**Setting-Static-Website-using-Terraform-and-AWS**

Automated DevOps project using Terraform and AWS to deploy a static website hosted on an S3 bucket.

**Implementation**

You can find the project on my github: https://github.com/lakshmir2023/Setting-Static-Website-using-Terraform-and-AWS

**Creating an AWS account**

If you don’t have an AWS account already, follow this official documentation for creating an AWS standalone account: <https://docs.aws.amazon.com/accounts/latest/reference/manage-acct-creating.html>

**Install Terraform**

To use Terraform we will need to install it. HashiCorp distributes Terraform as a binary package. We can also install Terraform using popular package managers. Install terraform in your machine by following: <https://developer.hashicorp.com/terraform/downloads>

**Install AWS CLI**

We will also need AWS CLI in this project. AWS CLI enables us to interact with AWS services using commands in our command-line shell.

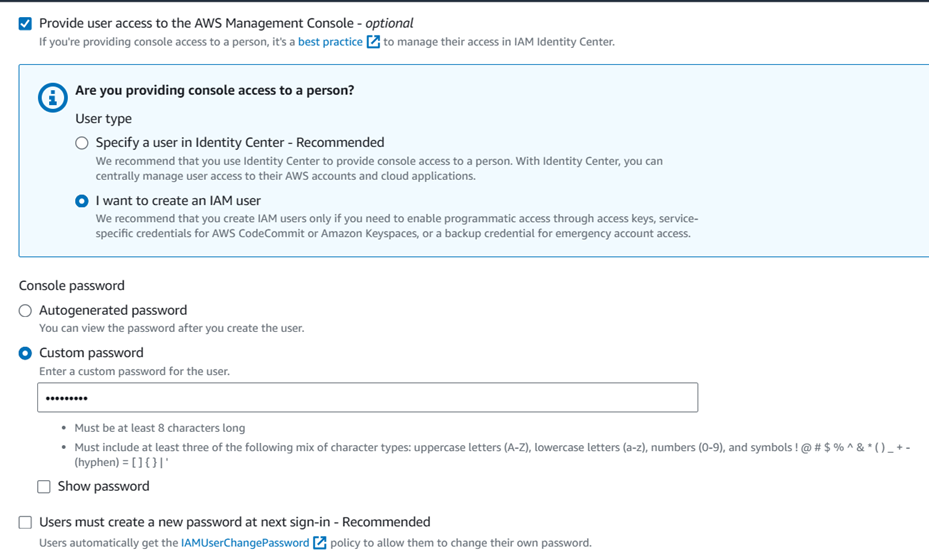
To install AWS CLI make use of official AWS documentation: <https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

**Connect your AWS account to Terraform**

Login to AWS using either your root user or IAM user.

**Creating an IAM user**

Once logged in, if you don’t have an IAM user, best to create one. Navigate to IAM service > Users > Create user > provide a user name > select Provide user access to the AWS Management Console option and then select rest of the options as below:



After this, click Next > Under Permissions options, choose Attach Policies directly and select Administrator Access permission policy for this user. After setting the policy, click Next and then create user. Copy the console sign-in URL and paste it in a new browser to sign in using the IAM user

**Creating the access key**

After you login using the IAM user> on the right side under the account info, click on Security credentials > it will take you to IAM service, scroll down below and find Access keys section. Access keys are used to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls.

Next, click on Create access key > Choose option CLI > confirm > Next > add a Description tag value, for example: tf-access-key > click on create access key

Once the access key is created, save the Access key and Secret access key (you can also download it in a .csv file).

**Authenticate with IAM user credentials**

To configure the AWS CLI, we are going to use aws configure. For general use, the aws configure command is the fastest way to set up AWS CLI installation. Open your local machine’s terminal and use aws configure.

