CLCM3504 - Lab 2 Provisioning EC2 as a Web Server and S3 to Host a Static Website via Terraform

Akilandeshwari Srinivasan (451036)

Due Date: 5th of November 2023

Preparation:

Launch aws website and get credentials.

Go through the hashicorp website to know more about how to create and implement S3 and EC2 instances

Difference between EC2 and S3 web hosting:

When it comes to hosting a website on AWS, Amazon EC2 is like having your own virtual computer where you can fully customize everything. It's great for complex websites that need a lot of control.

Amazon S3, on the other hand, is more like a simple storage space for your web files. It's perfect for basic websites with static content like HTML and images, and it's easy and cost-effective.

So, choose EC2 for complex websites and S3 for simple ones.

Alternatives for CI/CD & third party applications:

For cloud-based applications, AWS-native CI/CD services like AWS CodePipeline and CodeBuild offer seamless integration with AWS resources. They are easy to set up but may be AWS-centric.

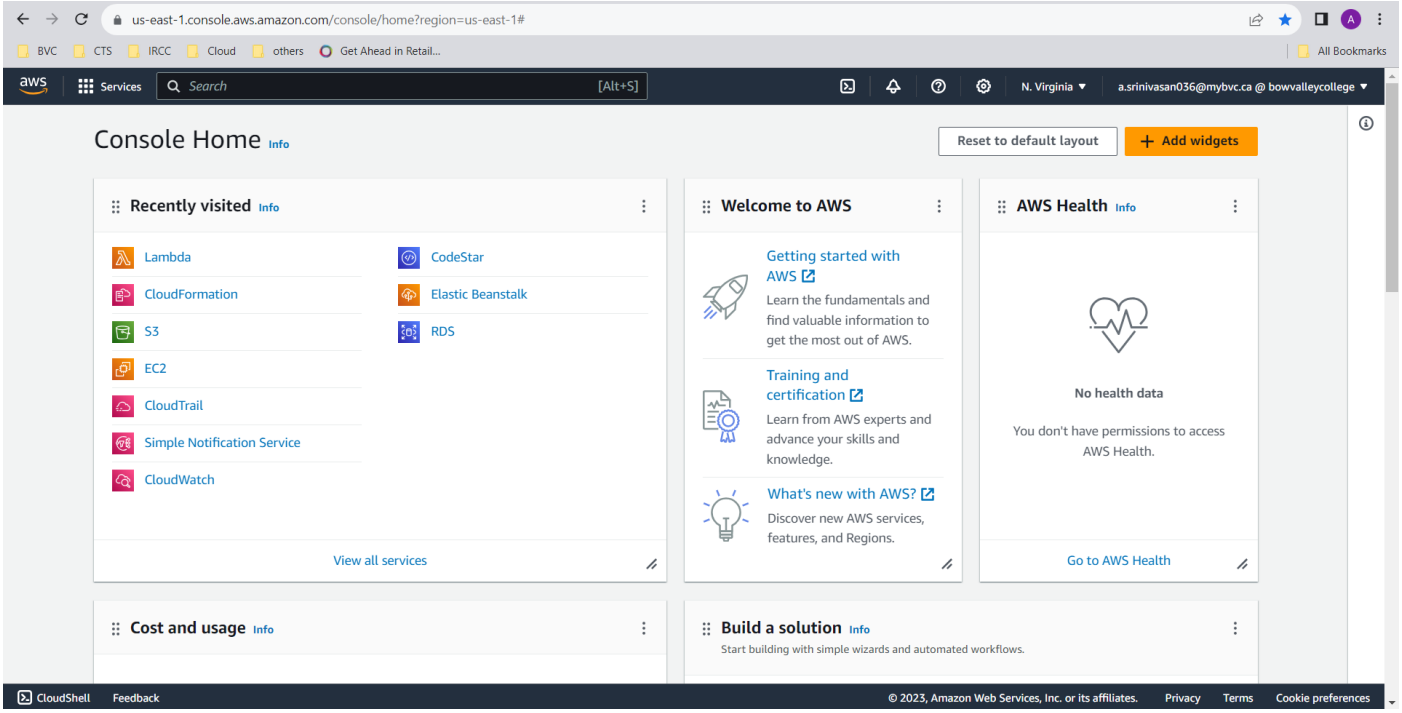
Third-party solutions such as Jenkins and CircleCI provide more flexibility and support multiple cloud providers. They require more configuration but offer broader compatibility. Choose based on your specific cloud environment and project requirements.

Benefits of considering other CI/CD applications:

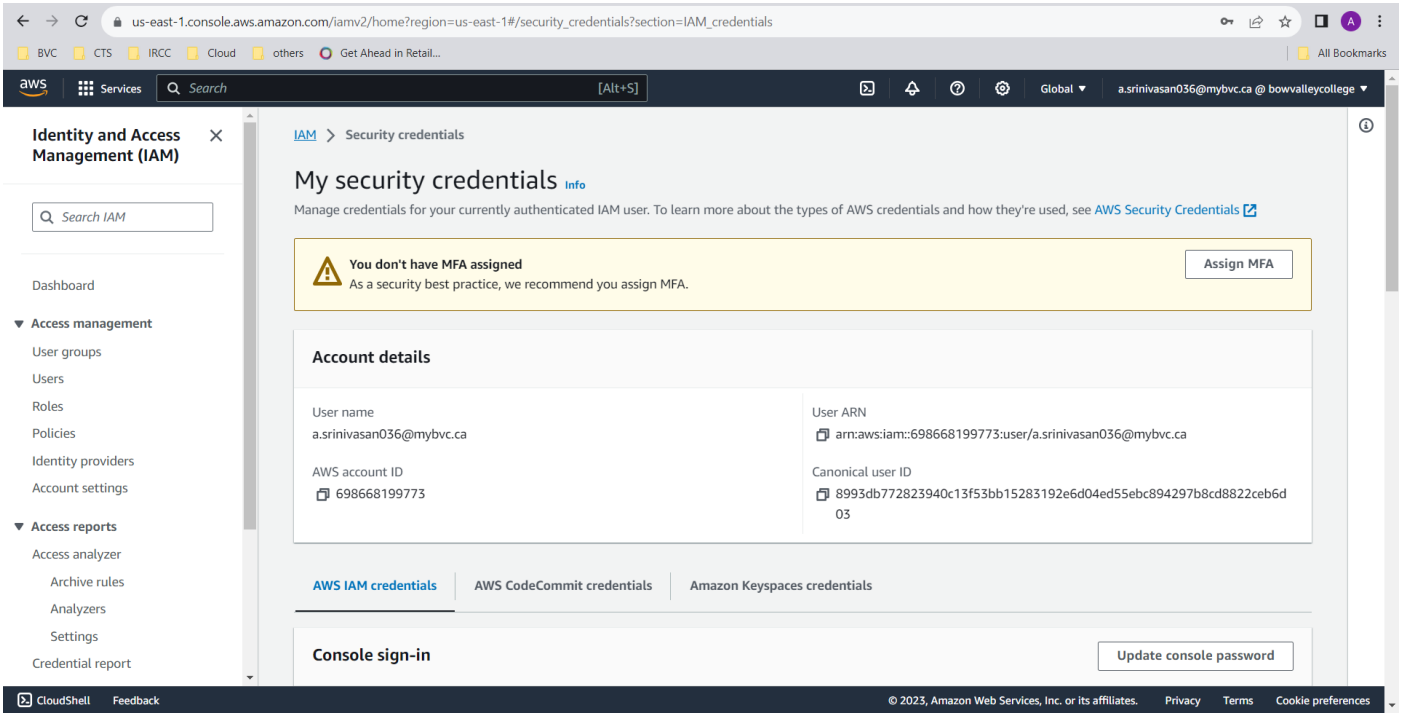
The choice of CI/CD approach for cloud-based applications should align with specific project needs. AWS-native services provide seamless integration and ease of use, particularly for AWS-centric projects. Third-party solutions offer flexibility and support for multiple cloud providers, making them suitable for diverse environments. The decision should consider factors like project scope, required integrations, and long-term scalability to deliver the most effective CI/CD pipeline.

