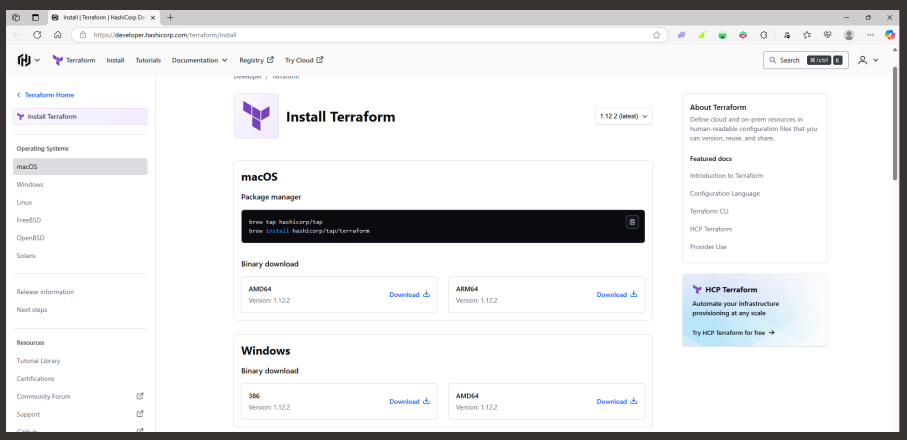
nextwork.org

Create S3 Buckets with Terraform

ME Melvin J Bonner



ME
Melvin J Bonner
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to use Terraform to automate the deployment of an S3 bucket. The goal is to become more familiar with Terraform.

Tools and concepts

Services I used were Terraform and Amazon S3. Key concepts I learned include infrastructure as code, resource blocks, Terraform configuration, terraform apply, and terraform init.

Project reflection

This project took me approximately 2 hours. The most challenging part was getting Terraform installed on my computer. It was most rewarding when the image appeared in my S3 bucket.

I did this project today to get some exposure to Terraform and get a better understanding of how it works. This project met my goals.

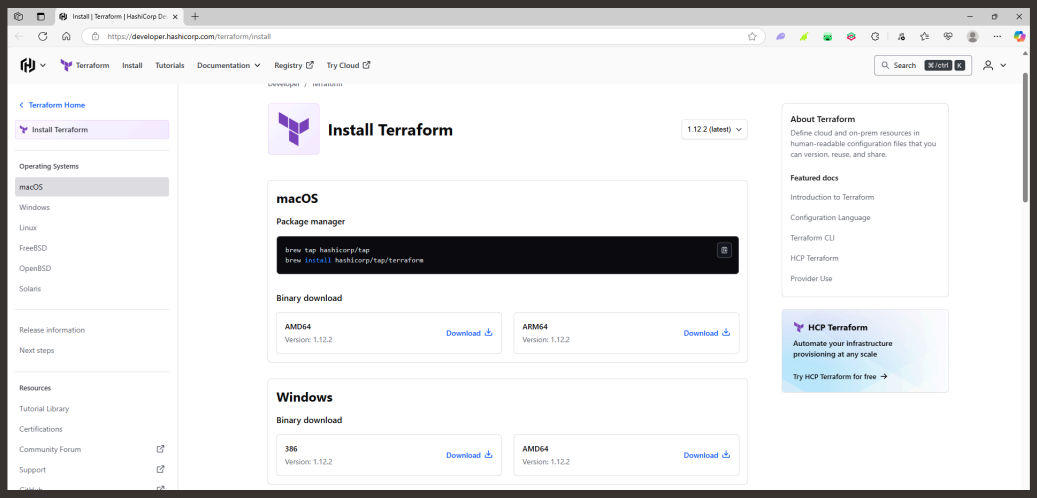
ME
Melvin J Bonner
NextWork Student

Introducing Terraform

Terraform is a tool that helps you build and manage your cloud infrastructure using code.

Terraform is one of the most popular tools used for infrastructure as code (IaC), which is a way to manage your IT infrastructure. Instead of manually setting up your resources one by one in the console, you're writing configuration files in code.

Terraform uses configuration files to to define and manage infrastructure; main.tf is is a central file in a Terraform project. It's where you write down what you want your infrastructure to look like, using Terraform's language



Configuration files

The configuration is structured in blocks. The advantage of doing this is help to organize the code better. Each block handles a specific piece of your setup, like a resource or a configuration, which makes it easier to read, manage, and adjust things separately without affecting the rest of your infrastructure.

