**DevOps Project Documentation – Kai Cyber**

**Overview:**

This project demonstrates a comprehensive DevOps pipeline using Azure services, covering infrastructure provisioning, CI/CD pipelines, monitoring, and data processing. The purpose is to build and manage a scalable deployment pipeline for a sample application while ensuring monitoring and security best practices.

**Project Structure:**

The project is divided into four parts:

1. **Infrastructure as Code (IaC)**: Provision Azure infrastructure using Terraform.
2. **CI/CD Pipeline**: Automate testing, containerization, deployment, and rollback using Azure Pipelines.
3. **Monitoring Solution**: Implement application and infrastructure monitoring using Azure Monitor, Log Analytics, and Grafana.
4. **Data Processing**: Set up a database and process vulnerability data using SQL.

**Part 1: Infrastructure as Code**

**Objective**

Use Terraform to provision the following infrastructure in Azure:

* A virtual network with public and private subnets.
* An AKS (Azure Kubernetes Service) cluster.
* A managed database service (e.g., Azure SQL Database).
* Azure Storage for data storage.
* Network Security Groups (NSGs) and appropriate IAM roles.

**Steps:**

1. **Terraform Setup**:
   * Write Terraform configuration files for provisioning resources.
   * Use Terraform modules for reusability and maintainability.
2. **Infrastructure Provisioning**:

Execute the following commands:

* + terraform init
  + terraform plan
  + terraform apply

Include a destroy script:

* + terraform destroy

1. **Verification**:

Screenshot of services

**Part 2: CI/CD Pipeline**

**Objective:**

Automate the build, test, and deployment process using Azure Pipelines.

**Steps**

1. **Pipeline Setup**:
   * Create a pipeline in Azure DevOps.
   * Define a pipeline.yaml file with stages:
     + **Build**: Containerize the sample application.
     + **Test**: Perform unit testing and infrastructure validation.
     + **Security**: Scan the container for vulnerabilities.
     + **Deploy**: Deploy the container to the AKS cluster.
     + **Rollback**: Automate rollback on deployment failure.
2. **Key Features**:
   * Implement branching strategy (e.g., GitFlow).
   * Include quality gates and secret management using Azure Key Vault.
   * Configure triggers for pipeline execution.
3. **Verification**:
   * Provide pipeline run evidence (e.g., screenshots of successful runs):

**Part 3: Monitoring Solution**

**Objective:**

Implement a monitoring solution to track application and infrastructure health, aggregate logs, and trigger alerts.

**Steps**

1. **Metrics Collection**:
   * Use Azure Monitor and Log Analytics to collect metrics and logs from Azure resources.
   * Integrate Azure Kubernetes Service (AKS) with **Container Insights**.
2. **Dashboard Creation**:
   * Set up **Grafana** with Azure Monitor as a data source.
   * Build custom dashboards to visualize metrics, such as:
     + CPU and memory utilization.
     + Application request rates.
     + Error rates.
3. **Alerts**:
   * Configure alerts in Azure Monitor for:
     + High CPU or memory usage.
     + Application performance issues.
     + Error rates above a defined threshold.
4. **Log Retention Policies**:
   * Set log retention policies in Log Analytics and Azure Storage to ensure efficient storage management.
5. **Verification**:
   * Include screenshots of dashboards and alert configurations.

**Part 4: Data Processing**

**Objective:**

Set up a database to store and process vulnerability data, and write queries to analyze it.

**Steps**

1. **Database Setup**:
   * Provision an **Azure SQL Database** using the infrastructure created in Part 1.
   * Create a table for vulnerabilities with columns such as ID, Description, Severity, and DetectedAt.
2. **Data Insertion**:
   * Insert sample vulnerability data into the table.
3. **Query Writing**:
   * Write SQL queries for:
     + Retrieving all high-severity vulnerabilities.
     + Counting vulnerabilities by severity.
     + Identifying recent vulnerabilities (e.g., within the last 7 days).
4. **Verification**:
   * Save all queries in a .sql file.

**Architecture Diagram:**

Include a visual representation of your project architecture showing:

1. Virtual network with subnets.
2. AKS cluster connected to the managed database.
3. Azure Storage and Log Analytics Workspace for monitoring.
4. CI/CD pipeline interacting with the infrastructure.



**Technologies Used**

* **Infrastructure**: Azure, Terraform
* **CI/CD**: Azure DevOps Pipelines
* **Monitoring**: Azure Monitor, Log Analytics, Grafana
* **Database**: Azure SQL Database
* **Scripting Languages**: SQL, YAML, Terraform