



Zomato Clone App with DevSecOps CI/CD

Capstone Project

Personal Information

Full Name: Stanley A
Email Address: antony.stanley@cprime.com
Phone Number: 9941127755
Location: Chennai, TamilNadu

Professional Information

Company Name: CPrime
Job Title: Apprentice

Github URL

<https://github.com/stanleycprime/Zomato-Clone-App-with-DevSecOps-CI-CD>

Description

This project implements a DevSecOps CI/CD pipeline for a Zomato Clone App using Jenkins, Docker, SonarQube, Trivy, and OWASP Dependency Check to automate code scanning, containerization, and deployment on an AWS EC2 instance.

Resources

<https://mrcloudbook.com/zomato-clone-app-with-devsecops-ci-cd/>

Zomato Clone App with DevSecOps CI/CD

Overview

This DevSecOps project implements a secure CI/CD pipeline for a **Zomato Clone Node.js app** on an **Ubuntu 22.04 AWS EC2 instance**. It uses **Jenkins** for orchestration, **SonarQube** and **OWASP Dependency-Check** for static code analysis, and **Trivy** for container vulnerability scanning. Docker is used to containerize the application, which is then pushed to **DockerHub** and deployed locally on the instance. Jenkins automates the entire process from code checkout to deployment. The setup ensures early detection of security issues and code quality enforcement. Finally, the EC2 instance is terminated to optimize resource usage and cost.

Tools and Services

1. GitHub

GitHub is a cloud-based platform for version control and collaboration, built around Git. It allows developers to host and manage code repositories, track changes, and collaborate on projects. In this project, GitHub is used to store and retrieve the Zomato Clone source code, enabling CI/CD automation.

2. Node.js

Node.js is a JavaScript runtime built on Chrome's V8 engine, used to run JavaScript code outside the browser. It powers server-side applications and supports scalable development. In your project, Node.js runs the backend of the Zomato Clone app and manages dependencies using npm.

3. Amazon EC2 (Elastic Compute Cloud)

EC2 is a web service from AWS that provides resizable compute capacity in the cloud. It allows you to launch virtual machines (instances) for hosting applications. Here, an Ubuntu EC2 instance is used as the base server for setting up Jenkins, Docker, SonarQube, and the deployment environment.

4. PuTTY & Pageant

PuTTY is a free SSH and telnet client for Windows used to remotely access servers. Pageant is its companion tool that holds private SSH keys. In your setup, PuTTY connects to the EC2 instance, while Pageant manages your SSH key for secure authentication.

5. Jenkins

Jenkins is an open-source automation server used to automate parts of software development related to building, testing, and deploying. It supports plugins to integrate with almost any tool. In this project, Jenkins orchestrates the CI/CD pipeline, running scans, builds, and deployments automatically.

6. Jenkins Plugins

Jenkins plugins extend the platform's capabilities. Plugins like **NodeJS**, **JDK**, **SonarQube Scanner**, **Docker Pipeline**, and **OWASP Dependency Check** are added to enable integrations with tools such as SonarQube, Docker, and static analysis scanners, making Jenkins a powerful DevSecOps orchestrator.

7. Docker

Docker is a containerization platform that allows packaging applications with their dependencies into isolated units (containers). It simplifies deployment and scalability. In your pipeline, Docker is used to run SonarQube, build Docker images of the app, and deploy containers from those images.

8. SonarQube

SonarQube is an open-source platform for continuous inspection of code quality. It performs automatic reviews with static code analysis to detect bugs, vulnerabilities, and code smells. In this pipeline, it is used to ensure code quality of the Node.js app before build and deployment stages.

9. Trivy

Trivy is a simple and comprehensive vulnerability scanner for containers and other artifacts. It checks for OS vulnerabilities and misconfigurations. Here, Trivy is integrated into the Jenkins pipeline to scan Docker images, adding a security layer before pushing images to Docker Hub.

10. OWASP Dependency-Check

OWASP Dependency-Check is a software composition analysis (SCA) tool that identifies known vulnerabilities in project dependencies. It checks for CVEs in third-party libraries. In this project, it's integrated into Jenkins to scan the application's node_modules and display detailed security reports in the pipeline.

11. SonarQube Webhooks

Webhooks in SonarQube enable integration with external systems by sending build status updates. In this project, a webhook sends code analysis results back to Jenkins, allowing the pipeline to decide whether to proceed based on the SonarQube Quality Gate results.

12. DockerHub

DockerHub is a cloud-based container registry that lets you store and distribute Docker images. It integrates with Jenkins to push container images after a successful build. The Zomato Docker image is pushed here, enabling easy deployment or sharing.

13. Jenkins Pipeline

Jenkins Pipeline is a suite of plugins that supports integrating and implementing continuous delivery pipelines as code. You define your build, test, and deploy steps in a Jenkinsfile. In this setup, the Zomato Clone CI/CD process is fully automated through a scripted pipeline.

14. Jenkins Credentials

Jenkins securely stores credentials (tokens, passwords, SSH keys) used by pipelines. You use it to store the SonarQube token and DockerHub credentials. These credentials are referenced in the pipeline to authenticate with SonarQube and DockerHub during analysis and image push stages.

15. Quality Gates (SonarQube)

A Quality Gate is a set of conditions SonarQube uses to determine whether a project passes or fails the quality check. It can block builds with critical vulnerabilities or low test coverage. You used the default Quality Gate to ensure code quality before deployment.

16. Jenkins Plugin - Pipeline Stage View

The Pipeline Stage View plugin provides a visual representation of stages in a Jenkins pipeline. It shows stage-by-stage status of the pipeline including success, failure, or skipped steps. This visual feedback helps quickly identify issues in complex pipelines like yours.

17. Jenkins Tool Configuration

Jenkins Tool Configuration allows you to define versions of tools like JDK, Node.js, SonarQube Scanner globally. You configured JDK 17 and Node 16 for consistent builds across jobs. This ensures compatibility between your code and the environment it runs in.

18. Ubuntu (on EC2)

Ubuntu is a widely used, Debian-based Linux OS known for its stability and community support. In this project, Ubuntu is the operating system of the EC2 instance, chosen for its ease of package management and compatibility with Jenkins, Docker, and DevSecOps tools.

19. Security Groups (AWS EC2)

Security Groups in AWS act like virtual firewalls that control inbound and outbound traffic for EC2 instances. You configure rules to allow specific ports (3000, 3001, 8080, 9000) for the app, Jenkins, and SonarQube, ensuring secure and controlled network access.

20. Terminating Instances (AWS EC2)

Terminating an EC2 instance permanently deletes the virtual machine and stops billing. It's a best practice to terminate unused instances, especially large ones (like t2.large) after project completion to save on AWS costs.

Workflow

Step 1 — Launch an Ubuntu (22.04) T2 Large EC2 Instance

An AWS EC2 instance with **Ubuntu 22.04 (t2.large)** is launched to serve as the CI/CD host. It offers enough CPU and memory to run Jenkins, Docker, and SonarQube simultaneously. Security groups are configured to allow SSH, HTTP and HTTPS.

Step 2 — Install Jenkins, Docker, and Trivy. Create SonarQube Container Using Docker

Jenkins is installed for automation, Docker for containerization, and Trivy to scan images for vulnerabilities. A SonarQube container is created using Docker, exposing it on port 9000. This setup ensures quick deployment and avoids manual installation overhead for SonarQube.

Step 3 — Install Jenkins Plugins (JDK, Node.js, SonarQube Scanner, OWASP Dependency Check)

Jenkins plugins for **Java (JDK 17)**, **Node.js**, **SonarQube Scanner**, and **OWASP Dependency-Check** are installed and configured. These tools enable Jenkins to run Java-based scanners, execute Node.js apps, and analyze the codebase for vulnerabilities and quality issues during pipeline execution.

Step 4 — Create a Pipeline Project in Jenkins Using a Declarative Pipeline

A new Jenkins pipeline project is created using a **Declarative Pipeline syntax** in a `Jenkinsfile`. This file defines all CI/CD steps like code checkout, static analysis, Trivy scan, Docker build, push, and deployment. It ensures the entire workflow is version-controlled and automated.

Step 5 — Install and Configure OWASP Dependency Check

The **OWASP Dependency-Check plugin** is configured in Jenkins to scan the project's Node.js dependencies for known vulnerabilities. It generates an HTML report and fails the pipeline if high-severity vulnerabilities are detected, enforcing dependency-level security checks.

Step 6 — Docker Image Build and Push

Jenkins builds a Docker image of the Zomato Clone app and tags it with the appropriate version. After a successful build and vulnerability scan with Trivy, the image is **pushed to DockerHub** using credentials securely stored in Jenkins.

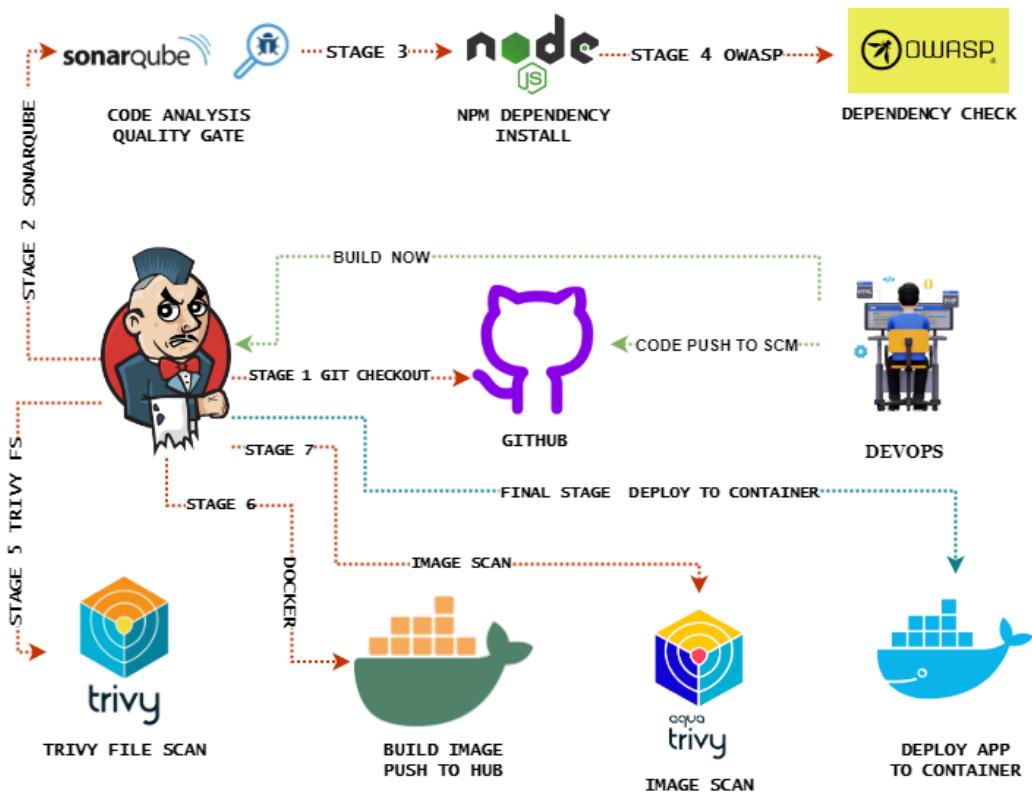
Step 7 — Deploy the Image Using Docker

The pushed Docker image is pulled and deployed on the EC2 instance using Docker. This step validates that the container runs correctly and exposes the Node.js app on ports like 3000 or 3001. This marks the end-to-end DevSecOps deployment cycle.

Step 8 — Terminate the AWS EC2 Instance

After verifying successful deployment, the EC2 instance is **terminated** to stop billing and release resources. This is a best practice to avoid unnecessary charges, especially after running compute-intensive tools like SonarQube on t2.large instances.

Architecture Diagram



Pre Check the Code

1. Get the code from Github:

<https://github.com/stanleycprime/Zomato-Clone>

2. Install the node modules for the Source Code

npm install

```
C:\Users\A Stanley\Downloads\Zomato-Clone-main\Zomato-Clone-main>npm install
npm warn deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility
npm warn deprecated rollup-plugin-terser@7.0.2: This package has been deprecated and is no longer maintained. Please use @rollup/plugin-terser
npm warn deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead
npm warn deprecated w3c-hr-time@1.0.2: Use your platform's native performance.now() and performance.timeOrigin.
npm warn deprecated svgo@1.3.2: This SVGO version is no longer supported. Upgrade to v2.x.x.

added 1506 packages, and audited 1507 packages in 4m

235 packages are looking for funding
  run 'npm fund' for details

35 vulnerabilities (3 low, 13 moderate, 17 high, 2 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run 'npm audit' for details.
```

