

# 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.

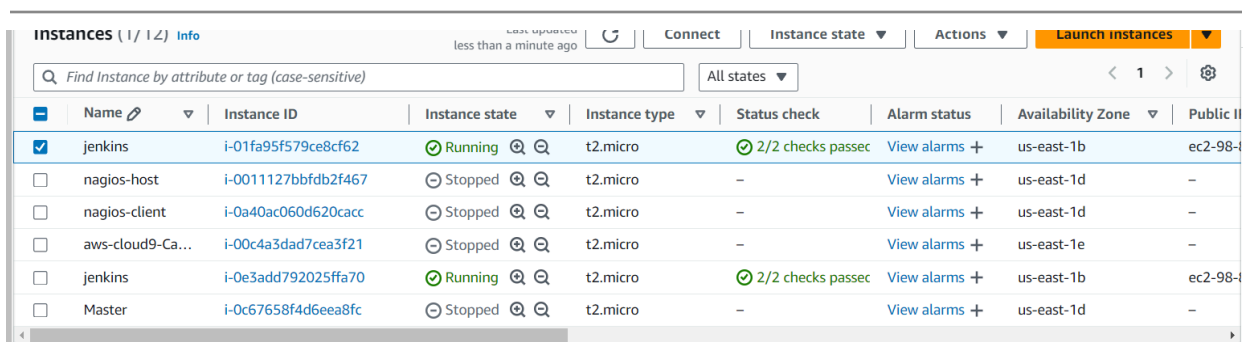
## # Step-by-Step Explanation

### 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 **t2.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.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
<input checked="" type="checkbox"/>	jenkins	i-01fa95f579ce8cf62	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-98-...
<input type="checkbox"/>	nagios-host	i-0011127bbfdb2f467	Stopped	t2.micro	-	View alarms +	us-east-1d	-
<input type="checkbox"/>	nagios-client	i-0a40ac060d620cacc	Stopped	t2.micro	-	View alarms +	us-east-1d	-
<input type="checkbox"/>	aws-cloud9-Ca...	i-00c4a3dad7cea3f21	Stopped	t2.micro	-	View alarms +	us-east-1e	-
<input type="checkbox"/>	jenkins	i-0e3add792025ffa70	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1b	ec2-98-...
<input type="checkbox"/>	Master	i-0c67658f4d6eea8fc	Stopped	t2.micro	-	View alarms +	us-east-1d	-

### Step 2: Install Jenkins on EC2 (Ubuntu)

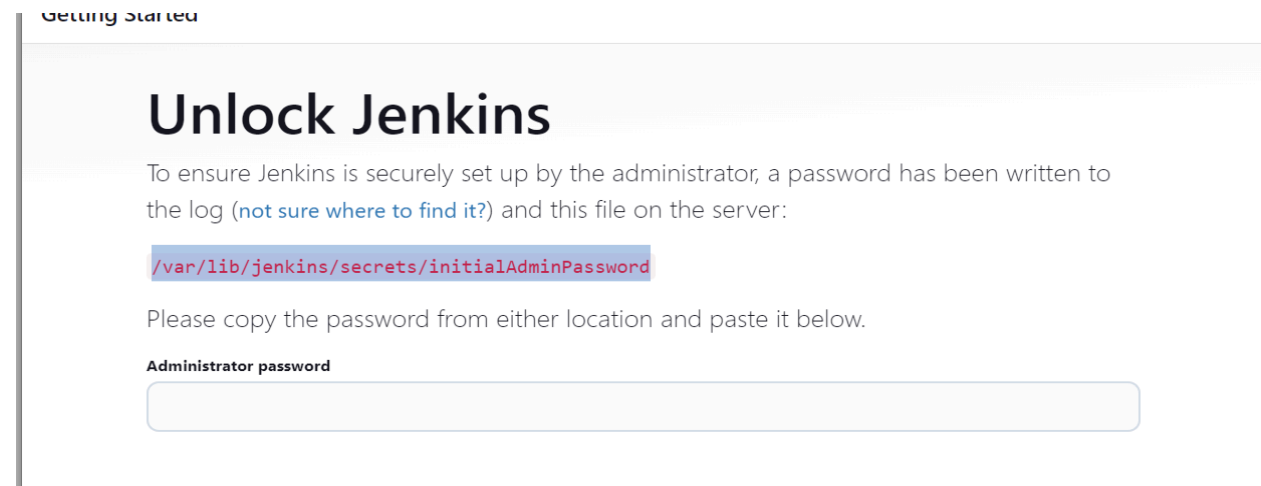
- `ssh -i path/to/your-key.pem ubuntu@<your-EC2-IP>`
- `sudo apt update`

- `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>.



```
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

## Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Q pipel

Install

Install	Name	Released
<input checked="" type="checkbox"/>	<div>SonarQube Scanner 2.17.2</div> <div><a href="#">External Site/Tool Integrations</a> <a href="#">Build Reports</a></div> <div>This plugin allows an easy integration of <a href="#">SonarQube</a>, the open source platform for Continuous Inspection of code quality.</div>	8 mo 3 days ago
<input type="checkbox"/>	<div>Pipeline: REST API 2.34</div> <div><a href="#">User Interface</a></div> <div>Provides a REST API to access pipeline and pipeline run data.</div>	11 mo ago
<input checked="" type="checkbox"/>	<div>Pipeline: Stage View 2.34</div> <div><a href="#">User Interface</a></div> <div>Pipeline Stage View Plugin.</div>	11 mo ago
<input type="checkbox"/>	<div>Docker Pipeline 580.vc0c340686b_54</div> <div><a href="#">pipeline</a> <a href="#">DevOps</a> <a href="#">Deployment</a> <a href="#">docker</a></div> <div>Build and use Docker containers from pipelines.</div>	5 mo 1 day ago

### Step 3: To setup sonarqube in ec2

Follow :

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

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

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.

Set Up

## 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:**
  - 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

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

+ Add

Advanced ▾

#### 4. Set sonarqube Scanner

##### Manage Jenkins → Tools

Add SonarQube Scanner

SonarQube Scanner

Name

Sonarqube

☒ Install automatically ?

Install from Maven Central

Version

SonarQube Scanner 6.2.1.4610

Add Installer ▾

Add SonarQube Scanner

#### Step 5: Create Pipeline project

##### Pipeline code:

