Helm — Présentation + Labs pratiques

Qu’est-ce que Helm ?

Helm est le gestionnaire de paquets de Kubernetes. Il permet de :

- Modéliser des applications via des charts (paquet contenant manifests + templates + valeurs).
- Paramétrer facilement par environnement via values.yaml.
- Déployer / mettre à jour / rollback des releases versionnées.
- Réutiliser des composants (dépendances), tester via helm test, automatiser via hooks.

Concepts clés :

- Chart : dossier avec Chart.yaml, values.yaml, templates/.
- Release : instance déployée d’un chart dans un namespace.
- Repo de charts : source distante de charts (ex. Bitnami).
- Values : variables de configuration injectées dans les templates.

Prérequis (pour tous les labs)

- Un cluster K8s accessible (kubectl get nodes OK) – minikube/kind/cluster existant.
- Helm v3 installé (helm version).
- Contexte/namespace de travail (on utilisera demo).

```
kubectl create namespace demo

kubectl config set-context --current --namespace=demo
```

Lab 1 — Premier déploiement avec un chart public (NGINX)

Objectif : ajouter un repo, chercher un chart, déployer, vérifier.

1) Ajouter un repo et le mettre à jour

```
helm repo add bitnami https://charts.bitnami.com/bitnami

helm repo update
```

2) Chercher un chart

```
helm search repo nginx
```

3) Installer une release

```
helm install web bitnami/nginx --set service.type=ClusterIP
```

4) Vérifier

```
helm ls

kubectl get pods,svc

kubectl port-forward svc/web-nginx 8080:80

# Ouvrir http://localhost:8080
```

À retenir : helm install <release> <chart>, helm ls, helm uninstall.

Lab 2 — Créer votre propre chart « hello-app »

Objectif : générer un squelette, personnaliser values.yaml, ajouter un ConfigMap, monter dans NGINX.

1) Générer le squelette

```
helm create hello-app

tree hello-app
```

Modifiez hello-app/values.yaml :

```
replicaCount: 1


image:
  repository: bitnami/nginx
  tag: latest
  pullPolicy: IfNotPresent


service:
  type: ClusterIP
  port: 80
```

Dans values.yaml, mets simplement du contenu statique ou une placeholder.

Ajoutez à la fin de hello-app/values.yaml :

```
# message HTML à servir par NGINX
htmlMessage: |
htmlMessage: |
  <html>
    <body>
      Bonjour Helm <br/>
      Namespace: __NAMESPACE__ <br/>
      Release: __RELEASE__
    </body>
  </html>
```

Ajoutez un nouveau fichier hello-app/templates/configmap.html.yaml :

On utilise Helm pour injecter les vraies infos.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ include "hello-app.fullname" . }}-html
data:
  index.html: |-
    <html>
      <body>
        Bonjour Helm <br/>
        Namespace: {{ .Release.Namespace }} <br/>
        Release: {{ .Release.Name }}
      </body>
    </html>
```

Modifiez hello-app/templates/deployment.yaml pour monter le ConfigMap (ajoutez volume + volumeMount).

Cherchez la section containers: puis ajoute/complète :

```
# ... spec: template: spec: containers: [...]
        volumeMounts:
          - name: html
            mountPath: /usr/share/nginx/html
      volumes:
        - name: html
          configMap:
            name: {{ include "hello-app.fullname" . }}-html
```

Déployer :

```
helm upgrade --install hello ./hello-app
kubectl get pods,svc
kubectl port-forward svc/hello-hello-app 8080:80
```

Lab 3 — Paramétrer par environnement (values override)

Objectif : utiliser des fichiers de valeurs dédiés (dev/prod).

Créez values-dev.yaml :

```
replicaCount: 1
resources:
  requests: { cpu: "50m", memory: "64Mi" }
  limits:   { cpu: "200m", memory: "128Mi" }
ingress:
  enabled: false
```

Vérifier la classe d'Ingress supportée

Les Ingress Controllers déclarent une IngressClass. Liste-les avec :

kubectl get ingressclass

```
Exemple de sortie :

NAME      CONTROLLER             PARAMETERS   AGE
nginx     k8s.io/ingress-nginx   <none>       5d
traefik   traefik.io/ingress     <none>       10d
```

La colonne NAME est la valeur à mettre dans className: dans ton values-prod.yaml

Si tu n'as pas d'Ingress Controller installé (liste vide), voici un pas-à-pas pour installer NGINX Ingress Controller via Helm, puis exposer la release hello.

1) Installer NGINX Ingress Controller (via Helm)

```
# Namespace dédié
kubectl create namespace ingress-nginx

# Repo officiel
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
```

Sur bare-metal/lab sans LoadBalancer, vous allez utiliser NodePort:

```
helm upgrade --install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --set controller.service.type=NodePort \
  --set controller.service.nodePorts.http=30080 \
  --set controller.service.nodePorts.https=30443
```

Vérifie que le controller tourne :

```
kubectl -n ingress-nginx get pods
kubectl -n ingress-nginx get svc
kubectl get ingressclass   # tu devrais voir 'nginx'
```

Créez values-prod.yaml :

```
replicaCount: 3
resources:
  requests: { cpu: "100m", memory: "128Mi" }
  limits:   { cpu: "500m", memory: "256Mi" }
ingress:
  enabled: true
  className: "traefik"   # ou "nginx" selon votre ingress controller
  hosts:
    - host: hello.example.com
      paths:
        - path: /
          pathType: Prefix
```

Éditez le fichier /etc/hosts sur ta machine d'accès et ajoutez :

```
192.168.1.50   hello.lab.local
```

Déployer :

