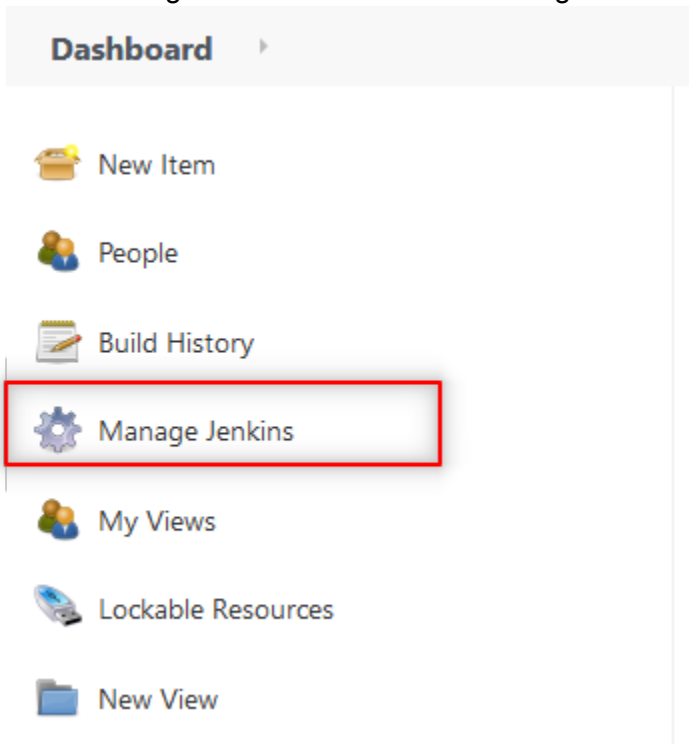


# Deploy AWS Cloud Infra using Terraform IaC Automated using Jenkins Pipeline.


## Configure Terraform on Jenkins


1. Click Manage Jenkins from left hand navigation.





2. Select Manage Plugins from System Configuration section.

### System Configuration


 **Configure System**  
Configure global settings and paths.


 **Global Tool Configuration**  
Configure tools, their locations and automatic installers.


 **Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.  
▲ There are updates available

 **Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

### Security

 **Configure Global Security**  
Secure Jenkins: define who is allowed to access/use the system.

 **Manage Credentials**  
Configure credentials

 **Configure Credential Providers**  
Configure the credential providers and types

 **Manage Users**  
Create/delete/modify users that can log in to this Jenkins

### Status Information

3. Click the Available tab and search Terraform.

The screenshot shows the Jenkins Update Center interface. At the top, there is a search bar with the text "terraform" entered. Below the search bar, there are tabs for "Updates", "Available", "Installed", and "Advanced". The "Available" tab is selected. Below the tabs, there is a table with columns "Name", "Version", and "Released". The table contains one entry for "Terraform" with version "1.0.10" and release date "1 yr 4 mo ago". Below the table, there are buttons for "Install without restart", "Download now and install after restart", and "Check now".

Name	Version	Released
Terraform	1.0.10	1 yr 4 mo ago

4. Select Terraform and click Install without restart.

The screenshot shows the "Installing Plugins/Upgrades" page in Jenkins. The page has a heading "Installing Plugins/Upgrades" and a section "Preparation" with a list of steps: "Checking internet connectivity", "Checking update center connectivity", and "Success". Below this, there are two rows: "Terraform" and "Loading plugin extensions", both with a green checkmark and the word "Success". At the bottom, there are two links: "Go back to the top page" and "Restart Jenkins when installation is complete and no jobs are running".

## Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Terraform ✓ Success

Loading plugin extensions ✓ Success

[Go back to the top page](#)  
(you can start using the installed plugins right away)

☐ Restart Jenkins when installation is complete and no jobs are running

5. Click Manage Jenkins from left hand navigation. Click Global Tool Configuration from System configuration section.

## System Configuration



### Configure System

Configure global settings and paths.



### Global Tool Configuration

Configure tools, their locations and automatic installers.



### Manage Plugins

Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

There are updates available

6. Scroll down to the Terraform section and click Add Terraform. Enter a Name of your choice. I'm going to use "Terraform" to make things simple. Ensure Install automatically is NOT selected. It will be selected by default.

