

**Name: Jyoti Fabiyani**

**Class: D15B/14**

## **Case Study: Continuous Integration with Simple Code Analysis**

### **1. Introduction**

This case study focuses on setting up a Continuous Integration (CI) pipeline using Jenkins and an AWS EC2 instance for performing a simple code analysis with SonarQube. The goal is to configure a Jenkins job that will analyze a JavaScript file's code quality using SonarQube, leveraging the EC2 instance as the infrastructure for Jenkins and SonarQube setup.

### **2. Problem Statement**

"Set up a Jenkins pipeline using an AWS EC2 instance to perform a simple code analysis on a JavaScript file using SonarQube."

### **3. Tools and Concepts Used**

- **Jenkins:** Automates software development workflows such as building, testing, and deploying code.
- **AWS EC2:** A virtual server in the cloud, used here to host Jenkins and SonarQube.
- **SonarQube:** A tool for continuous code quality inspection using static code analysis.

### **4. Key Features and Application**

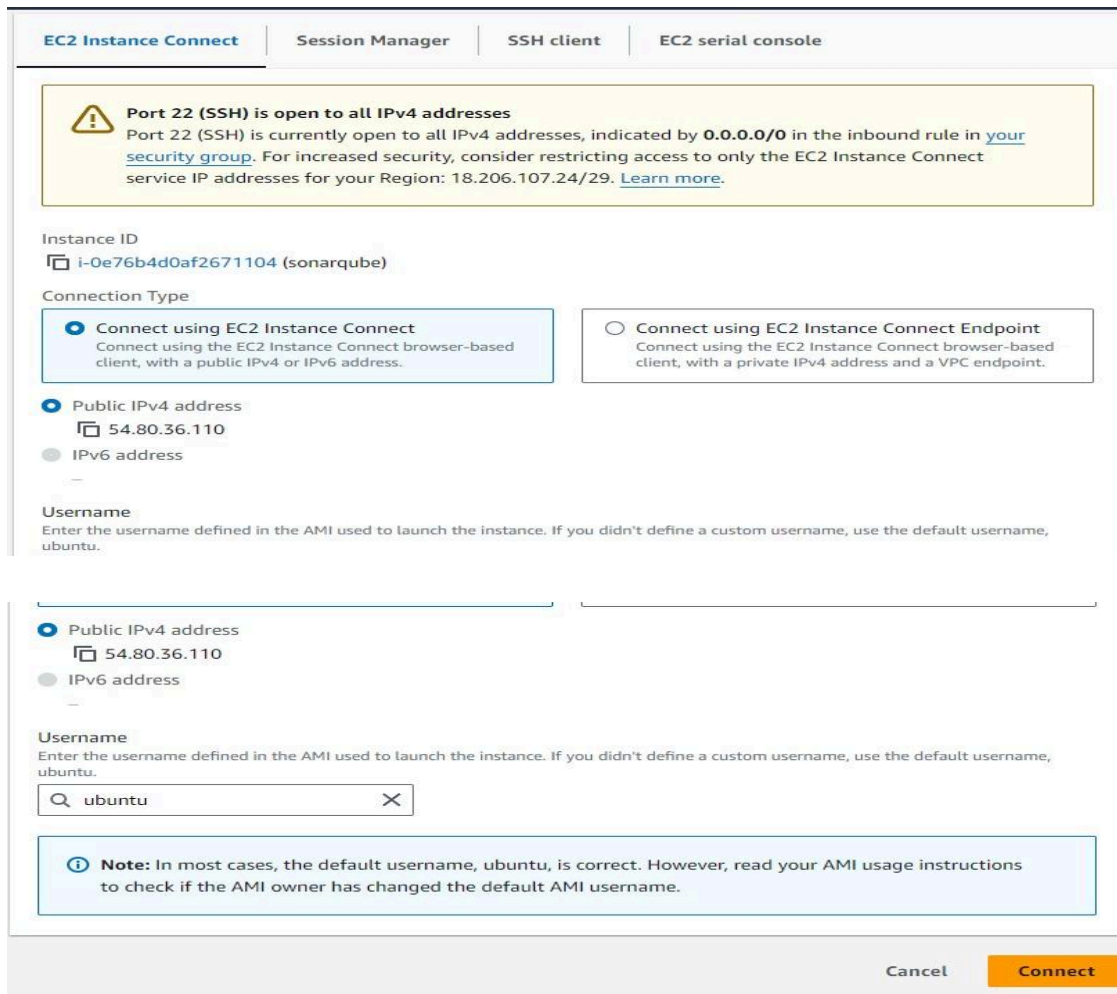
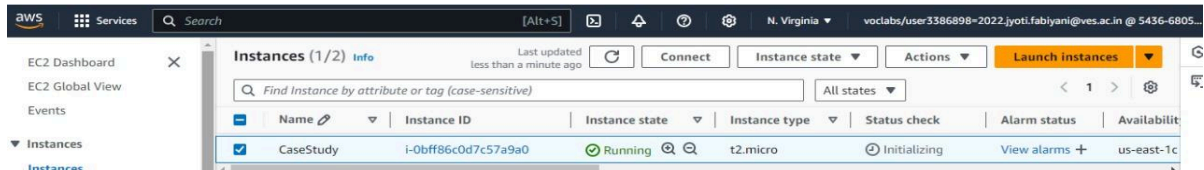
- **Jenkins Pipeline:** Automates the process of running code analysis.
- **AWS EC2:** Provides a flexible cloud-based infrastructure to run Jenkins and SonarQube.
- **SonarQube Integration:** Detects code quality issues, including bugs and vulnerabilities.

