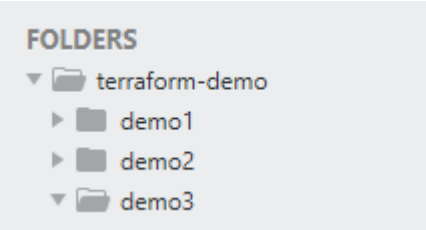
😊 Provisioners

1. Now for this lab, open your code editor and create a new sub folder in the main folder.

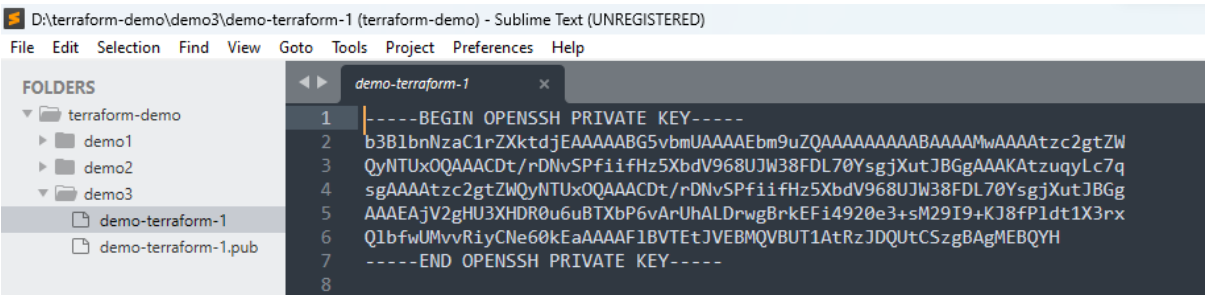


2. Once it is done now you are going to create keys. For that open your gitbash and write this command. Then press enter now you have to give your key a name and press enter then you will see that your key has been generated. There will be two keys private and public.

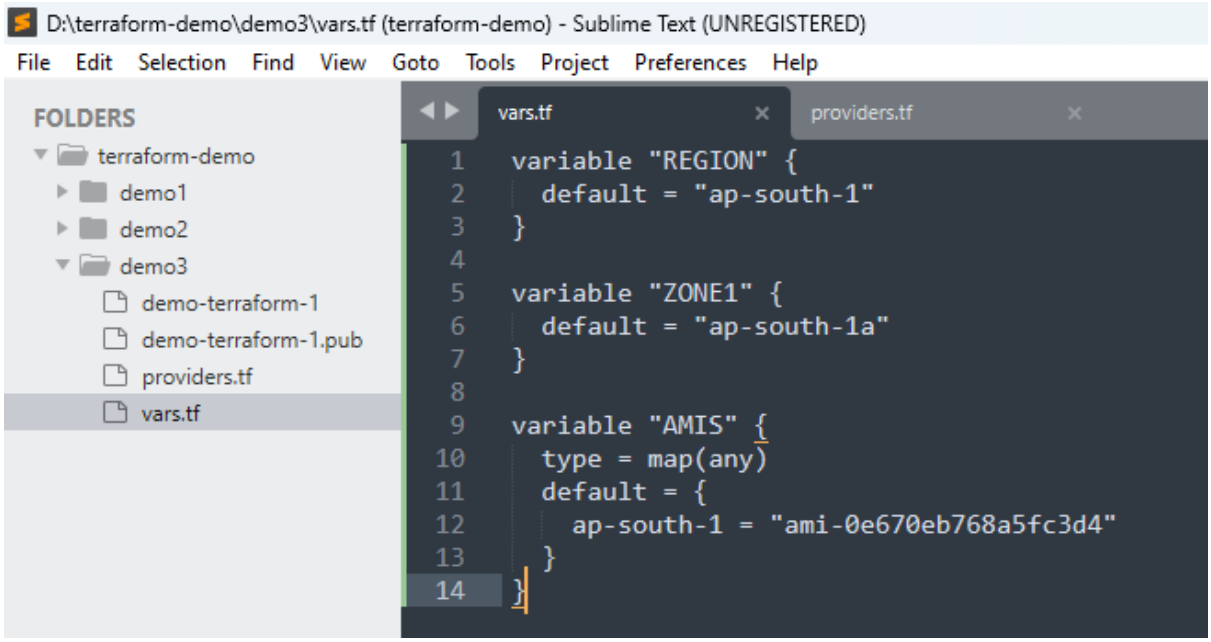
ssh-keygen

```
$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/PULKIT/.ssh/id_ed25519): demo-terraform-1
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in demo-terraform-1
Your public key has been saved in demo-terraform-1.pub
The key fingerprint is:
SHA256:QZmq5dwEF6nmt8+b4muTEtrWWhcnr/gp2UZOWa8aXLI        @LAPTOP-G2CAKBK8
The key's randomart image is:
+--[ED25519 256]--+
|         o=      |
|       ..=       |
|        =.       |
|       = ..  .   |
|      B oS +oo.  |
|     . = o.+O  . |
|      o +.OE ..  |
|     . +.X++=.   |
|      ..=+OXo    |
+----[SHA256]-----+
```

