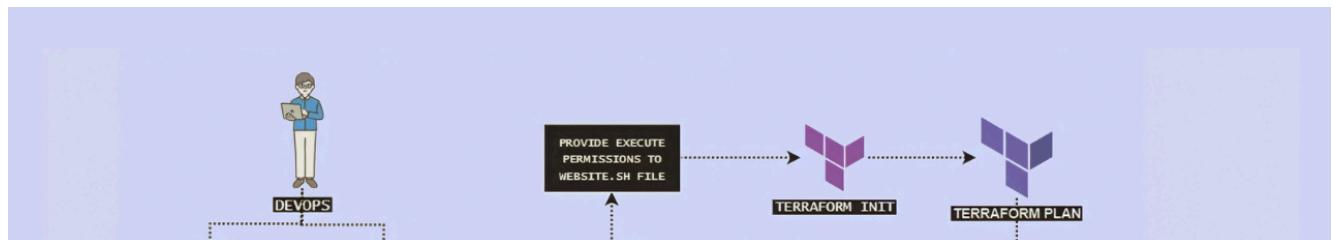


DevOps

# AWS Resources with Terraform, Jenkins ci-cd, and Hosting a static website in s3



mrclobook.com · 8 January 2024



Mr Cloud Book

Home Blog DevSecOps Contact About Me Testimonials

Search Blogs



## Contents [ hide ]

### STATIC WEBSITE USING S3 FROM TERRAFORM

Prerequisites:

Launch an Ubuntu(22.04) T2 Large Instance

Install Jenkins, Docker and Trivy

To Install Jenkins

Install Docker

2C – Install Trivy

Install Plugins like JDK, Sonarqube Scanner,Terraform

Install Plugin

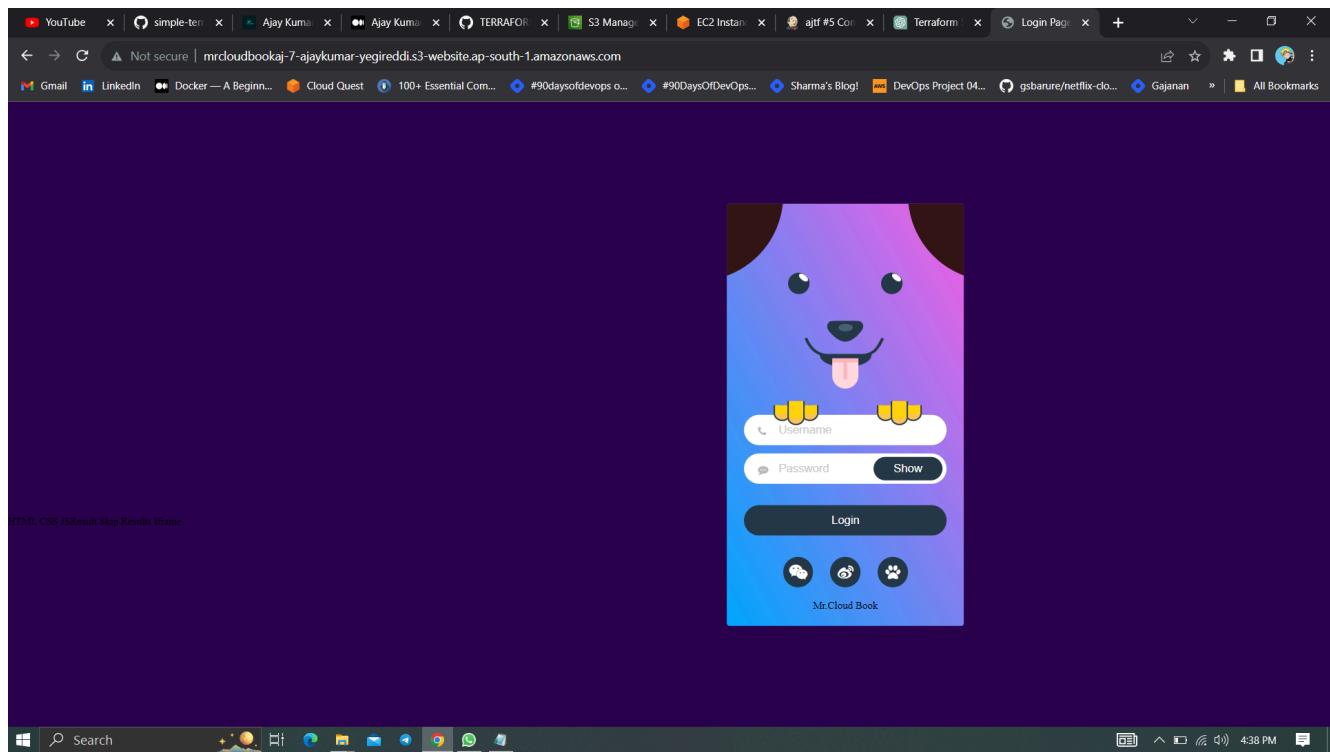
Configure Java and Terraform in Global Tool Configuration

Configure Sonar Server in Manage Jenkins

Create an IAM, S3 bucket and Dynamo DB table.

Docker Plugin setup

## STATIC WEBSITE USING S3 FROM TERRAFORM



In today's fast-paced world of cloud computing, the ability to rapidly and efficiently provision infrastructure is a game-changer. This is where Infrastructure as Code (IaC) comes into play, allowing us to define and manage our infrastructure in a code-based manner. In this blog post, we will explore how to harness the power of IaC by using two essential tools: Terraform and Jenkins, in conjunction with Amazon Web Services (AWS).

Terraform is an open-source IaC tool that enables us to define, create, and manage our infrastructure using declarative configuration files. Jenkins, on the other hand, is a widely adopted automation server that helps streamline the software development and deployment process.

Our journey will encompass several key objectives:

- 1. Setting up Terraform and Jenkins:** We'll start by ensuring you have all the prerequisites in place, including an AWS account, Terraform, Jenkins, and Docker. We'll walk you through the installation and configuration of these essential tools.
- 2. Creating the Terraform Scripts:** We'll delve into the heart of IaC by crafting Terraform scripts to provision AWS resources. Along the way, we'll introduce the concept of user data, a powerful feature that allows us to automate tasks like launching containers within our instances.
- 3. Running Two Application Containers with User Data:** To demonstrate the practical application of user data, we'll guide you through launching not just one but two application containers within your AWS instances. This step showcases the versatility and automation capabilities of IaC.
- 4. DevOps project we will be using Terraform and AWS Cloud to set up static website hosting fully automated in seconds. This Terraform project will help beginners understand the concept and working of Terraform with AWS and how you can create a one-click automated solution using Terraform in DevOps**
- 5. Setting up Infrastructure State Management:**
  - **S3 Bucket for Terraform State:** We'll create an AWS S3 bucket dedicated to securely storing your Terraform state files. This is essential for maintaining the state of your infrastructure in a central location.
  - **DynamoDB Table for Locking:** In addition to the S3 bucket, we'll set up an AWS DynamoDB table to enable locking capabilities. This ensures that your infrastructure remains in a consistent state when multiple users are working concurrently.
- 6. Integrating Jenkins and Terraform:** To tie it all together, we'll demonstrate how to integrate Jenkins with Terraform. This integration will empower you to automate the provisioning process, enhance the efficiency of your infrastructure management, and ensure that your Terraform state is securely stored and locked when needed.

## Prerequisites:

Before you embark on the journey of provisioning AWS resources using Terraform and Jenkins, it's crucial to ensure that you have all the necessary components and configurations in place. Here are the prerequisites you should have before starting this tutorial:

**1. AWS Account:** You must have an active AWS account with administrative privileges or the necessary permissions to create and manage AWS resources.

**2. S3 Bucket for Terraform State:**

- **Purpose:** To securely store your Terraform state files remotely.
- **Steps:**
  - Log in to your AWS Management Console.
  - Navigate to the S3 service.
  - Create an S3 bucket with a unique name in the desired AWS region.
  - Note down the bucket name as you'll use it in your Terraform scripts.

**3. DynamoDB Table for Locking Capability:**

- **Purpose:** To enable locking for Terraform state management.
- **Steps:**
  - Access the AWS Management Console.
  - Go to the DynamoDB service.
  - Create a DynamoDB table with a unique name and primary key.
  - Configure the table's read and write capacity settings as needed.
  - Note down the table name for reference.

**4. Jenkins Setup:**

- Ensure that Jenkins is up and running in your environment.
- Configure Jenkins with the necessary plugins for AWS and Terraform integration.

## 5. Terraform Installation in Jenkins:

- Terraform should be installed on the Jenkins server to execute Terraform scripts as part of your CI/CD pipeline.

## 6. Terraform Files in Source Code Management (SCM):

- Your Terraform configuration files should already be available in your Source Code Management system (e.g., Git). Make sure you have the necessary access rights to the repository.

## 7. IAM Role for Jenkins EC2 Instance:

- **Purpose:** To grant the Jenkins EC2 instance the necessary permissions to interact with AWS resources.
- **Steps:**
  - Create an IAM role in AWS.
  - Attach the appropriate policy that grants permissions for AWS resource provisioning, DynamoDB access, S3 bucket operations, and any other required permissions.
  - Associate the IAM role with the Jenkins EC2 instance.

## 8. GitHub Repository (Optional):

- If you're using a public repository as an example, you can fork the repository and start making changes in your own forked repository. Ensure that you have the necessary access to the repository.

With these prerequisites in place, you'll be well-prepared to dive into the tutorial and learn how to leverage Terraform, Jenkins, AWS S3, and DynamoDB to automate the provisioning and state management of your AWS resources. These foundational components are key to a successful IaC implementation and CI/CD pipeline for infrastructure.

## Launch an Ubuntu(22.04) T2 Large Instance

Launch an AWS T2 Large Instance. Use the image as Ubuntu. You can create a new key pair or use an existing one. Enable HTTP and HTTPS settings in the Security Group and open all ports (not best case to open all ports but just for learning purposes it's okay).

