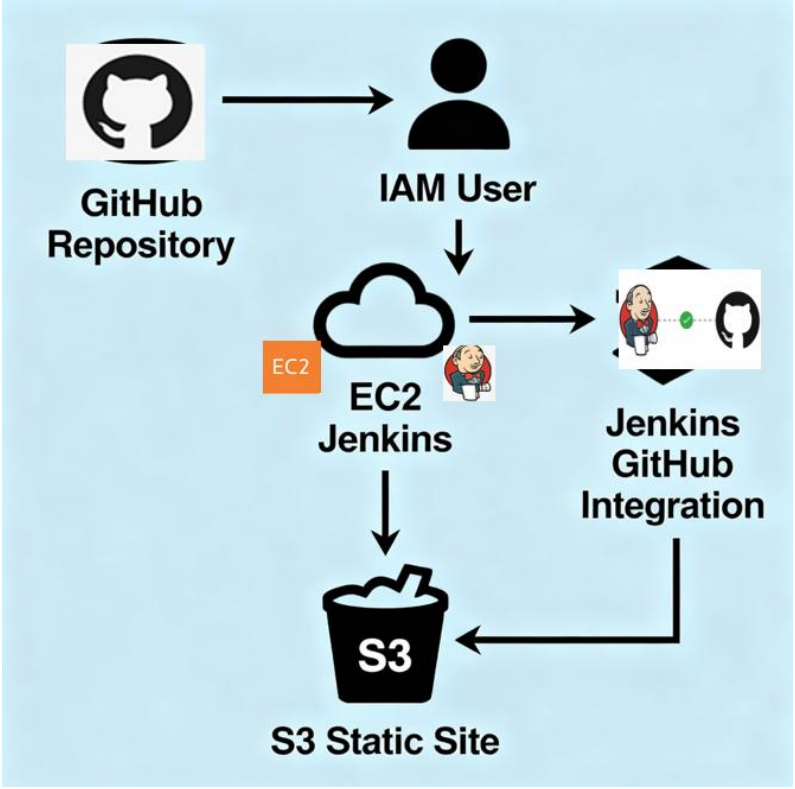
CI/CD Pipeline Setup: IAM, EC2, S3, GitHub, and Jenkins Integration

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-SlKuRsZcRBebm1SvWB9YRA?0=i*

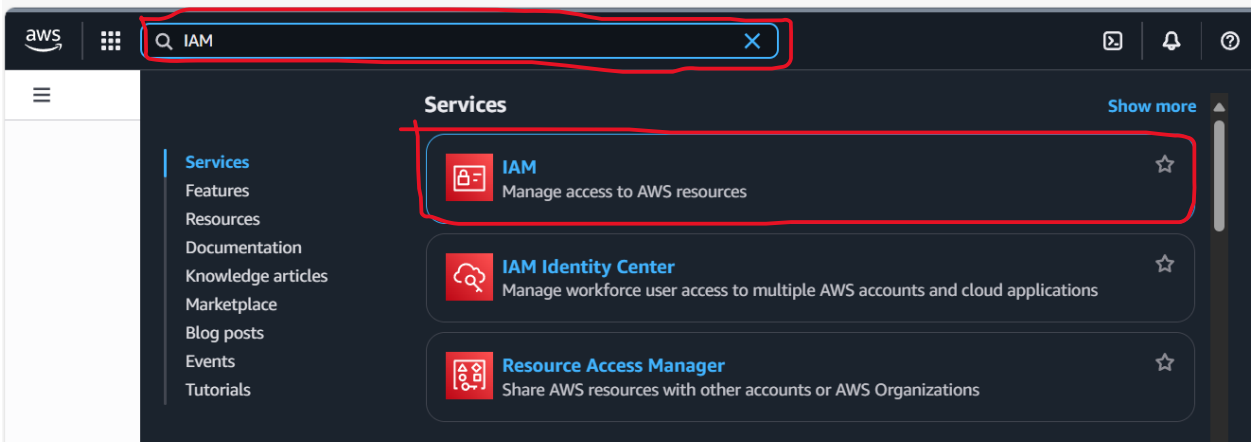
1. AWS IAM Configuration

1.1 Create an IAM User or Role for Jenkins

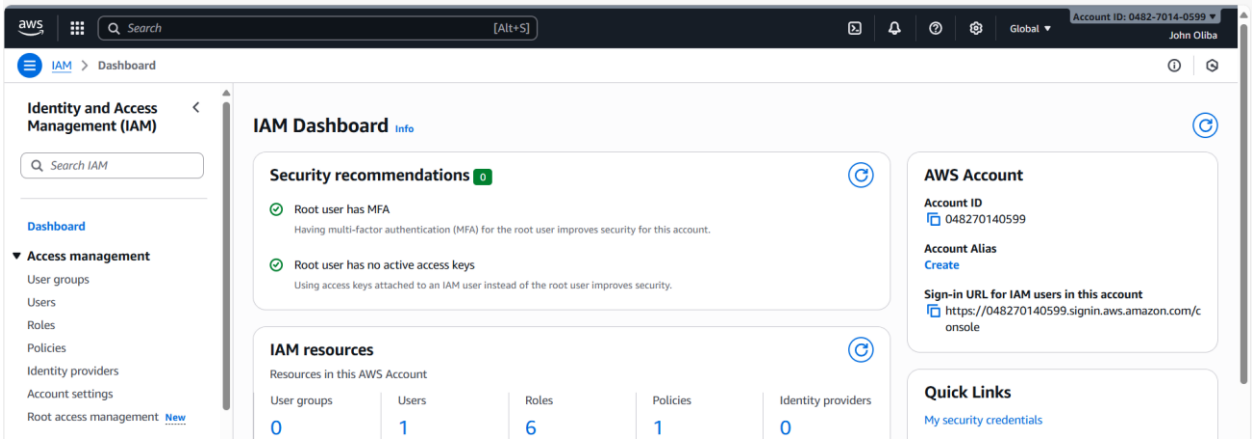
- **Purpose:** Grant permissions for EC2, S3, and CloudWatch integration.
- **Required Policies:**
  - AmazonS3FullAccess
  - AmazonEC2FullAccess
  - CloudWatchLogsFullAccess
  - *(Optional for full control)* AdministratorAccess

Create Access Key and Secret Access Key

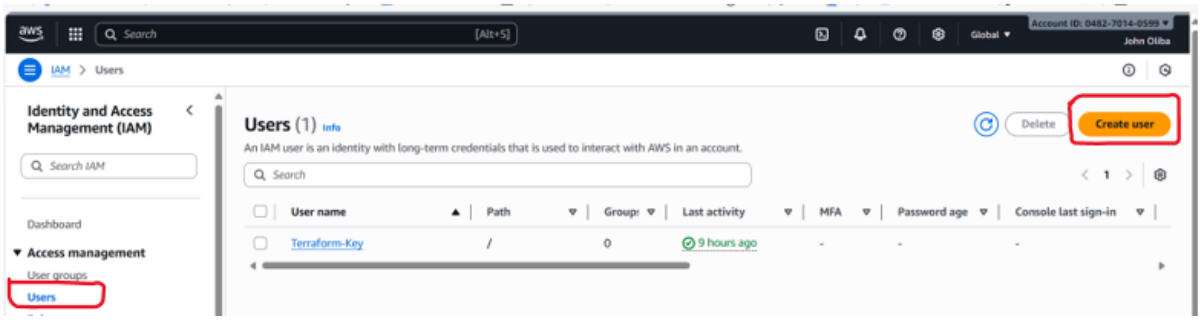
a) Log in to the AWS Management Console and type IAM in the search box



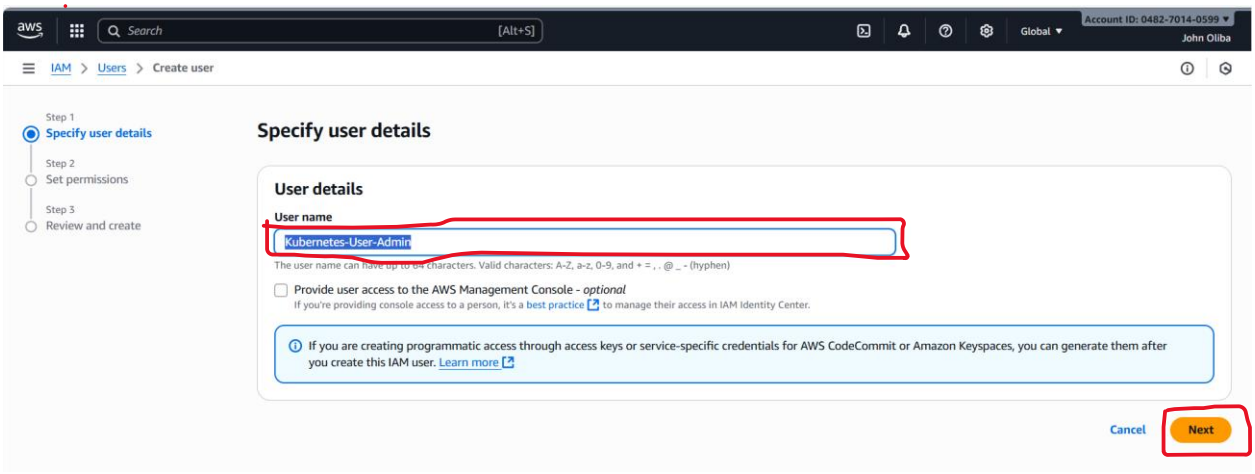
b) Navigate to the IAM (Identity and Access Management) dashboard.



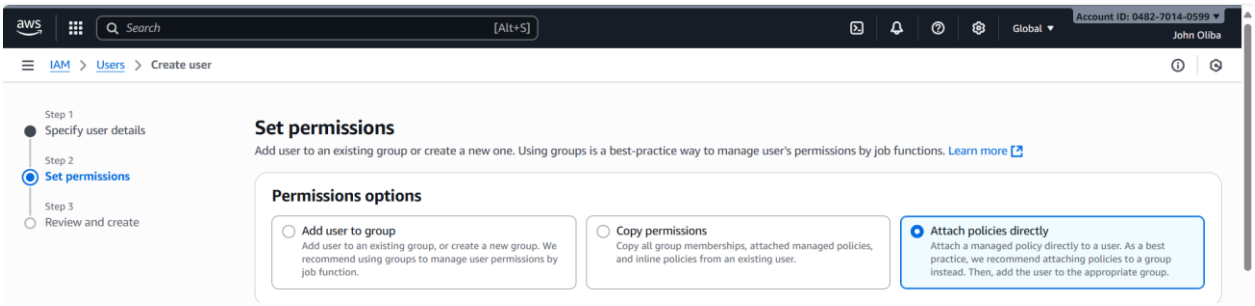
c) In the left sidebar, select Users and click on create user



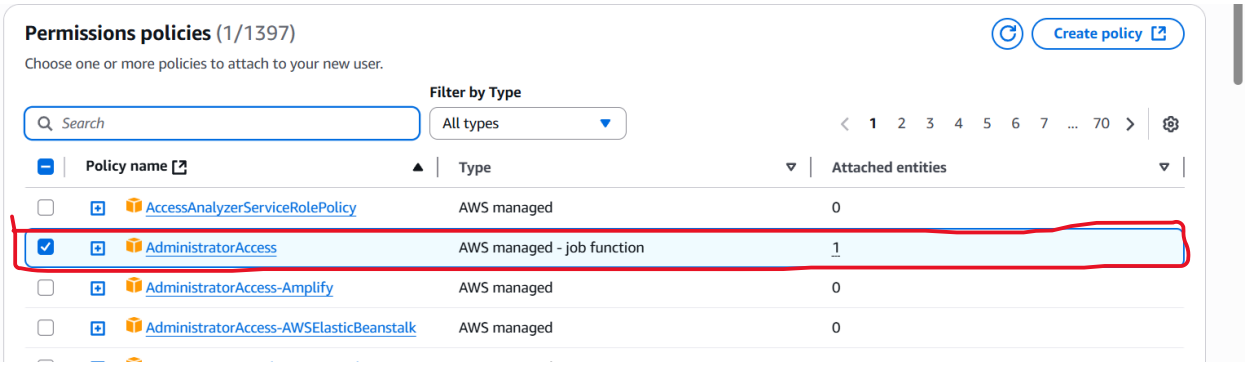
d) Specify User details by giving a user name and click next (e.g Jenkins-EC2)



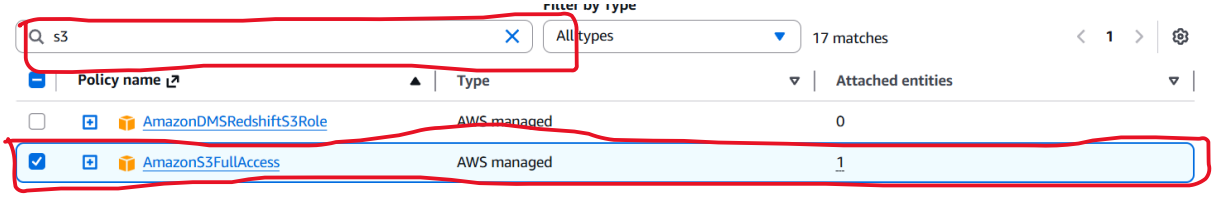
e) To set permissions; click on attach policies directly and scroll down



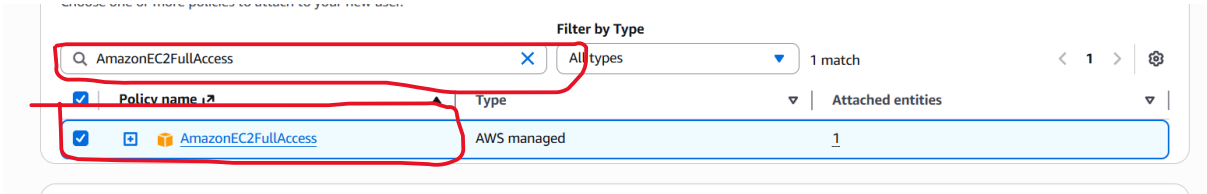
f) Select **AdministratorAccess**



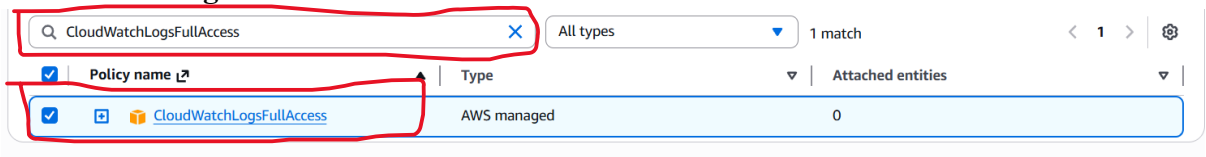
g) Type **S3** in the search box and select **AmazonS3FullAcess**



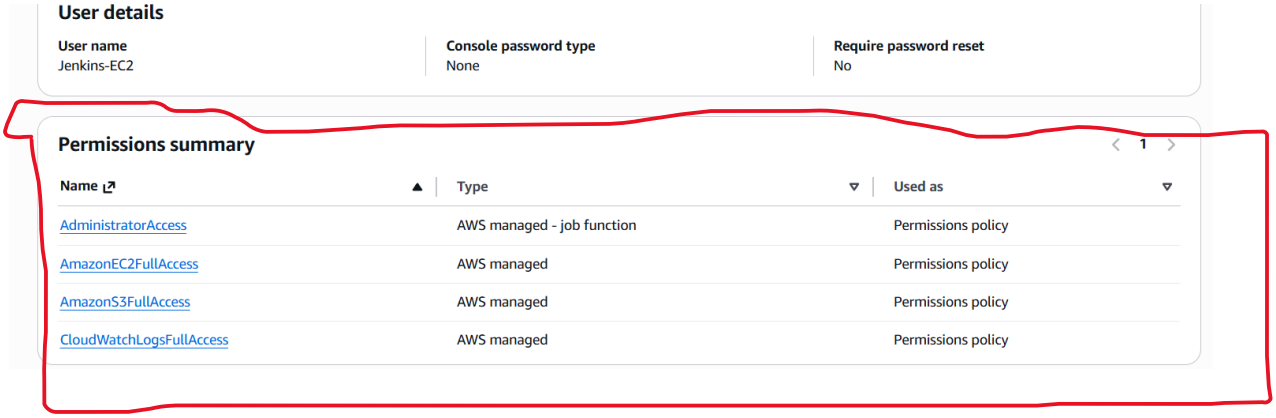
h). Type **AmazonEC2FullAccess** in the search box and select **AmazonEC2FullAccess**



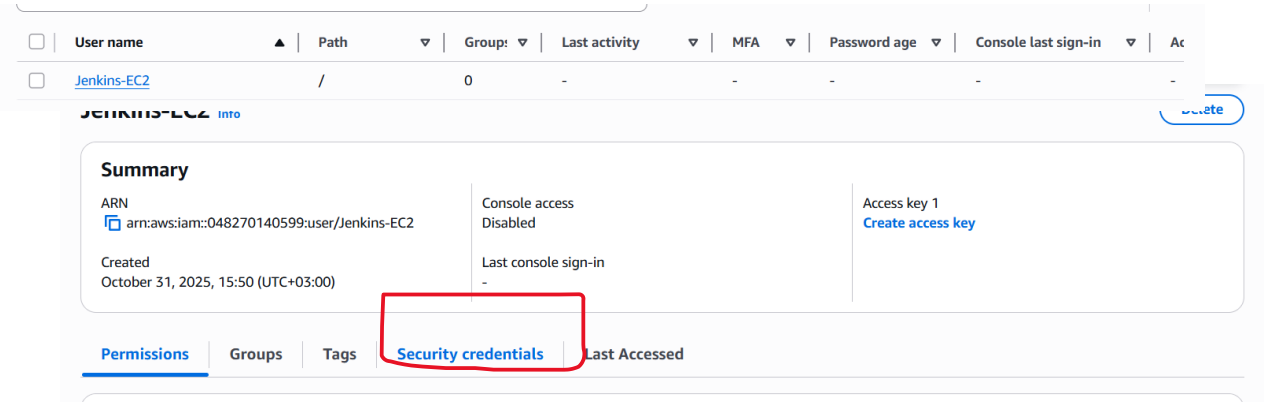
i) Type **CloudWatchLogsFullAccess** in the search box and select **CloudWatchLogsFullAccess** then click **Next**



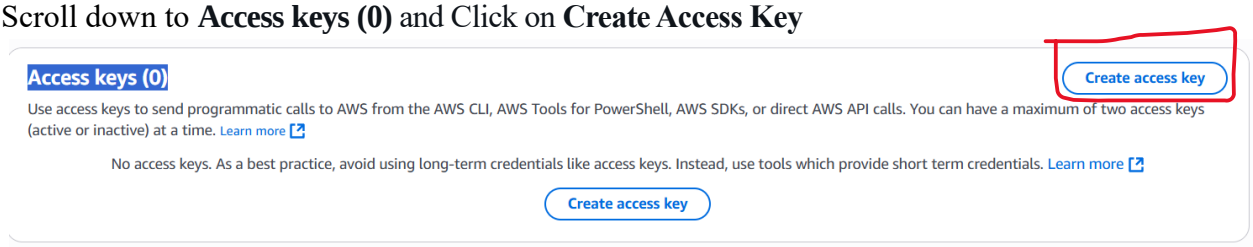
j) You should be able to see a summary of the permissions you have created then click on **Create User**



