

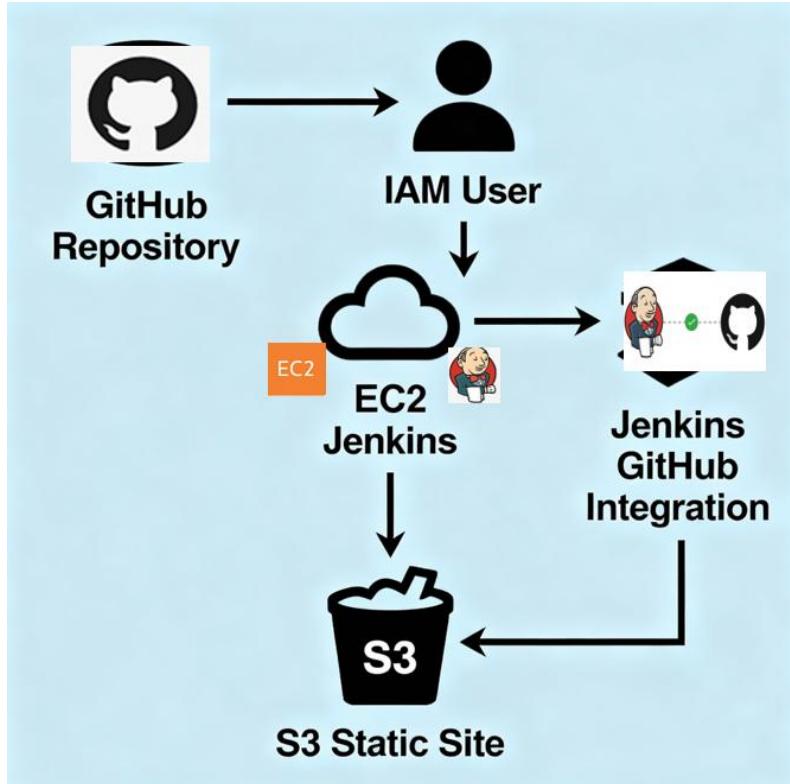
CI/CD Pipeline Setup: IAM, EC2, S3, GitHub, and Jenkins Integration

Objective

To set up a complete CI/CD pipeline that automatically builds and deploys a static website from GitHub to Amazon S3, using Jenkins running on an EC2 instance.

GitHub Link: <https://github.com/johnmicky1/lms-static-website>

Flow Diagram:



Source: <https://www.perplexity.ai/search/create-flow-diagram-with-image-SIKuRsZcRBebm1SvWB9YRA?0=i>

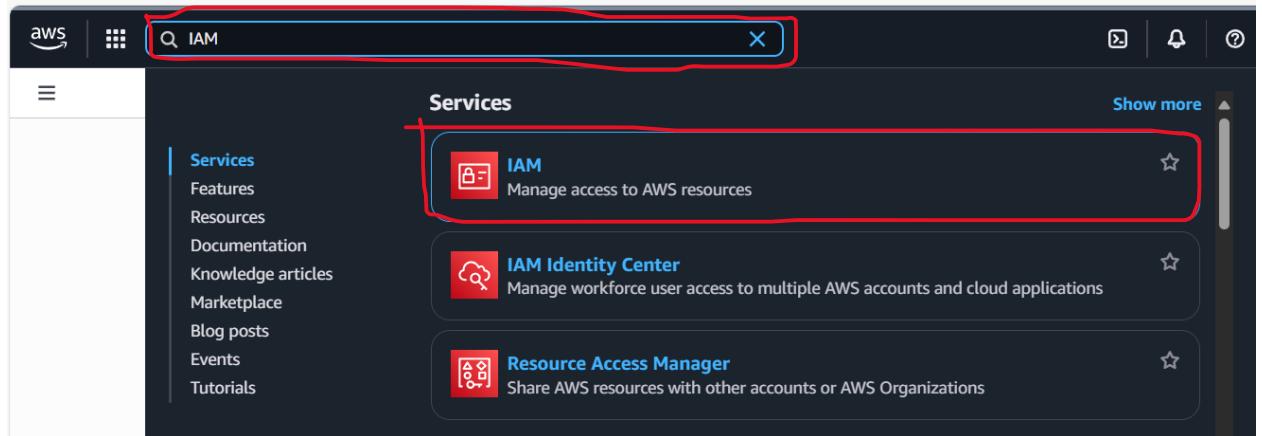
1. AWS IAM Configuration

1.1 Create an IAM User or Role for Jenkins

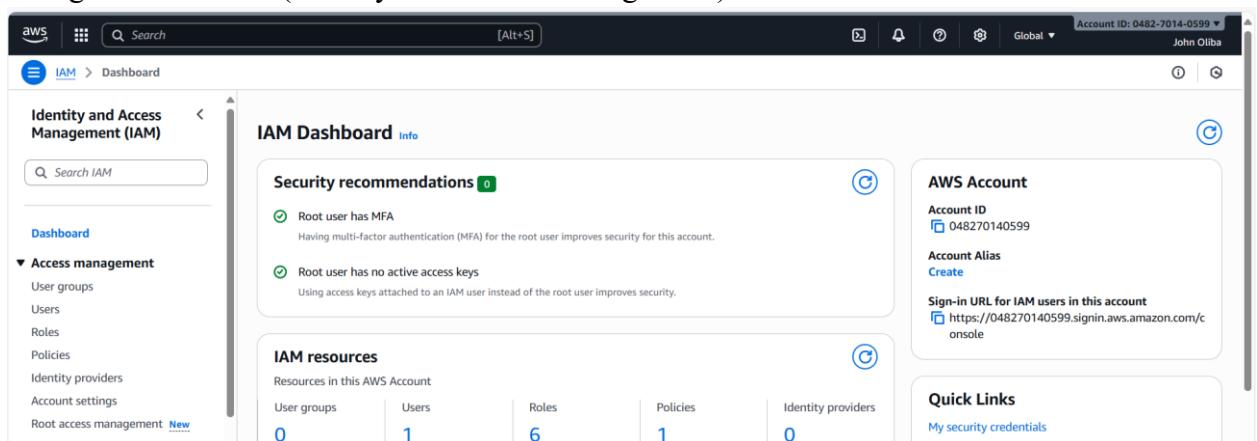
- **Purpose:** Grant permissions for EC2, S3, and CloudWatch integration.
- **Required Policies:**
 - AmazonS3FullAccess
 - AmazonEC2FullAccess
 - CloudWatchLogsFullAccess
 - (*Optional for full control*) AdministratorAccess

Create Access Key and Secret Access Key

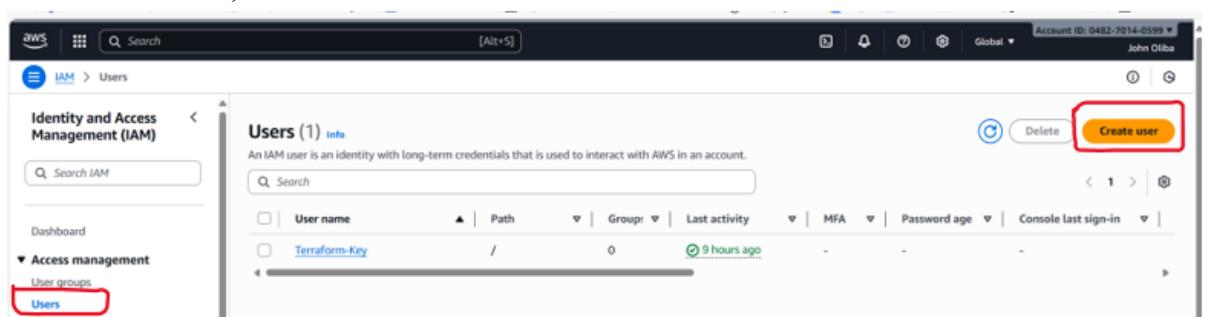
- Log in to the AWS Management Console and type IAM in the search box



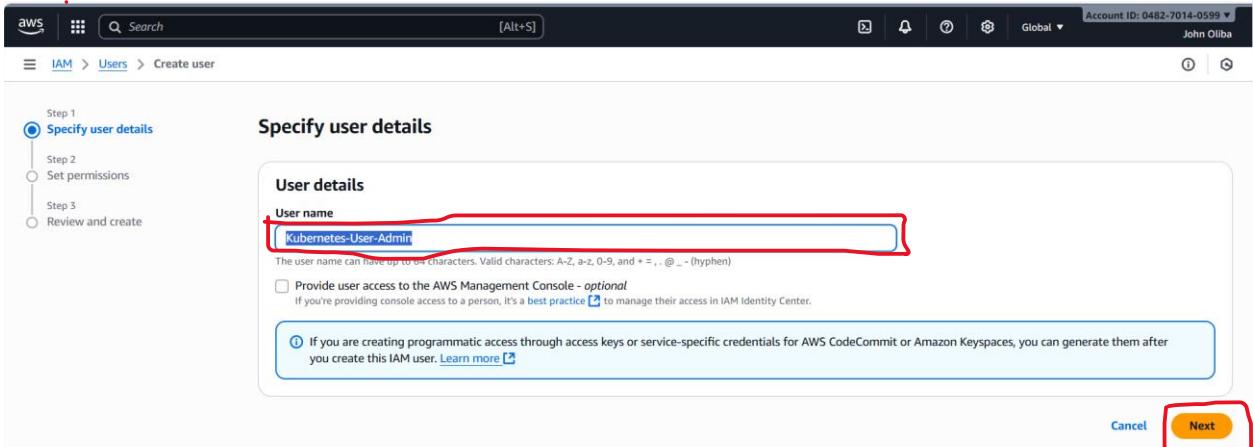
- Navigate to the IAM (Identity and Access Management) dashboard.



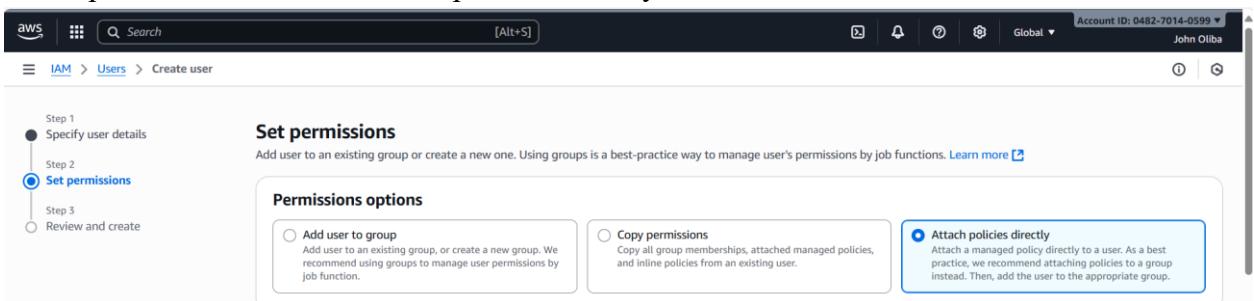
- In the left sidebar, select Users and click on create user



- d) Specify User details by giving a user name and click next (e.g Jenkins-EC2)



- e) To set permissions; click on attach policies directly and scroll down



- f) Select AdministratorAccess

The screenshot shows the 'Permissions policies' selection screen. The 'AdministratorAccess' policy is selected and highlighted with a red box. Other policies listed include 'AccessAnalyzerServiceRolePolicy' (AWS managed), 'AdministratorAccess-Amplify' (AWS managed), and 'AdministratorAccess-AWSElasticBeanstalk' (AWS managed). The 'Create policy' button is visible at the top right.

- g) Type S3 in the search box and select AmazonS3FullAccess

The screenshot shows the 'Permissions policies' selection screen with a search bar containing 's3'. The 'AmazonS3FullAccess' policy is selected and highlighted with a red box. Other policies listed include 'AmazonDMSRedshiftF3Role' (AWS managed) and 'AmazonS3FullAccess' (AWS managed).

- h). Type AmazonEC2FullAccess in the search box and select AmazonEC2FullAccess

The screenshot shows the 'Permissions policies' selection screen with a search bar containing 'AmazonEC2FullAccess'. The 'AmazonEC2FullAccess' policy is selected and highlighted with a red box. The 'Filter by Type' dropdown is set to 'All types'.

- i) Type **CloudWatchLogsFullAccess** in the search box and select **CloudWatchLogsFullAccess** then click **Next**

A screenshot of the AWS IAM Policies search interface. A red box highlights the search bar at the top containing "CloudWatchLogsFullAccess". Below the search bar, a table lists policies. A second red box highlights the "Policy name" column, where "CloudWatchLogsFullAccess" is checked. A third red box highlights the "Type" column, which shows "AWS managed". The table also includes columns for "Attached entities" (0) and "AWS managed".

- j) You should be able to see a summary of the permissions you have created then click on **Create User**

A screenshot of the AWS IAM Create User page. The "User details" section shows a user named "Jenkins-EC2" with "Console password type" set to "None" and "Require password reset" set to "No". A large red box surrounds the "Permissions summary" section below. This section contains a table with five rows, each representing a policy: "AdministratorAccess", "AmazonEC2FullAccess", "AmazonS3FullAccess", "CloudWatchLogsFullAccess", and another "CloudWatchLogsFullAccess". The table has columns for "Name", "Type", and "Used as".

- k) The User has been created

A screenshot of the AWS IAM User Details page for "Jenkins-EC2". The table header includes columns for "User name", "Path", "Group", "Last activity", "MFA", "Password age", "Console last sign-in", and "Access keys". The user "Jenkins-EC2" is listed with a path of "/". The "Summary" section shows ARN (arn:aws:iam::048270140599:user/Jenkins-EC2), Console access (Disabled), and Last console sign-in (-). The "Security credentials" tab is highlighted with a red box. A "Create access key" button is located in the top right corner of this section.

- m) Scroll down to **Access keys (0)** and Click on **Create Access Key**

A screenshot of the AWS IAM Access Keys page for "Jenkins-EC2". The title "Access keys (0)" is highlighted with a red box. A note below states: "Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time." A "Create access key" button is located in the top right corner of the page.