**Terraform**

Terraform installations

[Add Terraform](#)

Terraform

Name

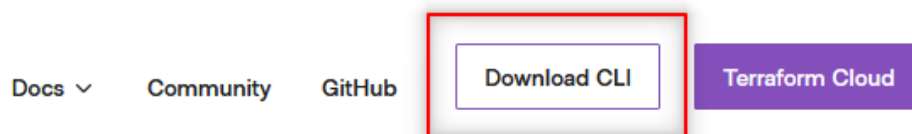
Terraform

Install directory

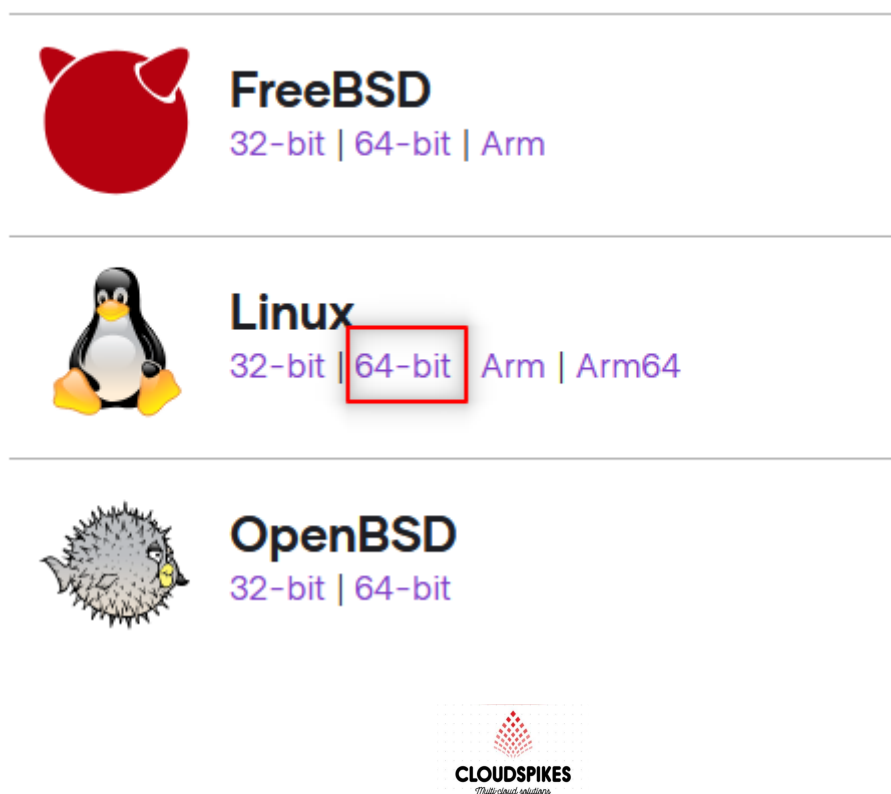
☐ Install automatically

[Delete Terraform](#)

7. We will need to install Terraform onto our Jenkins server via Terminal bash on the same machine where you hosted Jenkins server. Once you are on your Jenkins instance, navigate to terraform.io in your browser. Click Download CLI.



