

## Case Study Overview

In the world of cloud computing and infrastructure automation, the ability to efficiently manage and deploy resources is paramount. This case study highlights how Terraform, a powerful Infrastructure as Code (IaC) tool, can be used to automate the provisioning of AWS resources. Specifically, we focus on creating an EC2 instance and an S3 bucket, deploying a static website on the S3 bucket, and utilizing the EC2 instance as a backend server to interact with the S3 bucket.

## Key Features

1. **Automated Provisioning:** Leveraging Terraform to script the creation of AWS resources, ensuring consistent and repeatable deployments.
2. **Scalability:** Using AWS S3 for hosting a static website provides a scalable and cost-effective solution.
3. **Backend Server:** An EC2 instance serves as a backend server, allowing dynamic interactions with the static website hosted on S3.
4. **Logging and Monitoring:** The EC2 instance logs actions performed on the S3 bucket, providing insights into usage and access patterns.
5. **Infrastructure as Code (IaC):** The use of Terraform enables version control, modularization, and collaboration in managing infrastructure.

## Application

This case study exemplifies the practical application of Infrastructure as Code principles to streamline resource management in AWS. By automating the provisioning of an EC2 instance and an S3 bucket, we can deploy and manage a static website efficiently. The EC2 instance's role as a backend server demonstrates how to extend static websites with dynamic capabilities, logging interactions to monitor and optimize performance. This approach ensures scalability, reliability, and ease of maintenance, making it ideal for modern cloud-based applications. Through this example, we see how Terraform, AWS S3, and EC2 come together to provide a robust solution for deploying and managing cloud resources, reinforcing the value of automation and cloud-native practices.

## Implementation

1. Initialize a new Terraform configuration:

Create a directory for your Terraform configuration and navigate to it.

2. Create a new file named `main.tf` and add the following content:

```
main.tf
1  provider "aws" {
2      region = "us-west-2"
3      access_key = "AKIAVY2PGSUVMQXKNK5F"
4      secret_key = "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"
5  }
6
7  resource "aws_s3_bucket" "static_website_bucket" {
8      bucket = "my-static-website-bucket-unique123"
9  }
10
11 resource "aws_s3_bucket_policy" "bucket_policy" {
12     bucket = aws_s3_bucket.static_website_bucket.id
13
14     policy = <<POLICY
15     {
16         "Version": "2012-10-17",
17         "Statement": [
18             {
19                 "Effect": "Allow",
20                 "Principal": "*",
21                 "Action": "s3:GetObject",
22                 "Resource": "arn:aws:s3:::${aws_s3_bucket.static_website_bucket.bucket}/*"
23             }
24         ]
25     }
26     POLICY
27 }
28
29 resource "aws_s3_bucket_website_configuration" "static_website_config" {
30     bucket = aws_s3_bucket.static_website_bucket.bucket
```

3. Add the index.html file to the directory

```
<> index.html > html > body > form
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Custom T-Shirt Order Form</title>
7  </head>
8  <body>
9      <h1>Custom T-Shirt Order Form</h1>
10     <form action="/submit-order" method="post">
11         <fieldset>
12             <legend>T-Shirt Details</legend>
13
14             <label for="tagline">Tagline on the Shirt:</label>
15             <input type="text" id="tagline" name="tagline" required placeholder="Enter your t
16
17             <label for="color">Color:</label>
18             <select id="color" name="color" required>
19                 <option value="">Select a color</option>
20                 <option value="white">White</option>
21                 <option value="black">Black</option>
22                 <option value="red">Red</option>
23                 <option value="blue">Blue</option>
24                 <option value="green">Green</option>
25             </select>
26
27             <label for="size">Size:</label>
28             <select id="size" name="size" required>
29                 <option value="">Select a size</option>
30                 <option value="s">Small</option>
```

Ln 10, Col 48 Spaces: 4 UTF-8 CRLF HTML Go Live AI Code Chat

#### 4. Open cmd and perform these commands

terraform init

terraform apply

```
C:\Users\Atharva Prabhu\Desktop\terraformaws>terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.72.1
```

