Lab6. Terraform Pipelining

Contents

Start Lab

1. Ensure you have completed Lab0 before attempting this lab.
2. In the IDE terminal pane, enter the following commands...

```
cd ~/environment/awslabs/06
```

3. This shifts your current working directory to awslabs/labs/06. ***Ensure all IDE commands are executed in this directory***

Task1: Create an S3 Bucket for Remote State

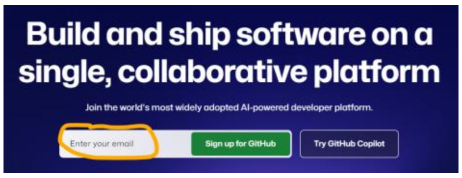
1. Switch to the AWS Console.
2. Search for and then navigate to the **S3** service. Click on **Create bucket**
3. Ensure you are focussed on the **Oregon (us-west-2)** region
4. Name your bucket **jenkins-state-<your-name>**. Every bucket name must be globally unique; therefore you may get a message indicating that a bucket

already exists with your chosen name. If so, then simply append a random number after your name. Record the name of this bucket in your session-info file against ***Jenkins-state-bucket***

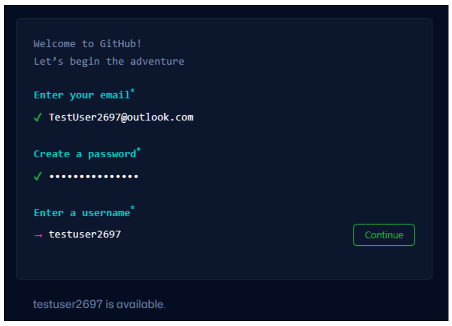
5. Leaving all settings at their default values, scroll down and select Create bucket.

Task2. Create a Github account and repository

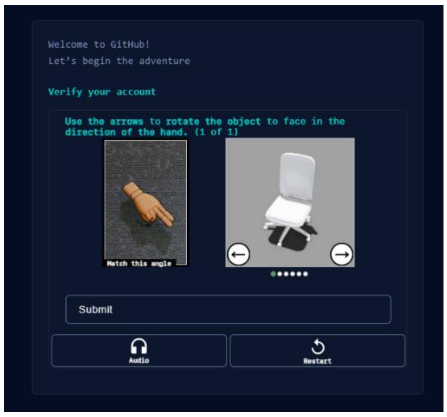
1. Sign up to Github using a personal email address that is not currently associated with Github…
   a. Navigate to https://github.com/
   b. Enter a personal email address and select "Sign up for Github"…


Build and ship software on a single, collaborative platform
Join the world's most widely adopted AI-powered developer platform.

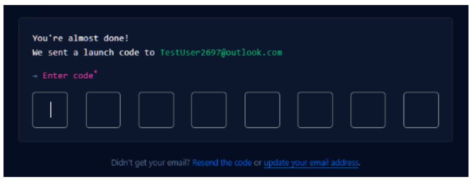
   c. Enter a password and a unique username of your choice..


Welcome to GitHub!
Let's begin the adventure
Enter your email*
✓ TestUser2697@outlook.com
Create a password*
✓ •••••••••••••••
Enter a username*
→ testuser2697
testuser2697 is available.

   d. Record and save your chosen **username** and **password** in your session-info file for safekeeping.
   e. Complete the challenge to prove you are a human…


Welcome to GitHub!
Let's begin the adventure
Verify your account
Use the arrows to rotate the object to face in the direction of the hand. (1 of 1)

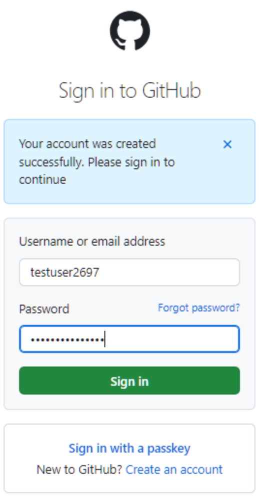
f. An email will sent containing your launch code. Retrieve this and enter it..


You're almost done!
We sent a launch code to TestUser2697@outlook.com

→ Enter code*

Didn't get your email? Resend the code or update your email address.

g. You will then be prompted to log into github using your new account..

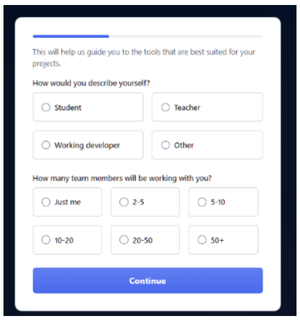

Sign in to GitHub

Your account was created successfully. Please sign in to continue

Username or email address: testuser2697

Password: •••••••••••••••

h. Complete the questionnaire…


This will help us guide you to the tools that are best suited for your projects.

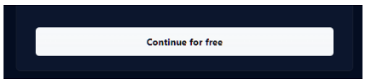
How would you describe yourself?

○ Student ○ Teacher ○ Working developer ○ Other

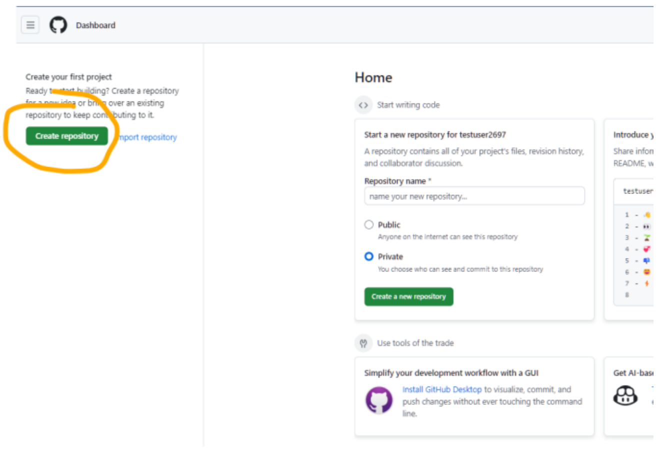
How many team members will be working with you?

○ Just me ○ 2-5 ○ 5-10 ○ 10-20 ○ 20-50 ○ 50+

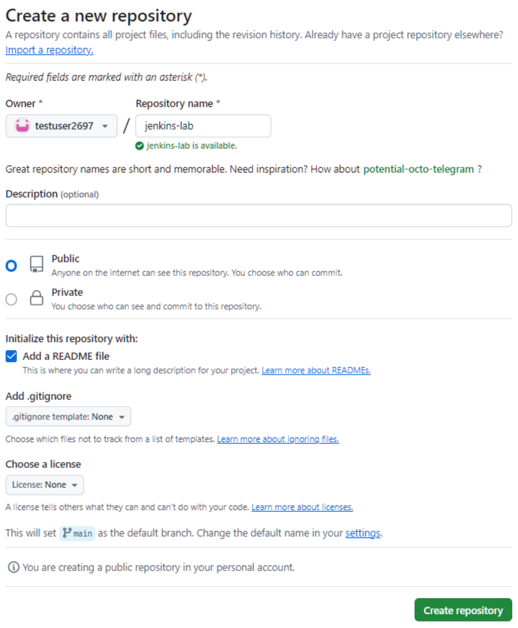
i. When asked to select a subscription, select "**Continue for free**"..



2. Create a public repository …
   a. Click on New …


Dashboard
Create your first project
Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.
Create repository
Import repository
Home
Start writing code
Start a new repository for testuser2697
A repository contains all of your project's files, revision history, and collaborator discussion.
Repository name *
name your new repository...
Public
Anyone on the internet can see this repository
Private
You choose who can see and commit to this repository
Create a new repository
Use tools of the trade
Simplify your development workflow with a GUI
Install GitHub Desktop to visualize, commit, and push changes without ever touching the command line.

b. Enter a Repository name of your choice. Ensure **Public** is selected and check the ‘**Add a README file’** option. Then click on **Create repository**…



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner *: testuser2697 / Repository name *: jenkins-lab

jenkins-lab is available.