S3 for web hosting using terraform:

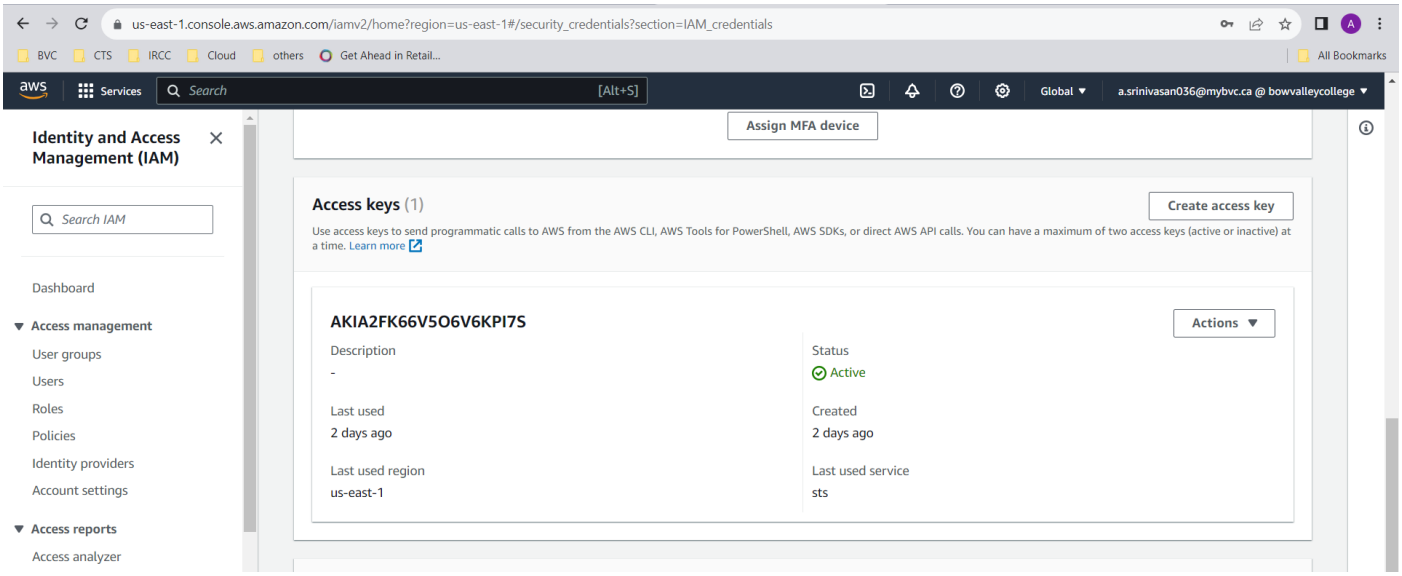
Open the console management for obtaining the security credentials.

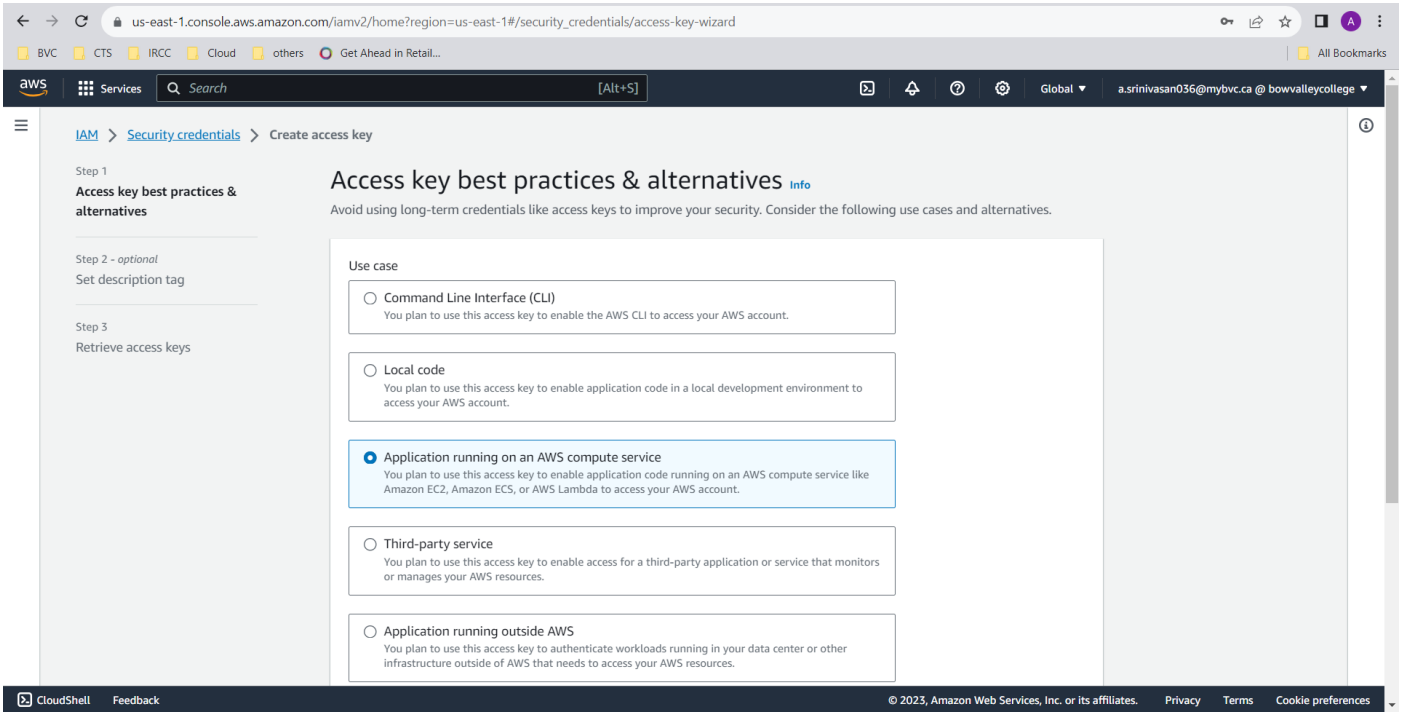


On the profile you can find the security credentials if you have two or more security credentials delete one and create a new one.

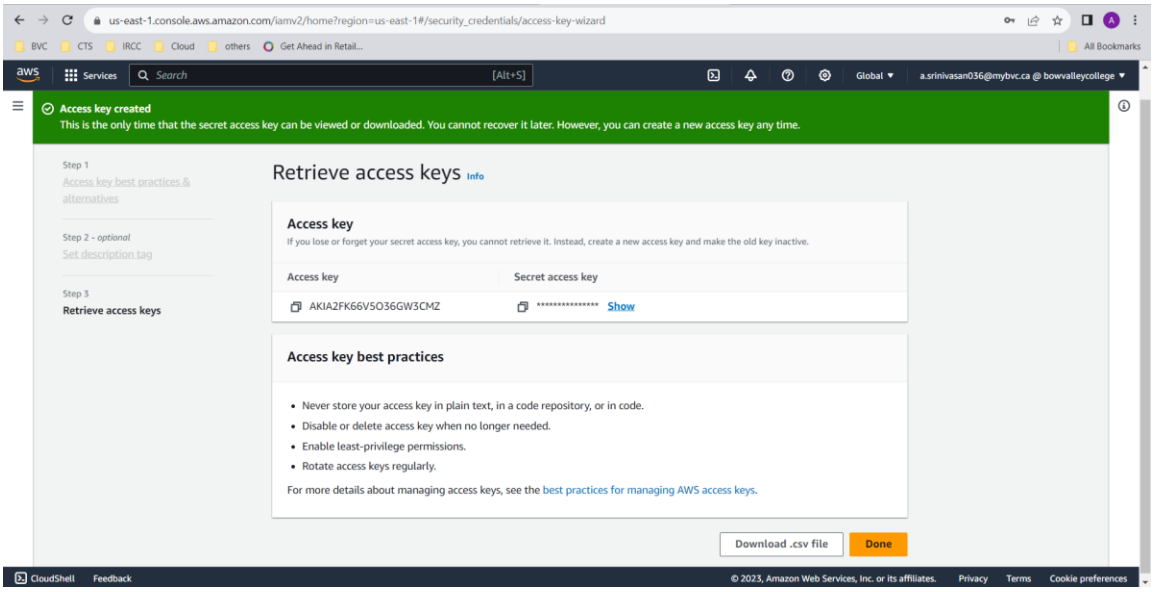


Click the button create access key to create new one





Copy the secret key and access to configure



Now open the cmd and type “aws configure”. And give the credentials which you have copied.

```
Command Prompt

Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\akila>aws configure
AWS Access Key ID [****************PI7S]: AKIA2FK66V5O36GW3CMZ
AWS Secret Access Key [****************B8W/]: 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
Default region name [None]:
Default output format [None]:

C:\Users\akila>

C:\Users\akila>
```

Post that navigate to the path where you have created a folder for the s3 terraform code. Give initiate the terraform code by giving the cmd “terraform init”

```
Command Prompt

Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\akila>aws configure
AWS Access Key ID [****************3CMZ]: AKIA2FK66V5O36GW3CMZ
AWS Secret Access Key [****************PJcH]: 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
Default region name [None]:
Default output format [None]:

C:\Users\akila>cd C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\s3terraform

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\s3terraform>terraform init
```

