

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 : This case study is about **Continuous integration (CI)** of pipeline using **Jenkins** and **SonarQube** on **AWS Cloud9**. This setup is used to ensure that analysis of code (Here JavaScript) is automated during the development stage.

Here we have used an **EC2** instance because AWS has stopped giving access to new users from **25th July 2024**. Along with this we have used **GitHub** for code storage and version control.

This Key Feature and Application : The main feature of this case study is to automate the build process with Jenkins combined with SonarQube for analysis. When a new commit is initiated on Github it triggers the Build Now process on Jenkins.

Tools

- **Jenkins:** A CI/CD automation server that helps build, test, and deploy applications efficiently. Jenkins will be used to trigger the pipeline for code analysis.
- **AWS Cloud9:** A cloud-based IDE from AWS that provides a flexible and fully-featured environment for developing, testing, and debugging the code. Cloud9 will serve as the coding platform where the JavaScript file is created and stored.
- **SonarQube:** A tool for automatic code quality inspection, detecting issues such as bugs, security vulnerabilities, and code smells. SonarQube will analyze the JavaScript file for compliance with coding standards.

The following steps will be used to implement the CI pipeline:

1. Development on AWS Cloud9: The JavaScript code file is created and maintained in AWS Cloud9.
2. Jenkins Setup: A Jenkins job is created and configured to execute a pipeline.
3. SonarQube Integration: The Jenkins pipeline integrates SonarQube to analyze the code quality of the JavaScript file.
4. Analysis Report: After the pipeline runs, SonarQube generates a report that highlights code issues like potential bugs, code smells, and security vulnerabilities.

2. Step-by-Step Explanation

Allow the following inbound rules on EC2 instance of Jenkins and SonarQube:

- **HTTP (port 80):** For accessing Jenkins and SonarQube.
- **SSH (port 22):** For secure shell access and SonarQube.
- **Custom TCP (port 8080):** For accessing Jenkins.
- **Custom TCP (port 9000):** For accessing sonarqube.

Edit inbound rules [info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type info	Protocol info	Port range info	Source info	Description - optional info	
sgr-09800e2a77808c1eb	HTTP	TCP	80	Custom	Q	Delete
sgr-0f32134e6817c3613	Custom TCP	TCP	9000	Custom	Q	Delete
sgr-0e5948858dd3e6b03	SSH	TCP	22	Custom	Q	Delete
sgr-005c2ebfd0449e859	Custom TCP	TCP	8080	Custom	Q	Delete

[Add rule](#)

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Preview changes](#) [Save rules](#)

Step 1: Initial Setup and Configuration

1. Launch a **t2.medium** EC2 instance with **Ubuntu**.
2. SSH into the instance using a terminal with the command

Find Instance by attribute or tag (case-sensitive)									
All states									
Instance state = running									
Clear filters									
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Put
<input type="checkbox"/>	sonarqube	i-0414949718d3d024c	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-54-175-252-91.compute-1.amazonaws.com	54.
<input type="checkbox"/>	Jenkins_Cs	i-06cc89c5495b959c7	Running	t2.micro	2/2 checks passed	View alarms	us-east-1b	ec2-54-196-15-121.compute-1.amazonaws.com	54.

Step 2: Install Jenkins on EC2 (Ubuntu)

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


```

Adding debian:USERTrust_RSA_Certification_Authority.pem
Adding debian:XRamp_Global_CA_Root.pem
Adding debian:certSIGN_ROOT_CA.pem
Adding debian:certSIGN_Root_CA_G2.pem
Adding debian:e-Szigno_Root_CA_2017.pem
Adding debian:ePKI_Root_Certification_Authority.pem
Adding debian:emSign_ECC_Root_CA_-_C3.pem
Adding debian:emSign_ECC_Root_CA_-_G3.pem
Adding debian:emSign_Root_CA_-_C1.pem
Adding debian:emSign_Root_CA_-_G1.pem
Adding debian:vTrus_ECC_Root_CA.pem
Adding debian:vTrus_Root_CA.pem
done.
Setting up openjdk-17-jre:amd64 (17.0.12+7-1ubuntu2-24.04) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.10+dfsg-3ubuntu3.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
openjdk version "17.0.12" 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu224.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu224.04, mixed mode, sharing)
ubuntu@ip-172-31-29-196:~$