Great repository names are short and memorable. Need inspiration? How about potential-octo-telegram ?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private: You choose who can see and commit to this repository.

Initialize this repository with:

[✓] Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.

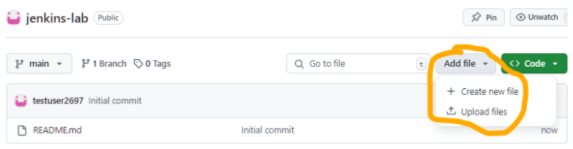
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

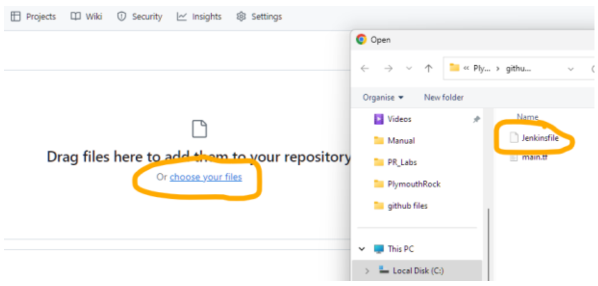
You are creating a public repository in your personal account.

Create repository

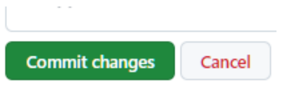
c. Click on **Add file**, **Upload file**..



d. Select the file **Jenkinsfile** from the **lab6** folder of your student bundle files that you downloaded at the start of the course

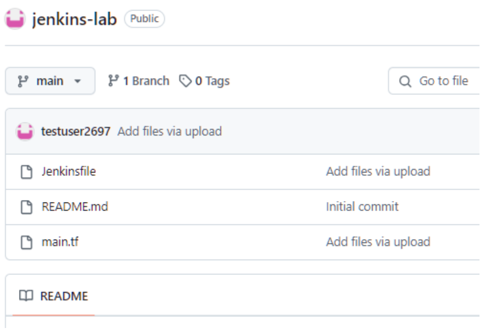


e. Click on Commit changes..



f. Repeat the previous 2 steps to upload the **main.tf** file from the Lab6 folder

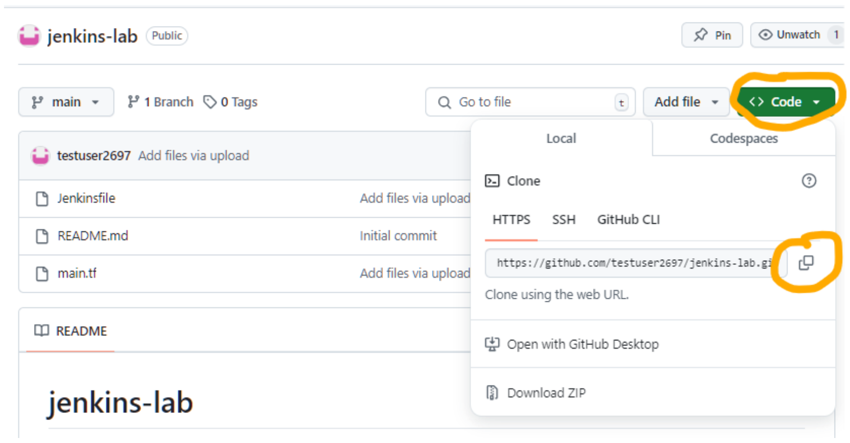
g. The 2 files should now be listed..



testuser2697 Add files via upload

Jenkinsfile	Add files via upload
README.md	Initial commit
main.tf	Add files via upload

h. Copy the URL of your repository to your clipboard..



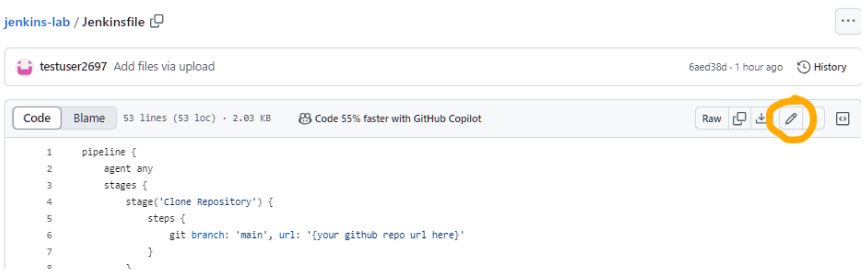
testuser2697 Add files via upload

Jenkinsfile	Add files via upload
README.md	Initial commit
main.tf	Add files via upload

```
https://github.com/testuser2697/jenkins-lab.gi
```

i. Record this URL in your **session-info** file against **Repo-URL**

j. Click on the file **Jenkinsfile** and then click on the **edit** icon to open the file for editing...

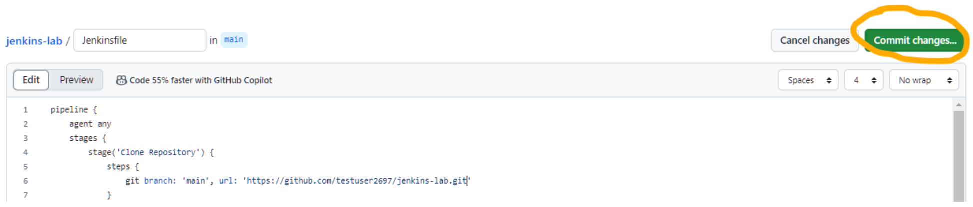

jenkins-lab / Jenkinsfile

testuser2697 Add files via upload 6aed38d · 1 hour ago History

Code Blame 53 lines (53 loc) · 2.03 KB Code 55% faster with GitHub Copilot Raw

```
1   pipeline {
2       agent any
3       stages {
4           stage('Clone Repository') {
5               steps {
6                   git branch: 'main', url: '{your github repo url here}'
7               }
```

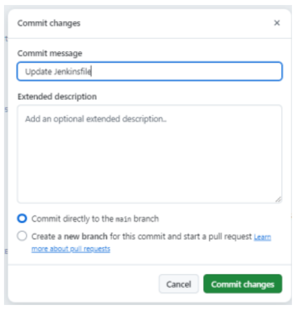
k. On line 6, replace **{your github repo url here},** including the braces, with your **Repo-URL** and then click on **Commit changes** twice...


jenkins-lab / Jenkinsfile in main

Cancel changes Commit changes...

Edit Preview Code 55% faster with GitHub Copilot Spaces 4 No wrap

```
1   pipeline {
2       agent any
3       stages {
4           stage('Clone Repository') {
5               steps {
6                   git branch: 'main', url: 'https://github.com/testuser2697/jenkins-lab.git'
7               }
```


Commit changes

Commit message: Update Jenkinsfile

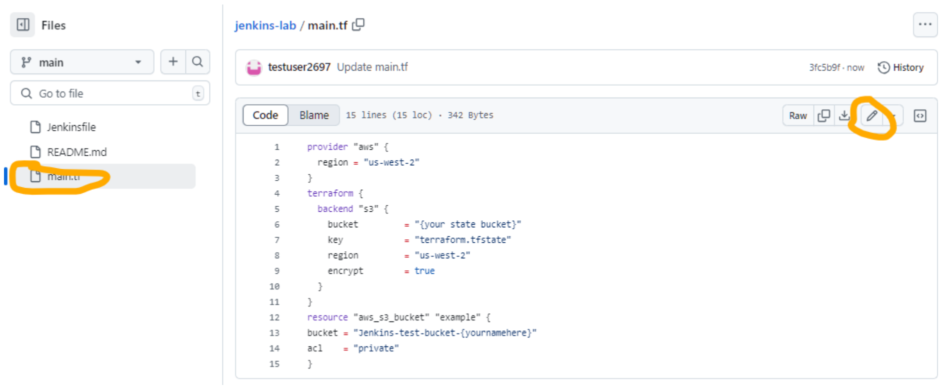
Extended description: Add an optional extended description..

