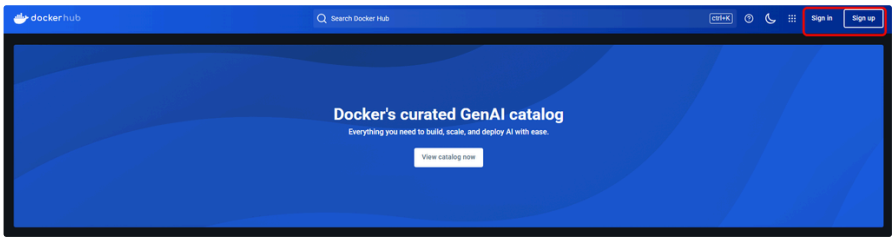
Terraform Deployment in Jenkins

PREQUISITIES

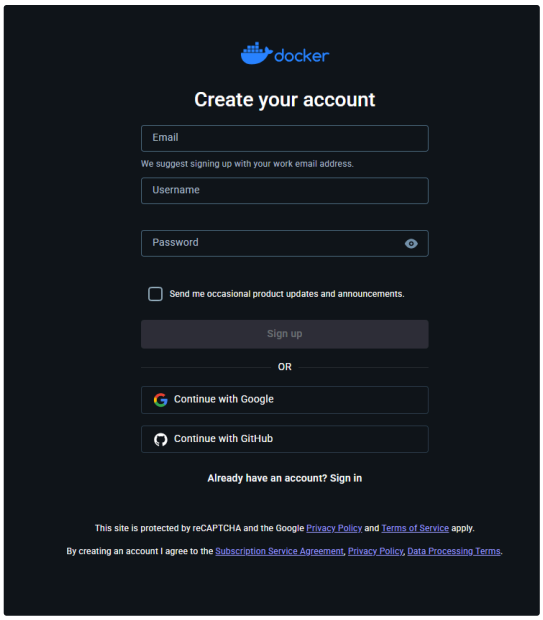
- ***Docker Desktop*** installed on your local machine
- ***Docker Container*** and ***Engine*** up and running
- ***Jenkins Username and Password*** stored in your Notepad
- ***AWS IAM User & Credentials*** for Jenkins (Access Key & Secret Access Key)
- ***GitHub Repo with "Jenkinsfile"*** SAMPLE (GitHub - LarvariousM/JenkinsExperiment1: Docker Experimentation with Jenkins)

SIGN UP FOR DOCKER HUB AND INSTALL DOCKER DESKTOP

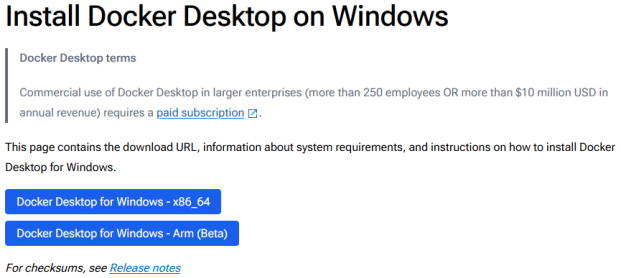
Sign up for sign in to docker hub - Docker Hub Container Image Library | App Containerization



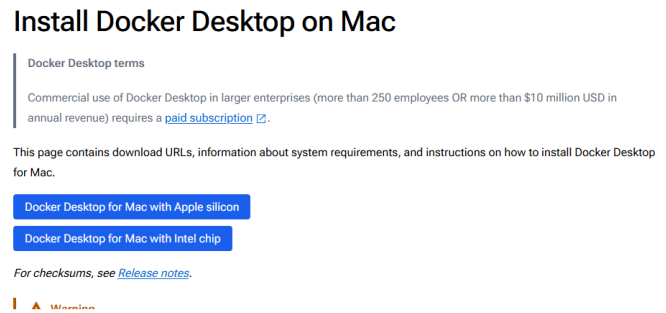
Create docker account - https://app.docker.com/signup



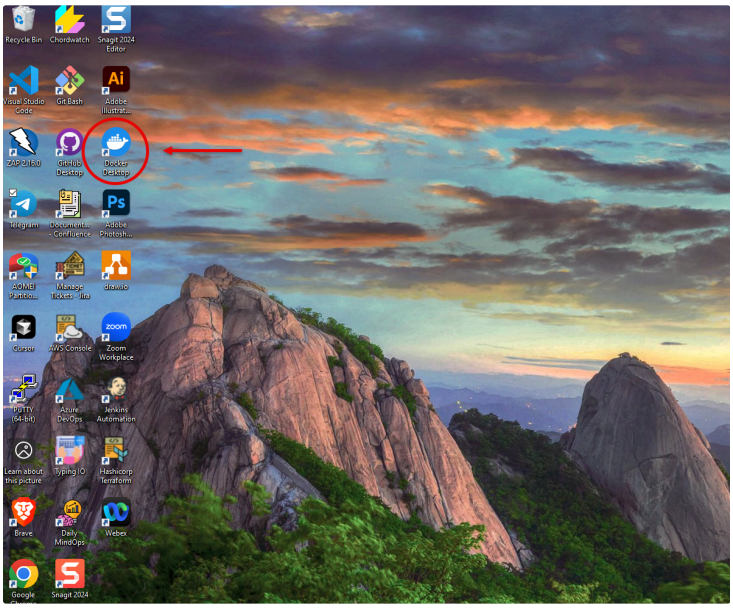
Install docker desktop - Windows = https://docs.docker.com/desktop/setup/install/windows-install/



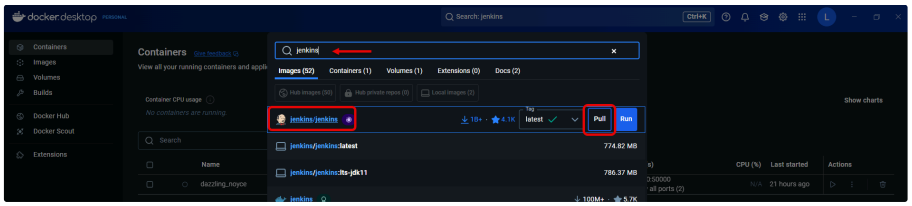
Install docker desktop - Mac Mac



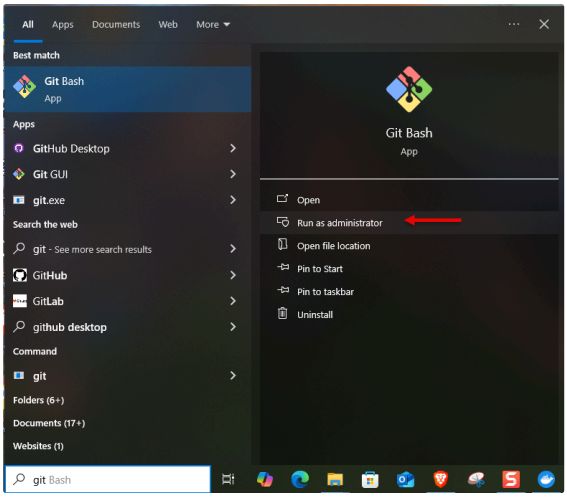
Open Docker Desktop



Search for Jenkins in search bar, look for jenkins/jenkins, and pull

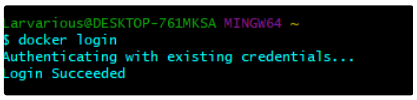


Open GitBash and run as administrator



Type in this command to login to docker

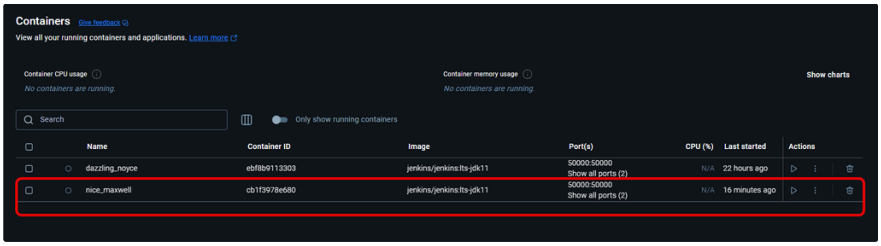
```
docker login
```



Run this command in GitBash

```
docker run -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk11
```

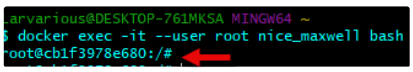
Previous code creates new docker container for you



Click on Start button to start the docker containter.

