QATIPv3 Lab6
Jenkins Setup and Terraform Pipeline Configuration

Contents

Start Lab

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

   cd ~/googlelabs/lab06
3. This shifts your current working directory to googlelabs/lab06. ***Ensure all IDE commands are executed in this directory***

Task1 Create a Jenkins GCE instance

1. Examine the script file **create_jenkins_gce.sh**. This will create an GCE instance in us-west-2 into which we will install Jenkins.
2. Update lines **4** and **15**, replacing **<your lab project id>** with your lab project id, there are **3** replacements in total
3. The script file **jenkins_startup_script.sh** will install jenkins into the instance.

4. Make these script executable...

   chmod +x create_jenkins_gce.sh

   chmod +x jenkins_startup_script.sh

5. Run the create script

   ./create_jenkins_gce.sh

6. Click on **Authorize** when prompted to Authorize Cloud Shell

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

Task2 Create an GCS Bucket for Remote State

1. Switch to the Console.
2. Search for and then navigate to the **Cloud Storage** service. Click on **Create bucket**
3. 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***
4. Leaving all settings at their default values, scroll down and select Create bucket.
5. Click on **Confirm** for Public access prevention

Task3 Update main.tf

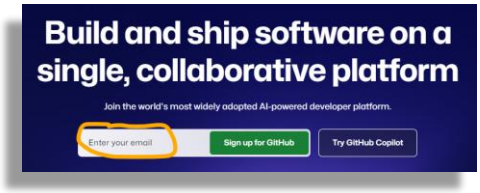
1. In the IDE, open **main.tf** for editing
2. Update line **2** with your **lab project id**
3. Update line **8** with the name of the storage bucket you have created
4. Download **main.tf** (right-click over the file and select Download) to your local machine for later uploading to Github.

Task4 Create a Github account and repository

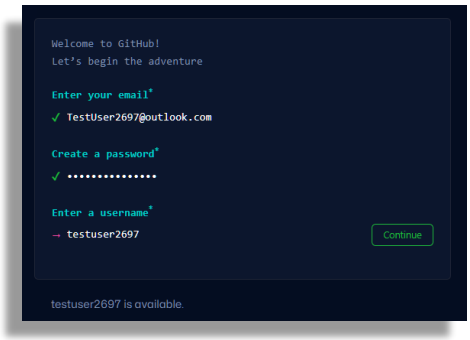
1. Open a new browser session and Sign up to Github using a personal email address that is not currently associated with Github...

a. Navigate to https://github.com/

b. Enter a personal email address and select “Sign up for Github”...



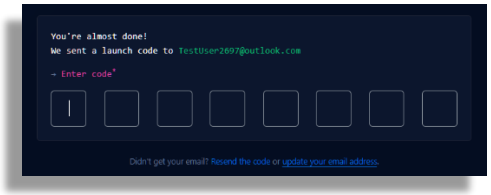
c. Enter a password and a unique username of your choice..



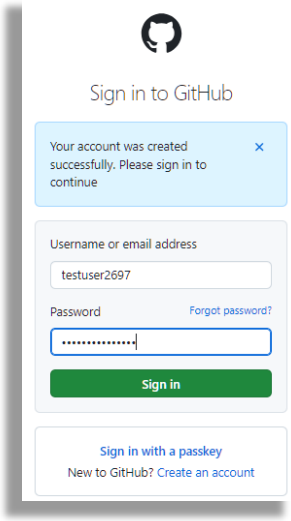
d. Record and save your chosen **username** and **password** in your session-info file for safekeeping.

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

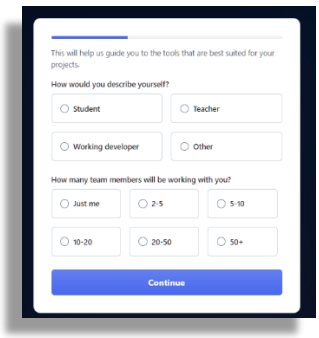
f. An email will sent containing your launch code. Retrieve this and enter it..



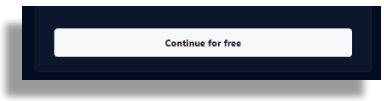
g. You will then be prompted to log into github using your new account..



