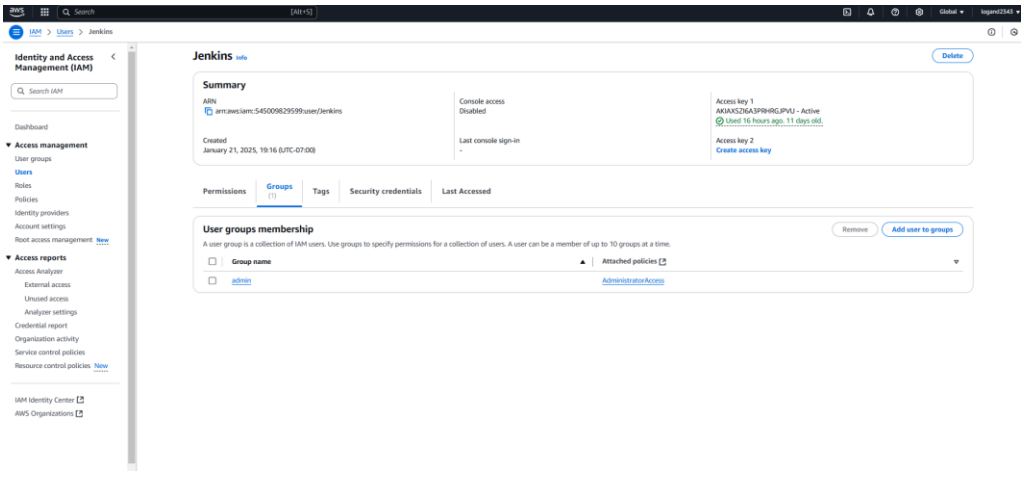
Jenkins Documentation

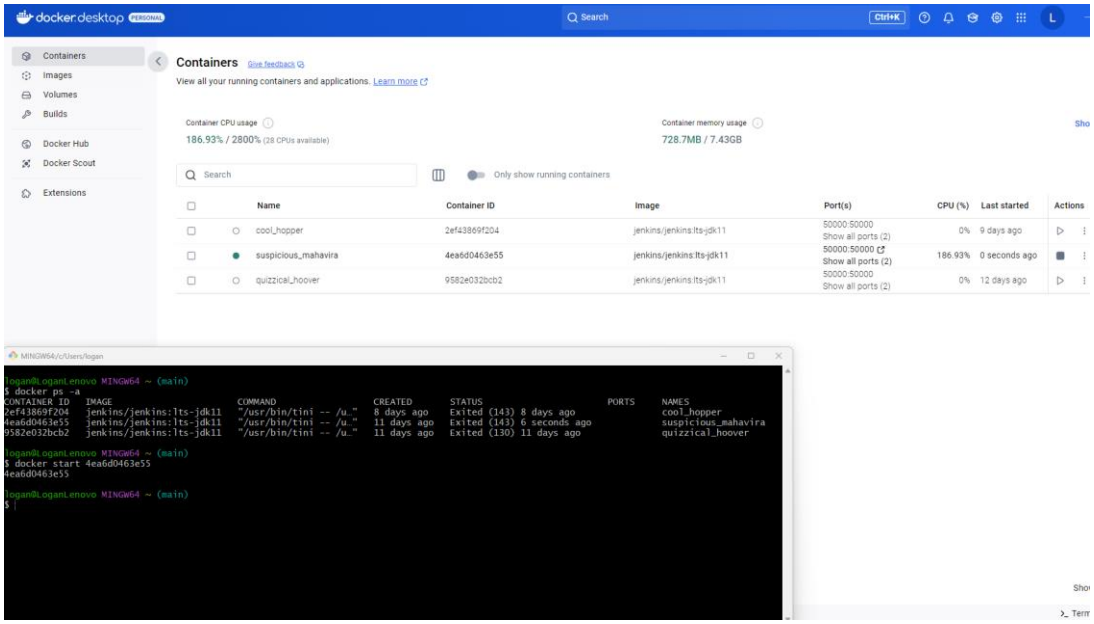
Overview Steps

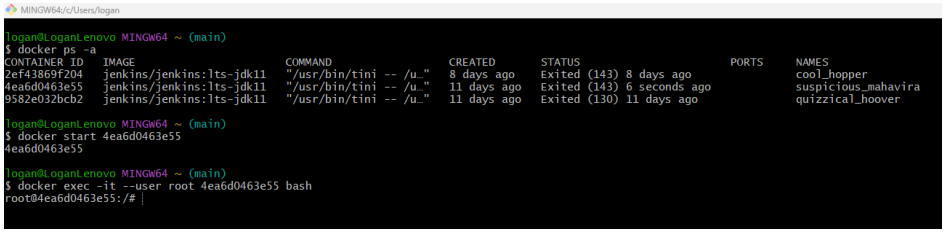
1. For Jenkins service, create a dedicated AWS IAM user with Access and Secret Keys.
2. Turn on Jenkins container via the CLI.
3. Within the CLI, install both Terraform and AWSCLI.
4. Access Jenkins in web browser via the localhost80:80 url and login with Jenkins credentials.
5. Add our Jenkins AWS IAM user keys to our Jenkins credentials settings.
6. Create new Github repository with working Terraform project and ensure the Jenkins file is properly configured and included in repository.
7. Create a new Pipeline in Jenkins and configure it to utilize our Github repository.
8. Run our new Pipeline in Jenkins to fire off the build of our Terraform project.
9. Confirm the Pipeline build was successful.
10. Perform Terraform teardown within our container in the CLI.

