NAME:ARYAN PATANKAR **ROLL.NO:34** **CLASS:D15A**

Case Study

14. Continuous Integration with Simple Code Analysis

- **Concepts Used**: Jenkins, AWS Cloud9, and SonarQube.
- **Problem Statement**: "Set up a Jenkins pipeline using AWS Cloud9 to perform a simple code analysis on a JavaScript file using SonarQube."
- **Tasks**:
    - Create a Jenkins job using AWS Cloud9.
    - Configure the job to integrate with SonarQube for basic code analysis.
    - Run the Jenkins job with a JavaScript file and review the analysis report.

---

1. Introduction

Case Study Overview:

The chosen case study focuses on setting up **Continuous Integration (CI)** for a JavaScript project using tools like **Jenkins, AWS Cloud9, and SonarQube**. The goal is to create a **Jenkins pipeline** that automates code quality analysis through SonarQube, ensuring high code standards and early detection of potential issues. The pipeline will run on AWS Cloud9, a cloud-based Integrated Development Environment (IDE), where the code is developed, tested, and analyzed. This case study showcases the process of setting up a basic CI pipeline and performing simple code analysis on a JavaScript file to maintain code integrity and improve quality over time.

Key Feature and Application:

Key Feature:

- **Continuous Code Analysis using SonarQube**:
  The unique feature of this case study is the integration of **SonarQube** with Jenkins to automate the analysis of a JavaScript codebase. SonarQube performs a detailed static code analysis, identifying issues such as code smells, bugs, security vulnerabilities, and technical debt, ensuring that the code follows best practices.

Practical Use:

- **Improving Code Quality and Reducing Bugs**:
  In a real-world software development process, integrating **SonarQube** with **Jenkins** allows developers to automatically review their code every time they make changes. This ensures that any potential issues are caught early in the development cycle, preventing buggy or vulnerable code from making its way into production. By using AWS Cloud9 as the development environment, the pipeline ensures scalability and accessibility for remote teams, improving collaboration and speeding up the development process.

Third-Year Project Integration (E-Mart):

The **E-Mart** project is an e-commerce platform that requires a robust and scalable infrastructure to ensure smooth functionality and continuous improvements. By applying **Continuous Integration** principles, you can maintain high code quality and reduce deployment risks, which directly ties into the case study's focus on Jenkins and SonarQube.

Relating to the Case Study:

1. **Jenkins CI Pipeline**:
   - You can set up a **Jenkins pipeline** for the E-Mart project to automatically build, test, and deploy the platform when new features are added, following the approach described in the case study.
   - This ensures that any updates to the platform (such as adding new features like group shopping or personalized recommendations) do not break the existing functionality.
2. **SonarQube Code Analysis**:
   - Similar to the case study where SonarQube was used for basic code analysis on a JavaScript file, you can integrate **SonarQube** in the Jenkins pipeline to continuously analyze the **JavaScript code** of E-Mart for code quality, security vulnerabilities, and maintainability.
   - This ensures that the codebase remains clean, efficient, and secure, which is crucial for the stability of the e-commerce platform.
3. **AWS Cloud9 for Development**:
   - You can leverage **AWS Cloud9** for cloud-based development of the E-Mart platform, ensuring a scalable and collaborative environment for development teams. This aligns with the setup described in the case study and provides an efficient cloud infrastructure to develop, test, and deploy the project.
4. **Problem Resolution & Automation**:
   - Applying CI/CD practices from the case study to **E-Mart** can help streamline the development process by automating testing and deployments. This makes the project more reliable and easier to manage, especially when multiple developers are working on different features like group shopping and social activities.