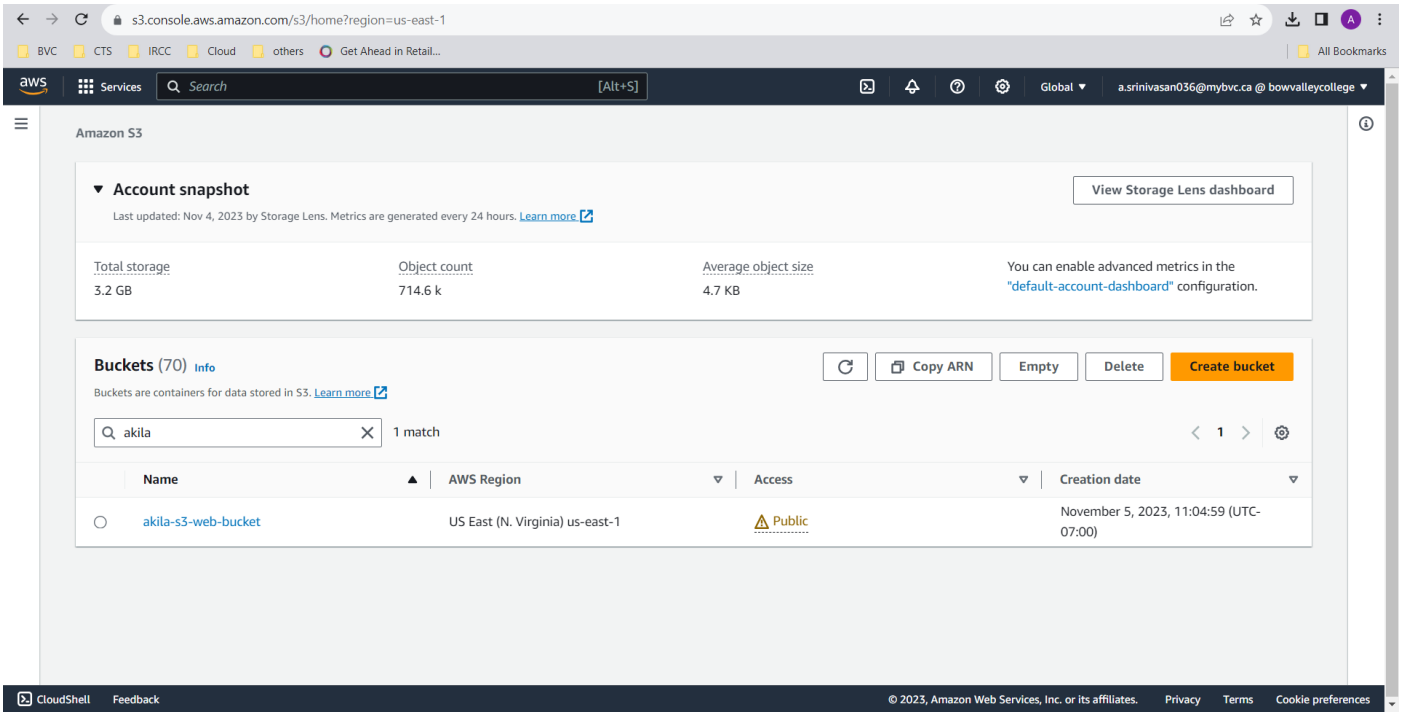
Once initialized then give” terraform plan ” followed by “terraform apply”

```
Command Prompt

Default region name [None]:
Default output format [None]:

C:\Users\akila>cd C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\s3terraform

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\s3terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "4.0.0"...
- Installing hashicorp/aws v4.0.0...
- Installed hashicorp/aws v4.0.0 (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.

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\s3terraform>
```

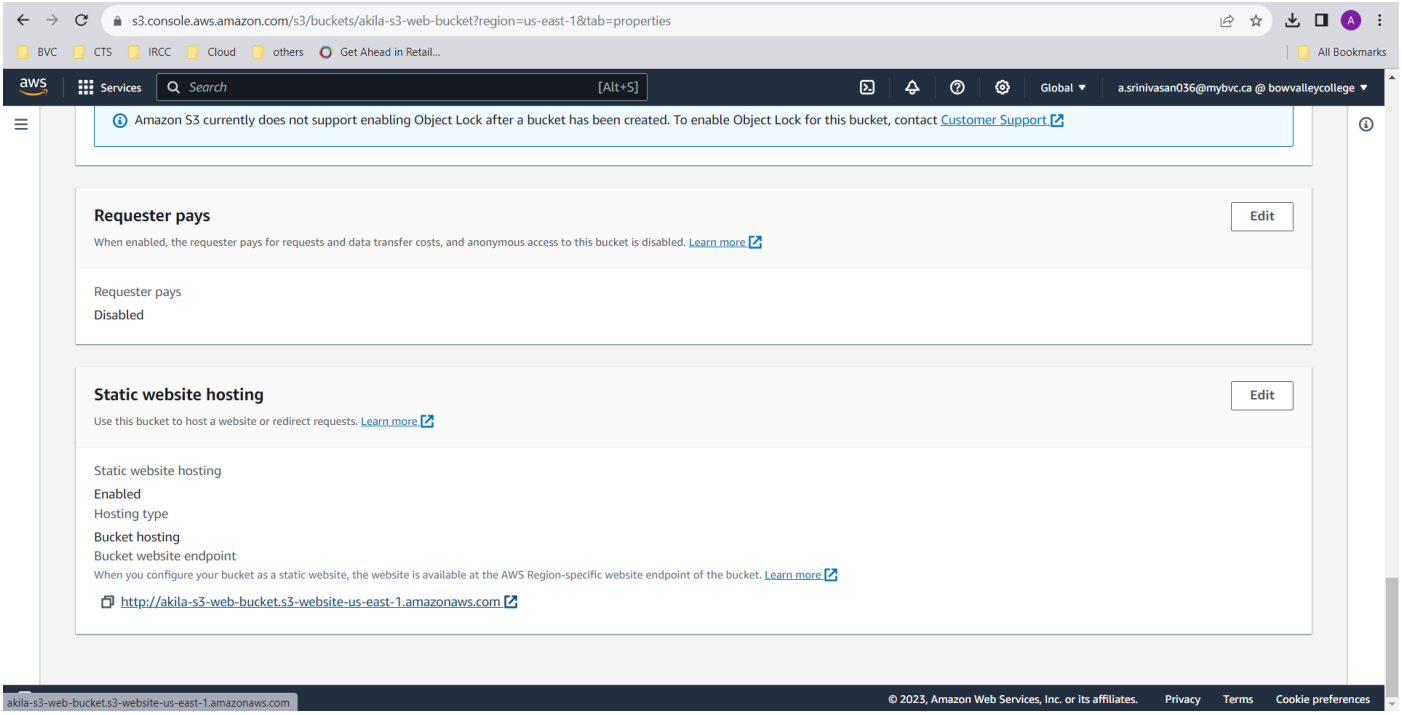
```
Command Prompt

      + bucket           = (known after apply)
      + id               = (known after apply)
      + website_domain   = (known after apply)
      + website_endpoint = (known after apply)

      + index_document {
          + suffix = "index.html"
        }
    }

  # aws_s3_object.object will be created
  + resource "aws_s3_object" "object" {
      + acl                    = "private"
      + bucket                 = "akila-s3-static-bucket"
      + bucket_key_enabled     = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "index.html"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "./html/index.html"
      + storage_class          = (known after apply)
      + tags_all               = (known after apply)
      + version_id             = (known after apply)
    }

Plan: 7 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + bucket_endpoint     = (known after apply)
  + domain_name         = "akila-s3-static-bucket"
  + website_bucket_name = (known after apply)

──────────────────────────────────────────────────────────────────────────────────────────

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.
```

