ANASIEZE IKENNA – CLOUD ENGINEER

Project: Deploy Multi-Tier Java App using AWS CI/CD Pipeline

Overview

This project focuses on building a fully automated CI/CD pipeline using AWS native services to deploy the Profile application, which is a multi-tier Java-based application requiring backend database, caching, and messaging layers.

I leveraged Bitbucket, AWS CodePipeline, Elastic Beanstalk, and other managed services for a cloud-native DevOps pipeline.

Problem Statement

Traditional deployments using Jenkins and EC2 are:

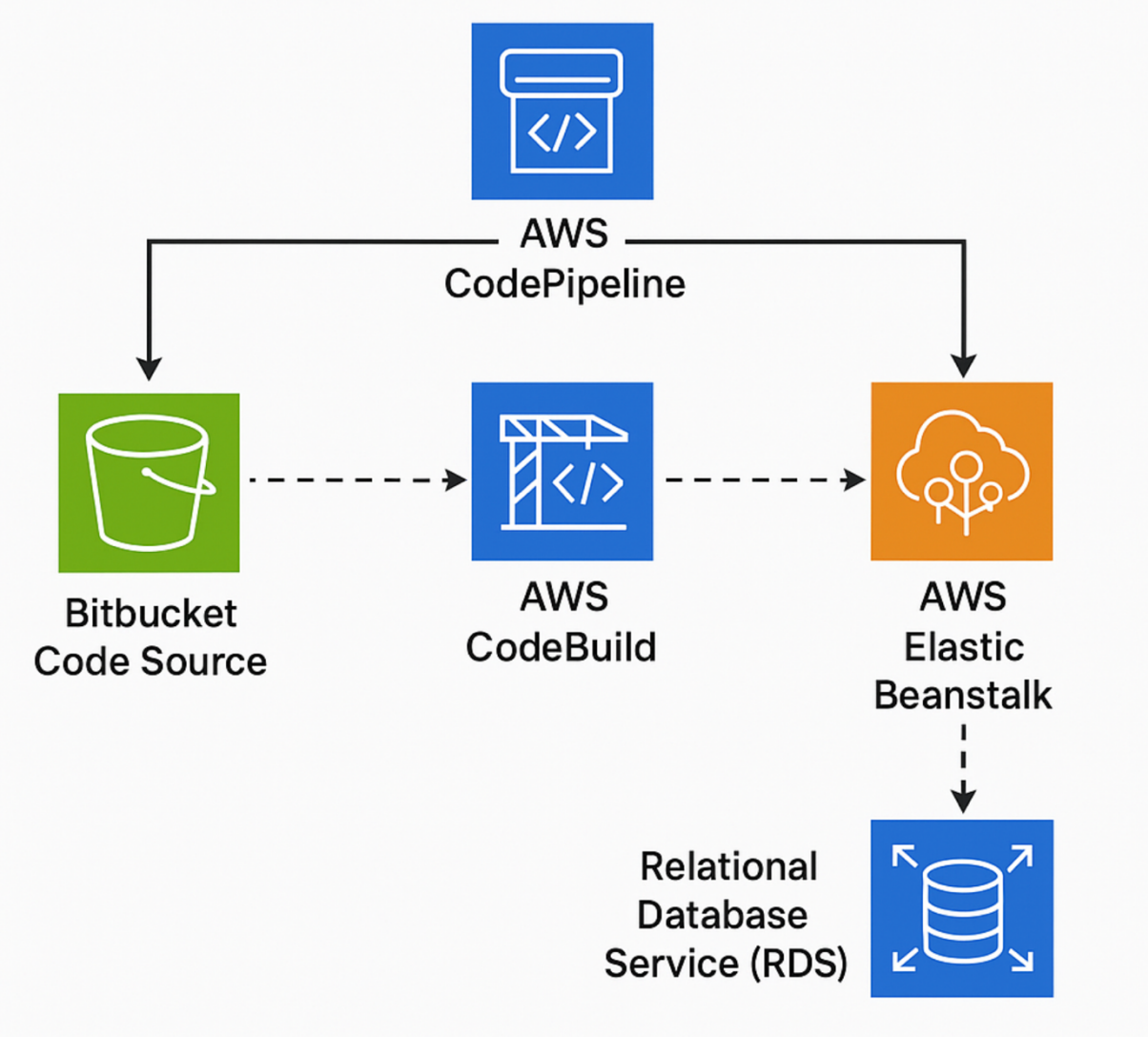
- Manual, error-prone, and not scalable
- Require maintenance of servers for Jenkins, DB, MQ, etc.
- Have slower release cycles and higher Ops overhead