- n) Select **Command Line Interface (CLI)** Scroll Down and Click Check box having **I understand the above recommendation and want to proceed to create an access key**. Then Click **Next**

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

- Command Line Interface (CLI)**
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation

I understand the above recommendation and want to proceed to create an access key.

Cancel **Next**

- o) Set description tag – optional (Give it a name e.g Jenkin-EC2-Access) and Click on **Create Access Key**

Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Jenkin-EC2-Access

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

Create access key

Cancel **Previous**

- p) Save the Access Key in a sure location on your local machine because it will be

necessary for Accessing the AWS CLI using your local machine.

The screenshot shows the AWS IAM 'Create access key' page. A green banner at the top says 'Access key created' with a note: 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below this, there are three steps: Step 2 - optional 'Set description tag', Step 3 'Retrieve access keys' (which is selected), and 'Access key best practices'. The 'Access key' section contains two fields: 'Access key' (containing 'AKIAQWPI74S3ZMQHAX5Z') and 'Secret access key' (containing '*****'). A red box highlights the 'Access key' field. At the bottom right are 'Download .csv file' and 'Done' buttons.

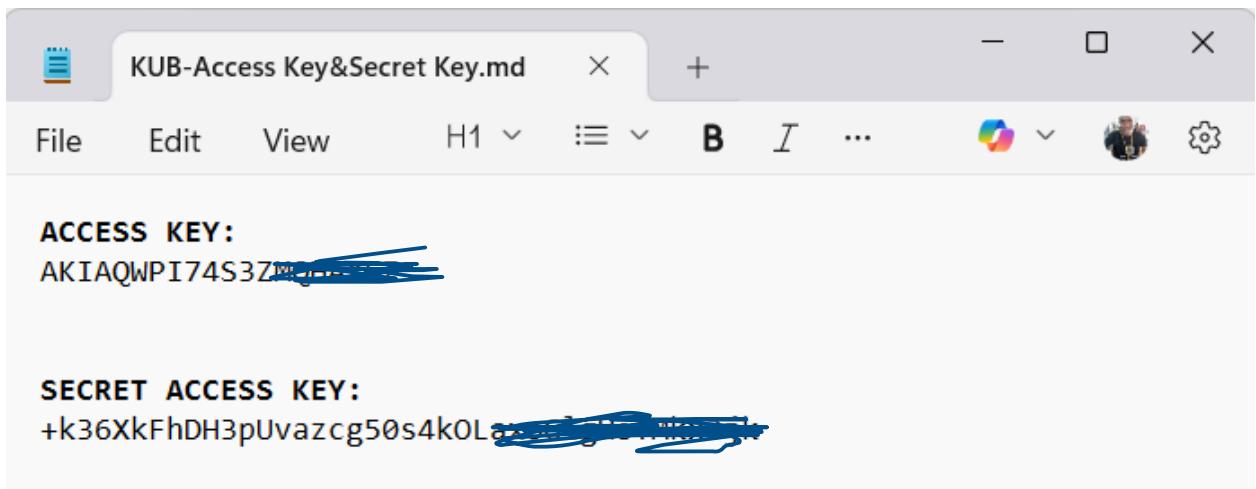
The screenshot shows the AWS IAM 'Secret access key' page. A green banner says 'Access key Copied'. Below it, the 'Access key' field contains 'AKIAQWPI74S3ZMQHAX5Z', which is also highlighted with a red box. To the right, there is a 'Show' button next to a redacted secret access key. The text 'If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.' is visible above the secret access key field.

Open Note Pad, Type **ACCESS KEY** and paste the Access Key as illustrated in the Image below:

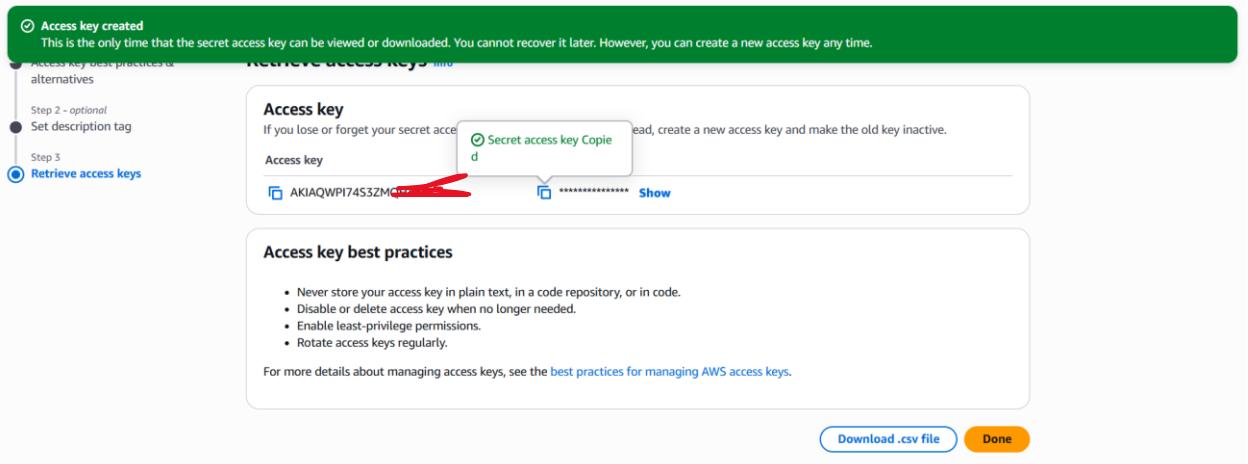
The screenshot shows a Notepad window titled 'KUB-Access Key&Secret Key.txt'. The content of the note pad is: 'ACCESS KEY:' followed by the copied access key 'AKIAQWPI74S3ZMQHAX5Z'. The entire text is highlighted with a red box.

- q) Type on the same notepad page **SECRET ACCESS KEY** then copy and paste it as seen below:

The screenshot shows the AWS IAM 'Secret access key' page again. A green banner says 'Secret access key Copied'. Below it, the 'Secret access key' field contains '*****', which is highlighted with a red box. To the left, the 'Access key' field contains 'AKIAQWPI74S3ZMQHAX5Z', also highlighted with a red box. The text 'If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.' is visible above the secret access key field.



- r) Download the CSV File As well and Save it in a secure location on your local machine.

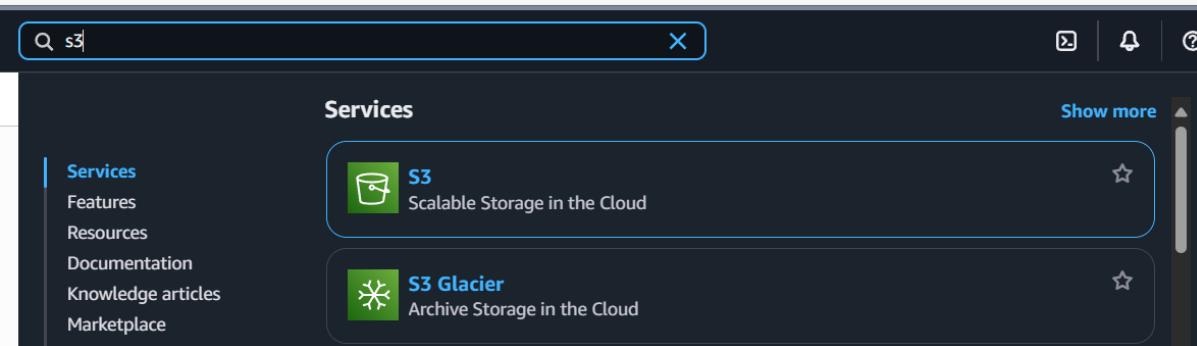


A C.S.V file will be download, make sure it is saved in a secure location

■ Step 2: Set Up AWS S3 Bucket

2.1. Create an S3 Bucket

- ❖ Log in to the AWS Management Console. And Type **S3** in the search box.



❖ Click on S3 → Click Create bucket.

❖ Enter a **unique bucket name**, e.g., *my-static-website-bucket-jmo-2025*

Choose your preferred **AWS Region** (e.g., US East (N. Virginia) us-east-1).

⚠ Important: Bucket names must be globally unique.

[Create bucket](#) [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type

[Info](#)

General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

[Bucket name](#) [Info](#)

my-static-website-bucket-jmo-2025

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn More](#)

❖ Configure Bucket Settings for Static Hosting

a. To Disable, scroll down "Block Public Access"

1. In the "Block Public Access" section, uncheck:

- o Block *all* public access

2. Confirm acknowledgment that your bucket will be public.

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

[Block all public access](#)

Turning off this setting will have no effect on turning on all four settings below. Each of the following settings are independent of one another.

[Block public access to buckets and objects granted through new access control lists \(ACLs\)](#)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

[Block public access to buckets and objects granted through any access control lists \(ACLs\)](#)

S3 will ignore all ACLs that grant public access to buckets and objects.

[Block public access to buckets and objects granted through new public bucket or access point policies](#)

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

[Block public and cross-account access to buckets and objects through any public bucket or access point policies](#)

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

❖ Enable Object Ownership

b. Under Object Ownership, select **ACLs enabled**.

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Object Ownership

[ACLs disabled \(recommended\)](#)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

[ACLs enabled](#)

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

⚠ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

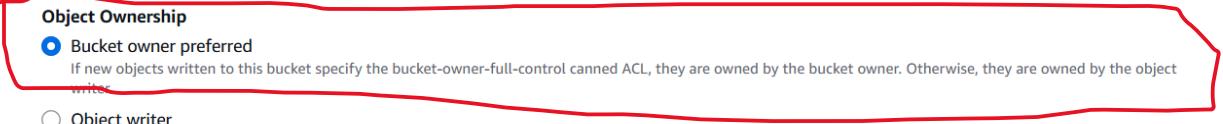
Object Ownership

❖ Choose **Bucket owner preferred**.

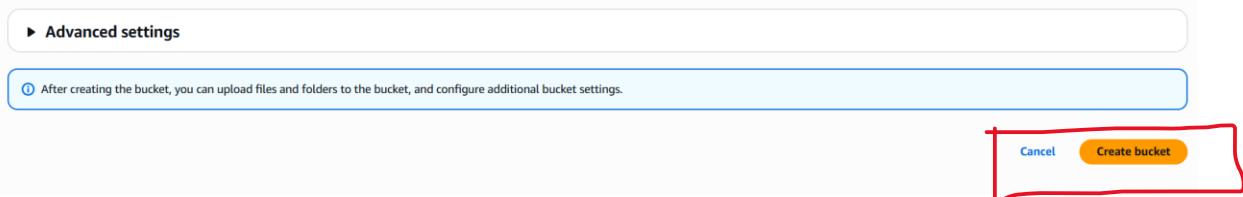
Object Ownership

Bucket owner preferred
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

Object writer
The object writer remains the object owner.



1. Click on Create Bucket.



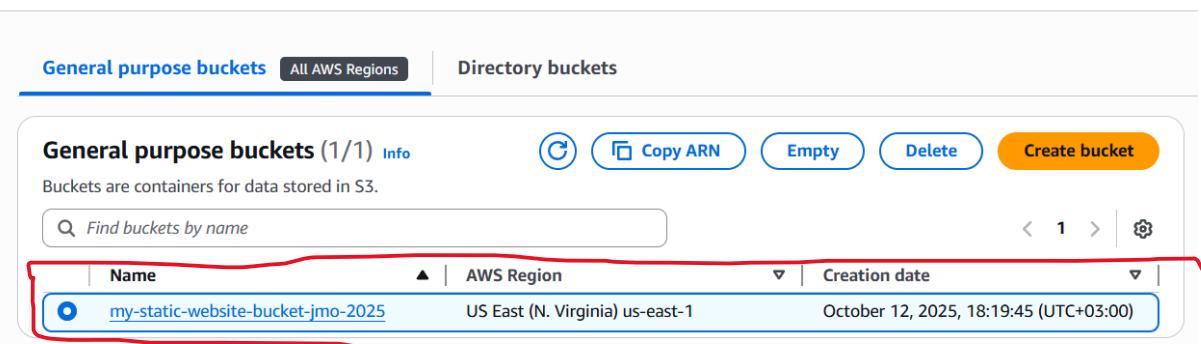
► Advanced settings

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

Create bucket

❖ Enable Static Website Hosting

C. Click on the Bucket created



General purpose buckets All AWS Regions

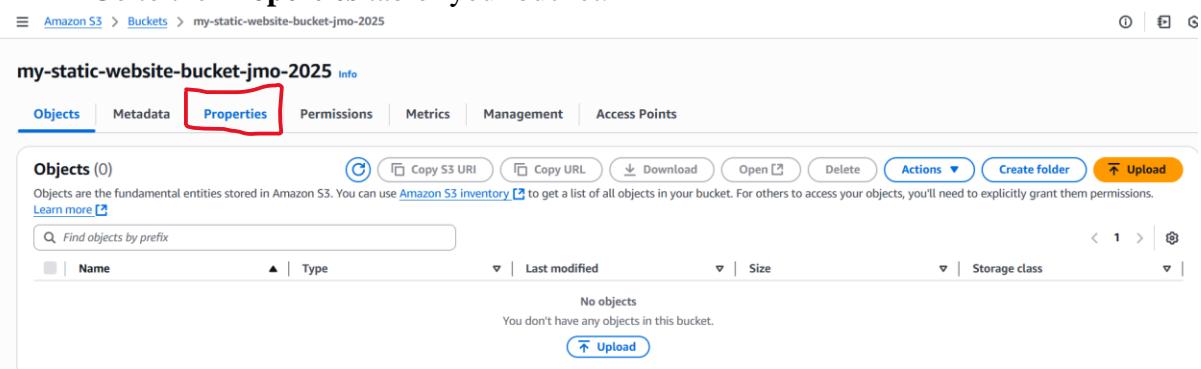
General purpose buckets (1/1) Info

Buckets are containers for data stored in S3.

