QATIP Intermediate
AWS Lab8
Jenkins Terraform Pipelining

Contents

Lab Objectives

In this lab you will:

- Deploy a network and an EC2 instance with Jenkins installed via a script.
- Configure an S3 remote backend
- Configure and test terraform pipelining using Jenkins to deploy, modify and destroy AWS resources

Teaching Points

This lab introduces the concept of Terraform pipelining with Jenkins, demonstrating how infrastructure as code (IaC) can be automated for consistent and repeatable deployments in AWS. You will gain hands-on experience in setting up a Jenkins pipeline to execute Terraform configurations, allowing them to deploy, modify, and destroy AWS resources efficiently. By

integrating Terraform with Jenkins, users can enforce version control, automate approvals, and implement CI/CD best practices for infrastructure management.

Key concepts covered include provisioning infrastructure with Terraform, configuring Jenkins to automate deployments, and managing Terraform state remotely using an S3 backend. Additionally, you will explore how Jenkins integrates with GitHub for source control, manage AWS credentials within Jenkins securely, and implement automated triggers using GitHub Webhooks. By the end of the lab, you should understand how to structure Terraform pipelines in Jenkins, troubleshoot deployment issues, and automate resource lifecycle management, ensuring scalable and reliable infrastructure provisioning in AWS.

Solution

Given the nature of this lab, there is no solution section. Please reach out to your instructor if you encounter any issues

Before you begin

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 a Jenkins EC2 instance

1. Examine the script file create_ec2.sh in awslabs/06. This will create network resources and an EC2 instance in us-west-2 into which Jenkins is installed. Make the script executable…

   chmod +x create_ec2.sh

2. Run the script

   ./create_ec2.sh

3. You can continue with the next task whilst the script runs.

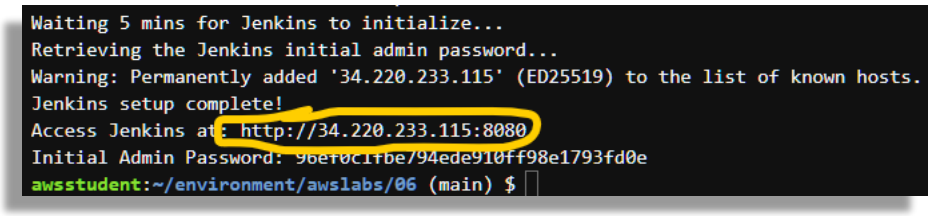
Task2 Create an S3 Bucket for Remote State

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.

Task3 Configure Jenkins

1. Open Jenkins in a new browser tab using the url displayed in your IDE. (Your IP address will differ)

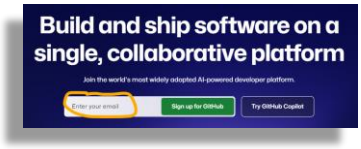


2. Copy and paste the admin password from the IDE into the **Unlock Jenkins** screen the click on Continue
3. Select "**Install suggested plugins**"
4. On the "**Create First Admin User**" screen; Select "**Skip and continue as admin**"
5. Click "**Save and Finish**" to complete the Jenkins configuration.
6. Click "**Start using Jenkins**"

Task4 Create a Github account and repository

Note: The process that follows was correct at time of writing. Github enrolment steps may change over time, so apply your own logic as you work through the process if it does not exactly match the steps that follow.

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"…

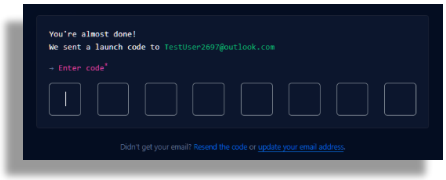


c. Enter a password and a unique username of your choice..

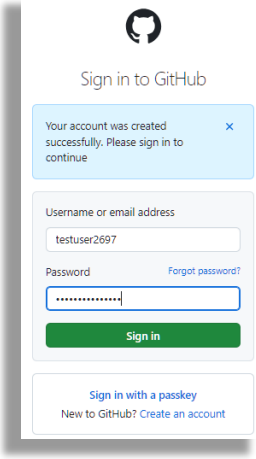
d. Record and save your chosen **username** and **password** in your session-info file for safekeeping.

e. Complete the challenge to prove you are a human…

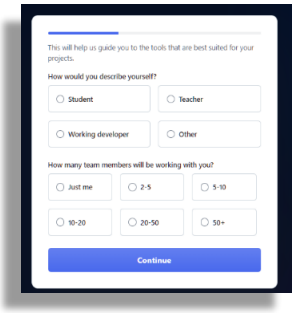
f. An email will sent containing your launch code. Retrieve this and enter it..



