**Project Report on**

**Implementing SSDLC Framework in DevSecOps**



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 Gomi** Presented by:

# Mr. Suryadev Chaudhary PRN: 220940127056 Ms. Shital Jagtap PRN: 220940127007 Mr. Dhammdip Landge PRN: 220940127043 Mr. Sunil Gaikwad PRN: 220940127037

**Ms. Aishwarya PRN: 220940127002**

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



CERTIFICATE

# TO WHOMSOEVER IT MAY CONCERN

**This is to certify that**

# Mr. Suryadev Chaudhary Ms. Shital Jagtap

**Mr. Dhammdip Landge Mr. Sunil Gaikwad**

# Ms. Aishwarya

**have successfully completed their project on**

**Implementing SSDLC Framework in DevSecOps**

**Under the Guidance of Mr. Roshan Gomi**

**Project Guide Project Supervisor**

# HOD ACTS

**Mr. Aditya Sinha**

PG-DHPCSA

**ACKNOWLEDGEMENT**

This project “Implementing SSDLC Framework in DevSecOps

**”**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 Gomi for his valuable guidance to work on this project. His guidance and support helped us to overcomevarious 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.

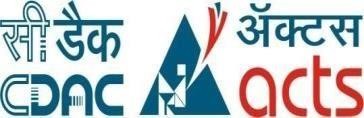
Our most heartfelt thank goes to Ms. Swati salunkhe (Course Coordinator, PG- DHPCSA) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day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

### From:

**Mr. Suryadev Chaudhary (220940127056) Ms. Shital Jagtap (220940127007)**

### Mr. Dhammdip Landge (220940127043) Mr. Sunil Gaikwad (220940127037)

**Ms. Aishwarya ( 220940127002)**



**TABLE OF CONTENTS**

1. Abstract………………………………………………………………………

1. Introduction and Overview of Project……………………………………….
2. Implementing SSDLC Framework in DevSecOps………………………….

4. System Requirement………………………………………………………...

4.1 Software Requirement…………………………………………………..

4.2 Hardware Requirement………………………………………………….

4.3 User Interface…………………………………………………………...

5. System Architecture………………………………………………………...

5.1 Data Flow Diagram…………………………………………………….

5.2 Activity Diagram……………………………………………………….

6. Introduction to Repository………………………………………………….

7. Git Concepts………………………………………………………………..

8. Jenkin server configuration…………………………………………………

9. Docker Configuration………………………………………………………

10. Pulling Sonarqube image………………………………………………….

11. OWASP Installation………………………………………………………

12. Maven……………………………………………………………………..

13. Webgoat installation………………………………………………………

14. Jenkins files configuration……………………………………………….

15. Test Plan………………………………………………………………….

16. Application server………………………………………………………..

17. Results……………………………………………………………………

18. Conclusion………………………………………………………………..

19. References………………………………………………………………..

**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 Secure Software Development Lifecycle (SSDLC) framework 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 will be 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 that it 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 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.

# Introduction and overview of project

This project aims to implement the SSDLC 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.

# Implementing SSDLC Framework in DevSecOps

SSDLC (Secure Software Development Lifecycle) is a framework for integrating security into the software development process. DevSecOps is a methodology that combines development, security, and operations into a single continuous process. By implementing SSDLC in DevSecOps, you can ensure that security is built into every stage of the software development process.

Here are the steps to implement SSDLC in DevSecOps:

1. Plan: In this stage, you need to define the project scope, objectives, and requirements. You also need to identify potential security risks and define security requirements.
2. Design: In this stage, you need to design the architecture, components, and interfaces of the software system. You also need to define security controls, such as access control, authentication, and encryption.
3. Develop: In this stage, you need to develop the software 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. Test: In this stage, you need to conduct functional and non-functional testing to ensure that the software system meets the requirements and is secure.
5. Deploy: In this stage, you need to deploy the software system in a secure manner. You also need to ensure that the deployment environment meets the security requirements.
6. Operate: In this stage, you need to monitor the software system for security incidents and vulnerabilities. You also need to perform security maintenance, such as patching and updating.
7. Maintain: In this stage, you need to maintain the software system by fixing security vulnerabilities and updating security controls. You also need to ensure that the software system remains secure over time.

By following these steps, you can implement SSDLC in DevSecOps and ensure that security is built into every stage of the software development process.

### System Requirement

* 1. **User Interface**

1.EC2 Instance Ubuntu

### Software Requirement

1. GIT
2. Jenkins 3.Docker 4.Sonarqubes 5.OWASP ZAP
3. Dependency-check 8.1.0
   1. Apache Maven version- 3.6.3 7.Trufflehog

8.webgoat

### 4.3 Hardware Requirement

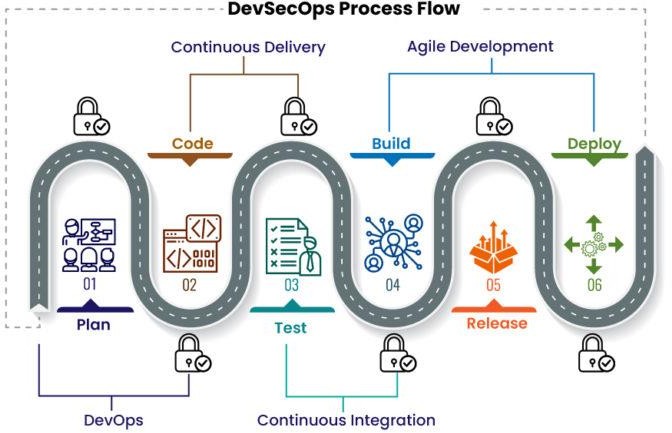
1. Windows 10 or 11
2. Ubuntu 30GB HD, 1GB RAM
3. t2-micro medium 4GBRAM