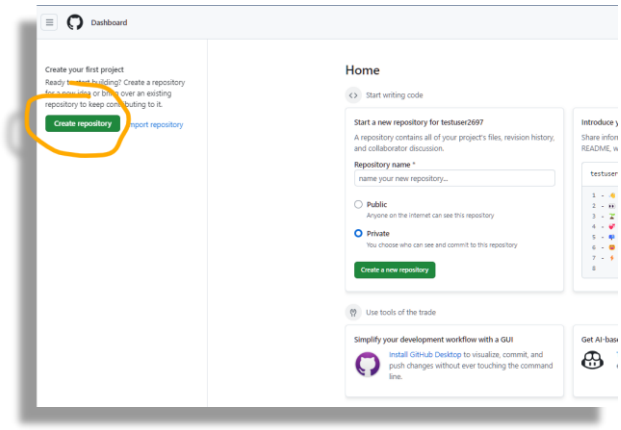
h. Complete the questionnaire...



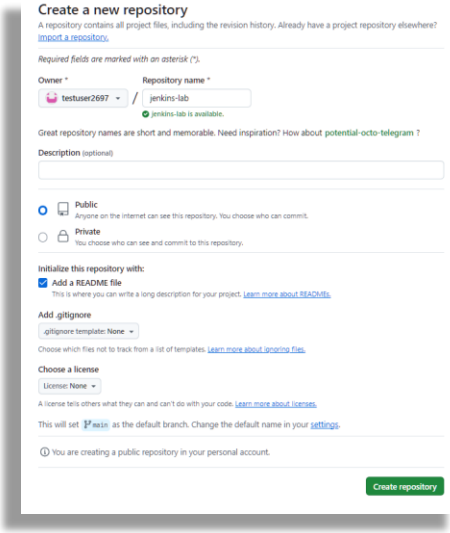
i. When asked to select a subscription, select "**Continue for free**"..



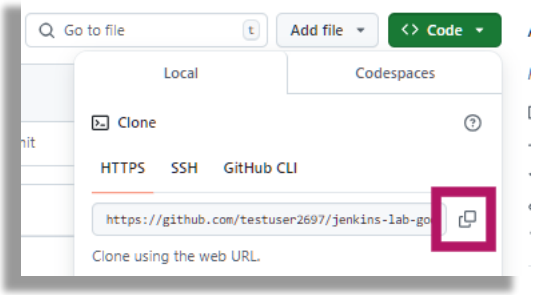
2. Create a public repository ...
   a. Click on New ...



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. Copy the repository url...

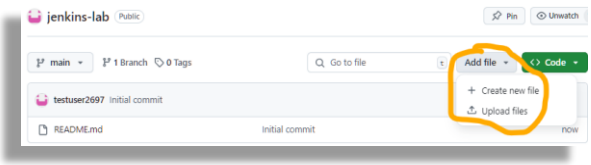


d. Switch to the IDE and update line 6 of **Jenkinsfile** with your repo url...

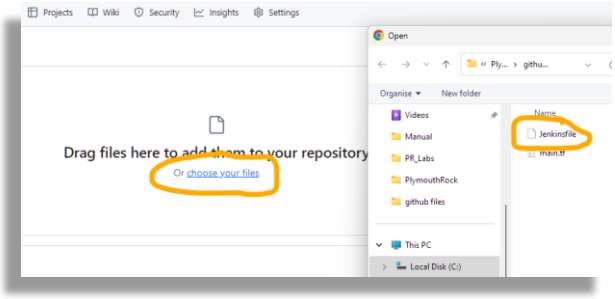
```
1  pipeline {
2      agent any
3      stages {
4          stage('Clone Repository') {
5              steps {
6                  git branch: 'main', url: '{your github repo url here}'
7              }
```

e. Download the modified file to your local machine. If necessary, remove the **.txt** extension that may be added during the download.

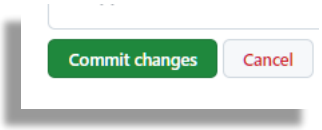
f. Back in Github...Click on **Add file**, **Upload file**..



g. Select the file **Jenkinsfile** you just downloaded..

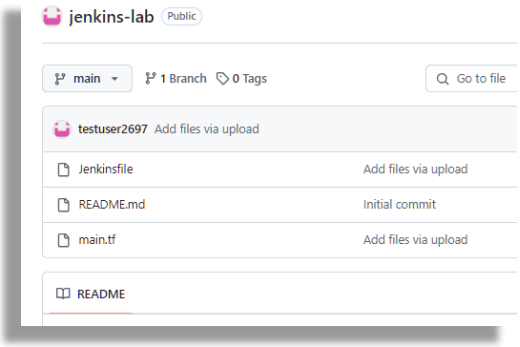


h. Click on Commit changes..



i. Repeat the previous 3 steps to upload the **main.tf** file you downloaded earlier

j. The 2 files should now be listed..

