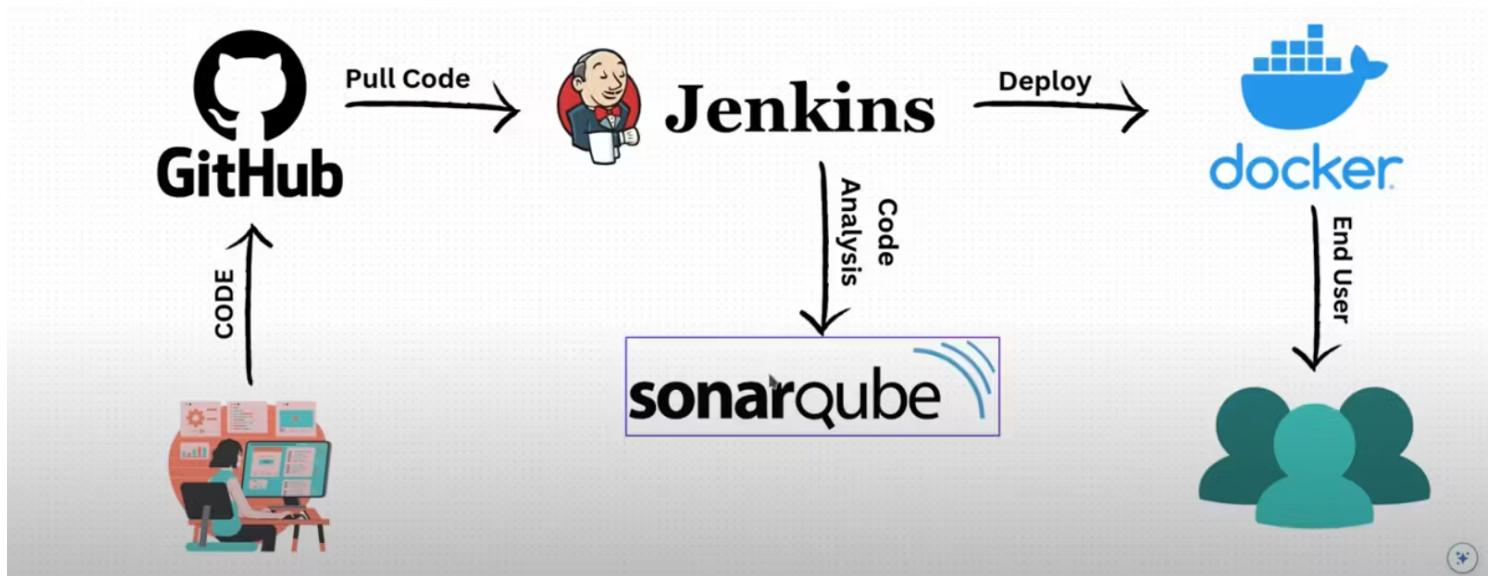




DevOps by Sagar Aulakh



Jenkins CI/CD Pipeline – SonarQube, Docker, Github Webhooks on AWS

SA Sagar Aulakh

Apr 26, 2023 · 7 min read

TABLE OF CONTENTS

- [Launch Instances](#)
- [Install Jenkins](#)
- [Creating a Pipeline in Jenkins Server](#)
- [Adding a Webhooks](#)
- [Starting a SonarQube Server](#)
- [Installing Docker](#)
- [Password-based Server Connectivity](#)
- [Building the Image and running the container](#)

Launch Instances

I have created 3 t2.medium instances

- Instance for Jenkins
- Instance for Sonarqube
- Instance for Docker

The screenshot shows the AWS EC2 Instances page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Tags, and Limits. The main content area displays a table titled "Instances (3) Info" with the following data:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Avai
Jenkins Server	i-031f5900dc31f82c7	Running	t2.medium	-	No alarms	us-e
Sonarqube Server	i-0f660f69ad652925f	Running	t2.medium	Initializing	No alarms	us-e
Docker Server	i-08e7ffec98479bb14	Running	t2.medium	Initializing	No alarms	us-e

Install Jenkins

Jenkins Server



COPY

```
sudo apt update  
clear
```

Now we have to install java first before installing Jenkins

COPY

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

Now we have to paste the Jenkins commands

COPY

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



Now we have to allow port 8080 in the Jenkins server security group inbound settings

Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>
sgr-0af5e974813efdcbb	SSH	TCP	22	Custom ▾	0.0.0.0/0 X
sgr-0492599f316be4209	All traffic	All	All	Custom ▾	0.0.0.0/0 X
-	Custom TCP	TCP	8080	Anywh... ▾	0.0.0.0/0 X

[Add rule](#)

Now use the public URL and port 8080 to view the Jenkins server

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

Continue

Copy the command which was highlighted in the above picture and use the command

COPY

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

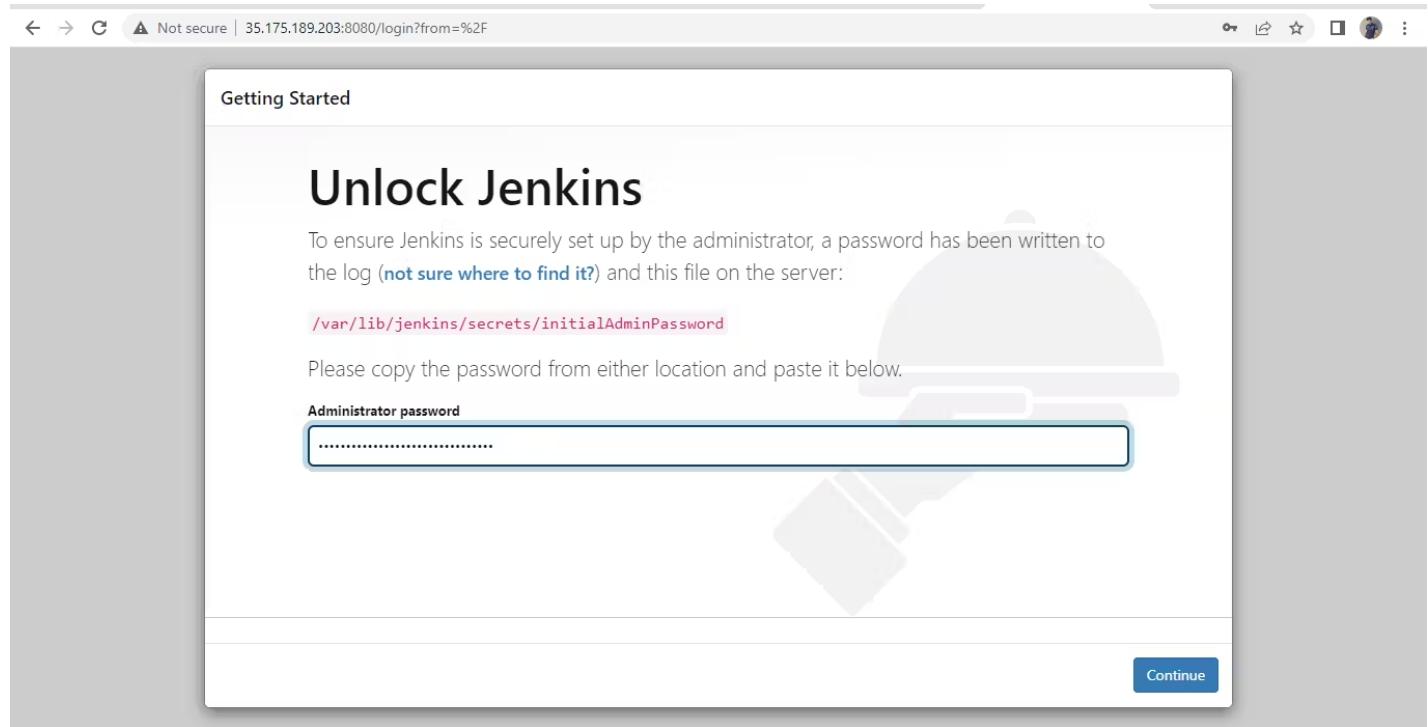
It will show the output a part

```
ubuntu@ip-172-31-20-97:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
2de334ec6d5c42afab841ca42fd91e84  
ubuntu@ip-172-31-20-97:~$
```

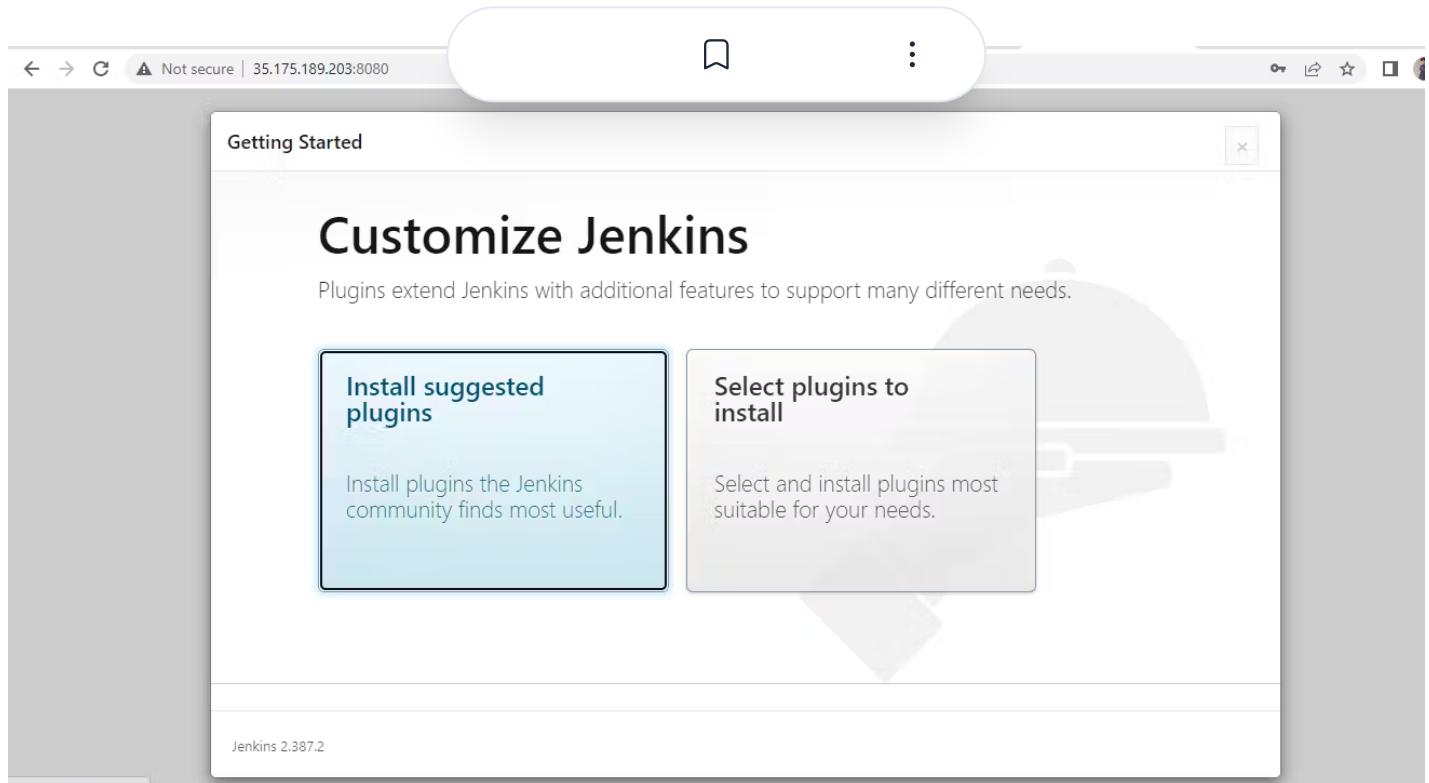
i-031f5900dc31f82c7 (Jenkins Server)