**Used Language**

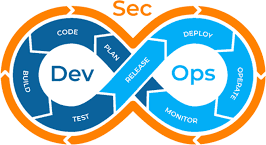
1. Python script

### System Architecture

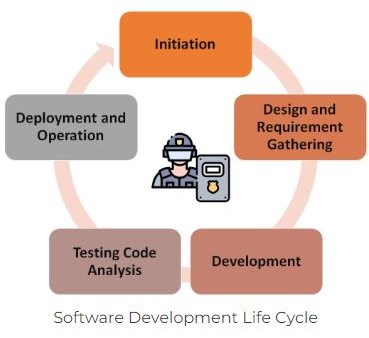
* 1. **Data Flow Diagram**

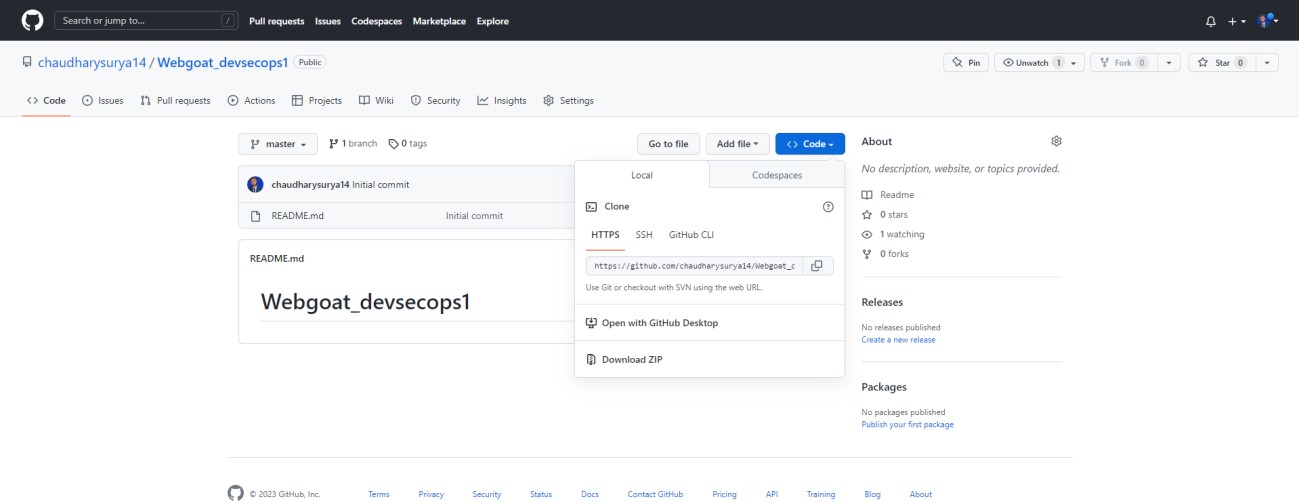


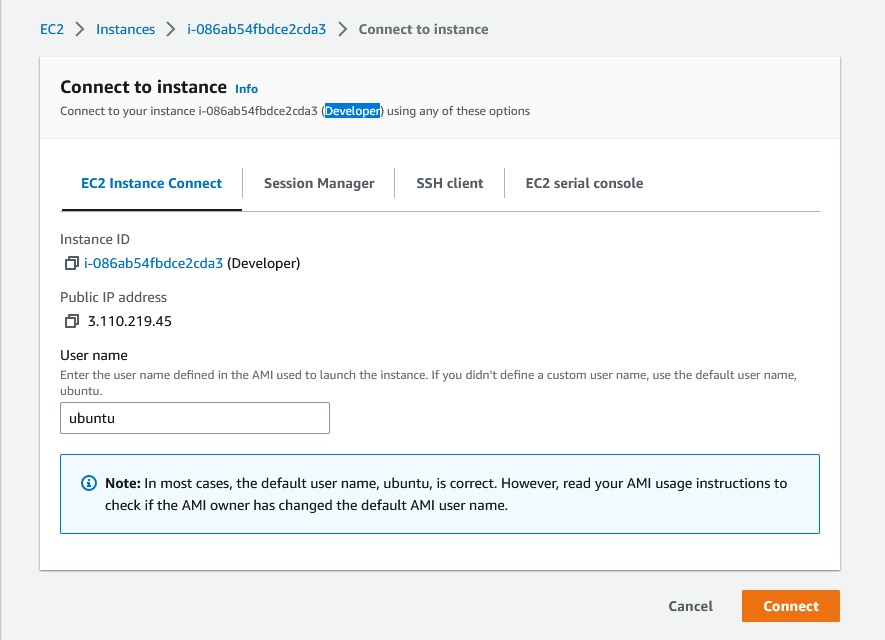
### Activity Diagram



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



1. Introduction to Repository



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

### 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 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

### 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.

# Project setup with CI CD Pipeline:

## Pre-requisites:

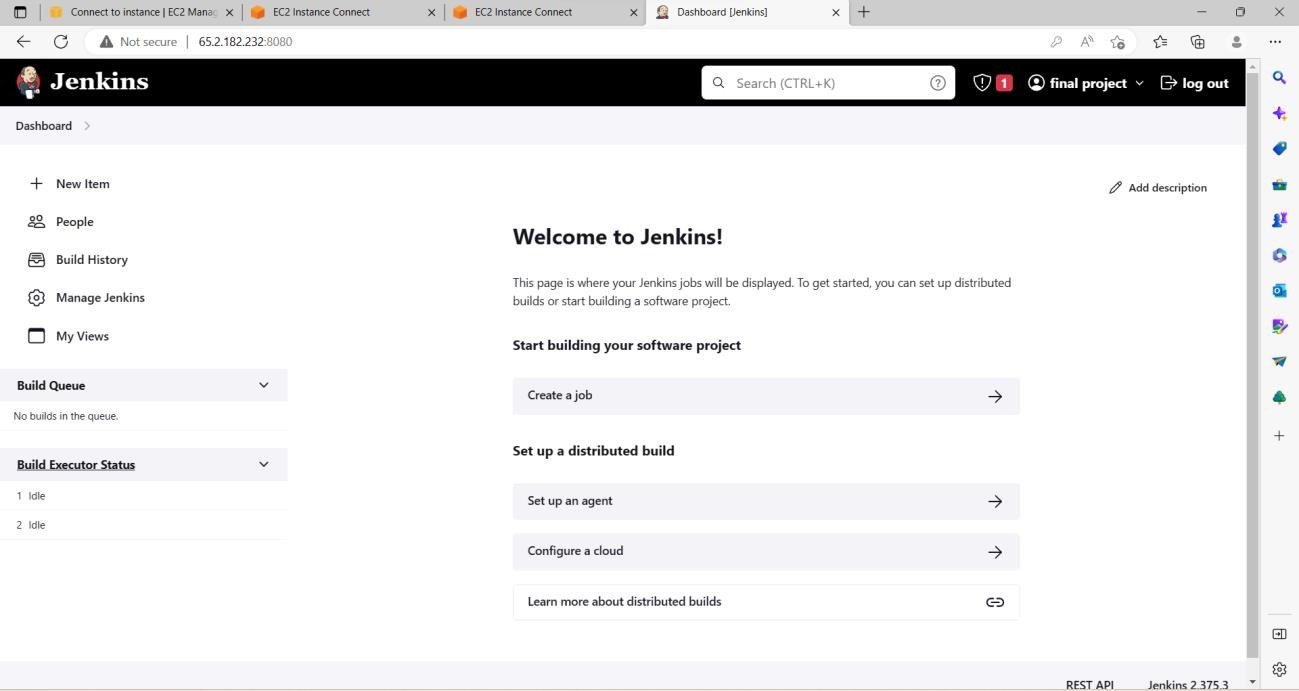
### 1) GitHub

**( Repo url:** https://github.com/chaudharysurya14/Webgoat\_devsecops1.git**)**

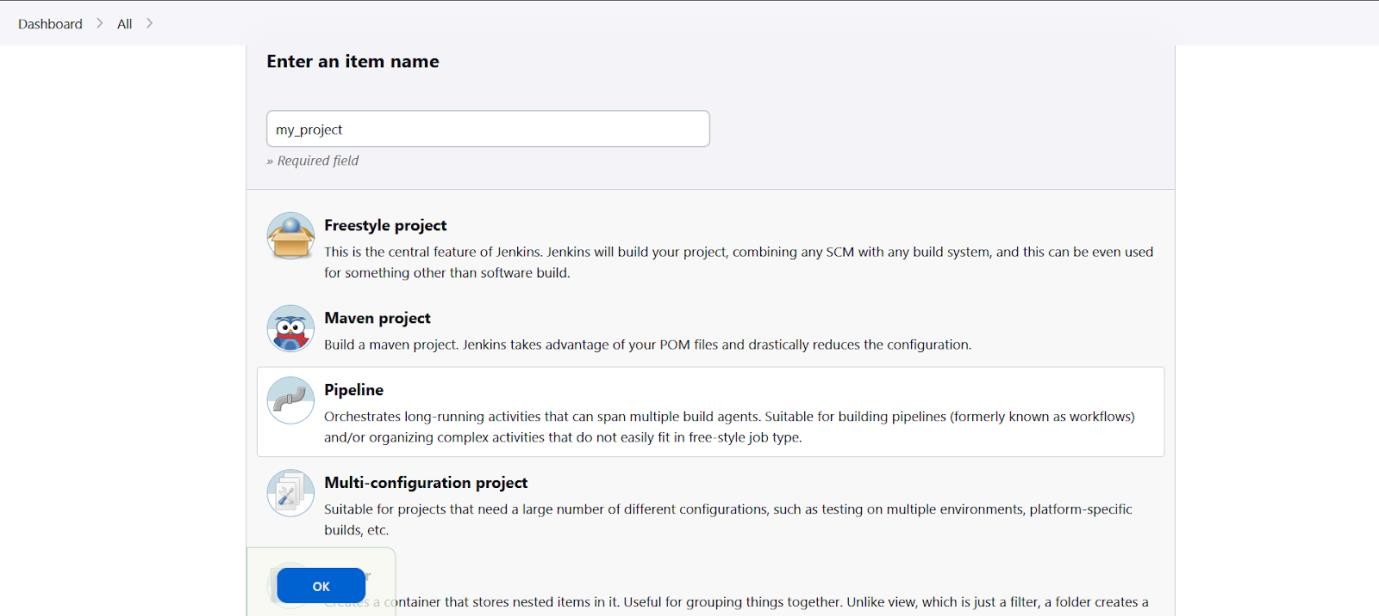
1. **SonarQube (url:**http://public-ip:9000 **)**
2. **Jenkins Server (url:**http://public-ip:8080 **)**
3. **Maven (**as a global tool in Jenkins**)**