➤ **Goal**: Replace this setup with a modern, AWS-native CI/CD pipeline that is fully automated, scalable, and cost-efficient.

Tech Stack

Layer	Service	Purpose
Source Control	Bitbucket	Git repository for application source
CI/CD Orchestration	AWS CodePipeline	Automates pipeline from commit to deploy
Build Tool	AWS CodeBuild	Compiles and packages WAR artifacts
Deployment	AWS Elastic Beanstalk	Manages app server, scaling, load balancer
Artifact Storage	Amazon S3	Stores build outputs
Database	Amazon RDS (MySQL)	Persistent database backend
DNS + CDN	Route 53 + CloudFront	Domain routing + low latency CDN
Monitoring	CloudWatch	Logs, metrics, and alarms

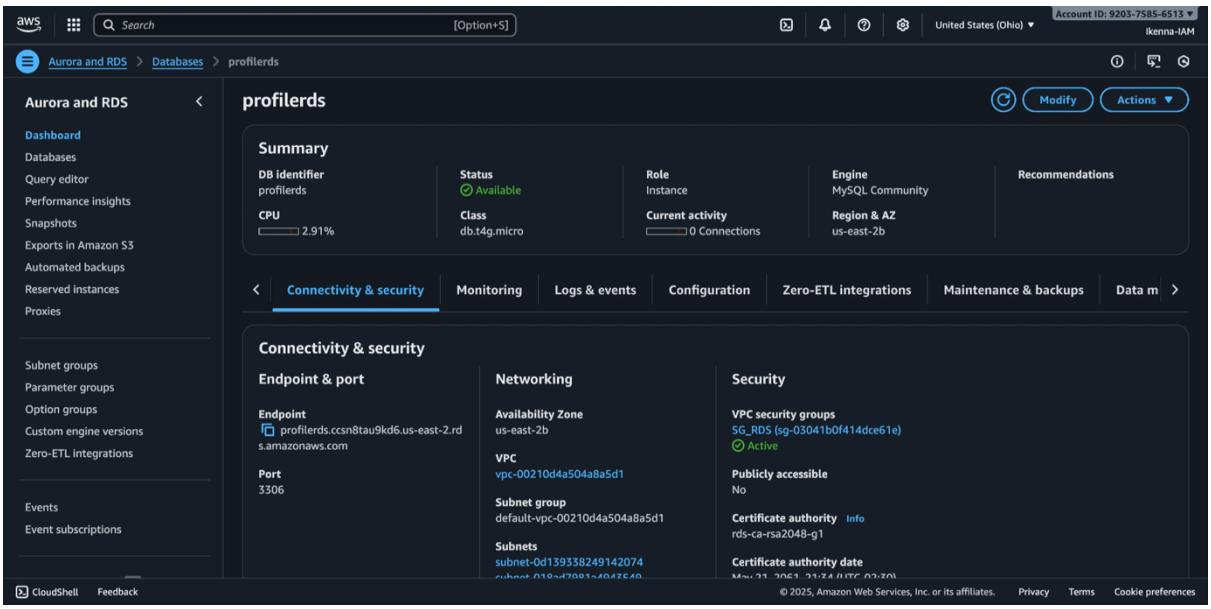