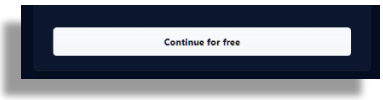
g. You will then be prompted to log into github using your new account..



h. Complete the questionnaire…

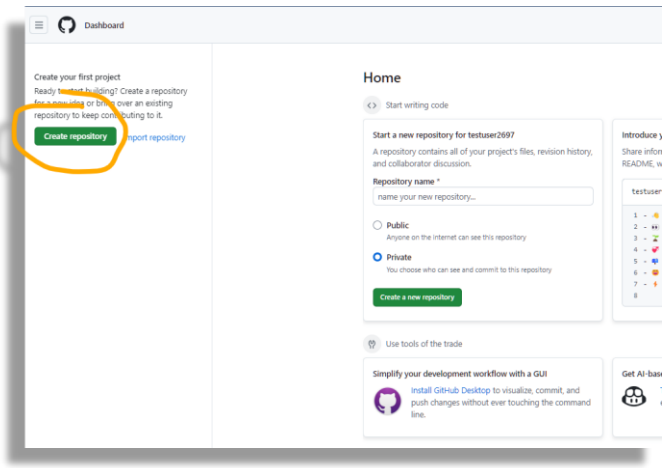


i. When asked to select a subscription, select "**Continue for free**"..

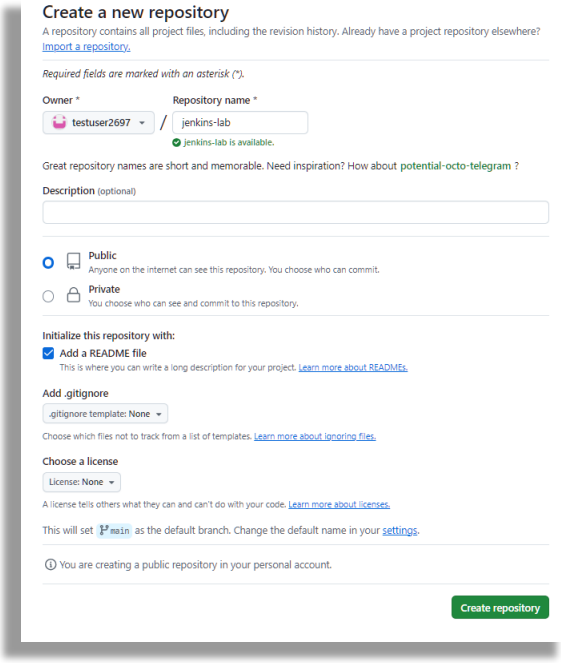


2. Create a public repository …

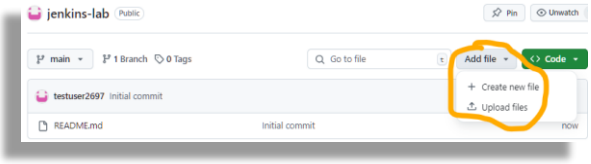
a. Click on New …



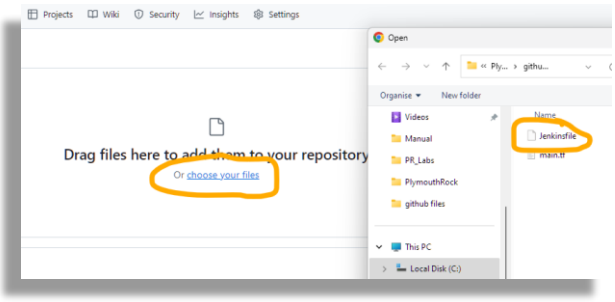
b. Enter a Repository name of your choice. Ensure **Public** is selected and check the '**Add a README file'** option. Then click on **Create repository**…



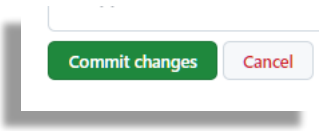
c. Click on **Add file**, **Upload file**..



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

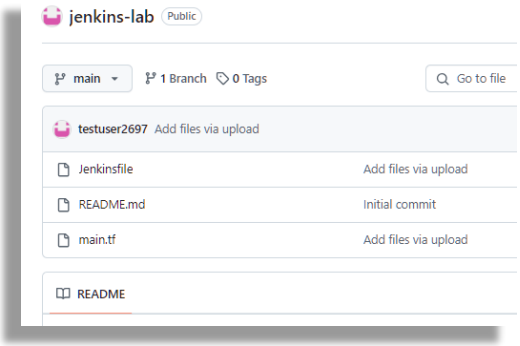


e. Click on Commit changes..