Return to GitBash and run this command to login as root user, we need to find your password for further installation

```
docker exec -it --user root <your container name> bash
```



We need to navigate to this folder as root user to find our password.

```
ini/var/jenkins_home/secrets/initialAdminPassword
```

Once you find your password in this file, copy it and store it in your Notepad. It will look similar to this image below. 👇

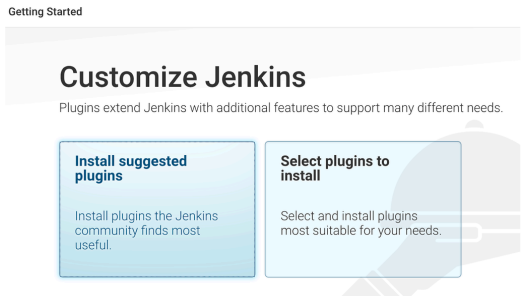
```
206z1eacdc1d4ac1f1f565768za5f20G
```

Once you have your password ready, open up your favorite browser and enter http://localhost:8080 into the address bar

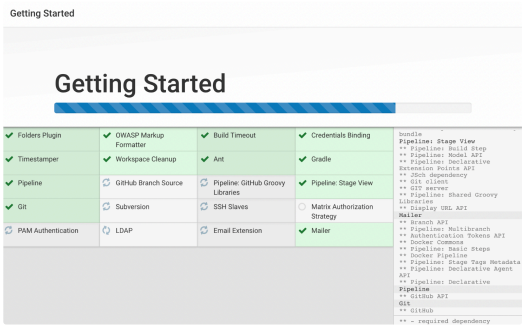
You will be brought to this next screen to Unlock Jenkins, enter your Administrator password you just put into your Notepad.

After putting in your Administrator password, you will brought to a webpage to create a User Name and Password for Jenkins. Go ahead and create one and store those also in your Notepad!

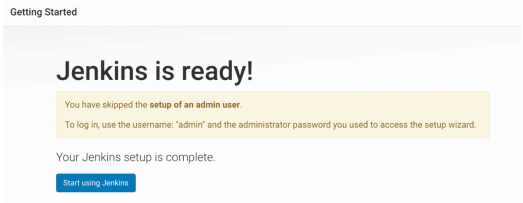
On the next page, the Customize Jenkins webpage. Choose 'Install suggested plugins'.



Suggested plugins will commence to installing.



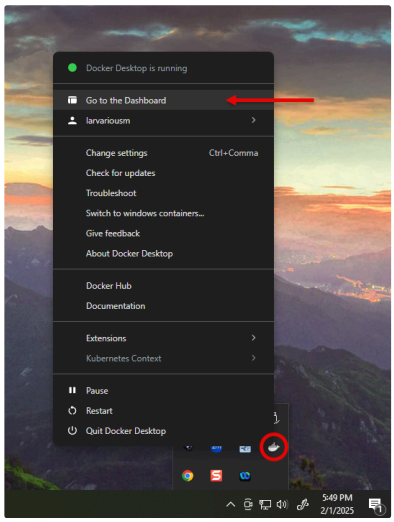
Once all plugins have been installed, your Jenkins is ready!



STARTING YOUR DOCKER CONTAINER AND ENGINE

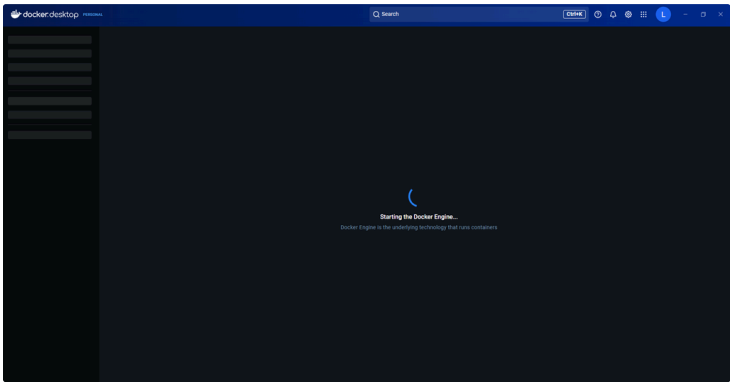
Locate the ***Docker Desktop*** icon on your desktop and double click on it.

If Docker Desktop doesn't load immediately on your screen, go to the bottom right hand side of your desktop and click the upwards arrow to show hidden icons running in the background of your computer, it will look like this.

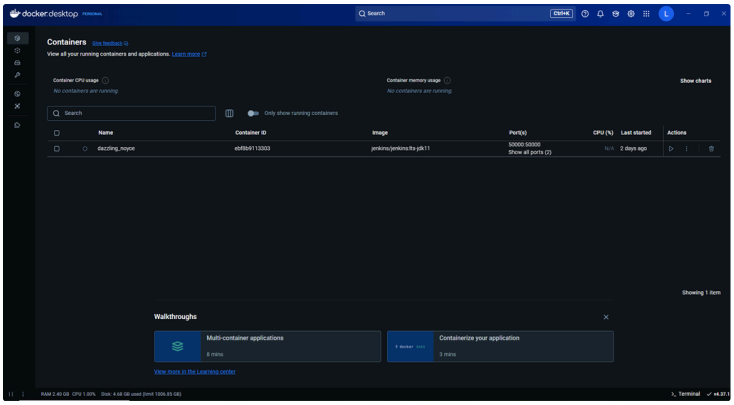


Either double-click the Docker Desktop icon, or right-click the icon to bring up a prompt and click on GO TO DASHBOARD.

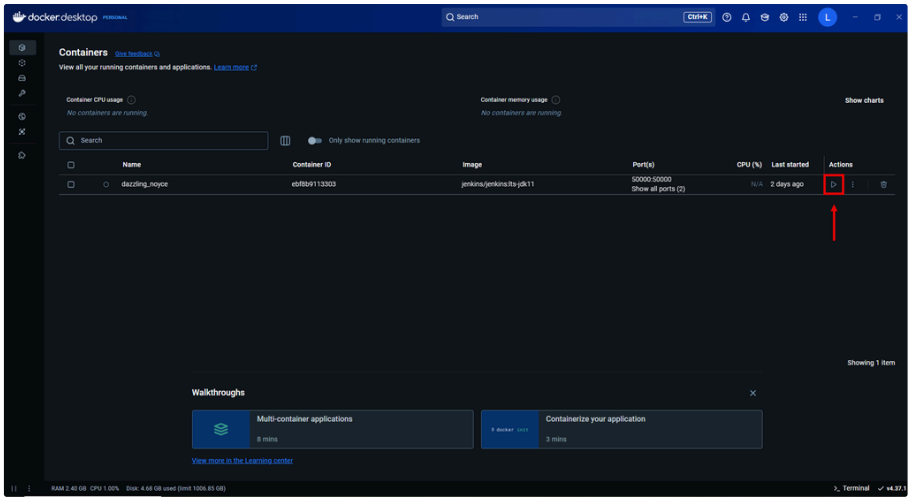
Once you've open Docker Desktop, the engine should automatically start running. 👇



