

Project Report

Title: Cloud & DevOps Engineering Projects Report

Name: M Anusuya

Course: Cloud Computing & DevOps

Institute: Besant Technologies, Bengaluru

Duration: Jun 2025 – Present

Role: Cloud & DevOps Trainee

Declaration

I hereby declare that this project report titled “**Cloud & DevOps Engineering Projects**” is my original work carried out during my training period at Besant Technologies. This work has not been submitted previously for any academic qualification. All references and resources used are duly acknowledged.

Acknowledgement

I would like to express my sincere gratitude to Besant Technologies and my trainers for their continuous guidance and support throughout my Cloud & DevOps training. I also thank my peers for their collaboration and encouragement, which helped me successfully complete these projects.

Abstract

This report presents a comprehensive overview of multiple Cloud and DevOps projects implemented using AWS, Azure, Linux, networking concepts, and modern DevOps tools. The projects focus on automating infrastructure provisioning, implementing CI/CD pipelines, deploying scalable cloud-native applications, and improving system reliability and availability. These implementations demonstrate realworld DevOps practices such as Infrastructure as Code (IaC), containerization, orchestration and automation.

Table of Contents

- Introduction
 - Tools & Technologies Used
 - Projects
 - Learning Outcomes
 - Conclusion
-

Introduction

Cloud computing and DevOps have become essential for delivering scalable, reliable, and fast software solutions. This project report showcases hands-on implementations that combine cloud infrastructure, automation, and CI/CD practices. The objective is to reduce manual effort, improve deployment speed, and ensure high availability and security of applications.

