

**Project Report**

**on**

**Vulnerability Assessment using  
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 Gami**

**Presented by:**

<b>Ms. Shradha Sonawane</b>	<b>PRN:230340127008</b>
<b>Mr. Vimeet K. Patil</b>	<b>PRN:230340127054</b>
<b>Ms. Prachi Nimgade</b>	<b>PRN:230340127043</b>
<b>Ms. Priyanka Kumari</b>	<b>PRN:230340127044</b>
<b>Ms. Saloni Bhole</b>	<b>PRN: 230340127002</b>
<b>Ms. Kranti Saknure</b>	<b>PRN: 230340127040</b>

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



# **CERTIFICATE**

**TO WHOMSOEVER IT MAY CONCERN**

**This is to certify that**

**Ms. Shradha Sonawane**

**Mr. Vimeet K. Patil**

**Ms. Prachi Nimgade**

**Ms. Priyanka Kumari**

**Ms. Saloni Bhole**

**Ms. Kranti Saknure**

**have successfully completed their project on**

## **Vulnerability Assessment using DevSecOps**

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

**Project Guide**

**Project Supervisor**

**HOD ACTS  
Mr. Aditya Sinha**

## ACKNOWLEDGEMENT

This project “**Vulnerability Assessment using 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 Gami** for his valuable guidance to work on this project. His guidance and support helped us to 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:

**Ms. Shradha Sonawane (230340127008)**

**Mr. Vimeet K. Patil (230340127054)**

**Ms. Prachi Nimgade (230340127043)**

**Ms. Priyanka Kumari (230340127042)**

**Ms. Saloni Bhole (230340127002)**

**Ms. Kranti Saknure (230340127040)**

## **TABLE OF CONTENTS**

1.	Abstract.....
2.	Introduction and Overview of Project.....
3.	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 Pipelines.....
9.	TruffleHog.....
10.	OWASP Dependency Check.....
11.	SonarQube.....
12.	Maven.....
13.	Tomcat.....
14.	OWASP ZAP.....
15.	Kubernetes.....
16.	Microk8s.....
17.	Implementation.....
18.	Testing.....
19.	Results.....
20.	CODE.....
21.	Conclusion.....

20. Reference.....

## **1. Abstract**

In the rapidly evolving landscape of software development, security vulnerabilities present a significant risk to organizations, their data, and their users. The emergence of DevSecOps, a symbiotic integration of Development, Security, and Operations practices, offers a comprehensive approach to addressing these vulnerabilities. This project delves into the realm of "Vulnerability Assessment using DevSecOps," where security is infused into every stage of the software development life cycle.

The project's primary objective is to establish an effective framework for identifying, mitigating, and managing vulnerabilities within a DevSecOps context. By merging security practices with continuous integration, continuous deployment (CI/CD), automation, and collaborative workflows, this project seeks to create an ecosystem that inherently prioritizes security.

The framework employs a range of automated tools and processes to scan for vulnerabilities in both code and dependencies, offering real-time feedback to development teams. Moreover, the project explores the integration of security policies into the pipeline, allowing for instantaneous checks and balances that prevent insecure code from progressing further. Through the lens of vulnerability assessment, this project unveils the power of "shifting left" – tackling security concerns early in the development cycle to prevent the perpetuation of vulnerabilities.

As a result of this project, organizations can anticipate reduced risk exposure, higher software quality, and more efficient development cycles. Security becomes an intrinsic component of the development process, transforming vulnerabilities from liabilities into opportunities for growth. By embracing Vulnerability Assessment using DevSecOps, organizations empower themselves to thrive in the face of an ever-evolving threat landscape, fortifying their software against vulnerabilities and safeguarding their digital assets.

## **2. Introduction and overview of project**

More and more companies are in the process of adopting modern continuous software development practices and approaches like continuous integration (CI), continuous delivery (CD), or DevOps. These approaches can support companies in order to increase the development speed, the frequency of product increments, and the time to market. To be able to get these advantages, especially the tooling and infrastructure need to be reliable and secure. In case CI/CD is compromised or even unavailable, all mentioned advantages are at stake. Potentially, this could also even hinder the forthcoming of the software development. Therefore, our goal was to identify which vulnerabilities are present in industry CI/CD pipelines and how they can be detected. In this project, we present our results of an industry case study which includes a qualitative survey of JAVA based project teams regarding the awareness of security in CI/CD, the analysis and abstraction of CI/CD pipelines, and a threat analysis based on the deducted CD pipeline to identify vulnerabilities. In this case study, we found that the team members that work with the CD pipeline in different roles do not have a strong security background but are aware of security attributes in general. Furthermore, CI/CD pipelines from industry projects were analyzed using the ZAP by OSAWP. In total, we identified 15 warnings that have been confirmed by the project teams.

### **3. Implementing SSDLC Framework in DevSecOps**

Vulnerability assessment within a DevSecOps framework involves integrating security practices into the entire software development life cycle to proactively identify and address vulnerabilities. Here's a concise flow of vulnerability assessment using DevSecOps:

DevSecOps integrates security into every phase of the software development life cycle, from planning and coding to testing and deployment. In the context of vulnerability assessment:

1. Early Detection: DevSecOps emphasizes identifying vulnerabilities early in the development process. Automated tools and code analysis are used to detect security weaknesses as code is being written, ensuring issues are caught before they reach production.
2. Continuous Monitoring: Vulnerability scanning tools are integrated into the CI/CD pipeline, scanning code and dependencies at each stage. This ensures that any new vulnerabilities are identified as changes are made and code is built and deployed.
3. Automated Testing: DevSecOps employs automated security tests, including static analysis for code vulnerabilities, dynamic tests for application behavior, and interactive tests for real-time analysis. These tests provide quick feedback to developers, enabling them to fix issues promptly.
4. Dependency Management: DevSecOps tools assess third-party libraries for vulnerabilities, ensuring that dependencies are up-to-date and free from known security flaws.
5. Policy Enforcement: DevSecOps enforces security policies through automated checks. If code or configurations violate security rules, the pipeline can halt or flag the deployment, preventing insecure code from reaching production.
6. Collaboration: DevSecOps promotes collaboration between development, operations, and security teams. By integrating security considerations into the development workflow, teams work together to address vulnerabilities efficiently.
7. Remediation: When vulnerabilities are identified, the DevSecOps approach facilitates rapid remediation. Automated tests verify fixes, ensuring that vulnerabilities are resolved effectively.
8. Continuous Improvement: DevSecOps incorporates feedback from security incidents and assessments to continuously enhance security practices, refining the vulnerability assessment process over time.

In summary, vulnerability assessment using DevSecOps is a proactive approach that integrates security practices into every step of the software development life cycle. This ensures that potential vulnerabilities are detected, addressed, and remediated early, resulting in more secure software releases and reduced risk of security breaches.

## **4. System Requirement**

### **4.1 User Interface**

1. EC2 Instance Ubuntu

### **4.2 Software Requirement**

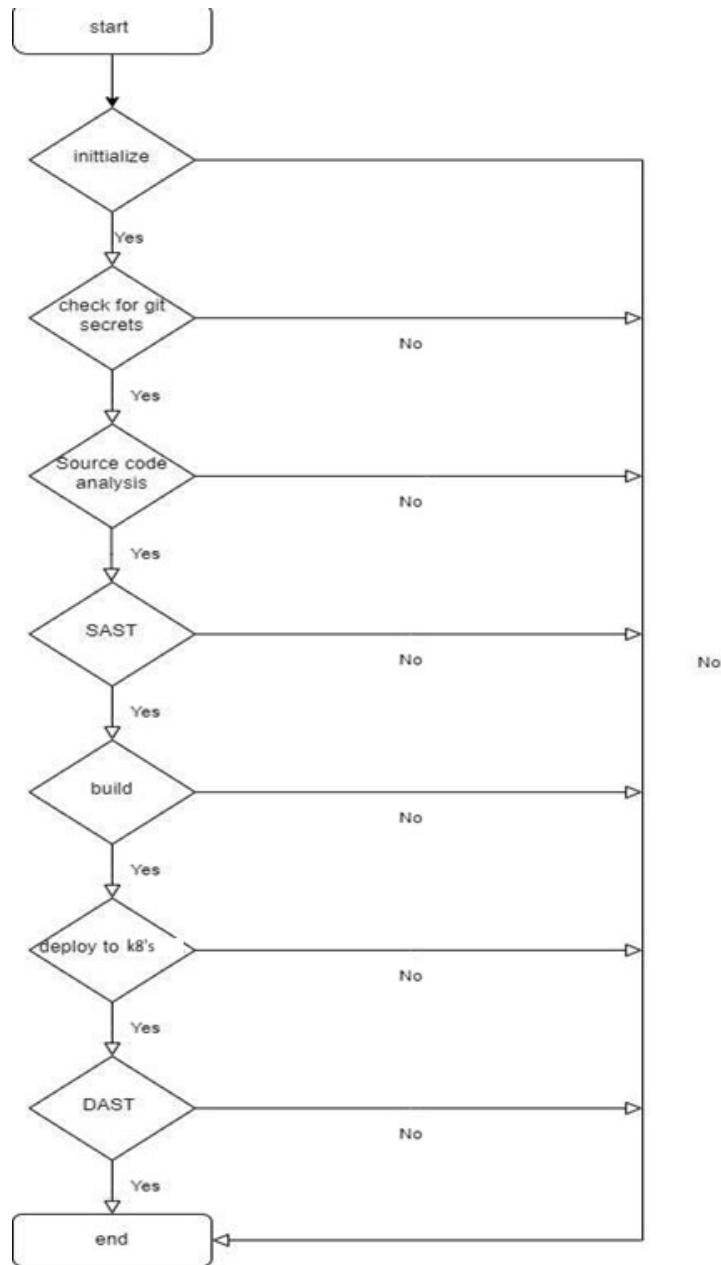
1. GIT
2. Jenkins
3. Docker
4. Kubernetes  
(Microk8s)
5. SAST  
Sonarqube
6. OWASP  
ZAP
7. Dependency-check 8.1.0
8. Apache Maven version- 3.6.37.
9. Trufflehog
10. Apache Tomcat 9(Docker Image)

### **4.3 Hardware Requirement**

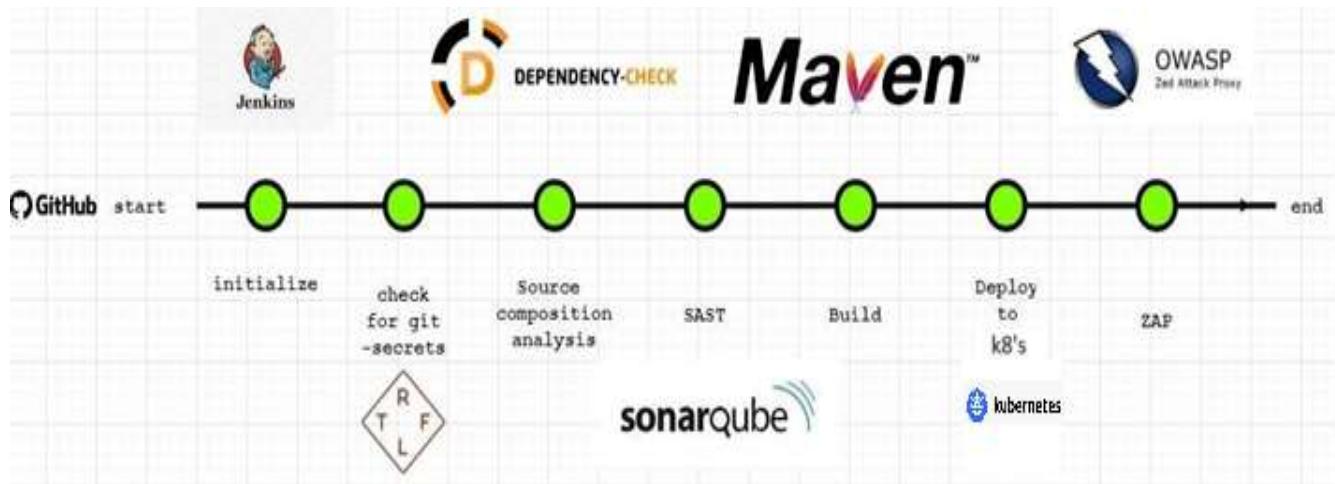
1. Windows 10 or 11
- 2.Ubuntu 20GB Harddrive
- 3.t2-micro medium 4GBRAM