### **5. Steps to Set Up the Jenkins Pipeline with SonarQube**

#### **Step 1: EC2 Instance Setup**

## 1. Launch an EC2 instance from the AWS Management Console:

- Choose **Ubuntu** as the Amazon Machine Image (AMI).
- Select the instance type i.e., **t2.micro** and configure security groups to allow access to Jenkins (port 8080) and SonarQube (port 9000).
- Launch the instance and connect it.



## 2. Jenkins Initial Setup using following commands:

- `sudo yum update -y`
- `sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo`
- `sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key`
- `sudo yum install jenkins java-1.8.0-openjdk-devel -y`

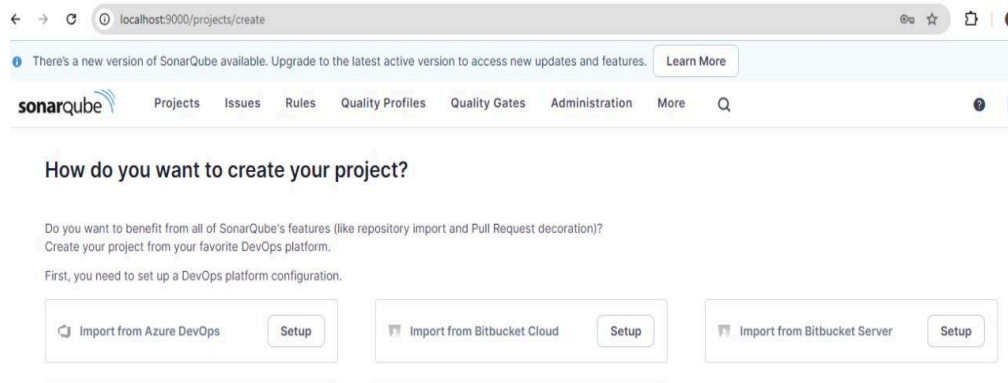
- `sudo systemctl start jenkins`
- `sudo systemctl enable jenkins`

```
ubuntu@ip-172-31-33-119:~$ sudo systemctl restart jenkins
ubuntu@ip-172-31-33-119:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2024-10-15 19:42:38 UTC; 20min ago
     Main PID: 12064 (java)
       Tasks: 39 (limit: 1130)
      Memory: 228.6M
         CPU: 12.025s
        CGroup: /system.slice/jenkins.service
                └─12064 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/jenkins/jenkins.war

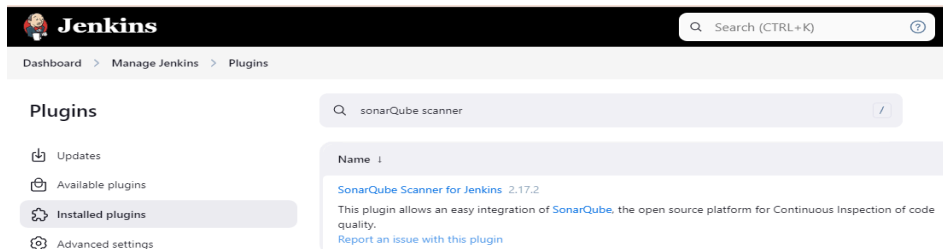
Oct 15 19:42:37 ip-172-31-33-119 jenkins[12064]: WARNING: Please consider
Oct 15 19:42:37 ip-172-31-33-119 jenkins[12064]: WARNING: Use --illegal-access=warn to enable
Oct 15 19:42:37 ip-172-31-33-119 jenkins[12064]: WARNING: All illegal access warnings.
Oct 15 19:42:38 ip-172-31-33-119 jenkins[12064]: 2024-10-15 19:42:38.36
Oct 15 19:42:38 ip-172-31-33-119 jenkins[12064]: 2024-10-15 19:42:38.36
Oct 15 19:42:38 ip-172-31-33-119 jenkins[12064]: 2024-10-15 19:42:38.39
Oct 15 19:42:38 ip-172-31-33-119 jenkins[12064]: 2024-10-15 19:42:38.41
Oct 15 19:42:38 ip-172-31-33-119 jenkins[12064]: 2024-10-15 19:42:38.49
Oct 15 19:42:38 ip-172-31-33-119 jenkins[12064]: 2024-10-15 19:42:38.52
Oct 15 19:42:38 ip-172-31-33-119 systemd[1]: Started Jenkins Continuous
```

