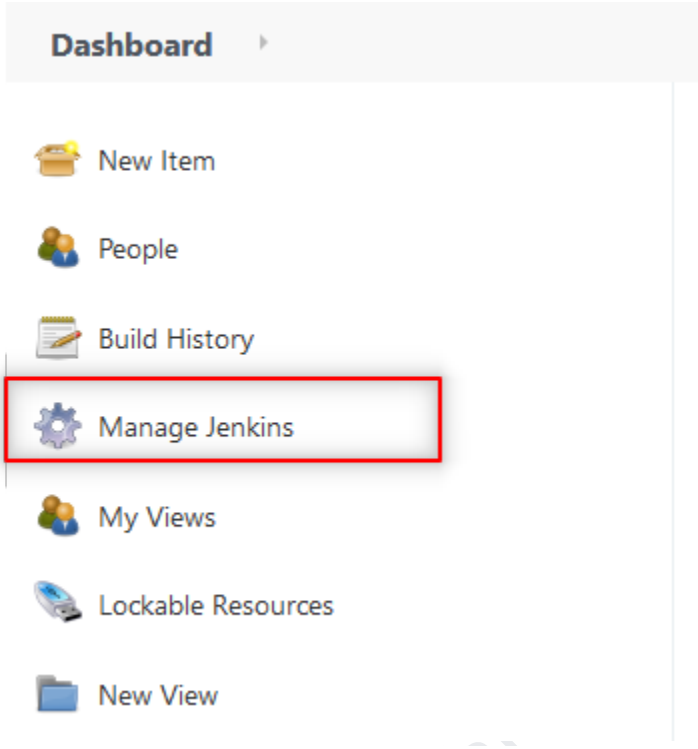
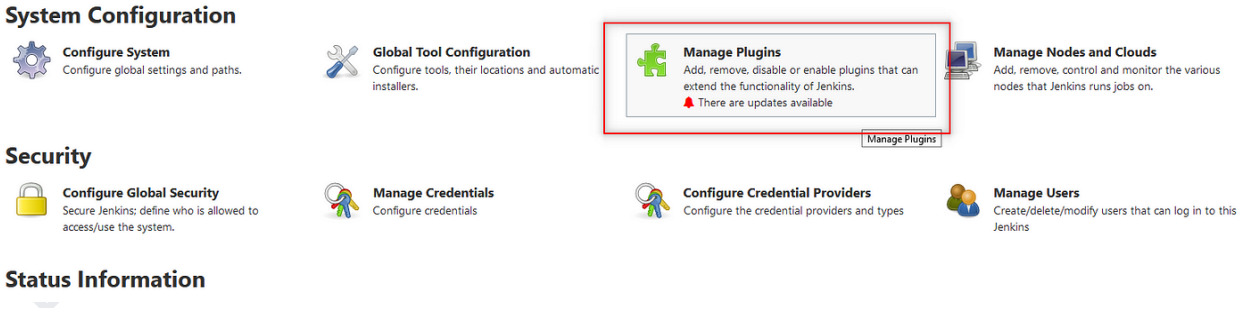
Deploy AWS Cloud Infra using Terraform IaC Automated using Jenkins Pipeline.

Configure Terraform on Jenkins

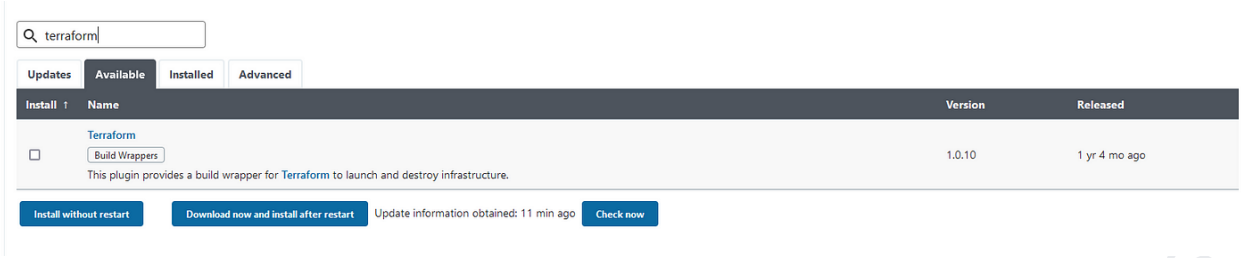
1. Click Manage Jenkins from left hand navigation.



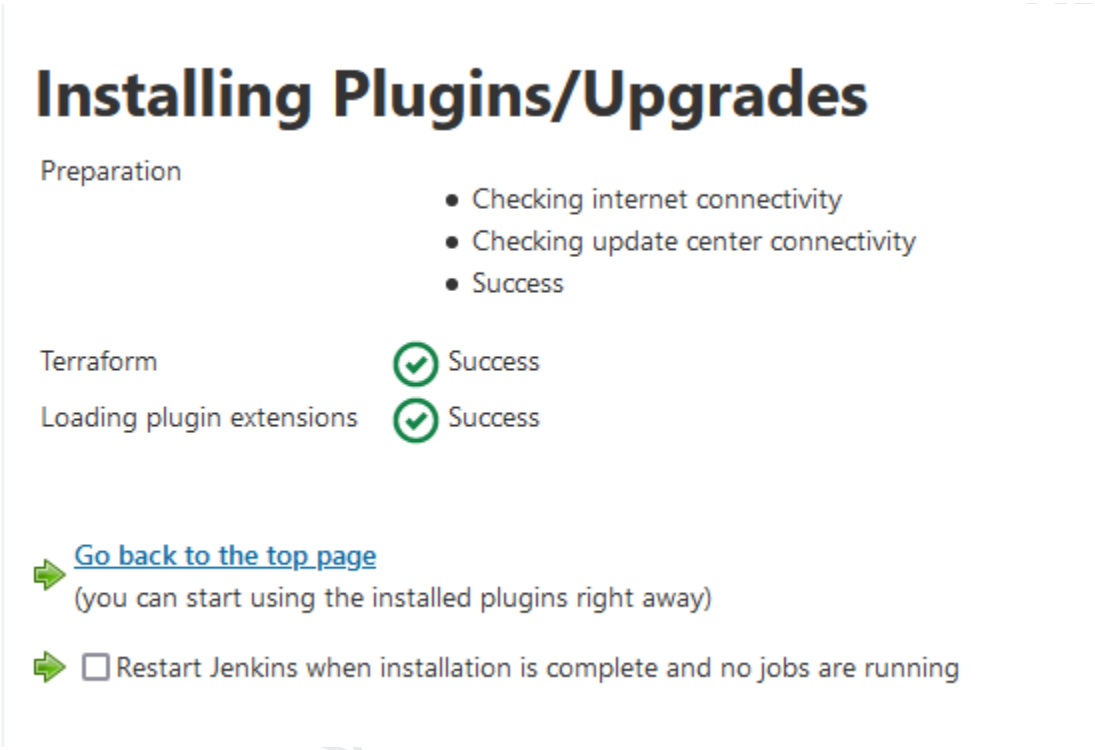
2. Select Manage Plugins from System Configuration section.