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

install jenkin



jenkin id configuration



### Docker Configuration

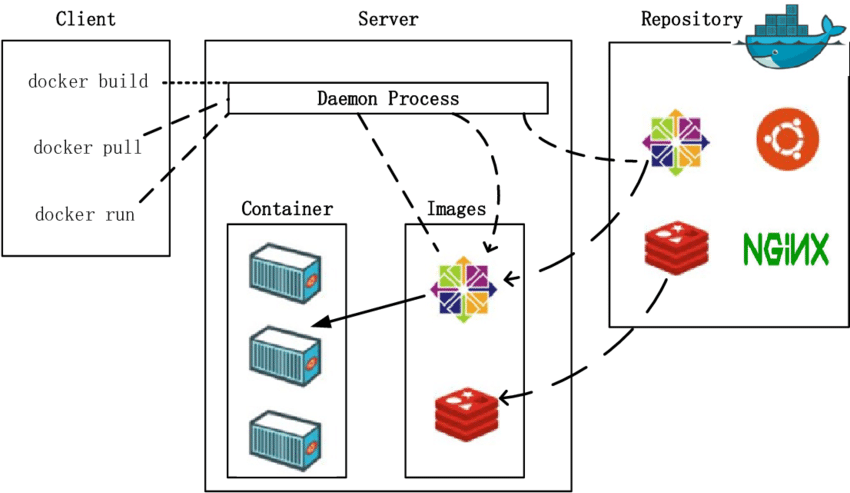
Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers are lightweight and portable units that can run on any machine with Docker installed, providing a consistent environment for applications to run in.

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 to install it on your machine.
2. Verify the installation: Once you have installed Docker, you can verify the installation by running the docker --version command in your terminal or command prompt. This command should return the version 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. You can 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 includes everything 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 environment run inside this container. You can use Docker API or CLI to start, stop, delete a docker container.

