Helm Chart Structure

A Helm Chart is a package that contains Kubernetes resource definitions, along with templates and values to allow dynamic configuration of resources

Typical Helm Chart Structure:

```
my-chart/
  ├── charts/            # Subcharts (dependencies)
  ├── templates/         # Resource definitions (with templates)
  ├── values.yaml        # Default values
  ├── Chart.yaml         # Chart metadata
```

Variables

You can use the values defined in the values.yaml file in the template.

values.yaml:

```
replicaCount: 2
image:
  repository: nginx
  tag: latest
```

Deployment Template (in templates/deployment.yaml):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-deployment
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app: {{ .Release.Name }}
  template:
    metadata:
      labels:
        app: {{ .Release.Name }}
    spec:
      containers:
        - name: {{ .Values.image.repository }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
```

Conditional Statements (if-else)

You can use if-else statements to handle conditional logic in your templates.

```
{{- if .Values.replicaCount }}
replicas: {{ .Values.replicaCount }}
{{- else }}
replicas: 1
{{- end }}
```

Ranges (Loops)

You can iterate over collections (like lists or maps) using range:

```
{{- range .Values.pods }}
- name: {{ .name }}
  image: {{ .image }}
{{- end }}
```

Built-in Objects in Helm Templates

Helm provides several built-in objects that are useful for templating.

.Release.Name: The release name (e.g., my-release).

.Release.Namespace: The namespace of the release.

.Values: The values defined in values.yaml or passed via --set at install/upgrade time.

.Chart.Name: The name of the chart.

.Chart.Version: The version of the chart.

.Chart.AppVersion: The application version.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-deployment
spec:
  replicas: {{ .Values.replicaCount | default 1 }}
  template:
    spec:
      containers:
        - name: {{ .Values.image.repository }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
```

```
# kubectl config commands

# 1. Set a Cluster in the kubeconfig file
kubectl config set-cluster <cluster-name> --server=<api-server-url> --certificate-authority=<path-to-ca-cert>

# 2. Set a User in the kubeconfig file
kubectl config set-credentials <user-name> --client-certificate=<path-to-client-cert> --client-key=<path-to-client-key>

# 3. Set a Context in the kubeconfig file
kubectl config set-context <context-name> --cluster=<cluster-name> --user=<user-name> --namespace=<namespace>

# 4. Switch to a Context
kubectl config use-context <context-name>

# 5. View the Current Context
kubectl config current-context

# 6. List All Contexts
```

```
kubectl config get-contexts

# 7. Delete a Context
kubectl config delete-context <context-name>

# 8. View the kubeconfig file
kubectl config view

# 9. Set the Namespace in the Current Context
kubectl config set-context --current --namespace=<namespace-name>

# kubectl commands for resource management

# 10. Create Resources from a YAML file
kubectl apply -f <resource-file>.yaml

# 11. Get All Resources in the current namespace
kubectl get all

# 12. Get a specific resource
kubectl get <resource-type> <resource-name>

# 13. Describe a resource
kubectl describe <resource-type> <resource-name>

# 14. Delete a resource
kubectl delete <resource-type> <resource-name>

# 15. View Logs of a Pod
kubectl logs <pod-name>

# 16. Execute command in a Pod
kubectl exec -it <pod-name> -- <command>

# 17. Use Go templates to fetch information (e.g., pod name, status)
kubectl get pods -o go-template='{{range .items}}{{.metadata.name}}{{"\n"}}
{{end}}'
kubectl get pod <pod-name> -o go-template='{{.status.phase}}'

# Helm commands

# 18. Install a Helm chart
helm install <release-name> <chart-name>

# 19. Upgrade a Helm release
helm upgrade <release-name> <chart-name>

# 20. Uninstall a Helm release
helm uninstall <release-name>

# 21. List installed Helm releases
helm list

# 22. Create a new Helm chart
helm create <chart-name>

# 23. Lint a Helm chart
helm lint <chart-directory>

# 24. Install a Helm chart with custom values
helm install <release-name> <chart-name> --set key=value

# 25. Upgrade a Helm release with custom values
helm upgrade <release-name> <chart-name> --set key=value
```