Once the ***Docker Engine*** has started, you should be at your ***Docker Dashboard***. It should look like this image below 👇



Press The START Button under Actions to start your Docker Container



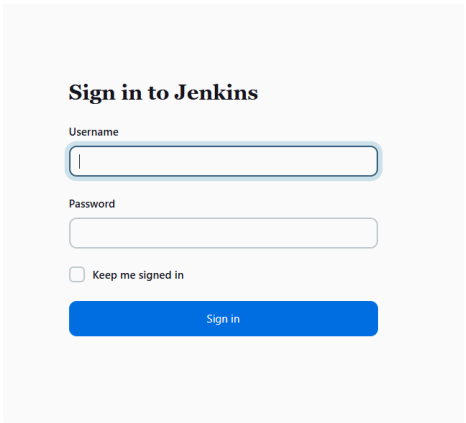
- Screenshot of Docker Engine running, and Docker Container running -

LOGIN TO JENKINS LOCALHOST

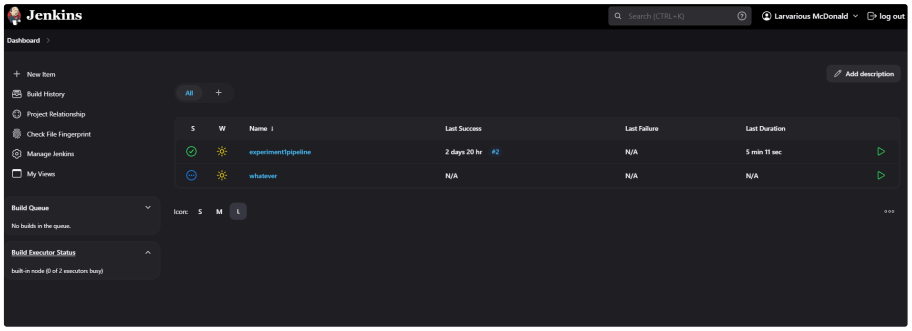
Now that we have Docker Engine and a Docker Container running simultaneously, we need to go to our Jenkins localhost, login and install some much needed plugins! 😀

In order to do that, open up your favorite browser, and in the address bar type in http://localhost:8080

After you've typed in http://localhost:8080, it should bring you to the glorious ***Jenkins Sign In Page***

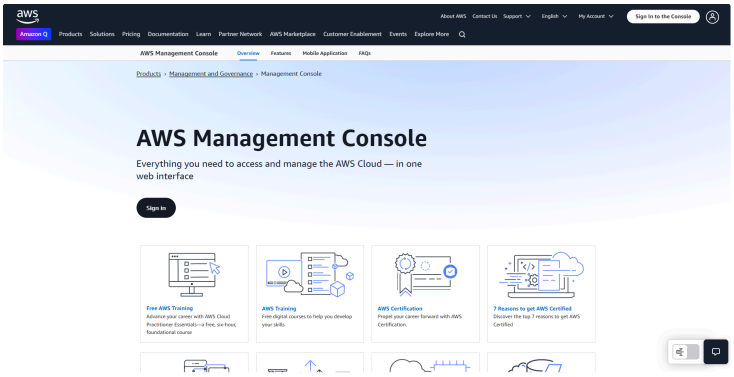


After you've signed in, it should bring you to the Jenkins Dashboard.

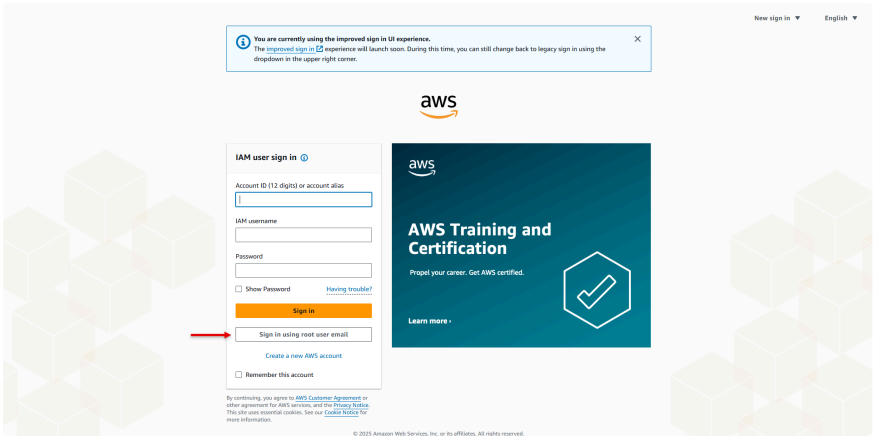


Now, we need to configure ***AWS Credentials*** into Docker. We'll need to go to your AWS Account, go to IAM, Users, create a New User for Jenkins, and create new Access and Secret Access Keys.

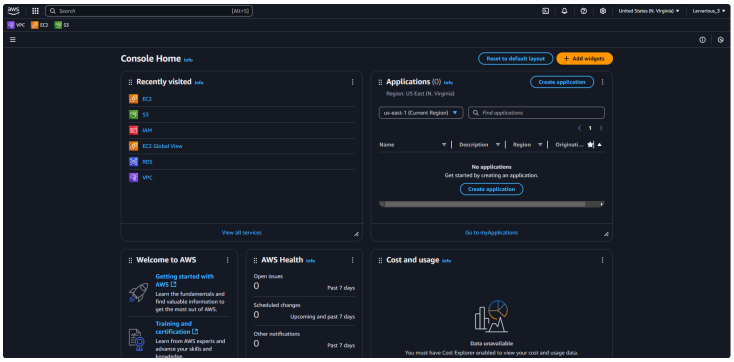
Go to AWS Management Console, and click on Sign In 👉 Manage AWS Resources - AWS Management Console - AWS



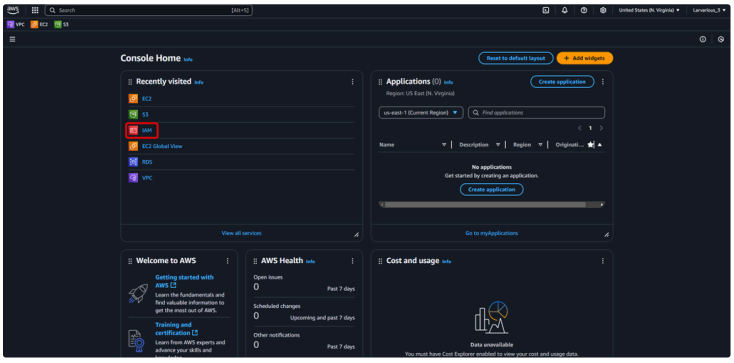
Click on "Sign in using root user email"



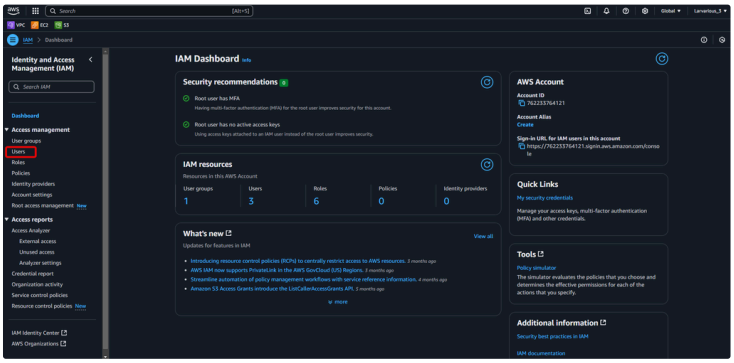
Once you've signed in to AWS using your email address, password, and MFA Authentication number(**If you don't have a MFA Authentication setup yet, please do so right now or bad things will happen!**). It will bring you to the Console Homepage that looks like this!