```

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`

```

ubuntu@ip-172-31-29-196:~$ 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
--2024-10-21 15:25:05-- https://pkg.jenkins.io/debian/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 146.75.30.133, 2a04:4e42:79::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)[146.75.30.133]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins-keyring.as 100%[=====] 3.10K --.-KB/s in 0s

2024-10-21 15:25:05 (40.9 MB/s) - '/usr/share/keyrings/jenkins-keyring.asc' saved [3175/3175]
ubuntu@ip-172-31-29-196:~$

```

- `sudo apt-get update`
- `sudo apt-get install jenkins`

```

ubuntu@ip-172-31-29-196:~$ sudo apt-get update
sudo apt-get install jenkins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian binary/ InRelease
Get:5 https://pkg.jenkins.io/debian binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:8 https://pkg.jenkins.io/debian binary/ Packages [65.3 kB]
Fetched 68.2 kB in 1s (87.7 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 26 not upgraded.
Need to get 94.4 MB of archives.
After this operation, 96.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 net-tools amd64 2.10-0.1ubuntu4 [204 kB]
Get:2 https://pkg.jenkins.io/debian binary/ jenkins 2.481 [94.2 MB]
32% [2 jenkins 25.6 MB/94.2 MB 27%]

```

```

The following additional packages will be installed:
 net-tools
The following NEW packages will be installed:
 jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 26 not upgraded.
Need to get 94.4 MB of archives.
After this operation, 96.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 net-tools amd64 2.10-0.1ubuntu4 [204 kB]
Get:2 https://pkg.jenkins.io/debian binary/ jenkins 2.481 [94.2 MB]
Fetched 94.4 MB in 21s (4440 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 82524 files and directories currently installed.)
Preparing to unpack .../net-tools_2.10-0.1ubuntu4_amd64.deb ...
Unpacking net-tools (2.10-0.1ubuntu4) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../archives/jenkins_2.481_all.deb ...
Unpacking jenkins (2.481) ...
Setting up net-tools (2.10-0.1ubuntu4) ...
Setting up jenkins (2.481) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-29-196:~$

```

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

```

ubuntu@ip-172-31-29-196:~$ sudo systemctl start jenkins
sudo systemctl enable jenkins
sudo systemctl status jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Mon 2024-10-21 15:26:31 UTC; 48s ago
     Main PID: 3944 (java)
       Tasks: 44 (limit: 1130)
      Memory: 341.3M (peak: 367.4M)
         CPU: 18.956s
    CGroup: /system.slice/jenkins.service
            └─3944 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Oct 21 15:26:23 ip-172-31-29-196 jenkins[3944]: c4ad97e6075e491c8e53b146f5f9dbad
Oct 21 15:26:23 ip-172-31-29-196 jenkins[3944]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 21 15:26:23 ip-172-31-29-196 jenkins[3944]: *****
Oct 21 15:26:23 ip-172-31-29-196 jenkins[3944]: *****
Oct 21 15:26:23 ip-172-31-29-196 jenkins[3944]: *****
Oct 21 15:26:31 ip-172-31-29-196 jenkins[3944]: 2024-10-21 15:26:31.330+0000 [id=30] INFO jenkins.InitReactorRunner$1#onAttained:
Oct 21 15:26:31 ip-172-31-29-196 jenkins[3944]: 2024-10-21 15:26:31.374+0000 [id=23] INFO hudson.lifecycle.Lifecycle#onReady: Jen
Oct 21 15:26:31 ip-172-31-29-196 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 21 15:26:31 ip-172-31-29-196 jenkins[3944]: 2024-10-21 15:26:31.526+0000 [id=46] INFO h.m.DownloadService$Downloadable#load:
Oct 21 15:26:31 ip-172-31-29-196 jenkins[3944]: 2024-10-21 15:26:31.528+0000 [id=46] INFO hudson.util.Retrier#start: Performed th
lines 1-20/20 (END)