```
# Dev
helm upgrade --install hello ./hello-app -f values-dev.yaml

# Prod (exemple)
helm upgrade --install hello ./hello-app -f values-prod.yaml
```

Astuce : utilisez --set key=value pour des overrides ponctuels.

Pour vérifier le NodePort en cours d'usage

```
kubectl -n helm-demo get ingress

kubectl -n ingress-nginx get svc ingress-nginx-controller

curl -I http://hello.lab.local:30080

# ou dans un navigateur: http://hello.lab.local:30080
```

Lab 4 — Dépendances de chart (ex: Redis)

Objectif : déclarer une dépendance (Bitnami Redis) et la piloter via values.

Dans hello-app/Chart.yaml, ajoutez :

```
dependencies:
  - name: redis
    version: 19.x.x
    repository: https://charts.bitnami.com/bitnami
    condition: redis.enabled
```

Dans hello-app/values.yaml, ajoutez :

```
redis:
  enabled: true
  architecture: standalone
```

```
helm lint hello-app

==> Linting hello-app

[INFO] Chart.yaml: icon is recommended

[WARNING] /home/admuser/hello-app: chart directory is missing these
dependencies: redis

1 chart(s) linted, 0 chart(s) failed
```

Dans notre Chart.yaml, nous avons déclaré une dépendance redis, mais le sous-chart n'a pas été téléchargé dans le dossier charts/.

Mettez à jour les deps puis déployez :

```
helm dependency update ./hello-app
```

```
helm upgrade --install hello ./hello-app
```

Bonnes pratiques : piloter la dépendance via condition, exposer des valeurs de connexion dans NOTES.txt.

Lab 5 — Tests Helm & Hooks

Objectif : écrire un test qui vérifie que le service répond, et un hook pre-upgrade.

Créez hello-app/templates/tests/test-connection.yaml :

```
apiVersion: v1
kind: Pod
metadata:
  name: "{{ include "hello-app.fullname" . }}-test"
  annotations:
    "helm.sh/hook": test
spec:
  restartPolicy: Never
  containers:
    - name: curl
      image: curlimages/curl:8.7.1
      args: ["-sS", "http://{{ include "hello-app.fullname" . }}:{{
.Values.service.port }}"]
```

Exécutez le test :

```
helm test hello
```

Hook pre-upgrade (optionnel) — hello-app/templates/hook-preupgrade.yaml :

```
apiVersion: batch/v1
kind: Job
metadata:
  name: "{{ include "hello-app.fullname" . }}-preupgrade"
  annotations:
    "helm.sh/hook": pre-upgrade
    "helm.sh/hook-delete-policy": before-hook-creation,hook-succeeded
spec:
  template:
    spec:
      restartPolicy: Never
      containers:
        - name: precheck
          image: busybox:1.36
          command: ["sh","-c","echo Pre-upgrade checks OK"]
```

Le hook pre-upgrade n'est déclenché qu'à l'upgrade, modifiez une valeur dans values.yaml

Par exemple changez le nombre de réplicas :

```
replicaCount: 2
```

```
helm upgrade hello ./hello-app -n helm-demo
```

Pendant l'upgrade, Helm crée un Job avec le nom basé sur ton hook (depuis un autre terminal):

```
kubectl -n helm-demo get jobs
NAME                          STATUS    COMPLETIONS   DURATION   AGE
hello-hello-app-preupgrade    Running   0/1           0s         0s
kubectl -n helm-demo get pods
```

Lab 6 — Upgrade, diff, history & rollback

Objectif : simuler une modification, visualiser le diff, revenir en arrière.

```
# (Optionnel) installer le plugin helm-diff
helm plugin install https://github.com/databus23/helm-diff


# Voir le diff
helm diff upgrade hello ./hello-app || true


# Appliquer la mise à jour
helm upgrade hello ./hello-app


# Historique & rollback
helm history hello
helm rollback hello 1     # revenir à la révision 1
```

Lab 7 — Templating avancé (helpers, conditions, required)

Objectif : factoriser des noms et imposer des valeurs requises.

Dans hello-app/templates/_helpers.tpl :

```
{{- define "hello-app.labels" -}}
app.kubernetes.io/name: {{ include "hello-app.name" . }}
app.kubernetes.io/instance: {{ .Release.Name }}
app.kubernetes.io/part-of: hello-suite
{{- end -}}
```

Dans vos manifests :

```
metadata:

  labels:

{{ include "hello-app.labels" . | indent 4 }}
```

Imposer des valeurs requises dans un template :

```
{{- $host := required "ingress.hosts[0].host est obligatoire en prod" (first
.Values.ingress.hosts).host -}}
```

Afficher des blocs conditionnels :

```
{{- if .Values.metrics.enabled }}

# manifest des métriques…

{{- end }}
```

Bonnes pratiques (résumé)

- Versionnez vos charts (semver) et gardez appVersion à jour.
- Évitez la logique complexe dans les templates ; préférez de petites fonctions dans helpers.tpl.
- Paramétrez par fichiers values-*.yaml par environnement.
- Testez (helm test) et documentez (NOTES.txt).
- Délimitez les permissions RBAC au strict nécessaire.
- Nettoyez les hooks (hook-delete-policy) pour éviter les ressources orphelines.

Dépannage rapide

- Pods en ImagePullBackOff : vérifiez image.repository/tag et Secret d'auth (imagePullSecrets).
- CrashLoopBackOff : examinez kubectl logs, readiness/liveness probes.
- Service inaccessible : port, type (ClusterIP/NodePort/LoadBalancer), Ingress class/règles.
- RBAC : messages Forbidden → Role/RoleBinding/ServiceAccount à corriger.
- Templates : helm template ./chart | kubectl apply -f - pour tester le rendu local.

Nettoyage

```
helm uninstall web

helm uninstall hello

kubectl delete ns demo
```