**Cloud Automation Deployment**

**Overview**

This project automates the deployment of a static HTML/CSS website on AWS EC2 using Infrastructure as Code and containerization tools. It demonstrates DevOps best practices using Terraform for provisioning, Ansible for configuration, and Docker for application deployment.

**Tools & Technologies Used**

| **Tool/Service** | **Purpose** |
| --- | --- |
| Terraform | Provision AWS EC2 instance & networking |
| Ansible | Configure the EC2 server and install Docker |
| Docker | Containerize and run the sample web app |
| GitHub Actions | Automate CI/CD workflow for deployment |
| AWS EC2 | Cloud infrastructure |
| VS Code / WSL2 | Local development environment |

**Project Structure**

| **Path/File** | **Description** |
| --- | --- |
| terraform/ | Terraform scripts (main.tf, key.tf, etc.) |
| ansible/ | Ansible playbooks and inventory file |
| static-site/ | Dockerfile and HTML/CSS web page |
| .github/workflows/ | GitHub Actions pipeline (deploy.yml) |
| README.md | Documentation and setup instructions |

**Setup Instructions**

**Part 1: Infrastructure Provisioning with Terraform**

1. Clone the GitHub Repository:  
   git clone <https://github.com/shubh2220/Cloud.git>  
   *cd project*
2. Configure AWS credentials in your environment or via AWS CLI.
3. Initialize and deploy with Terraform: Refering *main.tf and security\_group.tf*  to run :-

*terraform init*

*terraform validate*

*terraform plan*

*terraform apply*

1. Confirm to create EC2 instance and output the public IP.

**Part 2: Server Configuration with Ansible**

1. Installation of Ansible by

*sudo apt install ansible -y*

1. Update *inventory.ini* with the correct EC2 public IP.

*nano inventory.ini*

1. Run the Ansible Playbook:  
    *ansible-playbook -i inventory.ini install\_docker.yml*
2. Docker is installed and started on boot.

**Part 3: Docker Container Deployment**

1. Move into static-site/ directory: Contains your Dockerfile and index.html (HTML/CSS combined)
2. Upload the static-site folder to EC2:  
    *scp -i /.ssh/First\_Key.pem -r ./static-site ubuntu@3.250.99.137:/*
3. SSH into EC2:  
    *ssh -i ~/.ssh/First\_Key.pem* [*ubuntu@3.250.99.137*](mailto:ubuntu@3.250.99.137)
4. Build and run Docker container:  
    *docker build -t static-site ./static-site  
    docker run -d -p 80:80 static-site*
5. Visit [http://*3.250.99.137*](http://3.250.99.137)to view the webpage.

**Part 4: CI/CD Pipeline with GitHub Actions**

CI/CD automation has been implemented using GitHub Actions:

* Workflow file: .github/workflows/deploy.yml
* Trigger: Automatically runs on push to main branch.
* Steps:
  + Checkout code
  + Build Docker image by using command

*docker images*

* + Deploy to EC2 via SSH

This ensures every code update to the repo is tested, built, and deployed automatically to the cloud environment.

**Final Deliverables**

* Terraform scripts
* Ansible playbook
* Dockerfile and web app
* GitHub Actions workflow
* Functional public web app (on EC2)
* GitHub repository with README
* Screenshots and architecture diagram in report

**Repository**

GitHub Repository: <https://github.com/shubh2220/Cloud>