```

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

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

```
ubuntu@ip-172-31-26-91:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
eb9f423e26f144539e4f0bcbfa3e78cb
```

To get Administrator Password

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

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.

1. Install SonarQube Scanner Plugin in Jenkins:

- Go to **Manage Jenkins** → **Manage Plugins**.
- Search for **SonarQube Scanner** and install it.

The screenshot shows the Jenkins 'Manage Plugins' interface. On the left, a sidebar contains links for 'Updates', 'Available plugins', 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area displays a table of installed and available plugins. The 'SonarQube Scanner' plugin (version 2.17.2) is checked under the 'Install' column. Below it, the 'Pipeline: REST API' (version 2.34) and 'Pipeline: Stage View' (version 2.34) plugins are also listed and checked. The 'Released' column shows the date each plugin was released.

Install	Name	Released
<input checked="" type="checkbox"/>	SonarQube Scanner 2.17.2 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube , the open source platform for Continuous Inspection of code quality.	8 mo 3 days ago
<input type="checkbox"/>	Pipeline: REST API 2.34 User interface Provides a REST API to access pipeline and pipeline run data.	11 mo ago
<input checked="" type="checkbox"/>	Pipeline: Stage View 2.34 User interface Pipeline Stage View Plugin.	11 mo ago

Step 3: Install Sonarqube in new EC2 (Ubuntu)

1. Prepare your Ubuntu server.

```
sudo apt update  
  
sudo apt upgrade -y
```

2. Install OpenJDK 11 - install java development kit 11 or higher version as now

- `sudo apt install -y openjdk-11-jdk`

3. Install and Configure PostgreSQL

- `sudo sh -c 'echo "deb
http://apt.postgresql.org/pub/repos/apt/ `lsb_release
-cs`-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'`

```
ubuntu@ip-172-31-92-10:~$ sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" >> /etc/apt/sources.list.d/pgdg.lis  
ubuntu@ip-172-31-92-10:~$
```

Add PostgreSQL signing key.

- `wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc
-O - | sudo apt-key add -`

```
ubuntu@ip-172-31-92-10:~$ wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo apt-key add -  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).  
OK  
ubuntu@ip-172-31-92-10:~$
```

Install PostgreSQL.

- `sudo apt install -y postgresql postgresql-contrib`

```
ubuntu@ip-172-31-92-10:~$ sudo apt install -y postgresql postgresql-contrib  
sudo systemctl enable postgresql  
sudo systemctl start postgresql  
sudo passwd postgres  
su - postgres  
createuser sonar  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  libcommon-sense-perl libjson-perl libjson-xs-perl libpq5 libtypes-serialiser-perl postgresql-16 postgresql-client-16 postgresql-client-common postgresql  
Suggested packages:  
  postgresql-doc postgresql-doc-16  
The following NEW packages will be installed:  
  libcommon-sense-perl libjson-perl libjson-xs-perl libpq5 libtypes-serialiser-perl postgresql postgresql-16 postgresql-client-16 postgresql-client-commo  
  postgresql-contrib ssl-cert  
0 upgraded, 12 newly installed, 0 to remove and 26 not upgraded.  
Need to get 17.3 MB of archives.  
After this operation, 56.8 MB of additional disk space will be used.  
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libjson-perl all 4.10000-1 [81.9 kB]  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-client-common all 257build1.1 [36.4 kB]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 ssl-cert all 1.1.2ubuntu1 [17.8 kB]  
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-common all 257build1.1 [161 kB]  
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libcommon-sense-perl amd64 3.75-3build3 [20.4 kB]  
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libtypes-serialiser-perl all 1.01-1 [11.6 kB]  
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 libjson-xs-perl amd64 4.030-2build3 [83.6 kB]  
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 libpq5 amd64 16.4-0ubuntu0.24.04.2 [141 kB]  
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-client-16 amd64 16.4-0ubuntu0.24.04.2 [1271 kB]  
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-16 amd64 16.4-0ubuntu0.24.04.2 [15.5 MB]  
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql all 16+257build1.1 [11.6 kB]  
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 postgresql-contrib all 16+257build1.1 [11.6 kB]  
Fetched 17.3 MB in 0s (83.5 MB/s)  
Preconfiguring packages ...
```