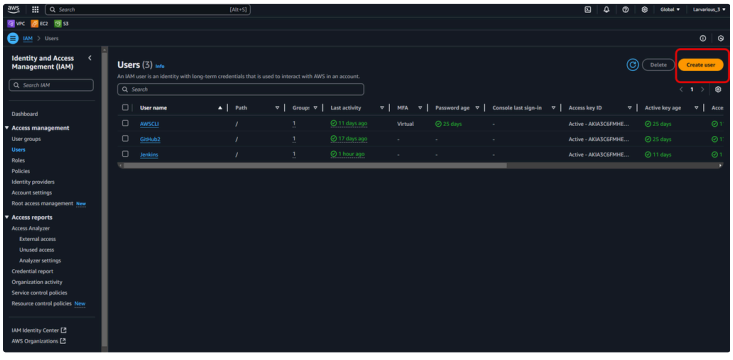
Either go to the Search bar up top and type in IAM, or click on under Recently visited.



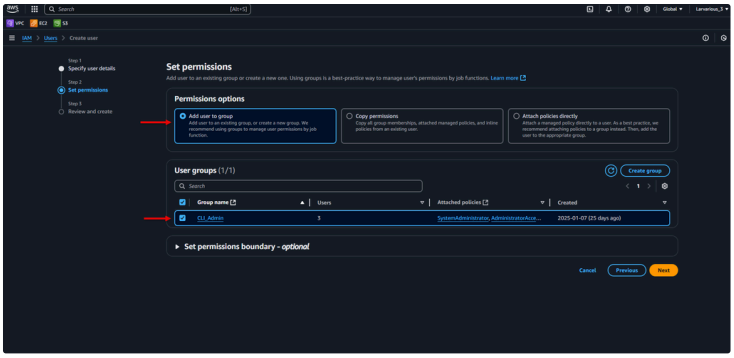
Click on Users, under Access management.



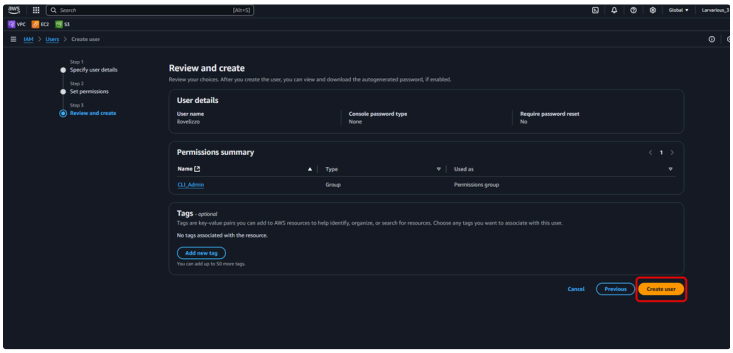
Click on Create user.



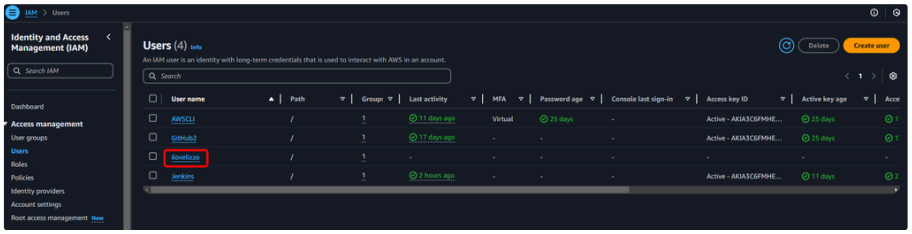
Give your username a crazy name you'll never forget, make sure to not add any spaces unlike I did.



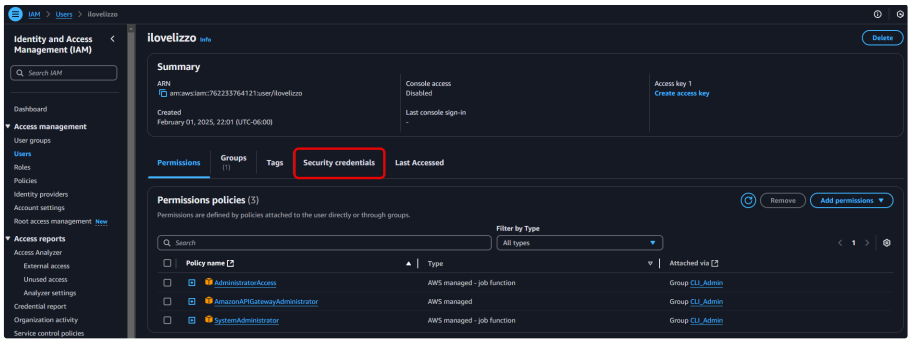
Let’s add this new user to a group I’ve already created.



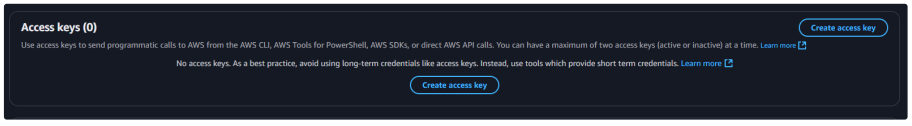
Click on Create user.



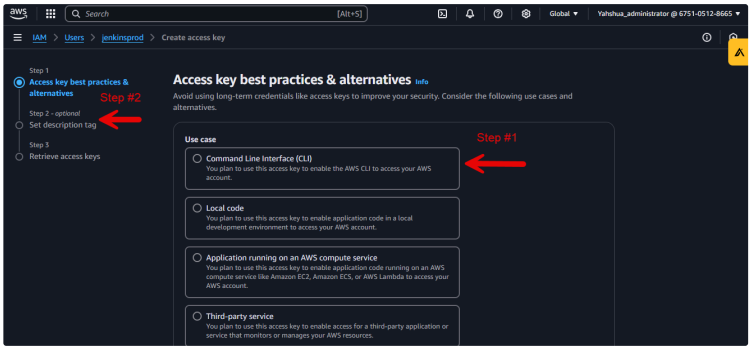
Click on the user you’ve just created.



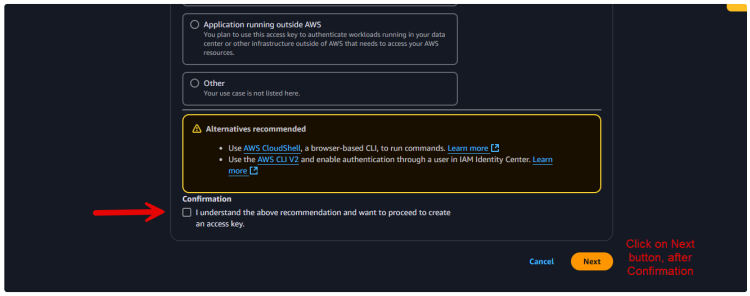
Click on Security credentials, we’ve got to give this user some ***G14 Classified*** assignments soon! 😎



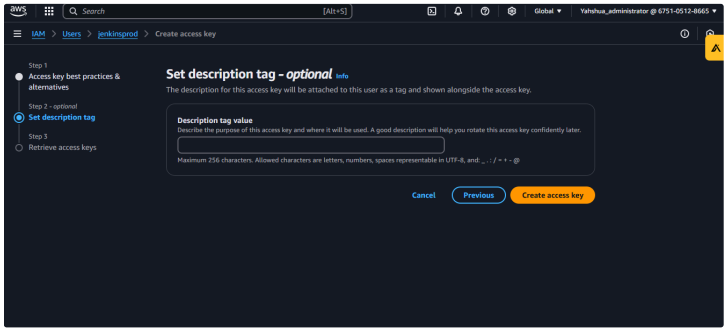
Scroll down on the next page and click on Create Access Key, and it will bring you to this next page.