[✓ Commit directly to the main branch]

Create a new branch for this commit and start a pull request Learn more about pull requests

Cancel Commit changes

l. Click on **main.tf** and then open it for edits...


Files

main

Go to file

Jenkinsfile

README.md

main.tf

jenkins-lab / main.tf

testuser2697 Update main.tf 3fc5b9f · now History

Code Blame 15 lines (15 loc) · 342 Bytes Raw

```
1   provider "aws" {
2     region = "us-west-2"
3   }
4   terraform {
5     backend "s3" {
6       bucket         = "{your state bucket}"
7       key            = "terraform.tfstate"
8       region         = "us-west-2"
9       encrypt        = true
10    }
11  }
12  resource "aws_s3_bucket" "example" {
13  bucket = "Jenkins-test-bucket-{yournamehere}"
14  acl    = "private"
15  }
```

m. On line 6, replace **{your state bucket},** including the braces, with the name of the bucket you created in Task1 (recorded as **Jenkins-state-bucket** in your session-info file). This is where your statefile will be created when you initialize Terraform

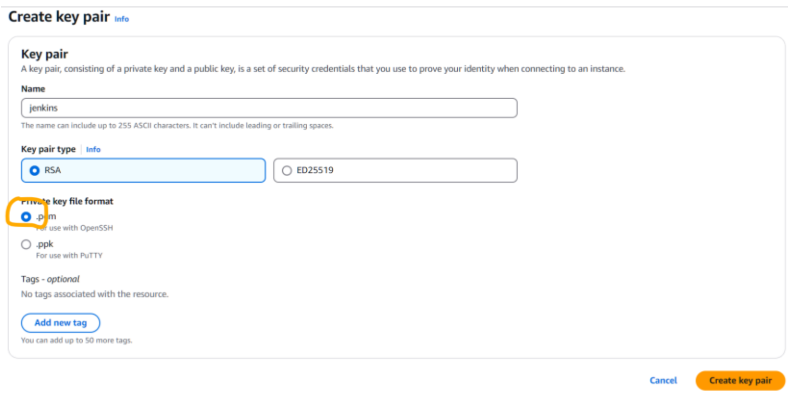
n. On line 13, replace **{younamehere},** including the braces, with your name followed by 2 random digits (to guarantee uniqueness). This will be the name of the bucket that Terraform will create when we run a Jenkins pipeline.

o. Commit these changes as before.

p. Leave the Github tab open as we will return to it later.

Task3. Create EC2 instance for Jenkins

1. Use the AWS console to navigate to **EC2** and then **create** and download a **.pem key-pair** file named **jenkins**



Create key pair Info

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

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

Key pair type | Info
[✓ RSA] ED25519

Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

Tags - *optional*
No tags associated with the resource.

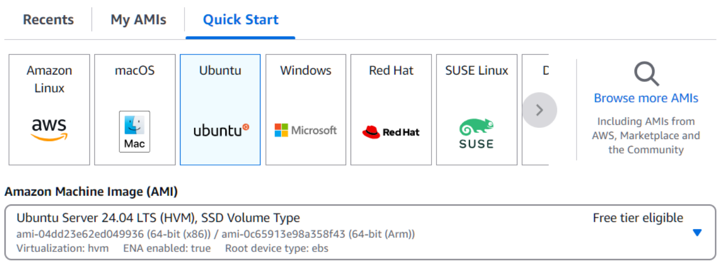
You can add up to 50 more tags.

2. Launch an Ubuntu EC2 instance with the following specifications:

Region: **Oregon (us-west-2)**

Name: **JenkinsServer**

Image: Select **Ubuntu** from Quick Start..

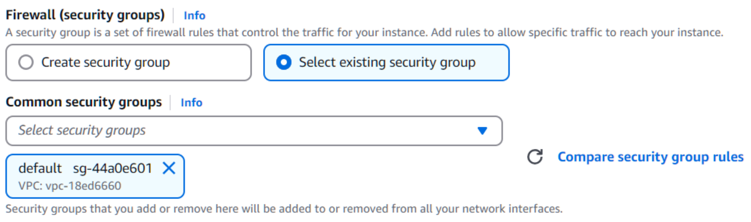


Amazon Machine Image (AMI)
Ubuntu Server 24.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-04dd23e62ed049936 (64-bit (x86)) / ami-0c65913e98a358f43 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Instance type: **t2.small**

Key pair: select **jenkins**

Network Settings (Firewall) : Select the **default** security group


Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

○ Create security group [✓ Select existing security group]

Common security groups | Info

Select security groups

default sg-44a0e601
VPC: vpc-18ed6660

Compare security group rules

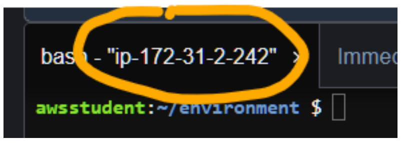
Security groups that you add or remove here will be added to or removed from all your network interfaces.

Select “**Launch Instance**” to create and start the instance

Task4. Modify the default security group

We are going to allow all inbound traffic from your Cloud9 instance terminal IP address and your local machine’s IP address. We will also allow all inbound TCP port 8080 traffic to communicate with Jenkins. (**Note**: In production environments the firewall rules would be more restrictive)

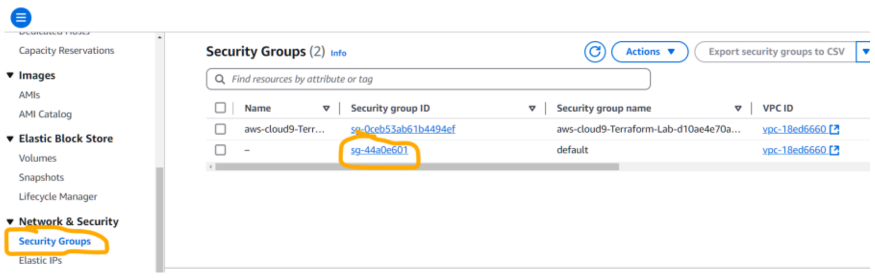
1. Identify your Cloud9 IDE terminal address. Here it is 172.31.2.242 (**replace hyphens with periods**), yours will be different (hover over the address if it is slightly obscured)...



```
bash - "ip-172-31-2-242"    Immed
awsstudent:~/environment $
```

2. Record this in your session-info as IDE IP, in this example it is **172.31.2.242**

3. In the **EC2** dashboard, scroll down, select **Security Groups** and open the default Security group...



Security Groups (2) Info

Name	Security group ID	Security group name	VPC ID
aws-cloud9-Terr...	sg-0ceb53ab61b4494ef	aws-cloud9-Terraform-Lab-d10ae4e70a...	vpc-18ed6660
–	sg-44a0e601	default	vpc-18ed6660

4. Select **Edit inbound rules**

5. Select to **Add rule** ...

   Type: **All traffic**

   Source: **My IP**

6. Click **Add rule** to add a second rule ...

Type: **All traffic**

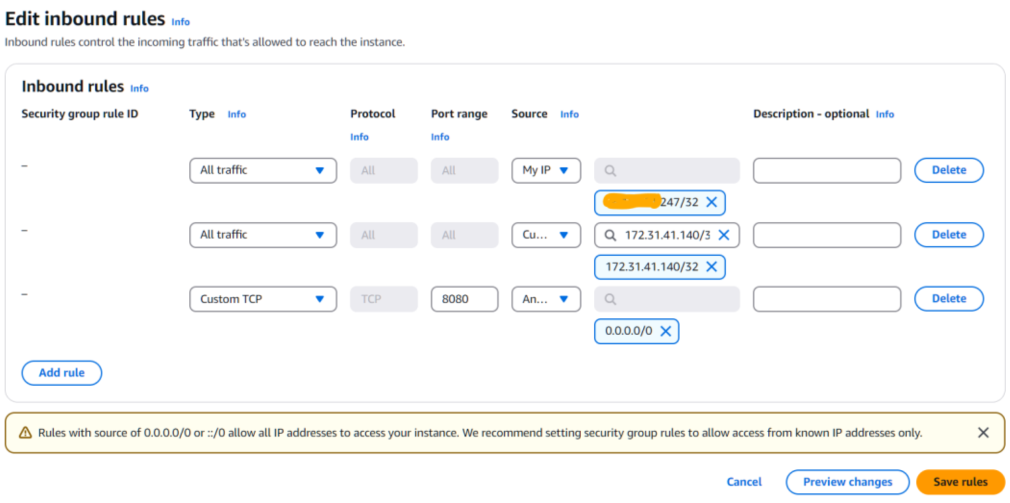
Source: Select **Custom** and enter the IP address of your IDE followed by "/32"...