My main.tf configuration has three blocks

The first block indicates that AWS is the provider. The second block provisions an S3 bucket. The third block manages the public access policies for the S3 bucket.

main.tf

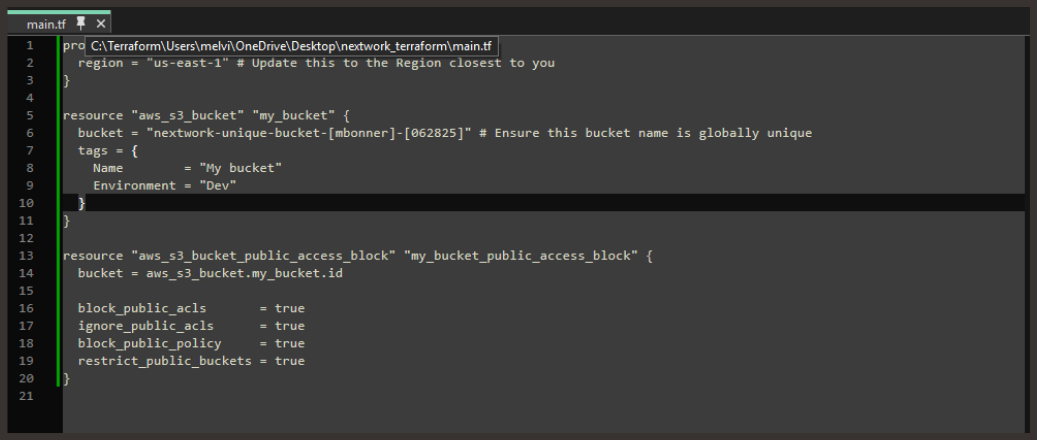
```
provider "aws" {
  region = "us-east-1" # Update this to the Region closest to you
}

resource "aws_s3_bucket" "my_bucket" {
  bucket = "nextwork-unique-bucket-[mbonner]-[062825]" # Ensure this bucket name is globally unique
}

resource "aws_s3_bucket_public_access_block" "my_bucket_public_access_block" {
  bucket = aws_s3_bucket.my_bucket.id

  block_public_acls       = true
  ignore_public_acls      = true
  block_public_policy     = true
  restrict_public_buckets = true
}
```

ME
Melvin J Bonner
NextWork Student

Customizing my S3 Bucket

For my project extension, I visited the official Terraform documentation to find setup instructions, descriptions of each available resource, best practice tips and examples, and parameters for customization. The documentation shows how to use Terraform to create and manage AWS S3 buckets. It covers how to configure these buckets, set their properties, and manage interactions with other AWS services.