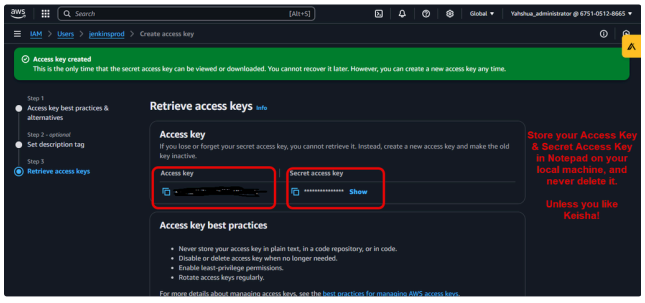
Click on Command Line Interface (CLI).



And choose, “I understand the above recommendation and want to proceed to create an access key.“, then click the Next button.



Click on create access key.

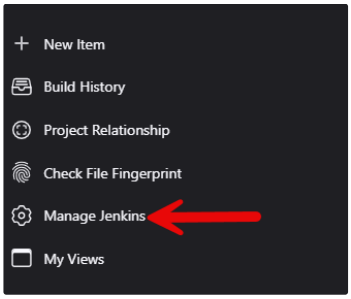


A new Access key and Secret access key will be created. Store your Access key and Secret Access Key in your Notepad on your PC, and never delete it! Unless you like Keisha or Lizzo!

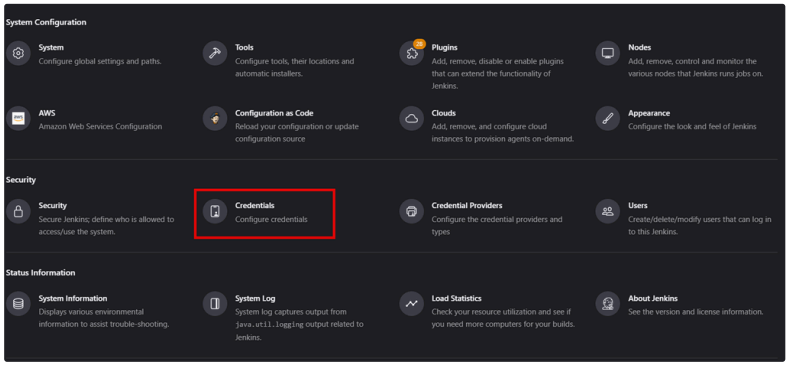
INSERTING AWS CREDENTIALS INTO JENKINS LOCALHOST

Hey, are you still here with me? Are you okay? Just remember….everything is fine. We’re almost not halfway there. Just follow the Confluence!

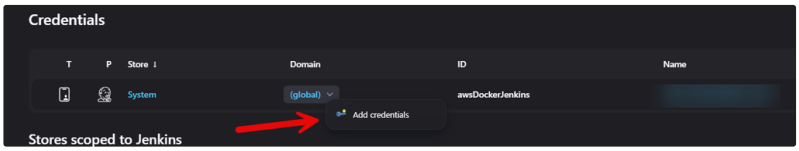
Now, we're going to add these AWS Credentials into our Jenkins localhost. Go back to your Jenkins Dashboard, and click on Manage Jenkins.



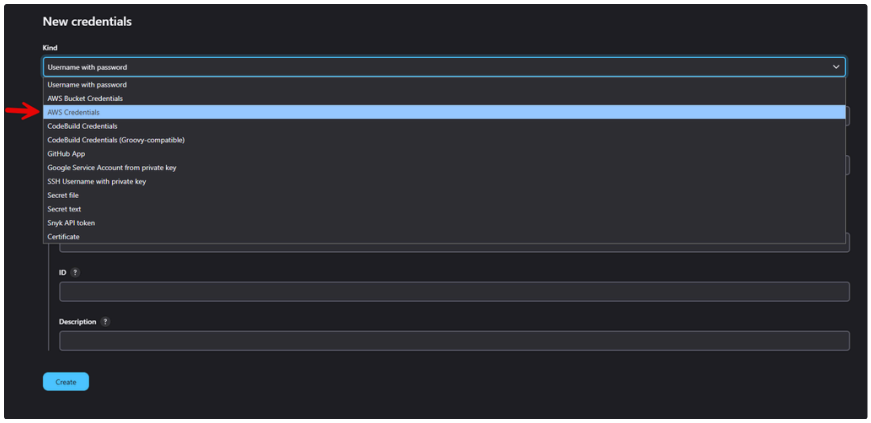
Under Security, click on Credentials.



After you click on Credentials, under Domain, we're going to hover over (global) and click on the drop down arrow, and select Add credentials.



Under Kind, change "Username with password" to AWS Credentials.



In the ID field, you're going to create a random ID, insert your AWS Access Key & Secret Access Key ID, and click the Create button. **Write down this random ID...you're gonna need it later! TRUST ME**

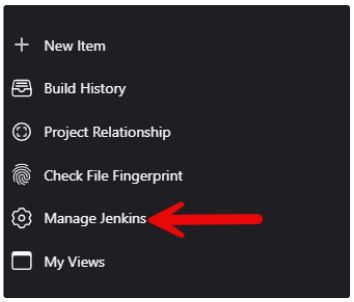
INSTALLING PLUGINS TO JENKINS

We need to install some plugins to Jenkins localhost to make this project work.

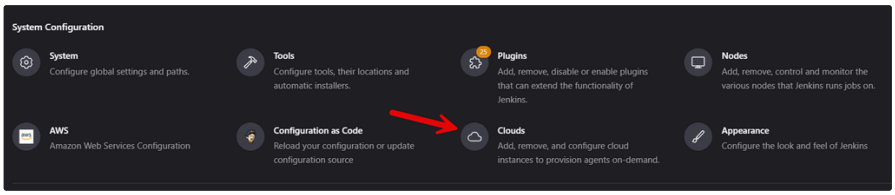
Here is the list of plugins to install:

1. AWS
2. AWS Credentials
3. Pipeline
4. Amazon EC2
5. Amazon Elastic Container Service
6. AWS Code Deploy
7. AWS Lambda
8. AWS S3 Bucket Credentials
9. AWS CodeBuild
10. AWS CodePipeline
11. AWS Secrets Manager Secret Source
12. Configuration as Code
13. CloudFormation
14. AWS SAM
15. Terraform
16. Google
17. Kubernetes
18. Google Cloud Storage
19. Google Cloud SDK
20. Pipeline
21. Snyk
22. SonarQube Scanner
23. Aqua - all 3
24. GitHub

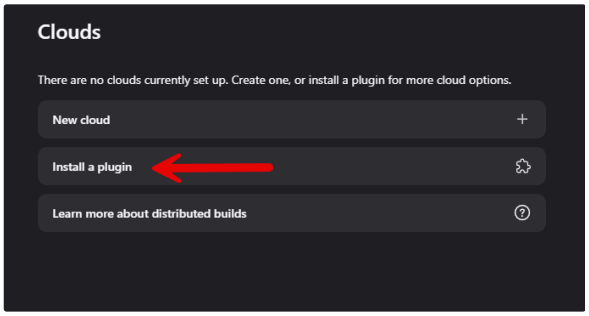
25. GitHub Integration
26. GitHub Authentication
27. Pipeline GitHub
28. Pipeline GitHub Notify Step