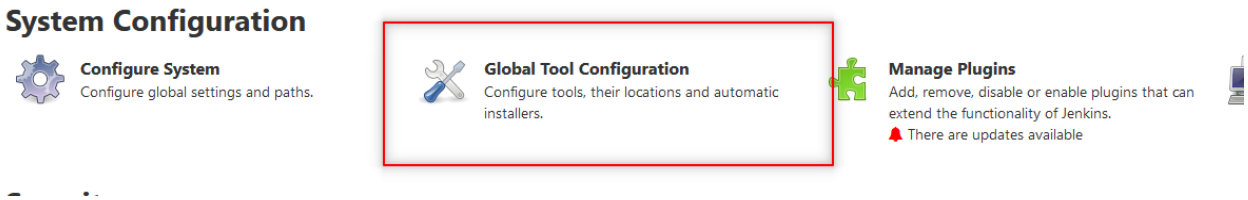
3. Click the Available tab and search Terraform.



4. Select Terraform and click Install without restart.

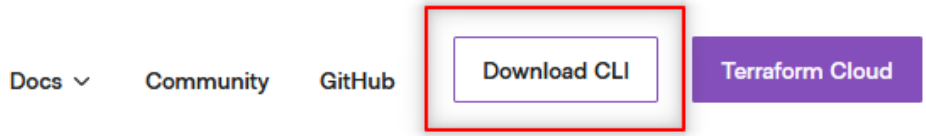


5. Click Manage Jenkins from left hand navigation. Click Global Tool Configuration from System configuration section.

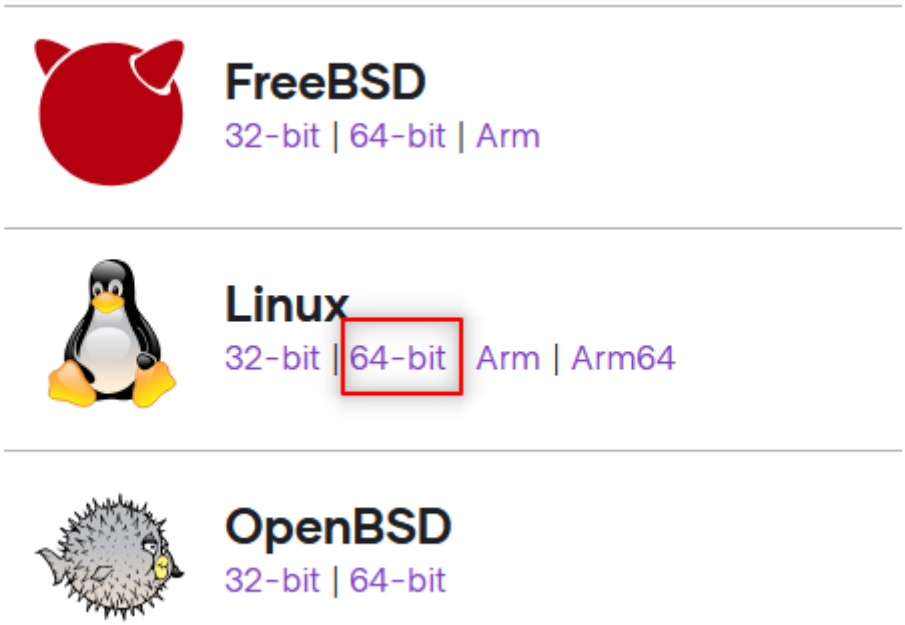


6. Scroll down to the Terraform section and click Add Terraform. Enter a Name of your choice. I'm going to use "Terraform" to make things simple. Ensure Install automatically is NOT selected. It will be selected by default.