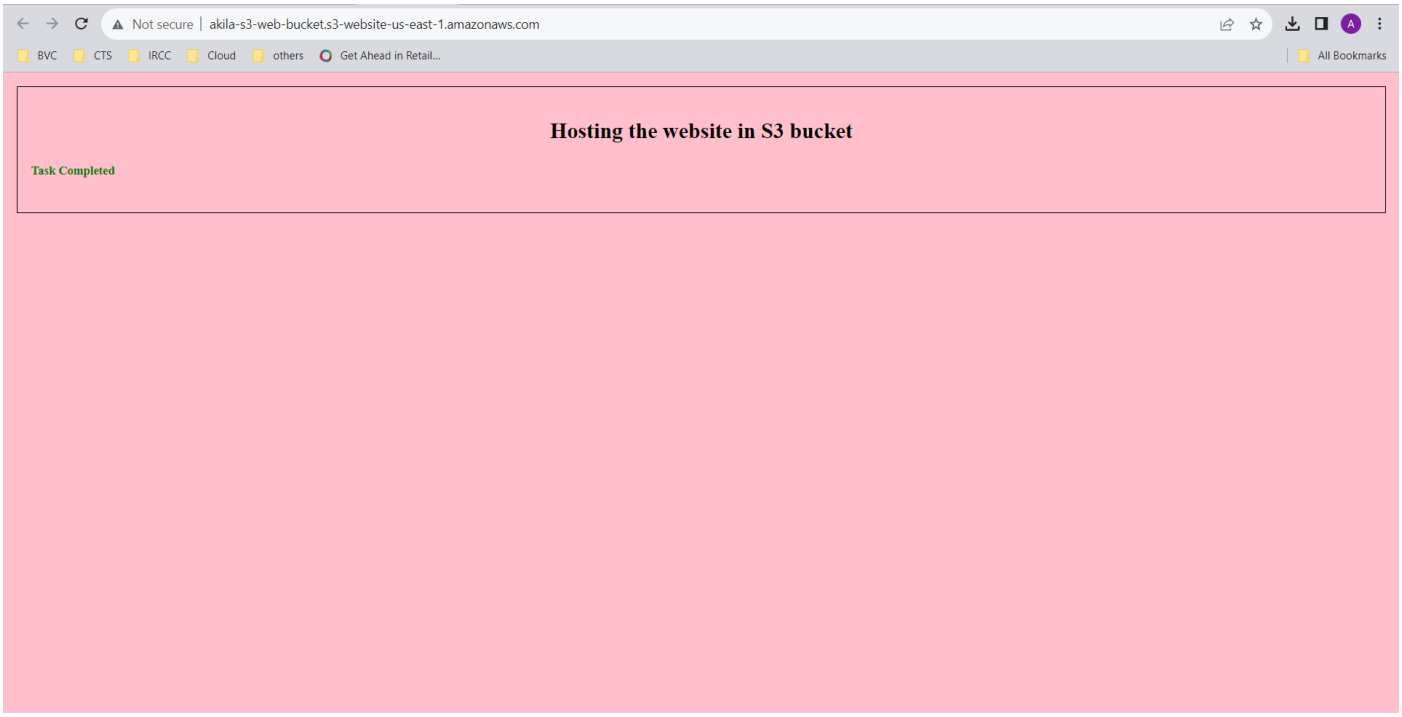
```
Command Prompt

Terraform will perform the following actions:

  # aws_s3_object.object will be created
  + resource "aws_s3_object" "object" {
      + acl                    = "private"
      + bucket                 = "akila-s3-static-bucket"
      + bucket_key_enabled     = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "index.html"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "./html/index.html"
      + storage_class          = (known after apply)
      + tags_all               = (known after apply)
      + version_id             = (known after apply)
    }

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

aws_s3_object.object: Creating...
aws_s3_object.object: Creation complete after 2s [id=index.html]

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

Outputs:

bucket_endpoint = "akila-s3-static-bucket.s3-website-us-east-1.amazonaws.com"
domain_name = "akila-s3-static-bucket"
website_bucket_name = "akila-s3-static-bucket"

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\s3terraform>
```

Once the terraform apply is done we can see the bucket in the s3 in aws management console.



Move to the properties and scroll down to bottom for the link for the static web hosting website.

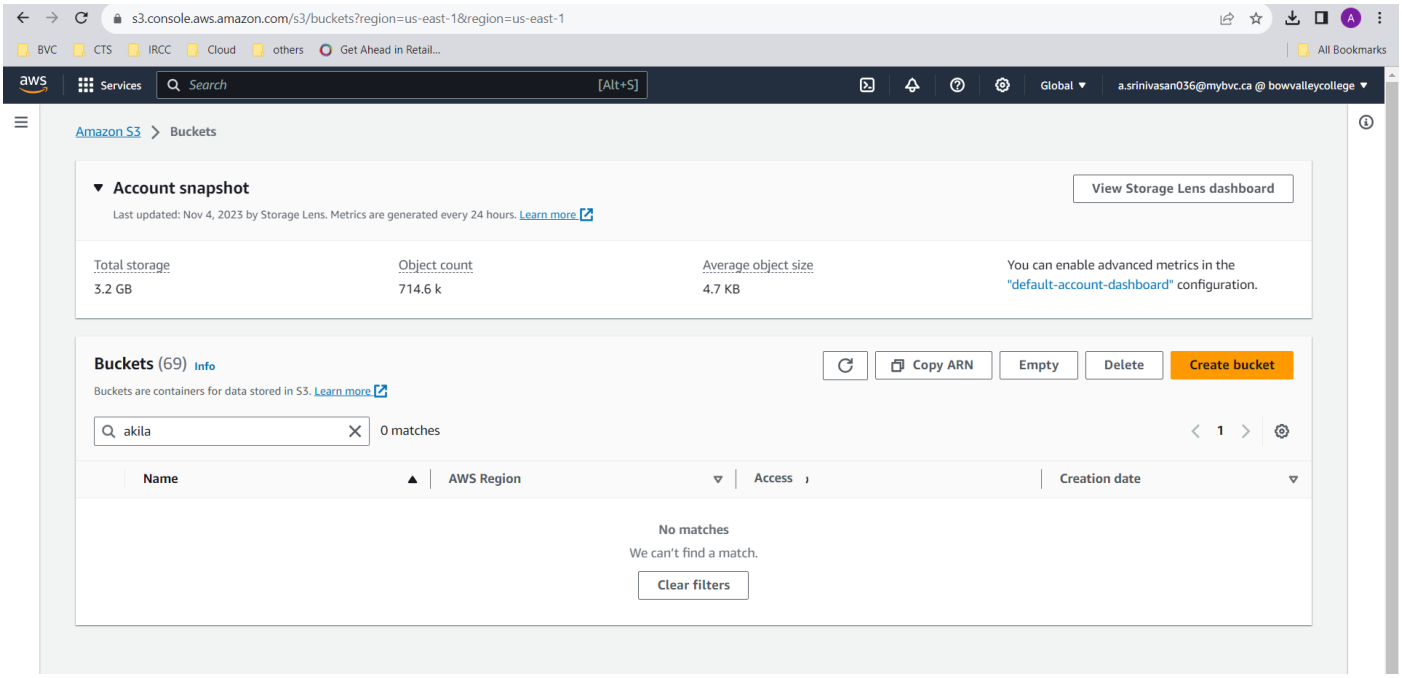




Then delete the s3 by giving the cmd “terraform destroy”

```
Command Prompt

Terraform will perform the following actions:

  # aws_s3_object.object will be created
  + resource "aws_s3_object" "object" {
      + acl                    = "private"
      + bucket                 = "akila-s3-static-bucket"
      + bucket_key_enabled     = (known after apply)
      + content_type           = (known after apply)
      + etag                   = (known after apply)
      + force_destroy          = false
      + id                     = (known after apply)
      + key                    = "index.html"
      + kms_key_id             = (known after apply)
      + server_side_encryption = (known after apply)
      + source                 = "./html/index.html"
      + storage_class          = (known after apply)
      + tags_all               = (known after apply)
      + version_id             = (known after apply)
    }

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

aws_s3_object.object: Creating...
aws_s3_object.object: Creation complete after 2s [id=index.html]

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

Outputs:

bucket_endpoint = "akila-s3-static-bucket.s3-website-us-east-1.amazonaws.com"
domain_name = "akila-s3-static-bucket"
website_bucket_name = "akila-s3-static-bucket"

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\s3terraform>terraform destroy
```

After running cmd the bucket got deleted.



In provider give the region. Here I have used the variables files where I have given the region name, domain name,etc.,

Line 5: create the bucket with the name as the value in the variable site_domain.

