Prajit Venkatachalam

Hosting a Static Website in Amazon S3 using Terraform

Prajit Venkatachalam
https://www.linkedin.com/in/prajit-venkatachalam/

Prajit Venkatachalam

What is Terraform

Terraform is a Infrastructure as Code (IaC) tool developed by HashiCorp, which allows to define, provision, and manage cloud infrastructure resources like EC2 instance, S3, VPC and others. It is useful for automating the setup of networking, databases, compute resources and other cloud services.

How did I use terraform in this project

I used Terraform in this project to create an S3 bucket in Amazon S3 and hosted a static website using Infrastructure as Code (IaC). I also destroyed the entire infrastructure using the command terraform destroy and then rebuilt it from scratch with just one command, terraform apply, since all the code was ready. I verified the automatic setup of the infrastructure.

One thing I did not expect in this project…

One thing I did not expect in this project was the simplicity of using Terraform, which eliminated the need for manually setting up the infrastructure.

This Project took me…

This project took me 1.5 hours to complete, including downloading and installing Terraform, writing the code, and preparing the report.

Prajit Venkatachalam

Creating S3 bucket

Setting up Terraform

1. I downloaded and installed terraform, and copied the path of the terraform to the environment variables. (double click- path- new- paste the path of the terraform where installed.
2. I opened terminal from my local computer and created a folder "**terfrms3buckprajit**" by typing the command: **mkdir mys3staticwebsite,** and then went into the directory of the folder using the command: **cd mys3staticwebsite**
3. Now, I opened the **terfrms3buckprajit** folder in Visual Studio Code using the command **Code** . (make sure you copied the path of vscode file)

What is provider in terraform

A provider defines the resources we create in AWS and instructs terraform on how to interact with those resources. It is essential to configure the provider with proper credentials before using it. In my case, I configured the provider with the below code for the terraform to connect to AWS resources.

Note: You can refer to the official Terraform documentation for code examples and to clarify any doubts. (https://registry.terraform.io/providers/hashicorp/aws/latest/docs)

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.70.0"
    }
  }
}

provider "aws" {
  # Configuration options
  region = "ap-southeast-2"
}
```

Make sure you update the correct region of your AWS account.

Prajit Venkatachalam

4. Next, I opened a new terminal and ran the command terraform init to install all the required files and dependencies for the provider to connect and function properly.

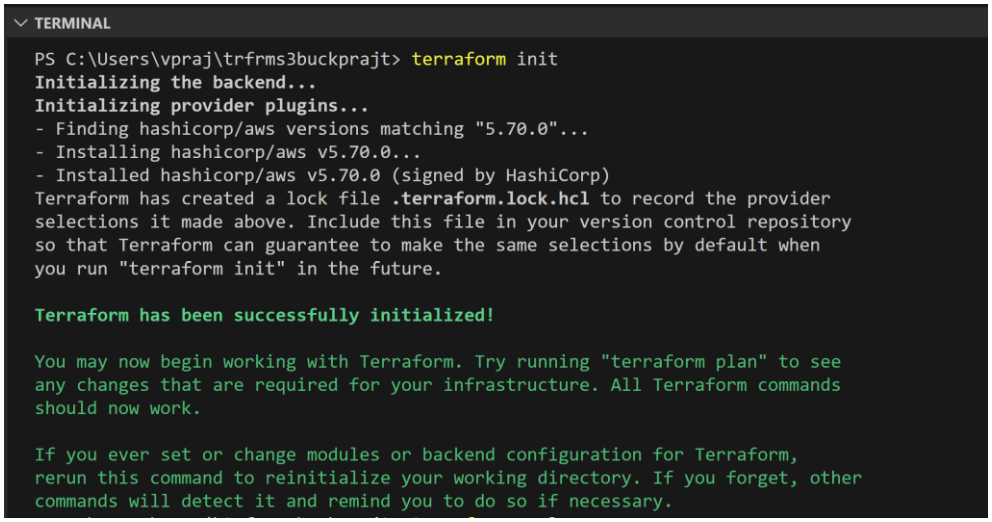


Figure 1: Terraform initialized

5. I opened a new file named variables.tf to declare the bucket's name separately, ensuring it was unique.

```
variable "bucketname" {
  default = "terfrms3buckprajit"
}
```

6. Then, I opened a new file named main.tf in the project folder in vscode to write the code for creating an S3 bucket. I ensured that the name of the bucket was globally unique.

```
resource "aws_s3_bucket" "mybucket" {
  bucket = var.bucketname
}
```

I created a variable in another file “variables.tf” to define the bucket name in “main.tf” file.

```
variable "bucketname" {
  default = "terfrms3buckprajit"
}
```

7. Next, I ran the command **terraform plan** to preview the changes Terraform intended to make—in this case, creating the S3 bucket.

