```yaml
# Kubernetes Helm Chart Scripts with Summaries

# 1. Helm Chart for Deploying a Simple Web Application
# Summary:
# This script deploys a basic web application using Nginx. It defines Kubernetes
Deployment and Service manifests, allowing the application to be accessible
externally via a LoadBalancer service type.

---
apiVersion: v2
name: nginx-web-app
version: 0.1.0

# Chart Values
values:
  replicaCount: 3
  image:
    repository: nginx
    tag: latest
  service:
    type: LoadBalancer
    port: 80

# Deployment Template
templates:
  - apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: nginx-deployment
    spec:
      replicas: {{ .Values.replicaCount }}
      selector:
        matchLabels:
          app: nginx
      template:
        metadata:
          labels:
            app: nginx
        spec:
          containers:
            - name: nginx
              image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
              ports:
                - containerPort: 80

# Service Template
  - apiVersion: v1
    kind: Service
    metadata:
      name: nginx-service
    spec:
      type: {{ .Values.service.type }}
      selector:
```

```
      app: nginx
    ports:
      - protocol: TCP
        port: {{ .Values.service.port }}
        targetPort: 80
```

# 2. Helm Chart for Configuring a Stateful Application

# Summary:

# This script creates a stateful application for MySQL. It provisions a PersistentVolumeClaim (PVC) for data persistence, ensuring storage is retained across pod restarts.

```
---
apiVersion: v2
name: mysql-stateful-app
version: 0.1.0

# Chart Values
values:
  image:
    repository: mysql
    tag: 8.0
  storage:
    size: 10Gi
  service:
    type: ClusterIP
    port: 3306

# StatefulSet Template
templates:
  - apiVersion: apps/v1
    kind: StatefulSet
    metadata:
      name: mysql-statefulset
    spec:
      serviceName: mysql
      replicas: 1
      selector:
        matchLabels:
          app: mysql
      template:
        metadata:
          labels:
            app: mysql
        spec:
          containers:
            - name: mysql
              image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
              env:
                - name: MYSQL_ROOT_PASSWORD
                  valueFrom:
```

```
          secretKeyRef:
           name: mysql-secret
           key: root-password
       ports:
        - containerPort: {{ .Values.service.port }}
       volumeMounts:
        - name: mysql-persistent-storage
         mountPath: /var/lib/mysql
   volumeClaimTemplates:
    - metadata:
      name: mysql-persistent-storage
     spec:
      accessModes: ["ReadWriteOnce"]
      resources:
       requests:
        storage: {{ .Values.storage.size }}


---


# 3. Helm Chart for Autoscaling Application Deployment
# Summary:
# This script defines a Horizontal Pod Autoscaler (HPA) for a web application, ensuring the number of pods dynamically adjusts based on CPU usage thresholds.


---
apiVersion: v2
```

```yaml
name: autoscaling-web-app
version: 0.1.0

# Chart Values
values:
  replicaCount: 2
  image:
    repository: web-app
    tag: latest
  resources:
    requests:
      cpu: 100m
      memory: 128Mi
    limits:
      cpu: 250m
      memory: 256Mi
  hpa:
    enabled: true
    minReplicas: 2
    maxReplicas: 10
    targetCPUUtilizationPercentage: 80

# Deployment Template
templates:
  - apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: web-app-deployment
    spec:
      replicas: {{ .Values.replicaCount }}
      selector:
        matchLabels:
          app: web-app
      template:
        metadata:
          labels:
            app: web-app
        spec:
          containers:
            - name: web-app
              image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
              resources:
                requests:
                  cpu: {{ .Values.resources.requests.cpu }}
                  memory: {{ .Values.resources.requests.memory }}
                limits:
                  cpu: {{ .Values.resources.limits.cpu }}
                  memory: {{ .Values.resources.limits.memory }}

# HPA Template
  - apiVersion: autoscaling/v1
    kind: HorizontalPodAutoscaler
    metadata:
      name: web-app-hpa
    spec:
        scaleTargetRef:
        apiVersion: apps/v1
        kind: Deployment
        name: web-app-deployment
      minReplicas: {{ .Values.hpa.minReplicas }}
      maxReplicas: {{ .Values.hpa.maxReplicas }}
      targetCPUUtilizationPercentage: {{
.Values.hpa.targetCPUUtilizationPercentage }}
```