8. Scroll down to Linux. Right click 64-bit and copy link.



9. In your terminal run the following command.  
\$ wget <copied url>

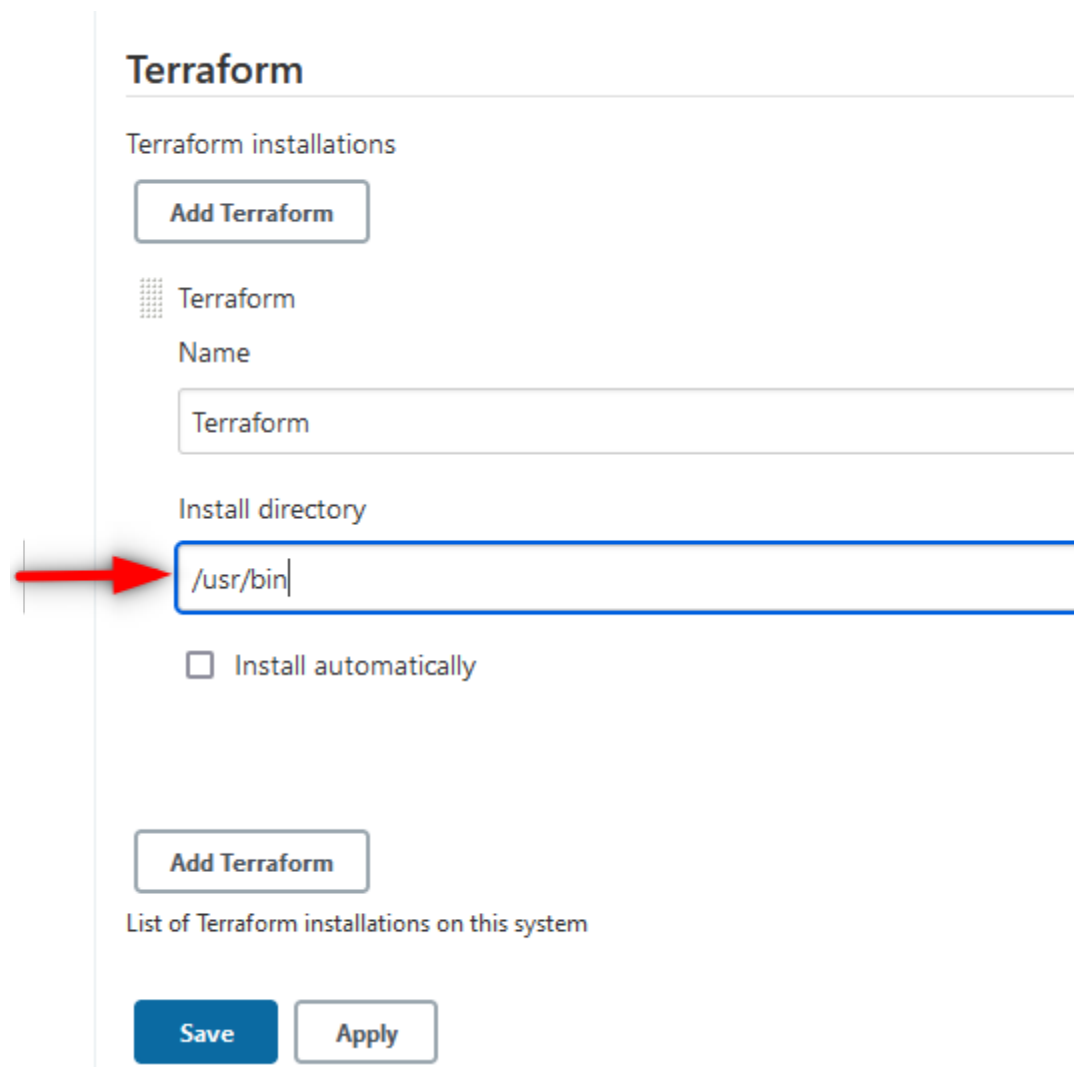
If you run ls you should see a Terraform zip file. Run  
\$ unzip <terraform zip file name> .

The unzipped file is an executable and we will need to move it to /usr/bin.  
\$ sudo mv terraform /usr/bin

10. To verify run which terraform

```
bitnami@ip-172-31-16-164:~$ which terraform
/usr/bin/terraform
```


11. Now switch back to Jenkins. For Install directory enter /usr/bin. Click Save.



**Terraform**


Terraform installations

[Add Terraform](#)

 Terraform

Name

Install directory



☐ Install automatically

[Add Terraform](#)

List of Terraform installations on this system

[Save](#) [Apply](#)

## Manage AWS Credentials on Jenkins

1. Click Manage Jenkins. Click Manage Credentials in the Security section.

### Security



#### Configure Global Security

Secure Jenkins; define who is allowed to access/use the system.



#### Manage Credentials

Configure credentials



#### Configure Credential Providers

Configure the credential providers and types

2. Click Jenkins.



## Credentials

T	P	Store ↓	Domain
---	---	---------	--------

Icon: S M L

### Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins	(global)

3. Click Global credentials (unrestricted).



### System

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

4. For Kind select Secret text. For ID type "AWS\_ACCESS\_KEY\_ID". For Secret paste your Access Key for your user. Then click OK.

Kind

Secret text

Scope

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

Secret

.....