Public IPs: 35.175.189.203 Private IPs: 172.31.20.97

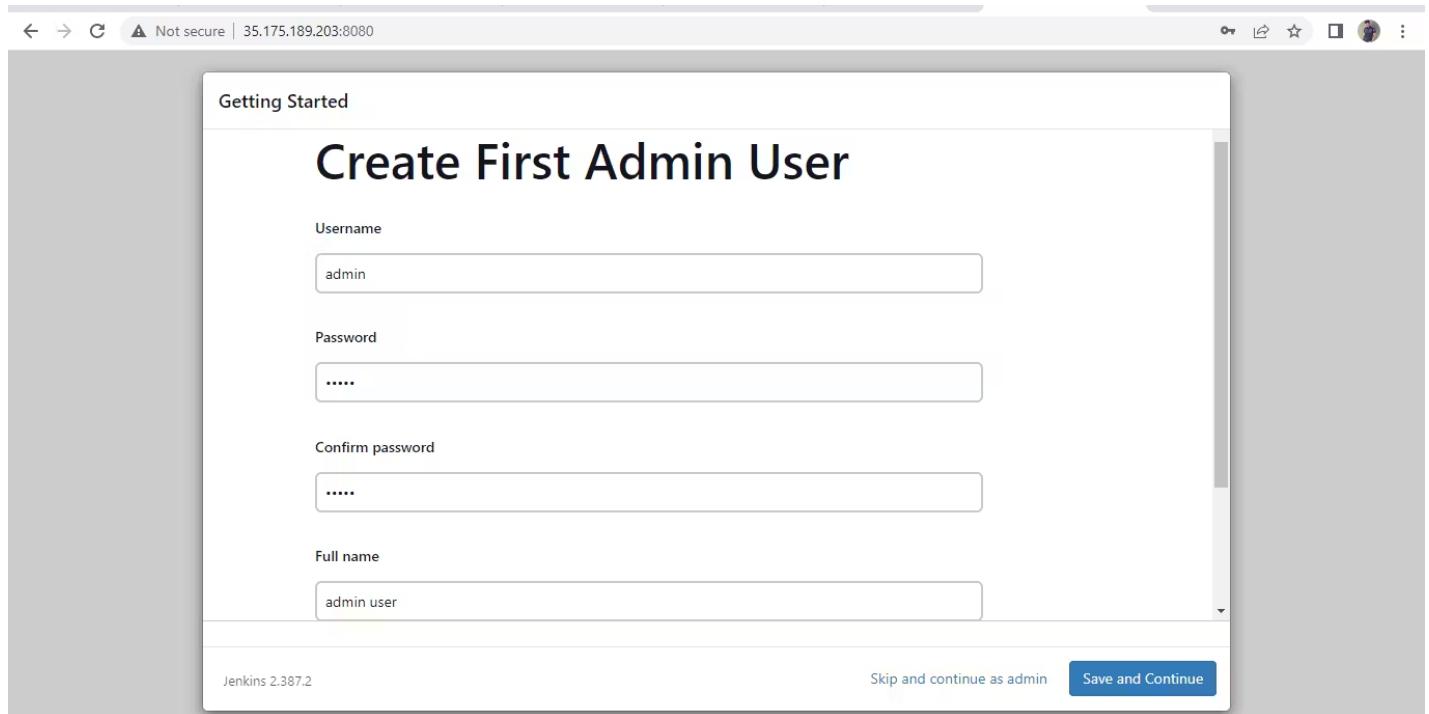
Copy the password and paste it to the Jenkins server



Install suggested plugins



You can set admin credentials



Jenkins Installation done

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job

Set up a distributed build

1 Idle Set up an agent →

2 Idle Configure a cloud →

Learn more about distributed builds ↗

Creating a Pipeline in Jenkins Server

In this, I have selected Freestyle project

Enter an item name

Automated-pipeline

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

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.

OK ↗

In the source code management, I have selected the Git option

The screenshot shows the Jenkins configuration interface for an 'Automated-pipeline' job. The left sidebar has tabs for General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main area is titled 'Source Code Management' and shows 'Git' selected. It includes fields for 'Repository URL' (set to <https://github.com/BroDevOps/website-sample.git>) and 'Credentials' (set to '- none -'). There is also an 'Advanced' dropdown and buttons for 'Save' and 'Apply'.

Also select branch of your git

The screenshot shows the Jenkins configuration interface for an 'Automated-pipeline' job. The left sidebar has tabs for General, Source Code Management (selected), Build Triggers, Build Environment, Build Steps, and Post-build Actions. The main area is titled 'Source Code Management' and shows 'Branches to build' selected. It includes a 'Branch Specifier' field (set to */master) and a 'Repository browser' dropdown (set to '(Auto)'). There is also an 'Advanced' dropdown and buttons for 'Save' and 'Apply'.

Adding a Webhooks

Now open the settings of your Git Repository, select webhooks

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](#).

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules (Beta)

Actions

Webhooks (Selected)

Environments

Codespaces

Click to Add Webhook, that time it will ask you to enter your password, once you enter your password, you're able to add Webhooks.

- Enter the Jenkins URL such as <http://35.175.189.203:8080/>
- and extension github-webhook after the URL

Payload URL *

http://35.175.189.203:8080/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

Just the push event.

Send me everything.

Let me select individual events.

Branch or tag creation
Branch or tag created.

Branch or tag deletion
Branch or tag deleted.

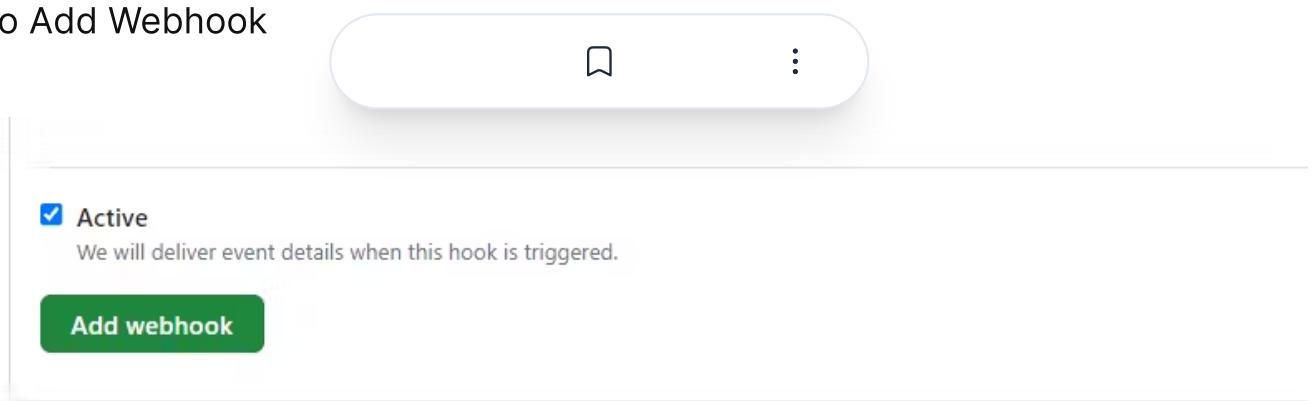
Check runs
Check run is created, requested, rerequested, or completed.

Branch protection rules
Branch protection rule created, deleted or edited.

Code scanning alerts
Code Scanning alert created, fixed in branch, or closed.

Check suites
Check suite is requested, rerequested, or completed.

Click to Add Webhook



Now in Build Triggers, select the **GitHub hook trigger for GITScm polling** (because this function can trigger the pipeline automatically whenever we make changes to the repository).

Configure

Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

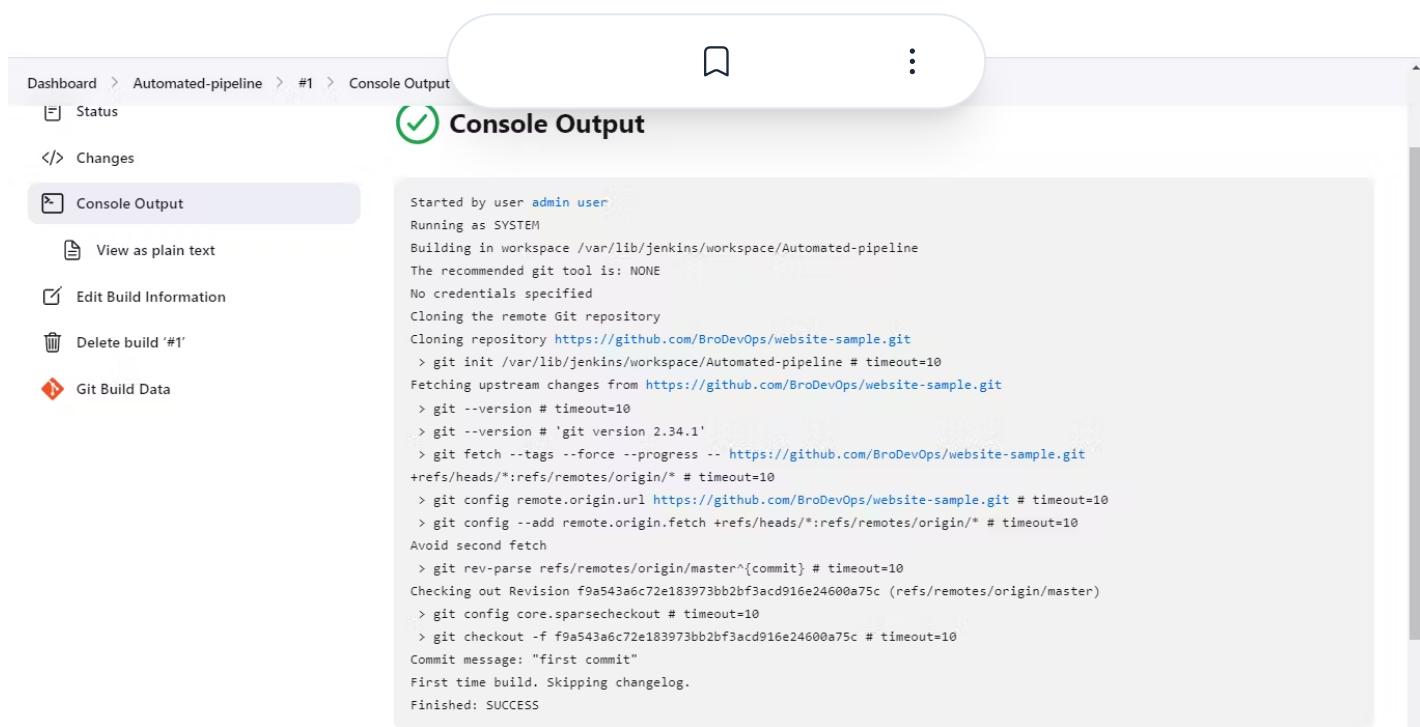
Build Environment

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Add timestamps to the Console Output
- Inspect build log for published build scans

Save Apply

Click to save.

