Project Report

on

Infra SecOps

Submitted in partial fulfillment for the award of

Post Graduate Diploma in High Performance Computing System Administration from **C-DAC ACTS (Pune)**

Guided by:

Mr. Roshan Gami

Presented by:

- **Anushka Kale PRN:230340127031**
- **Kundan Pagare PRN:230340127056**
- **Vishakha Kharat PRN:230340127055**
- **Saurabh Singh PRN:230340127049**
- **Sonali Zol PRN:230340127051**

Centre of Development of Advanced Computing (C-DAC), Pune

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

- **Anushka Kale**
- **Kundan Pagare**
- **Vishakha Kharat**
- **Saurabh Singh**
- **Sonali Zol**

have successfully completed their project on

Infra SecOps

Under the Guidance of Mr. Roshan Gami

Project Guide **Project Supervisor**

HOD ACTS
Mr. Aditya Sinha

ACKNOWLEDGEMENT

This project “Infra SecOps” was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS). We all are very glad to mention the name of Mr. Roshan Gami for his Valuable guidance to work on this project. His guidance and support helped usto overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Mr. Kaushal Sharma (Manager (ACTS training Centre), C-DAC), for his guidance and support whenever necessary while doing this course Post Graduate Diploma in High Performance Computing System Administration (PG-DHPCSA) through C-DAC ACTS, Pune.

Our most heartfelt thank goes to Ms. Swati Salunkhe (Course Coordinator, PG-DHPCSA) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

From:

Anushka Kale	**(230340127031)**
Kundan Pagare	**(230340127056)**
Saurabh Singh	**(230340127049)**
Vishakha Kharat	**(230340127055)**
Sonali Zol	**(230340127051)**

TABLE OF CONTENTS

1. Abstract

The increasing number of cyber-attacks has made it essential for organizations to prioritize security in their software development process. This project aims to implement the Infrastructure as a code in DevSecOps to integrate security into the software development process.

The project will involve the following phases:

1. Planning and Analysis: In this phase, security requirements will be identified and analyzed, and security testing tools will be selected.
2. Design and Architecture: In this phase, security controls will be defined and integrated into the software design and architecture.
3. Implementation: In this phase, security controls will be implemented in the code.
4. Testing: In this phase, the software will be tested for security vulnerabilities, and any issues found willbe fixed.
5. Deployment: In this phase, the software will be deployed to production.
6. Maintenance: In this phase, the software will be continuously monitored and maintained to ensure thatit remains secure.

The project will also involve the following practices:

1. Automation of Security Testing: Automated security testing tools will be integrated into the DevSecOps pipeline to detect vulnerabilities early in the development process.

2. Continuous Integration and Deployment: Continuous integration and deployment practices will be implemented to ensure that the software is continuously tested and deployed in a secure manner.

The expected outcome of the project is to provide Infrastructure of a secure software development process that meets the security requirements of the organization. The project will help reduce the risk of security breaches and ensure that the software is developed and deployed securely.

2. Introduction and overview of project

This project aims to implement the Infrastructure framework in DevSecOps to provide a secure software development process that meets the security requirements of the organization. The project will involve the identification and analysis of security requirements, the integration of security controls into the software design and architecture, the implementation of security controls in the code, testing the software for security vulnerabilities, deployment of the software to production, and continuous monitoring and maintenance of the software to ensure that it remains secure. With the increasing number of cyber-attacks, security has become a major concern for organizations. The traditional approach of adding security measures as an afterthought to the software development process is no longer effective. Therefore, there is a need for a more proactive approach that integrates security into the software development lifecycle. The Secure Software Development Lifecycle (SSDLC) framework is a process that integrates security measures into the software development lifecycle. DevSecOps, on the other hand, is an approach that combines development, operations, and security teams to automate and integrate security into the software development process.

3. Implementing Infra SecOps

Infrastructure as a code is a framework for integrating security into the Application development process. SecOps is a methodology that combines security, and operations in to asingle continuous process. By implementing Infra in SecOps, you can ensure that security is built into every stage of the Application development process.

Here are the steps to implement Infra in SecOps:

1. Plan: In this stage, you need to define the project scope, objectives, and requirements. You also needto identify potential security risks and define security requirements.

2. Design: In this stage, you need to design the architecture, components, and interfaces of the Application. You also need to define security controls, such as access control, authentication, and encryption.

3. Develop: In this stage, you need to develop the Application system and ensure that it meets the security requirements defined in the previous stages. You also need to conduct security testing, such as penetration testing and vulnerability scanning.

4. Deploy: In this stage, you need to deploy the application server in a secure manner. You also need to ensure that the deployment environment meets the security requirements.

5. Maintain: In this stage, you need to maintain the Application by fixing security vulnerabilities andupdating security controls. You also need to ensure that the Application Server remains secure over time.

4. System RequirementUser Interface

1.EC2 Instance Ubuntu

4.1 Software Requirement

1. GIT
2. Jenkins
3. Docker
4. OWASP ZAP
5. Apache Maven
6. Trufflehog3
7. WebGoat
8. DAST

4.2 Hardware Requirement

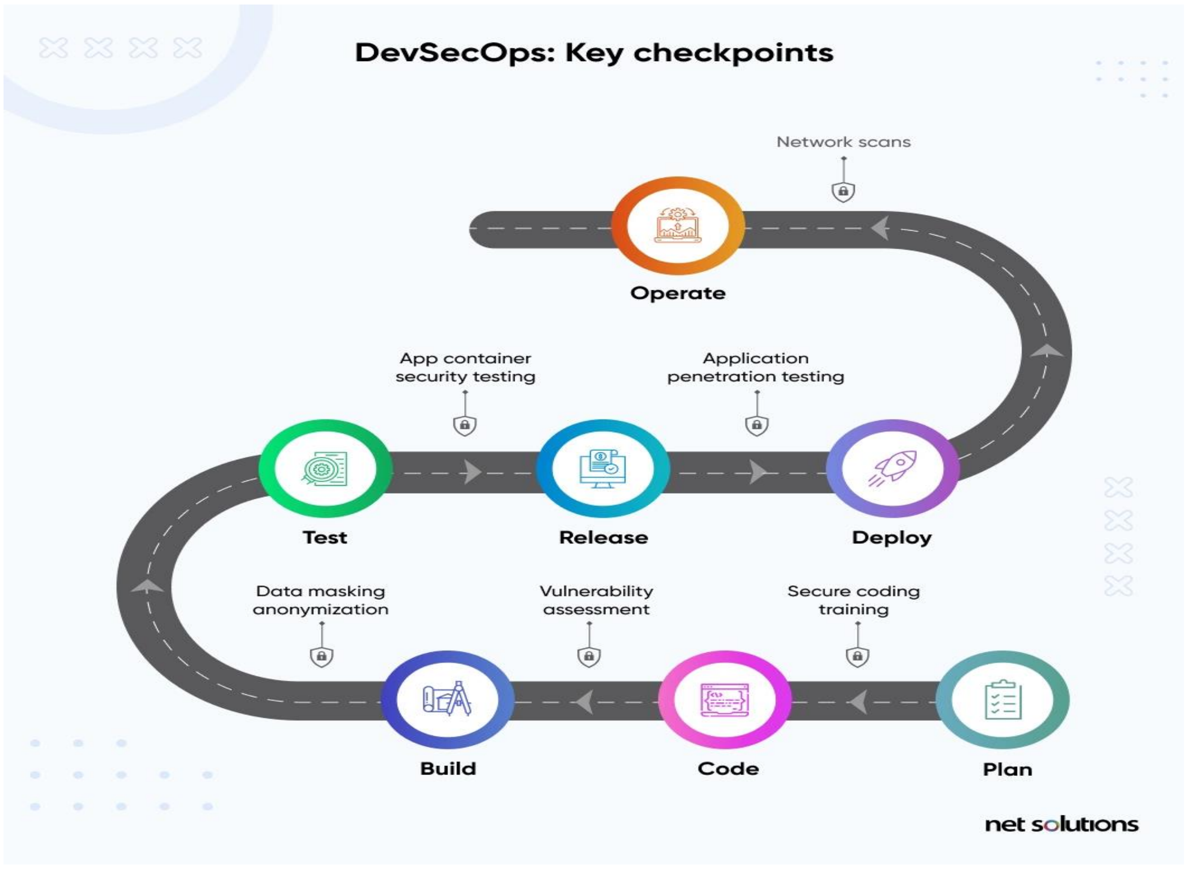
1. Windows 10 or 11
2. Ubuntu 30GB HD, 1GB RAM
3. t2-medium 4GB RAM
4. t2-micro 1GB RAM

4.3 Used Language

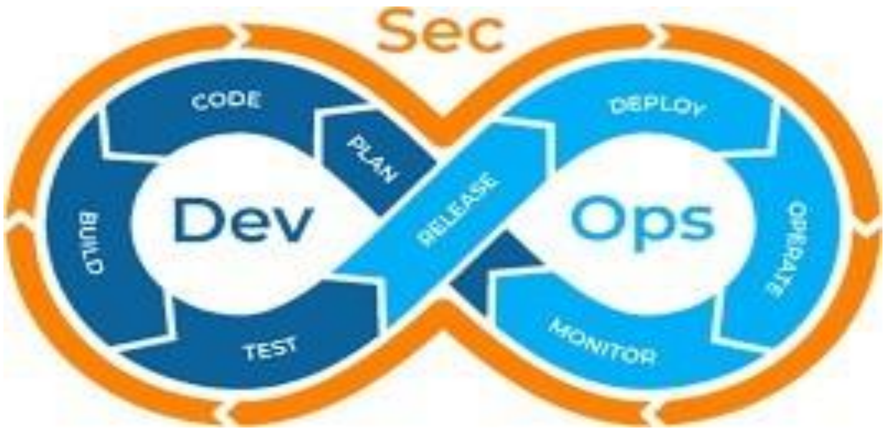
1. Python script
2. Yamal script

5. System Architecture

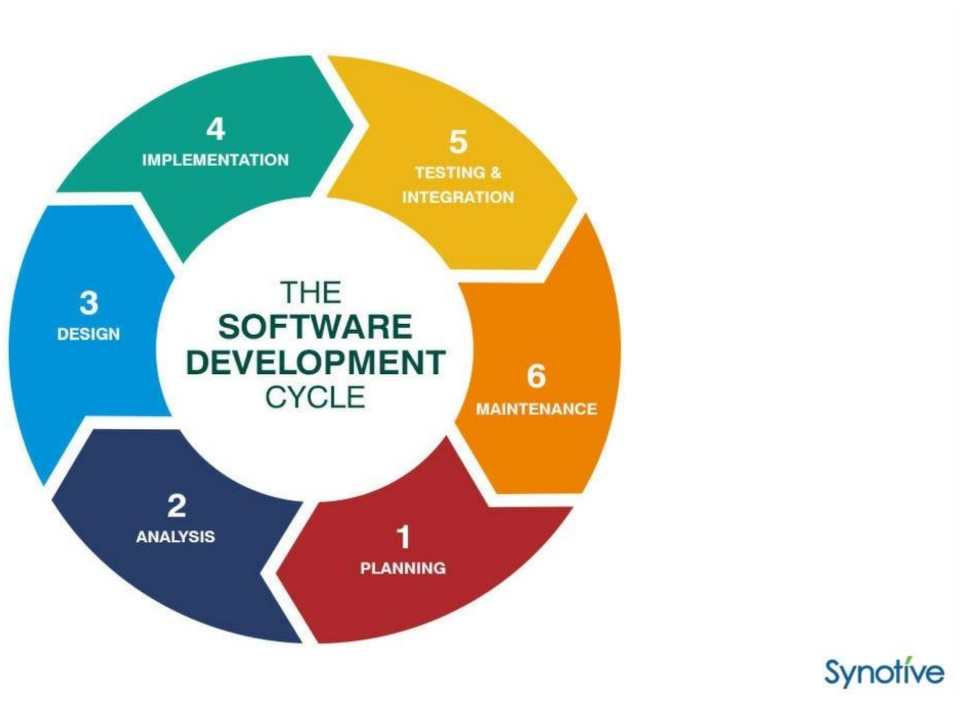
➢ **Data Flow Diagram**


DevSecOps: Key checkpoints
Network scans
Operate
App container security testing
Application penetration testing
Test
Release
Deploy
Data masking anonymization
Vulnerability assessment
Secure coding training
Build
Code
Plan
net solutions

➢ Activity Diagram

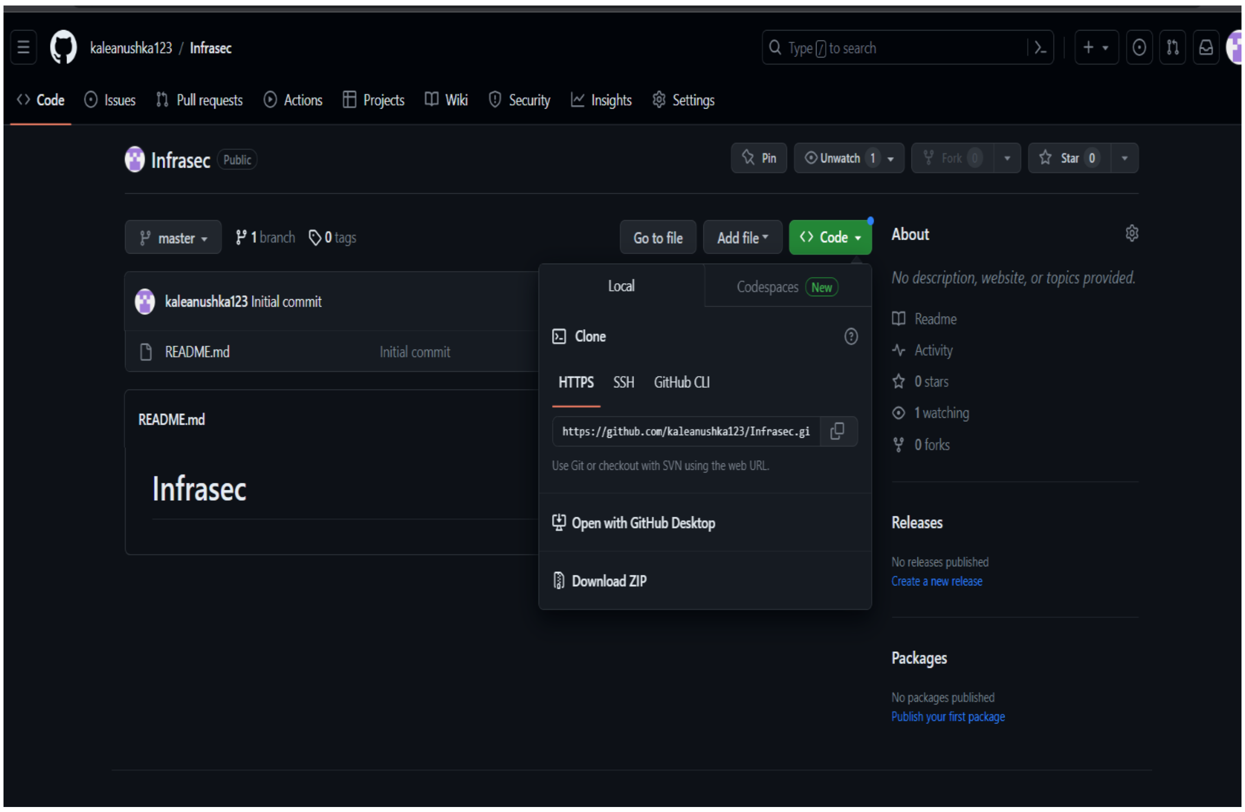


- **Initiation**
- **Design and Requirement gathering**
- **Development**
- **Testing and Code Analysis**
- **Deployment and Operation**



6. Introduction to Repository

First log-in with your GitHub account, then create new repository for the project.Now, copy the URL of the repository.



https://github.com/kaleanushka123/Infrasec.gi

Use Git or checkout with SVN using the web URL.

No description, website, or topics provided.

No releases published

No packages published

7. Git Concepts

Git is a version control system that is used to manage and track changes made to source code and other files. It was created by Linus Torvalds in 2005 to manage the development of the Linux operating system.With Git, developers can track changes to their code, collaborate with others on a project, and maintain multiple versions of their codebase. Git uses a distributed model, which means that every developer has a complete copy of the codebase on their local machine. This allows developers to work offline and independently, and then synchronize their changes with others when they're ready.

