Kartik bhat AdvancedDevops Case Study

D15A 03

**Case study overview**

This case study focuses on utilizing Terraform, a popular infrastructure-as-code tool, to provision cloud infrastructure on AWS. Specifically, the study will cover how to create an EC2 instance and an S3 bucket with Terraform and deploy a static website on the S3 bucket while using the EC2 instance as the backend for logging and interactions.

**Key Feature and Application**:  
The primary feature of this case study is automating cloud infrastructure management with Terraform. This includes provisioning both an EC2 instance and an S3 bucket, which are foundational elements in cloud services. The practical application involves deploying a website and logging interactions between the EC2 instance and the S3 bucket, showcasing a real-world scenario of infrastructure automation.

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.

**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**

#### **Step 1: Initial Setup**

1. **Install Terraform**:  
   If Terraform is not already installed, install it on your local machine. Follow the official Terraform documentation here for platform-specific instructions.
2. **Configure AWS CLI**:  
   Ensure the AWS CLI is installed and configured with your access keys:

#### **Step 2: Write the Terraform Script**

1. **Main Configuration File (main.tf)**: Create a new directory for the project and create a file called main.tf. This file will contain the code to provision the EC2 instance and S3 bucket.

provider "aws" {

region = "ap-south-1" # Change to your desired region

}

resource "aws\_s3\_bucket" "website\_bucket" {

bucket = "my-terraform-static-website-bucket"

# Enable bucket versioning if needed

versioning {

enabled = true

}

# Disable ACLs (this line can be removed since it's the default behavior)

# acl = "private"

# Optional: Enable logging

logging {

target\_bucket = aws\_s3\_bucket.logging\_bucket.bucket

target\_prefix = "log/"

}

}

# Define the S3 bucket for logging (if you want to enable logging)

resource "aws\_s3\_bucket" "logging\_bucket" {

bucket = "my-terraform-logs-bucket"

acl = "private"

}

# Configure the static website settings

resource "aws\_s3\_bucket\_website\_configuration" "website" {

bucket = aws\_s3\_bucket.website\_bucket.id

index\_document {

suffix = "index.html"

}

error\_document {

key = "404.html" # Optional, define if you have an error document

}

}

### **Step 3: Deploy the Static Website on S3**

#### **Create an index.html File: Create a simple HTML file in your project directory:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Custom T-Shirt Order Form</title>

</head>

<body>

<h1>Custom T-Shirt Order Form</h1>

<form action="/submit-order" method="post">

<fieldset>

<legend>T-Shirt Details</legend>

<label for="tagline">Tagline on the Shirt:</label>

<input type="text" id="tagline" name="tagline" required placeholder="Enter your tagline">

<label for="color">Color:</label>

<select id="color" name="color" required>

<option value="">Select a color</option>

<option value="white">White</option>

<option value="black">Black</option>

<option value="red">Red</option>

<option value="blue">Blue</option>

<option value="green">Green</option>

</select>

<label for="size">Size:</label>

<select id="size" name="size" required>

<option value="">Select a size</option>

<option value="s">Small</option>

<option value="m">Medium</option>

<option value="l">Large</option>

<option value="xl">X-Large</option>

<option value="xxl">XX-Large</option>

</select>

<label for="quantity">Quantity:</label>

<input type="number" id="quantity" name="quantity" min="1" max="100" required>

<label for="delivery-date">Delivery Date:</label>

<input type="date" id="delivery-date" name="delivery-date" required>

</fieldset>

<fieldset>

<legend>Delivery Details</legend>

<label for="recipient-name">Recipient's Name:</label>

<input type="text" id="recipient-name" name="recipient-name" required>

<label for="address">Address:</label>

<input type="text" id="address" name="address" required>

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<label for="phone">Phone Number:</label>

<input type="tel" id="phone" name="phone" pattern="[0-9]{10}" required placeholder="1234567890">

</fieldset>

<label for="comments">Additional Comments:</label>

<textarea id="comments" name="comments" rows="4" cols="50" placeholder="Any special instructions?"></textarea>