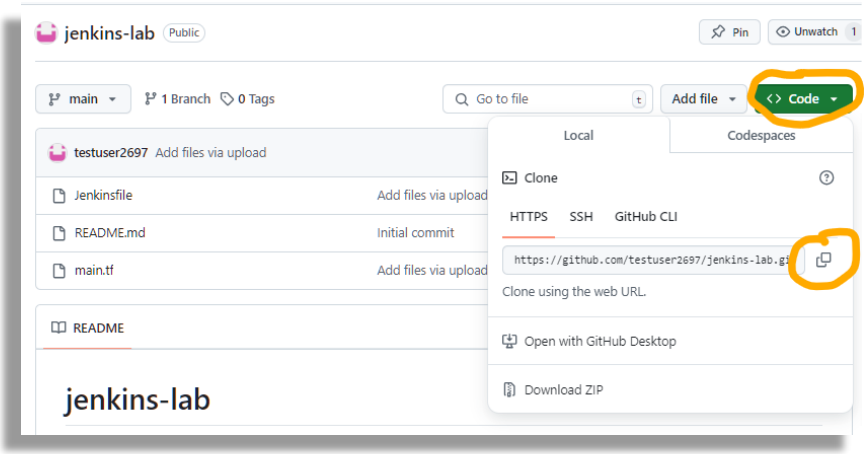


f. Repeat the previous 2 steps to upload the **main.tf** file

g. The 2 files should now be listed..



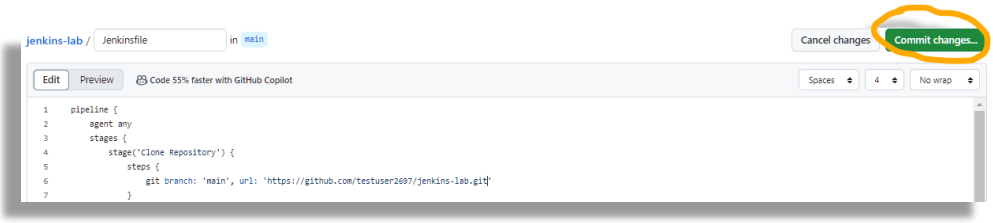
h. Copy the URL of your repository to your clipboard..

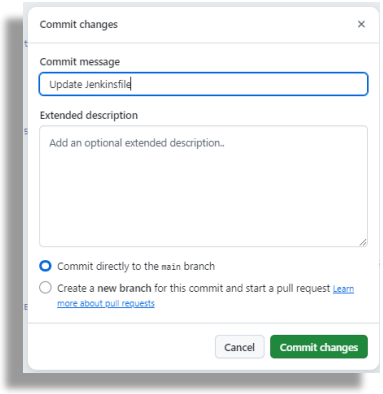


i. Record this URL in your **session-info** file against **Repo-URL**

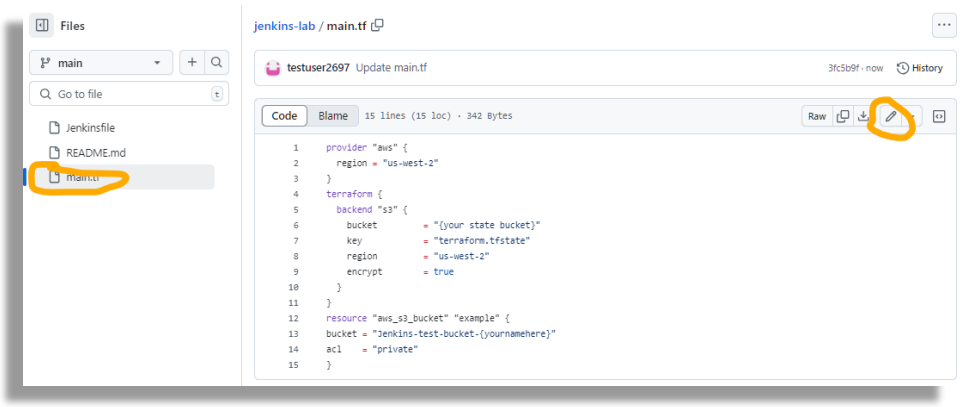
j. Click on the file **Jenkinsfile** and then click on the **edit** icon to open the file for editing…

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





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



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

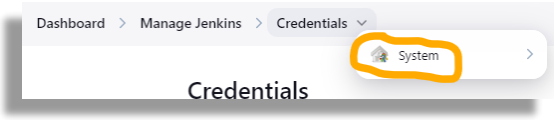
n. On line 13, replace **{younamehere},** including the braces, with your name followed by 2 random digits (to guarantee uniqueness). This will be the name of the bucket that Terraform will create when we run a Jenkins pipeline.

o. Commit these changes as before.

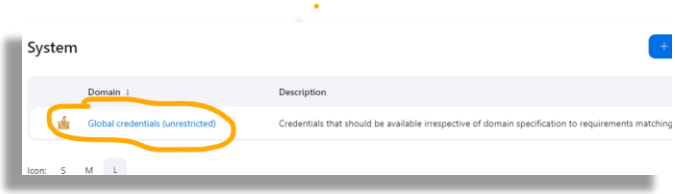
p. Leave the Github tab open as we will return to it later.

