Creating AWS Resources with GitHub Actions and Terraform

It’s not a terrible argument at all. Almost everyone in IT has heard of GitHub, and most have used it. It is extremely friendly to open source projects, and that friendliness continues with [GitHub Actions](https://github.com/features/actions) — they are free to open source repositories.

Which is not to say that it’s expensive for private repos. Pricing is based around how many minutes are consumed per month, with a generous amount of minutes provided for free to hook new users, then a simple per-minute charged based on the instance type. And, as with other cloud CI/CD providers, the self-hosted option (where you spin up your own builder host) is 100% free.

In this POC we will:

1. Create an IAM user in AWS with do-anything permissions

2. Bootstrap AWS with an S3 bucket (for terraform storage) and a DynamoDB table for terraform state locking

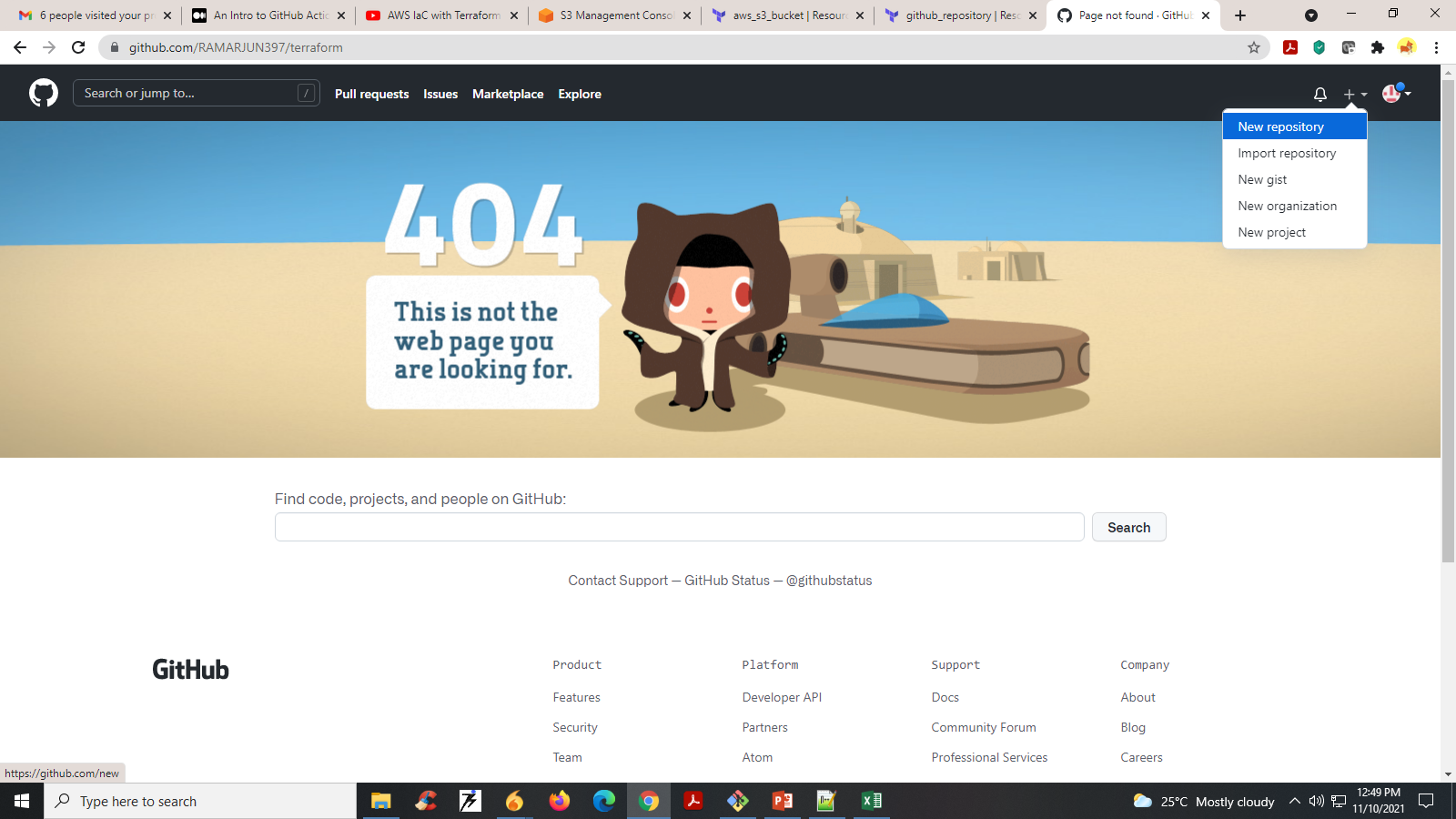
3. Set up a new GitHub repository

4. Store the IAM key and secret key as encrypted keys in GitHub for Actions to consume

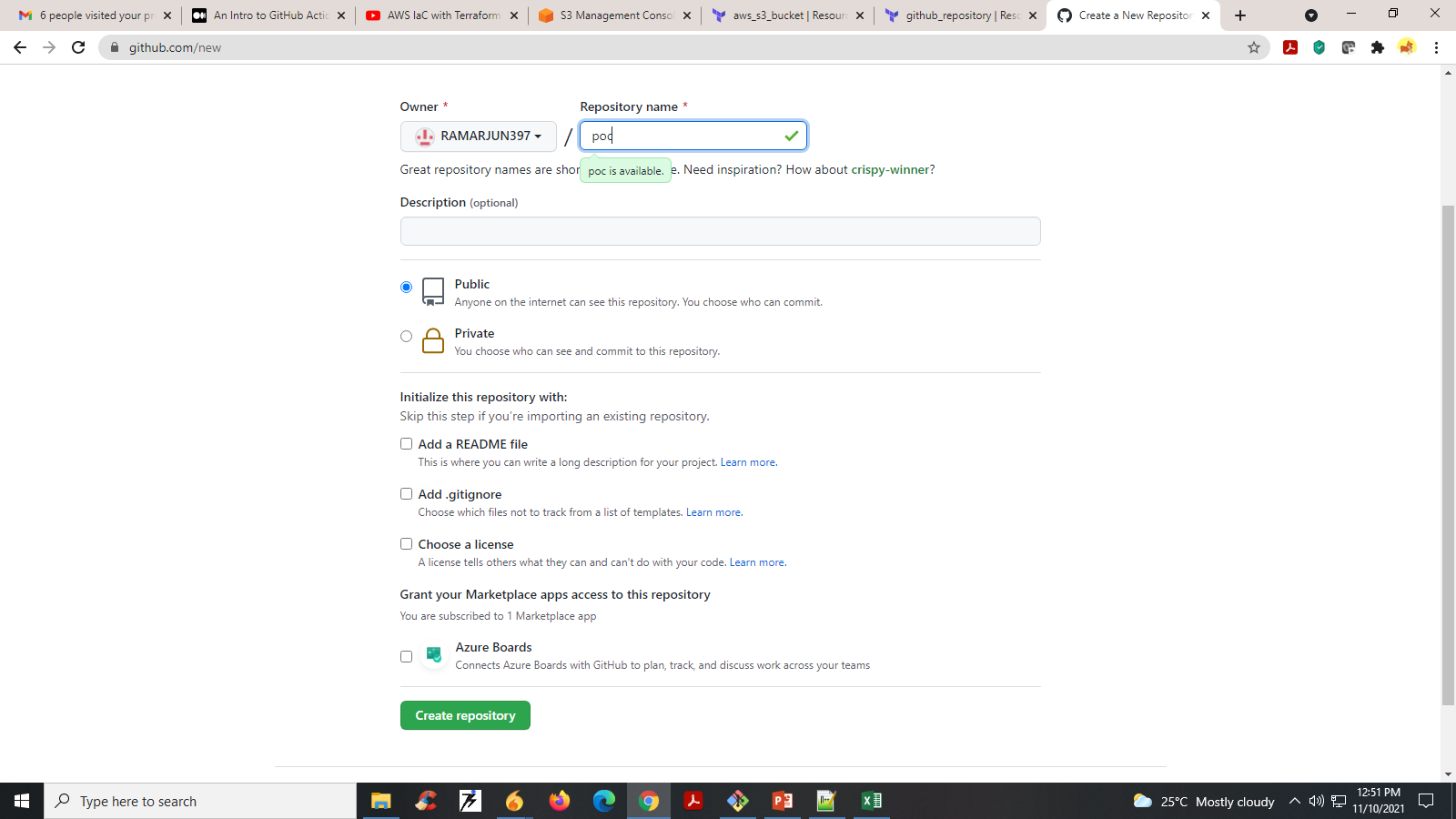
5. Create some GitHub actions that execute automatic terraform plan when code is committed to our repository