3. Now start running the application

npm start

```
C:\Users\A Stanley\Downloads\Zomato-Clone-main\Zomato-Clone-main>npm start
> zomato_clone@0.1.0 start
> react-scripts start
✖ Something is already running on port 3000.

Would you like to run the app on another port instead? ... yes
Browserslist: caniuse-lite is outdated. Please run:
  npx update-browserslist-db@latest
  Why you should do it regularly: https://github.com/browserslist/update-db#readme
  (Node:15080) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
  (Use 'node --trace-deprecation ...' to show where the warning was created)
  (Node:15080) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

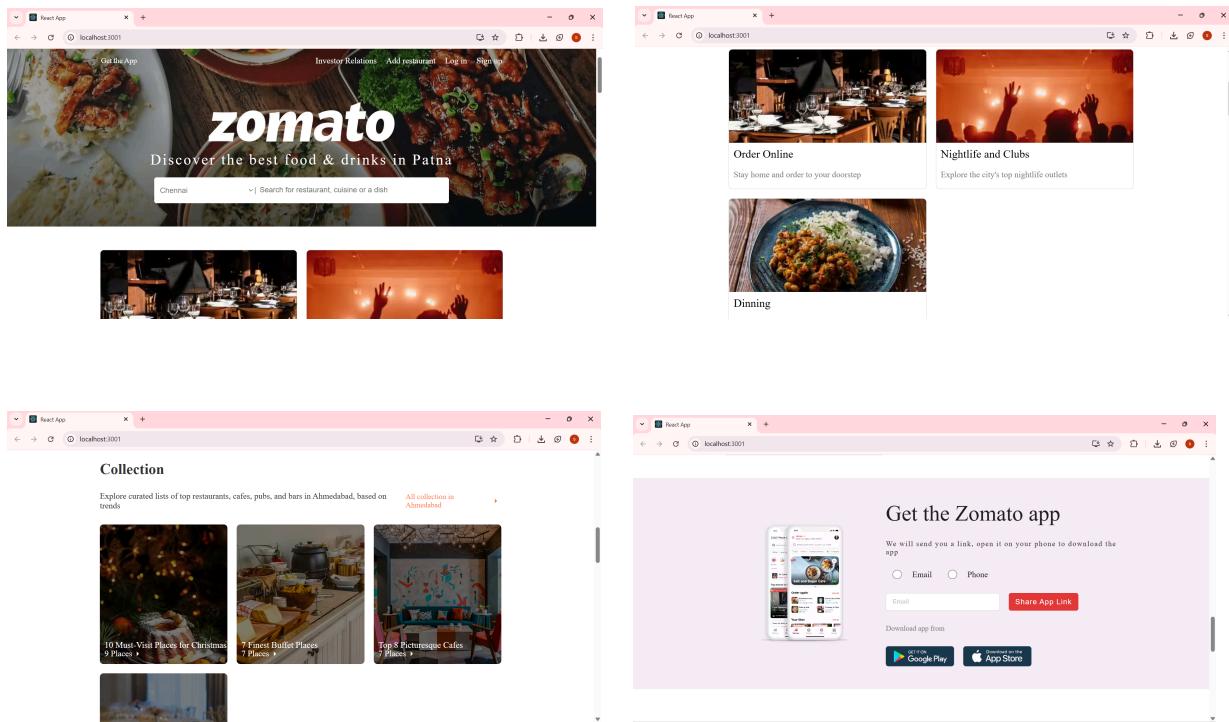
You can now view zomato_clone in the browser.

  Local:          http://localhost:3001
  On Your Network: http://172.26.0.1:3001

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

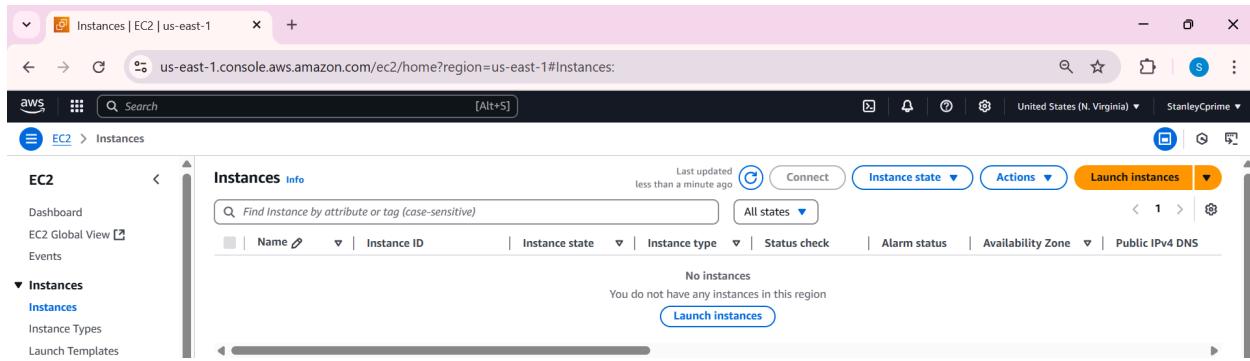
4. Open the browser port as mentioned (3001). Here we run as localhost.
<http://localhost:3001/>



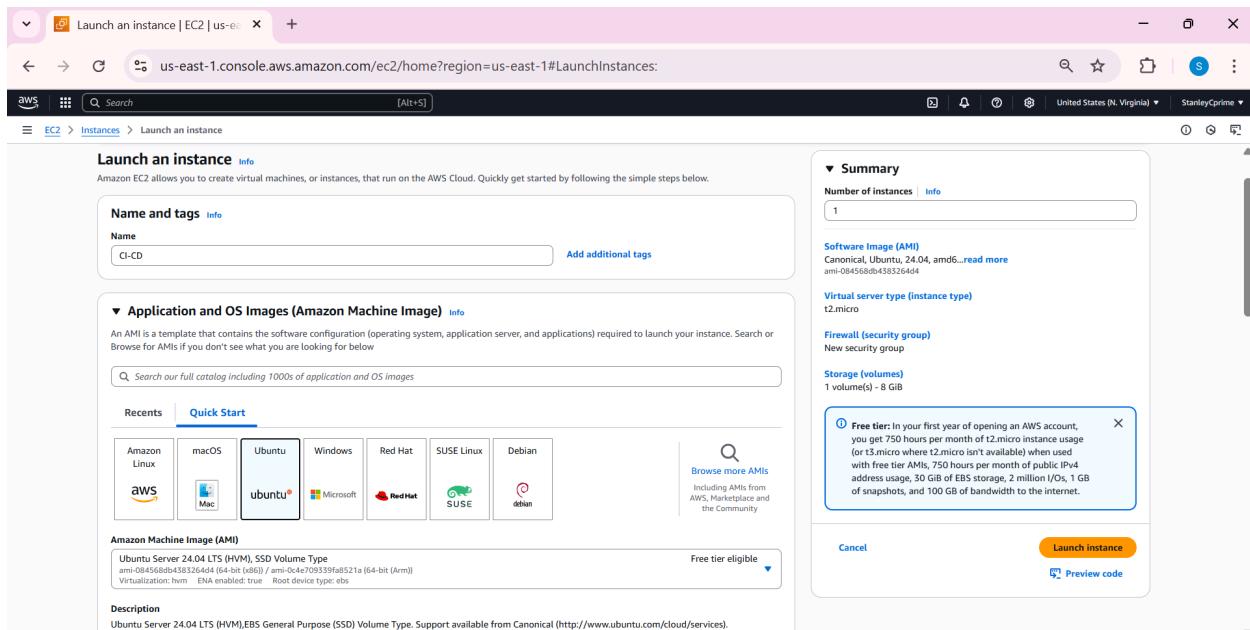
Step by Step Implementation

Step 1 : Launch an Ubuntu - t2.large Instance

1. Launch an AWS instance by going to AWS EC2 services
2. There click Launch Instance on top right corner



3. Give CI-CD as Instance name and choose Ubuntu OS



4. Select t2.large as Instance type
5. Choose the already existing key pair (nvirginia-key)

6. Choose default VPC
7. Create a new Security Group (that allows HTTP and HTTPS)
(don't allow any unnecessary ports)

8. Choose storage as 40 GB

The screenshot shows the 'Configure storage' section of the AWS EC2 instance creation wizard. It displays a 40 GiB gp3 root volume with 3000 IOPS and 'Not encrypted'. A note states that the selected AMI contains instance store volumes, which are not accessible from the instance. Below this, there's a link to 'Click refresh to view backup information' and a section for 'File systems'. On the right, there are tabs for 'virtual server type instance type' (t2.large), 'Firewall (security group)' (New security group), 'Storage (volumes)' (1 volume(s) - 40 GiB), and a 'Free tier' information box. At the bottom are 'Launch instance' and 'Preview code' buttons.

9. Then click Launch Instance

The screenshot shows the AWS EC2 Instances page. It lists one instance named 'CI-CD' with Instance ID 'i-05ffb9e2c6f027a75'. The instance is shown as 'Running' with an 't2.large' instance type. Other columns include 'Name', 'Status check', 'Alarm status', 'Availability Zone', 'Public IPv4 DNS', and 'Public IP'. The 'Actions' dropdown menu for this instance includes 'Launch instances'.

10. Add our instance key in Pageant (nvirginia-key)

The screenshot shows the Pageant Key List window. It displays an RSA key entry with a fingerprint of 'SHA256:SOb7EIYqhxCXSYkia2UQRqxZN+v+G1RgnSdBSE+UVQ'. The window includes fields for 'Fingerprint type' (set to 'SHA256'), 'Add Key' (disabled), 'Add Key (encrypted)', 'Re-encrypt', 'Remove', 'Help', and 'Close' buttons. At the bottom, there are tabs for 'AMI Catalog', 'Elastic Block Store Volumes', and 'Instances' (with 'i-05ffb9e2c6f027a75' selected).

11. Now open Putty and add the IP address of our instance

The screenshot shows the AWS Management Console with the EC2 service selected. On the left, the navigation pane includes 'EC2', 'Dashboard', 'EC2 Global View', 'Events', 'Instances' (selected), 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Instances', 'Capacity Reservations', 'Images' (AMIs, AMI Catalog), and 'Elastic Block Store' (Volumes, Snapshots). The main area displays 'Instances (1/1)' with one item: 'i-05ffb9e2c6f027a75 (CI-CD)'. Below it are 'Details' and 'Status and alarms' tabs, and a section for 'Instance summary' with 'Instance ID' i-05ffb9e2c6f027a75 and 'IPv6 address' listed as '-'. To the right, a 'PuTTY Configuration' window is open over the EC2 interface. The window has a sidebar titled 'Category' with sections like Session, Logging, Terminal, Keyboard, Bell, Features, Window, Appearance, Behaviour, Translation, Selection, Colours, and Connection. The 'Connection' section is expanded, showing Data, Proxy, SSH, Serial, Telnet, Rlogin, and SUPDUP. The 'SSH' tab is selected. In the main pane, under 'Basic options for your PuTTY session', the 'Host Name (or IP address)' field contains '107.22.148.208' and the 'Port' field is set to '22'. The 'Connection type' dropdown is set to 'SSH'. At the bottom of the PuTTY window are 'About', 'Help', 'Open', and 'Cancel' buttons.

12. In SSh - Auth - allow Agent Port Forwarding

This screenshot is similar to the previous one, showing the AWS EC2 interface and the PuTTY configuration window. The EC2 page shows the same instance 'CI-CD'. The PuTTY configuration window is open, specifically on the 'Session' tab. The 'Category' sidebar shows the 'Connection' section is expanded, with the 'SSH' tab selected. In the main pane, under 'Basic options for your PuTTY session', the 'Host Name (or IP address)' is '107.22.148.208' and the 'Port' is '22'. The 'Connection type' is 'SSH'. The 'Session' tab is selected. At the bottom are 'About', 'Help', 'Open', and 'Cancel' buttons. The status bar at the bottom of the PuTTY window shows 'Running'.

13. Now type ubuntu in login as user

14. With this our instance is set and connected with putty.

```
ubuntu@ip-172-31-39-134: ~
 2. Login as: ubuntu
 3. Authenticating with public key "nvirginia-key" from agent
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Wed May 14 05:31:56 UTC 2025

System load: 0.0 Processes: 113
Usage of /: 4.4% of 37.70GB Users logged in: 0
Memory usage: 2%
Swap usage: 0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-39-134:~$
```

Step 2 : Install Jenkins, Docker and Trivy

2A : To Install Jenkins

1. Now create a new file in Instance to run and install the dependencies needed for the Jenkins to run

```
ubuntu@ip-172-31-39-134:~$ ls
ubuntu@ip-172-31-39-134:~$ vi jenkins.sh
ubuntu@ip-172-31-39-134:~$ sudo chmod 777 jenkins.sh
```

2. Add this code inside the jenkins.sh

```
#!/bin/bash
sudo apt update -y
#sudo apt upgrade -y
wget -O https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk '$1~/^VERSION_CODENAME/ {print $2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.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
~
```

3. Now give executable permissions to the file (we give 777 permission)

4. Now run the script and download the packages

```
ubuntu@ip-172-31-39-134:~$ ls
ubuntu@ip-172-31-39-134:~$ vi jenkins.sh
ubuntu@ip-172-31-39-134:~$ sudo chmod 777 jenkins.sh
ubuntu@ip-172-31-39-134:~$ ./jenkins.sh
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:5 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1067 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [230 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [13.5 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1062 kB]
```

Error:

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
Job for jenkins.service failed because the control process exited with error code.
See "systemctl status jenkins.service" and "journalctl -xeu jenkins.service" for details.
* Jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
  Active: failed (Result: exit-code) since Wed 2025-05-14 05:41:45 UTC; 5s ago
    Process: 2708 ExecStart=/usr/bin/jenkins (code=exited, status=1/FAILURE)
   Main PID: 2708 (code=exited, status=1/FAILURE)
     CPU: 7ms

May 14 05:41:45 ip-172-31-39-134 systemd[1]: jenkins.service: Main process exited, code=exited, status=1/FAILURE
May 14 05:41:45 ip-172-31-39-134 systemd[1]: jenkins.service: Failed with result 'exit-code'.
May 14 05:41:45 ip-172-31-39-134 systemd[1]: Failed to start jenkins.service - Jenkins Continuous Integration Server.
May 14 05:41:45 ip-172-31-39-134 systemd[1]: jenkins.service: Scheduled restart job, restart counter is at 5.
May 14 05:41:45 ip-172-31-39-134 systemd[1]: jenkins.service: Start request repeated too quickly.
May 14 05:41:45 ip-172-31-39-134 systemd[1]: jenkins.service: Failed with result 'exit-code'.
May 14 05:41:45 ip-172-31-39-134 systemd[1]: Failed to start jenkins.service - Jenkins Continuous Integration Server.
May 14 05:41:51 ip-172-31-39-134 systemd[1]: jenkins.service: Start request repeated too quickly.
May 14 05:41:51 ip-172-31-39-134 systemd[1]: jenkins.service: Failed with result 'exit-code'.
May 14 05:41:51 ip-172-31-39-134 systemd[1]: Failed to start jenkins.service - Jenkins Continuous Integration Server.
ubuntu@ip-172-31-39-134:~$ ^C
```

This is because Java is not installed properly

```
ubuntu@ip-172-31-39-134:~$ java -version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-17-jre-headless # version 17.0.14+7-1~24.04, or
sudo apt install openjdk-21-jre-headless # version 21.0.6+7-1~24.04.1
sudo apt install default-jre          # version 2:1.17-75
sudo apt install openjdk-11-jre-headless # version 11.0.26+4-1ubuntul~24.04
sudo apt install openjdk-8-jre-headless # version 8u442-b06~us1~ubuntul~24.04
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-4
sudo apt install openjdk-20-jre-headless # version 20.0.2+9-1
sudo apt install openjdk-22-jre-headless # version 22~22ea-1
ubuntu@ip-172-31-39-134:~$ java --version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-17-jre-headless # version 17.0.14+7-1~24.04, or
sudo apt install openjdk-21-jre-headless # version 21.0.6+7-1~24.04.1
sudo apt install default-jre          # version 2:1.17-75
```

So we try changing the commands in jenkins.sh

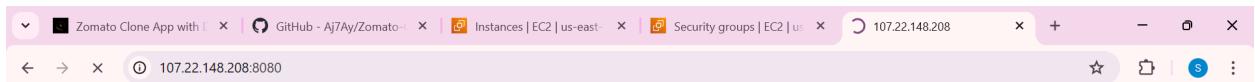
```
ubuntu@ip-172-31-39-134:~$ 
# /bin/bash
sudo apt update -y
#sudo apt upgrade -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk '/^VERSION_CODENAME/ {print $2}' /etc/os-release)" | tee /etc/apt/sources.list.d/adoptium.list
sudo apt update -y
sudo apt install openjdk-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
```

5. With this script, we also run and enable Jenkins

```
Reading state information... Done
jenkins is already the newest version (2.504.1).
0 upgraded, 0 newly installed, 0 to remove and 83 not upgraded.
• jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
  Active: active (running) since Wed 2025-05-14 05:48:11 UTC; 31ms ago
    Main PID: 6172 (java)
      Tasks: 48 (limit: 9505)
        Memory: 959.6M (peak: 963.7M)
          CPU: 14.879s
        CGroup: /system.slice/jenkins.service
           └─6172 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

May 14 05:48:08 ip-172-31-39-134 jenkins[6172]: Jenkins initial setup is required. An admin user has been created and a password generated.
May 14 05:48:08 ip-172-31-39-134 jenkins[6172]: Please use the following password to proceed to installation:
May 14 05:48:08 ip-172-31-39-134 jenkins[6172]: 85b0acf8237f445f9dd14e3854b4ec9
May 14 05:48:08 ip-172-31-39-134 jenkins[6172]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
May 14 05:48:08 ip-172-31-39-134 jenkins[6172]: ****
May 14 05:48:08 ip-172-31-39-134 jenkins[6172]: ****
May 14 05:48:08 ip-172-31-39-134 jenkins[6172]: ****
May 14 05:48:11 ip-172-31-39-134 jenkins[6172]: 2025-05-14 05:40:11.533+0000 [id=33]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed in
May 14 05:48:11 ip-172-31-39-134 jenkins[6172]: 2025-05-14 05:40:11.562+0000 [id=23]      INFO      hudson.lifecycle.Lifecycle#onReady: Jenkins is fully
May 14 05:48:11 ip-172-31-39-134 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Lines 1-20/20 (END)
```

6. Open the browser and open <public-ip>:8080



This site can't be reached

107.22.148.208 took too long to respond.