7. Click Add rule to add a third rule ...

   Type: **Custom TCP**

   Port range: **8080**

   Source: **Anywhere-IPv4**



Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
-	All traffic	All	All	My IP	[redacted]247/32		Delete
-	All traffic	All	All	Cu...	172.31.41.140/3 172.31.41.140/32		Delete
-	Custom TCP	TCP	8080	An...	0.0.0.0/0		Delete

Add rule

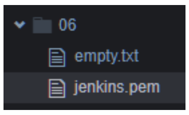
⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel | Preview changes | Save rules

8. Take note of the advisory regarding allowing all IP addresses. As mentioned, in production environments, the firewall rules would be more precise. Save the newly created rules

Task5. Connect to the Instance

1. In the IDE; Use Explorer to navigate to and select the awslabs/06 folder. Then upload the **jenkins.pem** key-pair file from your local machine to this folder. (File, Upload Local Files)..

   ```
   06
     empty.txt
     jenkins.pem
   ```

2. Set the correct permissions for the file:

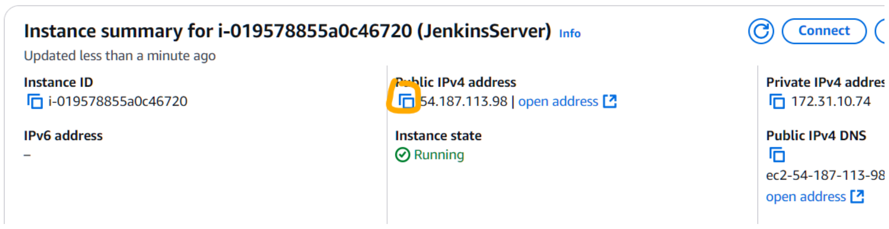
   ```
   chmod 400 jenkins.pem
   ```

3. Connect to the JenkinsServer EC2 instance from the IDE using SSH;

   a. Navigate to your **EC2** instances in the console.

b. Click on the **instance Id** of **JenkinsServer** to display the instance summary

c. Make a note of the Public IP address allocated to your instance. Record this in your session-info file as **Jenkins Public IP**...



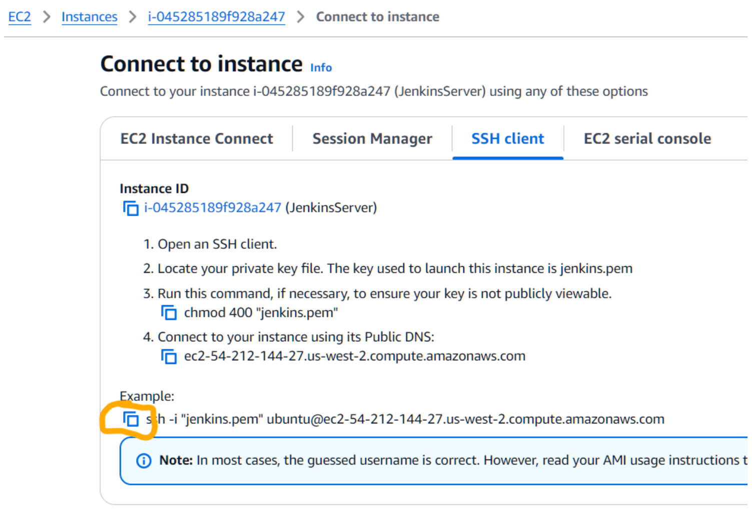
Instance summary for i-019578855a0c46720 (JenkinsServer) Info
Updated less than a minute ago

Instance ID: i-019578855a0c46720
Public IPv4 address: 54.187.113.98 | open address
Private IPv4 addres: 172.31.10.74
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-54-187-113-98 open address

d. Click on the **Connect** option

e. Select the **SSH client** tab

f. Copy the Example connection string and paste it into your IDE...



Connect to instance Info
Connect to your instance i-045285189f928a247 (JenkinsServer) using any of these options

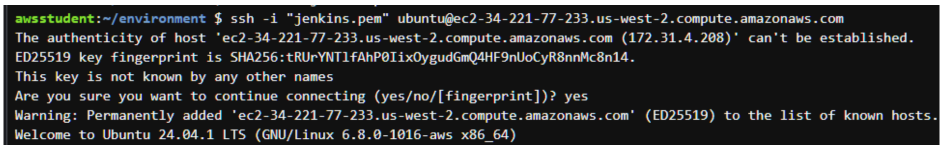
Instance ID
i-045285189f928a247 (JenkinsServer)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is jenkins.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "jenkins.pem"
4. Connect to your instance using its Public DNS:
   ec2-54-212-144-27.us-west-2.compute.amazonaws.com

Example:
ssh -i "jenkins.pem" ubuntu@ec2-54-212-144-27.us-west-2.compute.amazonaws.com

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions t

g. When prompted to continue with the connection, type **yes**



```
awsstudent:~/environment $ ssh -i "jenkins.pem" ubuntu@ec2-34-221-77-233.us-west-2.compute.amazonaws.com
The authenticity of host 'ec2-34-221-77-233.us-west-2.compute.amazonaws.com (172.31.4.208)' can't be established.
ED25519 key fingerprint is SHA256:tRUrYNT1fAhP0IixOygudGmQ4HF9nUoCyR8nnMc8n14.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-34-221-77-233.us-west-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)
```

Task6. Install Jenkins

1. You are now connected to the Jenkins instance from your IDE. Run the following commands one at a time to update the system and install the Jenkins software (reduced font size for 4th and 5th commands is to avoid word-wrap issues in PDF)...

```
sudo apt update
sudo apt upgrade -y

sudo apt install -y openjdk-17-jdk

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

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update

sudo apt install -y jenkins

sudo systemctl start jenkins

sudo systemctl enable jenkins

sudo apt-get update

sudo apt-get install -y unzip
```

Task7. Grant Jenkins Sudo Access

1. In your IDE ssh session; edit the sudoers file:

   ```
   sudo visudo
   ```

2. Use your down key to move to the end of the file and then paste in the following line:

   ```
   jenkins ALL=(ALL) NOPASSWD: /bin/mv
   ```

3. Save and exit. (Press "Ctrl+x", then "y" when prompted to save changes and then press enter)

Task8. Configure Jenkins

1. Obtain the initial admin password for Jenkins:

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

2. Open Jenkins in a new browser tab using the IP address recorded in your session-info file

   ```
   http://Jenkins Public IP:8080
   ```

3. Copy and paste the admin password from the IDE into the **Unlock Jenkins** screen the click on Continue

4. Select "**Install suggested plugins**"

5. On the "**Create First Admin User**" screen; Select "**Skip and continue as admin**"

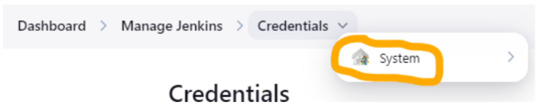
6. Click "**Save and Finish**" to complete the Jenkins configuration.