### Pulling Sonarqube image

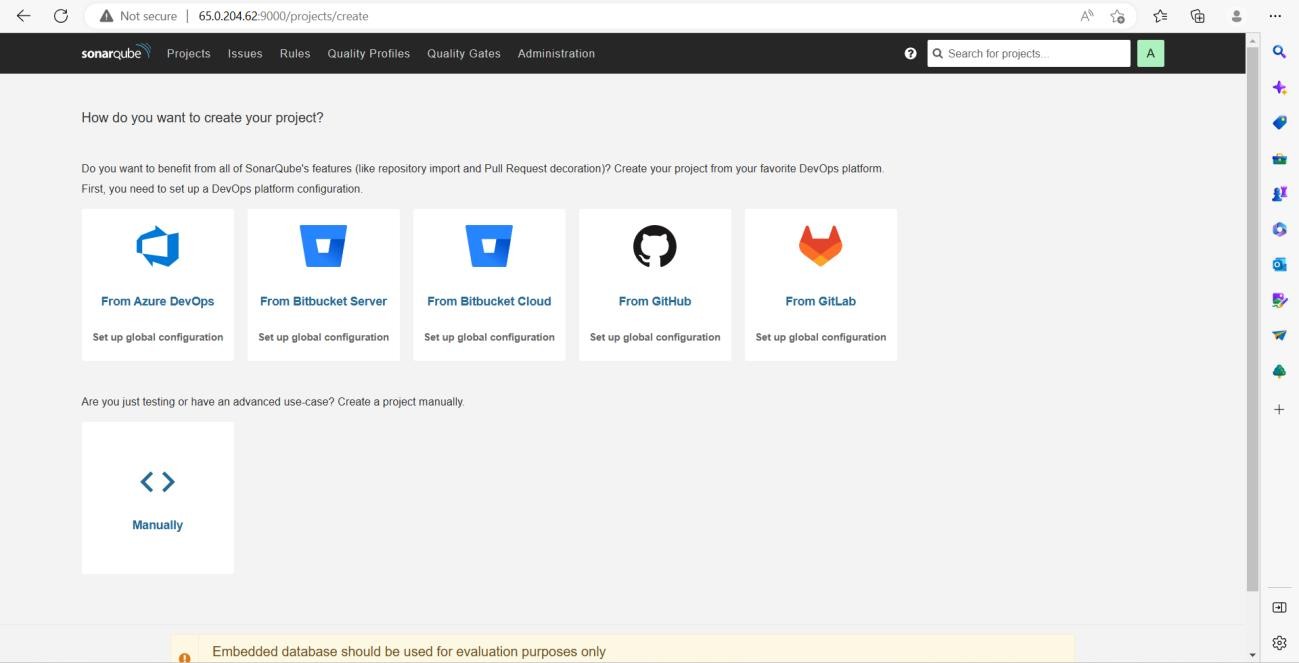
SonarQube is an open-source static code analysis tool used to measure the quality of source code. It provides a comprehensive set of code quality metrics and helps to identify issues and vulnerabilities in the code. It supports a variety of programming languages, including Java, C++, C#, Python, and JavaScript.

SonarQube analyzes the code and generates a detailed report on various code quality metrics, such as code coverage, code duplication, complexity, security vulnerabilities, and coding standards violations. It also provides a visual representation of the code quality metrics, allowing developers and teams to easily identify areas that need improvement.

In addition to analyzing the code, SonarQube integrates with popular CI/CD tools like Jenkins, GitLab, and GitHub, allowing developers to automatically trigger code analysis and receive feedback on code quality as part of their development workflow.

Overall, SonarQube is a powerful tool for improving code quality, identifying and addressing vulnerabilities, and ensuring the overall health of software projects.

Create docker container using sonarqube image Sonarqube installation complete



### OWASP Installation

OWASP stands for the Open Web Application Security Project. It is a non-profit organization that aims to 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 applications.

### Download 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.

### Trufflehog

Trufflehog 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.

Trufflehog 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. Trufflehog can be run locally or as part of a continuous integration/continuous deployment (CI/CD) pipeline.

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

Overall, Trufflehog 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.

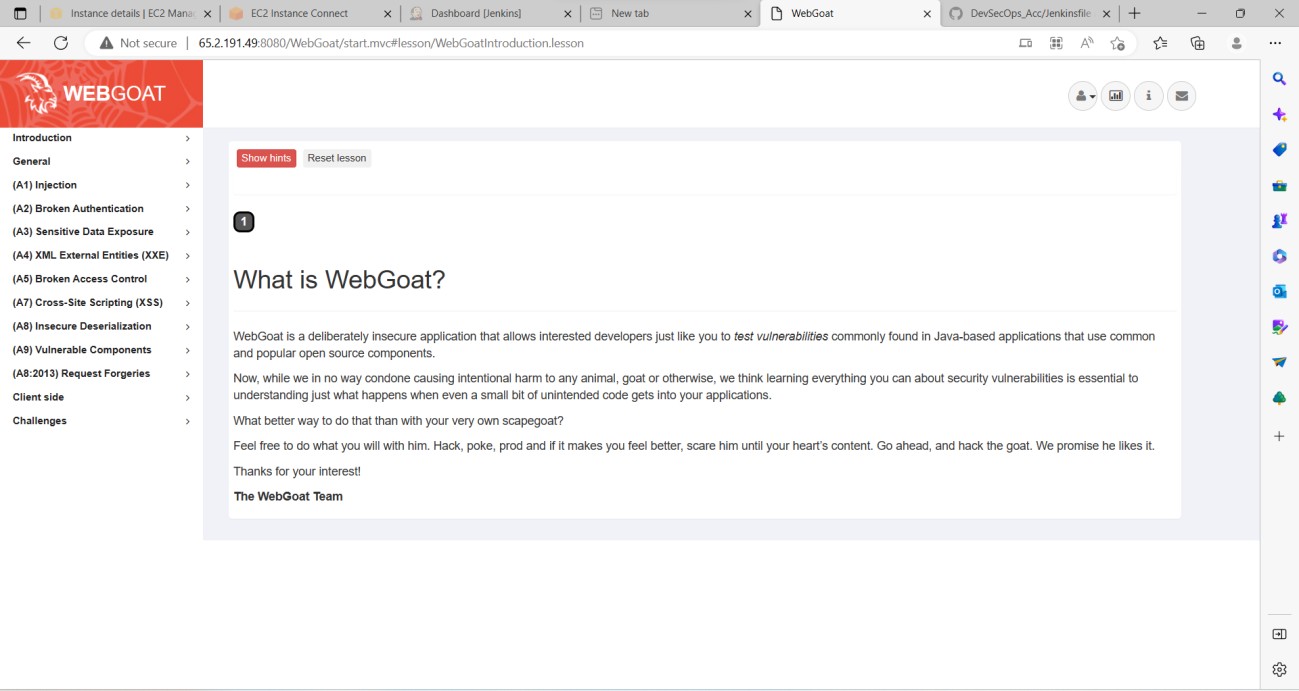
### Webgoat installation

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.