Name AWS Region Creation date

my-static-website-bucket-jmo-2025	US East (N. Virginia) us-east-1	October 12, 2025, 18:19:45 (UTC+03:00)
-----------------------------------	---------------------------------	--

❖ Go to the Properties tab of your bucket.



Amazon S3 > Buckets > my-static-website-bucket-jmo-2025

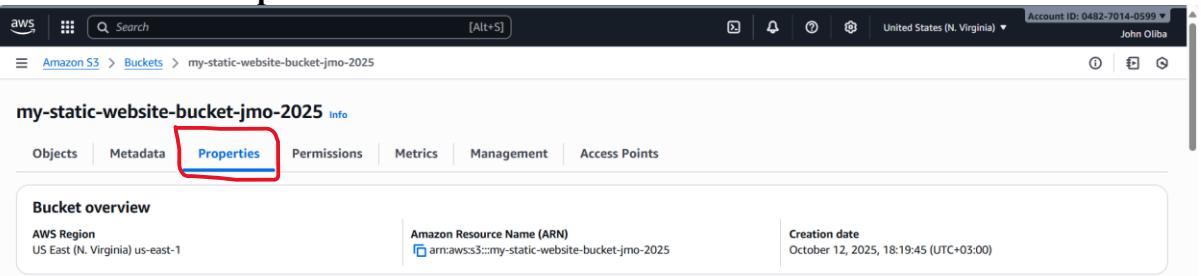
my-static-website-bucket-jmo-2025 Info

Objects (0)

No objects

Upload

❖ Click on Properties



Search [Alt+S]

United States (N. Virginia) Account ID: 0482-7014-0599 John Oliba

my-static-website-bucket-jmo-2025 Info

Objects Metadata Permissions Metrics Management Access Points

Bucket overview

AWS Region: US East (N. Virginia) us-east-1

Amazon Resource Name (ARN): arn:aws:s3:::my-static-website-bucket-jmo-2025

Creation date: October 12, 2025, 18:19:45 (UTC+03:00)

- ❖ Scroll down to **Static website hosting** → Click **Edit**→Select **Enable**.

Edit static website hosting Info

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#) 

Static website hosting

- Disable
 Enable

- ❖ Choose **Host a static website**

Hosting type

- Host a static website

Use the bucket endpoint as the web address. [Learn more](#) 

- ❖ Enter:

- **Index document:** index.html
- **Error document:** error.html

Index document

Specify the home or default page of the website.

index.html

Error document - optional

This is returned when an error occurs.

error.html

- ❖ Save changes.

[Cancel](#)

[Save changes](#)

- ❖ Set Bucket Policy for Public Access

d) Click on the Bucket created and select permissions.

my-static-website-bucket-jmo-2025 Info

Objects

Metadata

Properties

Permissions

Metrics

Management

Access Points

Permissions overview

Access finding

Access findings are provided by IAM external access analyzers. Learn more about [How IAM analyzer findings work](#) 

[View analyzer for us-east-1](#)

❖ Edit the Bucket Policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

No policy to display.

Copy

❖ Click on policy generator

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Bucket ARN
arn:aws:s3:::my-static-website-bucket-jmo-2025

Policy

Policy examples

Policy generator

❖ Under type of policy; select **S3 Bucket Policy**

Type of Policy

S3 Bucket Policy

❖ Under Principal, type * (To make it public)

Principal

*

Use a comma to separate multiple values.

❖ Under Actions; Clock on the Drop Down Menu and select **GetObject**

Actions

All Actions ("*")

-Select Actions-

get*

GetObject

❖ To get Amazon Resource Name (ARN) go back to the previous Tab and **COPY** the bucket policy as shown in the Image below.

Edit bucket policy [Info](#)

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Bucket ARN copied

arn:aws:s3:::my-static-website-bucket-jmo-2025

Policy

1

- ❖ Paste it to the section having **Amazon Resource Name (ARN)**
- ❖ Add **/*** at the end to make it public

Amazon Resource Name (ARN)

All Resources ("*")

arn:aws:s3:::my-static-website-bucket-jmo-2025/*

ARN should follow the following format: arn:aws:s3:::\${BucketName}/\${KeyName}. Use a comma to separate multiple values.

- ❖ Go back to the **previous Tab** and Click On **Add Statement**
- ▶ **Add conditions (optional)**

Add Statement

- ❖ You will be able to get the output below (**Confirming that Statements have been added**)

Statements added (1)					
Principal(s)	Effect	Action	Resource(s)	Condition(s)	Remove
*	Allow	s3:GetObject	arn:aws:s3:::my-static-website-bucket-jmo-2025/*	None	Remove

- ❖ Click on **Generate Policy**

Step 3: Generate policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

Generate Policy

- ❖ Copy the Policy

Policy JSON Document

Click below to edit. To save the policy, copy the text below to a text editor. Changes made below will **not be reflected in the policy generator tool**.

```

1 * []
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "Statement1",
6       "Effect": "Allow",
7       "Principal": "*",
8       "Action": [
9         "s3:GetObject"
10       ],
11       "Resource": "arn:aws:s3:::my-static-website-bucket-jmo-2025/*"
12     }
13   ]
14 
```

11 JSON

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided as is without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

Close **Copy Policy**

- ❖ Go back to the **previous Tab** and **Paste** it in the section having **Edit Bucket Policy**; scroll down and **Save Changes**

Edit bucket policy Info

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Bucket ARN

arn:aws:s3:::my-static-website-bucket-jmo-2025

Policy

```

1 ▼ {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "Statement1",
6       "Effect": "Allow",
7       "Principal": "*",
8       "Action": [
9         "s3:GetObject"
10      ],
11       "Resource": "arn:aws:s3:::my-static-website-bucket-jmo-2025/*"
12     }
13   ]
}

```

e) Upload the Folder having html, css and js files to the bucket

This step involves uploading a folder containing website files (HTML, CSS, and JS) to an S3 bucket, making them accessible online via a URL:

<https://my-static-website-bucket-jmo-2025.s3.us-east-1.amazonaws.com/Capstone-Project-LMS/index.html>

- ❖ Select the Bucket you created and click on **Objects**

Amazon S3 > Buckets > my-static-website-bucket-jmo-2025

my-static-website-bucket-jmo-2025 Info

Objects

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Name	Type	Last modified	Size	Storage class
------	------	---------------	------	---------------

- ❖ To Upload the folder; Click **Upload**, followed by **Add Folder** then **Navigate to** where the folder is **and Upload**

Amazon S3 > Buckets > my-static-website-bucket-jmo-2025 > Upload

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDKs or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (0)
All files and folders in this table will be uploaded.

Name	Folder	Type	Size
------	--------	------	------

- ❖ You will get a pages like this showing that all files have been successively uploaded

Files and folders (13 total, 97.1 KB)						
Name	Folder	Type	Size	Status	Error	
course-github.html	Capstone-Project-LMS/	text/html	7.6 KB	Succeeded	-	
course-jenkins.html	Capstone-Project-LMS/	text/html	8.6 KB	Succeeded	-	
courses.html	Capstone-Project-LMS/	text/html	28.0 KB	Succeeded	-	
css-auth.css	Capstone-Project-LMS/	text/css	3.5 KB	Succeeded	-	
css-dashboard.css	Capstone-Project-LMS/	text/css	7.4 KB	Succeeded	-	
css-style.css	Capstone-Project-LMS/	text/css	11.2 KB	Succeeded	-	
dashboard.html	Capstone-Project-LMS/	text/html	7.7 KB	Succeeded	-	
file-structure	Capstone-Project-LMS/	-	411.0 B	Succeeded	-	
index.html	Capstone-Project-LMS/	text/html	10.9 KB	Succeeded	-	
js-app.js	Capstone-Project-LMS/	text/javascript	2.5 KB	Succeeded	-	

f) Accessing the Static Website

- Click on index.html

index.html	Capstone-Project-LMS/	text/html	10.9 KB	Succeeded
----------------------------	-----------------------	-----------	---------	-----------

- Copy the Object URL from the bottom right corner.

Entity tag (Etag)

a1dfbe819c9d360dc6bdf68c19299668

Object URL

<https://my-static-website-bucket-jmo-2025.s3.us-east-1.amazonaws.com/Capstone-Project-LMS/index.html>

<https://my-static-website-bucket-jmo-2025.s3.us-east-1.amazonaws.com/Capstone-Project-LMS/index.html>

- The website should now be live and accessible!

The screenshot shows the homepage of the DevOps LMS. At the top, there's a navigation bar with 'DevOps LMS' and links for 'Home', 'Courses', 'Dashboard', and 'Get Started'. Below the navigation, there's a large purple banner with the text 'Transform Your Career with DevOps Mastery'. Underneath the banner, it says 'Master GitHub and Jenkins through industry-proven courses designed for modern software development. Learn CI/CD, automation, and collaboration from experts.' There are three course cards: 'GitHub Version Control' (with a GitHub icon), 'DevOps Best Practices' (with a rocket icon), and 'Jenkins CI/CD Automation' (with a gear icon). At the bottom, there's a 'Start Learning Free' button and a 'Explore Courses' link.

💻 Step 3: GitHub Setup

- ✓ Log in to [GitHub](#). And click on **Repositories**

The screenshot shows the GitHub interface for the user 'johnmicky1'. At the top, there's a navigation bar with tabs for 'Overview', 'Repositories' (which is highlighted with a red box), 'Projects', 'Packages', and 'Stars'. Below the navigation is a search bar and some filters ('Type', 'Language', 'Sort'). A prominent green button labeled 'New' is also highlighted with a red box. The main area displays three repositories: 'program-calculator' (Python Program to Calculate Word Frequency in a Sentence), 'leap-year-checker' (Python program to check whether a year is a leap year or not), and 'Voice-Recorder' (Python program to record audio). Each repository has a star icon and a 'Star' dropdown menu.

- ✓ Click **New Repository**.

The screenshot shows the 'Create a new repository' form. It starts with a header 'Create a new repository' and a note about repositories containing project files and version history. There are two main sections: 'General' and 'Configuration'. In the 'General' section, the 'Owner' is set to 'johnmicky1' and the 'Repository name' is 'lms-static-website'. The 'Description' field is empty. In the 'Configuration' section, the 'Choose visibility' dropdown is set to 'Public' (highlighted with a red box). There are options to 'Add README' (with a 'No README' button) and 'Add .gitignore' (with a 'No .gitignore' dropdown).

- ✓ Name it, for example: lms-static-website.

The screenshot shows the 'Repository name' input field with the value 'lms-static-website'. Below the input, a green message says 'lms-static-website is available.' The entire input field and its message are highlighted with a large red box.

- ✓ Initialize with a **README** and optionally add no gitignore.

The screenshot shows the 'Add README' and 'Add .gitignore' sections. Under 'Add README', there's a note about READMEs being longer descriptions. Under 'Add .gitignore', there's a note about .gitignore files not being tracked. To the right, there's a 'No .gitignore' dropdown with a 'No .gitignore' button highlighted with a red box.

- ✓ Create Repository

The screenshot shows the final step of creating the repository. A large red box highlights the green 'Create repository' button at the bottom of the page.

- ✓ Clone the repo locally:

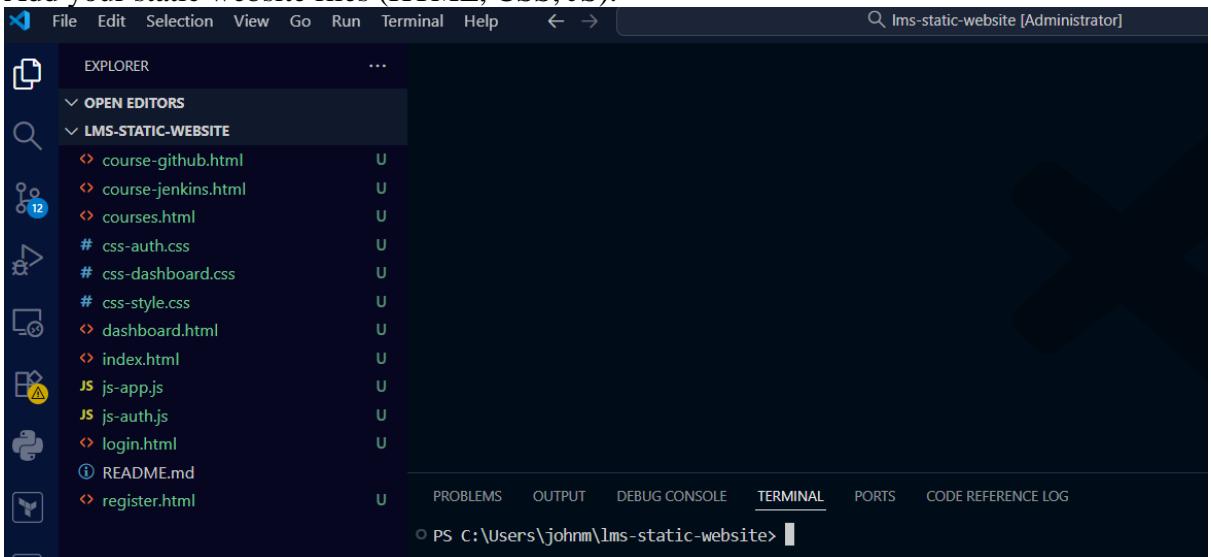
```
git clone git@github.com:johnmicky1/lms-static-website.git
johnm@JMO26-04-2025 MINGW64 ~ (master)
$ git clone git@github.com:johnmicky1/lms-static-website.git
Cloning into 'lms-static-website'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
```

- ✓ Opening VS Code to Add the (HTML, CSS, JS) files
- ✓ **cd lms-static-website** then followed by **code .** (to open VS Code)

```
johnm@JMO26-04-2025 MINGW64 ~ (master)
$ cd lms-static-website

johnm@JMO26-04-2025 MINGW64 ~/lms-static-website (main)
$ code .
```

- ✓ Add your static website files (HTML, CSS, JS).