ID

AWS\_ACCESS\_KEY\_ID

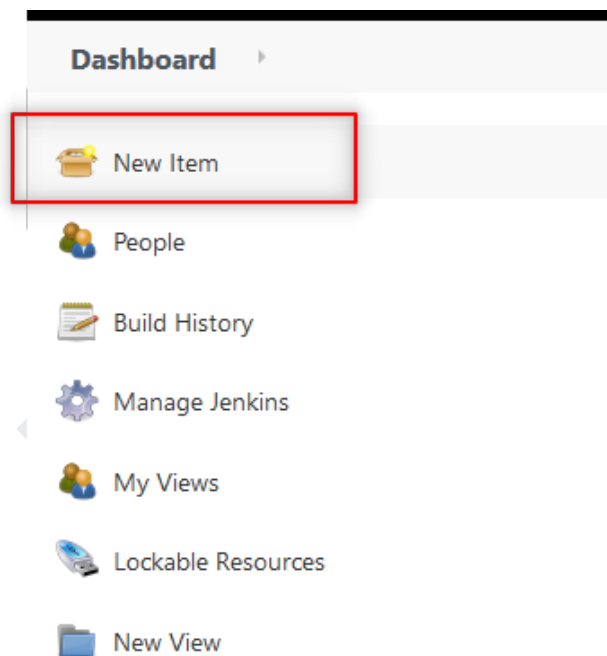
Description

OK

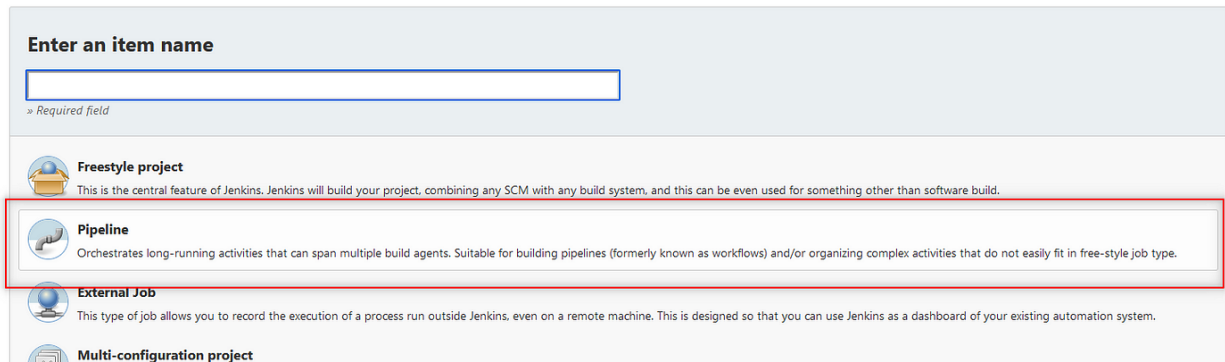
5. Repeat step 4 for your “AWS\_SECRET\_ACCESS\_KEY”

## Configure Jenkins Pipeline

1. Click New Item.



## 2. Select Pipeline.



The screenshot shows the 'Enter an item name' dialog in Jenkins. At the top is a text input field for the item name, with a small '» Required field' note below it. Below the input field are four radio button options, each with an icon and a description:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**: Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type. (This option is highlighted with a red border.)
- External Job**: This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.
- Multi-configuration project**

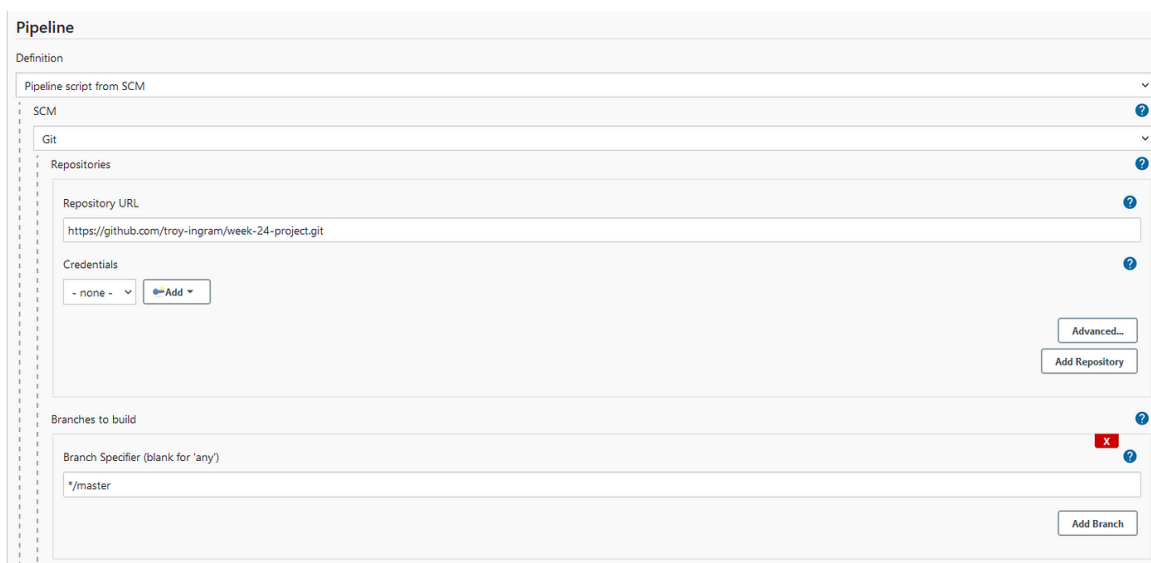
## 3. From the Definition drop down select Pipeline script from SCM.