Line 10: making the public accessible of the bucket as false

Line 18: website configuring like where to start the website “index.html”

main.tf s3terraform × ec2.tf main.tf terraformEC2

s3terraform > main.tf > resource "aws_s3_object" "object" > source

```
1  provider "aws" {
2    region = var.aws_region
3  }
4
5  resource "aws_s3_bucket" "akila_s3_static_bucket" {
6    bucket = var.site_domain
7  }
8
9
10 resource "aws_s3_bucket_public_access_block" "akila_s3_static_bucket" {
11   bucket = aws_s3_bucket.akila_s3_static_bucket.id
12   block_public_acls       = false
13   block_public_policy     = false
14   ignore_public_acls      = false
15   restrict_public_buckets = false
16 }
17
18 resource "aws_s3_bucket_website_configuration" "akila_s3_static_bucket" {
19   bucket = aws_s3_bucket.akila_s3_static_bucket.id
20
21   index_document {
22     suffix = "index.html"
23   }
24
25 }
26
27 resource "aws_s3_bucket_ownership_controls" "akila_s3_static_bucket" {
28   bucket = aws_s3_bucket.akila_s3_static_bucket.id
29   rule {
30     object_ownership = "BucketOwnerPreferred"
```

main.tf s3terraform × ec2.tf main.tf terraformEC2

s3terraform > main.tf > resource "aws_s3_object" "object" > source

```
26
27  resource "aws_s3_bucket_ownership_controls" "akila_s3_static_bucket" {
28    bucket = aws_s3_bucket.akila_s3_static_bucket.id
29    rule {
30      object_ownership = "BucketOwnerPreferred"
31    }
32  }
33
34  resource "aws_s3_bucket_acl" "akila_s3_static_bucket" {
35    bucket = aws_s3_bucket.akila_s3_static_bucket.id
36
37    acl = "public-read"
38    depends_on = [
39      aws_s3_bucket_ownership_controls.akila_s3_static_bucket,
40      aws_s3_bucket_public_access_block.akila_s3_static_bucket
41    ]
42  }
43
44  resource "aws_s3_bucket_policy" "akila_s3_static_bucket" {
45    bucket = aws_s3_bucket.akila_s3_static_bucket.id
46
47    policy = jsonencode({
48      Version = "2012-10-17"
49      Statement = [
50        {
51          Sid       = "PublicReadGetObject"
52          Effect    = "Allow"
53          Principal = "*"
54          Action    = ["s3:GetObject","s3:PutObject"]
55          Resource = [
```

Ln 70, Col 31 Spaces: 2 UTF-8 LF {} Terraform Go Live Analysis Tools Missing Go Update Available

In line 67: upload the object once the specific bucket is created

main.tf s3terraform × ec2.tf main.tf terraformEC2

s3terraform > main.tf > resource "aws_s3_object" "object" > source

```
47    policy = jsonencode({
48      Version = "2012-10-17"
49      Statement = [
50        {
51          Sid       = "PublicReadGetObject"
52          Effect    = "Allow"
53          Principal = "*"
54          Action    = ["s3:GetObject","s3:PutObject"]
55          Resource = [
56            aws_s3_bucket.akila_s3_static_bucket.arn,
57            "${aws_s3_bucket.akila_s3_static_bucket.arn}/*",
58          ]
59        },
60      ]
61    })
62
63    depends_on = [
64      aws_s3_bucket_public_access_block.akila_s3_static_bucket
65    ]
66  }
67  resource "aws_s3_object" "object" {
68    bucket = "akila-s3-static-bucket"
69    key = "index.html"
70    source = "./html/index.html"
71    content_type = "text/html"
72  }
73
74
```

<u>Dynamic web hosting in EC2 using terraform:</u>

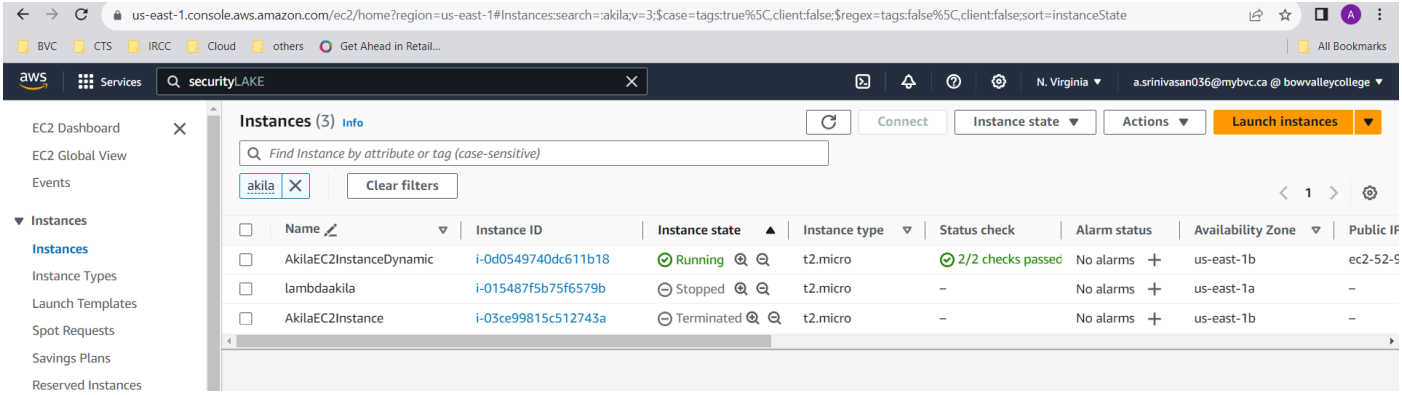
Now repeat the same for the same thing for like “terraform init” -> “terraform plan” -> “terraform apply”

```
C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2>cd ec2terraform

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\ec2terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.24.0...
- Installed hashicorp/aws v5.24.0 (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.

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\ec2terraform>
```

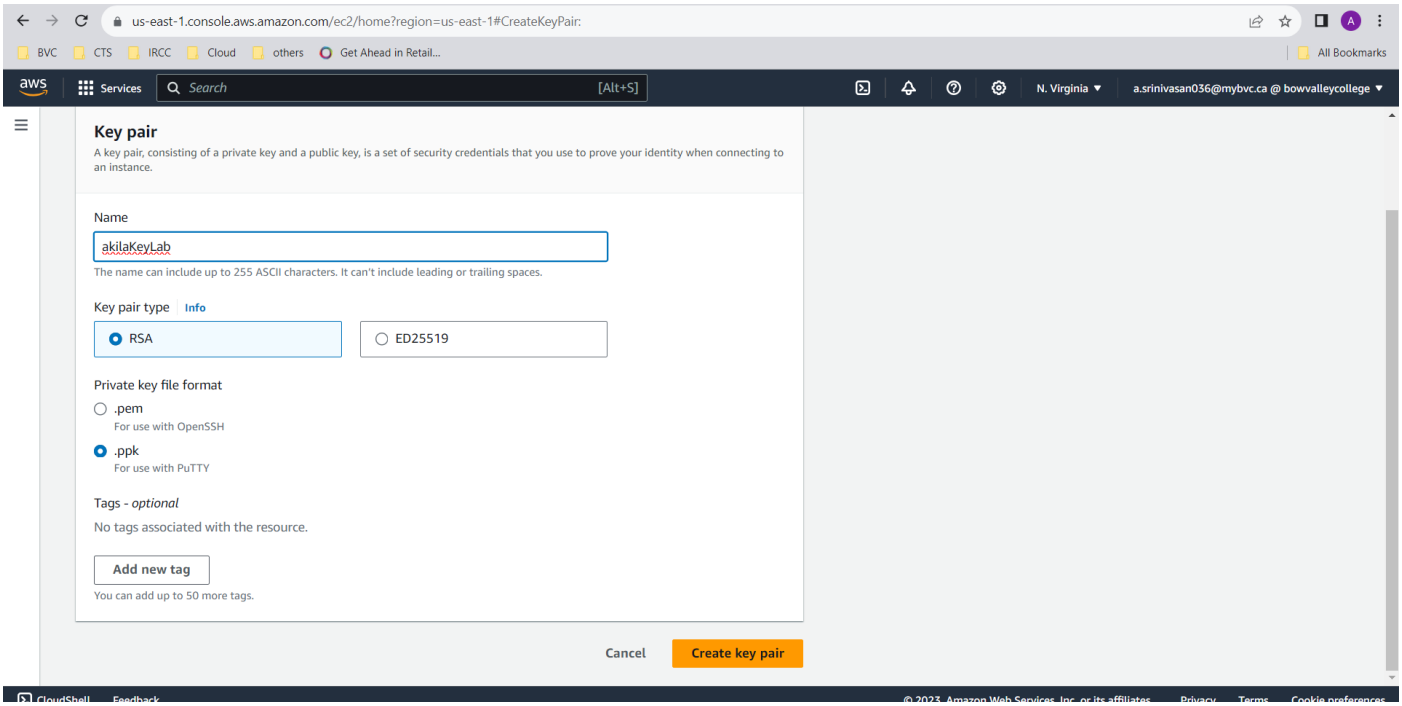
```
Command Prompt

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\ec2terraform>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:

  # aws_instance.web_server will be created
  + resource "aws_instance" "web_server" {
      + ami                                  = "ami-05c13eab67c5d8861"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
```

```
Command Prompt

              + to_port          = 80
            },
          + {
              + cidr_blocks      = []
              + description      = ""
              + from_port        = 22
              + ipv6_cidr_blocks = []
              + prefix_list_ids  = []
              + protocol         = "tcp"
              + security_groups  = []
              + self             = false
              + to_port          = 22
            },
        ]
      + name                   = (known after apply)
      + name_prefix            = "web-server-Akila-sg"
      + owner_id               = (known after apply)
      + revoke_rules_on_delete = false
      + tags_all               = (known after apply)
      + vpc_id                 = (known after apply)
    }

Plan: 2 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

aws_security_group.web_server_security_group: Creating...
aws_instance.web_server: Creating...
aws_security_group.web_server_security_group: Creation complete after 4s [id=sg-031952ea45f12beda]
aws_instance.web_server: Still creating... [10s elapsed]
aws_instance.web_server: Still creating... [20s elapsed]
aws_instance.web_server: Creation complete after 23s [id=i-0fd66c34ae0cde982]

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

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\ec2terraform>
```