Commit and push:

- ✓ **git add .**

```
● PS C:\Users\johnm\lms-static-website> git add .
```

- ✓ **git commit -m "Added Static Webfiles"**

```
PS C:\Users\johnm\lms-static-website> git commit -m "Added static webfiles"
[main 72ebcf9] Added static webfiles
 12 files changed, 3318 insertions(+)
 create mode 100644 course-github.html
 create mode 100644 course-jenkins.html
```

✓ **git status**

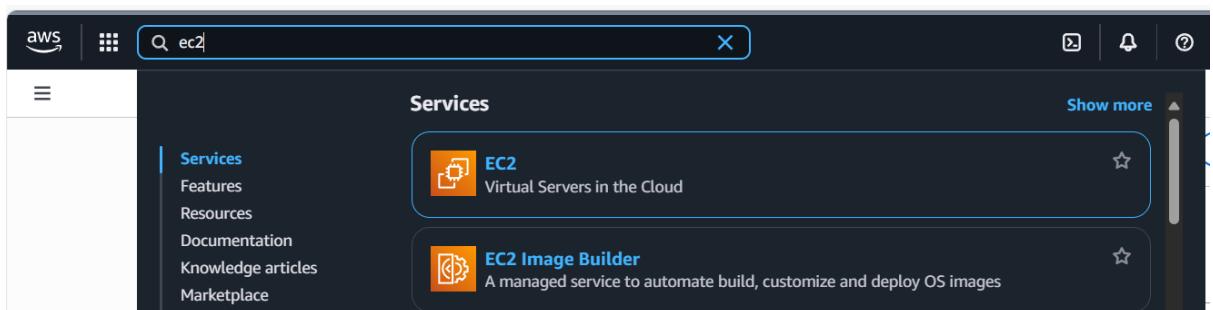
```
● PS C:\Users\johnm\lms-static-website> git status
  On branch main
    Your branch is ahead of 'origin/main' by 1 commit.
```

✓ **git push**

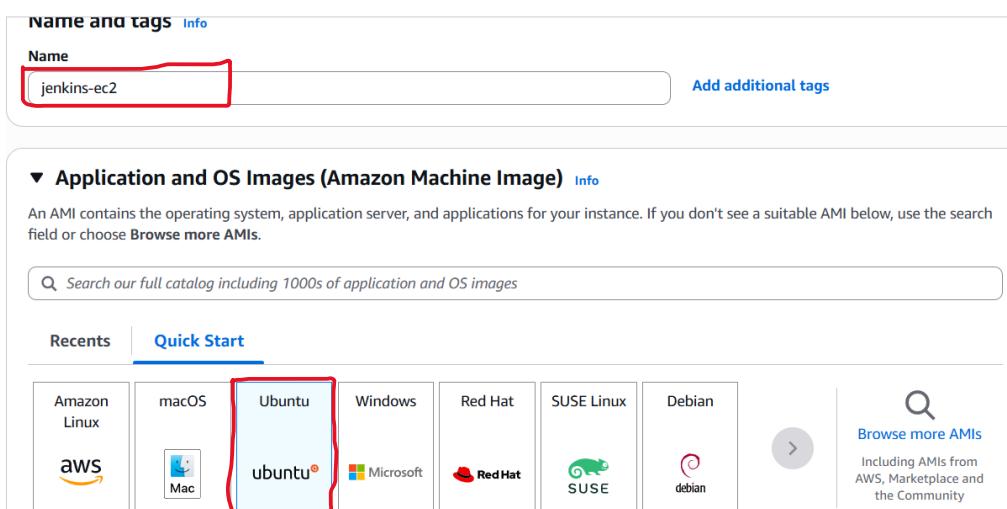
```
PS C:\Users\johnm\lms-static-website> git push
  Enumerating objects: 15, done.
  Counting objects: 100% (15/15), done.
  Delta compression using up to 8 threads
  Compressing objects: 100% (14/14), done.
  Writing objects: 100% (14/14), 19.67 KiB | 915.00 KiB/s, done.
  Total 14 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
  remote: Resolving deltas: 100% (4/4), done.
  To github.com:johnmicky1/lms-static-website.git
    e4e5c20..72ebcf9  main -> main
```

4. EC2 Setup (for Jenkins)

4.1 Launch EC2 Instance



- Give it a name of your choice e.g. **jenkins-ec2** & select **Ubuntu** for Image.



- Scroll down and create a key pair

Instance type [Info](#) | [Get advice](#)

Instance type

t3.micro Free tier eligible

Family: t3 2 vCPU 1 GiB Memory Current generation: true
 On-Demand Ubuntu Pro base pricing: 0.0139 USD per Hour
 On-Demand SUSE base pricing: 0.0104 USD per Hour On-Demand Linux base pricing: 0.0104 USD per Hour
 On-Demand RHEL base pricing: 0.0392 USD per Hour On-Demand Windows base pricing: 0.0196 USD per Hour

Additional costs apply for AMIs with pre-installed software

All generations [Compare instance types](#)

Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

Select [Create new key pair](#)

Create a Key Pair

- Give the key pair a name (e.g., "jenkins-key").
- Click "Create key pair."
- Save the downloaded key file in a secure location to use for SSH access to your EC2 instance.

Create key pair

X

Key pair name

Key pairs allow you to connect to your instance securely.

jenkins-key

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

RSA

RSA encrypted private and public key pair

ED25519

ED25519 encrypted private and public key pair

Private key file format

.pem

For use with OpenSSH

.ppk

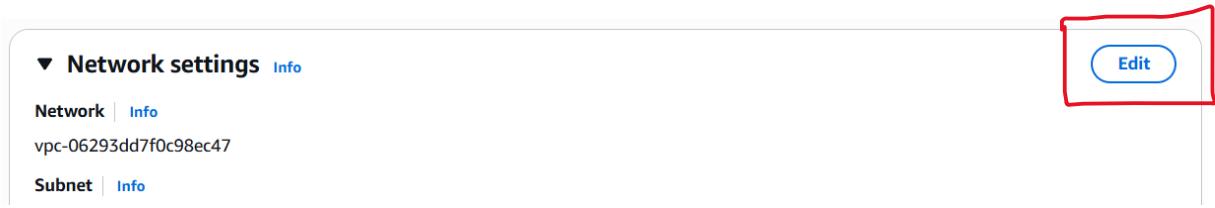
For use with PuTTY

⚠️ When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. [Learn more](#)

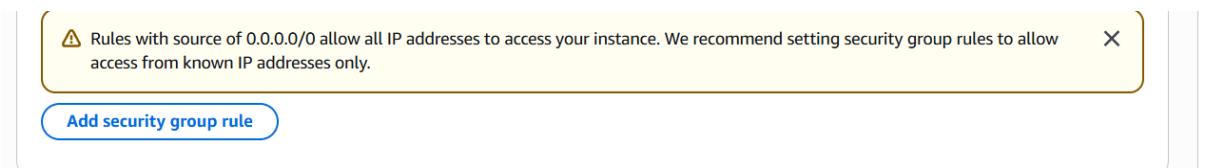
Cancel

Create key pair

2. Under Network Settings, click Edit

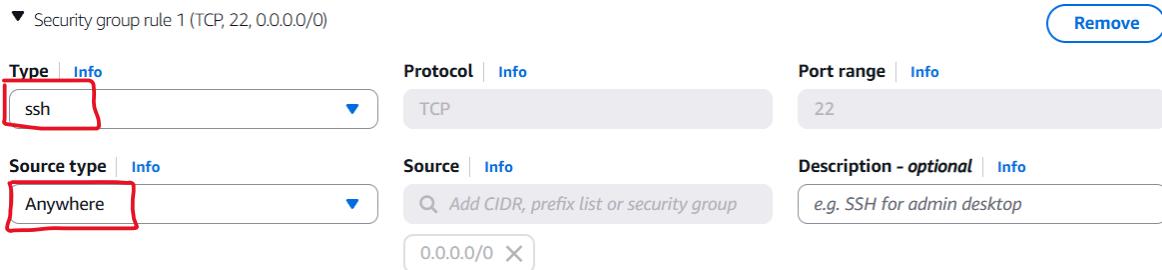


Click Add Security Group Rule (Security Group (inbound rules)) 😊

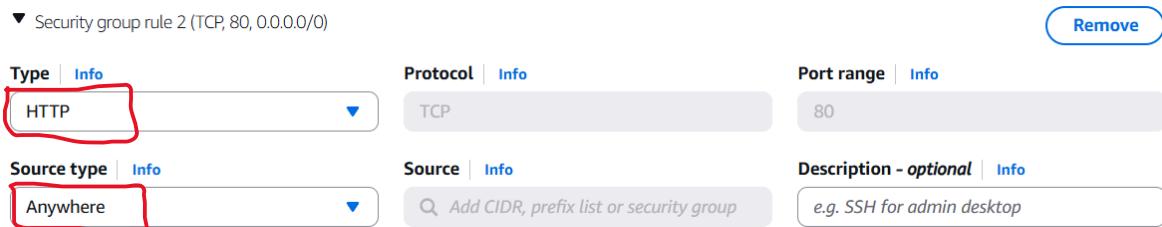


Add the port numbers as shown in the Images below

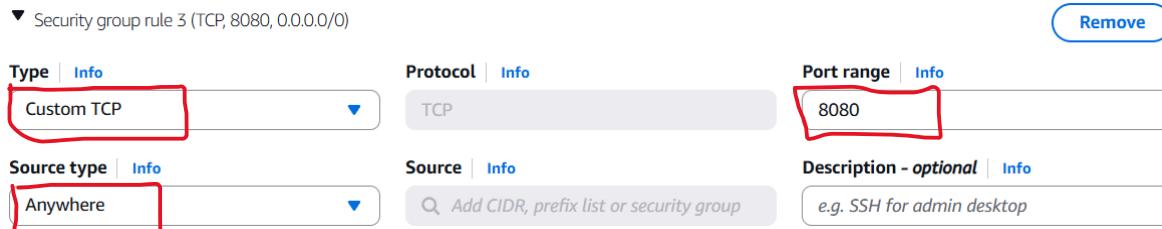
- SSH (TCP 22)



- HTTP (TCP 80)



- Jenkins (TCP 8080) — restrict to your IP or workplace IP range



3. Launch and wait until instance is **running**.

4.2 Connect via SSH

- Select Instance and Click **Connect**

The screenshot shows the AWS EC2 Instances page. At the top, there's a search bar and several filter buttons: 'All states', 'Actions', and 'Launch instances'. Below the header, a table lists one instance: 'jenkins-ec2' (Instance ID: i-03eb5653188c68560), which is 'Running'. Other columns include 'Status check' (3/3 checks passed), 'View alarms', 'Availability Zone' (us-east-1c), and 'Public IPv4' (ec2-54-163-2-3). The 'Connect' button in the top right is highlighted with a red box.

- Under **SSH Client Tab**, copy the **ssh key** at the bottom

The screenshot shows the 'Connect to instance' page for the 'jenkins-ec2' instance. It has tabs for 'EC2 Instance Connect', 'Session Manager', and 'SSH client', with 'SSH client' highlighted with a red box. Below the tabs, it says 'Connect to an instance using the browser-based client.' Under 'Instance ID', it shows 'i-03eb5653188c68560 (jenkins-ec2)'. It provides instructions for using an SSH client, including steps to open a private key file and run a command. It also shows the Public DNS and IP address. At the bottom, there's an example command: 'ssh -i "jenkins-key.pem" ubuntu@ec2-54-163-2-3.compute-1.amazonaws.com'. A note at the bottom says 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.'

- Click **EC2 Instance connect followed by connect**, the **CLI** will open then paste the **SSH key** you copied from the **SSH Client**

From your terminal:

Path·(of the key pair-where it was downloaded): <C:\Users\johnm\OneDrive\Desktop>

SSH key: jenkins-key.pem" <ubuntu@ec2-54-163-2-3.compute-1.amazonaws.com>

Combine Path plus SSH key (Put \ to separate the two)

ssh -i "C:\Users\johnm\OneDrive\Desktop\jenkins-key.pem" <ubuntu@ec2-54-163-2-3.compute-1.amazonaws.com>

```
PS C:\Windows\system32> ssh -i "C:\Users\johnm\OneDrive\Desktop\jenkins-key.pem" ubuntu@ec2-54-163-2-3.compute-1.amazonaws.com
The authenticity of host 'ec2-54-163-2-3.compute-1.amazonaws.com (54.163.2.3)' can't be established.
ED25519 key fingerprint is SHA256:XA/b3bxTYsVYzDWNmF6WH3PCzXVd4d1vtQMktBM2bCM.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-163-2-3.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)
```

Install Git on EC2 Instance:

1. Update the package index:

```
sudo apt update
```

```
ubuntu@ip-172-31-25-49:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Hit:5 https://pkg.jenkins.io/debian-stable binary/ Release
Hit:6 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

2. Install Git:

```
sudo apt install git
```

```
ubuntu@ip-172-31-25-49:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

3. Verify the installation:

```
git --version
```

```
ubuntu@ip-172-31-25-49:~$ git --version
git version 2.43.0
```

Configure Git on the EC2 instance with a GitHub account, follow these steps:

1. Generate a new SSH key pair using the following command:

```
mkdir ~/.ssh
```

```
ssh-keygen -t rsa -b 4096
```

```
ubuntu@ip-172-31-25-49:~$ mkdir ~/.ssh
ssh-keygen -t rsa -b 4096
mkdir: cannot create directory '/home/ubuntu/.ssh': File exists
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
```

2. When prompted for the file path, you can press Enter to accept the default location (/home/ubuntu/.ssh/id_rsa).

3. Copy and paste command below

```
/home/ubuntu/.ssh/id_rsa
```

```
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa): /home/ubuntu/.ssh/id_rsa
Enter passphrase (empty for no passphrase):
Enter passphrase again:
```

4. Enter A unique passphrase for your SSH key twice when prompted(make sure they are the same and securely store it in a safe place for reference purposes)