<button type="submit">Place Order</button>

<button type="reset">Reset Form</button>

</form>

</body>

</html>

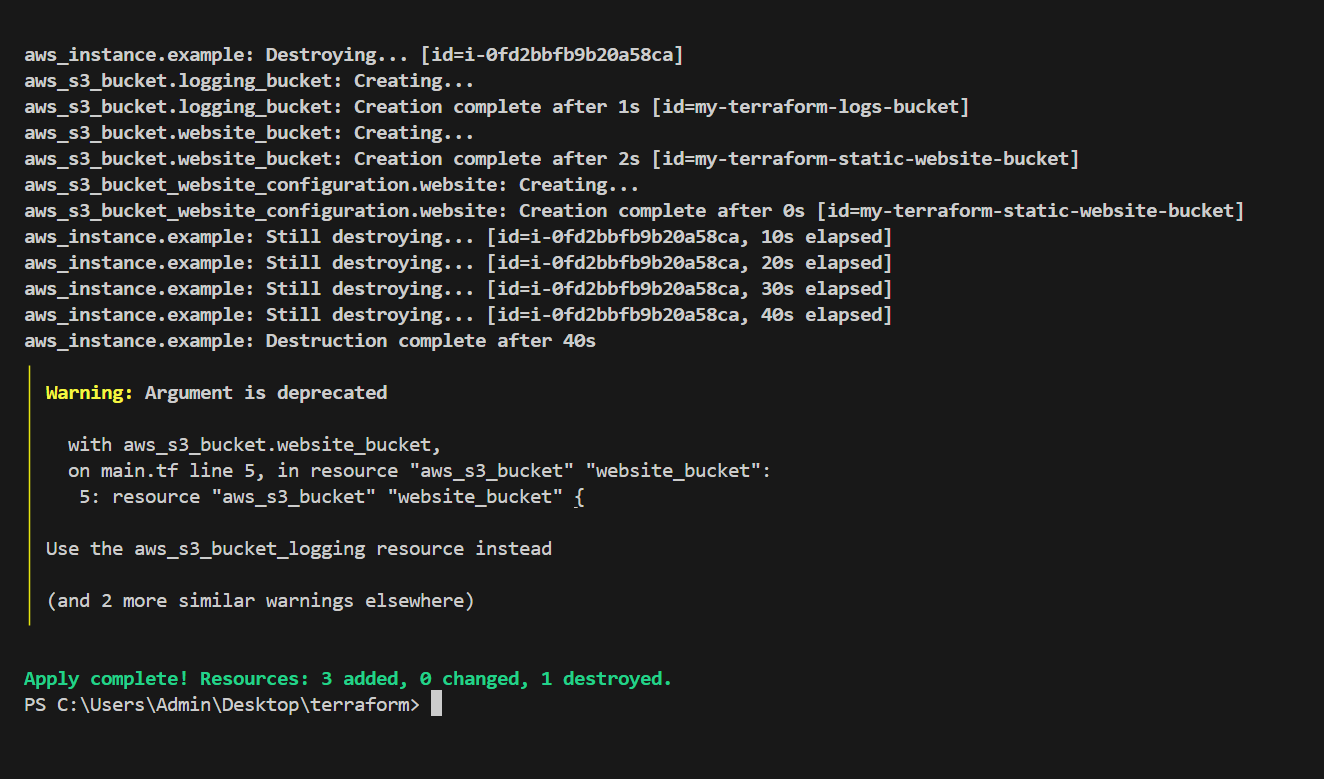
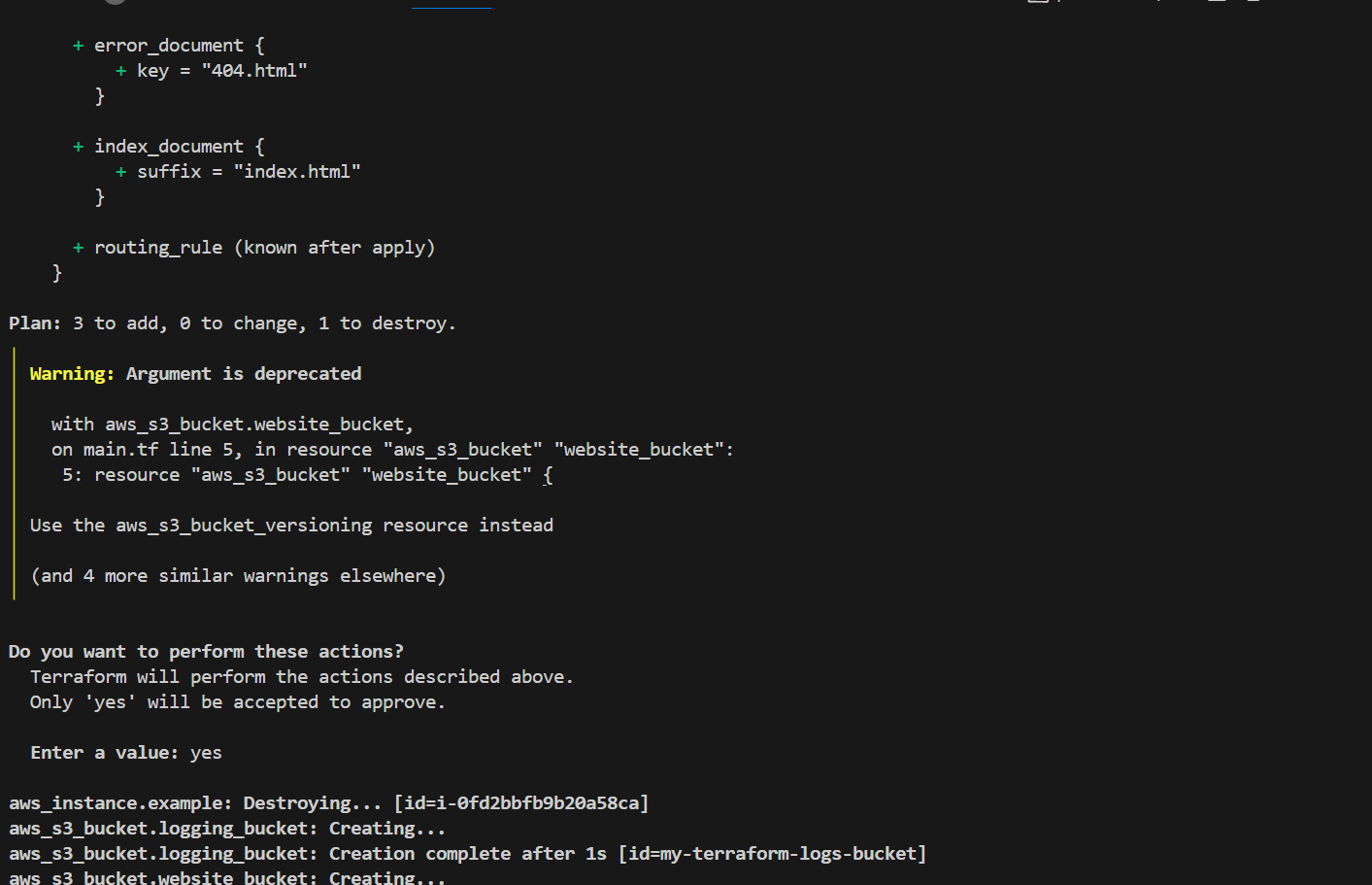