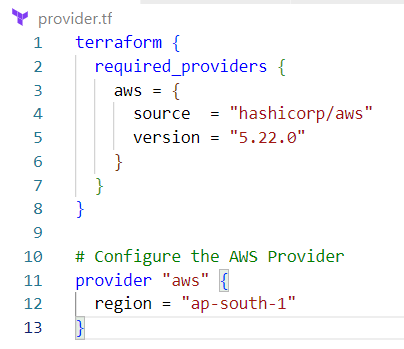
To check if the connection has been successfully established, run this command: aws iam list-users

**Configuration using Terraform**

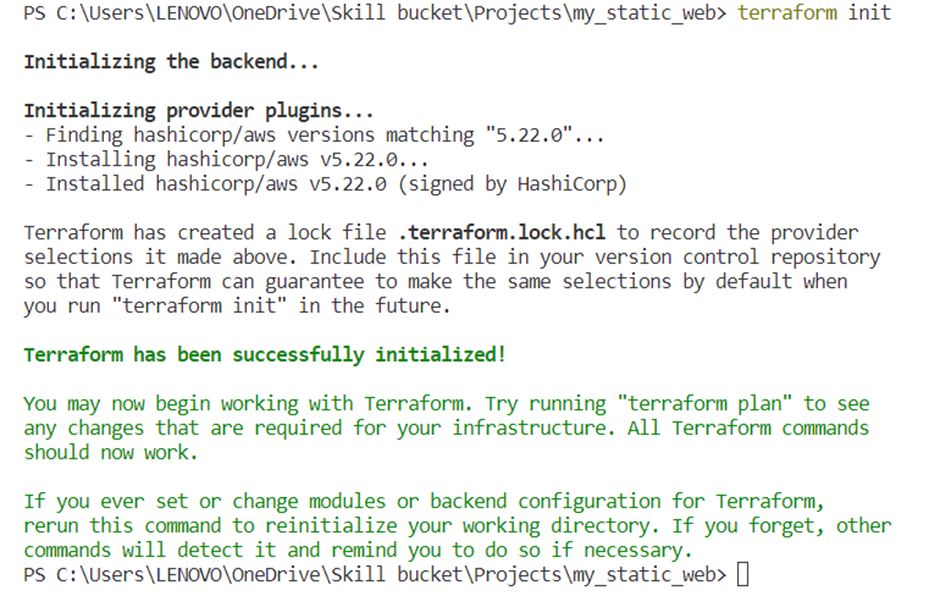
**Adding Provider and creating a bucket**

Create a new folder using your terminal and open that folder in Visual Studio Code.

Whenever we need to work with AWS in Terraform we need to create a provider. Create a provider.tf file. We will make use of Terraform’s documentation as much as possible. Search for AWS Provider in terraform documentation and copy the configuration as provided here: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs>

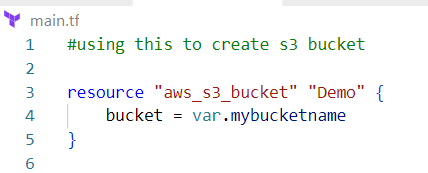


You can now initialize terraform using terraform init, it will help download all the required dependencies or files for the provider.

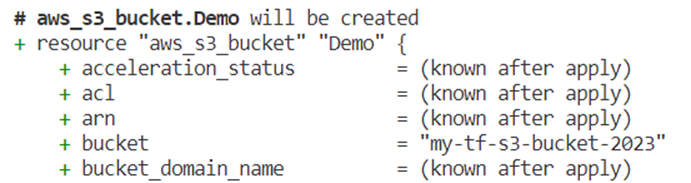


Now, let’s create a new file under our folder my-static-web and name that file main.tf. We will use this file to define our bucket configuration. Going back to Terraform documentation, we will look for S3 bucket configuration: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket>

Make sure that you have a globally unique name of the bucket. In our case, we will define the name of the bucket in the form of a variable. For this, we will create a separate file called variables.tf



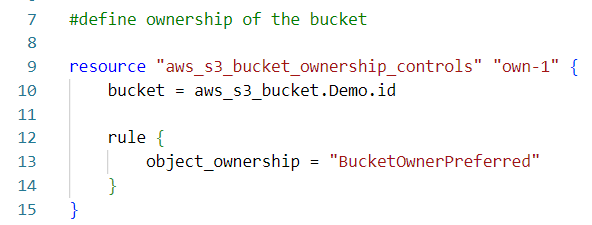
Once done, run terraform plan and then terraform apply -auto-approve (use -auto-approve only if you want to skip typing yes to apply) and then check if the bucket is getting created successfully in AWS



login into your AWS account > S3 > you will find the newly created bucket.

