**Question 1**

Explain Helm’s role in Kubernetes.

1. Why is Helm preferred over managing plain Kubernetes YAML files?
2. List and describe the key components of a Helm chart.

Answer a:

* **Templating**: allows to customize deployments per environment

(development, staging, production).

* **Reusability**: Helm charts can be deployed in multiple environments.
* **Versioning**: keeps a release history and allows easy rollback to earlier versions.
* **Dependencies**: automatically deploy dependencies.
* **Automation**: automates deployment with a single command

" helm upgrade -i [Release-Name] [path] ".

Answer b:

A Helm chart is a directory with:

* **Chart.yaml**: metadata such as name, version, description, and API version.
* **values.yaml**: default configuration values for the chart. can be overridden.
* **templates/:** Kubernetes manifest templates. Helm fills these templates when deploying.
* **Optional:**
  + **charts/**: contains subcharts.
  + **Chart.lock**: o locks the versions of dependencies to ensure repeatable deployments.
  + **templates/tests/**: run after installation to verify the deployment is working.
  + **README.md**: documentation.

**Question 2**

Environment-specific Configurations:

How does Helm handle environment-specific configurations? Provide an example.

Answer:

Helm handles environment-specific configurations by overriding the default values.

Step by step:

1. create separate YAMLs for each environment, for example:

values-dev.yaml for development

values-staging.yaml for staging

values-prod.yaml for production

1. override the defaults using the -f:

" helm upgrade -i myrelease ./mychart **-f values-dev.yaml "**

" helm upgrade -i myrelease ./mychart **-f values-prod.yaml "**

Example

values-dev.yaml

replicaCount: 1

image:

repository: myapp

tag: dev

service:

type: ClusterIP

port: 8080

values-prod.yaml

replicaCount: 3

image:

repository: myapp

tag: latest

service:

type: LoadBalancer

port: 80

Deployment template (templates/deployment.yaml)

apiVersion: apps/v1

kind: Deployment

metadata:

name: {{ .Release.Name }}

spec:

replicas: {{ .Values.replicaCount }}

template:

spec:

containers:

* + - * name: app

image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"

ports:

* + - * + containerPort: {{ .Values.service.port }}

**Question 3**

Helm Chart Repositories:

What is a Helm chart repository, and how can it be hosted? List at least three hosting options.

Answer:

A **Helm chart repository** is a collection of Helm charts that can be shared and installed by others.

Charts can be hosted in several ways, such as:

* + - public repositories like Artifact Hub
    - cloud storage services like Amazon S3
    - private Git repositories

**Question 4**

CI/CD Integration:

How can Helm be integrated into a CI/CD pipeline? Explain the typical steps involved.

Answer:

Helm can be integrated into a CI/CD pipeline to automate the deployment of Kubernetes applications.

Typical steps involved:

1. Docker image is pushed to a container registry.
2. Helm chart is packaged and optionally tested.
3. Helm deploys the application to the Kubernetes cluster with " helm upgrade -i ", allowing environment-specific configurations.
4. Helm enables rollbacks if a deployment fails.