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>

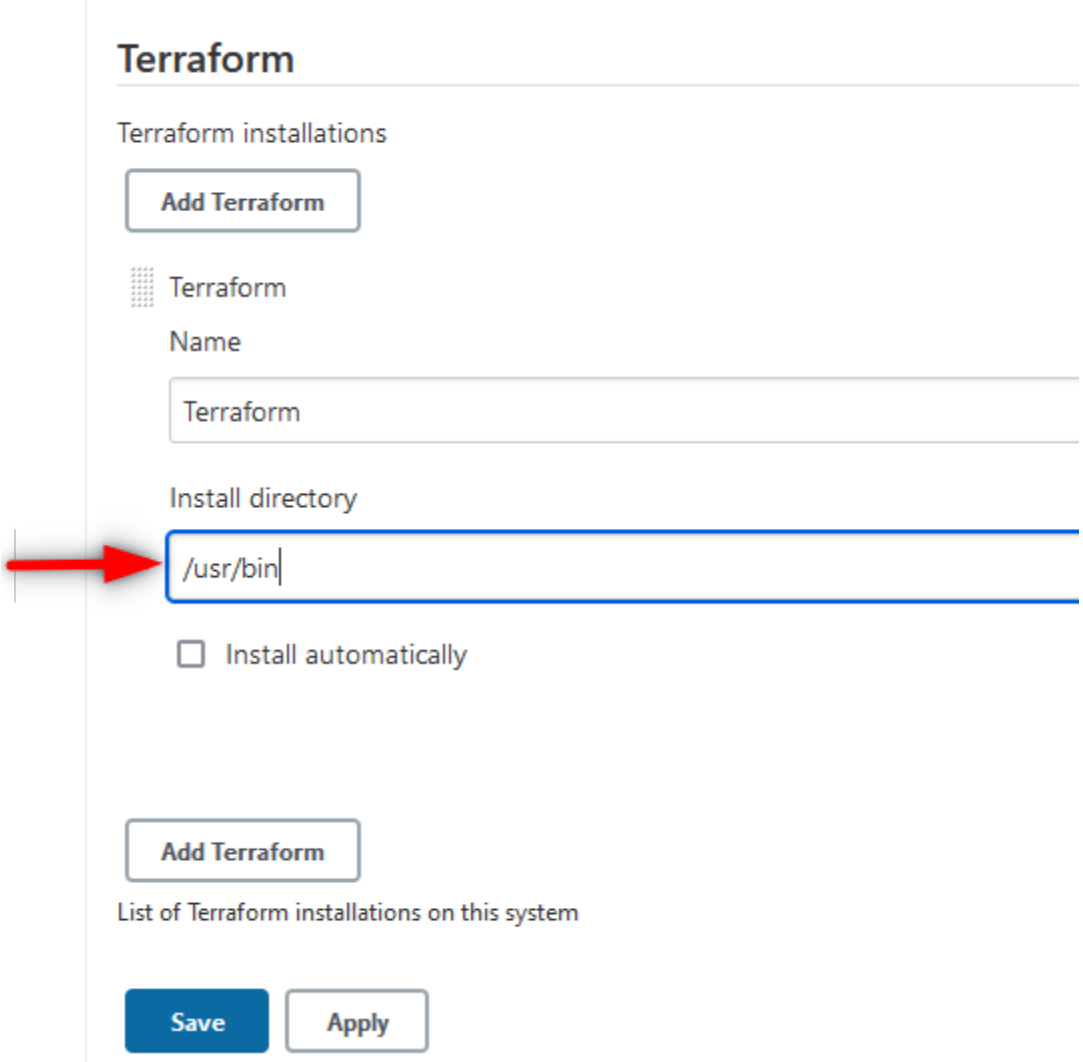
If you run ls you should see a Terraform zip file. Run
$ unzip <terraform zip file name> .

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

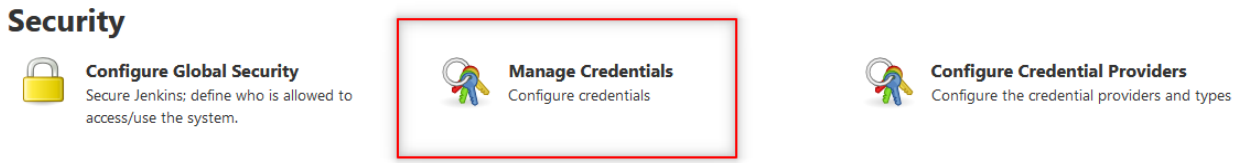
```
bitnami@ip-172-31-16-164:~$ which terraform
/usr/bin/terraform
```

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

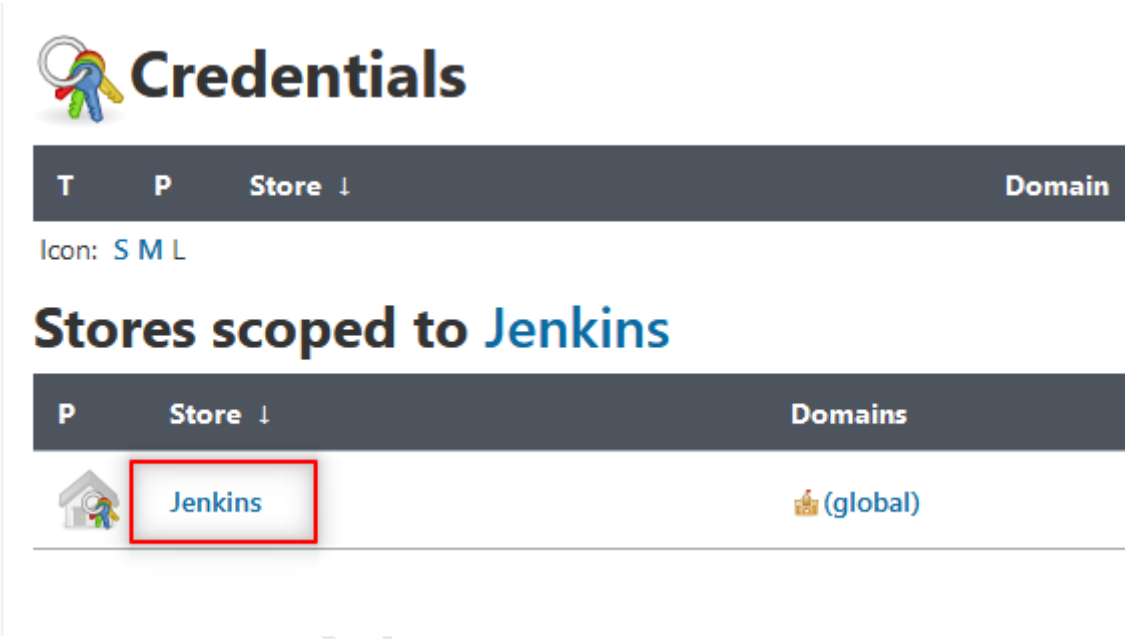


Manage AWS Credentials on Jenkins

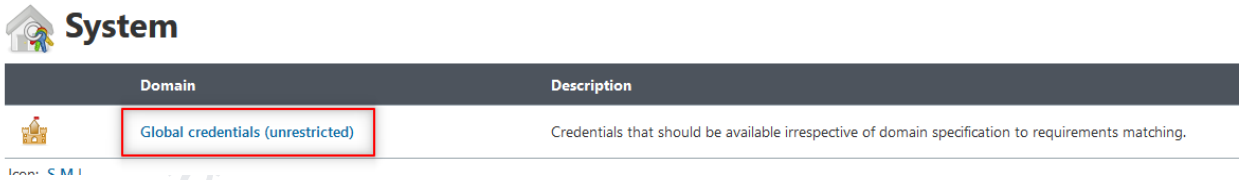
1. Click Manage Jenkins. Click Manage Credentials in the Security section.