Git also provides features like branching and merging, which allow developers to create parallel versions of their codebase and merge changes made by multiple developers back into the main codebase. These features make it easier to manage complex development workflows and collaborate on large projects.

Overall, Git is a powerful and widely used tool for managing software development projects.

Git clone

git clone is a command used to create a copy of a Git repository on your local machine. To use git clone, you need to specify the URL of the remote repository you want to clone, as well as the location on your local machine where you want to create the copy. The basic syntax of the command is:

```
git clone <repository URL> <local directory>
```

Git commit

A Git commit is a fundamental operation in the Git version control system that allows you to record and save changes to your project's files. When you make changes to your code or other project files, a commit captures those changes as a snapshot and saves it in the Git repository's history

Git push

git push is a command used to upload local repository content to a remote repository.

When you make changes to your local repository, you can use git push to send those changes to a remote repository that you have previously cloned or created. The basic syntax of the command is:

```
git push <remote> <branch>
```

File pushed

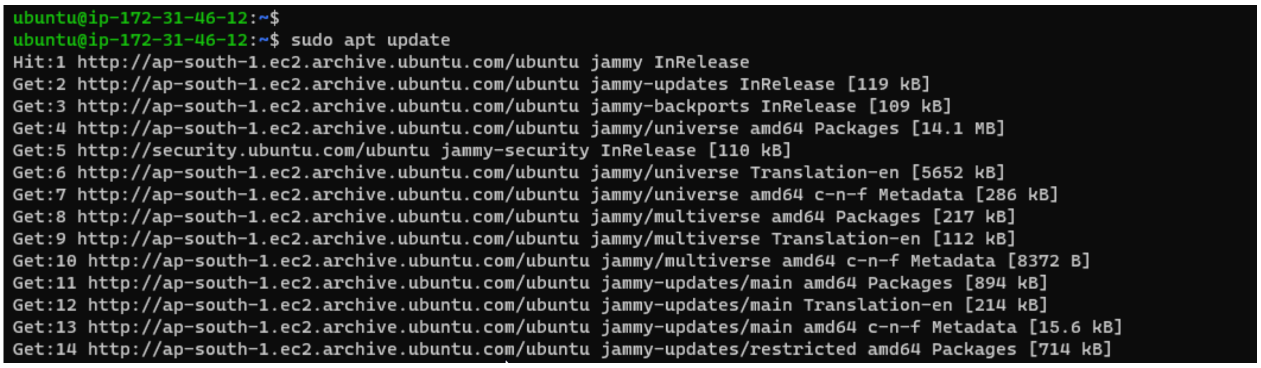
8. Jenkin server configuration

Jenkin server start

Jenkins is used to continually create and test software projects, making it easier for developers and DevOps engineers to integrate changes to the project and for consumers to get a new build.

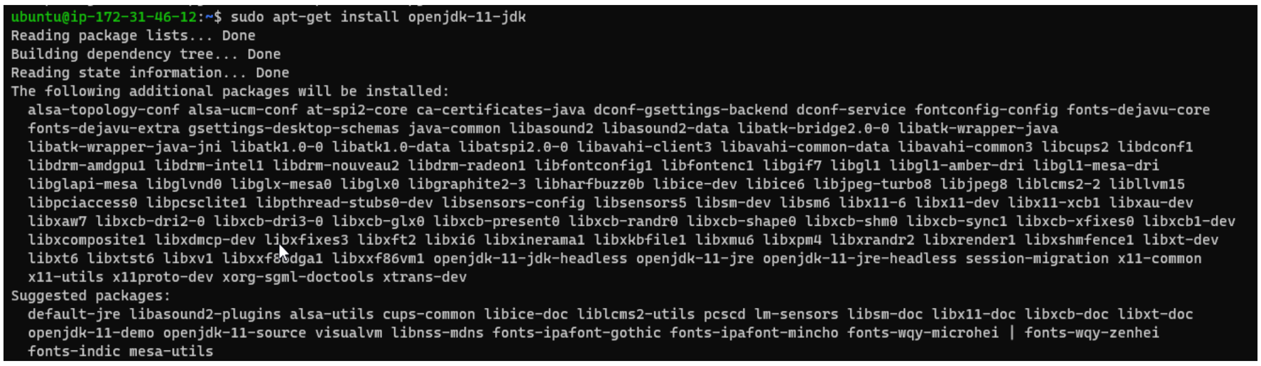
➔ Steps to Configure Jenkin-Server

Step 1: - Connect the Jenkins-server with the SSH-Client to the terminal, than update the Ubuntu packages lists by,



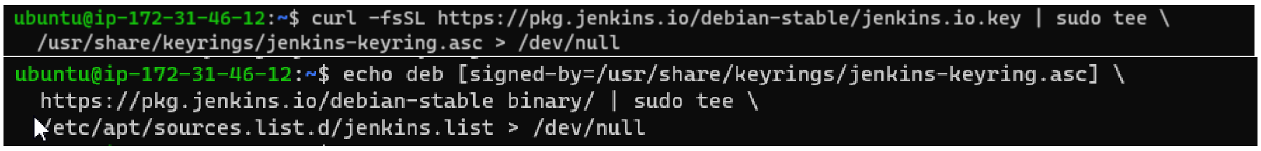
```
ubuntu@ip-172-31-46-12:~$
ubuntu@ip-172-31-46-12:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:6 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [894 kB]
Get:12 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [214 kB]
Get:13 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [15.6 kB]
Get:14 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [714 kB]
```

Step 2: - Install the Java JDK 11 package



```
ubuntu@ip-172-31-46-12:~$ sudo apt-get install openjdk-11-jdk
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  alsa-topology-conf alsa-ucm-conf at-spi2-core ca-certificates-java dconf-gsettings-backend dconf-service fontconfig-config fonts-dejavu-core
  fonts-dejavu-extra gsettings-desktop-schemas java-common libasound2 libasound2-data libatk-bridge2.0-0 libatk-wrapper-java
  libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data libavahi-common3 libcups2 libdconf1
  libdrm-amdgpu1 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libfontconfig1 libfontenc1 libgif7 libgl1 libgl1-amber-dri libgl1-mesa-dri
  libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libharfbuzz0b libice-dev libice6 libjpeg-turbo8 libjpeg8 liblcms2-2 libllvm15
  libpciaccess0 libpcsclite1 libpthread-stubs0-dev libsensors-config libsensors5 libsm-dev libsm6 libx11-6 libx11-dev libx11-xcb1 libxau-dev
  libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-shape0 libxcb-shm0 libxcb-sync1 libxcb-xfixes0 libxcb1-dev
  libxcomposite1 libxdmcp-dev libxfixes3 libxft2 libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev
  libxt6 libxtst6 libxv1 libxxf86dga1 libxxf86vm1 openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless session-migration x11-common
  x11-utils x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  default-jre libasound2-plugins alsa-utils cups-common libice-doc liblcms2-utils pcscd lm-sensors libsm-doc libx11-doc libxcb-doc libxt-doc
  openjdk-11-demo openjdk-11-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei | fonts-wqy-zenhei
  fonts-indic mesa-utils
```

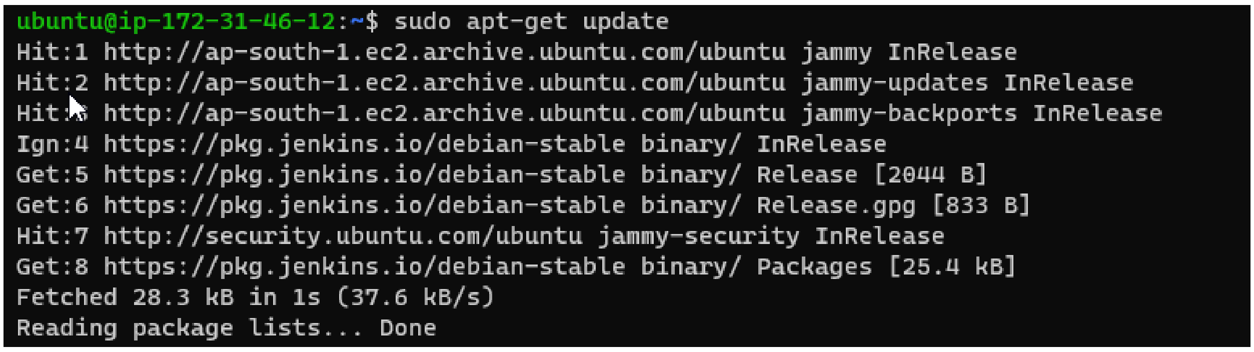
Step 3: - Install Jenkins from the Jenkins repository as,



```
ubuntu@ip-172-31-46-12:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

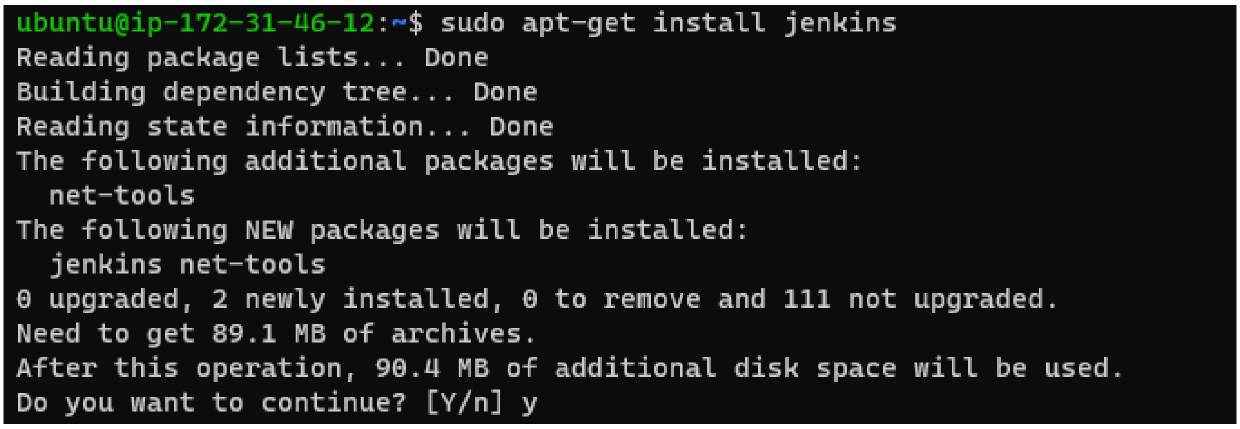
```
ubuntu@ip-172-31-46-12:~$ echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
```

Step 4: - Again update the packages list for Jenkins

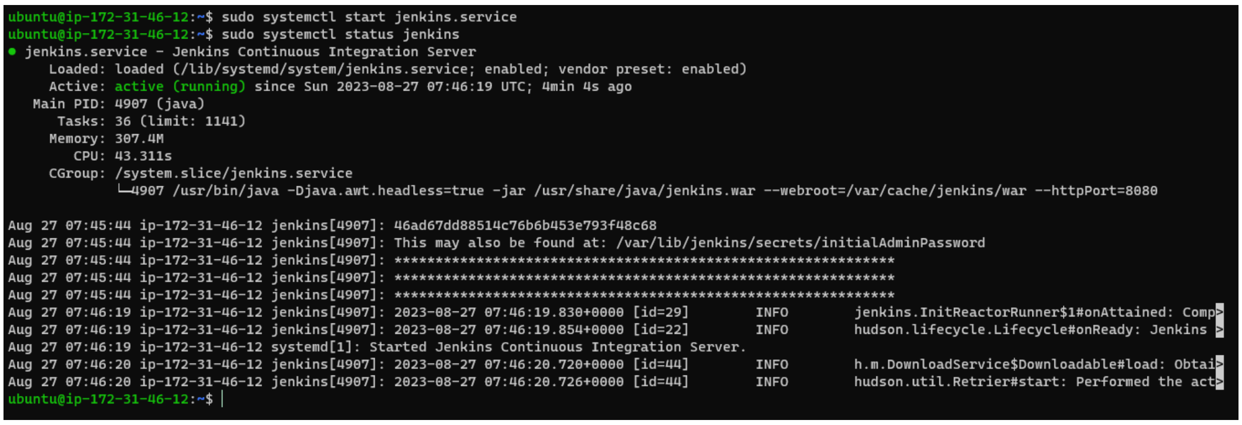


```
ubuntu@ip-172-31-46-12:~$ sudo apt-get update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [25.4 kB]
Fetched 28.3 kB in 1s (37.6 kB/s)
Reading package lists... Done
```

Step 5: - Install the Jenkins, start the Jenkins service & check the status of Jenkins by,

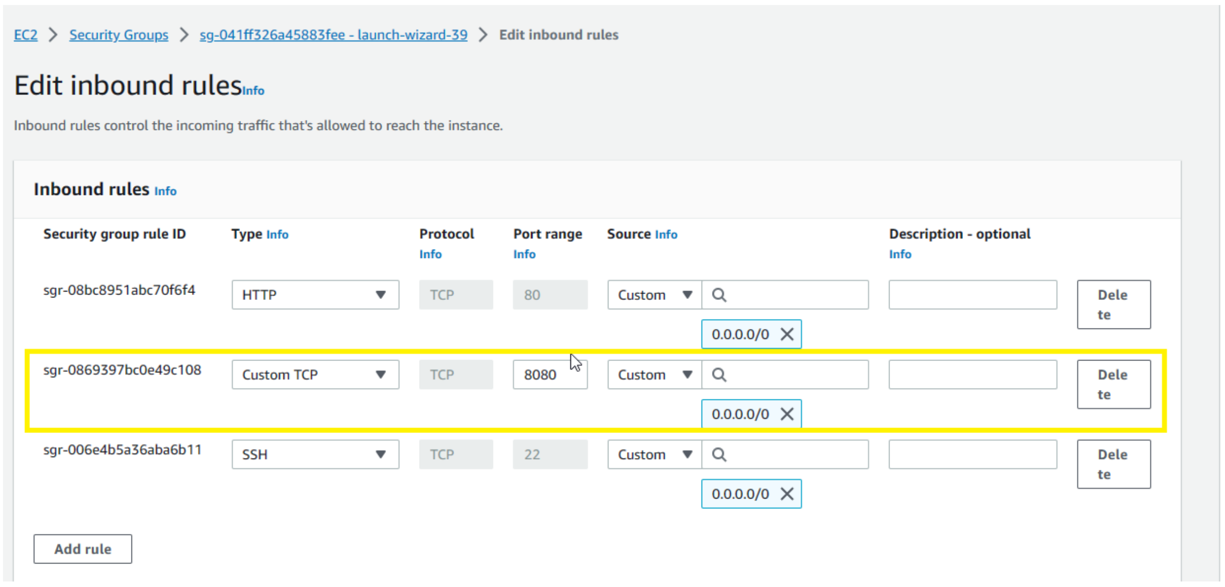


```
ubuntu@ip-172-31-46-12:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 111 not upgraded.
Need to get 89.1 MB of archives.
After this operation, 90.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```



```
ubuntu@ip-172-31-46-12:~$ sudo systemctl start jenkins.service
ubuntu@ip-172-31-46-12:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-08-27 07:46:19 UTC; 4min 4s ago
   Main PID: 4907 (java)
      Tasks: 36 (limit: 1141)
     Memory: 307.4M
        CPU: 43.311s
     CGroup: /system.slice/jenkins.service
             └─4907 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Aug 27 07:45:44 ip-172-31-46-12 jenkins[4907]: 46ad67dd88514c76b6b453e793f48c68
Aug 27 07:45:44 ip-172-31-46-12 jenkins[4907]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Aug 27 07:45:44 ip-172-31-46-12 jenkins[4907]: *************************************************************
Aug 27 07:45:44 ip-172-31-46-12 jenkins[4907]: *************************************************************
Aug 27 07:45:44 ip-172-31-46-12 jenkins[4907]: *************************************************************
Aug 27 07:46:19 ip-172-31-46-12 jenkins[4907]: 2023-08-27 07:46:19.830+0000 [id=29]        INFO        jenkins.InitReactorRunner$1#onAttained: Comp>
Aug 27 07:46:19 ip-172-31-46-12 jenkins[4907]: 2023-08-27 07:46:19.854+0000 [id=22]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins >
Aug 27 07:46:19 ip-172-31-46-12 systemd[1]: Started Jenkins Continuous Integration Server.
Aug 27 07:46:20 ip-172-31-46-12 jenkins[4907]: 2023-08-27 07:46:20.720+0000 [id=44]        INFO        h.m.DownloadService$Downloadable#load: Obtai>
Aug 27 07:46:20 ip-172-31-46-12 jenkins[4907]: 2023-08-27 07:46:20.726+0000 [id=44]        INFO        hudson.util.Retrier#start: Performed the act>
ubuntu@ip-172-31-46-12:~$ |
```

