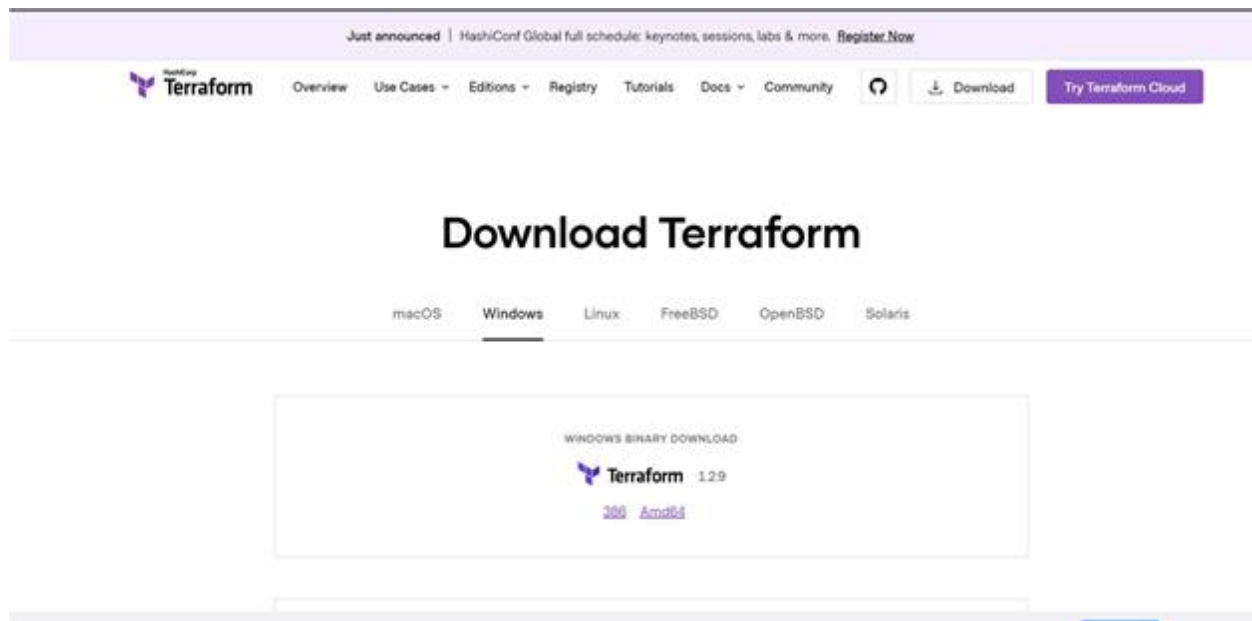
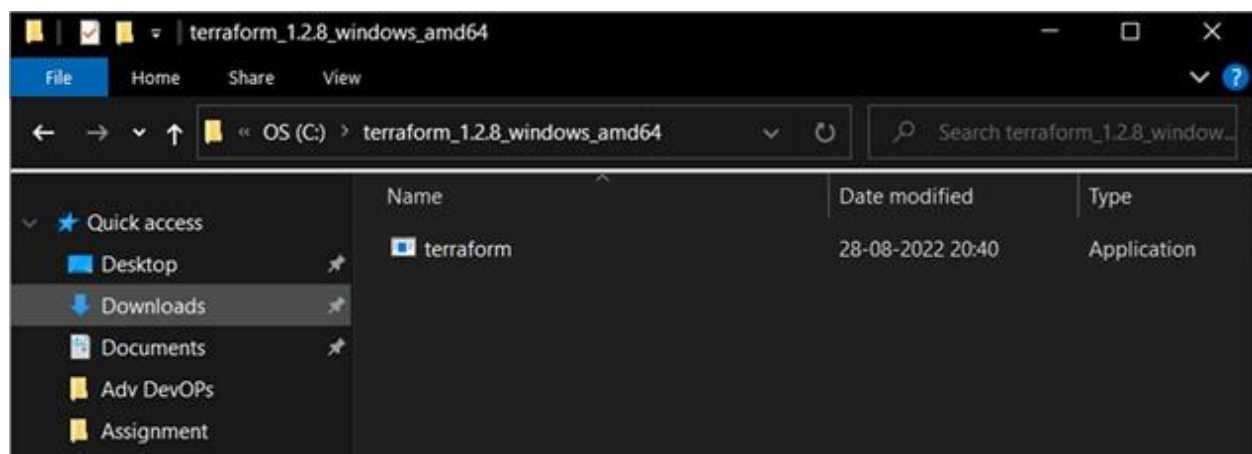