You will be able to get the output below:

```
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/ubuntu/.ssh/id_rsa  
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:AO+ReMuyAyz6VMgBk8To/l/uVJ/u4zmRA6j/vCphZ9s ubuntu@ip-172-31-25-49  
The key's randomart image is:  
+---[RSA 4096]----+  
| *o . |  
| o+ + . |  
| . . . * . |  
| + o + = . |  
| o = o = S.. . |  
| .o o * o. .+. |  
| . o + =oo oo |  
| o . o+o.E.o. |  
| . ..o+o++=o |  
+---[SHA256]----+
```

5. Use the cat command to display the contents of the public key file:

bash

1. Use the cat command to display the contents of the public key file:

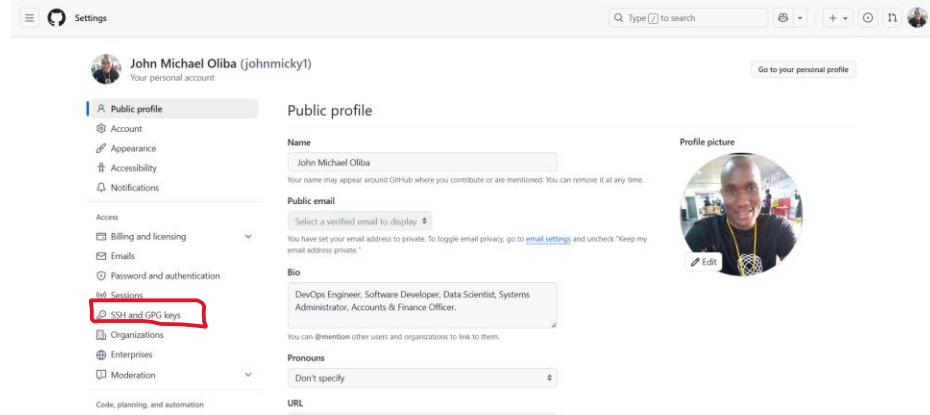
bash

cat ~/.ssh/id_rsa.pub

```
ubuntu@ip-172-31-25-49: $ cat ./~/.ssh/id_rsa.pub
ssh-rsa AAAAB3C123...C1yC2EAAADAOQDIBxUeHP4zBvS1MhpjQzQ6layoBeokS15Z10nWNgOP8yHeeyZMHAgB9cZkFvhbCrGTD5Dw+Omd0lRsKxLafS1svanuM0t0YppwRFzDlRke0zsd9eLvT6+jFHKEFSez+1pmB8e23CwEqZd3X0hbaekh4Q/C1pGeqf87ykh0WEadIEb/Pk5FBp#mPm4y5X50S5L9Jwle1.2518SPxV9rVewS/nemf3xtRr/tvvL7ox7rNrre+CjkW6Cx1ulgWkbMs0JGQo0Erq011Y1HEhmLo5+1pffhxa2C2px2lpiZd1t4Zv5PVGHeL6y1p+knpdrXvTJ7KwN50C8dHmJNLL697107dVly+7t+hayshGPNy+at10kFQd0Q3C9uXbphS3Ca1Qp5y5tRzMeM6L0DF5SpPgdD1g+jAyTz79HuHm0083W3juHw1Fy7WzN7DyGvC0e19j9p2z7nukNrrRrTzJhMaRpyB8f3ZxspRPhMaR9+yxhWle19FQp8QvWeWz6A1KU6CsSmx+C1bzN825J1Rb/HwAp1Rmh7Eyhd1knODNmXJw2pB73s+Tw4J7Gd+c/WtMy/8x7Zz+rz+ERSBVNsPhLm59wDf4Pjg1o-Zn9kph1Yo10/LxbD9+Fu5Lke0= ubuntu@ip-172-31-25-49
```

Add SSH key to GitHub

2. Log in to your GitHub account and go to Settings > SSH and GPG keys.



3. Click New SSH key and paste the public SSH key into the Key field.

The screenshot shows the GitHub Settings interface. At the top, there are navigation links for 'Public profile', 'Account', 'Appearance', and 'Accessibility'. Below these, a search bar contains the placeholder 'Type / to search'. On the right, there are icons for 'Go to your personal profile' and other account settings. The main area is titled 'SSH keys' and contains the sub-titles 'Public keys' and 'Authentication keys'. A green button labeled 'New SSH key' is highlighted with a red box. Below the titles, a note says 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' There is also a link 'Go to your personal profile'.

4. Give the key a label (e.g., "EC2 instance")

Add new SSH Key

Title

EC2 instance

5. Go back to ssh and copy the whole key generated when you typed command (cat ~/ssh/id_rsa.pub)

```
ubuntu@ip-172-31-25-49:~$ cat ~/ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQADIBxEu4PzBy8SiMHpcjQz6LayoBoekS15Z1QnWNgOP8yHeeyZMHAgB9cZKfBvhcBrrGTDN5dwe+OMdOloRSKx
9elVt/6jFHKEQfSeIz+ynPmBe84CZwEqDJzxDhAbKq4/WClpGeqf8Lykw0HEaeDIks/Pb5fpm0m4mPyM50XS6JLw9eL2158XSpv9RevS/nnef3xTmFR/ttVLn
Qo0re6QqI1IYEhqmcLQi+5fyhrksA22Fxp2liTjd4ZI5vPGHElY6ylp+nK+grdXvTJVNKo7SnPC8dhMEUJNL69f701LDYgv+t7hasyhPGNY+atI0KfdgDQqC3
D5Fh5PgPd6ULjcYAztT9chHNwuua8DpN3MwNjfuWSnZYDrdGyQCWoevj09pzo7NukNtRRrqTjhMpRANygi8ZF3xuysPpMRHa9x+yWxhe19FzQ0Pg8VVeQW6zAk0UCs
iknODNOMXJ2WP687J+Tw4x7GdC+/w7Myc/8x7Z7Zz+eRSBVNsVhpPlmS9dWK4F/PgjL0rZn9kZhTpQYIOh0/LXDb9+Fu5LKe9Q== ubuntu@ip-172-31-25-49
```

6.click Add SSH key.

The screenshot shows the 'Add new SSH Key' form. It has a 'Key' input field containing the copied SSH key. The key starts with 'ssh-rsa' and includes a long string of characters. Below the input field is a green 'Add SSH key' button, which is also highlighted with a red box.

Your Output will look like this

The screenshot shows the GitHub SSH keys list. A specific key entry is highlighted with a red box. The key is labeled 'EC2 instance' and has the following details: 'SSH', 'SHA256:AO+ReMuyAyZ6VMgBk8To/1/uVJ/u4zmRA6j/vCphZ9s', 'Added on Oct 30, 2025', and 'Never used — Read/write'. To the right of the key entry is a 'Delete' button.

Step 3: Configure Git

1. Set your GitHub username and email address:

bash

```
git config --global user.name "Your GitHub Username"
git config --global user.email "your_email@example.com"
```

Replace "Your GitHub Username" and "your_email@example.com" with your actual GitHub username and email address.

bash

```
git config --global user.name "johnmicky1"  
git config --global user.email johnmicky1@gmail.com
```

Output:

```
ubuntu@ip-172-31-25-49:~$ git config --global user.name "johnmicky1"  
git config --global user.email "johnmicky1@gmail.com"  
ubuntu@ip-172-31-25-49:~$ git config --global user.name "johnmicky1" && git config --global user.email "johnmicky1@gmail.com"  
ubuntu@ip-172-31-25-49:~$ git config --list  
user.name=johnmicky1  
user.email=johnmicky1@gmail.com
```

Step 4: Test the connection

1. Test the SSH connection to GitHub:

bash

```
ssh -T git@github.com  
ubuntu@ip-172-31-25-49:~$ ssh -T git@github.com  
The authenticity of host 'github.com (140.82.114.4)' can't be established.  
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.  
This key is not known by any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.  
Enter passphrase for key '/home/ubuntu/.ssh/id_rsa':  
Hi johnmicky1! You've successfully authenticated, but GitHub does not provide shell access.
```

If everything is set up correctly, you should see a message indicating that you've successfully authenticated with GitHub.

Setting up JDK(java) and Jenkins:

```
# update  
sudo apt update -y && sudo apt upgrade -y  
User sessions running outdated binaries:  
  ubuntu @ session #5: apt[2009], sshd[1612,1670]  
  ubuntu @ user manager service: systemd[1438]  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-26-70:~$
```

```
# install Java (Jenkins requirement)  
sudo apt install openjdk-17-jdk -y  
ubuntu@ip-172-31-26-70:~$ sudo apt install openjdk-17-jdk -y  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-common at-spi2-core ca-certificates  
  fonts-dejavu-mono gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanistic  
  libatk-wrapper-java-jni libatk1.0-0t64 libatspi2.0-0t64 libavahi-client3 libavahi-common-data  
  libdrm-intel1 libfontconfig1 libgail-common libgail18t64 libgbm1 libgdk-pixbuf-2.0-0 libgdk  
  libgtk2.0-0t64 libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice-dev libice6 libjbig0 lib
```

```
# add Jenkins repo + import key
```

Step 1: Update and Upgrade

```
sudo apt update
```

```
sudo apt upgrade
```

```
ubuntu@ip-172-31-26-70:~$ sudo apt update
sudo apt upgrade
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Reading package lists... Done
Building dependency tree... Done
```

Step 5: Install Jenkins

```
bash
```

```
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
```

```
ubuntu@ip-172-31-26-70:~$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

```
bash
```

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee
/usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

```
ubuntu@ip-172-31-26-70:~$ curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null
```

Then, add the Jenkins repository to your sources list:

```
bash
```

```
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
```

```
ubuntu@ip-172-31-26-70:~$ 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
```

Update your package list and install Jenkins:

```
bash
```

```
sudo apt update
```

```
sudo apt install jenkins
```

```
ubuntu@ip-172-31-26-70:~$ sudo apt update
sudo apt install jenkins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
```

Step 5: start jenkins

```
bash
```

```
sudo systemctl start jenkins
```

```
ubuntu@ip-172-31-26-70:~$ sudo systemctl start jenkins
```

Step 6: Enable Jenkins to Start at Boot

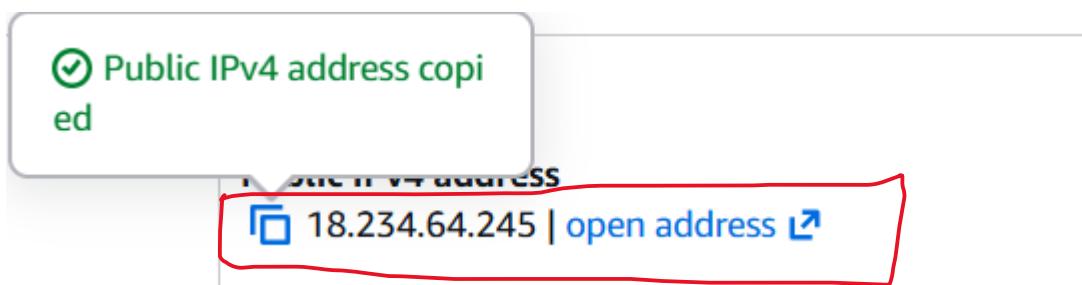
```
bash
```