Instances (1) <a href="#">Info</a>		<a href="#">Connect</a>	Instance state	Actions	Launch instances			
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 D
<input type="checkbox"/>	CI-CD	i-065c10200537a1eee	<span>Running</span>	t2.large	<span>2/2 checks passed</span>	No alarms	ap-south-1a	ec2-52-66-14

## Install Jenkins, Docker and Trivy

### To Install Jenkins

Connect to your console, and enter these commands to Install Jenkins

```
vi jenkins.sh
```



```
#!/bin/bash
sudo apt update -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/api/debian stable main" | sudo tee /etc/apt/sources.list.d/adoptium.list &| sudo apt update -y
sudo apt install temurin-17-jdk -y
/usr/bin/java --version
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 -y
sudo apt-get install jenkins -y
sudo systemctl start jenkins
sudo systemctl status jenkins
```



```
sudo chmod 777 jenkins.sh  
./jenkins.sh # this will install jenkins
```

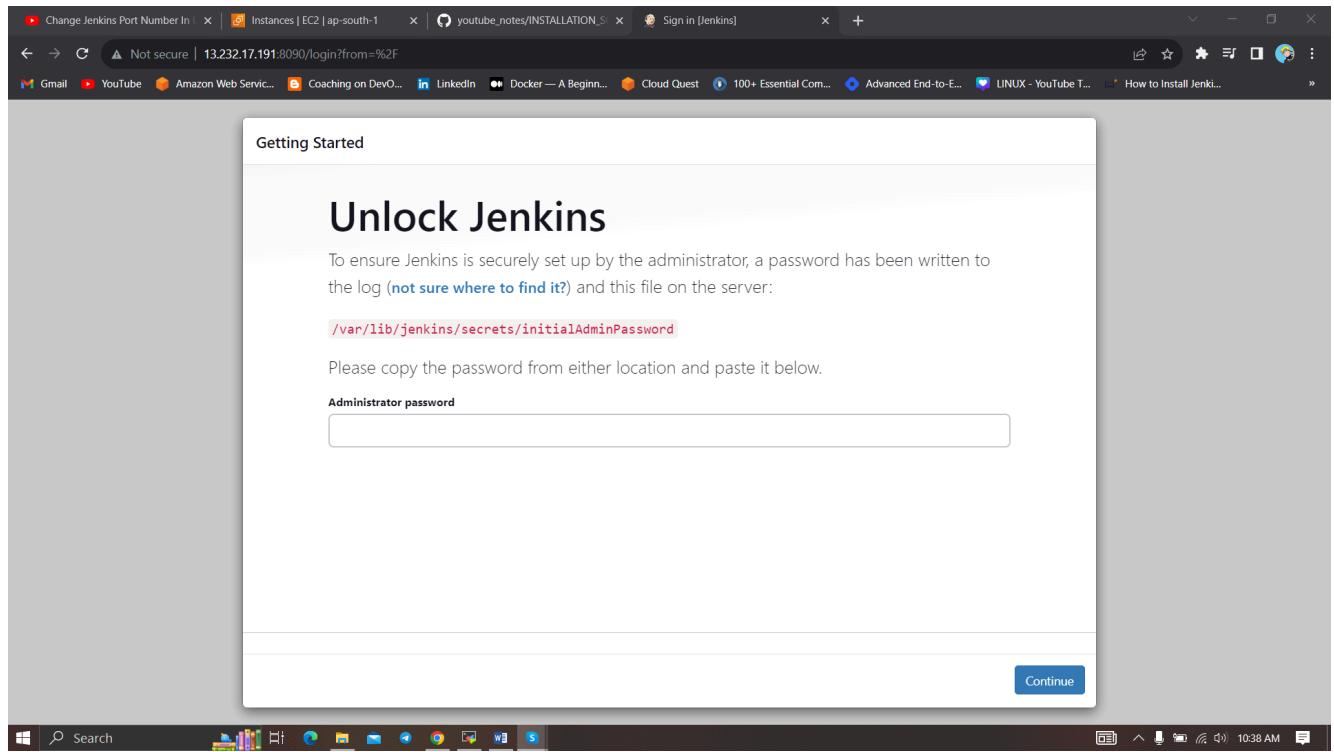


Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080.

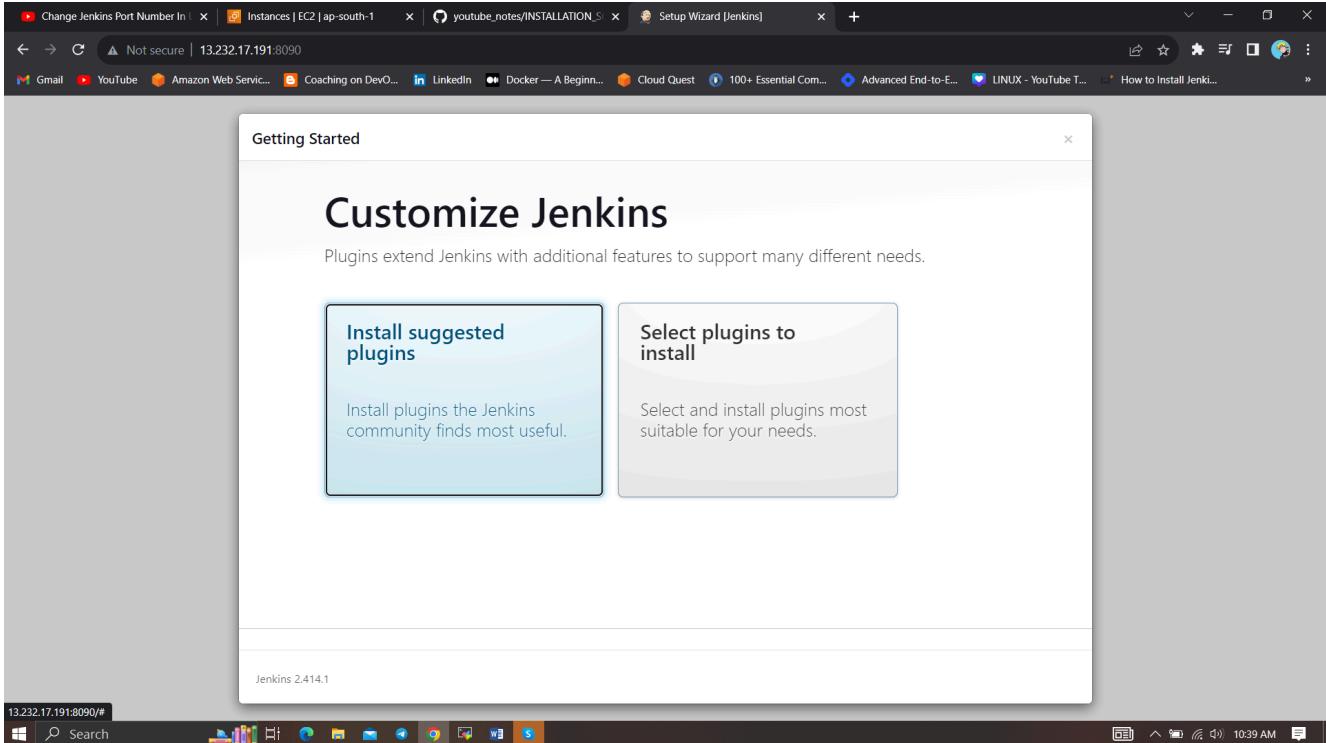
Now, grab your Public IP Address



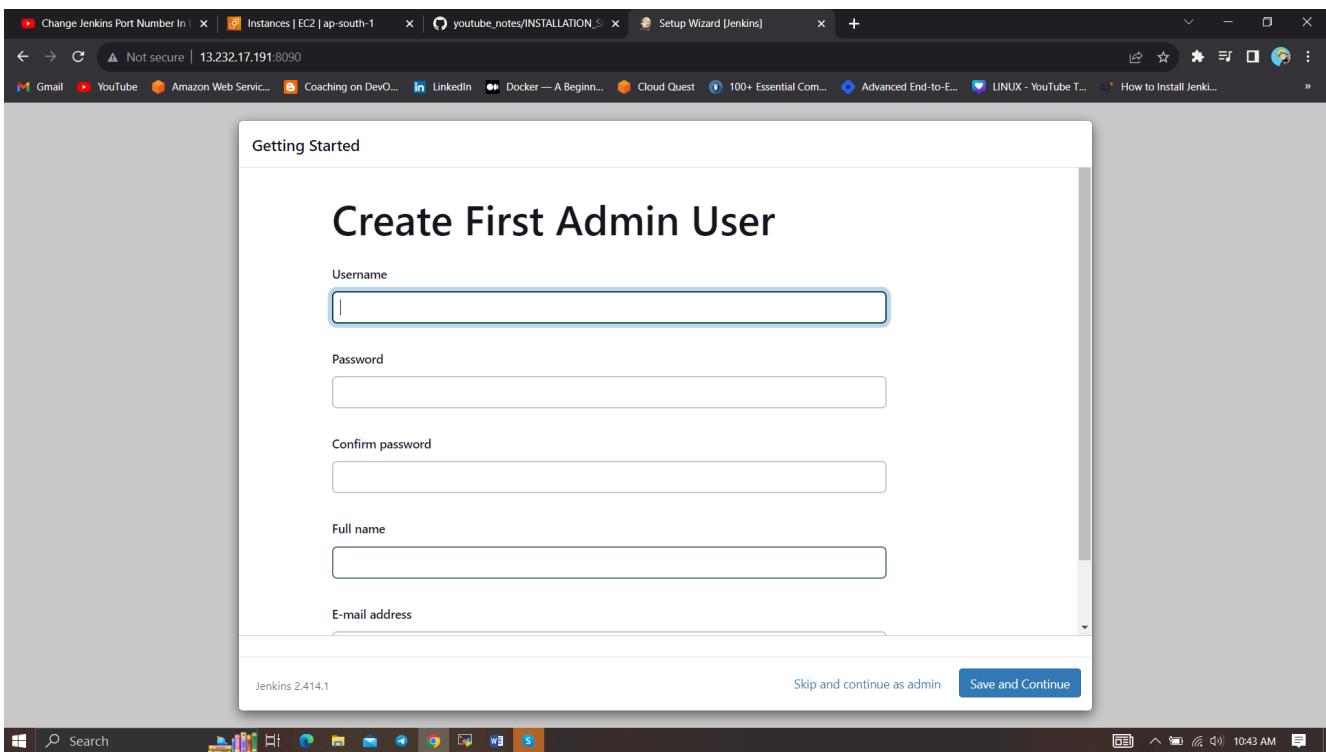
```
<EC2 Public IP Address:8080>  
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



Unlock Jenkins using an administrative password and install the suggested plugins.



Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

Jenkins Getting Started Screen.

Not secure | 13.232.17.191:8090

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

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle

2 Idle

**Set up a distributed build**

Create a job →

Set up an agent →

Configure a cloud →

Learn more about distributed builds ↗

## Install Docker

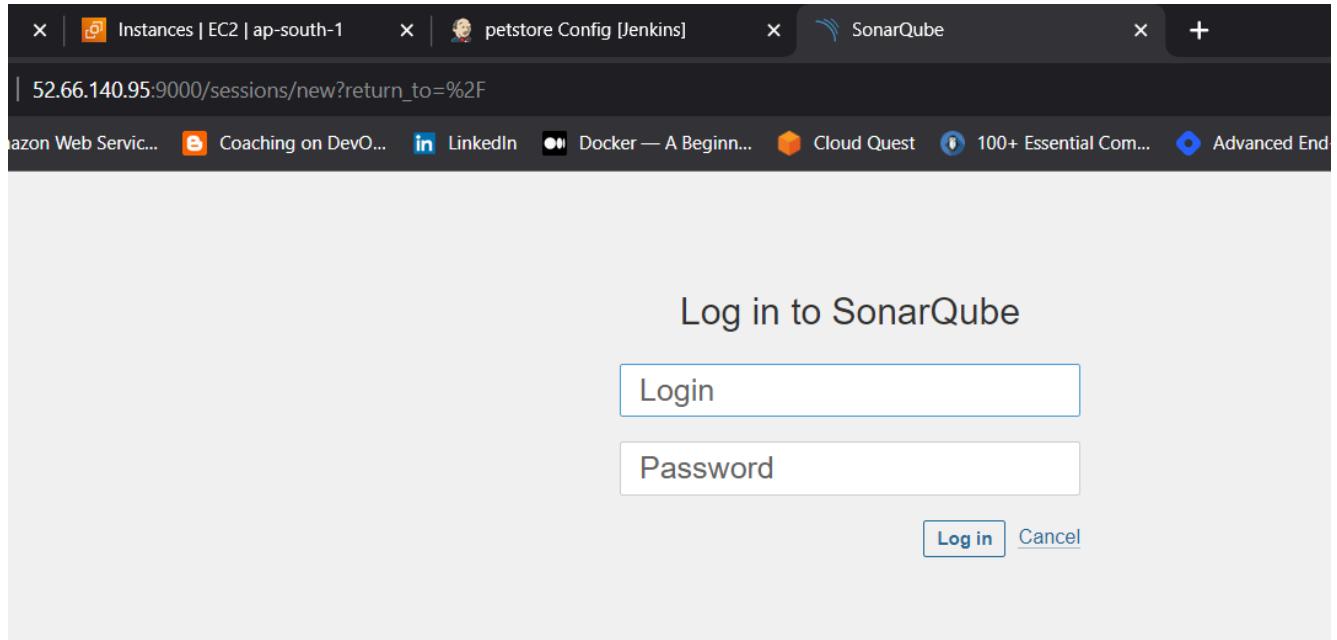
```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER      #my case is ubuntu
newgrp docker
sudo chmod 777 /var/run/docker.sock
```

After the docker installation, we create a sonarqube container (Remember to add 9000 ports in the security group).