Task5 Add AWS Credentials to Jenkins

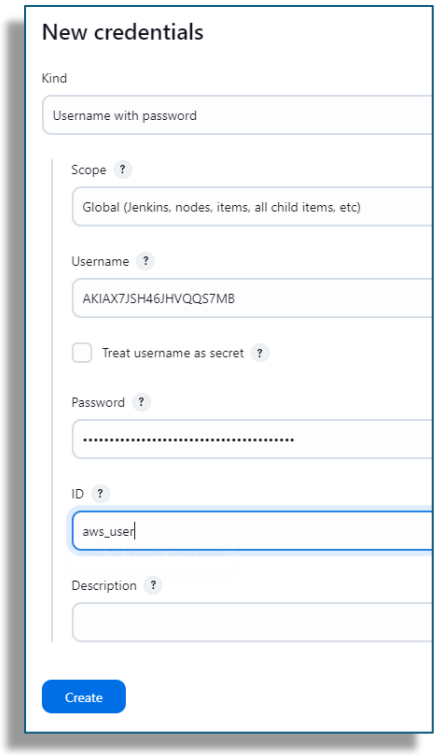
1. In order to use Terraform on Jenkins to interact with AWS, we must supply it with credentials to use. In your **Jenkins** browser session; Navigate to **Manage Jenkins > Credentials**.
2. On the breadcrumb menu; click on the **Credentials** dropdown and then select **System** …



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



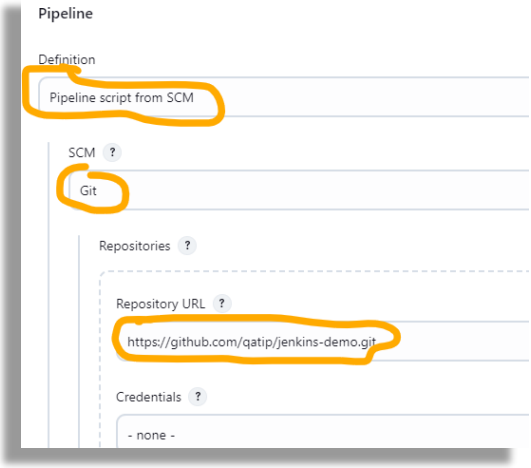
4. Click on "**Add Credentials**"
5. For Username: Enter the **Access key ID** generated for your lab user. (This and the Secret Access key required next, can be found in your **session-info** file)
6. For Password: Enter the **Secret Access key** generated for your lab user
7. For ID: enter **aws_user**
8. Select "**Create**"

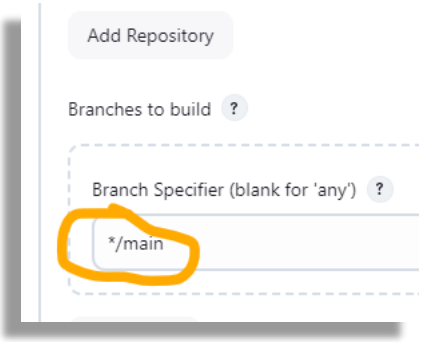


Task6 Configure Pipeline Job

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**

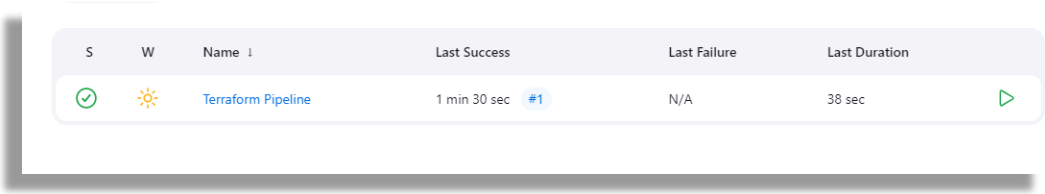




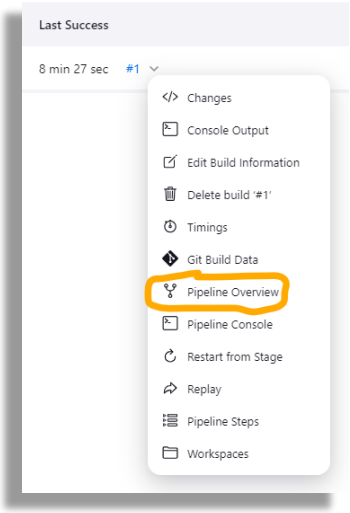
9. Click “**Save**”
10. Click “**Build Now**”