Without Webhook I have clicked to Build now and it is working perfectly fine



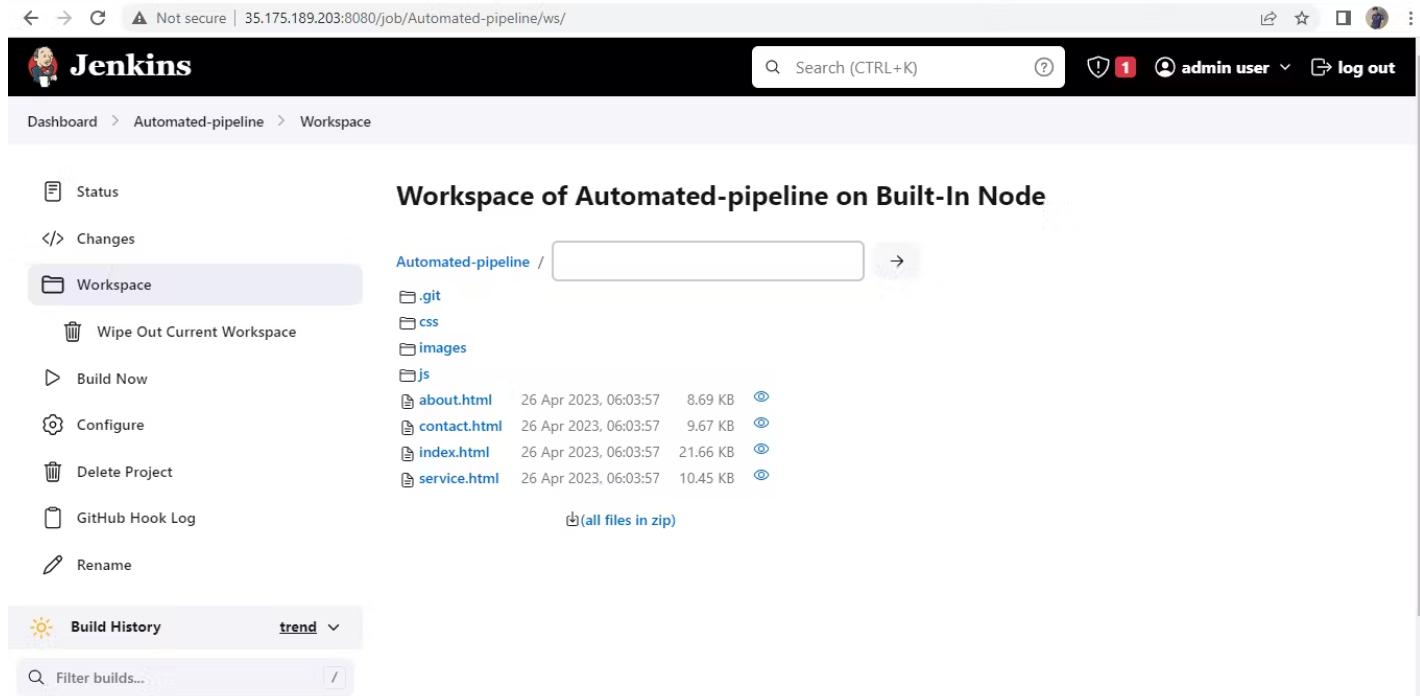
The screenshot shows the Jenkins 'Console Output' page for a build named '#1'. The output log is displayed in a large text area:

```

Started by user admin user
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Automated-pipeline
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
  Cloning repository https://github.com/BroDevOps/website-sample.git
    > git init /var/lib/jenkins/workspace/Automated-pipeline # timeout=10
Fetching upstream changes from https://github.com/BroDevOps/website-sample.git
  > git --version # timeout=10
  > git --version # 'git version 2.34.1'
  > git fetch --tags --force --progress -- https://github.com/BroDevOps/website-sample.git
    +refs/heads/*:refs/remotes/origin/*
  > git config remote.origin.url https://github.com/BroDevOps/website-sample.git # timeout=10
  > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
  > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision f9a543a6c72e183973bb2bf3acd916e24600a75c (refs/remotes/origin/master)
  > git config core.sparsecheckout # timeout=10
  > git checkout -f f9a543a6c72e183973bb2bf3acd916e24600a75c # timeout=10
Commit message: "first commit"
First time build. Skipping changelog.
Finished: SUCCESS

```

Now time to verify Webhooks, I have clicked on the workspace in Jenkins, and here text.txt file is not present which I am going to create to test the Webhook



The screenshot shows the Jenkins 'Workspace' page for the 'Automated-pipeline' project. The workspace directory structure is listed:

- .git
- css
- images
- js
- about.html (26 Apr 2023, 06:03:57) 8.69 KB
- contact.html (26 Apr 2023, 06:03:57) 9.67 KB
- index.html (26 Apr 2023, 06:03:57) 21.66 KB
- service.html (26 Apr 2023, 06:03:57) 10.45 KB

At the bottom, there is a link to download all files as a zip file.

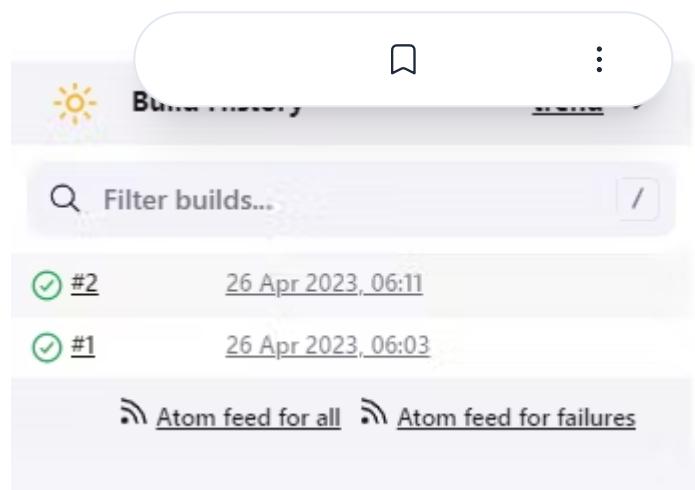
I have visited the Git Repository and created a new file by the name of test.txt

A screenshot of a GitHub repository page for 'BroDevOps / website-sample'. The page shows a single commit for 'test.txt' in the 'master' branch. The commit message is 'This is test file just to test the webhook'. The GitHub interface includes standard navigation links like Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A 'Cancel changes' button is visible in the top right of the commit editor.

I committed the file

A screenshot of a GitHub 'Commit new file' dialog. It shows fields for creating a new file ('Create test.txt') and adding an optional extended description. Two radio button options are present: one selected ('Commit directly to the master branch.') and one unselected ('Create a new branch for this commit and start a pull request.'). At the bottom are 'Commit new file' and 'Cancel' buttons.

Now I am back to the Jenkins server to check #2 build auto trigger and it is working fine.



Dashboard > Automated-pipeline > #2 > Console Output

Status
 </> Changes
 Console Output
[View as plain text](#)
 Edit Build Information
 Delete build '#2'
 Polling Log
 Git Build Data
[← Previous Build](#)

Console Output

```

Started by GitHub push by BroDevOps
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Automated-pipeline
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Automated-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/BroDevOps/website-sample.git # timeout=10
Fetching upstream changes from https://github.com/BroDevOps/website-sample.git
> git --version # timeout=10
> git --version # 'git' version 2.34.1'
> git fetch --tags --force --progress -- https://github.com/BroDevOps/website-sample.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision cc756c07bfbbef7bb35950ff9655923f893ae452 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f cc756c07bfbbef7bb35950ff9655923f893ae452 # timeout=10
Commit message: "Create test.txt"
> git rev-list --no-walk f9a543a6c72e183973bb2bf3acd916e24600a75c # timeout=10
Finished: SUCCESS

```

Now test.txt file is showing in the workspace

The screenshot shows the Jenkins interface for the 'Automated-pipeline' workspace. On the left, there's a sidebar with options: Status, Changes, Workspace (which is selected and highlighted in grey), Wipe Out Current Workspace, Build Now, Configure, Delete Project, and GitHub Hook Log. The main area is titled 'Workspace of Automated-pipeline on Built-In Node'. It shows a directory structure with '.git', 'css', 'images', and 'js' folders. Under 'js', there are five files: 'about.html', 'contact.html', 'index.html', 'service.html', and 'test.txt'. Each file has a timestamp (26 Apr 2023, 06:03:57), size (e.g., 8.69 KB, 9.67 KB, 21.66 KB, 10.45 KB, 43 B), and a preview icon.

Starting a SonarQube Server

Start the Sonarqube Instance

COPY

```
sudo apt update
clear
```

We need to install Java on the server but 17 version earlier we were using 11 version

COPY

```
sudo apt install openjdk-17-jre -y
```

Search for the **SonarQube** website and download the community for the free version

SonarQube Download Copy

Instance

The screenshot shows the SonarQube Downloads page. At the top, there's a navigation bar with links for Deployment, What's New, Roadmap, and Documentation. Below that, it says "Version 10.0 | Released April 2023". The main content area has three columns: "Community Edition", "Developer Edition", and "Enterprise Edition". Each column has a title, a brief description, and a large blue "DOWNLOAD" button. A context menu is open over the "Community Edition" download button, listing options like "Open link in new tab", "Save link as...", and "Copy link address". A purple banner at the bottom states: "SonarSource SA's websites use cookies to distinguish you from other users of our websites. This helps us to provide you with a good experience when you browse our websites and also allows us to improve them." The URL in the address bar is <https://www.sonarsource.com/products/sonarqube/downloads/success-download-community-edition/>.

COPY

```
sudo adduser sonarqube
```

```
ubuntu@ip-172-31-2:~$ Adding user `sonarqube' ...
Adding new group `sonarqube' (1001) ...
Adding new user `sonarqube' (1001) with group `sonarqube' ...
Creating home directory `/home/sonarqube' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for sonarqube
Enter the new value, or press ENTER for the default
      Full Name []:
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:
Is the information correct? [Y/n] y
ubuntu@ip-172-31-22-20:~$
```

i-0f660f69ad652925f (Sonarqube Server)

Public IPs: 54.88.67.190 Private IPs: 172.31.22.20

Use the command below with wget

COPY

`wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip`

```
ubuntu@ip-172-31-22-20:~$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
--2023-04-26 06:49:46-- https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.4.0.54424.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 99.84.191.75, 99.84.191.87, 99.84.191.23, ...
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|99.84.191.75|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 287666040 (274M) [binary/octet-stream]
Saving to: 'sonarqube-9.4.0.54424.zip'

sonarqube-9.4.0.54424.zip          100%[=====] 274.34M   109MB/s    in 2.5s

2023-04-26 06:49:49 (109 MB/s) - 'sonarqube-9.4.0.54424.zip' saved [287666040/287666040]
ubuntu@ip-172-31-22-20:~$ ls
sonarqube-9.4.0.54424.zip
ubuntu@ip-172-31-22-20:~$
```

i-0f660f69ad652925f (Sonarqube Server)

Public IPs: 54.88.67.190 Private IPs: 172.31.22.20

COPY

```
sudo apt install unzip  
unzip *\nls
```

Now the unzipped file is showing

```
ubuntu@ip-172-31-22-20:~$ ls\nsonarqube-9.4.0.54424  sonarqube-9.4.0.54424.zip\nubuntu@ip-172-31-22-20:~$
```

i-0f660f69ad652925f (Sonarqube Server)

PublicIPs: 54.88.67.190 PrivateIPs: 172.31.22.20

COPY

```
cd sonarqube-9.4.0.54424\nls
```

```
ubuntu@ip-172-31-22-20:~$ cd sonarqube-9.4.0.54424\nubuntu@ip-172-31-22-20:~/sonarqube-9.4.0.54424$ ls\nCOPYING  bin  conf  data  dependency-license.json  elasticsearch  extensions  lib  logs  temp  web\nubuntu@ip-172-31-22-20:~/sonarqube-9.4.0.54424$
```

Go to the bin folder

COPY

```
cd bin\ncd linux-x86-64\nls
```

```
ubuntu@ip-172-31-22-20:~/.sonarqube-9.4.0.54424/bin$ ls
jsw-license linux-x86-64 macosx-universal-64 windows-x86-64
ubuntu@ip-172-31-22-20:~/.sonarqube-9.4.0.54424/bin$ cd linux-x86-64
ubuntu@ip-172-31-22-20:~/.sonarqube-9.4.0.54424/bin/linux-x86-64$ ls
lib sonar.sh wrapper
ubuntu@ip-172-31-22-20:~/.sonarqube-9.4.0.54424/bin/linux-x86-64$
```

i-0f660f69ad652925f (Sonarqube Server)