NAME:ARYAN PATANKAR **ROLL.NO:34** **CLASS:D15A**

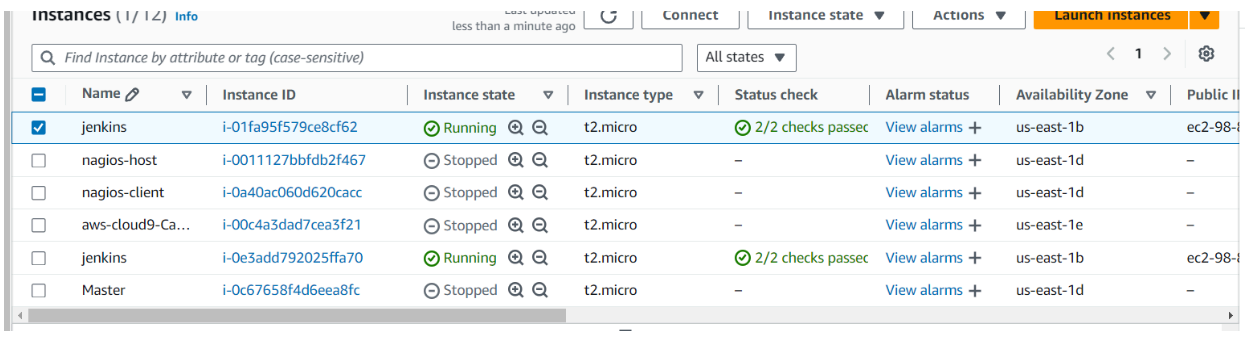
2. Step-by-Step Explanation

Step 1: Initial Setup and Configuration

- **Launch AWS EC2 Instance** for both **jenkins** and **sonarqube** :
  1. Create an AWS account if you haven't.
  2. Launch a **t3.medium** EC2 instance with **Ubuntu 20.04**.
  3. SSH into the instance using a terminal with the command

Allow the following inbound rules:

- **HTTP (port 80)**: For accessing Jenkins.
- **SSH (port 22)**: For secure shell access.
- **Custom TCP (port 8080)**: For accessing Jenkins.


Instances (1/12) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public I
✓	jenkins	i-01fa95f579ce8cf62	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-98-
	nagios-host	i-0011127bbfdb2f467	Stopped	t2.micro	–	View alarms +	us-east-1d	–
	nagios-client	i-0a40ac060d620cacc	Stopped	t2.micro	–	View alarms +	us-east-1d	–
	aws-cloud9-Ca...	i-00c4a3dad7cea3f21	Stopped	t2.micro	–	View alarms +	us-east-1e	–
	jenkins	i-0e3add792025ffa70	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-98-
	Master	i-0c67658f4d6eea8fc	Stopped	t2.micro	–	View alarms +	us-east-1d	–

Step 2: Install Jenkins on EC2 (Ubuntu)

- ssh -i path/to/your-key.pem ubuntu@<your-EC2-IP>
- sudo apt update
- sudo apt install fontconfig openjdk-17-jre
- java -version

# Add the Jenkins repository

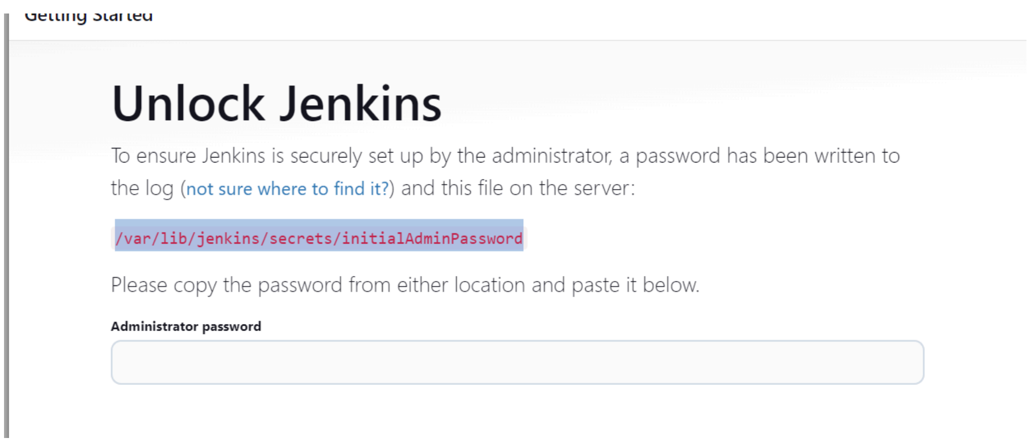
- sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian/jenkins.io-2023.key
- echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian binary/ | sudo tee \ /etc/apt/sources.list.d/jenkins.list > /dev/null
- sudo apt-get update
- sudo apt-get install jenkins

- sudo systemctl start jenkins

- sudo systemctl enable jenkins
- sudo systemctl status jenkins

Open a browser and navigate to `http://<your-EC2-IP>:8080`.