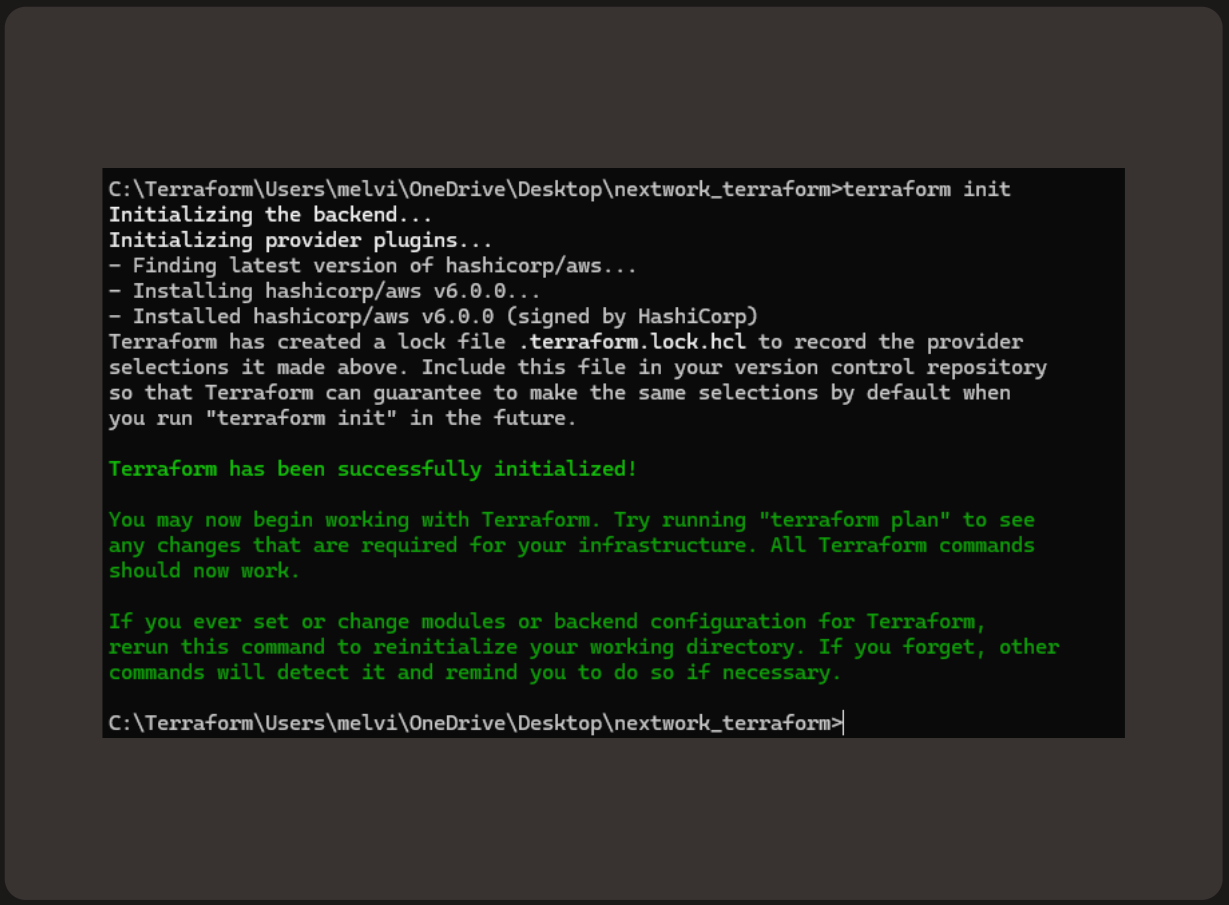
I chose to customize my bucket by adding a Dev tag. With a Dev tag, S3 only allows the Dev to access those resources. When I launch my bucket, I can verify my customization by checking both the AWS Console and Terraform outputs or state.


main.tf
C:\Terraform\Users\melvi\OneDrive\Desktop\nextwork_terraform\main.tf
pro
region = "us-east-1" # Update this to the Region closest to you
}
resource "aws_s3_bucket" "my_bucket" {
bucket = "nextwork-unique-bucket-[mbonner]-[062825]" # Ensure this bucket name is globally unique
tags = {
Name = "My bucket"
Environment = "Dev"
}
}
resource "aws_s3_bucket_public_access_block" "my_bucket_public_access_block" {
bucket = aws_s3_bucket.my_bucket.id
block_public_acls = true
ignore_public_acls = true
block_public_policy = true
restrict_public_buckets = true
}

Terraform commands

I ran terraform init to set up my project by initializing the backend, initialize the provider pluging, find the latest version of hashicorp, and create a lock file.

