Correction TD3 : Helm

Gestionnaire de package pour Kubernetes

Maxime Lambert
IUT Grand Ouest Normandie – BUT Informatique S5

Année 2024/2025

Table des matières

1 Introduction

Ce document présente la correction complète du TD3 sur Helm, le gestionnaire de packages pour Kubernetes. Les exercices couvrent la création de Charts, la gestion des dépendances, la variabilisation et les bonnes pratiques.

1.1 Prérequis

— Cluster Kubernetes fonctionnel (Minikube, Kind, ou autre)
— Helm 3 installé
— kubectl configuré
— Connaissances de base en Kubernetes

2 Exercice 1 : À la découverte de Helm

2.1 Énoncé

— Installer Helm
— Créer un répertoire `vs-code-chart`
— Reprendre les travaux du TP2 pour déployer l'application via Helm
— Packager et installer le Chart

2.2 Solution

2.2.1 Installation de Helm

```
# Sur Linux
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Verification
helm version
```

2.2.2 Création de la structure du Chart

```
# Creer le repertoire
mkdir vs-code-chart
cd vs-code-chart

# Creer la structure
mkdir -p templates charts

# Creer le fichier Chart.yaml
cat > Chart.yaml <<EOF
apiVersion: v2
name: vs-code-chart
description: A Helm chart for my application
version: 0.1
EOF
```

2.2.3 Création des manifestes dans templates/

templates/deployment.yaml :

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: code-server
  labels:
    app: code-server
spec:
  replicas: 1
  selector:
    matchLabels:
      app: code-server
  template:
    metadata:
      labels:
        app: code-server
    spec:
      containers:
      - name: code-server
        image: codercom/code-server:latest
        ports:
        - containerPort: 8080
        env:
        - name: PASSWORD
          value: "monmotdepasse"
        volumeMounts:
        - name: code-server-storage
          mountPath: /home/coder
      volumes:
      - name: code-server-storage
        persistentVolumeClaim:
          claimName: code-server-pvc
```

templates/service.yaml :

```
apiVersion: v1
kind: Service
metadata:
  name: code-server
  labels:
    app: code-server
spec:
  type: NodePort
  ports:
  - port: 8080
    targetPort: 8080
    nodePort: 30080
  selector:
    app: code-server
```

templates/pvc.yaml :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: code-server-pvc
  labels:
    app: code-server
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

2.2.4 Packaging du Chart

```
# Retour au repertoire parent
cd ..

# Packager le Chart
helm package vs-code-chart

# Resultat : vs-code-chart-0.1.tgz
```

2.2.5 Installation du Chart

```
# Creer le namespace
kubectl create namespace td3

# Installer le Chart
helm upgrade --install vs-code-release \
  vs-code-chart-0.1.tgz \
  --namespace td3

# Verifier l'installation
helm list -n td3
kubectl get all -n td3
```

3 Exercice 2 : Améliorer la modularité

3.1 Énoncé

Créer un Chart dépendant pour gérer le stockage (PVC) séparément de l'application principale.

3.2 Solution

3.2.1 Création du Chart de stockage

```
# Creer le Chart dependant
mkdir -p storage-chart/templates

# Chart.yaml du Chart de stockage
cat > storage-chart/Chart.yaml <<EOF
apiVersion: v2
name: storage-chart
description: Storage management for applications
version: 0.1
EOF
```

storage-chart/templates/pvc.yaml :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: code-server-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

3.2.2 Packaging du Chart de stockage

```
# Packager le Chart de stockage
helm package storage-chart

# Resultat : storage-chart-0.1.tgz
```

3.2.3 Ajout de la dépendance

vs-code-chart/Chart.yaml (version 0.2) :

```
apiVersion: v2
name: vs-code-chart
description: A Helm chart for my application
version: 0.2
dependencies:
  - name: storage-chart
    version: 0.1
    repository: "file://../storage-chart-0.1.tgz"
```

3.2.4 Suppression du PVC du Chart principal

```
# Supprimer templates/pvc.yaml du Chart principal
rm vs-code-chart/templates/pvc.yaml
```

3.2.5 Déploiement de la nouvelle version

```
# Construire les dependances
cd vs-code-chart
helm dependency build
cd ..