Public IPs: 54.88.67.190 Private IPs: 172.31.22.20

Now start the sonar server by using the command

COPY

`./sonar.sh start`

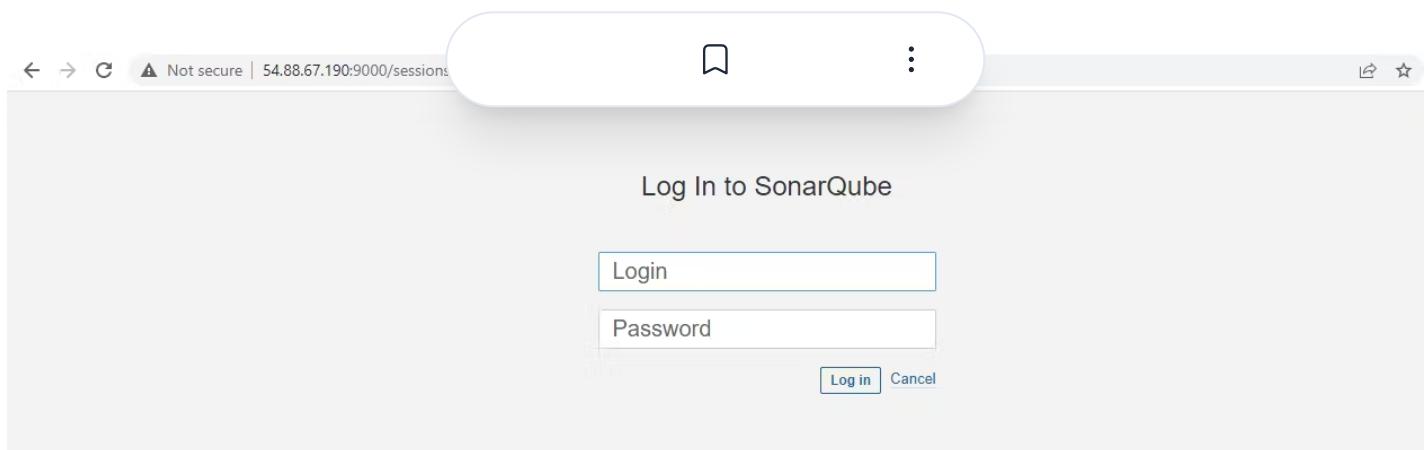
And also we need to port 9000 in the inbound setting of the security group

The screenshot shows the AWS CloudFormation 'Edit inbound rules' interface. It lists four rules:

- Rule 1: Security group rule ID sgr-0af5e974813efdcbb, Type SSH, Protocol TCP, Port range 22, Source 0.0.0.0/0.
- Rule 2: Security group rule ID sgr-0492599f316be4209, Type All traffic, Protocol All, Port range All, Source 0.0.0.0/0.
- Rule 3: A new rule being added, Type Custom TCP, Protocol TCP, Port range 9000, Source Anywhere (0.0.0.0/0).

An 'Add rule' button is visible at the bottom left.

Now it's time to check the URL <http://54.88.67.190:9000/>



Update the password

Update your password

This account should not use the default password.

Enter a new password

All fields marked with * are required

Old Password *

.....

New Password *

.....

Confirm Password *

.....|

Update

After updating the password we can access the Sonarqube

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.

First, you need to set up a DevOps platform configuration.

From Azure DevOps Set up global configuration

From Bitbucket Set up global configuration

From GitHub Set up global configuration

From GitLab Set up global configuration

Are you just testing or have an advanced use-case? Create a project manually.

Manually

Get the most out of SonarQube!

Take advantage of the whole ecosystem by using SonarLint, a free IDE plugin that helps you find and fix issues earlier in your workflow. Connect SonarLint to SonarQube to sync rule sets and issue states.

Learn More **Dismiss**

Now I have selected the project type **Manually**

All fields marked with * are required

Project display name *

Sample-website ✓

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

Project key *

Sample-website ✓

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

After this, I have selected the GitHub Actions integration.

The screenshot shows the SonarQube dashboard for the 'Sample-website' project. At the top, there's a navigation bar with links for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. Below the navigation bar, the project name 'Sample-website' is displayed along with a star icon and the branch 'master'. A blue '+' button is also present. The main content area has tabs for Overview, Issues, Security Hotspots, Measures, Code, and Activity, with 'Overview' being the active tab. A question 'How do you want to analyze your repository?' is followed by four options: 'With Jenkins' (represented by a Jenkins logo), 'With GitHub Actions' (represented by a GitHub Actions logo), 'With Bitbucket Pipelines' (represented by a Bitbucket logo), and 'With GitLab CI' (partially visible on the right). Below this, another question 'Are you just testing or have an advanced use-case? Analyze your project locally.' is shown, accompanied by a local analysis icon (a hand pointing to a computer screen).

Selected the DevOps platform **GitHub**

Analyze your project with Jenkins

Select your DevOps platform

Bitbucket Cloud Bitbucket Server GitHub GitLab

You can just simply select the options **Configure Analysis**

Select your DevOps platform

Prerequisites

! To run your project analyses with Jenkins, the following plugins must be **installed and configured**.

- SonarQube Scanner plugin for Jenkins - version 2.11 or later

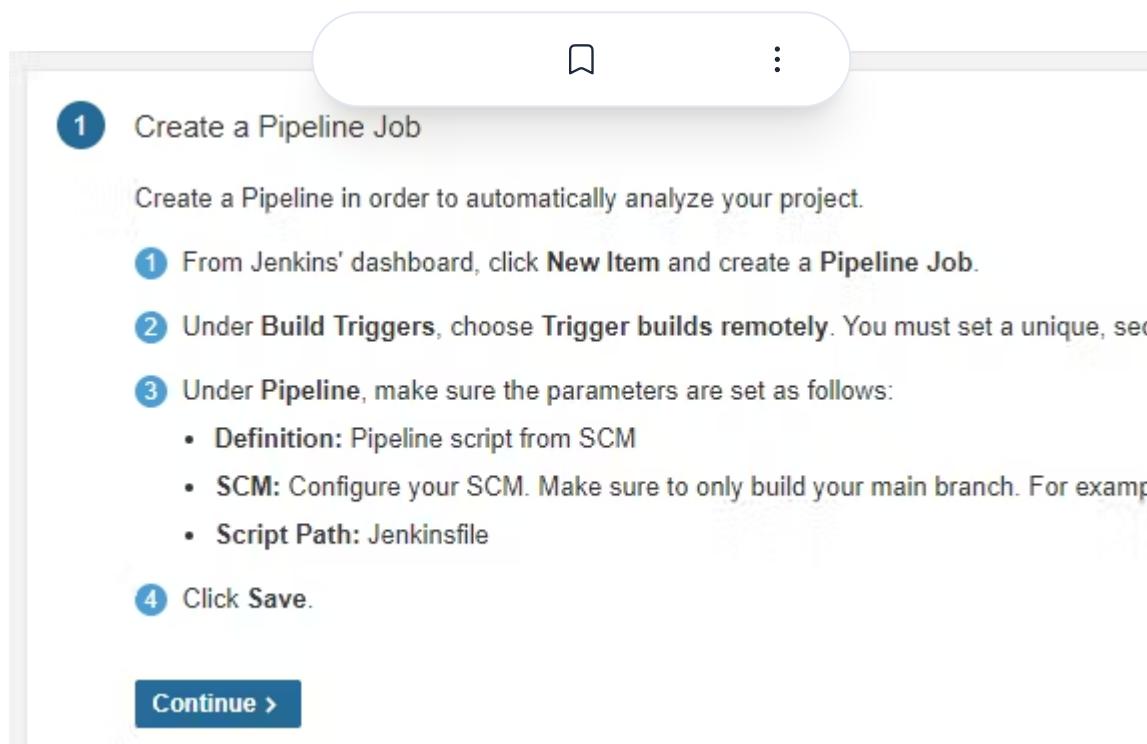
For a step by step guide on installing and configuring those plugins in Jenkins, visit the [Analysis Prerequisites documentation page](#).

We recommend using the configuration in the following steps for the best results, but you can customize it as needed.

Configure Analysis >

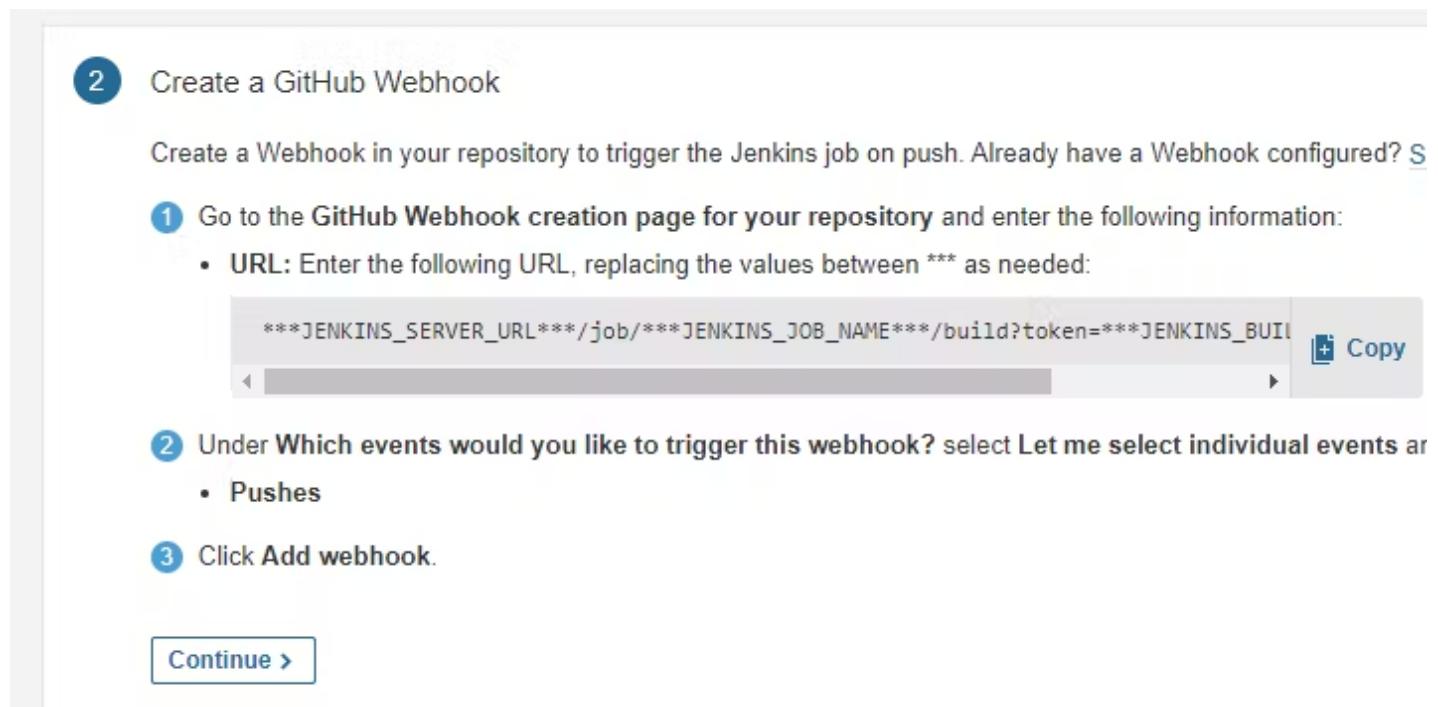
- 1 Create a Pipeline Job
- 2 Create a GitHub Webhook
- 3 Create a Jenkinsfile

Then select **Continue**



The screenshot shows the first step of a Jenkins pipeline creation wizard. The title is "Create a Pipeline Job". The instructions say: "Create a Pipeline in order to automatically analyze your project." Step 1: "From Jenkins' dashboard, click New Item and create a Pipeline Job." Step 2: "Under Build Triggers, choose Trigger builds remotely. You must set a unique, sec..." Step 3: "Under Pipeline, make sure the parameters are set as follows:" with three bullet points: "Definition: Pipeline script from SCM", "SCM: Configure your SCM. Make sure to only build your main branch. For example, Jenkinsfile", and "Script Path: Jenkinsfile". Step 4: "Click Save." A blue "Continue >" button is at the bottom.