Step 6: - Now add the Security Group rule to allow the 8080 port for Jenkins as Jenkins default port is 8080 so,



EC2 > Security Groups > sg-041ff326a45883fee - launch-wizard-39 > Edit inbound rules

Edit inbound rules Info

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

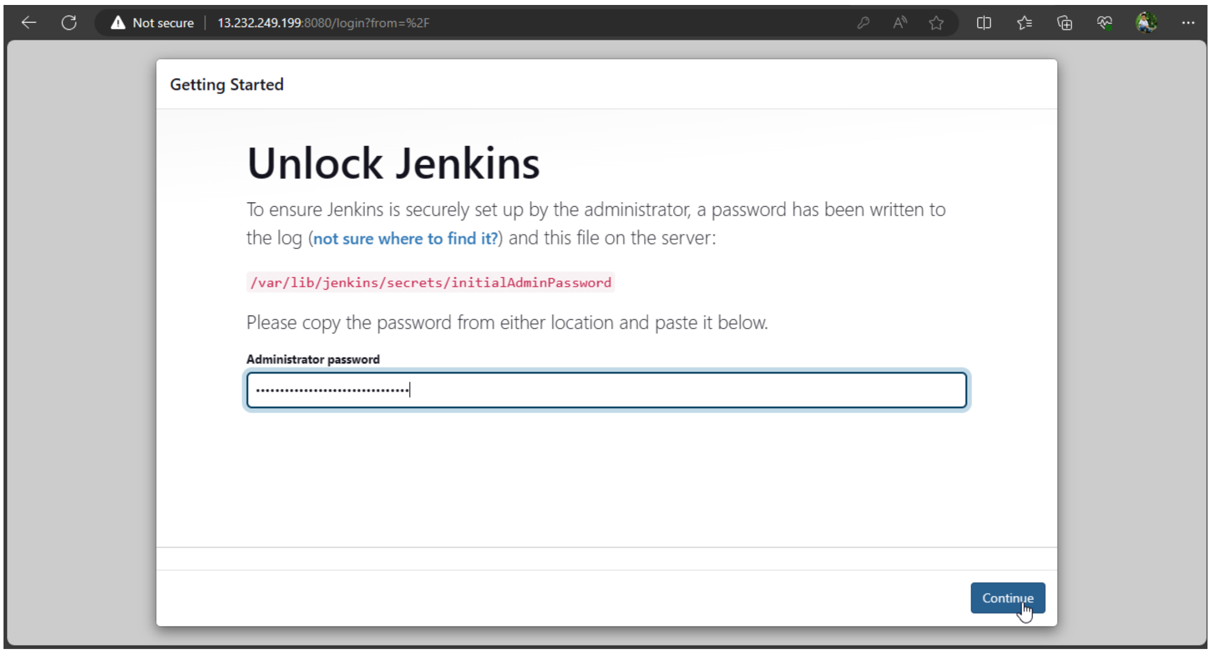
Inbound rules Info

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-08bc8951abc70f6f4	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-0869397bc0e49c108	Custom TCP	TCP	8080	Custom	0.0.0.0/0		Delete
sgr-006e4b5a36aba6b11	SSH	TCP	22	Custom	0.0.0.0/0		Delete

Step 7: - Copy the Administration Password from the below file

```
ubuntu@ip-172-31-46-12:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
46ad67dd88514c76b6b453e793f48c68
ubuntu@ip-172-31-46-12:~$
```

Step 8: - Now Go to Browser & write <Jenkins-server Public_IP:8080>
& then Paste the Administration Password → Continue



Not secure | 13.232.249.199:8080/login?from=%2F

Getting Started

Unlock Jenkins

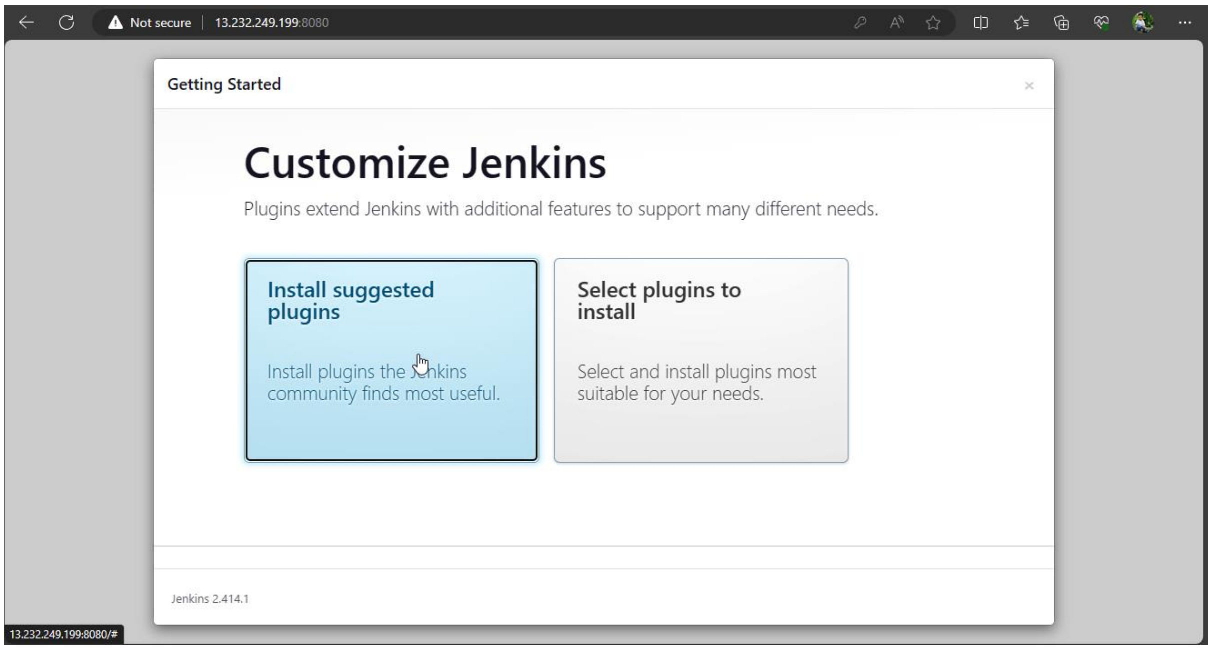
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

/var/lib/jenkins/secrets/initialAdminPassword

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

Administrator password: ································

Step 9: - Installed the Suggested Plugins,



Customize Jenkins

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

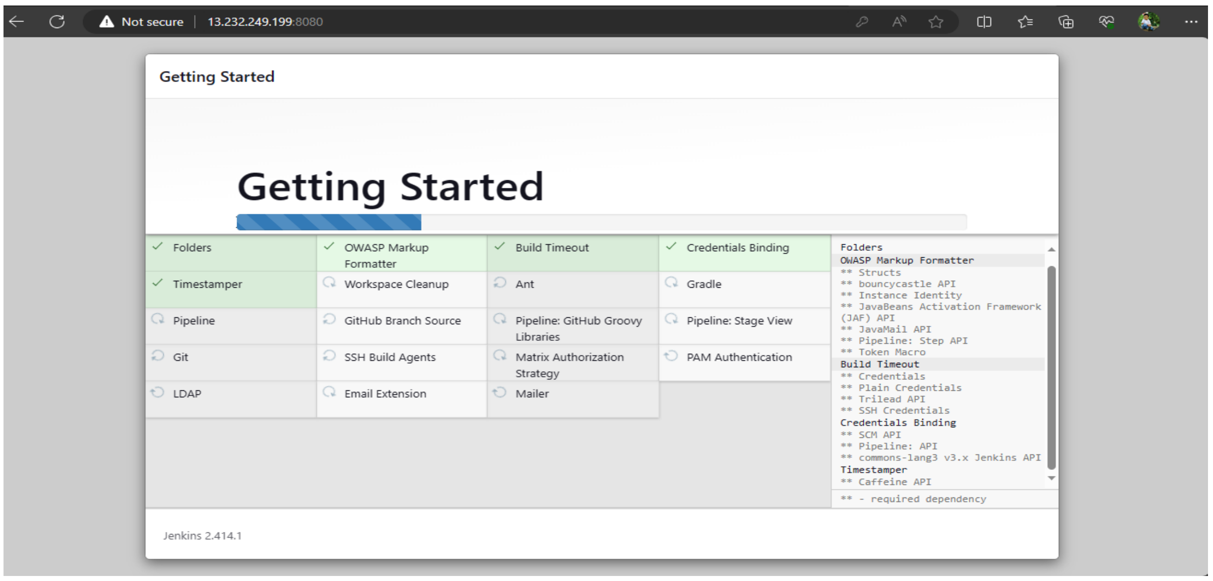
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.414.1



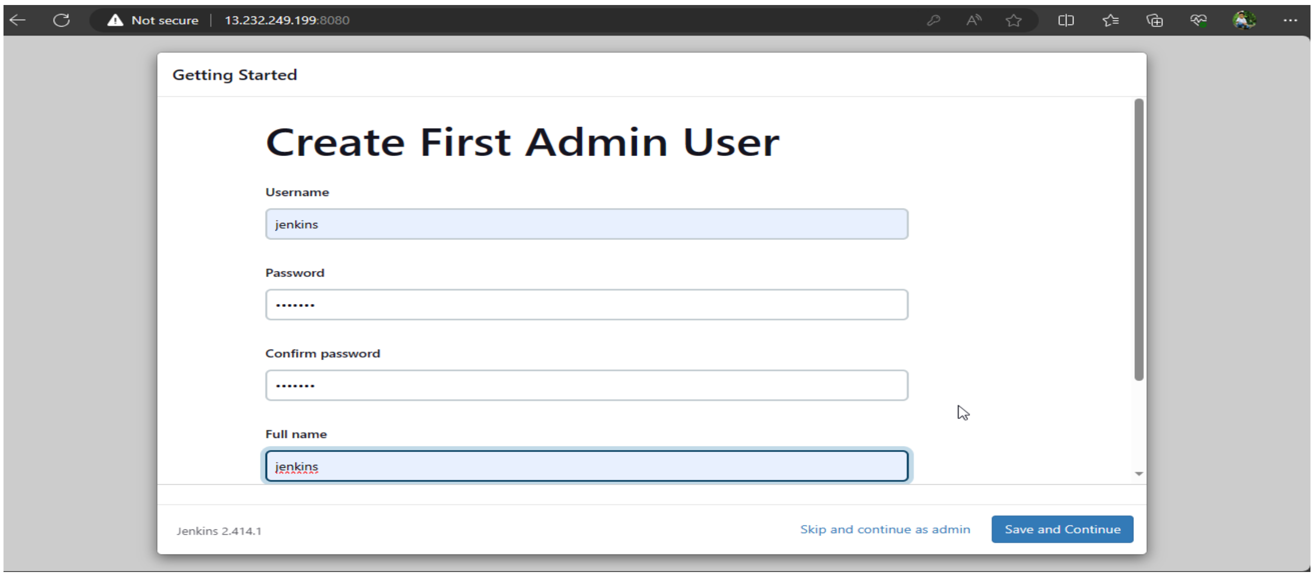
Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	

```
Folders
OWASP Markup Formatter
** Structs
** bouncycastle API
** Instance Identity
** JavaBeans Activation Framework
(JAF) API
** JavaMail API
** Pipeline: Step API
** Token Macro
Build Timeout
** Credentials
** Plain Credentials
** Trilead API
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
** commons-lang3 v3.x Jenkins API
Timestamper
** Caffeine API
** - required dependency
```

Jenkins 2.414.1

Step 10: - Create the Admin User → Save & continue → Save & Finish


Getting Started

Create First Admin User

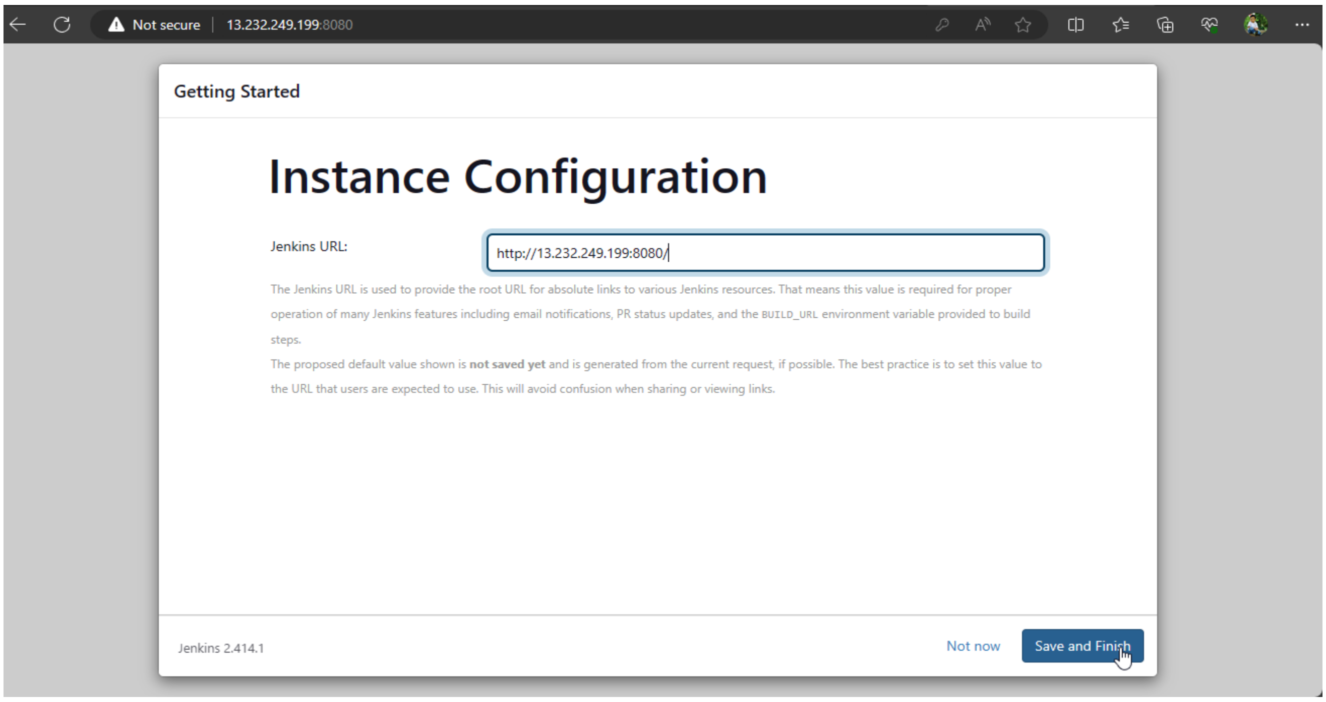
Username: jenkins

Password: •••••••

Confirm password: •••••••

Full name: jenkins

Jenkins 2.414.1


Getting Started

Instance Configuration

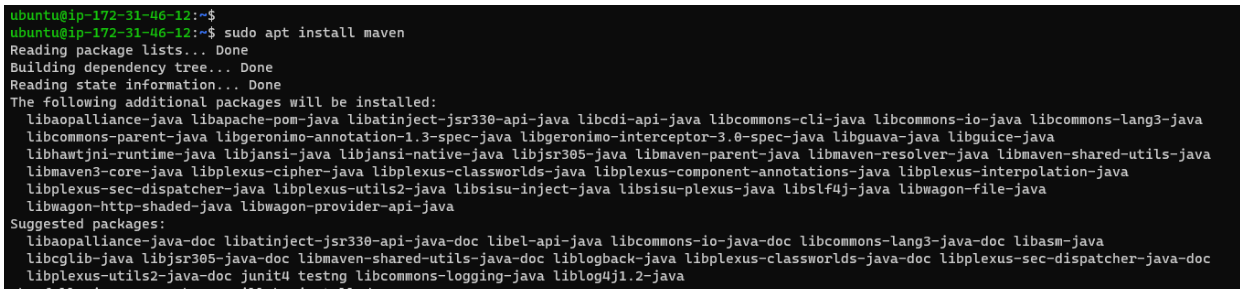
Jenkins URL: http://13.232.249.199:8080/

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.