Jenkins status Active


Getting Started

Unlock Jenkins

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

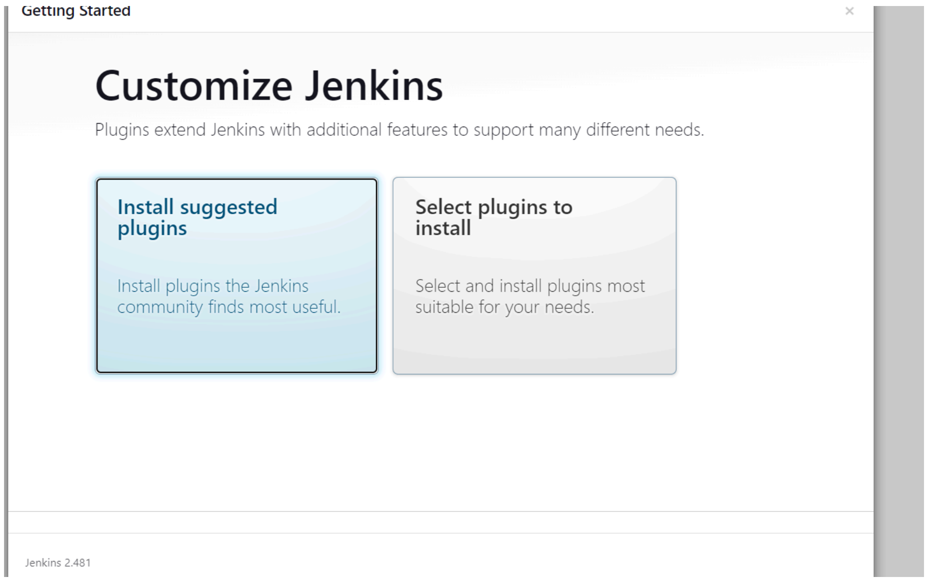
/var/lib/jenkins/secrets/initialAdminPassword

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

Administrator password

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

To get Administrator Password


Getting Started

Customize Jenkins

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

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.481

Step 3: To setup sonarqube in ec2

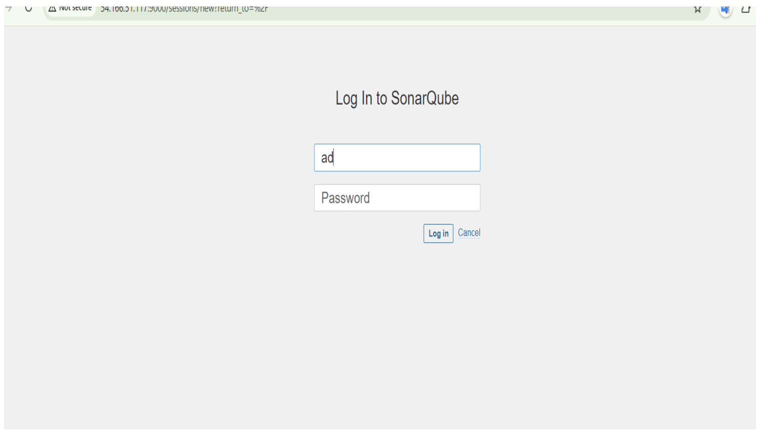
Follow :

[How To Install SonarQube on Ubuntu | by Joyal Saji | Medium](#) till **Setup Systemd Service**

NAME:ARYAN PATANKAR ROLL.NO:34 CLASS:D15A

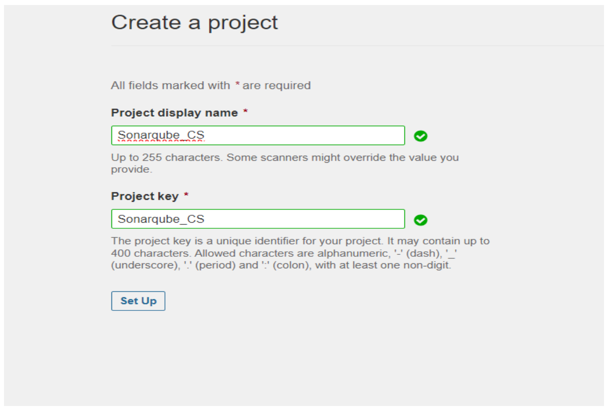
Open a browser and navigate to `http://<your-EC2-IP>:9000`.