Architecture Overview



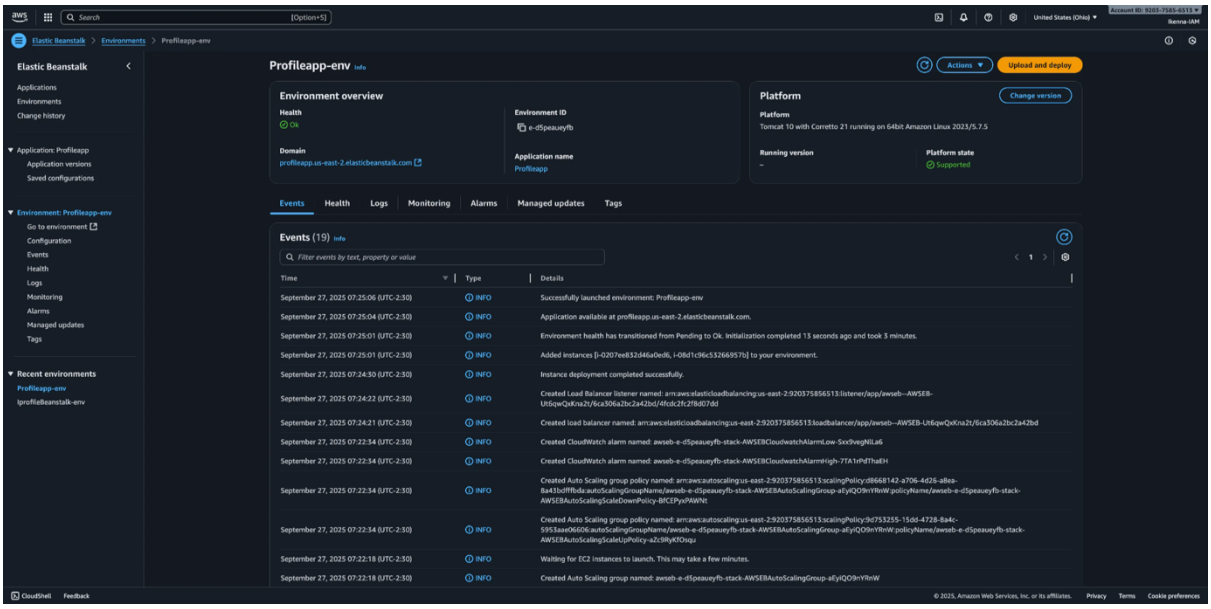
Implementation Steps

1. Migrate Profile source code from GitHub to Bitbucket
2. Create Elastic Beanstalk environment for app deployment (Tomcat-based)
3. Set up backend service: RDS
4. Configure Code Build with **buildspec.yml** to package WAR
5. Create S3 bucket for artifact storage
6. Set up Code Pipeline:
    - Source: Bitbucket
    - Build: Code Build
    - Deploy: Elastic Beanstalk
7. Set up CloudFront & Route 53 for global access and DNS
8. Push code to Bitbucket and verify full pipeline execution

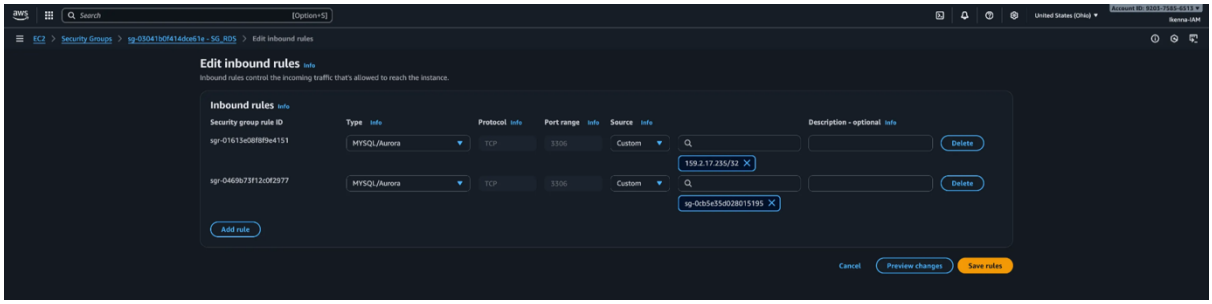
- **Set Up**
  - Create RDS Database



  - Create Elastic Beanstalk (Tomcat)