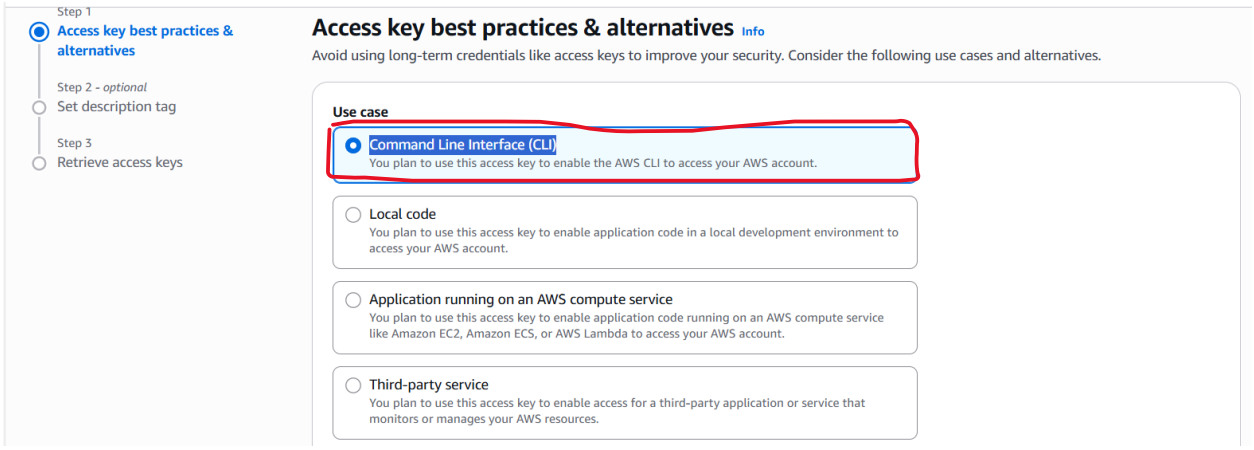
k) The User has been created

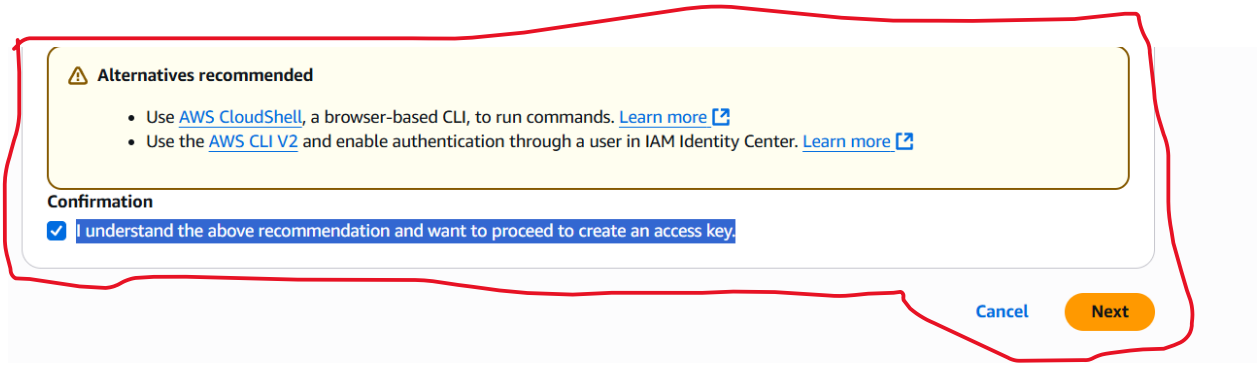


m) Scroll down to **Access keys (0)** and Click on **Create Access Key**

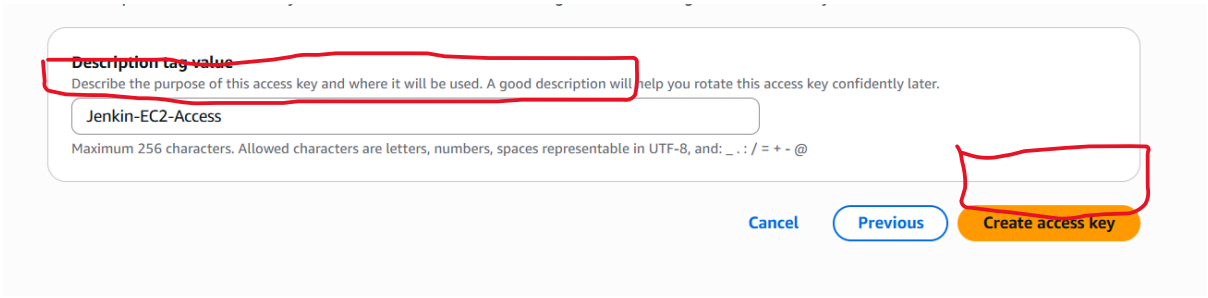


n) Select **Command Line Interface (CLI)** Scroll Down and Click Check box having **I understand the above recommendation and want to proceed to create an access key.** Then Click **Next**



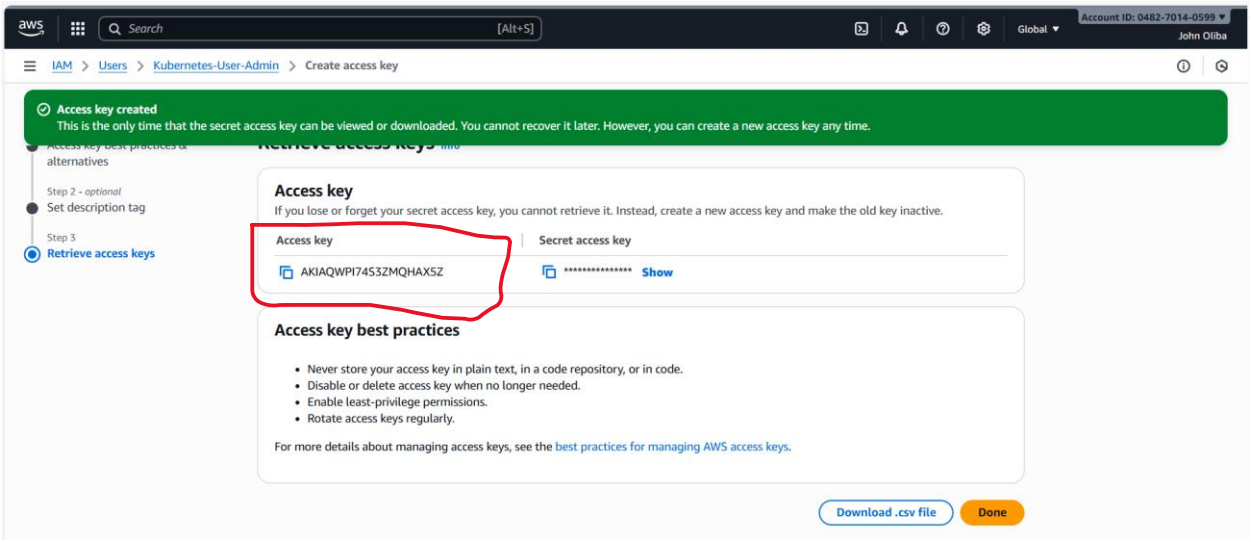


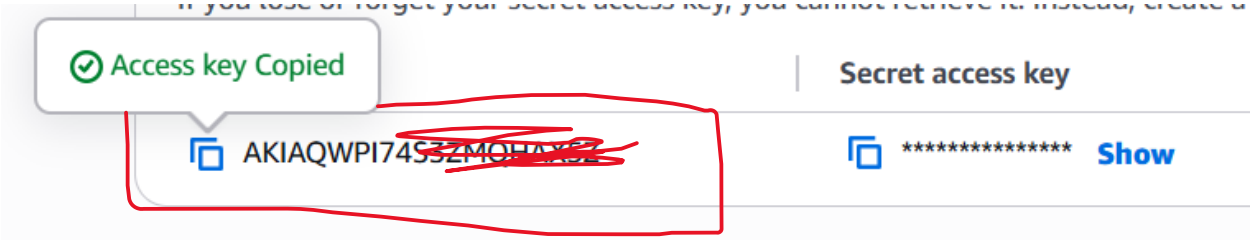
o) Set description tag – optional (Give it a name e.g Jenkin-EC2-Access) and Click on **Create Access Key**



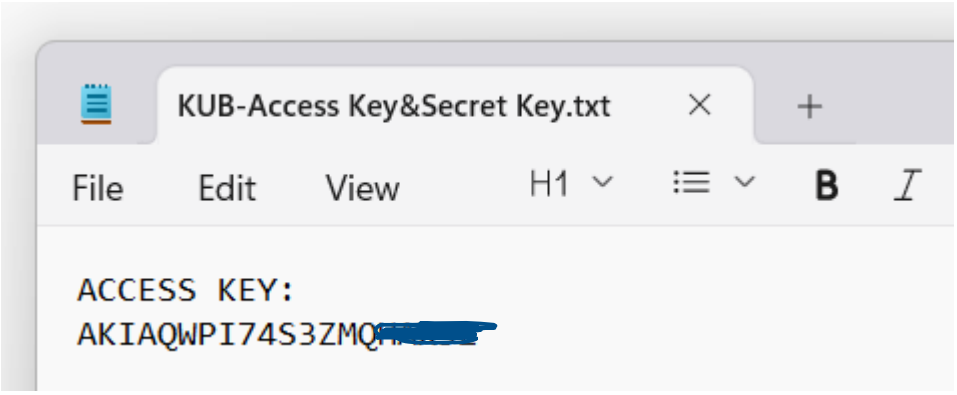
p) Save the Access Key in a sure location on your local machine because it will be

necessary for Accessing the AWS CLI using your local machine.

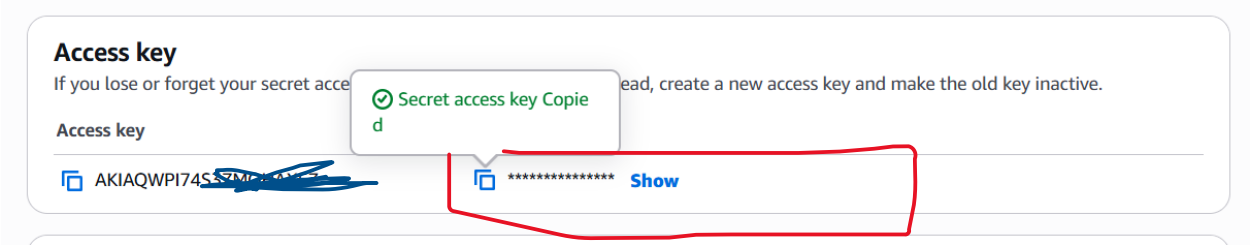


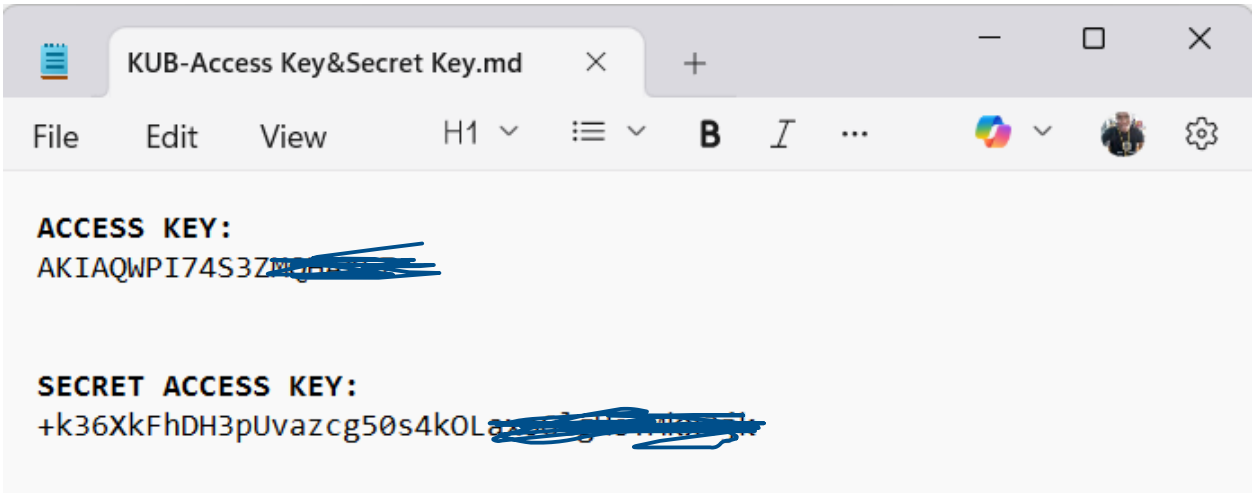


Open Note Pad, Type **ACCESS KEY** and paste the Access Key as illustrated in the Image below:

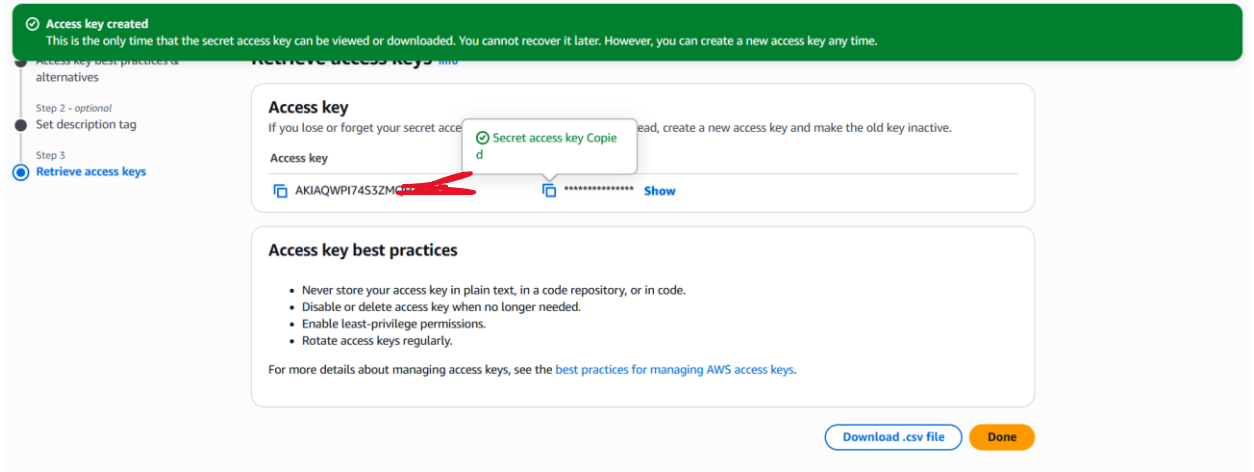


q) Type on the same notepad page **SECRET ACCESS KEY** then copy and paste it as seen below:





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



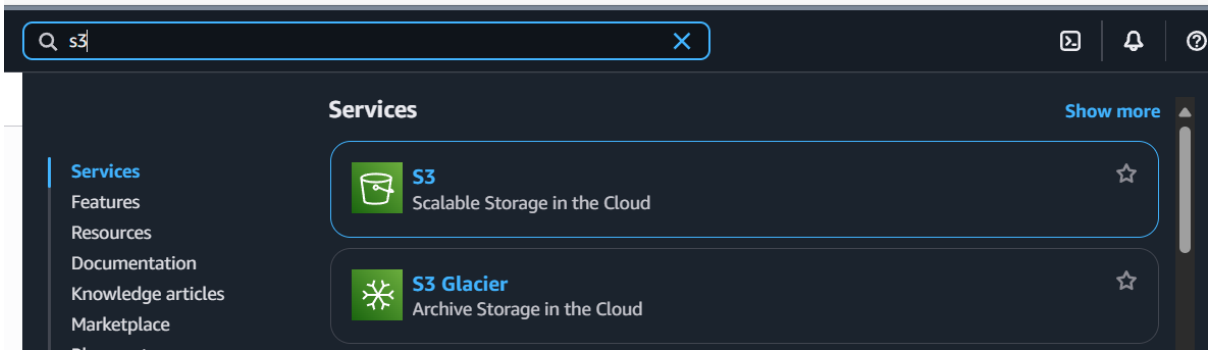
A C.S.V file will be download, make sure it is saved in a secure location

---

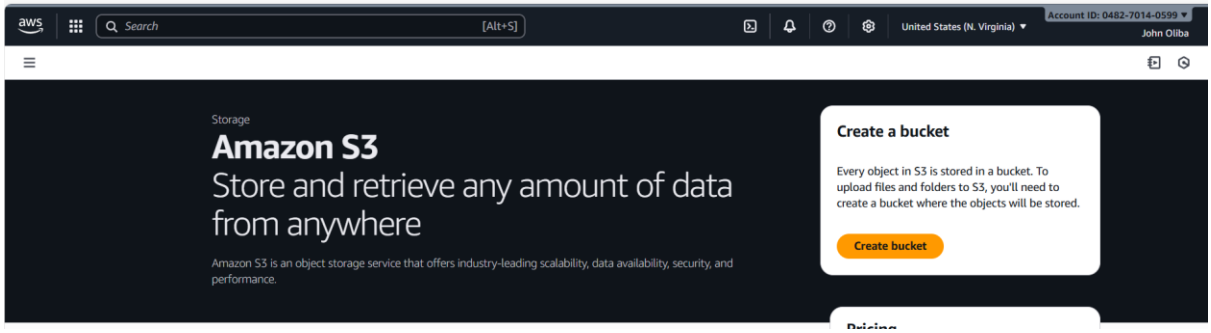
💻 Step 2: Set Up AWS S3 Bucket

2.1. Create an S3 Bucket

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

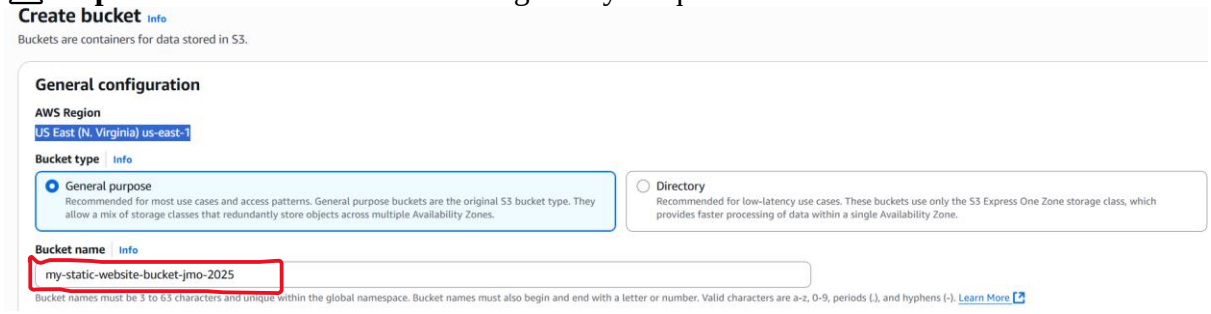


❖ Click on **S3** → Click **Create bucket**.