Enable DB server to start automatically on reboot.

- `sudo systemctl enable postgresql`

Start DB server.

- `sudo systemctl start postgresql`

Change the default PostgreSQL password.

- `sudo passwd postgres`

Switch to the postgres user.

- `su - postgres`

Create a user named sonar.

- `createuser sonar`

Log into PostgreSQL.

- `psql`
- `ALTER USER sonar WITH ENCRYPTED password '<your_password>';`
- `CREATE DATABASE sonarqube OWNER sonar;`
- `GRANT ALL PRIVILEGES ON DATABASE sonarqube to sonar;`

Exit PostgreSQL.

- `\q`

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
Synchronizing state of postgresql.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable postgresql
New password:
Retype new password:
passwd: password updated successfully
Password:
postgres@ip-172-31-92-10:~$ psql
ALTER USER sonar WITH ENCRYPTED PASSWORD 'my_strong_password';
CREATE DATABASE sonarqube OWNER sonar;
GRANT ALL PRIVILEGES ON DATABASE sonarqube TO sonar;
\q
exit
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
Type "help" for help.
postgres=#
```

Return to your non-root sudo user account.

- `exit`

4. Download and Install SonarQube

Install the zip utility, which is needed to unzip the SonarQube files.

- `sudo apt install -y zip`

Locate the latest download URL from SonarQube official download page. At the time of writing this document, the download URL was as follows:

<https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.0.1.46107.zip>

Download the SonarQube distribution files.

- `sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.0.1.46107.zip`

Unzip the downloaded file.

- `sudo unzip sonarqube-9.0.1.46107.zip`

Move the unzipped files to /opt/sonarqube directory

- `sudo mv sonarqube-9.0.1.46107 /opt/sonarqube`

```
ubuntu@ip-172-31-92-10:~$ sudo apt install -y zip
sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.0.1.46107.zip
sudo unzip sonarqube-9.0.1.46107.zip
sudo mv sonarqube-9.0.1.46107 /opt/sonarqube
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  unzip
The following NEW packages will be installed:
  unzip zip
0 upgraded, 2 newly installed, 0 to remove and 26 not upgraded.
Need to get 350 kB of archives.
After this operation, 933 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 unzip amd64 6.0-28ubuntu4.1 [174 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 zip amd64 3.0-13build1 [175 kB]
Fetched 350 kB in 0s (12.6 MB/s)
Selecting previously unselected package unzip.
(Reading database ... 85623 files and directories currently installed.)
Preparing to unpack .../unzip_6.0-28ubuntu4.1_amd64.deb ...
Unpacking unzip (6.0-28ubuntu4.1) ...
Selecting previously unselected package zip.
Preparing to unpack .../zip_3.0-13build1_amd64.deb ...
Unpacking zip (3.0-13build1) ...
Setting up unzip (6.0-28ubuntu4.1) ...
Setting up zip (3.0-13build1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...
```

5. Add SonarQube Group and User

Create a sonar group.

- `sudo groupadd sonar`
- `sudo useradd -d /opt/sonarqube -g sonar sonar`
- `sudo chown sonar:sonar /opt/sonarqube -R`