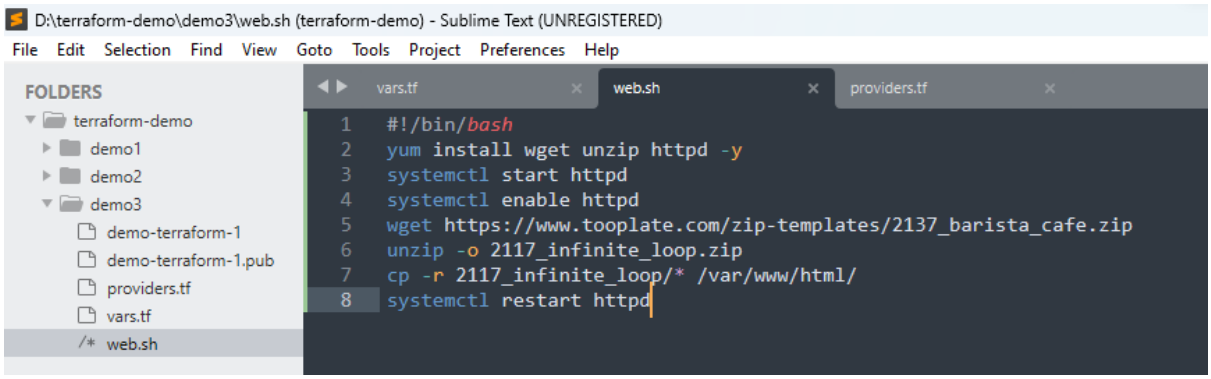
3. Now if you will come to your code editor you will see the keys in your folder.



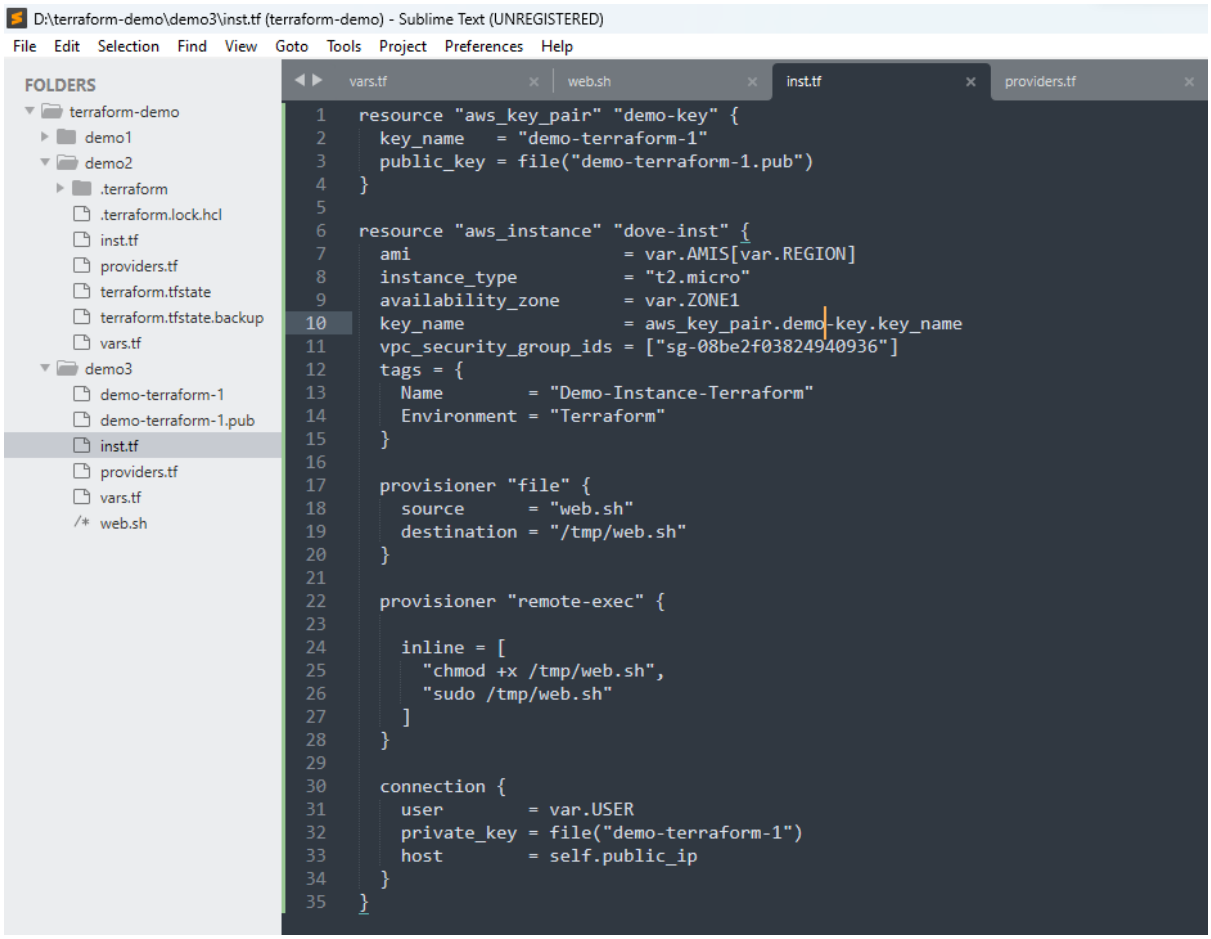
4. Now again like you did in the previous lab, first you are going to create the variables files and then you are going to create your provider's file.
5. The code is same as before you have to copy the code and paste it here.