❖ 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**).**

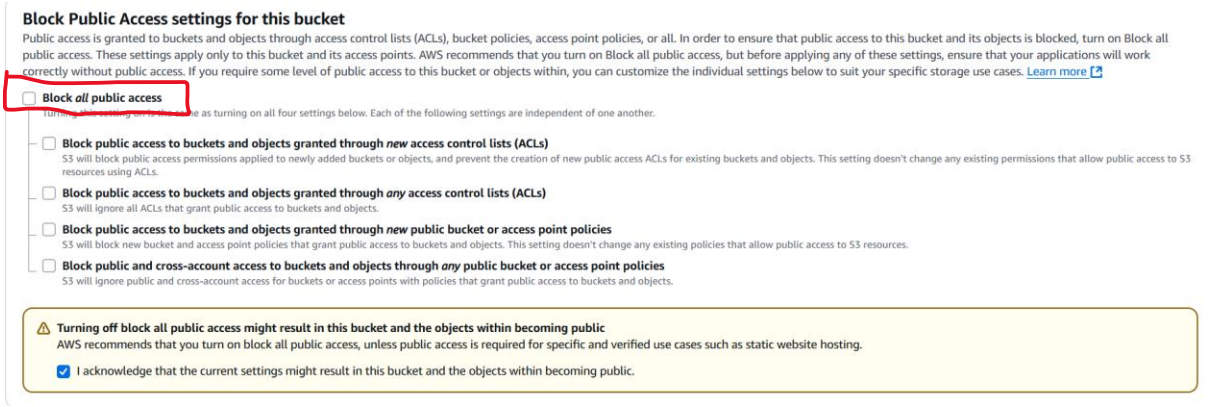
⚠ **Important:** Bucket names must be globally unique.



❖ Configure Bucket Settings for Static Hosting

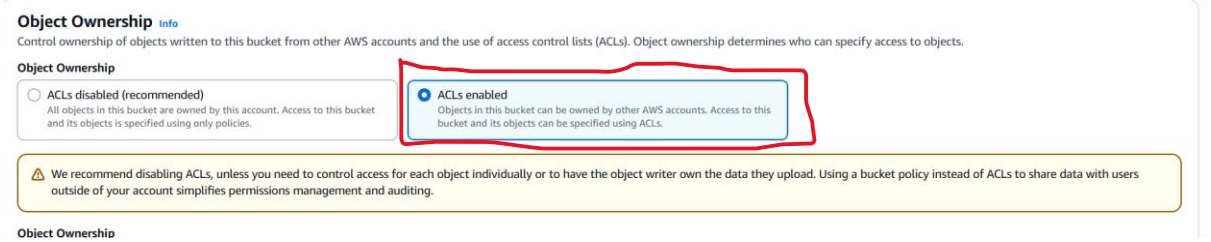
a. To Disable, scroll down "Block Public Access"

1. In the "Block Public Access" section, uncheck:
   - ☑ Block *all* public access
2. Confirm acknowledgment that your bucket will be public.

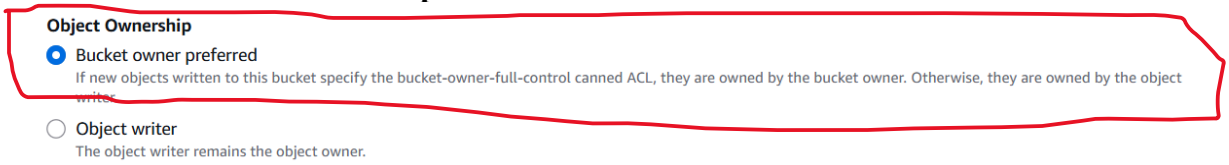


❖ Enable Object Ownership

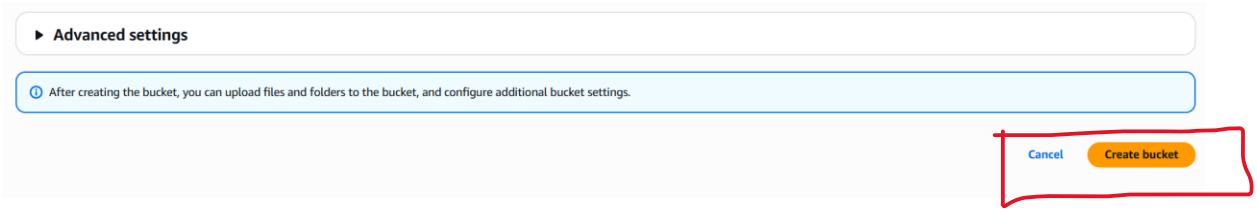
b. Under **Object Ownership**, select **ACLs enabled**.



❖ Choose **Bucket owner preferred**.

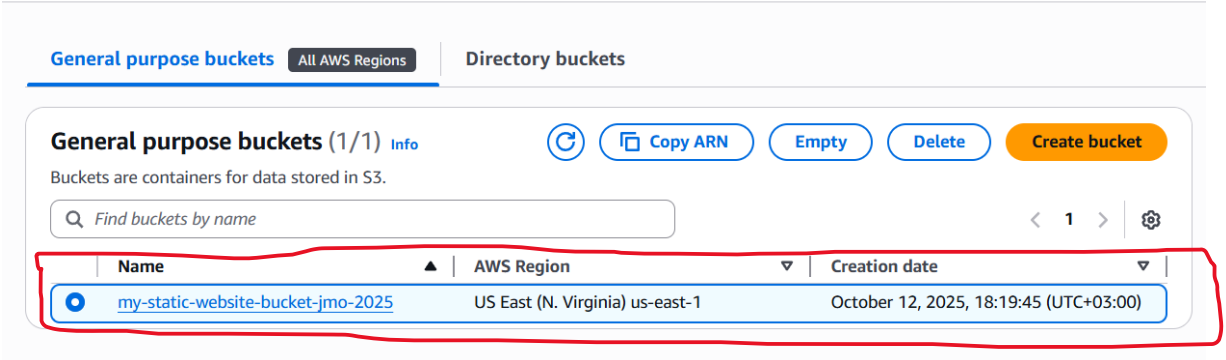


1. Click on Create Bucket.

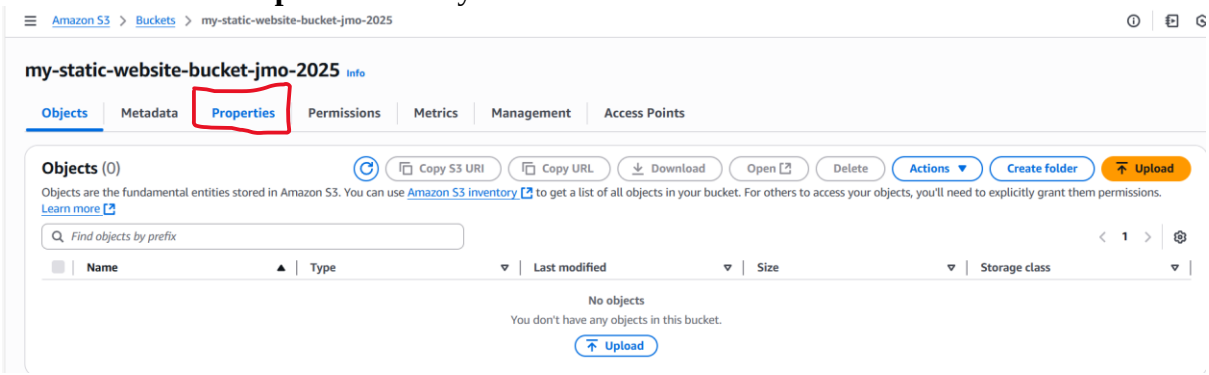


❖ Enable Static Website Hosting

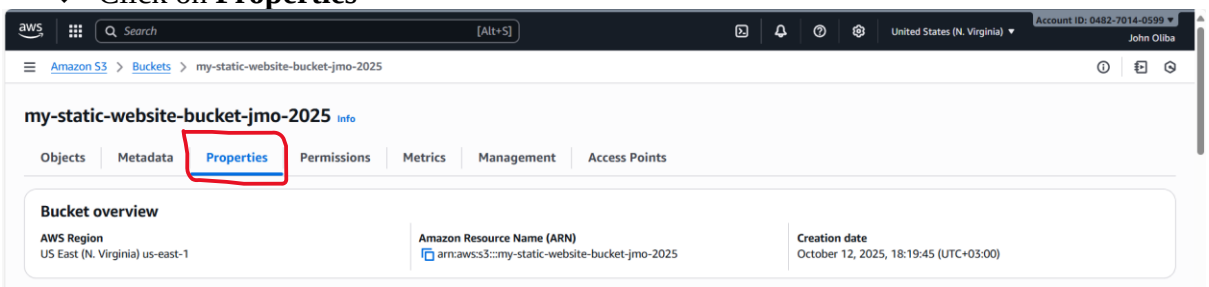
C. Click on the Bucket created



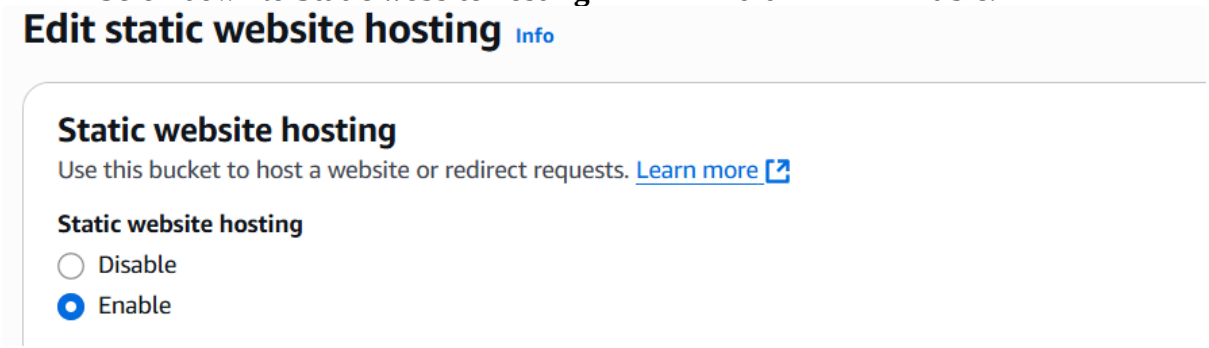
❖ Go to the **Properties** tab of your bucket.



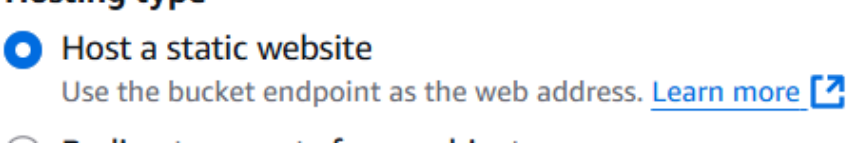
❖ Click on **Properties**



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



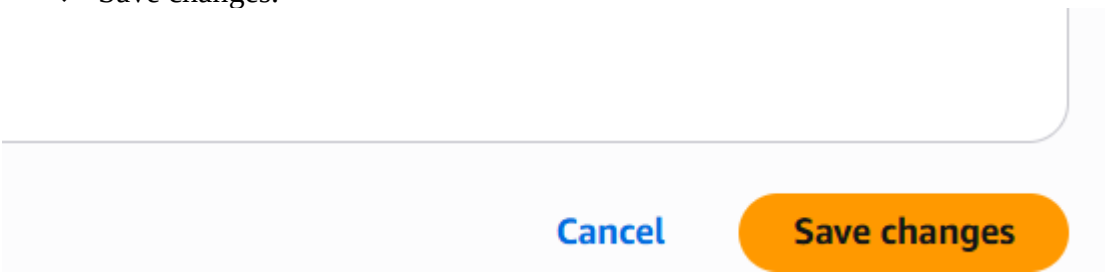
❖ Choose **Host a static website**



❖ Enter:

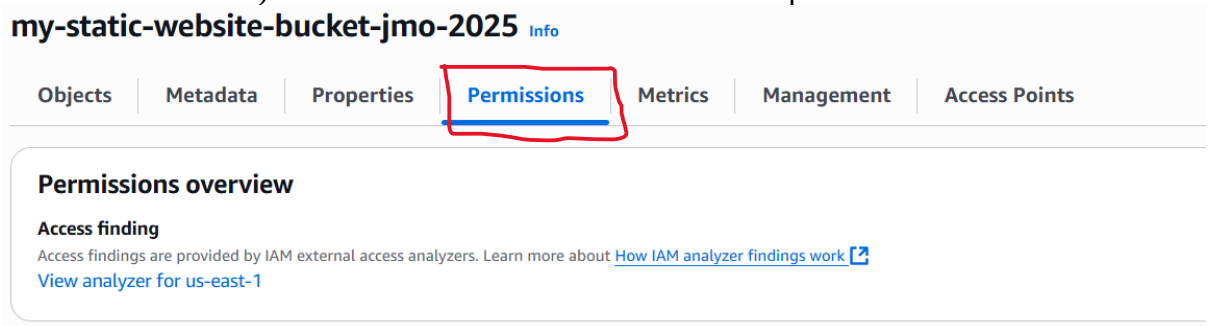
- **Index document**: index.html
- **Error document**: error.html

❖ Save changes.

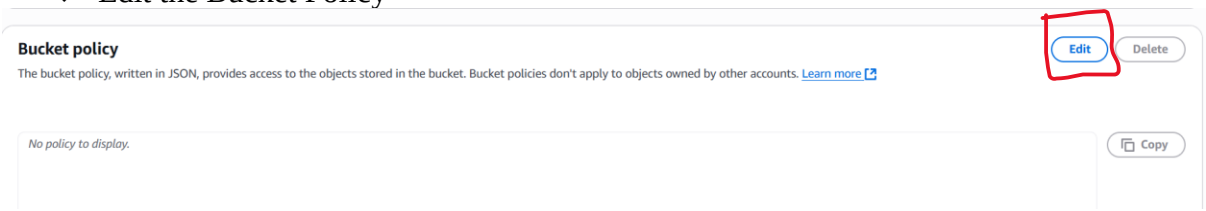


❖ **Set Bucket Policy for Public Access**

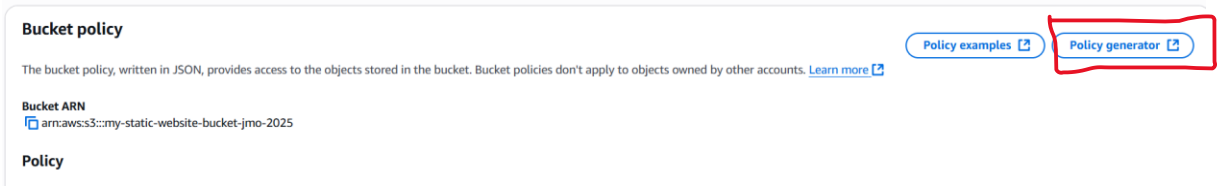
d) Click on the Bucket created and select permissions.



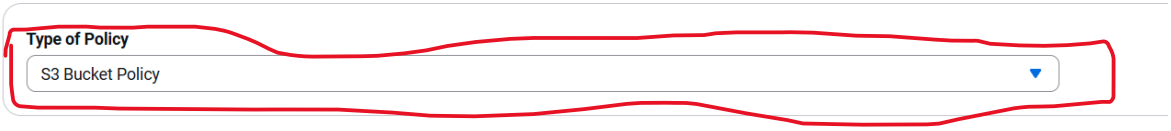
❖ Edit the Bucket Policy



❖ Click on policy generator

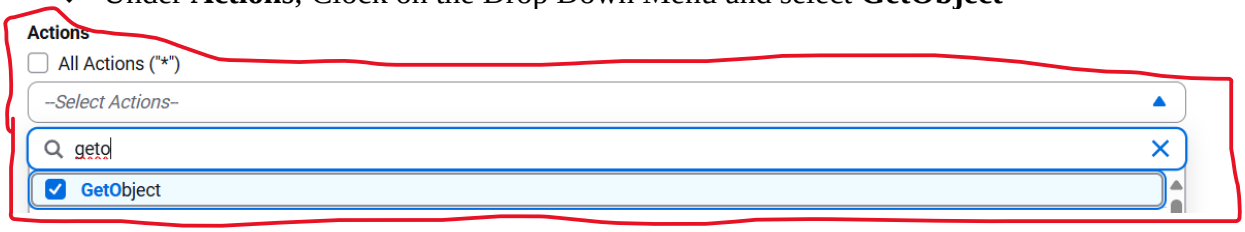


❖ Under type of policy; select **S3 Bucket Policy**

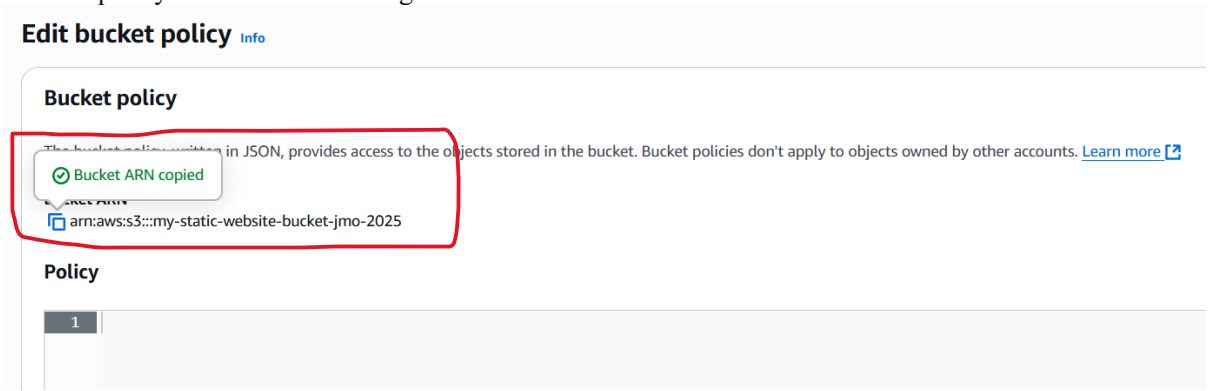


❖ Under Principal, type * (To make it public)

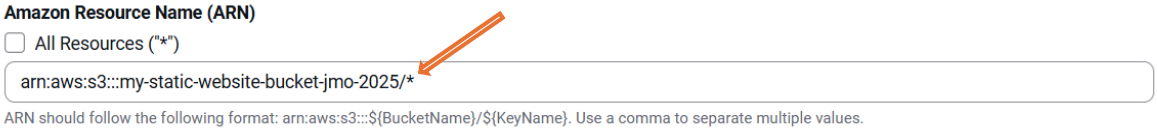
❖ Under **Actions**; Clock on the Drop Down Menu and select **GetObject**



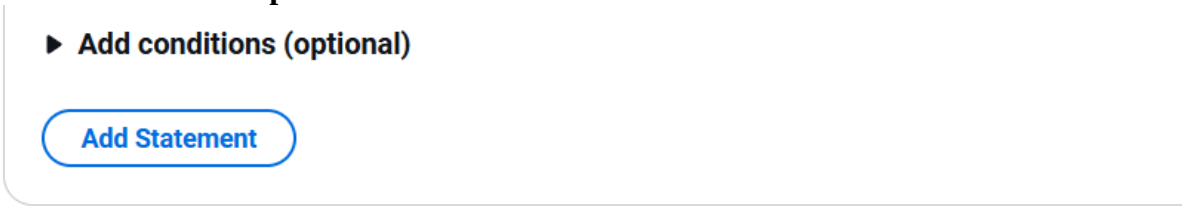
❖ To get **Amazon Resource Name (ARN)** go back to the previous Tab and **COPY** the bucket policy as shown in the Image below.