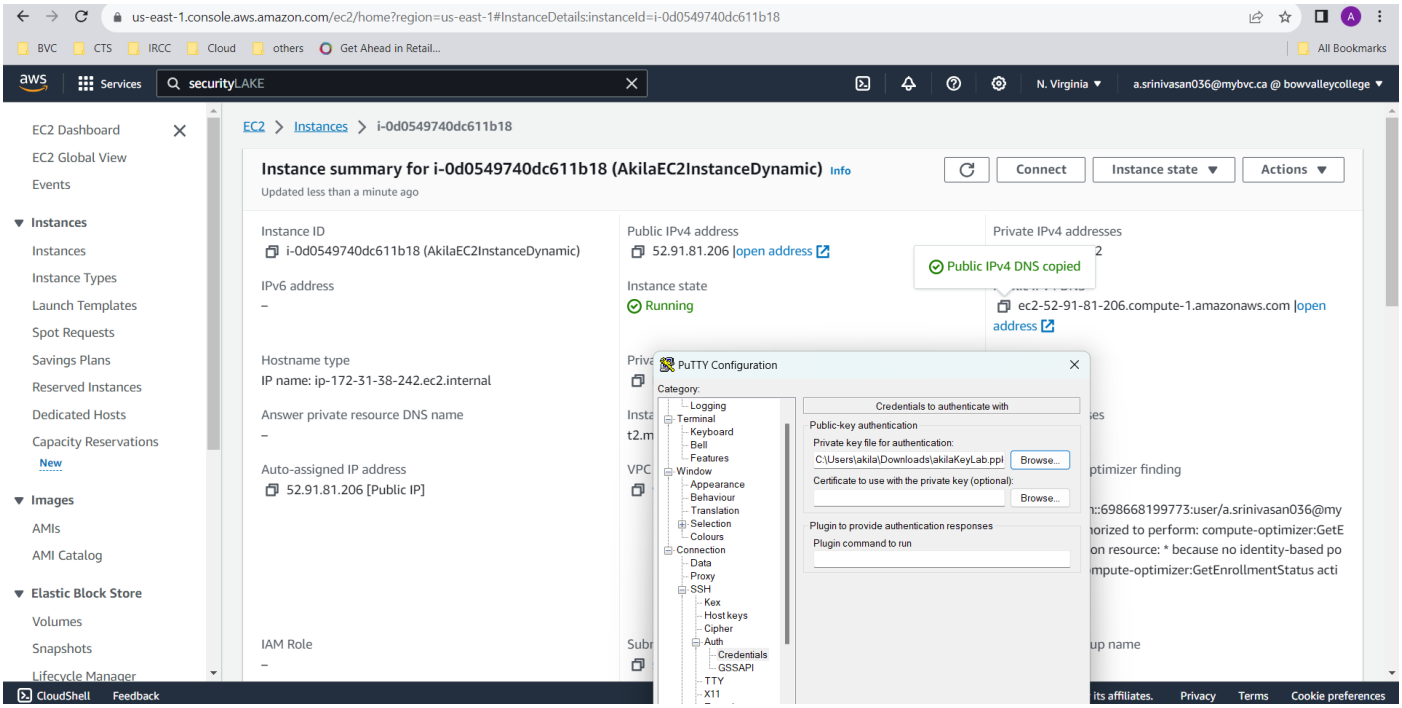
Now we have to create the key pairs for the ec2 which we have created. For that click the ec2 which is created.



Search for key pairs and click on create new key pair. Give the name for your key pair and click cerate button.



Now configure the putty to connect to the ec2 instance which we have created now.



Login as “ec2-user”

```
ec2-user@ip-172-31-40-115:~
login as: ec2-user
Authenticating with public key "akilaKeyLab"
       #_
       ~\_  ####_        Amazon Linux 2023
      ~~  \_#####\
      ~~     \###|
      ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
       ~~       V~' '->
        ~~~         /
          ~~._.   _/
             _/ _/
           _/m/'
Last login: Mon Nov  6 01:17:41 2023 from 50.99.196.75
[ec2-user@ip-172-31-40-115 ~]$
```