## System Architecture

### 5.1 Data Flow Diagram



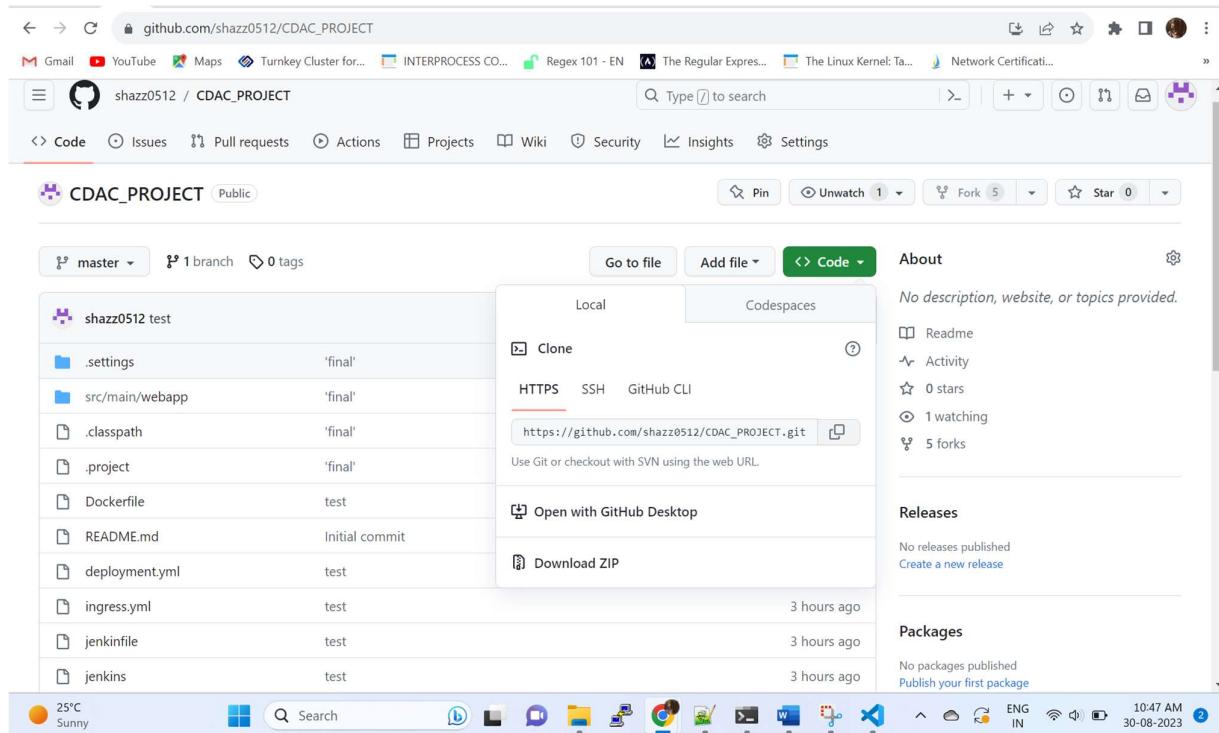
## 5.2 Activity Diagram



## 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.



## 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.

### 1. Repository (Repo):

- A repository is a collection of files and directories that make up a project, along with the version history of those files.
- Git repositories can be hosted locally or on remote platforms like GitHub, GitLab, and Bitbucket.

### 2. Commit:

- A commit is a snapshot of the changes made to the files in your repository at a specific point in time.
- Each commit has a unique identifier (hash) and includes information about the author, date, and a commit message describing the changes.

### 3. Branch:

- A branch is a parallel version of a repository that allows you to work on features or fixes without affecting the main codebase (usually the `master` branch).
- You can create, switch between, and merge branches.

### 4. Merge:

- Merging combines changes from one branch into another, typically to incorporate changes made on a feature branch back into the main branch.
- Merge conflicts can occur when Git cannot automatically reconcile different changes made in the same part of a file.

### 5. Pull Request (PR) / Merge Request (MR):

- A pull request (or merge request) is a way to propose changes from a branch to another branch (often from a feature branch to the main branch).
- It allows code review and discussion before the changes are merged.

### 6. Clone:

- Cloning creates a copy of a remote repository on your local machine. This allows you to work on the code locally and push your changes back to the remote repository.

### 7. Fetch:

- Fetch downloads the latest changes from a remote repository but doesn't apply them to your local branch. It updates your local tracking branches.

**8. Pull:**

- Pull is a combination of `fetch` and `merge`. It fetches the latest changes from the remote repository and automatically merges them into your local branch.

**9. Push:**

- Push sends your local commits to a remote repository. It updates the remote repository with the changes you've made locally.

**10. Remote:**

- A remote is a version of the repository that is hosted on a server, separate from your local machine.
- You can push and pull changes to and from remotes.

**11. Origin:**

- "Origin" is often the default name given to the remote repository from which you cloned your local repository.

**12. HEAD:**

- `HEAD` is a special pointer that points to the latest commit in the current branch. It's where you're currently working.

**13. Staging Area (Index):**

- The staging area is a space where you prepare and organize your changes before committing them. Changes added to the staging area are included in the next commit.

## 8. Jenkins Pipelines

Jenkins Pipeline is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.

A *continuous delivery (CD) pipeline* is an automated expression of your process for getting software from version control right through to your users and customers. Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment.

Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the [Pipeline domain-specific language \(DSL\) syntax](#)

The definition of a Jenkins Pipeline is written into a text file (called a [Jenkinsfile](#)) which in turn can be committed to a project's source control repository. This is the foundation of "Pipeline-as-code"; treating the CD pipeline a part of the application to be versioned and reviewed like any other code

### Pipeline

A Pipeline is a user-defined model of a CD pipeline. A Pipeline's code defines your entire build process, which typically includes stages for building an application, testing it and then delivering it.

### Node

A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline

### Stage

A `stage` block defines a conceptually distinct subset of tasks performed through the entire Pipeline (e.g. "Build", "Test" and "Deploy" stages), which is used by many plugins to visualize or present Jenkins Pipeline status/progress.

### Step

A single task. Fundamentally, a step tells Jenkins *what* to do at a particular point in time (or "step" in the process). For example, to execute the shell command `make` use the `sh` step: `sh 'make'`. When a plugin extends the Pipeline DSL, [that](#) typically means the plugin has implemented a new `step`

### Declarative Pipeline fundamentals

In Declarative Pipeline syntax, the `pipeline` block defines all the work done throughout your entire Pipeline.

## Jenkinsfile (Declarative Pipeline)

```
pipeline {  
    agent  
    any (1)  
    stages {  
        stage('Build')  
        { (2)steps  
            {  
                // (3)  
            }  
        }  
        stage('Test')  
        { (4)steps  
            {  
                // (5)  
            }  
        }  
        stage('Deploy')  
        { (6)steps {  
            // (7)  
        }  
    }  
}
```

← → ⌂ Not secure | 65.0.71.70:8080/view/all/newJob

Gmail YouTube Maps Turnkey Cluster for... INTERPROCESS CO... Regex 101 - EN The Regular Expressions The Linux Kernel: Ta... Network Certificati... »

**Jenkins** Search (CTRL+K) Jenkins Admin log out

Dashboard > All >

**Enter an item name**

» Required field

**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.

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

## 9. Trufflehog

TruffleHog is a utility that searches through git repositories for secrets, private keys and credentials so that you can protect your data before a breach occurs.

It is effective at finding secrets accidentally committed.

TruffleHog previously functioned by running entropy checks on git diffs. This functionality still exists, but high signal regex checks have been added, and the ability to suppress entropy checks have also been added.

Let's suppose there is a requirement from your client end to figure out all the secrets, sensitive API keys which are present in the code repositories.

The steps below will guide you on how to setup TruffleHog to meet the above requirement.

In the following section, we will setup a Jenkins pipeline in which we will launch a Docker container with the help of Dockerfile and run our TruffleHog utility in that Docker container.

Step 1: Go to your Jenkins server and start creating a Freestyle project.

Step 2: In the general section of your pipeline, give a brief description of your pipeline

Step 3: Give a log rotation strategy to your pipeline.

Step 4: Here, we are leaving the rest of the options to default values. You can change and give options according to your need

Step 5: Under the source code management section, pass the repository URL. Your git repo contains a Dockerfile that will build your docker container.

Step 6: Under the 'build' section, execute the shell commands:

Step 7: After your pipeline is executed successfully, you can go to console output and see the pipeline result.

## 10. OWASP Dependency-Check

Dependency-Check is a Software Composition Analysis (SCA) tool that attempts to detect publicly disclosed vulnerabilities contained within a project's dependencies. It does this by determining if there is a Common Platform Enumeration (CPE) identifier for a given dependency. If found, it will generate a report linking to the associated CVE entries.

Dependency-check has a command line interface, a Maven plugin, an Ant task, and a Jenkins plugin. The core engine contains a series of analyzers that inspect the project dependencies, collect pieces of information about the dependencies. The evidence is then used to identify the [Common Platform Enumeration \(CPE\)](#) for the given dependency. If a CPE is identified, a listing of associated [Common Vulnerability and Exposure \(CVE\)](#) entries are listed in a report. Other 3rd party services and data sources such as the Jenkins plugin, Maven plugin, SSH Agents.

Dependency-check works by collecting information about the files it scans (using Analyzers). The information collected is called Evidence; there are three types of evidence collected: vendor, product, and version. For instance, the Jar Analyzer will collect information from the Manifest, pom.xml, and the package names within the JAR files scanned and it has heuristics to place the information from the various sources into one or more buckets of evidence.

The OWASP Top 10 2013 contains a new entry: A9-Using Components with Known Vulnerabilities. Dependency Check can currently be used to scan applications (and their dependent libraries) to identify any known vulnerable components.

Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impact.

## 11. SonarQube

[SonarQube](#) is an automatic code review tool to detect bugs, vulnerabilities, and code smells in your code. It can integrate with your existing workflow to enable continuous code inspection across your project branches and pull requests.

SonarQube is a Code Quality Assurance tool that collects and analyzes source code, and provides reports for the code quality of your project. It combines static and dynamic analysis tools and enables quality to be measured continually over time. Everything from minor styling choices, to design errors are inspected and evaluated by SonarQube. This provides users with a rich searchable history of the code to analyze where the code is messing up and determine whether or not it is styling issues, code defeats, code duplication, lack of test coverage, or excessively complex code. The software will analyze source code from different aspects and drills down the code layer by layer, moving module level down to the class level, with each level producing metric values and statistics that should reveal problematic areas in the source code that needs improvement.

SonarQube also ensures code reliability, Application security, and reduces technical debt by making your codebase clean and maintainable. SonarQube also provides support for 27 different languages, including C, C++, Java, Javascript, PHP, GO, Python, and much more. SonarQube also provides CI/CD integration, and gives feedback during code review with branch analysis and pull request decoration.

SonarQube's integration with GitHub Enterprise and GitHub.com allows you to maintain code quality and security in your GitHub repositories.

With this integration, you'll be able to:

- Import your GitHub repositories - Import your GitHub repositories into SonarQube to easily set up SonarQube projects.
- Analyze projects with GitHub Actions - Integrate analysis into your build pipeline. Starting in [Developer Edition](#), SonarScanners running in GitHub Actions jobs can automatically detect branches or pull requests being built so you don't need to specifically pass them as parameters to the scanner
- Report your Quality Gate status to your branches and pull requests - (starting in [Developer Edition](#)) See your Quality Gate and code metric results right in GitHub so you know if it's safe to merge your changes.
- Authenticate with GitHub - Sign in to SonarQube with your GitHub credentials.

## Static Code Analysis

Static code analysis is done without executing any of the code. It is a collection of algorithms and techniques to analyze source code to automatically find potential errors and poor coding practices. This is done with compiler errors and run-time debugging techniques such as white box testing. Static code analysis is also considered a way to automate code review process. The tasks involved in static code analysis can be divided as such:

- Detecting errors in programs
- Recommendations on code formatting with a formatter
- Metrics computation, which gives you back a rating on how well your code is.
- Popular tools for static Code Analysis are Checkstyle, PMD, and Find Bug

## Getting SonarQube from a Docker Image

- Find the community version of SonarQube that you want to use on DockerHub: [https://hub.docker.com/\\_/sonarqube/](https://hub.docker.com/_/sonarqube/)
- Start the server by running:  
`docker run -d --name sonarqube -p 9000:9000 <image name>`
- Login to `http://localhost:9000` with the following credentials for system admin: (admin/admin)

## Analyzing a project with SonarQube

Once you are logged into sonarqube, to analyze a project follow the following steps:

- i. Click create new project button
- ii. When asked How do you want to create your project, select Manually.
- iii. Give your project a Project key and a Display name and click the Set Up button.
- iv. Under Provide a token, select Generate a token. Give your token a name, click the Generate button, and click Continue.
- v. Select your project's main language under Run analysis on your project, and follow the instructions to analyze your project. Here you'll download and execute a Scanner on your code (if you're using Maven or Gradle, the Scanner is automatically downloaded).

After successfully analyzing your code, you'll see your first analysis on SonarQube.