# 4. Helm Chart for Ingress Controller Deployment

# Summary:

# This script deploys an NGINX Ingress Controller, enabling HTTP and HTTPS routing to Kubernetes services. It sets up a custom ConfigMap for fine-grained control of ingress rules.

```
---
apiVersion: v2
name: nginx-ingress
version: 0.1.0

# Chart Values
values:
  controller:
    replicaCount: 2
    image:
      repository: k8s.gcr.io/ingress-nginx/controller
      tag: v1.6.4
    service:
      type: LoadBalancer
      externalPort: 80

# Deployment Template
templates:
  - apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: nginx-ingress-controller
    spec:
      replicas: {{ .Values.controller.replicaCount }}
      selector:
        matchLabels:
          app: nginx-ingress
      template:
        metadata:
          labels:
            app: nginx-ingress
        spec:
          containers:
```

```
          - name: nginx-ingress-controller
            image: {{ .Values.controller.image.repository }}:{{
.Values.controller.image.tag }}
            args:
              - /nginx-ingress-controller


# Service Template
  - apiVersion: v1
    kind: Service
    metadata:
      name: nginx-ingress-service
    spec:
      type: {{ .Values.controller.service.type }}
      selector:
        app: nginx-ingress
      ports:
        - protocol: TCP
          port: {{ .Values.controller.service.externalPort }}
          targetPort: 80


---


# 5. Helm Chart for Redis Cluster Deployment
# Summary:
# This script deploys a high-availability Redis cluster with master-replica
architecture. It uses StatefulSet for persistent storage and ConfigMap for
cluster configuration.
```

```yaml
---
apiVersion: v2
name: redis-cluster
version: 0.1.0

# Chart Values
values:
  image:
    repository: redis
    tag: 6.2
  replicas: 3
  storage:
    size: 5Gi

# StatefulSet Template
templates:
  - apiVersion: apps/v1
    kind: StatefulSet
    metadata:
      name: redis-cluster
    spec:
      serviceName: redis-headless
      replicas: {{ .Values.replicas }}
      selector:
        matchLabels:
```

```
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
        - name: redis
          image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
          ports:
            - containerPort: 6379
          volumeMounts:
            - name: redis-data
              mountPath: /data
  volumeClaimTemplates:
    - metadata:
        name: redis-data
      spec:
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: {{ .Values.storage.size }}
```

# 6. Helm Chart for Secure Application Deployment with Secrets

```
# Summary:
# This script deploys a secure application with sensitive credentials stored in
Kubernetes Secrets. The deployment template references these secrets to set
environment variables.

---
apiVersion: v2
name: secure-app
version: 0.1.0

# Chart Values
values:
  image:
    repository: secure-app
    tag: latest
  service:
    type: ClusterIP
    port: 8080

# Secret Template
templates:
  - apiVersion: v1
    kind: Secret
    metadata:
      name: secure-app-secret
    type: Opaque
    data:
```

```
    username: {{ .Values.secret.username | b64enc }}
    password: {{ .Values.secret.password | b64enc }}

# Deployment Template
templates:
  - apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: secure-app-deployment
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: secure-app
      template:
        metadata:
          labels:
            app: secure-app
        spec:
          containers:
            - name: secure-app
              image: {{ .Values.image.repository }}:{{ .Values.image.tag }}
              env:
                - name: APP_USERNAME
                  valueFrom:
                    secretKeyRef:
```

```
        name: secure-app-secret
        key: username
  - name: APP_PASSWORD
    valueFrom:
      secretKeyRef:
        name: secure-app-secret
        key: password
ports:
  - containerPort: {{ .Values.service.port }}
```