The screenshot shows the 'Pipeline' configuration page. The 'Definition' section has a dropdown menu open, showing three options: 'Pipeline script', 'Pipeline script', and 'Pipeline script from SCM'. The 'Pipeline script from SCM' option is highlighted with a dark background. Below the dropdown is a large empty text area for the pipeline definition.

## 4. Enter the following:

- SCM: Git
- Repository URL: Your GitHub Repo with your Jenkinsfile
- Branch: Your primary branch

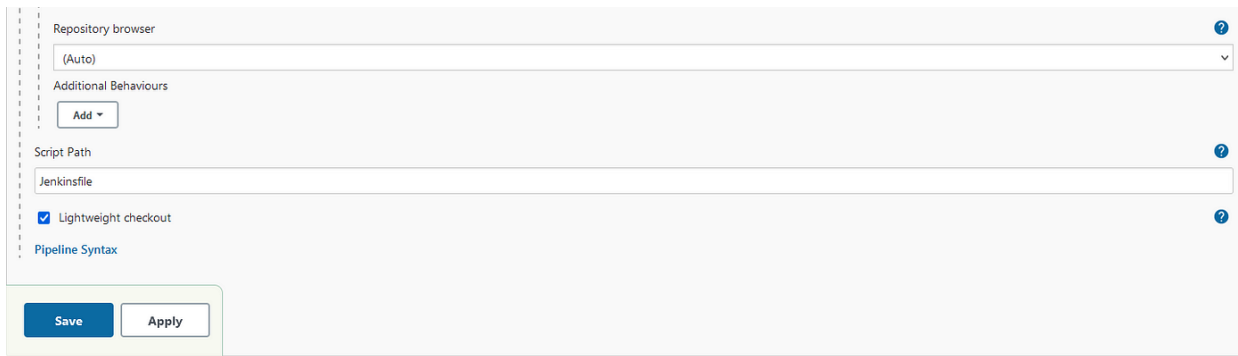


The screenshot shows the 'Pipeline' configuration page with the following fields and values:

- Definition**: Pipeline script from SCM
- SCM**: Git
- Repositories**:
  - Repository URL**: `https://github.com/troy-ingram/week-24-project.git`
  - Credentials**: - none - (Add button)
- Branches to build**:
  - Branch Specifier (blank for 'any')**: `*/master`

Buttons for 'Advanced...', 'Add Repository', and 'Add Branch' are visible on the right side of the configuration area.

- Repository browser: Auto
- Script Path: "Jenkinsfile"



Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

☒ Lightweight checkout

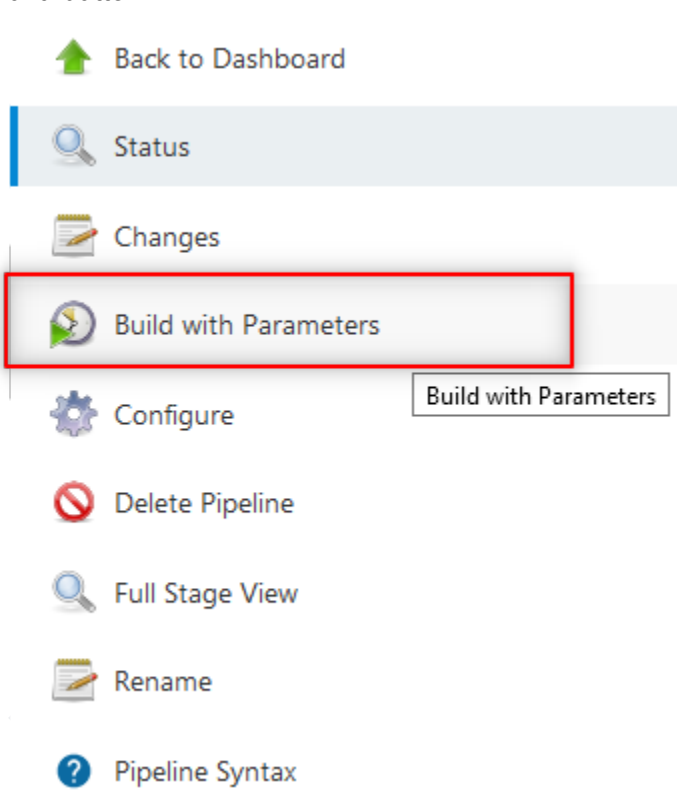
Pipeline Syntax

Save Apply

5. Click Save.

## Run Jenkins Pipeline

1. Select Build with Parameters from the left navigation. For the first time, you will see just the Build button.





2. For the environment parameter type the name you want to use for your Workspace. The default is "terraform". For now leave autoApprove and destroy unchecked. Click Build.