# Packager la nouvelle version
helm package vs-code-chart

# Deployer
helm upgrade vs-code-release \
  vs-code-chart-0.2.tgz \
  --namespace td3

# Verifier
helm status vs-code-release -n td3
kubectl get pvc -n td3
```

Résultat attendu : Le PVC existe toujours car Helm ne supprime pas les ressources qui ne sont plus dans le Chart lors d'un upgrade. Le PVC provient maintenant du Chart dépendant.

4 Exercice 3 : Variabilisation

4.1 Énoncé

Identifier les valeurs à variabiliser et les paramétrer via `values.yaml`.

4.2 Solution

4.2.1 Création du fichier values.yaml

vs-code-chart/values.yaml :

```
# Configuration de l'application
app:
  name: code-server
  image:
    repository: codercom/code-server
    tag: latest
    pullPolicy: IfNotPresent

  replicas: 1

  service:
    type: NodePort
    port: 8080
    nodePort: 30080

  password: "monmotdepasse"

  storage:
    size: 1Gi
    storageClass: ""

# Configuration du stockage (pour le Chart dependant)
storage-chart:
```

```
  pvc:
    name: code-server-pvc
    size: 1Gi
    accessModes:
      - ReadWriteOnce
```

4.2.2 Modification des templates

templates/deployment.yaml (variabilisé) :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Values.app.name }}
  labels:
    app: {{ .Values.app.name }}
spec:
  replicas: {{ .Values.app.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.app.name }}
  template:
    metadata:
      labels:
        app: {{ .Values.app.name }}
    spec:
      containers:
      - name: {{ .Values.app.name }}
        image: "{{ .Values.app.image.repository }}:{{ .Values.app.image.tag }}"
        imagePullPolicy: {{ .Values.app.image.pullPolicy }}
        ports:
        - containerPort: {{ .Values.app.service.port }}
        env:
        - name: PASSWORD
          value: {{ .Values.app.password | quote }}
        volumeMounts:
        - name: code-server-storage
          mountPath: /home/coder
      volumes:
      - name: code-server-storage
        persistentVolumeClaim:
          claimName: {{ .Values.app.name }}-pvc
```

templates/service.yaml (variabilisé) :

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.app.name }}
  labels:
    app: {{ .Values.app.name }}
spec:
  type: {{ .Values.app.service.type }}
  ports:
  - port: {{ .Values.app.service.port }}
    targetPort: {{ .Values.app.service.port }}
    {{- if eq .Values.app.service.type "NodePort" }}
    nodePort: {{ .Values.app.service.nodePort }}
    {{- end }}
  selector:
    app: {{ .Values.app.name }}
```

4.2.3 Déploiement avec surcharge de valeurs

```
# Deploiement avec valeurs par defaut
helm upgrade vs-code-release vs-code-chart/ --namespace td3

# Deploiement avec surcharge de valeurs
helm upgrade vs-code-release vs-code-chart/ \
  --namespace td3 \
  --set app.replicas=3 \
  --set app.password="nouveaumotdepasse"

# Deploiement avec fichier de valeurs custom
cat > custom-values.yaml <<EOF
app:
  replicas: 2
  password: "production-password"
  image:
    tag: "4.9.1"
EOF

helm upgrade vs-code-release vs-code-chart/ \
  --namespace td3 \
  -f custom-values.yaml
```

5 Exercice 4 : Éviter la redondance

5.1 Énoncé

Créer un Chart `common-chart` avec des labels communs définis dans `_helpers.tpl`.

5.2 Solution

5.2.1 Création du common-chart

```
# Creer la structure
mkdir -p common-chart/templates

cat > common-chart/Chart.yaml <<EOF
apiVersion: v2
name: common-chart
description: Common labels and helpers
version: 0.1
EOF
```

common-chart/_helpers.tpl :

```
{{- define "common-chart.labels" -}}
orga: "IUT-C3"
res: "R5-09"
app: {{ .Values.app.name | default "myapp" }}
version: {{ .Chart.Version }}
managed-by: {{ .Release.Service }}
{{- end }}
```

5.2.2 Packaging du common-chart

```
helm package common-chart
# Resultat : common-chart-0.1.tgz
```

5.2.3 Ajout comme dépendance

vs-code-chart/Chart.yaml (version 0.3) :