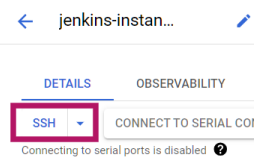


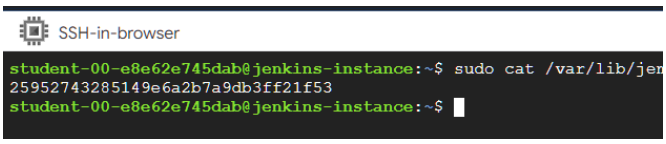
k. Leave the Github tab open as we will return to it later.

Task5 Configure Jenkins

1. Switch to your IDE and wait until the Jenkins installation completes
2. Using the GCE console, SSH into the Jenkins instance and retrieve the initial Jenkins password by running the following command...

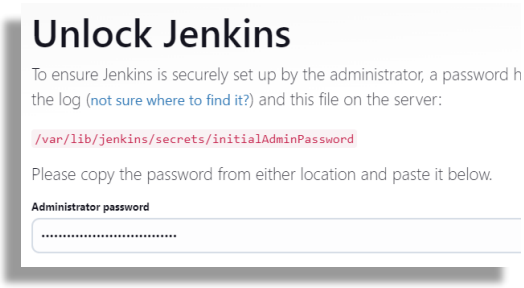
   sudo cat /var/lib/jenkins/secrets/initialAdminPassword



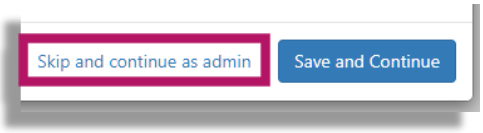


3. Record this in your session file (yours will differ) and then close the SSH session
4. Open Jenkins in a new browser tab using the url displayed

5. Paste the admin password from the IDE into the **Unlock Jenkins** screen the click on Continue..



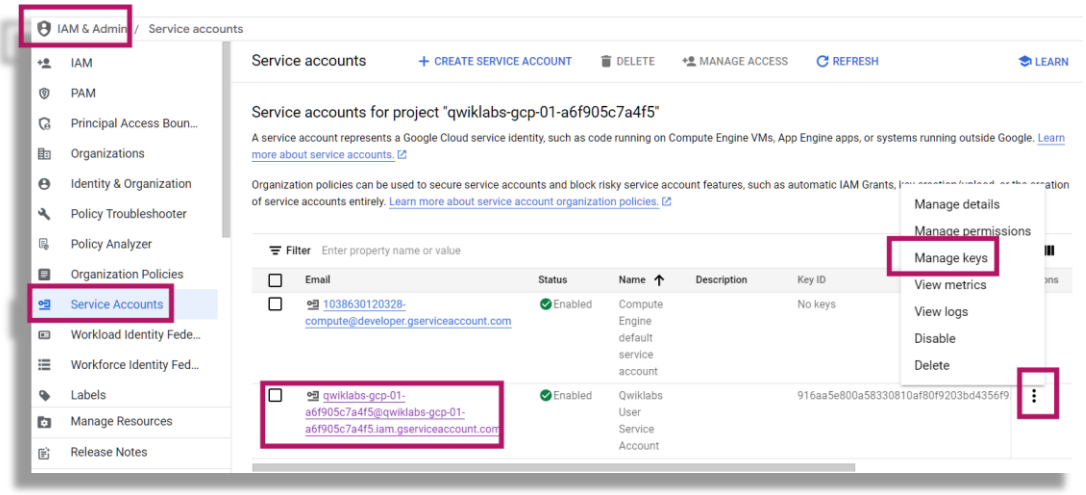
6. Select "**Install suggested plugins**"
7. On the "**Create First Admin User**" screen; Select "**Skip and continue as admin**"



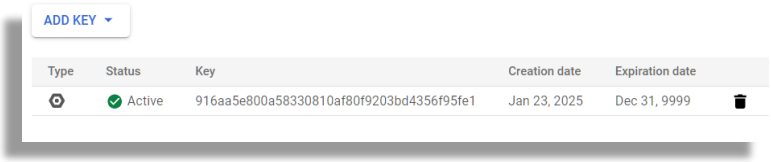
8. Click "**Save and Finish**" to complete the Jenkins configuration.
9. Click "**Start using Jenkins**"
10. Leave this browser session open as we will return to it.