## Pipeline terraform-pipeline

This build requires parameters:

environment

Workspace/environment file to use for deployment

☐ autoApprove

Automatically run apply after generating plan?

☐ destroy

Destroy Terraform build?

Build

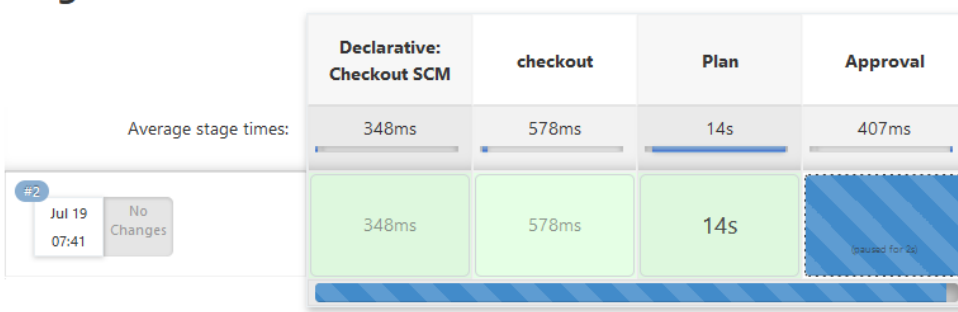
3. Now you should see the steps of the pipeline begin and the time it takes to complete each stage. The pipeline will pause on the Approval step because we didn't select the autoApprove parameter.

## Pipeline terraform-pipeline

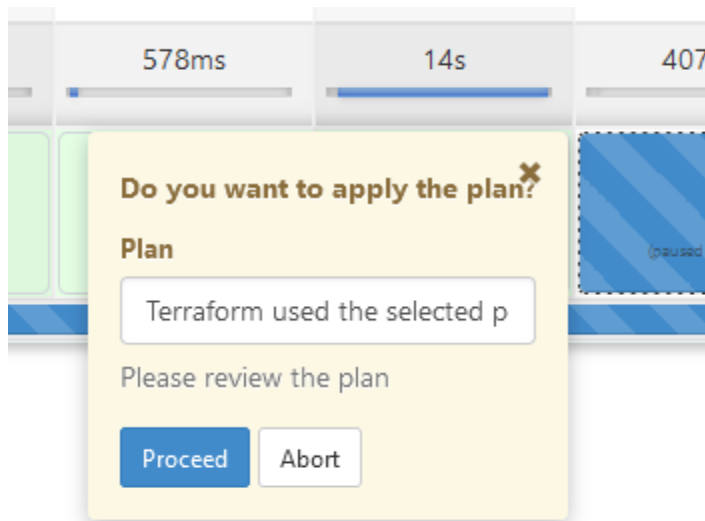


Recent Changes

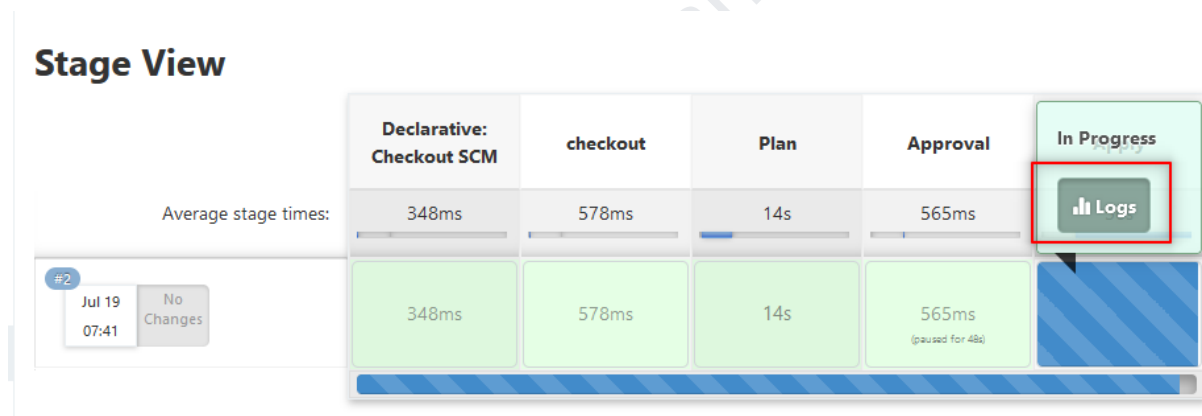
### Stage View



4. Click the Approval step and you'll see a pop up asking if you'd like to proceed. Review the plan and then click Proceed.