Tools & Technologies Used

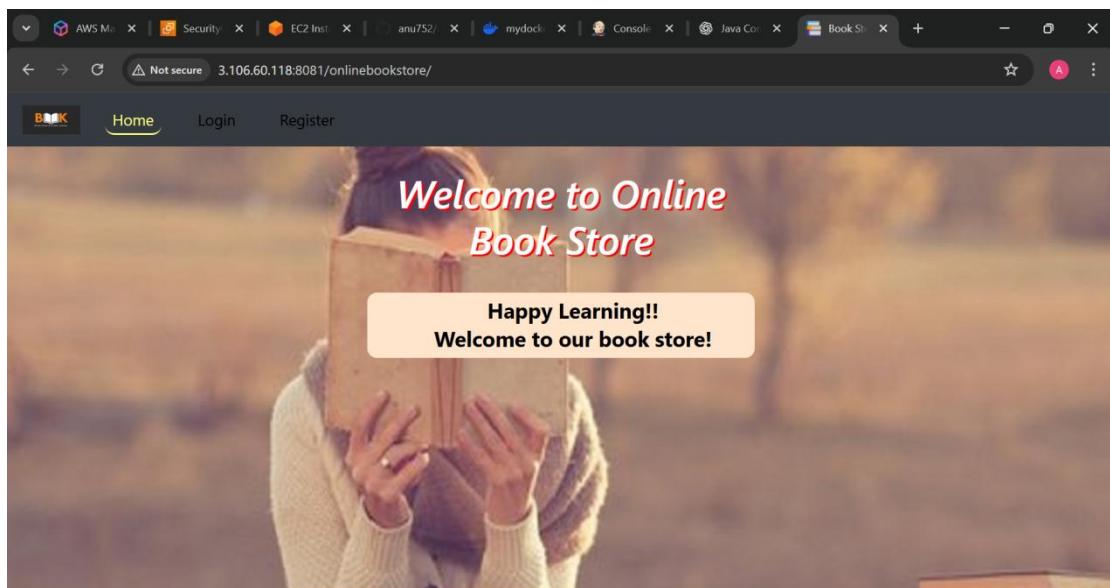
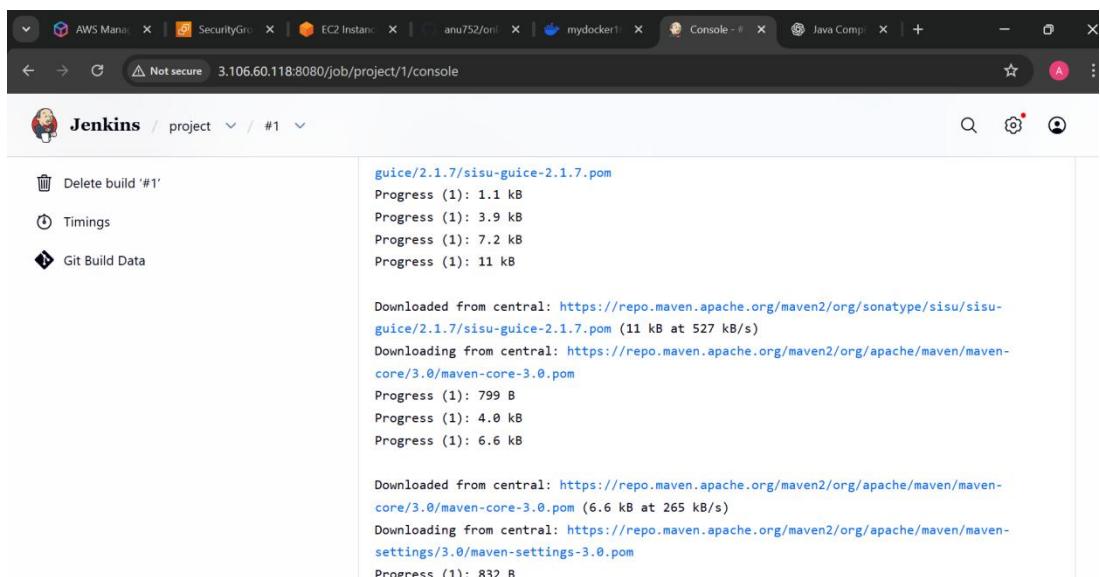
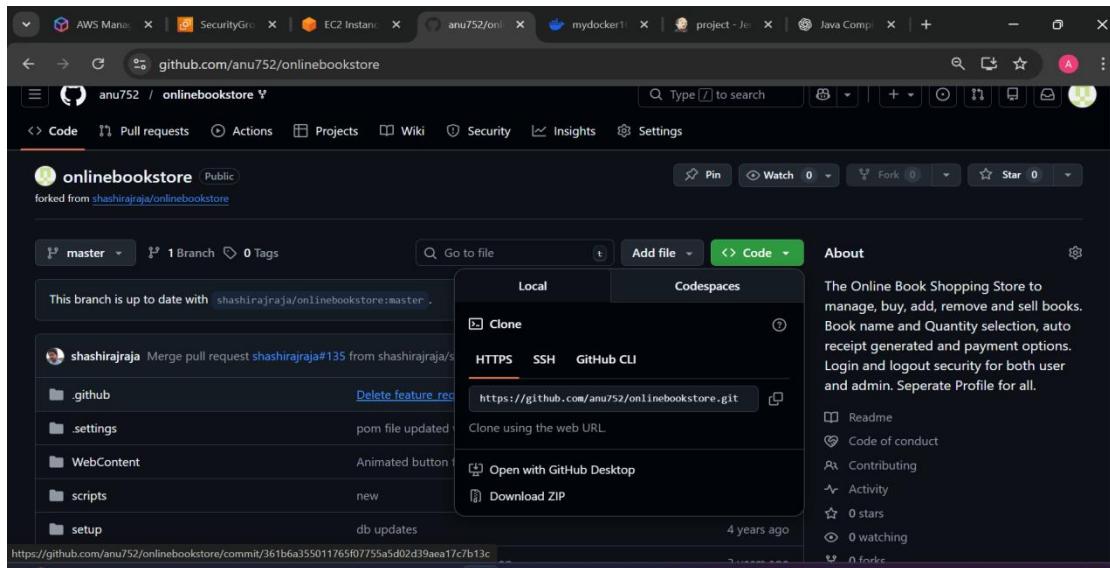
- **Cloud Platforms:** AWS
- **Operating Systems:** Linux (RedHat, Ubuntu)
- **DevOps Tools:** Git, GitHub, Jenkins, Docker, Kubernetes
- **Programming & Scripting:** Python, Shell Scripting
- **Networking:** VPC, Subnets, Routing, Load Balancers, Security Groups

Projects

Project – 1 : Automated Cloud-Native CI/CD Pipeline

This project focuses on building an automated CI/CD pipeline to deploy cloud-native applications in a consistent and reliable manner. Source code was managed using GitHub, and Jenkins was used to automate build and deployment processes. Docker was used to containerize the application, and Kubernetes handled deployment and scalability on AWS infrastructure.

