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 AZure with do-anything permissions

2. Set up a new GitHub repository

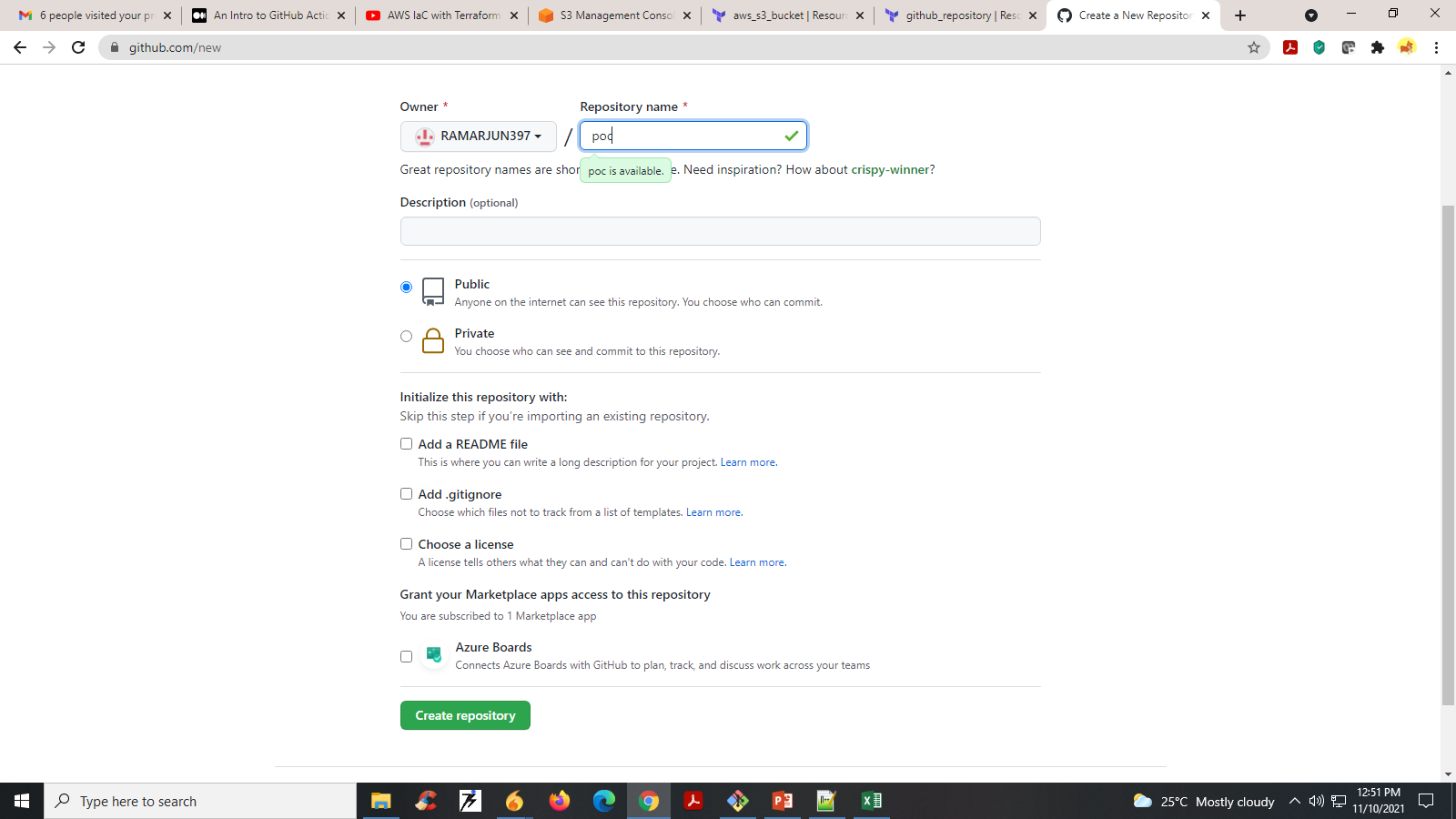
3. Store the AZURE\_AD\_CLIENT\_ID, AZURE\_AD\_TENANT\_ID, AZURE\_SUBSCRIPTION\_ID and AZURE\_AD\_CLIENT\_SECRET secret key as encrypted keys in GitHub for Actions to consume

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

Let’s get started. You can do this.