```
apiVersion: v2
name: vs-code-chart
description: A Helm chart for my application
version: 0.3
dependencies:
  - name: storage-chart
    version: 0.1
    repository: "file://../storage-chart-0.1.tgz"
  - name: common-chart
    version: 0.1
    repository: "file://../common-chart-0.1.tgz"
```

5.2.4 Utilisation dans les templates

templates/deployment.yaml (avec labels communs) :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Values.app.name }}
  labels:
    {{- template "common-chart.labels" . | nindent 4 }}
spec:
  replicas: {{ .Values.app.replicas }}
  selector:
    matchLabels:
      app: {{ .Values.app.name }}
  template:
    metadata:
      labels:
        {{- template "common-chart.labels" . | nindent 8 }}
    spec:
      containers:
      - name: {{ .Values.app.name }}
        image: "{{ .Values.app.image.repository }}:{{ .Values.app.image.tag }}"
        # ... reste du template
```

templates/service.yaml (avec labels communs) :

```
apiVersion: v1
kind: Service
metadata:
  name: {{ .Values.app.name }}
  labels:
    {{- template "common-chart.labels" . | nindent 4 }}
spec:
  # ... reste du template
```

5.2.5 Déploiement final

```
# Construire les dependances
cd vs-code-chart
helm dependency build
cd ..

# Packager
helm package vs-code-chart
```

```
# Deployer
helm upgrade vs-code-release vs-code-chart-0.3.tgz \
  --namespace td3

# Verifier les labels
kubectl get deployment code-server -n td3 -o yaml | grep -A 10 labels
```

6 Commandes Helm utiles

6.1 Gestion des releases

```
# Lister les releases
helm list -n td3

# Voir le statut d'une release
helm status vs-code-release -n td3

# Voir l'historique
helm history vs-code-release -n td3

# Rollback vers une revision precedente
helm rollback vs-code-release 1 -n td3

# Desinstaller une release
helm uninstall vs-code-release -n td3
```

6.2 Debugging

```
# Voir les manifestes qui seront appliques (dry-run)
helm install vs-code-release vs-code-chart/ \
  --namespace td3 \
  --dry-run --debug

# Rendre les templates sans installer
helm template vs-code-release vs-code-chart/ \
  --namespace td3

# Valider le Chart
helm lint vs-code-chart/
```

6.3 Gestion des dépendances

```
# Construire/telecharger les dependances
helm dependency build vs-code-chart/

# Lister les dependances
helm dependency list vs-code-chart/

# Mettre a jour les dependances
helm dependency update vs-code-chart/
```

7 Bonnes pratiques

7.1 Versioning

— Toujours incrémenter la version dans `Chart.yaml` lors de modifications

— Utiliser le versioning sémantique (semver) : MAJOR.MINOR.PATCH
— Documenter les changements dans un fichier CHANGELOG

7.2 Structure des Charts

— Un Chart = Une application logique
— Séparer les préoccupations (stockage, monitoring, etc.) en Charts dépendants
— Utiliser des helpers (_helpers.tpl) pour éviter la duplication

7.3 Variabilisation

— Tout ce qui peut varier entre environnements doit être dans `values.yaml`
— Fournir des valeurs par défaut sensées
— Documenter chaque valeur avec des commentaires

7.4 Sécurité

— Ne jamais mettre de secrets en clair dans `values.yaml`
— Utiliser des Kubernetes Secrets ou des outils comme Sealed Secrets
— Valider les entrées utilisateur dans les templates

8 Architecture finale

```
projet/
|-- vs-code-chart/
|   |-- Chart.yaml (v0.3)
|   |-- values.yaml
|   |-- charts/
|   |   |-- storage-chart-0.1.tgz
|   |   |-- common-chart-0.1.tgz
|   |-- templates/
|       |-- deployment.yaml
|       |-- service.yaml
|
|-- storage-chart/
|   |-- Chart.yaml (v0.1)
|   |-- templates/
|       |-- pvc.yaml
|
|-- common-chart/
    |-- Chart.yaml (v0.1)
    |-- _helpers.tpl
```

9 Conclusion

Ce TD vous a permis de :
— Comprendre l'architecture d'un Chart Helm
— Créer des Charts modulaires avec des dépendances
— Variabiliser les configurations pour différents environnements
— Utiliser les helpers pour éviter la duplication de code
— Appliquer les bonnes pratiques Helm

Helm est un outil puissant pour gérer des déploiements Kubernetes complexes de manière reproductible et maintenable. La maîtrise de Helm est essentielle pour tout DevOps travaillant avec Kubernetes.