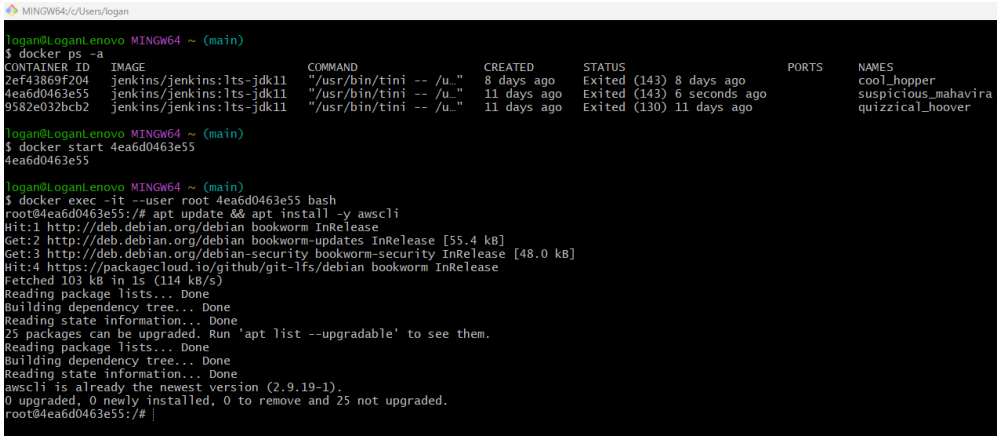
Steps Breakdown

1. For Jenkins service, create a dedicated AWS IAM user with Access and Secret Keys.
    a. Log into AWS and navigate to the IAM section.
    b. Create a new user for Jenkins.
    c. Assign the user to a group with administrator permissions.
    d. Create an access key and save the Access Key and Secret Key.
    e. Screenshots
        i. 

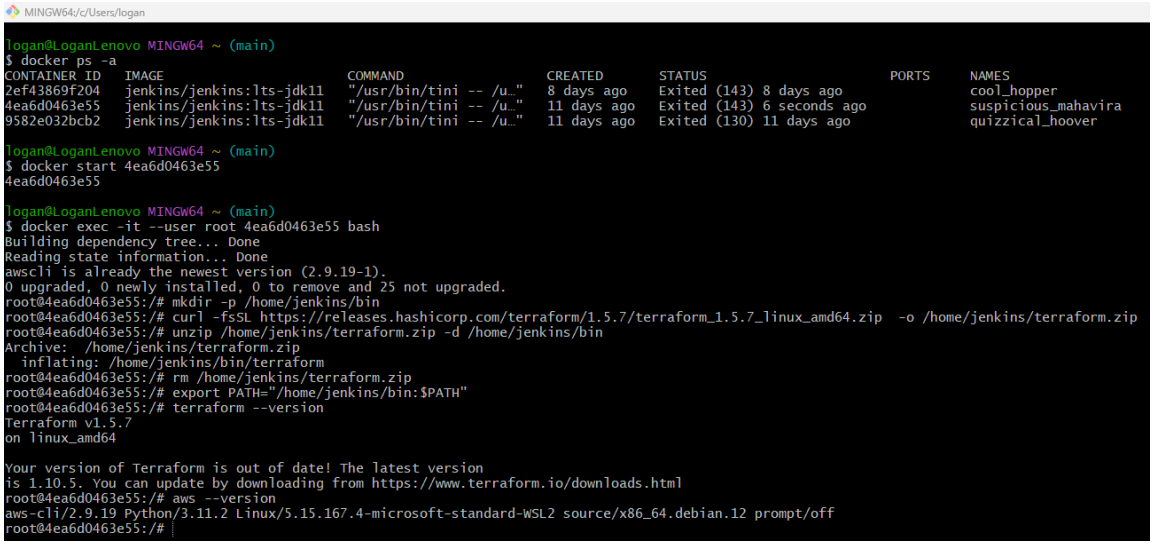
2. Turn on Jenkins container via the CLI.
    a. Open Gitbash

b. Run the docker command “docker ps -a” to get a list of all containers on local machine so we can retrieve the container id.
c. Run the docker command “docker start <container id>” to start the docker container.
d. 

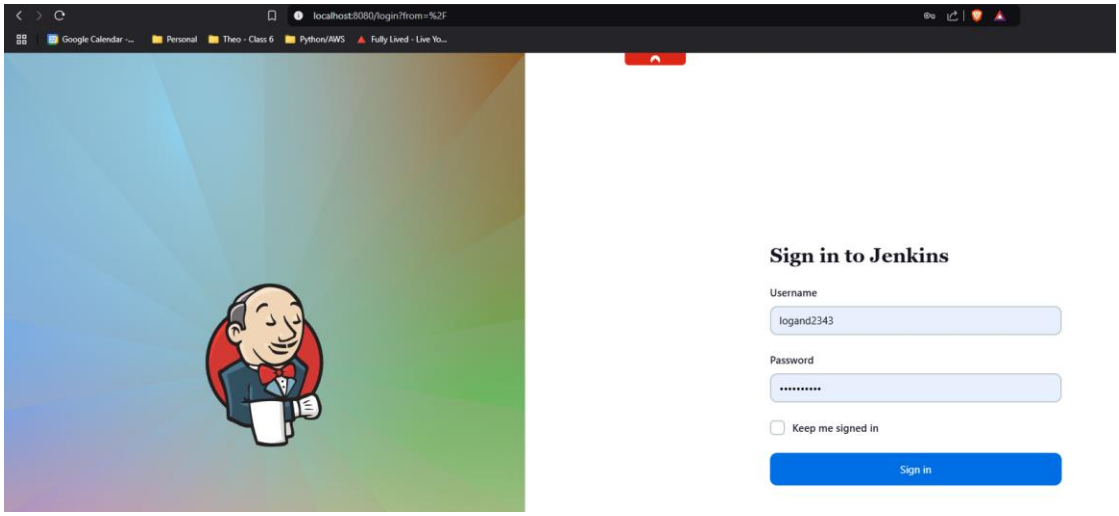
3. Within the CLI, install both Terraform and AWSCLI.
a. Run this docker command “docker exec -it --user root <container id> bash” to ssh into this running container.
i. 