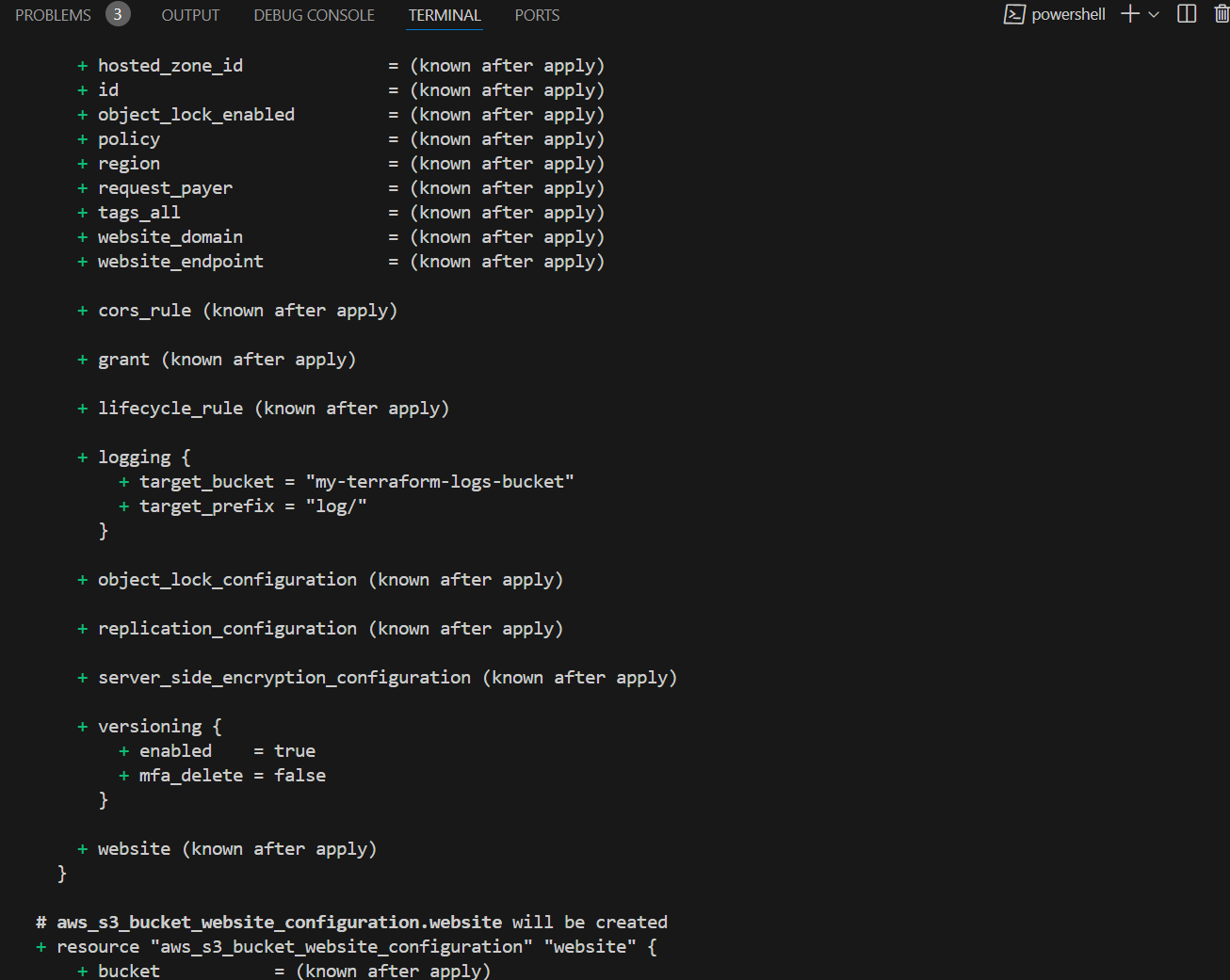
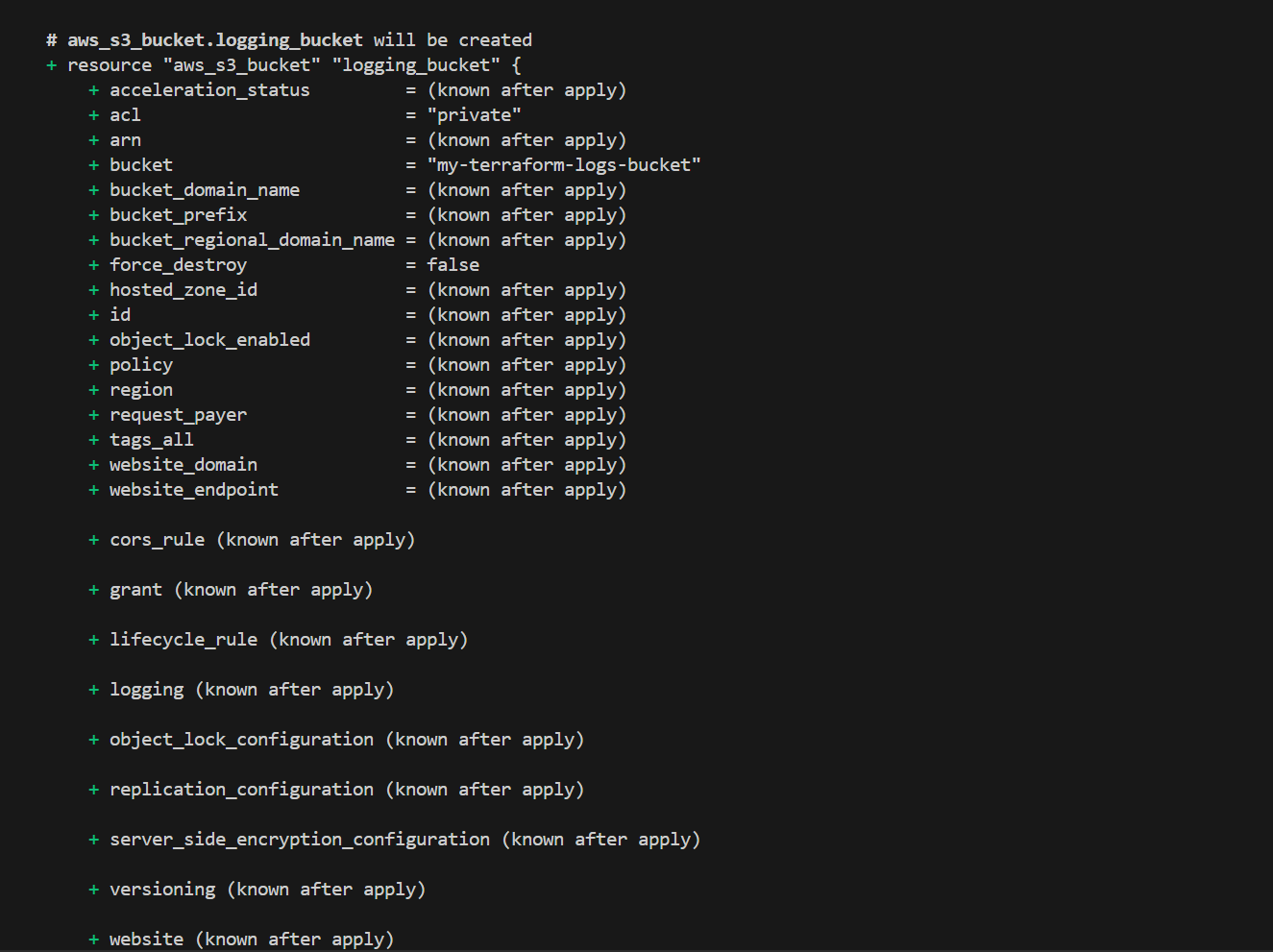
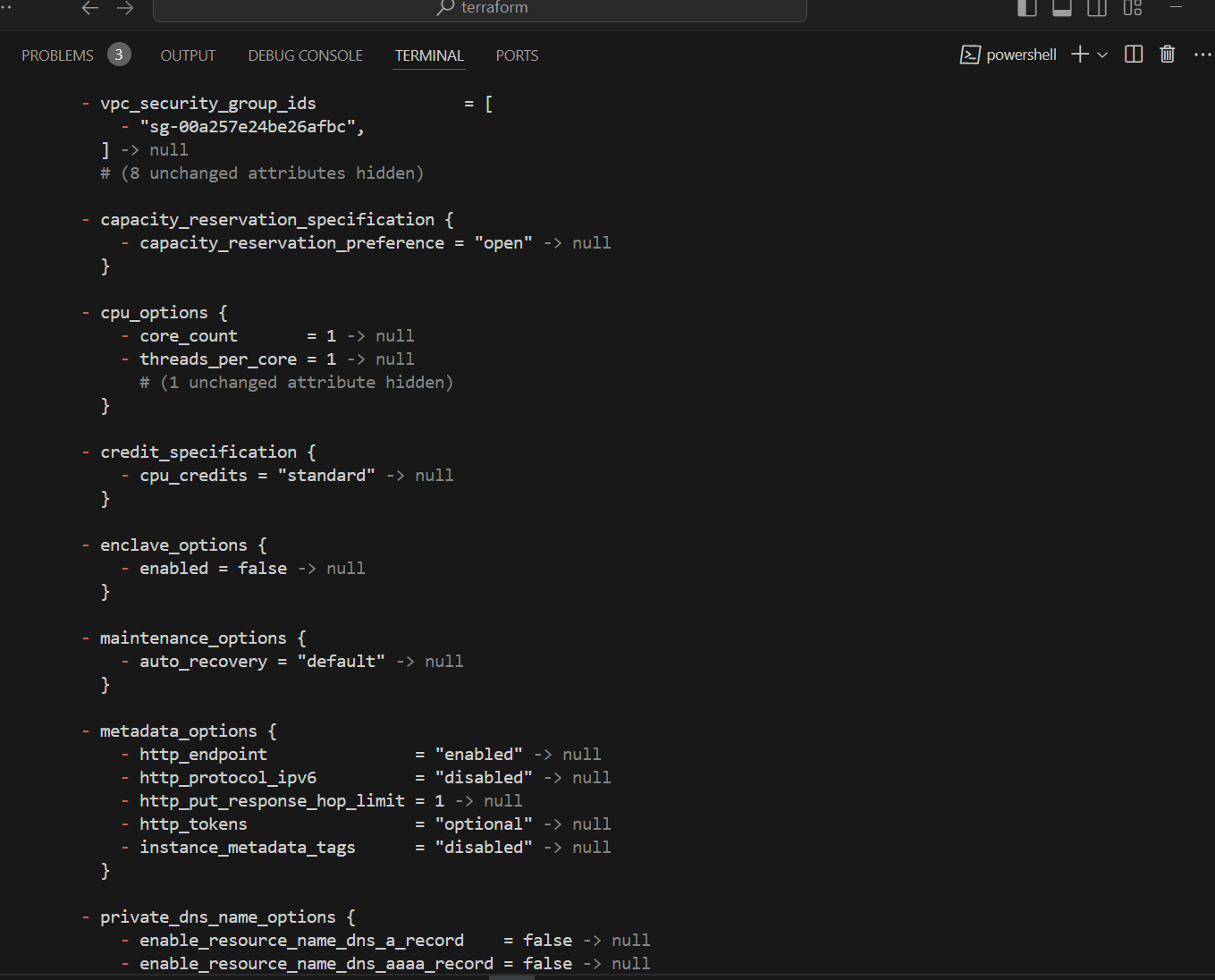