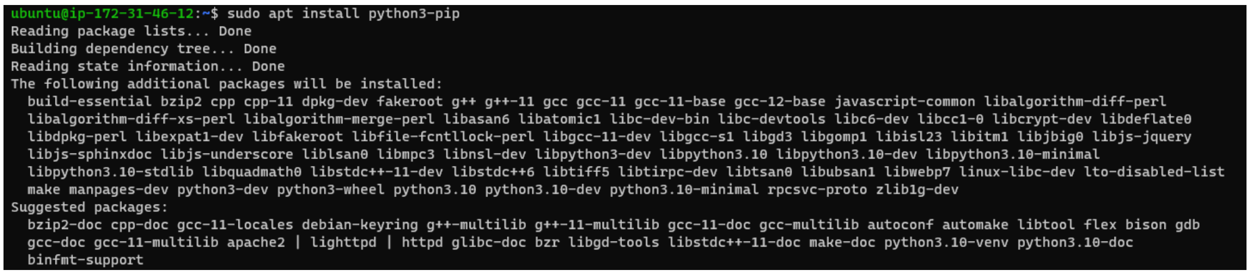
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.414.1

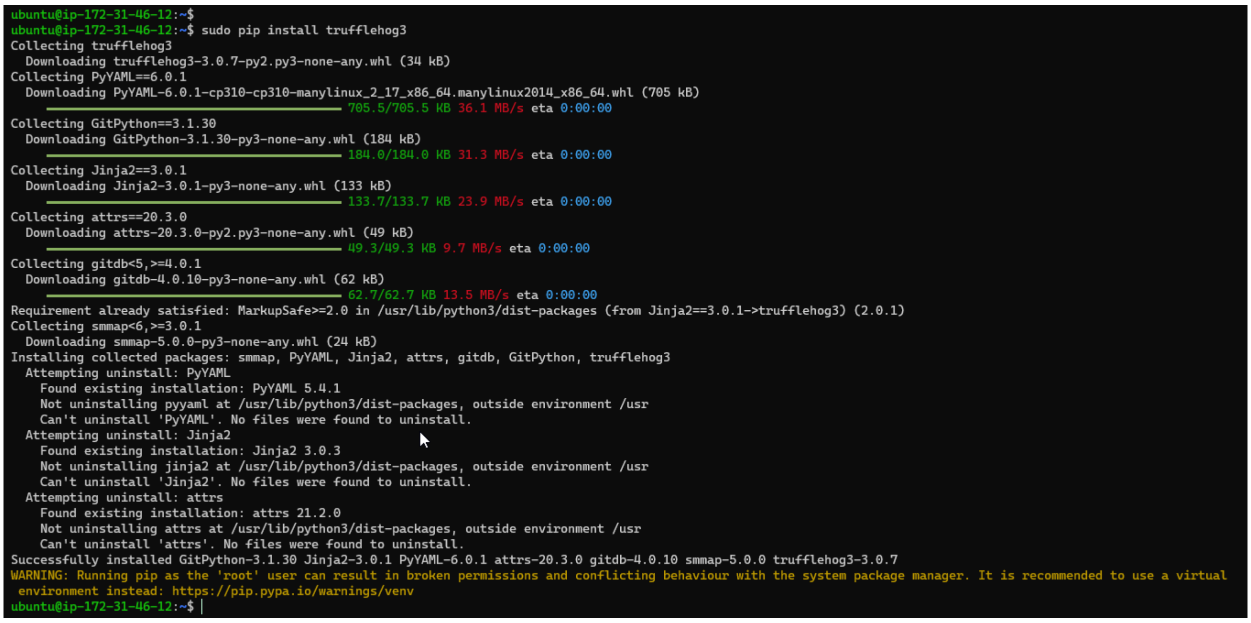
Step 11: - For Jenkins-Server we need Maven, so install maven as


```
ubuntu@ip-172-31-46-12:~$
ubuntu@ip-172-31-46-12:~$ sudo apt install maven
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java libcdi-api-java libcommons-cli-java libcommons-io-java libcommons-lang3-java
  libcommons-parent-java libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libguava-java libguice-java
  libhawtjni-runtime-java libjansi-java libjansi-native-java libjsr305-java libmaven-parent-java libmaven-resolver-java libmaven-shared-utils-java
  libmaven3-core-java libplexus-cipher-java libplexus-classworlds-java libplexus-component-annotations-java libplexus-interpolation-java
  libplexus-sec-dispatcher-java libplexus-utils2-java libsisu-inject-java libsisu-plexus-java libslf4j-java libwagon-file-java
  libwagon-http-shaded-java libwagon-provider-api-java
Suggested packages:
  libaopalliance-java-doc libatinject-jsr330-api-java-doc libel-api-java libcommons-io-java-doc libcommons-lang3-java-doc libasm-java
  libcglib-java libjsr305-java-doc libmaven-shared-utils-java-doc liblogback-java libplexus-classworlds-java-doc libplexus-sec-dispatcher-java-doc
  libplexus-utils2-java-doc junit4 testng libcommons-logging-java liblog4j1.2-java
```

Step 12: - Now we need to install trufflehog3, but to install trufflehog3 we need to install python3-pip, so



```
ubuntu@ip-172-31-46-12:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  build-essential bzip2 cpp cpp-11 dpkg-dev fakeroot g++ g++-11 gcc gcc-11 gcc-11-base gcc-12-base javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan6 libatomic1 libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libdeflate0
  libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-11-dev libgcc-s1 libgd3 libgomp1 libisl23 libitm1 libjbig0 libjs-jquery
  libjs-sphinxdoc libjs-underscore liblsan0 libmpc3 libnsl-dev libpython3-dev libpython3.10 libpython3.10-dev libpython3.10-minimal
  libpython3.10-stdlib libquadmath0 libstdc++-11-dev libstdc++6 libtiff5 libtirpc-dev libtsan0 libubsan1 libwebp7 linux-libc-dev lto-disabled-list
  make manpages-dev python3-dev python3-wheel python3.10 python3.10-dev python3.10-minimal rpcsvc-proto zlib1g-dev
Suggested packages:
  bzip2-doc cpp-doc gcc-11-locales debian-keyring g++-multilib g++-11-multilib gcc-11-doc gcc-multilib autoconf automake libtool flex bison gdb
  gcc-doc gcc-11-multilib apache2 | lighttpd | httpd glibc-doc bzr libgd-tools libstdc++-11-doc make-doc python3.10-venv python3.10-doc
  binfmt-support
```



```
ubuntu@ip-172-31-46-12:~$
ubuntu@ip-172-31-46-12:~$ sudo pip install trufflehog3
Collecting trufflehog3
  Downloading trufflehog3-3.0.7-py2.py3-none-any.whl (34 kB)
Collecting PyYAML==6.0.1
  Downloading PyYAML-6.0.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (705 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 705.5/705.5 KB 36.1 MB/s eta 0:00:00
Collecting GitPython==3.1.30
  Downloading GitPython-3.1.30-py3-none-any.whl (184 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 184.0/184.0 KB 31.3 MB/s eta 0:00:00
Collecting Jinja2==3.0.1
  Downloading Jinja2-3.0.1-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.7/133.7 KB 23.9 MB/s eta 0:00:00
Collecting attrs==20.3.0
  Downloading attrs-20.3.0-py2.py3-none-any.whl (49 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.3/49.3 KB 9.7 MB/s eta 0:00:00
Collecting gitdb<5,>=4.0.1
  Downloading gitdb-4.0.10-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 KB 13.5 MB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in /usr/lib/python3/dist-packages (from Jinja2==3.0.1->trufflehog3) (2.0.1)
Collecting smmap<6,>=3.0.1
  Downloading smmap-5.0.0-py3-none-any.whl (24 kB)
Installing collected packages: smmap, PyYAML, Jinja2, attrs, gitdb, GitPython, trufflehog3
  Attempting uninstall: PyYAML
    Found existing installation: PyYAML 5.4.1
    Not uninstalling pyyaml at /usr/lib/python3/dist-packages, outside environment /usr
    Can't uninstall 'PyYAML'. No files were found to uninstall.
  Attempting uninstall: Jinja2
    Found existing installation: Jinja2 3.0.3
    Not uninstalling jinja2 at /usr/lib/python3/dist-packages, outside environment /usr
    Can't uninstall 'Jinja2'. No files were found to uninstall.
  Attempting uninstall: attrs
    Found existing installation: attrs 21.2.0
    Not uninstalling attrs at /usr/lib/python3/dist-packages, outside environment /usr
    Can't uninstall 'attrs'. No files were found to uninstall.
Successfully installed GitPython-3.1.30 Jinja2-3.0.1 PyYAML-6.0.1 attrs-20.3.0 gitdb-4.0.10 smmap-5.0.0 trufflehog3-3.0.7
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual
 environment instead: https://pip.pypa.io/warnings/venv
ubuntu@ip-172-31-46-12:~$ |
```

9. Project setup with CI CD Pipeline:

Pre-requisites:

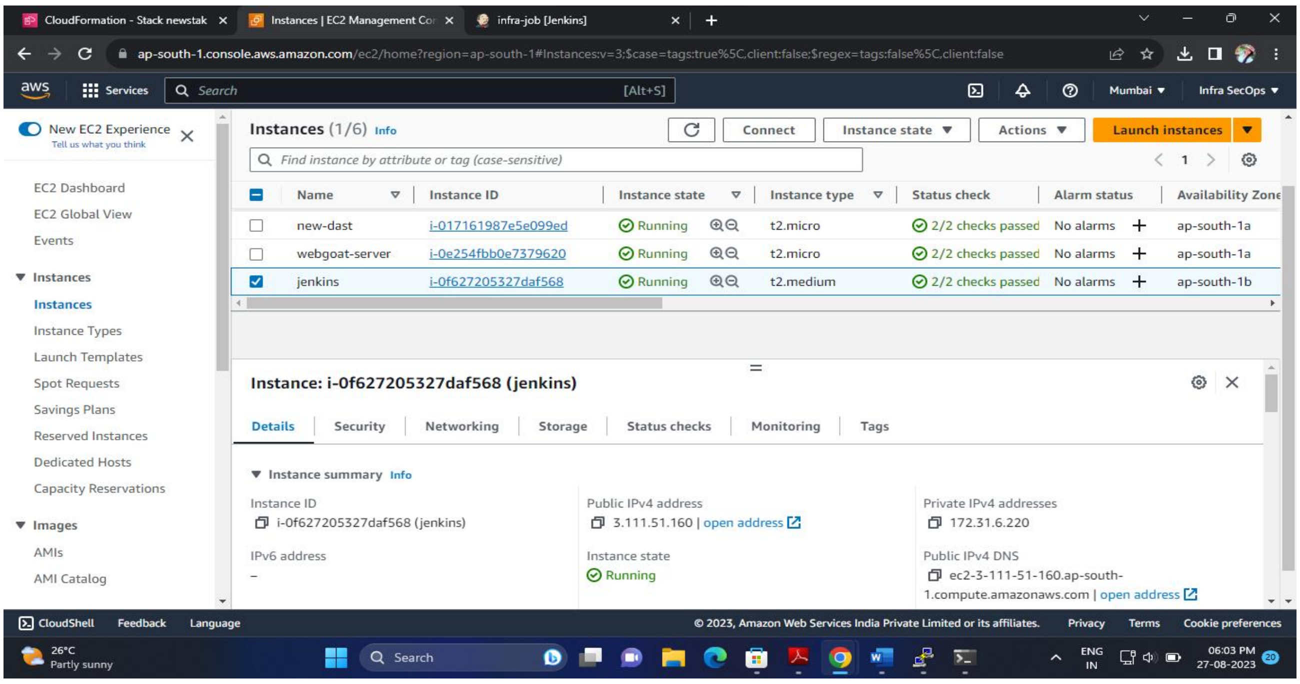
1) GitHub

(Repo url: https://github.com/kaleanushka123/Infra-secops.git**)**

2) Jenkins Server (url:http://public-ip:8080 **)**

3) Maven (as a global tool in Jenkins**)**

4) AWS Instance



Instances (1/6) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
new-dast	i-017161987e5e099ed	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a
webgoat-server	i-0e254fbb0e7379620	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a
jenkins	i-0f627205327daf568	Running	t2.medium	2/2 checks passed	No alarms	ap-south-1b

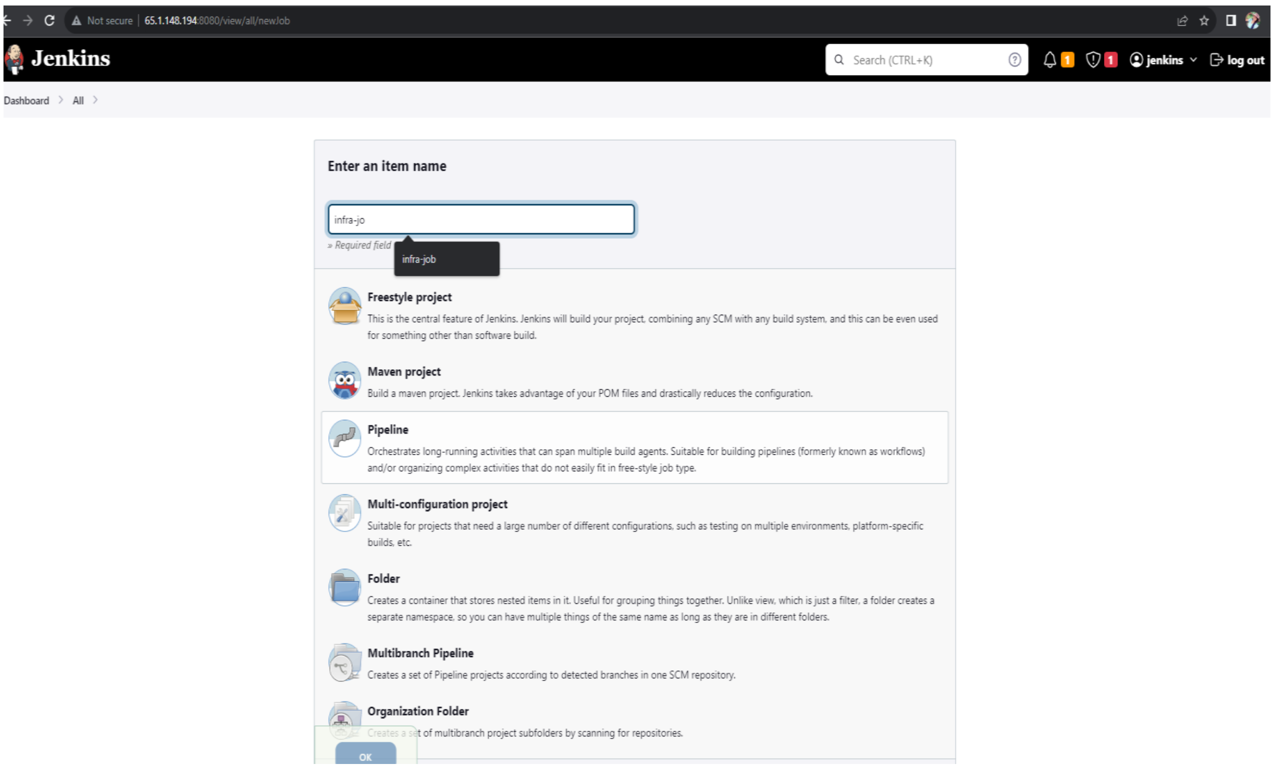
Instance: i-0f627205327daf568 (jenkins)

Instance summary Info

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0f627205327daf568 (jenkins)	3.111.51.160 | open address	172.31.6.220
IPv6 address	Instance state	Public IPv4 DNS
–	Running	ec2-3-111-51-160.ap-south-1.compute.amazonaws.com | open address

Note: ***The installation process of Pre-requisites are present in a GitHub repository link.**

- Making pipeline named as Infra-job


Not secure | 65.1.148.194:8080/view/all/newJob

Jenkins

Search (CTRL+K)

jenkins log out

Dashboard > All >

Enter an item name

infra-jo

» Required field

infra-job

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

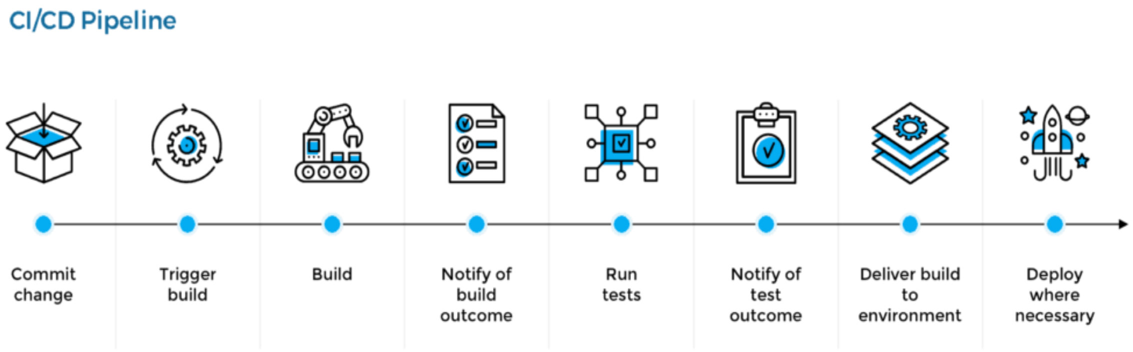
Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

<u>CI/CD PIPELINE</u>

A CI/CD pipeline automates the process of software delivery. It builds code, runs tests, and helps you to safely deploy a new version of the software. CI/CD pipeline reduces manual errors, provides feedback to developers, and allows fast product iterations.