**Manage S3 Bucket Ownership**

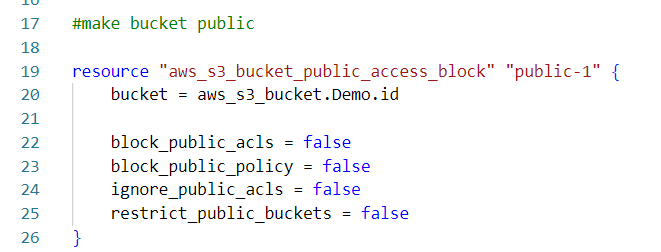
We can make use of Terraform documentation here to define bucket ownership: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_ownership_controls>



This will ensure that everything in this bucket is owned by the bucket owner.

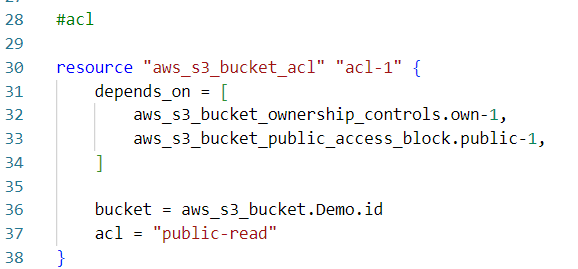
**Making the bucket Public and add ACL resource**

Using the documentation, we will now make our S3 bucket public: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_public_access_block>

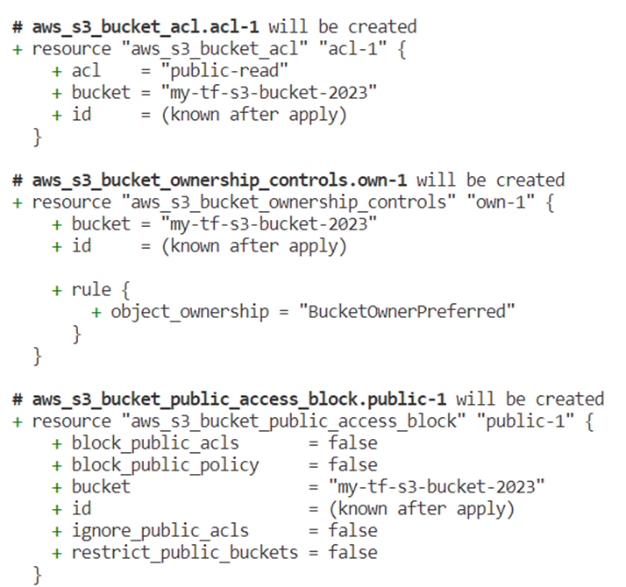


To add ACL resource use: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_acl>

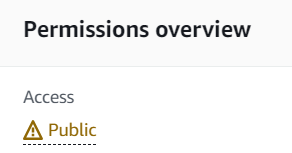
ACL is a set of permissions associated with an S3 bucket. It defines who can access the objects within the bucket and the type of access they have. It helps to manage public access to objects within the bucket. By setting permissions in the ACL, we can make objects within a bucket either publicly accessible or restricted.

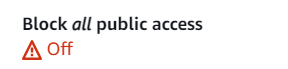


Now run terraform plan and terraform apply to apply the changes to the bucket.



To check the changes on AWS, navigate to bucket > Permissions





**Enable Static website hosting**

To set up S3 Static website, we need to enable static website hosting, for that, we need to make use of website configuration resource: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_website_configuration>.

To use this, we need to first create index.html and error.html and add those in our S3 bucket. In VSC, under our main folder my\_static\_web, create two files- index.html and error.html.

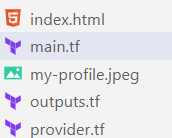
Index.html- serves as the default landing page of our website. This file typically contains the main content of our website, such as text, images, links. It’s the page visitors first see when they access a website.

Error.html- This file comes into play when visitors encounter errors while navigating our website. These errors could be due to broken links, mistyped URLs, or other issues that prevent a page from loading correctly.

You can simply visit my github and use both files as is: <https://github.com/lakshmir2023/Setting-Static-Website-using-Terraform-and-AWS>,

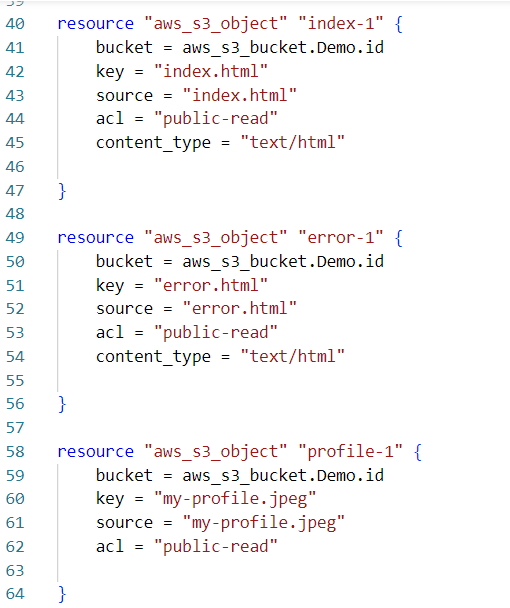
you will just need to tweak it a little which I will explain below.