1. Go to the dashboard in your Jenkins, and click on **Manage Jenkins**.

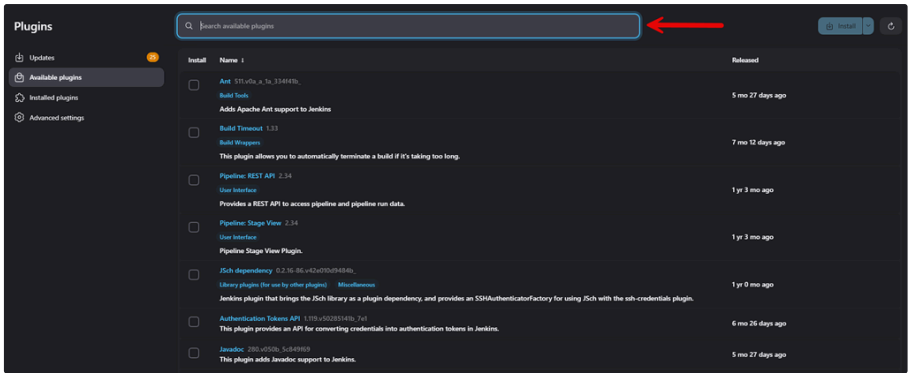


2. Click on **Clouds**, under System Configuration.



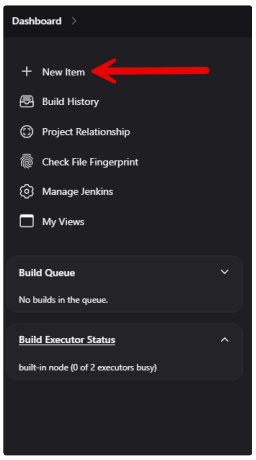
3. Under Clouds, click on **Install a plugin**

This page shows a list of all plugins to add to your Jenkins localhost, type in the name of each plugin and install each one.

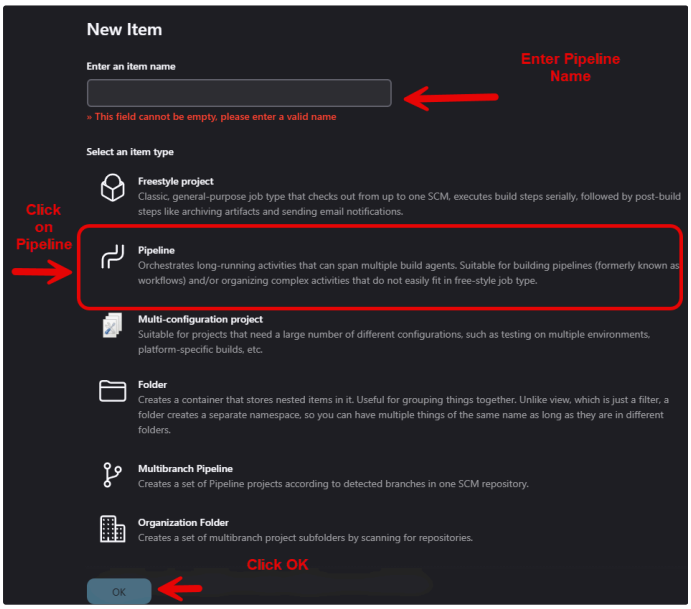


Once you're done adding all the plugins to the Jenkins localhost.

Now that we've configured our AWS Credentials into Jenkins and we've installed the necessary plugins, go back to your Jenkins Dashboard and select New Item.



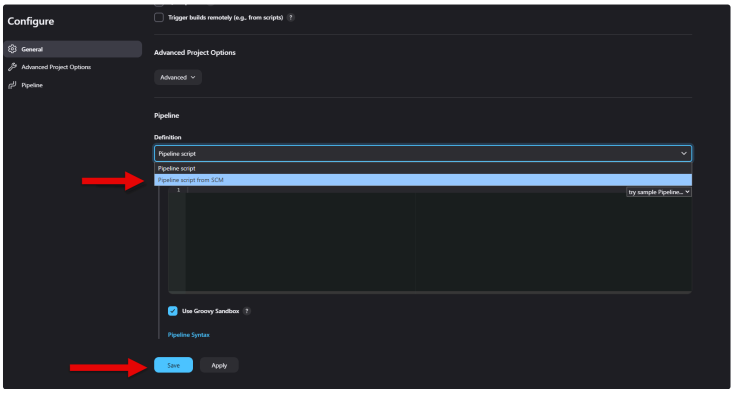
Create a Item Name, select Create Pipeline, and click OK at the bottom of the page.



I've created an Item named 'whatever', it should bring you to a page like this. 👇

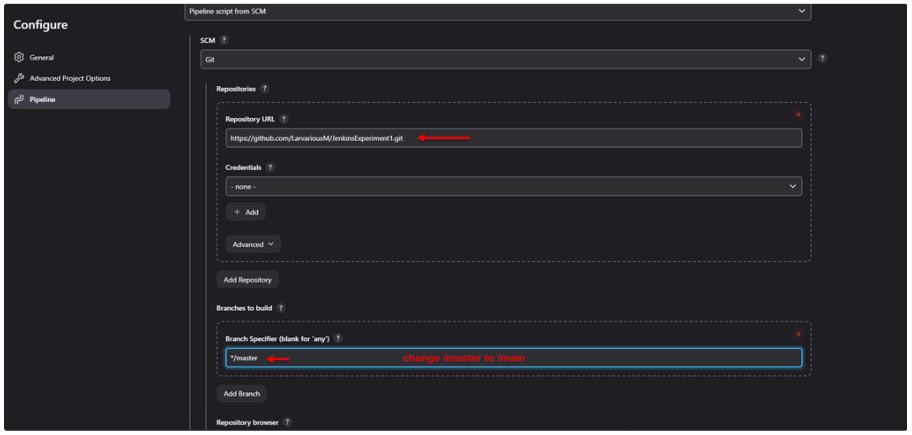
Now we have to configure this Item, which is our Pipeline. So that we can run our terraform code stored in our GitHub repository. **You do still have that link, right?** 😐 If so, click on the configure button.

Scroll down this page until you see Pipeline, change the Definition from “Pipeline script“ to “Pipeline script from SCM“, but don’t SAVE just yet!

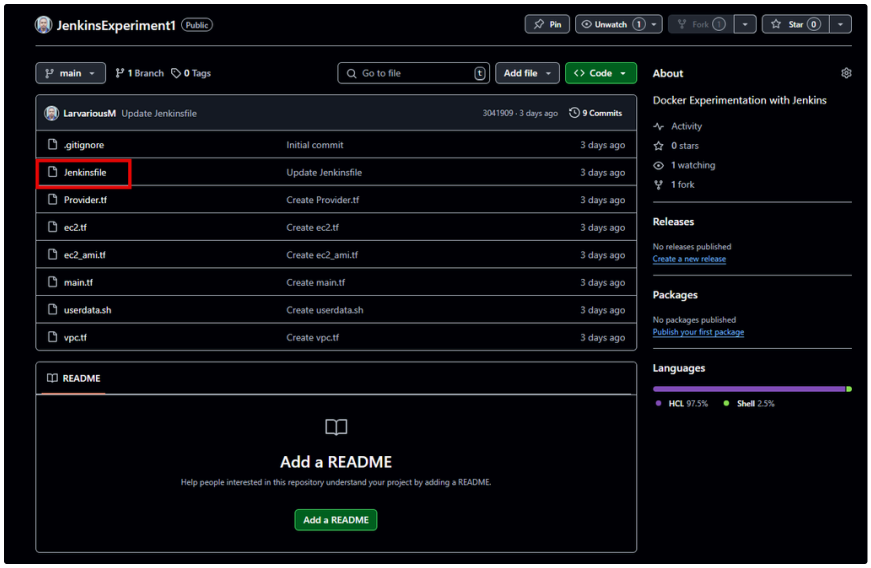


Under SCM, change “None“ to “Git“.