```
PS C:\Users\vpraj\trfrms3buckprajt> terraform plan

Terraform used the selected providers to generate the following ex
following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.mybucket will be created
  + resource "aws_s3_bucket" "mybucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "terfrms3buckprajit"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags_all                    = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + cors_rule (known after apply)

      + grant (known after apply)
```

```
      + server_side_encryption_configuration (known after apply)

      + versioning (known after apply)

      + website (known after apply)
    }

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

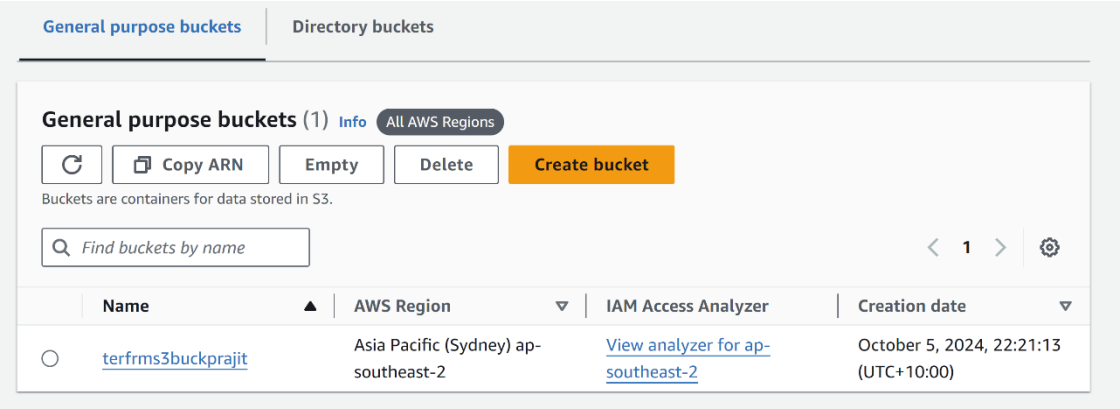
Figure 2: Terraform plan to execute the chnages

8. Then, I ran the command terraform apply to execute the plan, and it successfully created the bucket in the AWS console.

```
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_bucket.mybucket: Creating...
aws_s3_bucket.mybucket: Creation complete after 2s [id=terfrms3buckprajit]

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

Figure 3: Terraform creating the S3 bucket.

Prajit Venkatachalam



9. Once the bucket was created, I set the bucket ownership to ensure that all content inside the bucket belongs to the owner.

```
resource "aws_s3_bucket_ownership_controls" "mybucket" {
  bucket = aws_s3_bucket.mybucket.bucket

  rule {
    object_ownership = "BucketOwnerPreferred"
  }
}
```

10. Next, I configured the bucket to be publicly accessible, ensuring that the website could be accessed by anyone from anywhere on the internet.

```
resource "aws_s3_bucket_public_access_block" "mybucket" {
  bucket                  = aws_s3_bucket.mybucket.bucket
  block_public_acls       = false
  block_public_policy     = false
  ignore_public_acls      = false
  restrict_public_buckets = false
}
```

11. To make it completely public, I had to add ACLs (Access Control Lists).

```
resource "aws_s3_bucket_acl" "mybucket" {
  depends_on = [
    aws_s3_bucket_ownership_controls.mybucket,
    aws_s3_bucket_public_access_block.mybucket,
```

```
  ]

  bucket = aws_s3_bucket.mybucket.bucket
  acl    = "public-read"
}
```

12. The next step is enabling static website hosting for the bucket, which requires website configuration. To configure it, we need an index.html and an error.html. Before that, I uploaded the **index.html, error.html, and a profile.png** (picture) as objects in the S3 bucket.

    Note: I uploaded my resume and asked ChatGPT to write the index.html and error.html code for my website to showcase my portfolio. For profile.png, I uploaded the picture manually in the project folder.

```
resource "aws_s3_object" "index" {
  bucket      = aws_s3_bucket.mybucket.id
  key         = "index.html"
  source      = "index.html"
  acl         = "public-read"
  content_type = "text/html"
}

resource "aws_s3_object" "error" {
  bucket      = aws_s3_bucket.mybucket.id
  key         = "error.html"
  source      = "error.html"
  acl         = "public-read"
  content_type = "text/html"
}

