Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

Cloud Computing

Semester Final Project

Submitted by : Noor-ul-Fajar (048)

Ushna Saad(069)

Session： V-B

Submitted to：Sir Waqas Saleen

Table of Contents

Multi-Environment Static Website (S3 + CloudFront) with Terraform & Ansible

1. Executive Summary

This project demonstrates the design and implementation of a fully automated, secure, and globally distributed static website hosting solution on Amazon Web Services (AWS). By leveraging Infrastructure as Code (IaC) with Terraform and Configuration Management with Ansible, the project achieves a highly reproducible deployment pipeline across multiple environments (Dev, Test, Prod).

The core solution utilizes Amazon S3 for durable object storage and Amazon CloudFront for low-latency content delivery. Security is a primary focus, implemented through Origin Access Control (OAC) to ensure the S3 bucket remains private and accessible only via the Content Delivery Network (CDN). The result is a professional-grade hosting architecture that minimizes manual intervention, reduces human error, and optimizes performance for global users.

2. Architecture and Design

The architecture is built on a "Secured Content Delivery" model, separating the storage layer from the distribution layer. The design is modular, ensuring that each component can be scaled or updated independently.

Core Components

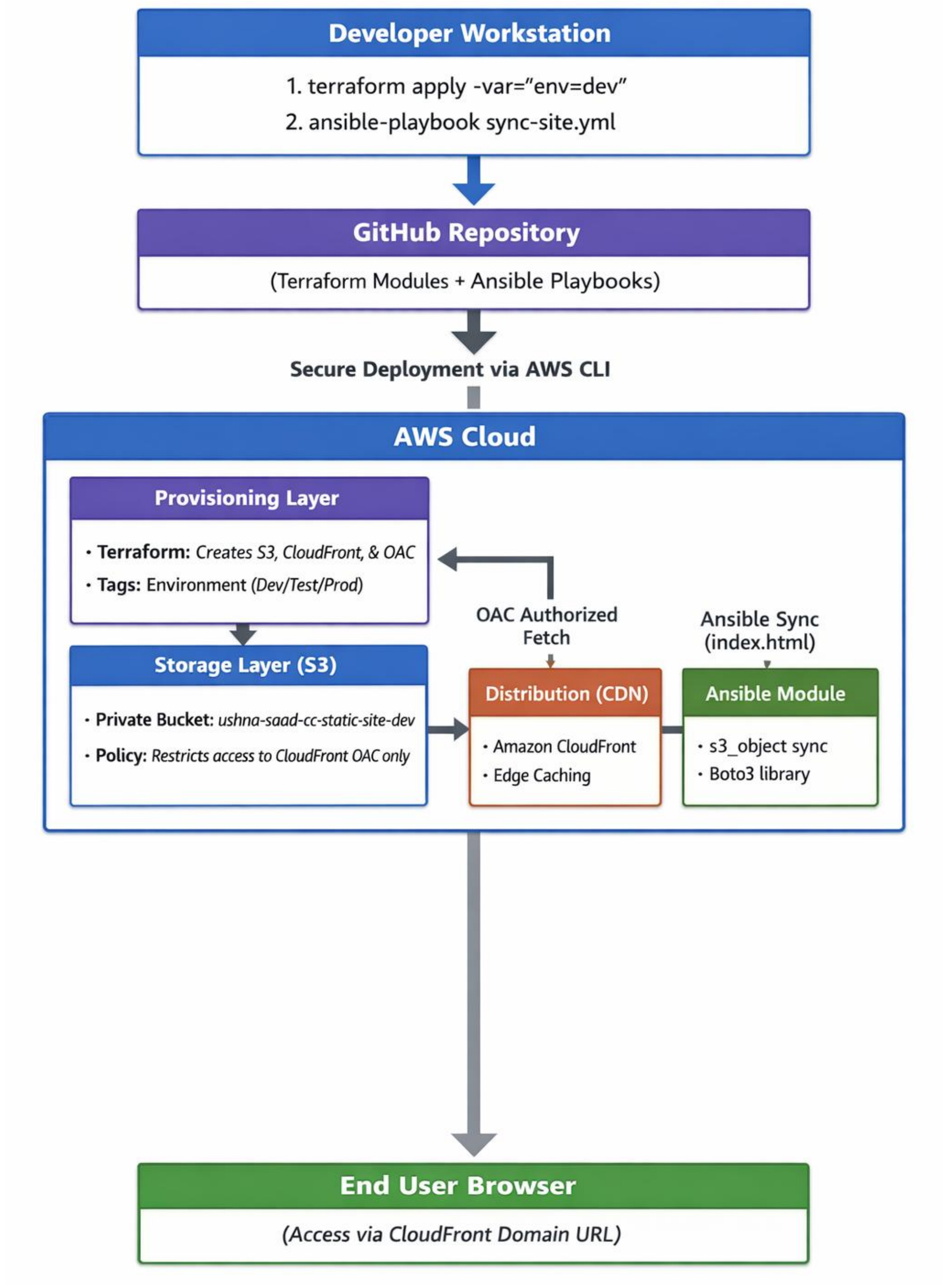
- **Storage Layer (Amazon S3):** Acts as the origin server. It is configured to block all public access, ensuring that the source files are not exposed directly to the internet.
- **Distribution Layer (Amazon CloudFront)**: A global CDN that caches content at edge locations. This reduces the load on the S3 bucket and provides faster load times for users by serving content from the nearest geographic location.
- **Security Layer (Origin Access Control - OAC):** This acts as the "virtual key." CloudFront uses OAC to sign requests to S3, proving to the bucket that the request is authorized.
- **Automation Layer:**

Terraform: Handles the "Provisioning." It creates the buckets, identity policies, and CDN distributions.

Ansible: Handles the "Deployment." It synchronizes the local index.html to the AWS cloud environment.

Multi-Environment Logic

The design employs a Variable-Driven Approach. Instead of hardcoding resource names, the architecture uses an env variable (e.g., dev, test, prod). This allows the same codebase to generate three identical but isolated environments, ensuring that the production site behaves exactly like the development site.


Developer Workstation
1. terraform apply -var="env=dev"
2. ansible-playbook sync-site.yml
GitHub Repository
(Terraform Modules + Ansible Playbooks)
Secure Deployment via AWS CLI
AWS Cloud
Provisioning Layer
• Terraform: Creates S3, CloudFront, & OAC
• Tags: Environment (Dev/Test/Prod)
Storage Layer (S3)
• Private Bucket: ushna-saad-cc-static-site-dev
• Policy: Restricts access to CloudFront OAC only
OAC Authorized Fetch
Distribution (CDN)
• Amazon CloudFront
• Edge Caching
Ansible Sync (index.html)
Ansible Module
• s3_object sync
• Boto3 library
End User Browser
(Access via CloudFront Domain URL)

3. Implementation details

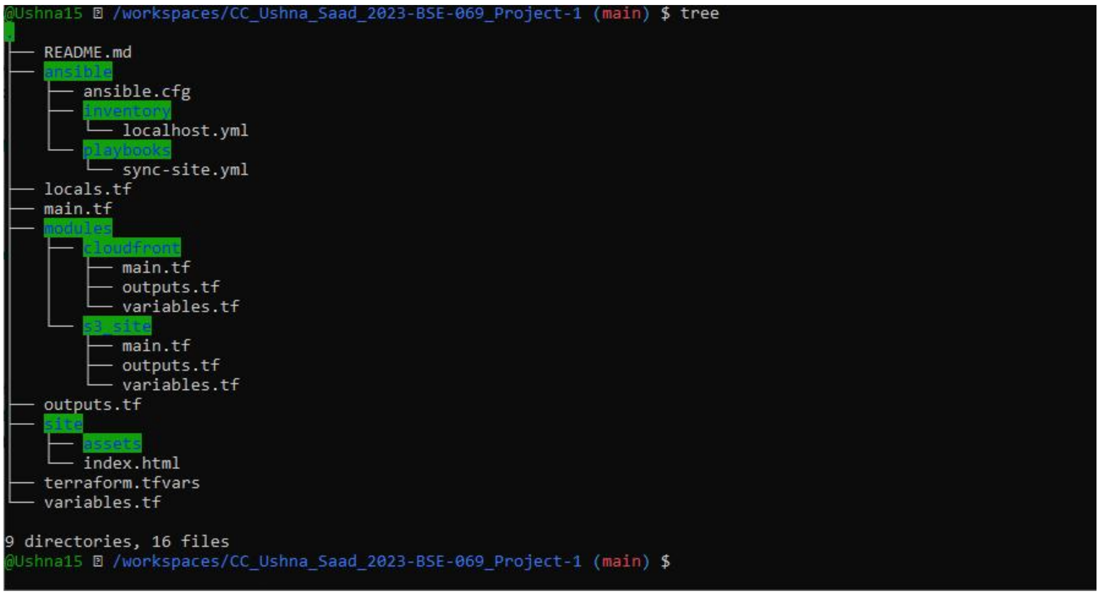
Part 1: Terraform Infrastructure Setup

This part focuses on setting up a well-structured and secure Terraform infrastructure for hosting a static website on AWS. The project is organized using a modular directory layout to improve readability, reusability, and scalability across different environments.

Terraform variables and local values are defined to enforce consistent naming, tagging, and environment-specific configurations. A reusable S3 module is implemented to provision a private S3 bucket per environment, with public access fully blocked in accordance with AWS security best practices.Overall, this part establishes a clean and secure Infrastructure as Code foundation that supports multi-environment deployments and prepares the project for further integration with CloudFront and Ansible in later stages.

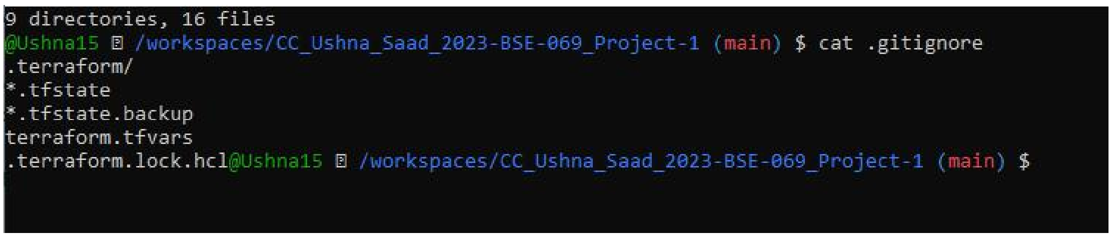
1.1 Project Structure

Create all necessary files and directories:



```
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ tree
.
├── README.md
├── ansible
│   ├── ansible.cfg
│   ├── inventory
│   │   └── localhost.yml
│   └── playbooks
│       └── sync-site.yml
├── locals.tf
├── main.tf
├── modules
│   ├── cloudfront
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   └── variables.tf
│   └── s3_site
│       ├── main.tf
│       ├── outputs.tf
│       └── variables.tf
├── outputs.tf
├── site
│   ├── assets
│   └── index.html
├── terraform.tfvars
└── variables.tf

9 directories, 16 files
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

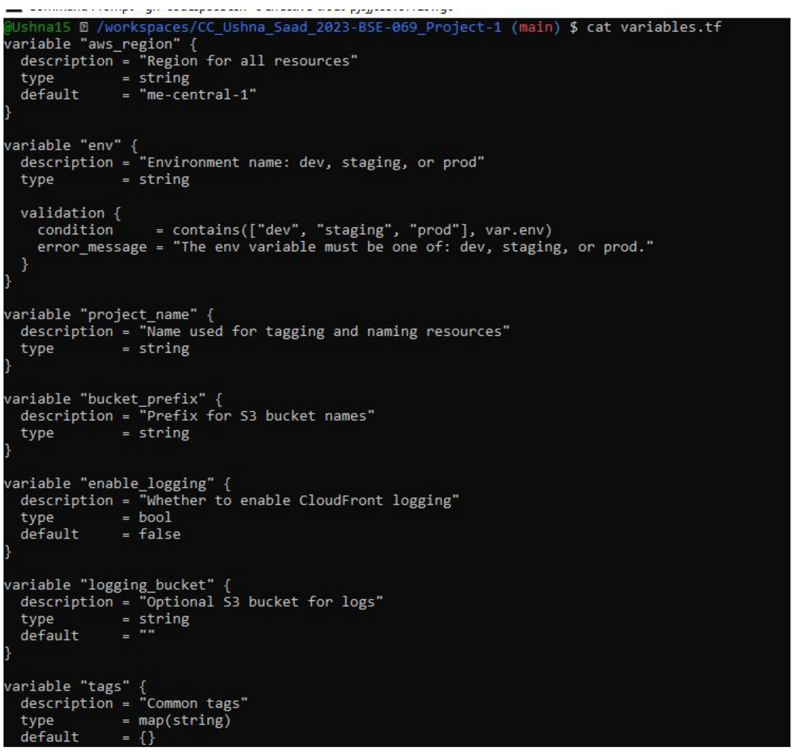
Implement proper .gitignore to exclude:



```
9 directories, 16 files
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat .gitignore
.terraform/
*.tfstate
*.tfstate.backup
terraform.tfvars
.terraform.lock.hcl@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

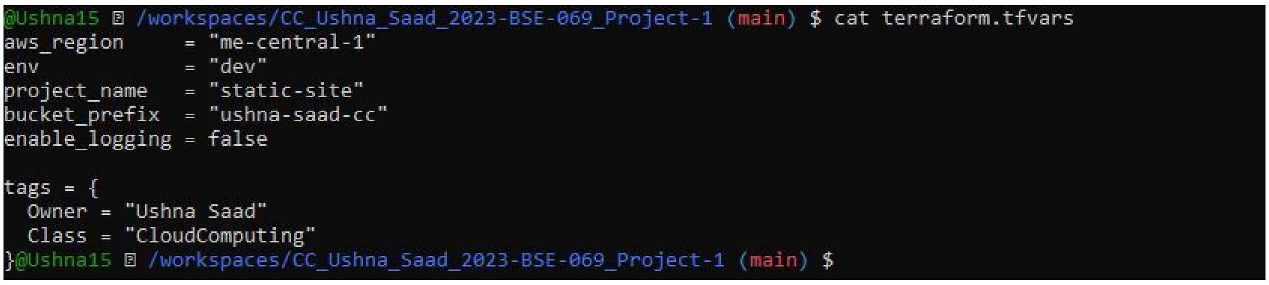
1.2 Variable Configuration

Variables.tf



```
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat variables.tf
variable "aws_region" {
  description = "Region for all resources"
  type        = string
  default     = "me-central-1"
}

variable "env" {
  description = "Environment name: dev, staging, or prod"
  type        = string

  validation {
    condition     = contains(["dev", "staging", "prod"], var.env)
    error_message = "The env variable must be one of: dev, staging, or prod."
  }
}

variable "project_name" {
  description = "Name used for tagging and naming resources"
  type        = string
}

variable "bucket_prefix" {
  description = "Prefix for S3 bucket names"
  type        = string
}

variable "enable_logging" {
  description = "Whether to enable CloudFront logging"
  type        = bool
  default     = false
}

variable "logging_bucket" {
  description = "Optional S3 bucket for logs"
  type        = string
  default     = ""
}

variable "tags" {
  description = "Common tags"
  type        = map(string)
  default     = {}
```

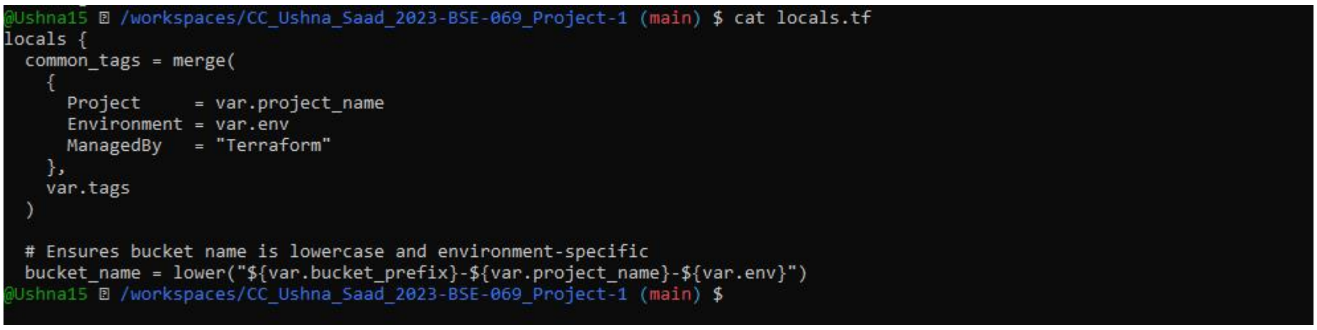
terraform.tfvars



```
@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat terraform.tfvars
aws_region     = "me-central-1"
env            = "dev"
project_name   = "static-site"
bucket_prefix  = "ushna-saad-cc"
enable_logging = false

tags = {
  Owner = "Ushna Saad"
  Class = "CloudComputing"
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

1.3 Locals Configuration

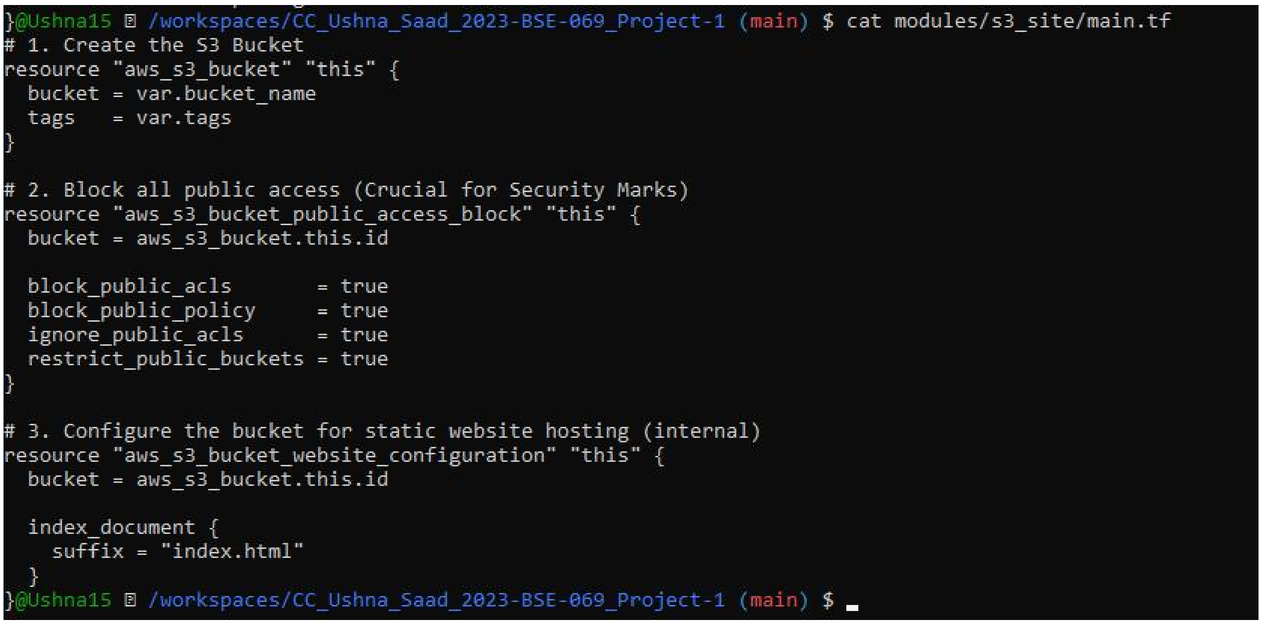
locals.tf



```
@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat locals.tf
locals {
  common_tags = merge(
    {
      Project     = var.project_name
      Environment = var.env
      ManagedBy   = "Terraform"
    },
    var.tags
  )

  # Ensures bucket name is lowercase and environment-specific
  bucket_name = lower("${var.bucket_prefix}-${var.project_name}-${var.env}")
@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

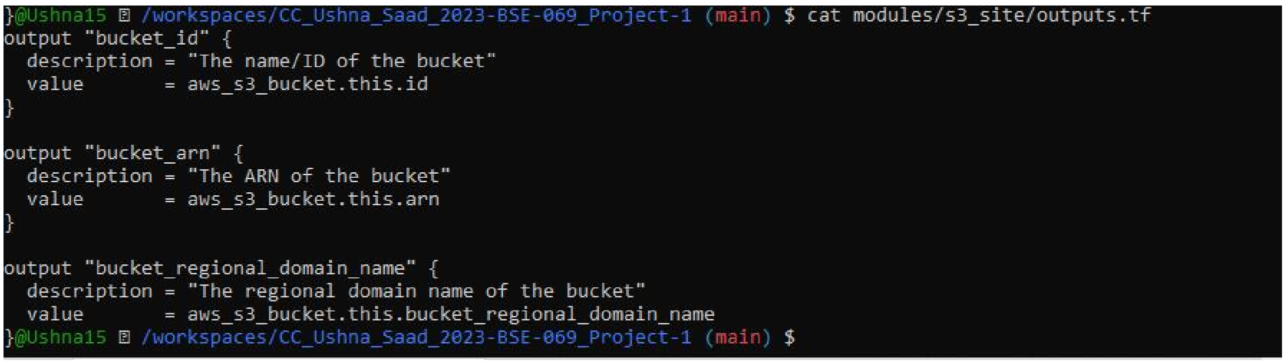
1.4 S3 Site Module

S3 Module main.tf



```
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat modules/s3_site/main.tf
# 1. Create the S3 Bucket
resource "aws_s3_bucket" "this" {
  bucket = var.bucket_name
  tags   = var.tags
}

# 2. Block all public access (Crucial for Security Marks)
resource "aws_s3_bucket_public_access_block" "this" {
  bucket = aws_s3_bucket.this.id

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

# 3. Configure the bucket for static website hosting (internal)
resource "aws_s3_bucket_website_configuration" "this" {
  bucket = aws_s3_bucket.this.id

  index_document {
    suffix = "index.html"
  }
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ _
```

