AdvanceDevOps Practical Case\_Study:

NITISH BHOSLE D15A – 04

## Continuous Integration with Static Code Analysis

**Concepts Used:** Jenkins, SonarQube, and AWS Cloud9(EC2 Instance).

**Problem Statement:** "Set up a Jenkins pipeline using AWS Cloud9 IDE to perform a static analysis of a Java/Python application. Integrate SonarQube for code quality checks."

**Tasks:**

* + Install Jenkins and set up a basic pipeline.
  + Configure SonarQube as part of the pipeline for static code analysis.
  + Run the pipeline and generate a report for code quality issues.

# Introduction

**Case Study Overview**

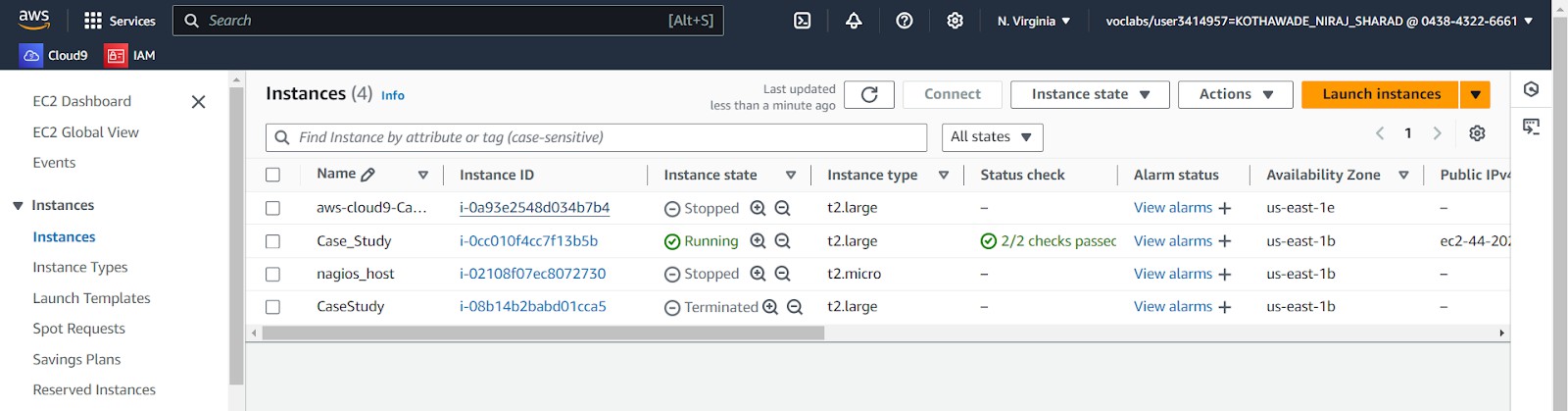
This case study demonstrates the setup of a Continuous Integration (CI) pipeline using Jenkins, integrated with SonarQube for static code analysis on AWS EC2 instance. The objective is to automate the quality checks of a Java/Python application and ensure that code quality issues are identified early in the development process. Jenkins, a popular CI/CD tool, automates code builds, testing, and deployment, while SonarQube provides comprehensive code quality and security analysis.

**Key Feature and Application**

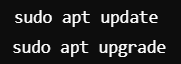
The key feature of this project is the integration of static code analysis into the CI pipeline. This allows developers to ensure code quality and adherence to best practices without manual intervention. It ensures early detection of bugs, code smells, and potential security vulnerabilities, making the development process smoother and more reliable.

# Step-by-Step Explanation

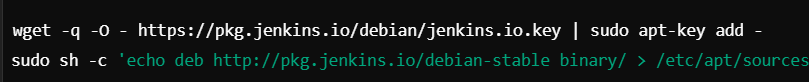
## Step 1: Create EC2 Instances for Jenkins and SonarQube

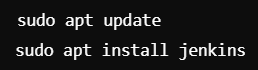
* 1. Login to AWS Console:
     + Navigate to the AWS Management Console and log in.
  2. Launch EC2 Instances:
     + Go to the EC2 Dashboard and click Launch Instance.
     + Choose Ubuntu Server 22.04 LTS as your AMI.
     + Select an instance type (t2.large for Jenkins and SonarQube).
     + Configure instance details (ensure Auto-assign Public IP).
     + Add storage (default 8 GB, adjust for larger projects).
     + Configure security group (allow ports 22, 8080 for Jenkins, 9000 for SonarQube).
     + Launch instances and download the key pair for SSH access.