5. While the final step is applying our infrastructure you can hover over and select the Logs button.



This will display in real time your infrastructure being created. Below is a snippet from the Apply step logs.

```
+ terraform apply -input=false tfplan
[0m[1module.networking.random_integer.random: Creating...[0m[0m
[0m[1module.networking.random_integer.random: Creation complete after 0s [id=219][0m
[0m[1module.networking.aws_vpc.vpc: Creating...[0m[0m
[0m[1module.networking.aws_vpc.vpc: Still creating... [10s elapsed][0m[0m
[0m[1module.networking.aws_vpc.vpc: Creation complete after 12s [id=vpc-0237dfd3f3b22cdb5][0m
[0m[1module.networking.aws_internet_gateway.internet_gateway: Creating...[0m[0m
[0m[1module.networking.aws_subnet.public_subnet[1]: Creating...[0m[0m
[0m[1module.networking.aws_subnet.public_subnet[0]: Creating...[0m[0m
[0m[1module.networking.aws_route_table.public_rt: Creating...[0m[0m
[0m[1module.networking.aws_security_group.web_sg: Creating...[0m[0m
[0m[1module.networking.aws_route_table.public_rt: Creation complete after 0s [id=rtb-0a1e55fab6d832cd1][0m
[0m[1module.networking.aws_internet_gateway.internet_gateway: Creation complete after 0s [id=igw-0aafe0d020be040d1][0m
[0m[1module.networking.aws_route.default_public_route: Creating...[0m[0m
[0m[1module.networking.aws_route.default_public_route: Creation complete after 1s [id=r-rtb-0a1e55fab6d832cd11080289494][0m
[0m[1module.networking.aws_security_group.web_sg: Creation complete after 1s [id=sg-07f3f7d5fa613edf4][0m
[0m[1module.compute.aws_launch_template.web: Creating...[0m[0m
[0m[1module.compute.aws_launch_template.web: Creation complete after 1s [id=lt-04750e915fbe29e82][0m
[0m[1module.networking.aws_subnet.public_subnet[1]: Still creating... [10s elapsed][0m[0m
```

## 6. Our Jenkins Pipeline has completed!

### Stage View

		Declarative: Checkout SCM	checkout	Plan	Approval	Apply	Destroy
Average stage times: (Average full run time: ~2min 22s)		348ms	578ms	14s	565ms	1min 15s	0ms
#2	Jul 19 07:41 No Changes	348ms	578ms	14s	565ms (paused for 48s)	1min 15s	

7. From the Jenkins console logs, you can get the ALB DNS Endpoint URL in form of Terraform Outputs. Copy that link and open it in the browser to get an nginx webserver default page to validate the overall Infra deployment pipeline.

## Destroying Our Infrastructure

1. Build another Pipeline, but this time select the destroy parameter.

# Pipeline terraform-pipeline

This build requires parameters:

environment

Workspace/environment file to use for deployment

☐ autoApprove

Automatically run apply after generating plan?

☒ destroy

Destroy Terraform build?

**Build**

That's it. If you look at the destroy pipeline console logs, you can see the AWS Infrastructure is been deleted by Terraform.

## Trigger the Jenkins pipeline via GitHub commits (push events)

1. Open the Jenkins pipeline and go to the Configure option. Under General configurations, you will find GitHub project config property. Check mark this property and provide the Project URL as per your GitHub repo URL.

☒ GitHub project

Project url ?