SonarQube status active


Log In to SonarQube

ad

Password


Create a project

All fields marked with * are required

Project display name *

Sonarqube_CS

Up to 255 characters. Some scanners might override the value you provide.

Project key *

Sonarqube_CS

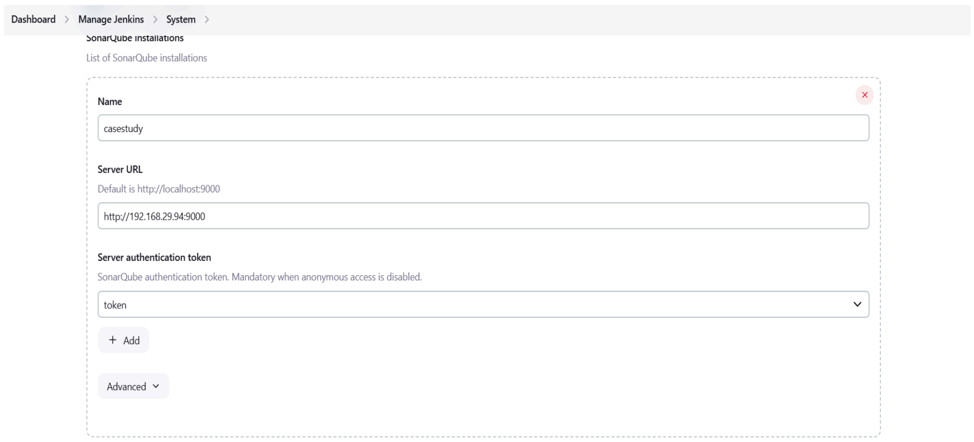
The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Step 4: Integrate Jenkins with SonarQube

1. **Install SonarQube Scanner Plugin in Jenkins**:
   - Go to **Manage Jenkins** → **Manage Plugins**.
   - Search for **SonarQube Scanner** and install it.
2. **Configure SonarQube Server in Jenkins**:
   - Go to **Manage Jenkins** → **Configure System**.
   - Find the **SonarQube servers** section and click **Add SonarQube**.
   - Enter:
     - **Name**: SonarQube
     - **Server URL**: `http://<your-local-IP>:9000` (use your local machine's IP, not `localhost`).
     - **Server authentication token**: Generate a token in SonarQube by going to **My Account** → **Security** → **Generate Tokens**.
3. **Add Credentials in Jenkins**:

NAME:ARYAN PATANKAR ROLL.NO:34 CLASS:D15A

- Go to **Manage Jenkins → Manage Credentials → Add a new credential**.
- Add your SonarQube token as a **Secret Text** credential.

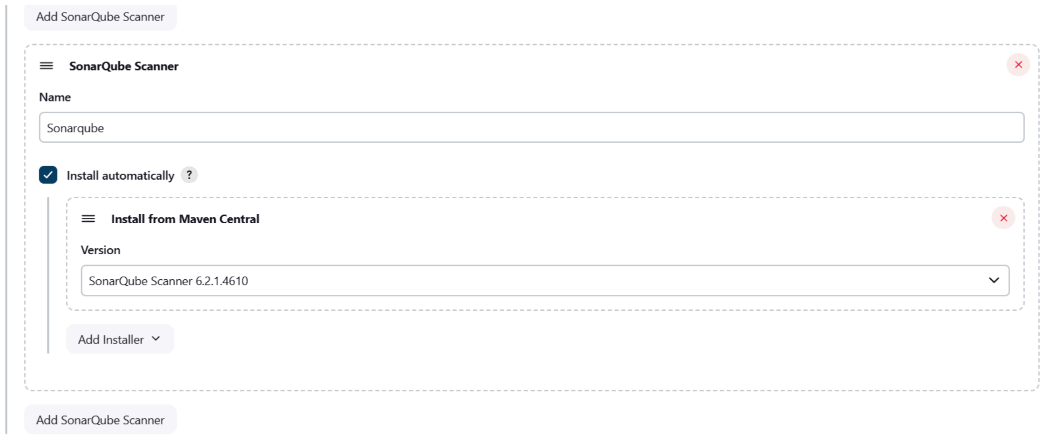

Dashboard > Manage Jenkins > System >
SonarQube installations
List of SonarQube installations
Add
Advanced