6. Once it is done now you are going to create a shell script. Now create a new file and save it as web.sh



7. Again, you can get this code from GitHub.
8. Once this is done now you are going to create the instance file.



9. Once all this is done now you need to go to gitbash and run the commands to execute your code.
10. First, you need to do a listing of your files in gitbash.

```
$ ls
demo-terraform-1  demo-terraform-1.pub  inst.tf  providers.tf  vars.tf  web.sh*
```

11. Then you are going to initialize it.

    terraform init

```
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.38.0...
- Installed hashicorp/aws v5.38.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

12. Now do the validate of your code. And then after that format your code.

    terraform validate
    terraform fmt

```
$ terraform validate
Success! The configuration is valid.
```

13. Now run the plan command to see for any changes in your code. As you can see two things are added which are your keys private and public and the instance.

    terraform plan

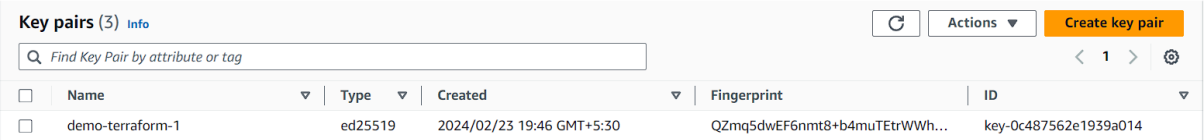
```
  # aws_key_pair.demo-key will be created
  + resource "aws_key_pair" "demo-key" {
      + arn             = (known after apply)
      + fingerprint     = (known after apply)
      + id              = (known after apply)
      + key_name        = "demo-terraform-1"
      + key_name_prefix = (known after apply)
      + key_pair_id     = (known after apply)
      + key_type        = (known after apply)
      + public_key      = "ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIO3+sM29I9+KJ8fPldt1X3rxQlbfwUMvvRiyCNe60kEa PULKIT@LAPTOP-G2CAKBK8"
      + tags_all        = (known after apply)
    }