Tools Used: GitHub, Jenkins, Docker, Kubernetes, AWS



```

project-app LoadBalancer 10.100.7.214 a7fa7e37fa4e94ed7bb109615db4f629-194266150.ap-southeast-2.elb.amazonaws.com 80:31724/TCP 36s
[root@ip-172-31-12-211 ~]# kubectl get all
NAME           READY   STATUS    RESTARTS   AGE
pod/project-app-84568db6d-tq2j8  1/1     Running   0          23m
pod/project-app-84568db6d-z822q  1/1     Running   0          23m

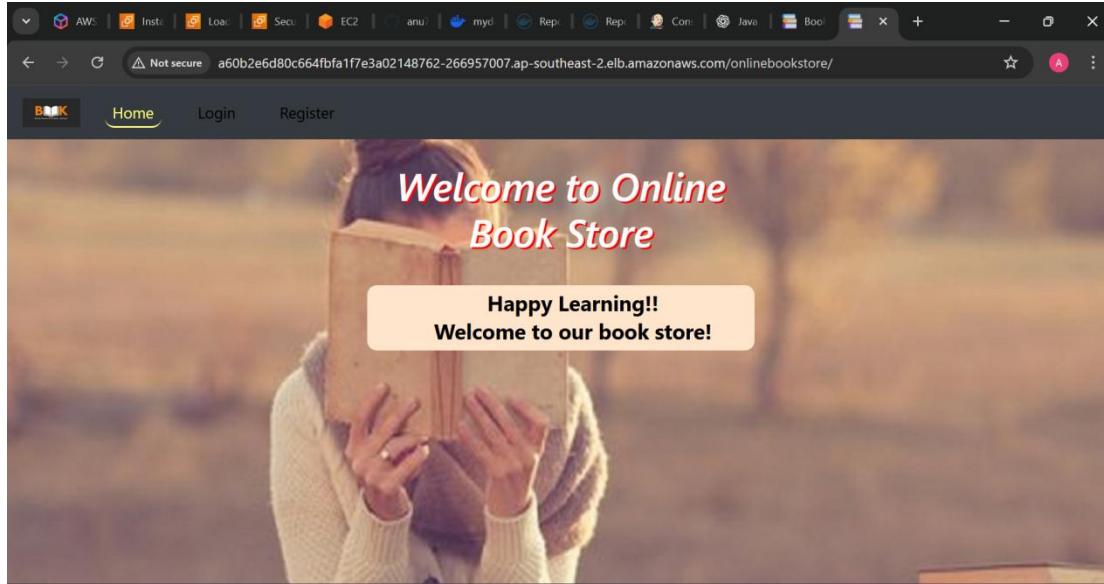
NAME            TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP  10.100.0.1   <none>        443/TCP       35m
service/project-app   LoadBalancer  10.100.7.214  a7fa7e37fa4e94ed7bb109615db4f629-194266150.ap-southeast-2.elb.amazonaws.com  80:31724/TCP  50s

NAME          READY  UP-TO-DATE  AVAILABLE  AGE
deployment.apps/project-app  2/2     2           2          23m

NAME          DESIRED  CURRENT  READY  AGE
replicaset.apps/project-app-84568db6d  2         2         2          23m
[root@ip-172-31-12-211 ~]# kubectl delete svc project-app
service "project-app" deleted
[root@ip-172-31-12-211 ~]# kubectl expose deployment project-app --type=LoadBalancer --port=80 --target-port=8080
service/project-app exposed
[root@ip-172-31-12-211 ~]# kubectl get service
NAME            TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP  10.100.0.1   <none>        443/TCP       39m
project-app   LoadBalancer  10.100.80.43  a60b2e6d80c664fbfa1f7e3a02148762-266957007.ap-southeast-2.elb.amazonaws.com  80:32313/TCP  6s
[root@ip-172-31-12-211 ~]#