## Step 2: Install Jenkins on EC2

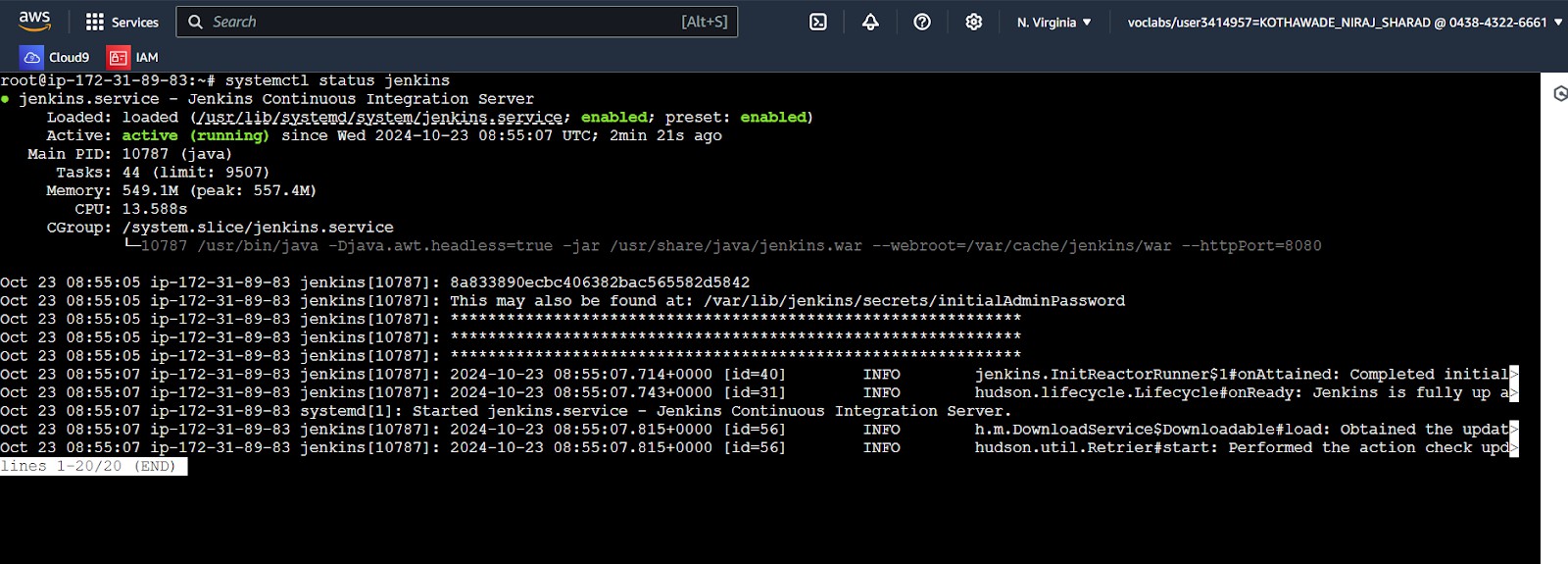
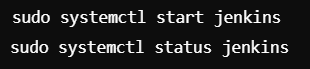
1. Connect to EC2 via SSH:
   * Use the key pair to SSH into the Jenkins EC2 instance.
2. Update the system:
3. Install Java 11:



1. Add Jenkins Repository:
   * Add Jenkins GPG key and repository.
2. Install Jenkins:

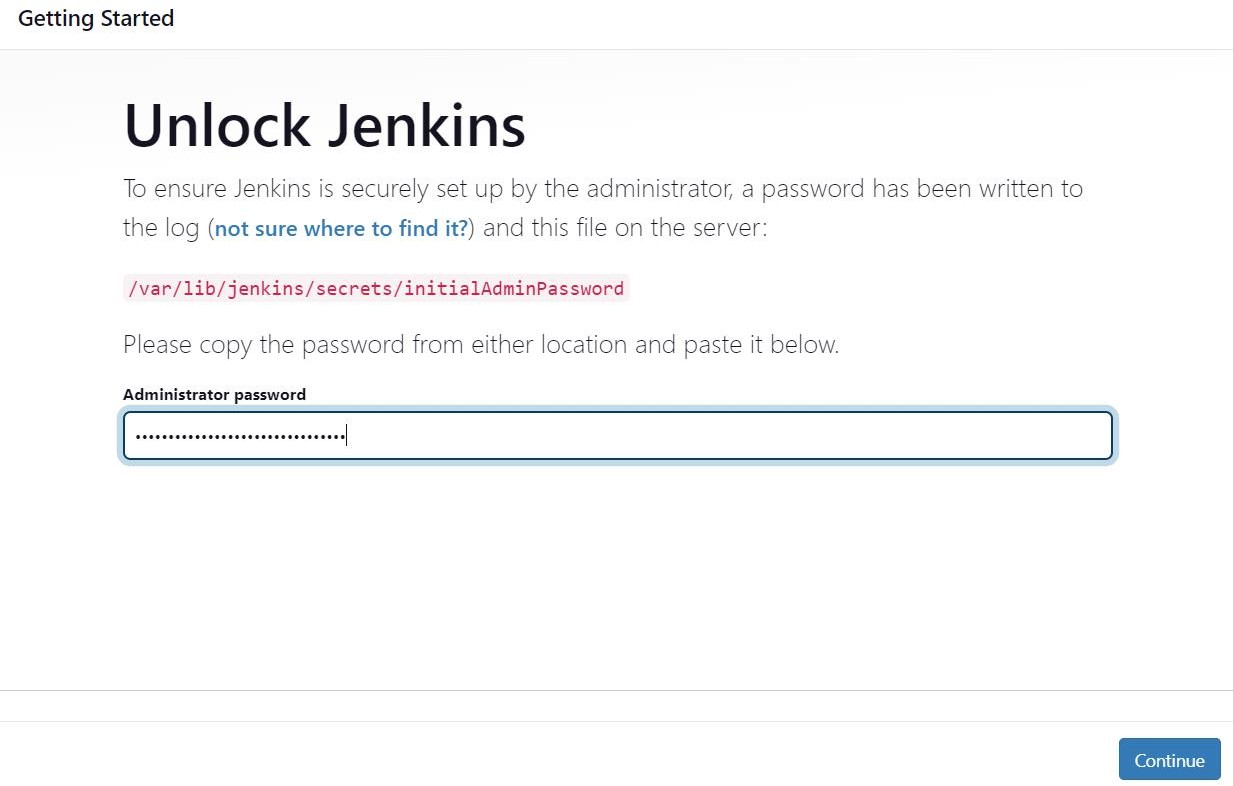


1. Start Jenkins:



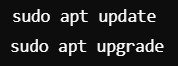
1. Access Jenkins:
   * Navigate to in your browser :