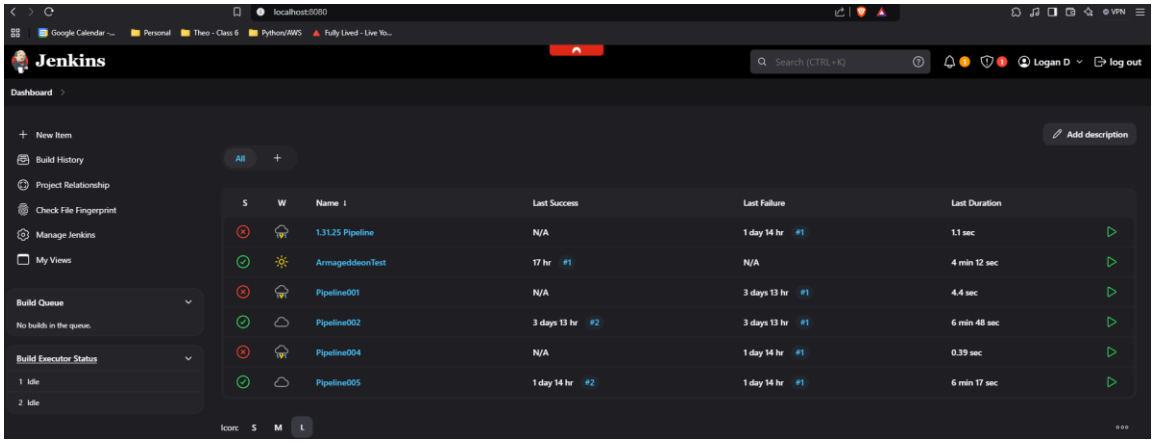
b. Run this docker command “apt update && apt install -y awscli” to update our container and install awscli.
i. 

c. Run this docker command “mkdir -p /home/jenkins/bin” to make a directory for Jenkins.
d. Run this docker command “curl -fsSL https://releases.hashicorp.com/terraform/1.5.7/terraform_1.5.7_linux_amd64.zip -o /home/jenkins/terraform.zip” to grab terraform from hashicorp and save the zip file inside our container.
e. Run this docker command “unzip /home/jenkins/terraform.zip -d /home/jenkins/bin” to unzip our terraform zip and place it in the Jenkins directory.
f. Run this docker command “rm /home/jenkins/terraform.zip” to remove the zip file since it is no longer necessary.
g. Run this docker command “export PATH="/home/jenkins/bin:$PATH" to let the container know where we installed terraform for future runs.
h. Run the following commands “terraform --version” and “aws --version” to ensure both are installed.
i.

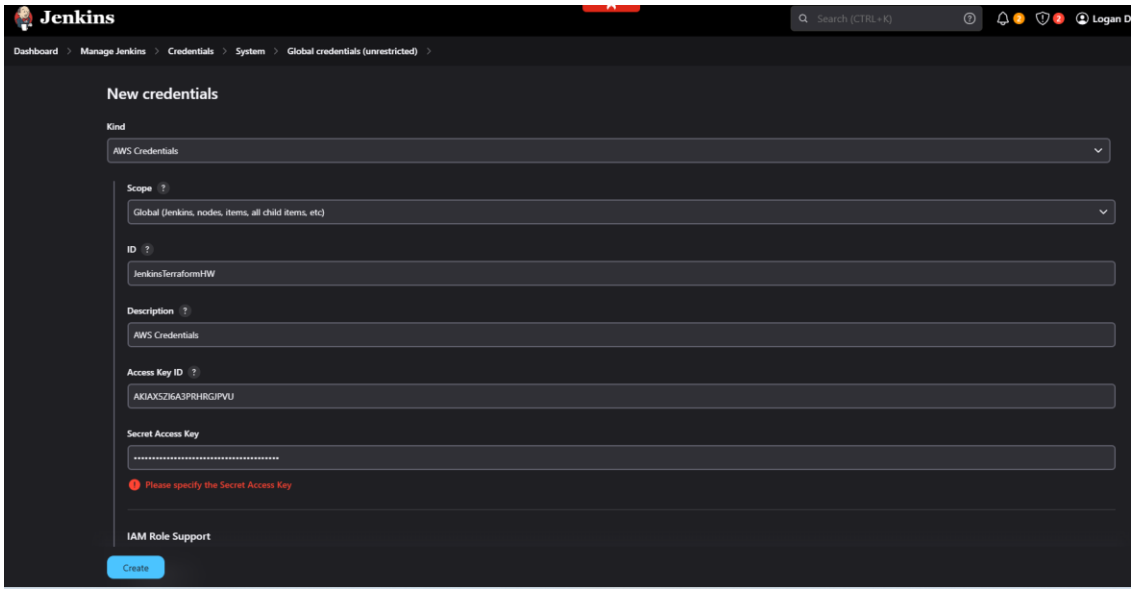


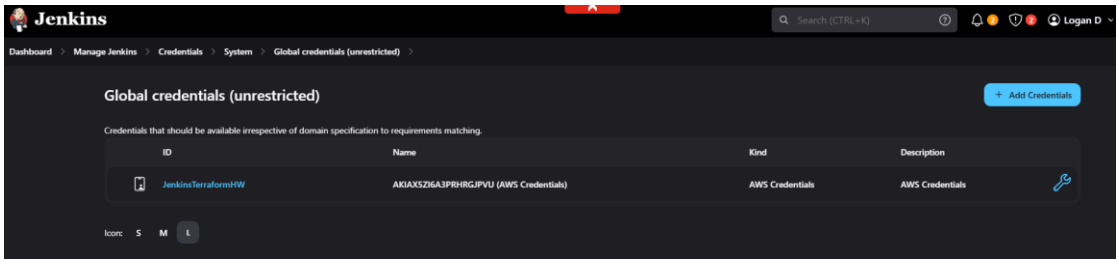
4. Access Jenkins in web browser via the localhost80:80 url and login with Jenkins credentials.

a. 