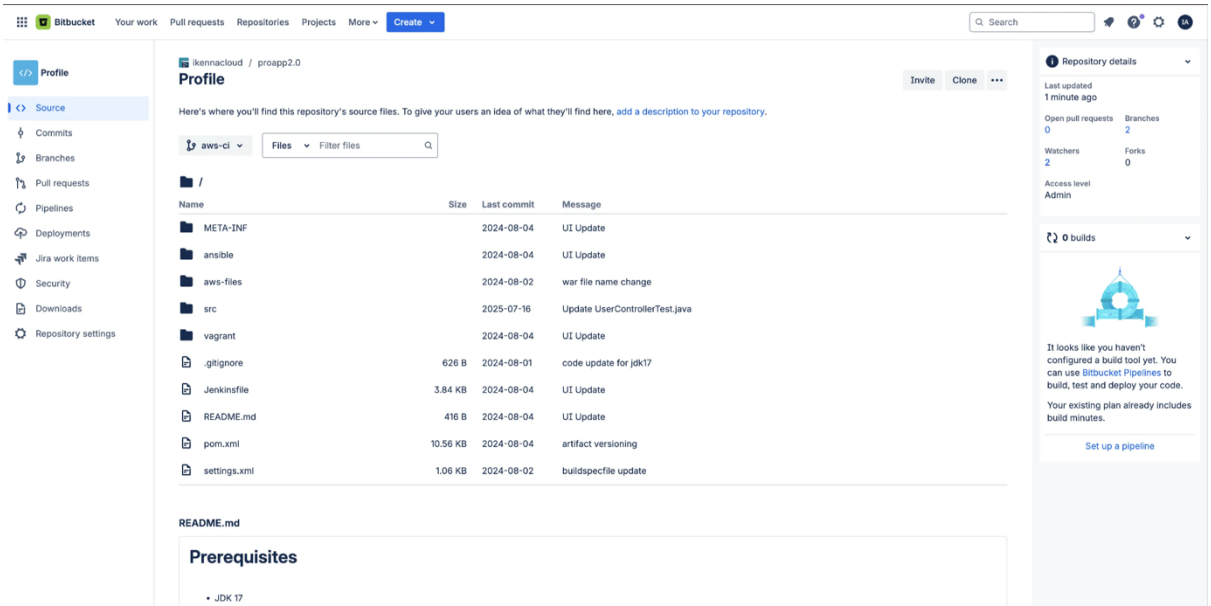
  - Configure Security Group Connections for access and restrictions