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

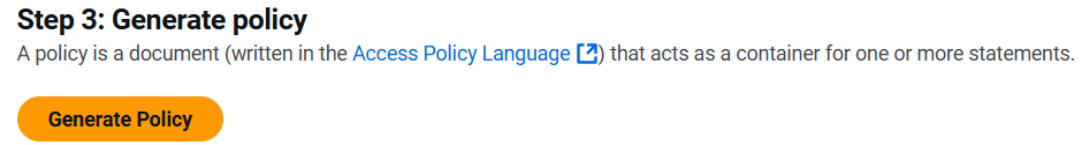


- Go back to the **previous Tab** and Click On **Add Statement**



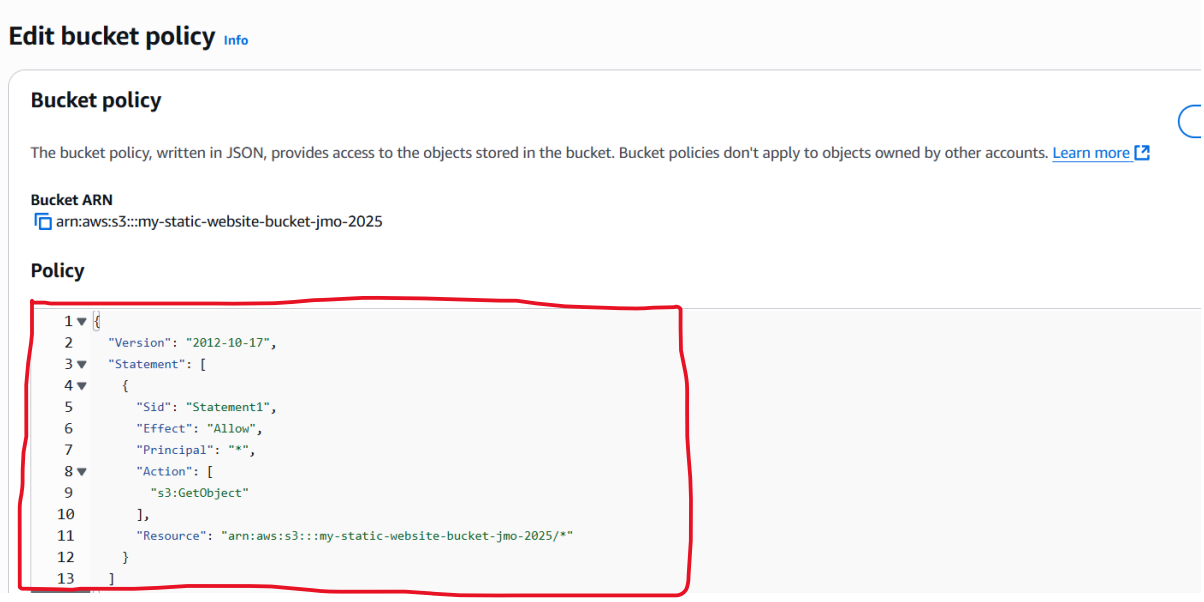
- You will be able to get the output below **(Confirming that Statements have been added)**

- Click on **Generate Policy**



- Copy the Policy

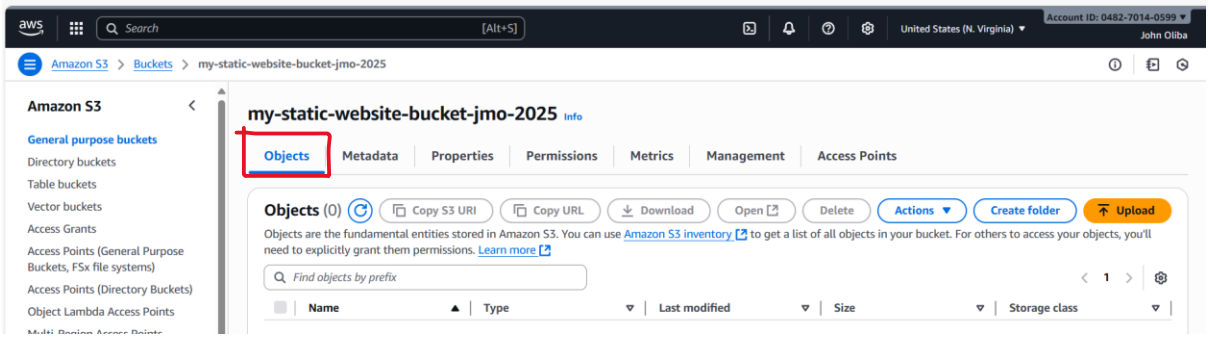
- Go back to the **previous Tab** and **Paste** it in the section having Edit **Bucket Policy**; scroll down and **Save Changes**



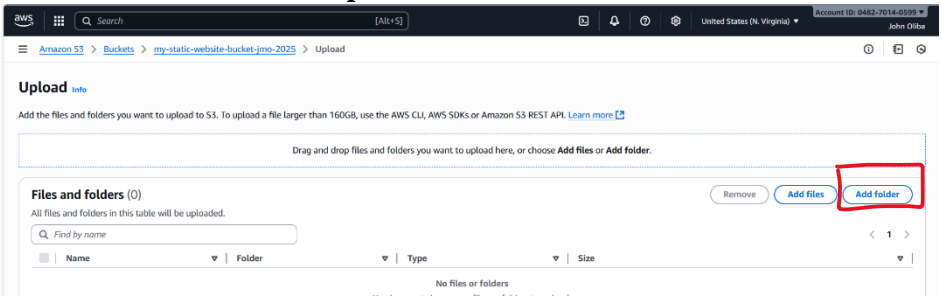
e) Upload the Folder having html, css and js files to the bucket

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

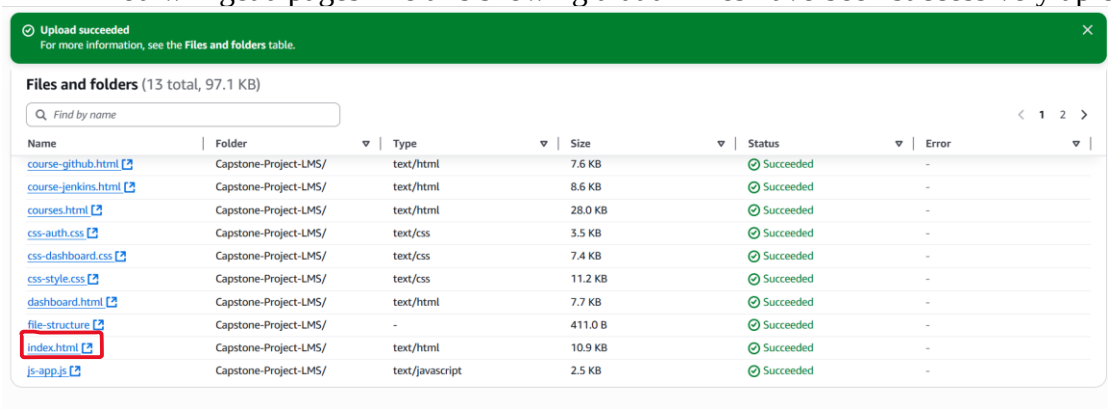
- Select the Bucket you created and click on **Objects**



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

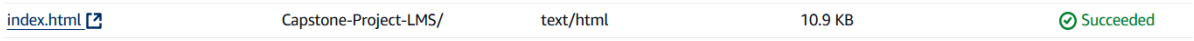


❖ You will get a pages like this showing that all files have been successively uploaded

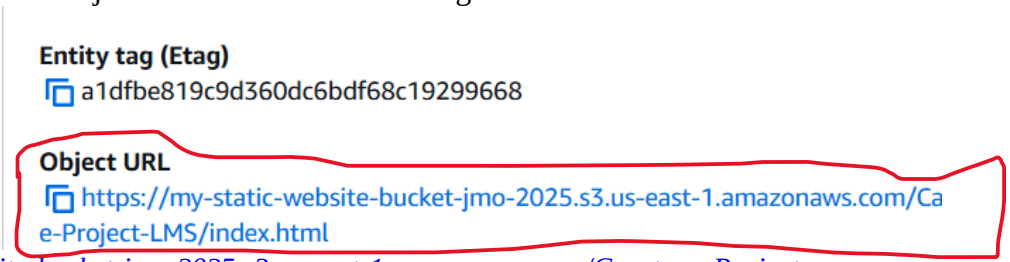


f) Accessing the **Static Website**

➢ .Click on index.html

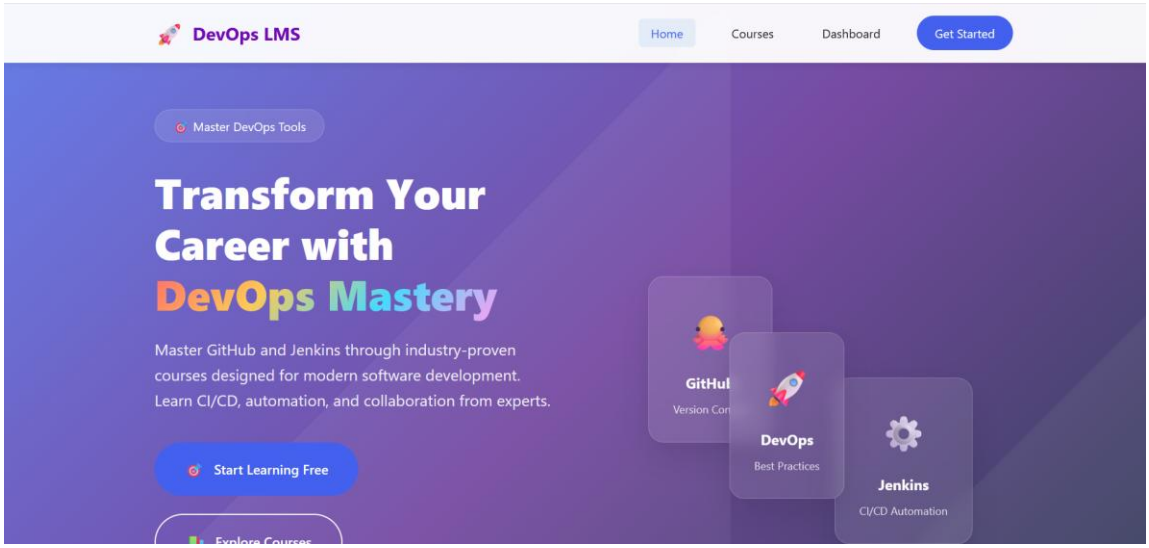


➢ Copy the Object URL from the bottom right corner.



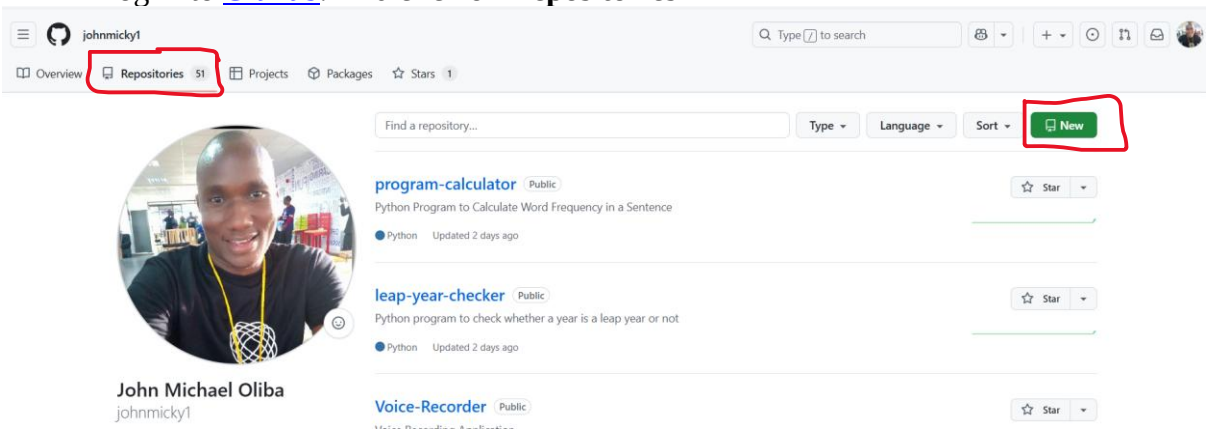
https://my-static-website-bucket-jmo-2025.s3.us-east-1.amazonaws.com/Capstone-Project-LMS/index.html

➢ The website should now be live and accessible!

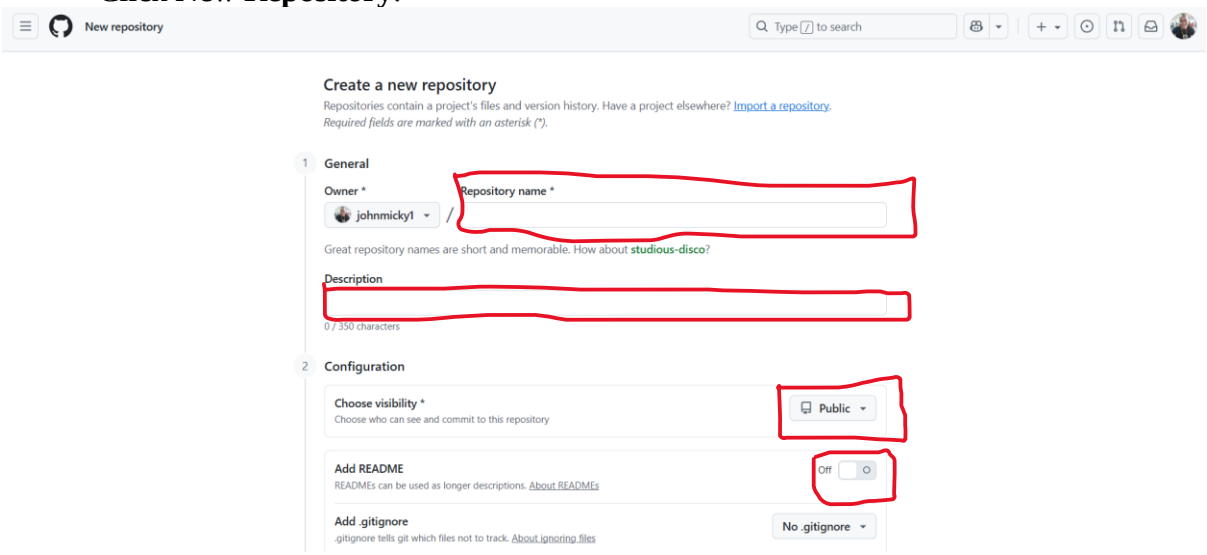


Step 3: GitHub Setup

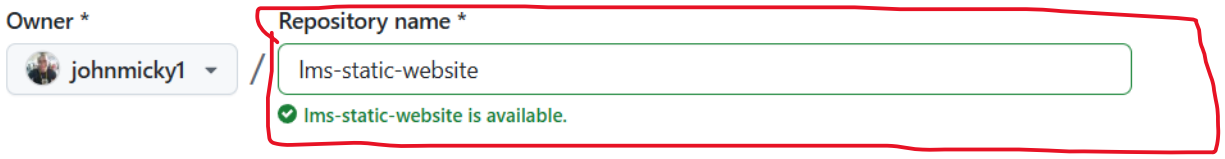
✓ Log in to GitHub. And click on **Repositories**



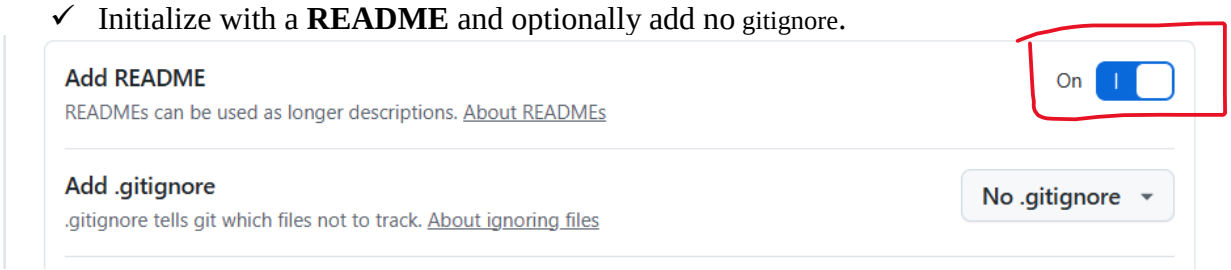
✓ Click **New Repository**.



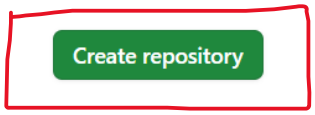
✓ Name it, for example: lms-static-website.



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



✓ Create Repository



✓ Clone the repo locally:
git clone git@github.com:johnmicky1/lms-static-website.git

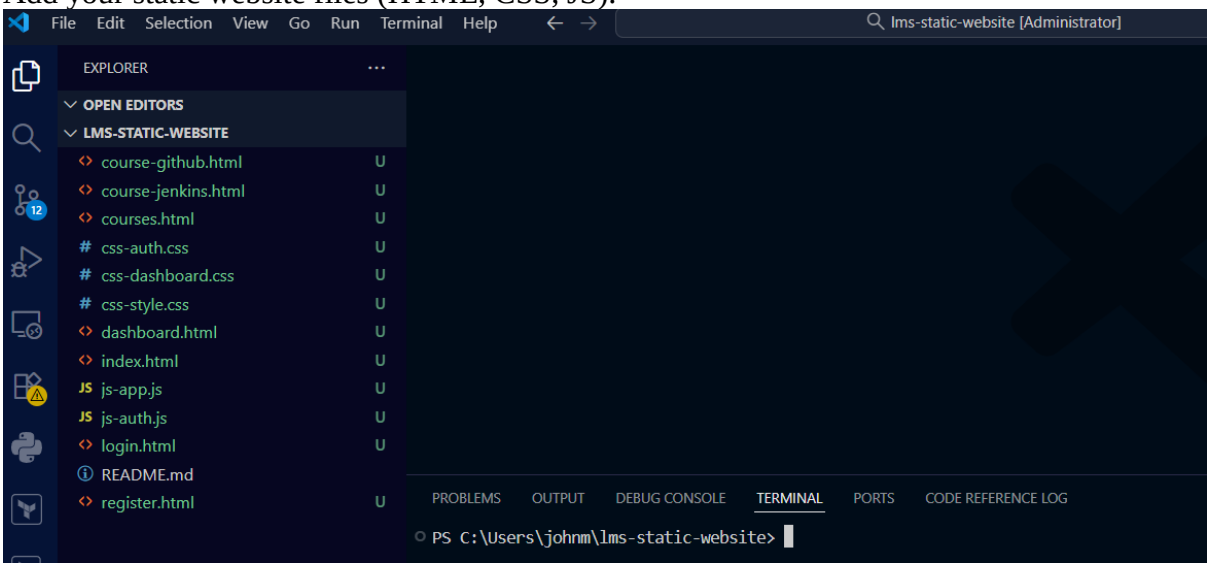
```
johnm@JMO26-04-2025 MINGW64 ~ (master)
$ git clone git@github.com:johnmicky1/lms-static-website.git
Cloning into 'lms-static-website'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused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

✓ **git status**

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

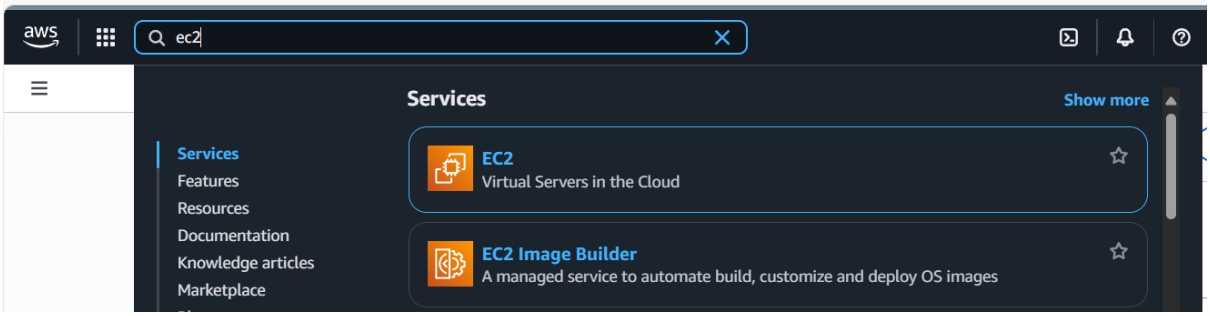
✓ **git push**

