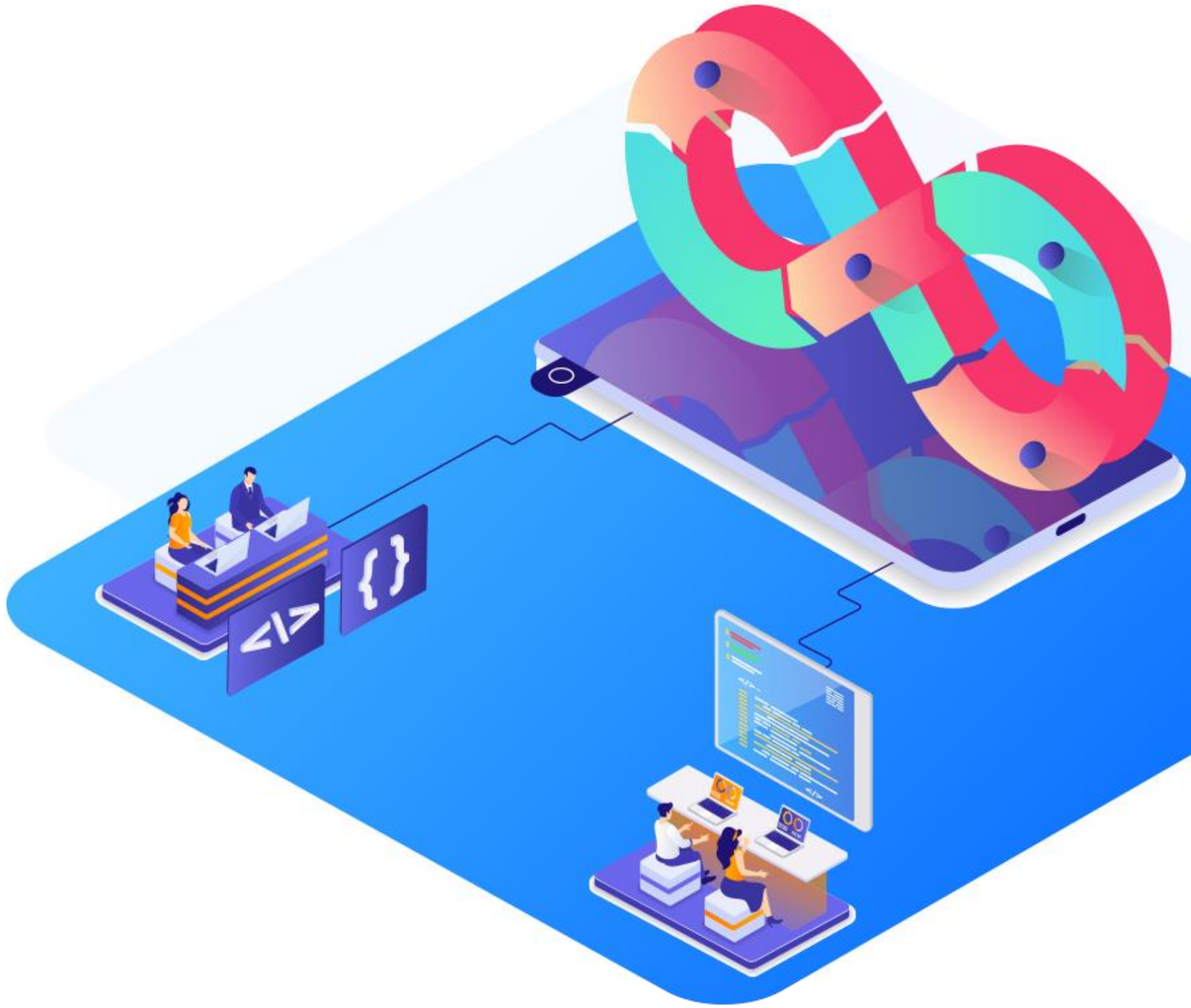
Capstone Project



AWS DevOps Spring Boot Application Deployment

Objective

To establish a seamless deployment strategy for the Spring Boot application, ensuring reliability, scalability, and automation while minimizing repetitive, error-prone manual steps that lead to inefficiencies and downtime