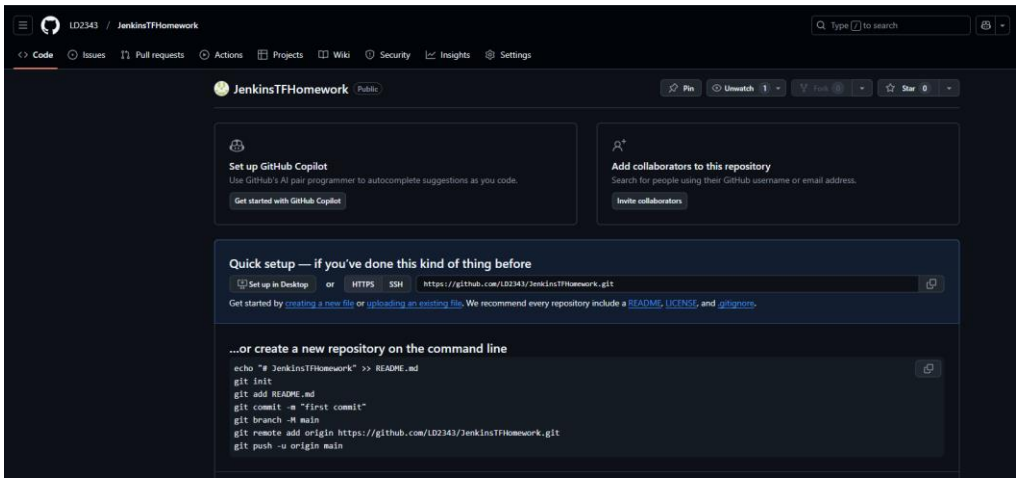
b. 

5. Add our Jenkins AWS IAM user keys to our Jenkins credentials settings.
    a. Inside Jenkins, go to Manage Jenkins and then Credentials.
    b. Click on System, then Global credentials (unrestricted) and then click on the Add Credentials button.
    c. For Kind select AWS Credentials. Add an ID label for this credential.
    d. Add the Access Key ID & Secret Access Key.
    e. Click Create to save credentials in Jenkins.

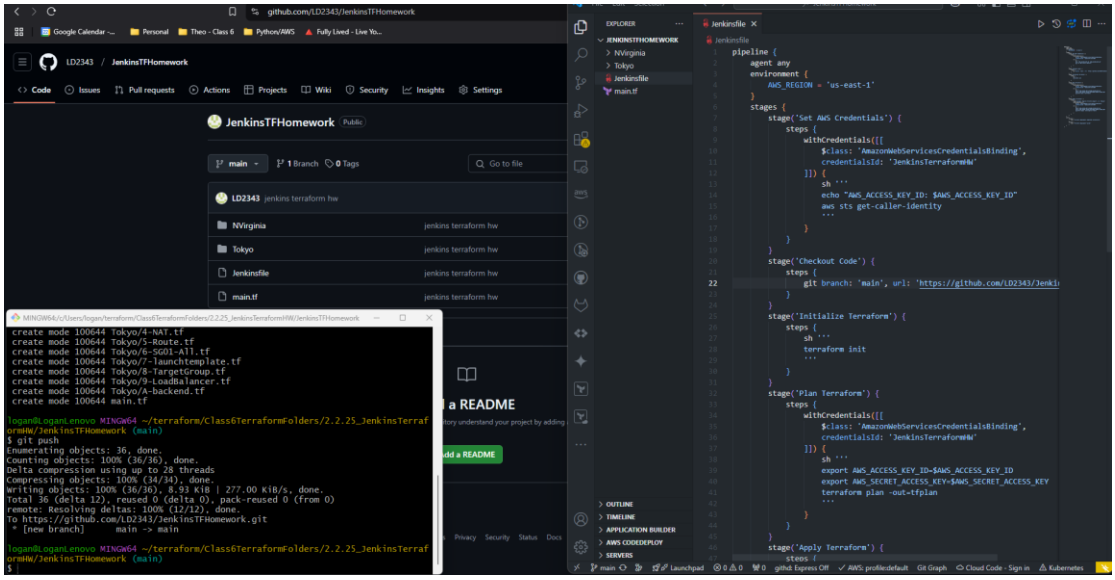
f. 

g. 

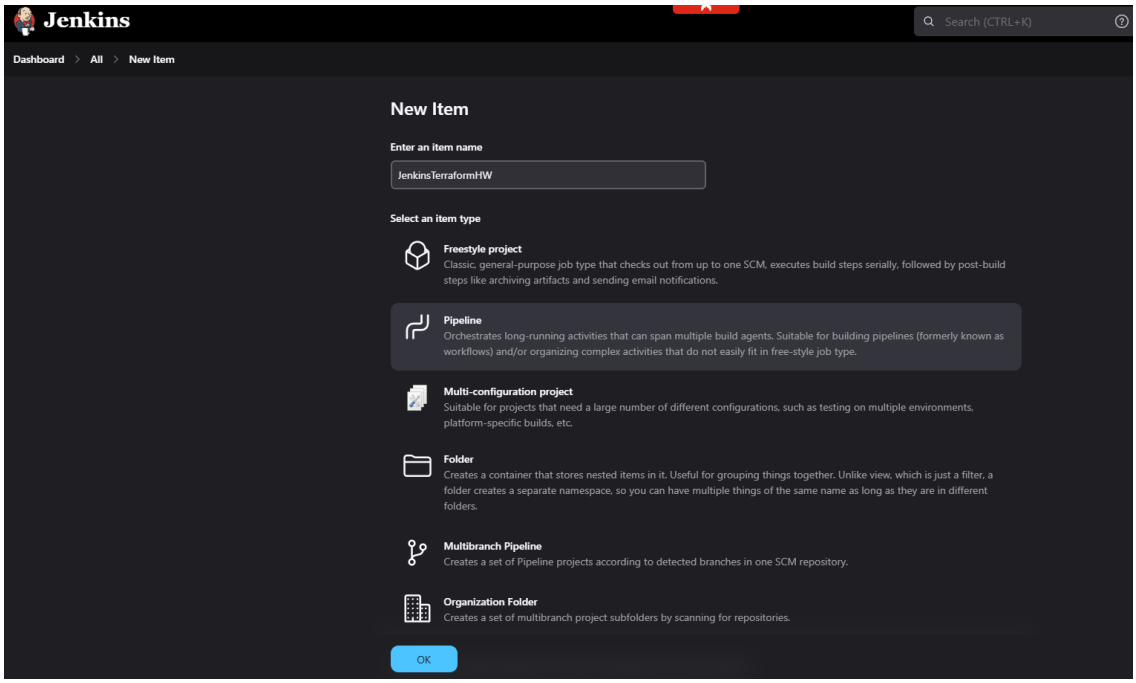
6. Create new Github repository with working Terraform project and ensure the Jenkins file is properly configured and included in repository.
   a. Log into Github and create a new repository.