```
docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
```

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonarqube -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882fb8eb: Pull complete
2cabec57ffa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b66c96bf9ad3d62289436af7f752fdb04993092d0ca5065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
4b66c96bf9ad sonarqube:lts-community "/opt/sonarqube/dock..." 9 seconds ago Up 5 seconds 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp sonar
ubuntu@ip-172-31-42-253:~$
```

Now our sonarqube is up and running



Enter username and password, click on login and change password



```
username admin
password admin
```



6.140.95.9000/account/reset\_password

Web Servic... Coaching on DevO... LinkedIn Docker — A Beginn... Cloud Quest 100+ Essential Com... Advanced End-to-E...

## 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**

Update New password, This is Sonar Dashboard.

← → C Not secure | 52.66.140.95:9000/projects/create

Gmail YouTube Amazon Web Service... Coaching on DevO... LinkedIn Docker — A Beginn... Cloud Quest 100+ Essential Com... Advanced End-to-E... LINUX - YouTube T... How to Install Jenk...

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration Search for projects... A

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 Server Set up global configuration	From Bitbucket Cloud Set up global configuration	From GitHub Set up global configuration	From GitLab Set up global configuration
--	--	---	--	--

## 2C – Install Trivy



```
vi trivy.sh
```



```
sudo apt-get install wget apt-transport-https gnupg lsb-release -y
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor > /usr/share/keyrings/trivy.gpg
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity"
sudo apt-get update
sudo apt-get install trivy -y
```



Next, we will log in to Jenkins and start to configure our Pipeline in Jenkins

## Install Plugins like JDK, Sonarqube Scanner,Terraform

### Install Plugin

Goto Manage Jenkins → Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → SonarQube Scanner (Install without restart)

3 → Terraform

The screenshot shows the Jenkins Plugins page. On the left sidebar, there are links for Updates, Available plugins (which is selected), Installed plugins, Advanced settings, and Download progress. A search bar at the top right says "Search available plugins". Below the search bar, there are two listed plugins:

- Eclipse Temurin installer 1.5**: Provides an installer for the JDK tool that downloads the JDK from <https://adoptium.net>. It is marked as "up for adoption" and was released 11 months ago.
- SonarQube Scanner 2.15**: External Site/Tool Integrations, Build Reports. This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality. It was released 9 months and 19 days ago.

The screenshot shows the Jenkins Plugins page. The search bar at the top right has "Terra" typed into it. Below the search bar, there is one listed plugin:

- Terraform 1.0.10**: Build Wrappers. This plugin provides a build wrapper for Terraform to launch and destroy infrastructure. It was released 3 years and 7 months ago.

let's install Terraform on our Jenkins machine

```
ubuntu@ip-172-31-42-229:~$ wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg --dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list
sudo apt update && sudo apt install terraform
```

The terminal window shows the command being run to add the HashiCorp GPG key and update the package list. The output indicates that the file '/usr/share/keyrings/hashicorp-archive-keyring.gpg' already exists, so it asks if we want to overwrite it. The user responds with 'yes'. The terminal then shows the package manager updating the list and installing the 'terraform' package. The progress bar shows the download of the package files.

check terraform version

```
terraform --version
```



let's find the path to our terraform (we will use it in the tools section of Terraform)

```
which terraform
```



The screenshot shows a terminal window with the following content:

```
Quick connect... 1. ubuntu@ip-172-31-42-229: ~
star  ubuntu@ip-172-31-42-229:~$ which terraform
/usr/bin/terraform
ubuntu@ip-172-31-42-229:~$
```

## Configure Java and Terraform in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) → Click on Apply and Save

The screenshot shows the Jenkins Global Tool Configuration page under the 'Tools' section. It displays a form for adding a new JDK configuration:

- Name:** jdk17
- Install automatically:**
- Install from adoptium.net:**
- Version:** jdk-17.0.8.1+1
- Add Installer:** A button to add more installers.

Tools -> Terraform

## Terraform installations

Add Terraform

**Terraform**

Name

Install directory

Install automatically ?

Add Terraform

**Save** **Apply**

Apply and save.

## Configure Sonar Server in Manage Jenkins

Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Administration

Configuration ▾ Security ▾ Projects ▾ System Marketplace

General Users Groups Global Permissions Permission Templates

Find i

click on update Token

SCM Accounts Last connection Groups Tokens

A Administrator admin < 1 hour ago sonar-administrators sonar-users 0 0 Update Tokens

Create a token with a name and generate

#### Tokens of Administrator

##### Generate Tokens

Name	Expires in
<input type="text" value="Enter Token Name"/>	30 days
<input type="button" value="Generate"/>	

! New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!



squ\_21d162904c1c72cf8b39665f96480185c99dc2f9

Name	Type	Project	Last use	Created	Expiration	
Jenkins	User		Never	September 8, 2023	October 8, 2023	<input type="button" value="Revoke"/>

copy Token

Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this

#### New credentials

##### Kind

##### Scope ?

##### Secret

**POST THE TOKEN HERE**

##### ID ?

**Sonar-token**

##### Description ?

**Sonar-token**

You will this page once you click on create

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description	
Sonar-token	sonar	Secret text	sonar	

Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

**SonarQube servers**

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

**Environment variables** Enable injection of SonarQube server configuration as build environment variables

**SonarQube installations**

List of SonarQube installations

Name	<input type="text" value="sonar-server"/>	<span style="color: red;">X</span>
Server URL	Default is http://localhost:9000 <input type="text" value="http://13.232.17.191:9000"/>	
Server authentication token	SonarQube authentication token. Mandatory when anonymous access is disabled. <input type="text" value="Sonar-token"/>	
<input style="margin-right: 10px;" type="button" value="Add"/> <input style="background-color: #0072bc; color: white; border-radius: 5px; padding: 5px; margin-right: 10px;" type="button" value="Save"/> <input type="button" value="Apply"/>		

Click on Apply and Save

**The Configure System option** is used in Jenkins to configure different server

**Global Tool Configuration** is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

**SonarQube Scanner installations****SonarQube Scanner****Name**

**Install automatically** ?

**Install from Maven Central****Version**

In the Sonarqube Dashboard add a quality gate also

## Administration-> Configuration->Webhooks

The screenshot shows the SonarQube administration interface. The top navigation bar has tabs for 'Administration' (which is highlighted with a red box), 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. Below the navigation is a secondary menu with 'General Settings', 'Encryption', and 'Webhooks' (also highlighted with a red box). A search bar 'Search by login or name...' is present. The main content area displays user information in a table with columns for 'SCM Accounts', 'Last connection', 'Groups', and 'Tokens'. One user, 'Administrator admin', is listed with a green profile icon. At the bottom right of the table is a 'Create User' button.

Click on Create

The screenshot shows the 'Webhooks' configuration page. The top navigation bar has tabs for 'Administration' (highlighted with a blue box), 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. Below the navigation is a secondary menu with 'Webhooks' (highlighted with a red box). A large 'Create' button is prominently displayed on the right side of the page. The main content area contains a brief description of what webhooks are used for and a note about documentation, followed by a message stating 'No webhook defined.'

Add details

```
#in url section of quality gate
<http://jenkins-public-ip:8080>/sonarqube-webhook/
```

**Create Webhook**

All fields marked with \* are required

**Name \***  
jenkins

**URL \***  
http://43.204.36.242:8090/sonarqube-webhook/

Server endpoint that will receive the webhook payload, for example:  
"http://my\_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example:  
"https://myLogin:myPassword@my\_server/foo"

**Secret**  
[Empty]

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header.

**Create** **Cancel**

**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. **Learn More** **Dismiss**

## Create an IAM, S3 bucket and Dynamo DB table.

Navigate to AWS CONSOLE

Click the “Search” field.