Task6 Add Google Cloud Credentials to Jenkins

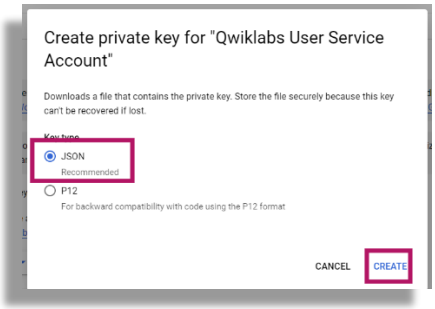
1. In order to use Terraform on Jenkins to interact with Google Cloud, we must supply it with credentials to use, typically in the form of a service account json key file.
2. In your Google Cloud console, navigate to **IAM & Admin, Service Accounts**. Select the Actions menu against your Qwiklabs service account and select **Manage Keys**...



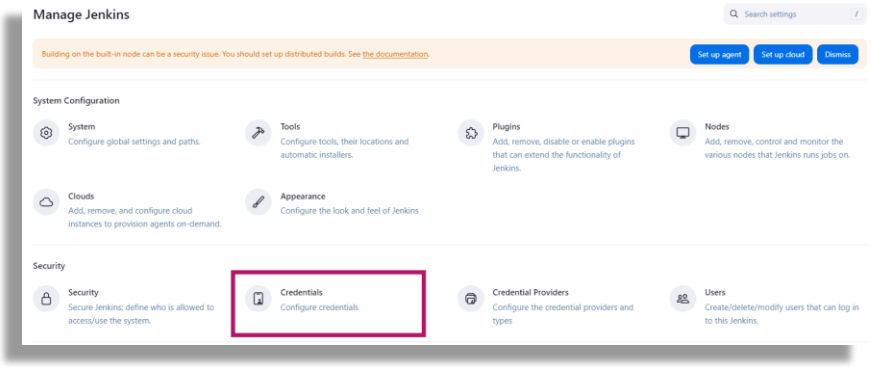
3. Select **ADD KEY, Create new key**..



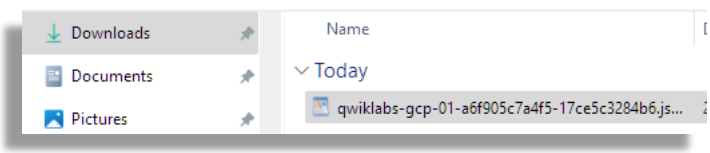
4. Select JSON type and click CREATE...



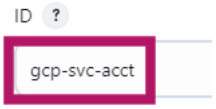
5. Switch to your **Jenkins** browser session and Navigate to **Manage Jenkins > Credentials**....

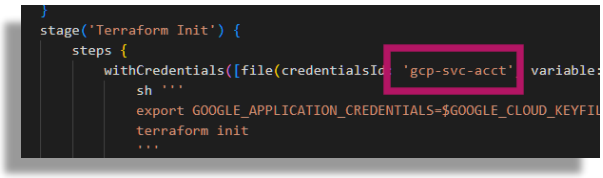


6. On the breadcrumb menu; click on the **Credentials** dropdown and then select **System** ...
7. Click on "**Global credentials (unrestricted)**"...
8. Click on "**Add Credentials**"
9. For **Kind**, select **Secret File**, select **Choose file**, browse to your downloads and select the json file you previously downloaded...



10. For ID, enter **gcp-svc-acct** (this can be any value but here it must match the credential used in the provided Jenkinsfile)