This is due to closed port in Security Group

7. So we choose our Security Group

The screenshot shows the AWS EC2 Security Groups page. The left sidebar is collapsed. The main area displays a table titled "Security Groups (1/3) Info". The columns are: Name, Security group ID, Security group name, VPC ID, and Description. The rows show:

Name	Security group ID	Security group name	VPC ID	Description
-	sg-028134b144e921da2	launch-wizard-2	vpc-08e4043f9384e6fad	launch-wizard-2 created 2025-05-14T0...
-	sg-09e8335bf17db5269	launch-wizard-1	vpc-08e4043f9384e6fad	launch-wizard-1 created 2025-04-24T1...
-	sg-03679d5c4583a92a8	default	vpc-08e4043f9384e6fad	default VPC security group

8. Select and Edit the Inbound Rules

The screenshot shows the AWS EC2 Security Groups page. The left sidebar is expanded, showing categories like Dashboard, Instances, Images, Elastic Block Store, and Network & Security. The main area shows the "Security Groups (1/3) Info" table with the same three security groups as before. Below it, a specific security group is selected: "sg-028134b144e921da2 - launch-wizard-2". The "Inbound rules" tab is selected. The table shows three inbound rules:

Name	Security group rule ID	Type	Protocol	Port range	Source	Description - optional
-	sgr-0422b25317c19b141	SSH	TCP	22	Custom	0.0.0.0/0
-	sgr-0464ea799f150e492	HTTPS	TCP	443	Custom	0.0.0.0/0
-	sgr-0f4d8a624d17478eb	HTTP	TCP	80	Custom	0.0.0.0/0
-	-	Custom TCP	TCP	8080	Anywhere	0.0.0.0/0

9. Add port 8080 as Custom TCP

The screenshot shows the "ModifyInboundSecurityGroupRules" page. The URL is "us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-028134b144e921da2". The main area is titled "Edit inbound rules". It shows the existing four rules and a new row being added at the bottom:

Type	Protocol	Port range	Source	Description - optional
SSH	TCP	22	Custom	0.0.0.0/0
HTTPS	TCP	443	Custom	0.0.0.0/0
HTTP	TCP	80	Custom	0.0.0.0/0
Custom TCP	TCP	8080	Anywhere	0.0.0.0/0

At the bottom right, there are buttons for "Cancel", "Preview changes", and "Save rules".

10. Click Save

The screenshot shows the AWS Management Console with the EC2 service selected. In the main pane, the 'Security Groups' section is displayed. A green banner at the top indicates that inbound security group rules were successfully modified. The table lists three security groups:

Name	Security group ID	Security group name	VPC ID	Description
-	sg-09e8335bf17db5269	launch-wizard-1	vpc-08e4043f9384e6fad	launch-wizard-1 created 2025-04-24T1...
-	sg-028134b144e921da2	launch-wizard-2	vpc-08e4043f9384e6fad	launch-wizard-2 created 2025-05-14T0...
-	sg-03679d5c4583a92a8	default	vpc-08e4043f9384e6fad	default VPC security group

11. Now try opening the jenkins dashboard

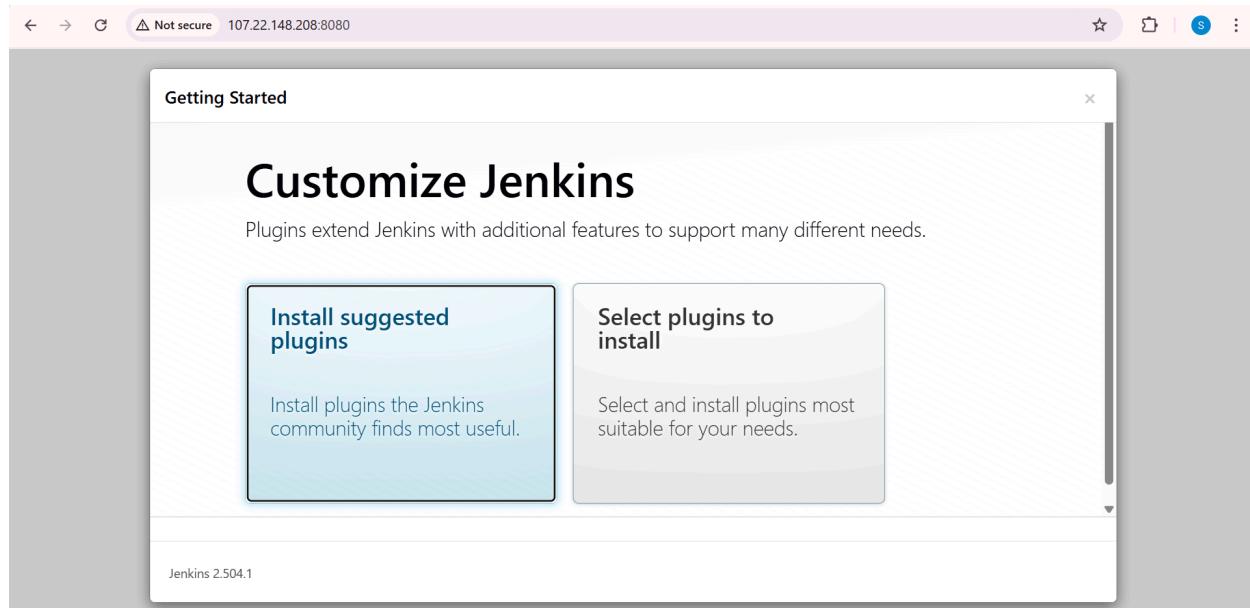
The screenshot shows a web browser window with the URL `107.22.148.208:8080/login?from=%2F`. The page title is "Sign in - Jenkins". The content is titled "Unlock Jenkins" and instructs the user to copy the password from the log file `/var/lib/jenkins/secrets/initialAdminPassword`. It includes a text input field for the "Administrator password" and a "Continue" button.

12. Now get the password from instance

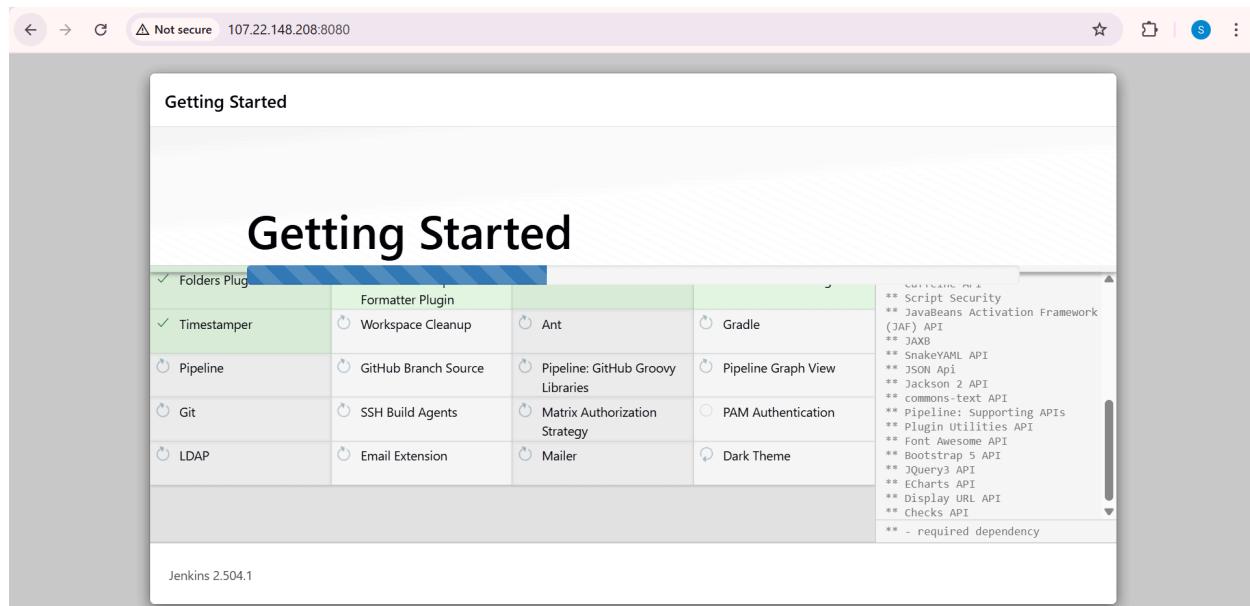
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

```
ubuntu@ip-172-31-39-134:~$ cat /var/lib/jenkins/secrets/initialAdminPassword
cat: /var/lib/jenkins/secrets/initialAdminPassword: Permission denied
ubuntu@ip-172-31-39-134:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
85b0acf8237f4d5f9ddd14e3854b4ec9
ubuntu@ip-172-31-39-134:~$
```

13. Click Install plugins and install necessary packages



14. Wait till the completion of the Plugins installation



15. Enter your jenkins credentials

Getting Started

Create First Admin User

Username
stan

Password
....

Confirm password
....

Jenkins 2.504.1

Skip and continue as admin **Save and Continue**

16. Click save and continue

Getting Started

Password
....

Confirm password
....

Full name
Stan

E-mail address
stan@gmail.com

Jenkins 2.504.1

Skip and continue as admin **Save and Continue**

17. With this we see the Jenkins URL

Getting Started

Instance Configuration

Jenkins URL: **http://107.22.148.208:8080**

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.504.1

Not now **Save and Finish**

18. This is your dashboard

The screenshot shows the Jenkins dashboard at the URL 107.22.148.208:8080. The top navigation bar includes links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. On the left, there are sections for 'Build Queue' (empty) and 'Build Executor Status' (0/2). The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' button, and a 'Set up a distributed build' section with 'Create a job' and '+' buttons, along with 'Set up an agent' and 'Configure a cloud' options.

2B : Install Docker

1. Open the instance in putty and install docker
2. Run these commands and install them
3. Also create a group for docker and add the user to it

```
ubuntu@ip-172-31-39-134:~$ sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER
newgrp docker
sudo chmod 777 /var/run/docker.sock
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker-doc
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 83 not upgraded.
Need to get 78.6 MB of archives.
```

4. Add port 9000 to Security Group to use Docker by choosing the Security Group.
5. Add the port number 9000 as Custom TCP

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0eb97e181be1b62e9	Custom TCP	TCP	8080	Custom	0.0.0.0/0
sgr-0422b25317c19b141	SSH	TCP	22	Custom	0.0.0.0/0
sgr-0464ea799f150e492	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0f4d8a624d17478eb	HTTP	TCP	80	Custom	0.0.0.0/0
-	Custom TCP	TCP	9000	Anywhere	0.0.0.0/0

Add rule

Save rules

6. Now save the changes made

Inbound security group rules successfully modified on security group (sg-028134b144e921da2 | launch-wizard-2)

Details

Security Groups (3)

Create security group

7. Now run and implement SonarQube in the container
(here we run using port 9000)

```
ubuntu@ip-172-31-39-134:~$ 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
215ed5a63843: Pull complete
094bfb4db7a: Pull complete
4df791be4da6: Pull complete
97a8e80e60c2: Pull complete
7be47dbb02a7: Pull complete
61c8ff67f948: Pull complete
2737e808f9c3: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:93f94e0ea6148cd02d52378049dad85d0f2d6c90c485578317f76addf2af9f3d
Status: Downloaded newer image for sonarqube:lts-community
ed94f18e4fead48d5a18c1d0b526362a5740c70552d2c3b5ab49d6afd23fa34e
ubuntu@ip-172-31-39-134:~$
```

8. Now open the SonarQube Dashboard by using <public-ip>:9000

Log in to SonarQube

Login

Password

Log in Cancel

9. Here we pass the credentials as Username - admin, Password - admin

Log in to SonarQube

admin

Log in Cancel

10. Also update and create new password

Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

New Password *

Confirm Password *

Update

11. This finally shows the SonarQube Dashboard

The screenshot shows the SonarQube 'Projects' creation page. At the top, there's a message about a new version available with a 'Learn More' link. Below the header, there are five cards for different CI/CD platforms:

- From Azure DevOps**: Shows the Azure DevOps logo and a 'Set up global configuration' button.
- From Bitbucket Server**: Shows the Bitbucket Server logo and a 'Set up global configuration' button.
- From Bitbucket Cloud**: Shows the Bitbucket Cloud logo and a 'Set up global configuration' button.
- From GitHub**: Shows the GitHub logo and a 'Set up global configuration' button.
- From GitLab**: Shows the GitLab logo and a 'Set up global configuration' button.

Below these cards, there's a section for manual project creation with a 'Manually' button and a double-angle bracket icon.

2C : Install Trivy

1. Now create a new file called trivy.sh
2. This file is to run and enable the Trivy package

```
ubuntu@ip-172-31-39-134:~$ 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 | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy -y
```

3. Now make the file executable using 777 permissions
4. Now execute the file and execute the commands

```
ubuntu@ip-172-31-39-134:~$ sudo chmod 777 trivy.sh
ubuntu@ip-172-31-39-134:~$ ./trivy.sh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wget is already the newest version (1.21.4-1ubuntu4.1).
wget set to manually installed.
lsb-release is already the newest version (12.0-2).
lsb-release set to manually installed.
The following additional packages will be installed:
  dirmngr gnupg-110n gnupg-utils gpg gpg-agent gpg-wks-client gpgconf gpgsm gpgv keyboxd
Suggested packages:
  pinentry-gnome3 tor parcellite xloadimage gpg-wks-server scdaemon
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  dirmngr gnupg gnupg-110n gnupg-utils gpg gpg-agent gpg-wks-client gpgconf gpgsm gpgv keyboxd
11 upgraded, 1 newly installed, 0 to remove and 72 not upgraded.
Need to get 2295 kB of archives.
After this operation, 35.8 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 gpg-wks-client amd64 2.4.4-2ubuntu17.2 [70.9 kB]
```

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

3A : Install Plugin

1. Open Jenkins Dashboard
2. Click Manage Jenkins and choose Plugins

The screenshot shows the Jenkins Manage Jenkins interface. At the top, there's a warning about Java 17 being end-of-life: "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)". Below this are buttons for "Set up agent", "Set up cloud", and "Dismiss". On the left, there are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (0/2). In the center, a prominent message says "Java 17 end of life in Jenkins" with a note: "You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details." Below this are "More Info" and "Ignore" buttons. On the right, there's a "System Configuration" sidebar with links to "System" (Configure global settings and paths), "Tools" (Configure tools, their locations and automatic installers), "Plugins" (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), and "Nodes" (Add, remove, control and monitor the various nodes that Jenkins runs jobs on).

3. Now add a plugin from Available Plugins Tab
4. Add plugins such as
 - a. Eclipse Temurin installer
 - b. SonarQube Scanner
 - c. NodeJS

The screenshot shows the Jenkins Plugin Manager. The top navigation bar includes a search icon, a bell icon with 1 notification, a shield icon with 1 error, and a "Stan" status indicator. The main area has tabs for "Updates", "Available plugins" (which is selected), "Installed plugins", "Advanced settings", and "Download progress". A search bar at the top right contains the text "Eclipse Temurin Installer". Below the search bar is an "Install" button. The main content area displays a table for the "Eclipse Temurin installer 1.7" plugin, showing its name, description ("Provides an installer for the JDK tool that downloads the Eclipse Temurin™ build based upon OpenJDK from the [Adoptium Working Group](#)."), release date ("2 mo 17 days ago"), and a "Released" status column.