```
sudo systemctl enable jenkins
```

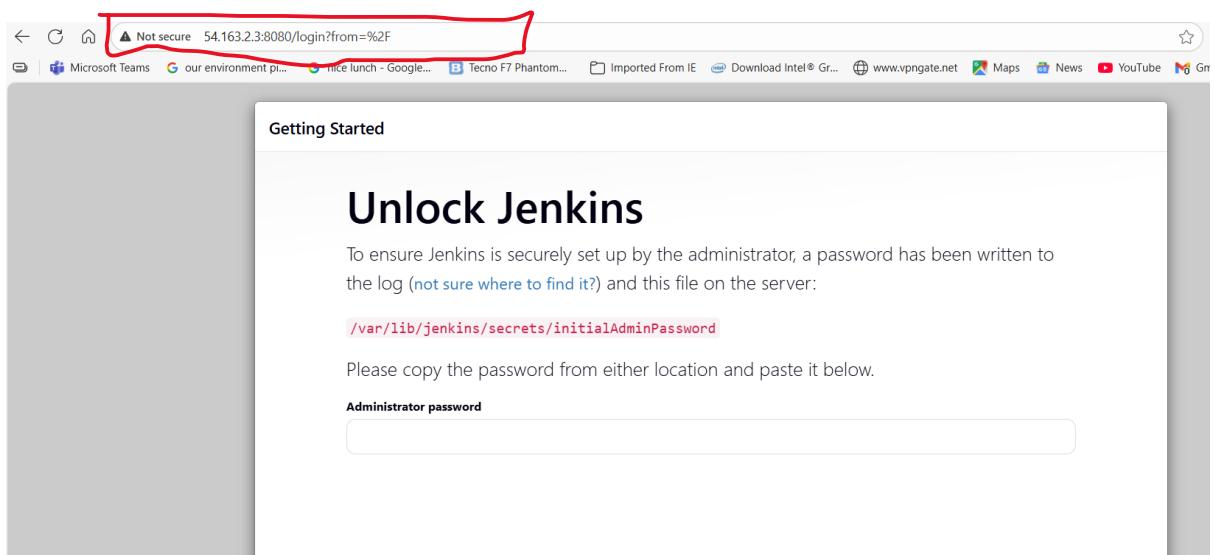
```
ubuntu@ip-172-31-26-70:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
```

5 — Unlock Jenkins & initial setup

1. Open browser: <http://Public IP Address os EC2 Instance:8080> → **Unlock Jenkins**.



e.g <http://18.234.64.245:8080>



2. In SSH session run to get the Administrator Password (copy it and paste it to the Administrator password section under unlock Jenkins)

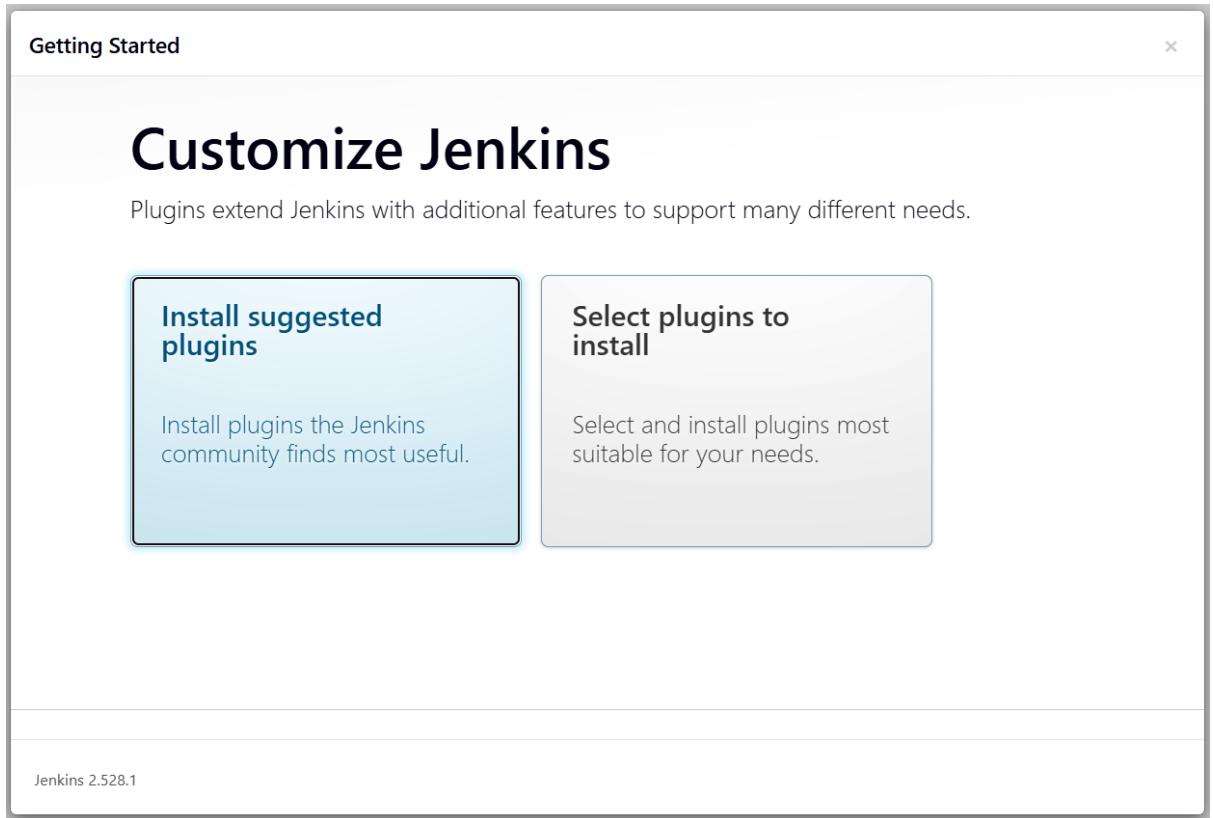
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

```
ubuntu@ip-172-31-26-70:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

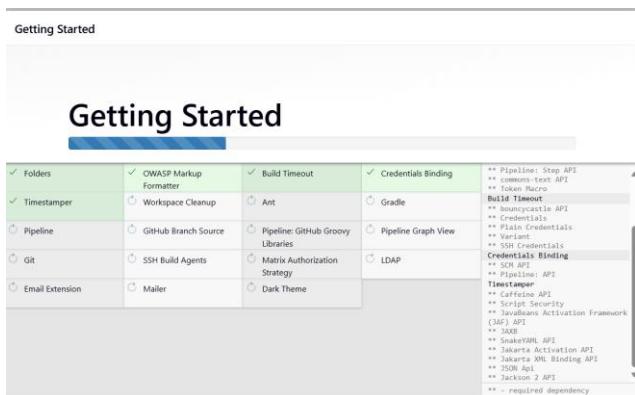
```
4ef3e471230146c0814ca59a08cb7a07
```

```
4ef3e471230146c0814ca59a08cb7a07
```

- Paste the password → install **Suggested plugins** → create admin user. Then click on Install suggested plugins



- Install these recommended plugins (if asked): Git, Pipeline, NodeJS, AWS Steps / AWS Credentials, GitHub Integration.



6 — Add AWS credentials to Jenkins

1. Jenkins → Manage Jenkins → Manage Credentials → Global → Add Credentials.
2. Kind: **AWS Credentials** or (if plugin not available) **Username with password** (use AccessKey as username, SecretKey as password) or **Secret text**

Getting Started

Username
AKIAQWPI74S3Q2MBVSY2

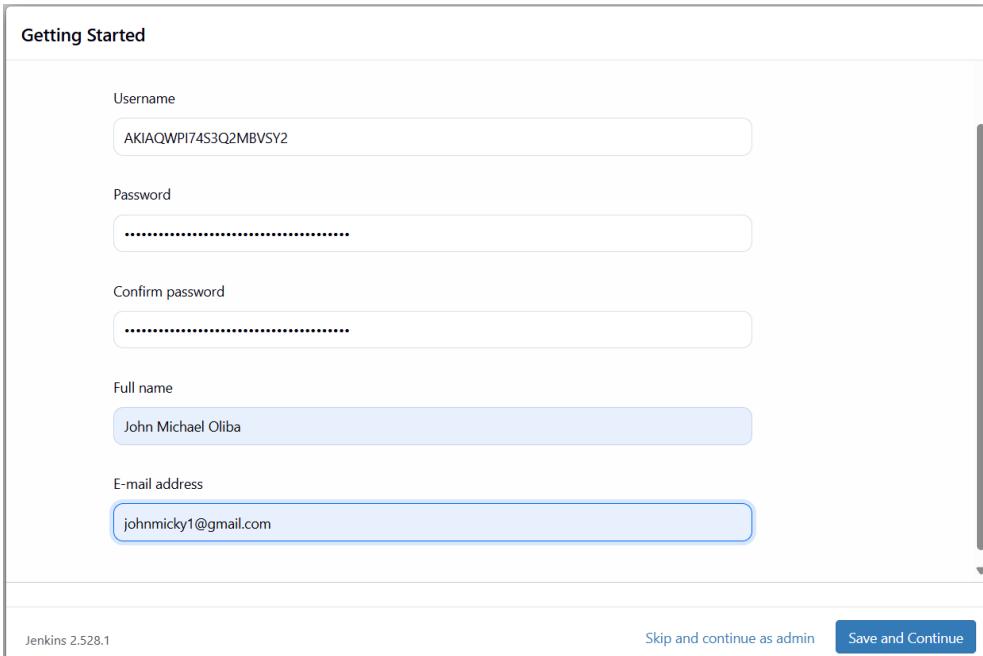
Password
.....

Confirm password
.....

Full name
John Michael Oliba

E-mail address
johnmicky1@gmail.com

Jenkins 2.528.1 Skip and continue as admin Save and Continue



Jenkins URL: <http://18.234.64.245:8080/> (needed to configure GitHub webhooks)

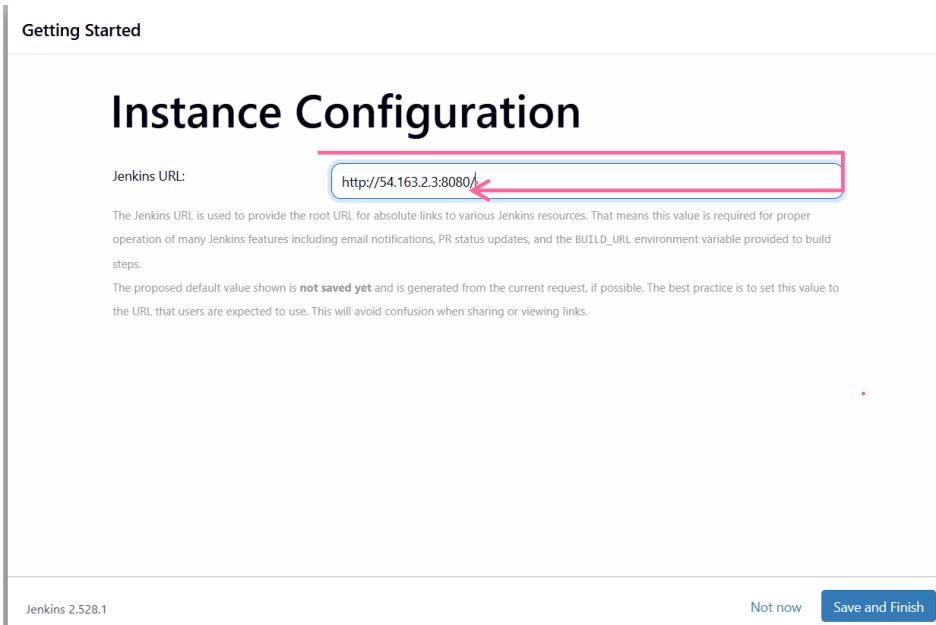
Getting Started

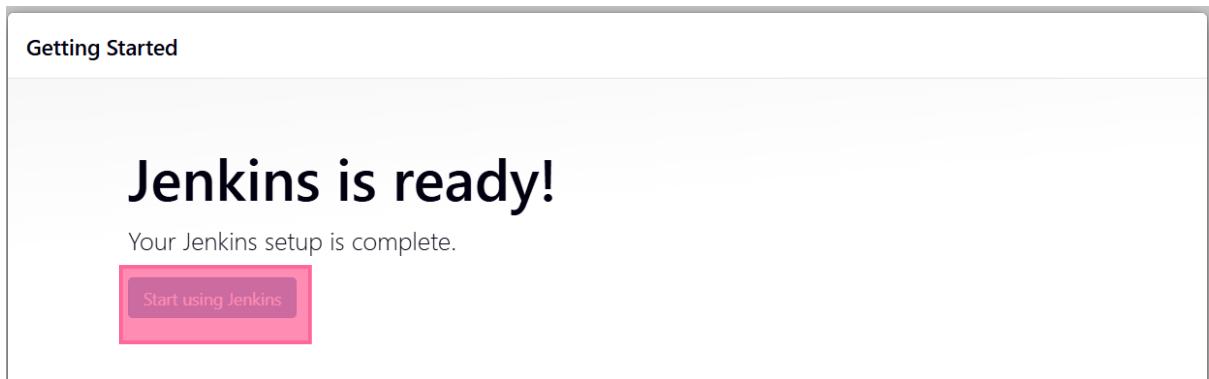
Instance Configuration

Jenkins URL: <http://54.163.2.3:8080/>

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.528.1 Not now Save and Finish





3. Give an ID (e.g. aws-jenkins-creds) — you will refer to this in the Jenkinsfile.

6. Configure Jenkins

- Click on the manage Jenkins (right hand top corner)

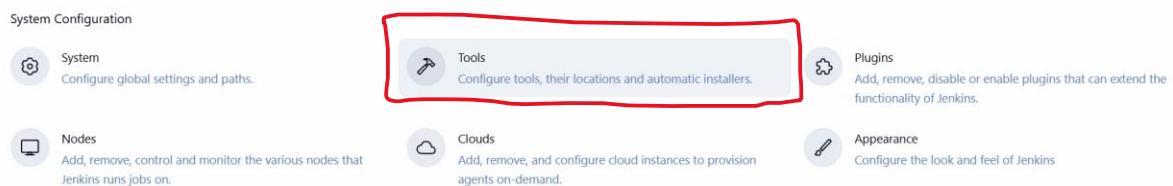


Welcome to Jenkins!

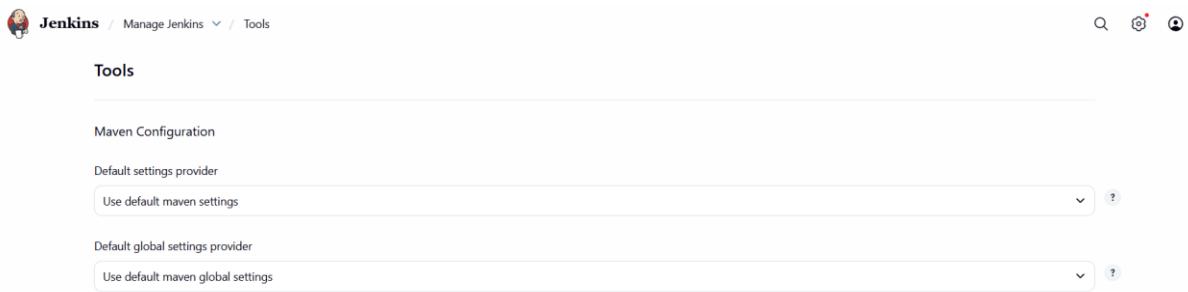
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

[Start building your software project](#)

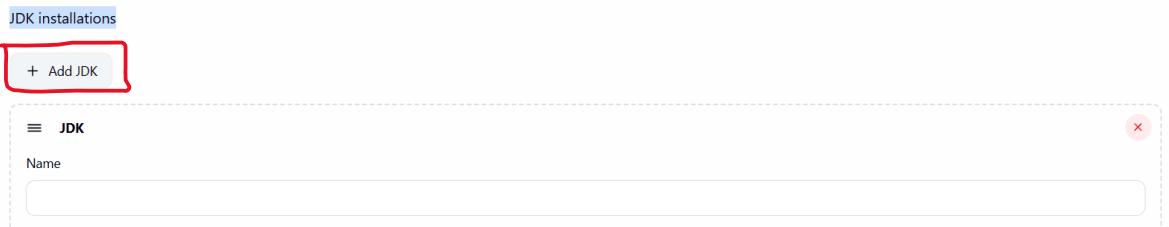
- Click on tools



- Your Display will look like this



- ⊕ Scroll Down to JDK → **Click Add JDK**

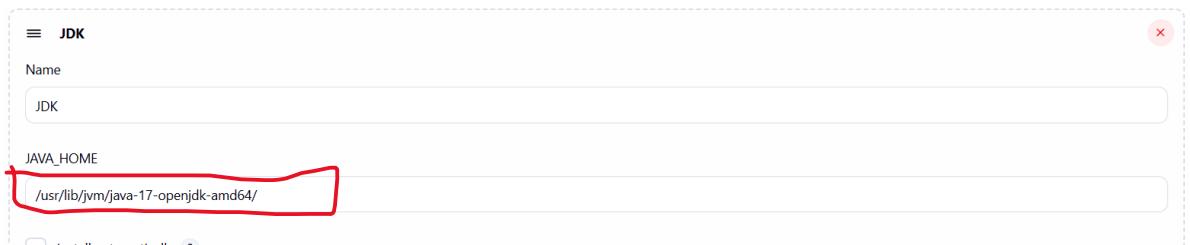


- ⊕ Go back to SSH and type `readlink -f /usr/bin/java | sed 's/bin\java//g'` to find the path

```
ubuntu@ip-172-31-25-49:~$ readlink -f /usr/bin/java | sed 's/bin\java//g'
/usr/lib/jvm/java-17-openjdk-amd64/
```