Also, click on **Continue** here



The screenshot shows the second step of a Jenkins GitHub webhook creation wizard. The title is "Create a GitHub Webhook". The instructions say: "Create a Webhook in your repository to trigger the Jenkins job on push. Already have a Webhook configured? [See documentation](#)". Step 1: "Go to the [GitHub Webhook creation page](#) for your repository and enter the following information:" with one bullet point: "URL: Enter the following URL, replacing the values between *** as needed:

```
***JENKINS_SERVER_URL***/job/**JENKINS_JOB_NAME***/build?token=***JENKINS_BUILD_TOKEN***
```

" followed by a "Copy" button. Step 2: "Under Which events would you like to trigger this webhook? select Let me select individual events and check Pushes." Step 3: "Click Add webhook." A blue "Continue >" button is at the bottom.

From here I have selected the **Others**

The screenshot shows a Jenkins pipeline configuration page for a 'Sample-website' repository on GitHub. The pipeline has three main steps:

- Create a Jenkinsfile**:
 - What option best describes your build? (Other (for JS, TS, Go, Python, PHP, ...))
- Create a sonar-project.properties file**:

```
sonar.projectKey=Sample-website
```

[Copy](#)
- Create a Jenkinsfile**:

Make sure to replace **SonarScanner** with the name you gave to your SonarQube Scanner tool in Jenkins.

```
node {  
    stage('SCM') {  
        checkout scm  
    }  
    stage('SonarQube Analysis') {  
        def scannerHome = tool 'SonarScanner';  
        withSonarQubeEnv() {  
            sh "${scannerHome}/bin/sonar-scanner"  
        }  
    }  
}
```

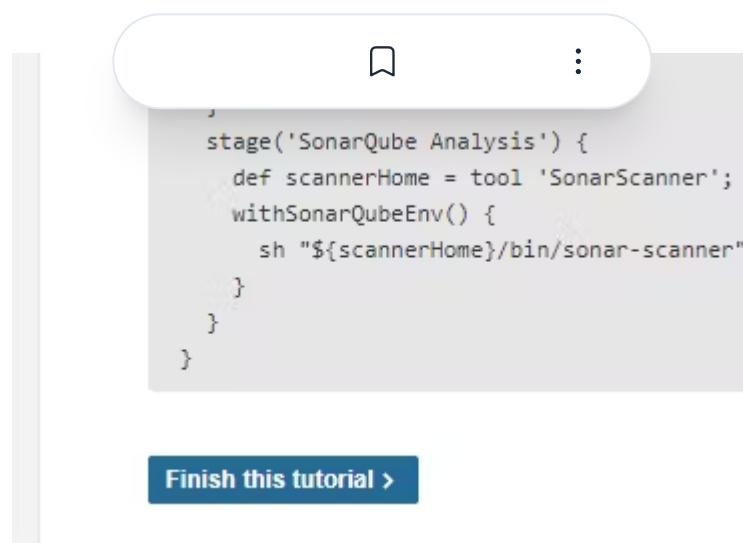
[Copy](#)

Copy the ProjectKey for now you can keep this Key in your notepad

COPY

sonar.projectKey=Sample-website

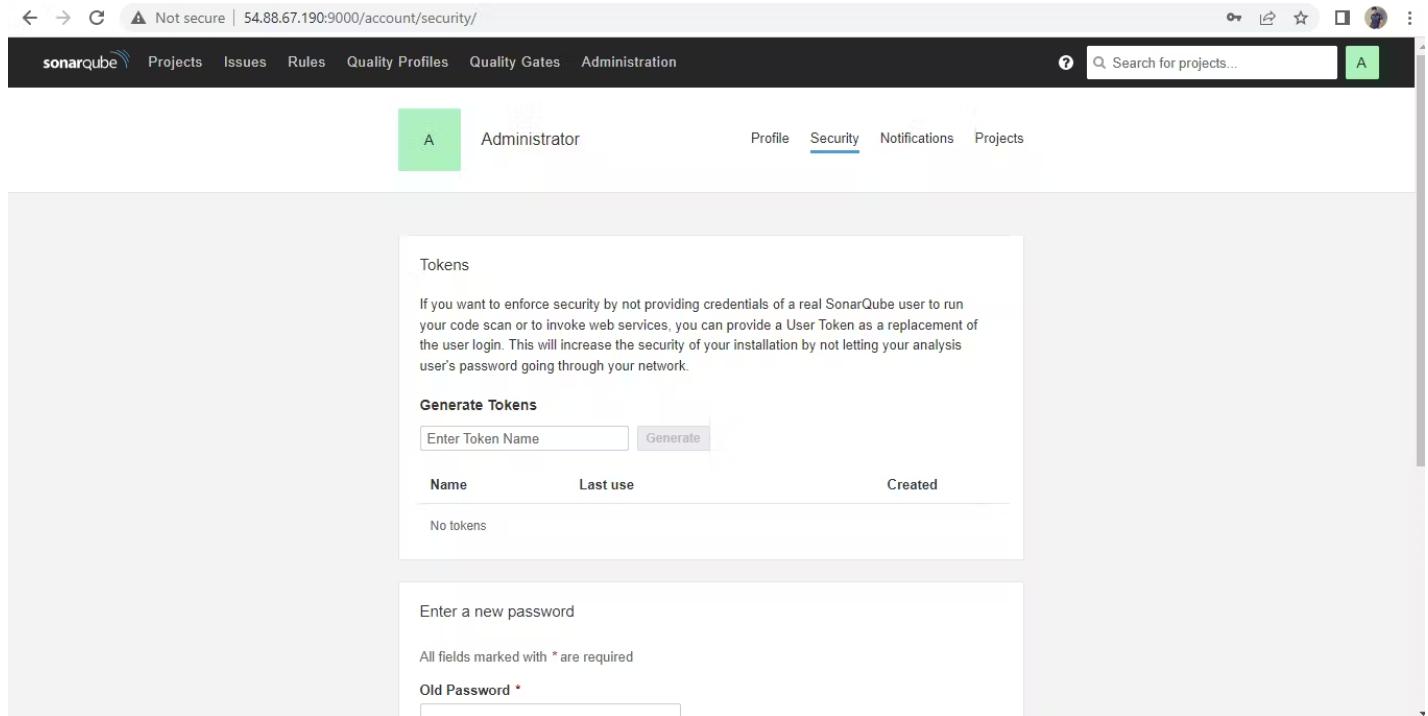
Finish the tutorial



```
stage('SonarQube Analysis') {  
    def scannerHome = tool 'SonarScanner';  
    withSonarQubeEnv() {  
        sh "${scannerHome}/bin/sonar-scanner"  
    }  
}
```

[Finish this tutorial >](#)

Now click to settings, we need to create token



A Not secure | 54.88.67.190:9000/account/security/

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

A Administrator Profile Security Notifications Projects

Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

Enter Token Name Generate

Name	Last use	Created
No tokens		

Enter a new password

All fields marked with * are required

Old Password *

Now I have created the Token and copy the token for now you can keep this token in your notepad

The screenshot shows the SonarQube interface with a navigation bar at the top. The main content area is titled "Tokens". It contains instructions about generating tokens for security purposes. A "Generate Tokens" section has a text input field "Enter Token Name" and a "Generate" button. Below this, a message box says "New token 'Sonarqube-Token' has been created. Make sure you copy it now, you won't be able to see it again!" with a "Copy" button and the token value "3ed473cad250b28bb5e20a53dd6cb12679f2b6dd". A table lists the generated token with columns "Name", "Last use", and "Created". The token details are: Name: Sonarqube-Token, Last use: Never, Created: April 26, 2023, with a "Revoke" button.

Now back to Jenkins, Manage Jenkins and Install the plugin

The screenshot shows the Jenkins "Plugin Manager" page. The left sidebar has links for "Updates", "Available plugins" (which is selected), "Installed plugins", "Advanced settings", and "Download progress". The main area is titled "Plugins" and has a search bar with "sonar". A table lists available plugins. The "SonarQube Scanner 2.15" plugin is selected for installation, indicated by a checked checkbox. The plugin details are: Name: SonarQube Scanner 2.15, Version: 2.15, Status: Released, Last updated: 5 mo 4 days ago. The description states: "This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality." Below the table, there is a warning message: "Warning: This plugin version may not be safe to use. Please review the following security notices:". At the bottom, there are two buttons: "Install without restart" and "Download now and install after restart".

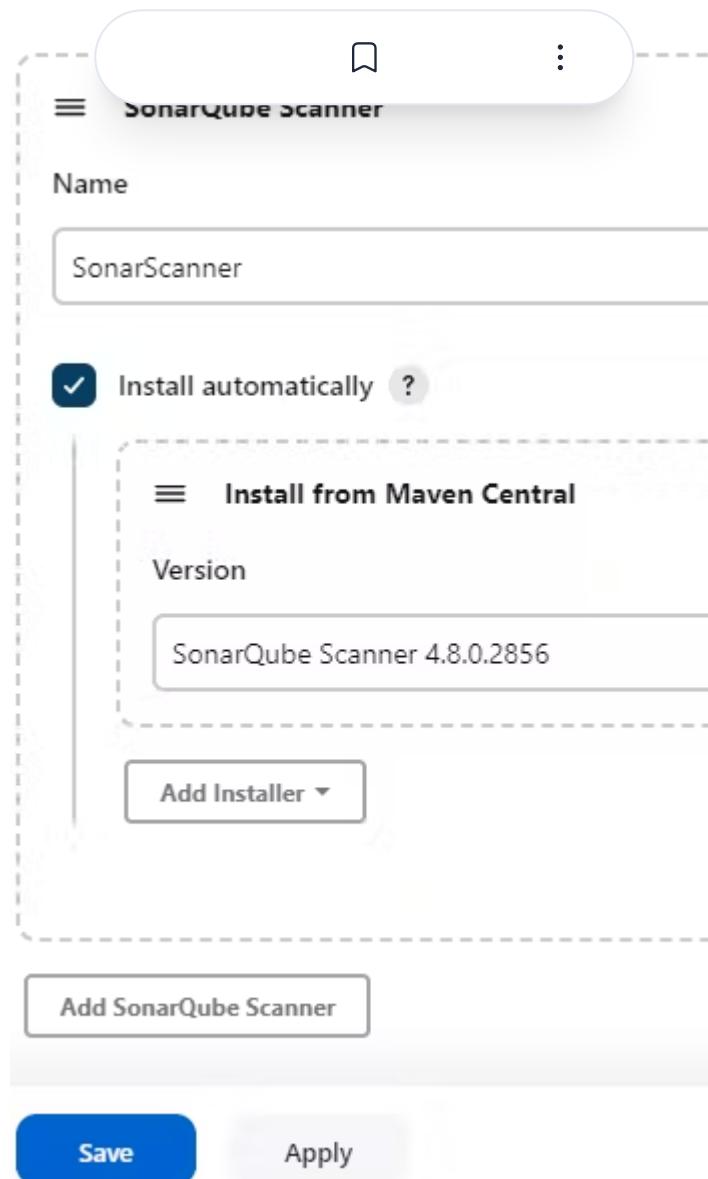
Install one more plugin

The screenshot shows the Jenkins Plugin Manager interface. On the left, there's a sidebar with links: 'Updates', 'Available plugins' (which is highlighted in grey), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area has a title 'Plugins' and a search bar containing 'ssh2'. Below the search bar, a table lists a single plugin: 'SSH2 Easy 1.4'. The plugin row includes a checked checkbox, an 'Install' button, and a description: 'This plugin allows you to ssh2 remote server to execute linux commands , shell , sftp download etc'. At the bottom, there are two buttons: 'Install without restart' (in blue) and 'Download now and install after restart' (in white). To the right of these buttons is the text 'Update information obtained:'.