Task7 Verify pipeline run and explore Jenkins

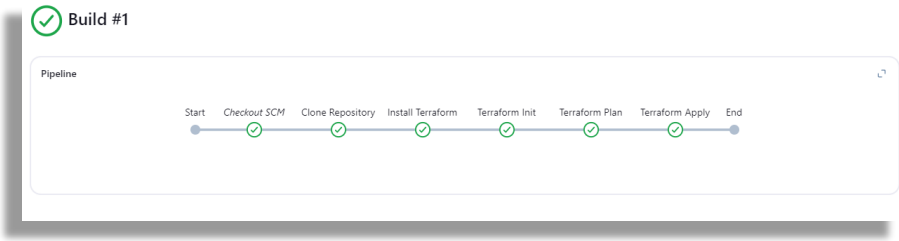
1. In Jenkins, return to the Dashboard. A record of the pipeline will be displayed showing run success and failure. Refresh the page until a result of the pipeline run is displayed…



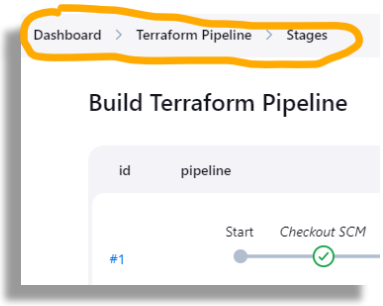
2. Select the drop-down menu against **#1** and choose Pipeline Overview..



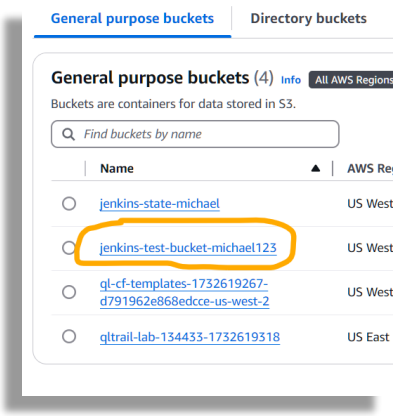
3. The stages of the pipeline are shown, with ticks (or crosses) indicating success or failure at that stage…



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



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

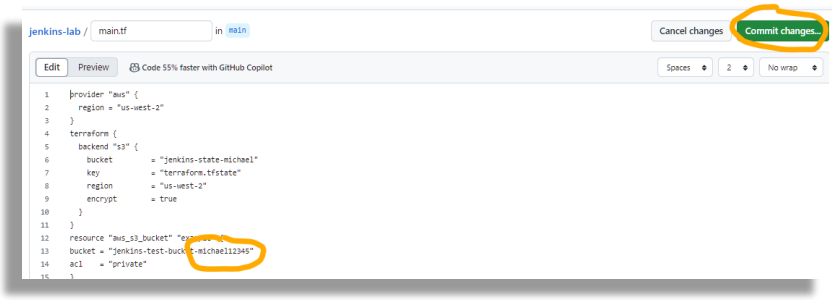


Task8 Updating the transformation pipeline

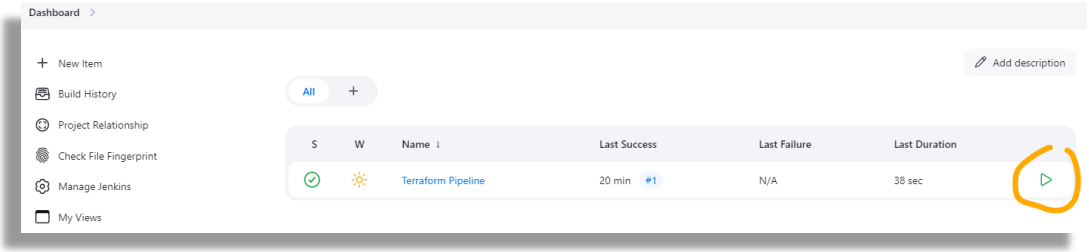
In a production environment, developers would now check-out the contents of the Github repo to their local machine, make updates to the terraform files and then check them back into the repo for approval. Once approved these changes would be merged with the current files and deployed by triggering a 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.

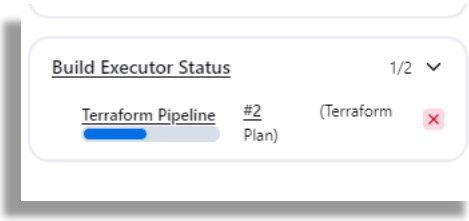
1. Return to your Github repository, logging back in if necessary.
2. Open **main.tf** for editing
3. Change the name of the bucket to be created by adding addition digits to the existing name. This will cause the original bucket to be deleted and a new one to be created. Commit this change..



