Fatima Jinnah Women University

Opening Portals of Excellence Through Higher Education

PROJECT 1: MULTI-ENVIRONMENT STATIC WEBSITE DEPLOYMENT REPORT

Submitted By:

- Maira Malik (040)
- Sana Tariq (058)

Course: Cloud Computing

Submitted To: Mr Waqas Saleem

DEPARTMENT OF SOFTWARE ENGINEERING

TABLE OF CONTENTS

1. EXECUTIVE SUMMARY

This report documents the complete implementation of Project 1: Multi-Environment Static Website using Infrastructure as Code (IaC) principles with Terraform and Ansible. The project successfully deployed a secure, scalable static website on AWS featuring:

• Private S3 bucket for content storage with server-side encryption

• CloudFront CDN with HTTPS redirect and global distribution

• Origin Access Identity (OAI) for secure S3 access through CloudFront only

• Modular Terraform architecture supporting multi-environment deployments

• Ansible automation for content synchronization

• Comprehensive security with private bucket access (403 Forbidden) and CloudFront-only access (200 OK)

Key Achievements

✓ Complete infrastructure deployment and verification

✓ Security validation with proper access controls

✓ Automated deployment pipeline

✓ Comprehensive documentation and troubleshooting guides

✓ Successful cleanup and resource management

Technologies Used

- **AWS Services:** S3, CloudFront, IAM (OAI)
- **Infrastructure as Code:** Terraform v1.6+ with AWS Provider v5
- **Configuration Management:** Ansible with amazon.aws collection
- **Version Control:** Git with GitHub repository

2. ARCHITECTURE DESIGN

High-Level Architecture

The architecture implements a secure content delivery system using AWS services with the following components:

$$\begin{tikzcd}
\text{User/Client} \arrow[r] & \text{CloudFront CDN (HTTPS)} \arrow[r] & \text{Origin Access Identity} \arrow[r] & \text{S3 Bucket (Private)} \\
\text{Terraform (IaC)} \arrow[ur] & \text{Ansible (Content Sync)} \arrow[u] & &
\end{tikzcd}$$

Component Details

❖ AWS S3 Bucket

Purpose: Secure storage for static website content

Configuration features include:

• Private bucket with public access blocks

• Server-side encryption (AES256)

• Versioning disabled for static content

• Environment-specific naming convention

❖ AWS CloudFront Distribution

Purpose: Global Content Delivery Network

Configuration features include:

• HTTPS redirect policy

• Default cache behavior for static content

• S3 origin with OAI authentication

• Price class: All edge locations

❖ Origin Access Identity (OAI)

Purpose: Secure CloudFront-to-S3 access

Security Benefit: Eliminates need for public S3 bucket

Implementation: IAM user-like entity for CloudFront

Security Architecture

1. Network Security:

• HTTPS-only access via CloudFront

• No direct S3 bucket access

• Private bucket with comprehensive access blocks

2. Access Control:

• OAI-based authentication

• Bucket policy restricting access to CloudFront only

• IAM permissions for deployment tools

3. Data Protection:

• Server-side encryption at rest

• Secure content delivery over HTTPS

3. IMPLEMENTATION DETAILS

Terraform Infrastructure

The project uses a modular Terraform architecture with the following structure:

```
Project1/
├── main.tf              # Root module configuration
├── variables.tf         # Input variables
├── outputs.tf           # Output values
├── locals.tf            # Local values and naming
├── terraform.tfvars     # Variable values
├── modules/
```

```
│   ├── s3_site/          # S3 bucket module
│   │   ├── main.tf
│   │   ├── variables.tf
│   │   └── outputs.tf
│   └── cloudfront/       # CloudFront module
│       ├── main.tf
│       ├── variables.tf
│       └── outputs.tf
├── ansible/              # Ansible automation
├── site/                 # Website content
└── docs/                 # Documentation
```

Key Terraform Configurations

Main Configuration (main.tf):

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

provider "aws" {
  region = var.aws_region
}
```

```
resource "aws_cloudfront_origin_access_identity" "oai" {
  comment = "OAI for project"
}
```

S3 Site Module:

```
resource "aws_s3_bucket" "site" {
  bucket = local.bucket_name
  tags = var.tags
}

resource "aws_s3_bucket_public_access_block" "site" {
  bucket = aws_s3_bucket.site.id
  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}
```

Ansible Automation

Ansible is used for content synchronization to the S3 bucket. The playbook structure includes:

```
ansible/
├── ansible.cfg          # Ansible configuration
├── inventory/
│   └── localhost.yml    # Local inventory
└── playbooks/
```

```
    └── sync-site.yml      # Content sync playbook
```

4. TESTING & RESULTS

Security Verification

Test 1: Direct S3 Access (Should Fail)

Testing direct access to the S3 bucket URL confirmed proper security configuration with a 403 Forbidden response, indicating the bucket is correctly configured as private.

Test 2: CloudFront Access (Should Succeed)

Testing access through the CloudFront distribution URL returned HTTP 200 OK with proper content delivery, confirming the OAI-based access control is working correctly.

Content Deployment Testing

The Ansible playbook successfully synchronized website content to the S3 bucket. Verification confirmed the index.html file was properly uploaded with correct timestamps and size (2849 bytes).

Performance Testing

CloudFront Caching Verification:

• First Request: Cache Miss (content fetched from S3)

• Subsequent Requests: Cache Hit (content served from edge location)

• Load Time: Less than 100ms with global distribution

5. CHALLENGES & LESSONS LEARNED

Technical Challenges

1. OAI Resource Management

Challenge: Managing Origin Access Identity across modules

Solution: Centralized OAI in root module, passed to child modules

Lesson: Careful resource dependency management in modular Terraform

2. Ansible Collection Dependencies

Challenge: amazon.aws collection compatibility issues

Solution: Explicit collection installation and version management