Restart Jenkins after installation of plugins

Now Go to the Global Tool Configuration

- Scrolling Down
- Click on SonarQube Scanner
- Give any name and rest default



After saving it, Click to Manage Jenkins again and click on **Configure System**

Manage Jenkins



Search settings /

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

[Set up agent](#)[Set up cloud](#)[Dismiss](#)

System Configuration



Configure System

Configure global settings and paths.



Global Tool Configuration

Configure tools, their locations and automatic installers.



Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.



Manage Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Configure System

- Scrolling down
- Click on the **SonarQube server**
- Click to add
- Give any name
- Paste the Sonarqube URL
- Save

The screenshot shows the Jenkins 'Manage Jenkins' section with the 'Configure System' tab selected. Under 'SonarQube installations', there is a single entry:

- Name:** Sonar-server
- Server URL:** Default is `http://localhost:9000`, with the value `http://54.88.67.190:9000/` entered.
- Server authentication token:** A dropdown menu shows '- none -' and an 'Add' button.

At the bottom, there are 'Save' and 'Apply' buttons.

After Saving, Click Configure of the Pipeline in Jenkins

- Click to Build Environment
- Select **Execute SonarQube Scanner** in Build Steps

The screenshot shows the Jenkins 'Configure' screen for a job named 'Automated-pipeline'. The left sidebar lists configuration sections: General, Source Code Management, Build Triggers, Build Environment (which is selected and highlighted in grey), Build Steps, and Post-build Actions. The right side of the screen displays build step options under the heading 'Build Steps'. A dropdown menu titled 'Add build step ▾' is open, showing a list of available steps. The 'Execute SonarQube Scanner' step is highlighted with a blue background.

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment**
- Build Steps
- Post-build Actions

35.175.189.203:8080/job/Automated-pipeline/configure#

Build Steps

Add build step ▾

Filter

- Execute SonarQube Scanner
- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Remote Command
- Remote SFTP Download
- Remote SFTP Upload
- Remote Shell
- Run with timeout

Ignore everything just paste the key here rest default and saved it

The screenshot shows the Jenkins 'Configure' screen for an 'Automated-pipeline' job. On the left, a sidebar lists configuration sections: General, Source Code Management, Build Triggers, Build Environment, **Build Steps**, and Post-build Actions. The 'Build Steps' section is currently selected. The main panel contains several configuration fields:

- JDK**: A dropdown menu set to '(Inherit From Job)'.
Description: JDK to be used for this SonarQube analysis.
- Path to project properties**: An empty input field.
Description: Path to project properties.
- Analysis properties**: A text area containing the value `sonar.projectKey=Sample-website`.
Description: Analysis properties.
- Additional arguments**: An empty input field.
Description: Additional arguments.
- JVM Options**: An empty input field.
Description: JVM Options.

At the bottom right are two buttons: **Save** (in a blue box) and **Apply**.

Now back to Manage Jenkins, select Configure System again

The screenshot shows the Jenkins 'Manage Jenkins' interface. On the left, there's a sidebar with links: '+ New Item', 'People', 'Build History', 'Manage Jenkins' (which is selected and highlighted in grey), and 'My Views'. Below these are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). The main content area is titled 'Manage Jenkins' and contains a yellow banner with the text 'Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#). [Set up agent](#)'.

System Configuration

- Configure System**
Configure global settings and paths.
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Now we have to add token here

The screenshot shows the Jenkins 'Configure System' page under 'Manage Jenkins'. A dashed vertical line on the left separates the header from the main configuration area. The configuration fields are as follows:

- Name:** Sonar-server
- Server URL:** Default is `http://localhost:9000`. Value: `http://54.88.67.190:9000/`
- Server authentication token:** SonarQube authentication token. Mandatory when anonymous access is disabled. Value: - none -
- Jenkins Credentials Provider:** Jenkins (selected), Advanced ▾

- Select Secret text
- ID - any name

The screenshot shows the Jenkins 'Secret text' configuration dialog. It includes fields for 'Scope' (set to 'Global'), 'Secret' (containing a redacted value), 'ID' (set to 'Sonar-Token'), and 'Description'. At the bottom are 'Add' and 'Cancel' buttons.

Select Token after adding and click on save

The screenshot shows a dropdown menu for 'Server authentication token' with two options: '- none -' and 'Sonar-Token'. The 'Sonar-Token' option is highlighted with a blue background.

Now go back to Pipeline to verify whether it's working or not, it's absolutely working fine

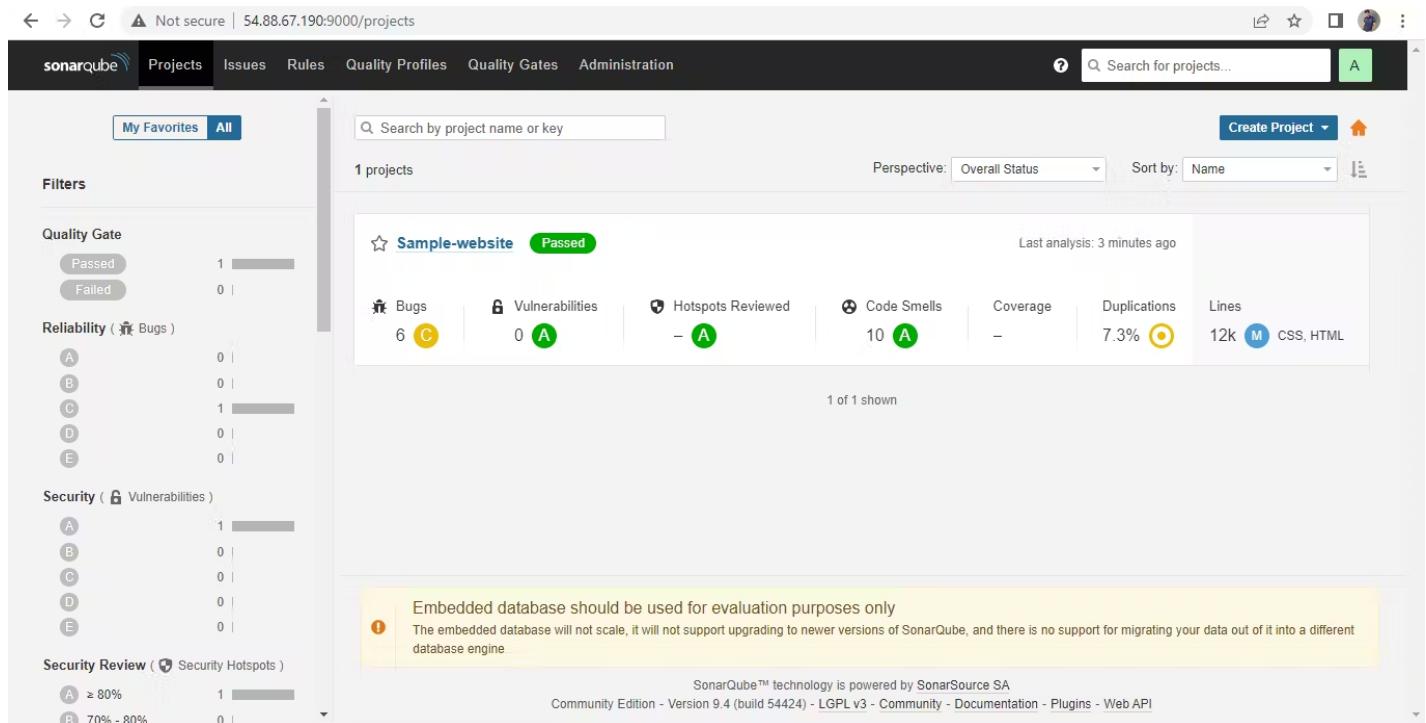


```

INFO: SENSOR VB.NET PROPERTIES [vbnet]
INFO: Sensor VB.NET Properties [vbnet] (done) | time=0ms
INFO: ----- Run sensors on project
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=1ms
INFO: SCM Publisher SCM provider for this project is: git
INFO: SCM Publisher 9 source files to be analyzed
INFO: SCM Publisher 9/9 source files have been analyzed (done) | time=188ms
INFO: CPD Executor Calculating CPD for 4 files
INFO: CPD Executor CPD calculation finished (done) | time=67ms
INFO: Analysis report generated in 83ms, dir size=824.5 kB
INFO: Analysis report compressed in 53ms, zip size=181.3 kB
INFO: Analysis report uploaded in 89ms
INFO: ANALYSIS SUCCESSFUL, you can find the results at: http://54.88.67.190:9000/dashboard?id=Sample-website
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://54.88.67.190:9000/api/ce/task?id=AYe8j\_dJJroWLDJSGCfj
INFO: Analysis total time: 8.822 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 14.420s
INFO: Final Memory: 16M/60M
INFO: -----
Finished: SUCCESS

```

Now going to check SonarQube, it's perfectly working fine



The screenshot shows the SonarQube interface for the "Sample-website" project. The project status is "Passed". Key metrics displayed include:

- Bugs: 6 (C)
- Vulnerabilities: 0 (A)
- Hotspots Reviewed: - (A)
- Code Smells: 10 (A)
- Coverage: 7.3% (Yellow)
- Duplications: -
- Lines: 12k (M)
- Technology: CSS, HTML

A note at the bottom states: "Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine."

Once our code is passed no

Docker



Installing Docker

Started Instance

Install Docker

Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

COPY

```
sudo apt-get update  
sudo apt-get install \  
  ca-certificates \  
  curl \  
  gnupg
```

Add Docker's official GPG key:

COPY

```
sudo install -m 0755 -d /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -  
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```



Use the following command to set up the repository:

COPY

```
echo \
```

```
"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker  
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Update the `apt` package index:

COPY

```
sudo apt update
```

To install the latest version, run:

COPY

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugi
```

Password-based Server Connectivity

Now I am going to make connection between Jenkins server and Docker server

After running all the commands go back to the **Jenkins server**

COPY

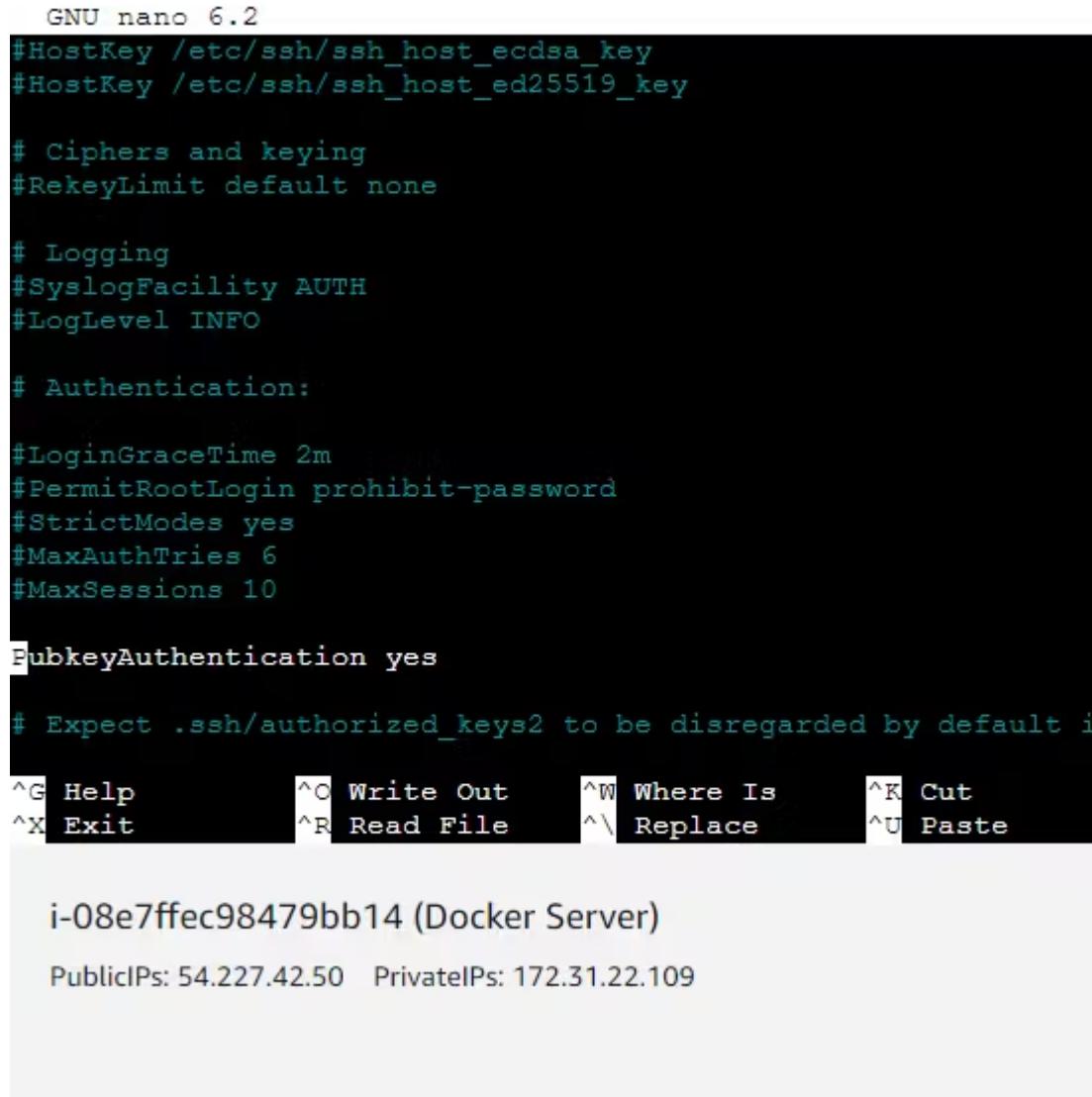
```
sudo su jenkins
```

Open Docker server

COPY