7. Click "**Start using Jenkins**"

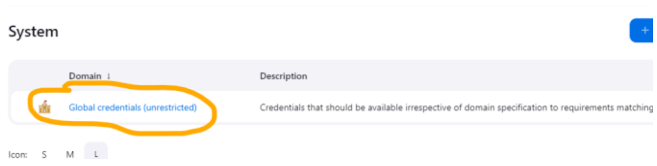
Task9. Add AWS Credentials to Jenkins

1. In order to use Terraform on Jenkins to interact with AWS, we must supply it with credentials to use. In your **Jenkins** browser session; Navigate to **Manage Jenkins > Credentials**.

2. On the breadcrumb menu; click on the **Credentials** dropdown and then select **System** ...



3. Click on "**Global credentials (unrestricted)"**...



Domain ↓	Description
Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching

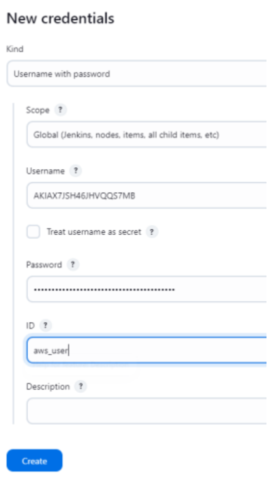
4. Click on "**Add Credentials**"

5. For Username: Enter the **Access key ID** generated for your lab user. (This and the Secret Access key required next, can be found in your **session-info** file)

6. For Password: Enter the **Secret Access key** generated for your lab user

7. For ID: enter **aws_user**

8. Select "**Create**"



New credentials

Kind: Username with password

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

Username: AKIAX7JSH46JHVQQS7MB

Treat username as secret

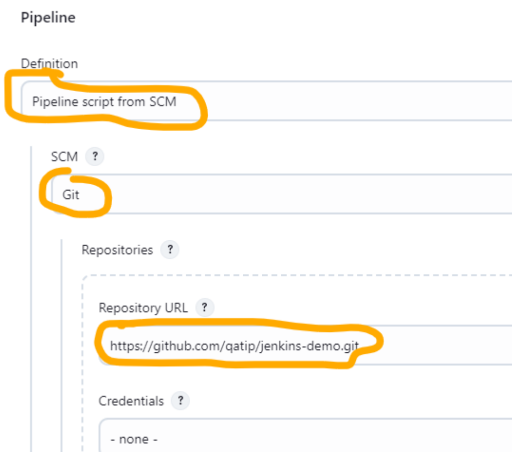
Password: ••

ID: aws_user

Description:

Task10. Configure Pipeline Job

1. Select “ **+ New Item**” from the Jenkins **dashboard**
2. Enter “**Terraform Pipeline**” as the item name
3. Select “**Pipeline**” as the item type
4. Click “**OK**”
5. On the **General** page displayed next, scroll down to the **Pipeline** section. Use the dropdown list to change the **Definition** from “**Pipeline script**” to “**Pipeline script from SCM**”
6. Select “**Git**” from the **SCM** dropdown list
7. In the **Repository URL**: Enter **your** Github repository URL, recorded in your session-info file as **Repo-URL**
8. In “**Branches to build**,” “**Branch Specifier**;” change from ***/master** to ***/main**



Pipeline

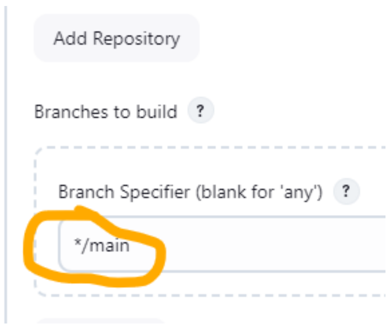
Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://github.com/qatip/jenkins-demo.git

Credentials: - none -


Add Repository

Branches to build

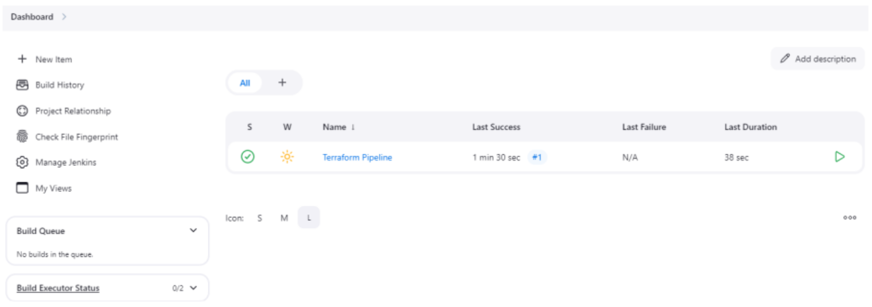
Branch Specifier (blank for 'any')

*/main

9. Click "**Save**"
10. Click "**Build Now**"

Task11. Verify pipeline run and explore Jenkins

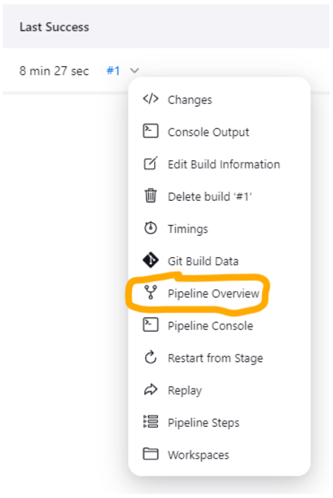
1. In Jenkins, return to the Dashboard. A record of the pipeline will be displayed showing run success and failure. Refresh the page until a result of the pipeline run is displayed...



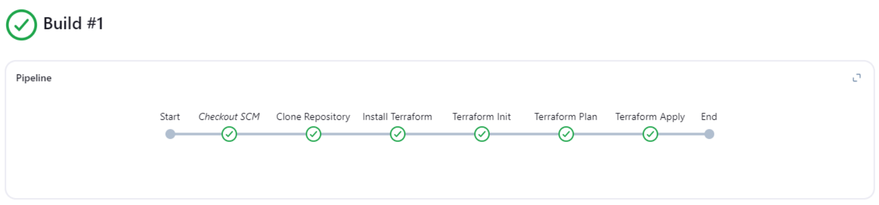
No builds in the queue.

S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	Terraform Pipeline	1 min 30 sec #1	N/A	38 sec

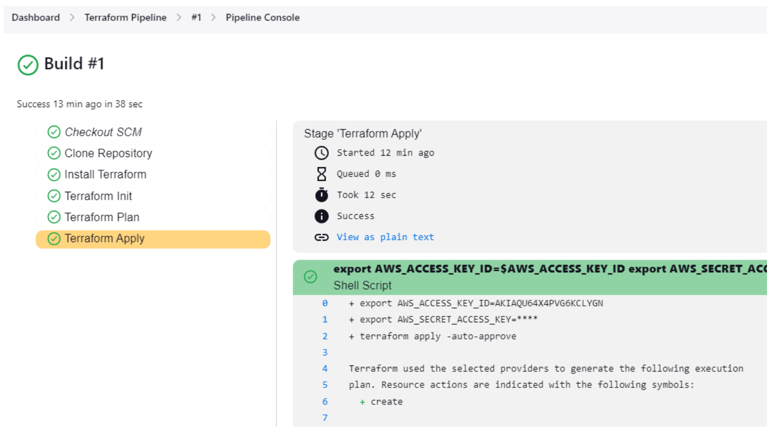
2. Select the drop-down menu against **#1** and choose Pipeline Overview..