For adding a profile picture in index.html, add the photo in your main folder, which in my case is my\_static\_web.

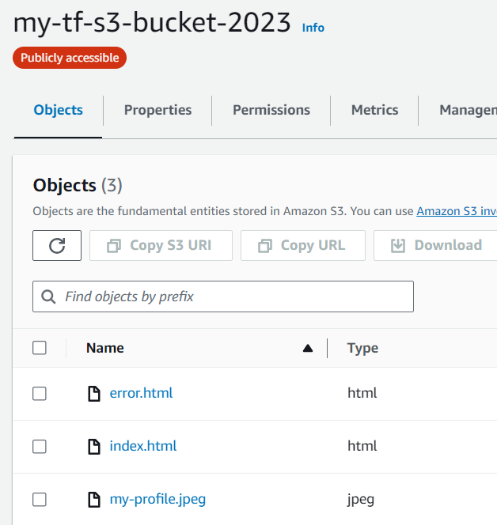


This file is now visible in our VSC, let’s first add this to our S3 bucket and then we can use the object URL to make changes to index.html file. To put objects to S3, we can make use of documentation: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/s3_object>

In our main.tf, define resources for all three objects.



Now, do a terraform plan and terraform apply. After that, check using AWS console that the objects have been added to our S3 bucket.



Let’s edit index.html to add profile photo, navigate to AWS > S3> your bucket > my-profile.jpeg > copy Object URL

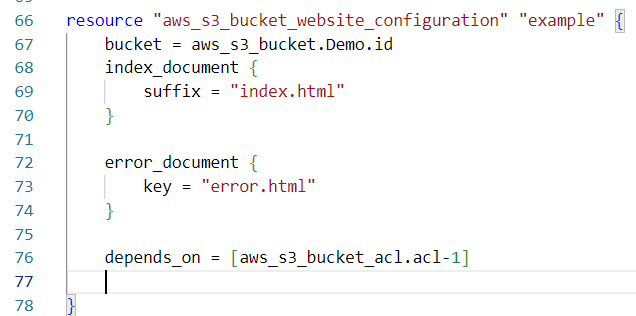


After copying object URL, go to VSC > index.html and add object URL to img src.



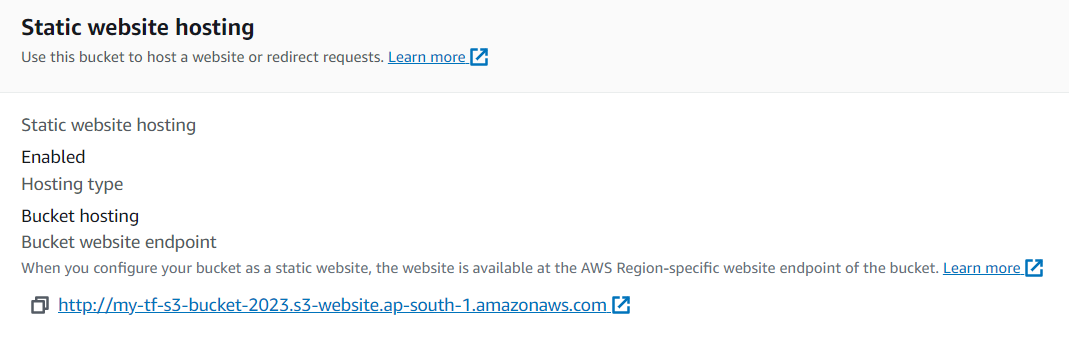
The index.html file contains around many occurrences of my name, you can replace it with your name.