S3 Module outputs.tf


```
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat modules/s3_site/outputs.tf
output "bucket_id" {
  description = "The name/ID of the bucket"
  value       = aws_s3_bucket.this.id
}

output "bucket_arn" {
  description = "The ARN of the bucket"
  value       = aws_s3_bucket.this.arn
}

output "bucket_regional_domain_name" {
  description = "The regional domain name of the bucket"
  value       = aws_s3_bucket.this.bucket_regional_domain_name
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

Part 2: CloudFront Module

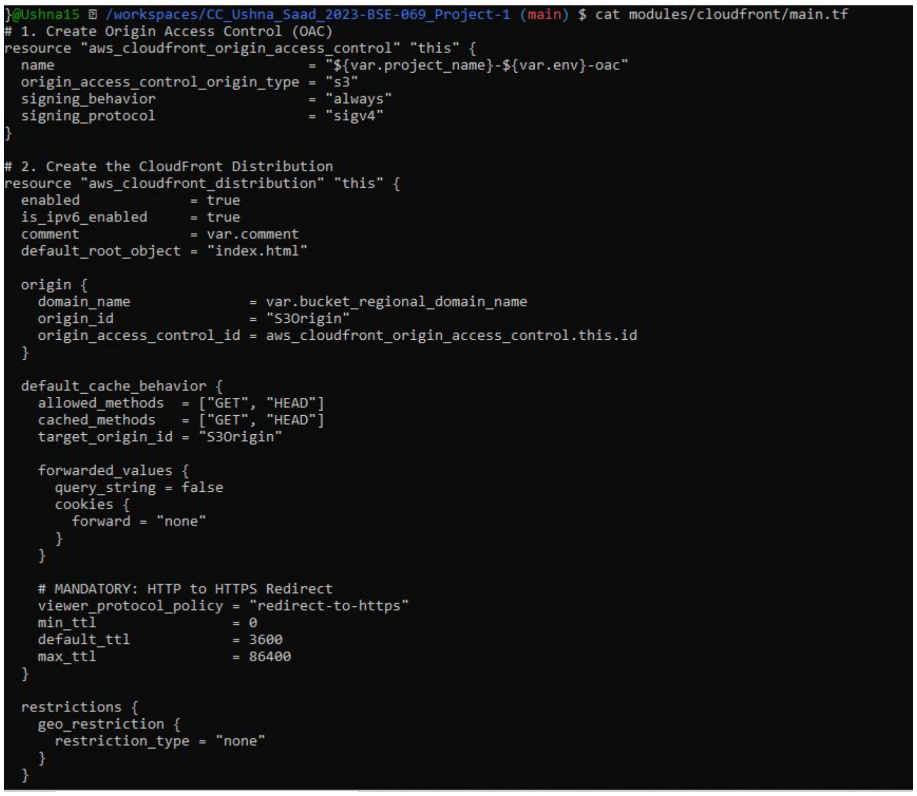
This part focuses on creating a reusable CloudFront module to securely deliver static website content stored in Amazon S3. The module is designed to be environment-independent, allowing the same configuration to support development, staging, and production deployments.

An Amazon CloudFront distribution is configured with the S3 bucket as its origin, enforcing HTTPS redirection, restricting allowed HTTP methods, and applying consistent tagging for traceability. Key outputs such as the distribution ID and domain name are exposed to enable seamless integration with other Terraform modules.

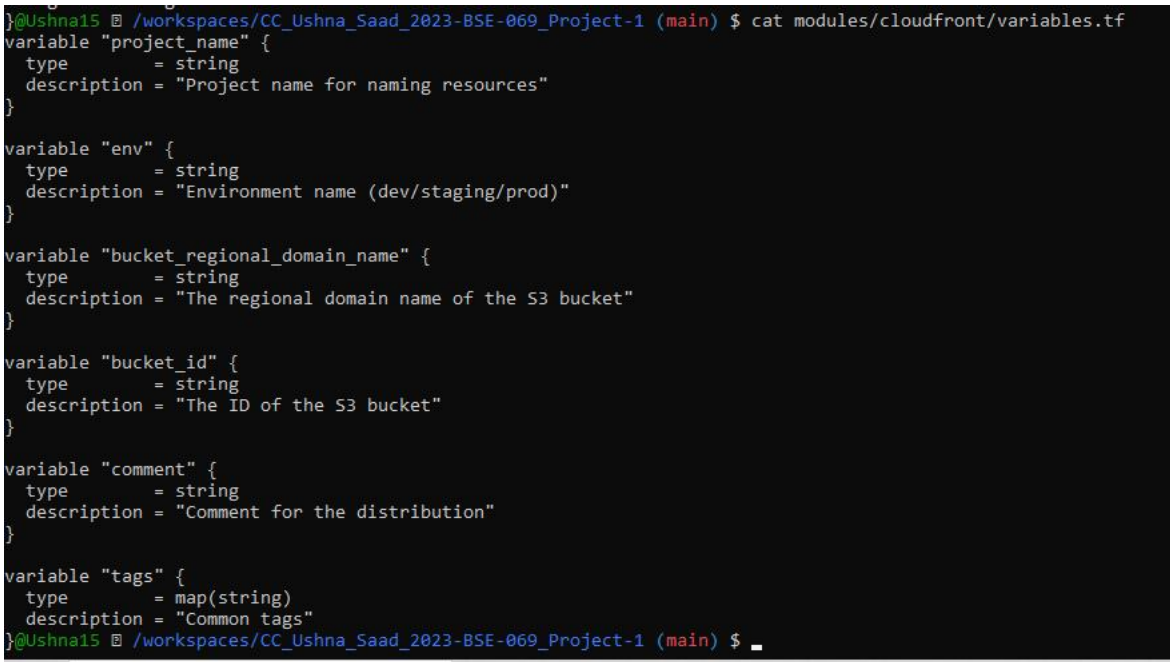
Overall, this implementation provides a secure and scalable content delivery layer, completing the connection between storage and global content distribution within the infrastructure.

2.1 Module Design

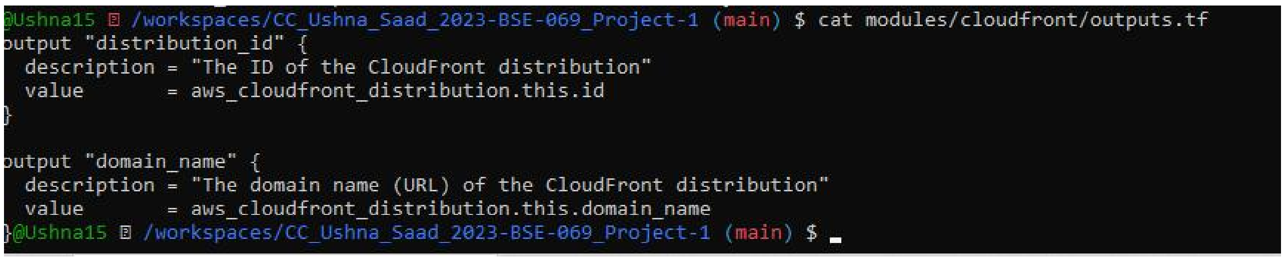
CloudFront Module main.tf


```
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat modules/cloudfront/main.tf
# 1. Create Origin Access Control (OAC)
resource "aws_cloudfront_origin_access_control" "this" {
  name                              = "${var.project_name}-${var.env}-oac"
  origin_access_control_origin_type = "s3"
  signing_behavior                  = "always"
  signing_protocol                  = "sigv4"
}

# 2. Create the CloudFront Distribution
resource "aws_cloudfront_distribution" "this" {
  enabled             = true
  is_ipv6_enabled     = true
  comment             = var.comment
  default_root_object = "index.html"

  origin {
    domain_name              = var.bucket_regional_domain_name
    origin_id                = "S3Origin"
    origin_access_control_id = aws_cloudfront_origin_access_control.this.id
  }

  default_cache_behavior {
    allowed_methods  = ["GET", "HEAD"]
    cached_methods   = ["GET", "HEAD"]
    target_origin_id = "S3Origin"

    forwarded_values {
      query_string = false
      cookies {
        forward = "none"
      }
    }

    # MANDATORY: HTTP to HTTPS Redirect
    viewer_protocol_policy = "redirect-to-https"
    min_ttl                = 0
    default_ttl            = 3600
    max_ttl                = 86400
  }

  restrictions {
    geo_restriction {
      restriction_type = "none"
    }
  }
```

CloudFront Module variables.tf

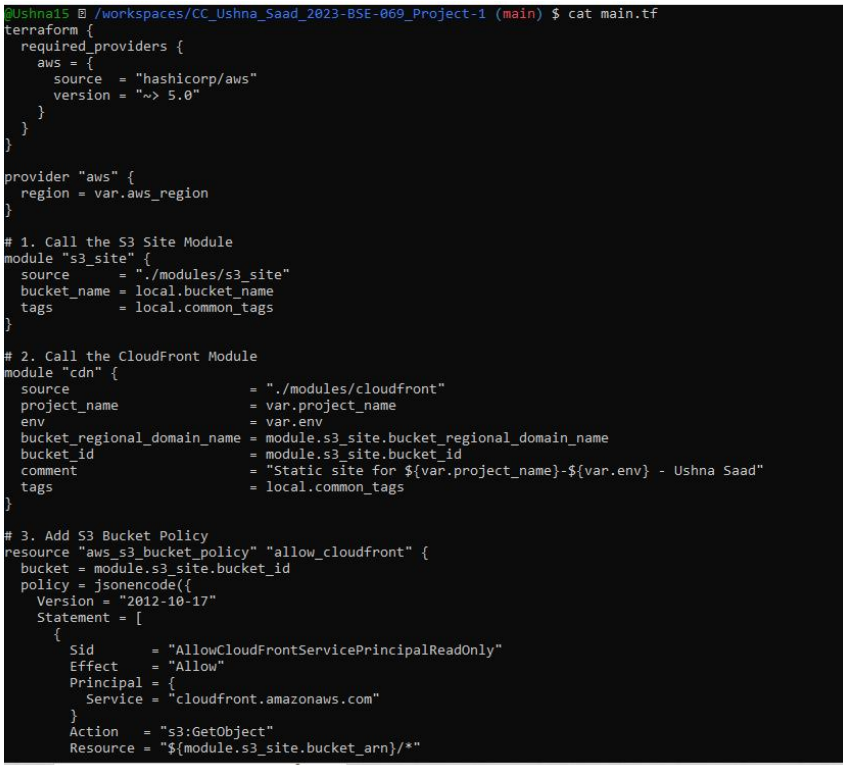

```
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat modules/cloudfront/variables.tf
variable "project_name" {
  type        = string
  description = "Project name for naming resources"
}

variable "env" {
  type        = string
  description = "Environment name (dev/staging/prod)"
}

variable "bucket_regional_domain_name" {
  type        = string
  description = "The regional domain name of the S3 bucket"
}

variable "bucket_id" {
  type        = string
  description = "The ID of the S3 bucket"
}

variable "comment" {
  type        = string
  description = "Comment for the distribution"
}

variable "tags" {
  type        = map(string)
  description = "Common tags"
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

CloudFront Module outputs.tf


```
@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat modules/cloudfront/outputs.tf
output "distribution_id" {
  description = "The ID of the CloudFront distribution"
  value       = aws_cloudfront_distribution.this.id
}

output "domain_name" {
  description = "The domain name (URL) of the CloudFront distribution"
  value       = aws_cloudfront_distribution.this.domain_name
}@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

2.2 Module Usage

Main.tf


```
@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ cat main.tf
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = var.aws_region
}

# 1. Call the S3 Site Module
module "s3_site" {
  source      = "./modules/s3_site"
  bucket_name = local.bucket_name
  tags        = local.common_tags
}