```
ubuntu@ip-172-31-92-10:~$ sudo groupadd sonar
ubuntu@ip-172-31-92-10:~$ sudo useradd -d /opt/sonarqube -g sonar sonar
ubuntu@ip-172-31-92-10:~$ sudo chown sonar:sonar /opt/sonarqube -R
ubuntu@ip-172-31-92-10:~$
```

6. Configure SonarQube

Edit the SonarQube configuration file.

- `sudo nano /opt/sonarqube/conf/sonar.properties`

```

GNU nano 7.2 /opt/sonarqube/conf/sonar.properties
# Property values can:
# - be overridden by environment variables. The name of the corresponding environment variable is the
#   upper-cased name of the property where all the dot ('.') and dash ('-') characters are replaced by
#   underscores ('_'). For example, to override 'sonar.web.systemPasscode' use 'SONAR_WEB_SYSTEMPASSCODE'.
# - be encrypted. See https://redirect.sonarsource.com/doc/settings-encryption.html
#-----
# DATABASE
#
# IMPORTANT:
# - The embedded H2 database is used by default. It is recommended for tests but not for
#   production use. Supported databases are Oracle, PostgreSQL and Microsoft SQLServer.
# - Changes to database connection URL (sonar.jdbc.url) can affect SonarSource licensed products.
#
# User credentials.
# Permissions to create tables, indices and triggers must be granted to JDBC user.
# The schema must be created first.
#sonar.jdbc.username=
#sonar.jdbc.password=
#----- Embedded Database (default)
# H2 embedded database server listening port, defaults to 9092
#sonar.embeddedDatabase.port=9092
#----- Oracle 12c/18c/19c
# The Oracle JDBC driver must be copied into the directory extensions/jdbc-driver/oracle/.
# Only the thin client is supported, and we recommend using the latest Oracle JDBC driver. See
# https://jira.sonarsource.com/browse/SONAR-9758 for more details.
# If you need to set the schema, please refer to http://jira.sonarsource.com/browse/SONAR-5000
#sonar.jdbc.url=jdbc:oracle:thin:@localhost:1521/XE
#----- PostgreSQL 9.6 or greater

```

Step 1: Find the following lines.