Name: casestudy

Server URL (Default is http://localhost:9000): http://192.168.29.94:9000

Server authentication token (SonarQube authentication token. Mandatory when anonymous access is disabled.): token

4. **Set sonarqube Scanner**
   Manage Jenkins → Tools



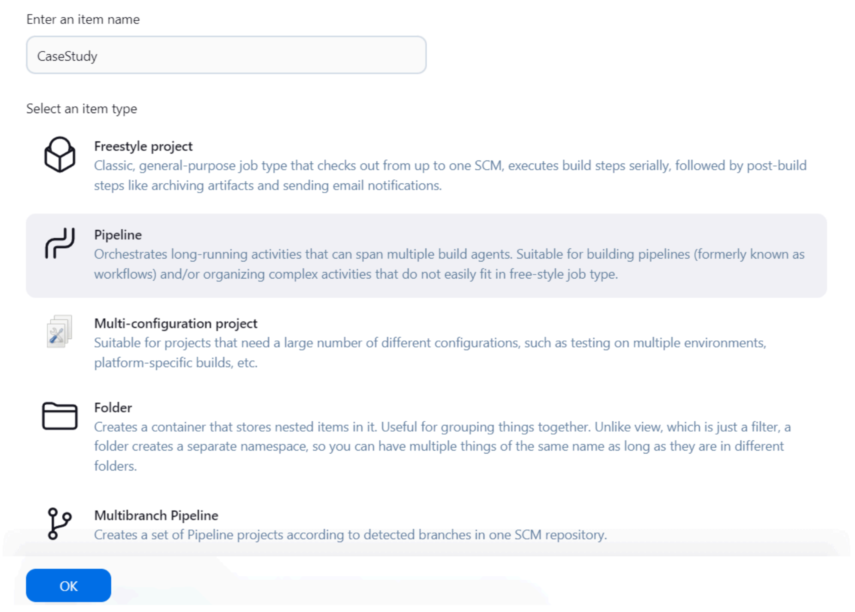
Name: Sonarqube

Install automatically: ✓

Version: SonarQube Scanner 6.2.1.4610

NAME:ARYAN PATANKAR **ROLL.NO:34** **CLASS:D15A**

Step 5: Create Pipeline project



Enter an item name

CaseStudy

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

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

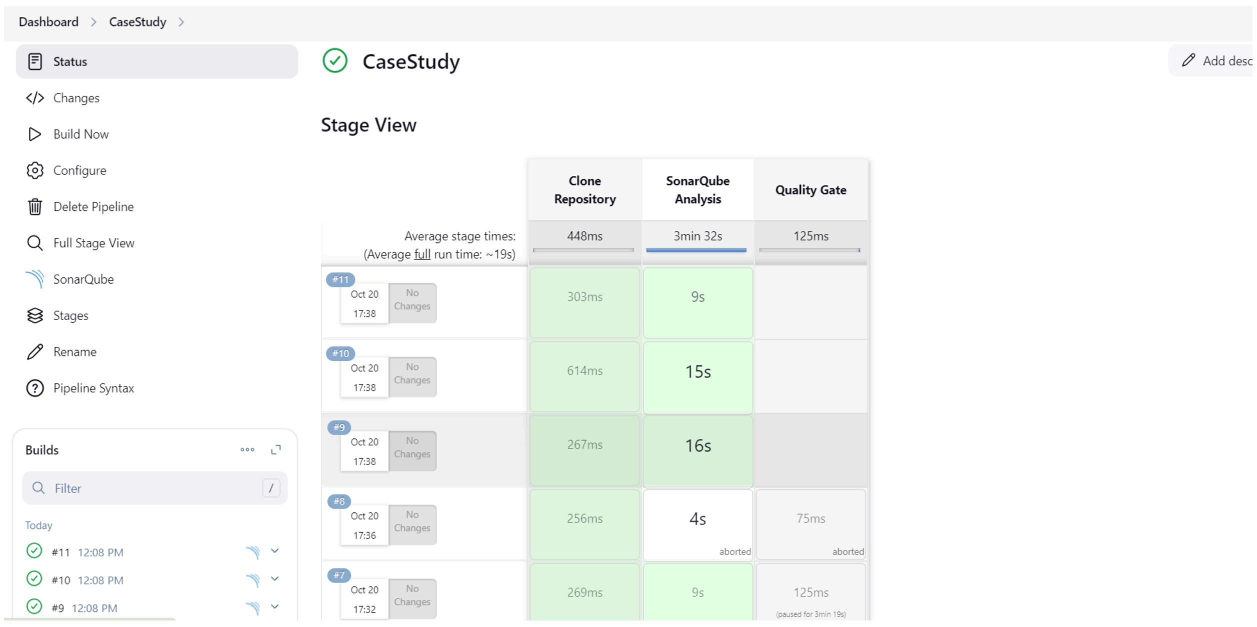
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