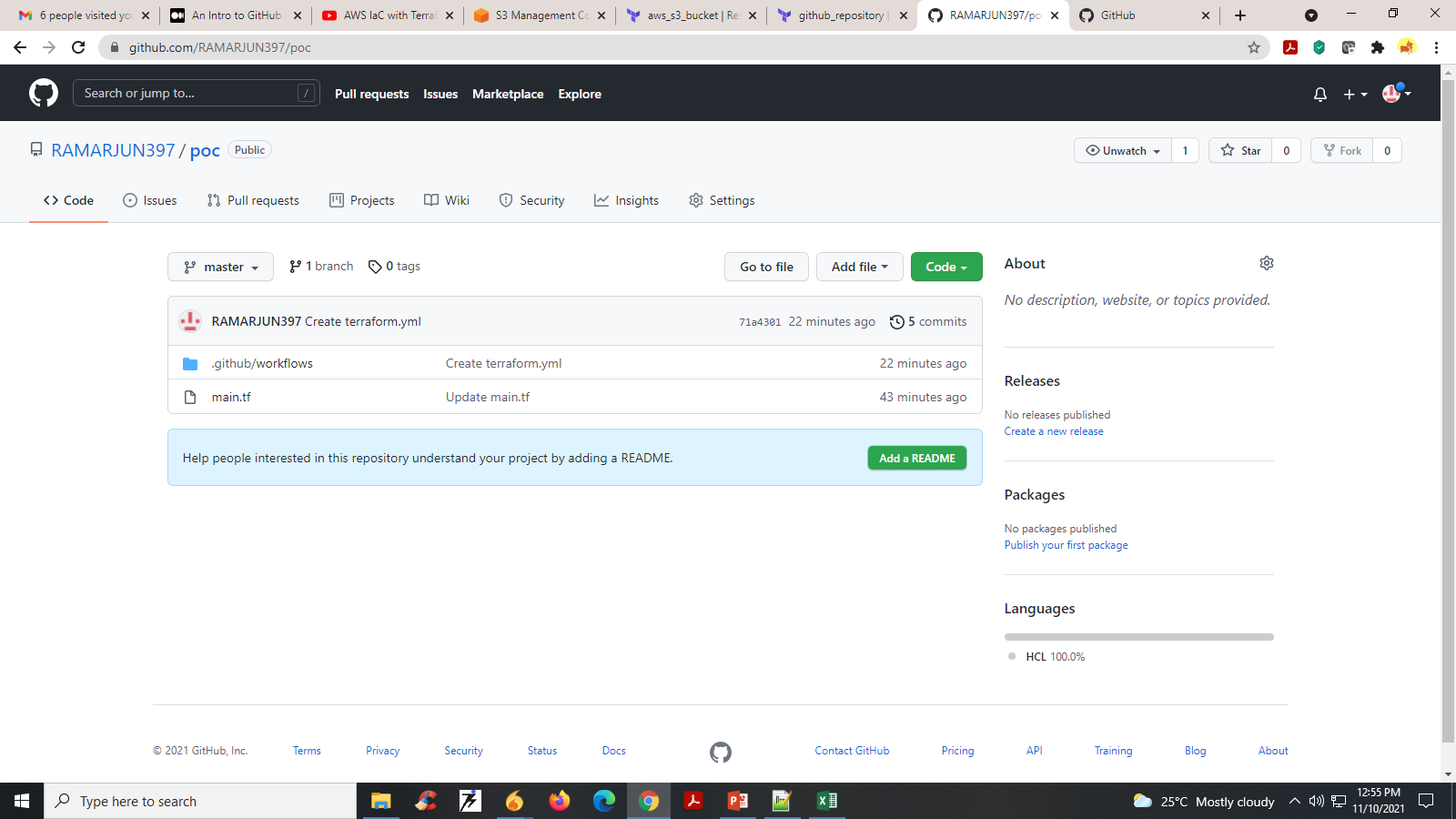
Let’s get started. You can do this.