CI/CD pipeline introduces automation and continuous monitoring throughout the lifecycle of a software product. It involves from the integration and testing phase to delivery and deployment. These connected practices are referred as CI/CD pipeline



10. Implementation of CI/CD pipeline

1. Source Stage

As we all know, with the help of CI, a source code repository gets created. Any change in the code or establishing a new code is notified to the associated pipeline. In this way, any error happening like automated scheduling or user initiating any workflow, or results from other pipelines are notified.

2. Build Stage

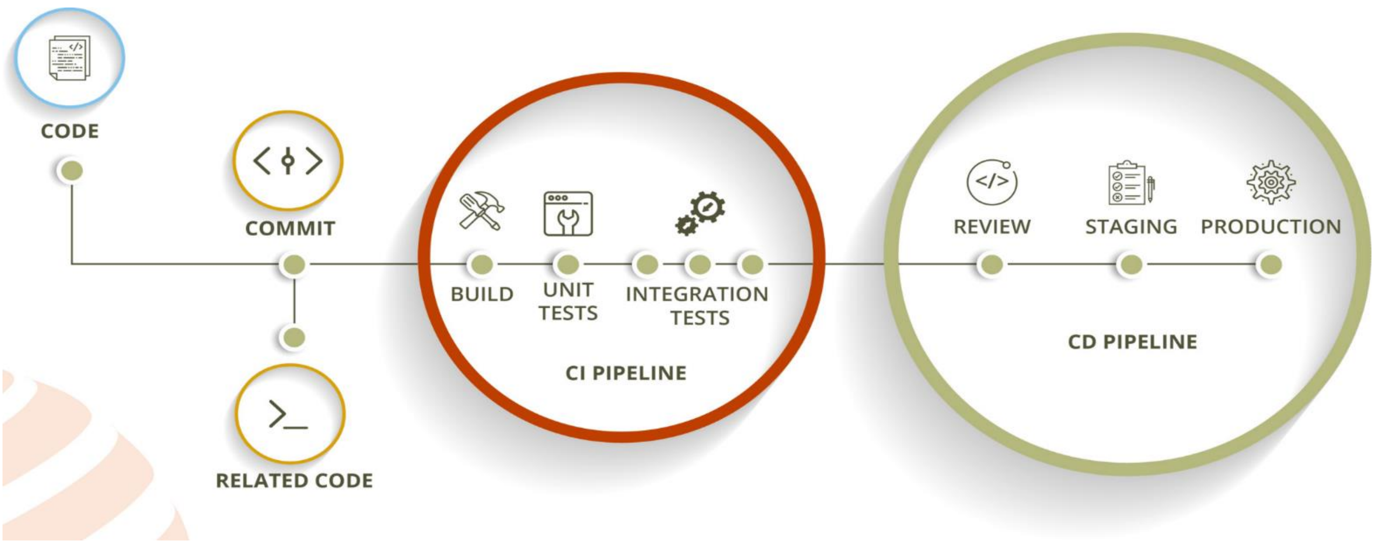
In this step, the application is compiled by putting together the codes and related dependencies. It helps in building a runnable entity of the product. Further, the runnable product or instance is shipped to the end-users.

3. Deployment Stage

After all the qualify and production checks, the instance then becomes ready to deploy on the product. Depending on the project type, there can be multiple layers in the deployment of the version. It can be a beta or staging version that runs internally by the product team. Once the instance is sorted, the codes get automatically deployed on the product. In this way, these elements work for the deployment of a new version of an application.

4. Testing Stage

The most crucial stage of all is the Testing phase. Automated testing is done to verify the ascertain the behavior of the product and exactness of the codes so implemented. It becomes a safety trap that stops reproducible bugs from reaching the end-user. The duration of testing depends upon the type of project and the skills required to validate the product.



- **After installation of Jenkins, we have to install plugins in Jenkins server**

1. Maven
2. Treffulhog3
3. ssh-agent
4. owasp-zap

11. Maven

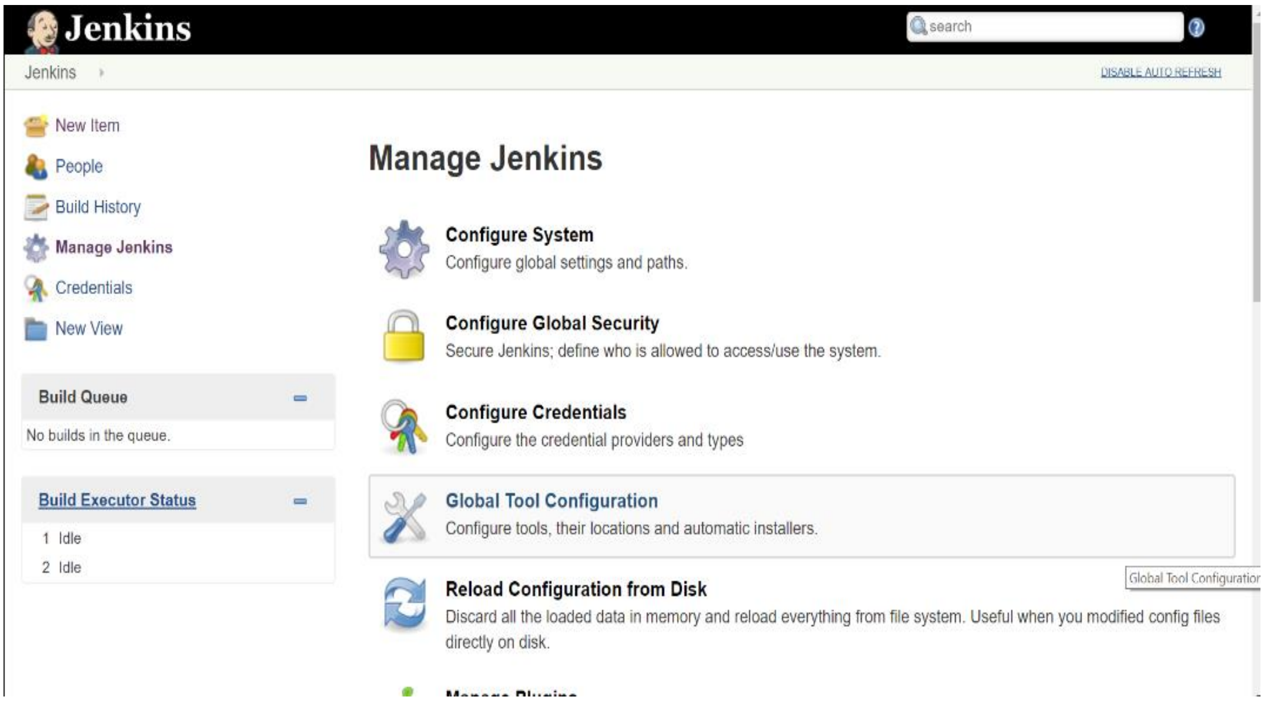
Maven is a build automation and dependency management tool primarily used for Java projects. It was developed by Apache Software Foundation and is built on the concept of "convention over configuration", which means that Maven uses a standard directory layout and a set of default configurations to build Java projects.

Maven simplifies the build process by managing dependencies and providing a consistent build process across different projects. It uses a central repository called Maven Central to download project dependencies, which makes it easy to manage and update dependencies for a project.

Maven also provides a standard way to manage the build lifecycle of a project, with a set of pre-defined phases like compile, test, package, install, and deploy. Developers can define custom build goals and plugins to extend the functionality of Maven.

Overall, Maven is a popular tool for Java developers to manage the build process and dependencies of their projects, and it can help to improve project organization and reduce build errors.

- After installing maven plugins, we have to set maven path in global tool configuration


Jenkins

Manage Jenkins

New Item
People
Build History
Manage Jenkins
Credentials
New View

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

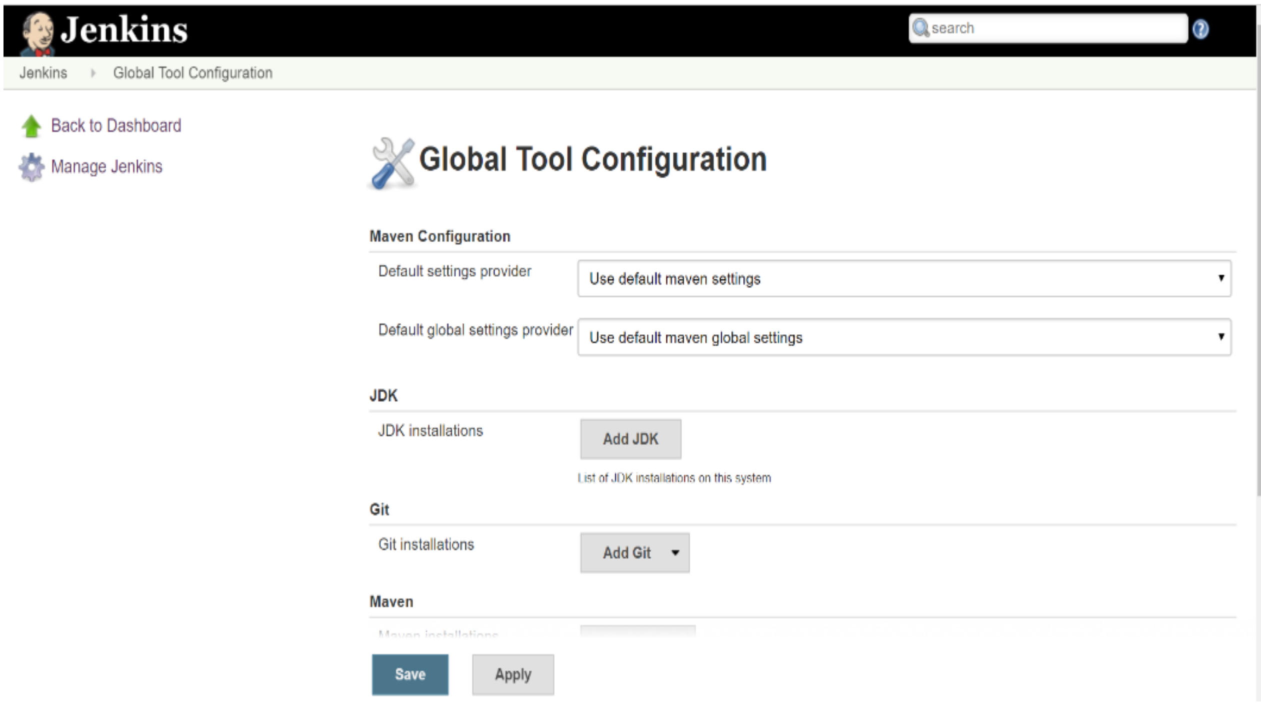
Configure System
Configure global settings and paths.

Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.

Configure Credentials
Configure the credential providers and types

Global Tool Configuration
Configure tools, their locations and automatic installers.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.


Jenkins › Global Tool Configuration

Back to Dashboard
Manage Jenkins

Global Tool Configuration

Maven Configuration
Default settings provider: Use default maven settings
Default global settings provider: Use default maven global settings

JDK
JDK installations: Add JDK
List of JDK installations on this system

Git
Git installations: Add Git

Maven

Save Apply

- Note: we have to add java path and maven path in add JDK and add maven in this global tool configuration. At first, we have to install java & maven in our Jenkins terminal.

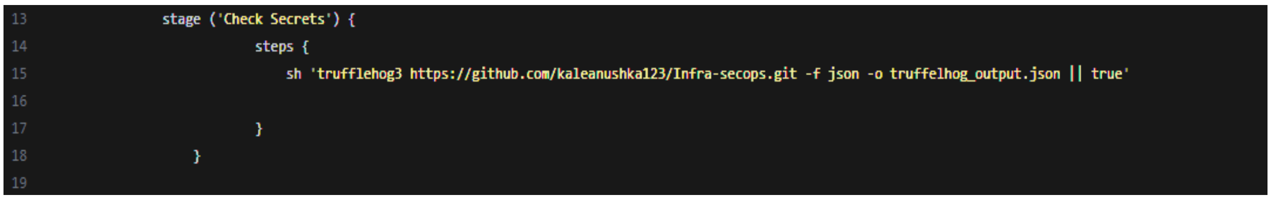
12.Trufflehog3

- Trufflehog3 is an open-source security tool that is used to search for secrets and sensitive information in code repositories. It is designed to identify and alert developers to any instances of sensitive information such as passwords, API keys, and other confidential data that may have been accidentally committed to a code repository.

- Trufflehog3 works by scanning the entire commit history of a code repository and analyzing the contents of each file to detect any instances of secrets or sensitive information. It supports various types of repositories, including Git, Mercurial, and Subversion. Trufflehog3 can be run locally or as part of a continuous integration/continuous deployment (CI/CD) pipeline.

- Trufflehog3 uses regular expressions to search for sensitive information and can be customized to search forspecific types of secrets or data. It can also be configured to ignore certain files or directories.

- Overall, Trufflehog3 is a useful tool for identifying and removing sensitive information from code repositories and preventing security breaches. It can be integrated into DevSecOps processes to ensure that security is considered throughout the software development life cycle.



```
        stage ('Check Secrets') {
                steps {
                    sh 'trufflehog3 https://github.com/kaleanushka123/Infra-secops.git -f json -o truffelhog_output.json || true'

                }
            }

```

13. SSH-Agent

- In a Jenkins pipeline, an SSH agent is used to manage SSH keys securely during the execution of tasks that involve SSH operations, such as connecting to remote servers, deploying code, or executing commands on remote machines. SSH (Secure Shell) is a cryptographic network protocol that allows secure communication between two systems over an insecure network.

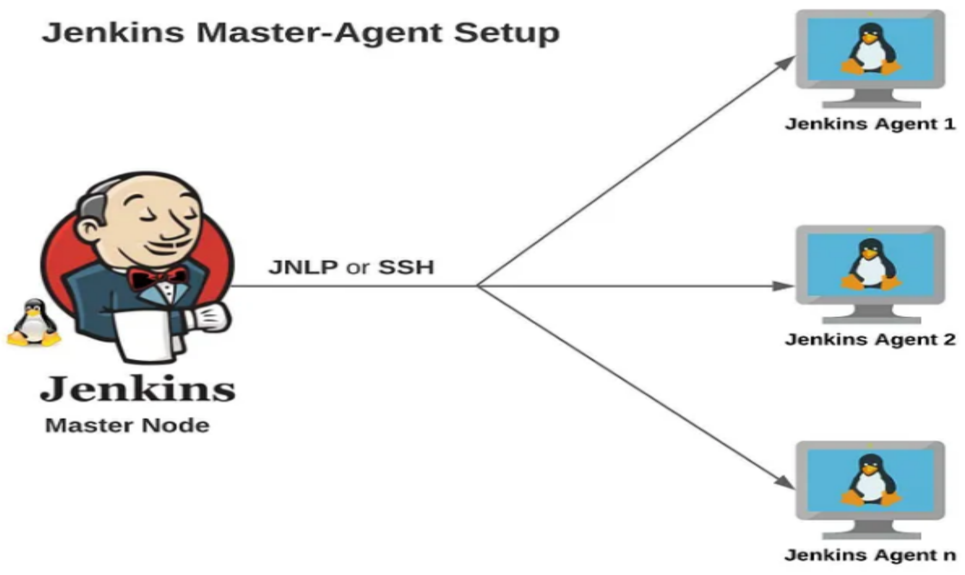
- The SSH agent in Jenkins pipeline helps in automating the process of providing authentication credentials (SSH keys) required to establish secure connections with remote servers. This is particularly useful when you want to interact with version control repositories, deploy applications, or perform other tasks that involve SSH-based authentication.