- ⊕ Come back to Jenkins and add the path under JDK

`/usr/lib/jvm/java-17-openjdk-amd64/`



- ⊕ Scroll Down to Git → **Click Add Git** → Under name type **Git**

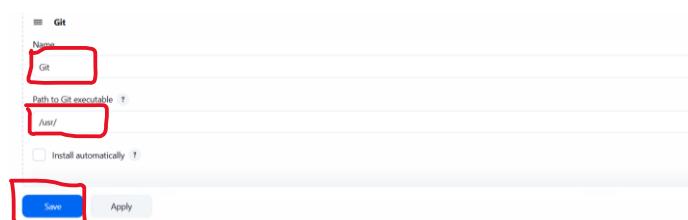


- ⊕ Go back to ssh and type the command below to get the path

`readlink -f $(which git) | sed 's/bin\git//g'`

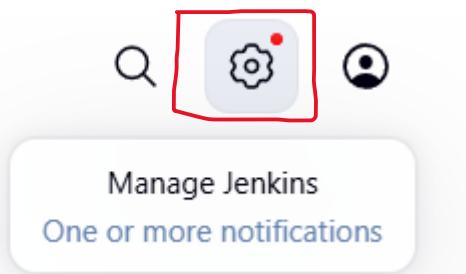
```
ubuntu@ip-172-31-25-49:~$ readlink -f $(which git) | sed 's/bin\git//g'
/usr/
```

1. Got back to the Jenkins Config Dash board and type as displayed in the image below then click on save

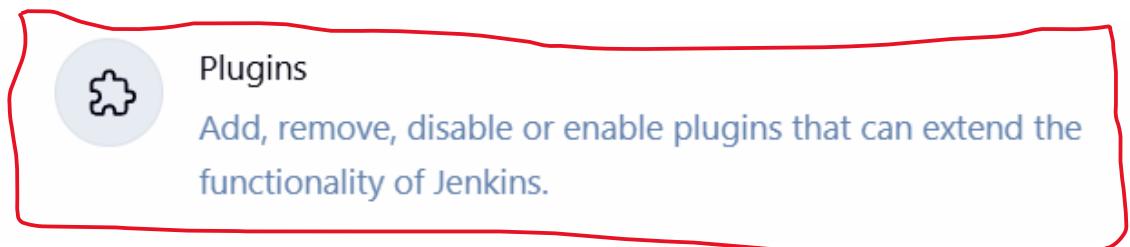


⊕ Install Required Plugins

- Go to **Manage Jenkins**



- → **Plugins**



- → **Available Plugins.**

A screenshot of the Jenkins Available Plugins page. The left sidebar has tabs for 'Updates' (selected), 'Available plugins' (highlighted with a red box), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main content area shows a message: 'No updates available'. A note at the bottom states: 'Disabled rows are already upgraded, awaiting restart. Shaded but selectable rows are in progress or failed.' A red box highlights the 'Available plugins' tab.

- Search and install:

- **Git** – type Git in the search box, select all git plugins (**Click Install**)
Restart Jenkins when the installation is complete

A screenshot of the Jenkins plugin search results for 'Git Plugin'. The search bar at the top contains 'Git Plugin'. The results table has columns: 'Install', 'Name ↓', 'Released', and 'Health'. One result is shown: 'GitPush 34.vd474e0fe7b_ec' by 'git'. A red box highlights the 'Install' button for this plugin. Another red box highlights the 'git' link under the plugin name.

- GitHub Integration – type GitHub Integration in the search box, select all git plugins (Click Install) Restart Jenkins when the installation is complete

The screenshot shows the Jenkins plugin store interface. A search bar at the top contains the text "GitHub Integration". Below it, a table lists the "GitHub Integration 0.7.2" plugin. The plugin details include its name, version (0.7.2), last release date (9 mo 18 days ago), and a green badge indicating 87 reviews. The "Install" button is highlighted with a red box.

Create GitHub Personal Access Token

- In GitHub → Settings → Developer Settings

The screenshot shows the GitHub developer settings page. It includes sections for "Sponsorship log", "ORCID.org", and "Connect your ORCID iD". A note at the bottom states: "All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our [privacy statement](#) to learn more about how we use this information." The "Developer settings" link is highlighted with a red box.

- Personal Access Tokens → Tokens (Classic).

The screenshot shows the GitHub "Personal access tokens (classic)" page. It has tabs for "GitHub Apps", "OAuth Apps", and "Personal access tokens". The "Personal access tokens" tab is highlighted with a red box. Below the tabs, there are two sections: "Fine-grained tokens" and "Tokens (classic)".

- Generate a token with:
 - repo and admin:repo_hook permissions.

The screenshot shows the "Personal access tokens (classic)" generation form. On the left, under "Permissions", the "admin:repo_hook" checkbox is checked and highlighted with a red box. To its right, the description "Full control of repository hooks" is shown. Below the checkboxes, "write:repo_hook" and "read:repo_hook" are also checked, with their descriptions "Write repository hooks" and "Read repository hooks" respectively.

- Copy the token (you'll need it for Jenkins).

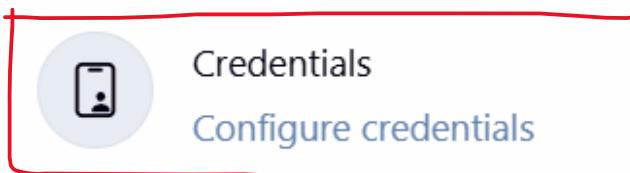
The screenshot shows the "Personal access tokens (classic)" list page. At the top, a "Generate new token" button is highlighted with a red box. Below it, a message says "Make sure to copy your personal access token now. You won't be able to see it again!" followed by a copied token "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". A note at the bottom explains that tokens function like ordinary OAuth access tokens and can be used for Git over HTTPS or API over Basic Authentication.

Token:

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

⊕ Add GitHub Credentials in Jenkins

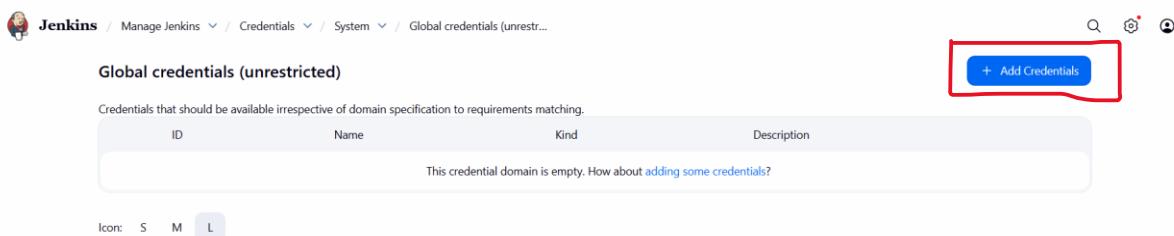
- Go to **Manage Jenkins → Credentials**



- → **System → Global credentials.**

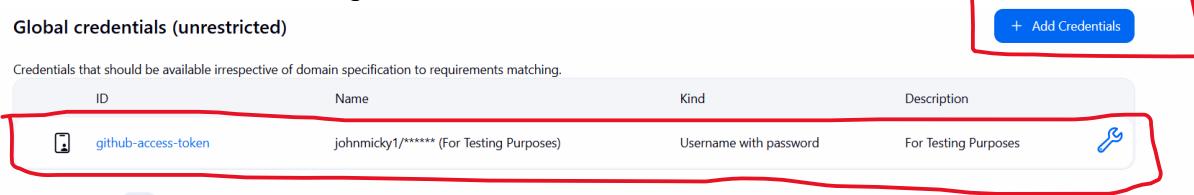


- Click **Add Credentials**:



- Kind: *Username with password*
- Username: your GitHub username
- Password: paste the GitHub token
- ID: e.g., github-access-token

Then Click Create. Your output will look like this below



Create and Configure Webhook in GitHub

⊕ Go to your GitHub repository → Settings → Webhooks → Add Webhook.

- Content type: application/json

- In the **Payload URL**, enter your Jenkins URL followed by /github-webhook/.
Example:
<http://18.234.64.245:8080/manage/credentials/store/system/domain/github-webhook/>

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](#).

● [http://18.234.64.245:8080/manage/... \(push\)](http://18.234.64.245:8080/manage/... (push))

Edit Delete

This hook has never been triggered.

Then Click → **Add webhook**

Add AWS Credentials

+ Add Credentials

Global credentials (unrestricted)

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

ID	Name	Kind	Description
github-access-token	johnmicky1/***** (For Testing Purposes)	Username with password	For Testing Purposes

Icon: S M L

Punch in Access Key, Secret Key and create

ID ?

s3-deployment-credentials

Description ?

credentials for S3 deployment

Access Key ID ?

AKIAQWP174S3ZMQHAX5Z

*

Secret Access Key

.....

*

! Please specify the Secret Access Key

IAM Role Support

Advanced ▾

Create

Step 7: Test CLI on local machine

- Open Powershell and run:

aws configure

```
PS C:\Users\johnm> aws configure
File association not found for extension .py
AWS Access Key ID [*****AX5Z]: AKIAQWPI74S3Q2MBVSY2
AWS Secret Access Key [*****x3jk]: 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
Default region name [us-east-1]: us-east-1
Default output format [json]: json
```

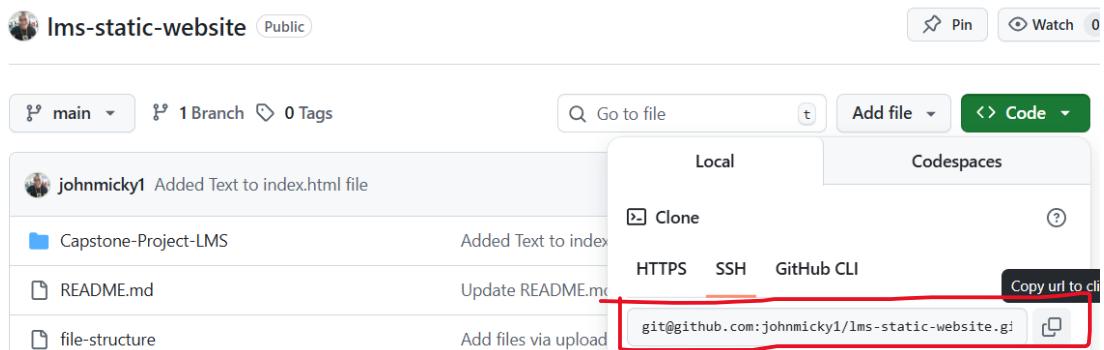
- Ensure your EC2 instance can run AWS CLI commands; Type command below

aws s3 ls

```
PS C:\Users\johnm> aws s3 ls
File association not found for extension .py
2025-10-30 12:28:45 my-static-website-bucket-jmo-2025
```

Step 8: Clone repository

1. Make a change to your GitHub repository (e.g., clone Repository ,add a new file or modify an existing one).
 - Clone Repository
 - Clone GitHub Repository and Deploy to S3



bash

Clone the GitHub repository

git clone [git@github.com:johnmicky1/lms-static-website.git](https://github.com/johnmicky1/lms-static-website.git)

```
PS C:\Users\johnm> cd C:\Users\johnm\OneDrive\Desktop
PS C:\Users\johnm\OneDrive\Desktop> git clone git@github.com:johnmicky1/lms-static-website.git
Cloning into 'lms-static-website'...
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 37 (delta 12), reused 20 (delta 7), pack-reused 0 (from 0)
Receiving objects: 100% (37/37), 28.64 KiB | 132.00 KiB/s, done.
Resolving deltas: 100% (12/12), done.
```

```
# Navigate to the repository directory
```

```
cd lms-static-website
```

```
PS C:\Users\johnm\OneDrive\Desktop> cd lms-static-website
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website>
```

```
# Sync the repository with the S3 bucket
```

```
aws s3 sync . s3://my-static-website-bucket-jmo-2025 --delete
```

```
delete: s3://my-static-website-bucket-jmo-2025/lms-static-website/Capstone-Project-LMS/register.html
delete: s3://my-static-website-bucket-jmo-2025/lms-static-website/README.md
delete: s3://my-static-website-bucket-jmo-2025/lms-static-website/file-structure
upload: .git\HEAD to s3://my-static-website-bucket-jmo-2025/.git/HEAD
upload: .git\hooks\sendemail-validate.sample to s3://my-static-website-bucket-jmo-2025/.git/hooks/sendemail-validate.sample
upload: .git\hooks\push-to-checkout.sample to s3://my-static-website-bucket-jmo-2025/.git/hooks/push-to-checkout.sample
upload: .git\hooks\commit-msg.sample to s3://my-static-website-bucket-jmo-2025/.git/hooks/commit-msg.sample
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> |
```

```
# Check if the website returns a 200 status code. if [ $? -eq 0 ]; then
```

```
echo "Website is up and running"
```

```
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> echo "Website is up and running"
Website is up and running
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website>
```

```
# List the contents of the S3 bucket
```

```
aws s3 ls s3://my-static-website-bucket-jmo-2025
```

```
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> aws s3 ls s3://my-static-website-bucket-jmo-2025
File association not found for extension .py
      PRE .git/
      PRE Capstone-Project-LMS/
2025-10-31 20:16:29      4477 README.md
2025-10-31 20:16:29      411 file-structure
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website>
```

```
# Check if the bucket contains the expected files
```

```
if aws s3 ls s3://your-bucket-name | grep -q "index.html"; then
```