## **12. Maven**

Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, Maven deals with several areas of concern:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Encouraging better development practices

### **Making the build process easy**

While using Maven doesn't eliminate the need to know about the underlying mechanisms, Maven does shield developers from many details

### **Providing a uniform build system**

Maven builds a project using its project object model (POM) and a set of plugins. Once you familiarize yourself with one Maven project, you know how all Maven projects build. This saves time when navigating many projects

### **Providing quality project information**

Maven provides useful project information that is in part taken from your POM and in part generated from your project's sources. For example, Maven can provide

- Change log created directly from source control
- Cross referenced sources
- Mailing lists managed by the project
- Third party code analysis products also provide Maven plugins that add their reports to the standard information given by Maven Dependencies used by the project
- Unit test reports including coverage

### **Providing guidelines for best practices development**

Maven aims to gather current principles for best practices development and make it easy to guide a project in that direction.

For example, specification, execution, and reporting of unit tests are part of the normal build cycle using Maven. Current unit testing best practices were used as guidelines:

- Keeping test source code in a separate, but parallel source tree
- Using test case naming conventions to locate and execute tests
- Having test cases setup their environment instead of customizing the build for test preparation

Maven also assists in project workflow such as release and issue management.

Maven also suggests some guidelines on how to layout your project's directory structure. Once you learn the layout, you can easily navigate other projects that use Maven.

While takes an opinionated approach to project layout, some projects may not fit with this structure for historical reasons. While Maven is designed to be flexible to the needs of different projects, it cannot cater to every situation without compromising its objectives.

If your project has an unusual build structure that cannot be reorganized, you may have to forgo some features or the use of Maven altogether.

The result is a tool that can now be used for building and managing any Java-based project. We hope that we have created something that will make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

## **13.Tomcat**

It is an open-source Java servlet container that implements many Java Enterprise Specs such as the Websites API, Java-Server Pages and last but not least, the Java Servlet. It began as the reference implementation for the very first Java-Server Pages and the [Java Servlet API](#). However, it no longer works as the reference implementation for both of these technologies, but it is considered as the first choice among the users even after that. It is still one of the most widely used java-servers due to several capabilities such as good extensibility, proven core engine, and well-tested and durable. Here we used the term "servlet" many times, so what is [java](#) servlet; it is a kind of software that enables the webserver to handle the dynamic(java-based) content using the Http protocols.

The Java ecosystem supports a wide variety of application servers, so let's have a little discussion on each of them and see where Tomcat fits in:

A servlet container is basically an implementation of the Java servlet specification, which is mainly used for the purpose of hosting Java servlets.

The Java enterprise application-server is an implementation of the Java specification.

A web- server is a kind of server designed to serve files using a local system such as Apache.

We can say that, at the center, the Tomcat is [JSP \(Java Server Pages\)](#) and Servlet. The JSP is one of the server-side programming technologies that enables the developers to create platform-independent dynamic content and also known as the server-side view rendering technology. A servlet is a java-based software component that helps in extending the capabilities of a server. However, it can also respond to several kinds of requests and generally implemented web server containers to host the web-applications on the

webservers. As the developer's point of view, we just have to write the java server pages (or JSP) or the servlet and not worry about routing; the Tomcat will handle the routing.

The Tomcat also consists of the webserver known as the Coyote engine due to which it's possible to extend the capability of Tomcat to include several java enterprise specs, and including the [Java Persistence API\(JPA\)](#). The Tomcat also has an extended version known as the "TomEE" that contains more enterprise features.

The Java ecosystem supports a wide variety of application servers, so let's have a little discussion on each of them and see where Tomcat fits in:

A servlet container is basically an implementation of the Java servlet specification, which is mainly used for the purpose of hosting Java servlets.

The Java enterprise application-server is an implementation of the Java specification.

A web- server is a kind of server designed to serve files using a local system such as Apache.

We can say that, at the center, the Tomcat is [JSP \(Java Server Pages\)](#) and Servlet. The JSP is one of the server-side programming technologies that enables the developers to create platform-independent dynamic content and also known as the server-side view rendering technology. A servlet is a java-based software component that helps in extending the capabilities of a server. However, it can also respond to several kinds of requests and generally implemented web server containers to host the web-applications on the webservers. As the developer's point of view, we just have to write the java server pages (or JSP) or the servlet and not required to worry about routing; the Tomcat will handle the routing.

The Tomcat also consists of the webserver known as the Coyote engine due to which it's possible to extend the capability of Tomcat to include several java enterprise specs, and including the [Java Persistence API\(JPA\)](#). The Tomcat also has an extended version known as the "TomEE" that contains more enterprise features.

## **14. ZAP**

OWASP ZAP is an ideal tool to use in automation (security testing). It can be run in headless mode and has a powerful API. The OWASP Zed Attack Proxy (OWASP ZAP) is an easy-to-use integrated penetration testing tool for finding vulnerabilities in web applications. ZAP passively scans all the requests and responses made during your exploration for vulnerabilities, continues to build the site tree, and records alert for potential vulnerabilities found during the exploration.

OWASP ZAP will proceed to crawl the web application with its spider and passively scan each page it finds. Then it will use the active scanner to attack all of the discovered pages, functionality, and parameters.

Software security testing is the process of assessing and testing a system to discover security risks and vulnerabilities of the system and its data. There is no universal terminology but for our purposes, we define assessments as the analysis and discovery of vulnerabilities without attempting to actually exploit those vulnerabilities. We define testing as the discovery and attempted exploitation of vulnerabilities.

Security testing is often broken out, somewhat arbitrarily, according to either the type of vulnerability being tested or the type of testing being done. A common breakout is

- Vulnerability Assessment – The system is scanned and analyzed for security issues.
- Penetration Testing – The system undergoes analysis and attack from simulated malicious attackers
- Runtime Testing – The system undergoes analysis and security testing from an end-user
- Code Review – The system code undergoes a detailed review and analysis looking specifically for security vulnerabilities

Pretesting usually follows these stages:

- Explore – The tester attempts to learn about the system being tested. This includes trying to determine what software is in use, what endpoints exist, what patches are installed, etc. It also includes searching the site for hidden content, known vulnerabilities, and other indications of weakness.
- Attack – The tester attempts to exploit the known or suspected vulnerabilities to prove they exist.
- Report – The tester reports back the results of their testing, including the vulnerabilities, how they exploited them and how difficult the exploits were, and the severity of the exploitation.

## 15. Kubernetes

Kubernetes, often abbreviated as K8s, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF). Kubernetes provides a powerful set of tools and features that simplify the process of managing complex containerized applications, making them more scalable, reliable, and efficient.

Key concepts and components of Kubernetes include:

1. Node: A physical or virtual machine that runs containerized applications. Nodes are grouped together into clusters.
2. Cluster: A collection of nodes that work together to run containerized applications. Clusters consist of a control plane and worker nodes.
3. Control Plane: The central management component of Kubernetes, responsible for making global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events.
  - API Server: Exposes the Kubernetes API and is the entry point for managing the cluster.
  - etcd: A distributed key-value store that stores configuration data for the cluster.
  - Scheduler: Assigns work (containers) to nodes based on resource availability and constraints.
  - Controller Manager: Monitors the state of the cluster and ensures that the desired state is maintained.
  - Cloud Controller Manager: Manages interactions with cloud providers if the cluster is deployed on a cloud platform.
4. Pod: The smallest deployable unit in Kubernetes, which can contain one or more containers. Containers within the same pod share the same network namespace and can communicate with each other via 'localhost'.
5. Deployment: A higher-level resource that defines how to manage the deployment of replica sets, which in turn manage the lifecycle of pods. Deployments enable rolling updates and rollbacks, ensuring application availability during updates.
6. Service: An abstraction that defines a set of pods and provides network access to them. Services enable load balancing and dynamic service discovery within the cluster.
7. ReplicaSet: Ensures that a specified number of pod replicas are running at all times. It is often managed by a deployment.
8. Namespace: A virtual cluster within the physical cluster that provides a way to divide cluster resources into logically named groups.
9. ConfigMap and Secret: Resources for managing configuration data and sensitive information, respectively, independently from application code.

10. Ingress: Manages external access to services within a cluster, often acting as a reverse proxy and providing features like SSL termination and routing.

11. Volume: An abstraction that allows pods to persist data beyond the lifetime of the containers within them.

Kubernetes offers numerous benefits, such as automating deployment processes, managing scaling and load balancing, providing self-healing capabilities, and supporting rolling updates and rollbacks without service disruptions. It has become an integral part of modern application deployment and infrastructure management, especially in cloud-native and containerized environments.

## 16. MicroK8s

MicroK8s is a lightweight, minimalistic distribution of Kubernetes designed for local development, testing, and single-node deployments. It is developed by Canonical, the company behind Ubuntu Linux, and it aims to make it easier for developers to set up and manage a Kubernetes environment on their local machines or small servers.

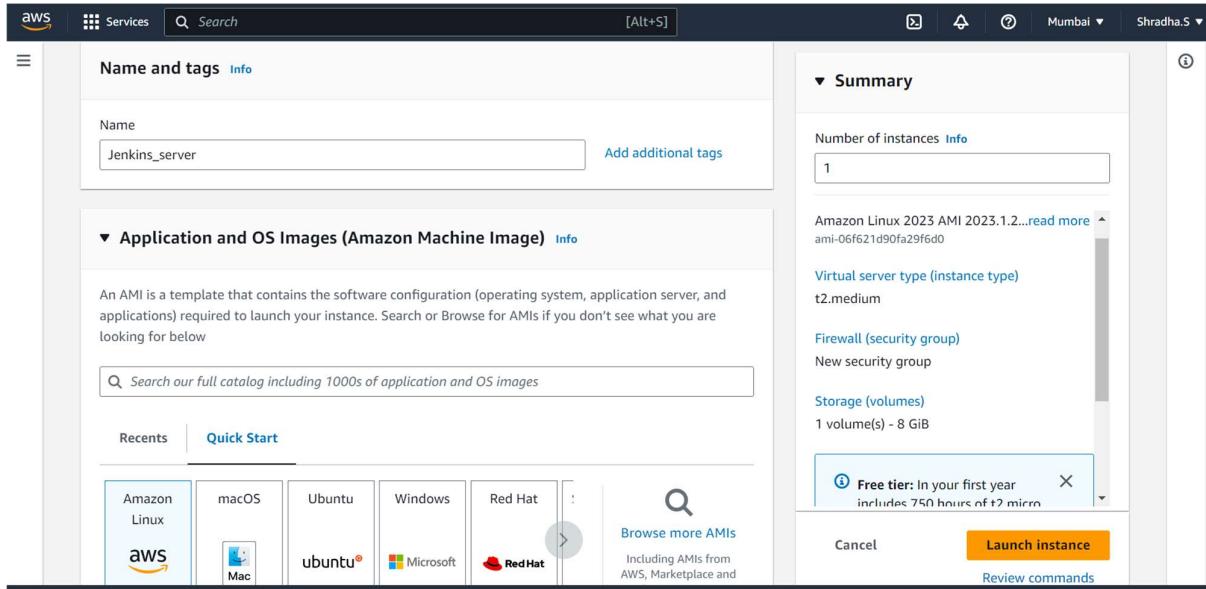
Key features of MicroK8s include:

1. Easy Installation: MicroK8s can be installed with a single command, making it quick and convenient to set up a local Kubernetes environment.
2. Single-Node Cluster: MicroK8s is designed to run as a single-node Kubernetes cluster. This makes it well-suited for local development and testing scenarios.
3. Addon Management: MicroK8s provides a range of optional add-ons that can be easily enabled or disabled. These add-ons offer extra functionality like storage, networking, and monitoring.
4. Isolation: MicroK8s uses container technology to isolate its components from the host system, reducing interference and ensuring a clean development environment.
5. Snap Package: MicroK8s is distributed as a snap package, which includes all the required dependencies and allows for easy installation, upgrades, and rollback.
6. Rapid Updates: MicroK8s is known for its fast update cycle, making it easy to stay up-to-date with the latest Kubernetes features and improvements.
7. Local Testing: Developers can use MicroK8s to test their applications in an environment that closely resembles a production Kubernetes cluster.
8. Support for CI/CD: MicroK8s can be used to set up continuous integration and continuous deployment (CI/CD) pipelines for testing and deploying applications in a Kubernetes environment.
9. Minimal Overhead: MicroK8s is optimized for minimal resource consumption and overhead, making it a good choice for resource-constrained systems.

MicroK8s is particularly useful for developers who want to experiment with Kubernetes concepts, test their applications, or develop microservices locally before deploying them to a larger Kubernetes cluster. It's important to note that MicroK8s is not intended for running large-scale or production workloads due to its single-node nature. For production deployments, traditional Kubernetes distributions or managed Kubernetes services are recommended.