1.First we can login github account after that create new git repository.



2. Here we can give repository name .



3. Next push our terraform code to GitHub repository using Git commands. After that the page look like. 

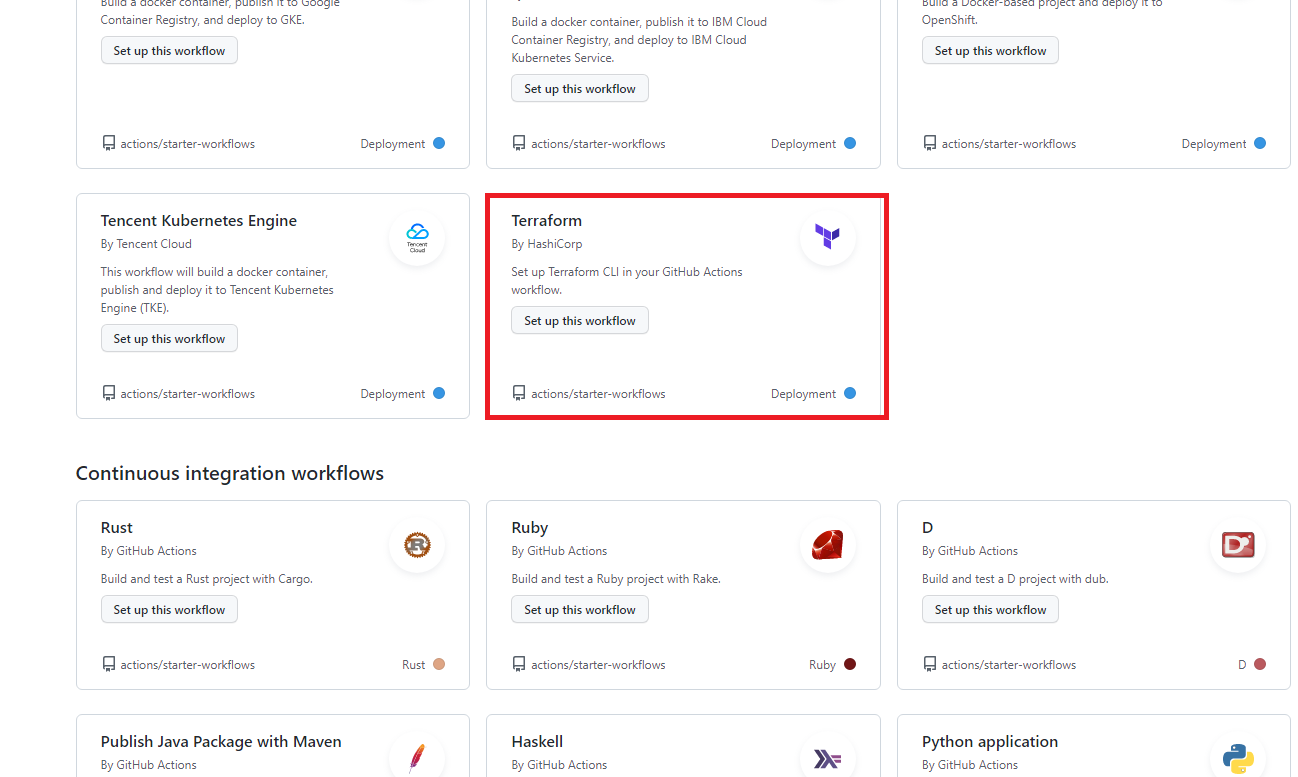
The main.tf contains the below code for creating s3 bucket in aws .