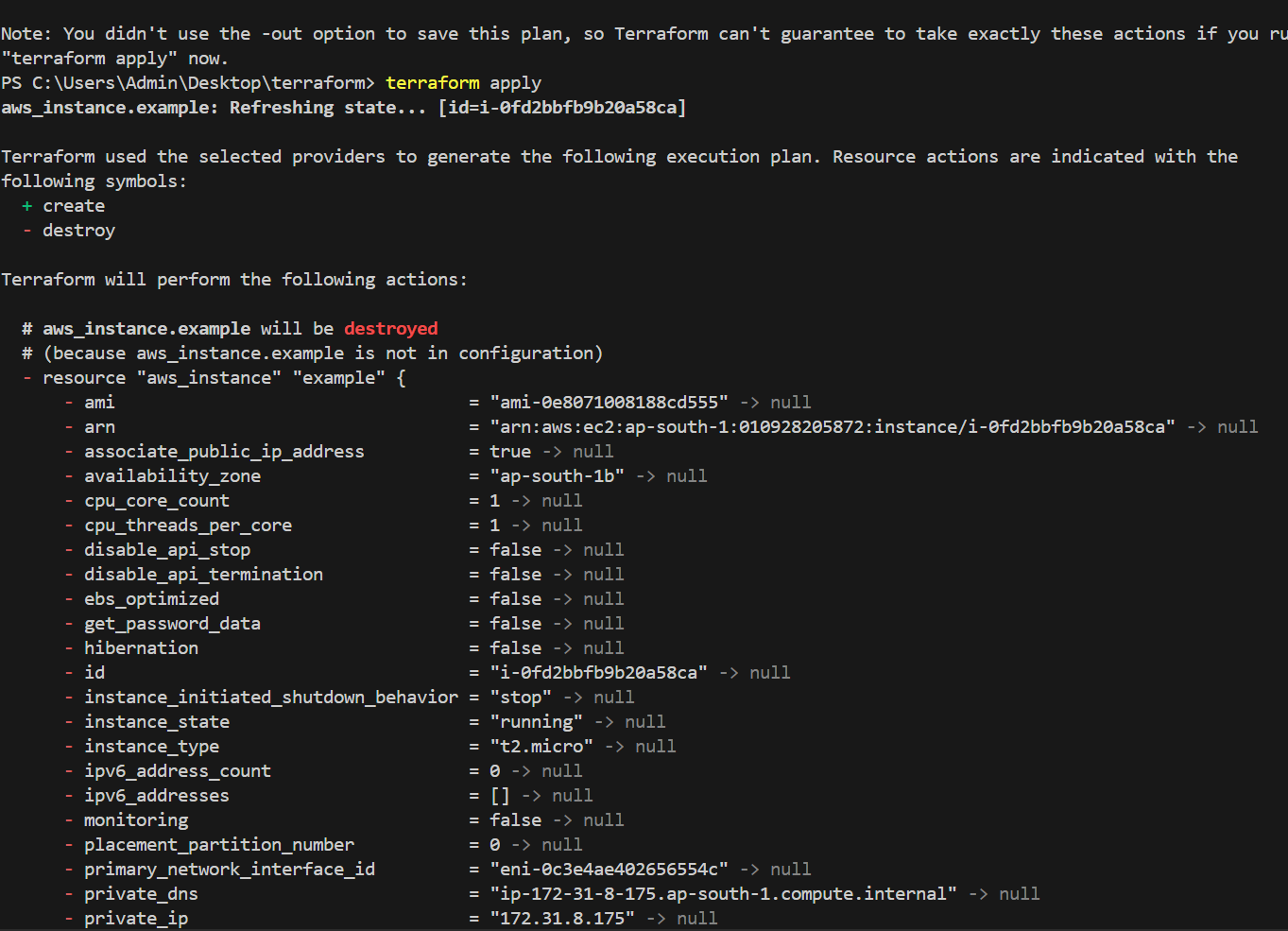
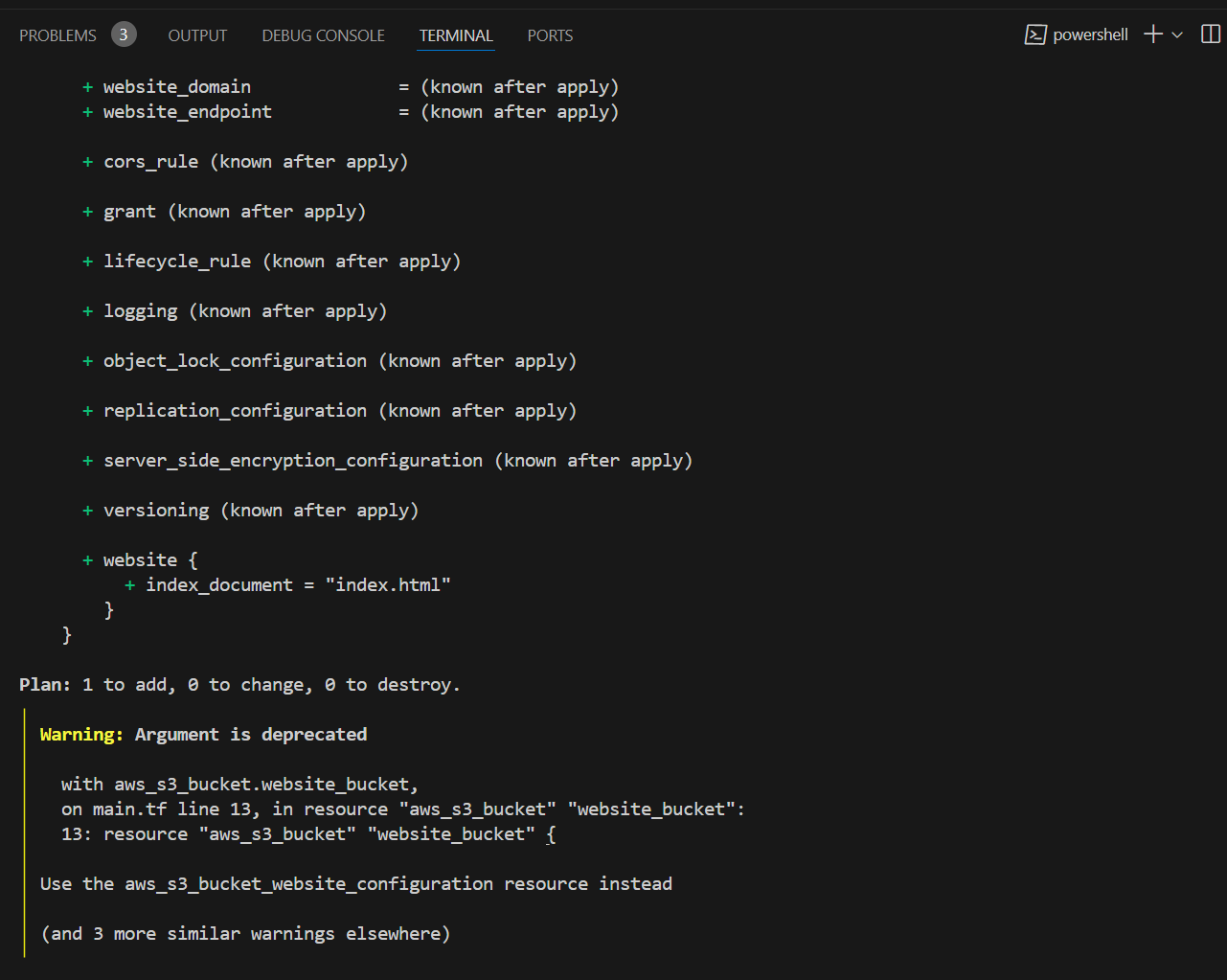
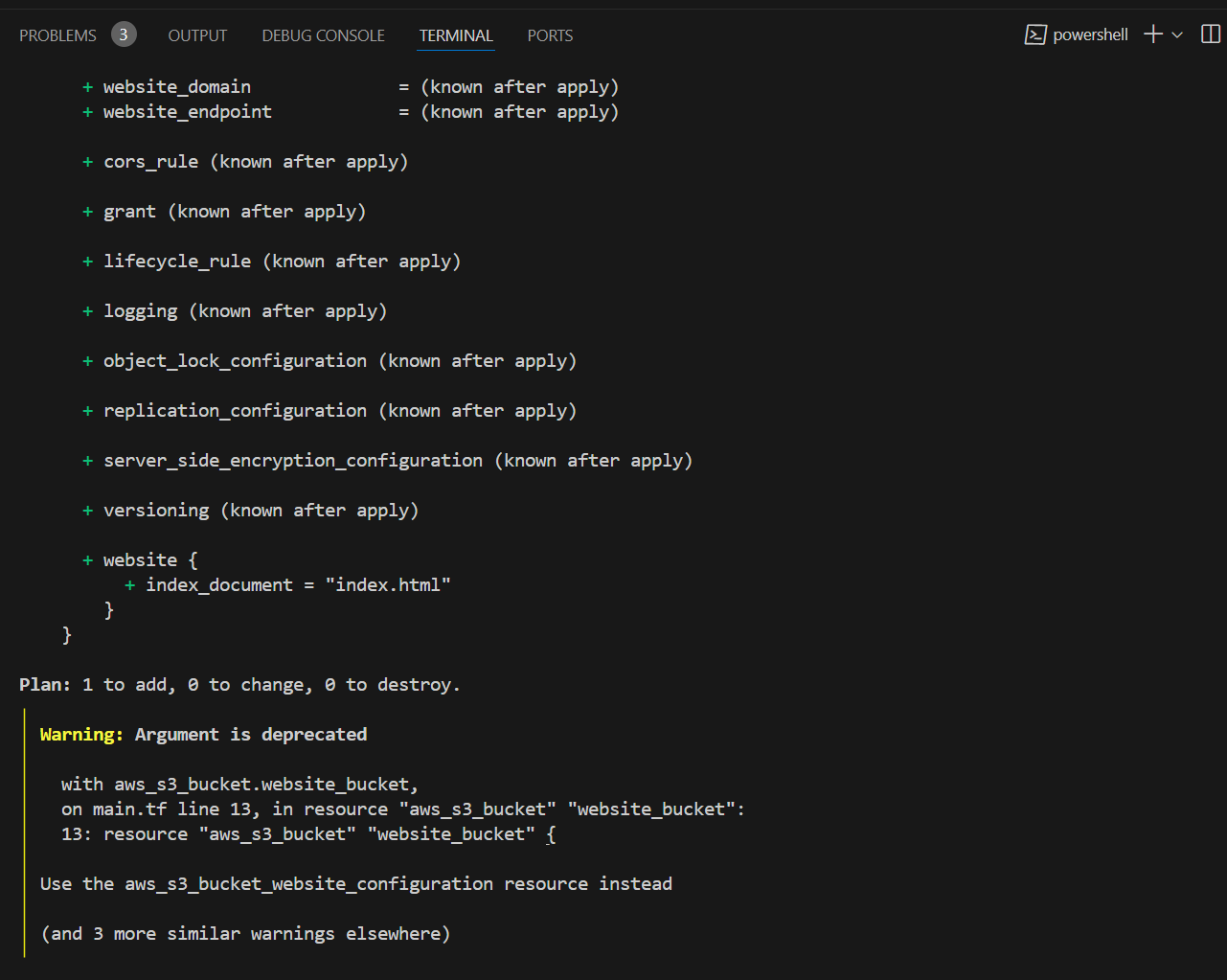