- Credential Setup: You would set up your SSH key pair in Jenkins as a credential. The private key is stored securely within Jenkins, and the public key is added to the authorized_keys file on the remote server you want to connect to.

- Pipeline Configuration: In your Jenkins pipeline script, you can use the ssh-agent step to load your SSH key for a specific block of code. This step loads the private key from the Jenkins credential store and makes it available to the pipeline script during its execution.

- SSH Operations: Inside the ssh-agent block, you can execute SSH-related operations, such as running commands on remote servers using SSH, pulling code from a Git repository over SSH, or deploying code to a remote server.

- Using an SSH agent in Jenkins pipelines helps maintain security by keeping sensitive credentials out of the pipeline script while still allowing automated SSH-based interactions with remote systems.



14. OWASP

- OWASP stands for the Open Web Application Security Project. It is a non-profit organization that aimsto improve the security of software by providing resources and tools to developers, security professionals, and organizations.

- OWASP provides a range of resources to help developers and organizations improve the security of their applications, including documentation, tools, best practices, and guidelines. The organization also conducts research on emerging security threats and vulnerabilities and provides information on how to mitigate them.

- One of the most well-known resources provided by OWASP is the OWASP Top 10, which is a list of the 10 most critical web application security risks. The OWASP Top 10 is widely used by security professionals and organizations as a benchmark for assessing the security of their web applications.

- Overall, OWASP is a valuable resource for anyone involved in software development or security, and it plays an important role in improving the security of software application

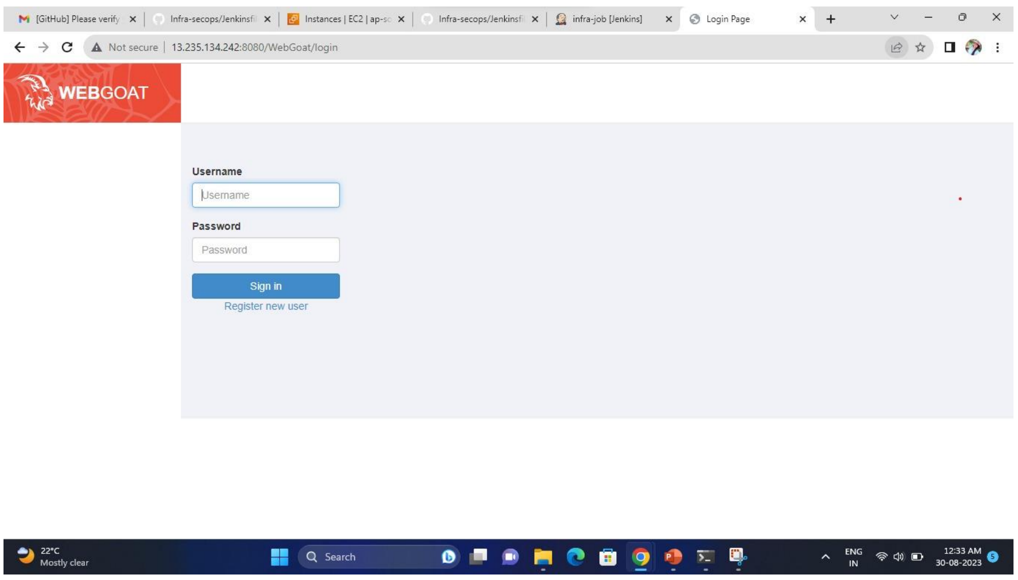
➢ OWASP-ZAP (zed attack proxy)

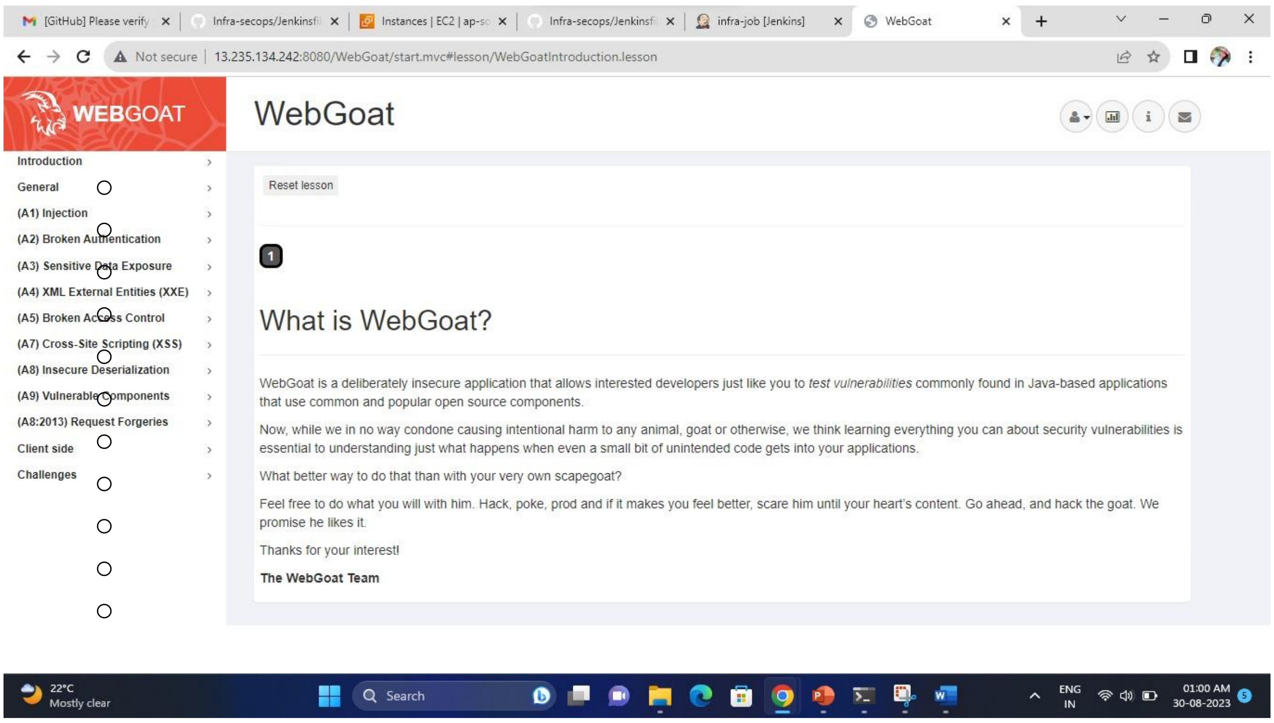
- Is an open-source security tool developed by the Open Web Application Security Project (OWASP). It is designed for finding security vulnerabilities in web applications during development and testing phases. ZAP helps identify various types of vulnerabilities, such as cross-site scripting (XSS), SQL injection, security misconfigurations, and more. Integrating OWASP ZAP into a Jenkins pipeline allows you to automate security testing as part of your continuous integration (CI) and continuous delivery (CD) process. This helps ensure that security checks are performed consistently whenever code changes are made, reducing the risk of vulnerabilities making their way into the production environment.

- By integrating OWASP ZAP into your Jenkins pipeline, you can proactively identify and address security vulnerabilities in your web applications, improving the overall security posture of your software development process.

15.Application Server (WebGoat)

- WebGoat is a deliberately insecure web application designed to help developers and security professionals learn about web application security vulnerabilities and how to identify and mitigate them. It was created by the Open Web Application Security Project (OWASP) as a way to provide a hands-on learning experience for those interested in web application security.
- WebGoat is a Java-based web application that runs on a web server, and it contains a variety of vulnerabilities that are commonly found in web applications, such as SQL injection, cross-site scripting (XSS), and broken authentication and session management. Users can interact with the application and attempt to exploit the vulnerabilities, and the application provides feedback on the effectiveness of their attacks and how to address the vulnerabilities.
- WebGoat is open source and freely available for download, and it is widely used as a tool for training and education in the field of web application security. It is often used in combination with other tools and resources provided by OWASP, such as the OWASP Top 10, to help developers and security professionals improve the security of their web applications.




Not secure | 13.235.134.242:8080/WebGoat/start.mvc#lesson/WebGoatIntroduction.lesson
WEBGOAT
WebGoat
Introduction
General
(A1) Injection
(A2) Broken Authentication
(A3) Sensitive Data Exposure
(A4) XML External Entities (XXE)
(A5) Broken Access Control
(A7) Cross-Site Scripting (XSS)
(A8) Insecure Deserialization
(A9) Vulnerable Components
(A8:2013) Request Forgeries
Client side
Challenges
Reset lesson
1

What is WebGoat?

WebGoat is a deliberately insecure application that allows interested developers just like you to *test vulnerabilities* commonly found in Java-based applications that use common and popular open source components.

Now, while we in no way condone causing intentional harm to any animal, goat or otherwise, we think learning everything you can about security vulnerabilities is essential to understanding just what happens when even a small bit of unintended code gets into your applications.

What better way to do that than with your very own scapegoat?

Feel free to do what you will with him. Hack, poke, prod and if it makes you feel better, scare him until your heart's content. Go ahead, and hack the goat. We promise he likes it.

Thanks for your interest!

The WebGoat Team

- When it comes to implementing the Secure Infrastructure framework in SecOps, the choice of application server can depend on various factors such as the specific needs of your organization, the technology stack being used, and the security requirements of your application.
- In general, the application server used should support secure coding practices and offer robust security features that align with the SSDLC framework. Some of the key security features to look for in an application server include:
- Role-based access control: The ability to assign different roles and permissions to users based on their level of access.
- Transport Layer Security (TLS) support: The ability to encrypt data in transit using SSL/TLS protocols to prevent eavesdropping, tampering, and other security threats.
- Authentication and authorization: The ability to authenticate users and authorize access to sensitive resources based on predefined rules and policies.
- Audit logging and monitoring: The ability to track user activities, monitor system events, and generate audit logs for compliance and forensic analysis.
- Hardening and patch management: The ability to harden the application server by applying security patches, updating software components, and disabling unnecessary services to reduce the attack surface.
- Some popular application servers that support this type of Infrastructure framework and offer robust security features include WebGoat Server, Apache Tomcat, Microsoft IIS, Nginx, and Oracle WebLogic Server. It's important to evaluate each server's features, strengths, and weaknesses to determine the best fit for your organization's needs.

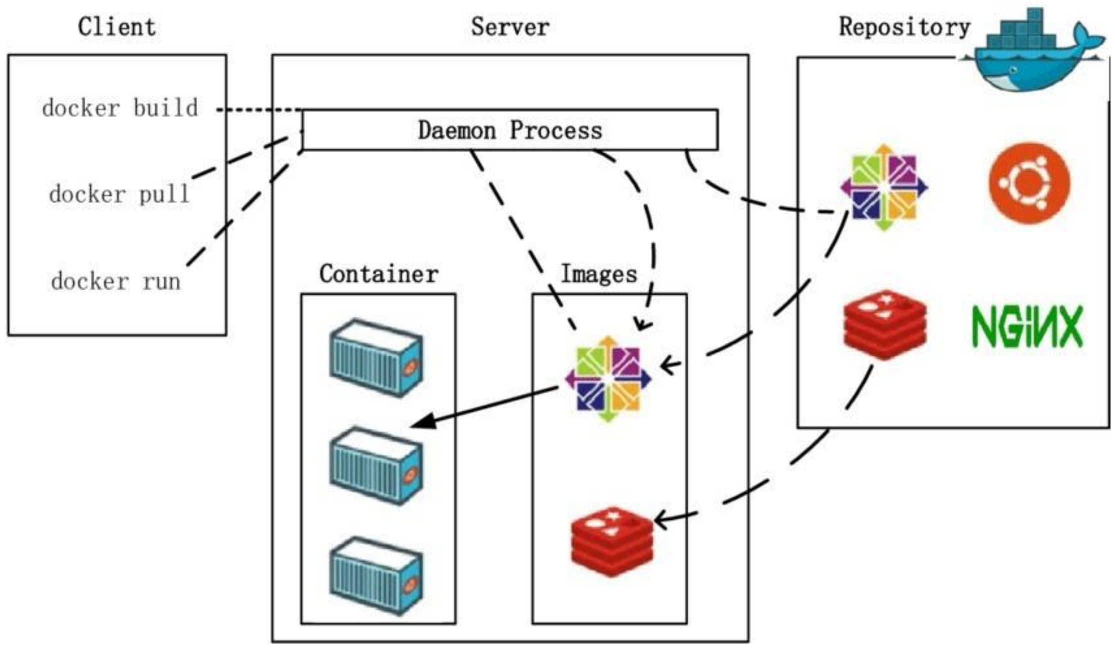
16. Docker Configuration

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers are lightweight and portable units that can run on any machine with Docker installed, providing aconsistent environment for applications to run in.

Here are the basic steps to configure Docker on your machine:

1. Install Docker: Docker provides installers for different operating systems. You can download the appropriate installer for your operating system from the Docker website and follow the instructions toinstall it on your machine.
2. Verify the installation: Once you have installed Docker, you can verify the installation by running thedocker --version command in your terminal or command prompt. This command should return theversion number of Docker that you have installed.
3. Create a Dockerfile: A Dockerfile is a text file that contains instructions to build a Docker image. Youcan create a Dockerfile in the root directory of your application by specifying the base image, copying files into the image, and running any commands to set up your application.
4. Build the Docker image: Once you have created a Dockerfile, you can use the docker build command to build the Docker image. The basic syntax of the command is:

docker build -t <image name>:<tag>



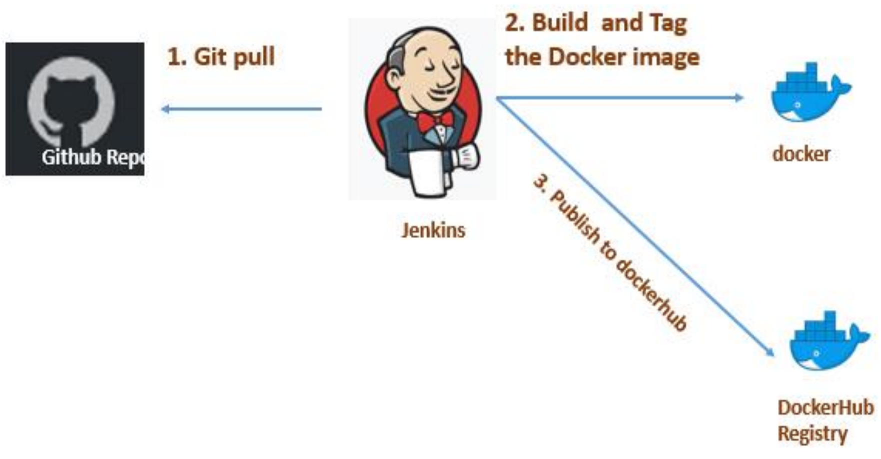
➢ Images and containers

A container is launched by running an image. An image is an executable package that includeseverything needed to run an application--the code, a runtime, libraries, environment variables, and configuration files. A container is a runtime instance of an image--what the image becomes in memory when executed (that is, an image with state, or a user process). You can see a list of your running containers with the command, docker pc, just as you would in Linux.

After you run a docker image, it creates a docker container. All the applications and their environmentrun inside this container. You can use Docker API or CLI to start, stop, delete a docker container.

#sudo apt-get install docker*

#docker pull owasp/zap2docker-stable



17. AWS CloudFormation

- AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles that.

- For a scalable web application that also includes a backend database, you might use an Auto Scaling group, an Elastic Load Balancing load balancer, and an Amazon Relational Database Service database instance. You might useeach individual service to provision these resources and after you create the resources, you would have to configure them to work together. All these tasks can add complexity and time before you even get your application up and running.

