Helm

- Helm Cheat Sheet

? Helm Cheat Sheet

Cette documentation regroupe les commandes Helm utilisées pour développer, valider et déboguer les Charts Helm de la plateforme.

Important : En production, les déploiements sont réalisés exclusivement par **ArgoCD**. Les commandes présentées ici sont destinées au développement local, au test et au diagnostic.

? Objectif

Cette documentation permet de :

- Créer un nouveau Chart Helm
- Valider un Chart
- Générer les manifestes Kubernetes
- Inspecter une Release existante
- Tester un Chart localement
- Gérer les dépôts Helm

? Créer un Chart

Créer la structure de base

```
helm create mon-app
```

Structure générée :

```
mon-app/

├── Chart.yaml

├── values.yaml

├── charts/
```

```
└── templates/
```

? Vérifier un Chart

Validation syntaxique

```
helm lint ./infra/kubernetes/mon-app
```

Cette commande vérifie la structure du Chart ainsi que les bonnes pratiques Helm.

? Générer les manifestes Kubernetes

Cette commande est la plus utilisée lors du développement.

Afficher le rendu complet

```
helm template ma-release ./infra/kubernetes/mon-app --debug
```

Tester un namespace spécifique

```
helm template ma-release ./infra/kubernetes/mon-app \
-n media
```

Cette étape permet de vérifier les boucles, les conditions et les variables avant tout déploiement.

? Inspecter une Release

Afficher toutes les Releases

```
helm list -A
```

Afficher l'historique

```
helm history ma-release \
-n mon-namespace
```

?? Consulter les valeurs appliquées

Valeurs personnalisées

```
helm get values ma-release \
-n mon-namespace
```

Toutes les valeurs

```
helm get values ma-release \
-n mon-namespace \
--all
```

? Afficher les manifestes générés

```
helm get manifest ma-release \
-n mon-namespace
```

Permet de visualiser exactement les objets Kubernetes installés.

? Tester un Chart localement

Cette procédure est réservée au développement.

⚠ Si l'application est déjà gérée par ArgoCD, celui-ci risque d'écraser les modifications.

Installation

```
helm install test-release \
./mon-chart \
-n test-ns \
--create-namespace
```

Installation ou mise à jour

```
helm upgrade \
--install \
test-release \
./mon-chart \
-n test-ns \
-f mes-valeurs.yaml
```

Désinstallation

```
helm uninstall test-release \
-n test-ns
```

? Dépôts Helm

Ajouter un dépôt

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

Mettre à jour les index

```
helm repo update
```

Rechercher un Chart

```
helm search repo redis
```

? Dépannage

Problème	Commande
Erreur de template	`helm template --debug`
Erreur de syntaxe	`helm lint`
Voir les valeurs réellement utilisées	`helm get values`
Voir les objets Kubernetes	`helm get manifest`
Historique des mises à jour	`helm history`

? Bonnes pratiques

- Versionner tous les Charts dans Git.
- Ne jamais modifier une Release directement sur le cluster.
- Utiliser `helm template` avant chaque commit.
- Toujours exécuter `helm lint`.
- Utiliser un fichier `values.yaml` par environnement lorsque cela est nécessaire.
- Laisser ArgoCD gérer les déploiements en production.

? Philosophie de la plateforme

Sur cette plateforme, Helm est uniquement un moteur de génération de manifestes Kubernetes.
Les responsabilités sont réparties de la manière suivante :

Composant	Responsabilité
Git	Source de vérité
Helm	Génération des manifestes Kubernetes
ArgoCD	Déploiement GitOps
Kubernetes	Exécution des workloads

Les commandes `helm install` et `helm upgrade` ne doivent être utilisées que pour des tests locaux.
En production, toutes les applications sont déployées automatiquement par ArgoCD.

? Voir également

- ArgoCD
- Sealed Secrets
- Kubernetes GitOps
- Développement des applications Helm internes

```
```