   * Follow on-screen instructions and use the initialAdminPassword to unlock Jenkins.



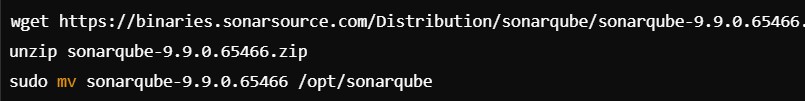


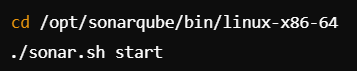
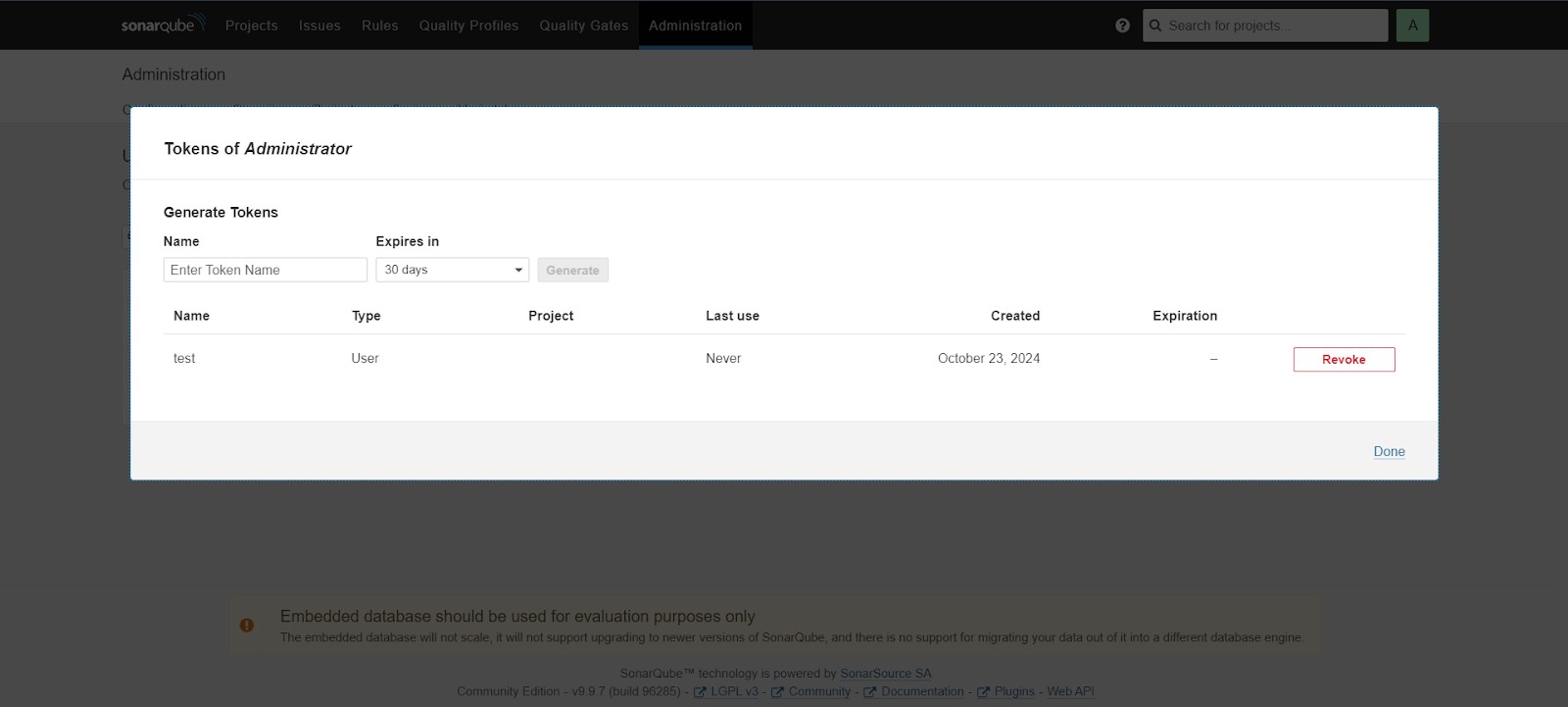
## Step 3: Install SonarQube on EC2