Webgoat dashboard



### Jenkins files configuration

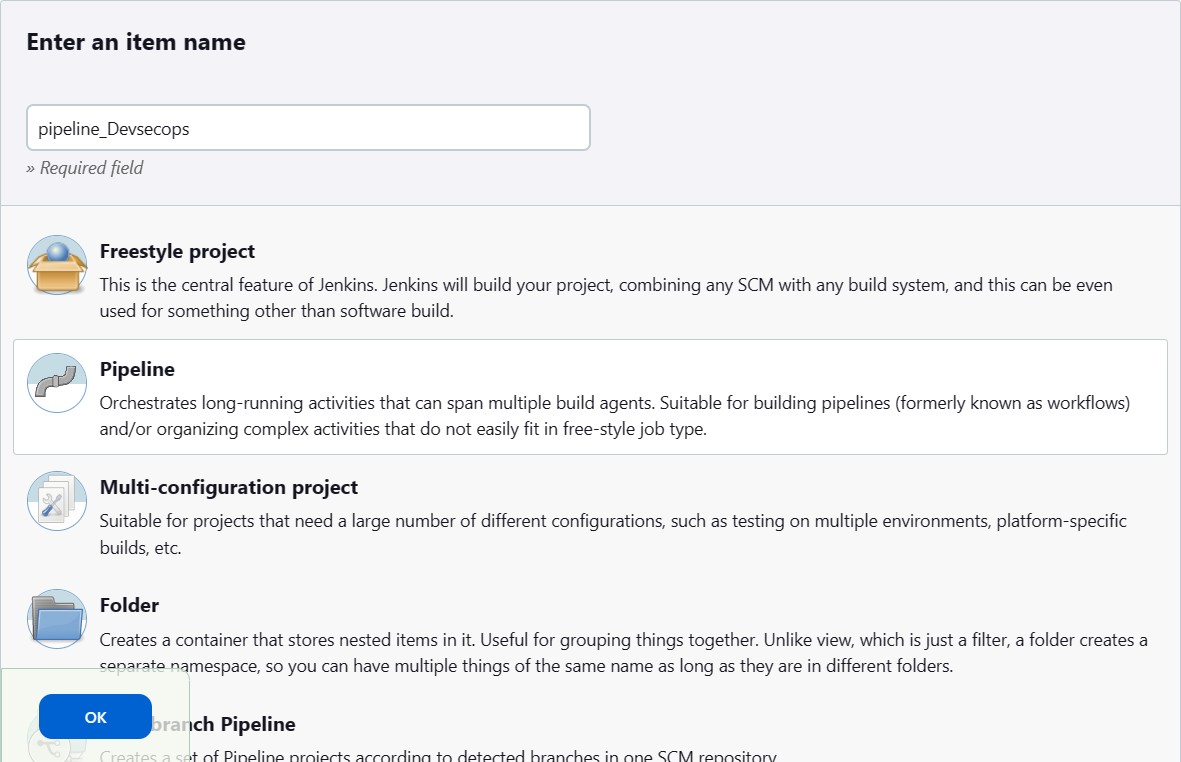
Jenkins is an open-source automation server that is used to automate various parts of the software development process. It was originally created as a fork of the Hudson project in 2011 and has since become one of the most widely used automation servers.

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.

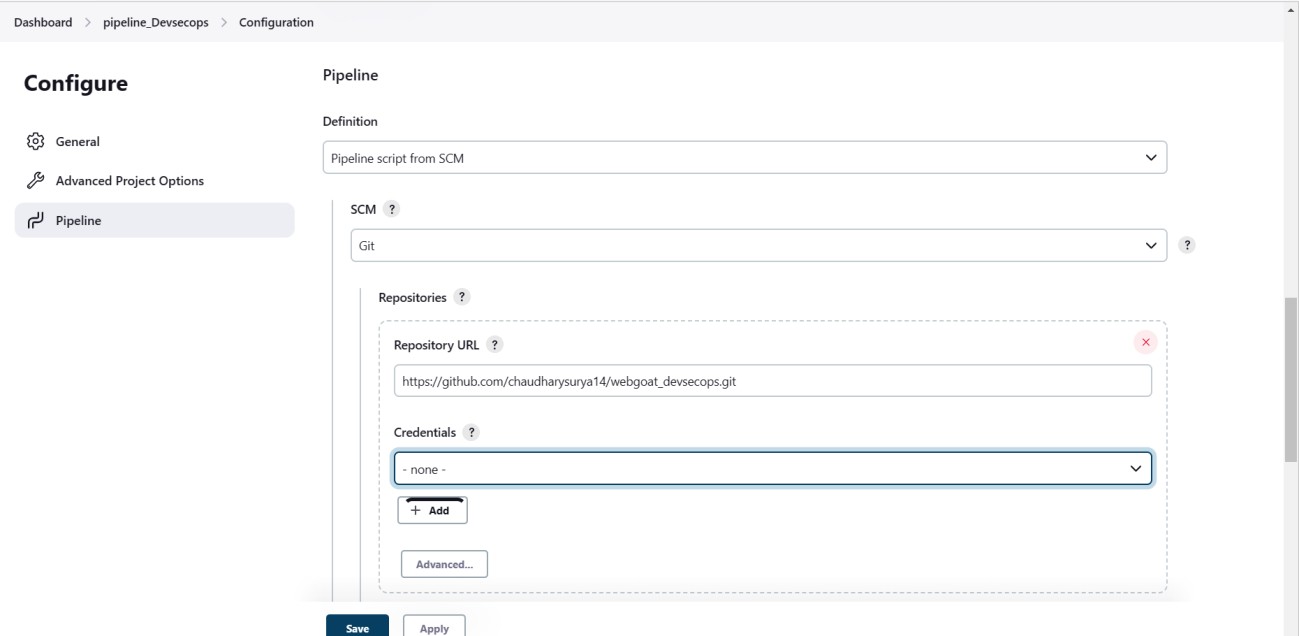
Jenkins works by creating jobs, which are sequences of steps that define how to build, test, and deploy software. These jobs can be scheduled to run at regular intervals or triggered by events like code changes or commits to a version control system. Jenkins provides a web-based interface for managing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