- Initialized RDS database schema setup using MySQL scripts

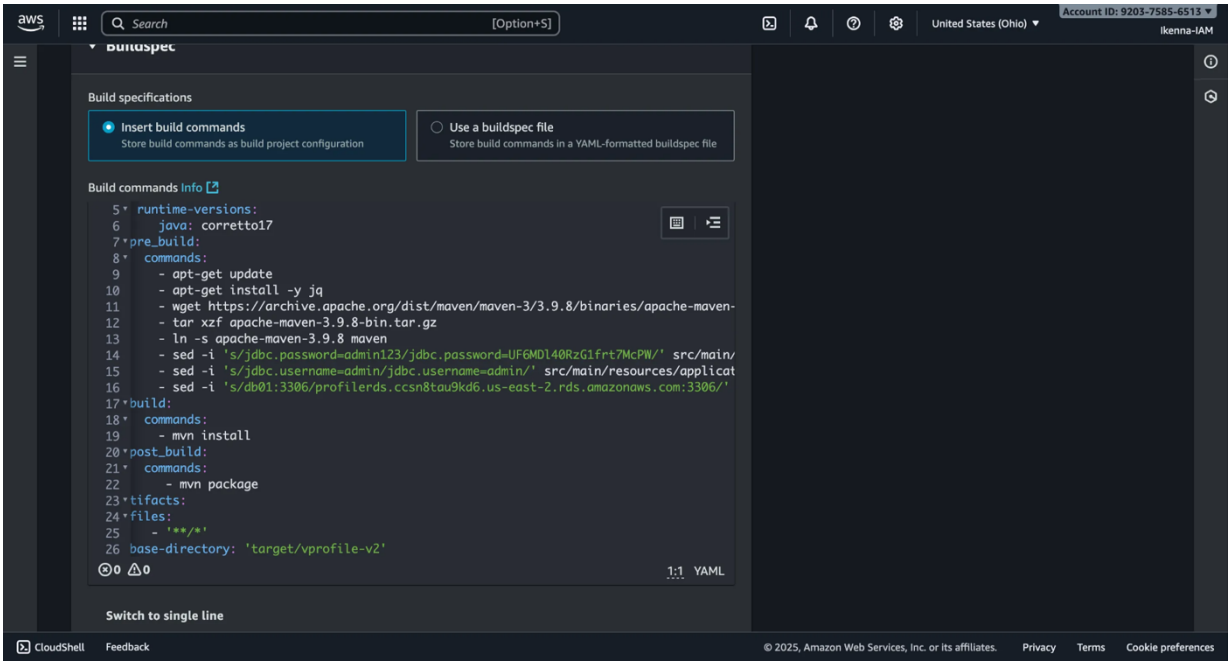
```
--2025-09-27 16:20:43--  https://raw.githubusercontent.com/hkhcoder/vprofile-project/refs/heads/aws-ci/src/main/resources/db_backup.sql
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.108.133, 185.199.111.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7027 (6.9K) [text/plain]
Saving to: 'db_backup.sql.1'

db_backup.sql.1                     100%[=========================================================================>]   6.86K  --.-KB/s    in 0s

2025-09-27 16:20:43 (90.6 MB/s) - 'db_backup.sql.1' saved [7027/7027]

[root@ip-172-31-47-200 ~]# mysql -h profilerds.ccsn8tau9kd6.us-east-2.rds.amazonaws.com -u admin -pUF6MDl40RzG1frt7McPW accounts < db_backup.sql
[root@ip-172-31-47-200 ~]# mysql -h profilerds.ccsn8tau9kd6.us-east-2.rds.amazonaws.com -u admin -pUF6MDl40RzG1frt7McPW accounts
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 142
Server version: 8.0.42 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [accounts]> show tables;
+--------------------+
| Tables_in_accounts |
+--------------------+
| role               |
| user               |
| user_role          |
+--------------------+
3 rows in set (0.002 sec)

MySQL [accounts]> show databases;
+--------------------+
| Database           |
+--------------------+
| accounts           |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.001 sec)

MySQL [accounts]>
```