```
sudo su  
nano /etc/ssh/sshd_config
```

Uncomment this first



```
GNU nano 6.2
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

SubkeyAuthentication yes

# Expect .ssh/authorized_keys2 to be disregarded by default in
# new clients (OpenSSH > 7.2)

^G Help          ^C Write Out      ^W Where Is      ^K Cut
^X Exit          ^R Read File      ^\ Replace       ^U Paste

i-08e7ffec98479bb14 (Docker Server)
Public IPs: 54.227.42.50  Private IPs: 172.31.22.109
```

And change the Password Authentication to Yes

```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
```



COPY

```
systemctl restart sshd
```

Back to Jenkins Server

Now you can see it's asking for the password earlier it's showing permission denied

```
ubuntu@ip-172-31-20-97:~$ sudo su jenkins
jenkins@ip-172-31-20-97:/home/ubuntu$ ssh ubuntu@172.31.22.109
The authenticity of host '172.31.22.109 (172.31.22.109)' can't be established.
ED25519 key fingerprint is SHA256:McuEpDZoKnsSai3oYh4WTPnR+qJRrVOt87bOWFyjG4M.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.31.22.109' (ED25519) to the list of known hosts.
ubuntu@172.31.22.109: Permission denied (publickey).
jenkins@ip-172-31-20-97:/home/ubuntu$ ssh ubuntu@172.31.22.109
ubuntu@172.31.22.109's password: [REDACTED]
```

i-031f5900dc31f82c7 (Jenkins Server)

Public IPs: 35.175.189.203 Private IPs: 172.31.20.97

Now we have to change the password of the Docker Ubuntu user

Back to Docker Server

COPY

```
passwd ubuntu
```

```
root@ip-172-31-22-109:~# 
New password:
Retype new password:
passwd: password updated successfully
root@ip-172-31-22-109:/home/ubuntu# 
```

i-08e7ffec98479bb14 (Docker Server)

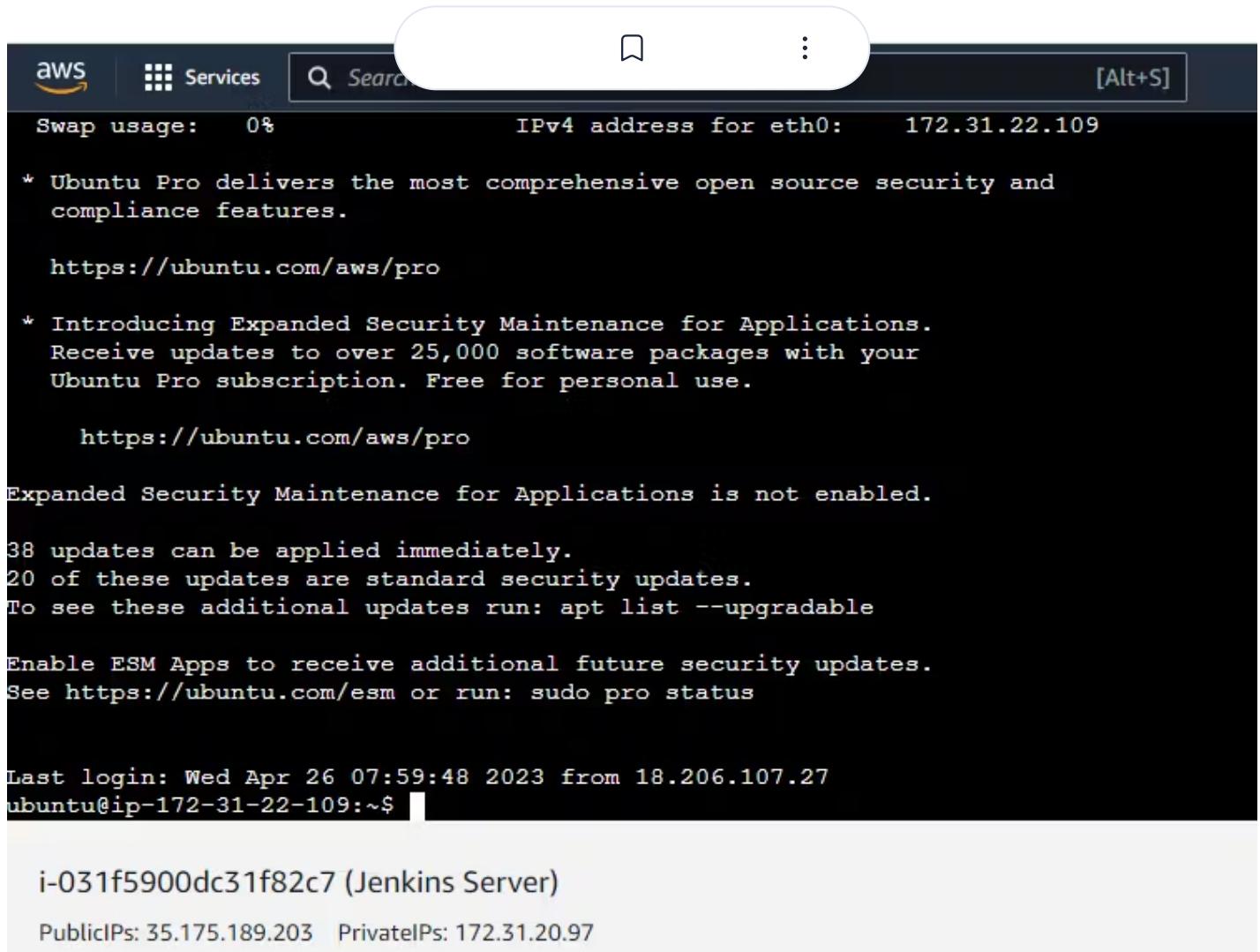
Public IPs: 54.227.42.50 Private IPs: 172.31.22.109

Back to Jenkins Server

COPY

ssh ubuntu@172.31.22.109

Access done now



AWS Services Search Swap usage: 0% IPv4 address for eth0: 172.31.22.109

* Ubuntu Pro delivers the most comprehensive open source security and compliance features.

<https://ubuntu.com/aws/pro>

* Introducing Expanded Security Maintenance for Applications. Receive updates to over 25,000 software packages with your Ubuntu Pro subscription. Free for personal use.

<https://ubuntu.com/aws/pro>

Expanded Security Maintenance for Applications is not enabled.

38 updates can be applied immediately.
20 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: sudo pro status

Last login: Wed Apr 26 07:59:48 2023 from 18.206.107.27
ubuntu@ip-172-31-22-109:~\$

i-031f5900dc31f82c7 (Jenkins Server)
Public IPs: 35.175.189.203 Private IPs: 172.31.20.97

Now I am going to generate SSH Key in Jenkins Server

```
jenkins@ip-172-31-20-97:~$ Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:j8F79JgA138khX89oi/7RWl4n8XMc2wJacseYEqH2gk jenkins@ip-172-31-20-97
The key's randomart image is:
+---[RSA 3072]---+
|          .. |
|          .. .. |
| .E.o.+.=.. |
| += =.++=*=|
| .S+. o=oOX|
|   B =..=.* |
|   o = o. o. |
|   . . . . |
|   .+.    |
+---[SHA256]---+
jenkins@ip-172-31-20-97:/home/ubuntu$
```

i-031f5900dc31f82c7 (Jenkins Server)

PublicIPs: 35.175.189.203 PrivateIPs: 172.31.20.97

After generating keygen I entered the command

COPY

ssh-copy-id ubuntu@172.31.22.109

After running the command enter the password

```
jenkins@ip-172-31-20-97:/home/ubuntu$ ssh-copy-id ubuntu@172.31.22.109
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/jenkins/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
ubuntu@172.31.22.109's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'ubuntu@172.31.22.109'"
and check to make sure that only the key(s) you wanted were added.

jenkins@ip-172-31-20-97:/home/ubuntu$ ssh ubuntu@172.31.22.109
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64)
```

i-031f5900dc31f82c7 (Jenkins Server)

PublicIPs: 35.175.189.203 PrivateIPs: 172.31.20.97



COPY

```
ssh ubuntu@172.31.22.109
```

Now we don't need to password again anymore

Back to **Jenkins** again

- Manage Jenkins
- Configure system
- Server group center

The screenshot shows the Jenkins Server Groups Center configuration page. At the top, there is a breadcrumb navigation: Dashboard > Manage Jenkins > Configure System. Below the title "Server Groups Center", it says "Server Group List : Create the server groups for your projects". There is a dashed-line box containing four input fields: "Group Name" (Docker-server), "SSH Port" (22), "User Name" (ubuntu), and "Password" (represented by four dots). At the bottom of the box are two buttons: "Save" (in a blue rounded rectangle) and "Apply".

Group Name ?
Docker-server

SSH Port ?
22

User Name ?
ubuntu

Password ?
.....

Save Apply

Now we have to add a server list

The screenshot shows the Jenkins 'Configure System' page. At the top, there are navigation links: 'Dashboard' > 'Manage Jenkins' > 'Configure System'. A blue 'Add' button is located in the top-left corner of the main content area. Below it, the text 'Server List :' is followed by the instruction 'add the server under this server group for your projects'. A dashed-line box contains the 'Server Group:' field, which has 'Docker-server' selected. Inside this box are three input fields: 'Server Name' with 'Docker-1', 'Server IP' with '54.227.42.50', and two empty 'Port' fields. At the bottom of the form are 'Save' and 'Apply' buttons.

Now go to **Pipeline**

- Configure
- Post-build action
- Add build step

The screenshot shows the Jenkins pipeline configuration interface. The left sidebar lists sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The Post-build Actions section is currently selected and highlighted with a light gray background. On the right, there are configuration panels for JVM Options, Remote Shell, Target Server, and shell. The shell panel contains the command `touch test.txt`. At the bottom, there are Save and Apply buttons.

JVM Options

Remote Shell

Disable

Target Server

Docker-server~~Docker-1~~54.227.42.50

shell

touch test.txt

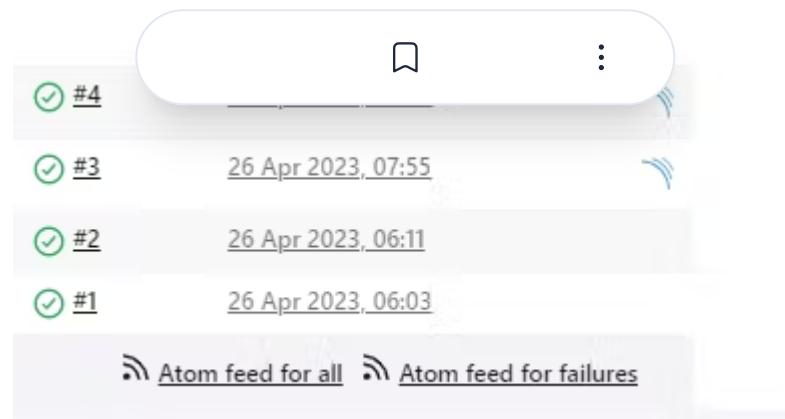
Add build step ▾

Save

Apply

Now I am gonna build the pipeline to verify whether it's working or not

And it seems to be working



Open Docker Server, as we can see our file got created here