3. The stages of the pipeline are shown, with ticks (or crosses) indicating success or failure at that stage...

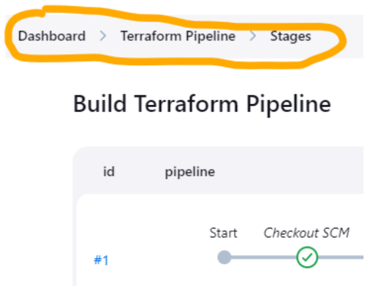


4. Examine the console logs generated by each stage by selecting the stage and clicking in the green area to toggle on/off the log display...

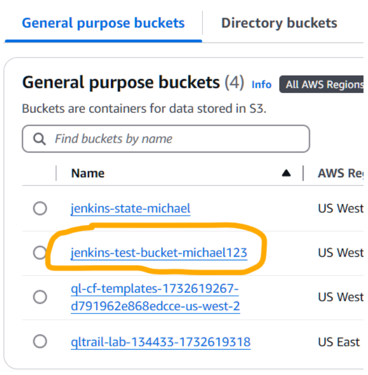


```
export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID export AWS_SECRET_ACC
Shell Script
+ export AWS_ACCESS_KEY_ID=AKIAQU64X4PVG6KCLYGN
+ export AWS_SECRET_ACCESS_KEY=****
+ terraform apply -auto-approve

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create

```

5. Spend a little time exploring the Jenkins interface, using the bread-crumb menu to navigate around, and finally return to the main Dashboard...



6. In the AWS console, navigate to the S3 service and verify the existence of your new storage bucket..


General purpose buckets
Directory buckets
General purpose buckets (4) Info
All AWS Regions
Buckets are containers for data stored in S3.
Find buckets by name

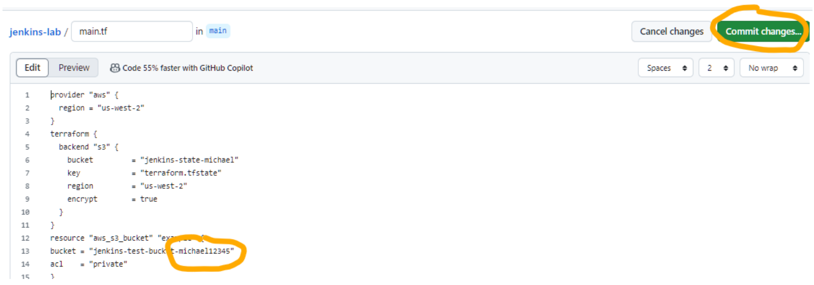
Name	AWS Re
jenkins-state-michael	US West
jenkins-test-bucket-michael123	US West
ql-cf-templates-1732619267-d791962e868edcce-us-west-2	US West
qltrail-lab-134433-1732619318	US East

Task12. Updating the transformation pipeline

In a production environment developers would now check-out the contents of the Github repo to their local machine, make updates to the terraform files and then check them back into the repo for approval. Once approved these changes would be merged with the current files and deployed by triggering a new pipeline run. This can be done manually in Jenkins or automatically using Webhooks, whereby Github notifies Jenkins of the changed files, and the pipeline run starts automatically to deploy these changes.

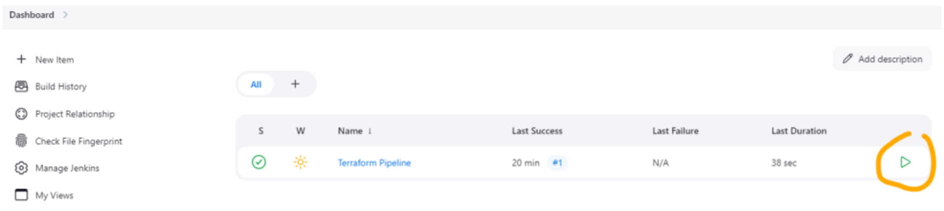
In this task we will modify the terraform files directly in Github before manually triggering a new pipeline run in Jenkins. We will then set up Webhooks to show how changes in Github can automatically trigger the pipeline run.

1. Return to your Github repository, logging back in if necessary.
2. Open main.tf for editing
3. Change the name of the bucket to be created by adding addition digits to the existing name. This will cause the original bucket to be deleted and a new one to be created. Commit this change..


jenkins-lab / main.tf in main
Cancel changes
Commit changes...
Edit
Preview
Code 55% faster with GitHub Copilot
Spaces
2
No wrap

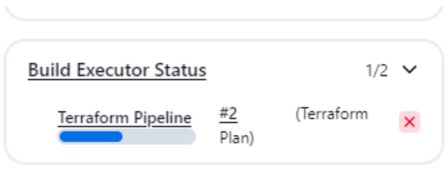
```
provider "aws" {
  region = "us-west-2"
}
terraform {
  backend "s3" {
    bucket         = "jenkins-state-michael"
    key            = "terraform.tfstate"
    region         = "us-west-2"
    encrypt        = true
  }
}
resource "aws_s3_bucket" "ex[illegible] {
bucket = "jenkins-test-bucket-michael12345"
acl    = "private"
}
```

4. Switch to Jenkins and click on the play icon to schedule a manual running of the pipeline..



S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	Terraform Pipeline	20 min #1	N/A	38 sec

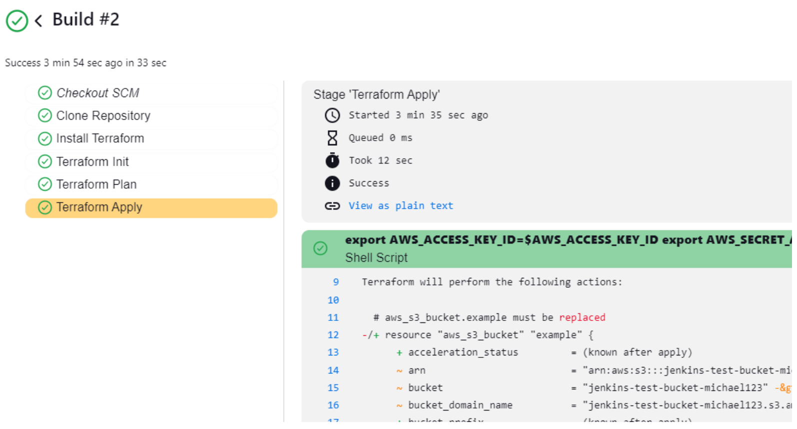
5. The build Executor Status will show the progress of the run..



Build Executor Status 1/2

Terraform Pipeline #2 (Terraform Plan)

6. The success runs count should increase to #2 indicating successful manual running of the pipeline. (You may need to refresh the page)
7. Looking at the logs for the **Terraform Apply** phase we see that the old bucket was replaced as bucket names are immutable and cannot be changed...


Build #2

Success 3 min 54 sec ago in 33 sec

- Checkout SCM
- Clone Repository
- Install Terraform
- Terraform Init
- Terraform Plan
- Terraform Apply

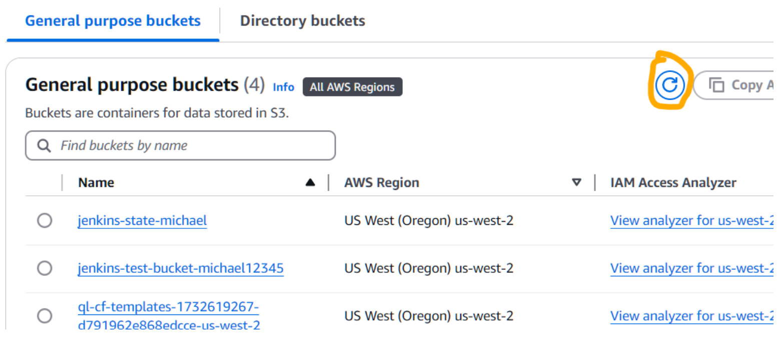
Stage 'Terraform Apply'

```
Started 3 min 35 sec ago
Queued 0 ms
Took 12 sec
Success
View as plain text
```

export AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID export AWS_SECRET_
Shell Script

```
Terraform will perform the following actions:

  # aws_s3_bucket.example must be replaced
-/+ resource "aws_s3_bucket" "example" {
      + acceleration_status         = (known after apply)
      ~ arn                         = "arn:aws:s3:::jenkins-test-bucket-mi
      ~ bucket                      = "jenkins-test-bucket-michael123" -&g
      ~ bucket_domain_name          = "jenkins-test-bucket-michael123.s3.a
```

8. Switch to your console and, in S3, verify the deletion of the old bucket and the creation of a new one, refreshing the display if necessary..