```
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
```

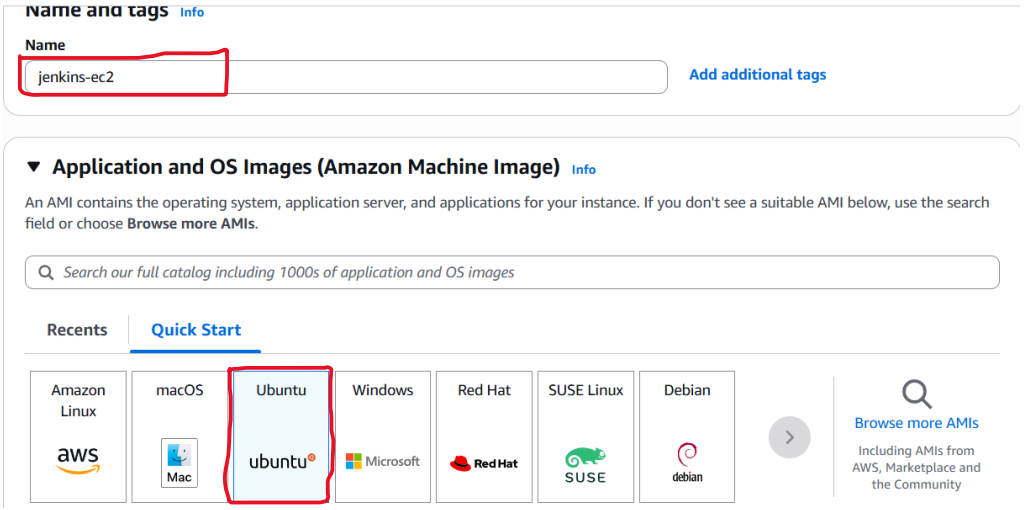
---

4. EC2 Setup (for Jenkins)

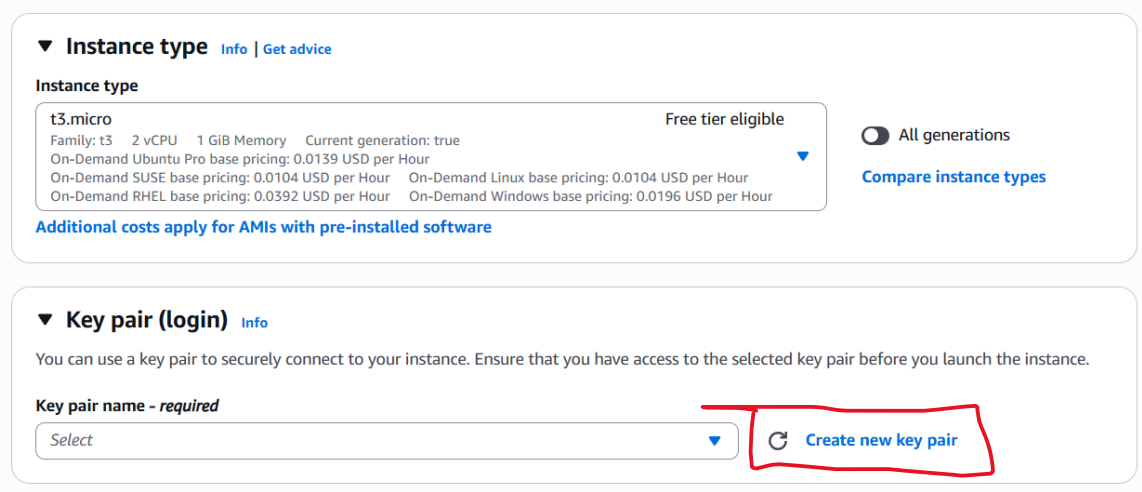
4.1 Launch EC2 Instance



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

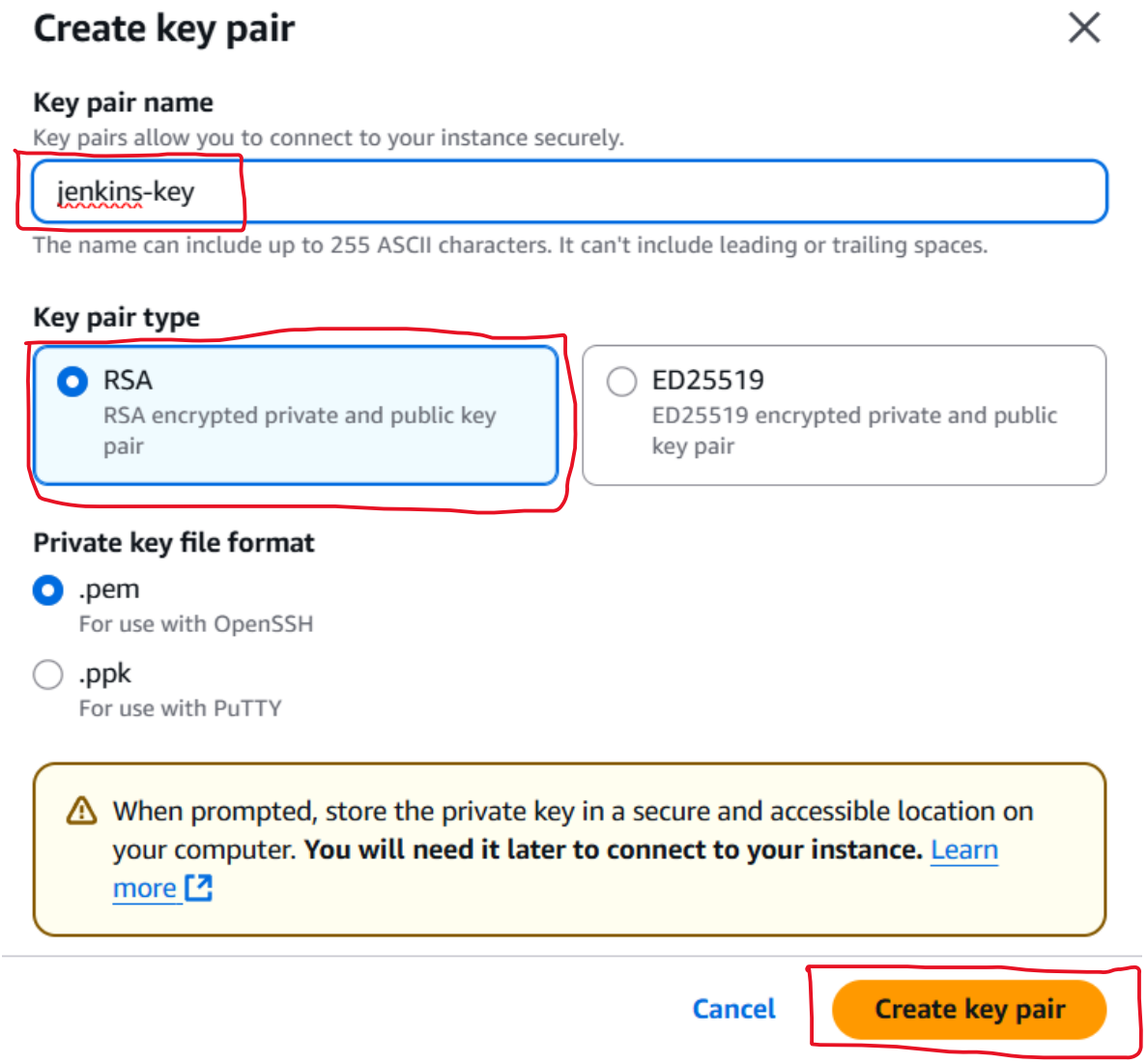


- Scroll down and create a key pair

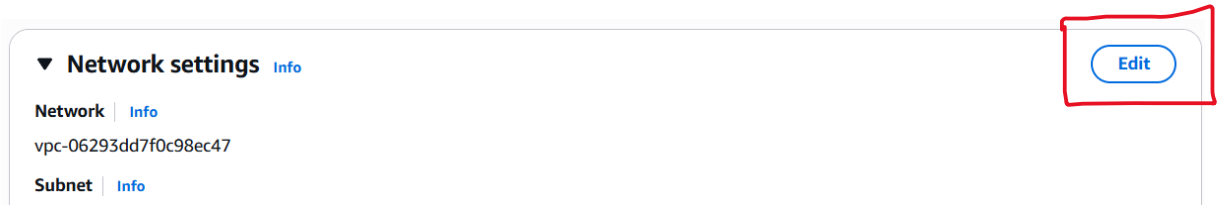


Create a Key Pair

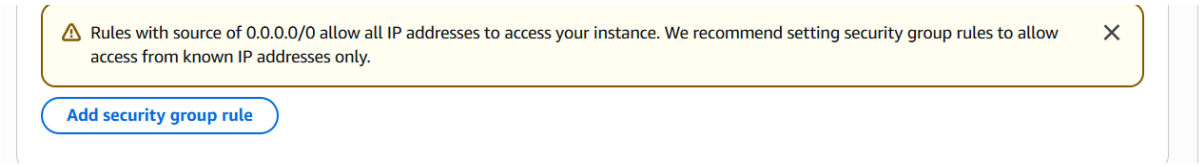
- Give the key pair a name (e.g., "jenkins-key").
- Click "Create key pair."
- Save the downloaded key file in a secure location to use for SSH access to your EC2 instance.



2. Under **Network Settings,** click **Edit**

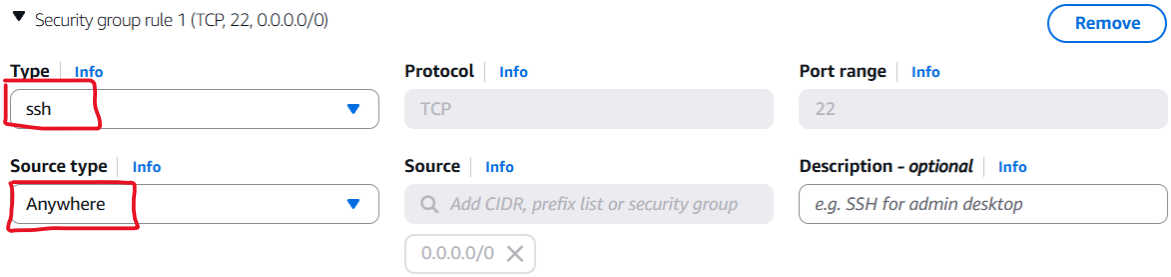


Click Add Security Group Rule (Security Group (inbound rules)😊

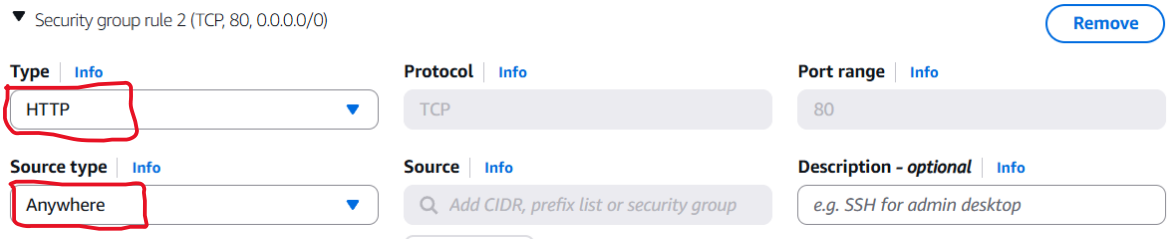


Add the port numbers as shown in the Images below

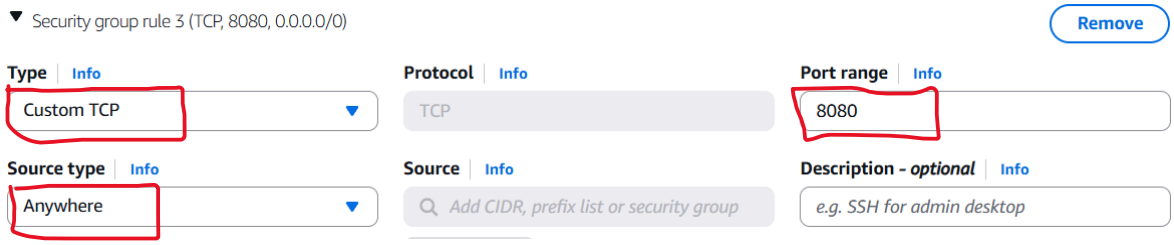
- SSH (TCP 22)



- HTTP (TCP 80)



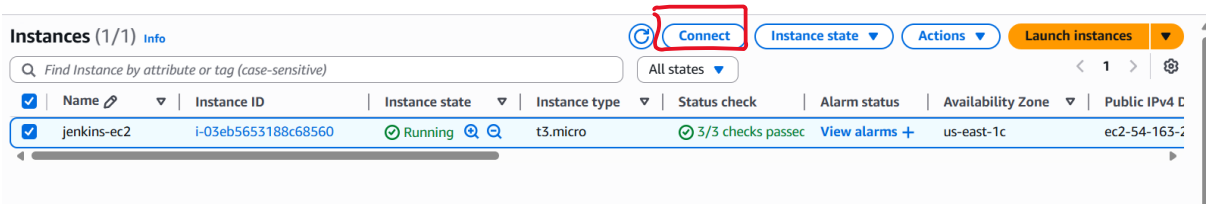
- Jenkins (TCP 8080) — restrict to your IP or workplace IP range



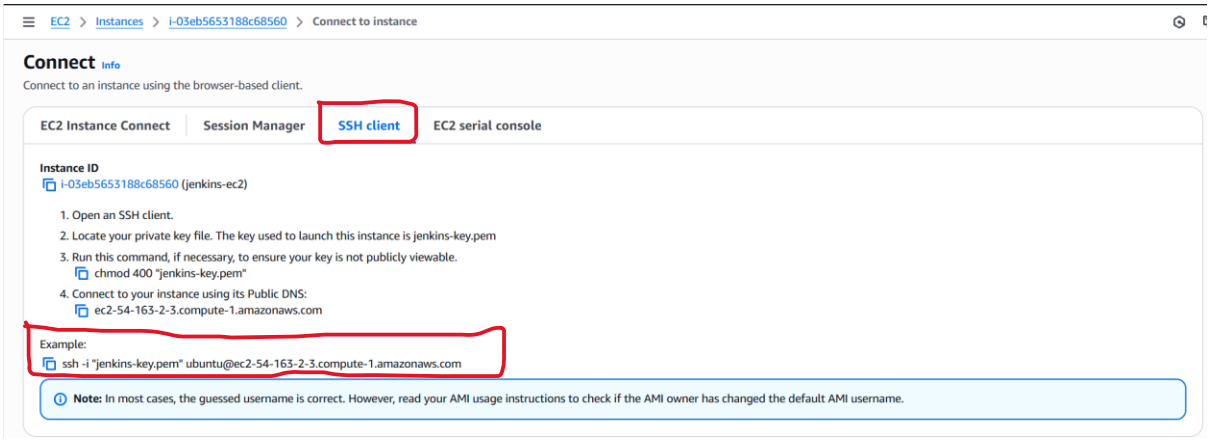
3. Launch and wait until instance is **running**.

4.2 Connect via SSH

- Select Instance and Click **Connect**



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



- 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/b3bxTYsVYzDWNmF6WH3PCzXVd4dlvtQMktBM2bCM.
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):
```

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]-----+
```

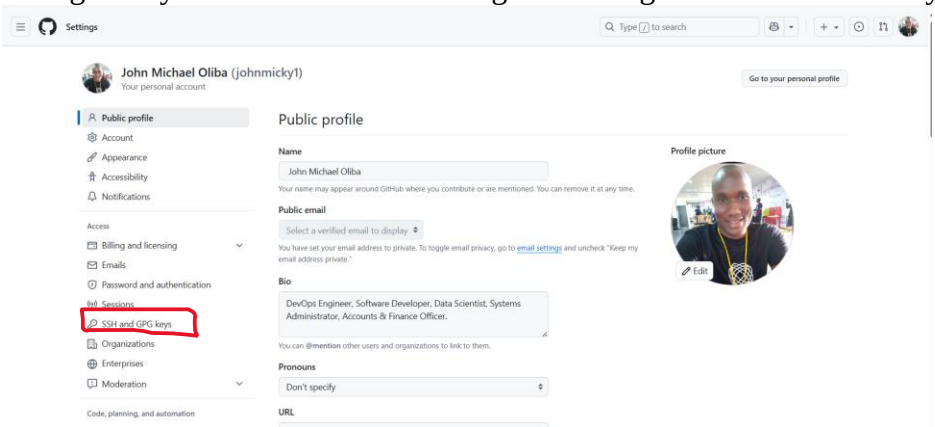
5. Use the cat command to display the contents of the public key file:

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

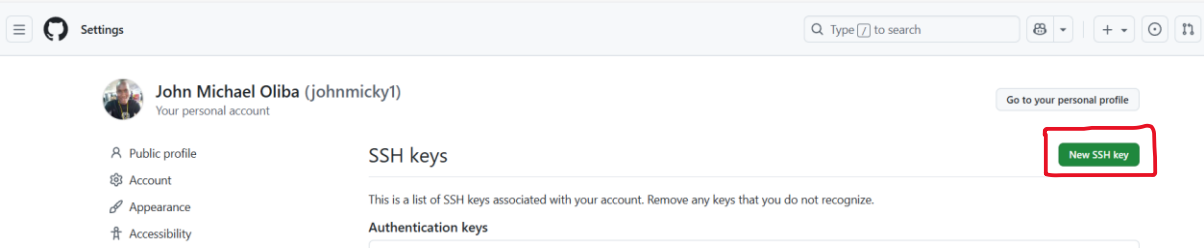
bash
cat ~/.ssh/id_rsa.pub