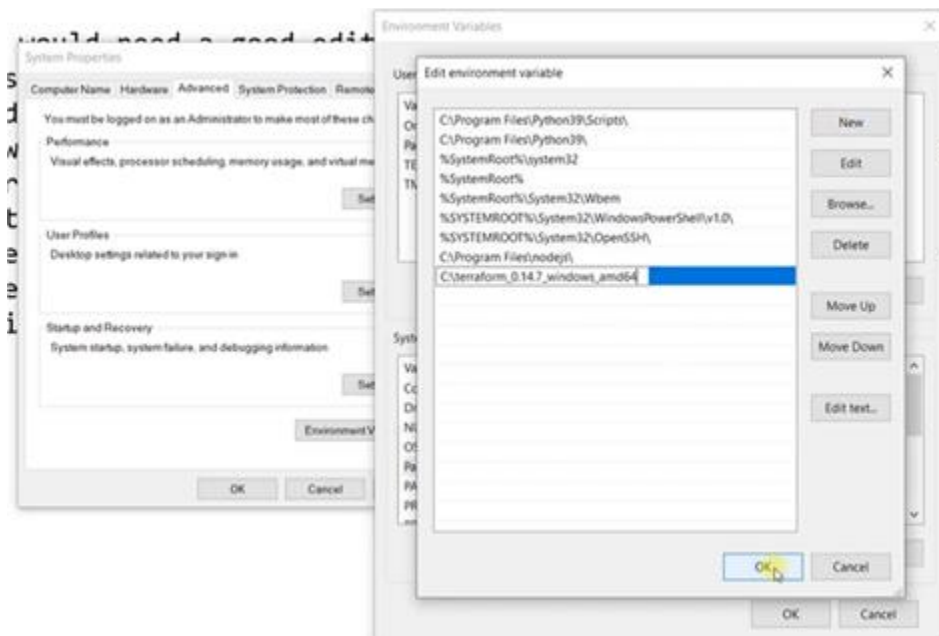
1. Download and Install Terraform and Setup Path



2. After Installation Extract the folder C:\terraform_1.2.8_windows_amd64



3. Edit Environment Variable path



4. Write the Code in main.tf file

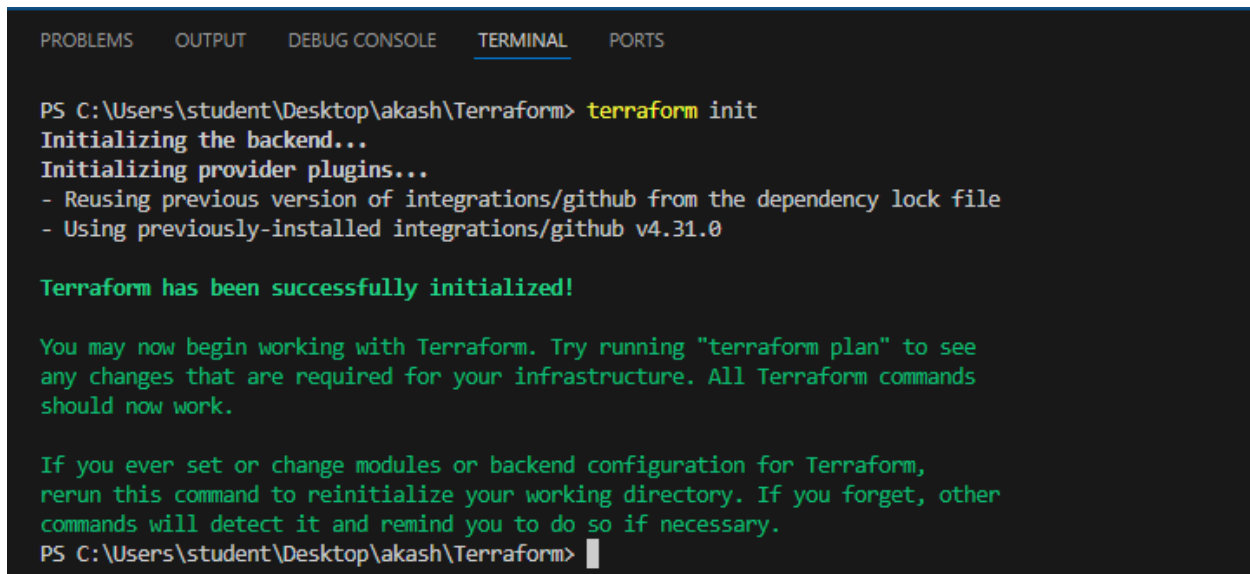
```
main.tf  ×  provider.tf
main.tf > resource "github_repository" "name"
1  resource "github_repository" "name" {
2      name="akash"
3      description="my code"
4      visibility="public"
5  }
```