**Instances (1) Info**

Find instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status
JENKINS	i-07885f1ce7fb7174	Running	t2.large	2/2 checks passed	No alarms

Type “IAM enter”

Click “Roles”

Click “Create role”

IAM > Roles

**Roles (3) Info**

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
EC2	AWS Service: ec2	1 hour ago

**Roles Anywhere Info**

Authenticate your non AWS workloads and securely provide access to AWS services.

**Create role**

Click “AWS service”

Click “Choose a service or use case”

Select trusted entity Info

Step 2 Add permissions

Step 3 Name, review, and create

**Trusted entity type**

- AWS service**  
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account**  
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity**  
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation**  
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy**  
Create a custom trust policy to enable others to perform actions in this account.

**Use case**

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**

Choose a service or use case

Click “EC2”

Click “Next”

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

**Service or use case**

**EC2**

Choose a use case for the specified service.

**Use case**

- EC2**  
Allows EC2 instances to call AWS services on your behalf.
- EC2 Role for AWS Systems Manager**  
Allows EC2 instances to call AWS services like CloudWatch and Systems Manager on your behalf.
- EC2 Spot Fleet Role**  
Allows EC2 Spot Fleet to request and terminate Spot instances on your behalf.
- EC2 - Spot Fleet Auto Scaling**  
Allows Auto Scaling to access and update EC2 spot fleets on your behalf.
- EC2 - Spot Fleet Tagging**  
Allows EC2 to launch spot instances and attach tags to the launched instances on your behalf.
- EC2 - Spot Instances**  
Allows EC2 Spot Instances to launch and manage spot instances on your behalf.
- EC2 - Spot Fleet**  
Allows EC2 Spot Fleet to launch and manage spot fleet instances on your behalf.
- EC2 - Scheduled Instances**  
Allows EC2 Scheduled Instances to manage instances on your behalf.

**Cancel** **Next**

Click the “Search” field.

Add permissions policies

AmazonEC2FullAccess

**AmazonEC2FullAccess**

AWS managed

Provides full access to Amazon EC2 via th...

Click the “Search” field.

AmazonS3FullAccess

**AmazonS3FullAccess**

AWS managed

Provides full access to all buckets via t...

**Search**

AmazonDynamoDBFullAccess

**AmazonDynamoDBFullAccess**

AWS managed

Provides full access to Amazon DynamoD...

click Next

Click the “Role name” field.

Type “Jenkins-cicd”

Click “Create role”

**Step 2: Add permissions**

Permissions policy summary

Policy name	Type	Attached as
AmazonDynamoDBFullAccess	AWS managed	Permissions policy
AmazonEC2FullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy

**Step 3: Add tags**

Add tags - optional [Info](#)  
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add new tag  
You can add up to 50 more tags.

Cancel Previous **Create role**

Click “EC2”

go to the Jenkins instance and add this role to the Ec2 instance.

select Jenkins instance -> Actions -> Security -> Modify IAM role

Add a newly created Role and click on Update IAM role.

EC2 > Instances > [i-07885f1ce7fb7174](#) > Modify IAM role

**Modify IAM role** [Info](#)

Attach an IAM role to your instance.

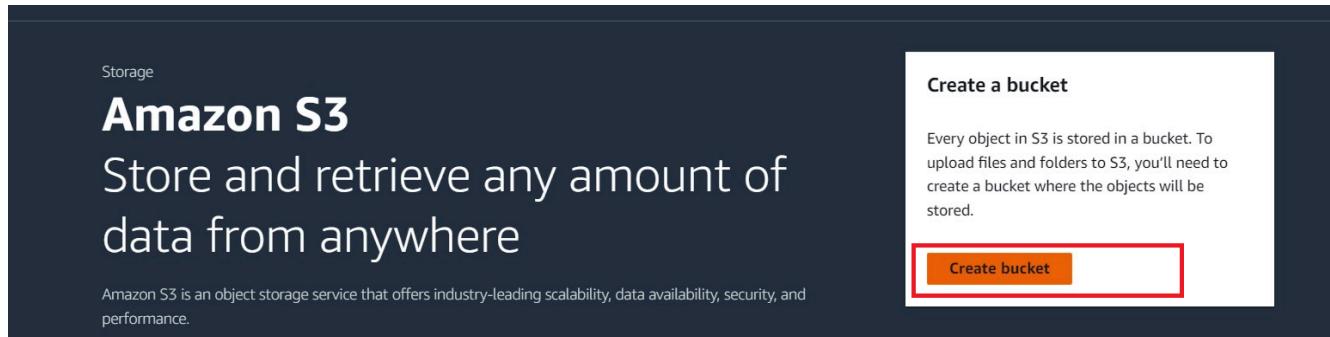
Instance ID  
 [i-07885f1ce7fb7174 \(JENKINS\)](#)

IAM role  
Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.  
Jenkins-cicd

Cancel **Update IAM role**

Search for S3 in console

Click “Create bucket”



Click the “Bucket name” field.

A screenshot of the "Create bucket" configuration page. The top navigation bar shows "Amazon S3 > Buckets > Create bucket". The main section is titled "Create bucket" with an "Info" link. It says "Buckets are containers for data stored in S3. [Learn more](#)". Below this is a "General configuration" section. The "Bucket name" field contains "myawsbucket NAME", which is highlighted with a red box. A note below the field states: "Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)". The "AWS Region" dropdown is set to "Asia Pacific (Mumbai) ap-south-1". Under "Copy settings from existing bucket - optional", it says "Only the bucket settings in the following configuration are copied." and has a "Choose bucket" button. The entire configuration form is enclosed in a light gray box.

Nothing to change, keep the remaining default.

Click “Create bucket”, Bucket will be created.

**Default encryption** [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

Server-side encryption with Amazon S3 managed keys (SSE-S3)

Server-side encryption with AWS Key Management Service keys (SSE-KMS)

Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)  
Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the **Storage** tab of the [Amazon S3 pricing page](#).

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable

Enable

▶ Advanced settings

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

[Cancel](#) [Create bucket](#)

Click the “Search” field. Search for DynamoDB and click on it.

The screenshot shows the AWS search interface. The search bar at the top has 'DynamoDB' typed into it. Below the search bar, the sidebar shows 'Services' selected. The main search results page displays 'Search results for 'Dynamo'' and 'Try searching with longer queries for more relevant results'. Under the 'Services' heading, the 'DynamoDB' service card is highlighted with a red box. The card shows the icon for DynamoDB, the text 'DynamoDB ☆', and 'Managed NoSQL Database'. Below the card, there are 'Top features' links: 'Tables', 'Imports from S3', 'Explore Items', 'Clusters', and 'Reserved Capacity'.

Click “Create table”

Click the “Table name” field. enter “dynamodb\_table = “mrcloudbook-dynamo-db-table””

Click the “Enter the partition key name” field.

Type “LockID”

Click “Create table”

## Tags

Tags are pairs of keys and optional values, that you can assign to AWS resources. You can use tags to control access to your resources or track your AWS spending.

No tags are associated with the resource.

[Add new tag](#)

You can add 50 more tags.

Cancel

[Create table](#)

## Docker Plugin setup

We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

Docker

Docker Commons

Docker Pipeline

Docker API

docker-build-step

and click on install without restart

The screenshot shows the Jenkins Manage Jenkins > Plugins page. The Docker plugin is listed as installed. The Docker Commons and Docker Pipeline plugins are also listed but are not checked. The Docker API plugin is listed but is not checked. The Docker plugin has a status bar indicating it was released 3 days 15 hr ago. The Docker Commons plugin was released 1 mo 29 days ago. The Docker Pipeline plugin was released 27 days ago. The Docker API plugin was released 3 mo 4 days ago. A red box highlights the 'Install' button for the Docker plugin.

Plugin	Status	Last Release
Docker	Installed	3 days 15 hr ago
Docker Commons	Available	1 mo 29 days ago
Docker Pipeline	Available	27 days ago
Docker API	Available	3 mo 4 days ago

Now, goto Dashboard → Manage Jenkins → Tools →

Docker installations

Add Docker

**Docker**

Name: docker

Install automatically

**Download from docker.com**

Docker version: latest

Add Installer

Add DockerHub Username and Password under Global Credentials

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: sevenajay

Treat username as secret

Password: .....

ID: docker

Description: docker

Create

Let's check the Terraform code Now.

### Backend.tf



```
terraform {
  backend "s3" {
    bucket      = "ajay-mrclocloudbook777"    #change name
    key         = "my-terraform-environment/main"
    region      = "ap-south-1"
```

```
dynamodb_table = "mrcloudbook-dynamo-db-table"
}
}
```

## provider.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }
  required_version = ">= 1.2.0"
}
provider "aws" {
  region = var.aws_region
}
```

## main.tf

```
resource "aws_instance" "Ajay" {
  ami                  = var.ami_id
  instance_type        = var.instance_type
  key_name             = var.key_name
  vpc_security_group_ids = [aws_security_group.ec2_security_group.id]
  user_data            = base64encode(file("website.sh"))
  tags = {
    Name = "Aj-EC2"
  }
}
resource "aws_security_group" "ec2_security_group" {
  name      = "ec2 security group"
  description = "allow access on ports 80 and 22 and 443"
```

```
ingress {
    description      = "ssh access"
    from_port       = 22
    to_port         = 22
    protocol        = "tcp"
    cidr_blocks     = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [":/:0"]
}

ingress {
    from_port      = 0
    to_port        = 0      # Allow all ports
    protocol       = "-1" # All protocols
    cidr_blocks    = ["0.0.0.0/0"]
}

ingress {
    description      = "https"
    from_port       = 443
    to_port         = 443
    protocol        = "tcp"
    cidr_blocks     = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [":/:0"]
}

ingress {
    description      = "http"
    from_port       = 80
    to_port         = 80
    protocol        = "tcp"
    cidr_blocks     = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [":/:0"]
}

egress {
    from_port      = 0
    to_port        = 0
    protocol       = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
    ipv6_cidr_blocks = [":/:0"]
}

tags = {
    Name = "Aj_sg"
}
```

s3.tf

```
#create s3 bucket
resource "aws_s3_bucket" "mybucket" {
    bucket = var.bucketname
}

resource "aws_s3_bucket_ownership_controls" "example" {
    bucket = aws_s3_bucket.mybucket.id
    rule {
        object_ownership = "BucketOwnerPreferred"
    }
}

resource "aws_s3_bucket_public_access_block" "example" {
    bucket = aws_s3_bucket.mybucket.id
    block_public_acls      = false
    block_public_policy     = false
    ignore_public_acls     = false
    restrict_public_buckets = false
}

resource "aws_s3_bucket_acl" "example" {
    depends_on = [
        aws_s3_bucket_ownership_controls.example,
        aws_s3_bucket_public_access_block.example,
    ]
    bucket = aws_s3_bucket.mybucket.id
    acl    = "public-read"
}

resource "aws_s3_object" "index" {
    bucket = aws_s3_bucket.mybucket.id
    key    = "index.html"
    source = "index.html"
    acl    = "public-read"
    content_type = "text/html"
}

resource "aws_s3_object" "error" {
    bucket = aws_s3_bucket.mybucket.id
    key    = "error.html"
    source = "error.html"
    acl    = "public-read"
    content_type = "text/html"
}

resource "aws_s3_object" "style" {
```

```

bucket = aws_s3_bucket.mybucket.id
key = "style.css"
source = "style.css"
acl = "public-read"
content_type = "text/css"
}

resource "aws_s3_object" "script" {
  bucket = aws_s3_bucket.mybucket.id
  key = "script.js"
  source = "script.js"
  acl = "public-read"
  content_type = "text/javascript"
}

resource "aws_s3_bucket_website_configuration" "website" {
  bucket = aws_s3_bucket.mybucket.id
  index_document {
    suffix = "index.html"
  }
  error_document {
    key = "error.html"
  }
  depends_on = [ aws_s3_bucket_acl.example.id ]
}

```

## Variables.tf