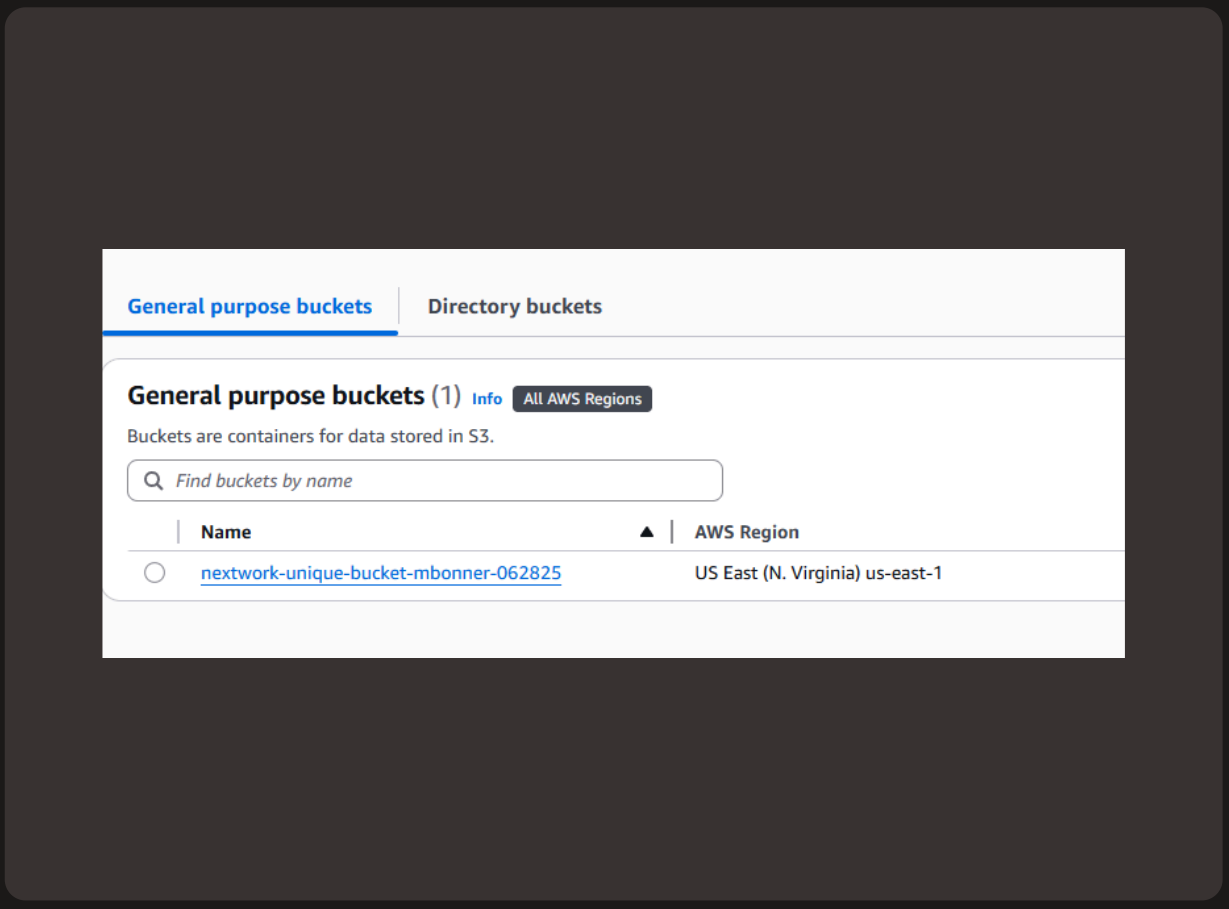
Next, I ran terraform plan to create an execution plan, showing what changes Terraform will make to the infrastructure on my configuration files.



Lanching the S3 Bucket

I ran terraform apply to apply the changes written in my Terraform configuration. Running terraform apply will affect my AWS account by adding the S3 bucket in my account.

The sequence of running terraform init, plan, and apply is crucial because I would have run into an error if not done in that order. Terraform init needed to be run first because it set my project up by downloading the necessary plugins and setting up the state line.



ME
Melvin J Bonner
NextWork Student
nextwork.org

Uploading an S3 Object

I created a new resource block to import an image to the S3 bucket.

We need to run terraform apply again because anytime you change Terraform configuration you have to run terraform apply. This will make sure the changes are correctly applied to your infrastructure.

To validate that I've updated my configuration successfully, I downloaded the image and compared it to the uploaded image.

main.tf

```
provider "aws" {
  region = "us-east-1" # Update this to the Region closest to you
}

resource "aws_s3_bucket" "my_bucket" {
  bucket = "nextwork-unique-bucket-mbonner-062825" # Ensure this bucket name is globally unique
  tags = {
    Name        = "My bucket"
    Environment = "Dev"
  }
}

resource "aws_s3_bucket_public_access_block" "my_bucket_public_access_block" {
  bucket = aws_s3_bucket.my_bucket.id

  block_public_acls       = true
  ignore_public_acls      = true
  block_public_policy     = true
  restrict_public_buckets = true
}

resource "aws_s3_object" "image" {
  bucket = aws_s3_bucket.my_bucket.id # Reference the bucket ID
  key    = "image.png" # Path in the bucket
  source = "image.png" # Local file path
}
```

nextwork.org

The place to learn & showcase your skills

Check out nextwork.org for more projects