CI/CD Pipeline Setup, Kubernetes and AWS Setup, Monitoring and Logging Configuration

1. CI/CD Pipeline Setup:

Objective: Implement a continuous integration and continuous deployment pipeline to automate the building, testing, and deployment of the frontend application.

Steps:

1. Version Control: Store the frontend application code in a version control system like Git.
2. Choose CI/CD Tool: Select a CI/CD tool such as GitHub Actions or Jenkins for automation.
3. Create Workflow File: In the repository, create a .github/workflows/main.yml file for GitHub Actions or a Jenkinsfile for Jenkins.
4. Define Workflow: Define jobs in the workflow file for building, testing, and deploying the frontend application.
5. Build Job: Use Docker to build the frontend application image.
6. Test Job: Run automated tests to ensure the application functions correctly.
7. Deploy Job: Use Kubernetes manifests or Helm charts to deploy the application to the Kubernetes cluster on AWS (EKS).
8. Access Control: Ensure appropriate access controls and secrets management for CI/CD pipelines.

2. Kubernetes and AWS Setup:

Objective: Set up a Kubernetes cluster on AWS (EKS) to host the frontend application.

Steps:

1. AWS Account: Sign in to the AWS Management Console or create a new account.
2. Navigate to EKS: Go to the Amazon EKS service in the AWS Management Console.
3. Create Cluster: Click on "Create Cluster" and follow the wizard to define cluster name, version, networking settings, and IAM role.
4. Wait for Provisioning: Wait for AWS to provision the EKS cluster, which may take several minutes.
5. Kubernetes Tools: Install and configure kubectl and aws-iam-authenticator on your local machine to interact with the EKS cluster.
6. Deploy Application: Use Kubernetes manifests or Helm charts to deploy the frontend application to the EKS cluster.

3. Monitoring and Logging Configuration:

Objective: Implement monitoring and logging for the frontend application and EKS cluster using AWS CloudWatch.

Steps:

1. Enable Logging for EKS Cluster: In the AWS Management Console, navigate to the Amazon EKS service, select your cluster, and enable logging for API server, audit, controller manager, or scheduler logs.
2. Deploy CloudWatch Agent: Create a ConfigMap and deploy the CloudWatch agent as a DaemonSet in the Kubernetes cluster to collect logs from application containers.
3. Set Up CloudWatch Alarms: In the AWS Management Console, navigate to the CloudWatch service, create alarms to monitor metrics like CPU, memory, and response time, and define thresholds and actions to trigger alerts in case of anomalies.
4. Configure Metrics: Use CloudWatch Container Insights to collect and analyze container-level metrics.
5. Access Logs and Metrics: Access logs and metrics in the CloudWatch Logs and CloudWatch Metrics sections of the AWS Management Console.