# 2. Call the CloudFront Module
module "cdn" {
  source                      = "./modules/cloudfront"
  project_name                = var.project_name
  env                         = var.env
  bucket_regional_domain_name = module.s3_site.bucket_regional_domain_name
  bucket_id                   = module.s3_site.bucket_id
  comment                     = "Static site for ${var.project_name}-${var.env} - Ushna Saad"
  tags                        = local.common_tags
}

# 3. Add S3 Bucket Policy
resource "aws_s3_bucket_policy" "allow_cloudfront" {
  bucket = module.s3_site.bucket_id
  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Sid       = "AllowCloudFrontServicePrincipalReadOnly"
        Effect    = "Allow"
        Principal = {
          Service = "cloudfront.amazonaws.com"
        }
        Action   = "s3:GetObject"
        Resource = "${module.s3_site.bucket_arn}/*"
```

Part 3: Ansible Content Deployment

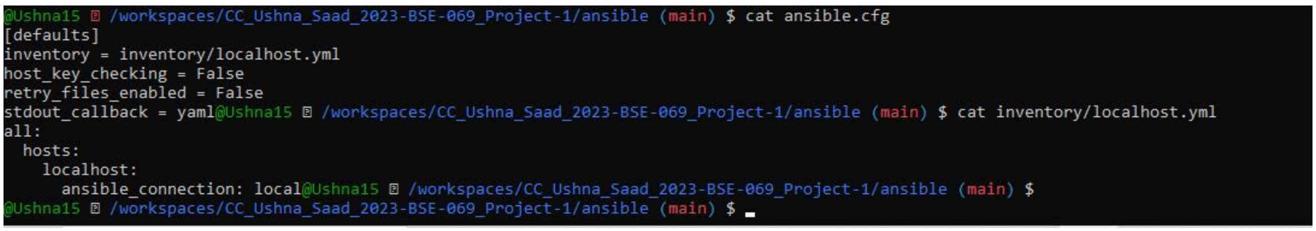
This part focuses on automating the deployment of static website content to Amazon S3 using Ansible. Ansible is configured to run locally, allowing content synchronization without managing remote hosts or SSH connections.

A dedicated playbook is implemented to idempotently sync the website files from the local site/ directory to the target S3 bucket using the amazon.aws.s3_sync module. The bucket name and AWS region are supplied dynamically through environment variables or extra variables, ensuring flexibility across environments.

Overall, this setup enables a repeatable and automated content deployment process, completing the infrastructure workflow by linking Terraform-provisioned resources with configuration management.

1.1 Ansible Setup

ansible.cfg

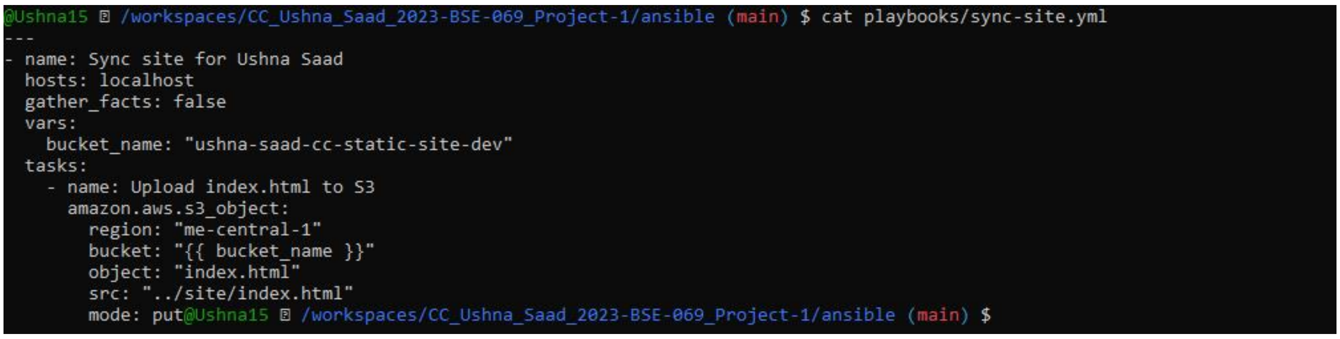


```
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ cat ansible.cfg
[defaults]
inventory = inventory/localhost.yml
host_key_checking = False
retry_files_enabled = False
stdout_callback = yaml@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ cat inventory/localhost.yml
all:
  hosts:
    localhost:
      ansible_connection: local@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ _
```

inventory list

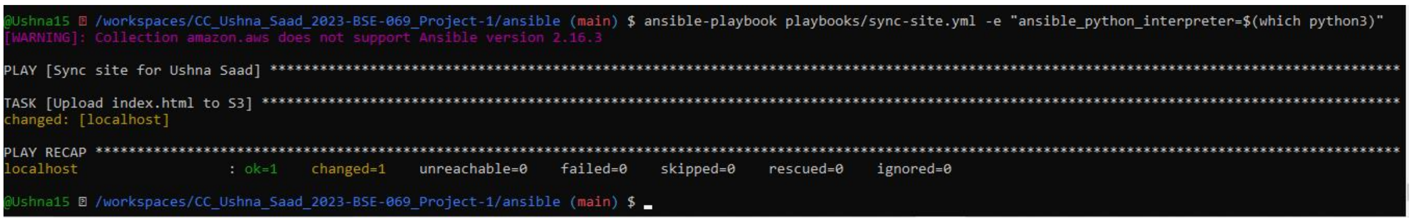
3.2 sync-site.yml Playbook

sync-site.yml



```
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ cat playbooks/sync-site.yml
---
- name: Sync site for Ushna Saad
  hosts: localhost
  gather_facts: false
  vars:
    bucket_name: "ushna-saad-cc-static-site-dev"
  tasks:
    - name: Upload index.html to S3
      amazon.aws.s3_object:
        region: "me-central-1"
        bucket: "{{ bucket_name }}"
        object: "index.html"
        src: "../site/index.html"
        mode: put@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $
```

successful playbook run



```
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ ansible-playbook playbooks/sync-site.yml -e "ansible_python_interpreter=$(which python3)"
[WARNING]: Collection amazon.aws does not support Ansible version 2.16.3

PLAY [Sync site for Ushna Saad] ***********************************************************************************************************************

TASK [Upload index.html to S3] ************************************************************************************************************************
changed: [localhost]

PLAY RECAP ********************************************************************************************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ _
```

Part 4: Deployment & Testing

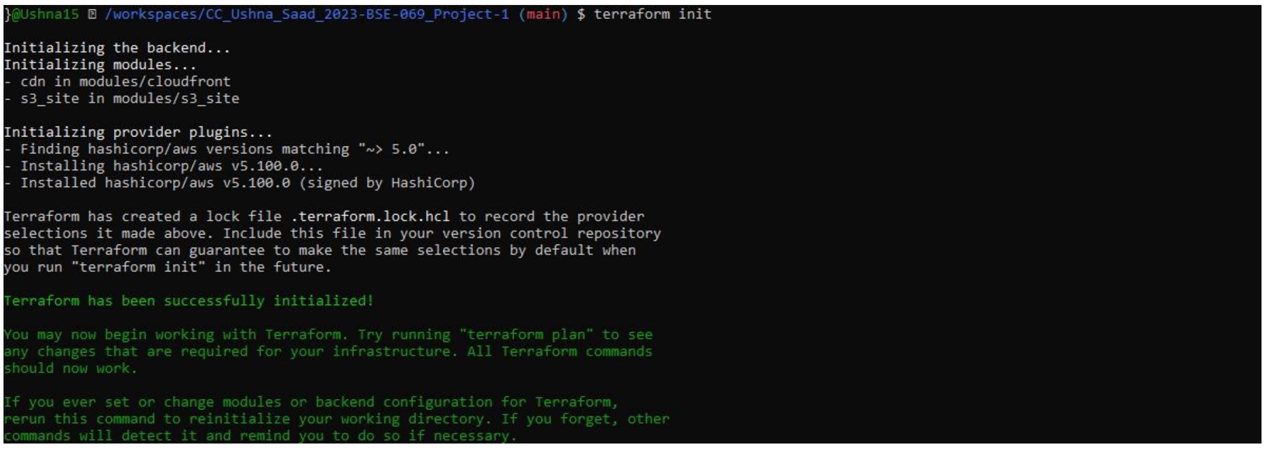
This part focuses on deploying and validating the complete infrastructure and content delivery pipeline. Terraform is used to initialize, validate, plan, and apply the infrastructure for a selected environment, ensuring that all resources are created successfully and without configuration errors.

Terraform outputs are then used to retrieve key deployment details such as the S3 bucket name and CloudFront distribution URL. These outputs are exported as environment variables and consumed by Ansible to automate the upload of static website content to the provisioned S3 bucket.

Finally, the deployment is verified by accessing the website through the CloudFront domain, confirming that the content is served correctly over HTTPS. This step validates the end-to-end integration of Terraform, Ansible, S3, and CloudFront.

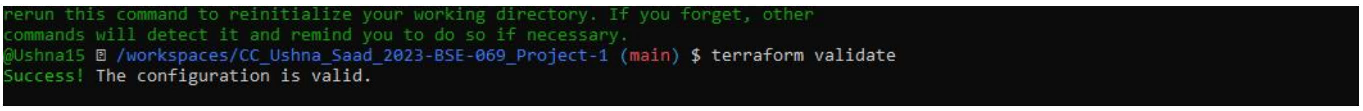
4.1 Terraform Deployment

terraform init



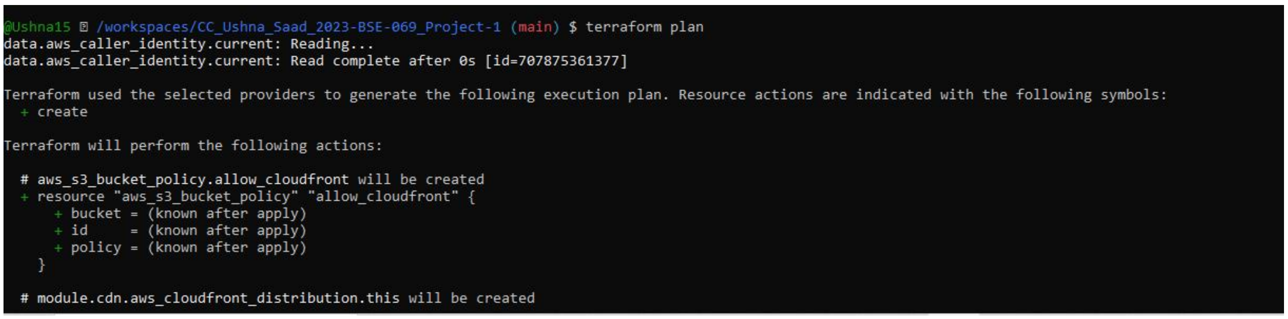
```
}@Ushna15 □ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ terraform init

Initializing the backend...
Initializing modules...
- cdn in modules/cloudfront
- s3_site in modules/s3_site