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

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

Pipeline code:

After adding pipeline : Build project by clicking **Build Now**



CaseStudy

Stage View

	Clone Repository	SonarQube Analysis	Quality Gate
Average stage times: (Average full run time: ~19s)	448ms	3min 32s	125ms
#11 Oct 20 17:38 No Changes	303ms	9s	
#10 Oct 20 17:38 No Changes	614ms	15s	
#9 Oct 20 17:38 No Changes	267ms	16s	
#8 Oct 20 17:36 No Changes	256ms	4s aborted	75ms aborted
#7 Oct 20 17:32 No Changes	269ms	9s	125ms (paused for 3min 19s)

Builds

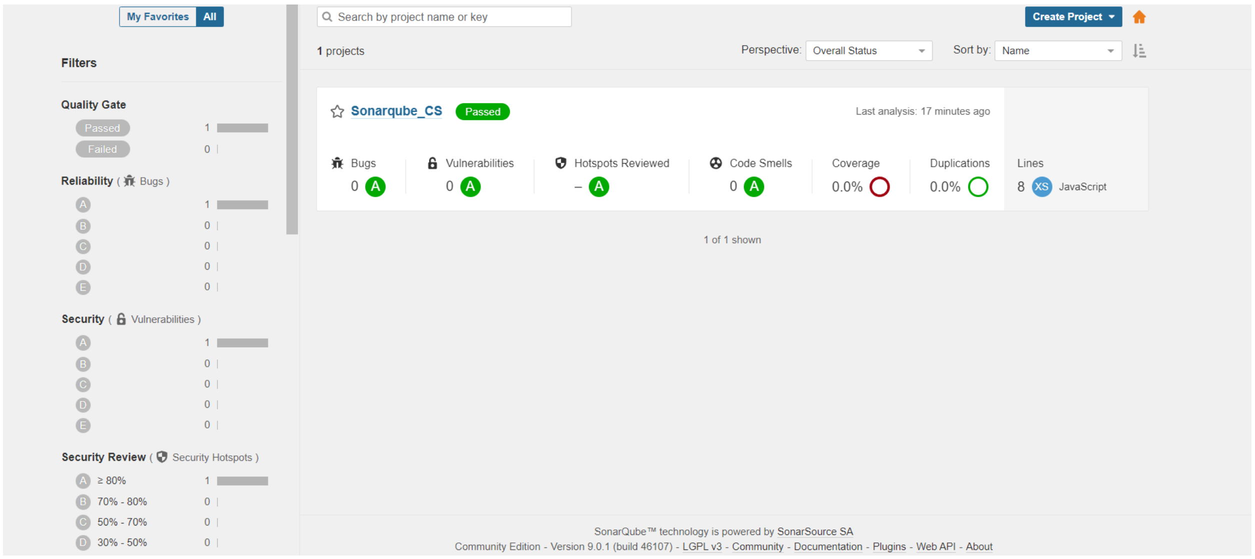
Filter

Today

✓ #11	12:08 PM
✓ #10	12:08 PM
✓ #9	12:08 PM

Sonarqube:

NAME:ARYAN PATANKAR **ROLL.NO:34** **CLASS:D15A**


My Favorites | All

Search by project name or key

Create Project

Filters

1 projects

Perspective: Overall Status Sort by: Name

Quality Gate

Passed	1	
Failed	0	

Reliability (Bugs)

A	1
B	0
C	0
D	0
E	0

Security (Vulnerabilities)

A	1
B	0
C	0
D	0
E	0

Security Review (Security Hotspots)