1. Connect to EC2 via SSH:
   * Use the key pair to SSH into the SonarQube EC2 instance.
2. Update the system:

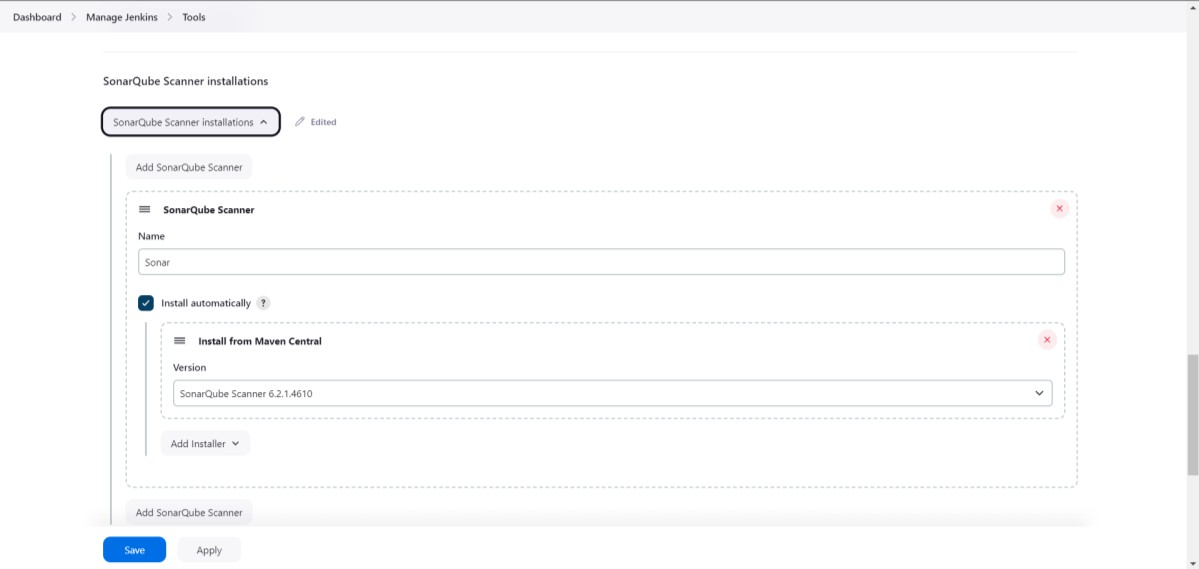
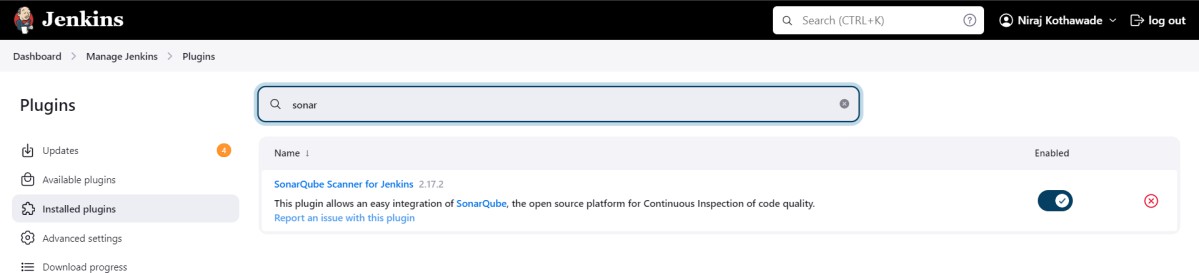