```
pipeline {
  agent any
  stages {
    stage('Clone Repository') {
      steps {
        git 'https://github.com/Ashloneer/adv-devops-casestudy'
      }
    }
  }
}
```

```

    }
    stage('SonarQube Analysis') {
        environment {
            SONAR_TOKEN = '48e56dcd98b809d4b5b4e90b25167a191efd9eca' // Your actual
token
        }
        steps {
            withSonarQubeEnv('Sonarqube') {
                sh """
                ${tool 'Sonarqube'}/bin/sonar-scanner \\\
                -Dsonar.projectKey=Sonarqube_CS \\\
                -Dsonar.sources=. \\\
                -Dsonar.host.url=http://54.166.51.117:9000 \\\
                -Dsonar.login=${SONAR_TOKEN}
                """
            }
        }
    }
}

```

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**

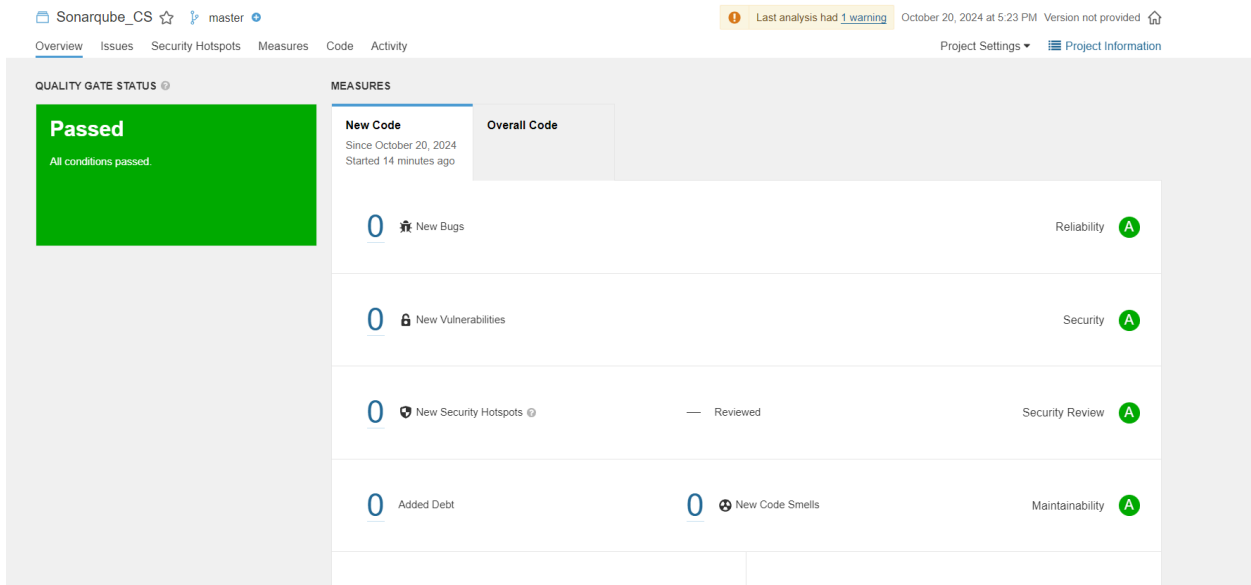
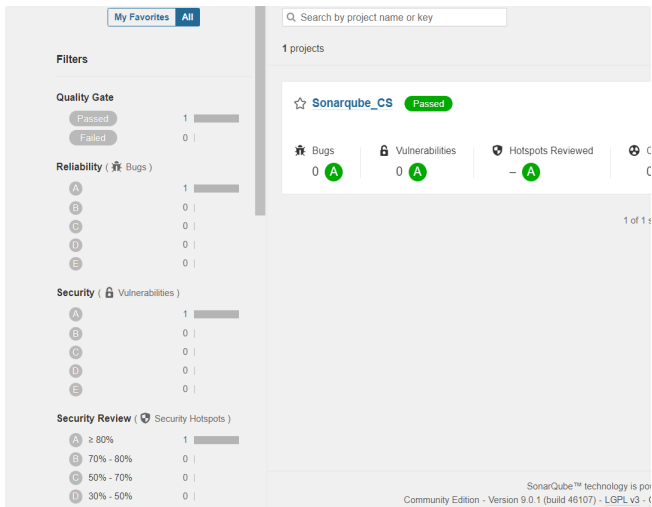
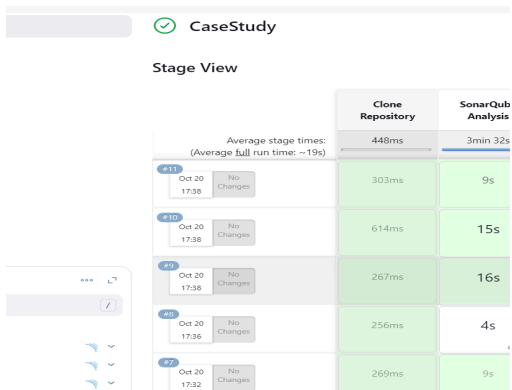
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**Multibranch Pipeline**

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

OK

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



done.....