5. Write the code in provide.tf file and provide the github account token in the code



```
main.tf provider.tf X
provider.tf > provider "github"
1 terraform {
2   required_providers {
3     github={
4       source="integrations/github"
5       version = "~> 6.0"
6     }
7   }
8 }
9 provider "github" {}
10   token="ghp_PHPMc9qBWckk1oLfrkju8r6G3aj1Lj3reR1D"
11
12 }
```

6. Run terraform init command in terminal to initialize terraform



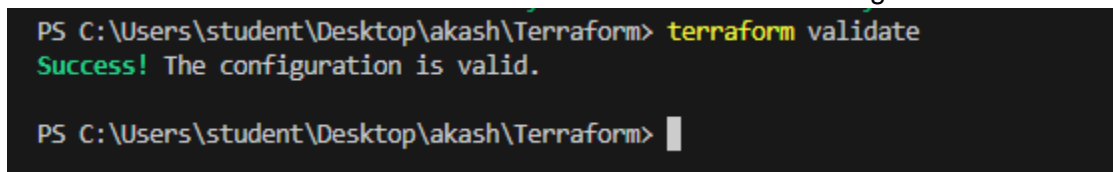
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\student\Desktop\akash\Terraform> terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of integrations/github from the dependency lock file
- Using previously-installed integrations/github v4.31.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\student\Desktop\akash\Terraform> 
```

7. Run terraform validate command in terminal to validate the configuration



```
PS C:\Users\student\Desktop\akash\Terraform> terraform validate
Success! The configuration is valid.

PS C:\Users\student\Desktop\akash\Terraform> 
```

8. Run terraform plan command in terminal

```
PS C:\Users\student\Desktop\akash\Terraform> terraform plan
```

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

```
# github_repository.git_AK_repo will be created
+ resource "github_repository" "git_AK_repo" {
  + allow_auto_merge      = false
  + allow_merge_commit    = true
  + allow_rebase_merge    = true
  + allow_squash_merge     = true
  + archived              = false
  + branches               = (known after apply)
  + default_branch        = (known after apply)
  + delete_branch_on_merge = false
  + description            = "My awesome codebase"
  + etag                   = (known after apply)
  + full_name              = (known after apply)
  + git_clone_url          = (known after apply)
  + html_url               = (known after apply)
  + http_clone_url         = (known after apply)
  + id                     = (known after apply)
  + merge_commit_message   = "PR_TITLE"
  + merge_commit_title     = "MERGE_MESSAGE"
  + name                   = "Akash"
  + node_id                = (known after apply)
  + private                = (known after apply)
  + repo_id                = (known after apply)
  + squash_merge_commit_message = "COMMIT_MESSAGES"
  + squash_merge_commit_title  = "COMMIT_OR_PR_TITLE"
  + ssh_clone_url           = (known after apply)
  + svn_url                 = (known after apply)
  + visibility              = "public"
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

9.Run terraform apply command in terminal to apply the changes

```
PS C:\Users\student\Desktop\akash\Terraform> terraform apply
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create

Terraform will perform the following actions:

```
# github_repository.git_AK_repo will be created
+ resource "github_repository" "git_AK_repo" {
  + allow_auto_merge      = false
  + allow_merge_commit    = true
  + allow_rebase_merge    = true
  + allow_squash_merge     = true
  + archived              = false
  + branches               = (known after apply)
  + default_branch        = (known after apply)
  + delete_branch_on_merge = false
  + description            = "My awesome codebase"
  + etag                  = (known after apply)
  + full_name              = (known after apply)
  + git_clone_url          = (known after apply)
  + html_url              = (known after apply)
  + http_clone_url        = (known after apply)
  + id                    = (known after apply)
  + merge_commit_message  = "PR_TITLE"
  + merge_commit_title    = "MERGE_MESSAGE"
  + name                  = "Akash"
  + node_id               = (known after apply)
  + private               = (known after apply)
  + repo_id               = (known after apply)
  + squash_merge_commit_message = "COMMIT_MESSAGES"
  + squash_merge_commit_title  = "COMMIT_OR_PR_TITLE"
  + ssh_clone_url         = (known after apply)
  + svn_url               = (known after apply)
  + visibility            = "public"
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?

Terraform will perform the actions described above.

Only 'yes' will be accepted to approve.

Enter a value: yes

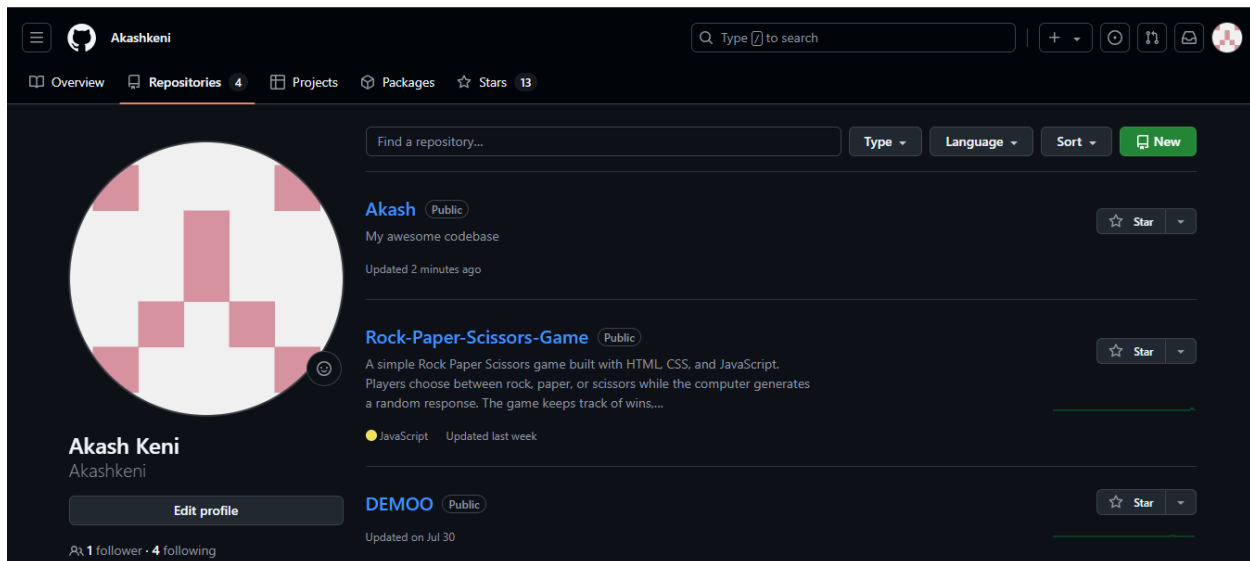
github_repository.git_AK_repo: Creating...

github_repository.git_AK_repo: Creation complete after 6s [id=Akash]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

```
PS C:\Users\student\Desktop\akash\Terraform> |
```