## Step 2: Jenkins Initial Setup

1. Open Jenkins from the public IP of the EC2 instance on port 8080 (<http://54.80.36.110:8080>).
2. Install the SonarQube Plugin in Jenkins:
  - Ensure that SonarQube is running on **port 9000**:



- In Jenkins, go to **Manage Plugins**, search for **SonarQube Scanner**, and install it.



3. **Configure SonarQube in Jenkins:**

- Go to **Manage Jenkins > Configure System**, scroll to the **SonarQube Servers** section, and add a new SonarQube server.
- Provide the server URL (e.g., `http://localhost:9000`) and the authentication token from SonarQube.

SonarQube installations

List of SonarQube installations

Name

sonarqube

Server URL

Default is `http://localhost:9000`

`http://localhost:9000`

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

### Step 3: Create a Jenkins Pipeline Job

#### 1. Create a New Pipeline Job:


- In the Jenkins dashboard, click **New Item** and choose **Pipeline** as the job type. Give it a name "javascript".


New Item


Enter an item name


javascript

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.

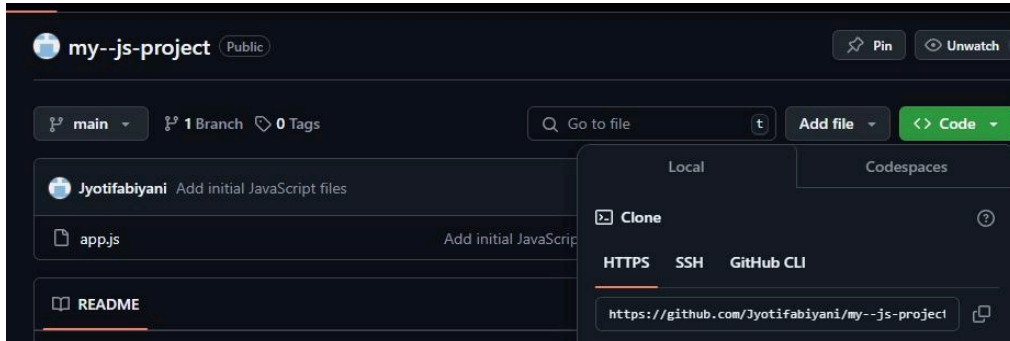
 **Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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

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

OK

- Setup a github repository (my-js-project) and add some simple javaScript code (app.js).



## 2. Configure the Pipeline to Fetch Code from GitHub:

- Choose **Git** as the source control and provide the repository URL where your JavaScript code resides.



## Step 4: SonarQube Code Analysis in the Pipeline Script

### 1. Pipeline script:



## Step 5: Save and Run the Pipeline

- After configuring the pipeline, click "Save".
- To trigger the job, click on "Build Now".

The screenshot displays the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, and the user 'Jyoti Fabiyani'. The breadcrumb trail shows 'Dashboard > javascript > #4'. The left sidebar contains a list of pipeline-related actions: Status, Changes, Console Output (selected), Edit Build Information, Delete build '#4', Timings, Git Build Data, Pipeline Overview, Pipeline Console, Restart from Stage, Replay, and Pipeline Steps. The main area is titled 'Console Output' and shows the following log:

```

Started by user Jyoti Fabiyani
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\jenkins\workspace\javascript
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Checkout)
[Pipeline] checkout
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\jenkins\workspace\javascript\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/Jyotifabiyani/my--js-project.git # timeout=10
Fetching upstream changes from https://github.com/Jyotifabiyani/my--js-project.git
> git.exe --version # timeout=10
> git --version # 'git version 2.46.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/Jyotifabiyani/my--js-project.git +refs/heads/*:refs/remotes/origin/* # timeout=10
Checking out Revision 84cca99cc38838629d4a0561868cd30941290c1d (refs/remotes/origin/main)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 84cca99cc38838629d4a0561868cd30941290c1d # timeout=10
Commit message: "Add initial JavaScript files"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Building...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Testing...
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

- Analysis and Results:

### Stage View



**6. Conclusion:** In this case study, I successfully set up a Jenkins pipeline on an AWS EC2 instance to analyze the quality of a JavaScript file using SonarQube. The integration of Jenkins and SonarQube provides an automated approach to code quality checks, ensuring cleaner and more secure code.