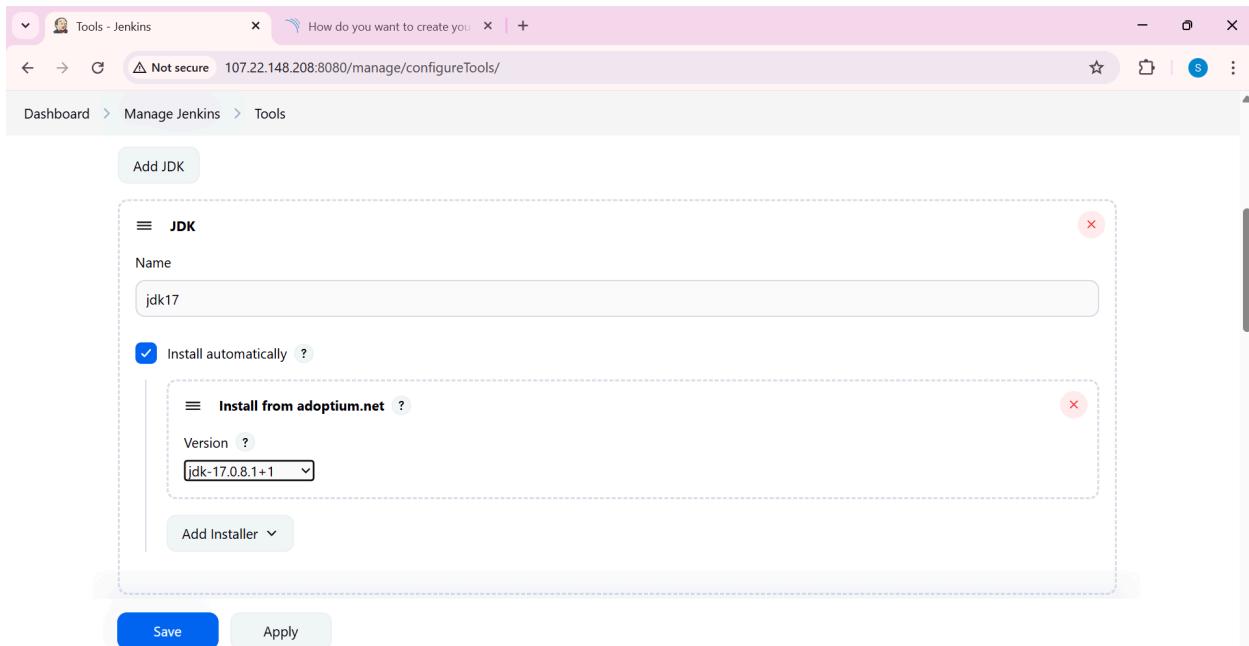
The screenshot shows the Jenkins Plugin Manager interface. The URL is 107.22.148.208:8080/manage/pluginManager/available. The search bar contains "SonarQube Scanner". There are two listed plugins: "Eclipse Temurin installer 1.7" and "SonarQube Scanner 2.18". Both have a checked checkbox next to them. The "SonarQube Scanner" plugin has a sub-section titled "External Site/Tool Integrations" and "Build Reports". The "Install" button is visible at the top right.

This screenshot is similar to the previous one, showing the Jenkins Plugin Manager interface. The search bar now contains "NodeJS". The "SonarQube Scanner" plugin is still listed with its checkboxes checked. A new plugin, "NodeJS 1.6.4", is now listed with a checked checkbox. This plugin is associated with "npm". The "Install" button is again visible at the top right.

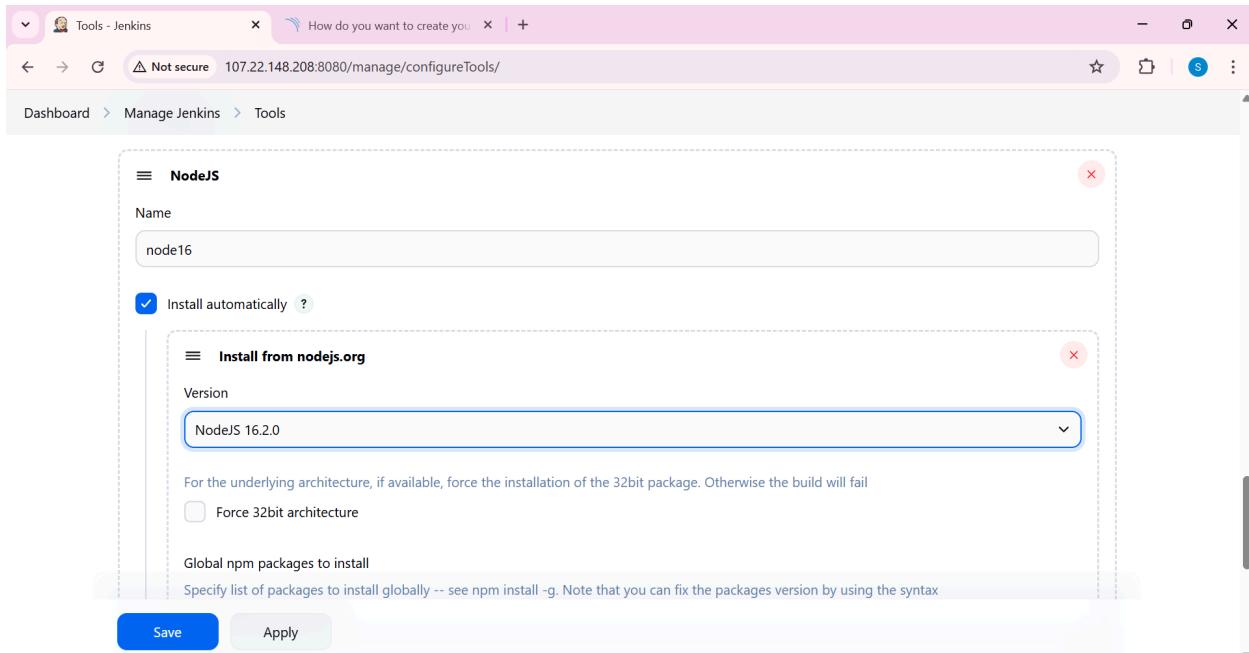
5. Now install them without restart and go back to Dashboard

3B : Configure Java and Nodejs in Global Tool Configuration

1. Now open Jenkins Dashboard and click Manage Jenkins
2. Choose Tools and search for JDK and NodeJs installation
3. Click Add JDK and name as jdk17
4. Choose Install Automatically and select version



5. Now click add NodeJS
6. Add name as node16 and choose Install Automatically
7. Select the desired version



8. Click Apply and Save

3C : Create a Job

1. In Jenkins Dashboard, click New Item
2. Name it as Zomato and choose Pipeline as Type

The screenshot shows the Jenkins 'New Item' creation dialog. At the top, there's a header bar with tabs for 'New Item - Jenkins' and 'How do you want to create your item'. Below the header, the URL is shown as 'Not secure 107.22.148.208:8080/view/all/newJob'. The main content area has a title 'New Item'. A field labeled 'Enter an item name' contains the value 'Zomato'. Below this, a section titled 'Select an item type' lists three options: 'Freestyle project', 'Pipeline' (which is highlighted with a black border), and 'Multi-configuration project'. At the bottom right of the dialog is a blue 'OK' button.

Step 4 — Configure Sonar Server in Manage Jenkins

1. Open SonarQube Dashboard using port 9000

The screenshot shows the SonarQube Administration interface. The top navigation bar includes tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration tab is selected. Below the navigation is a search bar labeled "Search for projects..." and a green button labeled "A". A banner at the top right indicates a new SonarQube version is available for upgrade. The main content area is titled "General Settings" and contains a "Find in Settings" search bar. On the left, a sidebar lists categories: Analysis Scope, Authentication, DevOps Platform Integrations, External Analyzers, and General. The "General" category is currently selected. The main content pane displays the "Cross project duplication detection" configuration, which is described as deprecated and detects duplicates at the project level. A toggle switch is shown, with "(default)" written below it.

2. Click Administrator in Navigation Bar
3. Within Security Tab, choose Users

The screenshot shows the SonarQube Administration interface with the Security tab selected from the top navigation bar. The navigation bar also includes tabs for Configuration, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration tab is highlighted. Below the navigation is a search bar labeled "Search for projects..." and a green button labeled "A". A banner at the top right indicates a new SonarQube version is available for upgrade. The main content area shows the "General" section of the Security tab, which includes options for Users, Groups, Global Permissions, and Permission Templates. A dropdown menu is open over the "General" tab, showing these four options. The "Users" option is currently selected.

4. Here Administrator will be available. Here click Update Token

The screenshot shows the SonarQube Administration - Users page. In the 'Tokens' section, there is a table with one row for the 'Administrator' user. The table columns are 'SCM Accounts', 'Last connection', 'Groups', and 'Tokens'. The 'Tokens' column shows 'sonar-administrators' and 'sonar-users' with a count of 0. A 'Create User' button is located at the top right of the 'Users' section. An 'Update Tokens' button is located at the bottom right of the 'Tokens' table.

5. Here add any Token name and set the Expiry date

The screenshot shows the SonarQube Administration - Tokens of Administrator page. It features a 'Generate Tokens' form with fields for 'Name' (containing 'Enter Token Name') and 'Expires in' (set to '30 days'). Below the form is a table titled 'Tokens of Administrator' with columns: 'Name', 'Type', 'Project', 'Last use', 'Created', and 'Expiration'. The table currently displays 'No tokens'. At the bottom right of the page is a 'Done' button.

6. We set it as For-Zomato with 30 days to expire

This screenshot shows the same 'Generate Tokens' form from the previous step, but with specific values entered: 'Name' is 'For-Zomato' and 'Expires in' is '30 days'. The 'Generate' button is visible at the bottom of the form.

7. Now copy paste the token generated somewhere safe

The screenshot shows the 'Tokens of Administrator' section in the SonarQube interface. A success message indicates a new token 'For-Zomato' has been created. The token value is displayed in a copyable field: `squ_fce64acd3eea48242b671c21f9b58b2e1e95fa1a`. A table lists the token details:

Name	Type	Project	Last use	Created	Expiration
For-Zomato	User		Never	May 14, 2025	June 13, 2025

A 'Revoke' button is available for the token. At the bottom right is a 'Done' button.

8. Now open the Jenkins Dashboard, and click Manage Plugins

The screenshot shows the Jenkins dashboard. The 'Manage Jenkins' link in the left sidebar is highlighted. The main area displays a table of build configurations:

S	W	Name ↓	Last Success	Last Failure	Last Duration
...	...	Zomato	N/A	N/A	N/A

Below the table, a note states 'No builds in the queue'.

9. Here we choose Credentials - System - Global Credentials

The screenshot shows the 'Manage Jenkins' page under the 'System' section. The 'Security' and 'Credentials' links are visible. The 'Security' section contains a brief description: 'Secure Jenkins; define who is allowed to access/use the system.'

Jenkins > Credentials - Jenkins > Users - Administration

Not secure 107.22.148.208:8080/manage/credentials/

Credentials

T	P	Store ↓	Domain	ID	Name
Stores scoped to Jenkins					
P	Store ↓	Domains			
	System	(global)			

System > Jenkins > Users - Administration

Not secure 107.22.148.208:8080/manage/credentials/store/system/

System

Domain ↓	Description
	Global credentials (unrestricted) <small>▼</small>
	Credentials that should be available irrespective of domain specification to requirements matching.

10. Here click Add Credentials

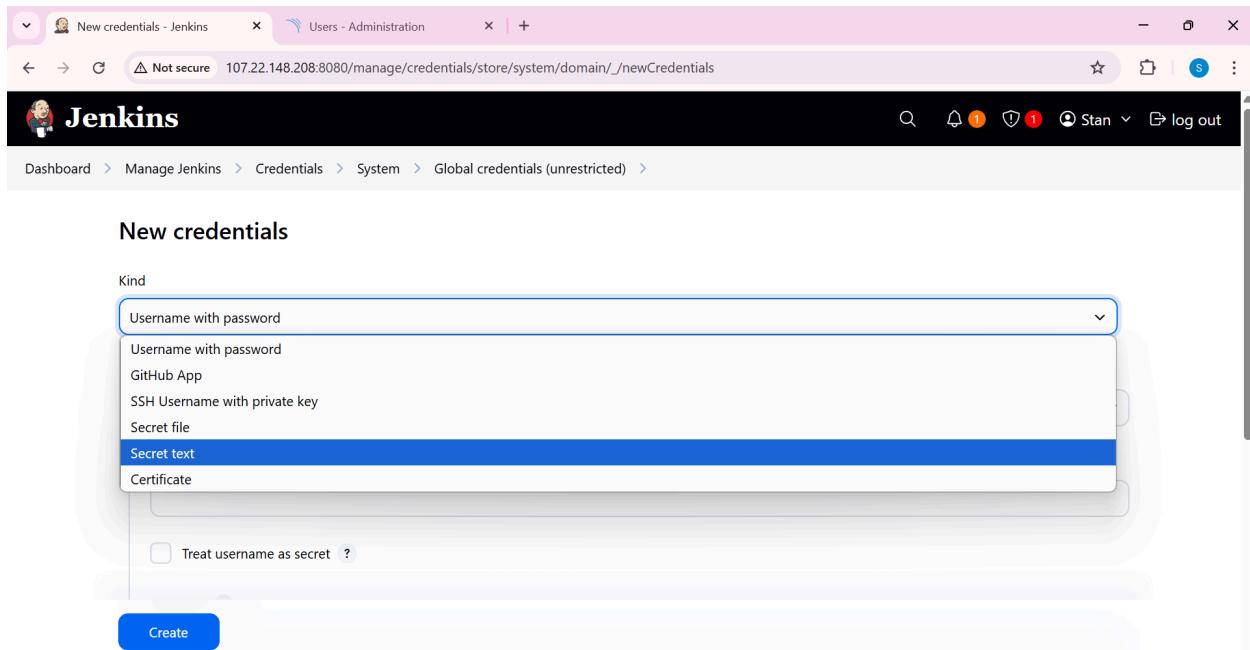
System > Global credentials (unrestricted) > Jenkins > Users - Administration

Not secure 107.22.148.208:8080/manage/credentials/store/system/domain/_/

Global credentials (unrestricted)

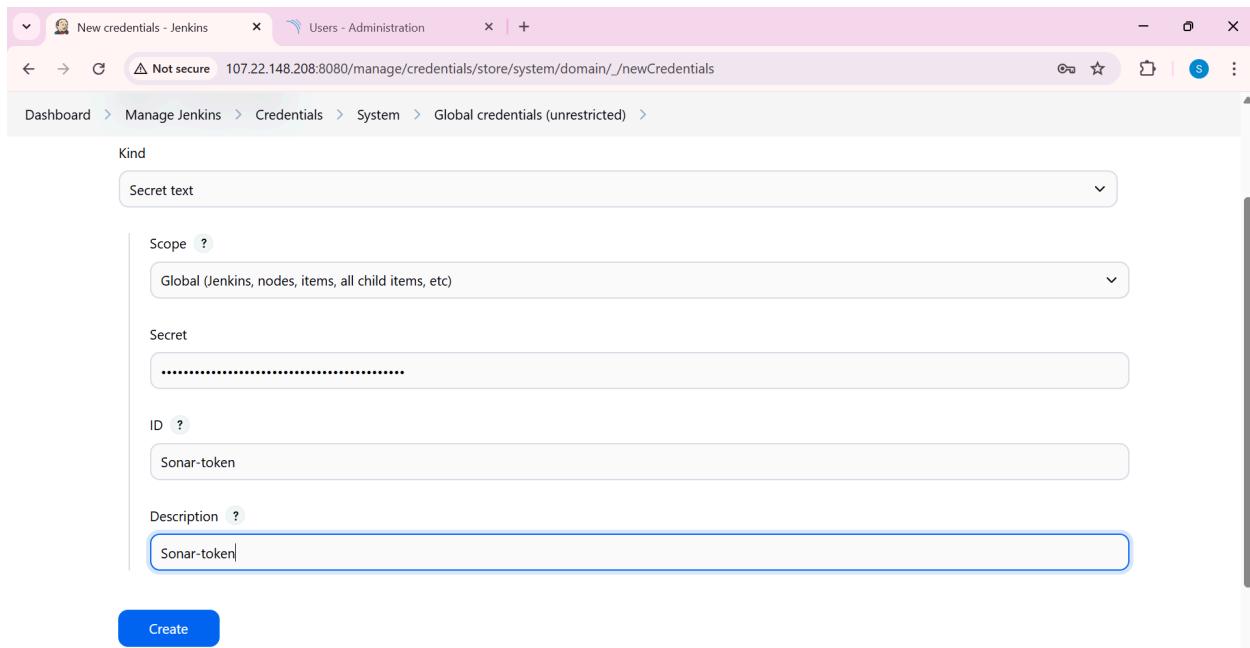
ID	Name	Kind	Description
This credential domain is empty. How about adding some credentials?			