General purpose buckets | Directory buckets

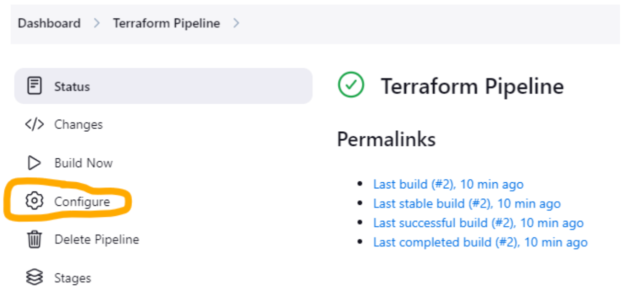
General purpose buckets (4) Info All AWS Regions

Buckets are containers for data stored in S3.

Find buckets by name

Name	AWS Region	IAM Access Analyzer
jenkins-state-michael	US West (Oregon) us-west-2	View analyzer for us-west-2
jenkins-test-bucket-michael12345	US West (Oregon) us-west-2	View analyzer for us-west-2
ql-cf-templates-1732619267-d791962e868edcce-us-west-2	US West (Oregon) us-west-2	View analyzer for us-west-2

9. To configure automatic pipeline running; **In Jenkins**, configure the pipeline by first selecting it on the main dashboard and then choosing the “Configure” option..

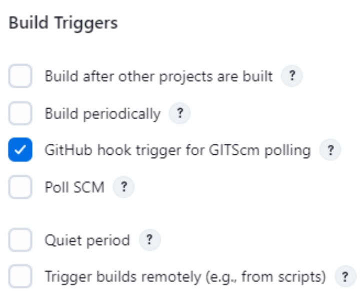


Terraform Pipeline

Permalinks

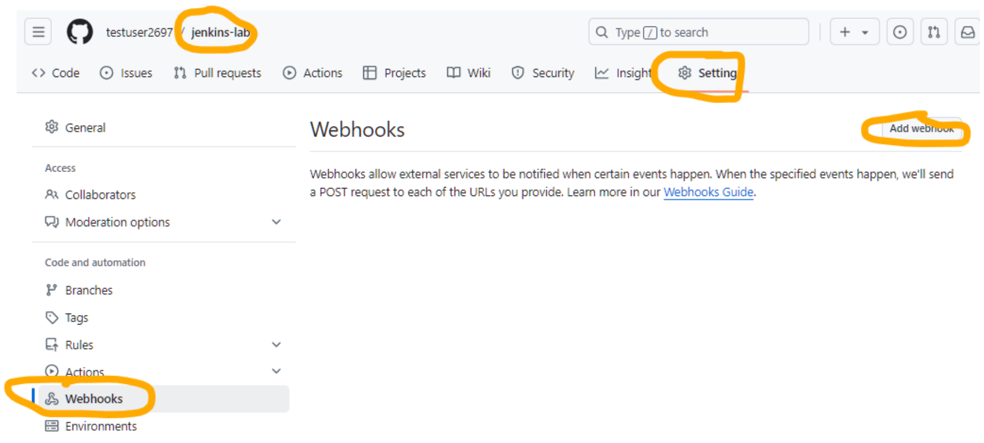
- Last build (#2), 10 min ago
- Last stable build (#2), 10 min ago
- Last successful build (#2), 10 min ago
- Last completed build (#2), 10 min ago

10. Scroll down to the **Build Triggers** section, select “**Github hook trigger for GITScm polling**” and click on **Save** ...


Build Triggers

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [x] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [ ] Quiet period ?
- [ ] Trigger builds remotely (e.g., from scripts) ?

11. **Switch to Github**. Select the **Settings** for your repo. Scroll down and select **Webhooks**. Click on **Add webhook** (you may be prompted to re-authenticate at this point)...



Webhooks

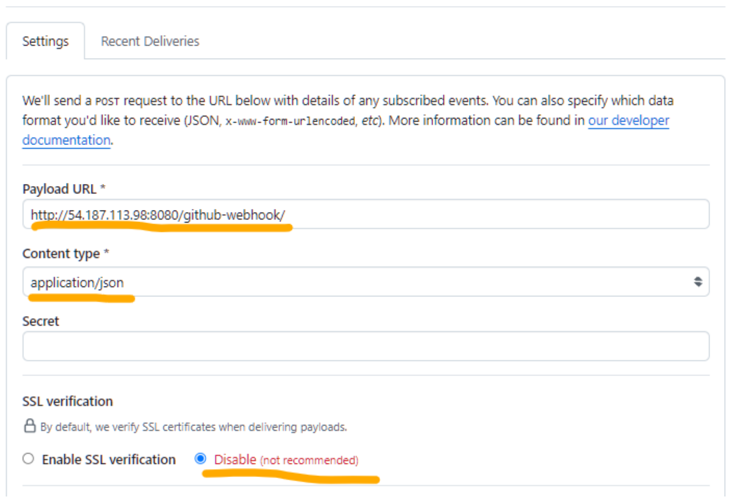
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

12. For **Payload URL;** enter **http://{Jenkins Public IP}:8080/github-webhook/** replacing (Jenkins Public IP} with the Public IP address of your Jenkins instance

13. For Content type; select application/json

14. Select to disable **SSL verification** for this lab environment.

15. Verify your settings as shown in example below (your IP will differ) before clicking on **Add webhook**..



We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *
http://54.187.113.98:8080/github-webhook/

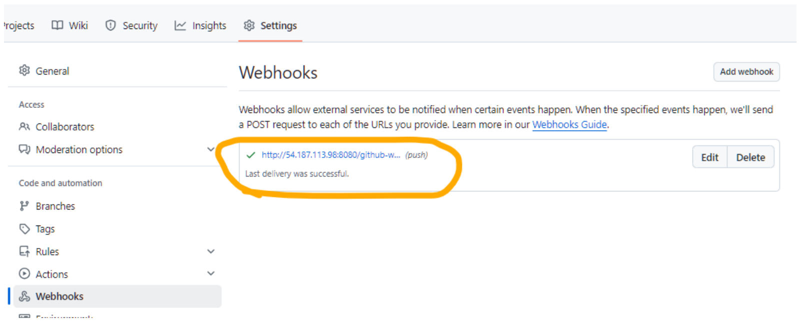
Content type *
application/json

Secret

SSL verification
By default, we verify SSL certificates when delivering payloads.
Enable SSL verification [✓ Disable (not recommended)]

16. Make another change to the name of your bucket in main.tf and commit the changes (refer to steps 2 and 3 above if you need guidance)

17. Re-visit your Webhooks setting and you should see confirmation that there was a successful push of the changes to Jenkins…

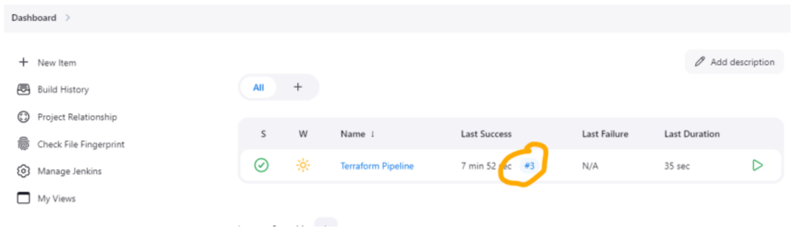


Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ http://54.187.113.98:8080/github-w… (push)
Last delivery was successful.

