DevOps Course - Helm Assignment:

Helm - Conceptual Questions

1. Explain Helm's role in Kubernetes

Helm is a package manager for Kubernetes that allows you to manage complex applications using reusable templates. It simplifies deployment by packaging Kubernetes manifests into a single, versioned unit called a Helm chart.

Helm is preferred over plain YAML files because it:

- Centralizes management of multiple Kubernetes resources.
- Enables templating and parameterization for different environments.
- Reduces duplication and manual editing of YAMLs.

Key components of a Helm chart:

- Chart.yaml: Metadata about the chart (name, version, dependencies, etc.).
- values.yaml: Default configuration values for the templates.
- templates/: Templates of Kubernetes manifests.
- charts/: Folder for dependencies (subcharts).

2. Environment-specific Configurations

Helm handles environment-specific configurations by allowing multiple values-<env>.yaml files. These override or extend the default values.yaml file during deployment. This enables using the same chart in multiple environments without duplicating code.

Example:

```
helm install nissim-app ./mychart -f values-prod.yaml
helm upgrade nissim-app ./mychart -f values-dev.yaml
helm upgrade nissim-app ./mychart -f values-nissim.yaml
```

This approach supports reusability and keeps the deployment logic centralized.

3. Helm Chart Repositories

A Helm chart repository is a location where packaged Helm charts (.tgz files) and an index.yaml file are stored. Repositories allow sharing and installing charts just like package managers (e.g., apt, npm).

Common hosting options:

- Artifact Hub
- GitHub Pages
- ChartMuseum

A private Helm repository allows organizations to:

- Maintain personalized charts for internal apps.
- Centralize updates and deployment logic.
- Improve security and version control.

4. CI/CD Integration

Helm is integrated into CI/CD pipelines to automate testing and deployment of Kubernetes applications.

Typical steps:

CI Phase:

- Run helm lint and helm template to validate chart structure.
- Create a temporary environment using helm install for integration/load testing.
- Run application-level tests against the deployed environment.

CD Phase:

- After passing tests, deploy to the desired environment using:

```
helm upgrade nissim-app ./mychart -f values-prod.yaml
```

- Use service account or kubeconfig in the CI/CD tool for Kubernetes access.

Artifact Management:

- Package the chart using helm package.
- Upload the .tgz file to a chart repository.
- Use versioning for traceability and rollback support.