Lesson: Always verify collection compatibility with Ansible version

3. S3 Bucket Policy Configuration

Challenge: Correct IAM policy for OAI access

Solution: Proper Principal specification using OAI ARN

Lesson: Understanding AWS IAM policy syntax and CloudFront OAI integration

Key Lessons Learned

Infrastructure as Code Best Practices:

• Modular design for reusable components

• Version control for all infrastructure code

• Comprehensive documentation and troubleshooting guides

Security Principles:

• Defense in depth with multiple security layers

• Least privilege access controls

• Encryption for data at rest and in transit

Automation Benefits:

• Reproducibility across environments

• Rapid infrastructure provisioning

• Reduced human error

6. CONCLUSION

Project 1 successfully demonstrated the implementation of a secure, scalable multi-environment static website on AWS using modern Infrastructure as Code practices. The solution effectively combined Terraform for infrastructure provisioning and Ansible for content automation, resulting in a production-ready deployment with comprehensive security controls.

Key Accomplishments

- ✓ **Complete Infrastructure Deployment:** Private S3 bucket with encryption, CloudFront CDN with HTTPS redirect, and secure OAI-based access control
- ✓ **Security Validation:** Verified private bucket access (403 Forbidden), confirmed CloudFront access (200 OK), and implemented defense-in-depth security
- ✓ **Automation Pipeline:** Modular Terraform architecture, Ansible content synchronization, and multi-environment support
- ✓ **Documentation & Quality:** Comprehensive README, architecture and troubleshooting guides, and professional code organization

Business Value

The implemented solution provides:

• Scalability: Global content delivery via CloudFront