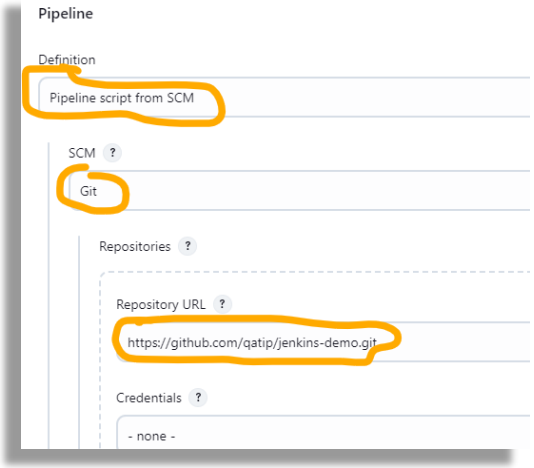


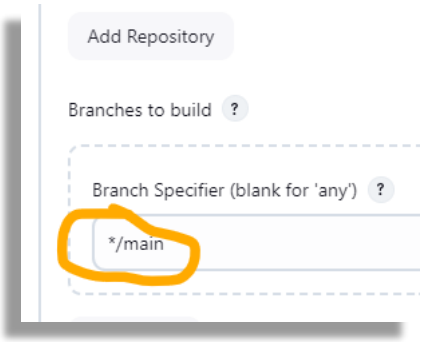


11. Select "**Create**"

Task7 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**

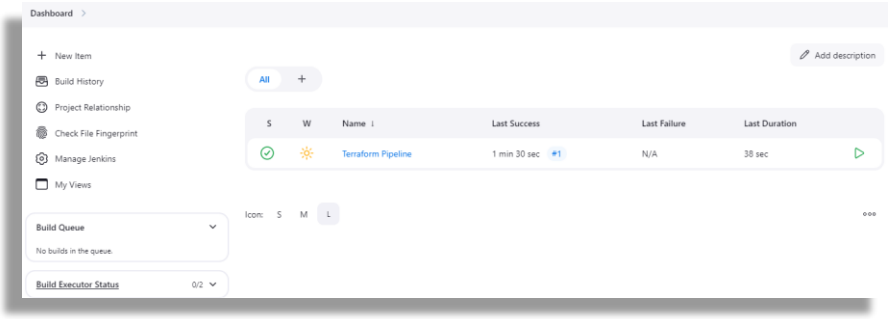




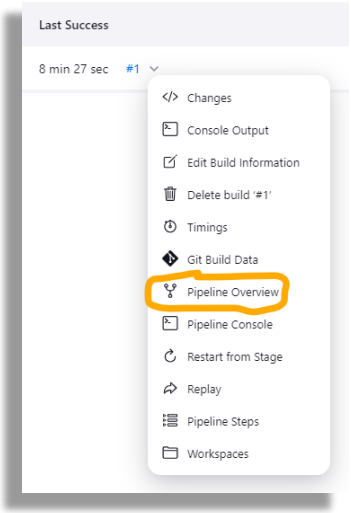
9. Click “**Save**”
10. Click “**Build Now**”

Task8 Verify pipeline run and explore Jenkins

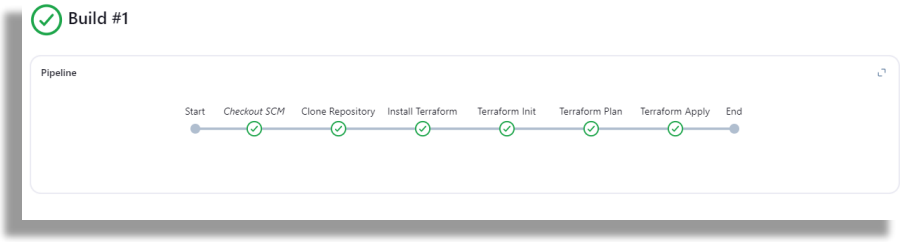
1. In Jenkins, return to the Dashboard. A record of the pipeline will be displayed showing run success and failure. Refresh the page until a result of the pipeline run is displayed...



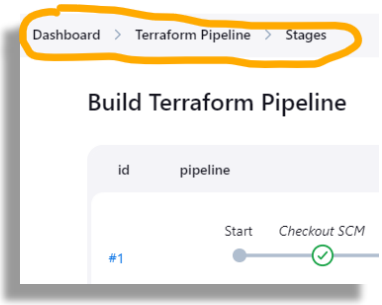
2. Select the drop-down menu against **#1** and choose Pipeline Overview..



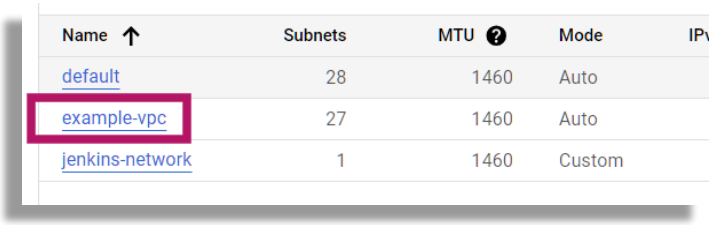
3. The stages of the pipeline are shown, with ticks (or crosses) indicating success or failure at that stage...



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 Google Cloud, navigate to the VPC service and verify the existence of your new network..