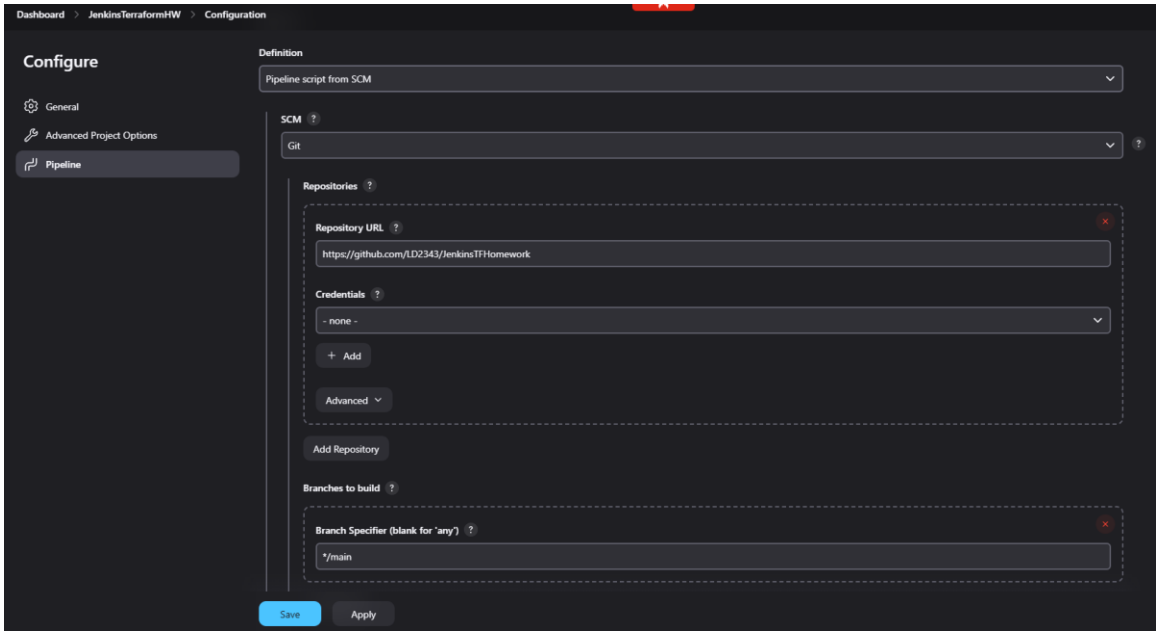
i. 

   b. Create a folder on pc to clone this empty repository to.
   c. Add working Terraform project and Jenkins file to this folder.
   d. In VS code update the Jenkins file with the correct
   e. Perform Git commands to push the terraform project and Jenkins file to repository.

f. 

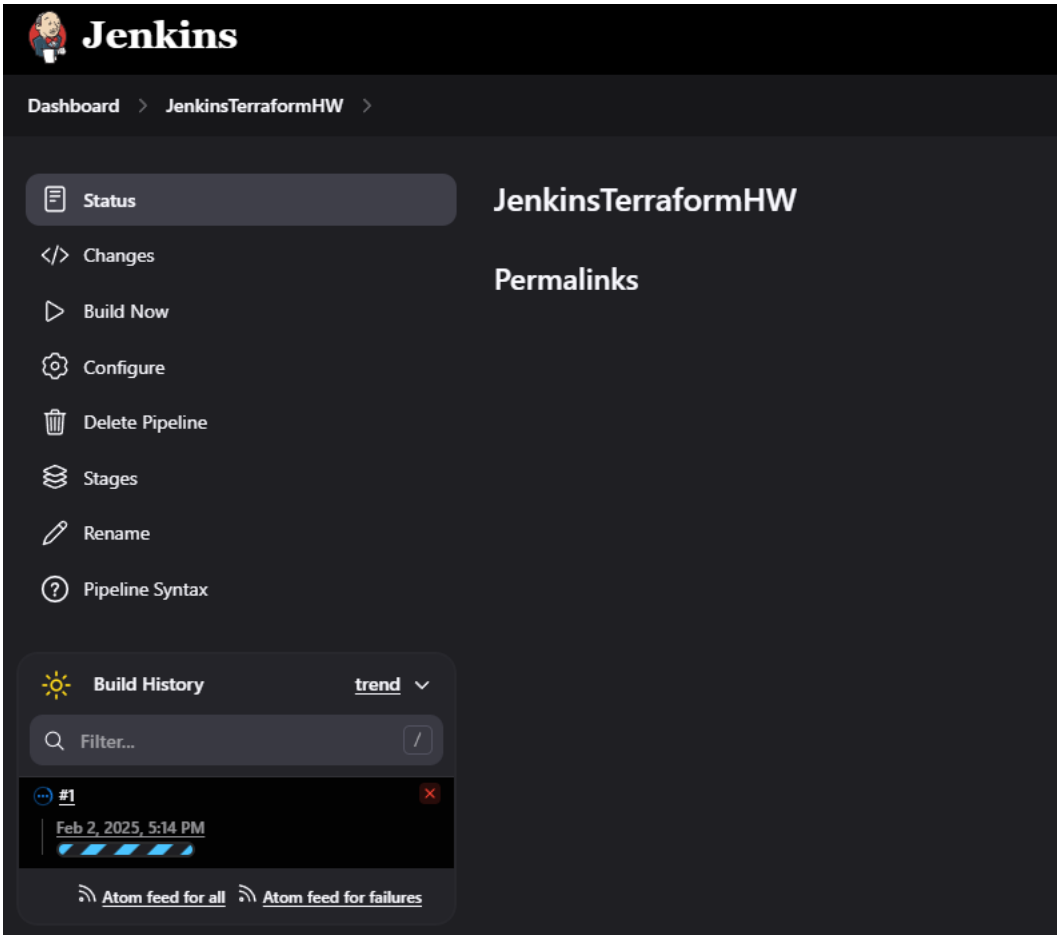
7. Create a new Pipeline in Jenkins and configure it to utilize our Github repository.
    a. In Jenkins under Dashboard, click on New Item.
    b. Name item and select Pipeline then click ok.
    c. In the Pipeline section, for Definition select Pipeline script from SCM.
    d. In SCM, select Git.
    e. Paste the Github repository url and update the Branch Specifier to */main.
    f. Click Save