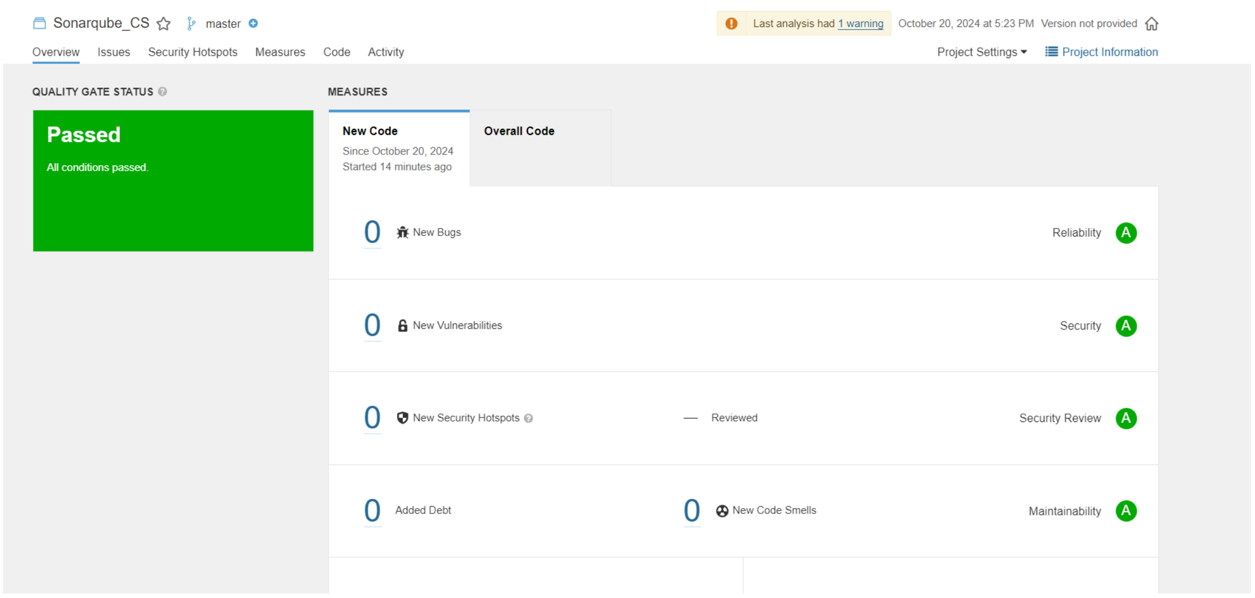
A ≥ 80%	1
B 70% - 80%	0
C 50% - 70%	0
D 30% - 50%	0

Sonarqube_CS Passed Last analysis: 17 minutes ago

Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications	Lines
0 A	0 A	– A	0 A	0.0%	0.0%	8 XS JavaScript

1 of 1 shown

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.0.1 (build 46107) - LGPL v3 - Community - Documentation - Plugins - Web API - About


Sonarqube_CS master

Last analysis had 1 warning October 20, 2024 at 5:23 PM Version not provided

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

QUALITY GATE STATUS

Passed
All conditions passed.

MEASURES

New Code
Since October 20, 2024
Started 14 minutes ago

Overall Code

0 New Bugs		Reliability A
0 New Vulnerabilities		Security A
0 New Security Hotspots	— Reviewed	Security Review A
0 Added Debt	0 New Code Smells	Maintainability A

Conclusion:

Conclusion:

In conclusion, this case study demonstrates the effective use of Continuous Integration (CI) by integrating Jenkins, AWS Cloud9, and SonarQube to maintain code quality in a JavaScript project. By automating code analysis, the setup ensures that potential issues like bugs, code smells, and security vulnerabilities are detected early in the development cycle, enhancing the overall reliability and maintainability of the codebase.

NAME:ARYAN PATANKAR ROLL.NO:34 CLASS:D15A

Through the use of SonarQube, developers receive continuous feedback on their code, promoting best practices and preventing the introduction of low-quality code into production. The cloud-based environment provided by AWS Cloud9 offers flexibility and scalability, allowing for easy collaboration and rapid development, while Jenkins automates the entire process, reducing manual effort and increasing productivity.

Ultimately, this CI pipeline ensures a streamlined, efficient, and high-quality development process, making it a practical and essential solution for modern software projects.