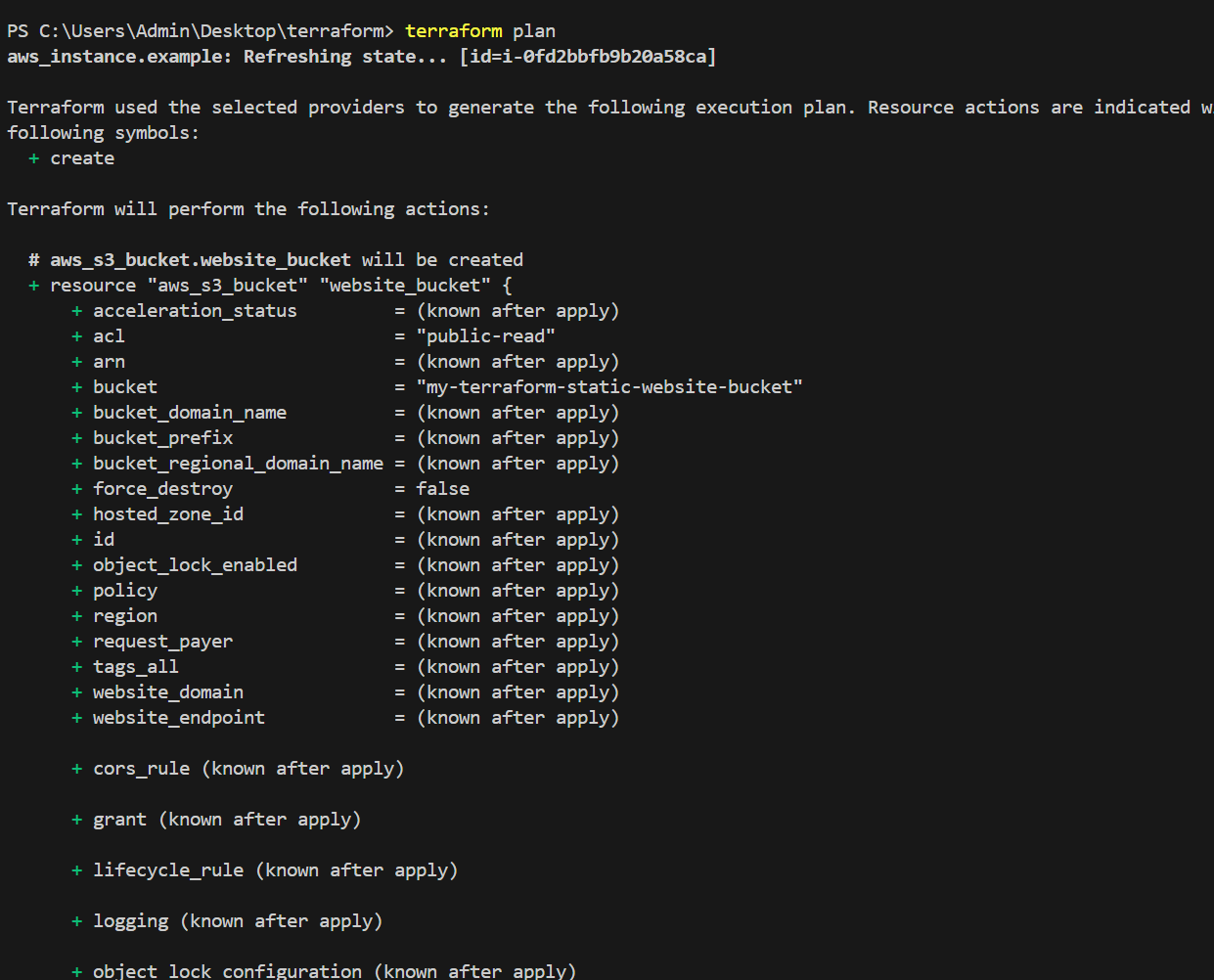
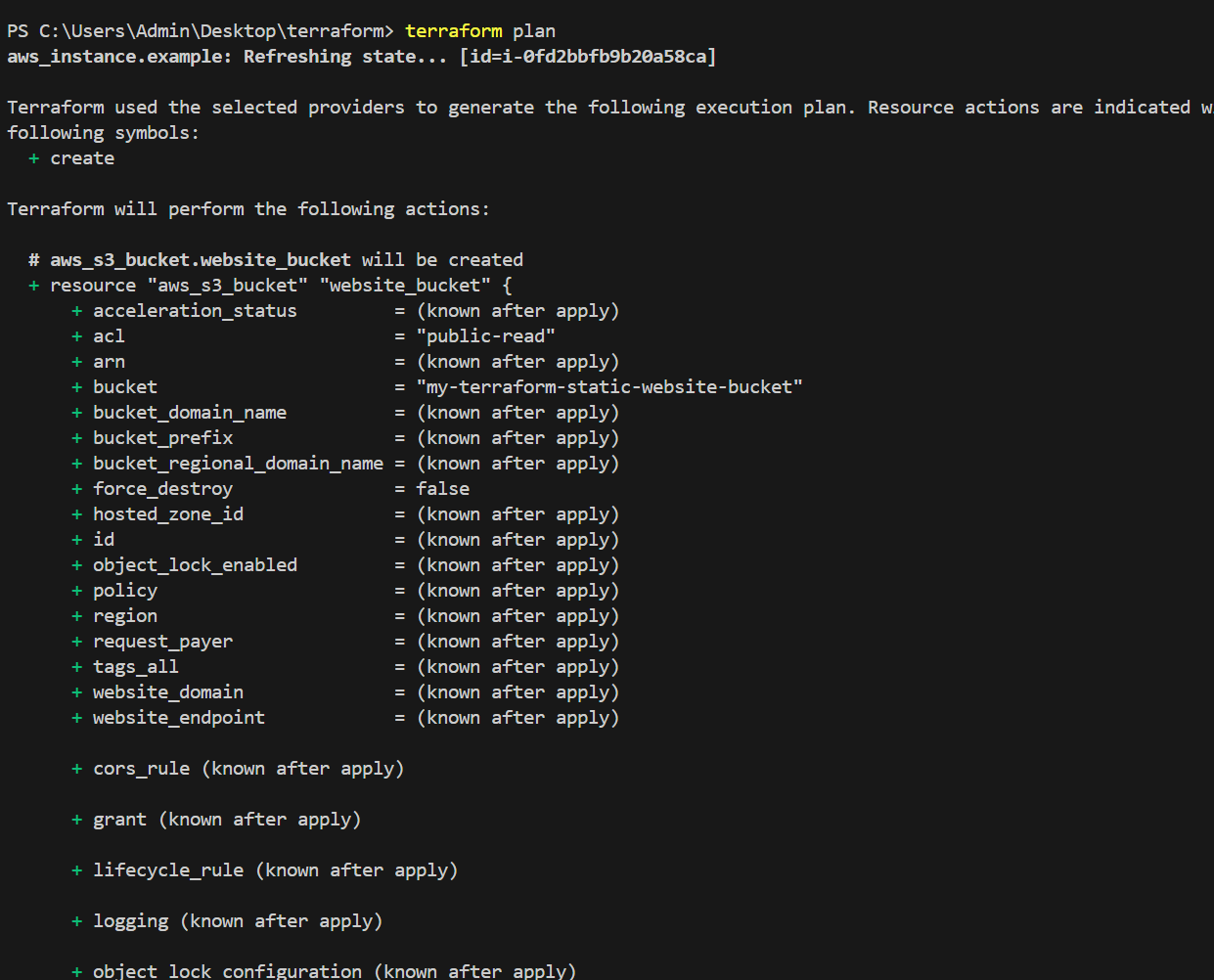
#### 