- Instead, you can create a CloudFormation template or modify an existing one. A template describes all your resources and their properties. When you use that template to create a CloudFormation stack, CloudFormation provisions the Auto Scaling group, load balancer, and database for you. After the stack has been successfully created, your AWS resources are up and running. You can delete the stack just as easily, which deletes all the resources in the stack. By using CloudFormation, you easily manage a collection of resources as a single unit.

- If your application requires additional availability, you might replicate it in multiple regions so that if one region becomes unavailable, your users can still use your application in other regions. The challenge in replicating your application is that it also requires you to replicate your resources. Not only do you need to record all the resources that your application requires, but you must also provision and configure those resources in each region.
- Reuse your CloudFormation template to create your resources in a consistent and repeatable manner. To reuse your template, describe your resources once and then provision the same resources over and over in multiple regions.

- When you provision your infrastructure with CloudFormation, the CloudFormation template describes exactly what resources are provisioned and their settings.

Because these templates are text files, you simply track differences in your templates to track changes to your infrastructure, similar to the way developers control revisions to source code. For example, you can use a version control system with your templates so that you know exactly what changes were made, who made them, and when. If at any point you need to reverse changes to your infrastructure, you can use a previous version of your template.

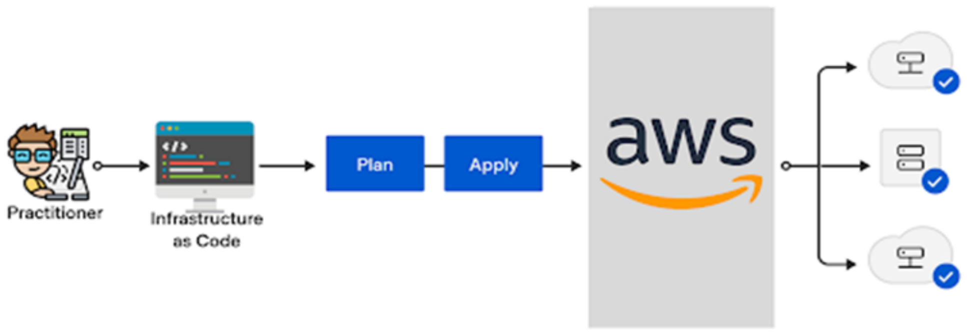
- In cloud formation, we use Yamal code through which it automatically creates application server instance with the help of stack.

➢ **Code for the cloud formation**

```
AWSTemplateFormatVersion: 2010-09-09
Description: Part 1 - Build a webapp stack with CloudFormation

Resources:
  WebAppInstance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0f5ee92e2d63afc18 # ImageID valid only in us-east-1 region
      InstanceType: t2.micro
      KeyName: dast-key
      SecurityGroupIds:
        - !Ref WebAppSecurityGroup

  WebAppSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: !Join ["-", [webapp-security-group, dev1]]
      GroupDescription: "Allow HTTP/HTTPS and SSH inbound and outbound
traffic"
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 443
          ToPort: 443
          CidrIp: 0.0.0.0/0
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0

WebAppEIP:
    Type: AWS::EC2::EIP
    Properties:
      Domain: vpc
      InstanceId: !Ref WebAppInstance
      Tags:
        - Key: Name
          Value: !Join ["-", [webapp-eip, dev1]]

Outputs:
  WebsiteURL:
    Value: !Sub http://${WebAppEIP}
    Description: WebApp URL
```

➢ How Does CloudFormation Work?

CloudFormation provides users with a simple and automated way to model and provision AWS and third-party resources. It enables users to create templates in YAML or JSON formats that define the desired infrastructure configuration. The template is then sent to an AWS CloudFormation stack, which deploys the resources specified in the template. CloudFormation is a highly reliable service that ensures the integrity of the infrastructure. It also provides version control, rollback, and other features to ensure that changes made to the infrastructure are done safely and consistently.



➢ Benefits of CloudFormation

AWS CloudFormation offers a number of benefits for organizations that are looking to automate the process of provisioning and managing their cloud resources. By using CloudFormation, organizations can increase their visibility into their infrastructure and dramatically reduce the time it takes to deploy and manage their cloud resources. Furthermore, CloudFormation is cost-effective and easy to use, allowing organizations to make the most of their investments in hardware and software. Additionally, CloudFormation integrates with other AWS services, making it easy to manage multiple services in a unified way. With CloudFormation, organizations can quickly and easily launch and configure their cloud resources with minimal effort.

➢ Jenkins file configuration

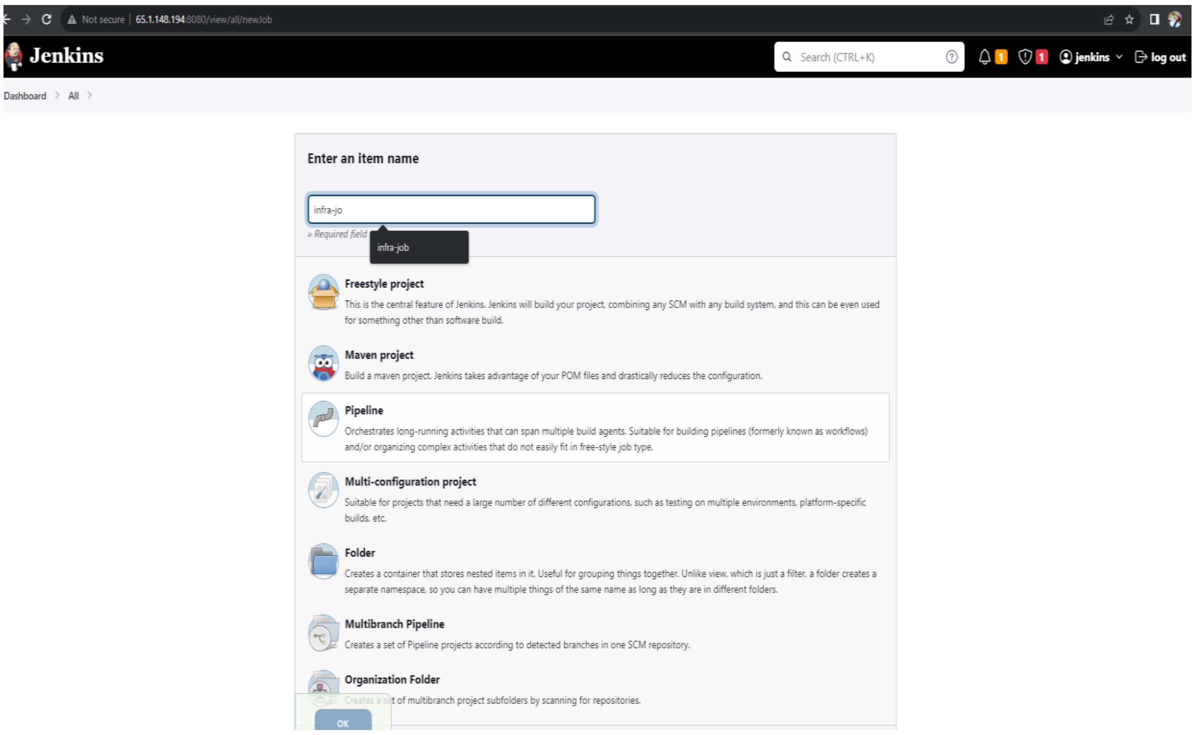
Jenkins is an open-source automation server that is used to automate various parts of the software development process. It was originally created as a fork of the Hudson project in 2011 and has sincebecome one of the most widely used automation servers.

Jenkins provides a wide range of plugins that can be used to automate tasks like building, testing, and deploying software. It integrates with a variety of development tools and systems, such as Git, SVN, Maven, and Gradle, and can be used for continuous integration and continuous delivery (CI/CD) pipelines.

Jenkins works by creating jobs, which are sequences of steps that define how to build, test, and deploysoftware. These jobs can be scheduled to run at regular intervals or triggered by events like code changes or commits to a version control system. Jenkins provides a web-based interface for managingjobs and viewing the status of builds and tests.

Jenkins is highly customizable and extensible, and its large ecosystem of plugins and integrations makesit a popular choice for automation and CI/CD pipelines in the software development industry.

- Create a pipeline



Enter an item name

infra-jo

» Required field

infra-job

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

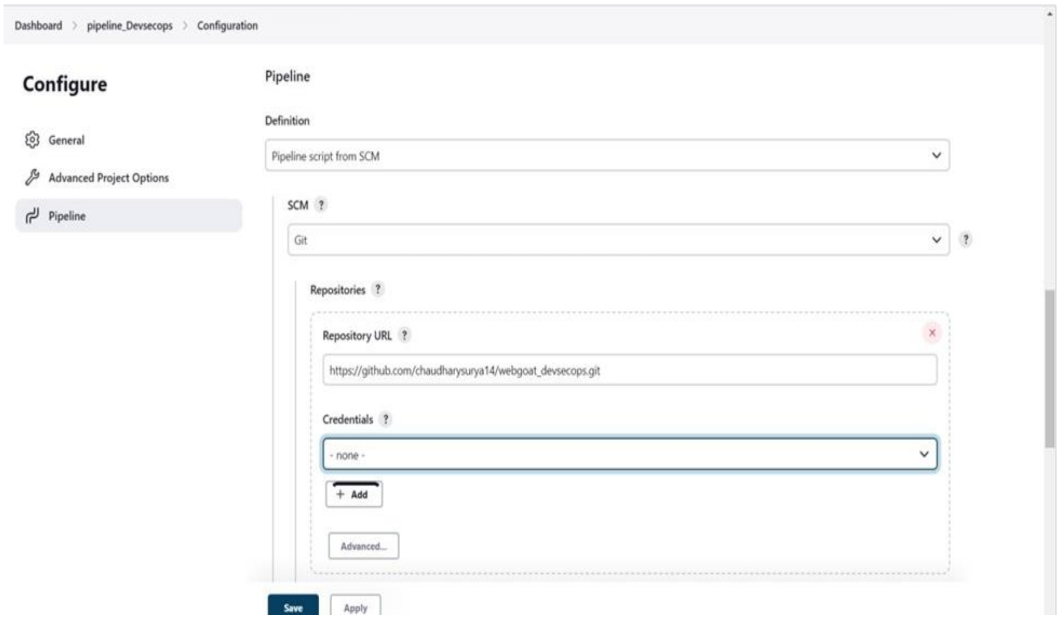
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

- Their configuration



Dashboard > pipeline_Devsecops > Configuration

Configure

Pipeline

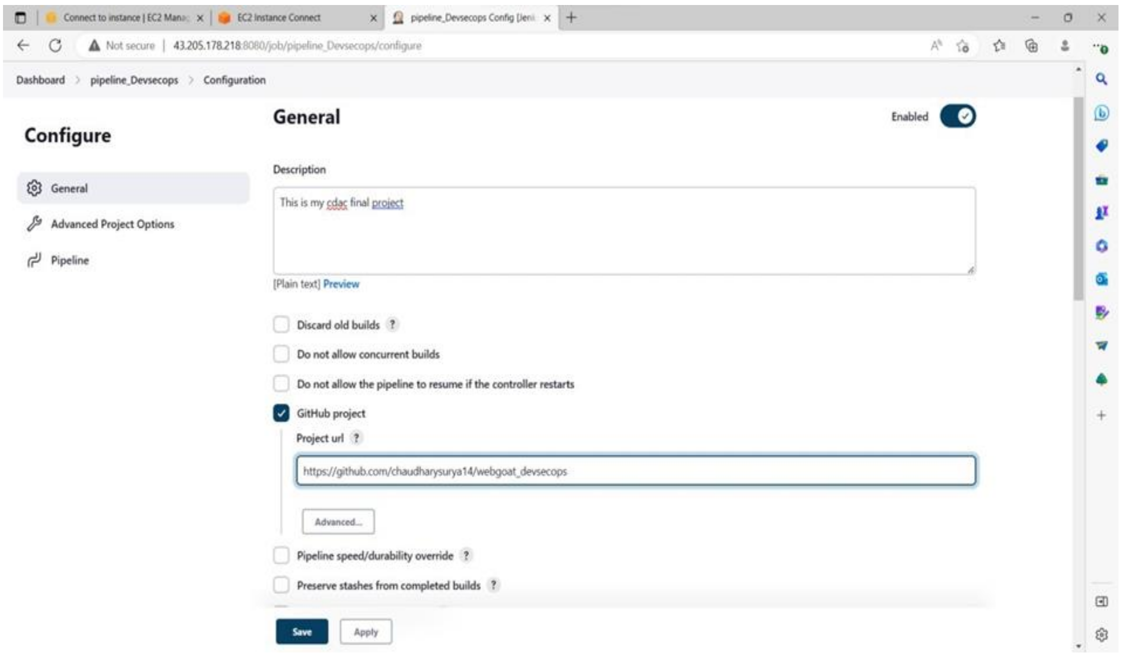
Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/chaudharysurya14/webgoat_devsecops.git

Credentials: - none -



43.205.178.218:8080/job/pipeline_Devsecops/configure

Dashboard > pipeline_Devsecops > Configuration

Configure

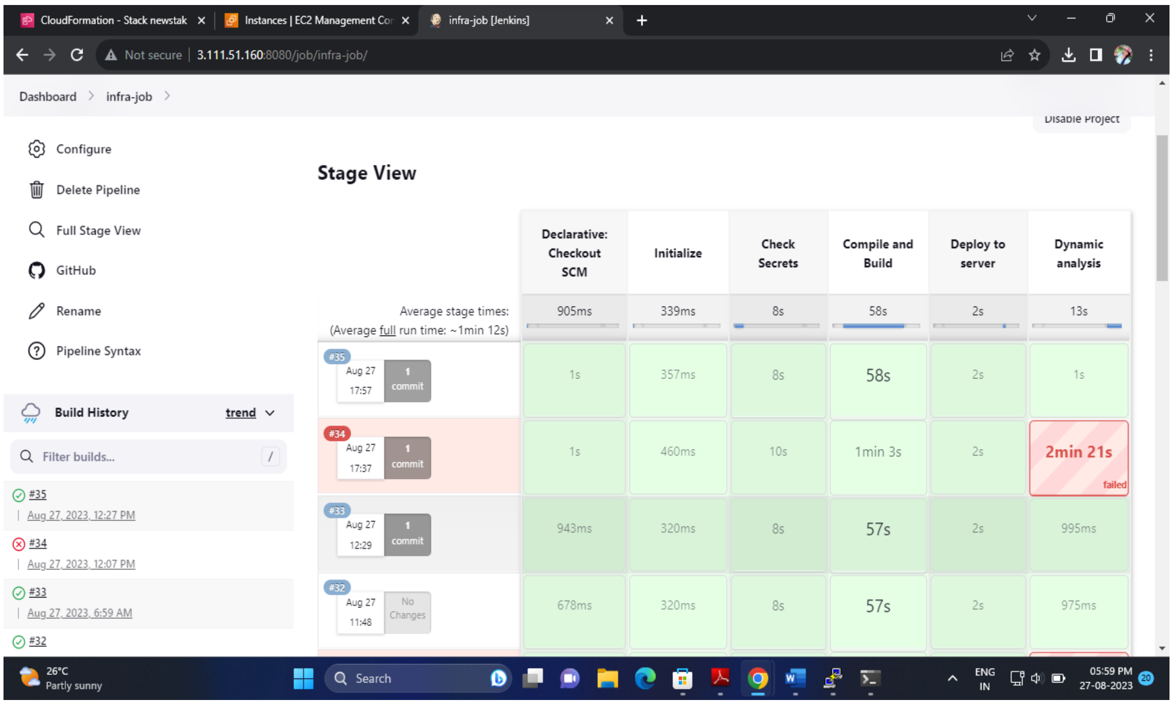
General — Enabled

Description: This is my cdac final project

[Plain text] Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project
  - Project url: https://github.com/chaudharysurya14/webgoat_devsecops
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds

- **CI/CD Pipeline Build**



Dashboard > infra-job >

Disable Project

Configure
Delete Pipeline
Full Stage View
GitHub
Rename
Pipeline Syntax

Build History trend

Filter builds...