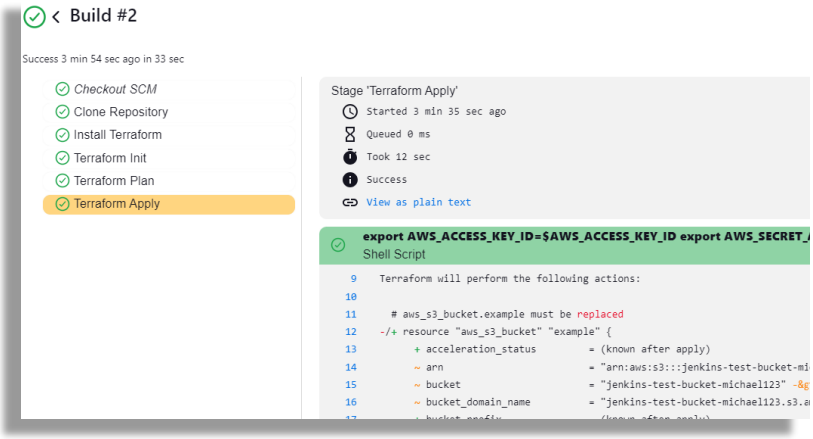
4. Switch to Jenkins and click on the play icon to schedule a manual running of the pipeline..



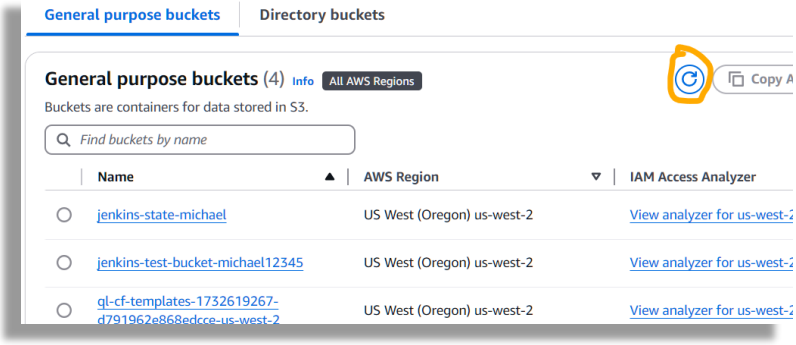
5. The build Executor Status will show the progress of the run..



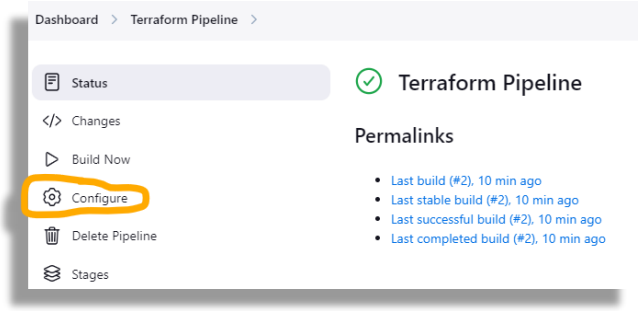
6. The success runs count should increase to #2 indicating successful manual running of the pipeline. (You may need to refresh the page)
7. Looking at the logs for the **Terraform Apply** phase we see that the old bucket was replaced as bucket names are immutable and cannot be changed…



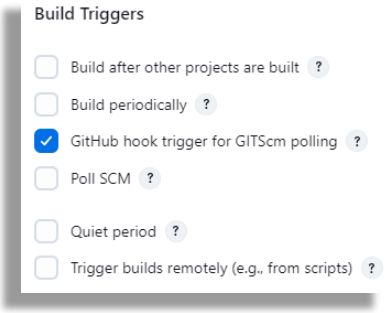
8. Switch to your console and, in S3, verify the deletion of the old bucket and the creation of a new one, refreshing the display if necessary..



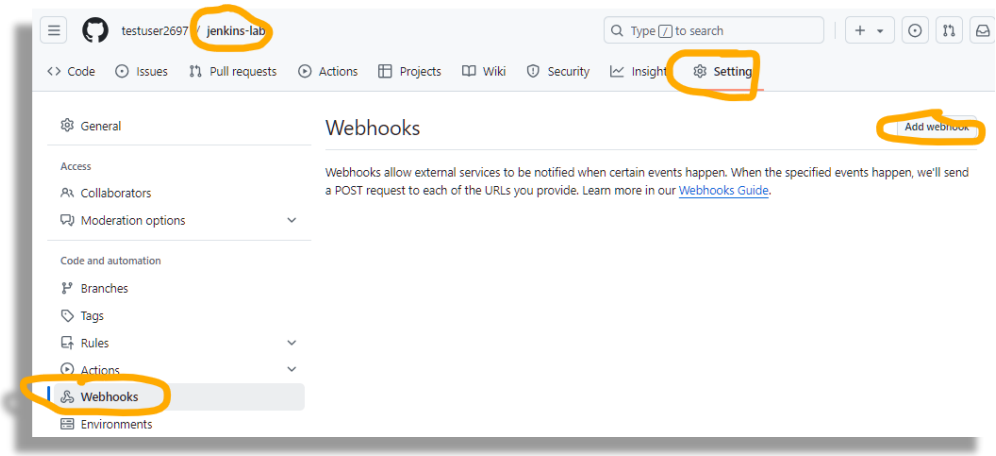
9. To configure automatic pipeline running; **In Jenkins**, configure the pipeline by first selecting it on the main dashboard and then choosing the "**Configure**" option..