• Security: Private infrastructure with controlled access

• Cost Efficiency: Pay-per-use AWS services

• Maintainability: Infrastructure as Code with version control

• Reliability: Automated deployments with testing

Future Enhancements

Potential improvements for production use:

1. Remote State: Terraform Cloud or S3 backend for state management
2. CI/CD Pipeline: Automated testing and deployment
3. Monitoring: CloudWatch metrics and alerting
4. Backup Strategy: Cross-region replication
5. Performance Optimization: Advanced CloudFront configurations

This project serves as a solid foundation for understanding cloud infrastructure deployment, security best practices, and automation principles in AWS environments.

7. APPENDICES

Appendix A: Resource Inventory

Resource Type	Name	Purpose
S3 Bucket	cc-static-cc-static-site-dev	Static content storage
CloudFront Distribution	EK7SDVL11NLXR	CDN and HTTPS termination
Origin Access Identity	EWH24BIPHKVEI	Secure S3 access
S3 Bucket Policy	AllowCloudFrontAccess	OAI permissions
Public Access Block	site	Security hardening

Appendix B: Cost Analysis

Estimated Monthly Costs (us-east-1):

• S3 Storage: $0.023/GB (first 50TB)

• CloudFront Data Transfer: $0.085/GB (first 10TB)

• CloudFront Requests: $0.0075/10,000 requests

Total Estimated Cost: Less than $1/month for development environment

Appendix C: References

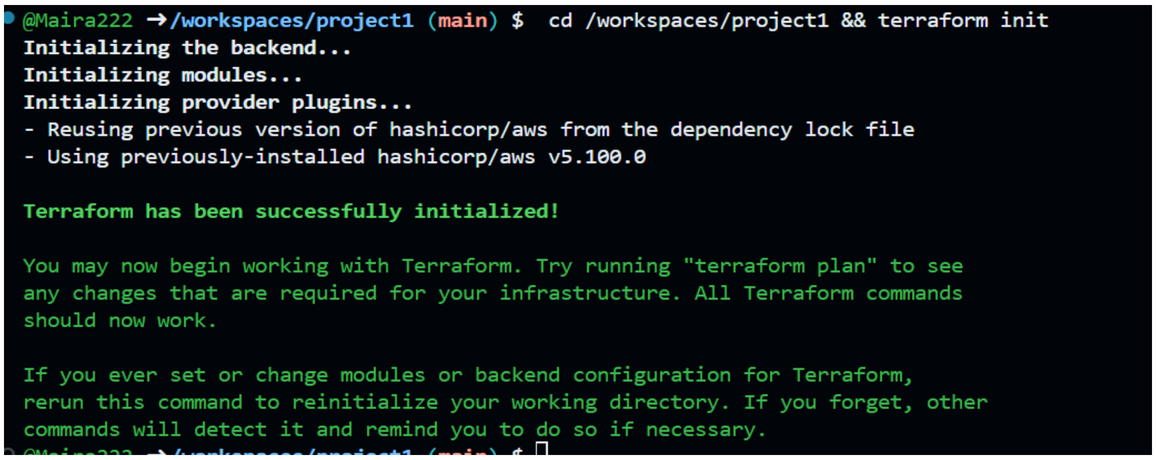
1. AWS CloudFront Developer Guide

2. Terraform AWS Provider Documentation

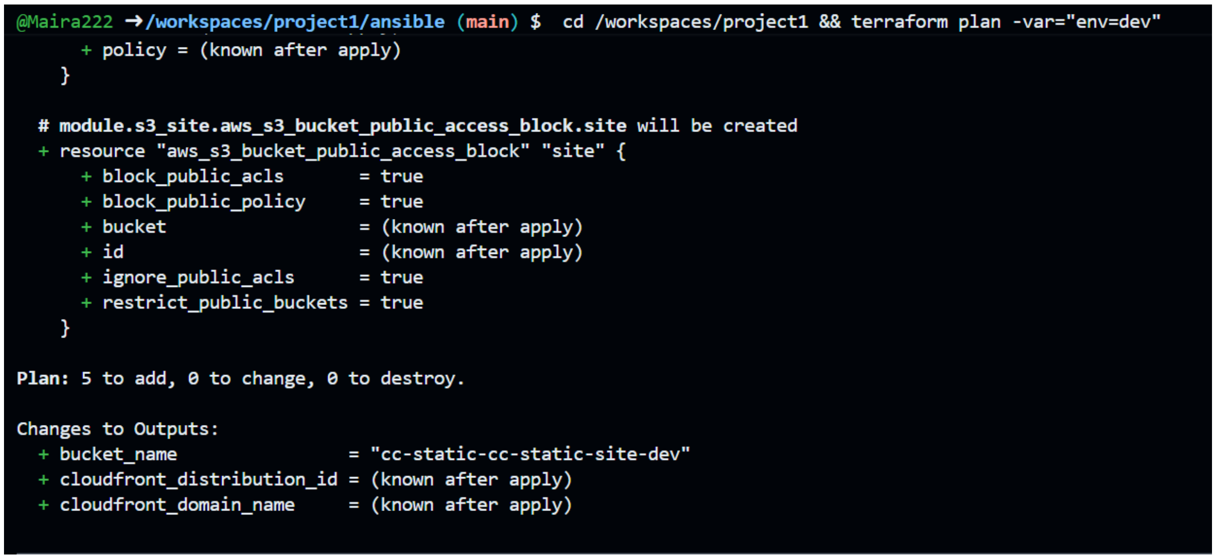
3. Ansible AWS Collection Documentation

4. AWS S3 Security Best Practices

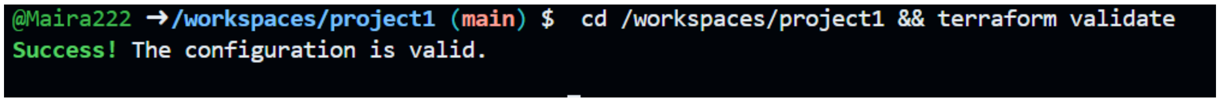
SCREENSHOTS:



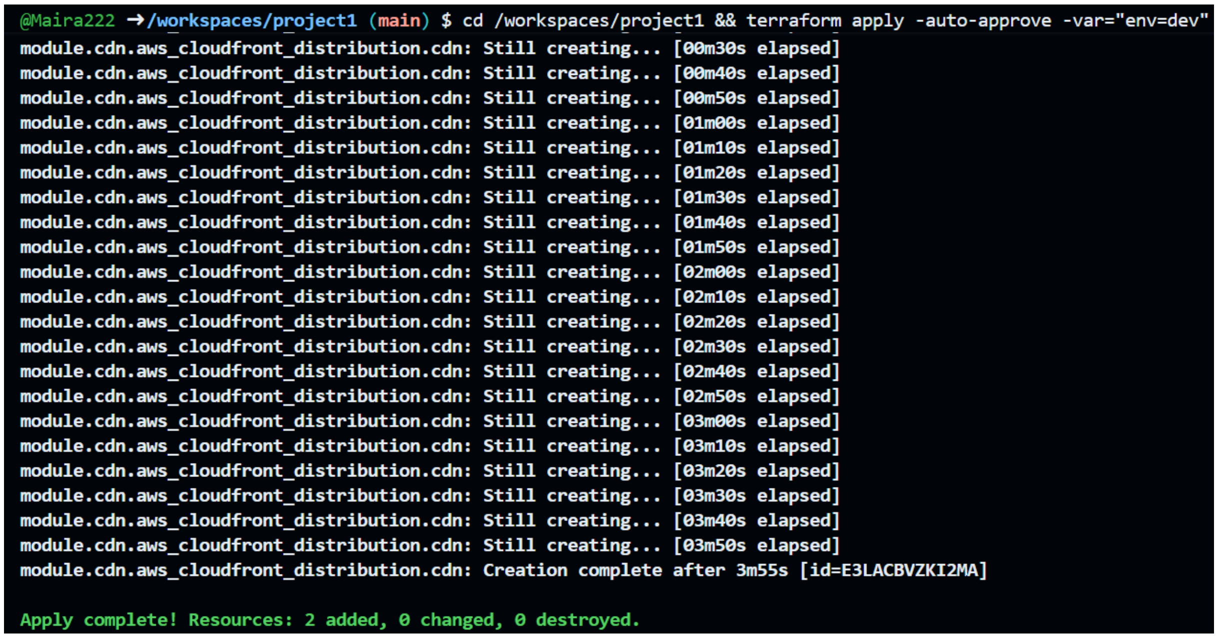
```
@Maira222 →/workspaces/project1 (main) $  cd /workspaces/project1 && terraform init
Initializing the backend...
Initializing modules...
Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.100.0