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.100.0...
- Installed hashicorp/aws v5.100.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

terraform validate



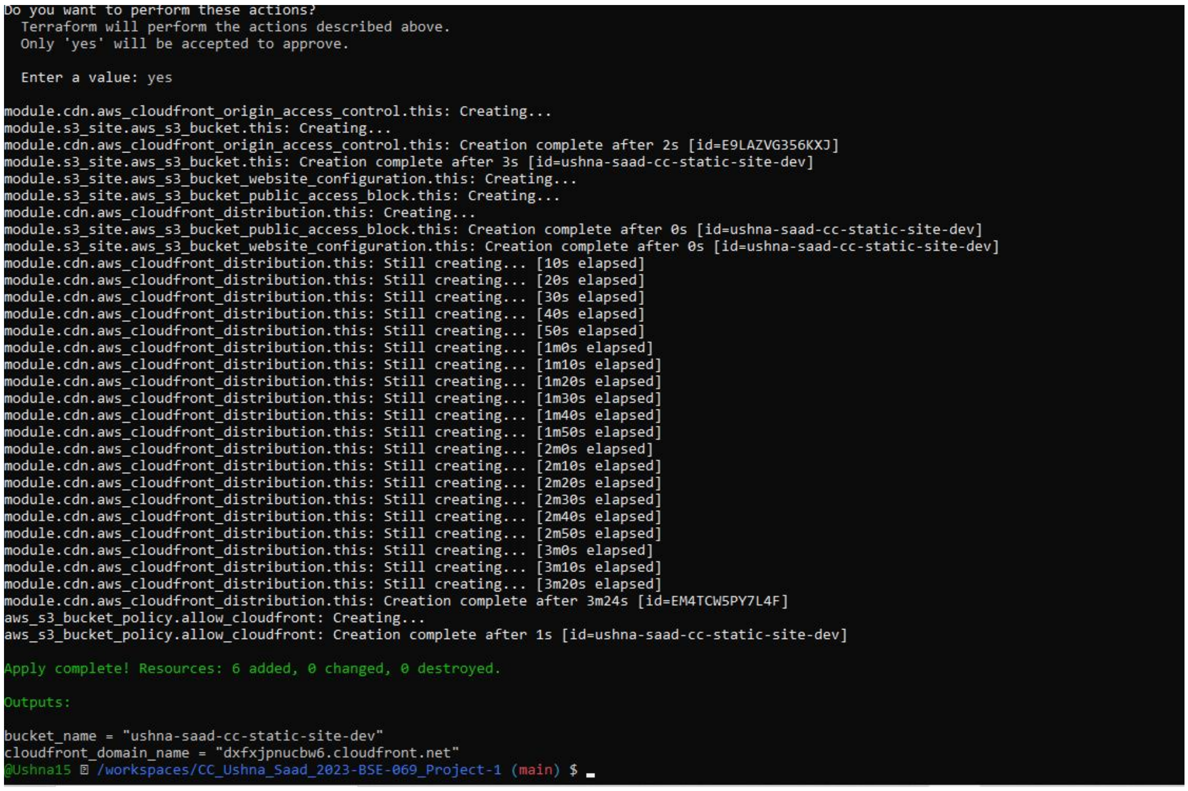
```
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
@Ushna15 □ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ terraform validate
Success! The configuration is valid.
```

terraform plan



```
@Ushna15 □ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ terraform plan
data.aws_caller_identity.current: Reading...
data.aws_caller_identity.current: Read complete after 0s [id=707875361377]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket_policy.allow_cloudfront will be created
  + resource "aws_s3_bucket_policy" "allow_cloudfront" {
      + bucket = (known after apply)
      + id     = (known after apply)
      + policy = (known after apply)
    }

  # module.cdn.aws_cloudfront_distribution.this will be created
```

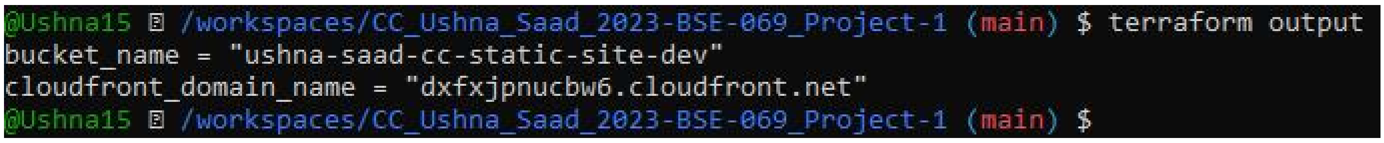
terraform apply



```
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

module.cdn.aws_cloudfront_origin_access_control.this: Creating...
module.s3_site.aws_s3_bucket.this: Creating...
module.cdn.aws_cloudfront_origin_access_control.this: Creation complete after 2s [id=E9LAZVG356KXJ]
module.s3_site.aws_s3_bucket.this: Creation complete after 3s [id=ushna-saad-cc-static-site-dev]
module.s3_site.aws_s3_bucket_website_configuration.this: Creating...
module.s3_site.aws_s3_bucket_public_access_block.this: Creating...
module.cdn.aws_cloudfront_distribution.this: Creating...
module.s3_site.aws_s3_bucket_public_access_block.this: Creation complete after 0s [id=ushna-saad-cc-static-site-dev]
module.s3_site.aws_s3_bucket_website_configuration.this: Creation complete after 0s [id=ushna-saad-cc-static-site-dev]
module.cdn.aws_cloudfront_distribution.this: Still creating... [10s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [20s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [30s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [40s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [50s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [1m0s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [1m10s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [1m20s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [1m30s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [1m40s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [1m50s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [2m0s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [2m10s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [2m20s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [2m30s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [2m40s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [2m50s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [3m0s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [3m10s elapsed]
module.cdn.aws_cloudfront_distribution.this: Still creating... [3m20s elapsed]
module.cdn.aws_cloudfront_distribution.this: Creation complete after 3m24s [id=EM4TCW5PY7L4F]
aws_s3_bucket_policy.allow_cloudfront: Creating...
aws_s3_bucket_policy.allow_cloudfront: Creation complete after 1s [id=ushna-saad-cc-static-site-dev]

Apply complete! Resources: 6 added, 0 changed, 0 destroyed.

Outputs:

bucket_name = "ushna-saad-cc-static-site-dev"
cloudfront_domain_name = "dxfxjpnucbw6.cloudfront.net"
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ 
```

4.2 Terraform Outputs

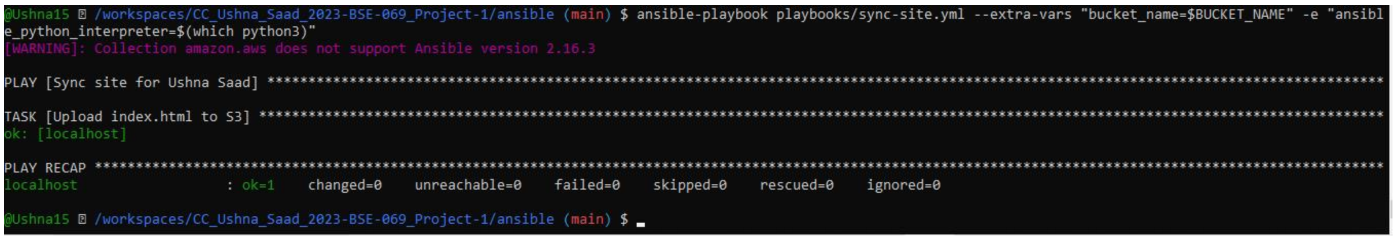
terraform output



```
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $ terraform output
bucket_name = "ushna-saad-cc-static-site-dev"
cloudfront_domain_name = "dxfxjpnucbw6.cloudfront.net"
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

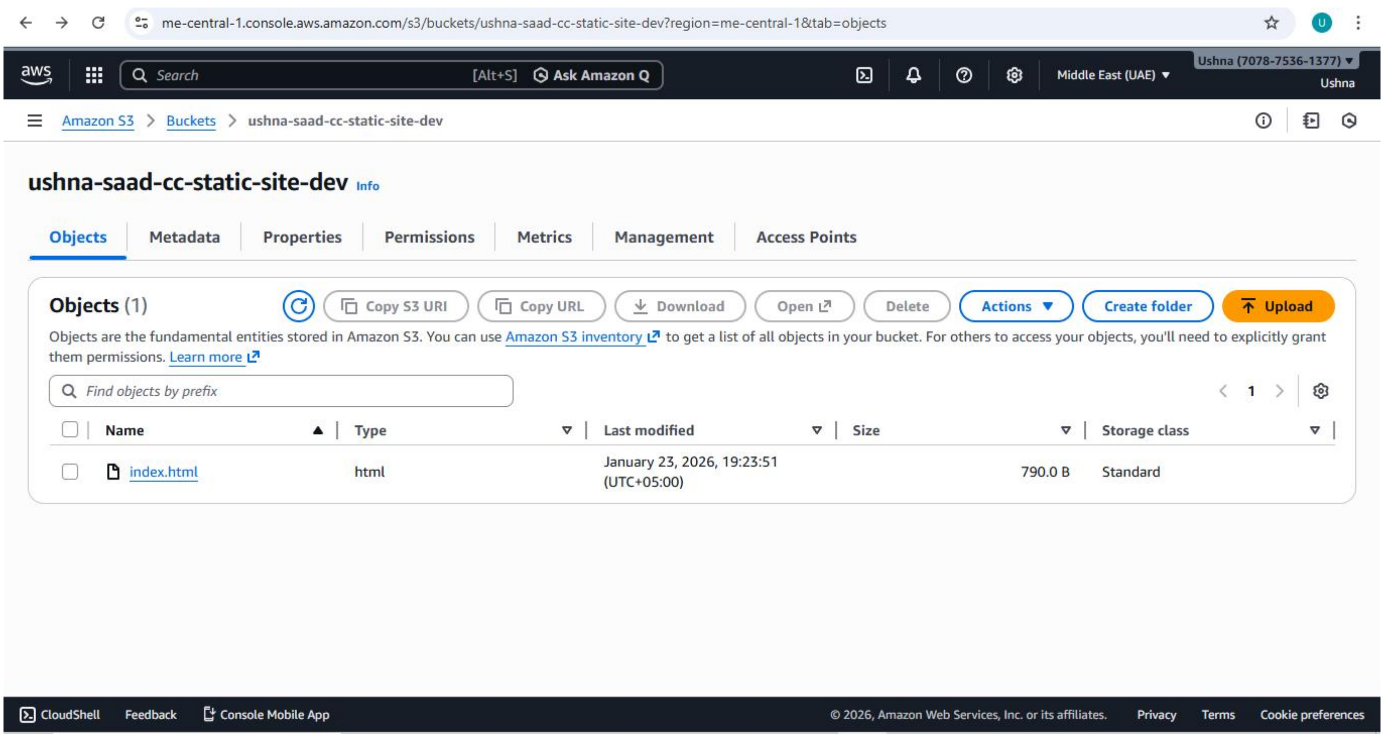
4.3 Ansible Deployment

Successful playbook run



```
@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ ansible-playbook playbooks/sync-site.yml --extra-vars "bucket_name=$BUCKET_NAME" -e "ansibl
e_python_interpreter=$(which python3)"
[WARNING]: Collection amazon.aws does not support Ansible version 2.16.3

PLAY [Sync site for Ushna Saad] **************************************************************************************************************************************

TASK [Upload index.html to S3] ***************************************************************************************************************************************
ok: [localhost]

PLAY RECAP ***********************************************************************************************************************************************************
localhost                  : ok=1    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