g. 

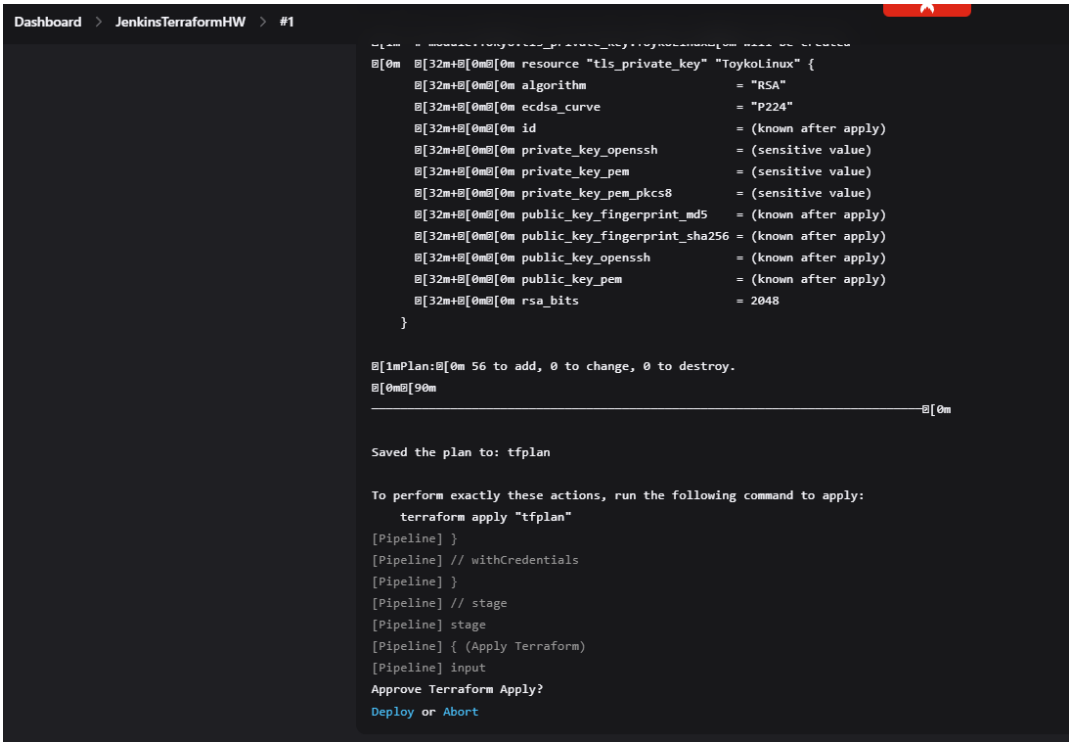
h. 

8. Run our new Pipeline in Jenkins to fire off the build of our Terraform project.
    a. Inside the Pipeline, click on Build Now to run.
    b. Click on the progress bar.
    c. When prompted click on Deploy in the Console Output.
    d. Wait for build to complete.
    e. ...

f.


Jenkins
Dashboard
JenkinsTerraformHW
Status
Changes
Build Now
Configure
Delete Pipeline
Stages
Rename
Pipeline Syntax
JenkinsTerraformHW
Permalinks
Build History
trend
Filter...
#1
Feb 2, 2025, 5:14 PM
Atom feed for all
Atom feed for failures

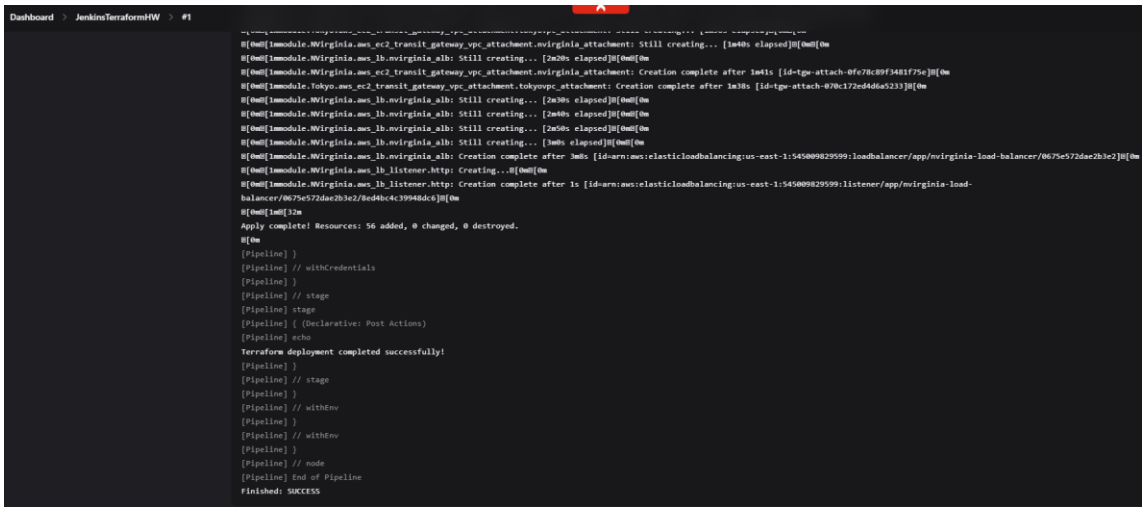
g.