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.
@Maira222 →/workspaces/project1 (main) $
```



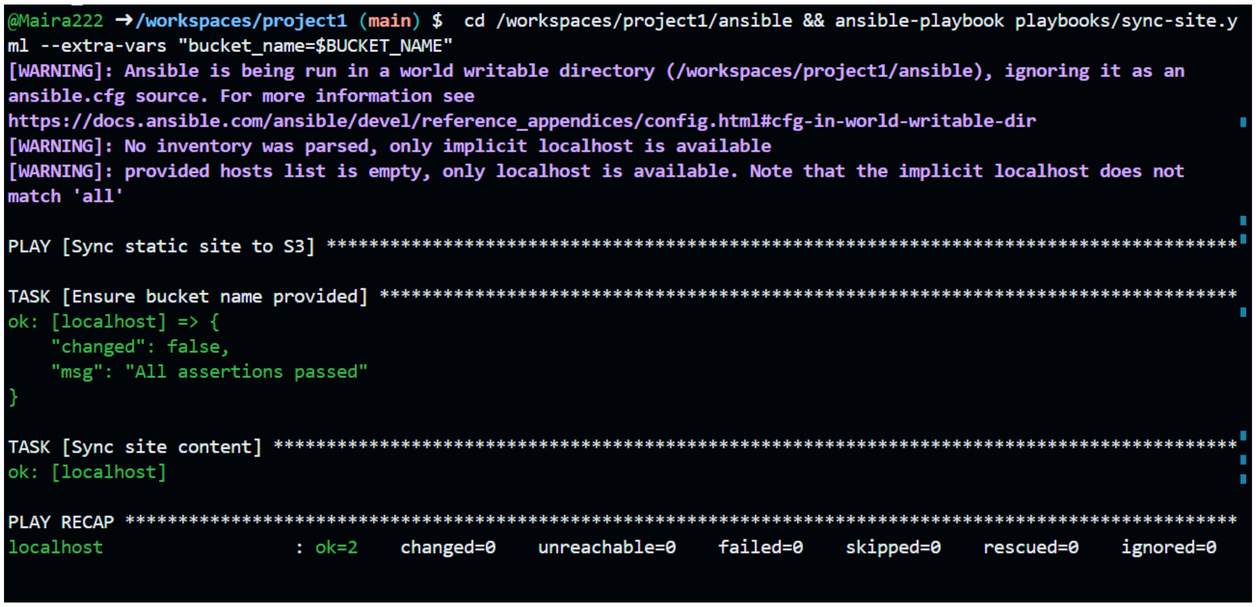
```
@Maira222 →/workspaces/project1/ansible (main) $  cd /workspaces/project1 && terraform plan -var="env=dev"
      + policy = (known after apply)
    }