@Ushna15 ▯ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1/ansible (main) $ 
```

S3 bucket on console


me-central-1.console.aws.amazon.com/s3/buckets/ushna-saad-cc-static-site-dev?region=me-central-1&tab=objects

Amazon S3 > Buckets > ushna-saad-cc-static-site-dev

ushna-saad-cc-static-site-dev Info

Objects (1)

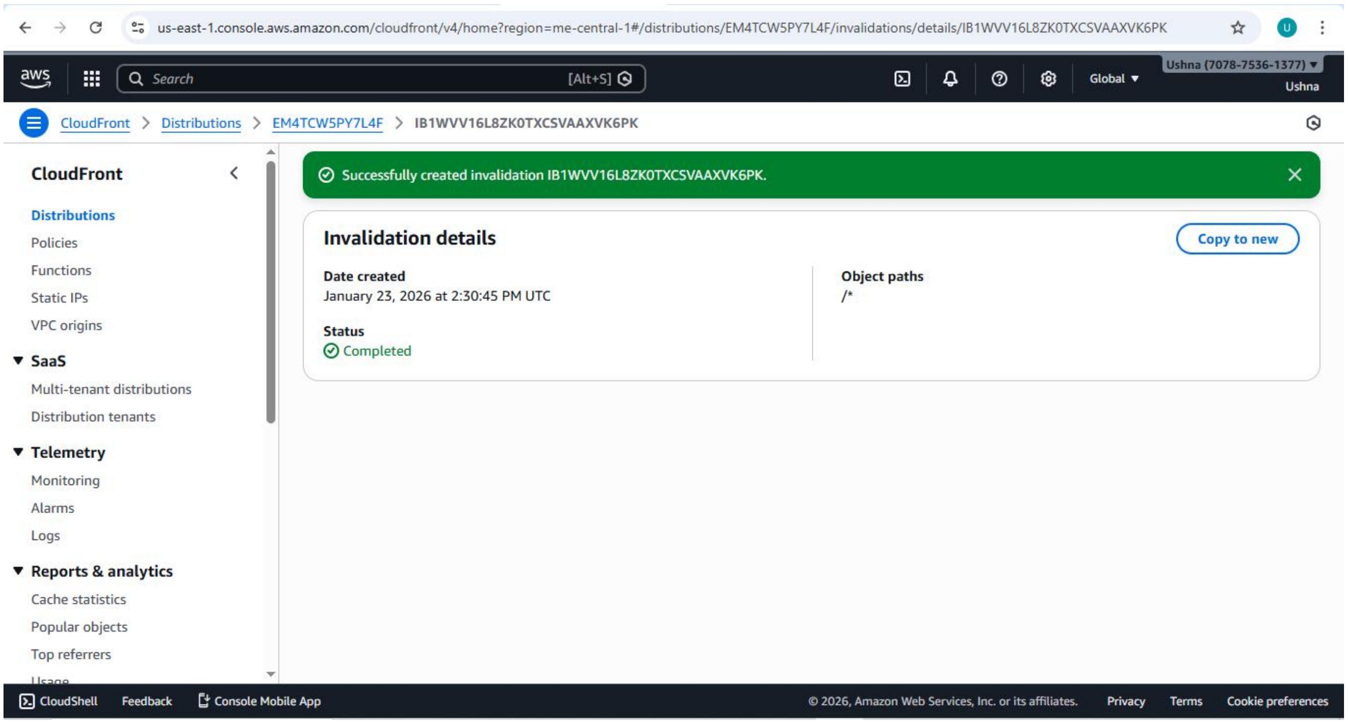
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
index.html	html	January 23, 2026, 19:23:51 (UTC+05:00)	790.0 B	Standard

© 2026, Amazon Web Services, Inc. or its affiliates.

4.4 Website Testing

During testing, the CloudFront URL initially displayed a blank page even though the S3 objects were successfully uploaded. This issue was resolved by updating the CloudFront distribution to include a wildcard path (/*) in the behavior configuration, ensuring that all objects from the S3 bucket were correctly served through the distribution. After this change, the website content became visible and accessible via the CloudFront domain.


us-east-1.console.aws.amazon.com/cloudfront/v4/home?region=me-central-1#/distributions/EM4TCW5PY7L4F/invalidations/details/IB1WVV16L8ZK0TXCSVAAXVK6PK

CloudFront > Distributions > EM4TCW5PY7L4F > IB1WVV16L8ZK0TXCSVAAXVK6PK

Successfully created invalidation IB1WVV16L8ZK0TXCSVAAXVK6PK.

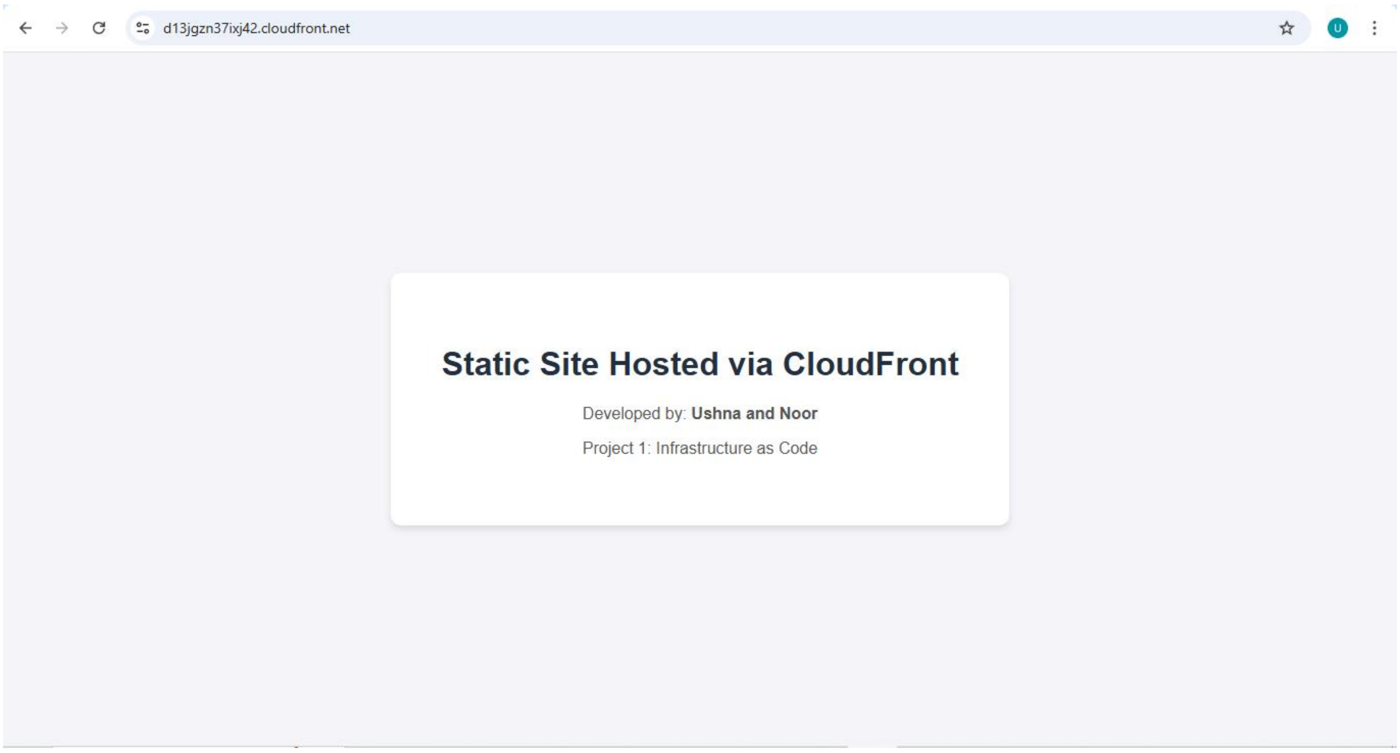
Invalidation details

Date created
January 23, 2026 at 2:30:45 PM UTC

Status
Completed

Object paths
/*

Cloudfront testing on browser



Static Site Hosted via CloudFront

Developed by: **Ushna and Noor**

Project 1: Infrastructure as Code

Part 5: Security & Cleanup

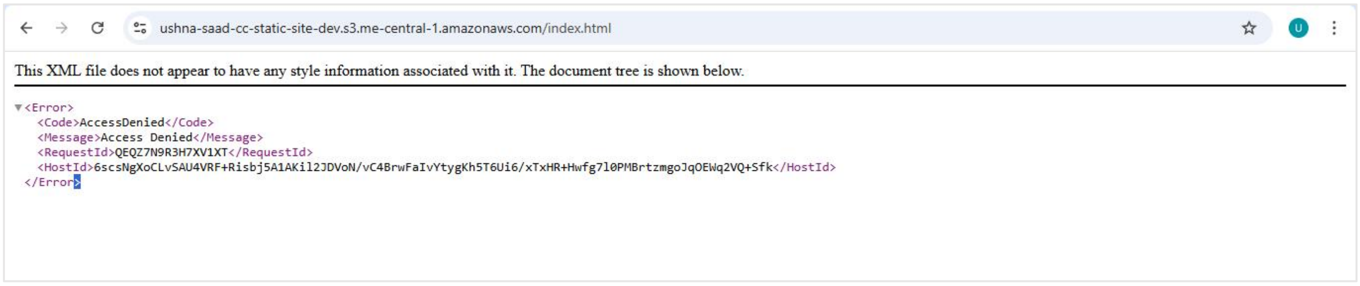
This step focuses on validating the security posture of the deployed infrastructure and ensuring proper resource cleanup. Access controls are verified by confirming that the S3 bucket denies direct public access while allowing content delivery exclusively through CloudFront.

Comprehensive project documentation is provided through a structured README.md, covering setup, usage, testing, troubleshooting, and cleanup procedures. This ensures the project is easy to understand, deploy, and maintain.

Finally, all AWS resources are safely removed using Terraform destroy commands, confirming that the environment can be cleanly decommissioned without leaving residual infrastructure or incurring unnecessary costs.

5.1 Bucket Access Verification

accessing an object via S3 url

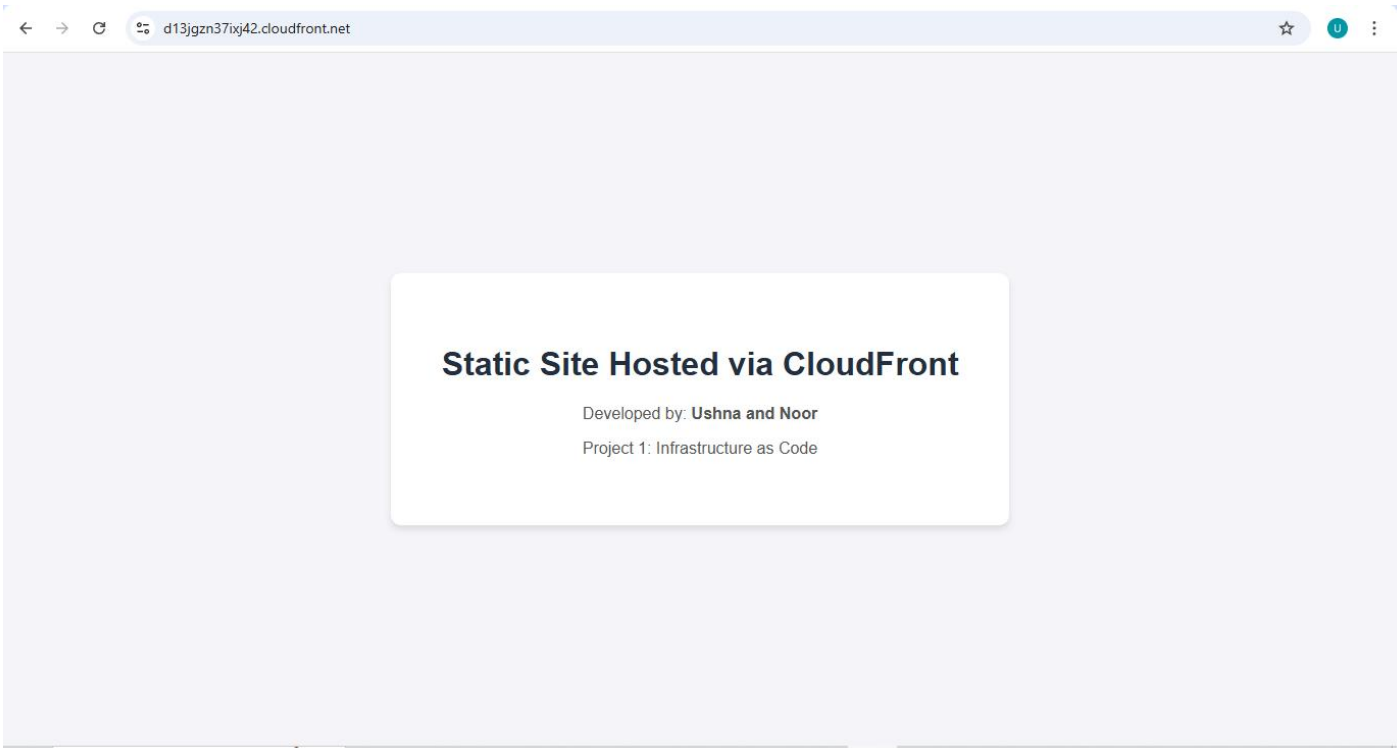


ushna-saad-cc-static-site-dev.s3.me-central-1.amazonaws.com/index.html

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Error>
   <Code>AccessDenied</Code>
   <Message>Access Denied</Message>
   <RequestId>QEQZ7N9R3H7XV1XT</RequestId>
   <HostId>6scsNgXoCLvSAU4VRF+Risbj5A1AKil2JDVoN/vC4BrwFaIvYtygKh5T6Ui6/xTxHR+Hwfg7l0PMBrtzmgoJqOEWq2VQ+Sfk</HostId>
 </Error>
```