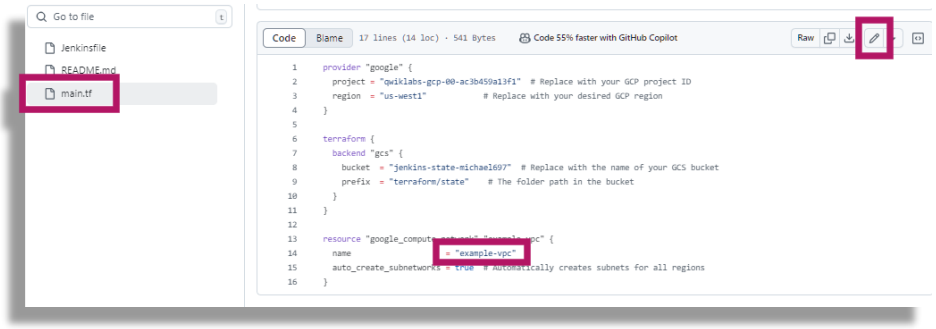


Task9 Updating the transformation pipeline

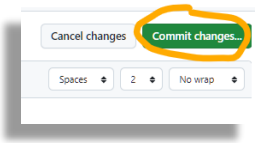
In a production environment developers would now check-out the contents of the Github repo to their local machine, make updates to the terraform files and then check them back into the repo for approval. Once approved these changes would be merged with the current files and deployed by triggering a new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

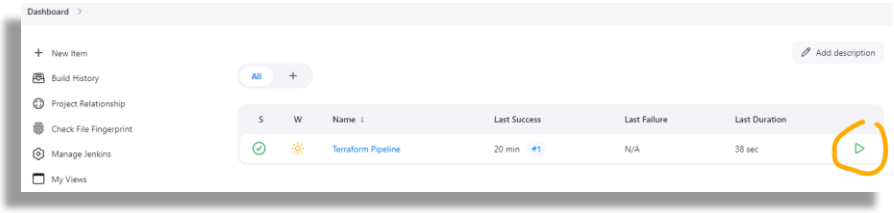
1. Return to your Github repository, logging back in if necessary.
2. Open main.tf for editing..



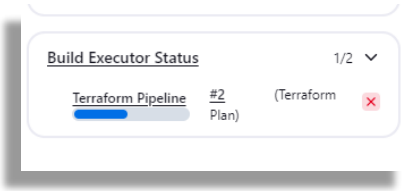
3. Change the name of the VPC to be created by adding addition digits to the existing name. This will cause the original VPC to be deleted and a new one to be created. Commit this change..



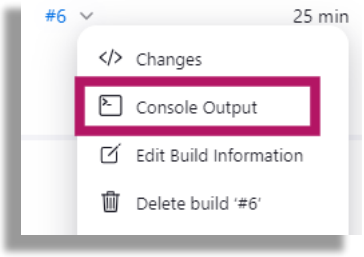
4. Switch to Jenkins and click on the play icon to schedule a manual running of the pipeline..



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



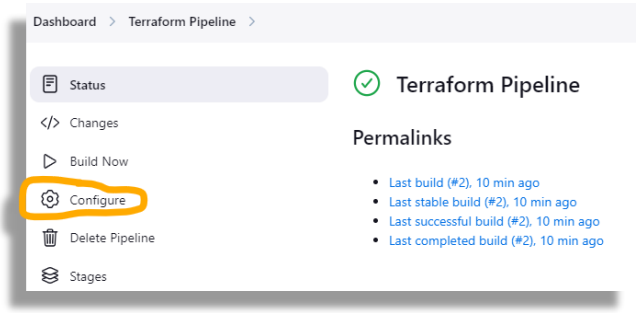
6. The success runs count should increase to #2 indicating successful manual running of the pipeline. (You may need to refresh the page)
7. Click to the right of the run number and from the drop-down, select **Console Output**