**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.

```
aws_instance.web_server (deposed object f790899c): Destroying... [id=i-0b445c1dbbde4225d]
aws_instance.web_server (deposed object 8104a9e8): Destroying... [id=i-02da99739a4d93ed8]
aws_instance.web_server: Still destroying... [id=i-0b445c1dbbde4225d, 10s elapsed]
aws_instance.web_server: Still destroying... [id=i-0dad81c9ae638d4f8, 10s elapsed]
aws_instance.web_server: Still destroying... [id=i-02da99739a4d93ed8, 10s elapsed]
aws_instance.web_server: Still destroying... [id=i-0b445c1dbbde4225d, 20s elapsed]
aws_instance.web_server: Still destroying... [id=i-02da99739a4d93ed8, 20s elapsed]
aws_instance.web_server: Still destroying... [id=i-0dad81c9ae638d4f8, 20s elapsed]
aws_instance.web_server: Still destroying... [id=i-0dad81c9ae638d4f8, 30s elapsed]
aws_instance.web_server: Still destroying... [id=i-02da99739a4d93ed8, 30s elapsed]
aws_instance.web_server: Still destroying... [id=i-0b445c1dbbde4225d, 30s elapsed]
aws_instance.web_server: Destruction complete after 32s
aws_instance.web_server: Still destroying... [id=i-0b445c1dbbde4225d, 40s elapsed]
aws_instance.web_server: Still destroying... [id=i-0dad81c9ae638d4f8, 40s elapsed]
aws_instance.web_server: Still destroying... [id=i-0dad81c9ae638d4f8, 50s elapsed]
aws_instance.web_server: Still destroying... [id=i-0b445c1dbbde4225d, 50s elapsed]
aws_instance.web_server: Still destroying... [id=i-0b445c1dbbde4225d, 1m0s elapsed]
aws_instance.web_server: Still destroying... [id=i-0dad81c9ae638d4f8, 1m0s elapsed]
aws_instance.web_server: Still destroying... [id=i-0dad81c9ae638d4f8, 1m10s elapsed]
aws_instance.web_server: Still destroying... [id=i-0b445c1dbbde4225d, 1m10s elapsed]
```

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

**Outputs:**

```
bucket_name = "my-static-website-bucket-unique123"
instance_id = "i-0ab3c7380269c6bdb"
public_ip = "54.191.86.65"
```

```
C:\Users\Atharva Prabhu\Desktop\terraformaws>
```

### 5. Connect ec2 instance using ssh command

```
PS C:\Users\Atharva Prabhu> ssh -i "C:\Users\Atharva Prabhu\Downloads\first.pem" ec2-user@ec2-54-218-79-250.us-west-2.compute.amazonaws.com
```

```
#_
#_   #####_      Amazon Linux 2023
#_   \#####\_
#_   ####|
#_   \|##/\#/    https://aws.amazon.com/linux/amazon-linux-2023
#_   V__!____->
#_   /
#_   _/_/___/___/
#_   _/_/___/___/
#_   _/m/'
```

```
Last login: Tue Oct 22 13:54:56 2024 from 49.36.101.86
[ec2-user@ip-172-31-21-83 ~]$
Broadcast message from root@localhost (Tue 2024-10-22 15:07:49 UTC):

The system will power off now!
```

```
Connection to ec2-54-218-79-250.us-west-2.compute.amazonaws.com closed by remote host.
Connection to ec2-54-218-79-250.us-west-2.compute.amazonaws.com closed.
```

### 6.Add textfile to s3 bucket using ec2 instance

```
[ec2-user@ip-172-31-23-146 ~]$ aws s3 ls s3://my-static-website-bucket-unique123/
2024-10-22 15:07:47          2834 index.html

Last login: Wed Oct 23 04:57:57 2024 from 18.237.140.163
[ec2-user@ip-172-31-23-146 ~]$ echo "Test log file" > textfile.txt
[ec2-user@ip-172-31-23-146 ~]$ aws s3 cp testfile.txt s3://my-static-website-bucket-unique123/

The user-provided path testfile.txt does not exist.
[ec2-user@ip-172-31-23-146 ~]$ aws s3 cp textfile.txt s3://my-static-website-bucket-unique123/
upload: ./textfile.txt to s3://my-static-website-bucket-unique123/textfile.txt
[ec2-user@ip-172-31-23-146 ~]$ aws s3 cp textfile.txt s3://my-static-website-bucket-unique123/ > upload_log.txt
[ec2-user@ip-172-31-23-146 ~]$ cat s3 logs.txt
2024-10-22 15:07:47          2834 index.html
[ec2-user@ip-172-31-23-146 ~]$ cat upload_log.txt
upload: ./textfile.txt to s3://my-static-website-bucket-unique123/textfile.txt
[ec2-user@ip-172-31-23-146 ~]$
```

Objects (2) Info

↻

Copy S3 URI

Copy URL

Download

Open

Delete

Actions ▼

Create folder



Upload

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

< 1 >

⚙

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 <a href="#">index.html</a>	html	October 22, 2024, 20:37:47 (UTC+05:30)	2.8 KB	Standard
<input type="checkbox"/>	 <a href="#">textfile.txt</a>	txt	October 23, 2024, 10:59:11 (UTC+05:30)	14.0 B	Standard

## Guidelines:

- IAM Policies: Ensure that the IAM user used in Terraform has the least privilege necessary to avoid over-permissioning.
- Bucket Naming: S3 bucket names must be globally unique, so modify the name accordingly.
- Cost Management: Keep track of your running instances and S3 storage usage to prevent unwanted costs.
- Backup your `.pem` file: The SSH key should be stored securely as it is required to access the EC2 instance.

## Certification Requirement

### Deadline:

October 21st

### Importance:

The certification is crucial for validating your proficiency in using Terraform for AWS resource management. It demonstrates your ability to automate infrastructure provisioning, ensuring repeatable and reliable deployments. Achieving certification underscores your expertise in cloud-native practices, which is essential for the successful execution and understanding of this case study. It shows you're not just implementing a solution, but doing so with best practices and recognized skills. Make sure to complete your certification by the deadline to fully leverage this case study's potential and your professional growth.