11. Choose Secret Text



The screenshot shows the Jenkins 'New credentials' interface. In the 'Kind' dropdown, 'Secret text' is selected. Below the dropdown, there is a checkbox labeled 'Treat username as secret'. At the bottom is a blue 'Create' button.

12. Add the token we generated in SonarQube Administrator and add a name to it (here we use Sonar-token) and click Create



The screenshot shows the Jenkins 'New credentials' interface for 'Secret text'. The 'Scope' is set to 'Global'. The 'Secret' field contains '.....'. The 'ID' field is filled with 'Sonar-token'. The 'Description' field also contains 'Sonar-token'. At the bottom is a blue 'Create' button.

13. Now we see the Credentials available here

The screenshot shows the Jenkins Global credentials (unrestricted) page. At the top, there are tabs for System, Global credentials (unrestricted), and Users - Administration. The URL is 107.22.148.208:8080/manage/credentials/store/system/domain/_/. The page title is "Jenkins". Below the title, the breadcrumb navigation shows: Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted). A blue button labeled "+ Add Credentials" is at the top right. The main content area is titled "Global credentials (unrestricted)" and contains a table with one row. The table columns are ID, Name, Kind, and Description. The row shows an icon of a smartphone, the ID "Sonar-token", the name "Sonar-token", the kind "Secret text", and the description "Sonar-token". There is also a gear icon for configuration. Below the table, there are icons for S, M, and L.

14. Now go back to Jenkins Dashboard

The screenshot shows the Jenkins Dashboard. The top navigation bar includes "Build History", "Manage Jenkins" (which is highlighted in grey), and "My Views". A message box states: "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)". Buttons for "Set up agent", "Set up cloud", and "Dismiss" are available. Below this, there are sections for "Build Queue" (No builds in the queue) and "Build Executor Status" (0/2). A prominent orange message box says: "Java 17 end of life in Jenkins" and "You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to [the documentation](#) for more details". A "More Info" button and an "Ignore" button are provided. Under "System Configuration", there are links for "System" (Configure global settings and paths) and "Tools" (Configure tools, their locations and automatic installers).

15. Click Manage Jenkins, where we choose System

16. Inside search for SonarQube installations and click Add SonarQube

The screenshot shows the Jenkins Manage Jenkins > System page. The top navigation bar includes "System - Jenkins", "Users - Administration", and "Manage Jenkins". The URL is 107.22.148.208:8080/manage/configure. The breadcrumb navigation shows: Dashboard > Manage Jenkins > System. The main content area is titled "SonarQube servers". It includes a note: "If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build." with an unchecked checkbox for "Environment variables". The "SonarQube installations" section has a link "List of SonarQube installations" and a "Add SonarQube" button. The "Metrics" section includes "Access keys" with a question mark and a "Add new access key" button. At the bottom, there are "Save" and "Apply" buttons.

17. Name it as sonar-server and add Url ad <public-ip>:9000
18. Finally select the credentials, we created for SonarQube

The screenshot shows the Jenkins 'System' configuration page under 'Manage Jenkins > System'. A 'SonarQube installations' section is displayed. It contains a table with one row for 'sonar-server'. The row includes fields for 'Name' (sonar-server), 'Server URL' (http://107.22.148.208:9000), and 'Server authentication token' (Sonar-token). Buttons for '+ Add', 'Save', and 'Apply' are visible at the bottom.

Name	Server URL	Server authentication token
sonar-server	http://107.22.148.208:9000	Sonar-token

19. Click Apply and Save

20. Now the the Configure System option is used in Jenkins to configure different server
21. So we use Global Tool Configuration is used to configure different tools that we install using Plugins

The screenshot shows the Jenkins dashboard. The 'Manage Jenkins' link in the left sidebar is highlighted with a black rectangle. The main content area displays a table of build items, with one item named 'Zomato' shown in the first row. The table columns are: S, W, Name (with a dropdown arrow), Last Success, Last Failure, and Last Duration.

S	W	Name	Last Success	Last Failure	Last Duration
...	...	Zomato	N/A	N/A	>

22. Choose Tools in Manage Jenkins
23. Search for SonarQube Scanner Installations and click add SonarQube Server

SonarScanner for MSBuild installations

Add SonarScanner for MSBuild

SonarQube Scanner installations

Add SonarQube Scanner

Ant installations

Add Ant

Maven installations

Save Apply

24. Here add name as sonar-scanner and choose Install automatically

25. Select the desired version and click Apply and Save

SonarQube Scanner

Name
sonar-scanner

Install automatically

Install from Maven Central

Version
SonarQube Scanner 5.0.1.3006

Add Installer

Add SonarQube Scanner

Save Apply

26. Here we can create a Quality Gate or use the Default one

27. To create a Custom Quality Gate

- Open SonarQube Dashboard and choose Quality Gates

The screenshot shows the SonarQube Quality Gates page. At the top, there is a banner indicating a new version of SonarQube is available for upgrade. The main navigation bar includes links for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates (which is highlighted in blue), and Administration. Below the navigation, there is a section for 'Quality Gates' with a 'Create' button. A status message says 'Sonar way BUILT-IN'. A green box at the bottom right states 'This quality gate complies with Clean as You Code'.

- Here click Create and see the Pop up

This screenshot is similar to the previous one, showing the SonarQube Quality Gates page. The 'Create' button is highlighted with a blue border. The status message 'Sonar way BUILT-IN' is visible, along with the green compliance box.

- Add name as For-Zomato and Click Save

The screenshot shows the 'Create Quality Gate' dialog box. It contains fields for 'Name *' (with 'For-Zomato' typed in) and a list of rules. The rules listed are: 'No new bugs', 'No new code smells', 'All new code is covered by tests', 'New code has limited technical debt', and 'New code has limited duplication'. There are 'Save' and 'Cancel' buttons at the bottom of the dialog.

d. We can see the Quality Gate created

The screenshot shows the SonarQube interface for managing quality gates. The top navigation bar includes links for Manage Jenkins, General Settings, and Quality Gates. The Quality Gates tab is selected. Below the navigation is a search bar and several action buttons: Rename, Copy, Set as Default, and Delete. On the left, there's a sidebar for 'Quality Gates' with a 'Create' button. The main content area displays a single quality gate for the 'For-Zomato' project, which is set to 'Sonar way' and is a 'DEFAULT BUILT-IN'. A green box highlights that it complies with 'Clean as You Code', listing several criteria such as no new bugs, vulnerabilities, or security hotspots, and limited technical debt and duplication. Below this, a 'Conditions' section shows a table for 'Conditions on New Code' with one row: Coverage is less than 80.0%.

e. But we move forward with Default Quality Gate

28. Now create Webhook by going to SonarQube Dashboard and choose Administration

The screenshot shows the SonarQube General Settings page under the Administration tab. The top navigation bar includes Manage Jenkins, General Settings, and Administration. The Administration tab is selected. The main content area is titled 'Administration' and contains sections for Configuration, Security, Projects, System, and Marketplace. Under 'General Settings', there is a note about editing global settings for the instance. A search bar at the bottom allows finding specific settings.

29. In Configuration Tab, by hovering we see the Webhooks option

This screenshot is identical to the previous one, showing the SonarQube General Settings page under the Administration tab. The difference is that the 'Configuration' tab is now selected in the top navigation bar. The rest of the interface remains the same, showing the 'General Settings' section and the 'Find in Settings' search bar.

30. Choose Webhook and click Create

The screenshot shows the SonarQube Administration interface under the Webhooks section. A message at the top indicates a new SonarQube version is available. The main content area displays a table with one row, showing a webhook named "jenkins" with the URL "http://107.22.148.208:8080/sonarqube-webhook/". The "Actions" column contains a "Edit" button.

31. In pop-up, add name as jenkins and add URL as http://<public-ip>:8080/sonarqube-webhook

The screenshot shows a modal dialog titled "Create Webhook" over the SonarQube interface. It has fields for "Name" (set to "jenkins") and "URL" (set to "http://107.22.148.208:8080/sonarqube-webhook/"). Below the URL field is a note about using a secret key for HMAC hex digests. At the bottom right of the dialog are "Create" and "Cancel" buttons.

32. Now click create and see the Webhook n Dashboard

The screenshot shows the SonarQube Administration interface under the Webhooks section. The newly created webhook "jenkins" is listed in the table. The "Actions" column for this entry includes a "Edit" button.

33. Now go back to Jenkins Dashboard, choose the Zomato Job

The screenshot shows the Jenkins dashboard with the Zomato job selected. The job card for 'Zomato' is displayed, showing 'N/A' for Last Success, Last Failure, and Last Duration. There are buttons for 'Edit' and 'Delete' at the top right of the card.

34. Inside Zomato, click Configure Now and search for Pipeline (or choose it)

The screenshot shows the 'Configuration' page for the Zomato job. The 'Pipeline' tab is selected. The 'Definition' dropdown is set to 'Pipeline script'. A code editor window shows a Groovy pipeline script:

```
1~ pipeline{
2~   agent any
3~   tools{
4~     jdk 'jdk17'
5~     nodejs 'node16'
6~   }
7~   environment {
8~     SCANNER_HOME=tool 'sonar-scanner'
9~   }
10~  stages {
11~    stage('clean workspace'){
12~      steps{
13~        cleanWs()
14~      }
15~    }
16~  }
17~}
```

35. Add the pipeline script and press Apply and Save

The screenshot shows the 'Configuration' page for the Zomato job. The 'Pipeline' tab is selected. The 'Definition' dropdown is set to 'Pipeline script'. The code editor window now contains the following Groovy pipeline script:

```
1~ pipeline{
2~   agent any
3~   tools{
4~     jdk 'jdk17'
5~     nodejs 'node16'
6~   }
7~   environment {
8~     SCANNER_HOME=tool 'sonar-scanner'
9~   }
10~  stages {
11~    stage('clean workspace'){
12~      steps{
13~        cleanWs()
14~      }
15~    }
16~  }
17~}
```

The screenshot shows the Jenkins Pipeline configuration page for the 'Zomato' job. The pipeline script is displayed in a code editor:

```

14      }
15    }
16    stage('Checkout from Git'){
17      steps{
18        git branch: 'main', url: 'https://github.com/stanleycprime/Zomato-Clone.git'
19      }
20    }
21    stage("Sonarqube Analysis"){
22      steps{
23        withSonarQubeEnv('sonar-server') {
24          sh """$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=zomato \
25              -Dsonar.projectKey=zomato """
26        }
27      }
28    }
29    stage("quality gate"){
30      steps {
31        script {
32          waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
33        }
34      }
35    }
36    stage('Install Dependencies') {
37      steps {
38        sh "npm install"
39      }
40    }

```

Below the code editor are two buttons: 'Save' and 'Apply'.

36. Now click Build Now (inside Zomato Job)

The screenshot shows the Jenkins dashboard for the 'Zomato' job. The sidebar menu includes options like Status, Changes, Build Now (which is highlighted with a red border), Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The main content area displays the 'Zomato' pipeline configuration.

37. Wait for the building

The screenshot shows the Jenkins dashboard for the 'Zomato' job. The sidebar menu includes Status, Changes, Build Now (which is now unhighlighted), Configure, Delete Pipeline, Stages, Rename, and Pipeline Syntax. The main content area displays the 'Zomato' pipeline configuration. A message at the bottom indicates the 'Last build (#1), 17 sec ago'.

38. After the completion of Build, we see the success message

The screenshot shows the Jenkins interface for the 'Zomato' pipeline. On the left, there's a sidebar with options like Status, Changes, Build Now, Configure, Delete Pipeline, SonarQube, Stages, Rename, and Pipeline Syntax. The main area is titled 'SonarQube Quality Gate' and shows a green checkmark next to 'zomato' with the status 'Passed'. It also indicates 'server-side processing: Success'. Below this, there's a section for 'Permalinks' listing recent builds. At the bottom, there's a 'Builds' section with a search bar and a 'Filter' button.

39. But to embed the Pipeline Stage View, do this

- In Manage Jenkins, go to Plugins

The screenshot shows the 'Manage Jenkins' page. In the sidebar, 'Manage Jenkins' is selected. The main area has sections for 'Java 17 end of life in Jenkins' (warning about support ending on Mar 31, 2026) and 'System Configuration' (with links to 'System', 'Tools', and 'Plugins'). The 'Plugins' section is highlighted, showing a search bar with 'pipeline stage' and a button to 'Install'.

- Search for Pipeline Stage View and Install it

The screenshot shows the 'Available plugins' page in the Jenkins plugin manager. The search bar contains 'pipeline stage'. A table lists the 'Pipeline: Stage View' plugin by 'User Interface', version 2.38, released 14 days ago. The 'Install' button is visible.

Install	Name	Released
<input checked="" type="checkbox"/>	Pipeline: Stage View 2.38	14 days ago

c. Now wait for the completion and go back to Dashboard

The screenshot shows the Jenkins Plugins page. On the left, there's a sidebar with links: Updates, Available plugins, Installed plugins, Advanced settings, and Download progress (which is selected). The main area is titled "Download progress" and shows the following steps:

- Preparation:**
 - Checking internet connectivity
 - Checking update center connectivity
 - Success
- Pipeline: REST API:** Success
- Pipeline: Stage View:** Success
- Loading plugin extensions:** Success

Below these steps are two buttons:

- Go back to the top page (you can start using the installed plugins right away)
- Restart Jenkins when installation is complete and no jobs are running

d. Now click Build Now for Zomato Job

The screenshot shows the Jenkins dashboard. In the center, there's a table for the "Build Queue". One row for the "Zomato" job is highlighted. The columns are labeled: S, W, Name, Last Success, Last Failure, and Last Duration.