## 17. Implementation

- Login onto the AWS console.
- Create EC2 instance for Jenkins Server.



- Install Jdk 11

```
Memory usage: 50%          IPv4 address for docker0: 172.17.0.1
Swap usage:  0%          IPv4 address for eth0:    172.31.6.250

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

53 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Mon Aug 28 06:28:23 2023 from 13.233.177.3
ubuntu@ip-172-31-6-250:~$ java --version
openjdk 11.0.20 2023-07-18
OpenJDK Runtime Environment (build 11.0.20+8-post-Ubuntu-1ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.20+8-post-Ubuntu-1ubuntu122.04, mixed mode, sharing)
ubuntu@ip-172-31-6-250:~$ []

i-0339bf25117c2e19b (JENKINS_SERVER)
PublicIPs: 65.0.71.70  PrivateIPs: 172.31.6.250
```

- curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \  
/usr/share/keyrings/jenkins-keyring.asc > /dev/null  
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \  
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null
- sudo apt-get update
- sudo apt-get install Jenkins
- sudo systemctl enable Jenkins
- sudo systemctl status Jenkins



Not secure | 65.0.71.70:8080/view/all/newJob

Gmail YouTube Maps Turnkey Cluster for... INTERPROCESS CO... Regex 101 - EN The Regular Express... The Linux Kernel: Ta... Network Certificati...

Jenkins Admin log out

Dashboard > All >

### Enter an item name

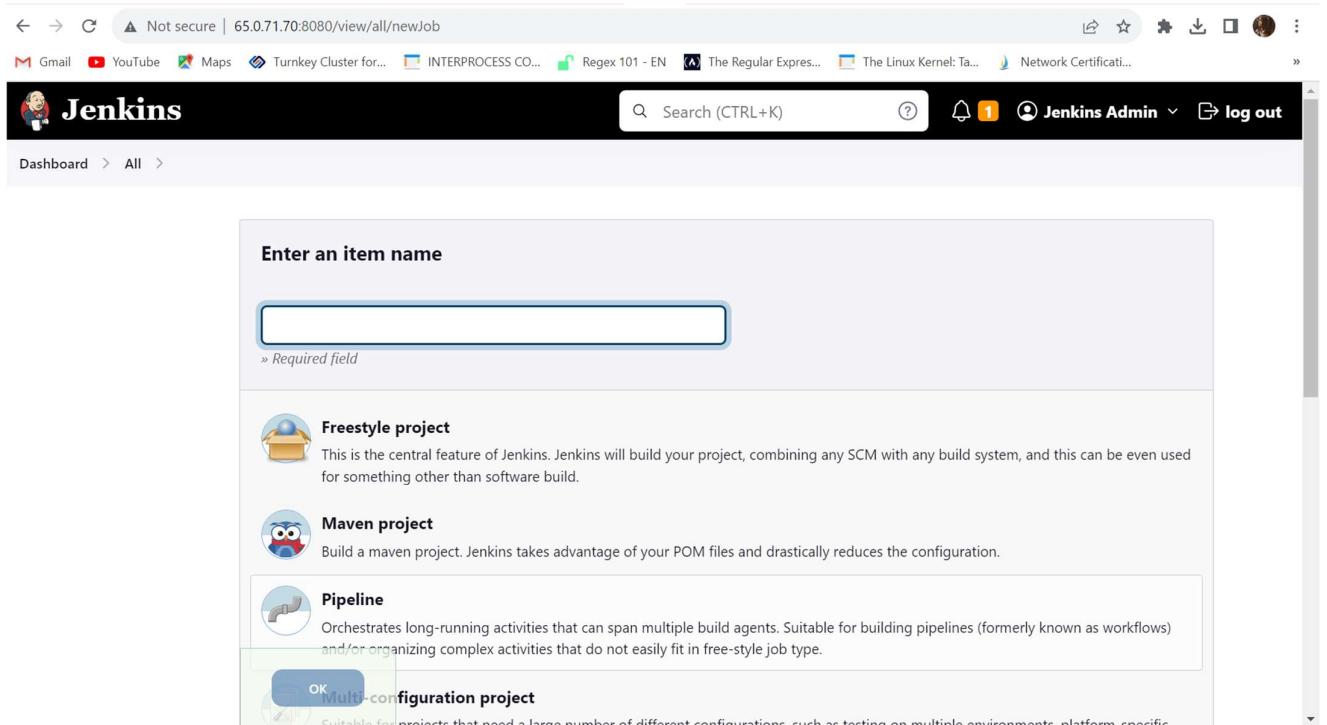
Required field

**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.

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



## • Configure Jenkins Job:

Not secure | 65.2.31.246:8080/job/CDAC\_Project/configure

Gmail YouTube Maps Turnkey Cluster for... INTERPROCESS CO... Regex 101 - EN The Regular Express... The Linux Kernel: Ta... Network Certificati...

Dashboard > CDAC\_Project > Configuration

### Configure

**Pipeline**

**Definition**  
Pipeline script from SCM

**SCM**  
Git

**Repositories**

Repository URL: https://github.com/shazz0512/CDAC\_PROJECT.git

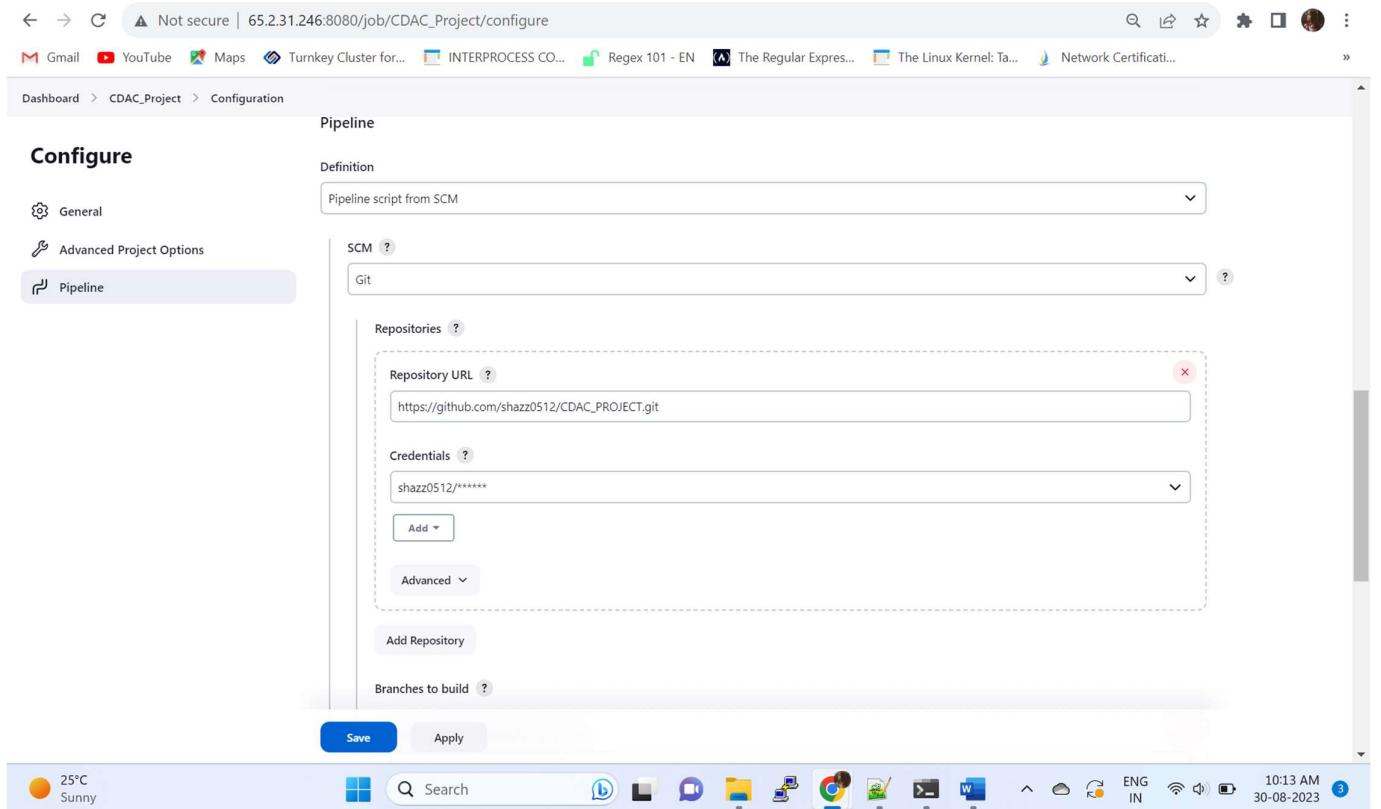
Credentials: shazz0512/\*\*\*\*\*

Add Advanced

Add Repository

Branches to build

Save Apply



25°C Sunny 10:13 AM 30-08-2023 ENG IN

## Maven installations

List of Maven installations on this system

[Add Maven](#)

Maven

Name

Maven

MAVEN\_HOME

/usr/share/maven



Install automatically [?](#)

Dependency-Check

Name

dependencyCheck



Install automatically [?](#)

[≡ Install from github.com](#)

Version

dependency-check 8.4.0

[Add Installer ▾](#)

[Add Dependency-Check](#)

- Configuring SonarQubes in Jenkins

SonarQube Scanner installations

List of SonarQube Scanner installations on this system

Add SonarQube Scanner

SonarQube Scanner  
Name  
sonar

Install automatically ?

≡ Install from Maven Central

Version  
SonarQube Scanner 5.0.1.3006

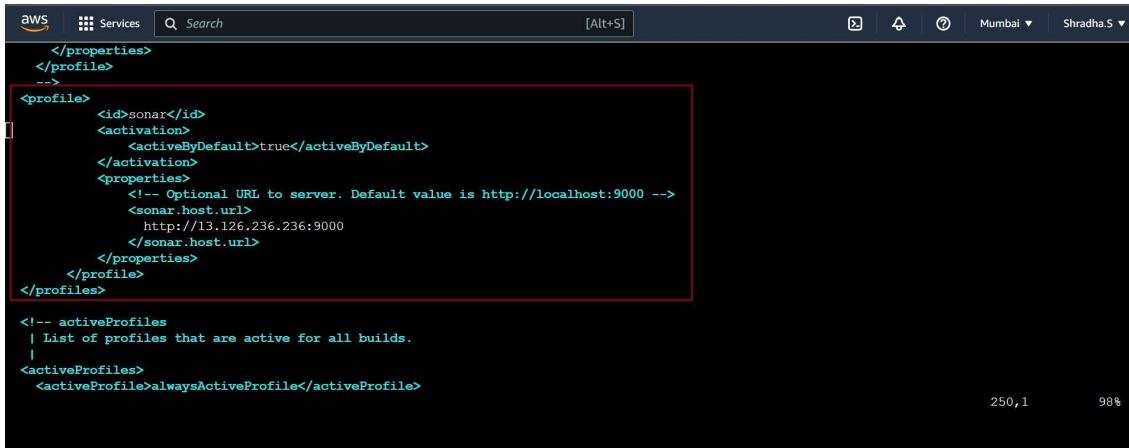
Add Installer ▾

```
ubuntu@ip-172-31-6-250:/usr/share/maven/conf$ vi settings.xml
```

```
<!-- pluginGroups
 | This is a list of additional group identifiers that will be searched when resolving plugins by their prefix, i.e.
 | when invoking a command line like "mvn prefix:goal". Maven will automatically add the group identifiers
 | "org.apache.maven.plugins" and "org.codehaus.mojo" if these are not already contained in the list.
 |-->
<pluginGroups>
  <!-- pluginGroup
   | Specifies a further group identifier to use for plugin lookup.
   <pluginGroup>com.your.plugins</pluginGroup>
  -->
  <pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
</pluginGroups>

<!-- proxies
 | This is a list of proxies which can be used on this machine to connect to the network.
 | Unless otherwise specified (by system property or command-line switch), the first proxy
 | specification in this list marked as active will be used.
 |-->
<proxies>
  <!-- proxy
   | Specification for one proxy, to be used in connecting to the network.