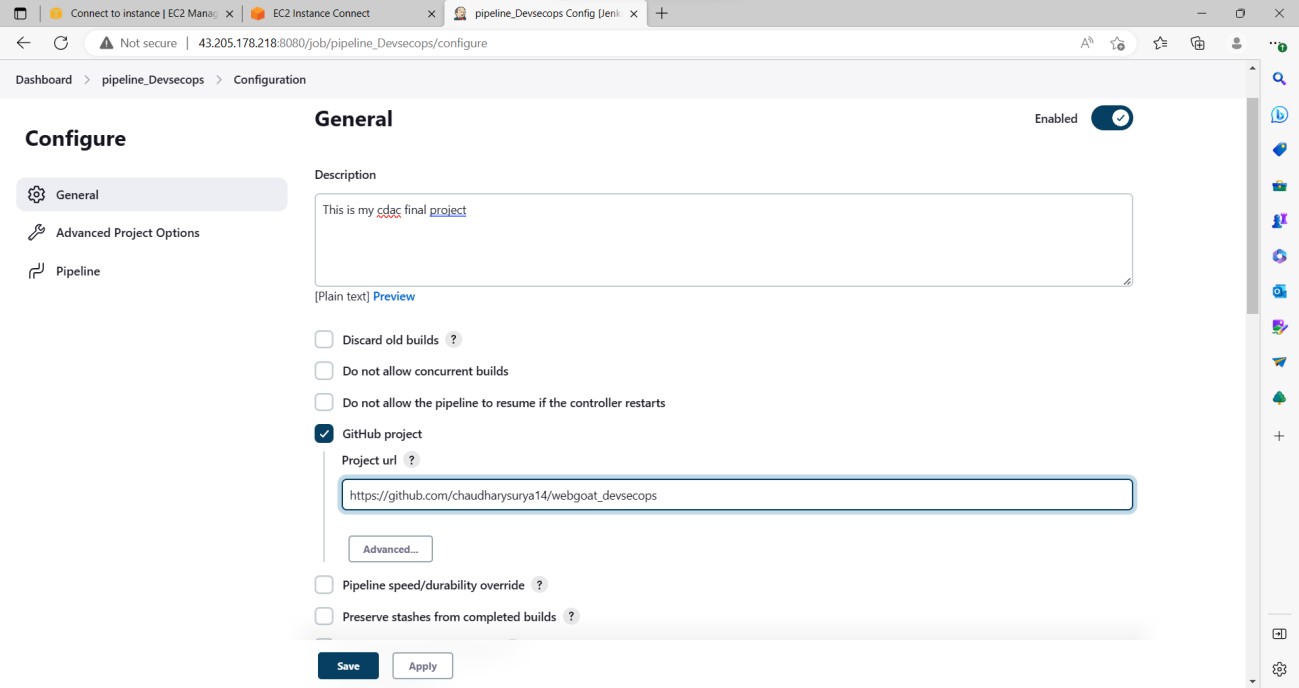
Jenkins is highly customizable and extensible, and its large ecosystem of plugins and integrations makes it a popular choice for automation and CI/CD pipelines in the software development industry.