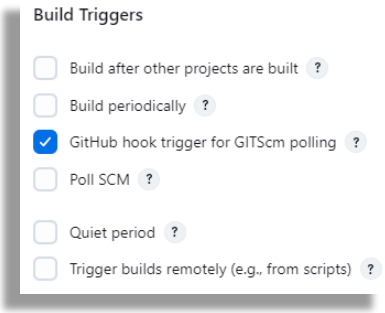
8. Scroll down to confirm the delete/recreate operation was successful..

```
Apply complete! Resources: 1 added, 0 changed, 1 destroyed.
[0m
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

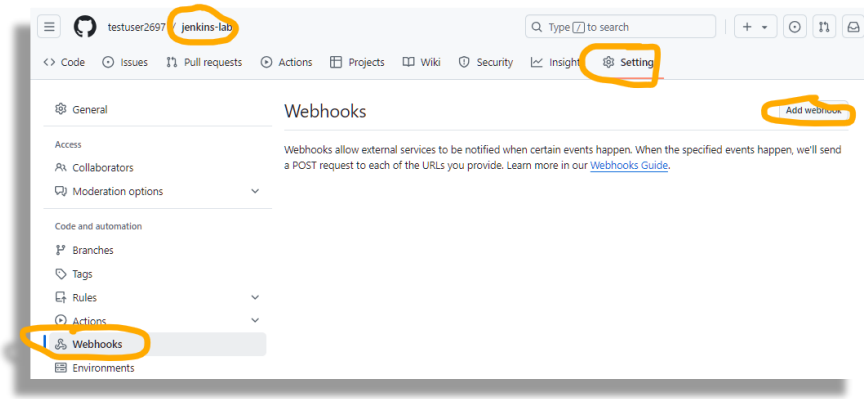
9. Switch to your console and verify the creation of the new VPC
10. To configure automatic pipeline running; **In Jenkins**, configure the pipeline by first selecting it on the main dashboard and then choosing the "Configure" option..



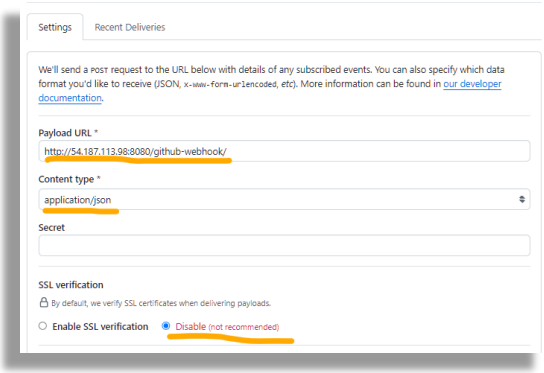
11. Scroll down to the **Build Triggers** section, select "**Github hook trigger for GITScm polling**" and click on **Save** ...



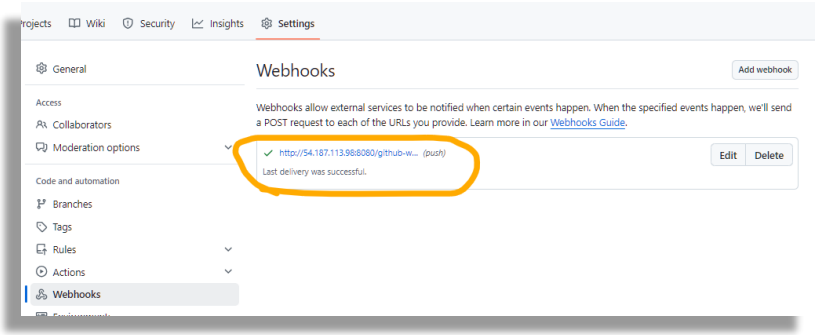
12. **Switch to Github**. Select the **Settings** for your repo. Scroll down and select **Webhooks**. Click on **Add webhook** (you may be prompted to re-authenticate at this point)...



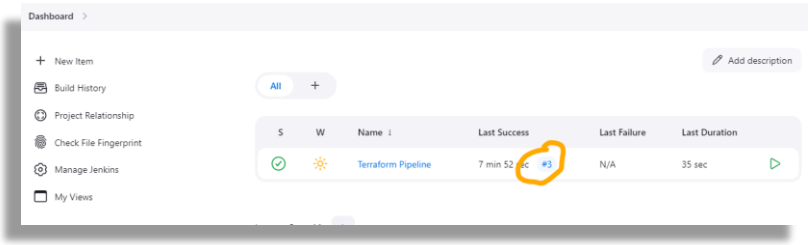
13. For **Payload URL;** enter **http://{Jenkins Public IP}:8080/github-webhook/** replacing {Jenkins Public IP} with the Public IP address of your Jenkins instance
14. For Content type; select **application/json**
15. Select to disable **SSL verification** for this lab environment.
16. Verify your settings as shown in example below (your IP will differ) before clicking on **Add webhook**..



17. Make another change to the name of your VPC in main.tf and commit the changes (refer to steps 2 and 3 above if you need guidance)