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



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)…

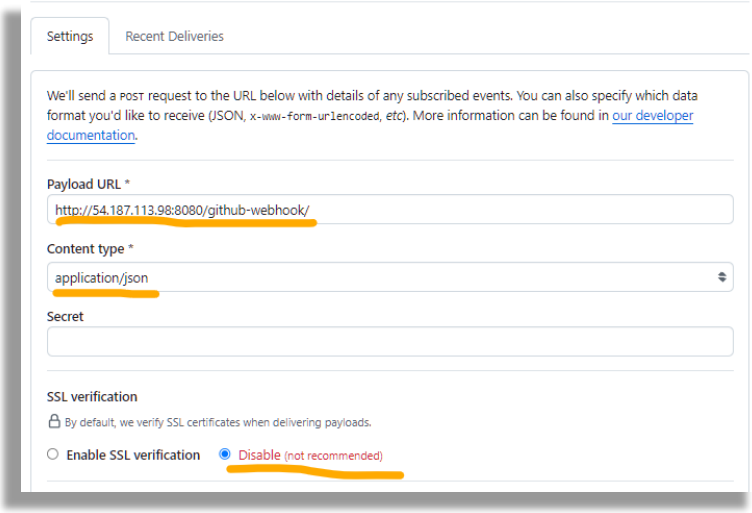


12. For **Payload URL;** enter **http://{Jenkins Public IP}:8080/github-webhook/** replacing **{Jenkins Public IP}** with the Public IP address of your Jenkins instance

13. For Content type; select **application/json**

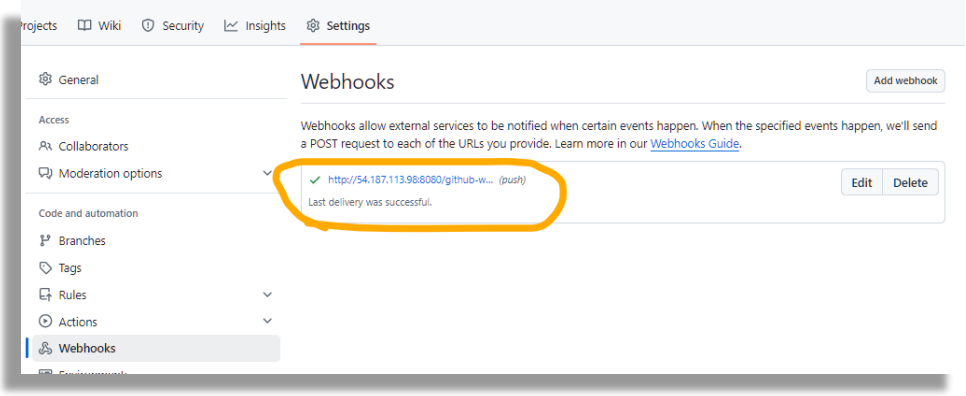
14. Select to disable **SSL verification** for this lab environment.

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

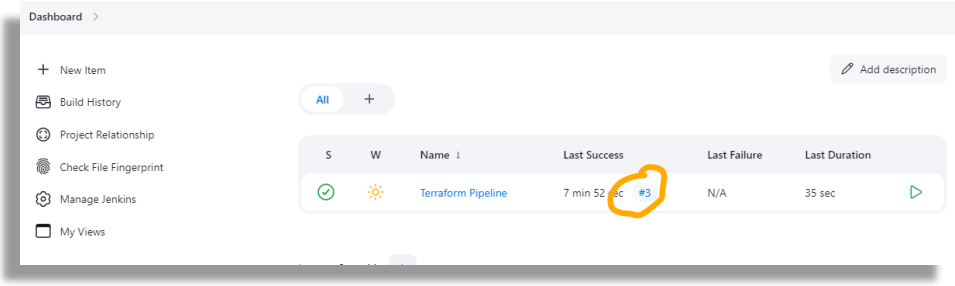


16. Make another change to the name of your bucket in **main.tf** and commit the changes (refer to steps 2 and 3 above if you need guidance)

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



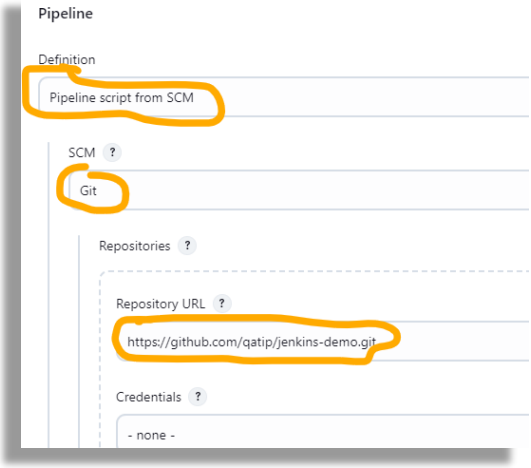
18. Switch to Jenkins. Return to the Dashboard and check that there is now a record of a third successful running of the pipeline…