✓ #35 Aug 27, 2023, 12:27 PM
✗ #34 Aug 27, 2023, 12:07 PM
✓ #33 Aug 27, 2023, 6:59 AM
✓ #32

Stage View

	Declarative: Checkout SCM	Initialize	Check Secrets	Compile and Build	Deploy to server	Dynamic analysis
Average stage times: (Average full run time: ~1min 12s)	905ms	339ms	8s	58s	2s	13s
#35 Aug 27 17:57 1 commit	1s	357ms	8s	58s	2s	1s
#34 Aug 27 17:37 1 commit	1s	460ms	10s	1min 3s	2s	2min 21s failed
#33 Aug 27 12:29 1 commit	943ms	320ms	8s	57s	2s	995ms
#32 Aug 27 11:48 No Changes	678ms	320ms	8s	57s	2s	975ms

18. DAST

- In the context of a Jenkins pipeline, "DAST" typically refers to "Dynamic Application Security Testing." DAST is a type of security testing that focuses on identifying vulnerabilities in running web applications by sending various inputs to the application and analyzing its responses. It is used to find security vulnerabilities that could potentially be exploited by attackers.

- In a Jenkins pipeline, you can integrate DAST tools and processes to automate security testing as part of your continuous integration and continuous delivery (CI/CD) pipeline. Here's how DAST can be useful in a Jenkins pipeline:

- Automated Security Testing: Including DAST as a step in your Jenkins pipeline allows you to automatically run security tests against your application after each code change. This helps identify security vulnerabilities early in the development process.

- Early Detection of Vulnerabilities: By integrating DAST into your pipeline, you can detect security vulnerabilities that might not be apparent during code reviews or static analysis. DAST simulates real-world attacks and helps identify issues that might only be visible in a running application.

- Immediate Feedback to Developers: When security vulnerabilities are identified by DAST, your pipeline can provide immediate feedback to developers, enabling them to address the issues promptly before the code progresses further.

- Shift Left Security: DAST in a Jenkins pipeline supports the "shift left" security approach, which aims to integrate security practices earlier in the software development lifecycle. This reduces the cost and effort of fixing vulnerabilities at later stages.

- Continuous Security: Just like other automated tests, DAST tests can be run as part of every build, ensuring that security is continuously monitored and maintained as the application evolves.

- Customizable and Configurable: DAST tools often come with customizable settings to match your application's specific behavior. You can configure the tool to scan different parts of the application and provide the desired level of security coverage.

- Complements Other Security Testing: DAST complements other security testing methods like SAST (Static Application Security Testing) and manual penetration testing. Each method focuses on different aspects of security and together provide a more comprehensive view of your application's security posture.

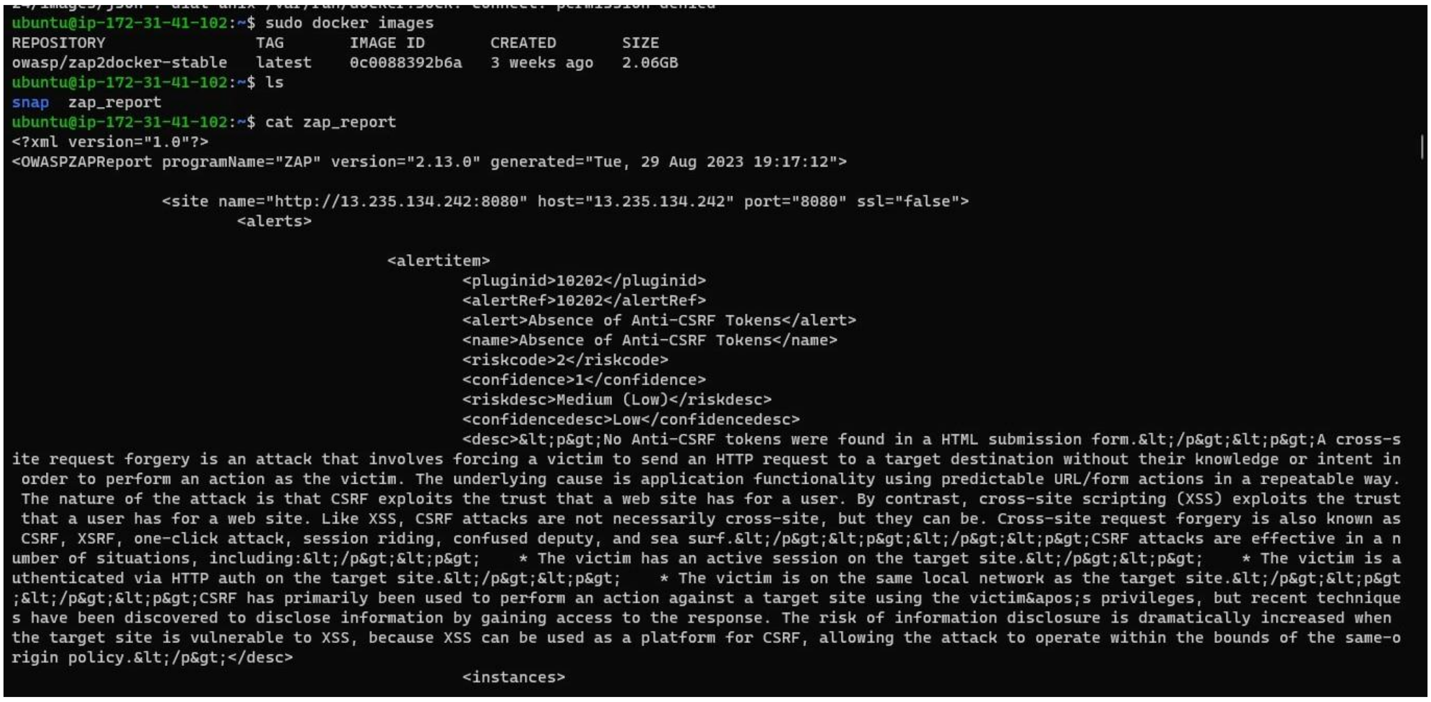
- Examples of DAST tools that can be integrated into Jenkins pipelines include OWASP ZAP (Zed Attack Proxy), Burp Suite, and Acunetix. The specifics of how you integrate DAST into your Jenkins pipeline depend on the DAST tool you're using and the steps required to run tests against your application

➢ Code

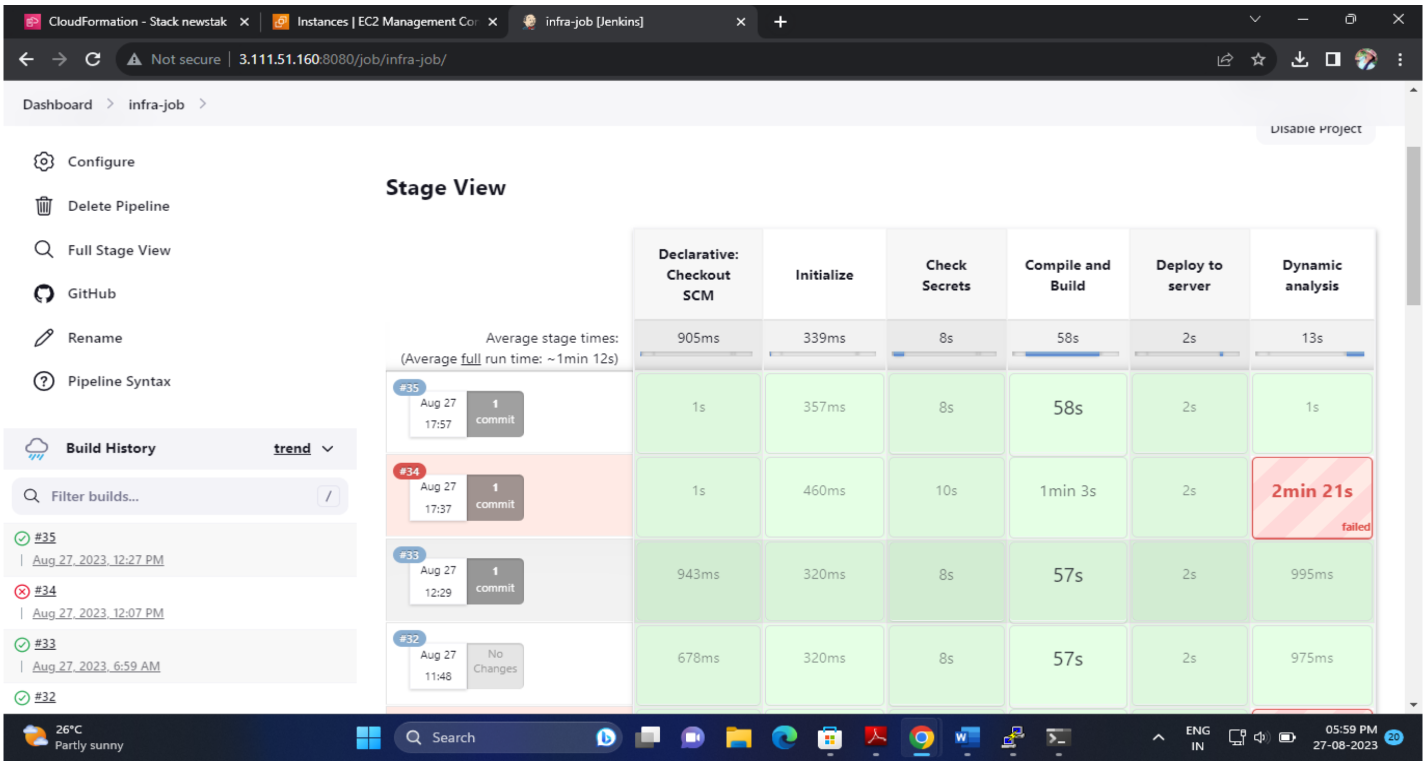
```
pipeline {
        agent any
        stages {
            stage ('Initialize') {
                     steps {
                         sh '''
                         echo "PATH = ${PATH}"
                                 echo "M2_HOME = ${M2_HOME}"
                         '''
                         }
                 }
        stage ('Check Secrets') {
                 steps {
            sh 'trufflehog3 https://github.com/kaleanushka123/Infra-secops.git -f json -o truffelhog_output.json || 
true'

                         }
                 }

        stage('Compile and Build'){
                 steps{
                         sh 'mvn clean install -DskipTests'
                         }
                 }

        stage ('Deploy to server') {
        steps {
             timeout(time: 4, unit: 'MINUTES') {
              sshagent(['web-server']) {
    sh 'scp -o StrictHostKeyChecking=no /var/lib/jenkins/workspace/infra-job/target/webgoat-2023.5-
SNAPSHOT.jar app3@13.126.140.19:WebGoat'
sh 'ssh -o  StrictHostKeyChecking=no app3@13.126.140.19 "nohup java -jar /home/app3/WebGoat/webgoat-
2023.5-SNAPSHOT.jar --server.address=13.126.140.19 --server.port=8080 &"'
                 }
             }
           }
        }
         stage ('Dynamic analysis') {
        steps {
       sshagent(['dast-test']) {
     sh 'ssh -o  StrictHostKeyChecking=no ubuntu@13.234.225.115 "sudo docker run --rm -v 
/home/ubuntu:/zap/wrk/:rw -t owasp/zap2docker-stable zap-full-scan.py -t http://13.126.140.19:8080/WebGoat -x 
zap_report || true" '

             }
        }
      }
   }
}
```

19. Result

✓ Build completed Successfully



```
ubuntu@ip-172-31-41-102:~$ sudo docker images
REPOSITORY                 TAG       IMAGE ID       CREATED       SIZE
owasp/zap2docker-stable    latest    0c0088392b6a   3 weeks ago   2.06GB
ubuntu@ip-172-31-41-102:~$ ls
snap  zap_report
ubuntu@ip-172-31-41-102:~$ cat zap_report
<?xml version="1.0"?>
<OWASPZAPReport programName="ZAP" version="2.13.0" generated="Tue, 29 Aug 2023 19:17:12">

                <site name="http://13.235.134.242:8080" host="13.235.134.242" port="8080" ssl="false">
                        <alerts>

                                        <alertitem>
                                                <pluginid>10202</pluginid>
                                                <alertRef>10202</alertRef>
                                                <alert>Absence of Anti-CSRF Tokens</alert>
                                                <name>Absence of Anti-CSRF Tokens</name>
                                                <riskcode>2</riskcode>
                                                <confidence>1</confidence>
                                                <riskdesc>Medium (Low)</riskdesc>
                                                <confidencedesc>Low</confidencedesc>
                                                <desc>&lt;p&gt;No Anti-CSRF tokens were found in a HTML submission form.&lt;/p&gt;&lt;p&gt;A cross-s
ite request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in
 order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way.
 The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust
 that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as
 CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.&lt;/p&gt;&lt;p&gt;&lt;/p&gt;&lt;p&gt;CSRF attacks are effective in a n
umber of situations, including:&lt;/p&gt;&lt;p&gt;    * The victim has an active session on the target site.&lt;/p&gt;&lt;p&gt;    * The victim is a
uthenticated via HTTP auth on the target site.&lt;/p&gt;&lt;p&gt;    * The victim is on the same local network as the target site.&lt;/p&gt;&lt;p&gt
;&lt;/p&gt;&lt;p&gt;CSRF has primarily been used to perform an action against a target site using the victim&apos;s privileges, but recent technique
s have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when
the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-o
rigin policy.&lt;/p&gt;</desc>
                                                <instances>
```



Stage View

	Declarative: Checkout SCM	Initialize	Check Secrets	Compile and Build	Deploy to server	Dynamic analysis
Average stage times: (Average full run time: ~1min 12s)	905ms	339ms	8s	58s	2s	13s
#35 Aug 27 17:57 1 commit	1s	357ms	8s	58s	2s	1s
#34 Aug 27 17:37 1 commit	1s	460ms	10s	1min 3s	2s	2min 21s failed
#33 Aug 27 12:29 1 commit	943ms	320ms	8s	57s	2s	995ms
#32 Aug 27 11:48 No Changes	678ms	320ms	8s	57s	2s	975ms

✓ #35 Aug 27, 2023, 12:27 PM
✗ #34 Aug 27, 2023, 12:07 PM
✓ #33 Aug 27, 2023, 6:59 AM
✓ #32

20. Conclusion

In conclusion, utilizing Infrastructure as Code (IaC) through tools like AWS CloudFormation to deploy an application server within a Jenkins CI/CD pipeline brings several benefits to the development and deployment process. This approach enhances efficiency, scalability, and reproducibility by allowing you to define your infrastructure using code, thereby reducing manual errors and enabling consistent environments across different stages of development and deployment.

By integrating CloudFormation with Jenkins, you establish a streamlined pipeline that automates the provisioning of necessary resources, such as servers, networks, and databases. This enables faster development iterations, reliable testing environments, and efficient collaboration among teams, as infrastructure changes are version-controlled and can be easily tracked.

Moreover, combining IaC and CI/CD practices improves the traceability of changes and simplifies troubleshooting. If issues arise during deployment, you can pinpoint the exact changes made to the infrastructure code and quickly roll back to a previous working state.

However, it's important to note that while IaC brings numerous advantages, it also requires careful planning, expertise in coding infrastructure, and a solid understanding of the cloud provider's services and features. Ensuring proper testing and validation of your infrastructure code is crucial to prevent potential issues in production.

In the end, adopting IaC through CloudFormation within a Jenkins CI/CD pipeline empowers your development teams to efficiently manage and deploy application servers, promoting automation, consistency, and reliability throughout the software development lifecycle.

21. Reference

Here are some references that can be used for further reading on implementing an SSDLC framework in DevSecOps:

1."Secure DevOps: A Pragmatic Approach to Securing Software in the Cloud" by Julien Vehent, O'Reilly Media, 2018.
2. "Implementing DevSecOps: Security in the DevOps Lifecycle" by Yogesh Singh and Prabath Siriwardena, Apress, 2018.

3. "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations" by Gene Kim, Jez Humble, Patrick Debois, and John Willis, IT Revolution Press, 2016.

4. "OWASP DevSecOps Maturity Model" by OWASP, available at: OWASP Devsecops Maturity Model _ OWASP Foundation_files

5. "Continuous Security in DevOps" by Paula Thrasher, IEEE Security & Privacy, vol. 16, no. 5, pp. 68-71, 2018.

6. "DevSecOps: How to Seamlessly Integrate Security into DevOps" by Fahmida Y. Rashid, SD Times, available at: DevSecOps Archives - SD Times_files

7. "Securing DevOps: Security in the Cloud" by Julien Vehent, O'Reilly Media, 2017.

8. "Secure Software Development: A Comprehensive Guide to Developing Secure Software" by Mark Merkow and Lakshmikanth Raghavan, Auerbach Publications, 2012. SSH to jenkins.devsecops.lab