```
ubuntu@ip-172-31-25-49:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDIBxEu4PzBy8SiMHpcjQz6LayoBoekS15Z1QnWNgOP8yHeeyZMHAgB9cZKfBvhcBrrGTDN5dwe+OMdOloRSKxLaFs1svanuM0IoYyppweRFZdLrekieOzd
9eLVt/6jFHKEQFseIz+ynPmBeS84CZwEqDJZx0HhabKq4/WCIpGeqf8Lykw0HEaeDIKs/Pb5fpm0m4mPyM50XS6JLw9eL2158XSPv9VRevS/nnef3xTmFR/ttVLnE7orvxNRe+CJkV6CxulkgWalkMsb0JGG
Qo0re6QqI1IYEhqmcLQi+5fyhrksA22Fxp2liTjd4ZI5vVPGHELy6ylp+nK+grdXvTJVNKo7SnPC8dhMEUJNLL69f701LDYgv+t7hasyhPGrNY+atI0KfDgOQqC3K/xOupbS0k3CA1QpSyB5rtEZMKMEl60F
D5Fh5PgPd6ULjcYAzTT9cHNwuu8DpN3MwNjFuWSnZYDrdGyQCWoevj09pzo7NukNtRRrqTJhMpRANygi8ZF3xuYsPpMRHa9x+yWxhe19FzQ0Pq8VVeQW6zAK0UCssmCx+EibzN825J1Rb/HwAp1wRemh7EyH
iKnODNOMXJ2WP687J+Tw4x7GdC+/w7Myc/8x7Z7Zz+eRSBVNsvHpPLmS9dWK4F/PgjL0rZn9kZhTpQYIOhO/lXDb9+Fu5LKe9Q== ubuntu@ip-172-31-25-49
```

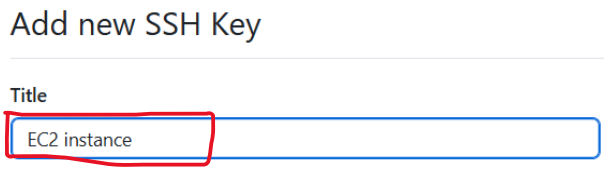
Add SSH key to GitHub
2. Log in to your GitHub account and go to Settings > SSH and GPG keys.



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



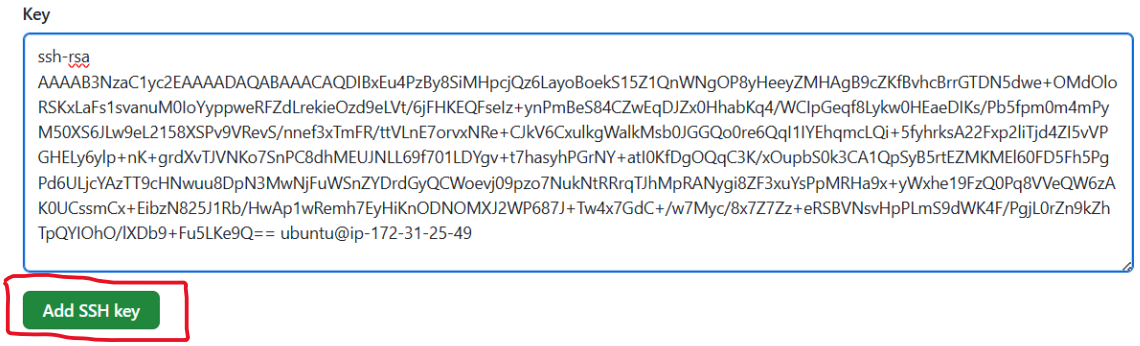
4. Give the key a label (e.g., "EC2 instance")



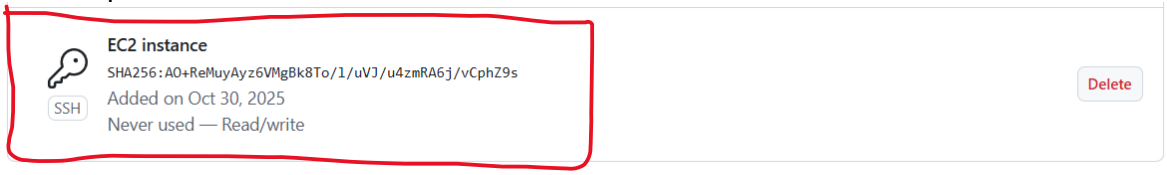
5. Go back to ssh and copy the whole key generated when you typed command (cat ~/.ssh/id_rsa.pub)

```
ubuntu@ip-172-31-25-49:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDIBxEu4PzBy8SiMHpcjQz6LayoBoekS15Z1QnWNgOP8yHeeyZMHAgB9cZKfBvhcBrrGTDN5dwe+OMdOloRSKxL
9eLVt/6jFHKEQFseIz+ynPmBeS84CZwEqDJZx0HhabKq4/WCIpGeqf8Lykw0HEaeDIKs/Pb5fpm0m4mPyM50XS6JLw9eL2158XSPv9VRevS/nnef3xTmFR/ttVLn
Qo0re6QqI1IYEhqmcLQi+5fyhrksA22Fxp2liTjd4ZI5vVPGHELy6ylp+nK+grdXvTJVNKo7SnPC8dhMEUJNLL69f701LDYgv+t7hasyhPGrNY+atI0KfDgOQqC3
D5Fh5PgPd6ULjcYAzTT9cHNwuu8DpN3MwNjFuWSnZYDrdGyQCWoevj09pzo7NukNtRRrqTJhMpRANygi8ZF3xuYsPpMRHa9x+yWxhe19FzQ0Pq8VVeQW6zAK0UCs
iKnODNOMXJ2WP687J+Tw4x7GdC+/w7Myc/8x7Z7Zz+eRSBVNsvHpPLmS9dWK4F/PgjL0rZn9kZhTpQYIOhO/lXDb9+Fu5LKe9Q== ubuntu@ip-172-31-25-49
```

6.click **Add SSH key.**



Your Output will look like this



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/zLDA0zPMSvHdkr4UvCOqU.
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-certific
  fonts-dejavu-mono gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humani
  libatk-wrapper-java-jni libatk1.0-0t64 libatspi2.0-0t64 libavahi-client3 libavahi-common-da
  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
```

💻 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/sour
```

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
```

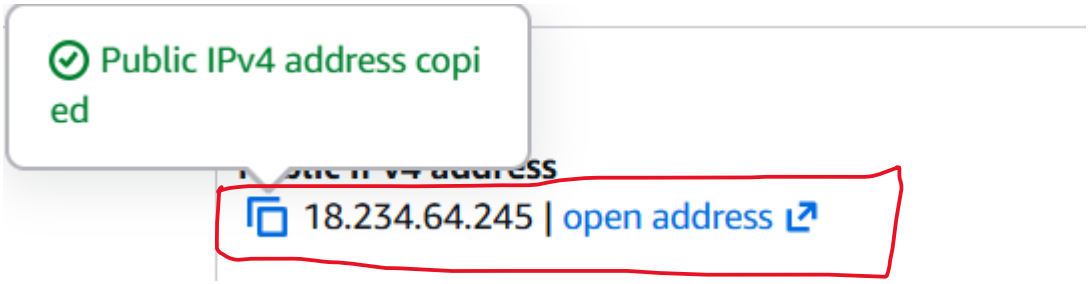
Step 6: Enable Jenkins to Start at Boot

bash
sudo systemctl enable jenkins

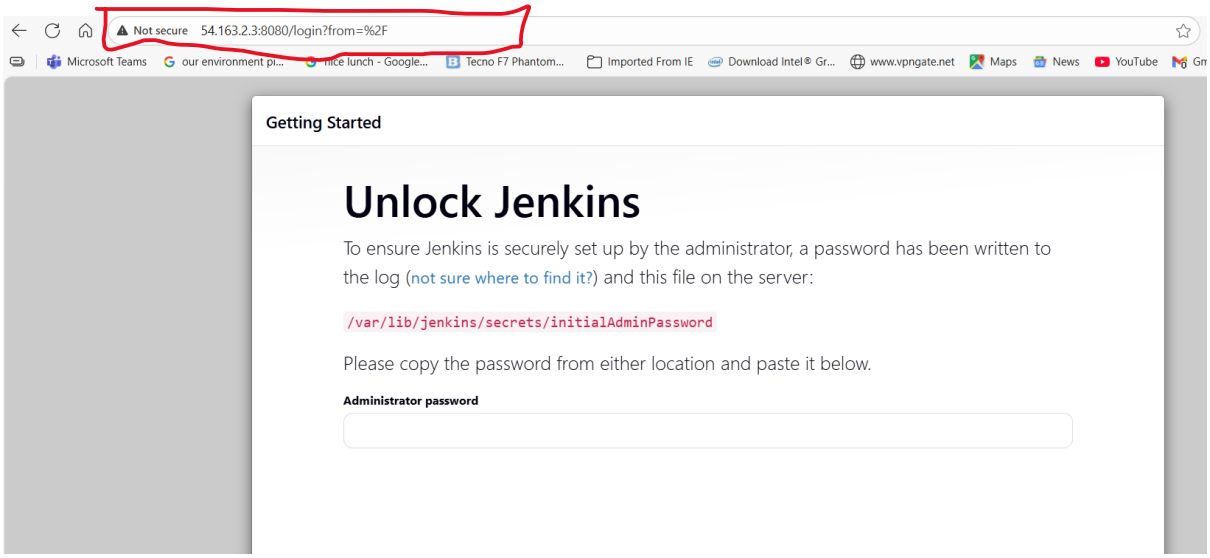
```
ubuntu@ip-172-31-26-70:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
```

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



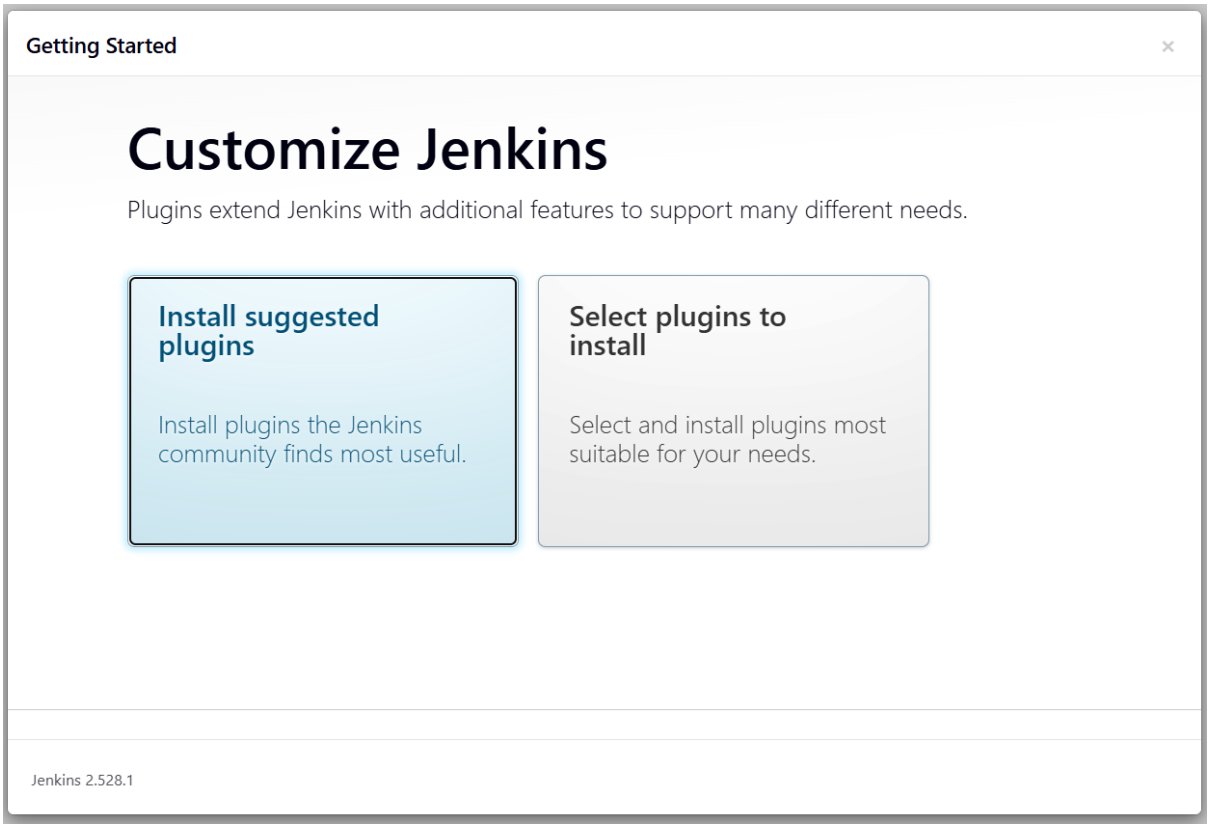
2. In SSH session run to get the Administrator Password (copy it ans paste it to the Administrator password section under unlock Jenkins)

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

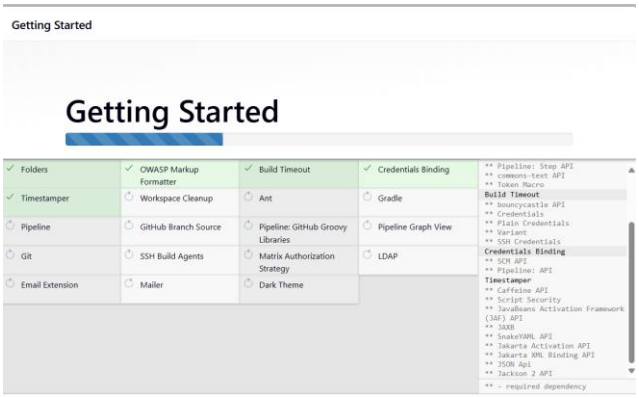
```
ubuntu@ip-172-31-26-70:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
4ef3e471230146c0814ca59a08cb7a07
```