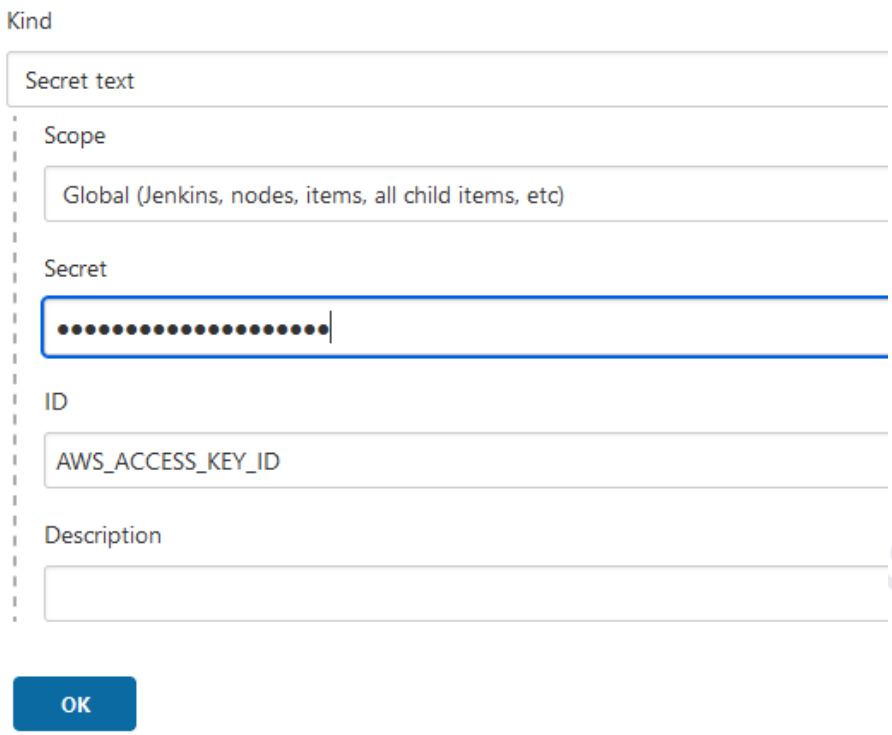
2. Click Jenkins.



3. Click Global credentials (unrestricted).



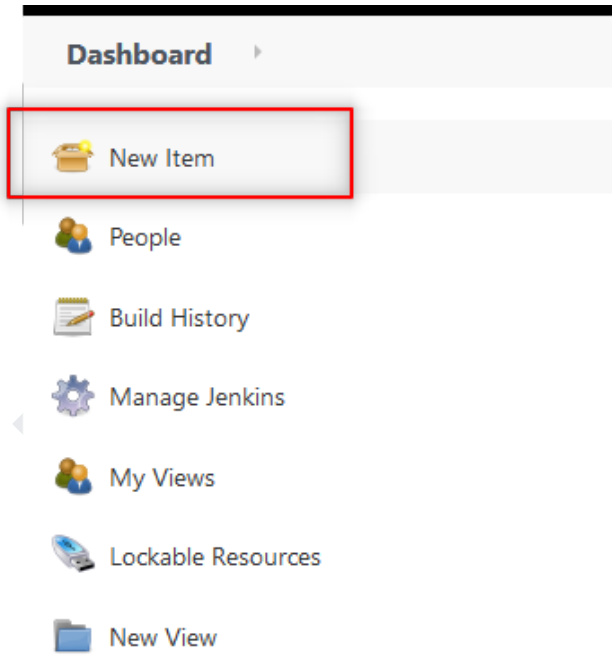
4. For Kind select Secret text. For ID type “AWS_ACCESS_KEY_ID”. For Secret paste your Access Key for your user. Then click OK.



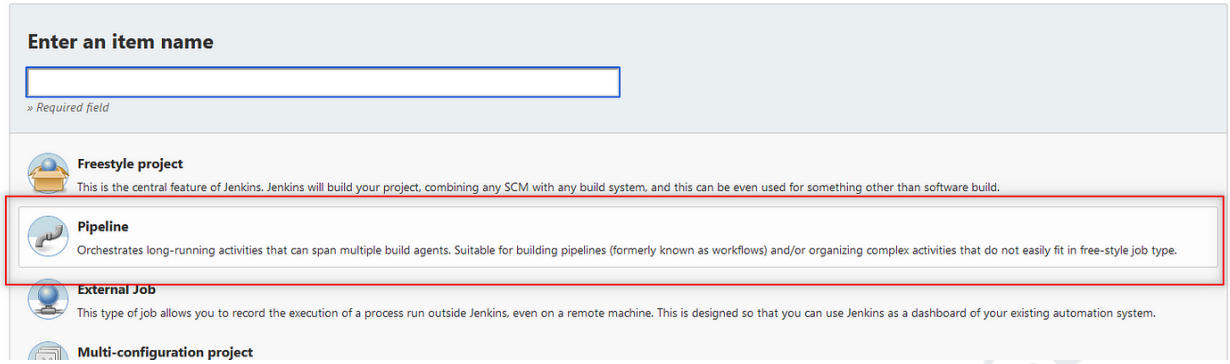
5. Repeat step 4 for your “AWS_SECRET_ACCESS_KEY”