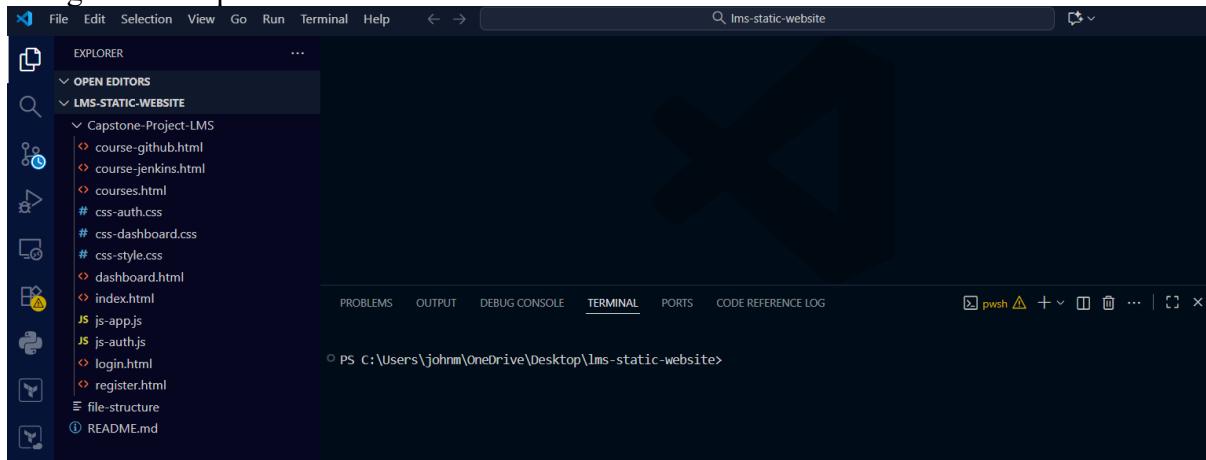
```
echo "Index.html file found in S3 bucket"
```

```
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> echo "Index.html file found in S3 bucket"
Index.html file found in S3 bucket
```

Open Repository on Vs Code or any other code Editor
Code .

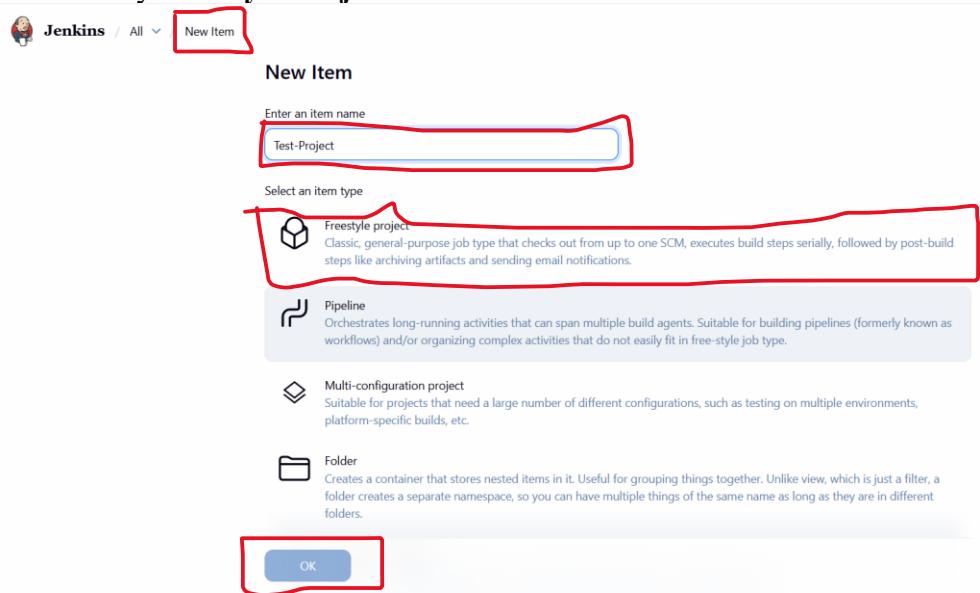
```
Windows PowerShell
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> code .
```

You get this Output

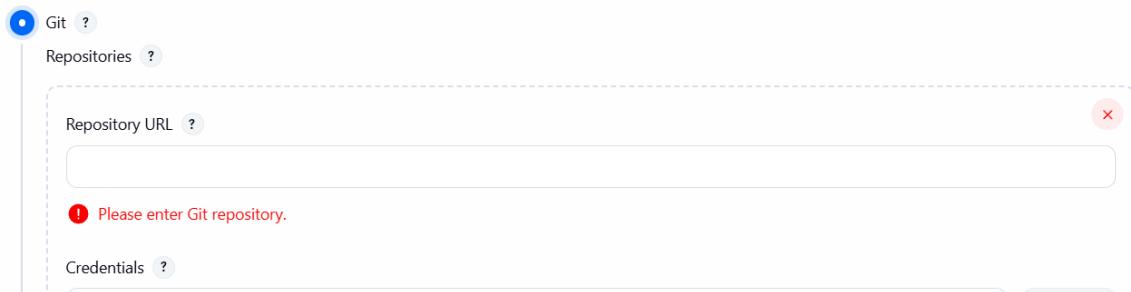


💻 Step 9: Test the Pipeline

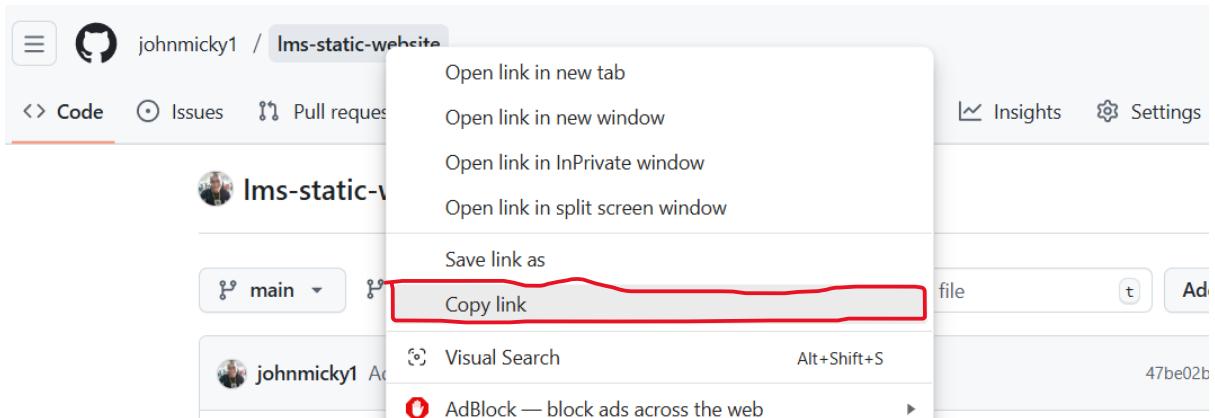
- ✓ Log in to your Jenkins instance and click on **New Item** then type **Test-Project** followed by **Freestyle Project** then **OK**



- ✓ Configure the GitHub repository
- ✓ In the project configuration, scroll down to the Source Code Management section.
- ✓ Select Git as the source code management system.



- ✓ Enter the URL of your GitHub repository (e.g., <https://github.com/johnmicky1/lms-static-website.git>).



Repository URL ? <https://github.com/johnmicky1/lms-static-website.git>

Credentials ?

Branches to build ?

Branch Specifier (blank for 'any') ? */*

Configure the build trigger

- ✓ Scroll down to the Build Triggers section.
- ✓ Check GitHub hook trigger for GITScm polling.

GitHub hook trigger for GITScm polling ?

Poll SCM ?

- ✓ Save and build the project

+ Add post-build action

Save

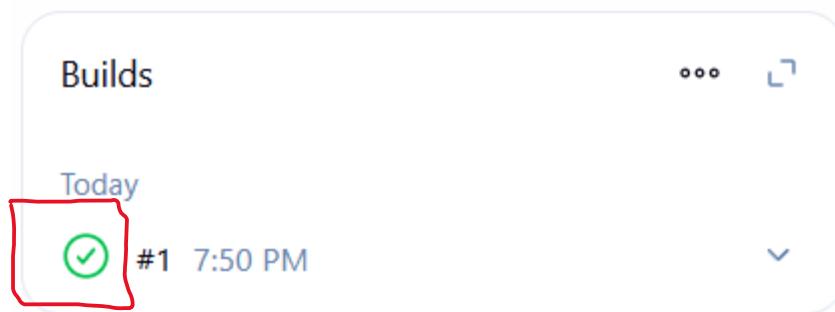
Apply

- ✓ Click Save to save the project configuration.
- ✓ Click Build Now to trigger a build manually.

Build Now

Configure

Once the build is successful, a green tick like in the image below will be displayed.



When you go to console output the script will have a green tick on top and the bottom will have success.

A screenshot of the Jenkins 'Console Output' page for build '#1'. The left sidebar shows navigation links: Status, Changes, Console Output (which is selected and highlighted in grey), Edit Build Information, Delete build '#1', Timings, and Git Build Data. The main content area displays the build logs:

```
Started by user John Michael Oliba
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Test-Project
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
  Cloning repository https://github.com/johnmicky1/lms-static-website.git
    > git init /var/lib/jenkins/workspace/Test-Project # timeout=10
Fetching upstream changes from https://github.com/johnmicky1/lms-static-website.git
  > git --version # timeout=10
  > git -version # 'git version 2.43.0'
  > git fetch --tags --force --progress -- https://github.com/johnmicky1/lms-static-website.git +refs/heads/*:refs/remotes/origin/*
timeout=10
  > git config remote.origin.url https://github.com/johnmicky1/lms-static-website.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/main^{commit} # timeout=10
Checking out Revision 47be02b90bc53b376d4ff0759f2007ff038dd32b (refs/remotes/origin/main)
  > git config core.sparsecheckout # timeout=10
  > git checkout -f 47be02b90bc53b376d4ff0759f2007ff038dd32b # timeout=10
Commit message: "Added Text to index.html file"
First time build. Skipping changelog.
Finished: SUCCESS
```

At the top of the log, there is a green checkmark icon followed by the word 'Console Output'.

Make Changes on the **index.html** file. (Changing the title from DevOps LMS – Reasons to Master Github & Jenkins to.....)

A screenshot of a browser's developer tools, specifically the 'Elements' tab. It shows the HTML code for the page, with the title element highlighted:

```
<title>DevOps LMS - Reasons to Master GitHub & Jenkins</title>
```

The title text has been changed from 'DevOps LMS – Reasons to Master Github & Jenkins' to 'DevOps LMS - Master GitHub & Jenkins'.

DevOps LMS – Master Github & Jenkins

A screenshot of a browser's developer tools, specifically the 'Elements' tab. It shows the HTML code for the page, with the title element highlighted:

```
<title>DevOps LMS - Master GitHub & Jenkins</title>
```

The title text has been changed from 'DevOps LMS – Reasons to Master Github & Jenkins' to 'DevOps LMS - Master GitHub & Jenkins'.

Commit and push the changes to GitHub.

```
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website>
● PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> git add .
● PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> git commit -m "Changed Title of index.html file"
[main 3f76500] Changed Title of index.html file
 1 file changed, 1 insertion(+), 1 deletion(-)
● PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
● PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 361 bytes | 120.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:johnmicky1/lms-static-website.git
  47be02b..3f76500 main -> main
○ PS C:\Users\johnm\OneDrive\Desktop\lms-static-website>
```

Verify the build trigger

- ✓ Make a change to your GitHub repository (e.g., add a new file or modify an existing one).
- ✓ Commit and push the changes to GitHub.
- ✓ Go back to your Jenkins project and check the build history. You should see a new build has been triggered automatically, as indicated by a new build number in the "Build History" section.

New build triggered successfully, confirming the connection between GitHub and Jenkins is working correctly.

The screenshot shows the Jenkins interface for a project named 'Test-Project'. On the left, there's a sidebar with links like Status, Changes, Console Output (which is selected and highlighted in grey), Edit Build Information, Delete build #2, Polling Log, Timings, Git Build Data, and Previous Build. The main area is titled 'Console Output' and contains the following log output:

```
Started by GitHub push by johnmicky1
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Test-Project
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Test-Project/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/johnmicky1/lms-static-website.git # timeout=10
Fetching upstream changes from https://github.com/johnmicky1/lms-static-website.git
> git --version # timeout=10
> git --version # 'git version 2.43.0'
> git fetch --tags --force --progress -- https://github.com/johnmicky1/lms-static-website.git +refs/heads/*:refs/remotes/origin/*
timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 3f7650093ac61c092136ae1020793680b624dd79 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 3f7650093ac61c092136ae1020793680b624dd79 # timeout=10
Commit message: "Changed Title of index.html file"
> git rev-list --no-walk 47be02b99bc53b376d4ff0759f2007ff03add32b # timeout=10
Finished: SUCCESS
```

Clean up

To avoid incurring further costs, make sure to delete all resources created during this process, including:

- EC2 Instance: Terminate the EC2 instance to stop incurring costs.
- S3 Bucket Files: Delete all files stored in the S3 bucket.
- S3 Bucket: Delete the S3 bucket itself.

By deleting these resources, you can ensure that you are no longer incurring costs associated with them.

Author Information

Prepared by: John Michael Oliba

Passionate about;

- DevOps
- Systems Engineering
- Administration
- Graphics Design
- Accounts & Finance

Date: November 01, 2025

Version: 1.0

GitHub Link: <https://github.com/johnmicky1/lms-static-website>