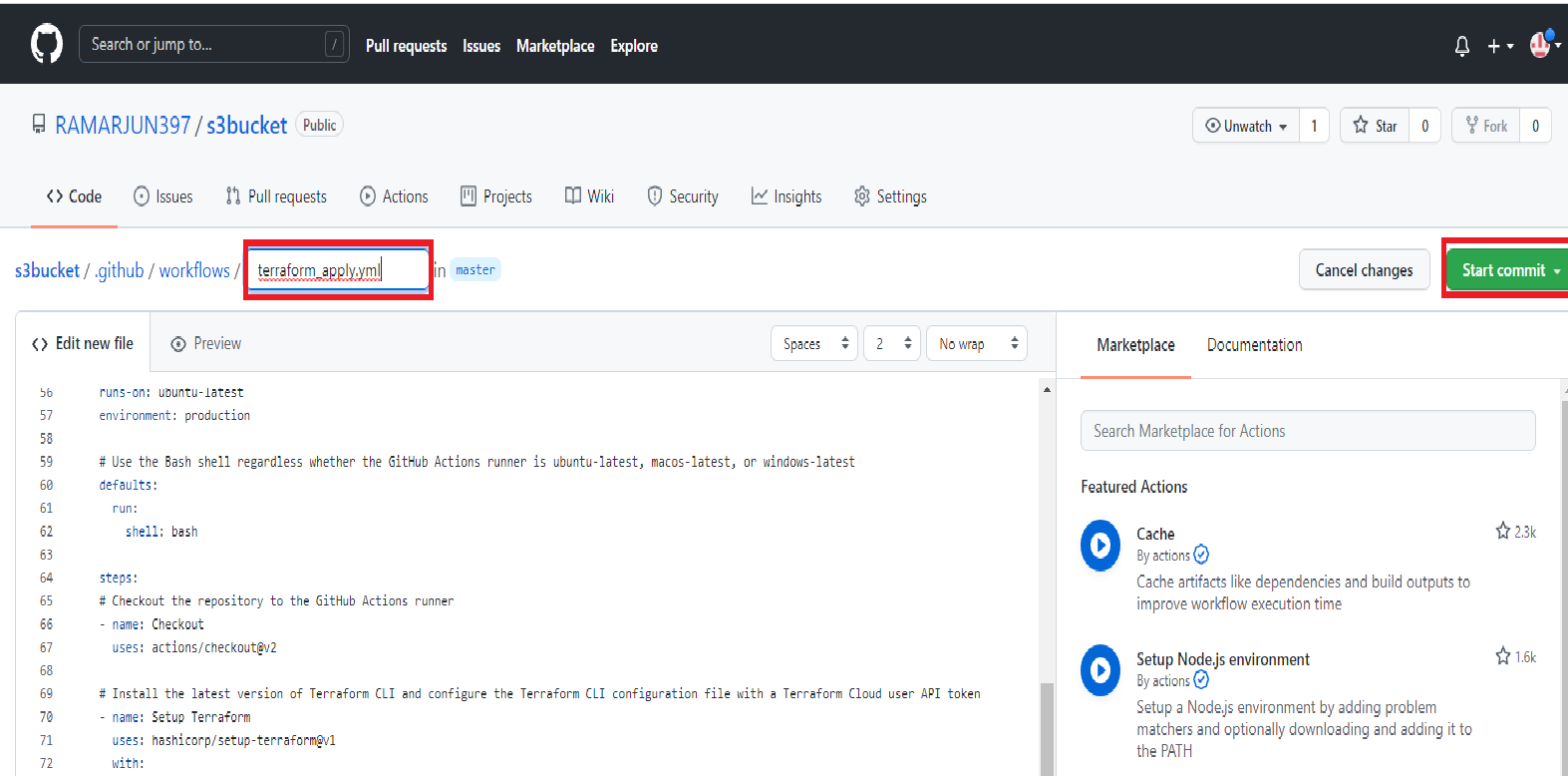
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | terraform { | |  | required\_version = ">=0.12.13" | |  | } | |  |  | |  | provider "aws" { | |  | region = "ap-southeast-1" | |  | access\_key = "<Access-Key" | |  | secret\_key = "<Secret-key>" | |  | } | |  |  | |  | #Aws s3bucket | |  | resource "aws\_s3\_bucket" "terraformsimplestorage3bucket" { | |  | bucket = "terraformsimplestorage3bucket" | |  | acl = "private" | |  |  | |  | tags = { | |  | Name = "terraformsimplestorage3bucket" | |  | Environment = "OPS" | |  | } | |  | } | |

