## Introduction

Dear Mr Jitta,

Thank you for taking the time to participate in this DevOps task. We're excited to see your approach to implementing a GitOps workflow using Flux on Kubernetes. This task is designed to test your skills in managing Kubernetes clusters and automating deployment processes in a secure, scalable, and efficient manner.

## Objectives

Your main objective is to set up Flux on a Kubernetes cluster and configure it to manage the deployment of applications automatically from a Git repository. You will demonstrate best practices in security, scalability, and maintainability.

## Task Details

### Part 1: Cluster Setup

- **Provision a Kubernetes Cluster:** Set up a Kubernetes cluster using Minikube, Kind, or any cloud provider like AWS EKS, Google GKE, or Azure AKS. Make sure to do this using Terraform (no restrictions on providers).

- **Ensure Cluster Security:** Apply best practices for Kubernetes security (e.g., use RBAC, Network Policies, and Pod Security Policies).

### Part 2: Install Flux

- **Install Flux on the Cluster:** Use Terraform to bootstrap the Kubernetes cluster to your repository.

- **Configure Repository:** Initialize a new Git repository with configurations and Kubernetes manifests that Flux will synchronize.

- **Verify Installation:** Ensure that Flux is properly managing the resources in your Kubernetes cluster by pushing a change to your repository and observing that Flux applies the change.

### Part 3: Application Deployment

- **Prepare Application Manifests:** Create Kubernetes manifests for a simple web application (you can use a sample image like `nginx` or `httpd` or `echo`).

- **Deploy Using Flux:** Commit these manifests to your Git repository in a structured directory (e.g., `staging` and `production` environments).

- **Implement Automated Rollbacks:** Configure Flux to automatically rollback deployments if they fail health checks and also to automatically upgrade the app version on patch/bugfix increment.

### Part 4: Monitoring and Logging

- **Set up Monitoring Tools:** Integrate Prometheus and Grafana for monitoring your Kubernetes resources. You can use whatever monitoring stack you are most comfortable, no need to add alerting and custom integrations, just the basics.

- **Configure Logging:** Use Fluentd or a similar log extraction/aggregation tool of your choice to aggregate logs from the Kubernetes nodes and pods.

### Part 5: Documentation

- **Document Your Setup:** Provide a README.md in your repository detailing the steps you took, issues faced, and how you resolved them. Include screenshots or diagrams where applicable.

## Best Practices and Bonus Points Criteria

- **Code Quality:** Clear, maintainable code with comments. A full destruction/creation of the cluster without manual steps is a good example.

- **Security:** Adherence to security best practices throughout the setup. Bonus points for using a VM with a CIS compliant base image.

- **Scalability:** Demonstrate how your setup can scale efficiently based on CPU.

- **Resilience:** Implement fault-tolerance and automatic rollback capabilities.

- **Documentation:** Comprehensive documentation that explains your process and choices.

## Submission

Please submit your GitHub/GitLab/other repository URL. Ensure that your repository is public.

We look forward to your innovative solutions and thank you again for participating in this challenge. Good luck!