resource "aws_s3_object" "profile" {
  bucket      = aws_s3_bucket.mybucket.id
  key         = "profile.png"
  source      = "profile.png"
  acl         = "public-read"
  content_type = "image/png"
}
```

Prajit Venkatachalam

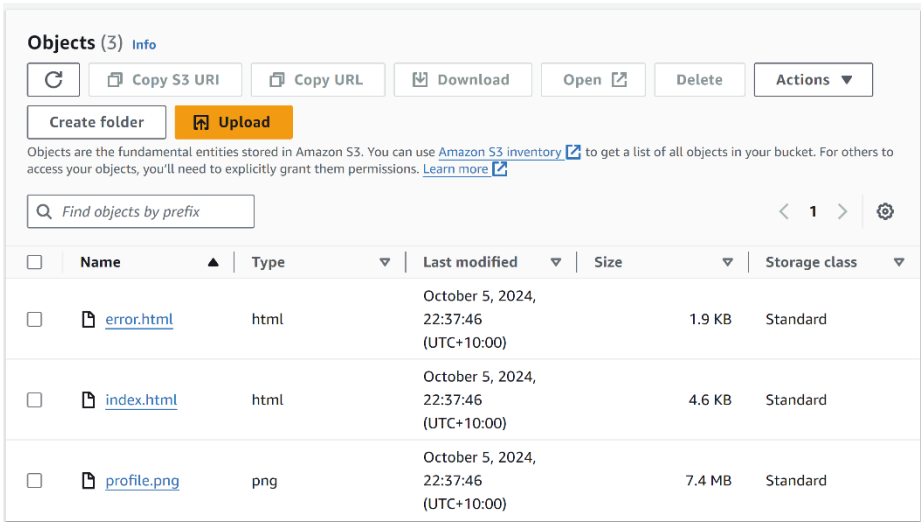


Figure 4: Objects uploaded in S3 bucket

13. I then opened the URL of my S3 bucket to check if my website pages were loading correctly to display my portfolio.

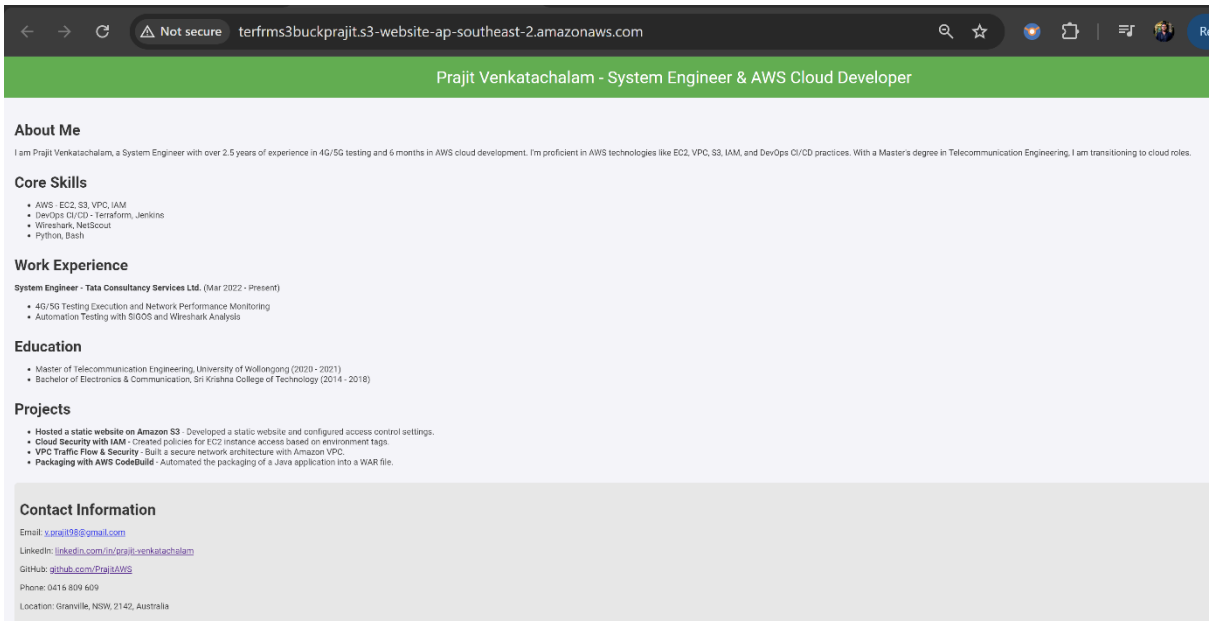
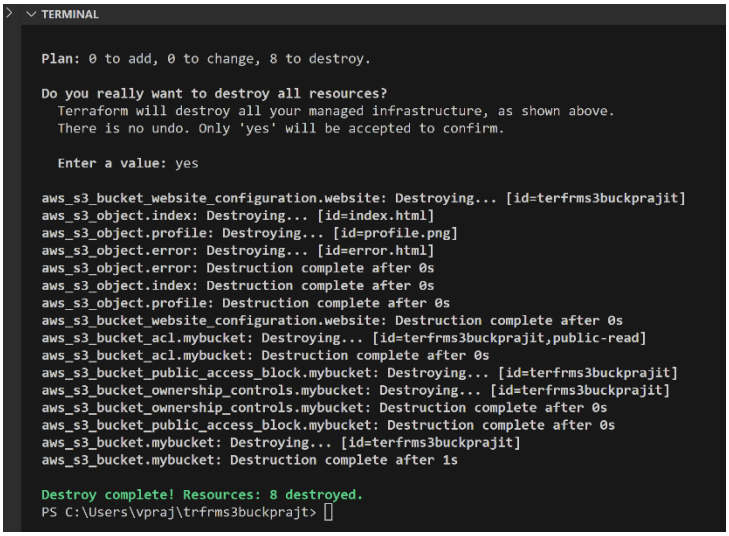


Figure 5: website showing my portfolio

Prajit Venkatachalam

14. I then re-edited the index.html file to include the profile.png that I uploaded to S3. After that, I destroyed all the resources to test if I could start from scratch and host the website in just a few seconds using the command terraform destroy.



15. I then used the command terraform apply again, with all the files prepared, to check if Terraform would automatically build the entire infrastructure based on the code.