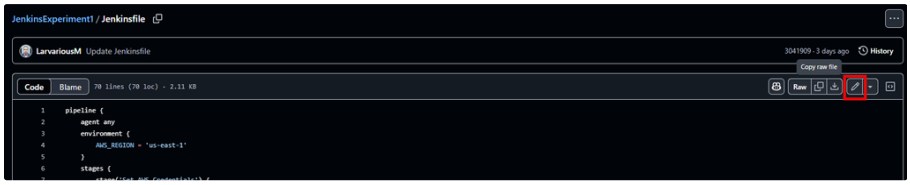
Scroll down, under Repositories. Insert that GitHub repository from the top of this tutorial page into the Repository URL. Under Branch Specifier, change master to */master to */main. Scroll down to the bottom, and click on Save.



Go to that GitHub Repository, fork it to your repositories. Then click on Jenkinsfile. We have to make some edits in that file for this Terraform deployment to work in Jenkins.



Once you've selected the Jenkinsfile, click on the Pencil icon to edit the file.



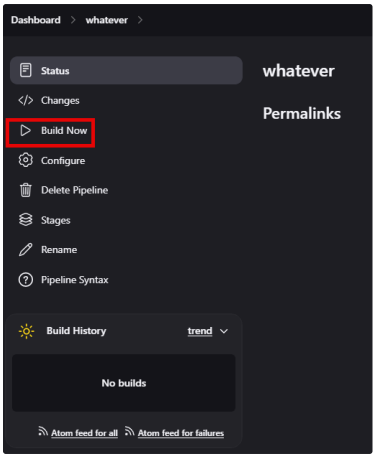
We're going to edit Lines 11, 36, and 51 with our credentialsID we created in the Jenkins credentials, remember??? Also, Line 22 should point to the main branch in GitHub.

```
pipeline {
    agent any
    environment {
        AWS_REGION = 'us-east-1'
    }
    stages {
        stage('Set AWS Credentials') {
            steps {
                withCredentials([[
                    $class: 'AmazonWebServicesCredentialsBinding',
                    credentialsId: 'INSERT JENKINS CREDENTIALS HERE'
                ]]) {
                    sh '''
                    echo "AWS_ACCESS_KEY_ID: $AWS_ACCESS_KEY_ID"
                    aws sts get-caller-identity
                    '''
                }
            }
        }
        stage('Checkout Code') {
            steps {
                git branch: 'main', url: 'https://github.com/LarvariousM/JenkinsExperiment1'
            }
        }
        stage('Initialize Terraform') {
            steps {
                sh '''
                terraform init
                '''
            }
        }
        stage('Plan Terraform') {
            steps {
                withCredentials([[
                    $class: 'AmazonWebServicesCredentialsBinding',
                    credentialsId: 'INSERT JENKINS CREDENTIALS HERE'
                ]]) {
                    sh '''
                    export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID
                    export AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY
                    terraform plan -out=tfplan
                    '''
                }
```

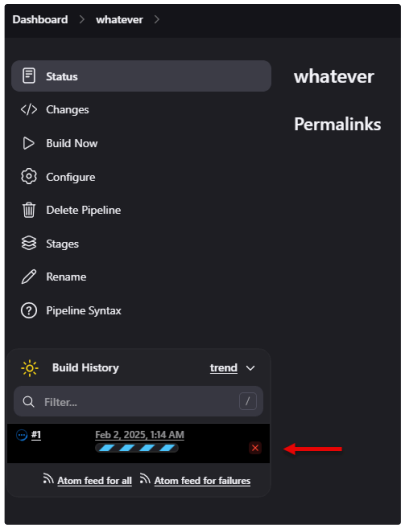
```
            }
        }
        stage('Apply Terraform') {
            steps {
                input message: "Approve Terraform Apply?", ok: "Deploy"
                withCredentials([[
                    $class: 'AmazonWebServicesCredentialsBinding',
                    credentialsId: 'INSERT JENKINS CREDENIALS HERE'
                ]]) {
                    sh '''
                    export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID
                    export AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY
                    terraform apply -auto-approve tfplan
                    '''
                }
            }
        }
    }
    post {
        success {
            echo 'Terraform deployment completed successfully!'
        }
        failure {
            echo 'Terraform deployment failed!'
        }
    }
}
```

Once you're done making those changes, click on Commit Changes.

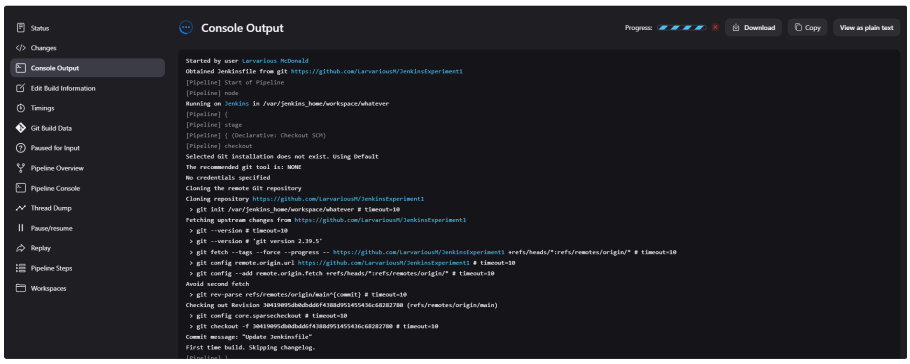
Let's go back to Jenkins localhost and build our first Pipeline. Go to your Item you've created, and click on Build Now.



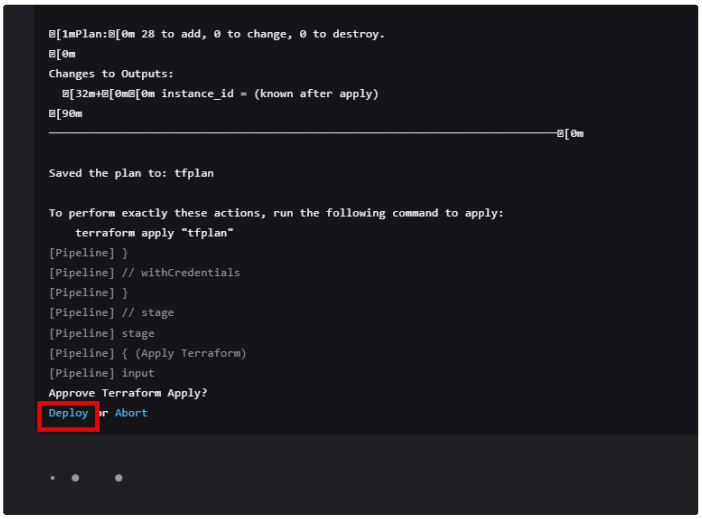
Click on the progress bar to show your current build in progress.



Once you've clicked on the progress bar, it should bring you to the Console Output page. Scroll all the way down the page. I wanna show you some cool stuff, if you've been following the Confluence correctly! 😎



My Terraform code has successfully INIT, VALIDATED, and PLANNED, and now wants my decision to Deploy 28 resources. Click on Deploy to build.

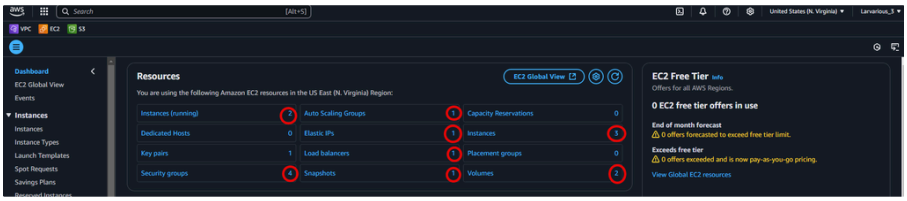


My Terraform Deployment in Jenkins is a success. We'll also check in the AWS Console to verify the resources has been created.