accessing via CloudFront url

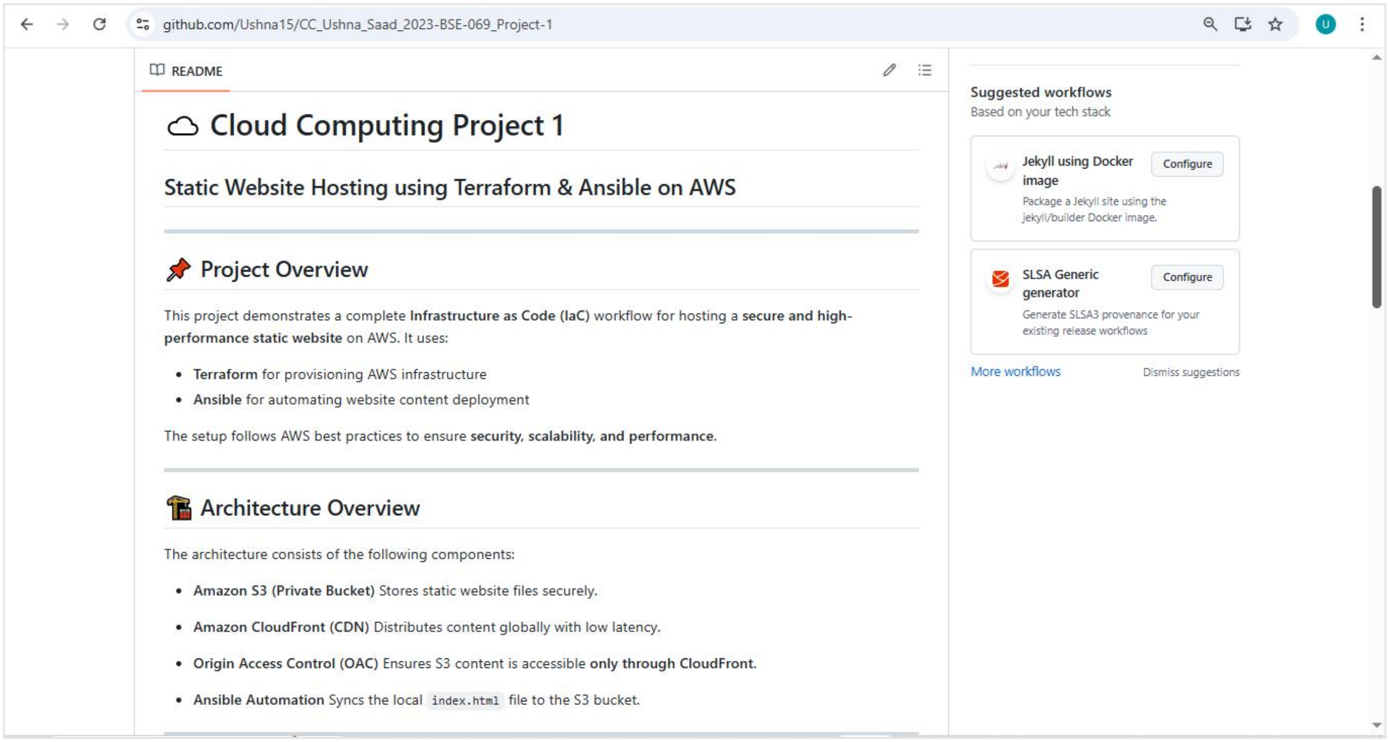

d13jgzn37ixj42.cloudfront.net

Static Site Hosted via CloudFront

Developed by: **Ushna and Noor**

Project 1: Infrastructure as Code

5.2 README Documentation


github.com/Ushna15/CC_Ushna_Saad_2023-BSE-069_Project-1

README

☁ Cloud Computing Project 1

Static Website Hosting using Terraform & Ansible on AWS

📌 Project Overview

This project demonstrates a complete **Infrastructure as Code (IaC)** workflow for hosting a **secure and high-performance static website** on AWS. It uses:

- **Terraform** for provisioning AWS infrastructure
- **Ansible** for automating website content deployment

The setup follows AWS best practices to ensure **security, scalability, and performance.**

🏗 Architecture Overview

The architecture consists of the following components:

- **Amazon S3 (Private Bucket)** Stores static website files securely.
- **Amazon CloudFront (CDN)** Distributes content globally with low latency.
- **Origin Access Control (OAC)** Ensures S3 content is accessible **only through CloudFront.**
- **Ansible Automation** Syncs the local `index.html` file to the S3 bucket.

Suggested workflows
Based on your tech stack

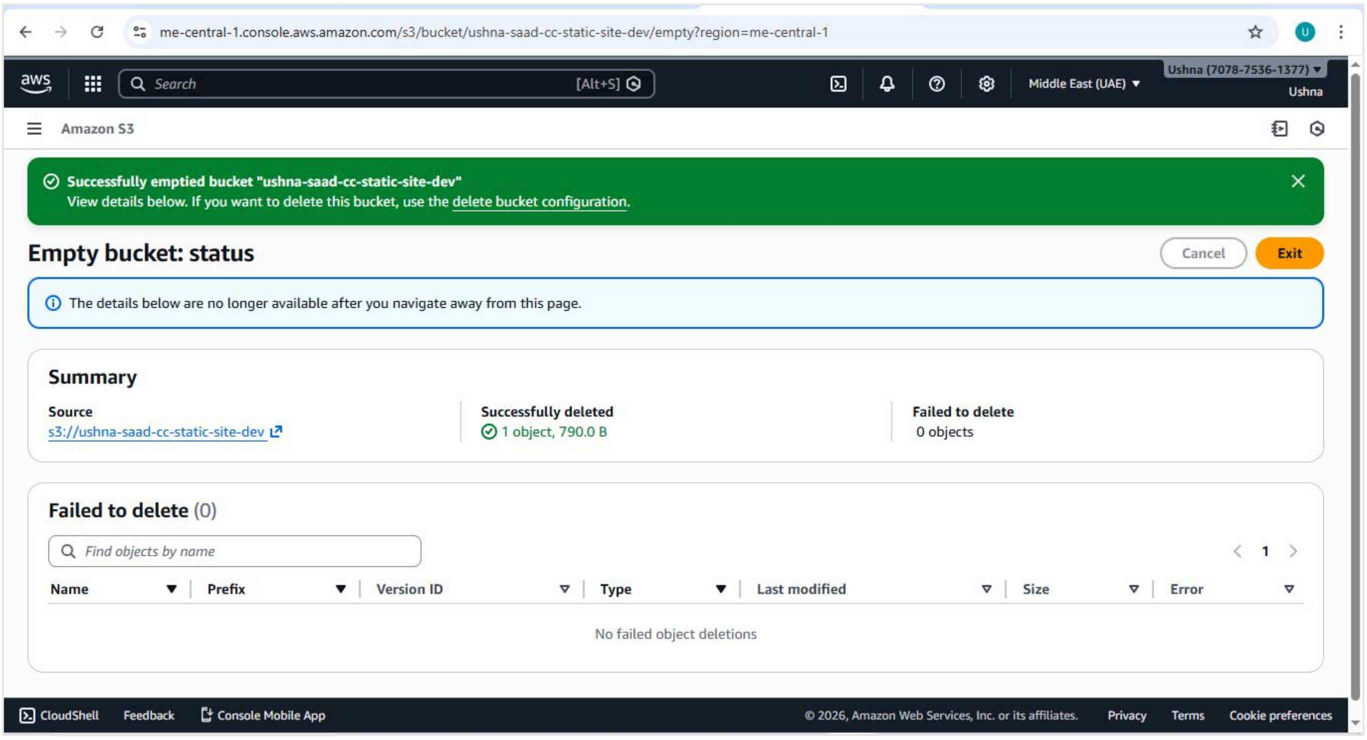
Jekyll using Docker image — Configure
Package a Jekyll site using the jekyll/builder Docker image.

SLSA Generic generator — Configure
Generate SLSA3 provenance for your existing release workflows

More workflows — Dismiss suggestions

5.3 Cleanup

During the cleanup phase, the S3 bucket was not deleted automatically when running terraform destroy because it still contained objects. Since Amazon S3 does not allow the deletion of non-empty buckets by default, the bucket contents were manually removed from the AWS Console. After emptying the bucket, the terraform destroy command was re-executed and completed successfully, removing all remaining resources.


me-central-1.console.aws.amazon.com/s3/bucket/ushna-saad-cc-static-site-dev/empty?region=me-central-1

Amazon S3

Successfully emptied bucket "ushna-saad-cc-static-site-dev"
View details below. If you want to delete this bucket, use the delete bucket configuration.

Empty bucket: status

The details below are no longer available after you navigate away from this page.