```



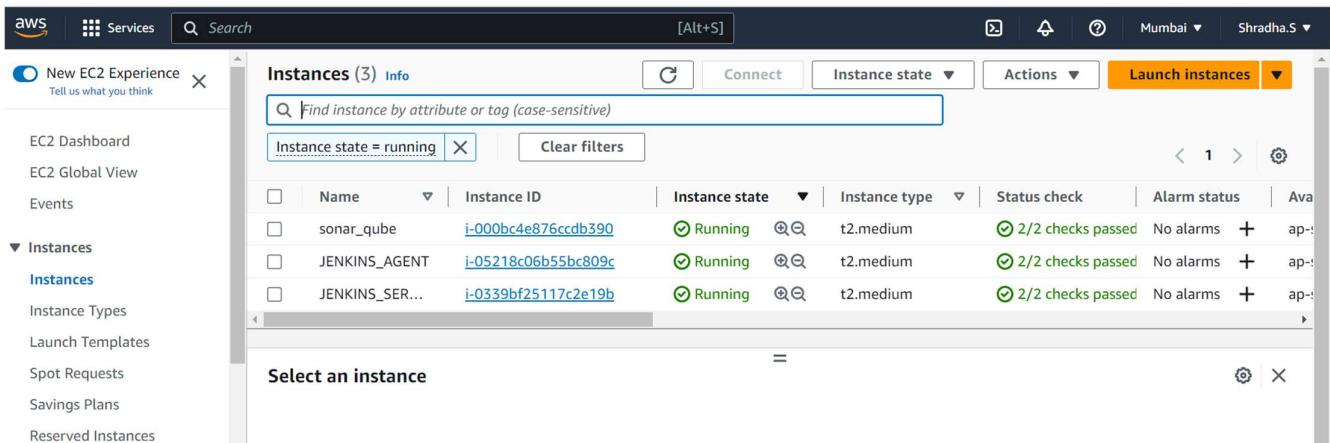
```
</properties>
</profile>
-->
<profile>
    <id>sonar</id>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <properties>
        <!-- Optional URL to server. Default value is http://localhost:9000 -->
        <sonar.host.url>
            http://13.126.236.236:9000
        </sonar.host.url>
    </properties>
</profile>
</profiles>

<!-- activeProfiles
| List of profiles that are active for all builds.
|
<activeProfiles>
<activeProfile>alwaysActiveProfile</activeProfile>
```

250,1

99%

- Created Similar 2 servers each for



Instance state = running	X	Clear filters
<input type="checkbox"/> sonar_qube <a href="#">i-000bc4e876ccdb390</a>	<span>Running</span>	t2.medium
<input type="checkbox"/> JENKINS_AGENT <a href="#">i-05218c06b55bc809c</a>	<span>Running</span>	t2.medium
<input type="checkbox"/> JENKINS_SER... <a href="#">i-0339bf25117c2e19b</a>	<span>Running</span>	t2.medium

- **Install tomcat9**

The screenshot shows a terminal window within the AWS CloudShell interface. The terminal title is "aws Services". The search bar contains "Search" and the keyboard shortcut "[Alt+S]" is shown. The terminal output displays the usage information for the catalina.sh script, followed by the command to start Tomcat 9, and the resulting log output. The log shows the configuration of CATALINA\_BASE, CATALINA\_HOME, CATALINA\_TMPDIR, JRE\_HOME, and CLASSPATH, and concludes with "Tomcat started.".

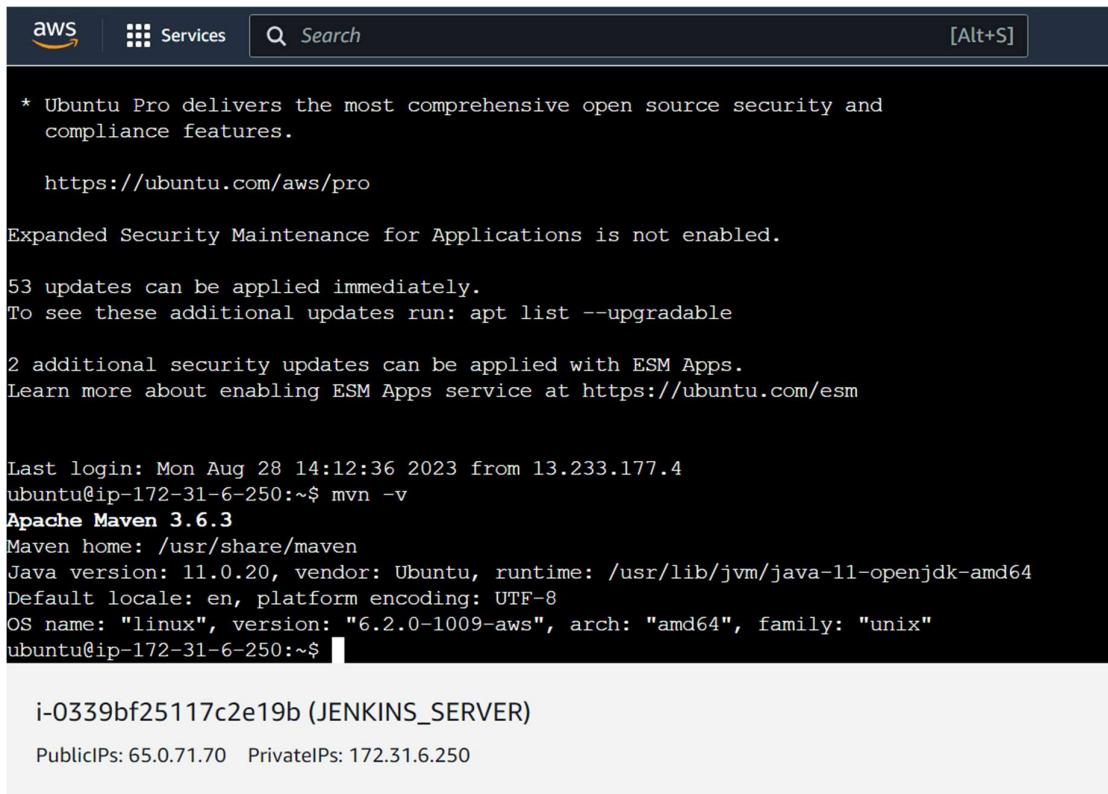
```
jpda start      Start Catalina under JPDA debugger
run             Start Catalina in the current window
run -security   Start in the current window with security manager
start           Start Catalina in a separate window
start -security Start in a separate window with security manager
stop            Stop Catalina, waiting up to 5 seconds for the process to end
stop n          Stop Catalina, waiting up to n seconds for the process to end
stop -force     Stop Catalina, wait up to 5 seconds and then use kill -KILL if still running
stop n -force   Stop Catalina, wait up to n seconds and then use kill -KILL if still running
configtest     Run a basic syntax check on server.xml - check exit code for result
version         What version of tomcat are you running?

Note: Waiting for the process to end and use of the -force option require that $CATALINA_PID is defined
root@ip-172-31-6-250:/opt# ^C
root@ip-172-31-6-250:/opt# ^C
root@ip-172-31-6-250:/opt# ./opt/tomcat9/bin/catalina.sh start
Using CATALINA_BASE:  /opt/tomcat9
Using CATALINA_HOME:  /opt/tomcat9
Using CATALINA_TMPDIR: /opt/tomcat9/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /opt/tomcat9/bin/bootstrap.jar:/opt/tomcat9/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
root@ip-172-31-6-250:/opt#
```

i-0339bf25117c2e19b (JENKINS\_SERVER)

PublicIPs: 65.0.71.70 PrivateIPs: 172.31.6.250

- Add Maven in Jenkins



```
* Ubuntu Pro delivers the most comprehensive open source security and
compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

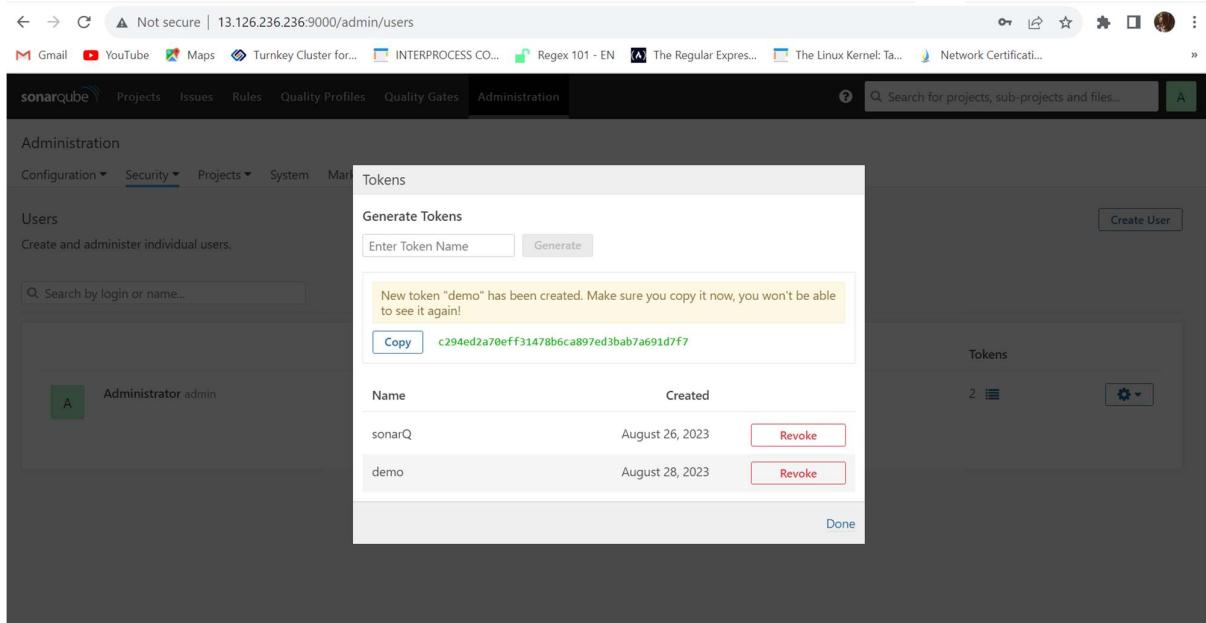
53 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

2 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Mon Aug 28 14:12:36 2023 from 13.233.177.4
ubuntu@ip-172-31-6-250:~$ mvn -v
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 11.0.20, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.2.0-1009-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-6-250:~$ [REDACTED]
```

i-0339bf25117c2e19b (JENKINS\_SERVER)  
PublicIPs: 65.0.71.70 PrivateIPs: 172.31.6.250

- Generate Token in SonarQube



Not secure | 13.126.236.236:9000/admin/users

Administration

Configuration ▾ Security ▾ Projects ▾ Issues ▾ Rules ▾ Quality Profiles ▾ Quality Gates ▾ Administration ▾

Tokens

Generate Tokens

Enter Token Name  Generate

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

**Copy** c294ed2a70eff31478b6ca897ed3bab7a691d7f7

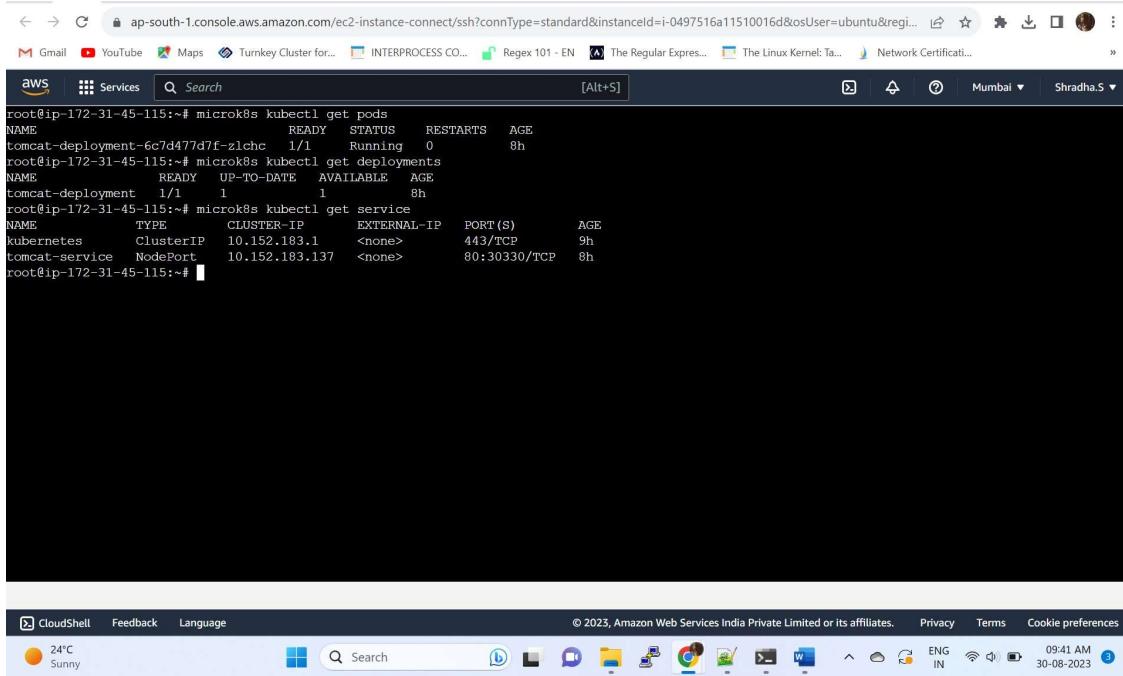
Name	Created	Action
sonarQ	August 26, 2023	<b>Revoke</b>
demo	August 28, 2023	<b>Revoke</b>