```
#sonar.jdbc.username=
```

```
#sonar.jdbc.password=
```

Step 2: Uncomment the lines, and add the database user sonar and password my_strong_password you created in Section 3.

- sonar.jdbc.username=sonar
- sonar.jdbc.password=my_strong_password

Step 3: Below those two lines, add sonar.jdbc.url.

- sonar.jdbc.url=jdbc:postgresql://localhost:5432/sonarqube

Save and exit the file.

Edit the sonar script file.

- sudo nano /opt/sonarqube/bin/linux-x86-64/sonar.sh

locate this line. **#RUN_AS_USER=** Uncomment the line and change it to.

- RUN_AS_USER=sonar

Save and exit the file.

7. Setup Systemd Service

- `sudo nano /etc/systemd/system/sonar.service`

Step 1: Paste the following lines to the file.

[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start

ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=sonar

Group=sonar

Restart=always

LimitNOFILE=65536

LimitNPROC=4096

[Install]

WantedBy=multi-user.target

Save and exit the file.

Start SonarQube

- `sudo systemctl enable sonar`
- `sudo systemctl start sonar`

- `sudo systemctl status sonar`

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

The left screenshot shows the 'Log In to SonarQube' page. It has a username field with 'ad' and a password field. Below the fields are 'Log in' and 'Cancel' buttons.

The right screenshot shows the 'Create a project' page. It has two required fields: 'Project display name' and 'Project key', both containing 'Sonarqube_CS'. Below these fields is a 'Set Up' button.

Step 4: Integrate Jenkins with SonarQube

1. **Generate authentication token**: Generate a token in SonarQube by going to **My Account** → **Security** → **Generate Tokens**.

The screenshot shows the SonarQube interface. At the top, there's a notification about a new version. Below that is the navigation bar with 'sonarqube' logo and links to Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and More. The user is logged in as 'Administrator'. The 'Security' tab is selected in the sub-navigation. The 'Generate Tokens' section is active, showing a form to generate a new token. A success message states: 'New token "token" has been created. Make sure you copy it now, you won't be able to see it again!'. Below the message is a copy icon and the token value: 'sqa_d0b37530e18947b4057de7c32a2140dde2a6a8ec'. At the bottom, there is a table of generated tokens.

Name	Type	Project	Last use	Created	Expiration	
token	Global		Never	October 21, 2024	November 20, 2024	Revoke

2. **Add Credentials in jenkins**:

- a) Go to **Manage Jenkins** → **Manage Credentials** → **Add a new credential**.
 - b) Add your SonarQube token as a **Secret Text** credential.
3. **Configure SonarQube Server in Jenkins:**
- a) Go to **Manage Jenkins** → **Configure System**.
 - b) Find the **SonarQube servers** section and click **Add SonarQube**.
