**Helm and Kubernetes Basics**

**I. Helm: A Kubernetes Package Manager**

**What is Helm?**

Helm is a tool that helps manage Kubernetes applications. It works like a package manager (similar to apt or yum) but is designed for Kubernetes. Helm makes it easy to define, install, and upgrade applications by bundling multiple Kubernetes resources into one unit.

**Key Components of Helm:**

1. Helm Client: A command-line tool used to interact with Helm.

2. Helm Chart: A package containing all the configuration files needed to deploy an app (like a template).

3. Helm Repository: A storage location where Helm charts are shared and stored.

4. Helm Release: An instance of a chart running in a Kubernetes cluster. Installing a chart creates a release.

**Understanding Helm Charts:**

A Helm chart includes files that describe a Kubernetes application:

- Chart.yaml: Contains metadata about the chart.

- values.yaml: Holds default configuration values.

- templates/: A folder with templates for Kubernetes resources.

**Advantages of Using Helm:**

- Simplified Deployments: Deploy complex apps easily.

- Version Control: Track changes, upgrade, or revert versions effortlessly.

- Reusable Configurations: Create charts once and use them in different environments.

- Rollback Capabilities: Quickly revert to previous versions when needed.

- Consistency: Ensure the same application setup across different clusters.

**II. Basic Helm Commands**

**These commands are used with the Helm client:**

**1. Install a Chart:**

helm install <release-name> <chart-name>

Installs an application with a specified release name.

**2. List Installed Charts:**

helm list

Displays all currently installed applications (releases).

**3. Upgrade a Chart:**

helm upgrade <release-name> <chart-name>

Updates an existing application to a new version.

**4. Rollback to a Previous Version:**

helm rollback <release-name> <revision>

Reverts the application to a previous version.

**5. Uninstall a Chart:**

helm uninstall <release-name>

Removes an installed application.

**6. View Release History:**

helm history <release-name>

Shows the history of changes for a specific release.

**III. Kubernetes Concepts: ReplicaSets and Services**

**ReplicaSets:**

**- What Are They?**

ReplicaSets ensure a specific number of identical Pods (containers running your application) are always running.

**- Purpose:**

To maintain application stability and high availability.

**- Key Points:**

- Used by Deployments: Deployments use ReplicaSets to manage Pods.

- Selectors: ReplicaSets identify which Pods to manage using labels.

**Services:**

**- What Are They?**

Services define how to access a group of Pods. They provide stable IPs, DNS names, and load balancing, even if Pods are restarted.

**- Purpose:**

To make applications accessible both within and outside the Kubernetes cluster.

**- Types of Services:**

1. ClusterIP (Default): Exposes the service within the cluster.

2. NodePort: Exposes the service on each node’s IP using a specific port.

3. LoadBalancer: Uses an external load balancer (usually provided by cloud providers).

4. ExternalName: Maps the service to an external DNS name.