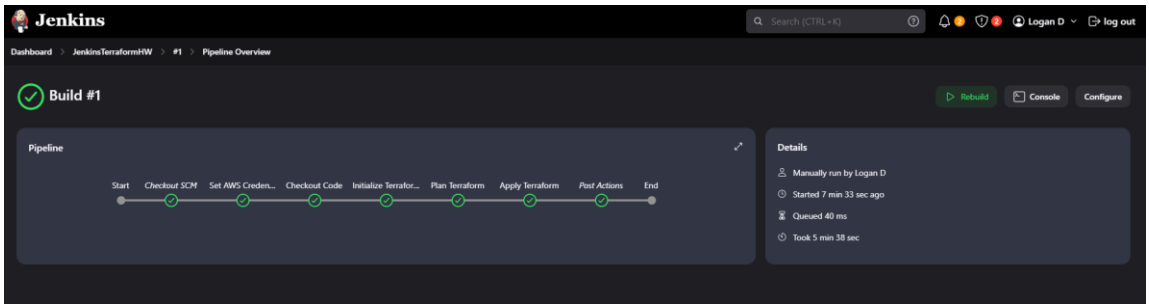

Dashboard
JenkinsTerraformHW
#1
resource "tls_private_key" "ToykoLinux" {
algorithm = "RSA"
ecdsa_curve = "P224"
id = (known after apply)
private_key_openssh = (sensitive value)
private_key_pem = (sensitive value)
private_key_pem_pkcs8 = (sensitive value)
public_key_fingerprint_md5 = (known after apply)
public_key_fingerprint_sha256 = (known after apply)
public_key_openssh = (known after apply)
public_key_pem = (known after apply)
rsa_bits = 2048
}
Plan: 56 to add, 0 to change, 0 to destroy.
Saved the plan to: tfplan
To perform exactly these actions, run the following command to apply:
terraform apply "tfplan"
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Apply Terraform)
[Pipeline] input
Approve Terraform Apply?
Deploy or Abort

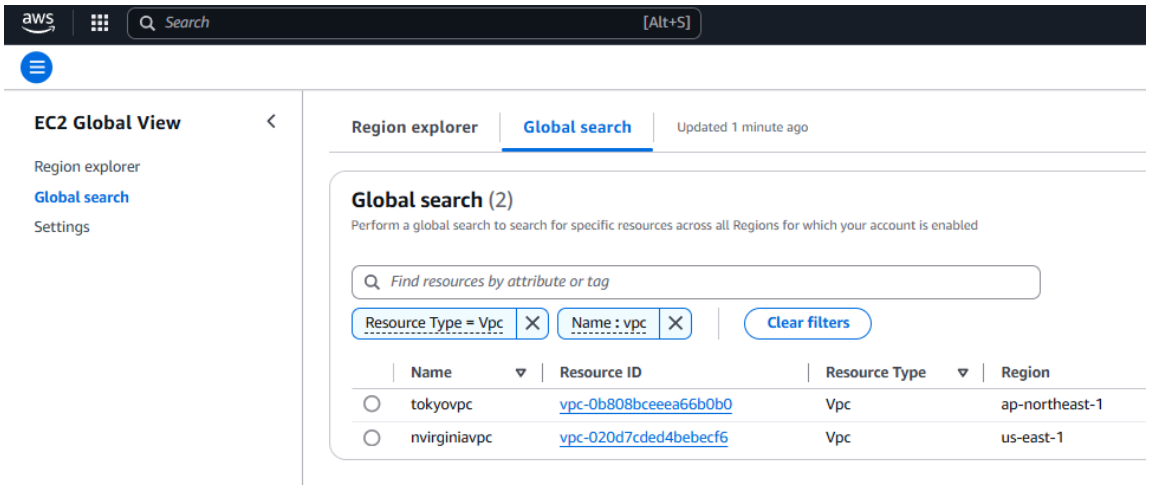
h. ...

i. ...

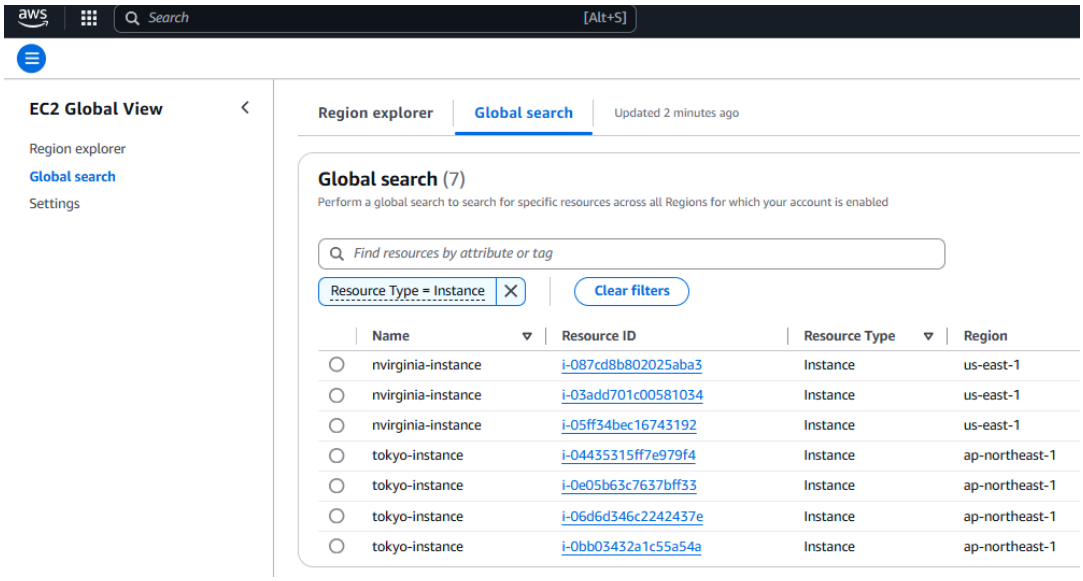
j. ...