4ef3e471230146c0814ca59a08cb7a07

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

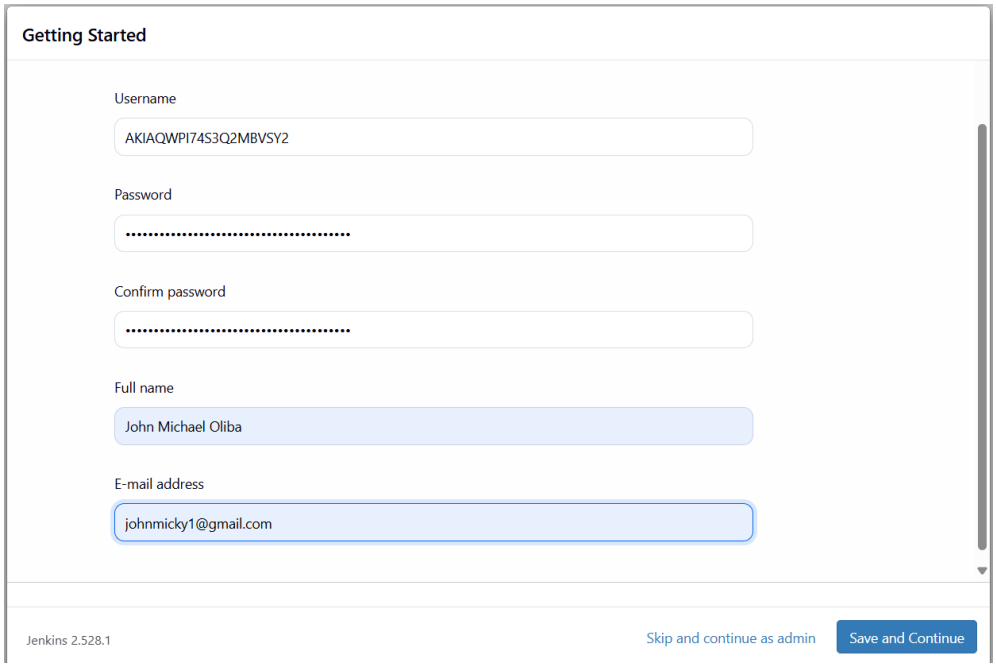


4. Install these recommended plugins (if asked): Git, Pipeline, NodeJS, AWS Steps / AWS Credentials, GitHub Integration.

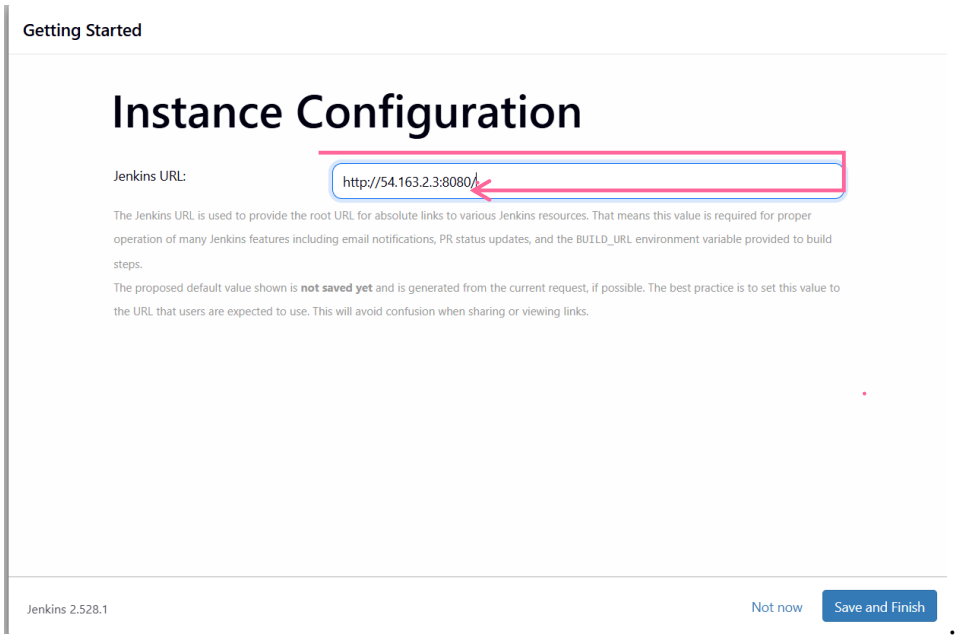


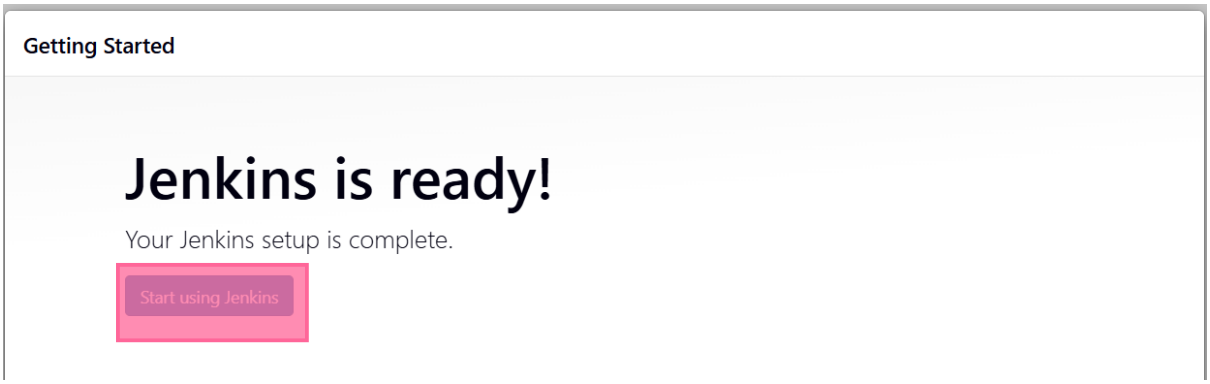
6 — Add AWS credentials to Jenkins

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**



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



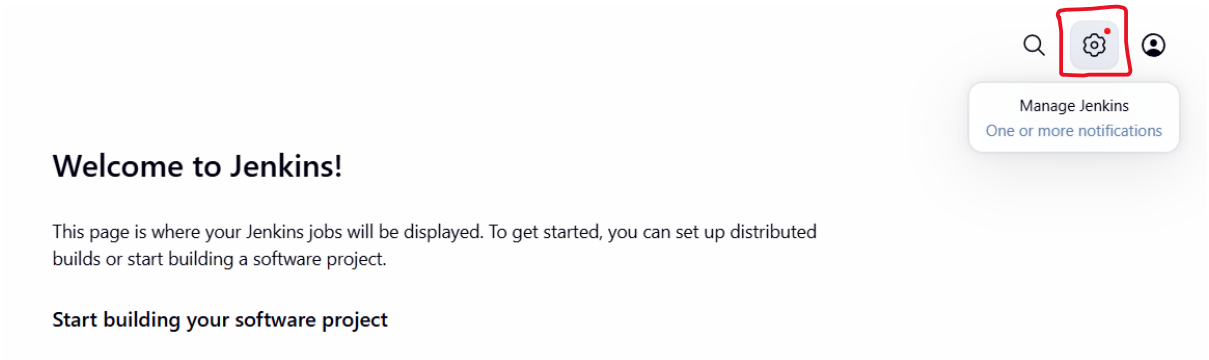


3. Give an ID (e.g. aws-jenkins-creds) — you will refer to this in the Jenkinsfile.

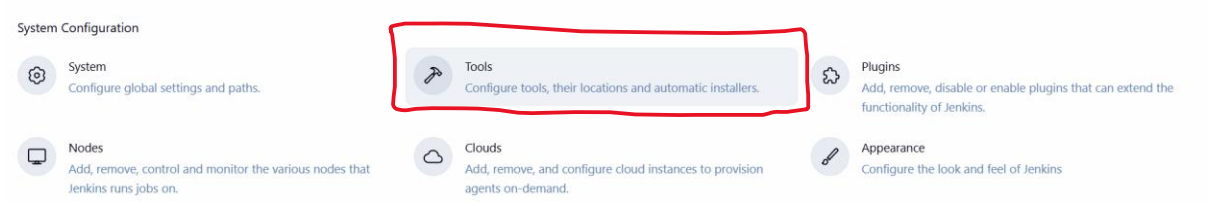
---

6. Configure Jenkins

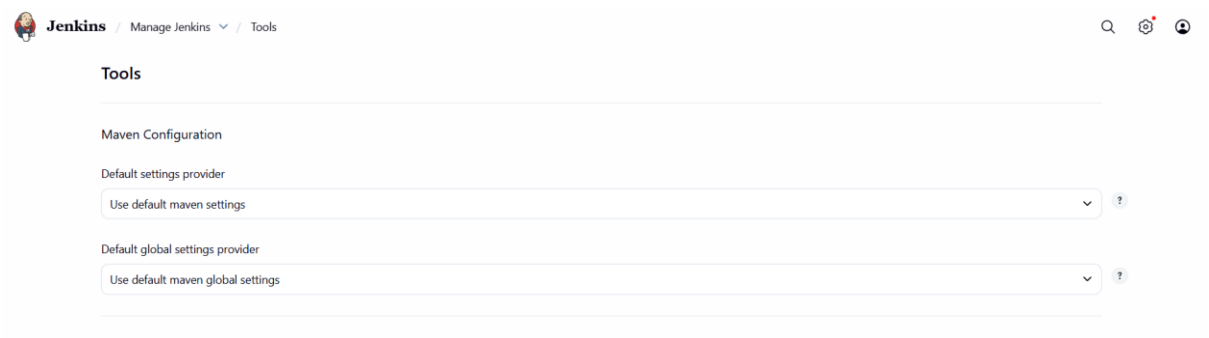
- Click on the manage Jenkins (right hand top corner)



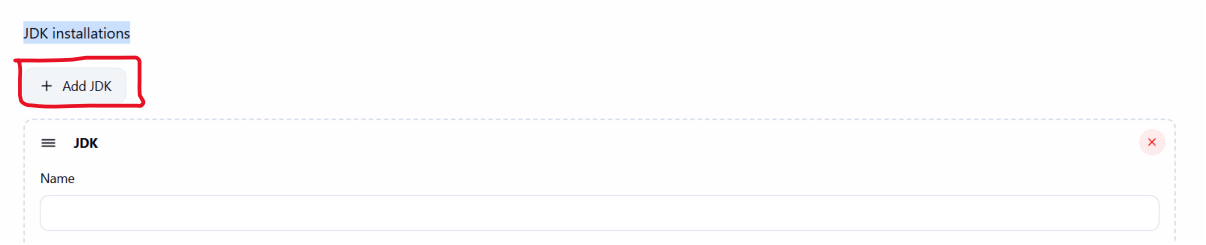
- Click on tools



- Your Display will look like this



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

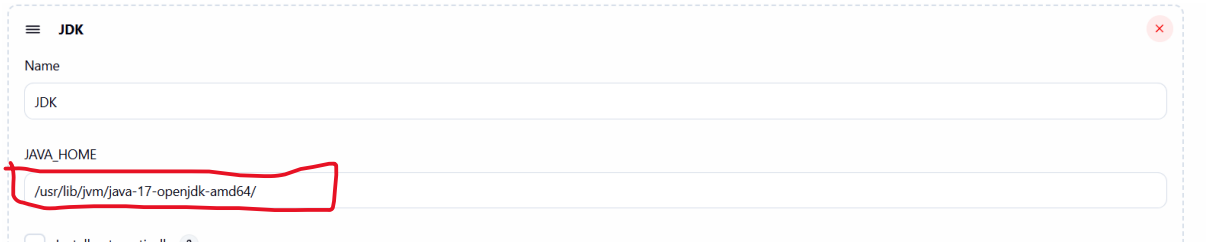


- Go back to SSH and type readlink -f /usr/bin/java | sed 's/bin\/java//g' to find the path

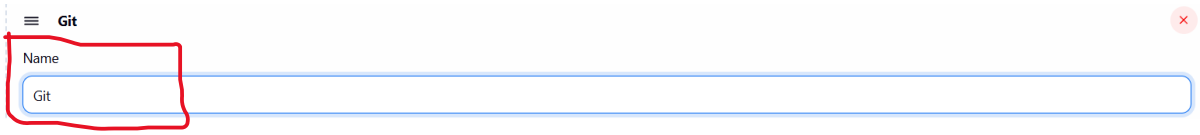
```
ubuntu@ip-172-31-25-49:~$ readlink -f /usr/bin/java | sed 's/bin\/java//g'
/usr/lib/jvm/java-17-openjdk-amd64/
```

- 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**

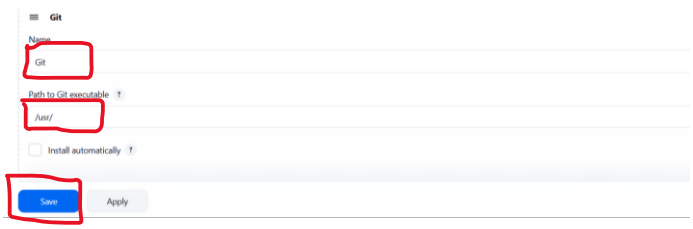


- Go back to ssh and type the command below to get the path

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

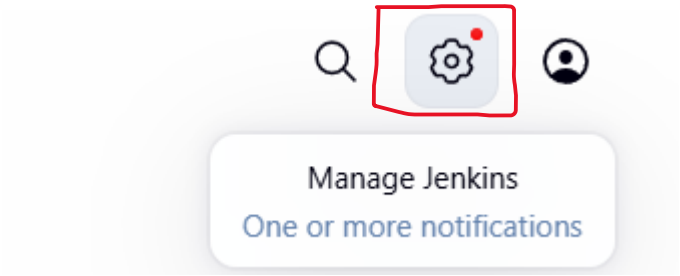
```
ubuntu@ip-172-31-25-49:~$ readlink -f $(which git) | sed 's/bin\/git//g'
/usr/
```

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

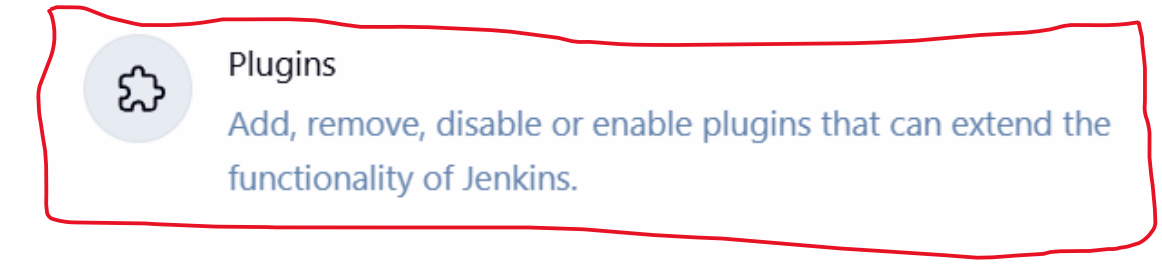


Install Required Plugins

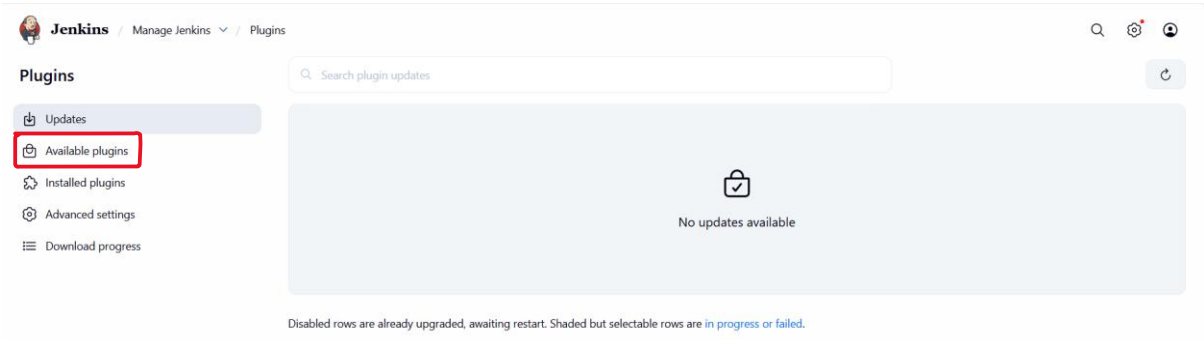
- Go to **Manage Jenkins**



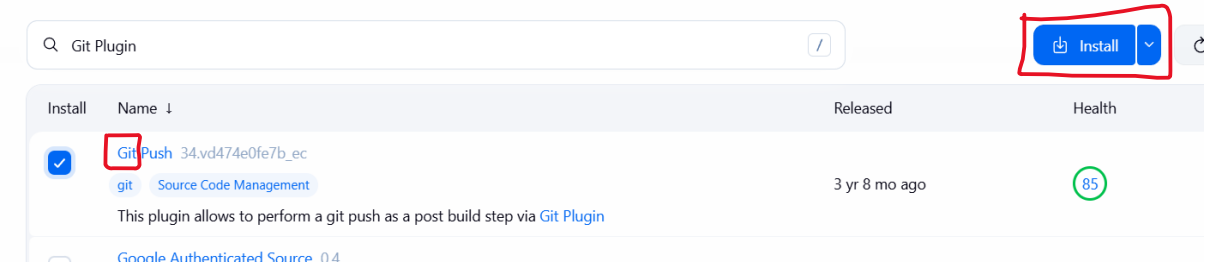
- → **Plugins**



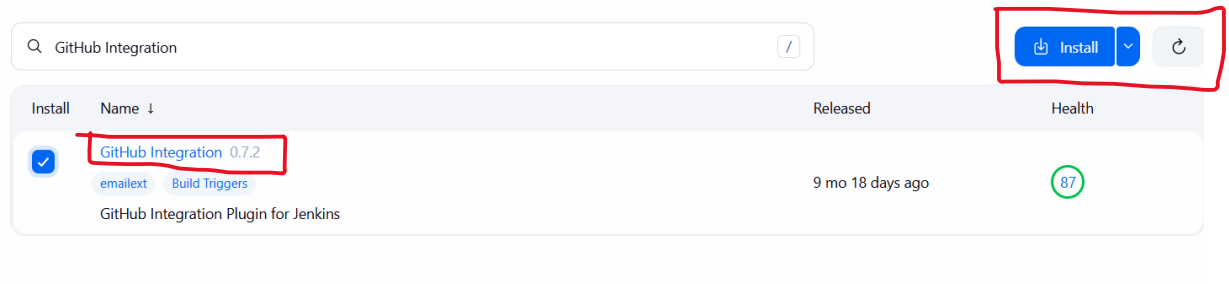
- → **Available Plugins**.



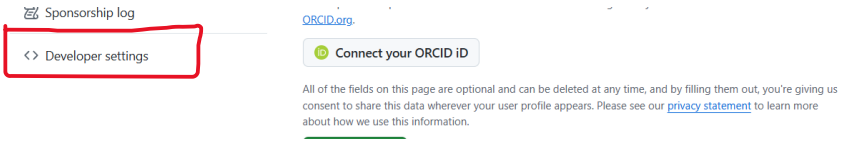
- Search and install:
  - **Git –** type Git in the search box, select all git plugins **(Click Install) Restart Jenkins when the installation is complete**



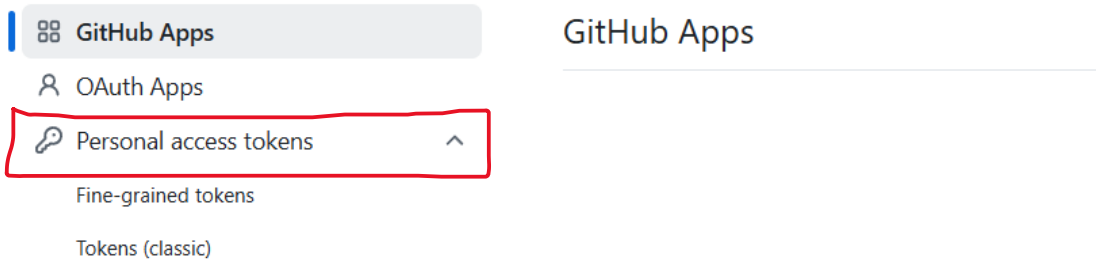
- **GitHub Integration –** type GitHub Integration in the search box, select all git plugins **(Click Install) Restart Jenkins when the installation is complete**



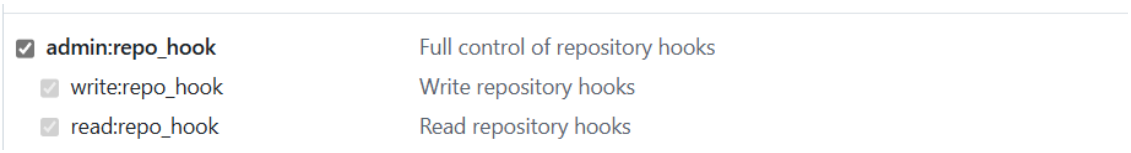
- **Create GitHub Personal Access Token**
  - In GitHub → **Settings → Developer Settings**



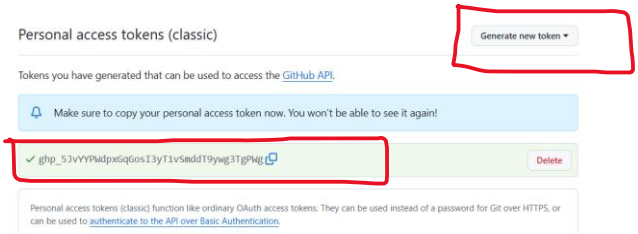
  - **→ Personal Access Tokens → Tokens (Classic).**



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

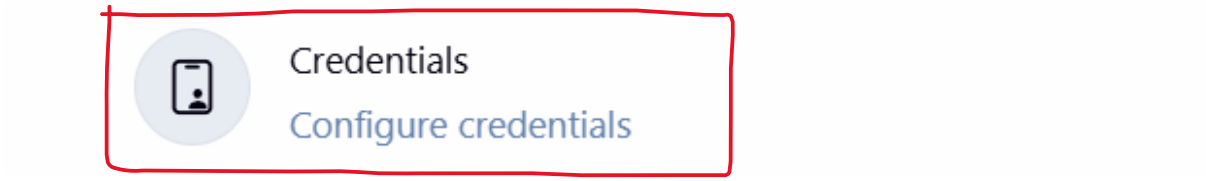


  - Copy the token (you'll need it for Jenkins).

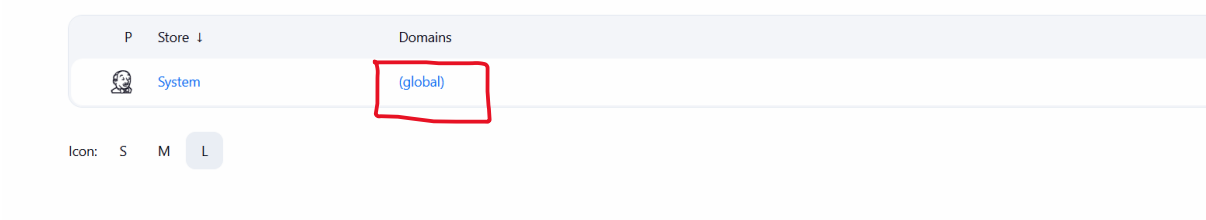


Token:
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

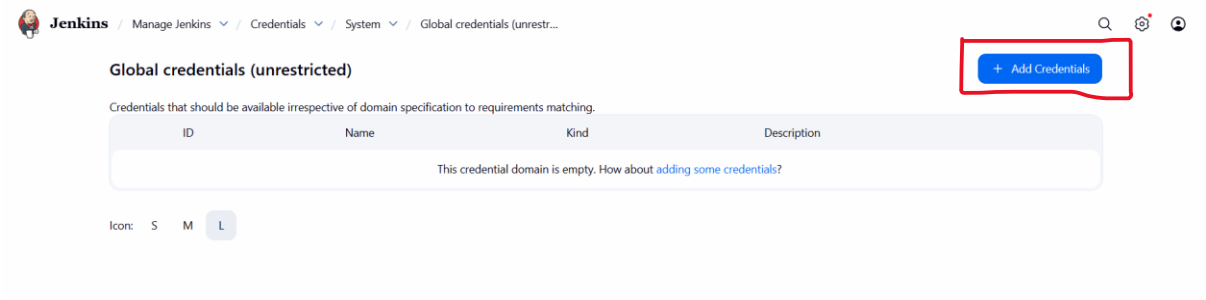
- **Add GitHub Credentials in Jenkins**
  - Go to **Manage Jenkins → Credentials**



  - **→ System → Global credentials.**



  - Click **Add Credentials**:



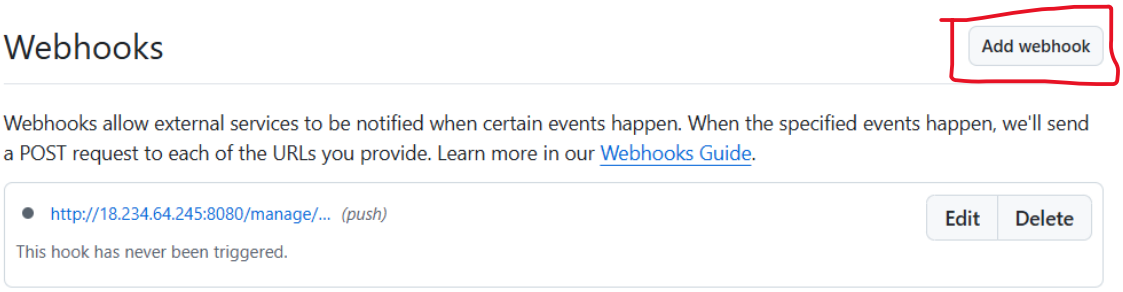
    - Kind: *Username with password*
    - Username: your GitHub username
    - Password: paste the GitHub token
    - ID: e.g., github-access-token