```

i-0b485f5779cf66758 (devops)



Outcome:

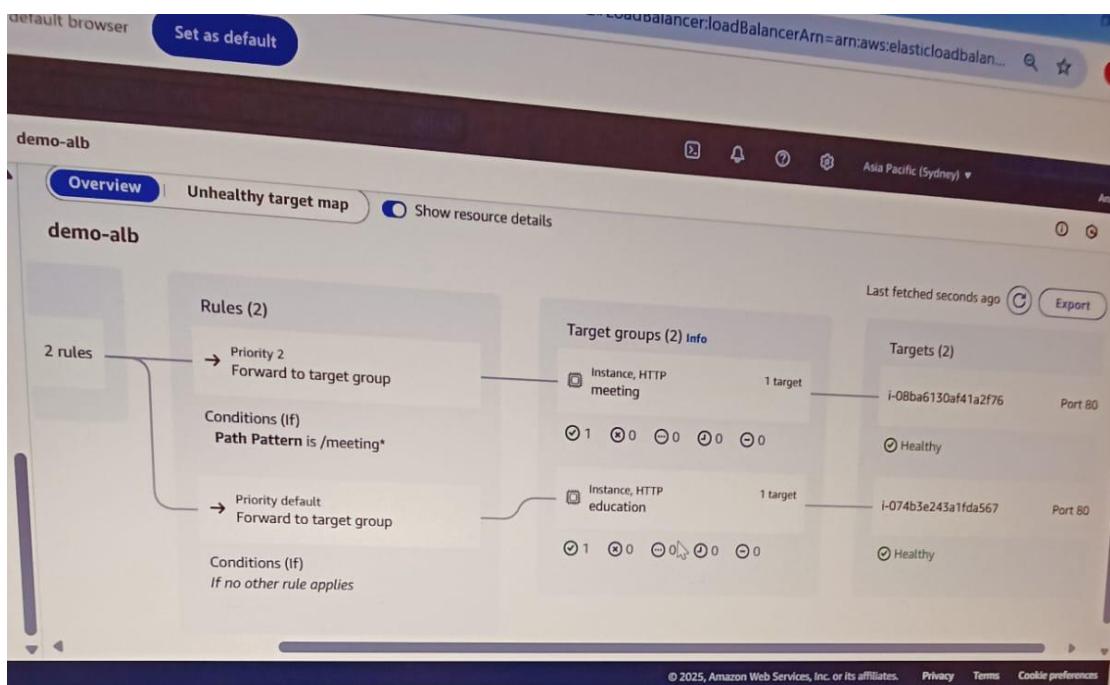
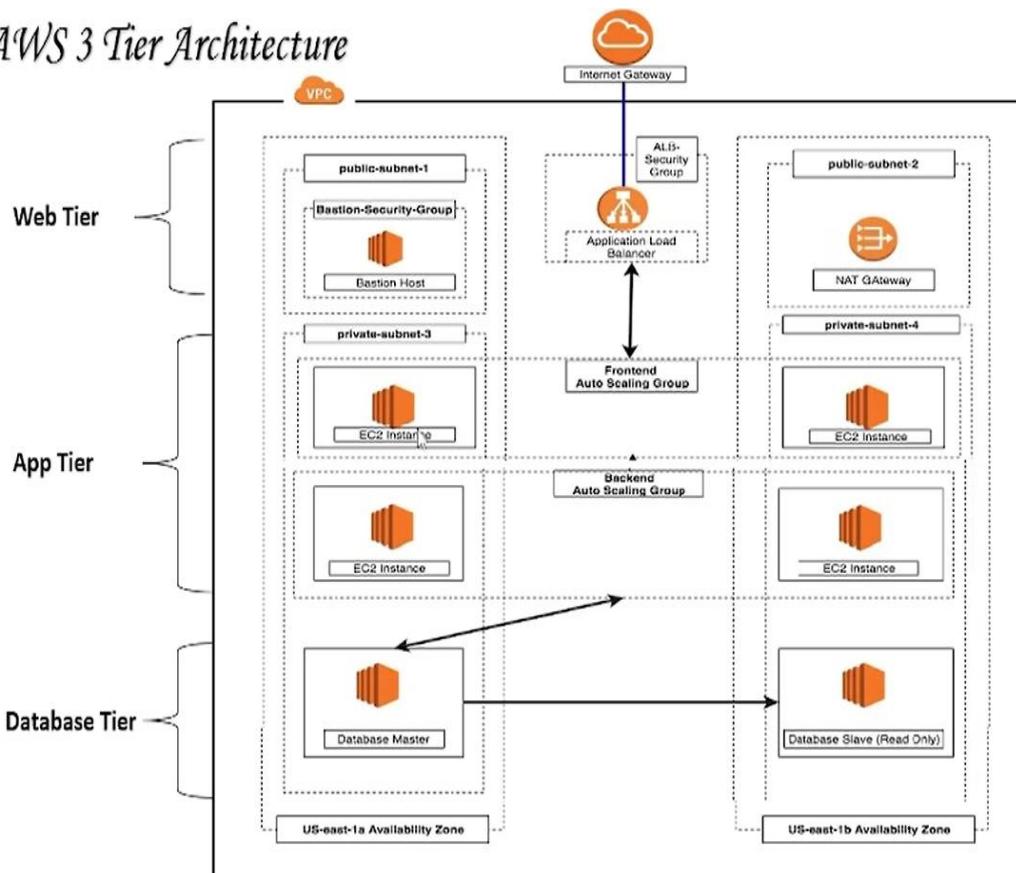
Improved deployment speed, reduced manual effort, and achieved reliable and scalable application delivery.