9. Confirm the Pipeline build was successful.
    a. Verify the build was successful in Jenkins in the Console Output, Pipeline Overview and AWS.
    b. 

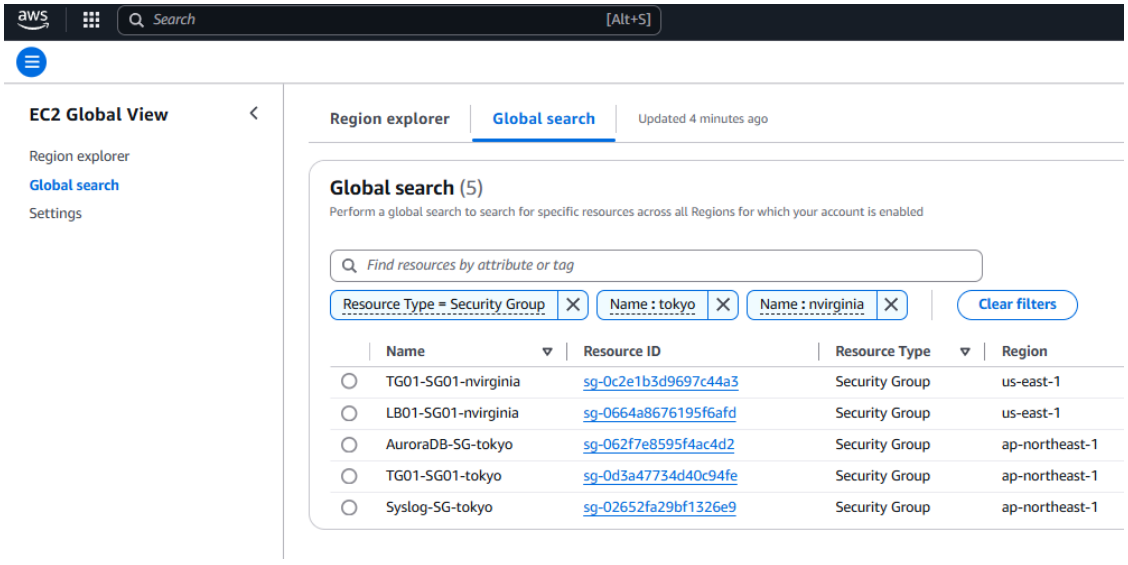
    c. 

    d. 

e.



f.



10. Perform Terraform teardown within our container in the CLI.
    a. In CLI inside our container, run to following command “cd var/jenkins_home/workspace/<PIPELINENAME>” to change directory to our pipeline.
    b. Run the following commands to input our AWS credentials and region in our container in order to run terraform destroy.
        i. export AWS_ACCESS_KEY_ID="xxxxxxx"
        ii. export AWS_SECRET_ACCESS_KEY="xxxxxxxx"
        iii. export AWS_REGION="xxxxxxx"
    c. Run the following command to teardown our terraform build.
        i. terraform destroy

d.

```
Plan: 0 to add, 0 to change, 56 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value:
```

e.

```
MINGW64:/c/Users/logan
Plan: 0 to add, 0 to change, 8 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

module.NVirginia.aws_autoscaling_group.nvirginia_asg: Destroying... [id=nvirginia-auto-scaling-group-2025020217172303210000003]
module.NVirginia.aws_autoscaling_group.nvirginia_asg: Destruction complete after 1s
module.NVirginia.aws_subnet.private-us-east-1c: Destroying... [id=subnet-01ad39ecc4d1e60cd]
module.NVirginia.aws_subnet.private-us-east-1a: Destroying... [id=subnet-0b7402f40121d3f0b]
module.NVirginia.aws_lb_target_group.nvirginia_tg: Destroying... [id=arn:aws:elasticloadbalancing:us-east-1:545009829599:targetgroup/nvirginia-
target-group/3aa163369f6f6b6d]
module.NVirginia.aws_subnet.private-us-east-1b: Destroying... [id=subnet-0e677860260252 8cf]
module.NVirginia.aws_launch_template.nvirginia_LT: Destroying... [id=lt-0a941dabf7669b384]
module.NVirginia.aws_lb_target_group.nvirginia_tg: Destruction complete after 0s
module.NVirginia.aws_launch_template.nvirginia_LT: Destruction complete after 0s
module.NVirginia.aws_security_group.TG01-SG01-nvirginia: Destroying... [id=sg-0c2e1b3d9697c44a3]
module.NVirginia.aws_subnet.private-us-east-1b: Destruction complete after 0s
module.NVirginia.aws_subnet.private-us-east-1c: Destruction complete after 0s
module.NVirginia.aws_subnet.private-us-east-1a: Destruction complete after 0s
module.NVirginia.aws_security_group.TG01-SG01-nvirginia: Destruction complete after 1s
module.NVirginia.aws_vpc.nvirginiavpc: Destroying... [id=vpc-020d7cded4bebecf6]
module.NVirginia.aws_vpc.nvirginiavpc: Destruction complete after 0s
```