S	W	Name	Last Success	Last Failure	Last Duration
(green)	(yellow)	Zomato	4 min 27 sec #1	N/A	1 min 36 sec

A context menu is open over the Zomato row, with the "Build Now" option highlighted. Other options in the menu include "Changes", "Configure", and "Delete Pipeline".

e. We see the Pipeline Stage View (stage wise) inside the Job

The screenshot shows the Pipeline Stage View for the "Zomato" job. On the left, there's a sidebar with links: Status (which is selected), Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax.

The main area is titled "Stage View" and displays a table of pipeline stages. The columns are:

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies
9s	467ms	2s	23s	817ms	36s

Below this table, there are two rows of stage details:

- #3 May 14 12:52 No Changes: 285ms, 457ms, 1s, 19s
- #2 May 14 12:52 No Changes: 241ms, 477ms, 1s, 27s

A message at the bottom states: "Average stage times: (full run time: ~1min 36s)"

40. Now we see the Build is completed and the Quality Gate is Success

The screenshot shows the Jenkins dashboard for the 'Zomato' pipeline. On the left, there's a sidebar with options like 'Configure', 'Delete Pipeline', 'Full Stage View', 'SonarQube', 'Stages', 'Rename', and 'Pipeline Syntax'. Below that is a 'Builds' section with three recent builds (#3, #2, #1) each showing a green status icon and a progress bar. To the right is a summary table with columns: Declarative: Tool Install (9s), clean workspace (467ms), Checkout from Git (2s), Sonarqube Analysis (26s), quality gate (773ms), and Install Dependencies (28s). Below the table is a 'SonarQube Quality Gate' section showing 'zomato' in a green 'Passed' state with 'server-side processing: Success'. At the bottom, there are 'Permalinks' for each build.

41. In SonarQube Dashboard, go to Projects and see the report

The screenshot shows the SonarQube 'Projects' dashboard. It features a search bar and a 'Create Project' button. On the left, there are filters for 'Quality Gate' (Passed, Failed) and 'Reliability' (A rating, B rating). The main area displays the 'zomato' project, which is marked as 'Passed'. It provides a summary of code quality metrics: Bugs (1 C), Vulnerabilities (0 A), Hotspots Reviewed (0.0% E), Code Smells (0 A), Coverage (0.0%), Duplications (0.0%), and Lines (1.3k S, CSS, Java). The last analysis was 1 minute ago.

42. Choose the Project - Zomato, for further details

The screenshot shows the SonarQube 'Project' dashboard for 'zomato'. It includes tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity. The 'Overview' tab is selected and shows a large green box stating 'Passed' with the subtext 'All conditions passed.' To the right, there are sections for 'New Code' (Since May 14, 2025, Started 6 minutes ago), 'Overall Code' (0 New Bugs), and 'Reliability' (A). Below these are sections for 'New Vulnerabilities' (0 New Vulnerabilities) and 'Security' (A).

Step 5 : Install OWASP Dependency Check Plugins

43. In Jenkins Dashboard, go to Manage Jenkins

The screenshot shows the Jenkins Manage Jenkins dashboard. At the top, there are links for 'New Item', 'Build History', 'Manage Jenkins' (which is highlighted), and 'My Views'. A message box at the top right says: 'Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.' Below this are two buttons: 'Set up agent', 'Set up cloud', and 'Dismiss'. A yellow warning box states: 'Java 17 end of life in Jenkins' and 'You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to the documentation for more details.' It has 'More Info' and 'Ignore' buttons. The 'System Configuration' section includes links for 'System', 'Tools', 'Nodes', 'Clouds', and 'Plugins'. The 'Plugins' link is highlighted with a tooltip: 'Add, remove, disable or enable plugins that can extend the functionality of Jenkins.' The URL in the address bar is 107.22.148.208:8080/manage/pluginManager.

44. Choose Plugins and install OWASP Dependency Check

The screenshot shows the Jenkins Available plugins page. The left sidebar has links for 'Updates', 'Available plugins' (which is highlighted), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area shows a search bar with 'Owasp de' and an 'Install' button. A table lists available plugins: 'OWASP Dependency-Check' (version 5.6.1, released 23 days ago) and 'OWASP Dependency-Track' (version 6.0.1, released 1 mo 21 days ago). Both descriptions mention their use for dependency analysis and visualization. The URL in the address bar is 107.22.148.208:8080/manage/pluginManager/available.

45. Wait for installation to complete and go back to Dashboard

The screenshot shows the Jenkins 'Plugins' section. On the left, there's a sidebar with links: 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress'. The 'Download progress' link is highlighted with a grey background. On the right, under 'Download progress', it says 'Preparation' with three items: 'Checking internet connectivity', 'Checking update center connectivity', and 'Success'. Below that, under 'Pipeline: REST API', there are two items: 'Success' and 'Success'. Under 'Loading plugin extensions', there are two items: 'Success' and 'Success'. Under 'OWASP Dependency-Check', there is one item: 'Success'. At the bottom, there are two buttons: 'Go back to the top page' (with a note '(you can start using the installed plugins right away)') and 'Restart Jenkins when installation is complete and no jobs are running'.

46. In Manage Jenkins, choose Tools

The screenshot shows the Jenkins 'Manage Jenkins' page. On the left, there are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2). In the center, there's a prominent orange box titled 'Java 17 end of life in Jenkins' with the message: 'You are running Jenkins on Java 17, support for which will end on or after Mar 31, 2026. Refer to the [documentation](#) for more details.' To the right of this box are 'More Info' and 'Ignore' buttons. Below this, there's a 'System Configuration' section with two main items: 'System' (Configure global settings and paths) and 'Tools' (Configure tools, their locations and automatic installers). There's also a 'Plugins' section (Add, remove, disable or enable plugins that can extend the functionality of Jenkins).

47. Search for Dependency Check Installations and click Add Dependency Check

The screenshot shows the Jenkins 'Tools' configuration page. It has sections for 'Maven installations' (with an 'Add Maven' button), 'NodeJS installations' (with a dropdown menu showing 'NodeJS installations' and 'Edited'), and 'Dependency-Check installations' (with an 'Add Dependency-Check' button). At the bottom, there are 'Save' and 'Apply' buttons.

48. Add name as DP-Check and choose Install automatically

49. Select the desired version and click Apply and Save

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 ▾

50. Now go back to Jenkins Dashboard and choose the Zomato Job

51. Click Configure and choose Pipeline option (or scroll down)

Jenkins

Dashboard > Zomato > Configuration

Configure General Enabled

General Triggers Pipeline Advanced

Description

Plain text Preview

```
stage('OWASP FS SCAN') {  
    steps {  
        dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odc  
        dependencyCheckPublisher pattern: '**/dependency-check-report.xml'  
    }  
}  
stage('TRIVY FS SCAN') {  
    steps {  
        sh "trivy fs . > trivyfs.txt"  
    }  
}
```

52. Here add code in Pipeline Script to implement OWASP Scan

Dashboard > Zomato > Configuration

Configure General Enabled

Definition: Pipeline script

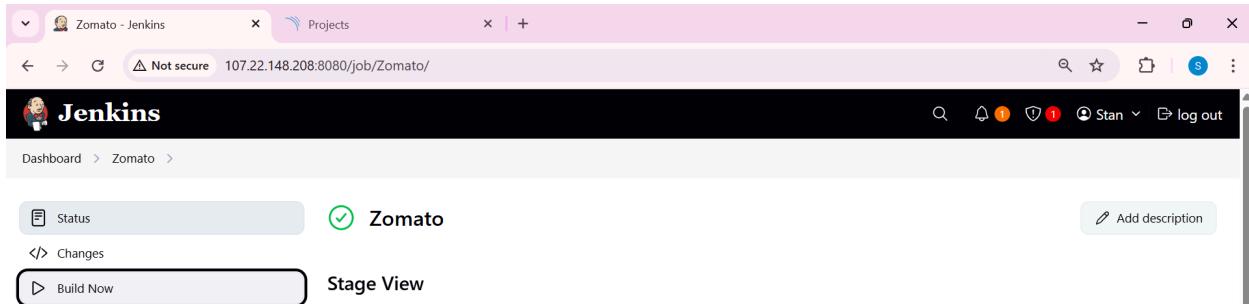
Script ?

```
stage('OWASP FS SCAN') {  
    steps {  
        dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odc  
        dependencyCheckPublisher pattern: '**/dependency-check-report.xml'  
    }  
}  
stage('TRIVY FS SCAN') {  
    steps {  
        sh "trivy fs . > trivyfs.txt"  
    }  
}
```

Use Groovy Sandbox ?

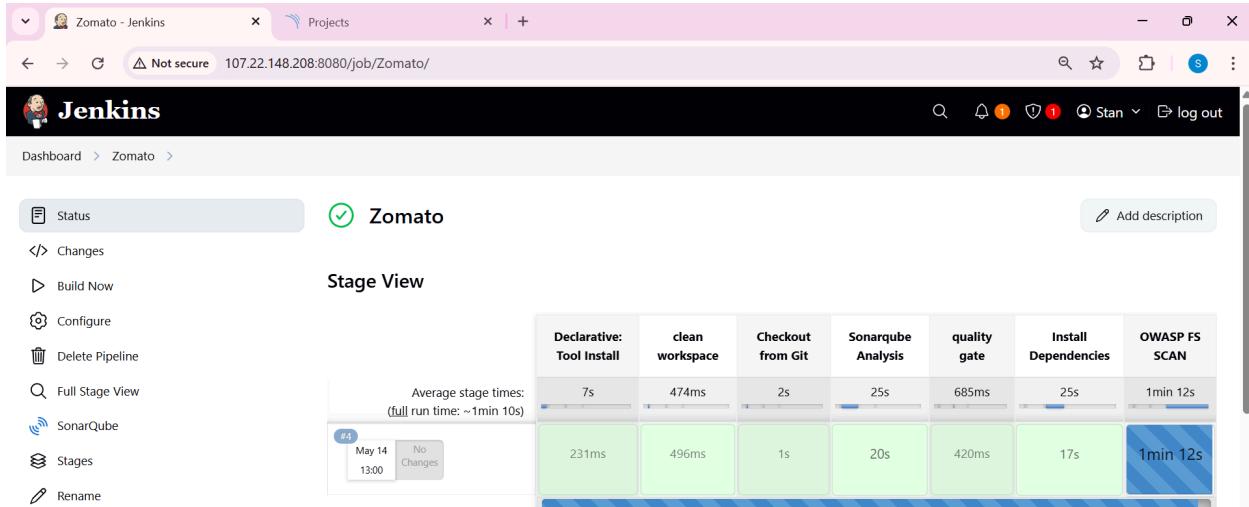
Save Apply

53. Now choose the Zomato Job and click Build Now



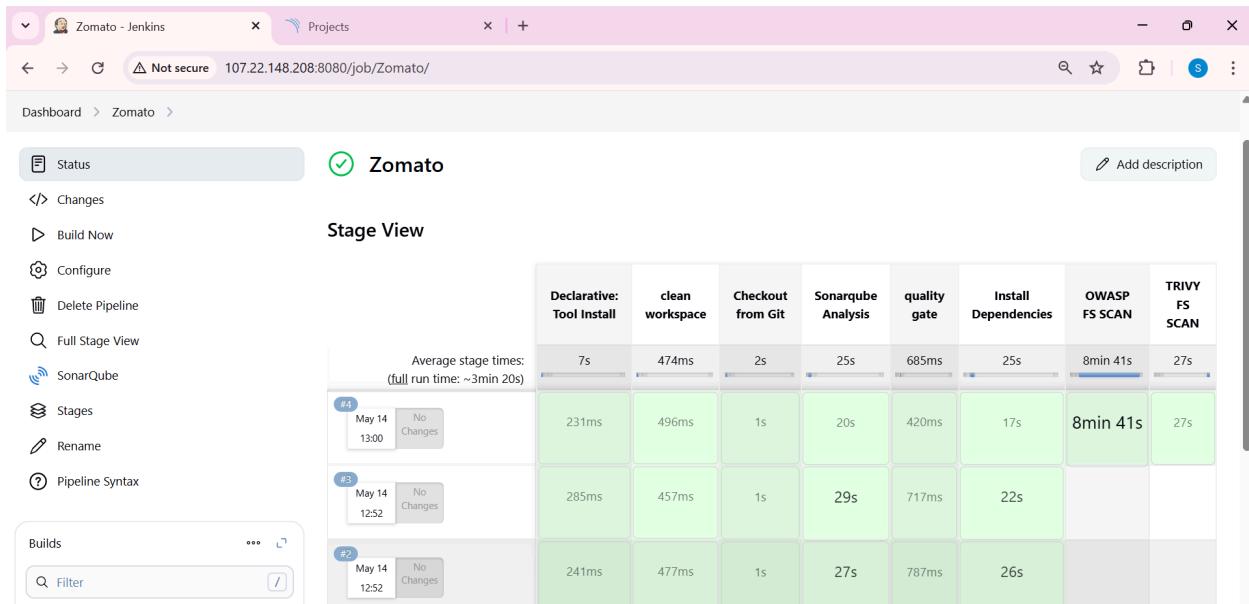
The screenshot shows the Jenkins Pipeline View for the 'Zomato' job. The top navigation bar includes links for 'Dashboard', 'Projects', and 'Not secure'. The main area displays the pipeline stages: 'Status' (green), 'Changes' (grey), 'Build Now' (highlighted in red), 'Configure', 'Delete Pipeline', 'Full Stage View', 'SonarQube', 'Stages', and 'Rename'. The 'Stage View' section shows the pipeline stages: Declarative: Tool Install (7s), clean workspace (474ms), Checkout from Git (2s), Sonarqube Analysis (25s), quality gate (685ms), Install Dependencies (25s), and OWASP FS SCAN (1min 12s). A summary at the bottom indicates an average stage time of 7s and a full run time of ~1min 10s.

54. Wait for the Build to be completed while viewing the stage progress in Pipeline View



The screenshot shows the Jenkins Pipeline View for the 'Zomato' job after the build has completed. The 'Build Now' button is no longer highlighted. The pipeline stages are now fully green: Declarative: Tool Install (7s), clean workspace (474ms), Checkout from Git (2s), Sonarqube Analysis (25s), quality gate (685ms), Install Dependencies (25s), and OWASP FS SCAN (1min 12s). The summary at the bottom indicates an average stage time of 7s and a full run time of ~1min 10s.

55. Here we see the build is completed



The screenshot shows the Jenkins Pipeline View for the 'Zomato' job, displaying multiple completed builds. The 'Build Now' button is no longer highlighted. The pipeline stages are shown for three builds: #4 (May 14 13:00), #3 (May 14 12:52), and #2 (May 14 12:52). Each build row includes a 'No Changes' indicator. The stages for each build are: Declarative: Tool Install (7s), clean workspace (474ms), Checkout from Git (2s), Sonarqube Analysis (25s), quality gate (685ms), Install Dependencies (25s), OWASP FS SCAN (8min 41s), and TRIVY FS SCAN (27s). The summary at the bottom indicates an average stage time of 7s and a full run time of ~3min 20s.

The screenshot shows the Jenkins Zomato job dashboard. On the left, there's a sidebar with build history for #4, #3, #2, and #1. The main area displays a SonarQube Quality Gate summary for build #4. It includes a table with metrics like duration (241ms, 477ms, 1s, 27s, 787ms, 26s) and a green bar indicating success. Below this is a "Latest Dependency-Check" section with a green "Success" status.