Follow The Steps Creating Azure Resource Group using GitHub and Terraform Actions:

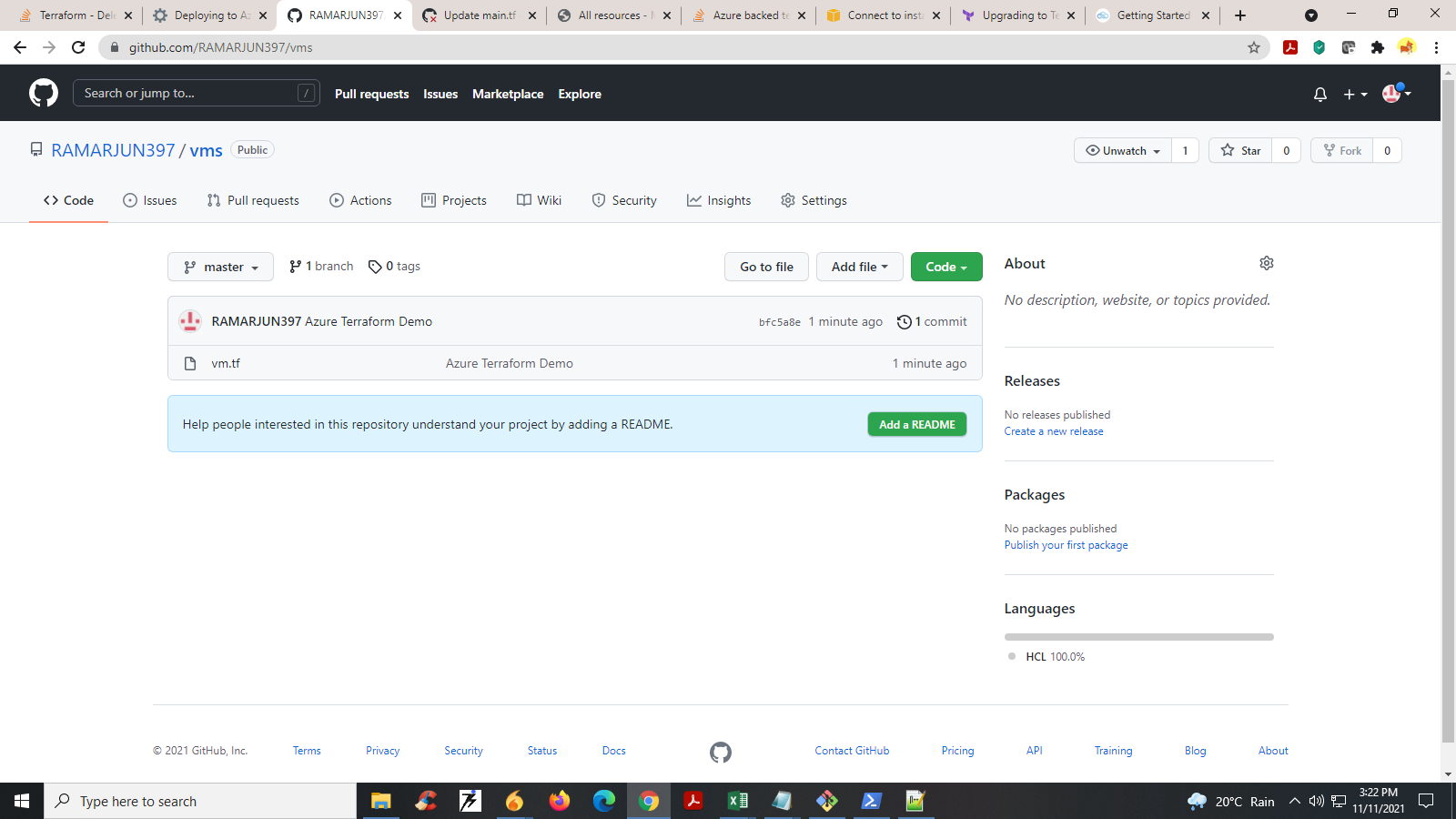
1.First we can login github account after that create new git repository and Here we can give repository name.



2. Sample Terraform Code

|  |
| --- |
| Provider"azurerm" { |
|  | version = "=2.84.0" |
|  | client\_id = "<Azure-ClientId>" |
|  | client\_secret = "<Azure-Client-Secret>" |
|  | subscription\_id = "<Azure-SubscriptionID>" |
|  | tenant\_id = "Azure-Tenant-ID" |
|  | features {} |
|  | } |
|  |  |
|  | #create resource group |
|  | resource "azurerm\_resource\_group" "rg" { |
|  | name = "rg-MyFirstTerraform" |
|  | location = "East US" |
|  | } |

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

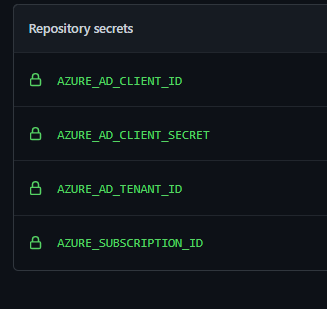


**Saving Service Principal credentials within GitHub Repository as secrets**

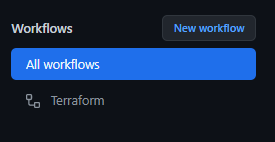
4. Within the GitHub repository to where you are going to be running the terraform from, select settings -> secrets

Add 4 secrets

* AZURE\_AD\_CLIENT\_ID – Will be the service principal ID from above
* AZURE\_AD\_CLIENT\_SECRET – The secret that was created as part of the Azure Service Principal
* AZURE\_AD\_TENANT\_ID – The Azure AD tenant ID to where the service principal was created
* AZURE\_SUBSCRIPTION\_ID – Subscription ID of where you want to deploy the Terraform