Tokens

2

Done

- Configuring Microk8s

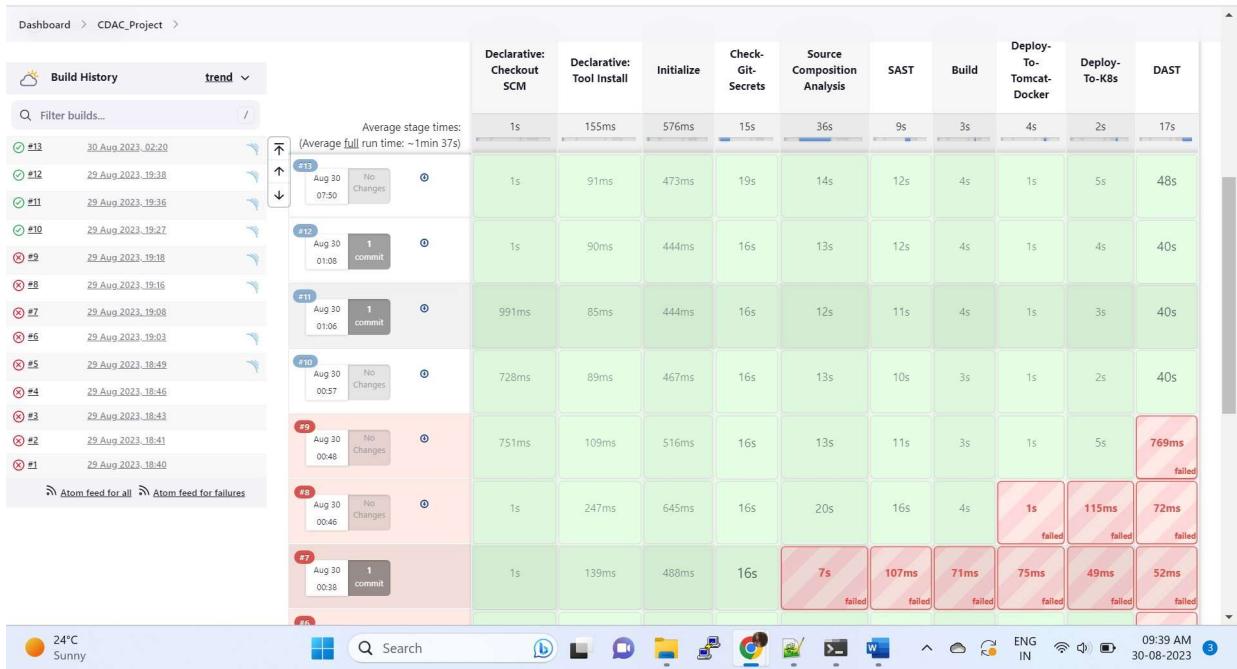


The screenshot shows a terminal window within an AWS CloudShell interface. The user is running several commands to inspect a Kubernetes cluster:

```
root@ip-172-31-45-115:~# microk8s kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
tomcat-deployment-6c7d477d7f-zl1hc  1/1     Running   0          8h
root@ip-172-31-45-115:~# microk8s kubectl get deployments
NAME            READY  UP-TO-DATE  AVAILABLE  AGE
tomcat-deployment  1/1      1          1          8h
root@ip-172-31-45-115:~# microk8s kubectl get service
NAME        TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)        AGE
kubernetes  ClusterIP  10.152.183.1 <none>       443/TCP        9h
tomcat-service  NodePort  10.152.183.137 <none>       80:30330/TCP  8h
root@ip-172-31-45-115:~#
```

Below the terminal, the CloudShell header includes links for CloudShell, Feedback, Language, and a weather widget showing 24°C and Sunny. The footer contains copyright information, privacy terms, cookie preferences, and system status indicators.

## 18. Testing



**A test sample which shows BUILD FAILURE in build step**

## 19. Result

The screenshot displays the Jenkins Pipeline interface for the 'CDAC\_Project'. The top navigation bar shows various links like Gmail, YouTube, Maps, Turnkey Cluster for..., INTERPROCESS CO..., Regex 101 - EN, The Regular Expressions..., The Linux Kernel: Ta..., Network Certificati..., and Admin. The main title is 'Pipeline CDAC\_Project'.

**Status:** Pipeline is successful (green). Last successful build was #13 on Aug 30, 2023, 07:50. Average stage times: Declarative: Checkout SCM (1s), Declarative: Tool Install (155ms), Initialize (576ms), Check-Git-Secrets (15s), Source Composition Analysis (36s), SAST (9s), Build (3s), Deploy-To-Tomcat-Docker (4s), Deploy-To-K8s (2s), and DAST (17s).

**Last Successful Artifacts:**

- dependency-check-jenkins.html (27.27 KB)
- dependency-check-junit.xml (708 B)
- dependency-check-report.csv (374 B)
- dependency-check-report.html (133.06 KB)
- dependency-check-report.json (3.19 KB)
- dependency-check-report.sarif (2.34 KB)
- dependency-check-report.xml (3.26 KB)

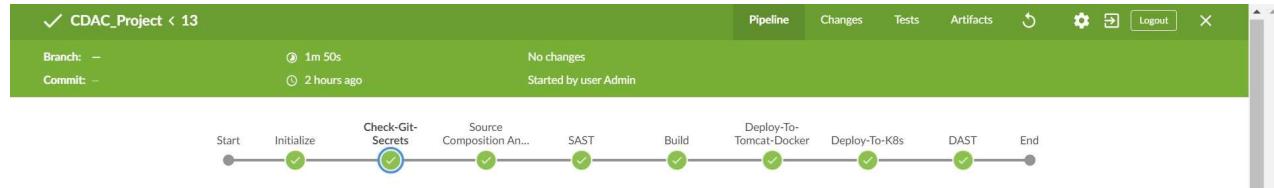
**Stage View:** Shows the stages of the pipeline: Initialize, Check-Git-Secrets, Source Composition An..., SAST, Build, Deploy-To-Tomcat-Docker, Deploy-To-K8s, and DAST. Each stage has a green checkmark indicating success.

**Pipeline Details:** Pipeline name is 'CDAC\_Project < 13'. Started by user 'Admin' at 09:36 AM on 30-08-2023. Last commit was 2 hours ago. No changes.

**Logs:** The 'Initialize' step log shows the following commands:

```
1 + echo PATH = $PATH
2 PATH = /usr/share/maven/bin:/usr/share/maven/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
3 + echo M2_HOME = /usr/share/maven
4 M2_HOME = /use/share/maven
```

## Output of trufflehog tool (git secrets)



**Check-Git-Secrets - 19s**

```

    > Maven -- Use a tool from a predefined Tool Installation <1s
    > Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. <1s
    ✓ sudo whoami -- Shell Script <1s
    ✓ sudo rm trufflehog || true -- Shell Script <1s
    ✓ sudo docker run gesellix/trufflehog --json https://github.com/shazz0512/test2.git > trufflehog -- Shell Script 18s
    ✓ sudo cat trufflehog -- Shell Script <1s

```

**sudo cat trufflehog -- Shell Script**

```

1  + sudo cat trufflehog
2  {
3      "branch": "master",
4      "commit": "Create Jenkinsfile",
5      "date": "2023-08-30 01:56:48",
6      "diff": "#@_1.61 +0.0 @@n-pipeline {n- agent any n- tools {n- maven 'Maven'n- }n- stages {n- stage ('Initialize') {n- steps {n- sh '''\n- echo\n'${PATH}'\n'${HOME} = ${HQ_HOME}'\n' -\n- sh 'sudo rm trufflehog' || true'n- sh 'sudo docker run gesellix/trufflehog --json https://github.com/shazz0512/test2.git > trufflehog'n- sh 'sudo cat trufflehog'n- }n- }\n- stage ('Source Composition Analysis') {n- steps {n- sh 'sudo rm oswasp' || true'n- sh 'sudo bash oswasp-dependency-check.sh'n- sh 'sudo chmod +x oswasp-dependency-check.sh'n- sh 'sudo cp target/sensorQubeEnv('sonar') {n- sh 'sudo mvn sonar:sonar' }n- sh 'sudo cat target/sensor/report-task.txt'n- }n- }\n- stage ('DAST') {n- steps {n- sh 'sudo mkdir -p /opt/zap/'n- sh 'sudo cp target/*.war /opt/tomcat/webapps/dacr.war'n- }n- }\n- stage ('DAST') {n- steps {n- sh 'sudo cp target/*.war /opt/tomcat/webapps/sonar.war'n- sh 'sudo ssh root@172.31.10.91 \"sudo docker run --rm -u root -v /opt/zap/wrk:rw -t oswasp/zap-docker-stable zap-baseline.py -t http://65.0.71.70:8181/ciac -x zap_report\" || true\"n- }\n- }\n- }\n- reason: "High Entropy"
9     "stringsFound":
10      [
11          "2d27be64d4de504b642547a47c2cef18ccad564"
12      ]
13  }
14  {
15      "branch": "master",
16      "commit": "Add all the file's",
17      "date": "Create Jenkinsfile",
18      "diff": "#@_1.61 +0.0 @@n-pipeline {n- agent any n- tools {n- maven 'Maven'n- }n- stages {n- stage ('Initialize') {n- steps {n- sh '''\n- echo\n'${PATH}'\n'${HOME} = ${HQ_HOME}'\n' -\n- sh 'sudo rm trufflehog' || true'n- sh 'sudo docker run gesellix/trufflehog --json https://github.com/shazz0512/test2.git > trufflehog'n- sh 'sudo cat trufflehog'n- }n- }\n- stage ('Source Composition Analysis') {n- steps {n- sh 'sudo rm oswasp' || true'n- sh 'sudo wget \"https://www.githubusercontent.com/shazz0512/test2/master/oswasp-dependency-check.sh\" -t /tmp/n- sh 'sudo chmod +x oswasp-dependency-check.sh'n- sh 'sudo cp target/sensorQubeEnv('sonar') {n- sh 'sudo mvn sonar:sonar' }n- sh 'sudo cp target/*.war /opt/tomcat/webapps/sonar.war'n- }n- }\n- stage ('DAST') {n- steps {n- sh 'sudo cp target/*.war /opt/tomcat/webapps/sonar.war'n- sh 'sudo cp target/*.war /opt/tomcat/webapps/sonar.war'n- sh 'sudo cp target/*.war /opt/tomcat/webapps/sonar.war'n- }\n- }\n- stage ('DAST') {n- steps {n- sh 'sudo cp target/*.war /opt/tomcat/webapps/sonar.war'n- sh 'sudo cp target/*.war /opt/tomcat/webapps/sonar.war'n- }\n- }\n- }\n- reason: "High Entropy"
20     "stringsFound":
21      [
22          "2d27be64d4de504b642547a47c2cef18ccad564"
23      ]
24  }
25  {
26      "branch": "master",
27      "commit": "Add all the file's",
28      "date": "Create Jenkinsfile"

```

## Output of OSWAP Dependency check (SCA)

✓ CDAC\_Project < 13

Branch: -      ① 1m 50s      No changes  
 Commit: -      ② 2 hours ago      Started by user Admin

Pipeline    Changes    Tests    Artifacts    ⚡    ⚙    ⌂    Logout

**Source Composition Analysis - 14s**

- ✓ > Maven — Use a tool from a predefined Tool Installation
- ✓ > Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.
- ✓ > sudo rm owasp\* || true — Shell Script
- ✓ > sudo wget "https://raw.githubusercontent.com/shazz0512/test2/master/owasp-dependency-check.sh" — Shell Script
- ✓ > sudo chmod +x owasp-dependency-check.sh — Shell Script
- ✓ > sudo bash owasp-dependency-check.sh — Shell Script

```

1  + sudo bash owasp-dependency-check.sh
2  Data directory is : /root/OWASP-Dependency-Check/data
3  Cache directory is : /root/OWASP-Dependency-Check/data/cache
4  latest: Pulling from owasp/dependency-check
5  Digest: sha256:d2cabdd9ea81d1d13f5f5d23d4c03946d9becd83b092fc1ee5465d9b5133615
6  Status: Image is up to date for owasp/dependency-check:latest
7  docker.io/owasp/dependency-check:latest
8  [INFO] Checking for updates
9  [INFO] Skipping NVD check since last check was within 4 hours.
10 [INFO] Skipping maven update since last update was within 24 hours.
11 [INFO] Skipping Hosted Suppressions file update since last update was within 2 hours.
12 [INFO] Skipping Known Exploited Vulnerabilities update check since last check was within 24 hours.
  
