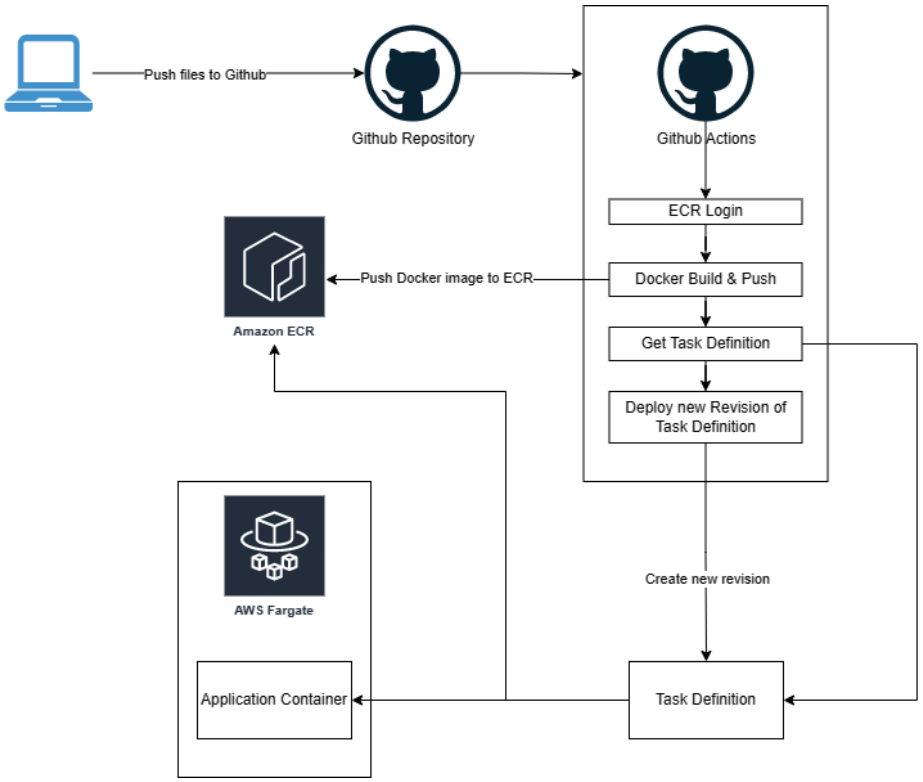
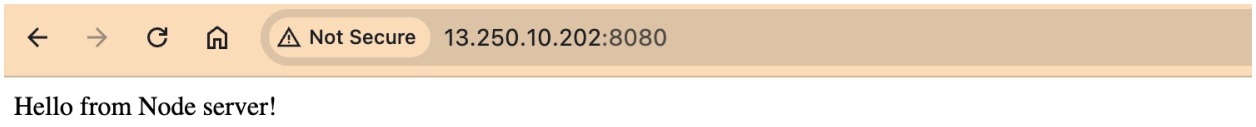
Assignment: CI/CD to AWS Fargate using Terraform + GitHub Actions

You are working as a **cloud engineer in a DevOps team**. Your team has been asked to deploy a containerized Node.js application to AWS in a **repeatable, automated, and production-ready way**.
The company requires all infrastructure to be defined using **Terraform** and all deployments to be automated using **GitHub Actions**.
Every code change pushed to the main branch must automatically build a Docker image, store it in **Amazon ECR**, and deploy it to **ECS Fargate** with minimal downtime.
Security, automation, and clean infrastructure design are key expectations.
Your task is to design and implement this end-to-end solution, following real-world cloud infrastructure and DevOps best practices.



- Provision AWS infrastructure using Terraform: VPC, ECS Fargate cluster, ECR, IAM, and ALB(optional).
    - Hint: Creating the ECS cluster use the public ECS terraform module
- Create a simple Node.js/Flask application that listens on port 8080
- Containerize the application using Docker and test it locally.
- Push the source code and Dockerfile to a GitHub repository.
- Configure GitHub Actions to trigger on every push to the main branch.
- Authenticate GitHub Actions to AWS using OIDC or GitHub Secrets.
- Build the Docker image and push it to Amazon ECR.
- Update the ECS task definition with a new revision using the new image tag.
- Deploy the updated task definition to the ECS Fargate service and wait for stability.

You should be able to access your deployed service through a public IP.



This is a simple web application; however, you may choose to implement an application that also **interacts with other AWS services** if you wish to extend the functionality.(e.g: Including a File upload functionality)

Marking Scheme (Total: 100 marks)

- Terraform Infrastructure – 40
  ECS Fargate, ECR, IAM, networking, and security groups.
- Application & Docker – 10
  Node.js app on port 8080, Dockerfile
- CI/CD (GitHub Actions) – 30
  Secure AWS auth, build & push to ECR, deploy to ECS.
- Documentation – 10
  Clear README and deployment evidence.
- Advanced / Forward-Thinking Implementation – 10 marks
  Eg:- Additional enhancements such as Application Load Balancer integration with health checks and ECS service access via port 80, or any other well-justified improvements.

Total: 100 marks

Objectives of the Assignment (Cloud Infrastructure Design)

By completing this assignment, students will be able to:

1. **Design cloud-native infrastructure** on AWS using ECS Fargate and supporting services.
2. **Apply Infrastructure as Code (IaC)** principles using Terraform to build repeatable, version-controlled infrastructure.
3. **Understand container-based architectures**, including task definitions, services, and networking.
4. **Implement CI/CD pipelines** with GitHub Actions for automated build and deployment.
5. **Apply cloud security best practices**, including Security groups, secret handling, custom task execution roles(if needed).
6. **Deploy and operate scalable workloads**, with optional use of an Application Load Balancer.
7. **Demonstrate production-oriented thinking**, such as health checks, logging.

Submission Guidelines

Students must submit **one ZIP file or one GitHub repository link** containing the following:

1. **GitHub Repository Link**

    - Public or shared access
    - Contains Terraform code, application code, and GitHub Actions workflow

Required Repository Structure

```
/terraform
/app
/.github/workflows
README.md
```

2. **README.md (Mandatory)**

    - Architecture overview (diagram or explanation)
    - Explanation of the CI/CD pipeline
    - How to test the application (URL or public IP + /health)

3. **Evidence of Working Deployment**

    - Screenshots of:
        - Successful GitHub Actions run
        - ECS service with running tasks
        - ECR repository with image tags
        - Application accessible via browser or curl
        - Include your name in the webpage you develop

4. **Access & Security**

    - No AWS credentials committed to the repository
    - Secrets handled via GitHub Secrets or OIDC

Deadline: 25th of Jan
This assignment will contribute **10 marks** toward your overall module assessment.