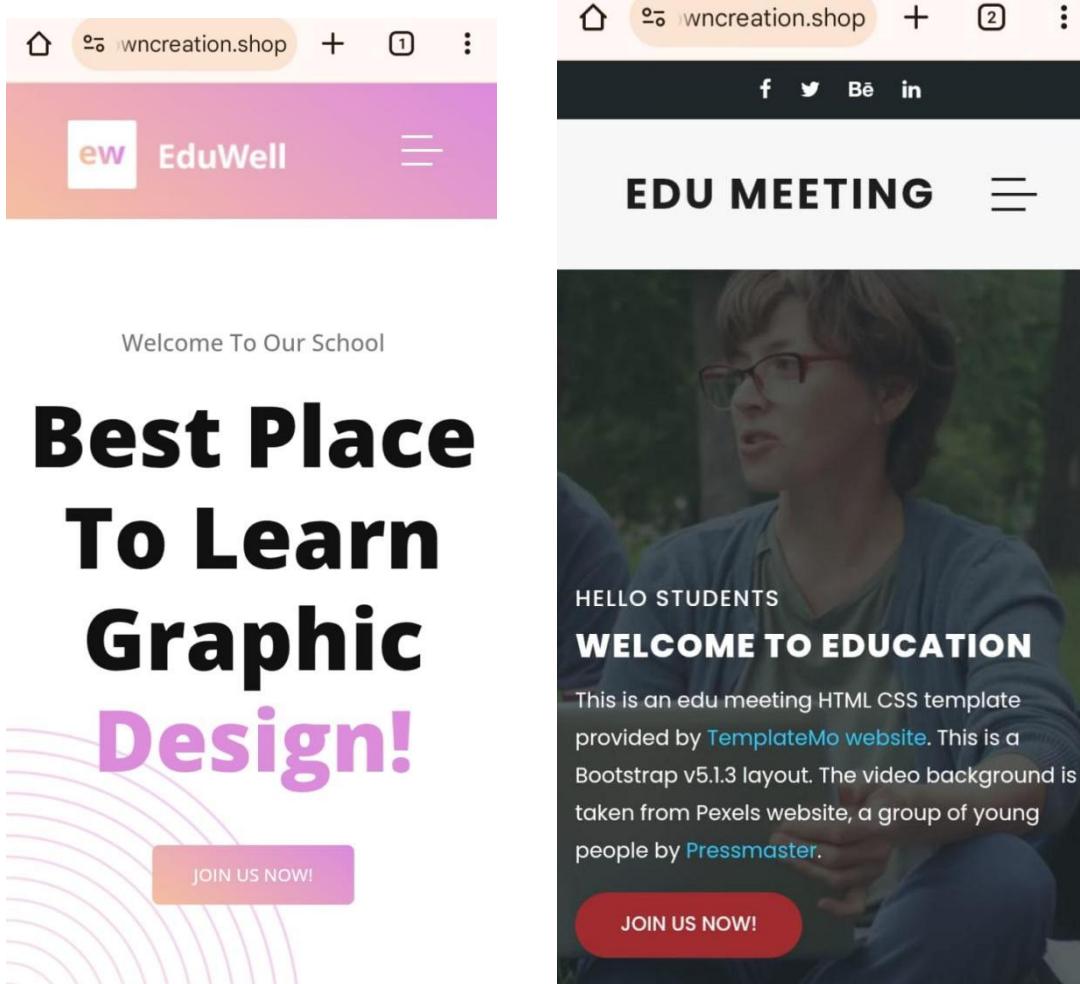
Project – 2: AWS 3-Tier Architecture Implementation