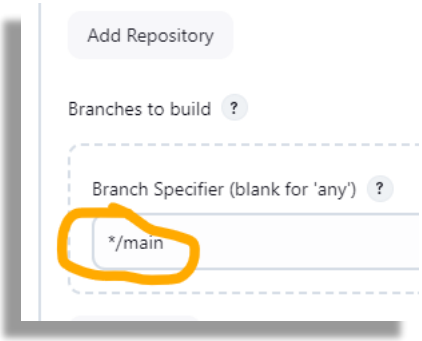


19. Switch to your AWS console, navigate to S3, refresh the display if necessary and verify the new bucket has been created.

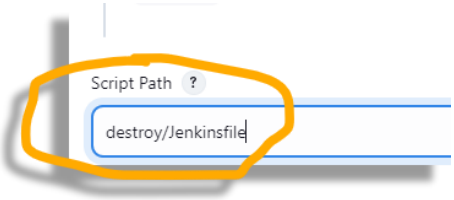
Task9 (Time permitting). Configure Destroy Pipeline Job

1. Select " **+ New Item**" from the Jenkins **dashboard**
2. Use "**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**





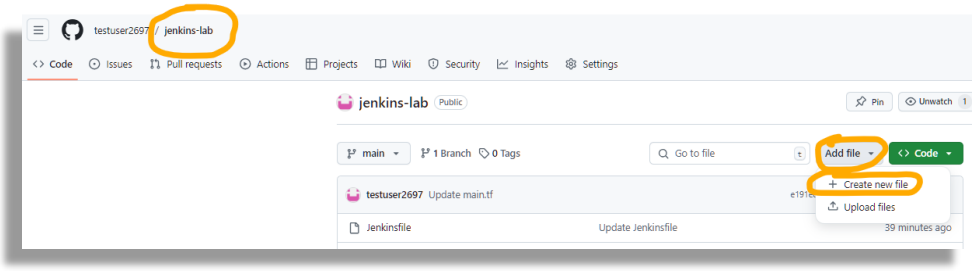
9. Change the Script Path to **destroy/Jenkinsfile**

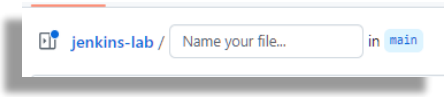


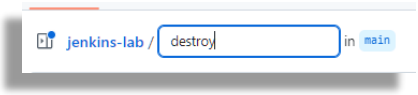
10. Save the new pipeline but do not build it yet

11. Switch to your Github account

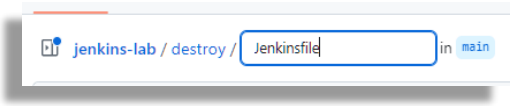
12. Create a new empty Jenkinsfile in a new folder “destroy”…







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

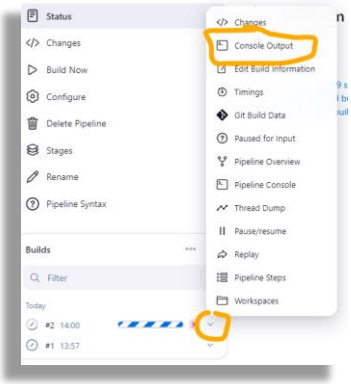


13. In your IDE, navigate to and to open **Jenkinsfile.txt** in your **Lab Instructions/lab6_Files/destroy** folder and copy the contents into your new github file. Then commit the changes..

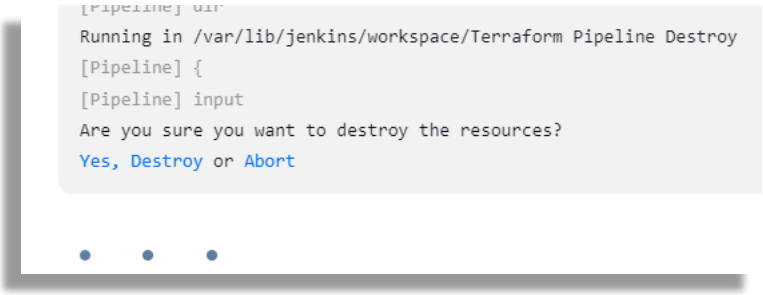
14. Switch back to Jenkins and **Build** the pipeline

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

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



17. The run is waiting for approval to continue…



18. Click on **Yes** to confirm the deletion
19. 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. If you wish to attempt optional lab06a then please do so now. Your instructor will end your lab environment for you at course end, which will destroy all AWS resources created. Destroy your Github repository at your own discretion or retain it for future use *****