```

variable "aws_region" {
  description = "The AWS region to create things in."
  default     = "ap-south-1"
}

variable "key_name" {
  description = " SSH keys to connect to ec2 instance"
  default     = "Mumbai"      #change key name here
}

variable "instance_type" {
  description = "instance type for ec2"
  default     = "t2.medium"
}

variable "ami_id" {

```

```
description = "AMI for Ubuntu Ec2 instance"
default      = "ami-0f5ee92e2d63afc18"
}

variable "bucketname" {
    description = "The name of the S3 bucket to create"
    type        = string
    default     = "ajaykumar-yegireddi-cloud" #change Bucket name also
}
```

## User data for Instance

### website.sh

```
#!/bin/bash
# Update the package manager and install Docker
sudo apt-get update -y
sudo apt-get install -y docker.io
# Start the Docker service
sudo systemctl start docker
# Enable Docker to start on boot
sudo systemctl enable docker
# Pull and run a simple Nginx web server container
sudo docker run -d --name zomato -p 3000:3000 sevenajay/zomato:latest
sudo docker run -d --name netflix -p 8081:80 sevenajay/netflix:latest
```

### index.html

```
HTML CSS JSResult Skip Results Iframe
<!DOCTYPE html>
<html lang="en" >
<head>
    <meta charset="UTF-8">
    <title> Login Page Form | Nothing4us</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/mr
```

```
<link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-variant-settings/1.0.0/font-variant-settings.min.css'>
</head>
<body>
<!-- partial:index.partial.html -->
<div class="center">
  <div class="ear ear--left"></div>
  <div class="ear ear--right"></div>
  <div class="face">
    <div class="eyes">
      <div class="eye eye--left">
        <div class="glow"></div>
      </div>
      <div class="eye eye--right">
        <div class="glow"></div>
      </div>
    </div>
    <div class="nose">
      <svg width="38.161" height="22.03">
        <path d="M2.017 10.987Q-.563 7.513.157 4.754C.877 1.994 2.976.1:1.512 0L0 0Z" style="fill:none;stroke:#fff;stroke-width:2px;stroke-linecap:round;stroke-linejoin:round;stroke-miterlimit:10;"/>
      </svg>
      <div class="glow"></div>
    </div>
    <div class="mouth">
      <svg class="smile" viewBox="-2 -2 84 23" width="84" height="23">
        <path d="M0 0c3.76 9.279 9.69 18.98 26.712 19.238 17.022.258 10.258 10.258 0 0L0 0Z" style="fill:none;stroke:#fff;stroke-width:2px;stroke-linecap:round;stroke-linejoin:round;stroke-miterlimit:10;"/>
      </svg>
      <div class="mouth-hole"></div>
      <div class="tongue breath">
        <div class="tongue-top"></div>
        <div class="line"></div>
        <div class="median"></div>
      </div>
    </div>
  </div>
<div class="hands">
  <div class="hand hand--left">
    <div class="finger">
      <div class="bone"></div>
      <div class="nail"></div>
    </div>
    <div class="finger">
      <div class="bone"></div>
      <div class="nail"></div>
    </div>
    <div class="finger">
      <div class="bone"></div>
      <div class="nail"></div>
    </div>
  </div>
</div>
```

```
<div class="bone"></div>
<div class="nail"></div>
</div>
</div>
<div class="hand hand--right">
  <div class="finger">
    <div class="bone"></div>
    <div class="nail"></div>
  </div>
  <div class="finger">
    <div class="bone"></div>
    <div class="nail"></div>
  </div>
  <div class="finger">
    <div class="bone"></div>
    <div class="nail"></div>
  </div>
</div>
<div class="login">
  <label>
    <div class="fa fa-phone"></div>
    <input class="username" type="text" autocomplete="on" placeholder="Username">
  </label>
  <label>
    <div class="fa fa-commenting"></div>
    <input class="password" type="password" autocomplete="off" placeholder="Password">
    <button class="password-button">Show</button>
  </label>
  <button class="login-button">Login</button>
</div>
<div class="social-buttons">
  <div class="social">
    <div class="fa fa-wechat"></div>
  </div>
  <div class="social">
    <div class="fa fa-weibo"></div>
  </div>
  <div class="social">
    <div class="fa fa-paw"></div>
  </div>
</div>
<div class="footer">Mr.Cloud Book</div>
<script src='https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.5/lodash.min.js'>
```

```
</body>
</html>
```

## error.html



```
<!DOCTYPE html>
<html lang="en" >
<head>
    <meta charset="UTF-8">
    <title> 404 page | Nothing4us </title>
    <link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/tu
<link rel='stylesheet' href='https://fonts.googleapis.com/css?family=Arv
</head>
<body>
<!-- partial:index.partial.html -->
<section class="page_404">
    <div class="container">
        <div class="row">
            <div class="col-sm-12 ">
                <div class="col-sm-10 col-sm-offset-1 text-center">
                    <div class="four_zero_four_bg">
                        <h1 class="text-center ">404</h1>
                    </div>
                    <div class="contant_box_404">
                        <h3 class="h2">
                            Look like you're lost
                        </h3>
                        <p>the page you are looking for not avaible!</p>
                        <a href="" class="link_404">Go to Home</a>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
<!-- partial -->
```

```
</body>
</html>
```

## style.css



```
* {
    box-sizing: border-box;
}
body {
    width: 100vw;
    height: 100vh;
    background-color: rgb(41, 0, 75);
    overflow: hidden;
    font-size: 12px;
}
.inspiration {
    position: fixed;
    bottom: 0;
    right: 0;
    padding: 10px;
    text-align: center;
    text-decoration: none;
    font-family: 'Gill Sans', sans-serif;
    font-size: 12px;
    color: #969696;
}
.inspiration img {
    width: 60px;
}
.center {
    position: relative;
    top: 50%;
    left: 50%;
    display: inline-block;
    width: 275px;
    height: 490px;
    border-radius: 3px;
    transform: translate(-50%, -50%);
    overflow: hidden;
```

```
background-image: linear-gradient(to top right, rgb(0 168 255), rgb(0 0 0));
}

@media screen and (max-height: 500px) {
    .center {
        transition: transform 0.5s;
        transform: translate(-50%, -50%) scale(0.8);
    }
}

.center .ear {
    position: absolute;
    top: -110px;
    width: 200px;
    height: 200px;
    border-radius: 50%;
    background-color: rgb(50 22 22);
}

.center .ear.ear--left {
    left: -135px;
}

.center .ear.ear--right {
    right: -135px;
}

.center .face {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 200px;
    height: 150px;
    margin: 80px auto 10px;
    --rotate-head: 0deg;
    transform: rotate(var(--rotate-head));
    transition: transform 0.2s;
    transform-origin: center 20px;
}

.center .eye {
    display: inline-block;
    width: 25px;
    height: 25px;
    border-radius: 50%;
    background-color: #243946;
}

.center .eye.ey--left {
    margin-right: 40px;
}

.center .eye.ey--right {
```

```
margin-left: 40px;
}

.center .eye .glow {
  position: relative;
  top: 3px;
  right: -12px;
  width: 12px;
  height: 6px;
  border-radius: 50%;
  background-color: #fff;
  transform: rotate(38deg);
}

.center .nose {
  position: relative;
  top: 30px;
  transform: scale(1.1);
}

.center .nose .glow {
  position: absolute;
  top: 3px;
  left: 32%;
  width: 15px;
  height: 8px;
  border-radius: 50%;
  background-color: #476375;
}

.center .mouth {
  position: relative;
  margin-top: 45px;
}

.center svg.smile {
  position: absolute;
  left: -28px;
  top: -19px;
  transform: scaleX(1.1);
  stroke: #243946;
}

.center .mouth-hole {
  position: absolute;
  top: 0;
  left: -50%;
  width: 60px;
  height: 15px;
  border-radius: 50%/100% 100% 0% 0;
  transform: rotate(180deg);
}
```

```
background-color: #243946;
z-index: -1;
}
.center .tongue {
position: relative;
top: 5px;
width: 30px;
height: 20px;
background-color: #ffd7dd;
transform-origin: top;
transform: rotateX(60deg);
}
.center .tongue.breath {
-webkit-animation: breath 0.3s infinite linear;
animation: breath 0.3s infinite linear;
}
.center .tongue-top {
position: absolute;
bottom: -15px;
width: 30px;
height: 30px;
border-radius: 15px;
background-color: #ffd7dd;
}
.center .line {
position: absolute;
top: 0;
width: 30px;
height: 5px;
background-color: #fcb7bf;
}
.center .median {
position: absolute;
top: 0;
left: 50%;
transform: translateX(-50%);
width: 4px;
height: 25px;
border-radius: 5px;
background-color: #fcb7bf;
}
.center .hands {
position: relative;
}
.center .hands .hand {
```