Plan: 2 to add, 0 to change, 0 to destroy.
```

14. Now execute the code. Now the data is too much to be covered in the screen shot.

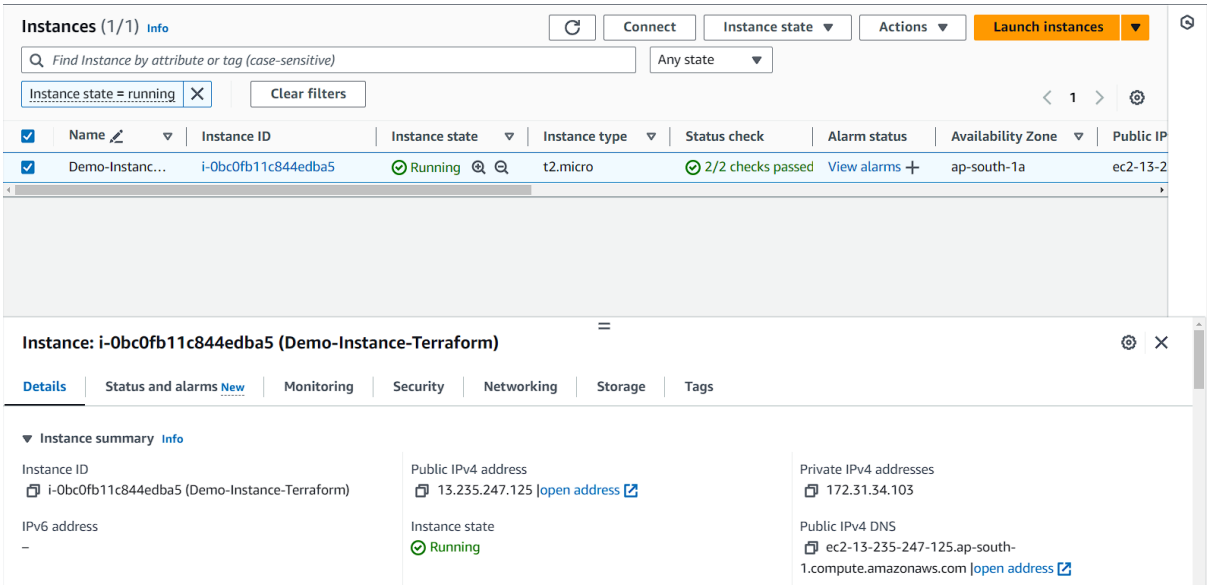
    terraform apply

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

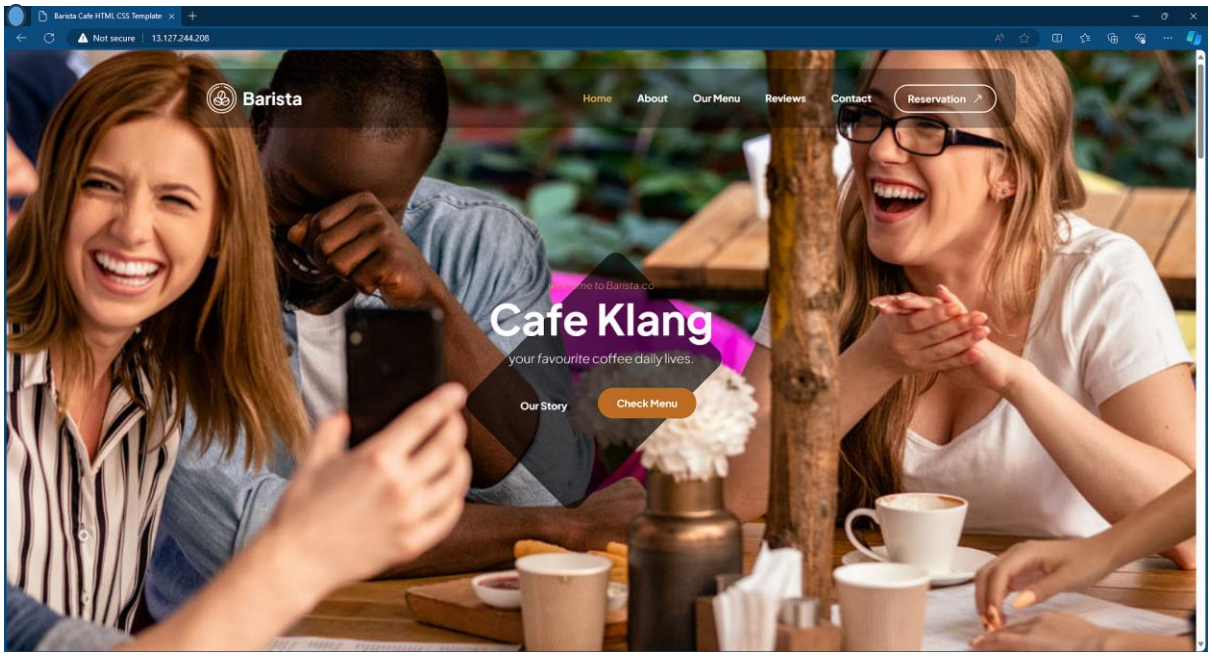
15. So, one thing more when you have executed the command terraform apply, go and check the security group for it. In the security group you have to add port 22 for SSH. If you did not add that so, it will get stuck.
16. Once your instance is created first go to the key pairs in AWS Console. You will see your key pair.



17. Now here is the instance that got created.



18. Now copy the public IP address of your instance and paste it in a new tab.
19. There you will see that the instance that we created is provisioned with the website.



20. Once you are done run the destroy command which will destroy all the resources including the key pair.

    terraform 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

aws_instance.demo-inst: Destroying... [id=i-0ef01ac7ab68944a7]
aws_instance.demo-inst: Still destroying... [id=i-0ef01ac7ab68944a7, 10s elapsed]
aws_instance.demo-inst: Still destroying... [id=i-0ef01ac7ab68944a7, 20s elapsed]
aws_instance.demo-inst: Still destroying... [id=i-0ef01ac7ab68944a7, 30s elapsed]
aws_instance.demo-inst: Destruction complete after 30s
aws_key_pair.demo-key: Destroying... [id=demo-terraform-1]
aws_key_pair.demo-key: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```