Configure Jenkins Pipeline

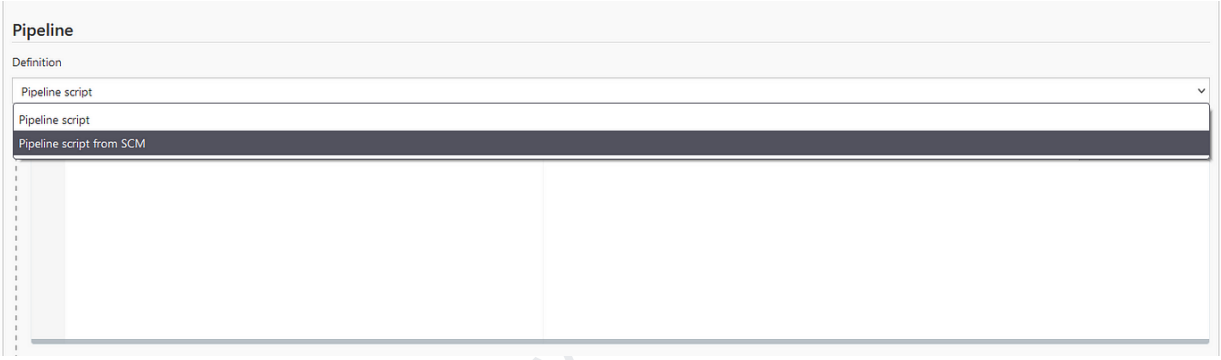
1. Click New Item.



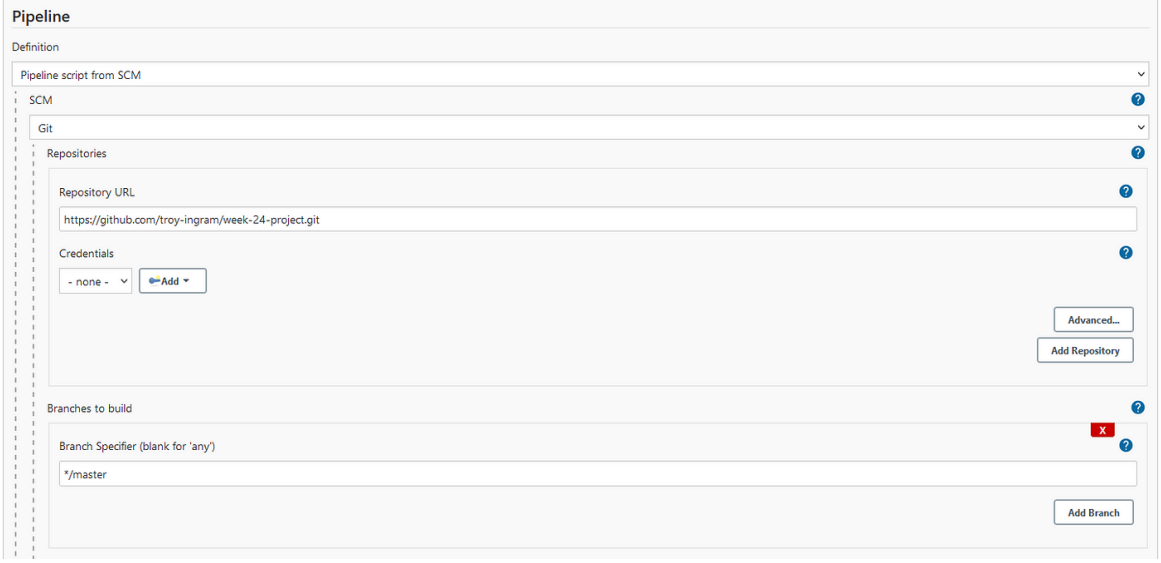
2. Select Pipeline.



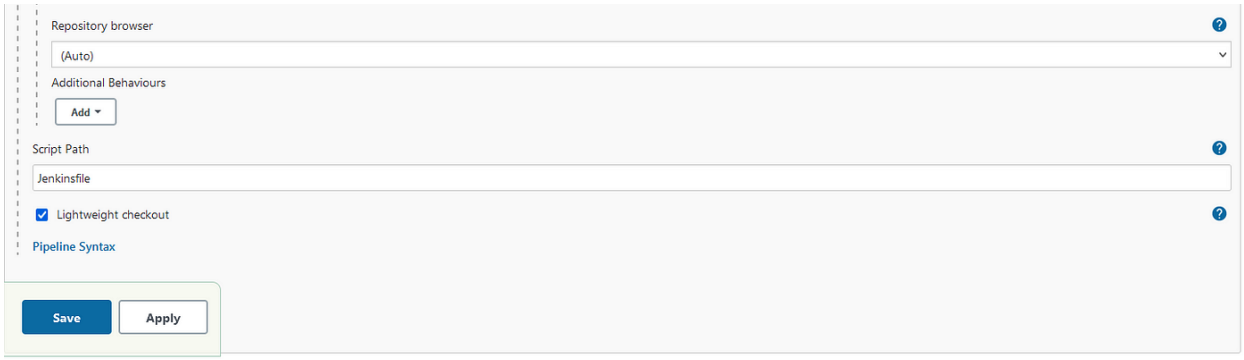
3. From the Definition drop down select Pipeline script from SCM.