56. Here click the Latest Dependency Check option which shows the Results
(Also available in Graph based manner)

The screenshot shows the Jenkins Dependency-Check Report for build #4. The left sidebar has options like Status, Changes, Console Output, and Dependency-Check (which is selected). The main area is titled "Dependency-Check Results" and contains a "SEVERITY DISTRIBUTION" chart and a detailed table of vulnerabilities. The table columns include File Name, Vulnerability, Severity, and Weakness. Most vulnerabilities are marked as OSSINDEX and have a medium severity level.

File Name	Vulnerability	Severity	Weakness
+ async:3.2.4	OSSINDEX CVE-2024-39249	Medium	CWE-1333
+ body-parser:1.20.1	OSSINDEX CVE-2024-45590	High	CWE-405
+ braces:3.0.2	OSSINDEX CVE-2024-4068	High	CWE-1050
+ cookie:0.5.0	OSSINDEX CVE-2024-47764	Medium	CWE-74
+ cross-spawn:7.0.3	OSSINDEX CVE-2024-21538	High	CWE-1333
+ css-what:3.4.2	OSSINDEX CVE-2022-21222	High	CWE-1333
+ ejss:3.1.8	OSSINDEX CVE-2024-33883	High	CWE-693
+ express:4.18.2	OSSINDEX CVE-2024-10491	Medium	CWE-74
+ express:4.18.2	OSSINDEX CVE-2024-29041	Medium	CWE-1286
+ express:4.18.2	OSSINDEX CVE-2024-43796	Medium	CWE-79

57. It shows Vulnerability and Severity with other details

Step 6 : Docker Image Build and Push

1. In Jenkins Dashboard, choose Plugins in Manage Jenkins
2. Add various plugins such as
 - a. Docker
 - b. Docker Commons
 - c. Docker Pipeline
 - d. Docker API
 - e. docker-build-step

The screenshot shows the Jenkins Manage Jenkins > Plugins page. A search bar at the top contains the text 'docker'. Below the search bar, there are two sections of plugin results:

- Install** section:
 - Docker** (1274.vc0203fdf2e74) - Released 2 mo 7 days ago. This plugin integrates Jenkins with Docker.
 - Docker Commons** (451.vd12c371eeeb_3) - Released 2 mo 4 days ago. Provides the common shared functionality for various Docker-related plugins.
 - Docker Pipeline** (611.v16e84da_6d3ff) - Released 2 mo 10 days ago. Build and use Docker containers from pipelines.
 - Docker API** (3.5.0-108.v211cdd21c383) - Released 1 mo 12 days ago. This plugin provides docker-java API for other plugins.
- Available plugins** section:
 - docker-build-step** (2.12) - Released 11 mo ago. This plugin allows to add various docker commands to your job as build steps. A warning message is displayed: "Warning: This plugin version may not be safe to use. Please review the following security notices:
 - CSRF vulnerability and missing permission check"

3. Install them and go back to dashboard (once completed, without restart)

The screenshot shows the Jenkins plugin manager interface. On the left, there's a sidebar with links like 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress'. The 'Download progress' link is highlighted. The main area lists various Jenkins plugins with their status: Docker Commons (Success), Apache HttpComponents Client 5.x API (Success), Commons Compress API (Success), Docker API (Success), Docker (Success), Docker Commons (Success), Docker Pipeline (Success), Docker API (Success), Javadoc (Success), Dev Tools Symbols API (Success), jsoup API (Success), JSch dependency (Success), Maven Integration (Installing), docker-build-step (Pending), and Loading plugin extensions (Pending). A progress bar indicates the status of the Maven Integration download.

4. Now go to Tools in Manage Jenkins

The screenshot shows the Jenkins Manage Jenkins dashboard. In the center, there's a 'System Configuration' section with three main items: 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers, which is currently selected and highlighted in blue), and 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins).

5. Scroll and search for Docker Installations and click Add Docker

The screenshot shows the Jenkins Tools configuration page. Under the 'Docker' section, there are fields for 'Name' (with a red 'Required' error message) and 'Installation root'. There's also a checkbox for 'Install automatically'. At the bottom of the form is a 'Save' button, which is highlighted with a black border, and an 'Apply' button.

6. Add name as docker and click Install automatically while selecting the desired version

The screenshot shows the Jenkins 'Tools' configuration page under 'Docker installations'. A new Docker instance is being added with the name 'docker'. The 'Install automatically' checkbox is checked, and the 'Download from docker.com' section specifies 'latest' as the Docker version. There is also an 'Add Installer' dropdown. At the bottom are 'Save' and 'Apply' buttons.

7. Now choose Credentials in Manage Jenkins

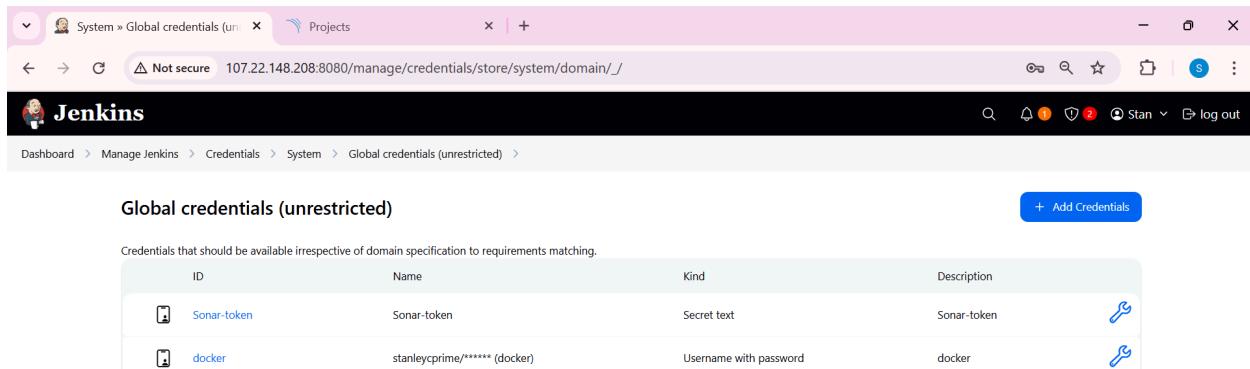
The screenshot shows the Jenkins 'Manage Jenkins' security page. The 'Credentials' link is highlighted, indicating it is the next step. Other options shown include 'Security' and 'Credential Providers'.

8. Choose Username with password and pass the username and password from DockerHub

9. Add ID and Description as Docker and click Create

The screenshot shows the Jenkins 'Global credentials (unrestricted)' creation page for a 'Username with password' credential. The fields filled are: Kind (Username with password), Scope (Global), Username (stanleycprime), Password (represented by dots), ID (docker), and Description (docker). The 'Create' button is at the bottom.

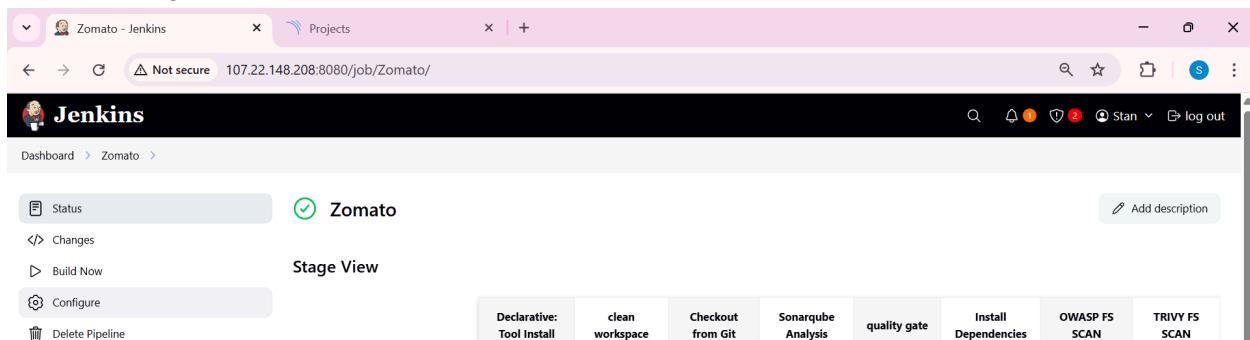
10. Now we see the Credentials we created in Global Dashboard



The screenshot shows the Jenkins Global credentials (unrestricted) page. It lists two entries:

ID	Name	Kind	Description
Sonar-token	Sonar-token	Secret text	Sonar-token
docker	stanleycprime***** (docker)	Username with password	docker

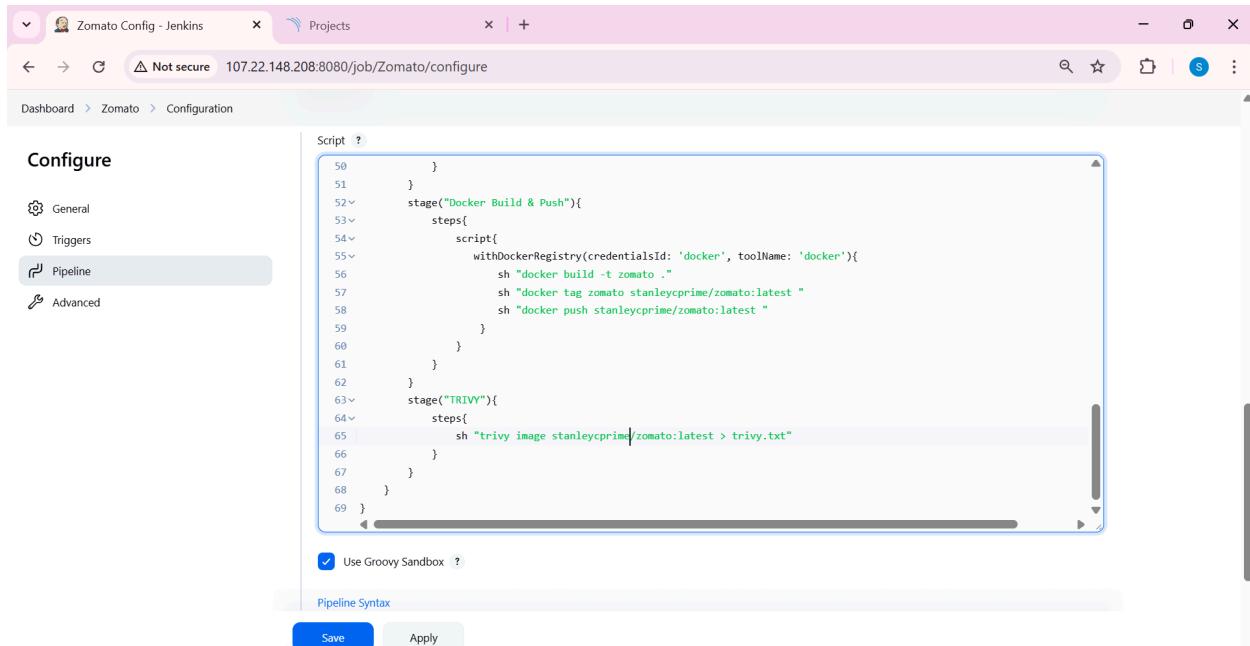
11. Now go back to Jenkins Dashboard and choose the Zomato Job



The screenshot shows the Jenkins Zomato job dashboard. The Stage View tab is selected. Pipeline stages shown include Declarative: Tool Install, clean workspace, Checkout from Git, Sonarqube Analysis, quality gate, Install Dependencies, OWASP FS SCAN, and TRIVY FS SCAN.

12. Choose Configure and go to Pipeline and add the code to implement Docker Build and Push stage

13. Now click Apply and Save



The screenshot shows the Jenkins Zomato Config - Jenkins pipeline configuration screen. The Pipeline tab is selected. The Groovy script in the Script editor is:

```
50
51
52    stage("Docker Build & Push"){
53        steps{
54            script{
55                withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
56                    sh "docker build -t zomato ."
57                    sh "docker tag zomato stanleycprime/zomato:latest"
58                    sh "docker push stanleycprime/zomato:latest"
59                }
60            }
61        }
62    }
63    stage("TRIVY"){
64        steps{
65            sh "trivy image stanleycprime/zomato:latest > trivy.txt"
66        }
67    }
68 }
69 }
```

At the bottom, there are Save and Apply buttons.

14. Click Build Now for the Zomato Job

15. We see the Pipeline progress here

The screenshots show the Jenkins Stage View for build #5. The top screenshot is for a successful build, and the bottom screenshot is for a failed build. Both screenshots show the following details:

- Build Now** button (highlighted in the top screenshot)
- Configure**, **Delete Pipeline**, **Full Stage View**, **SonarQube**, **Stages**, **Rename** buttons
- Stage View** section with a timeline showing the duration of each stage.
- Average stage times:** (full run time: ~3min 20s)
- Declarative: Tool Install**: 5s
- clean workspace**: 468ms
- Checkout from Git**: 1s
- Sonarqube Analysis**: 21s
- quality gate**: 685ms
- Install Dependencies**: 25s
- OWASP FS SCAN**: 8min 41s
- TRIVY FS SCAN**: 27s
- May 14 13:25 No Changes** (Build #5)
- May 14 13:00 No Changes** (Build #4)

In the bottom screenshot (failed build), the **Docker Build & Push** stage is highlighted in red with a 'failed' status, and the **TRIVY** stage is also highlighted in red with a 'failed' status.

16. Here we encounter an Error

- We see the Docker Build and Push stage got failed
- This is because we need jenkins to be installed within docker group

```

version: 0.62.1
ubuntu@ip-172-31-39-134:~$ sudo usermod -aG docker jenkins
ubuntu@ip-172-31-39-134:~$ ls
jenkins.sh trivy.sh
ubuntu@ip-172-31-39-134:~$ sudo systemctl restart jenkins
ubuntu@ip-172-31-39-134:~$ sudo systemctl status jenkins
sudo: systemctlstatus: command not found
ubuntu@ip-172-31-39-134:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Wed 2025-05-14 08:10:55 UTC; 13s ago
       Main PID: 19224 (java)
          Tasks: 50 (limit: 9505)
         Memory: 549.2M (peak: 552.7M)
            CPU: 20.543s
           CGroup: /system.slice/jenkins.service
                   └─19224 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

May 14 08:10:53 ip-172-31-39-134 jenkins[19224]: 2025-05-14 08:10:53.602+0000 [id=31]      INFO      h.p.b.g.GlobalTimeOutConfiguration#load: global ti
May 14 08:10:53 ip-172-31-39-134 jenkins[19224]: 2025-05-14 08:10:53.951+0000 [id=31]      WARNING    o.j.p.d.DockerBuildersDescriptorImpl#<init>: Do
May 14 08:10:54 ip-172-31-39-134 jenkins[19224]: 2025-05-14 08:10:54.821+0000 [id=29]      INFO      jenkins.initReactorRunner$1#onAttained: System con

```

17. Now rebuild the Job with the same Script

The screenshot shows the Jenkins interface for the Zomato job. On the left, there's a sidebar with various options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, and Pipeline Syntax. The main area is titled "Stage View" and shows a pipeline with several stages: Declarative: Tool Install (345ms), clean workspace (617ms), Checkout from Git (1s), Sonarqube Analysis (20s), quality gate (525ms), Install Dependencies (17s), Docker Build & Push (3min 31s), and TRIVY (47s). Above the stages, there's a chart titled "Dependency-Check Trend" with a legend for Unassigned (grey), Low (green), Medium (yellow), High (orange), and Critical (red). The chart shows a single red line at the zero mark for all stages.

