	gunicorn	0.40	0.29	1
1	gunicorn	0.28	0.20	1

Performance Monitor

Step 10 — Terminate the AWS EC2 Instance

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

If this post was helpful, please follow and click the like button below to show your support.

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 "CI-CD DevSecOps project with Jenkins | Python webapp"

reetesh
7 February 2025

HI Sir

i am fallow your document but not understanding wher to put this 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/Aj7Ay/Python-System-Monitoring.git’
}
}
stage(“Sonarqube Analysis “){
steps{
withSonarQubeEnv(‘sonar-server’) {
sh ”’ $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Python-Webapp \
-Dsonar.projectKey=Python-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’
}
}
}
}
```

Reply

Kiran Gowda
3 March 2025

Step 1: Open Jenkins

Go to Jenkins Dashboard → New Item
Choose Pipeline and name your job

Click OK
Step 2: Configure Pipeline

Scroll down to Pipeline section
Select Pipeline Script
Paste your pipeline script
Step 3: Save and Run

Click Save
Click Build Now to run the pipeline

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

Follow Us on YouTube