Design, build, and maintain. You must be able to break things down at a high level.

Dockerized Microservices Application with Kubernetes

## Phase 1: Define the Project Scope

#### Understand the requirements and deliverables:

* + Containerize a shopping cart app (3–4 microservices).
  + Deploy it on Kubernetes.
  + Implement a service mesh (Istio or Linkerd).
  + Set up CI/CD for automated deployments.
  + Configure monitoring and centralized logging.

#### Decide the Tech Stack:

* + Use Docker for containerization.
  + Kubernetes for orchestration (using Minikube, EKS, or AKS).
  + Choose a service mesh: Istio for advanced traffic control and security.
  + Use Jenkins or GitLab CI for CI/CD pipelines.
  + Monitoring: Prometheus + Grafana.
  + Logging: Loki or ELK stack.

## Phase 2: Prepare the Development Environment

#### Set up your local environment:

* + Install Docker and Kubernetes (e.g., Minikube or Kubernetes on cloud like EKS/AKS).
  + Install kubectl, Helm, and Istio CLI.
  + Install Prometheus and Grafana (optional for now).

#### Set up CI/CD tool:

* + Install Jenkins or configure GitLab CI.

#### Optional: Set up cloud resources:

* + If using AWS, create an EKS cluster.
  + For Azure, create an AKS cluster.

#### Install supporting tools:

* + Install Helm for Kubernetes package management.
  + Install Prometheus and Grafana locally or in the cluster.

## Phase 3: Develop and Containerize the Microservices

#### Design the application:

* + Identify services for the shopping cart app (e.g., product service, user service, cart service, order service).
  + Decide the APIs for communication between microservices.

#### Develop microservices:

* + Write each service in your preferred language (Node.js, Python, etc.).
  + Include a Dockerfile for each service.

#### Test locally:

* + Run each service independently using Docker Compose or directly with Docker.

## Phase 4: Deploy to Kubernetes

#### Set up Kubernetes resources:

* + Write YAML manifests for each microservice (Deployments, Services, ConfigMaps, etc.).
  + Test local Kubernetes deployment using Minikube or Kind.

#### Deploy to the cluster:

* + Apply the Kubernetes manifests to deploy the microservices.

#### Verify deployment:

* + Ensure all pods are running using kubectl get pods.
  + Expose services using NodePort/LoadBalancer.

## Phase 5: Add the Service Mesh

#### Install Istio or Linkerd:

* + Use Helm or the CLI to install Istio/Linkerd on the cluster.

#### Integrate the service mesh:

* + Inject Istio sidecars into the microservices.
  + Configure traffic policies (e.g., rate limiting, retries).

#### Secure communication:

* + Enable mutual TLS (mTLS) between services.

#### Add traffic control:

* + Set up advanced routing policies (e.g., canary deployments).

## Phase 6: Set Up CI/CD Pipelines

#### Configure Jenkins/GitLab CI:

* + Write pipeline scripts for building, testing, and deploying the microservices.

#### Automate deployment:

* + Set up GitOps or push Docker images and deploy with Kubernetes manifests.

#### Test the pipeline:

* + Validate CI/CD by making a code change and checking auto-deployment.

## Phase 7: Set Up Monitoring and Logging

#### Install monitoring tools:

* + Deploy Prometheus and Grafana for metrics.
  + Create dashboards in Grafana for visualizing cluster and application health.

#### Set up centralized logging:

* + Deploy Loki/ELK stack to capture logs from all services.

#### Validate observability:

* + Trigger application traffic and verify metrics and logs.

## Deliverables Checklist:

* A working shopping cart app containerized with Docker.
* Microservices deployed on Kubernetes.
* Service mesh configured with advanced traffic and security.
* CI/CD pipelines for automated deployments.
* Monitoring and logging dashboards.
* Comprehensive documentation.

# Project Workflow (SDLC Phases Mapped)

|  |  |
| --- | --- |
| **SDLC Phase** | **Actions** |
| Plan | Define project scope, deliverables and objectives |
| Design | Architect the microservices and their interactions (API design, db schema) |
| Implement | Write the code for microservices and containerize with Docker |
| Test | Test microservices independently (unit test) & together (integration test) |
| Deploy | Deploy microservices to Kubernetes using YAML manifests |
| Maintain | Monitor, log and enhance the app with observability tools and CI/CD pipelines |

## Tech Stack:

|  |  |
| --- | --- |
| **Tool/Technology** | **Purpose** |
| Docker | Containerizes the Microservices |
| Kubernetes | Orchestrates and manages containers |
| Istio/Linkerd | Adds service mesh capabilties |
| Jenkins/GitLab | Automates builds and deployments |
| Prometheus (optional) | Collects and monitors metrics |
| Grafana (optional) | Visualizes metrics and dashboards |
| Loki/ELK | Collects and centralizes logs |

**Install Prerequisite Tools** **Tools Needed:**

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| Docker | Containerize microservices |
| Kubernetes | Orchestration platform (MiniKube) |
| Kubectl | CLI to manage K8s clusters |
| Helm | Manages K8s resources (e.g., Istio) |

|  |  |
| --- | --- |
| Code Editor (optional) | Writing code and YAML files |
| Prometheus/Grafana (optional) | Monitoring and dashboards |
| Loki/ELK | Centralized logging |