10. Check github account for the new repository



11. Run terraform destroy command to delete the repository

```
PS C:\Users\student\Desktop\akash\Terraform> terraform destroy
github_repository.git_AK_repo: Refreshing state... [id-Akash]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
- destroy

Terraform will perform the following actions:

# github_repository.git_AK_repo will be destroyed
- resource "github_repository" "git_AK_repo" {
  - allow_auto_merge           = false -> null
  - allow_merge_commit         = true -> null
  - allow_rebase_merge         = true -> null
  - allow_squash_merge          = true -> null
  - archived                   = false -> null
  - branches                   = [] -> null
  - default_branch              = "main" -> null
  - delete_branch_on_merge      = false -> null
  - description                 = "My awesome codebase" -> null
  - etag                       = "W/\"74567b05f37ff0b95ed592138201cd2cbe1d448879fd68d23d5cbab447c24b20\"" -> null
  - full_name                   = "Akashkeni/Akash" -> null
  - git_clone_url               = "git://github.com/Akashkeni/Akash.git" -> null
  - has_downloads               = false -> null
  - has_issues                  = false -> null
  - has_projects                = false -> null
  - has_wiki                    = false -> null
  - html_url                   = "https://github.com/Akashkeni/Akash" -> null
  - http_clone_url              = "https://github.com/Akashkeni/Akash.git" -> null
  - id                         = "Akash" -> null
  - is_template                 = false -> null
  - merge_commit_message        = "PR_TITLE" -> null
  - merge_commit_title          = "MERGE_MESSAGE" -> null
  - name                       = "Akash" -> null
  - node_id                    = "R_kgDOMz4iTw" -> null
}
```

```

- node_id           = "R_kgDOMz4iTw" -> null
- private           = false -> null
- repo_id           = 859710031 -> null
- squash_merge_commit_message = "COMMIT_MESSAGES" -> null
- squash_merge_commit_title   = "COMMIT_OR_PR_TITLE" -> null
- ssh_clone_url        = "git@github.com:Akashkeni/Akash.git" -> null
- svn_url              = "https://github.com/Akashkeni/Akash" -> null
- visibility           = "public" -> null
- vulnerability_alerts = false -> null
  # (1 unchanged attribute hidden)
}

```

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

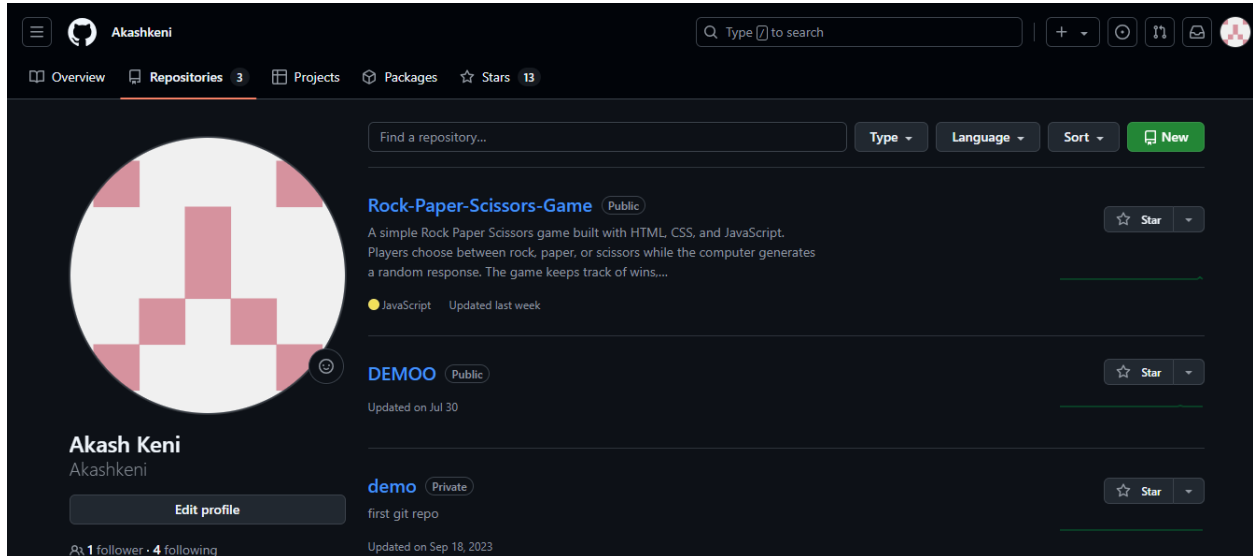
```

github_repository.git_AK_repo: Destroying... [id=Akash]
github_repository.git_AK_repo: Destruction complete after 1s

```

Destroy complete! Resources: 1 destroyed.

12. Check the account to see the deleted repository



The screenshot shows the GitHub profile of Akashkeni. The profile includes a bio, a profile picture, and a list of repositories. The repository 'Rock-Paper-Scissors-Game' is shown with a large grey 'X' over it, indicating it has been deleted. Below it, there is a repository named 'DEMOO' and a private repository named 'demo'.