```

[Restart Source Composition Analysis](#) ⌂ ⌂

## Output of SAST(Sonarqube)

✓ CDAC\_Project < 13

Branch: -      ① 1m 50s      No changes  
 Commit: -      ② 2 hours ago      Started by user Admin

Pipeline    Changes    Tests    Artifacts    ⚡    ⚙    ⌂    Logout

**SAST - 12s**

- ✓ > Maven — Use a tool from a predefined Tool Installation
- ✓ > Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.
- ✓ > sudo mvn sonar:sonar -Dsonar.login=2d27be64d4de504b642547a47c2cef18ccad564c — Shell Script
- ✓ > sudo cat target/sonar/report-task.txt — Shell Script

```

1  + sudo cat target/sonar/report-task.txt
2  projectKey=23.groupn023.demo:CDAC
3  serverPort=236.236.236.9000
4  serverVersion=7.3.0.15553
5  dashboardUrl=http://13.126.236.236:9000/dashboard?id=23.groupn023.demo$3ACDAC
6  ceTaskId=AyPp7101L57adKZELL
7  ceTaskUrl=http://13.126.236.236:9000/api/ce/task?id=AyPp7101L57adKZELL
  
```

[Restart SAST](#) ⌂ ⌂

SonarQube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects, sub-projects and files... A 1 / 15 rules

**Filters** Clear All Filters

Search for rules...

**Type** Bug

Language CSS 15  
Search for languages...

Tag Repository Default Severity Status Available Since Template Quality Profile Sonar way ...

Inheritance

"important" should not be used on "keyframes" CSS Bug

"at-rules" should be valid CSS Bug

"calc" operands should be correctly spaced CSS Bug

"linear-gradient" directions should be valid CSS Bug

Color definitions should be valid CSS Bug

CSS properties should be valid CSS Bug

Font declarations should contain at least one generic font family CSS Bug

Media features should be valid CSS Bug

Properties should not be duplicated CSS Bug

Pseudo-class selectors should be valid CSS Bug

Pseudo-element selectors should be valid CSS Bug

Selectors should be known CSS Bug

Shorthand properties that override related longhand properties should be avoided CSS Bug

Selected rules are not yet checked for analysis

SonarQube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects, sub-projects and files... A 1 / 9 rules

**Filters** Clear All Filters

Search for rules...

**Type** Code Smell

Language CSS 9  
Search for languages...

Tag Repository Default Severity Status Available Since Template Quality Profile Sonar way ...

CSS files should not be empty CSS Code Smell

Duplicate imports should be removed CSS Code Smell unused

Duplicated font names should be removed CSS Code Smell

Empty blocks should be removed CSS Code Smell

Extra semicolons should be removed CSS Code Smell cert, misra, unused

Multi-line comments should not be empty CSS Code Smell

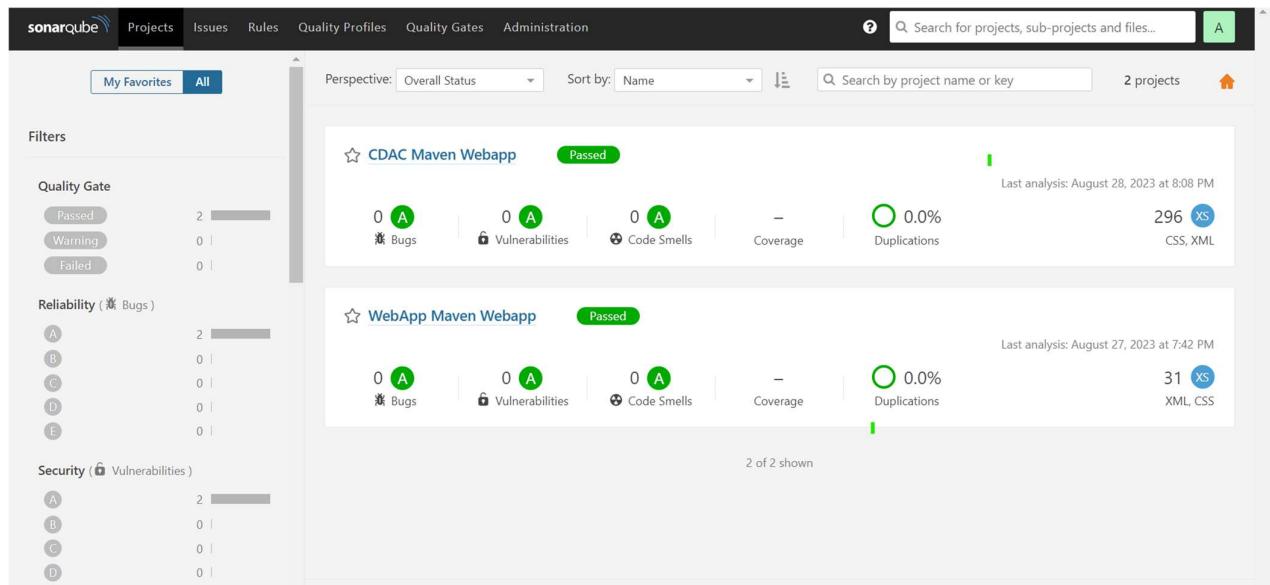
Selectors of lower specificity should come before overriding selectors of higher specificity CSS Code Smell

Selectors should not be duplicated CSS Code Smell

Strings should not contain new lines CSS Code Smell

9 of 9 shown

SonarQube™ technology is powered by SonarSource SA  
Community Edition - Version 7.3 (build 15553) - GPL v3 - Community - Documentation - Get Support - Plugins - Web API - About



## Output of Maven

✓ CDAC\_Project < 13

Branch: -      Commit: -      Pipeline      Changes      Tests      Artifacts      Logout

Start   Initialize   Check-Git-Secrets   Source Composition An...   SAST   Build   Deploy-To-Tomcat-Docker   Deploy-To-K8s   DAST   End

**Build - 4s**

- > Maven — Use a tool from a predefined Tool Installation
- > Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.
- sudo mvn clean package — Shell Script

```

1 + sudo mvn clean package
2 [INFO] Scanning for projects...
3 [INFO]
4 [INFO] -----< 23.groupno23.demo:CDAC >-----
5 [INFO] building CDAC Maven Webapp 1.0-SNAPSHOT
6 [INFO] -----
7 [INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ CDAC ---
8 [INFO] Deleting /var/lib/jenkins/workspace/CDAC_Project/target
9 [INFO]
10 [INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ CDAC ---
11 [WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
12 [INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/CDAC_Project/src/main/resources
13 [INFO]
14 [INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ CDAC ---
15 [INFO] No sources to compile
16 [INFO]
17 [INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ CDAC ---
18 [WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
19 [INFO] skip non existing resourceDirectory /var/lib/jenkins/workspace/CDAC_Project/src/test/resources
20 [INFO]
21 [INFO] --- maven-war-plugin:2.2:war (default-war) @ CDAC ---
22 [WARNING] An illegal reflective access operation has occurred
23 WARNING: Illegal reflective access by com.thoughtworks.xstream.core.util.Fields (file:/root/.m2/repository/com/thoughtworks/xstream/xstream/1.3.1/xstream-1.3.1.jar) to field
24 java.util.Properties.defaults
25 WARNING: Please consider reporting this to the maintainers of com.thoughtworks.xstream.core.util.Fields
26 WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
27 WARNING: All illegal access operations will be denied in a future release
28 [INFO] Packaging webapp
29 [INFO] Assembling webapp [CDAC] in [/var/lib/jenkins/workspace/Vulnerability_Assessment/target/cdac]
30 [INFO] Processing war project
31 [INFO] Copying war resources [/var/lib/jenkins/workspace/Vulnerability_Assessment/src/main/webapp]
32 [INFO] Webapp assembled in [43 msecs]
33 [INFO] Building war: /var/lib/jenkins/workspace/Vulnerability_Assessment/target/cdac.war
34 [INFO] WEB-INF/web.xml already added, skipping
35 [INFO] -----
36 [INFO] BUILD SUCCESS
37 [INFO] -----
38 [INFO] Total time: 1.732 s
39 [INFO] Finished at: 2023-08-28T14:38:11Z
40 [INFO] -----

```

3s

## Output of Deploy to Tomcat to local docker Registry

✓ CDAC\_Project < 13

Branch: - Commit: - 1m 50s 2 hours ago No changes Started by user Admin

**Deploy-To-Tomcat-Docker - 2s**

- > Maven — Use a tool from a predefined Tool Installation <1s
- > Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. <1s
- > sudo docker build -t cdac . — Shell Script 1s
- > sudo docker tag cdac localhost:5000/cdac — Shell Script <1s
- > sudo docker push localhost:5000/cdac — Shell Script <1s

```

1 + sudo docker push localhost:5000/cdac
2 Using default tag: latest
3 The push refers to repository [localhost:5000/cdac]
4 8d357a655a5a: Preparing
5 41344cfc0c1c: Preparing
6 89139e46dd: Preparing
7 1c674e43667: Preparing
8 9f16c18b1d5: Preparing
9 36ec17965e3d: Preparing
10 0860a5737d39: Preparing
11 7fc19fb33ab: Preparing
12 bcc45cce613d3: Preparing
13 36ec17965e3d: Waiting
14 0860a5737d39: Waiting
15 7fc19fb33ab: Waiting
  
```

Restart Deploy-To-Tomcat-Docker

24°C Sunny 09:32 AM 30-08-2023 ENG IN

## Output of Deploy to K8s

✓ CDAC\_Project < 13

Branch: - Commit: - 1m 50s 2 hours ago No changes Started by user Admin

**Deploy-To-K8s - 5s**

- > Maven — Use a tool from a predefined Tool Installation <1s
- > Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step. <1s
- > sudo microk8s kubectl apply -f deployment.yaml — Shell Script 4s
 

```

1 + sudo microk8s kubectl apply -f deployment.yaml
2 deployment.apps/tomcat-deployment unchanged
  
```
- > sudo microk8s kubectl apply -f service.yaml — Shell Script <1s
 

```

1 + sudo microk8s kubectl apply -f service.yaml
2 service/tomcat-service unchanged
  
```
- > sudo microk8s kubectl get svc tomcat-service — Shell Script <1s
 

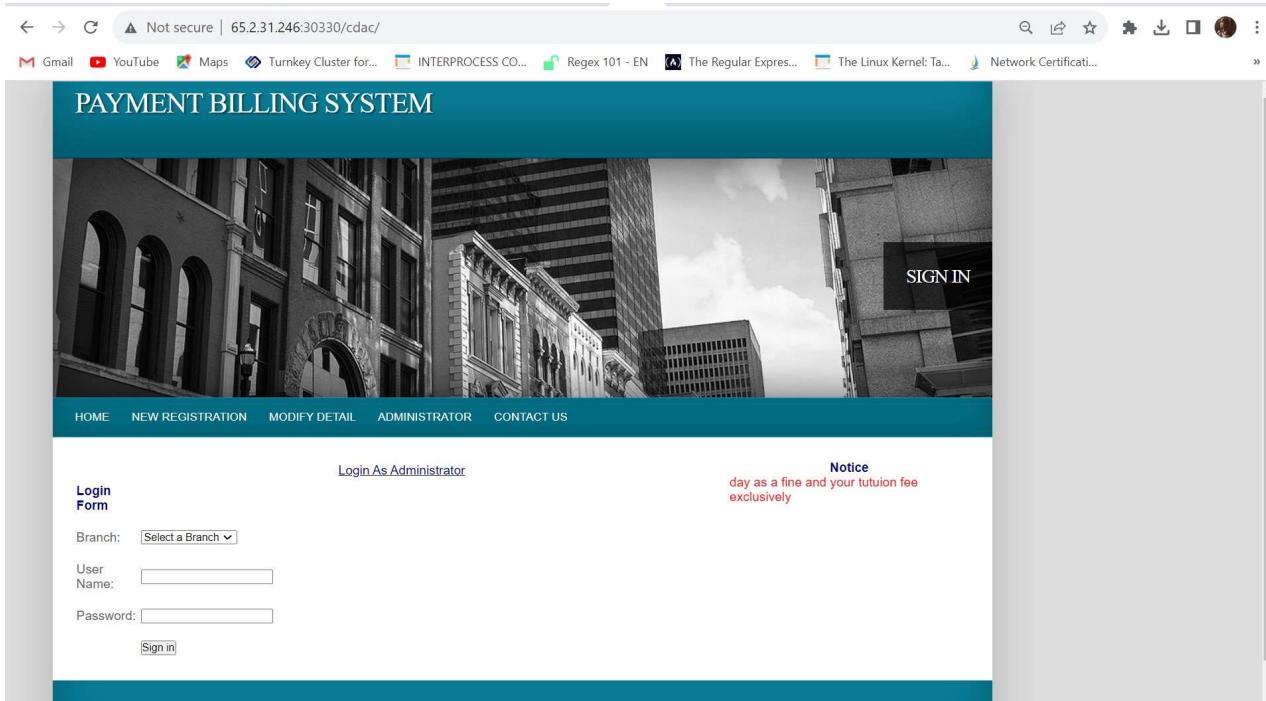
```

1 + sudo microk8s kubectl get svc tomcat-service
2 NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
3 tomcat-service  NodePort    10.152.183.137  <none>       80:30330/TCP  6h44m
  
```

Restart Deploy-To-K8s

24°C Sunny 09:33 AM 30-08-2023 ENG IN

Application is deployed



## Output of ZAP

✓ CDAC\_Project < 13

Branch: -      ① 1m 50s  
Commit: -      ② 2 hours ago

No changes      Started by user Admin

Pipeline      Changes      Tests      Artifacts      Logout

Start      Initialize      Check-Git-Secrets      Source      Composition An...      SAST      Build      Deploy-To-Tomcat-Docker      Deploy-To-KBs      DAST      End

**DAST - 49s**

> Maven — Use a tool from a predefined Tool Installation

> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.