4. Enter the following:
   - SCM: Git
   - Repository URL: Your GitHub Repo with your Jenkinsfile
   - Branch: Your primary branch



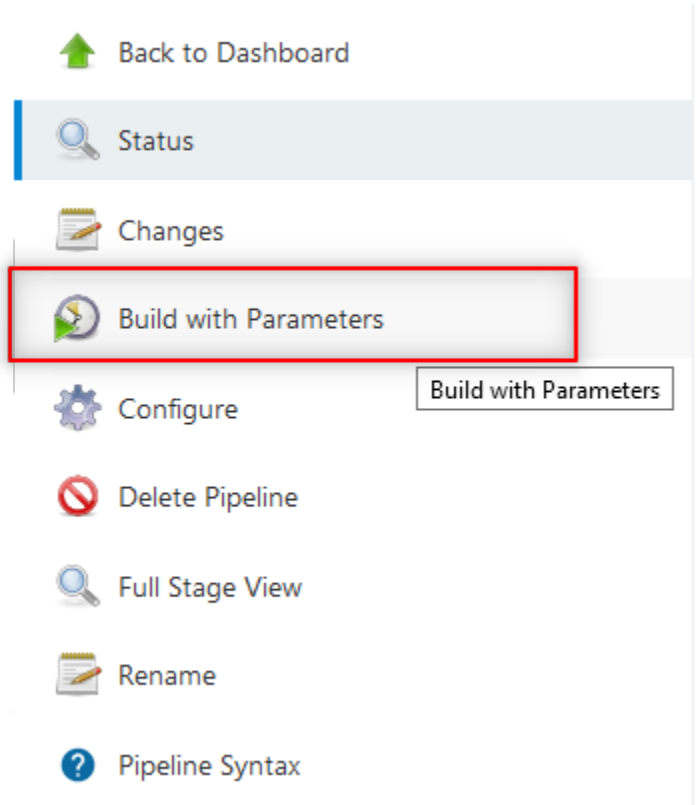
- Repository browser: Auto
- Script Path: “Jenkinsfile”



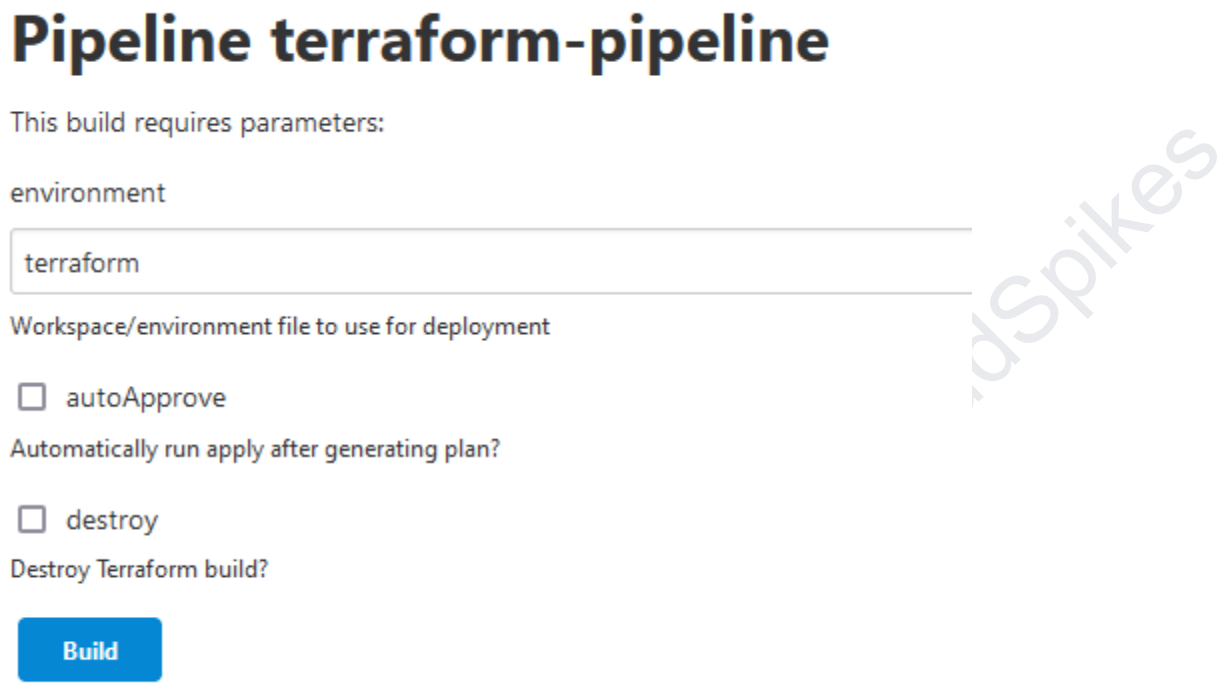
5. Click Save.

Run Jenkins Pipeline

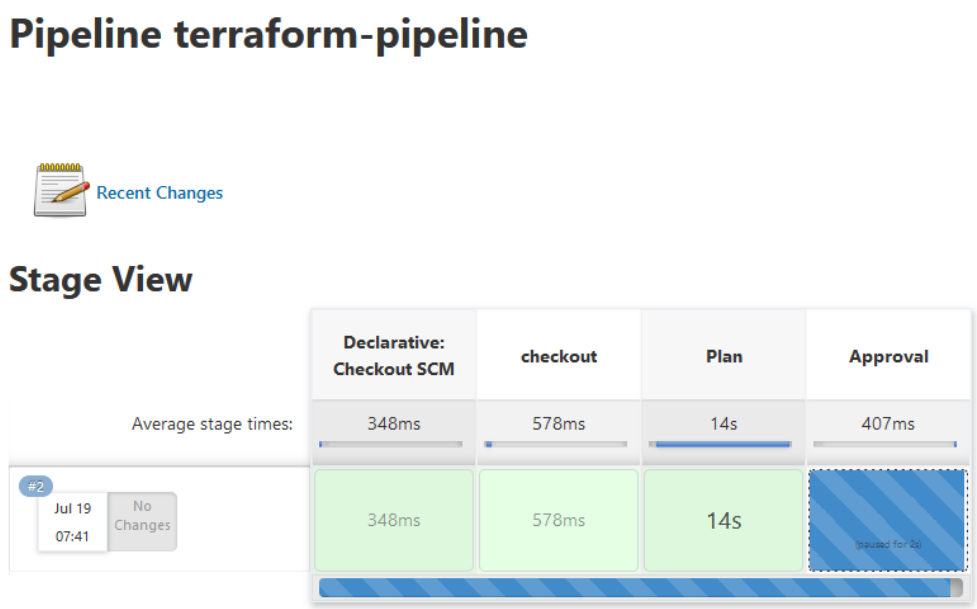
1. Select Build with Parameters from the left navigation. For the first time, you will see just the Build button.