The project aims to:

- Showcase the practical implementation of AWS DevOps tools and services
- Provide hands-on experience in creating a modern automated deployment pipeline
- Deliver a scalable and reliable solution for real-world applications

Problem Statement and Motivation

Real-time scenario:

A financial technology startup, FinPro Solutions, is developing a secure payment processing application built on Spring Boot. The company anticipates rapid growth due to its innovative features and needs to ensure that its application deployment is scalable, reliable, and automated.

The company struggles with manual, error-prone deployments, limited infrastructure scalability for handling traffic spikes, and the lack of automated pipelines, slowing feature releases and updates.

To address these challenges, the company implements AWS CodePipeline, CodeBuild, ECS with Fargate, and Docker with ECR, ensuring automated, scalable, and consistent deployments, addressing inefficiencies and enabling seamless updates.

Industry Relevance

The following tools used in this project serve specific purposes within the industry:

AWS CodePipeline and CodeBuild:

- Automate CI/CD workflows, widely used in industries like e-commerce, gaming, and financial services for rapid deployment and consistent integration
- Enable faster time-to-market, critical for competitive markets such as fintech and healthcare

AWS ECS with Fargate:

- Powers scalable, serverless container management; ideal for applications requiring dynamic resource allocation during traffic spikes, as seen in gaming, media streaming, and fintech

Industry Relevance

The following tools used in this project serve specific purposes within the industry:

Docker and ECR:

- Ensure application portability and environment consistency; widely adopted in industries using microservices architectures, such as logistics, telecom, and SaaS platforms

IAM Roles:

- Manage secure access to cloud resources, ensuring compliance with data protection regulations, which is crucial for sectors like finance, government, and healthcare

Business Impact of the Tools

- **Reduced downtime:** Automated deployments ensure minimal service interruptions during updates.
- **Improved scalability:** ECS Fargate dynamically scales to accommodate increased user traffic without manual intervention.
- **Accelerated time-to-market:** CI/CD pipelines streamline the release of new features, enabling faster adaptation to market demands.
- **Cost savings:** Automation reduces the need for large manual operations teams and optimizes infrastructure costs.

Business Impact of the Tools

- **Jenkins:** Automates the CI/CD pipeline for rapid updates and bug fixes, minimizing downtime and maintaining player engagement
- **AWS CLI:** Facilitates efficient management of AWS resources, accelerating configuration changes and deployment for the game application

Tasks

The following tasks outline the steps of the project execution:

1. **Create ECR and Fork GitHub repository:** Set up an Elastic Container Registry (ECR) for storing Docker images and fork the provided GitHub repository to update the buildspec.yml file

2. **Configure CodeBuild and permissions:** Create a CodeBuild project to compile the application and build the Docker image, and then update IAM roles for access to ECR and ECS

3. **Build and push Docker image**: Execute the CodeBuild project to build the Docker image and push it to the ECR repository

4. **Set up ECS cluster and task:** Create an ECS Fargate cluster, define ECS tasks, and link the Docker image from ECR to the task configuration

Tasks

The following tasks outline the steps of the project execution:

5. **Deploy application on ECS:** Run the ECS service to deploy the application and verify its functionality via the endpoint (:8080/demo/data)

6. **Automate with CodePipeline:** Build a CodePipeline to automate the CI/CD workflow for continuous integration and deployment of the application