Then Click Create. Your output will look like this below

Create and Configure Webhook in GitHub

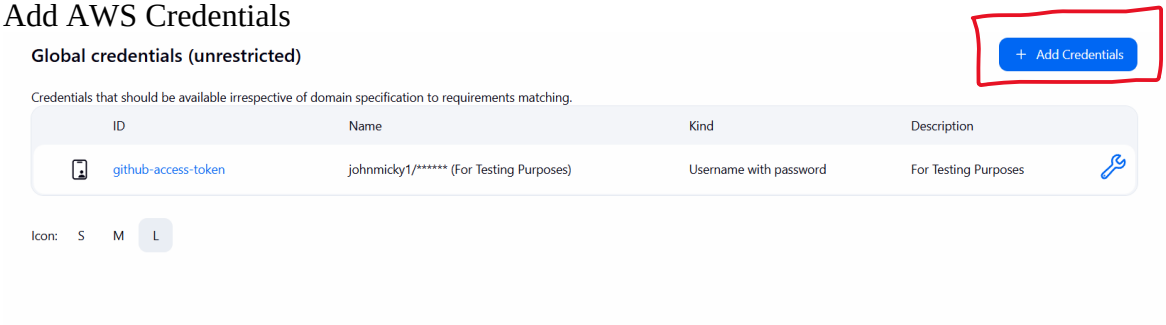
- Go to your **GitHub repository → Settings → Webhooks → Add Webhook**.
  - Content type: application/json

- 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/



- Then Click → **Add webhook**

- Add AWS Credentials



- Punch in Access Key, Secret Key and 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
```

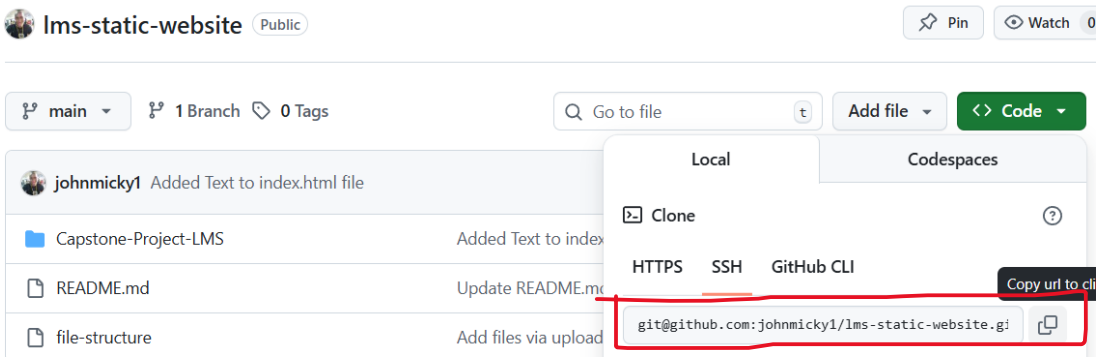
- **Ensure your EC2 instance can run AWS CLI commands; Type command below**

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

```
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.sam
ple
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"

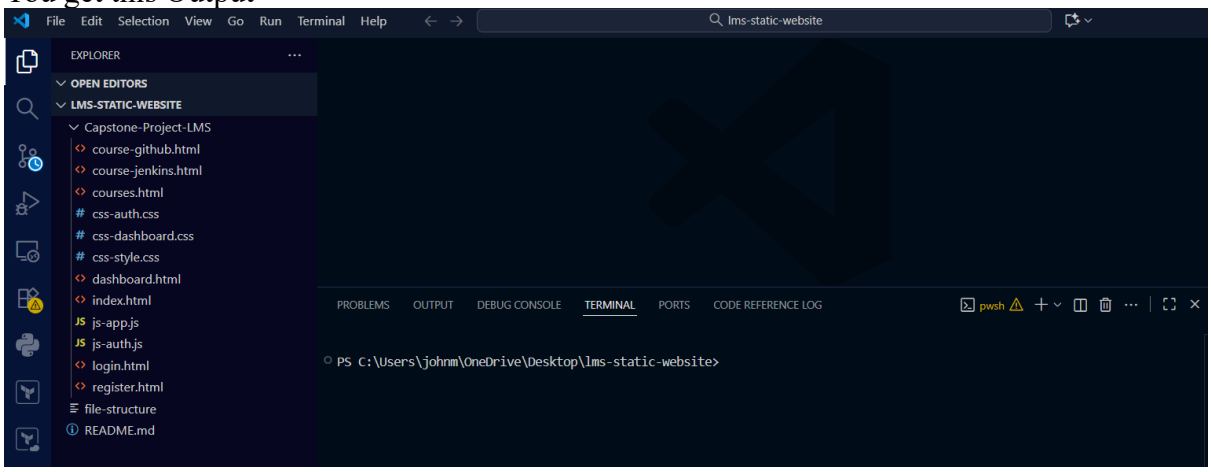
```
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> echo "Index.html file found in S3 bucket"
Index.html file found in S3 bucket
```

Open Repository on Vs Code or any other code Editor

Code .

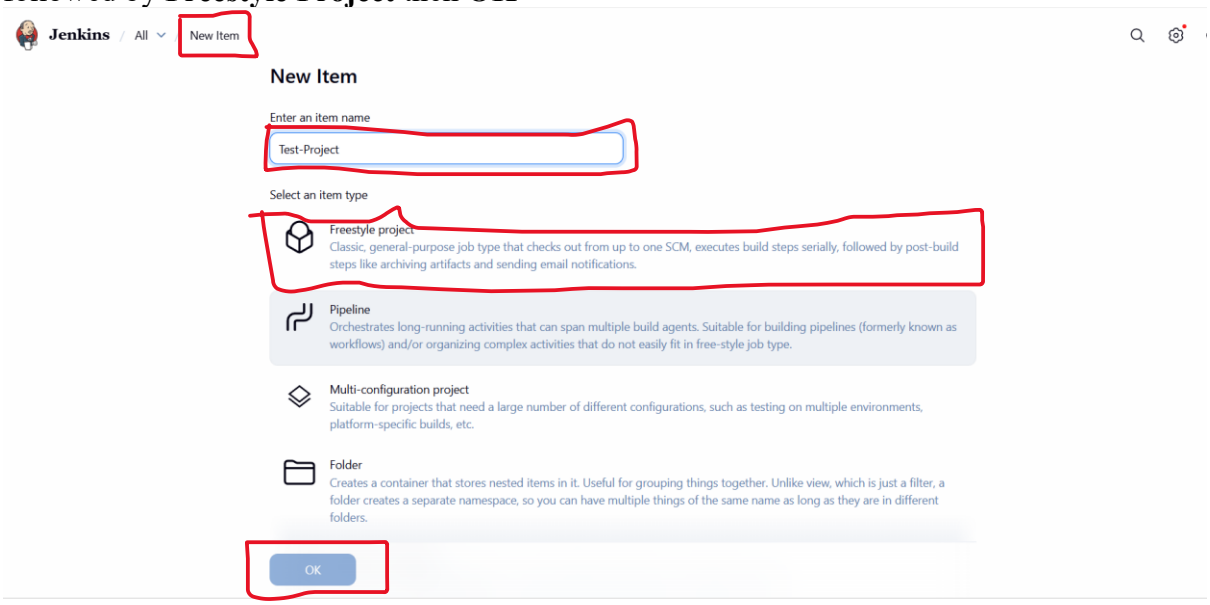
```
Windows PowerShell
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> code .
```

You get this Output

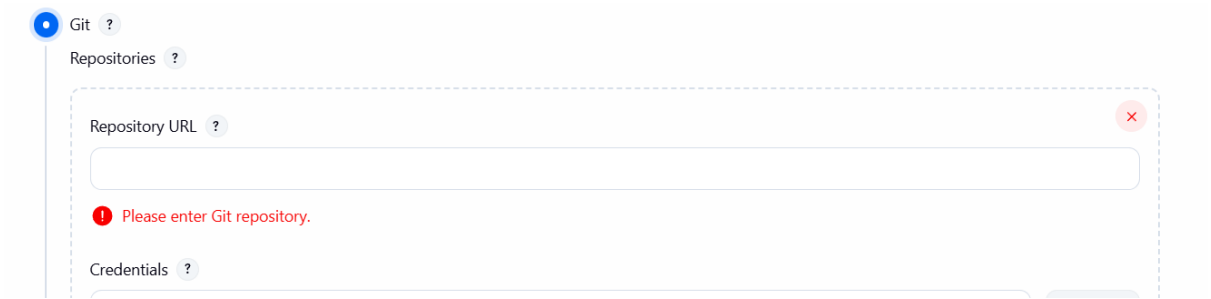


Step 9: **Test the Pipeline**

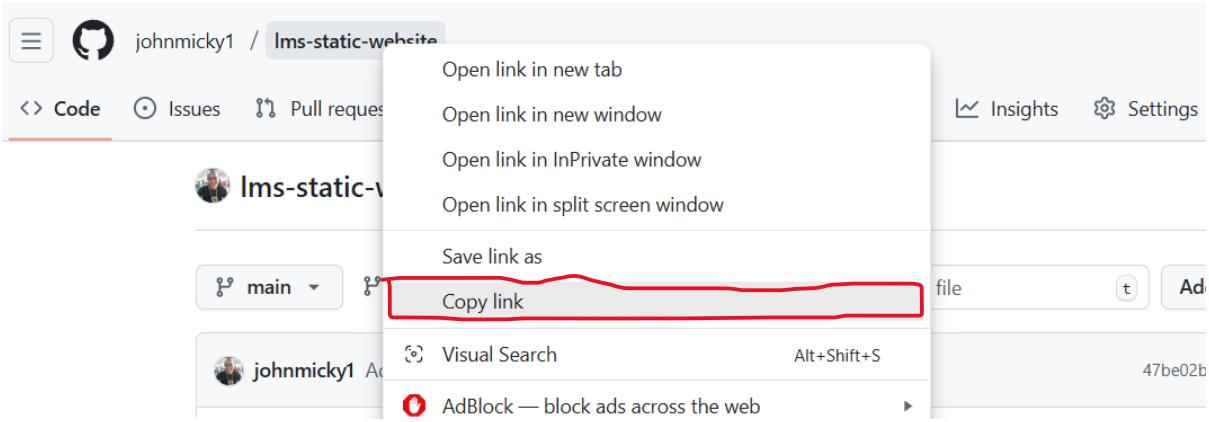
- ✓ Log in to your Jenkins instance and click on **New Item** then type **Test-Project** followed by **Freestyle Project** then **OK**

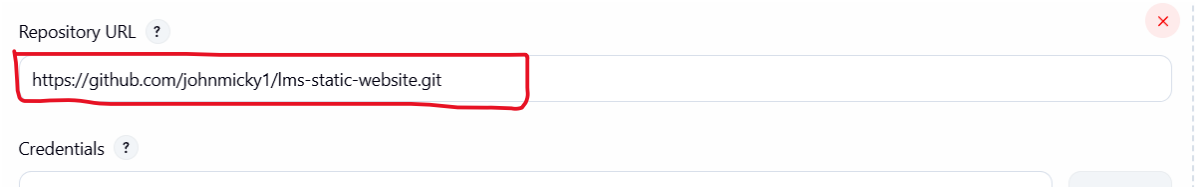


- ✓ Configure the GitHub repository
- ✓ In the project configuration, scroll down to the Source Code Management section.
- ✓ Select Git as the source code management system.

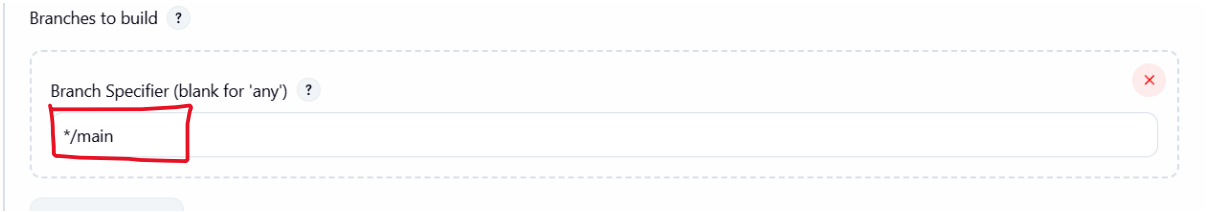


✓ Enter the URL of your GitHub repository (e.g., https://github.com/johnmicky1/lms-static-website.git).



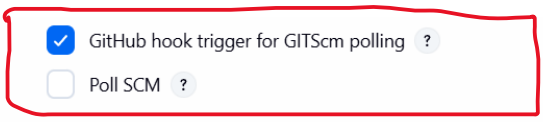


✓ Scroll down to Branches to build (change it to main since the Repo is on main)

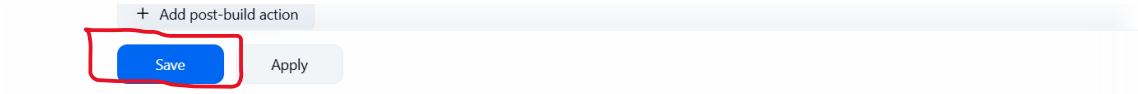


Configure the build trigger

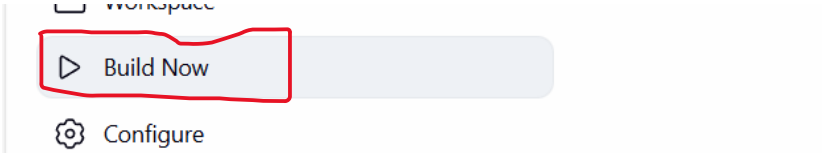
✓ Scroll down to the Build Triggers section.
✓ Check GitHub hook trigger for GITScm polling.



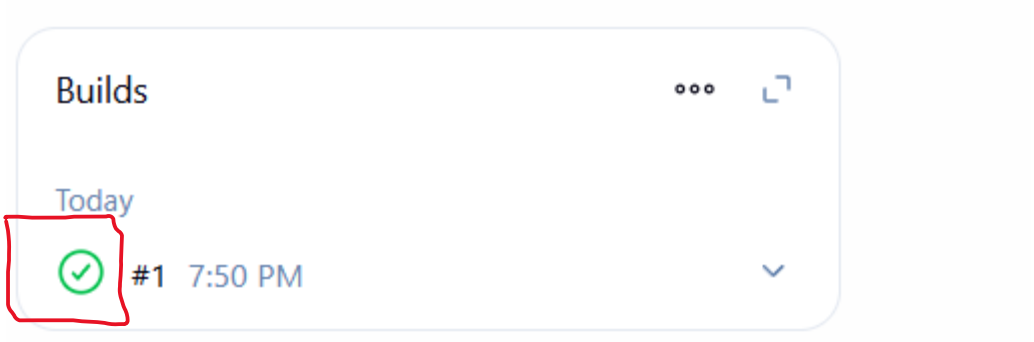
✓ Save and build the project



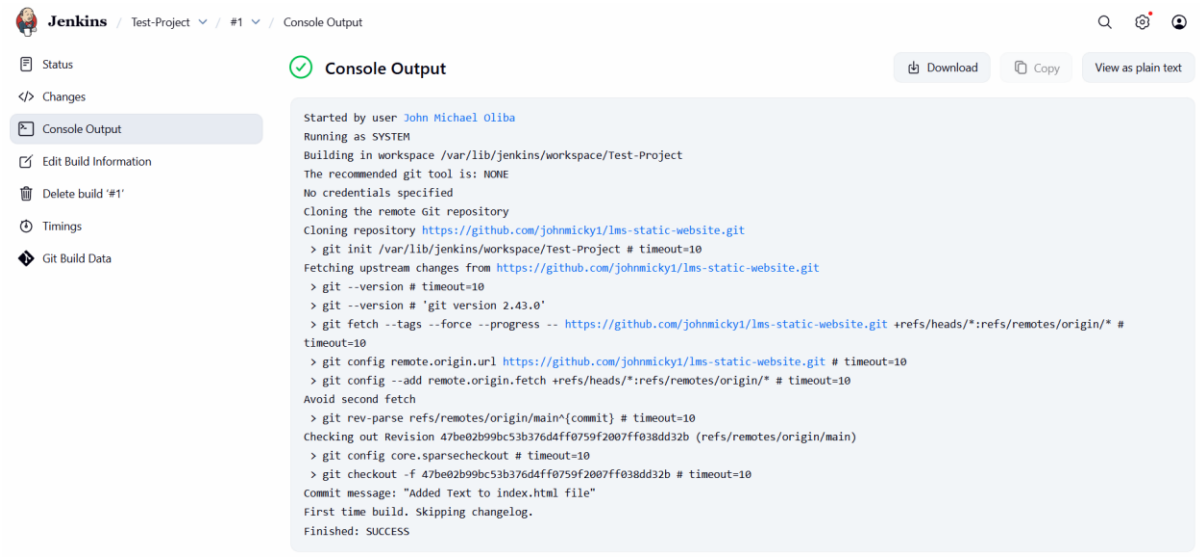
✓ Click Save to save the project configuration.
✓ Click Build Now to trigger a build manually.



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.



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

DevOps LMS – Master Github & Jenkins

Commit and push the changes to GitHub.

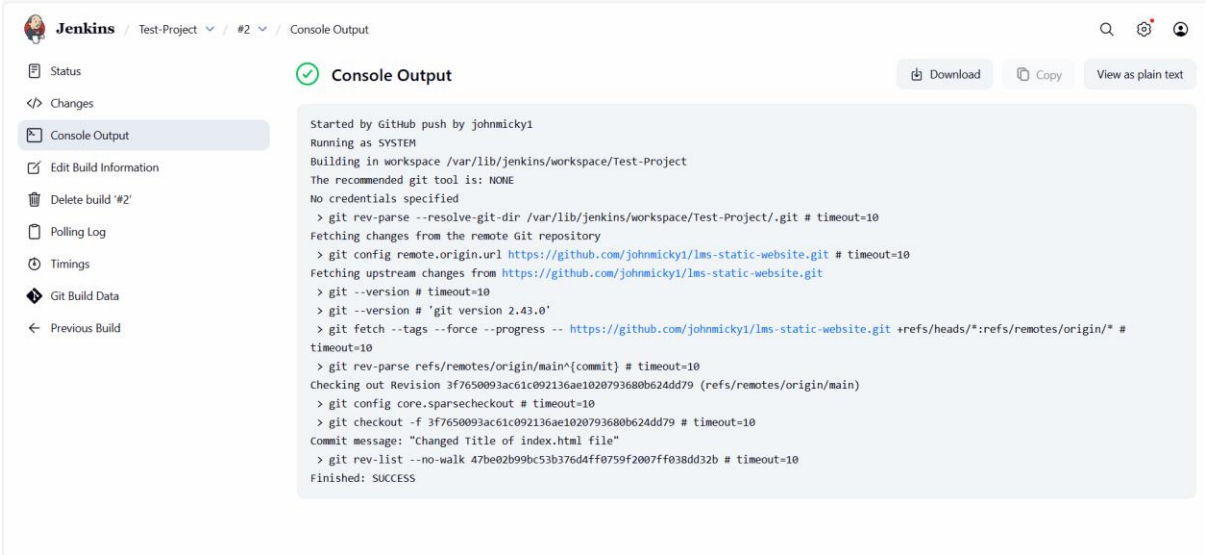
```
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website>
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> git add .
PS C:\Users\johnm\OneDrive\Desktop\lms-static-website> git commit -m "Changed Title of index.html file"
[main 3f76500] Changed Title of index.html file
 1 file changed, 1 insertion(+), 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.



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