 - c) Enter:
 - **Name:** SonarQube or <any name>
 - **Server URL:** *http://<your-new-EC2-Public-IP>:9000* .
 - **Server authentication token:** Use generated token.

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 installer

Manage Jenkins → **Tools** → **SonarQube Scanner** → **Add Installer**

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

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

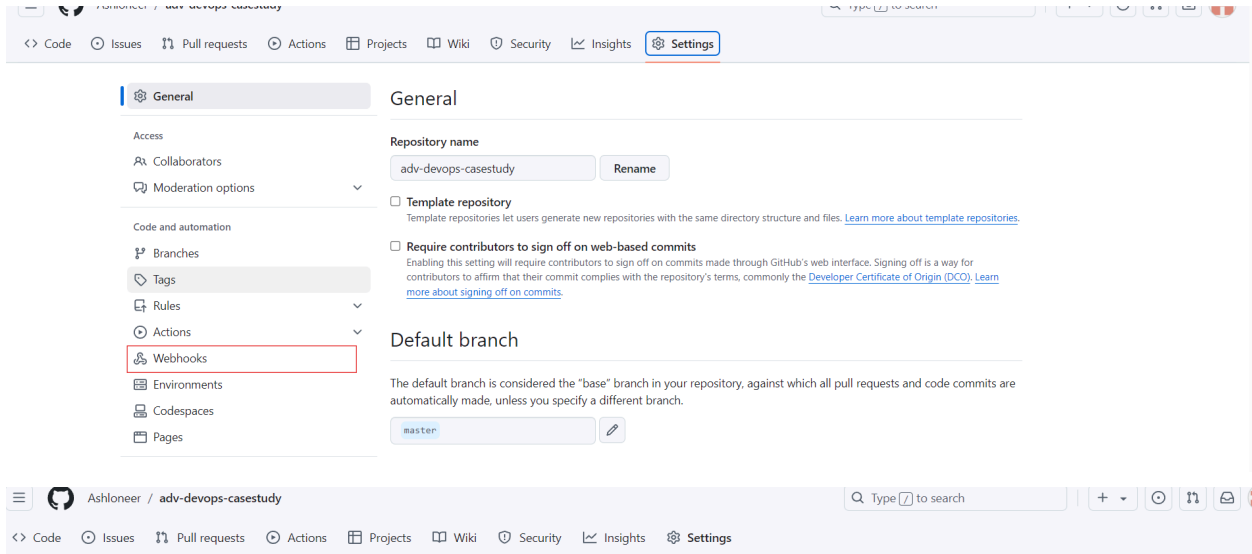
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different

For Continuous Integration:

1) Configure GitHub Webhook:

- Go to your GitHub repository. → Navigate to **Settings > Webhooks** → Click **Add webhook**.

The screenshot shows the GitHub repository page for 'adv-devops-casestudy' by user 'Ashloneer'. The 'Settings' tab is selected in the top navigation bar. The repository is public and has 1 branch (master) and 12 commits. The 'About' section shows 0 stars, 1 watching, and 0 forks. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Languages' section is partially visible at the bottom. The main content area shows the 'proj' directory and a 'README' file. A large green button labeled 'Add a README' is prominently displayed in the center of the page.



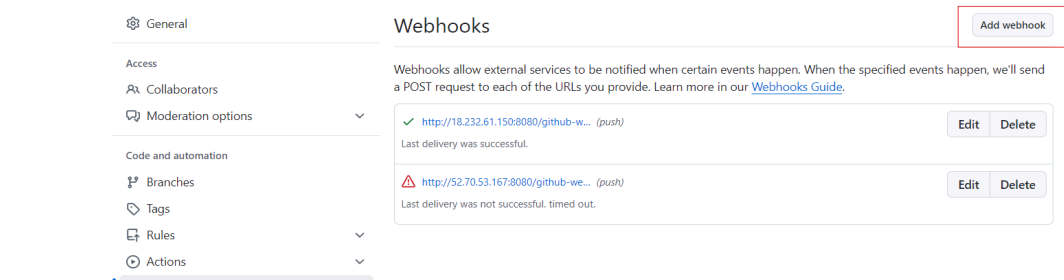
General

Repository name: adv-devops-casestudy

Template repository: ☐

Require contributors to sign off on web-based commits: ☐

Default branch: master



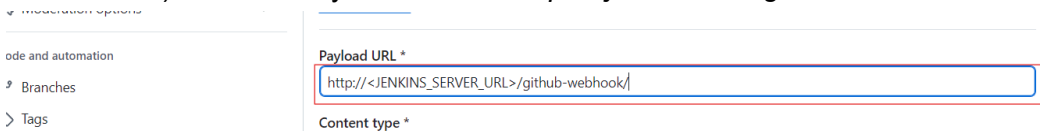
Webhooks

Add webhook

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

URL	Event	Status	Last delivery	Actions
http://18.232.61.150:8080/github-w...	(push)	✓	Last delivery was successful.	Edit Delete
http://52.70.53.167:8080/github-we...	(push)	✗	Last delivery was not successful, timed out.	Edit Delete

b) Set the Payload URL to: *http://<jenkins url>/github-webhook/*.



Payload URL *

http://<JENKINS_SERVER_URL>/github-webhook/

Content type *

c) Choose *application/json* for **Content type**.

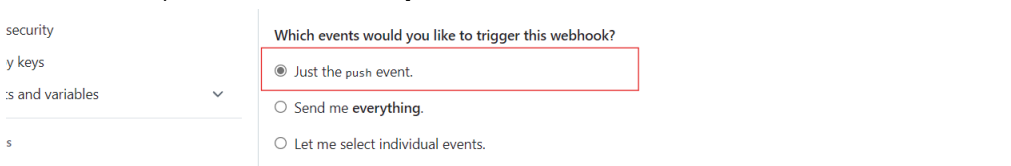


Content type *

application/json

Secret

d) Select **Just the push event**.



Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me everything.

☐ Let me select individual events.

e) Click **Add webhook**

Code and automation

- Branches
- Tags
- Rules
- Actions
- Webhooks**
- Environments
- Codespaces
- Pages

Security

- Code security
- Deploy keys
- Secrets and variables

Integrations

- GitHub Apps
- Email notifications

Payload URL *

http://<JENKINS_SERVER_URL>/github-webhook/A

Content type *

application/json

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

☒ Enable SSL verification ☐ Disable (not recommended)

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ Active

We will deliver event details when this hook is triggered.

Add webhook

Pipeline code:

```

pipeline {
    agent any
    triggers {
        githubPush()
    }
    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 """
                        ${tool 'Sonarqube'}/bin/sonar-scanner \\\
                        -Dsonar.projectKey=Sonarqube_CS \\\
                        -Dsonar.sources=. \\\
                        -Dsonar.host.url=http://184.72.130.101:9000 \\\
                        -Dsonar.login=${SONAR_TOKEN}
                    """
                }
            }
        }
    }
}

```



```
}  
}
```

After adding pipeline : **Save** it and Build project by clicking **Build Now**

Dashboard > case_study_pipeline > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

Pipeline

Definition

Pipeline script

```
1 pipeline {  
2   agent any  
3   triggers {  
4     githubPush()  
5   }  
6   stages {  
7     stage('Clone Repository') {  
8       steps {  
9         git 'https://github.com/Ashloneer/adv-devops-casestudy'  
10      }  
11    }  
12    stage('SonarQube Analysis') {  
13      environment {  
14        SONAR_TOKEN = '48e56dcd98b809d4b5b4e90b25167a191efd9eca' // Your actual token  
15      }  
16      steps {  
17        withSonarQubeEnv('Sonarqube') {  
18          sh '''  
19            ${tool 'Sonarqube'}/bin/sonar-scanner \\  
20            -Dsonar.projectKey=Sonarqube_CS \\  
21            -Dsonar.sources=. \\  
22            -Dsonar.host.url=http://184.72.130.101:9000 \\  
23            -Dsonar.login=${SONAR_TOKEN}  
24          '''  
25        }  
26      }  
27    }  
28  }  
29 }
```

try sample Pipeline...

Save Apply

Status CaseStudy Add desc

Changes Build Now Configure Delete Pipeline Full Stage View SonarQube Stages Rename Pipeline Syntax

Builds

Filter

Today

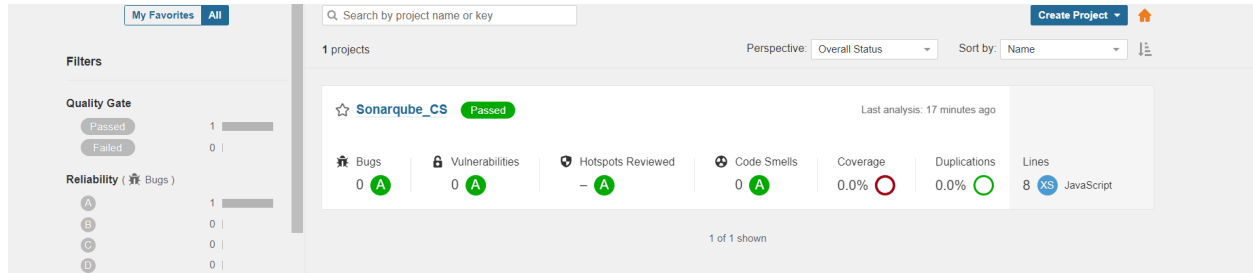
#11 12:08 PM

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	75ms

aborted aborted

Sonarqube:



Guidelines:

- Always update your instance (sudo apt update && sudo apt upgrade).
- Use an instance with storage of at least 4GiB RAM and 2 CPU (t2.medium or higher).

3. Key Points:

Jenkins Automation is the process of automatic builds, where Jenkins pulls the code from the GitHub to execute the builds and integrate changes continuously without any interference of humans.

SonarQube Integration, it is a step where SonarQube is integrated on the static level and analyzes the software at the time of building. Bugs, bugs, vulnerabilities, and code smells have been known to improve the quality of the code.

Rather than local deployment we have used an EC2 instance which is reliable, scalable and flexible.

Practice:

- Run through the demo multiple times to ensure everything works smoothly.
- Confirm that Jenkins and SonarQube are running before starting the presentation.
- Check Public IP each time we start presentation because each time new Public IP is allot to instances and according to that configurations are set on Jenkins