To configure website in terraform, let’s make use of the documentation: <https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_website_configuration>

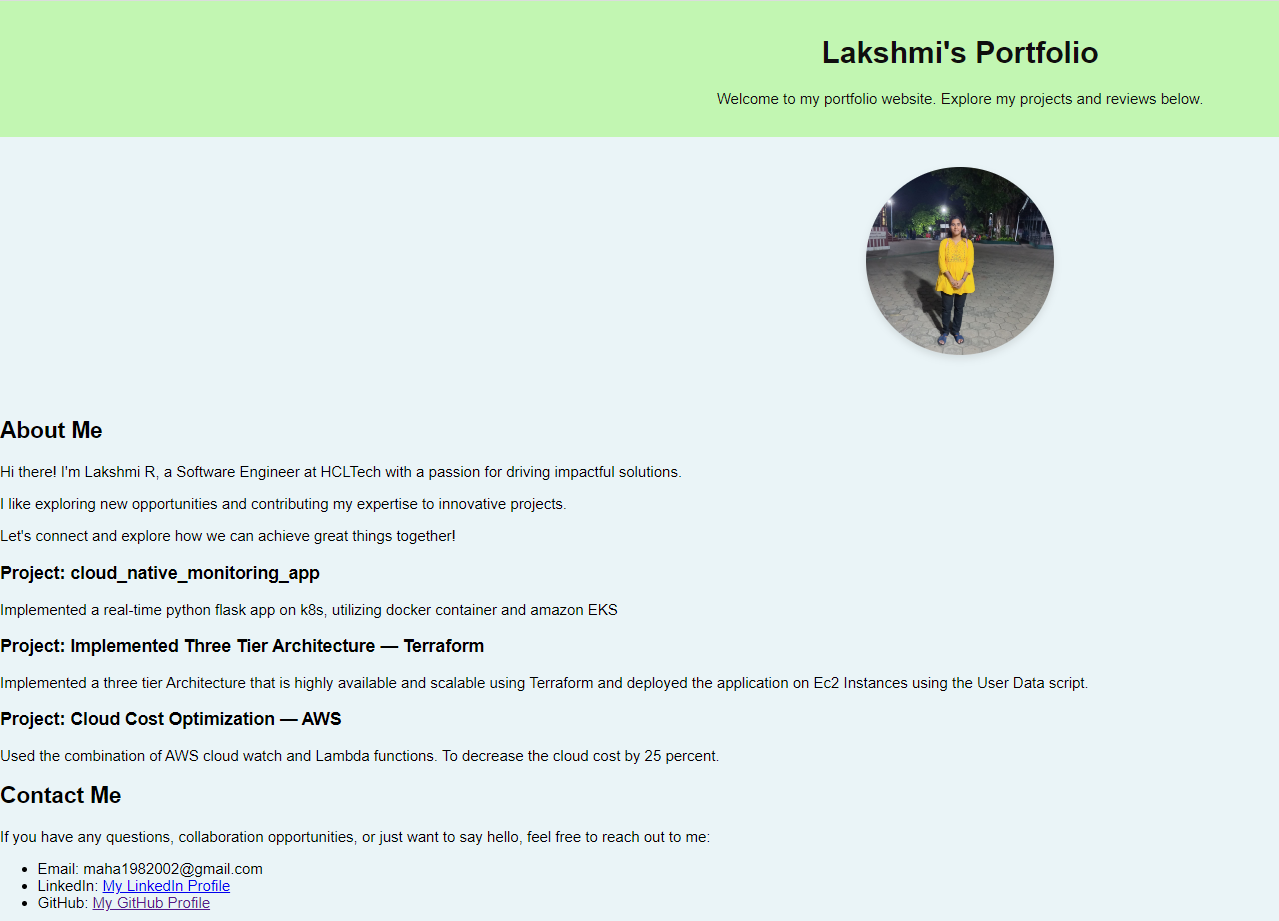


After adding website configuration resource, run terraform plan and apply

Navigate to AWS> S3 bucket > your bucket name > Properties > static web hosting

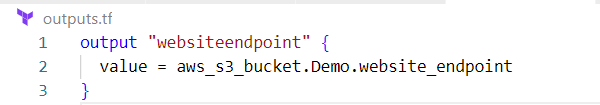


If we click on this link, we can see the portfolio!



Feel free to make changes to index.html file and customise it as you please.

We can also create outputs.tf file so that we can get endpoint right on our terminal. By creating this output variable, we enable Terraform to provide us with the URL endpoint of our static website after the infrastructure is created.



That’s it! I hope you enjoyed working on this short and pretty interesting project where you got to learn about AWS S3 and Infrastructure as Code using Terraform. If you feel anything can be made better in this project or you find any issues or run into any problems, please feel free to let me know.

Happy Learning!