18. Now we can see the build is complete and we can also see the Dependency check Trend Graph embedded during the Build Process

This screenshot shows the Jenkins interface for the Zomato job after a rebuild. The build number has changed to #6. The stages and their durations remain the same as in build #5. The "Dependency-Check Trend" chart now shows a green line at the zero mark, indicating that all dependencies have been successfully checked.

19. We can ensure this by going to Repositories in DockerHub, where we stanleycprime/zomato repository

The screenshot shows the Docker Hub interface. In the top left, the user's namespace 'stanleycprime' is selected. The main navigation bar has 'My Hub' highlighted. Below the navigation, there's a search bar and a 'Create a repository' button. On the left, a sidebar lists options like 'Repositories' (which is selected), 'Settings', 'Default privacy', 'Notifications', 'Billing', 'Usage', 'Pulls', and 'Storage'. The main content area is titled 'Repositories' and shows a single entry: 'stanleycprime/zomato' was last pushed 2 minutes ago and is a public image.

20. We open them, to see for further details

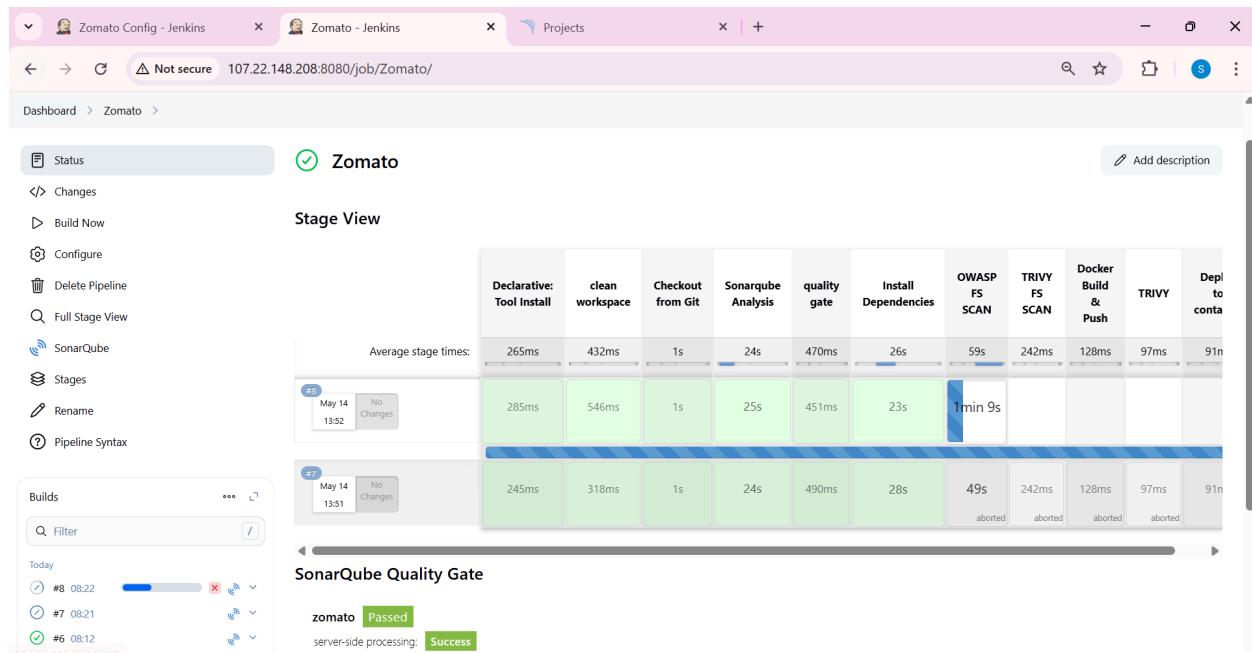
This screenshot shows the detailed view of the 'stanleycprime/zomato' repository. The left sidebar remains the same. The main page title is 'stanleycprime/zomato'. It shows the repository was last pushed about 1 hour ago and has a size of 303.2 MB. There are sections for 'Add a description' and 'Add a category'. The 'General' tab is selected, showing a table of tags. The table has columns for Tag, OS, Type, Pulled, and Pushed. One row is shown: 'latest' (OS: Image, Type: Image, Pulled: less than 1 day, Pushed: about 1 hour). A 'See all' link is at the bottom. To the right, there's a 'Docker commands' section with a 'Public view' button and a command line box containing 'docker push stanleycprime/zomato: tagname'. A 'buildcloud' sidebar is also visible.

Step 7 : Deploy the image using Docker

1. To implement the Deployment in pipeline script, we go back to Jenkins Dashboard and choose the Zomato Job
2. Choose Pipeline inside Configure Tab, where we insert the code to deploy to container stage

```
62      }
63  stage("TRIVY"){
64    steps{
65      sh "trivy image stanleycprime/zomato:latest > trivy.txt"
66    }
67  }
68 stage('Deploy to container'){
69   steps{
70     sh 'docker run -d --name zomato -p 3000:3000 stanleycprime/zomato:latest'
71   }
72 }
73
74 }
75 }
```

3. Now Build the Zomato Job and see the pipeline progress



4. Now we see the pipeline is completed

The screenshot shows the Jenkins Pipeline Summary for the 'Zomato' job. It displays a table of stages and their execution times. The stages are:

	Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container
Average stage times: (full run time: ~15min 37s)	265ms	432ms	1s	24s	470ms	26s	3min 44s	11s	2min 9s	1min 40s	492ms
y 14 :52	No Changes	285ms	546ms	1s	25s	451ms	23s	6min 39s	23s	4min 19s	3min 20s
y 14 :51	No Changes	245ms	318ms	1s	24s	490ms	28s	49s	aborted	242ms	128ms
							aborted	aborted	aborted	97ms	91ms
										aborted	aborted

5. Ensure them by opening the DockerHUb repositories

The screenshot shows the Docker Hub interface for the 'stanleycprime' namespace. The 'Repositories' section lists one repository: 'stanleycprime/zomato'. The repository details are as follows:

Name	Last Pushed	Contains	Visibility	Scout
stanleycprime/zomato	27 minutes ago	IMAGE	Public	Inactive

6. To open the Deployed application in Browser, we need to add port 3000 to Security Group. To do this:

- Choose the Security Group which we used for this instance
- Click Edit Inbound rules

Inbound security group rules successfully modified on security group (sg-028134b144e921da2 | launch-wizard-2)

Security Groups (1/3) Info

Inbound rules (5)

Name	Security group ID	Security group name	VPC ID	Description
-	sg-09e8335bf17db5269	launch-wizard-1	vpc-08e4043f9384e6fad	launch-wizard-1 created 2025-04-24T1...
selected	sg-028134b144e921da2	launch-wizard-2	vpc-08e4043f9384e6fad	launch-wizard-2 created 2025-05-14T0...
-	sg-03679d5c4583a92a8	default	vpc-08e4043f9384e6fad	default VPC security group

- Add port 3000 as Custom TCP with open subnet and click Save rules

Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

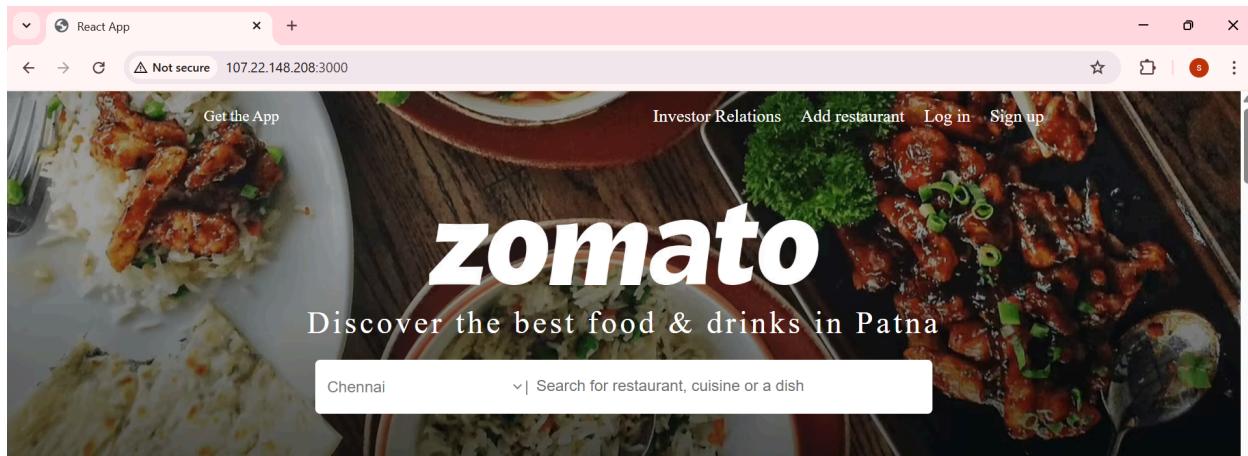
Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0fc8531c137be55ec	Custom TCP	TCP	9000	Custom	0.0.0.0/0
sgr-0eb97e181be1b62e9	Custom TCP	TCP	8080	Custom	0.0.0.0/0
sgr-0422b25317c19b141	SSH	TCP	22	Custom	0.0.0.0/0
sgr-0464ea799f150e492	HTTPS	TCP	443	Custom	0.0.0.0/0
sgr-0fd8a624d17478eb	HTTP	TCP	80	Custom	0.0.0.0/0
-	Custom TCP	TCP	3000	Anywhere	0.0.0.0/0

Add rule

Save rules

7. Now open the Application using
<http://<public-ip>:3000>



A screenshot of the same web browser window showing two featured sections. On the left, a section titled "Order Online" with the subtext "Stay home and order to your doorstep" is shown next to an image of a restaurant interior. On the right, a section titled "Nightlife and Clubs" with the subtext "Explore the city's top nightlife outlets" is shown next to an image of a night club scene with raised hands. The overall layout is identical to the first screenshot, with the Zomato logo and search bar at the top.



Popular localities in and around Ahmedabad

Bodakdev	>	Setellite	>
345 Places		336 Places	
Gurukul	>	Navrangpura	>
83 Places		302 Places	
Vastrapur	>	Thaltej	>
217 Places		222 Places	
Prahaldad Nagar	>	C G Road	>
181 Places		94 Places	
See more			

Get the Zomato app



We will send you a link, open it on your phone to download the app

Email Phone

Email

Share App Link

Download app from



Explore options near me

Top Restaurant Chains

India

English

Zomato

ABOUT ZOMATO

Who We Are

Blog

Work With Us

Investor Relations

Report Fraud

ZOMAVERSE

Zomato

Blinkit

Feeding India

HyperPure

Zomaland

FOR RESTAURANTS

Partner With Us

Apps For You

For Enterprises

Zomato For Work

LEARN MORE

Privecy

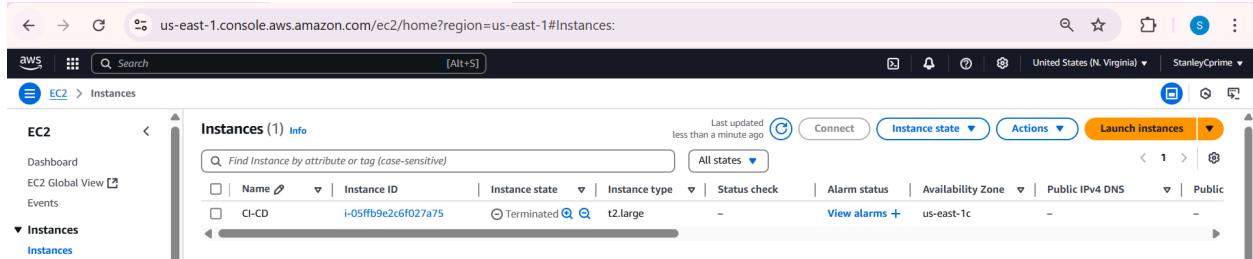
Security

Terms

Sitemap

Step 8 : Delete the Instances

1. At first glance, this step may be neglected
2. But the instances we use are heavy and huge
3. So it is mandatory to delete the instances as soon the Project is completed
4. Open EC2 in AWS service provider
5. Select the instance (or resource) you want to delete
6. Click Instance State and choose Terminate Instance
7. Wait for the status to be changed to Terminated



The screenshot shows the AWS EC2 Instances page. The URL is us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances. The page displays one instance named 'CI-CD' with the ID 'i-05ff9e2c6f027a75'. The instance state is 'Terminated' and the instance type is 't2.large'. The page includes filters for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, Public IPv4 DNS, and Public. There are also buttons for Connect, Instance state, Actions, and Launch instances.

Things we Achieved

End-to-End CI/CD Pipeline Setup:

- Automated the process of code checkout, build, test, scan, containerize, and deploy using Jenkins.

Secure DevSecOps Integration:

- Integrated security at every stage using **SonarQube**, **Trivy**, and **OWASP Dependency-Check** for code and container vulnerability analysis.

Containerization with Docker:

- Packaged the Node.js application into a portable Docker image, ensuring consistent deployment across environments.

DockerHub Image Push:

- Automatically pushed the application image to **DockerHub**, enabling image reuse and version control.

Jenkins Plugin Configuration:

- Installed and configured essential plugins (JDK, NodeJS, SonarQube Scanner, Docker, Dependency Check) for pipeline execution.

Custom Jenkins Declarative Pipeline:

- Defined all CI/CD steps in a version-controlled Jenkinsfile, allowing for easy updates and transparency.

Static Code Quality Checks:

- Enforced **SonarQube Quality Gates** to maintain high code standards and detect code smells, bugs, and security hotspots early.

Dependency Vulnerability Scanning:

- Used **OWASP Dependency Check** to scan third-party packages and libraries for known vulnerabilities.

Container Vulnerability Scanning:

- Leveraged **Trivy** to scan Docker images for OS-level and application-level security risks before deployment.

Real-Time Deployment:

- Deployed the Dockerized Zomato Clone application directly on an EC2 instance, accessible via browser on port 3000.

Cost Optimization:

- Terminated the EC2 instance after deployment testing to avoid unnecessary billing.
-

Conclusion

The Zomato Clone App deployment with a DevSecOps CI/CD pipeline is a comprehensive process that integrates various tools and services to automate the building, scanning, and deployment of a Node.js application. From launching an AWS EC2 instance to setting up Jenkins, Docker, SonarQube, Trivy, and OWASP Dependency Check, each step is crucial for ensuring the application is built with security and quality in mind.

Key stages in the setup include:

- **Instance Setup:** Launching an Ubuntu EC2 instance and connecting it via SSH using Putty, followed by configuring the necessary security settings and installing required services.
- **Tool Installations:** Installing Jenkins for automation, Docker for containerization, SonarQube for code quality checks, Trivy for vulnerability scanning, and OWASP Dependency Check for assessing application dependencies.
- **CI/CD Pipeline Configuration:** Setting up Jenkins pipelines to handle tasks such as code building, SonarQube analysis, vulnerability scanning, Docker image building, and deployment.
- **Automated Security Scanning:** Leveraging SonarQube and Trivy ensures that security flaws are detected and handled early in the development lifecycle.
- **Docker Image Management:** Building Docker images, pushing them to DockerHub, and deploying them in containers ensures a repeatable and scalable deployment process.
- **Cleaning up Resources:** Finally, after successfully deploying the application, terminating the EC2 instances helps avoid unnecessary costs.

This setup not only automates the entire process but also ensures that security and quality gates are integrated into every step, making the deployment process robust and reliable. The combination of Jenkins, SonarQube, Trivy, and Docker provides a modern DevSecOps approach that is well-suited for managing production-ready applications.