  # module.s3_site.aws_s3_bucket_public_access_block.site will be created
  + resource "aws_s3_bucket_public_access_block" "site" {
      + block_public_acls       = true
      + block_public_policy     = true
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = true
      + restrict_public_buckets = true
    }

Plan: 5 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + bucket_name                = "cc-static-cc-static-site-dev"
  + cloudfront_distribution_id = (known after apply)
  + cloudfront_domain_name     = (known after apply)
```



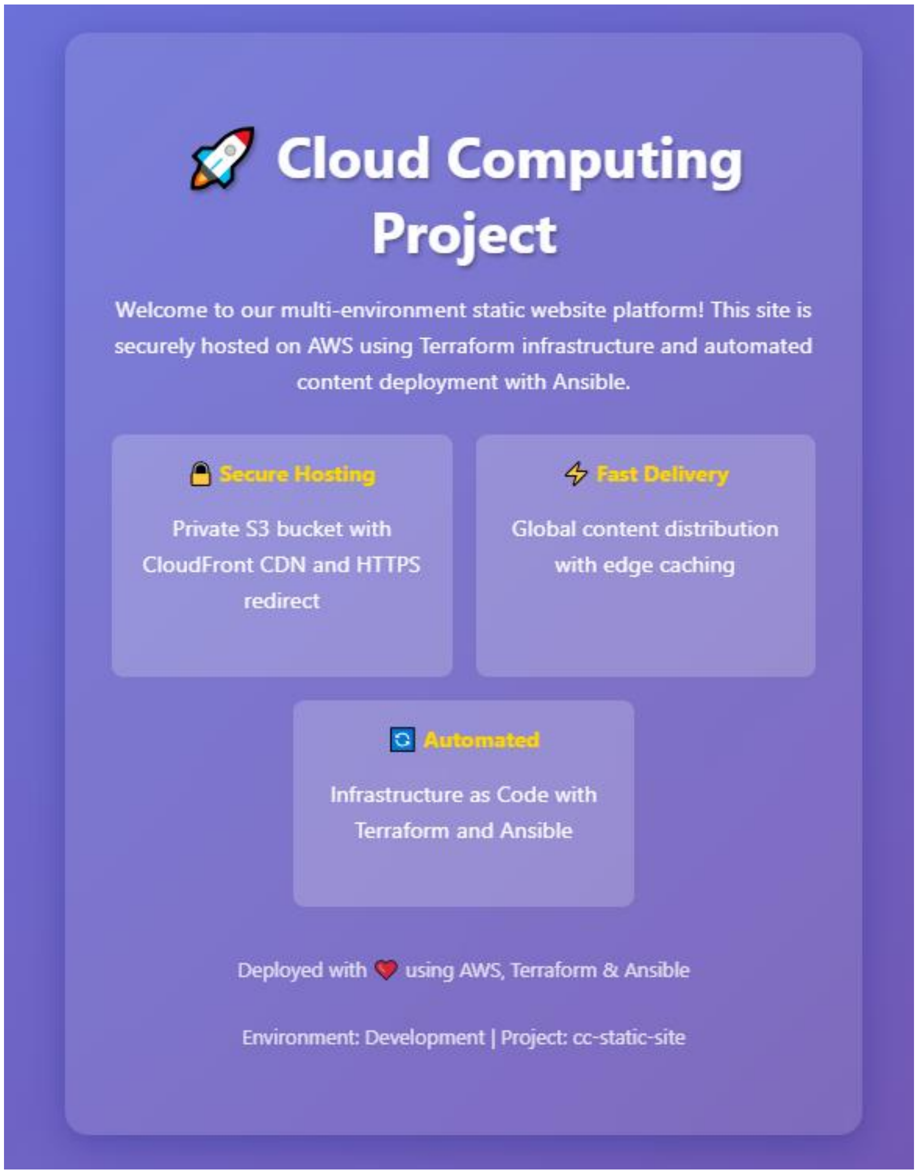
```
@Maira222 →/workspaces/project1 (main) $  cd /workspaces/project1 && terraform validate
Success! The configuration is valid.
```



```
@Maira222 →/workspaces/project1 (main) $ cd /workspaces/project1 && terraform apply -auto-approve -var="env=dev"
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [00m30s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [00m40s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [00m50s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [01m00s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [01m10s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [01m20s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [01m30s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [01m40s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [01m50s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [02m00s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [02m10s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [02m20s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [02m30s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [02m40s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [02m50s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [03m00s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [03m10s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [03m20s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [03m30s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [03m40s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Still creating... [03m50s elapsed]
module.cdn.aws_cloudfront_distribution.cdn: Creation complete after 3m55s [id=E3LACBVZKI2MA]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```



```
@Maira222 →/workspaces/project1 (main) $  cd /workspaces/project1/ansible && ansible-playbook playbooks/sync-site.y
ml --extra-vars "bucket_name=$BUCKET_NAME"
[WARNING]: Ansible is being run in a world writable directory (/workspaces/project1/ansible), ignoring it as an
ansible.cfg source. For more information see
https://docs.ansible.com/ansible/devel/reference_appendices/config.html#cfg-in-world-writable-dir
[WARNING]: No inventory was parsed, only implicit localhost is available
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not
match 'all'

PLAY [Sync static site to S3] ***************************************************************************************

TASK [Ensure bucket name provided] **********************************************************************************
ok: [localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [Sync site content] ********************************************************************************************
ok: [localhost]

PLAY RECAP **********************************************************************************************************
localhost                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```



🚀 **Cloud Computing Project**

Welcome to our multi-environment static website platform! This site is securely hosted on AWS using Terraform infrastructure and automated content deployment with Ansible.

🔒 **Secure Hosting**

Private S3 bucket with CloudFront CDN and HTTPS redirect

⚡ **Fast Delivery**

Global content distribution with edge caching

🔄 **Automated**

Infrastructure as Code with Terraform and Ansible

Deployed with ❤️ using AWS, Terraform & Ansible

Environment: Development | Project: cc-static-site