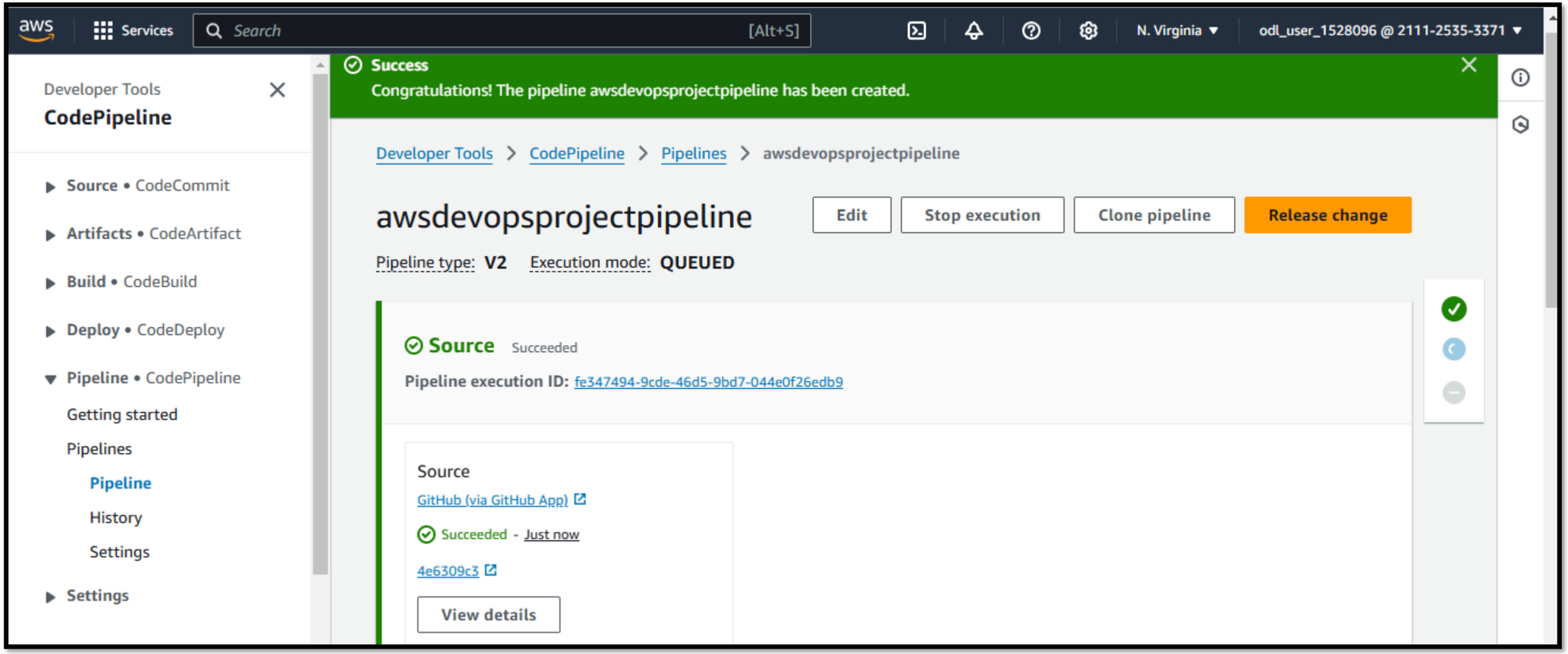
Project References

- **Task 1: Refer to the course:** Configuration Management with Ansible and Terraform and Containerization with Docker
- **Task 2: Refer to the course:** Containerization with Docker
- **Task 3: Refer to the course:** Containerization with Docker and Container Orchestration using Kubernetes
- **Task 4: Refer to the course:** DevOps Foundations Version Control and CICD with Jenkins
- **Task 5: Refer to the course:** Containerization with Docker and Container Orchestration using Kubernetes
- **Task 6: Refer to the course:** DevOps Foundations Version Control and CICD with Jenkins

COURSE-END PROJECT

Output Screenshots

The below screenshot shows successful deployment of a Spring Boot application using a CI/CD pipeline with AWS ECR, CodeBuild, and ECS:



Thank you