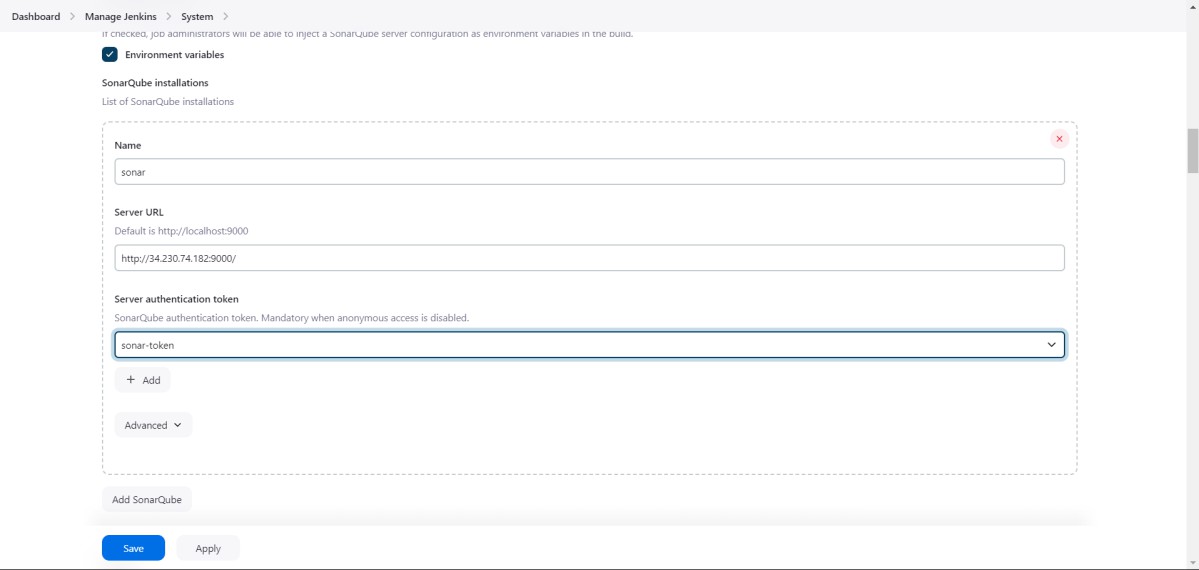
1. Install prerequisites (unzip, wget):
2. Download and install SonarQube:



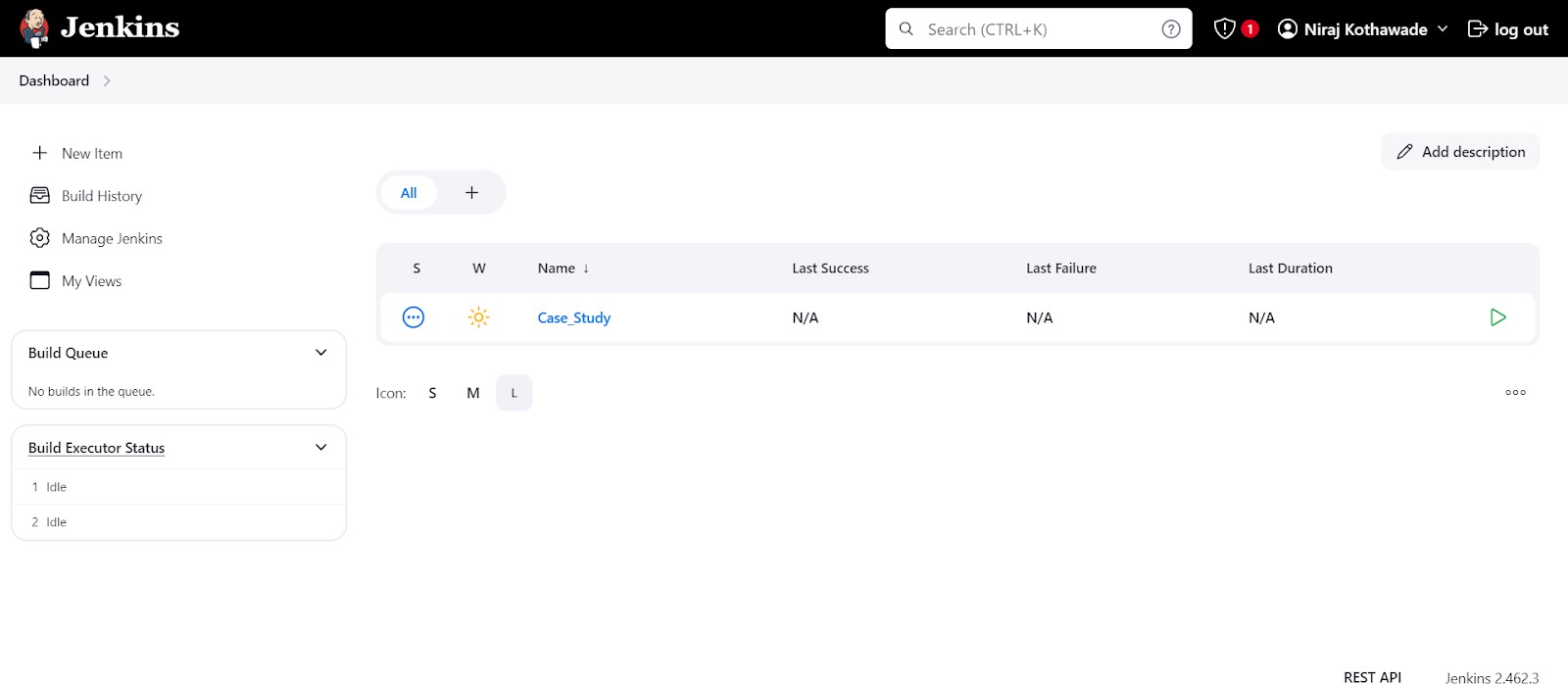
1. Start SonarQube:
2. Access SonarQube:
   * Navigate to in your browser :
   * Login using default credentials (admin/admin) and complete the setup.
3. Generate a token for Jenkins:

## Step 4: Configure Jenkins to Integrate with SonarQube

1. Install SonarQube Plugin in Jenkins:
   * Go to Manage Jenkins → Manage Plugins.
   * Install SonarQube Scanner plugin.
2. Configure SonarQube in Jenkins:
   * Go to Manage Jenkins → Configure System.
   * Add SonarQube server details (IP, token) and configure SonarQube Scanner under Global Tool Configuration.



## Step 5: Set Up a Jenkins Pipeline for Static Code Analysis

1. Create a new Jenkins Pipeline job.
2. In the pipeline script, add the following basic pipeline code:

pipeline {

agent any

stages {

stage('SCM Checkout') {

steps {

git 'https://github.com/your-repo.git'

}

}

stage('SonarQube Analysis') {

steps {

withSonarQubeEnv('SonarQube') {

sh 'mvn clean verify sonar:sonar'

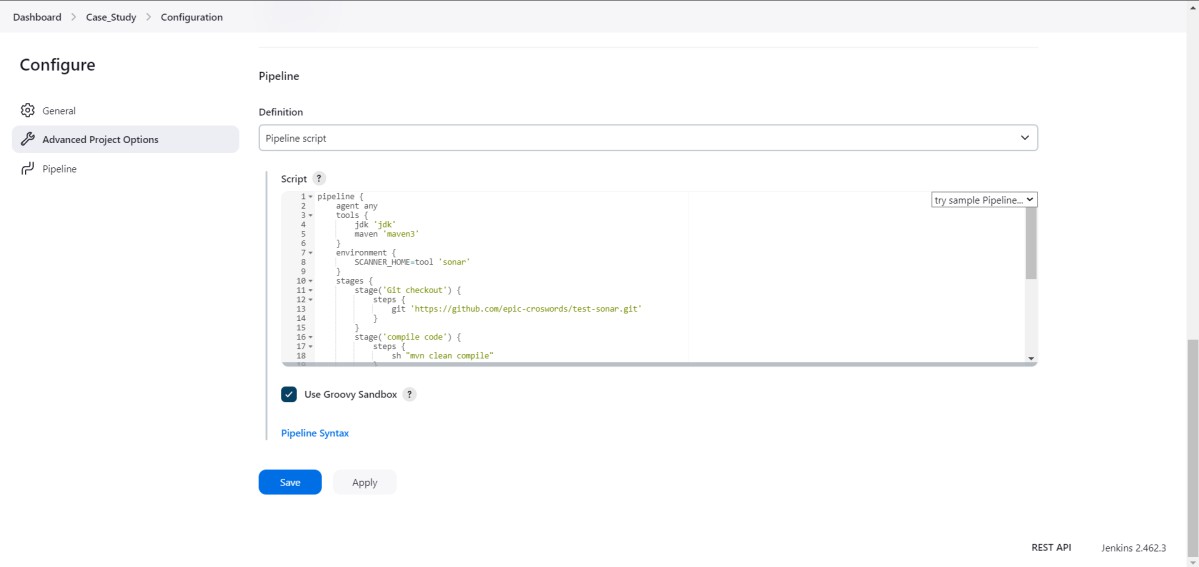
}

}

}

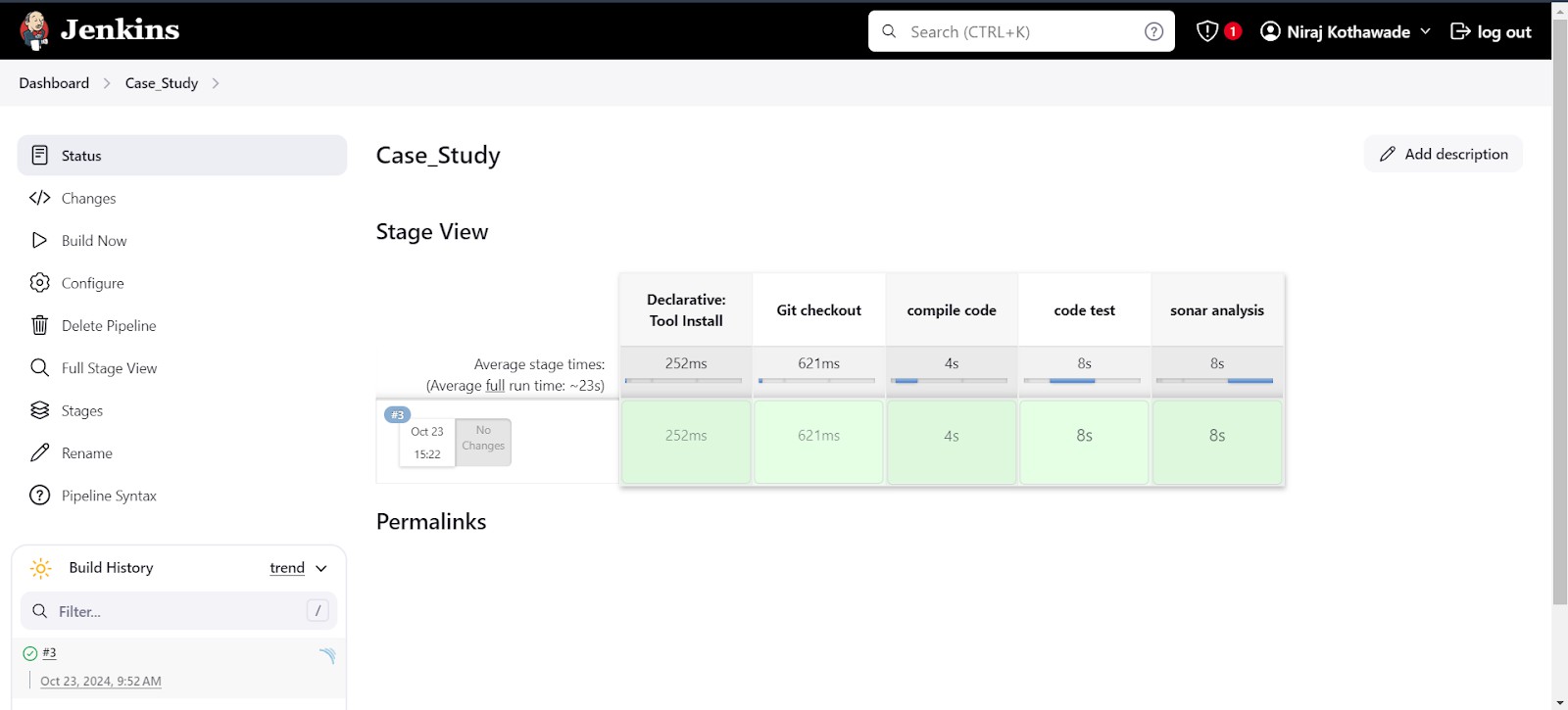
}

}

1. Configure the pipeline to run SonarQube analysis for your project.

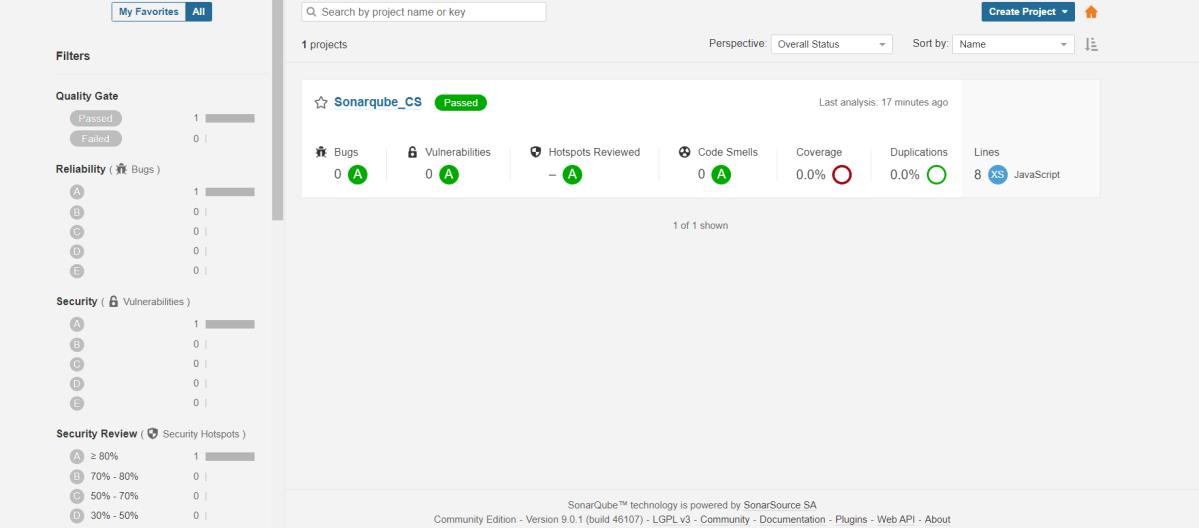
## Step 6: Run the Jenkins Pipeline

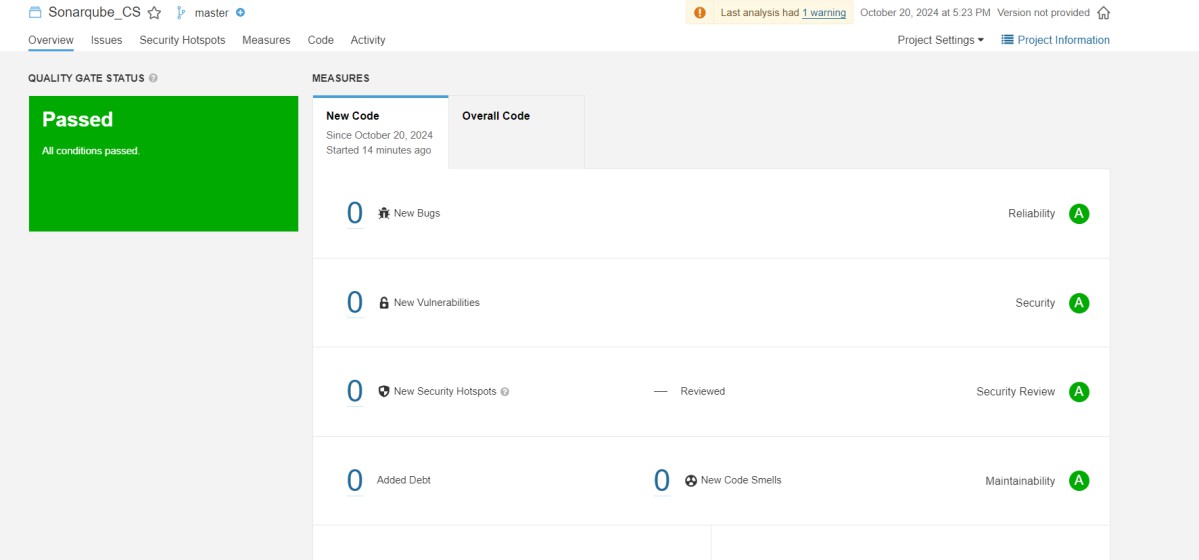
1. Trigger the pipeline manually by clicking Build Now, or set it to run automatically after code push.
2. Monitor pipeline execution via Jenkins console output.



## Step 7: Analyze SonarQube Report

1. Access SonarQube dashboard at http://<EC2-SonarQube-Public-IP>:9000.
2. View code quality reports (bugs, code smells, vulnerabilities) and address issues accordingly.





**UNIQUE FEATURES**

1. **Automated Slack Notifications:**
   * Configure Jenkins to send notifications to a Slack channel whenever the pipeline completes or issues are detected by SonarQube.
   * Use the Slack Notification plugin in Jenkins.
2. **GitHub Webhook Integration:**
   * Set up a GitHub webhook so that any new code pushed to the repository automatically triggers the Jenkins pipeline.
3. **SonarQube Quality Gate Enforcement:**
   * Set up quality gates in SonarQube to automatically fail the Jenkins build if the code doesn't meet specific quality standards (e.g., a certain number of bugs, coverage percentage, or vulnerability issues).
4. **Real-time Code Metrics Dashboard:**
   * Configure a Grafana Dashboard with SonarQube metrics to visualize real-time code quality data. This could include trend graphs of code coverage, technical debt, or issue count.