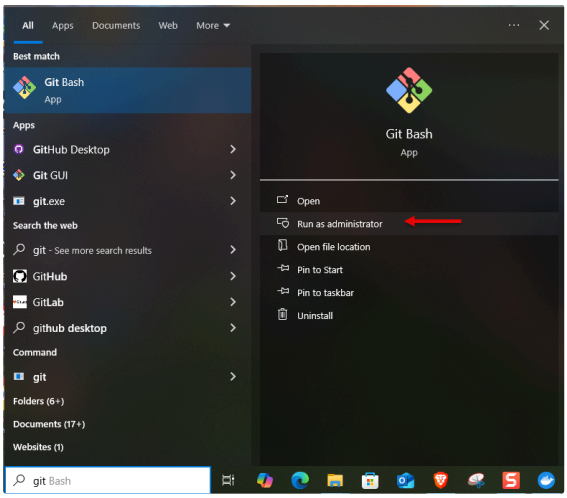
```
Apply complete! Resources: 28 added, 0 changed, 0 destroyed.
[0m[0m[1m[32m
Outputs:

[0minstance_id = "i-047769a9d6d9a7adf"
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] echo
Terraform deployment completed successfully!
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Resources that were created in Amazon Web Services Console, from the Terraform Deployment in Jenkins.

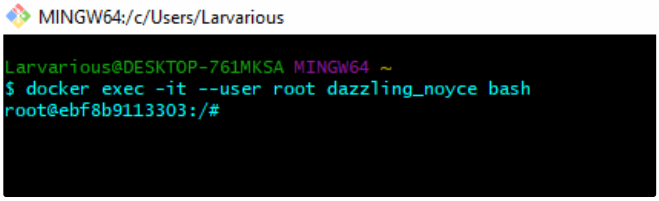


Now let's Terraform Destroy these resources from Git or PowerShell, whichever you choose. I'll open up Git Bash, and run it as an Administrator.



Once you're in Git Bash, run this command.

```
docker exec -it --user root YOUR CONTAINER NAME bash
```



Now, we have to cd aka Change Directory into the directory with terraform files. You can navigate the folders by using LS & CD commands. We have to navigate to the destination below: 👇

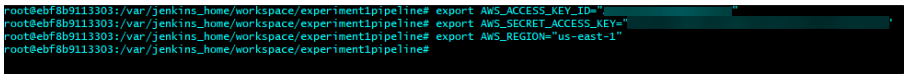
```
/var/jenkins_home/workspace/<PIPELINENAME>
```

As you can see on the screen,

cd → var, cd → jenkins_home, cd → workspace, cd → pipelinename

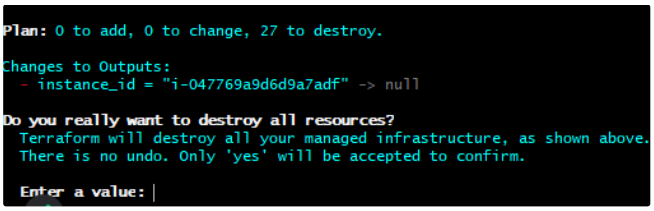
Now, you're going to enter in your key information into GitBash.

```
export AWS_ACCESS_KEY_ID="YOUR ACCESS KEY"
export AWS_SECRET_ACCESS_KEY="YOUR SECRET ACCESS KEY"
export AWS_REGION="YOUR AWS REGION"
```

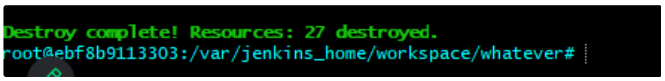


Now, enter Terraform Destroy into GitBash.

You will be prompted to enter a value, Yes or No, to terraform destroy these resources. Enter 'Yes', unless you got Diddy money.

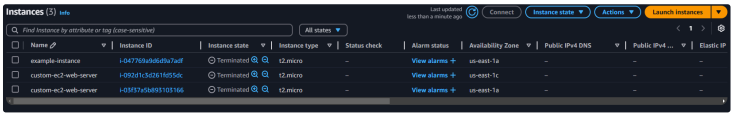


Terraform Destroy Completed

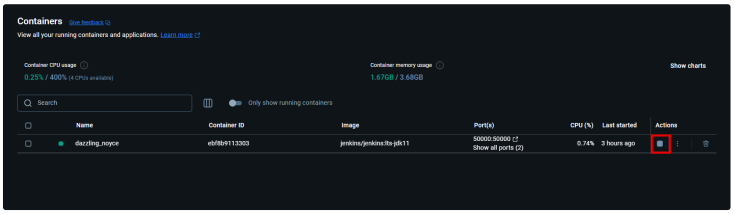


AWS Resource Gone!

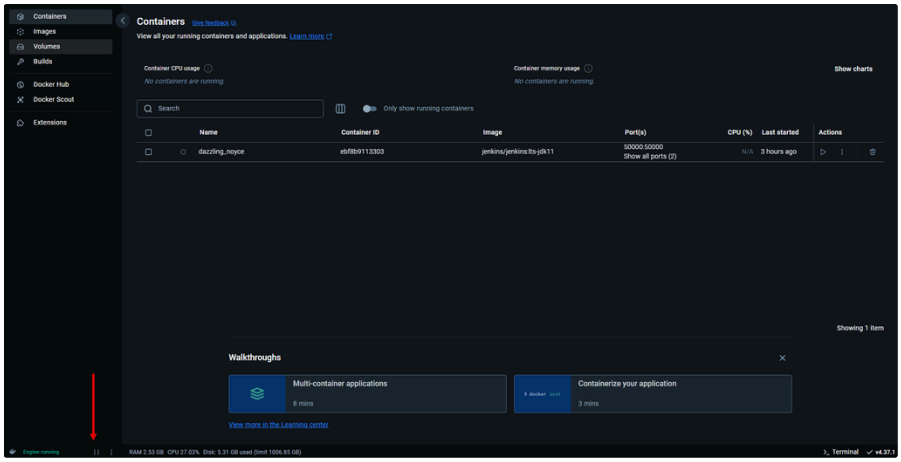
AWS Instances Terminated



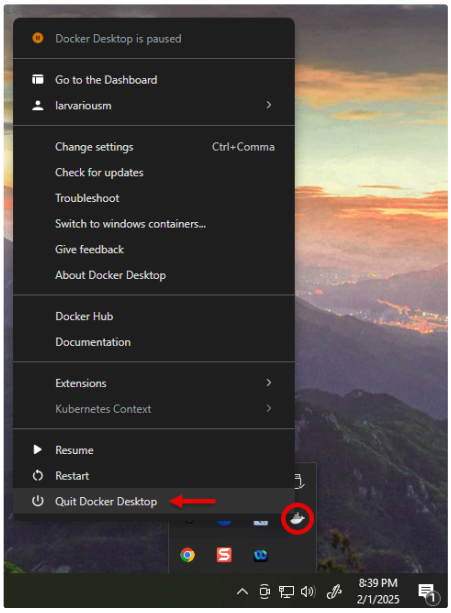
HOLD ON!!! WAIT! There's one more thing! ☹️ Let's stop our Docker Container!



Then, let's pause our Docker Engine.



Then, we can safely exit and close out Docker Desktop, but to be sure let's Quit Docker Desktop. So, on your desktop click on show hidden icons on the bottom right-hand side of the screen. Right Docker Desktop, and let's click on 'Quit Docker Desktop'.



That concludes our tutorial for Deploying Terraform in Jenkins! That's all folks!