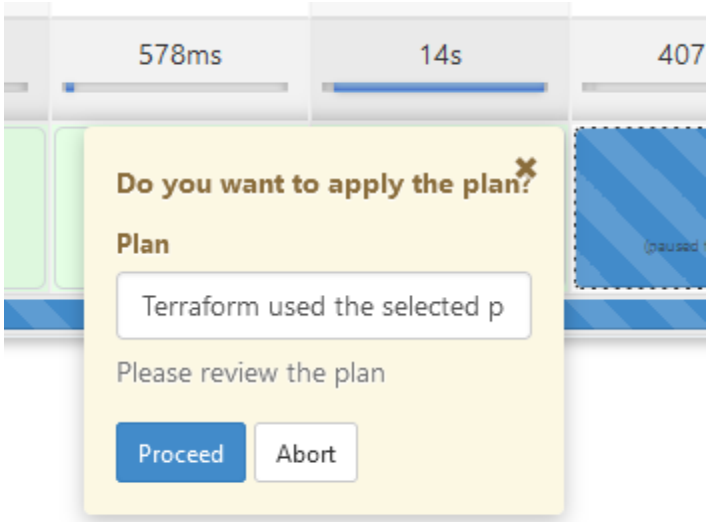
2. For the environment parameter type the name you want to use for your Workspace. The default is “terraform”. For now leave autoApprove and destroy unchecked. Click Build.



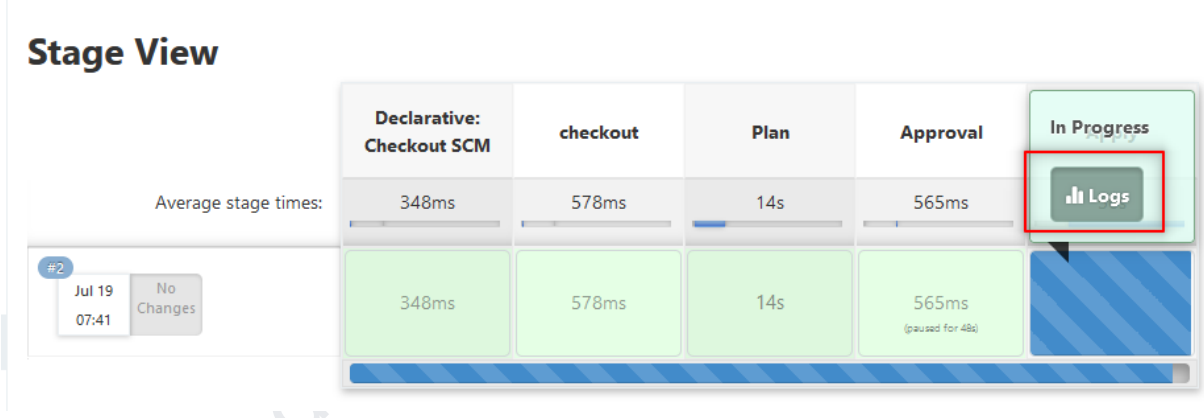
3. Now you should see the steps of the pipeline begin and the time it takes to complete each stage. The pipeline will pause on the Approval step because we didn’t select the autoApprove parameter.



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.



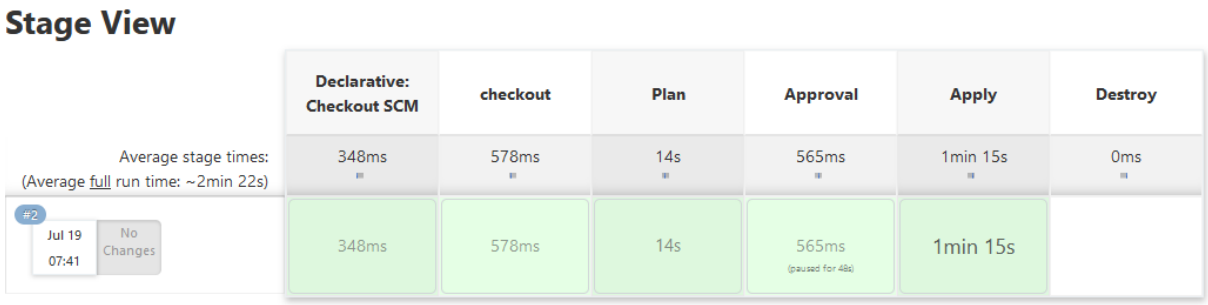
5. While the final step is applying our infrastructure you can hover over and select the Logs button.