# Terraform Actions In… Action

You’ll be dropped into your shiny new GitHub repo. Click on the “Actions” tab, and you should see the “Terraform Apply” workflow. That was built automatically because of the file we imported from my source repository in .github/workflows/TerraformApply.yml. It will run automatically each time any file in this repo is updated, including its own file, but will fail until we add some secrets.

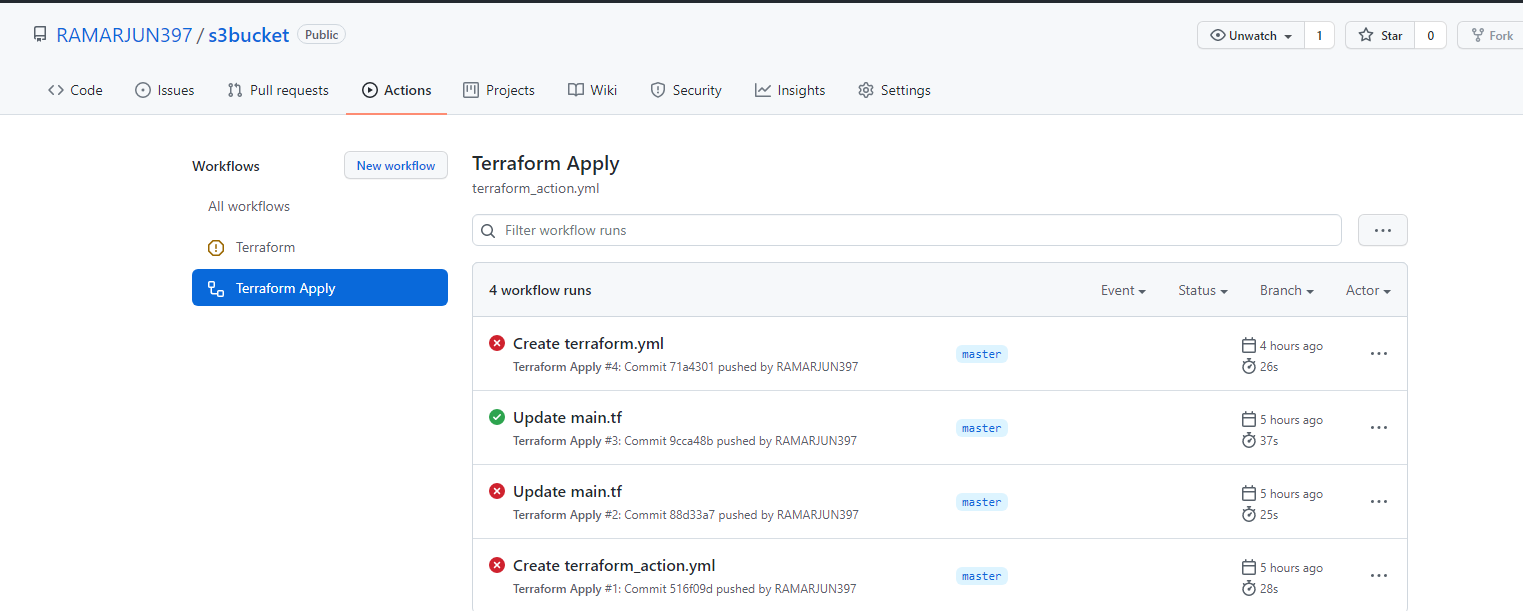
4. Click On Actions and Scroll Down You Select Terraform setup Work space.



5. Ater that give terraform action plan name and after click start commit and Save that action plane. 

Terraform action plan yml file contains the following code.

6. Goto Github action you will get the action plane.



7. The action plan will be automatically running and create our required resources in in aws console.