18. Switch to Jenkins. Return to the Dashboard and check that there is now a record of a third successful running of the pipeline…



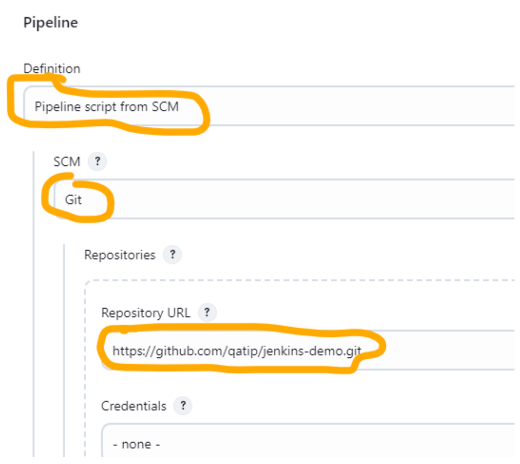
S	W	Name ↓	Last Success	Last Failure	Last Duration
✓	☀	Terraform Pipeline	7 min 52 sec #3	N/A	35 sec

19. Switch to your AWS console, navigate to S3, refresh the display if necessary and verify the new bucket has been created.

Task13 (Time permitting). Configure Destroy Pipeline Job

1. Select " **+ New Item**" from the Jenkins **dashboard**
2. Enter "**Terraform Pipeline Destroy**" as the item name
3. Select "**Pipeline**" as the item type
4. Click "**OK**"
5. On the **General** page displayed next, scroll down to the **Pipeline** section. Use the dropdown list to change the **Definition** from "**Pipeline script**" to "**Pipeline script from SCM**"
6. Select "**Git**" from the **SCM** dropdown list

7. In the **Repository URL**: Enter **your** Github repository URL, recorded in your session-info file as **Repo-URL**
8. In "**Branches to build**," "**Branch Specifier**;" change from ***/master** to ***/main**



Pipeline

Definition

Pipeline script from SCM

SCM

Git

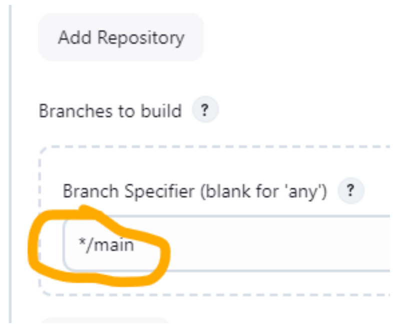
Repositories

Repository URL

https://github.com/qatip/jenkins-demo.git

Credentials

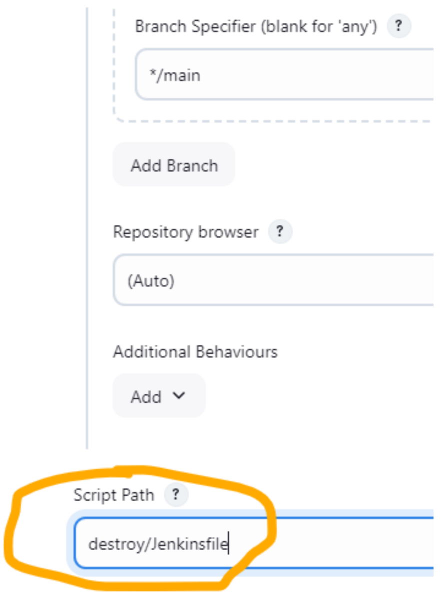
- none -



Branches to build

Branch Specifier (blank for 'any')

*/main



Branch Specifier (blank for 'any')

*/main

Repository browser

(Auto)

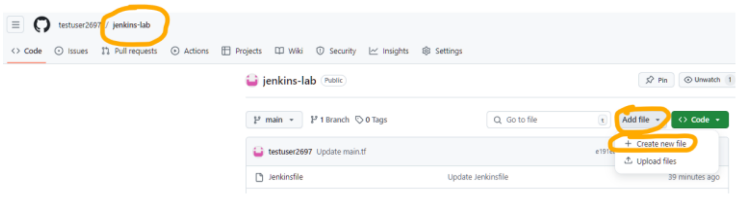
Additional Behaviours

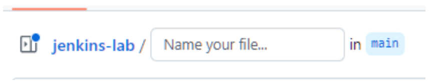
Script Path

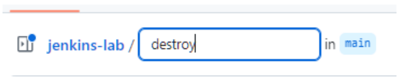
destroy/Jenkinsfile

9. Save the new pipeline but do not build it yet
10. Switch to your Github account

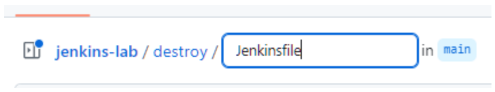
11. Create a new empty Jenkinsfile in a new folder “destroy”…



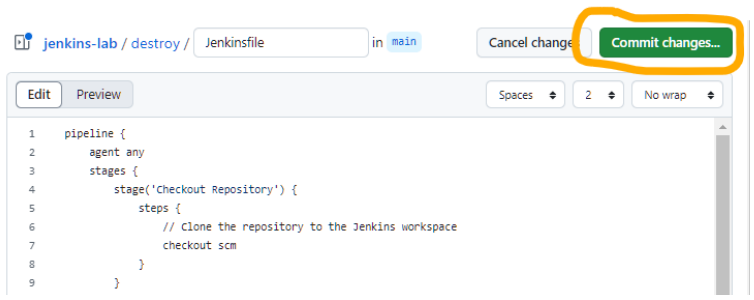




Note: Entering **destroy/** will create the **destroy** folder



12. Use a text editor such as notepad to open **Jenkinsfile.txt** in your **Student Material/lab6_Files/destroy** folder and copy the contents into your new github file. Then commit the changes..

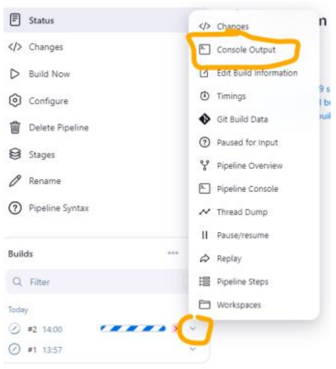


```
pipeline {
    agent any
    stages {
        stage('Checkout Repository') {
            steps {
                // Clone the repository to the Jenkins workspace
                checkout scm
            }
        }
```

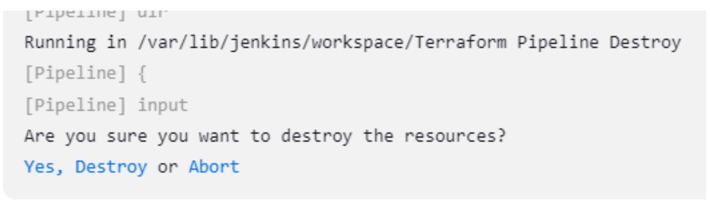
13. Switch back to Jenkins and **Build** the pipeline

14. This Jenkinsfile mandates that approval must be granted for the deletion to proceed.

15. Click on the pipeline and under **Builds**, select the running Terraform Pipeline Destroy job and select **Console Output**..



16. The run is waiting for approval to continue…



```
[Pipeline] dir
Running in /var/lib/jenkins/workspace/Terraform Pipeline Destroy
[Pipeline] {
[Pipeline] input
Are you sure you want to destroy the resources?
Yes, Destroy or Abort
```

17. Click on **Yes** to confirm the deletion

18. The destruction should now proceed. Switch to **S3** to verify the deletion of your test bucket.

***** Congratulations, you have completed the final lab of the course. Your instructor will end your lab environment for you which will destroy all AWS resources created. Destroy your Github repository at your own discretion or retain it for future use *****