```
position: absolute;
top: -6px;
display: flex;
transition: transform 0.5s ease-in-out;
z-index: 1;
}
.center .hands .hand--left {
  left: 50px;
}
.center .hands .hand--left.hide {
  transform: translate(2px, -155px) rotate(-160deg);
}
.center .hands .hand--left.peek {
  transform: translate(0px, -120px) rotate(-160deg);
}
.center .hands .hand--right {
  left: 170px;
}
.center .hands .hand--right.hide {
  transform: translate(-6px, -155px) rotate(160deg);
}
.center .hands .hand--right.peek {
  transform: translate(-4px, -120px) rotate(160deg);
}
.center .hands .finger {
  position: relative;
  z-index: 0;
}
.center .hands .finger .bone {
  width: 20px;
  height: 20px;
  border: 2px solid #243946;
  border-bottom: none;
  border-top: none;
  background-color: rgb(255 211 11);
}
.center .hands .finger .nail {
  position: absolute;
  left: 0;
  top: 10px;
  width: 20px;
  height: 18px;
  border-radius: 50%;
  border: 2px solid #243946;
  background-color: #fac555;
```

```
z-index: -1;
}
.center .hands .finger:nth-child(1),
.center .hands .finger:nth-child(3) {
  left: 4px;
  z-index: 1;
}
.center .hands .finger:nth-child(1) .bone,
.center .hands .finger:nth-child(3) .bone {
  height: 10px;
}
.center .hands .finger:nth-child(3) {
  left: -4px;
}
.center .hands .finger:nth-child(2) {
  top: -5px;
  z-index: 2;
}
.center .hands .finger:nth-child(1) .nail,
.center .hands .finger:nth-child(3) .nail {
  top: 0px;
}
.center .login {
  position: relative;
  display: flex;
  flex-direction: column;
}
.center .login label {
  position: relative;
  padding: 0 20px;
}
.center .login label .fa {
  position: absolute;
  top: 40%;
  left: 35px;
  color: #bbb;
}
.center .login label .fa:before {
  position: relative;
  left: 1px;
}
.center .login input,
.center .login .login-button {
  width: 100%;
  height: 35px;
```

```
border: none;
border-radius: 30px;
}
.center .login input {
padding: 0 20px 0 40px;
margin: 5px 0;
box-shadow: none;
outline: none;
}
.center .login input::-moz-placeholder {
color: #ccc;
}
.center .login input:-ms-input-placeholder {
color: #ccc;
}
.center .login input::placeholder {
color: #ccc;
}
.center .login input.password {
padding: 0 90px 0 40px;
}
.center .login .password-button {
position: absolute;
top: 9px;
right: 25px;
display: flex;
justify-content: center;
align-items: center;
width: 80px;
height: 27px;
border-radius: 30px;
border: none;
outline: none;
background-color: #243946;
color: #fff;
}
.center .login .password-button:active {
transform: scale(0.95);
}
.center .login .login-button {
width: calc(100% - 40px);
margin: 20px 20px 0;
outline: none;
background-color: #243946;
color: #fff;
```

```
transition: transform 0.1s;
}
.center .login .login-button:active {
  transform: scale(0.95);
}
.center .social-buttons {
  display: flex;
  justify-content: center;
  margin-top: 25px;
}
.center .social-buttons .social {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 35px;
  height: 35px;
  margin: 0 10px;
  border-radius: 50%;
  background-color: #243946;
  color: #fff;
  font-size: 18px;
}
.center .social-buttons .social:active {
  transform: scale(0.95);
}
.center .footer {
  text-align: center;
  margin-top: 15px;
}
@webkit-keyframes breath {
  0%, 100% {
    transform: rotateX(0deg);
  }
  50% {
    transform: rotateX(60deg);
  }
}
@keyframes breath {
  0%, 100% {
    transform: rotateX(0deg);
  }
  50% {
    transform: rotateX(60deg);
  }
}
```

```
    }
}
```

## script.js



```
let usernameInput = document.querySelector('.username');
let passwordInput = document.querySelector('.password');
let showPasswordButton = document.querySelector('.password-button');
let face = document.querySelector('.face');
passwordInput.addEventListener('focus', event => {
  document.querySelectorAll('.hand').forEach(hand => {
    hand.classList.add('hide');
  });
  document.querySelector('.tongue').classList.remove('breath');
});
passwordInput.addEventListener('blur', event => {
  document.querySelectorAll('.hand').forEach(hand => {
    hand.classList.remove('hide');
    hand.classList.remove('peek');
  });
  document.querySelector('.tongue').classList.add('breath');
});
usernameInput.addEventListener('focus', event => {
  let length = Math.min(usernameInput.value.length - 16, 19);
  document.querySelectorAll('.hand').forEach(hand => {
    hand.classList.remove('hide');
    hand.classList.remove('peek');
  });
  face.style.setProperty('--rotate-head', `${-length}deg`);
});
usernameInput.addEventListener('blur', event => {
  face.style.setProperty('--rotate-head', '0deg');
});
usernameInput.addEventListener('input', _.throttle(event => {
  let length = Math.min(event.target.value.length - 16, 19);
  face.style.setProperty('--rotate-head', `${-length}deg`);
}, 100));
showPasswordButton.addEventListener('click', event => {
  if (passwordInput.type === 'text') {
```

```

passwordInput.type = 'password';
document.querySelectorAll('.hand').forEach(hand => {
  hand.classList.remove('peek');
  hand.classList.add('hide');
});
} else {
  passwordInput.type = 'text';
  document.querySelectorAll('.hand').forEach(hand => {
    hand.classList.remove('hide');
    hand.classList.add('peek');
  });
}
});

```

Let's create a Job now in Jenkins

set a job name and add this pipeline



```

`aform'

>ol 'sonar-scanner'

>rkspace'){
;()

: from Git'){

nch: 'main', url: 'https://github.com/Aj7Ay/TERRAFORM-JENKINS-CICD.git'

`m version'){


```

```

  terraform --version

  >e Analysis "){

  |arQubeEnv('sonar-server') {
    ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Terraform \
  |sonar.projectKey=Terraform '''

  gate"){

  {
  |tForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'

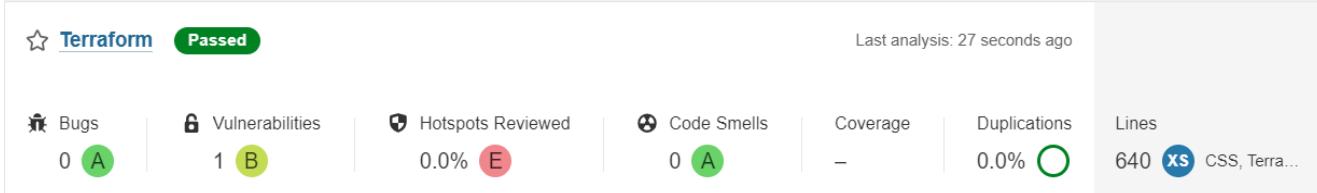
  ; SCAN') {

  |vy fs . > trivyfs.txt"

```

stage view

Declarative: Tool Install	checkout from Git	check terraform version	Sonarqube Analysis	quality gate	TRIVY FS SCAN
76ms	1s	420ms	8s	253ms	1s
70ms	1s	422ms	8s	241ms (paused for 2s)	1s



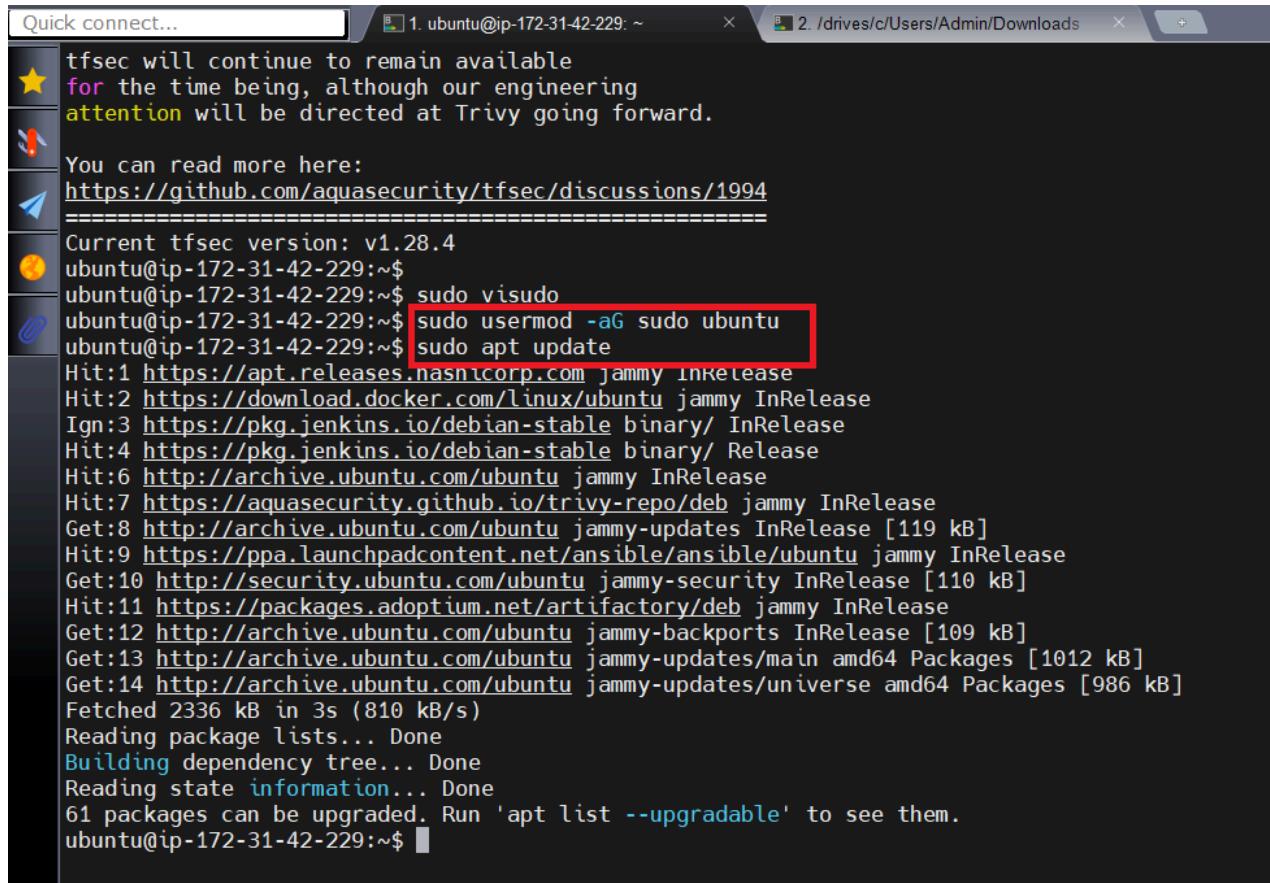
we have to provide Executable permissions for our user data otherwise it won't run.

If we run with sudo directly we will get the error

To give a user sudo permissions on an Ubuntu system, you need to add the user to the `sudo` group or grant them specific sudo access by editing the sudoers file. Here are two common ways to give a user sudo permissions:

### Method 1: Add User to the sudo Group

1. Log in to your Ubuntu system as a user with sudo privileges, or log in as the root user.
2. Open a terminal.
3. Run the following command to add a user (replace `<username>` with the actual username) to the `sudo` group
4. After running the command, the user will have sudo privileges. They can now execute commands with superuser privileges using `sudo`.