Summary

Source	Successfully deleted	Failed to delete
s3://ushna-saad-cc-static-site-dev	1 object, 790.0 B	0 objects

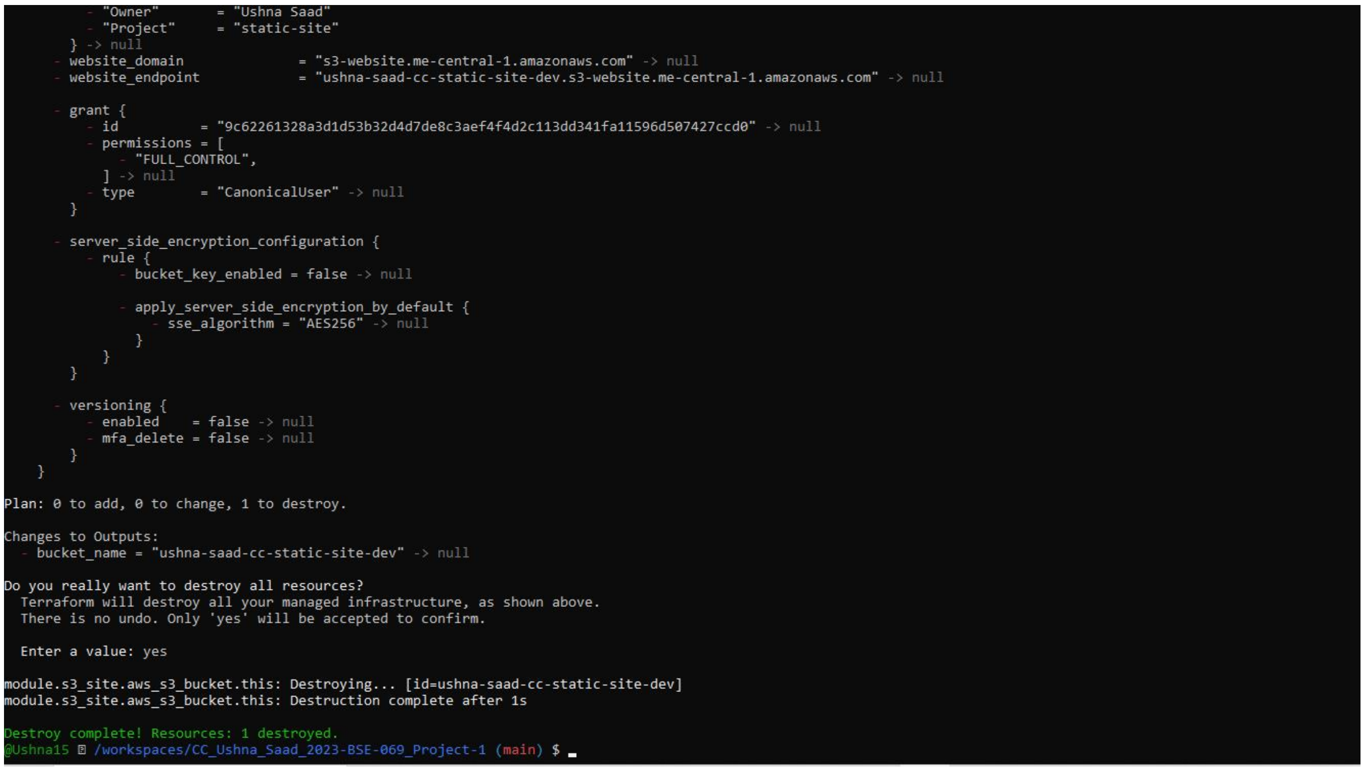
Failed to delete (0)

Name	Prefix	Version ID	Type	Last modified	Size	Error

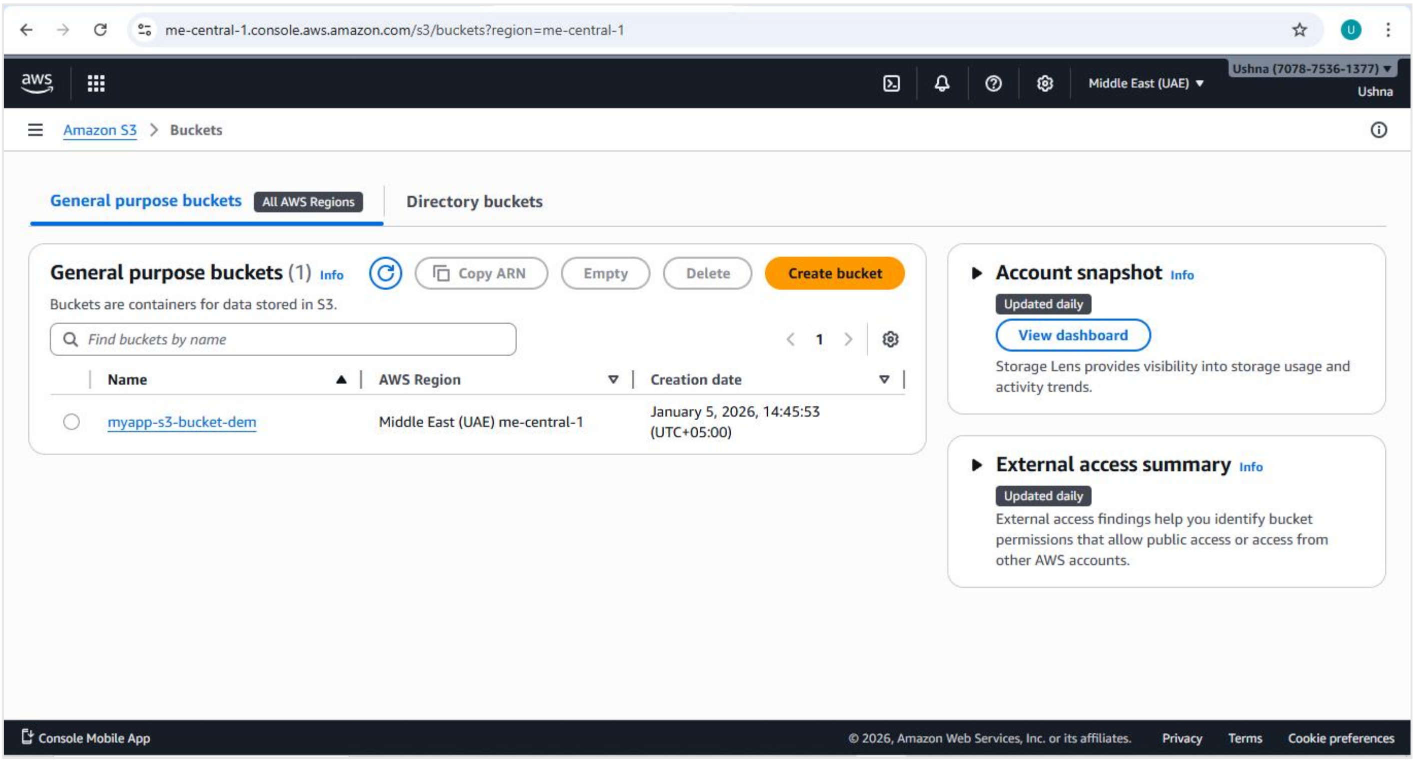
No failed object deletions

© 2026, Amazon Web Services, Inc. or its affiliates.

terraform destroy


```
          - "Owner"       = "Ushna Saad"
          - "Project"     = "static-site"
        } -> null
      - website_domain              = "s3-website.me-central-1.amazonaws.com" -> null
      - website_endpoint            = "ushna-saad-cc-static-site-dev.s3-website.me-central-1.amazonaws.com" -> null

      - grant {
          - id          = "9c62261328a3d1d53b32d4d7de8c3aef4f4d2c113dd341fa11596d507427ccd0" -> null
          - permissions = [
              - "FULL_CONTROL",
            ] -> null
          - type        = "CanonicalUser" -> null
        }

      - server_side_encryption_configuration {
          - rule {
              - bucket_key_enabled = false -> null

              - apply_server_side_encryption_by_default {
                  - sse_algorithm = "AES256" -> null
                }
            }
        }

      - versioning {
          - enabled    = false -> null
          - mfa_delete = false -> null
        }
    }

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

Changes to Outputs:
  - bucket_name = "ushna-saad-cc-static-site-dev" -> null

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

module.s3_site.aws_s3_bucket.this: Destroying... [id=ushna-saad-cc-static-site-dev]
module.s3_site.aws_s3_bucket.this: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
@Ushna15 ➜ /workspaces/CC_Ushna_Saad_2023-BSE-069_Project-1 (main) $
```

Confirming all resources (S3, CloudFront) are deleted in AWS Console


me-central-1.console.aws.amazon.com/s3/buckets?region=me-central-1

Amazon S3 > Buckets

General purpose buckets (1)

Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
myapp-s3-bucket-dem	Middle East (UAE) me-central-1	January 5, 2026, 14:45:53 (UTC+05:00)

Account snapshot

Storage Lens provides visibility into storage usage and activity trends.

External access summary

External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

© 2026, Amazon Web Services, Inc. or its affiliates.

4. Troubleshooting & Lessons Learned

During the implementation of the CI/CD pipeline, several technical challenges were encountered. Resolving these provided deep insight into how AWS services interact with automation tools.

1. Python Interpreter Mismatch (Environment Pathing)

- **The Issue:** When executing the Ansible playbook, the system returned an error stating that the boto3 library was missing, even though it had been installed via pip. This occurred because Ansible was defaulting to a system-level Python version rather than the version where the AWS SDK was installed.

- **The Solution:** The issue was resolved by explicitly defining the Python interpreter during the playbook execution using the command: -e "ansible_python_interpreter=$(which python3)"

- **Lesson Learned:** Explicitly defining environment paths is critical when working with automation tools to ensure library dependencies are correctly mapped.

2. CloudFront Caching (The "Blank Page" Latency)

- **The Issue:** After successfully uploading the index.html file via Ansible, the CloudFront URL continued to display a "403 Access Denied" or a blank page. This was due to the CDN's edge locations caching the "Error" state that existed before the file was uploaded.

- **The Solution**: A manual CloudFront Invalidation was created for the path /*. This forced the global edge locations to purge their cache and fetch the newly uploaded content from the S3 origin.

- **Lesson Learned**: In a CDN-based architecture, content updates require a cache-clearing strategy to ensure immediate visibility for end-users.

3. S3 Bucket Deletion Error (Resource Dependency)

- **The Issue**: Running terraform destroy resulted in a failure. Terraform cannot delete an S3 bucket if it still contains objects (like the index.html file), as this is an AWS safety mechanism to prevent accidental data loss.

- **The Solution:** The AWS CLI was used to empty the bucket contents before re-running the destroy command. In a production environment, this could also be managed by setting the force_destroy = true argument within the Terraform S3 resource block.

- **Lesson Learned:** Infrastructure lifecycle management requires a clear understanding of resource dependencies—specifically that "Content" must be managed or deleted before "Infrastructure" can be decommissioned.

5. Conclusion

The project successfully achieved its goal of deploying a secure, automated, and high-performance static website on AWS. By moving away from manual "click-ops" and adopting Infrastructure as Code (IaC), we have created a system that is:

- **Secure:** The S3 origin is completely private, protected by AWS's modern Origin Access Control.

- **Scalable:** The use of Amazon CloudFront ensures that the website can handle traffic spikes globally with minimal latency.

- **Repeatable:** The combination of Terraform and Ansible ensures that the entire environment can be destroyed and rebuilt in minutes, guaranteeing consistency across Dev, Test, and Prod stages.

This project serves as a robust foundation for modern web hosting, proving that automation and security can work hand-in-hand to provide a superior developer and end-user experience.