In this project, a secure and scalable 3-tier architecture was designed and implemented on AWS. The architecture separates web, application, and database layers using public and private subnets. Load Balancers and Auto Scaling ensured high availability, while security groups and IAM improved overall security.

Tools Used: AWS EC2, VPC, ALB, Auto Scaling, RDS, IAM, Route53

AWS 3 Tier Architecture





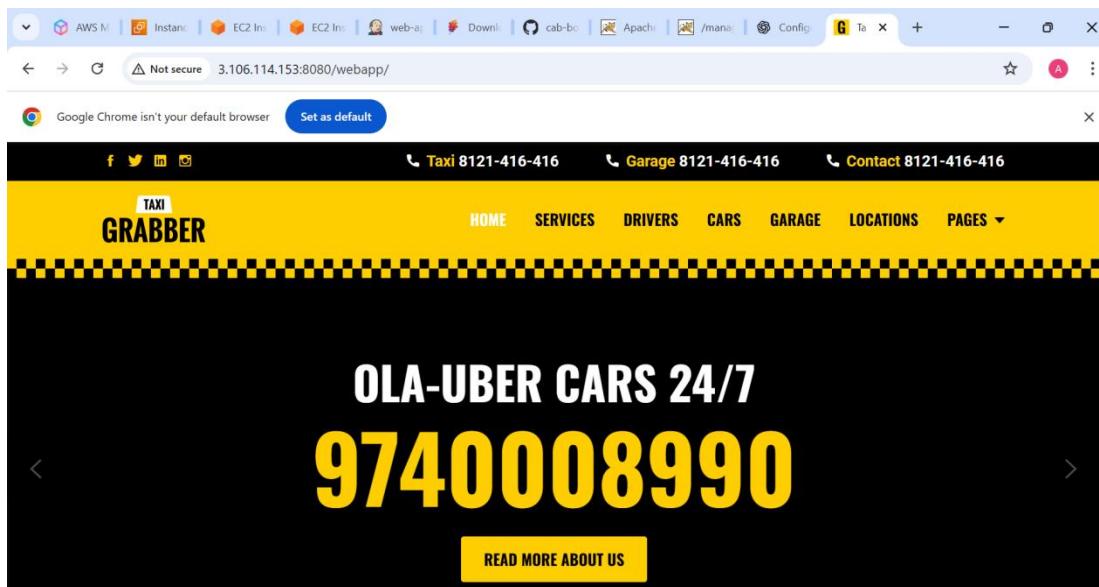
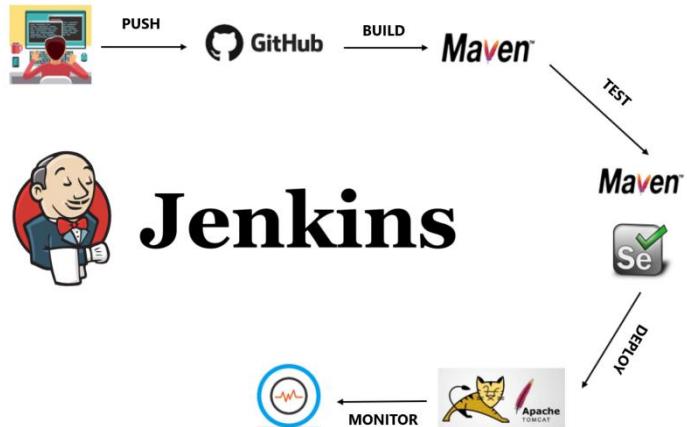
Outcome:

Achieved high availability, improved security, and better performance through layered architecture design.

Project- 3 : Jenkins-Based CI/CD Pipeline for Java Web Application

This project focuses on implementing an automated CI/CD pipeline for a Java-based web application using Jenkins. The pipeline triggers automatically when code is pushed to GitHub, builds and tests the application using Maven and Selenium, and deploys the application to an Apache Tomcat server.

Tools Used: GitHub, Jenkins, Maven, Selenium, Apache Tomcat, Linux



Outcome:

Automated build, test, and deployment process, reduced manual intervention, and improved reliability and delivery speed of the application.

Project – 4: Hosting a Static Website on AWS S3 with Terraform

This project demonstrates the use of Infrastructure as Code, a simple and efficient way to host a basic website using Terraform and Amazon Web Services (AWS) S3. The main aim of this project is to show how to automatically create and deploy a website without much hassle.

Tools Used: AWS S3, Terraform, VS code

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, ...
- Search Bar:** static web-hosting
- Explorer:** Shows the project structure:
 - OPEN EDITORS: main.tf, index.html
 - STATIC WEB-HOSTING: .terraform, terraform.lock.hcl, index.html, main.tf
 - Others: terraform.tfstate, terraform.tfstate.backup
- Editor:** The main.tf file is open, showing Terraform code for AWS S3 bucket creation and configuration.

```
1 terraform {  
2   required_providers {  
3     aws = {  
4       source  = "hashicorp/aws"  
5       version = "6.11.0"  
6     }  
7     random = {  
8       source = "hashicorp/random"
```
- Terminal:** Shows the output of the terraform apply command, indicating the creation of resources.

```
aws_s3_bucket.website: Creation complete after 7s [id=terraformproject123final-173e626a]  
aws_s3_bucket_public_access_block.website: Creating...  
aws_s3_bucket_ownership_controls.website: Creating...  
aws_s3_bucket_website_configuration.website: Creating...  
aws_s3_object.index_html: Creating...  
aws_s3_object.index_html: Creation complete after 1s [id=terraformproject123final-173e626a/index.html]  
aws_s3_bucket_ownership_controls.website: Creation complete after 1s [id=terraformproject123final-173e626a]  
aws_s3_bucket_website_configuration.website: Creation complete after 1s [id=terraformproject123final-173e626a]  
aws_s3_bucket_website_configuration.website: Creation complete after 1s [id=terraformproject123final-173e626a]  
aws_s3_bucket_public_access_block.website: Creation complete after 1s [id=terraformproject123final-173e626a]  
aws_s3_bucket_policy.website: Creating...  
aws_s3_bucket_policy.website: Creation complete after 1s [id=terraformproject123final-173e626a]
```
- Status Bar:** Anusuya@NJ-I00AA5692M7K /mnt/c/Users/Administrator/Desktop/terraform/static web-hosting\$

The screenshot shows the AWS S3 console with the following details:

- Buckets**:
 - General purpose buckets
 - Directory buckets
 - Table buckets
 - Vector buckets
- Access management and security**:
 - Access Points
 - Access Points for FSx
 - Access Grants
 - IAM Access Analyzer
- Storage management and insights**:
 - Storage Lens

Static website hosting is enabled for the bucket. The configuration includes:

- Requester pays**: Disabled
- Static website hosting**:
 - Use this bucket to host a website or redirect requests. [Learn more](#)
 - We recommend using AWS Amplify Hosting for static website hosting:
Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about [Amplify Hosting](#) or [View your existing Amplify apps](#).
 - [Create Amplify app](#)
- S3 static website hosting**: Enabled
- Hosting type**: Bucket hosting
- Bucket website endpoint**:
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)
<http://terraformproject123final-173e626a.s3-website-ap-southeast-2.amazonaws.com>

Book Your Ride

Pickup Location:

Drop-off Location:

Pickup Date & Time: - yyyy

Vehicle Type: Sedan

Outcome:

Automatically configured static website hosting, bucket policies, and public access successfully on AWS S3 bucket using Terraform without the need to access the AWS Console.

Project 5: Terraform — Deploy Dockerized App on AWS ECS with Load Balancer

This project demonstrates how to deploy a Dockerised app on Amazon ECS. Amazon ECS is the AWS service to run Docker applications on a scalable cluster. It is a container orchestration/management service similar to Kubernetes

Tools Used: AWS EC2, ELB, IAM, ECS, VPC, Terraform, vs code

The screenshot shows the VS Code interface with several files open in the Explorer sidebar: provider.tf, variables.tf, outputs.tf, vpc.tf, security.groups.tf, alb.tf, ecs.tf, main.tf, and outputs.tf. The provider.tf file contains the following Terraform code:

```

provider "aws" {
  region = var.aws_region
}

```

The terminal window displays AWS CLI logs for the creation of an ALB and its targets:

```

aws_lb.this: Still creating... [02m36s elapsed]
aws_lb.this: Still creating... [02m46s elapsed]
aws_lb.this: Creation complete after 1s [id=arn:aws:elasticloadbalancing:ap-southeast-2:405457752889:loadbalancer/app/ecs-demo-app-alb/1c176a07950de8a8]
aws_lb.listener.http: Creating...
aws_lb.listener.http: Creation complete after 1s [id=arn:aws:elasticloadbalancing:ap-southeast-2:405457752889:listener/app/ecs-demo-app-alb/1c176a07950de8a8/f1e48298353daad5]
aws_ecs_service.this: Creating...
aws_ecs_service.this: Creation complete after 2s [id=arn:aws:ecs:ap-southeast-2:405457752889:service/ecs-demo-app/ecs-demo-app]

```

Outputs:

```

alb_dns_name = "ecs-demo-app-alb-1236501212.ap-southeast-2.elb.amazonaws.com"
anusuya@MJ-I00AA5692M7K:/mnt/c/Users/Administrator/Desktop/terraform/project$ 

```

Outputs:

```

alb_dns_name = "ecs-demo-app-alb-1236501212.ap-southeast-2.elb.amazonaws.com"
anusuya@MJ-I00AA5692M7K:/mnt/c/Users/Administrator/Desktop/terraform/project$ 

```

The status bar indicates 12 lines, 172 selected characters, 4 spaces, UTF-8 encoding, CRLF line endings, and the Terraform extension.

The screenshot shows the AWS Management Console Load Balancers page for the 'ecs-demo-app-alb'. It displays the Resource map for the load balancer, showing one listener (HTTP:80), one rule (Priority default: Forward to target group), one target group (IP, HTTP: ecs-demo-app-tg), and two healthy targets (IP addresses 10.0.0.87 and 10.0.1.50).

The screenshot shows a web browser displaying the 'Welcome to nginx!' page, indicating successful deployment of the application.

Outcome:

This project demonstrated the process of deploying a Dockerized application on Amazon ECS using Terraform. It covered setting up ECS resources, and configuring networking. We can automate the deployment of containerized apps on ECS for scalable and reliable infrastructure on AWS.

Learning Outcomes

- Practical exposure to real-world DevOps workflows
 - Strong understanding of cloud infrastructure and automation
 - Improved troubleshooting and problem-solving skills
 - Hands-on experience with CI/CD, IaC, and monitoring tools
-

Conclusion

These projects strengthened my understanding of Cloud and DevOps principles and provided hands-on experience in designing, deploying, and automating modern cloud infrastructures. The work aligns with industry standards and prepares me for entry-level Cloud or DevOps roles.