Then give “sudo yum update” command for the latest updates

Later install httpd (apache) and golang using the following commands

Sudo yum install httpd ---------- apache

Sudo yum install golang install ---------- golang

```
ec2-user@ip-172-31-40-115:~
login as: ec2-user
Authenticating with public key "akilaKeyLab"
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Mon Nov  6 01:17:41 2023 from 50.99.196.75
[ec2-user@ip-172-31-40-115 ~]$ sudo yum update
Last metadata expiration check: 5:37:28 ago on Sun Nov  5 20:12:56 2023.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-40-115 ~]$ sudo yum install httpd
Last metadata expiration check: 5:37:39 ago on Sun Nov  5 20:12:56 2023.
Package httpd-2.4.56-1.amzn2023.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-40-115 ~]$ sudo yum install golang
Last metadata expiration check: 5:38:26 ago on Sun Nov  5 20:12:56 2023.
Dependencies resolved.
================================================================================================================================================
 Package                                    Architecture                    Version                                         Repository                                  Size
================================================================================================================================================
Installing:
 golang                                     x86_64                          1.20.10-1.amzn2023.0.1                          amazonlinux                                599 k
Installing dependencies:
 annobin-docs                               noarch                          10.93-1.amzn2023.0.1                            amazonlinux                                 92 k
 annobin-plugin-gcc                         x86_64                          10.93-1.amzn2023.0.1                            amazonlinux                                887 k
 cpp                                        x86_64                          11.4.1-2.amzn2023.0.2                           amazonlinux                                 10 M
 emacs-filesystem                           noarch                          1:28.2-3.amzn2023.0.6                           amazonlinux                                9.5 k
 gc                                         x86_64                          8.0.4-5.amzn2023.0.2                            amazonlinux                                105 k
 gcc                                        x86_64                          11.4.1-2.amzn2023.0.2                           amazonlinux                                 32 M
 git-core                                   x86_64                          2.40.1-1.amzn2023.0.1                           amazonlinux                                4.3 M
 git-core-doc                               noarch                          2.40.1-1.amzn2023.0.1                           amazonlinux                                2.6 M
 glibc-devel                                x86_64                          2.34-52.amzn2023.0.7                            amazonlinux                                 45 k
 glibc-headers-x86                          noarch                          2.34-52.amzn2023.0.7                            amazonlinux                                446 k
 golang-bin                                 x86_64                          1.20.10-1.amzn2023.0.1                          amazonlinux                                 58 M
 golang-src                                 noarch                          1.20.10-1.amzn2023.0.1                          amazonlinux                                 11 M
 guile22                                    x86_64                          2.2.7-2.amzn2023.0.3                            amazonlinux                                6.4 M
 kernel-headers                             x86_64                          6.1.59-84.139.amzn2023                          amazonlinux                                1.4 M
 libmpc                                     x86_64                          1.2.1-2.amzn2023.0.2                            amazonlinux                                 62 k
 libserf                                    x86_64                          1.3.9-23.amzn2023.0.3                           amazonlinux                                 58 k
```

```
ec2-user@ip-172-31-40-115:~
  Installing       : git-2.40.1-1.amzn2023.0.1.x86_64                                                                                   27/30
  Installing       : golang-1.20.10-1.amzn2023.0.1.x86_64                                                                               28/30
  Installing       : golang-bin-1.20.10-1.amzn2023.0.1.x86_64                                                                           29/30
  Running scriptlet: golang-bin-1.20.10-1.amzn2023.0.1.x86_64                                                                           29/30
  Installing       : annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64                                                                     30/30
  Running scriptlet: annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64                                                                     30/30
  Verifying        : git-2.40.1-1.amzn2023.0.1.x86_64                                                                                    1/30
  Verifying        : make-1:4.3-5.amzn2023.0.2.x86_64                                                                                    2/30
  Verifying        : golang-bin-1.20.10-1.amzn2023.0.1.x86_64                                                                            3/30
  Verifying        : git-core-2.40.1-1.amzn2023.0.1.x86_64                                                                               4/30
  Verifying        : perl-lib-0.65-477.amzn2023.0.5.x86_64                                                                               5/30
  Verifying        : utf8proc-2.6.1-2.amzn2023.0.2.x86_64                                                                                6/30
  Verifying        : libmpc-1.2.1-2.amzn2023.0.2.x86_64                                                                                  7/30
  Verifying        : gc-8.0.4-5.amzn2023.0.2.x86_64                                                                                      8/30
  Verifying        : libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64                                                                            9/30
  Verifying        : kernel-headers-6.1.59-84.139.amzn2023.x86_64                                                                       10/30
  Verifying        : subversion-1.14.2-5.amzn2023.0.3.x86_64                                                                            11/30
  Verifying        : subversion-libs-1.14.2-5.amzn2023.0.3.x86_64                                                                       12/30
  Verifying        : annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64                                                                     13/30
  Verifying        : golang-1.20.10-1.amzn2023.0.1.x86_64                                                                               14/30
  Verifying        : gcc-11.4.1-2.amzn2023.0.2.x86_64                                                                                   15/30
  Verifying        : libxcrypt-devel-4.4.33-7.amzn2023.x86_64                                                                           16/30
  Verifying        : glibc-devel-2.34-52.amzn2023.0.7.x86_64                                                                            17/30
  Verifying        : mercurial-5.7.1-1.amzn2023.0.3.x86_64                                                                              18/30
  Verifying        : perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64                                                                        19/30
  Verifying        : guile22-2.2.7-2.amzn2023.0.3.x86_64                                                                                20/30
  Verifying        : cpp-11.4.1-2.amzn2023.0.2.x86_64                                                                                   21/30
  Verifying        : libserf-1.3.9-23.amzn2023.0.3.x86_64                                                                               22/30
  Verifying        : perl-Error-1:0.17029-5.amzn2023.0.2.noarch                                                                         23/30
  Verifying        : git-core-doc-2.40.1-1.amzn2023.0.1.noarch                                                                          24/30
  Verifying        : glibc-headers-x86-2.34-52.amzn2023.0.7.noarch                                                                      25/30
  Verifying        : golang-src-1.20.10-1.amzn2023.0.1.noarch                                                                           26/30
  Verifying        : perl-Git-2.40.1-1.amzn2023.0.1.noarch                                                                              27/30
  Verifying        : annobin-docs-10.93-1.amzn2023.0.1.noarch                                                                           28/30
  Verifying        : perl-File-Find-1.37-477.amzn2023.0.5.noarch                                                                        29/30
  Verifying        : emacs-filesystem-1:28.2-3.amzn2023.0.6.noarch                                                                      30/30

Installed:
  annobin-docs-10.93-1.amzn2023.0.1.noarch      annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64  cpp-11.4.1-2.amzn2023.0.2.x86_64              emacs-filesystem-1:28.2-3.amzn2023.0.6.noarch
  gc-8.0.4-5.amzn2023.0.2.x86_64                gcc-11.4.1-2.amzn2023.0.2.x86_64                git-2.40.1-1.amzn2023.0.1.x86_64              git-core-2.40.1-1.amzn2023.0.1.x86_64
  git-core-doc-2.40.1-1.amzn2023.0.1.noarch     glibc-devel-2.34-52.amzn2023.0.7.x86_64         glibc-headers-x86-2.34-52.amzn2023.0.7.noarch golang-1.20.10-1.amzn2023.0.1.x86_64
  golang-bin-1.20.10-1.amzn2023.0.1.x86_64      golang-src-1.20.10-1.amzn2023.0.1.noarch        guile22-2.2.7-2.amzn2023.0.3.x86_64           kernel-headers-6.1.59-84.139.amzn2023.x86_64
  libmpc-1.2.1-2.amzn2023.0.2.x86_64            libserf-1.3.9-23.amzn2023.0.3.x86_64            libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64      libxcrypt-devel-4.4.33-7.amzn2023.x86_64
  make-1:4.3-5.amzn2023.0.2.x86_64              mercurial-5.7.1-1.amzn2023.0.3.x86_64           perl-Error-1:0.17029-5.amzn2023.0.2.noarch    perl-File-Find-1.37-477.amzn2023.0.5.noarch
  perl-Git-2.40.1-1.amzn2023.0.1.noarch         perl-TermReadKey-2.38-9.amzn2023.0.2.x86_64     perl-lib-0.65-477.amzn2023.0.5.x86_64         subversion-1.14.2-5.amzn2023.0.3.x86_64
  subversion-libs-1.14.2-5.amzn2023.0.3.x86_64  utf8proc-2.6.1-2.amzn2023.0.2.x86_64

Complete!
[ec2-user@ip-172-31-40-115 ~]$ sudo yum install httpd
```

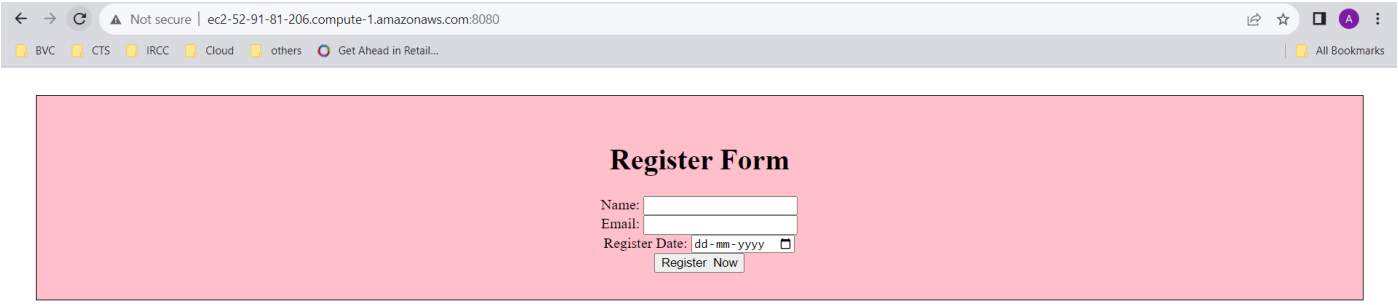
Now start and enable the apache server

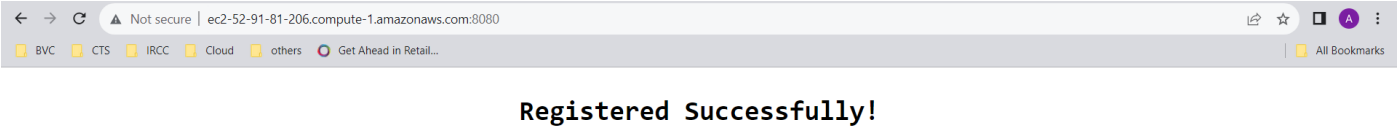
```
[ec2-user@ip-172-31-40-115 ~]$ sudo yum install httpd
Last metadata expiration check: 5:40:48 ago on Sun Nov  5 20:12:56 2023.
Package httpd-2.4.56-1.amzn2023.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-40-115 ~]$ Sudo systemctl start httpd
-bash: Sudo: command not found
[ec2-user@ip-172-31-40-115 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-40-115 ~]$ sudo systemctl enable httpd
[ec2-user@ip-172-31-40-115 ~]$
```