```

Quick connect... 1. ubuntu@ip-172-31-42-229: ~ 2. /drives/c/Users/Admin/Downloads
tfsec will continue to remain available
for the time being, although our engineering
attention will be directed at Trivy going forward.

You can read more here:
https://github.com/aquasecurity/tfsec/discussions/1994
=====
Current tfsec version: v1.28.4
ubuntu@ip-172-31-42-229:~$ sudo visudo
ubuntu@ip-172-31-42-229:~$ sudo usermod -aG sudo ubuntu
ubuntu@ip-172-31-42-229:~$ sudo apt update
Hit:1 https://apt.releases.nasnicorp.com jammy InRelease
Hit:2 https://download.docker.com/linux/ubuntu jammy InRelease
Ign:3 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:4 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:6 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:7 https://aquasecurity.github.io/trivy-repo/deb jammy InRelease
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:9 https://ppa.launchpadcontent.net/ansible/ansible/ubuntu jammy InRelease
Get:10 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:11 https://packages.adoptium.net/artifactory/deb jammy InRelease
Get:12 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1012 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [986 kB]
Fetched 2336 kB in 3s (810 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
61 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-42-229:~$ 

```

sudo usermod -aG sudo ubuntu  
 sudo apt update



## Method 2: Edit the sudoers File

In some cases, you may want more fine-grained control over a user's sudo permissions. To do this, you can edit the sudoers file using the `visudo` command:

1. Log in to your Ubuntu system as a user with sudo privileges, or log in as the root user.
2. Open a terminal.
3. Run the following command to edit the sudoers file:
4. `sudo visudo`





```
# User privilege specification  
<username> ALL=(ALL:ALL) ALL
```



If you want to limit their sudo access to specific commands, you can specify those commands instead of ALL .

Save and exit the text editor. On most systems, you can save and exit `visudo` by pressing `Ctrl + X`, followed by `Y`, and then `Enter`.

After completing one of these methods, the user should have sudo permissions and can run commands with superuser privileges using `sudo`. Make sure to exercise caution when granting sudo access to users, as it can provide them with significant control over the system.

After granting sudo permissions to a user on a Ubuntu system, you do not need to restart the system for the changes to take effect. The user can immediately start using `sudo` to run commands with superuser privileges.

To test whether the user can use `sudo`, you can simply open a terminal and have the user run a command with `sudo`. For example:



```
sudo apt update
```



If the user is prompted for their password and the command executes without errors, it means that the sudo permissions have been successfully granted and applied.

There is no need to restart the system to apply changes to sudo permissions.

Now add the below stages to your pipeline



```
stage('Excutable permission to userdata'){
    steps{
        sh 'chmod 777 website.sh'
    }
}
stage('Terraform init'){
    steps{
        sh 'terraform init'
    }
}
stage('Terraform plan'){
    steps{
        sh 'terraform plan'
    }
}
```



## stage view

Declarative: Tool Install	checkout from Git	check terraform version	Sonarqube Analysis	quality gate	TRIVY FS SCAN	Provide Executable permissions	Terraform init	Terraform plan
76ms	1s	420ms	8s	253ms	1s	419ms	3s	5s
70ms	1s	422ms	8s	241ms (paused for 2s)	1s	418ms	3s	5s

if you want to add a security check for terraform files that also works fine but if we use that now we will get so many errors. coz we just wrote simple terraform files that's why it throws errors. we have `aqua tfsec` and `checkov` for terraform scan.

install aqua security [tfsec](#)



```
curl -s https://raw.githubusercontent.com/aquasecurity/tfsec/master/scr:
```



```

Quick connect... 1. ubuntu@ip-172-31-42-229: ~
ubuntu@ip-172-31-42-229:~$ curl -s https://raw.githubusercontent.com/aquasecurity/tfsec/master/scripts/install_linux.sh | bash
arch=amd64
remote_filename=tfsec-linux-amd64
local_filename=tfsec
checkgen_filename=tfsec-checkgen-linux-amd64

=====
Looking up the latest version...
Downloading tfsec v1.28.4
Downloading tfsec-linux-amd64...
Downloaded file "tfsec-linux-amd64" successfully
Downloading tfsec-checkgen-linux-amd64...
Downloaded file "tfsec-checkgen-linux-amd64" successfully
Downloading tfsec_checksums.txt...
Downloaded file "tfsec_checksums.txt" successfully
Checksum verified successfully

=====
Installing /tmp/tfsec.uRGdViAKy6/tfsec to /usr/local/bin/...
'/tmp/tfsec.uRGdViAKy6/tfsec' -> '/usr/local/bin/tfsec'
Cleaning downloaded files...

=====
tfsec is joining the Trivy family

tfsec will continue to remain available
for the time being, although our engineering
attention will be directed at Trivy going forward.

You can read more here:
https://github.com/aquasecurity/tfsec/discussions/1994
=====
Current tfsec version: v1.28.4
ubuntu@ip-172-31-42-229:~$ 

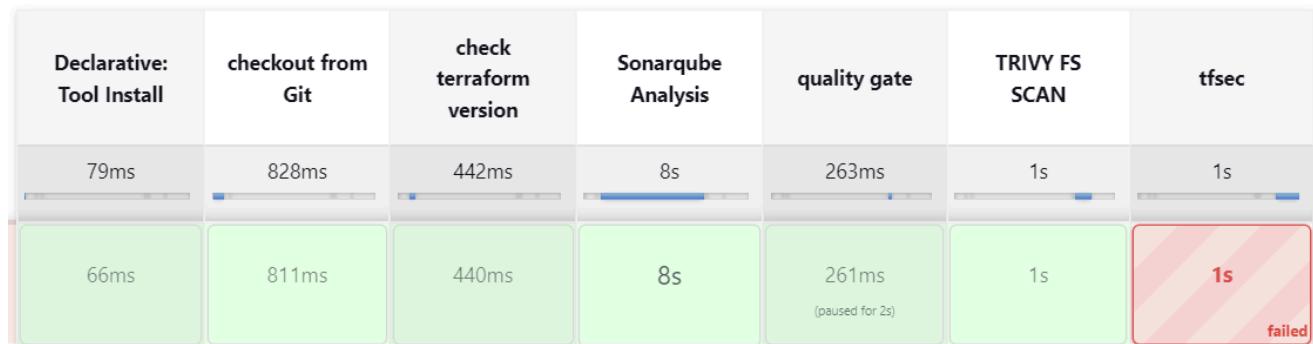
```

Add this stage. For this project, our learning purpose isn't recommended.

```

stage('Trivy terraform scan'){
    steps{
        sh 'tfsec . --no-color'
    }
}

```



Let's continue without that step.

Add this stage to the pipeline



```
stage('Terraform apply'){
    steps{
        sh 'terraform apply --auto-approve'
    }
}
```



you will succeed but I want to do this with build parameters to apply and destroy while building only.

you have to add this inside job like below image

The screenshot shows the Jenkins Pipeline configuration page. Under the 'Configure' section, there is a 'General' tab selected. A checkbox labeled 'This project is parameterized' is checked. Below it, there is a 'Choice Parameter' configuration block. It has a 'Name' field set to 'action'. In the 'Choices' field, there are two options: 'apply' and 'destroy'. There is also a 'Description' field which is currently empty. At the bottom of the configuration block are 'Save' and 'Apply' buttons.



```
stage('Terraform apply'){
    steps{
        sh "terraform ${action} --auto-approve"
    }
}
```



**Pipeline Terraform**

This build requires parameters:

action  
apply

Build

While at apply stage it automatically takes apply option and creates infrastructure in AWS and runs containers

```
[Pipeline] sh
+ terraform apply --auto-approve
@[0m@[1maws_s3_bucket.example_bucket: Refreshing state... [id=mrcloudbook-777-ajaykumar-yegireddi]@[0m
@0m@[1maws_security_group.ec2_security_group: Refreshing state... [id=sg-0a0997e8c2d4c35bb]@[0m

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
@[32m+[0m create@[0m
@0m@[32m-[0m destroy and then create replacement@[0m

Terraform will perform the following actions:

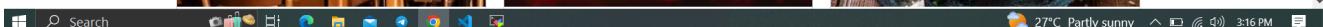
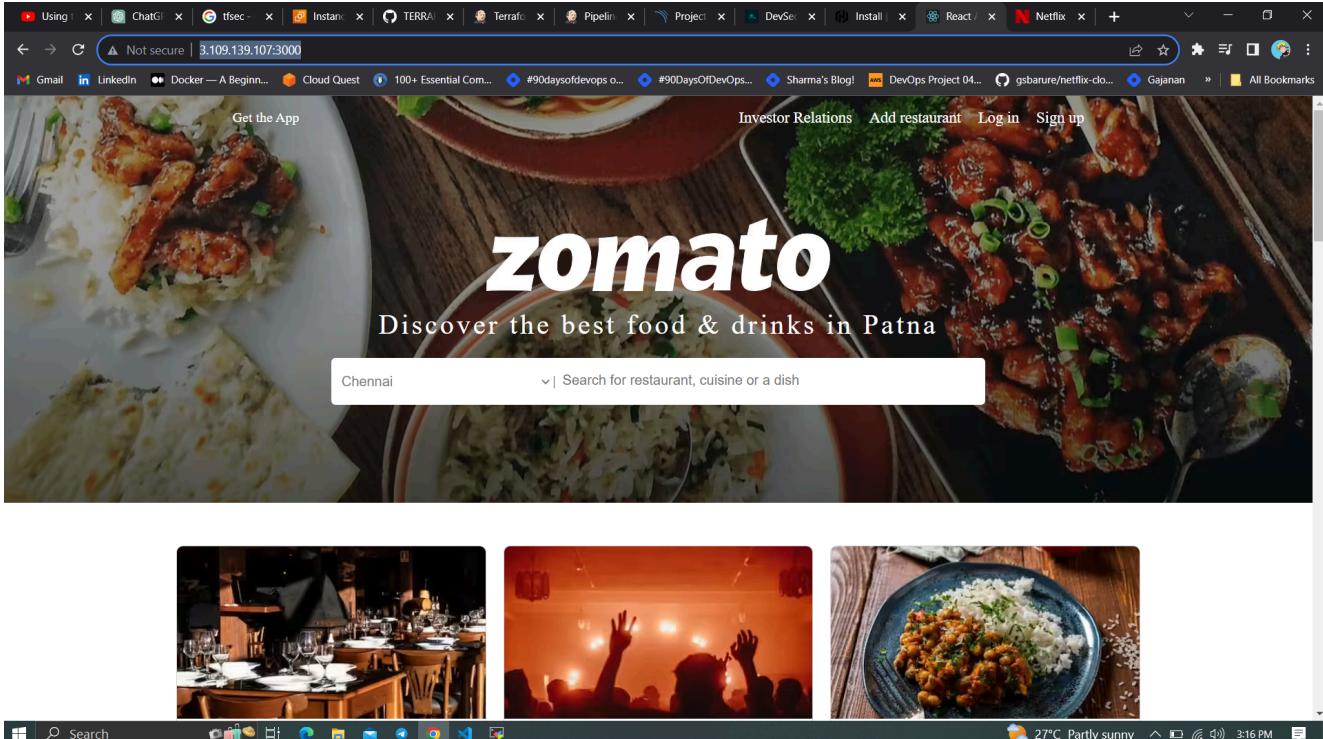
@1m # aws_instance.Ajay@[0m will be created
@[0m @[32m+[0m@[0m resource "aws_instance" "Ajay" {
    ami = "ami-0f5ee92e2d63afc18"
    arn = (known after apply)
    associate_public_ip_address = (known after apply)
    availability_zone = (known after apply)
    cpu_core_count = (known after apply)
    cpu_threads_per_core = (known after apply)
    disable_api_stop = (known after apply)
    disable_api_termination = (known after apply)
    ebs_optimized = (known after apply)
```

Now copy the newly created Instance Ip address

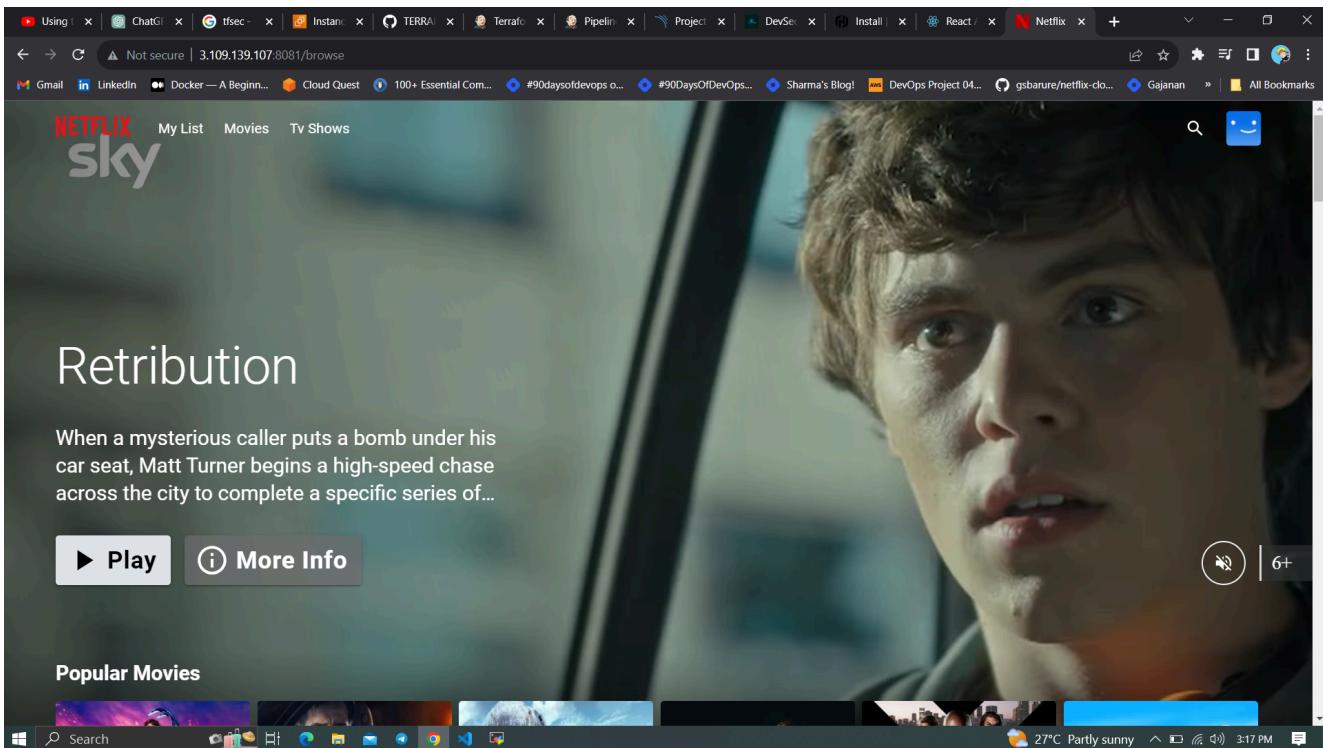


<instance-ip:3000> #zomato app container





<instance-ip:8081> #netflix container



check s3 bucket is created or not

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with links like YouTube, simple-ter..., Ajay Kumar, TERRAFOR..., mrcloudb..., EC2 Instan..., ajtf Jenkins, Terraform, Login Page, and others. Below the navigation bar, the main content area displays a bucket named 'mrcloudbookaj-7-ajaykumar-yegireddi'. The bucket is set to 'Publicly accessible'. The 'Objects' tab is selected, showing a list of 4 objects:

Name	Type	Last modified	Size	Storage class
error.html	html	September 29, 2023, 16:37:45 (UTC+05:30)	912.0 B	Standard
index.html	html	September 29, 2023, 16:37:45 (UTC+05:30)	3.2 KB	Standard
script.js	js	September 29, 2023, 16:37:45 (UTC+05:30)	1.7 KB	Standard
style.css	css	September 29, 2023, 16:37:45 (UTC+05:30)	7.1 KB	Standard

Below the object list are buttons for Copy S3 URI, Copy URL, Download, Open, Delete, Actions (with a dropdown menu), Create folder, and Upload. There's also a search bar labeled 'Find objects by prefix'.

The screenshot shows a web browser window with multiple tabs open at the top, including YouTube, simple-ter..., Ajay Kumar, TERRAFOR..., mrcloudb..., EC2 Instan..., ajtf Jenkins, Terraform, Login Page, and others. The main content area displays a static website from the S3 bucket 'mrcloudbookaj-7-ajaykumar-yegireddi.s3-website.ap-south-1.amazonaws.com'. The page has a dark purple background with a central graphic of a smiling dog's face. Below the graphic is a login form with fields for 'Username' and 'Password', a 'Show' button for the password field, and a 'Login' button. At the bottom of the form is the text 'Mr Cloud Book'. The browser's taskbar at the bottom shows various icons for system functions.

Check your s3 bucket for the tf state file with the name main

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various navigation links like 'Buckets', 'Access Points', 'Object Lambda Access Points', etc. The main content area shows a bucket named 'my-terraform-environment'. Inside this bucket, there is one object named 'main'. The object details show it's a standard file type, 180.0 B in size, last modified on September 27, 2023, at 15:16:40 (UTC+05:30).

## Let's Destroy everything

The screenshot shows the Jenkins Pipeline Terraform configuration page. Under the 'action' dropdown, the value 'destroy' is selected. At the bottom of the page, there is a green 'Build' button.

while at the apply stage, it automatically takes the destroy option and deletes everything that we created till now.

The screenshot shows the Jenkins Pipeline Terraform execution log. It highlights the command '+ terraform destroy --auto-approve' in red. The log then displays the execution plan and the actions that Terraform will perform, including destroying resources like AWS buckets, security groups, instances, and associate public IP addresses.

Dashboard &gt; Terraform &gt; #14

```
[0m[1maws_security_group.ec2_security_group: Destruction complete after 0s[0m
[33m[0m[0m[0m
[33m[0m[0m[1m[33mWarning: [0m[0m[1mArgument is deprecated[0m
[33m[0m[0m[0m
[33m[0m[0m[0m with aws_s3_bucket.example_bucket,
[33m[0m[0m[0m on s3.tf line 1, in resource "aws_s3_bucket" "example_bucket":
[33m[0m[0m[0m 1: resource "aws_s3_bucket" "example_bucket" [4m{[0m[0m
[33m[0m[0m[0m
[0m[1m[32m
Destroy complete! Resources: 3 destroyed.

[0m
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

If you find this blog Helpful make sure to give a like and subscribe.

complete pipeline



```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        terraform 'terraform'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanws()
            }
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Aj7Ay/TERRA'
            }
        }
    }
}
```

```
stage('Terraform version'){
    steps{
        sh 'terraform --version'
    }
}

stage("Sonarqube Analysis"){
    steps{
        withSonarQubeEnv('sonar-server') {
            sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=sonar-project -Dsonar.projectKey=Terraform '''
        }
    }
}

stage("quality gate"){
    steps {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: ''
        }
    }
}

stage('TRIVY FS SCAN') {
    steps {
        sh "trivy fs . > trivyfs.txt"
    }
}

stage('Executable permission to userdata'){
    steps{
        sh 'chmod 777 website.sh'
    }
}

stage('Terraform init'){
    steps{
        sh 'terraform init'
    }
}

stage('Terraform plan'){
    steps{
        sh 'terraform plan'
    }
}

stage('Terraform apply'){
    steps{
        sh 'terraform ${action} --auto-approve'
    }
}
```

```
    }  
}
```



**Ajay Kumar Yegireddi** is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

## Comments

2 responses to “AWS Resources with Terraform, Jenkins ci-cd, and Hosting a static website in s3”



**Paul**

26 July 2025

Great stuff, as a junior devopps, what tools am I required to be proficient in?

[Reply](#)



**mrcloudbook.com**

7 August 2025

AI, Docker and Kubernetes and Security and Network

[Reply](#)

## Leave a Reply

Your email address will not be published. Required fields are marked \*

Comment \*

Name \*

Email \*

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot

reCAPTCHA  
[Privacy](#) - [Terms](#)

Post Comment

Uncategorized

**Day -1: Kick Off Cloud Security with AWS Registration**

18 August 2025

Uncategorized

**How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point**

22 July 2025

AI

**How to Run Docker Model Runner on Ubuntu 24.04**

11 July 2025

## Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.



## Important Links

[Privacy Policy](#)

[Terms & Conditions](#)

[Contact](#)

## Resources

[Blog](#)

[YouTube Channel](#)