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

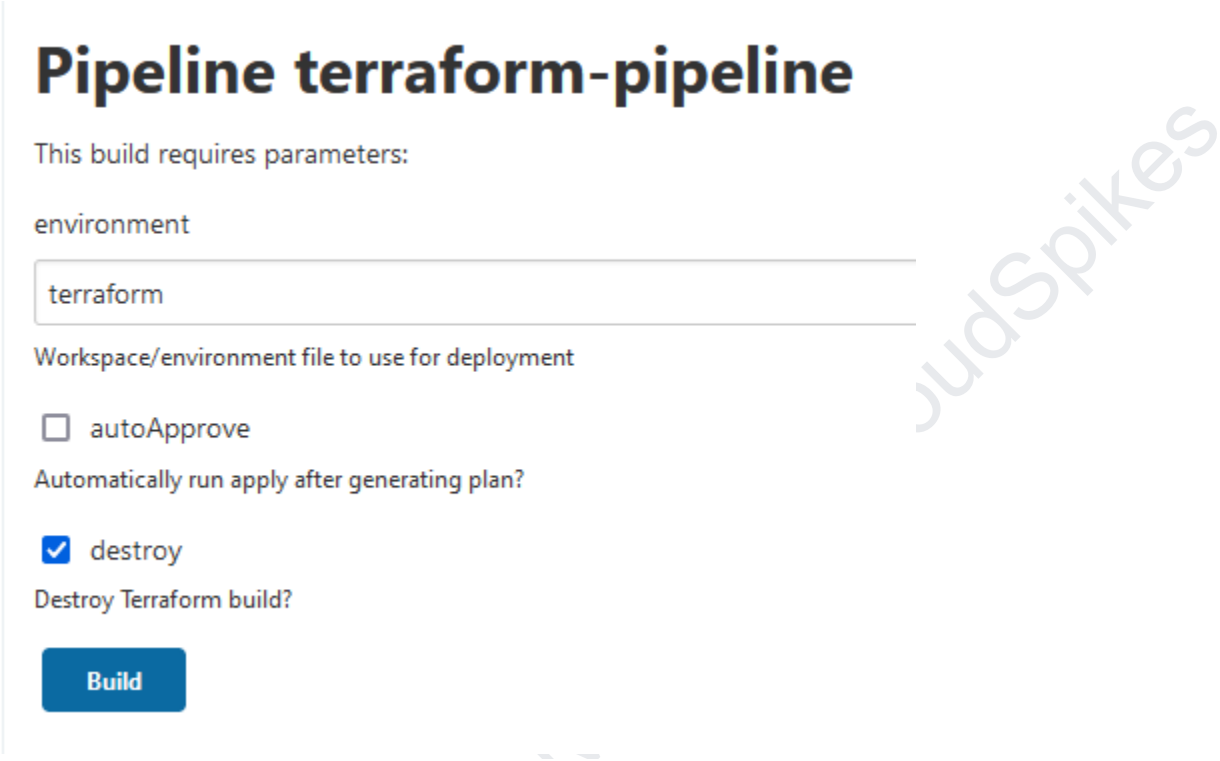
6. Our Jenkins Pipeline has completed!



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.



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.

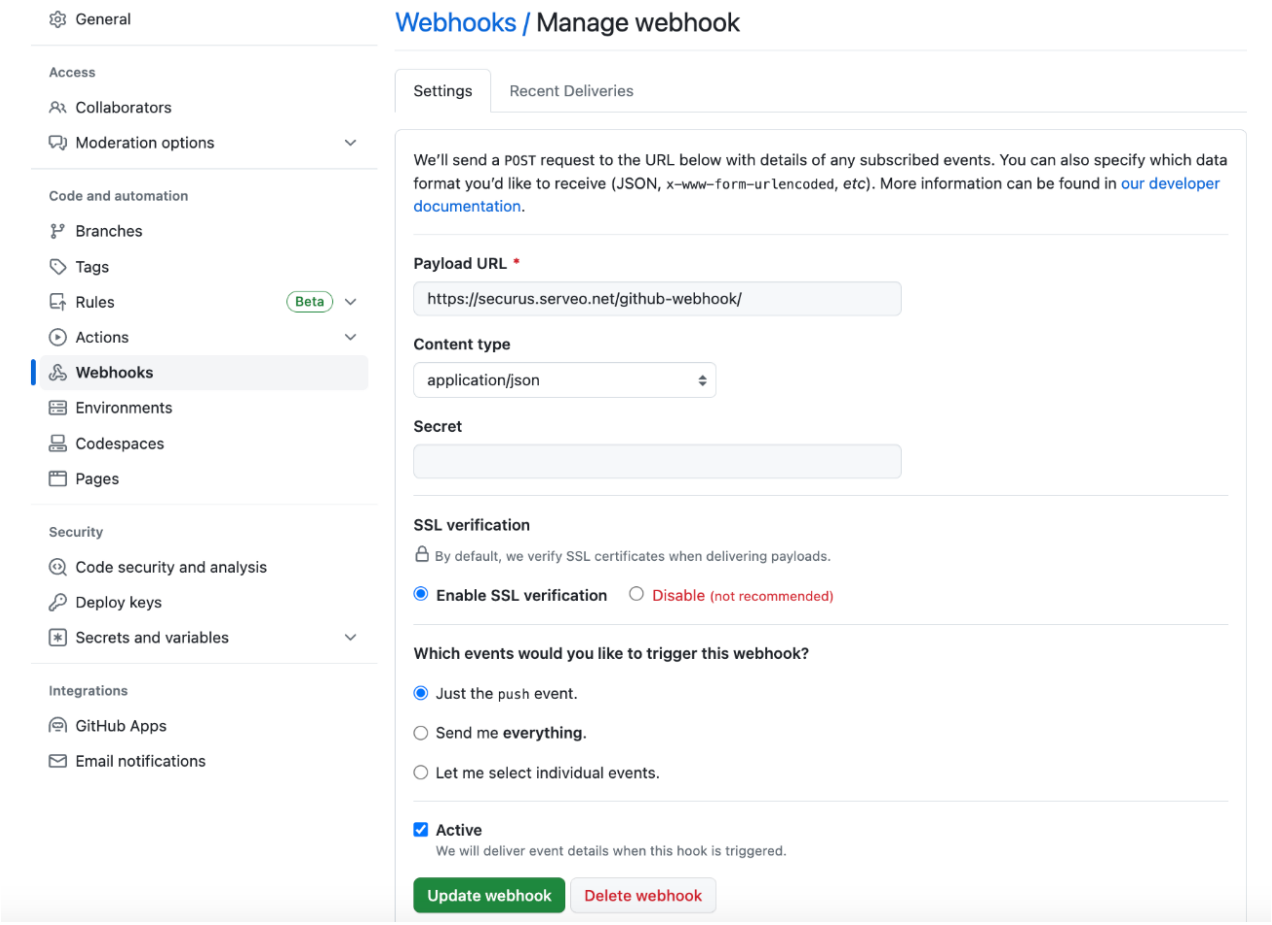
2. You can make your localhost reachable on internet by port forwarding 8080 Jenkins port to Serveo.net portal.

```
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/*



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.