Create a pipeline



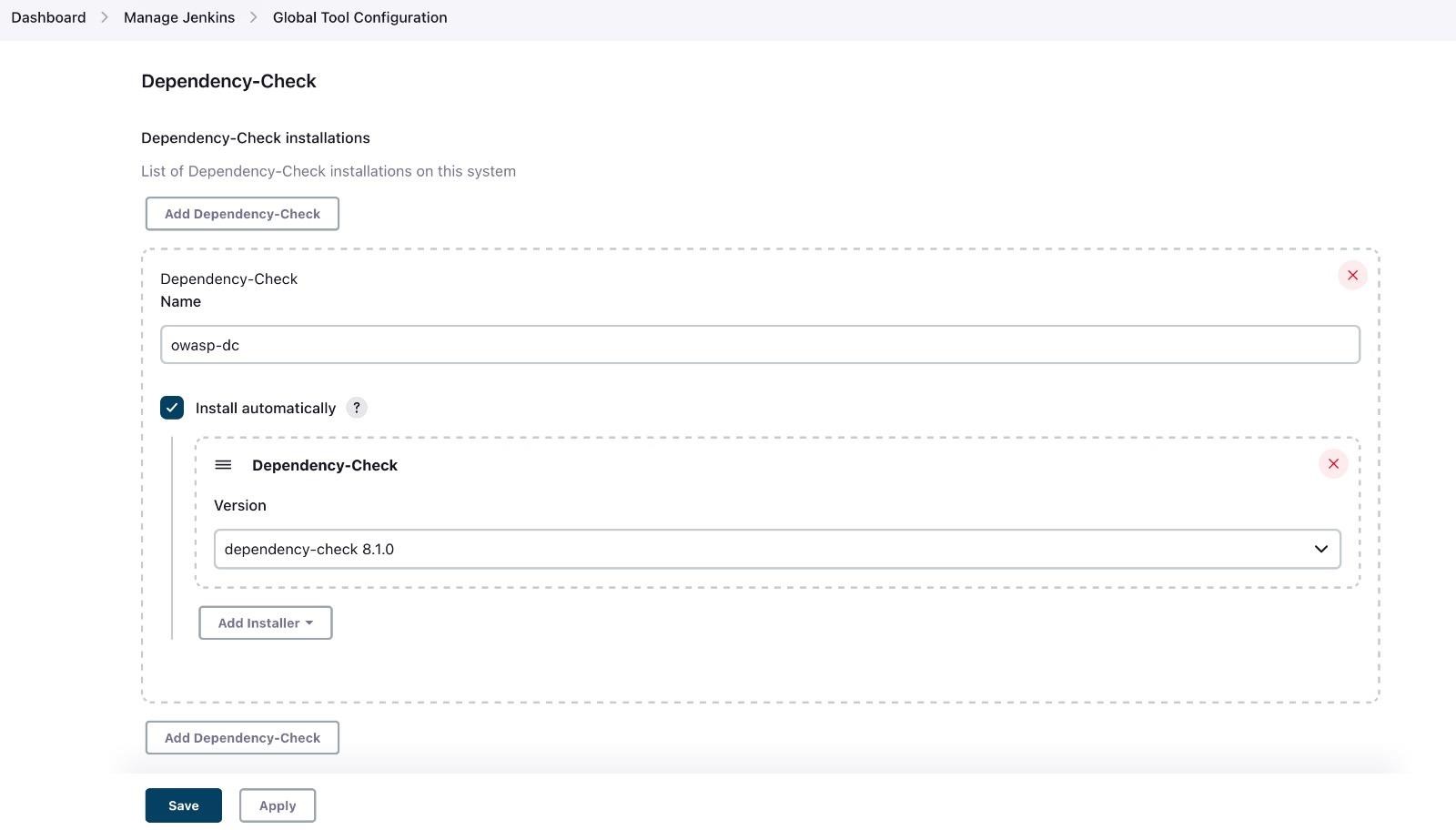
Their configuration



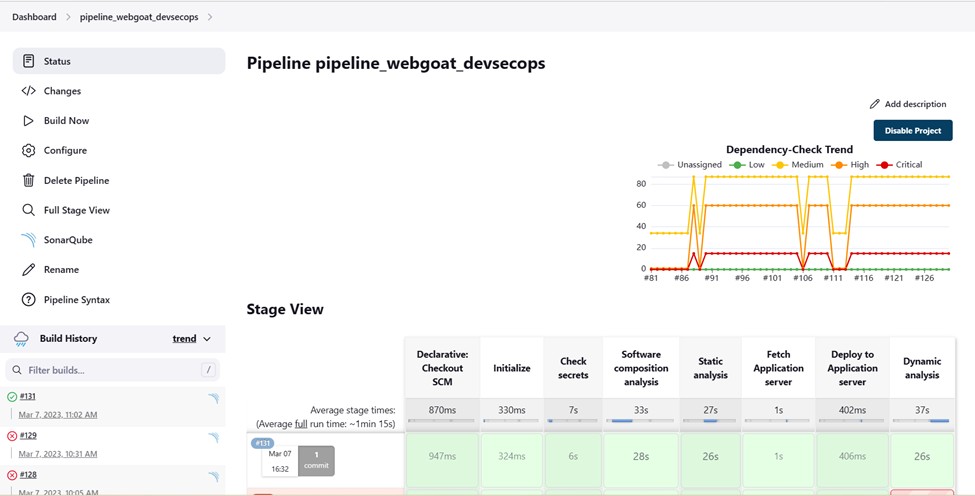


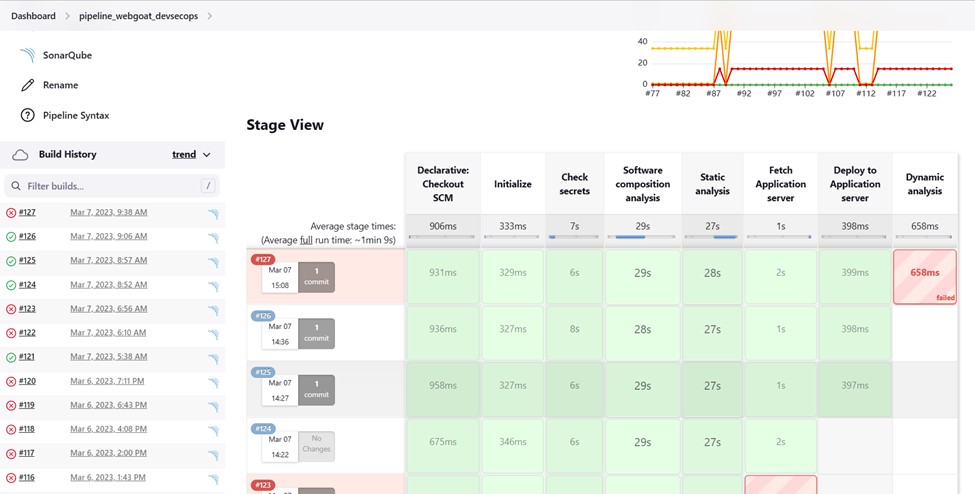
Dependency checkers are important tools that help in identifying vulnerabilities in software dependencies. Integrating a dependency checker into your Secure Software Development Life Cycle (SSDLC) framework is a good practice that can help prevent security breaches and ensure that your software is secure. Here are some popular dependency checkers that you can consider for your DevSecOps pipeline:

OWASP Dependency-Check: OWASP Dependency-Check is a popular open-source dependency checker that can scan a wide range of software dependencies and identify known vulnerabilities. It supports various package managers and programming languages, including Java, .NET, Ruby, and Python. It can be integrated into your DevSecOps pipeline using plugins for popular CI/CD tools like Jenkins, GitLab CI, and Travis CI.



CI/CD Pipeline Build





* 1. Test Plan

Implementing the Secure Software Development Lifecycle (SSDLC) framework in a DevSecOps environment requires a well-designed test plan to ensure that all aspects of the development process are covered. Here are some key elements that should be included in a test plan for implementing the SSDLC framework in DevSecOps:

1. Define the Objectives: The first step is to define the objectives of the SSDLC framework implementation. This should include identifying the specific security risks that need to be addressed, the security goals that the framework aims to achieve, and the timeline for implementing the framework.
2. Identify the Stakeholders: Identify the stakeholders who will be impacted by the implementation of the SSDLC framework. This may include the development team, operations team, security team, management team, and any other relevant stakeholders.
3. Assess the Current Process: Conduct an assessment of the current development process to identify the gaps and areas that need improvement. This may include reviewing the current tools and processes used, as well as identifying any security vulnerabilities in the system.
4. Define the SSDLC Process: Define the SSDLC process and document the steps involved in each phase. This should include defining the security requirements, threat modeling, security testing, and other relevant activities.
5. Develop Test Cases: Develop test cases that cover each phase of the SSDLC process. This should include functional and security testing, as well as testing for performance and scalability.
6. Plan for Automation: Plan for automation of the SSDLC process where possible. This may include using automation tools to identify security vulnerabilities, testing for performance, and automating the deployment process.
7. Define Metrics: Define metrics to measure the success of the SSDLC framework implementation. This should include metrics for security, performance, and other relevant areas.
8. Conduct Testing: Conduct testing of the SSDLC process to ensure that it is effective and meets the defined objectives. This should include testing for each phase of the SSDLC process and verifying that the metrics are being met.
9. Review and Refine: Review the results of the testing and refine the SSDLC process as needed. This should include identifying areas for improvement and making adjustments to the process as necessary.
10. Document and Communicate: Document the SSDLC process and communicate it to all stakeholders. This should include creating documentation that outlines the steps involved in the SSDLC process and communicating the results of the testing to relevant stakeholders.

By following these steps, you can develop a comprehensive test plan for implementing the SSDLC framework in a DevSecOps environment.

* 1. Application server

When it comes to implementing the Secure Software Development Life Cycle (SSDLC) framework in DevSecOps, 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 the SSDLC framework and offer robust security features include 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.