18. Re-visit your Webhooks setting and you should see confirmation that there was a successful push of the changes to Jenkins...



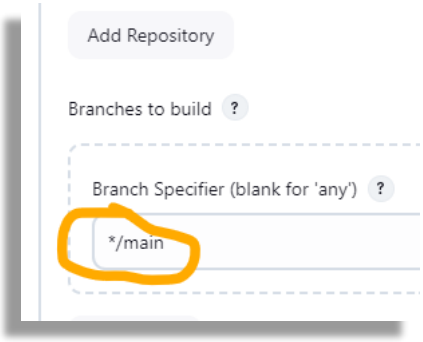
19. Switch to Jenkins. Return to the Dashboard and check that there is now a record of a third successful running of the pipeline...

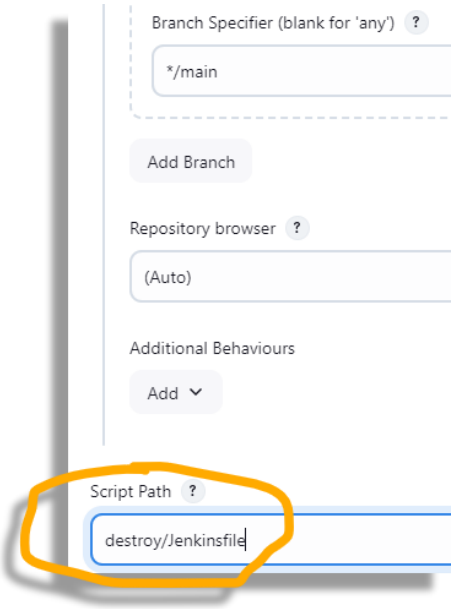


20. Switch to your Cloud console and verify the new VPC has been created.

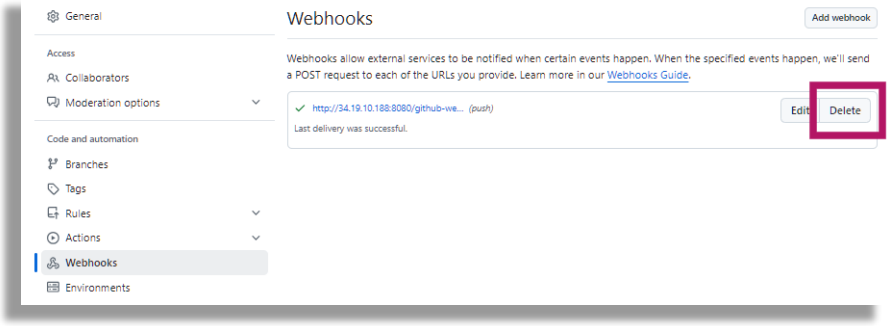
Task10 (Time permitting). Configure Destroy Pipeline Job

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**

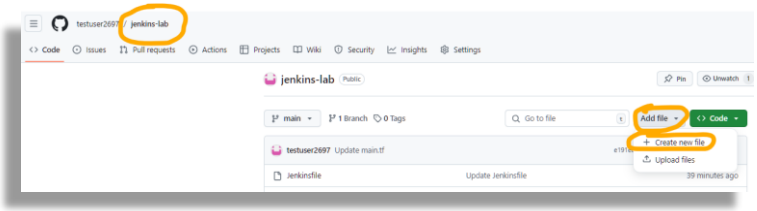


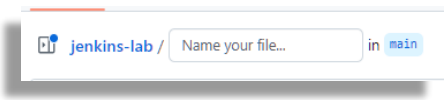


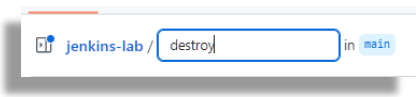
9. Save the new pipeline **but do not build it yet**
10. Switch to your Github account
11. Delete the previously configured Webhook..



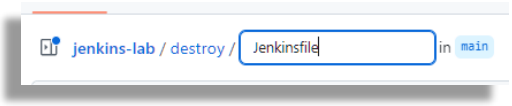
12. Create a new empty Jenkinsfile in a new folder “destroy”...







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

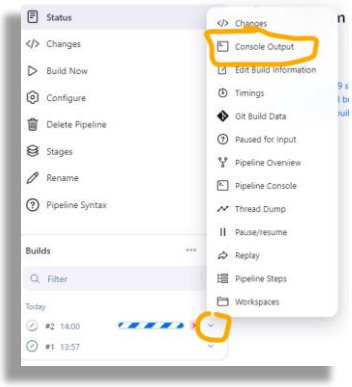


13. In your IDE, navigate to and open **Jenkinsfile.txt** in your **lab06/destroy** folder and copy the contents into your new github file. Then commit the changes..

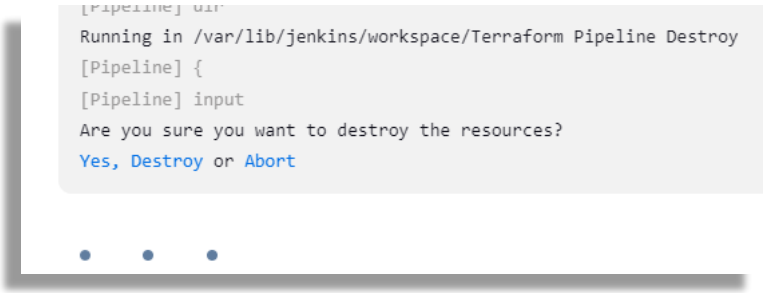
14. Switch back to Jenkins and **Build** the pipeline

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 the console to verify the deletion of your VPC

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