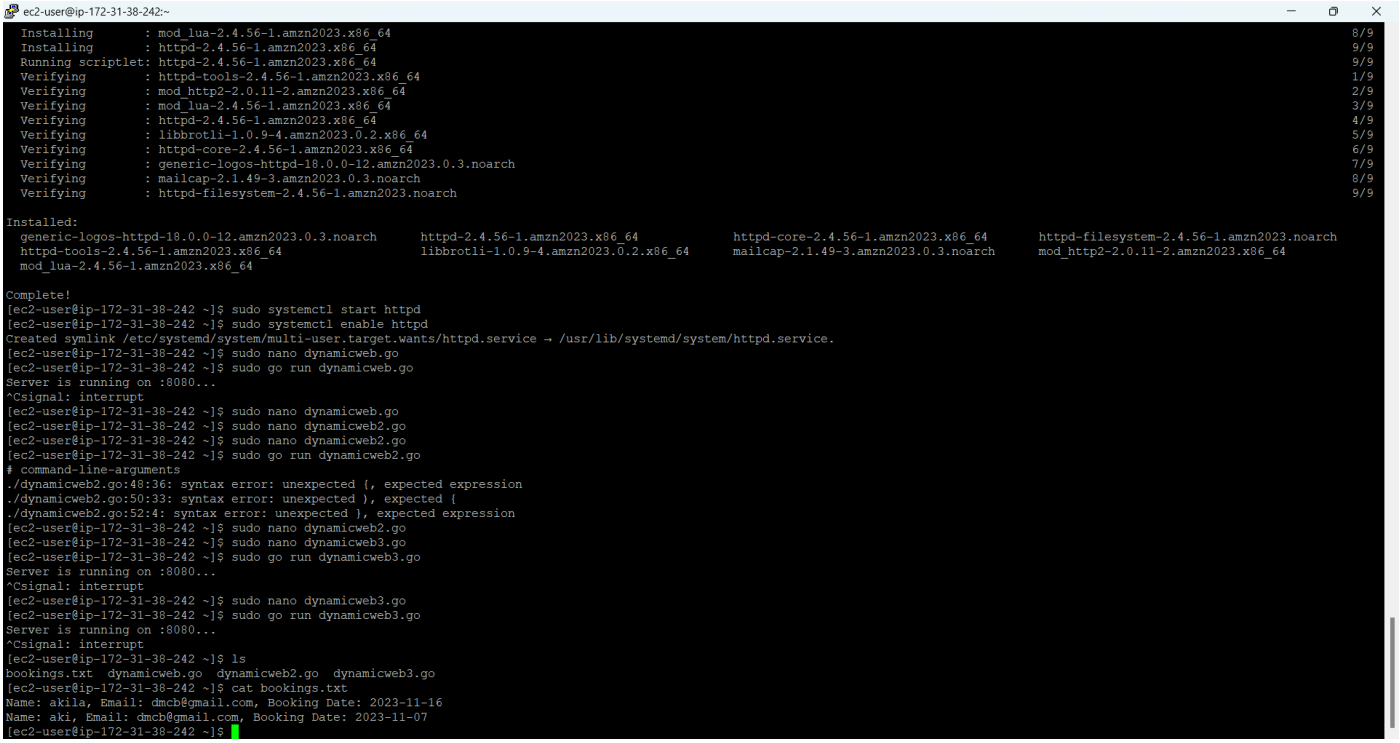
In this I have chosen golang code to run for that I have created a file using the cmd “sudo nano file.name.go”

```
ec2-user@ip-172-31-38-242:~
  Installing       : httpd-filesystem-2.4.56-1.amzn2023.noarch                                                    2/9
  Installing       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                                            3/9
  Installing       : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                                                        4/9
  Installing       : httpd-tools-2.4.56-1.amzn2023.x86_64                                                         5/9
  Installing       : httpd-core-2.4.56-1.amzn2023.x86_64                                                          6/9
  Installing       : mod_http2-2.0.11-2.amzn2023.x86_64                                                           7/9
  Installing       : mod_lua-2.4.56-1.amzn2023.x86_64                                                             8/9
  Installing       : httpd-2.4.56-1.amzn2023.x86_64                                                               9/9
  Running scriptlet: httpd-2.4.56-1.amzn2023.x86_64                                                               9/9
  Verifying        : httpd-tools-2.4.56-1.amzn2023.x86_64                                                         1/9
  Verifying        : mod_http2-2.0.11-2.amzn2023.x86_64                                                           2/9
  Verifying        : mod_lua-2.4.56-1.amzn2023.x86_64                                                             3/9
  Verifying        : httpd-2.4.56-1.amzn2023.x86_64                                                               4/9
  Verifying        : libbrotli-1.0.9-4.amzn2023.0.2.x86_64                                                        5/9
  Verifying        : httpd-core-2.4.56-1.amzn2023.x86_64                                                          6/9
  Verifying        : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch                                            7/9
  Verifying        : mailcap-2.1.49-3.amzn2023.0.3.noarch                                                         8/9
  Verifying        : httpd-filesystem-2.4.56-1.amzn2023.noarch                                                    9/9

Installed:
  generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch      httpd-2.4.56-1.amzn2023.x86_64            httpd-core-2.4.56-1.amzn2023.x86_64       httpd-filesystem-2.4.56-1.amzn2023.noarch
  httpd-tools-2.4.56-1.amzn2023.x86_64                   libbrotli-1.0.9-4.amzn2023.0.2.x86_64     mailcap-2.1.49-3.amzn2023.0.3.noarch      mod_http2-2.0.11-2.amzn2023.x86_64
  mod_lua-2.4.56-1.amzn2023.x86_64

Complete!
[ec2-user@ip-172-31-38-242 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-38-242 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-38-242 ~]$ sudo nano dynamicweb.go
[ec2-user@ip-172-31-38-242 ~]$ sudo go run dynamicweb.go
Server is running on :8080...
^Csignal: interrupt
[ec2-user@ip-172-31-38-242 ~]$ sudo nano dynamicweb.go
[ec2-user@ip-172-31-38-242 ~]$ sudo nano dynamicweb2.go
[ec2-user@ip-172-31-38-242 ~]$ sudo nano dynamicweb2.go
[ec2-user@ip-172-31-38-242 ~]$ sudo go run dynamicweb2.go
# command-line-arguments
./dynamicweb2.go:48:36: syntax error: unexpected {, expected expression
./dynamicweb2.go:50:33: syntax error: unexpected ), expected {
./dynamicweb2.go:52:4: syntax error: unexpected }, expected expression
[ec2-user@ip-172-31-38-242 ~]$ sudo nano dynamicweb2.go
[ec2-user@ip-172-31-38-242 ~]$ sudo nano dynamicweb3.go
[ec2-user@ip-172-31-38-242 ~]$ sudo go run dynamicweb3.go
Server is running on :8080...
^Csignal: interrupt
[ec2-user@ip-172-31-38-242 ~]$ sudo nano dynamicweb3.go
[ec2-user@ip-172-31-38-242 ~]$ sudo go run dynamicweb3.go
Server is running on :8080...
```

This is the output screen of executing our golang program.







After these we have to delete the ec2 by giving cmd “terraform destroy”

```
Command Prompt
                ]
              - description      = ""
              - from_port        = 80
              - ipv6_cidr_blocks = []
              - prefix_list_ids  = []
              - protocol         = "tcp"
              - security_groups  = []
              - self             = false
              - to_port          = 80
            },
        ] -> null
      - name                   = "web-server-Akila-sg20231106024604732500000001" -> null
      - name_prefix            = "web-server-Akila-sg" -> null
      - owner_id               = "698668199773" -> null
      - revoke_rules_on_delete = false -> null
      - tags                   = {} -> null
      - tags_all               = {} -> null
      - vpc_id                 = "vpc-551dcc28" -> null
    }

Plan: 0 to add, 0 to change, 2 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

aws_security_group.web_server_security_group: Destroying... [id=sg-07256e19f60077b8f]
aws_instance.web_server: Destroying... [id=i-0d0549740dc611b18]
aws_security_group.web_server_security_group: Destruction complete after 1s
aws_instance.web_server: Still destroying... [id=i-0d0549740dc611b18, 10s elapsed]
aws_instance.web_server: Still destroying... [id=i-0d0549740dc611b18, 20s elapsed]
aws_instance.web_server: Still destroying... [id=i-0d0549740dc611b18, 30s elapsed]
aws_instance.web_server: Still destroying... [id=i-0d0549740dc611b18, 40s elapsed]
aws_instance.web_server: Destruction complete after 40s

Destroy complete! Resources: 2 destroyed.

C:\Users\akila\Desktop\BVC\terraform\lab2-s3 , ec2\terraformEC2>
```

In Provider we have to mention the region

Line 9: create the instance with linux os and type t2.micro. I have already created the key pair which I have mentioned here.

Tag name is the name which u see in the instance list.

Line 19: create the security key for the inbound and outbound rules.

```
main.tf s3terraform    main.tf terraformEC2
terraformEC2 > main.tf > ...
1   # hcl
2   provider "aws" {
3     region = "us-east-1"
4   }
5   # resource "aws_key_pair" "my_key_pair" {
6   #   key_name   = "akilakey"   # Change to your desired key pair name
7   #   public_key = file("~/.ssh/akilakey.pub")  # Provide the path to your public key file
8   # }
9   resource "aws_instance" "web_server" {
10    ami           = "ami-05c13eab67c5d8861"
11    instance_type = "t2.micro"
12    key_name      = "akilaKeyLab"
13    tags = {
14      Name = "AkilaEC2InstanceDynamic"
15    }
16
17  }
18
19  resource "aws_security_group" "web_server_security_group" {
20    name_prefix = "web-server-Akila-sg"
21
22
23    ingress {
24      from_port   = 80
25      to_port     = 80
26      protocol    = "tcp"
27      cidr_blocks = ["0.0.0.0/0"]
28    }
29
```

main.tf s3terraform | main.tf terraformEC2

terraformEC2 > main.tf > ...

```
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"

  }
  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "ssh"

  }
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

Reference:

www.google.com , www.youtube.com, terraform official website (Hashicorp)