> sudo mkdir -p /opt/zap — Shell Script

> sudo ssh root@172.31.10.91 "sudo docker run --rm -u root -v /opt/zap:/zap:rw -t owasp/zap2docker-stable zap-baseline.py -t http://65.2.31.246:30330/cdac/ -x zap\_report || true" — Shell Script

```

1 + sudo ssh root@172.31.10.91 sudo docker run --rm -u root -v /opt/zap:/zap:rw -t owasp/zap2docker-stable zap-baseline.py -t http://65.2.31.246:30330/cdac/ -x zap_report || true
2 Using the Automation Framework
3 Total time: 49s
4 PASS: Vulnerable JS Library (Powered by Retire.js) [10003]
5 PASS: In Page Banner Information leak [10000]
6 PASS: Cookie No HttpOnly Flag [10010]
7 PASS: Cookies Without Secure Flag [10011]
8 PASS: Re-examine Cache-control Directives [10015]
9 PASS: Cross-Domain JavaScript Source File Inclusion [10017]
10 PASS: Content-Type Header Missing [10019]
11 PASS: Information Disclosure - Sensitive Information in URL [10024]
12 PASS: Information Disclosure - Sensitive Information in HTTP Referrer Header [10025]
13 PASS: HTTP Parameter Override [10026]
14 PASS: Information Disclosure - Suspicious Comments [10027]
15 PASS: Open Redirect [10028]
16 PASS: Cookie Poisoning [10029]
17 PASS: User Controllable Charset [10030]

```

Restart DAST

24°C Sunny      Search      Task View      File Explorer      File Manager      Google Chrome      Microsoft Edge      Microsoft Word      Microsoft Excel      Microsoft Powerpoint      Microsoft Word      Microsoft Word      ENG IN      09:33 AM      30-08-2023

```

✓ sudo ssh root@172.31.10.91 "sudo docker run --rm -u root -v /opt/zap/wrkcrw -t owasp/zap2docker-stable zap-baseline.py -t http://65.2.31.246:30330/cdac -x zap_report || true" — Shell Script      48s
66 http://65.2.31.246:30330/cdac/admin.jsp (200 OK)
67 http://65.2.31.246:30330/cdac/ahome.jsp (200 OK)
68 http://65.2.31.246:30330/cdac/contactus.jsp (200 OK)
69 http://65.2.31.246:30330/cdac/generalinfo.jsp (200 OK)
70 WARN-NEW: Information Disclosure - Debug Error Messages [10023] x 2
71 http://65.2.31.246:30330/cdac/aloginprocess.jsp (500 Internal Server Error)
72 http://65.2.31.246:30330/cdac/aloginprocess.jsp (500 Internal Server Error)
73 WARN-NEW: Content Security Policy (CSP) Header Not Set [10058] x 10
74 http://65.2.31.246:30330/cdac/ (404 Not Found)
75 http://65.2.31.246:30330/cdac/ (200 OK)
76 http://65.2.31.246:30330/cdac/ahome.jsp (200 OK)
77 http://65.2.31.246:30330/cdac/contactus.jsp (200 OK)
78 http://65.2.31.246:30330/cdac/generalinfo.jsp (200 OK)
79 WARN-NEW: Non-Storable Content [10049] x 10
80 http://65.2.31.246:30330/cdac/ (302 Found)
81 http://65.2.31.246:30330/cdac/ (404 Not Found)
82 http://65.2.31.246:30330/cdac/ (200 OK)
83 http://65.2.31.246:30330/cdac/aloginprocess.jsp (200 OK)
84 http://65.2.31.246:30330/cdac/generalinfo.jsp (200 OK)
85 WARN-NEW: Cookie without SameSite Attribute [10054] x 1
86 http://65.2.31.246:30330/cdac/ (200 OK)
87 WARN-NEW: Permissions Policy Header Not Set [10063] x 10
88 http://65.2.31.246:30330/ (404 Not Found)
89 http://65.2.31.246:30330/cdac/ (200 OK)
90 http://65.2.31.246:30330/cdac/aloginprocess.jsp (200 OK)
91 http://65.2.31.246:30330/cdac/contactus.jsp (200 OK)
92 http://65.2.31.246:30330/cdac/generalinfo.jsp (200 OK)
93 WARN-NEW: Authentication Request Identified [10111] x 2
94 http://65.2.31.246:30330/cdac/aloginprocess.jsp (500 Internal Server Error)
95 http://65.2.31.246:30330/cdac/loginprocess.jsp (500 Internal Server Error)
96 WARN-NEW: Session Management Response Identified [10112] x 3
97 http://65.2.31.246:30330/cdac/ (200 OK)
98 http://65.2.31.246:30330/cdac/ (200 OK)
99 http://65.2.31.246:30330/cdac/ (200 OK)
100 WARN-NEW: Absence of Anti-CSRF Tokens [10202] x 7
101 http://65.2.31.246:30330/cdac/ (200 OK)
102 http://65.2.31.246:30330/cdac/admin.jsp (200 OK)
103 http://65.2.31.246:30330/cdac/ahome.jsp (200 OK)
104 http://65.2.31.246:30330/cdac/contactus.jsp (200 OK)
105 http://65.2.31.246:30330/cdac/generalinfo.jsp (200 OK)
106 WARN-NEW: Application Error Disclosure [90022] x 2
107 http://65.2.31.246:30330/cdac/aloginprocess.jsp (500 Internal Server Error)
108 http://65.2.31.246:30330/cdac/loginprocess.jsp (500 Internal Server Error)
109 FAIL-NEW: 0 FAIL-INPROG: 0 WARN-NEW: 11 WARN-INPROG: 0 INFO: 0 IGNORE: 0      PASS: 54
110 Automation plan warnings:
111 Job spider error accessing URL http://65.2.31.246:30330/ status code returned : 404 expected 200

```



We got to see 12 warnings at the end of the ZAP Analysis.

Hence Successfully implemented Vulnerability Assessment using DevSecOps.

## 20 . CODE

### Jenkinsfile

```
pipeline {
    agent any
    tools {
        maven 'Maven'
    }
    stages {
        stage ('Initialize') {
            steps {
                sh """
                    echo "PATH = ${PATH}"
                    echo "M2_HOME = ${M2_HOME}"
                    """
            }
        }
        stage ('Check-Git-Secrets') {
            steps {
                sh 'sudo whoami'
                sh 'sudo rm trufflehog || true'
                sh 'sudo docker run geselliix/trufflehog --json https://github.com/shazz0512/test2.git > trufflehog'
                sh 'sudo cat trufflehog'
            }
        }
        stage ('Source Composition Analysis') {
            steps {
                sh 'sudo rm owasp* || true'
                sh 'sudo wget "https://raw.githubusercontent.com/shazz0512/test2/master/owasp-dependency-check.sh" '
                sh 'sudo chmod +x owasp-dependency-check.sh'
                sh 'sudo bash owasp-dependency-check.sh'
                archiveArtifacts artifacts: '**/odc-reports/*', allowEmptyArchive: true
            }
        }
    }
}
```

```

        }
    }

stage ('SAST') {
    steps {
        withSonarQubeEnv('sonar') {
            sh 'sudo mvn sonar:sonar -Dsonar.login=2d27be64d4de504b642547a47c2cef18ccad564c'
            sh 'sudo cat target/sonar/report-task.txt'
        }
    }
}

stage ('Build') {
    steps {
        sh 'sudo mvn clean package'
    }
}

stage ('Deploy-To-Tomcat-Docker') {
    steps {
        sh 'sudo docker build -t cdac .'
        sh 'sudo docker tag cdac localhost:5000/cdac'
        sh 'sudo docker push localhost:5000/cdac'
    }
}

stage ('Deploy-To-K8s') {
    steps {
        sh 'sudo microk8s kubectl apply -f deployment.yml'
        sh 'sudo microk8s kubectl apply -f service.yml'
        sh 'sudo microk8s kubectl get svc tomcat-service'
    }
}

stage ('DAST') {
    steps {
        sh 'sudo mkdir -p /opt/zap'
        sh 'sudo ssh root@172.31.10.91 "sudo docker run --rm -u root -v /opt/zap:/zap:rw -t

```

---

```

owasp/zap2docker-stable zap-baseline.py -t http://65.2.31.246:30330/cdac/ -x zap_report || true" '
}
}

}

```

## Dockerfile

```

FROM tomcat:latest
COPY target/cdac.war /usr/local/tomcat/webapps/
EXPOSE 8080
CMD ["catalina.sh", "run"]

```

## Deployment.yml

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: tomcat-deployment
  labels:
    app: tomcat
spec:
  replicas: 1
  selector:
    matchLabels:
      app: tomcat
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
        - name: tomcat
          image: localhost:5000/cdac:latest
          ports:
            - containerPort: 8080

```

## Service.yml

```

kind: Service
apiVersion: v1
metadata:

```

```
name: tomcat-service
spec:
  selector:
    app: tomcat
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
    type: NodePort
```

## **21. Conclusion:**

In conclusion, the project "Vulnerability Assessment using DevSecOps" bridges the critical gap between security and development by integrating security practices seamlessly into the DevOps workflow. This project showcases the synergy between development and security teams to proactively identify and mitigate vulnerabilities throughout the software development lifecycle. By adopting a DevSecOps approach, organizations enhance their ability to identify and address security weaknesses early in the development process, reducing the potential for security breaches and minimizing the associated risks.

## 22. Reference

1. An Introduction to CI/CD Best Practices. <https://www.digitalocean.com/community/tutorials/introduction-to-cicd-best-practices>. [Online; accessed 14-May-2020]. 2018.
2. CVE security vulnerability database. Security vulnerabilities, exploits, references and more. <https://www.cvedetails.com/>. [Online; accessed 1-June-2020]. 2020.
3. CWE-531: Inclusion of Sensitive Information in Test Code. <https://cwe.mitre.org/data/definitions/531.html>. [Online; accessed 17-May-2020]. 2020.
4. CWE-94: Improper Control of Generation of Code ('Code Injection'). <https://cwe.mitre.org/data/definitions/94.html>. [Online; accessed 17-May-2020]. 2020.
5. CWE311: Missing encryption of sensitive data. <https://cwe.mitre.org/data/definitions/311.html>. [Online; accessed 14-May-2020]. 2020.
6. Martin Fowler. Continuous Delivery. 2013. url: <https://martinfowler.com/bliki/ContinuousDelivery.html> (visited on 05/18/2020).
7. Martin Fowler. Deployment Pipeline. 2013. url: <https://martinfowler.com/bliki/DeploymentPipeline.html> (visited on 04/06/2020).
8. Volker Gruhn, Christoph Hannebauer, and Christian John. "Security of public continuous integration services". In: Proceedings of the 9th International Symposium on open collaboration. 2013, pp. 1–10.
9. Red Hat. What is DevSecOps. url: <https://www.redhat.com/en/topics/devops/what-is-devsecops> (visited on 05/17/2020).
- 10 Identify vulnerabilities in third-party software libraries. <https://attack.mitre.org/techniques/T1389/>. [Online; accessed 21-May-2020]. 2018.
- 11 Identify vulnerabilities in third-party software libraries. <https://attack.mitre.org/techniques/T1389/>. [Online; accessed 21-May-2020]. 2018.
- 12 Christina Paule. "Securing DevOps: detection of vulnerabilities in CD pipelines". MA thesis. 2018.
- 13 Install Java : [C:\Users\Desktop\How To Install Java with Apt-Get on Ubuntu 16.04 \\_ DigitalOcean\\_files](C:\Users\Desktop\How To Install Java with Apt-Get on Ubuntu 16.04 _ DigitalOcean_files)
- 14 <https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+on+Ubuntu>.