#### 

#### **Step 4: Initialize and Apply the Terraform Script**

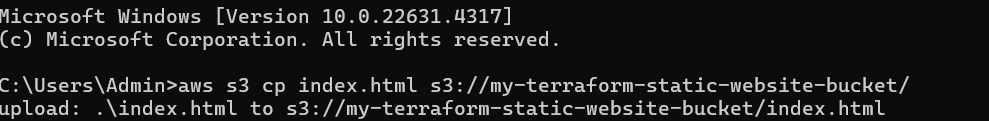
1. **Initialize the Directory**: Run terraform init in your project directory to initialize Terraform and install required providers.

**Apply the Terraform Configuration**: Run terraform apply to provision the resources. Terraform will prompt for confirmation before applying changes.



#### **Step 5: Deploy the Static Website on S3**

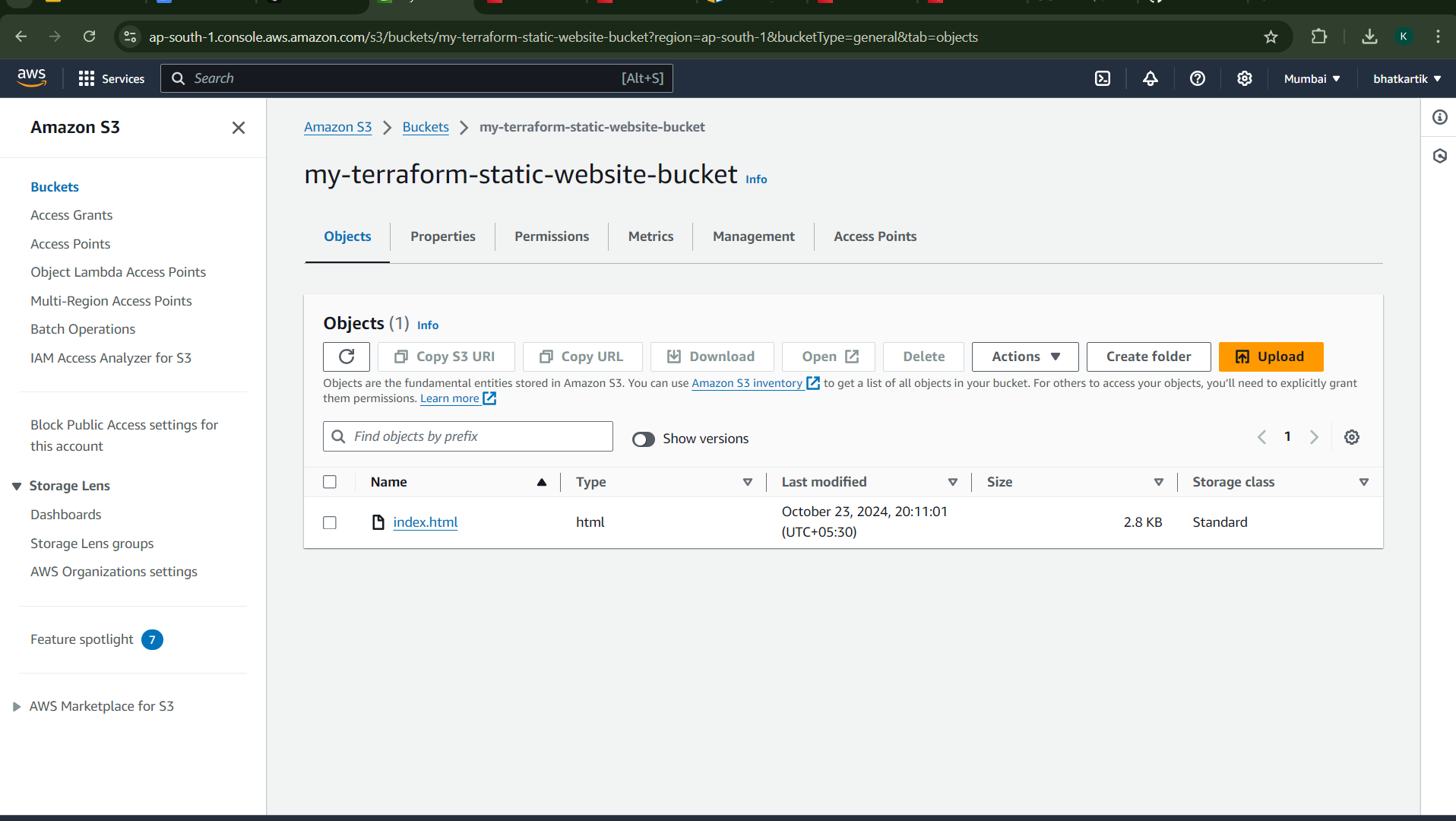
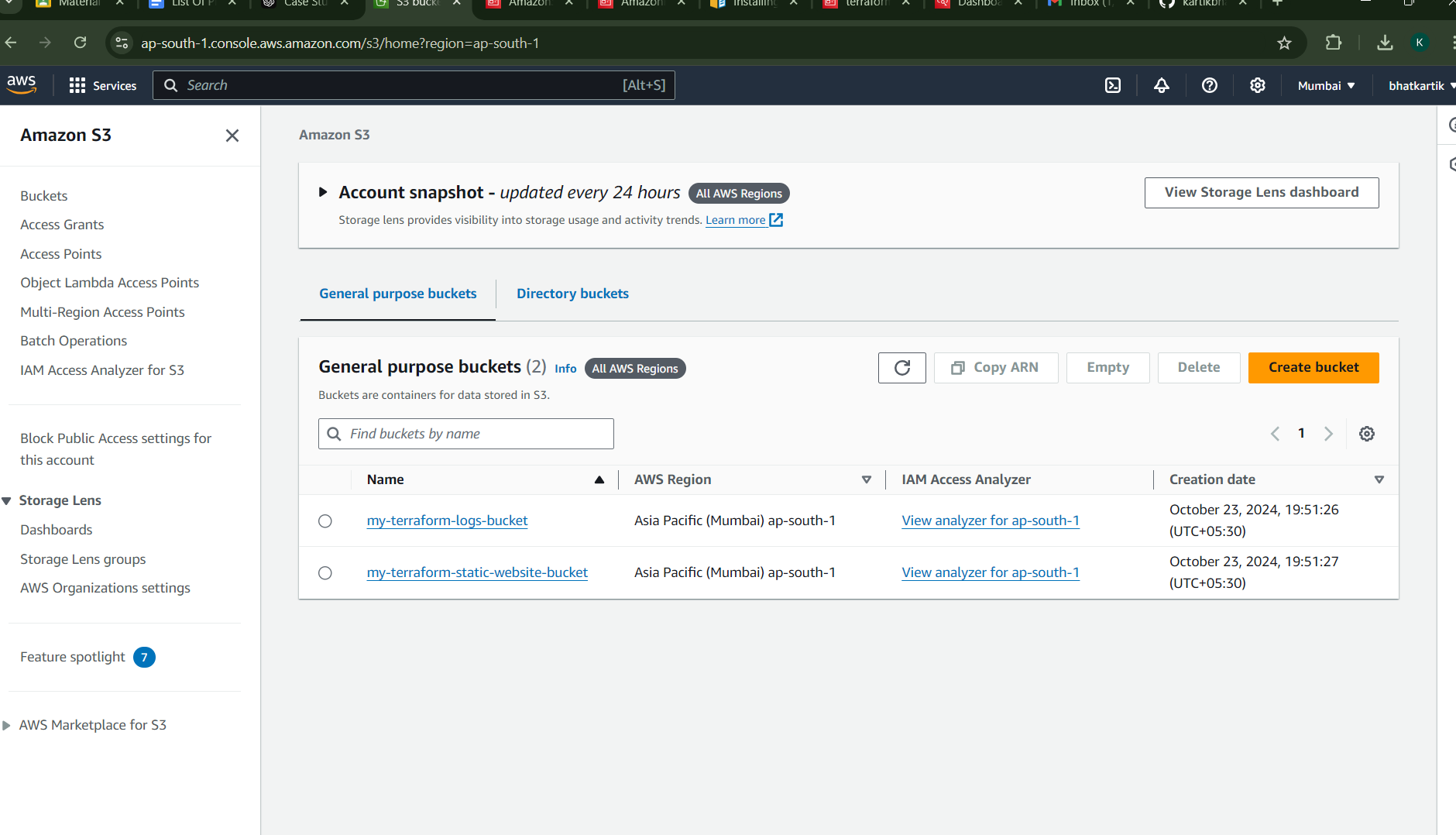
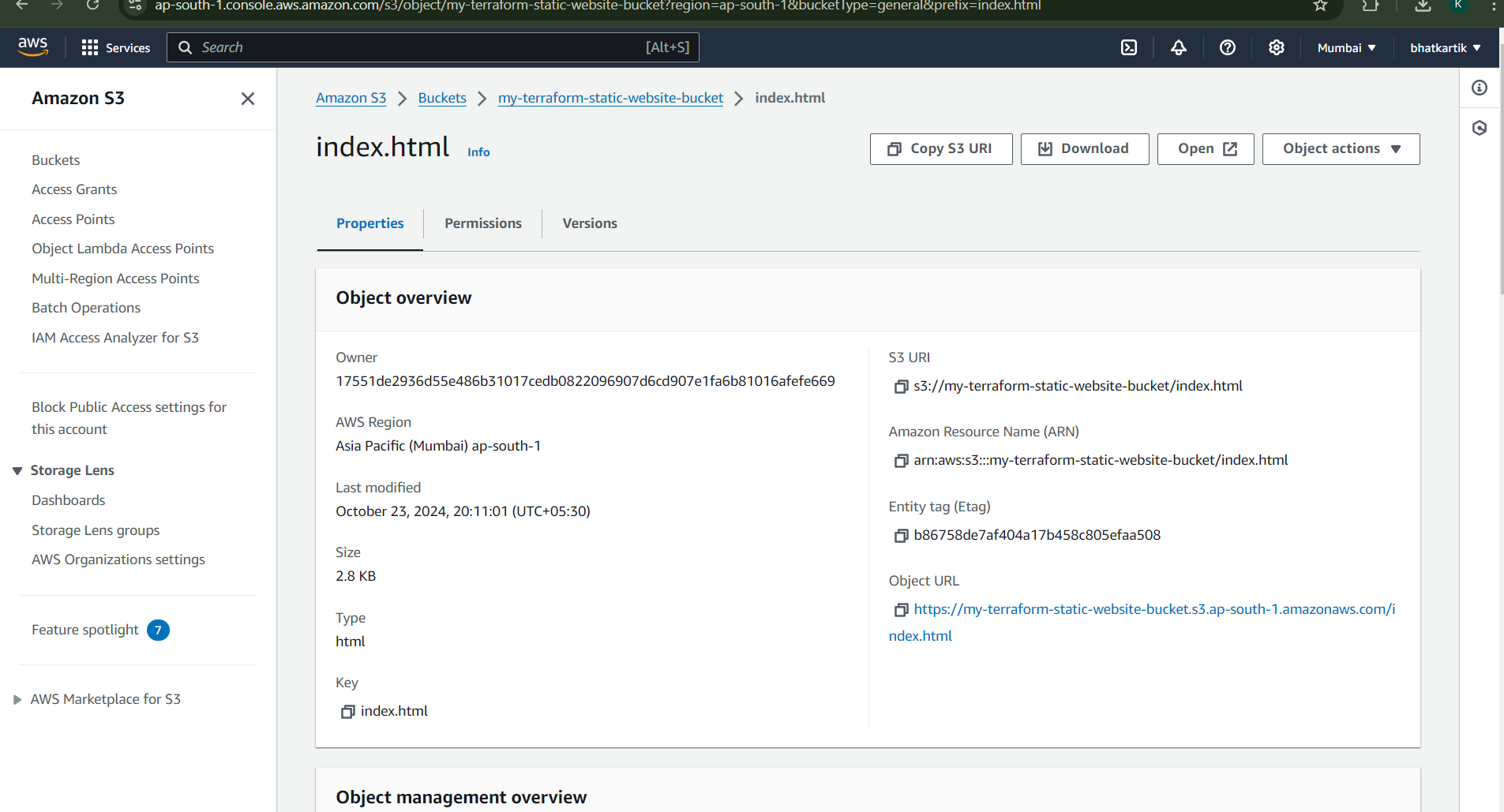
1. **Upload the Static Website Files**: Create a simple index.html file for your static website. Upload it to the S3 bucket created by Terraform.



**Step 6**

.Connect ec2 instance using ssh command





**Guidelines:**

● IAMPolicies: 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.

● CostManagement: Keep track of your running instances and S3 storage usage to prevent unwanted costs.

● Backupyour .pem file: The SSH key should be stored securely as it is required to access the EC2 instance

### **3. Demonstration Preparation**

* **Key Points**:
  + Highlight the use of Terraform to automate AWS infrastructure creation.
  + Demonstrate the process of creating an EC2 instance and S3 bucket.
  + Show how to deploy the website and log interactions.
* **Practice**:
  + Walk through each step of the process to ensure familiarity.
  + Test the website and log commands to verify everything works as expected.
* **Questions**:
  + What advantages does Terraform provide over manual cloud resource creation?
  + How can you scale this solution for larger infrastructure setups?
  + What other AWS services could be integrated using Terraform?

### **Certification Requirement**

* **Deadline**: Ensure submission of the complete report and certification by **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