- Set up BitBucket & Migrate Source code from Github

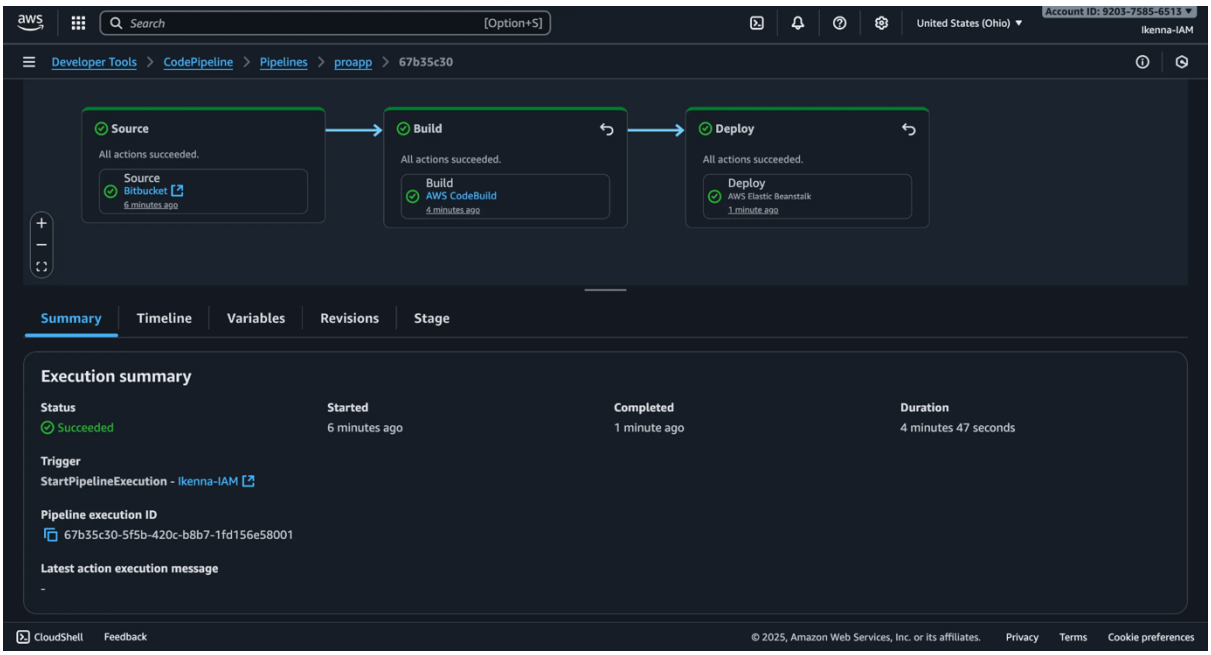


```
apple@MacBookPro vprofile-project % git fetch --tags
apple@MacBookPro vprofile-project % git remote rm origin
apple@MacBookPro vprofile-project % cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        ignorecase = true
        precomposeunicode = true
apple@MacBookPro vprofile-project % git remote add origin git@bitbucket.org:ikennacloud/profile.git
apple@MacBookPro vprofile-project % cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
        ignorecase = true
        precomposeunicode = true
[remote "origin"]
        url = git@bitbucket.org:ikennacloud/profile.git
        fetch = +refs/heads/*:refs/remotes/origin/*
apple@MacBookPro vprofile-project % git pus origin --all
git: 'pus' is not a git command. See 'git --help'.

The most similar commands are
        push
        pull
apple@MacBookPro vprofile-project % git push origin --all
Enumerating objects: 1004, done.
Counting objects: 100% (1004/1004), done.
Delta compression using up to 8 threads
Compressing objects: 100% (550/550), done.
Writing objects: 100% (1004/1004), 12.80 MiB | 13.33 MiB/s, done.
Total 1004 (delta 335), reused 1004 (delta 335), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (335/335), done.
To bitbucket.org:ikennacloud/profile.git
 * [new branch]      aws-ci -> aws-ci
 * [new branch]      local -> local
apple@MacBookPro vprofile-project %
```