```
root@ip-172-31-22-109:/home/ubuntu# ls
test.txt
root@ip-172-31-22-109:/home/ubuntu#
```

i-08e7ffec98479bb14 (Docker Server)

Public IPs: 54.227.42.50 Private IPs: 172.31.22.109

Open the Git Repository now

- Create a Dockerfile
- Commit the file

The screenshot shows a GitHub repository named 'BroDevOps / website-sample'. The 'Code' tab is selected. Below the repository name, there's a search bar with 'Dockerfile' and a dropdown 'in master'. A modal window is open over the code editor, showing two tabs: 'Edit new file' and 'Preview'. The 'Preview' tab is active, displaying the following Dockerfile content:

```
1 FROM nginx
2 COPY . /usr/share/nginx/html
```

See Auto trigger started

The screenshot shows the Jenkins Build History page. It displays five builds, each with a green checkmark icon and a build number. The builds are listed from bottom to top: #1 (26 Apr 2023, 06:03), #2 (26 Apr 2023, 06:11), #3 (26 Apr 2023, 07:55), #4 (26 Apr 2023, 08:46), and #5 (pending—In the quiet period. Expires in 4 sec). There are also links for 'Atom feed for all' and 'Atom feed for failures'.

Now returned to Pipeline, configure

- I have deleted the Remote shell
- Clicked to execute shell

Created a folder in the Docker server

```
ubuntu@ip-172-31-22-109:~$ mkdir website
ubuntu@ip-172-31-22-109:~$ ls
test.txt  website
ubuntu@ip-172-31-22-109:~$ cd website
ubuntu@ip-172-31-22-109:~/website$ pwd
/home/ubuntu/website
ubuntu@ip-172-31-22-109:~/website$
```

i-08e7ffec98479bb14 (Docker Server)

Public IPs: 54.227.42.50 Private IPs: 172.31.22.109

Clicked to execute shell and fill the details, here I entered docker server IP and folder which is website

The screenshot shows the Jenkins Pipeline configuration interface. The top navigation bar includes 'Dashboard', 'Automated-pipeline', and 'Configuration'. The left sidebar lists 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build Steps', and 'Post-build Actions', with 'Post-build Actions' currently selected. The main panel has a title 'Execute shell' with a question mark icon. It contains a 'Command' field with the value 'scp -r ./* ubuntu@172.31.22.109:~/website'. Below this is an 'Advanced' dropdown and a 'Save' button. At the bottom, there's a 'Post-build Actions' section with a 'Save' and 'Apply' button.

We have got the success message here and let us check our docker server

The screenshot shows the Jenkins 'Build history' page. At the top, there's a search bar labeled 'Filter builds...' and a dropdown menu set to 'trend'. Below the search bar is a table listing eight build entries:

Build #	Date
#8	26 Apr 2023, 09:13
#7	26 Apr 2023, 09:11
#6	26 Apr 2023, 09:08
#5	26 Apr 2023, 08:58
#4	26 Apr 2023, 08:46
#3	26 Apr 2023, 07:55
#2	26 Apr 2023, 06:11
#1	26 Apr 2023, 06:03

Each entry has a green checkmark icon next to it, except for builds #6 and #7 which have a red X icon. At the bottom of the table, there are two links: 'Atom feed for all' and 'Atom feed for failures'.

All the files copied to the Docker server

```
ubuntu@ip-172-31-22-109:~/website$ ls
Dockerfile  about.html  contact.html  css  images  index.html  js  service.html  test.txt
ubuntu@ip-172-31-22-109:~/website$
```

i-08e7ffec98479bb14 (Docker Server)

PublicIPs: 54.227.42.50 PrivateIPs: 172.31.22.109

Building the Image and running the container

Back to the Docker server and need to give permission so that we can run all the commands without sudo

COPY

```
sudo usermod -aG docker ubuntu
newgrp docker
```

After giving the permission

Run docker without sudo



COPY

docker ps

```
ubuntu@ip-172-31-22-109:~/website$ sudo usermod -aG docker ubuntu
ubuntu@ip-172-31-22-109:~/website$ newgrp docker
ubuntu@ip-172-31-22-109:~/website$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS          NAMES
ubuntu@ip-172-31-22-109:~/website$
```

i-08e7ffec98479bb14 (Docker Server)

Public IPs: 54.227.42.50 Private IPs: 172.31.22.109

Now back to Jenkins

- Click to Pipeline
- Click to configure
- Click to post-build actions
- Select the Remote shell again in the build steps
- I gave any random name

The screenshot shows the Jenkins Pipeline configuration interface. On the left, a sidebar lists navigation items: Dashboard, Automated-pipeline, Configuration, General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The 'Post-build Actions' item is currently selected and highlighted with a grey background. The main panel is titled 'Remote Shell' and contains a 'Target Server' section with the value 'Docker-server~ ~ Docker-1~ ~ 54.227.42.50'. Below it is a 'shell' section containing the command:

```
cd /home/ubuntu/website  
docker build -t mywebsite .  
docker run -d -p 8085:80 --name=website-sample mywebsite
```

At the bottom of the main panel is a button labeled 'Add build step ▾'. The 'Post-build Actions' section below has a 'Save' button and an 'Apply' button.

Time to check our docker container got created or not

It got created but we have to add port 8085 in the inbound setting of the security group

A terminal window displays the following Docker container information:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
681f37f29afe	mywebsite	"/docker-entrypoint..."	About a minute ago	Up About a minute	0.0.0.0:8085->80/tcp, :::8085->80/tcp	website-sample

Ubuntu@ip-172-31-22-109:~/website\$

i-08e7ffec98479bb14 (Docker Server)

PublicIPs: 54.227.42.50 PrivateIPs: 172.31.22.109

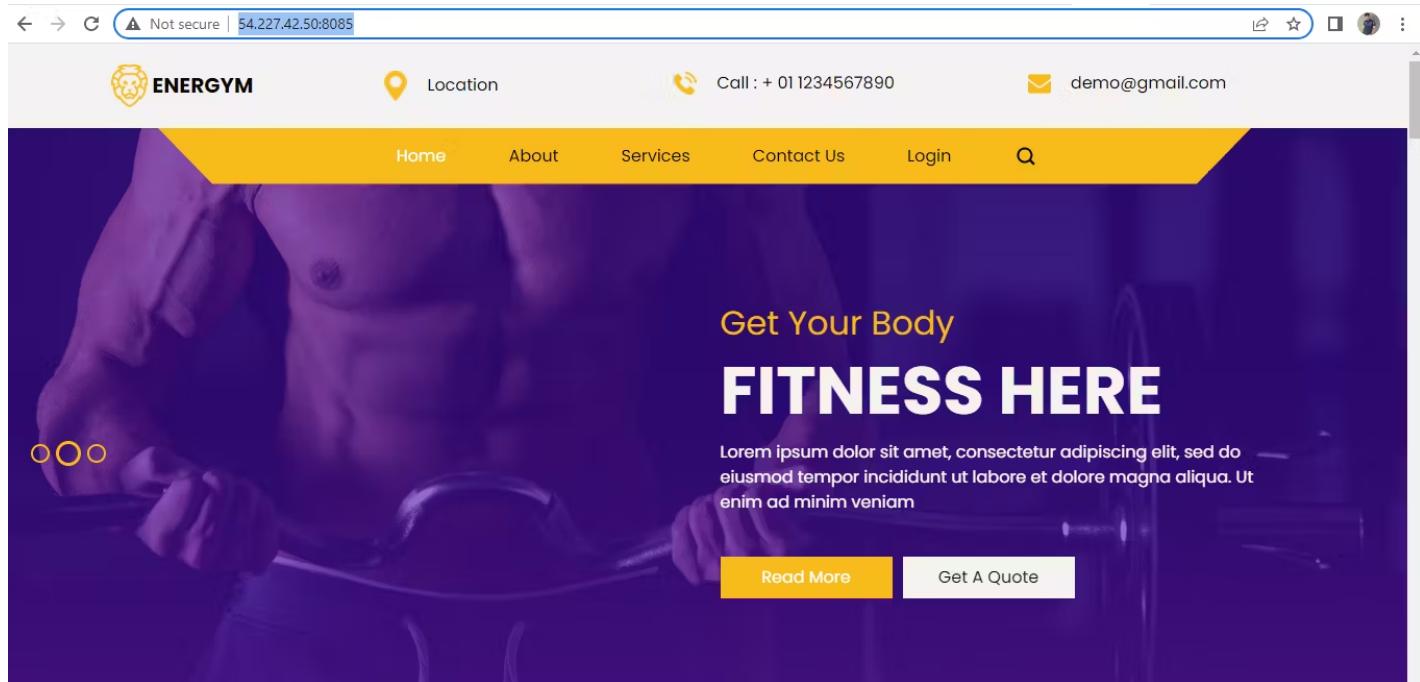
Inbound rules [Info](#)

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-0af5e974813efdcbb	SSH	TCP	22	Custom ▾ <input type="text" value="0.0.0.0"/> X	<input type="text"/>	Delete
sgr-0492599f316be4209	All traffic	All	All	Custom ▾ <input type="text" value="0.0.0.0"/> X	<input type="text"/>	Delete
-	Custom TCP	TCP	8085	Anywh... ▾ <input type="text" value="0.0.0.0"/> X	<input type="text"/>	Delete

[Add rule](#)

Now time to check the public URL of the docker Instance

<http://54.227.42.50:8085/> (our code is successfully deployed on docker container)



ter



Read articles from **DevOps by Sagar Aulakh** directly inside your inbox. Subscribe to the newsletter, and don't miss out.

parandhamareddy026@gmail.com **SUBSCRIBE**

[Jenkins](#)[Docker](#)[Ubuntu](#)[ci-cd](#)[Linux](#)

MORE ARTICLES

Sagar Aulakh

Installing Java and MySQL DB using Ansible Playbook

Launch EC2 Instance Create 1 Controller Instance Create 2 Slave Instances We need to install Ansible...

Sagar Aulakh



How to push a local project to GitHub repository

Step: 1 Open the file manager and the path of the code Below I have created 3 sample files under Fol...

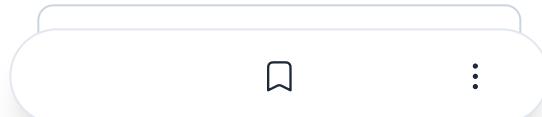
Sagar Aulakh

Jenkins CI with GitHub Integration

Launch Instance Create AWS EC2 - t2.micro sudo apt update clear Install JAVA because Jenkins is bui...

©2023 DevOps by Sagar Aulakh

[Archive](#) • [Privacy policy](#) • [Terms](#)



Powered by [Hashnode](#) - Home for tech writers and readers