### 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/chaudharysurya14/Webgoat\_devsecops1.git -f json -o truffelhog\_output.json || true'

}

}

stage ('Software composition analysis') { steps {

dependencyCheck additionalArguments: '''

-o "./"

-s "./"

-f "ALL"

--prettyPrint''', odcInstallation: 'DP-Check'

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

}

}

stage ('Static analysis') { steps {

withSonarQubeEnv('SonarQube') { sh 'mvn sonar:sonar'

//sh 'sudo python3 Devsecops.py'

}

}

}

stage ('Fetch Application server') { steps { sshagent(['application\_server']) {

sh 'scp -o StrictHostKeyChecking=no /var/lib/jenkins/workspace/pipeline\_webgoat\_devsecops/target/webgoat- server-v8.2.0.jar ubuntu@13.235.24.37:~/WebGoat'

}

}

}

stage ('Deploy to Application server') { steps { sshagent(['application\_server']) {

sh 'ssh -o StrictHostKeyChecking=no [ubuntu@13.235.24.37](mailto:ubuntu@13.235.24.37) "nohup java -jar /WebGoat/webgoat-server- v8.2.0.jar --server.address=0.0.0.0 --server.port=8081 &"'

}

}

}

stage ('Dynamic analysis') { steps {

sshagent(['application\_server']) {

sh 'ssh -o StrictHostKeyChecking=no [ubuntu@13.234.136.161](mailto:ubuntu@13.234.136.161) "sudo docker run --rm -v

/home/ubuntu:/zap/wrk/:rw -t owasp/zap2docker-stable zap-full-scan.py -t http://13.235.24.37:8081/WebGoat -x zap\_report || true" '

}

}

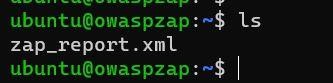
}

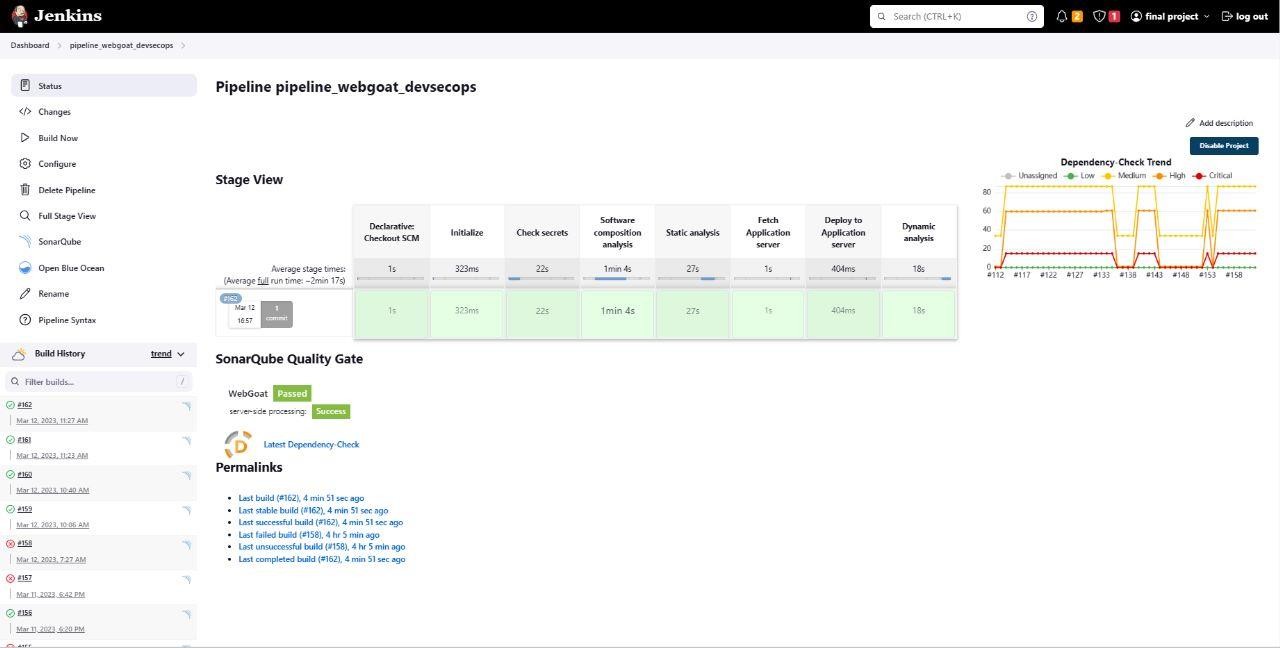
}

}

# Result

Build completed Successfully





# Conclusion

In conclusion, implementing a Secure Software Development Life Cycle (SSDLC) framework is crucial for successful DevSecOps implementation. SSDLC helps organizations to integrate security throughout the entire software development process, from planning to deployment. By doing so, security vulnerabilities and risks can be identified early in the development cycle, reducing the likelihood of costly security incidents later on.

DevSecOps focuses on a collaborative approach to software development, bringing together development, operations, and security teams to work towards a common goal. Implementing an SSDLC framework within DevSecOps provides a structured approach to security, enabling teams to work together more effectively and efficiently.

Key elements of an SSDLC framework include threat modeling, code analysis, testing, and continuous monitoring. By integrating these processes into the development cycle, teams can identify and address security vulnerabilities early in the process and reduce the risk of security incidents.

Implementing an SSDLC framework within DevSecOps requires a culture shift towards security as a shared responsibility across teams, a commitment to continuous improvement, and a focus on collaboration and communication. With the right tools, processes, and mindset, organizations can build secure, high-quality software that meets the needs of their customers and protects their data and assets.

At this point your SSDLC project with DevSecOps is being configured successfully.

### References.

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.

1. "Implementing DevSecOps: Security in the DevOps Lifecycle" by Yogesh Singh and Prabath Siriwardena, Apress, 2018.
2. "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.
3. "OWASP DevSecOps Maturity Model" by OWASP, available at: OWASP Devsecops Maturity Model \_ OWASP Foundation\_files
4. "Continuous Security in DevOps" by Paula Thrasher, IEEE Security & Privacy, vol. 16, no. 5, pp. 68-71, 2018.
5. "DevSecOps: How to Seamlessly Integrate Security into DevOps" by Fahmida Y. Rashid, SD Times, available at: DevSecOps Archives - SD Times\_files
6. "Securing DevOps: Security in the Cloud" by Julien Vehent, O'Reilly Media, 2017.
7. "Secure Software Development: A Comprehensive Guide to Developing Secure Software" by Mark Merkow and Lakshmikanth Raghavan, Auerbach Publications, 2012. SSH to jenkins.devsecops.lab

Install Java : C:\Users\Desktop\How To Install Java with Apt-Get on Ubuntu 16.04 \_ DigitalOcean\_files

Install Jenkins: https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Ubuntu

Install Docker: https://[www.digitalocean.com/community/tutorials/how-to-install-docker-with-apt-on-ubuntu-18-04](http://www.digitalocean.com/community/tutorials/how-to-install-docker-with-apt-on-ubuntu-18-04) Modify rule in Security Groups for port 8080

Activate Jenkins from /var/lib/Jenkins/secrets/initialAdminPassword

### sdbfjsdfj