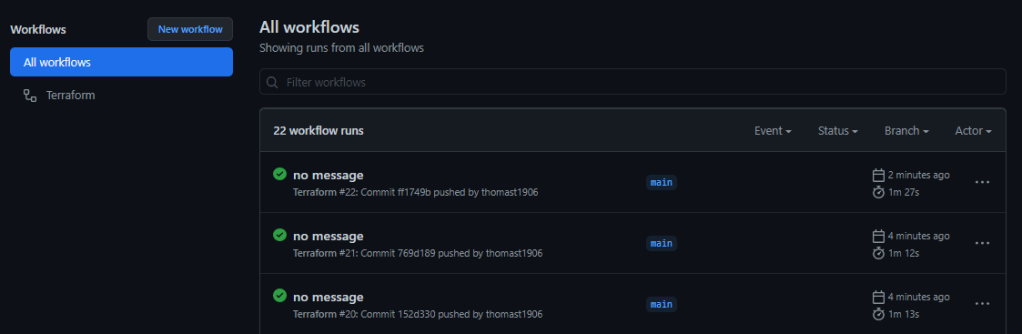
5. To add this GitHub Action to your repository, within your GitHub Repo – select Actions -> Workflows -> New workflow



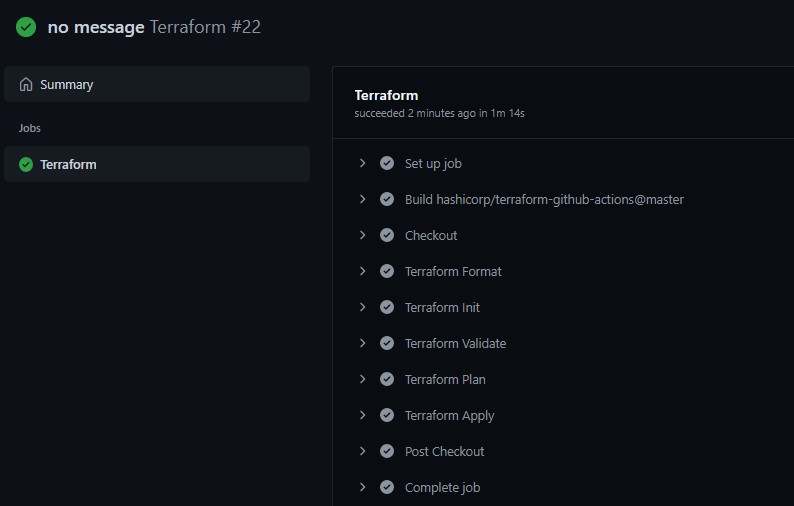
6. The Terraform Create Yml File It Contains the Following Code.

|  |
| --- |
| name: 'Terraform'    on:  push:  branches:  - main  pull\_request:    jobs:  terraform:  name: 'Terraform'  env:  ARM\_CLIENT\_ID: ${{ secrets.AZURE\_AD\_CLIENT\_ID }}  ARM\_CLIENT\_SECRET: ${{ secrets.AZURE\_AD\_CLIENT\_SECRET }}  ARM\_SUBSCRIPTION\_ID: ${{ secrets.AZURE\_SUBSCRIPTION\_ID }}  ARM\_TENANT\_ID: ${{ secrets.AZURE\_AD\_TENANT\_ID }}  runs-on: ubuntu-latest  environment: production    # Use the Bash shell regardless whether the GitHub Actions runner is ubuntu-latest, macos-latest, or windows-latest  defaults:  run:  shell: bash    steps:  # Checkout the repository to the GitHub Actions runner  - name: Checkout  uses: actions/checkout@v2    - name: 'Terraform Format'  uses: hashicorp/terraform-github-actions@master  with:  tf\_actions\_version: 0.14.8  tf\_actions\_subcommand: 'fmt'  tf\_actions\_working\_dir: "./terraform"    - name: 'Terraform Init'  uses: hashicorp/terraform-github-actions@master  with:  tf\_actions\_version: 0.14.8  tf\_actions\_subcommand: 'init'  tf\_actions\_working\_dir: "./terraform"    - name: 'Terraform Validate'  uses: hashicorp/terraform-github-actions@master  with:  tf\_actions\_version: 0.14.8  tf\_actions\_subcommand: 'validate'  tf\_actions\_working\_dir: "./terraform"    - name: 'Terraform Plan'  uses: hashicorp/terraform-github-actions@master  with:  tf\_actions\_version: 0.14.8  tf\_actions\_subcommand: 'plan'  tf\_actions\_working\_dir: "./terraform"    - name: Terraform Apply  if: github.ref == 'refs/heads/main'  uses: hashicorp/terraform-github-actions@master  with:  tf\_actions\_version: 0.14.8  tf\_actions\_subcommand: 'applya'  tf\_actions\_working\_dir: "./terraform" |

7.Once the Action has been ran, we can review each step



8. Select any of these workflow runs and you can drill-down into each step and review its output.



Hopefully you find this POC useful and give you a look into deploying Terraform using GitHub Actions.

|  |
| --- |
|  |