- Set up CodeBuild



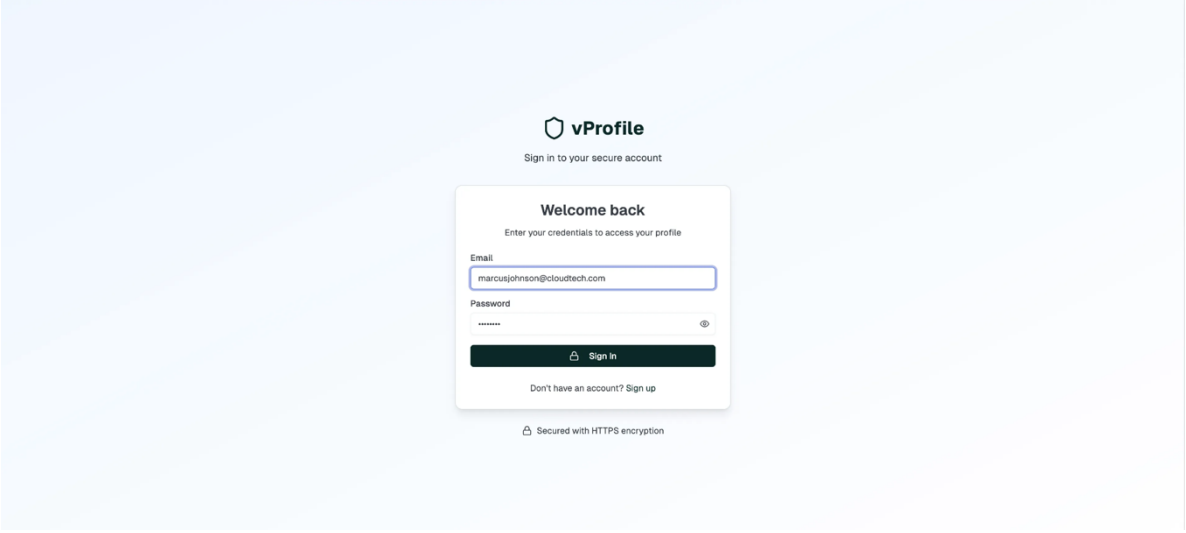
- Set up & Run Code Pipeline

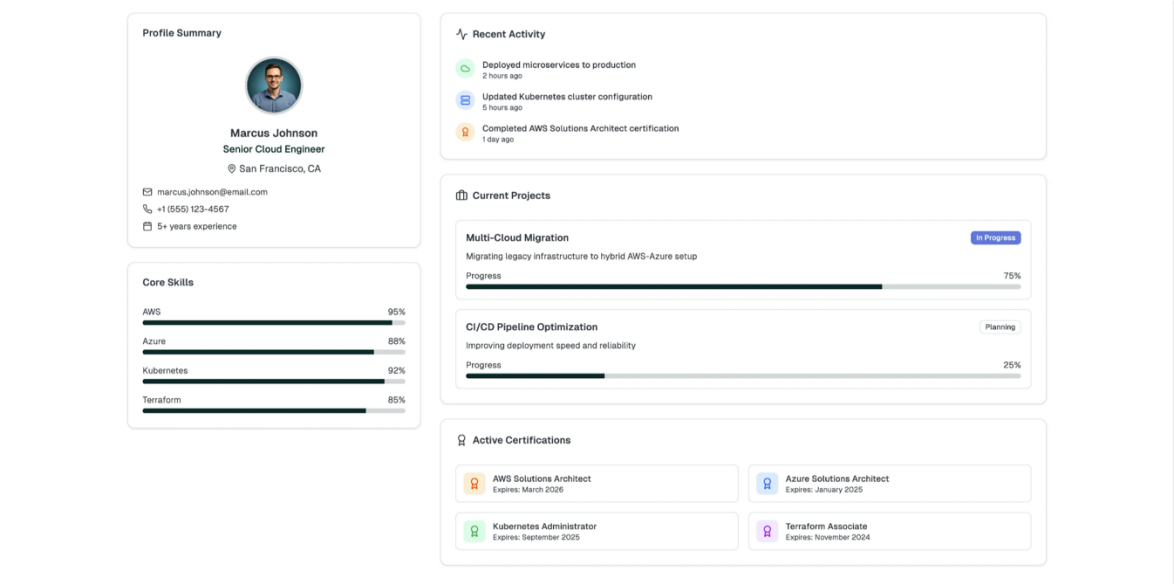


- Test Pipeline

  Working Pipeline Video

- Solution Deployed & Running





- **Key Benefits**
  - **Reduced deployment time by 80%**
    AWS-native CI/CD, cutting deployment time from ~25 minutes to under 5 minutes per release.
  - **Lowered operational workload by 60%**
    Saving ~15+ hours/month in DevOps overhead.
  - **Cut infrastructure costs by ~30%**
    Shifted from self-managed EC2 and databases to AWS-managed services (Elastic Beanstalk, RDS, MQ), reducing idle resource waste.
  - **Achieved 99.99% application availability**
    Used RDS Multi-AZ, Elastic Beanstalk auto-recovery, and CloudWatch monitoring to ensure high uptime and fault tolerance.
- **Future Improvements**
  - Convert infrastructure to IaC using Terraform or AWS CDK
  - Explore container-based deployments using ECS or EKS

This project demonstrates my ability to architect and implement a complete CI/CD pipeline using AWS-native services — integrating Bitbucket for version control, Code Build for compilation, and Elastic Beanstalk for deployment. By automating the build-test-deploy workflow end-to-end, I reduced release time by 80%, eliminated operational bottlenecks, and delivered a scalable, secure, and production-ready application infrastructure.