Advanced ▾

2. You can make your localhost reachable on internet by port forwarding 8080 Jenkins port to Serveo.net portal.

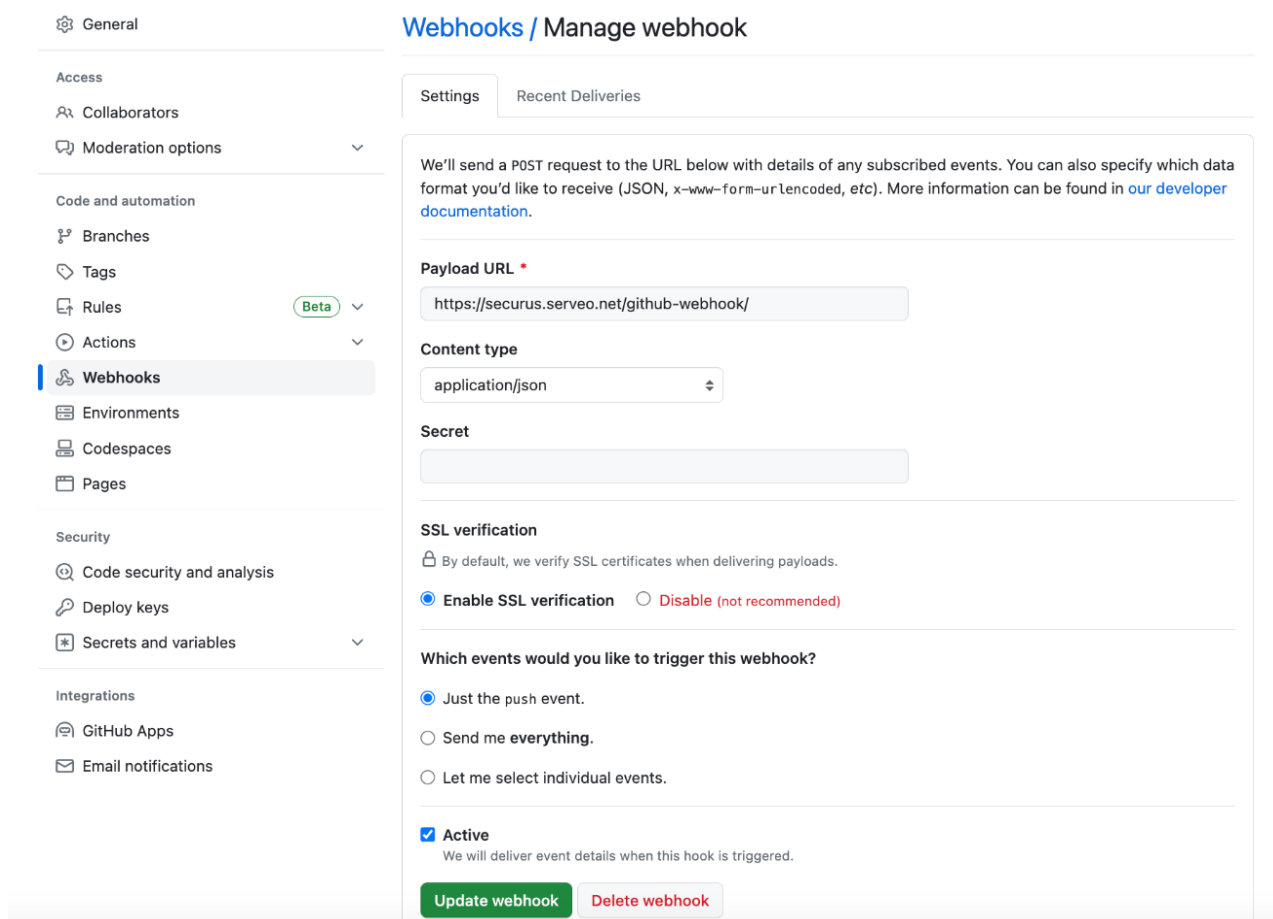
```
avani@Avanis-MacBook-Pro infra % ssh -R 80:localhost:8080 serveo.net
Forwarding HTTP traffic from https://plico.serveo.net
HTTP request from 41.223.98.122 to https://plico.serveo.net/
```

3. As per the above screenshot now, you can use <https://plico.serveo.net/> URL as per the logs in the GitHub WebHook configuration.

4. Go to the GitHub repo from where you want to trigger the commit/push event based Jenkins pipelines and go to Settings → Webhooks → Add webhook. You can provide the config properties as below to setup the WebHook triggers by referring following Payload URL format:

<Serveo-URL>/github-webhook/

For ex: <https://securus.serveo.net/github-webhook/>

The image shows the GitHub 'Webhooks / Manage webhook' configuration page. On the left is a sidebar with navigation links: General, Access (Collaborators, Moderation options), Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), and Integrations (GitHub Apps, Email notifications). The 'Webhooks' link is highlighted. The main content area has two tabs: 'Settings' and 'Recent Deliveries'. The 'Settings' tab is active and contains the following fields: 'Payload URL' with the value 'https://securus.serveo.net/github-webhook/', 'Content type' set to 'application/json', and a 'Secret' field. Below these is the 'SSL verification' section with a note 'By default, we verify SSL certificates when delivering payloads.' and two radio buttons: 'Enable SSL verification' (selected) and 'Disable (not recommended)'. The 'Which events would you like to trigger this webhook?' section has three radio buttons: 'Just the push event.' (selected), 'Send me everything.', and 'Let me select individual events.'. At the bottom, there is an 'Active' checkbox (checked) with the text 'We will deliver event details when this hook is triggered.' and two buttons: 'Update webhook' and 'Delete webhook'.

5. Now whenever you commit a new change from the main branch of the repo, you can see a new pipeline job is getting triggered unless until the serveo URL is up and running from your local environment.