ILKI
YOUR CLOUD DESIGNER

Formation Kubernetes

-

Pratique

kubernetes

Atelier 7

-

Deployments & Rolling updates

ILKI
YOUR CLOUD DESIGNER

Table des matières

1. Création et manipulation des Deployments

Les **Deployments** sont des ressources de type déclaratives qui permettent de mieux administrer les applications déployées sur une infrastructure **Kubernetes**.

Dans **Kubernetes**, les **Deployments** sont considérés comme des ressources de haut niveau (par opposition aux **ReplicaSet** qui eux, sont de bas niveau). Ils permettent donc d'utiliser des fonctionnalités de gestion des applications plus avancées que les **ReplicaSet**s.

Cet atelier a pour but de vous familiariser avec les **Deployments** et vous permettre de réaliser le « rolling-update » d'une application.

1.1. Déploiement de l'application « webapp » initial.

1.1.1. Création d'un Deployment

Dans cette partie, vous allez déployer une application nommée **webapp**, sur le même principe que lors de l'atelier précédent (atelier sur les **ReplicaSets**), mais cette fois, avec un **Deployment**. Cette application disposera de 3 répliquas basés sur le POD **fortune**.

1) Créez le fichier « deployment.yaml » avec le contenu suivant :

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 3
  template:
    metadata:
      name: webapp
      labels:
        app: app
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
        - containerPort: 80
        - containerPort: 22
      - name: generator
        image: ilkiformation/fortunegen:v1
```

2) Créez le **Deployment**.

3) Vérifiez que votre application est bien déployée grâce à la commande :
   - « kubectl get -f deployment.yaml ».

4) Listez les **ReplicaSets**. Que remarquez-vous ?

1.1.2. Configuration d'un service

Maintenant que vous avez créé un **Deployment** avec 3 répliquas, créez un service interne pour accéder à vos 3 répliquas.

1) Créez le fichier « service.yaml » suivant :

```yaml
kind: Service
apiVersion: v1
metadata:
  name: webapp
spec:
  selector:
    app: app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80

```

2) Lancez le déploiement de votre service.

3) Vérifiez que votre service est bien créé et que ses **EndPoints** sont bien renseignés.

4) Depuis un des nœuds de votre cluster, faites un test de connexion à votre application web par l'intermédiaire de votre service (commande : « curl IP_SERVICE »)

À ce stade, vous devez disposer d'une application web **webapp** qui affiche des pages HTML « fortune ». Cette application est accessible via un service interne **Kubernetes**.

2. Rolling update

Dans cette partie vous procéderez à une mise complète de votre application **webapp** en mode « rolling update ».

L'objectif de cette mise à jour sera de migrer l'application d'une version qui affiche des pages web « fortune » à une version qui affiche des pages web « Chuck Norris Facts ».

Pour rappel, le **Deployment** est une ressource haut niveau de **Kubernetes**. En temps réel, **Kubernetes** analyse la définition du déploiement et s'assure que cette définition est bien respectée.

En cas de modification du déploiement, **Kubernetes** le détecte et prend alors les mesures nécessaires.

Un **ReplicaSet** ne fonctionne pas de la même manière, car une modification de sa définition n'entraine pas d'action particulière de **Kubernetes** et de ce fait, si aucun pod n'est supprimé la mise à jour ne se fait pas.

1) Modifiez votre fichier « deployment.yaml » comme suit :

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 3
  template:
    metadata:
      name: webapp
      labels:
        app: app
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
        - containerPort: 80
        - containerPort: 22
      - name: generator
        image: ilkiformation/chuckgen:v1
```

2) Appliquez cette modification à votre déploiement :

```
kubectl apply -f deployment.yaml
```

3) Listez vos **ReplicaSet**s. Que remarquez-vous ?

4) Listez vos pods. Que remarquez-vous ?

5) Connectez-vous à un de vos nœuds **Kubernetes** et interrogez votre application « webapp » par l'intermédiaire de votre service interne. Que remarquez-vous ?

Comme vous avez pu le constater, la modification de l'image de référence du container « generator » de votre déploiement a entrainé une mise à jour progressive de votre application.

On appelle ce type de mise à jour, une « rolling update ». Tous les pods de votre déploiement ont été mis à jour progressivement.

Pour faire cette « rolling update », **Kubernetes** a déployé un second **ReplicaSet**, qui a progressivement « scale-up » le nombre de pods basés sur la nouvelle version de **webapp** à 3 tout en diminuant progressivement le nombre de répliquas gérés par le **ReplicaSet** initial à 0.

3. Rolling back

Dans cette partie vous allez faire un « rolling-back » sur votre déploiement. Ce mécanisme permet de revenir rapidement à une version de déploiement antérieure de votre application.

3.1.Revenir à l'état précédent

Vous allez mettre à jour votre application **webapp** (grâce au **Deployment**) en changeant l'image du générateur de page HTML par « ilkiformation/funjokegen :v1 ».

Une fois la mise à jour effectuée, vous pourrez effectuer une restauration de votre application **webapp** à sa version précédente.

1) Modifiez votre fichier « deployment.yaml » comme suit :

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: webapp
spec:
  replicas: 3
  template:
    metadata:
      name: webapp
      labels:
        app: app
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
        - containerPort: 80
        - containerPort: 22
      - name: generator
        image: ilkiformation/funjokegen:v1
```

2) Lancez votre mise à jour grâce à la commande : ***kubectl apply -f deployment.yaml***

3) Vous pouvez visualiser en temps réel les modifications faites par le déploiement lors de la mise à jour grâce à la commande suivante :

```
kubectl rollout status deployment webapp
```

4) Une fois la mise à jour terminée, vérifiez que l'application **webapp** a bien été mise à jour.

À ce stade, votre application **webapp** affiche le texte de la version « funjokegen ». Vous allez maintenant passer au « rolling-back ».

5) Pour faire un « rolling-back » à la version précédente de votre application, utilisez la commande suivante :

```
kubectl rollout undo deployment webapp
```

6) Vérifiez que votre application est bien revenue à son état précédent « chuckgen ».
   a. Vous pouvez vérifier l'avancement du « rolling back » de la même manière que vous le feriez avec une « rolling-update ».

3.2.Revenir à un état antérieur.

À ce stade, vous avez fait une « rolling back » qui a restauré votre application à son état précédent.

Dans cette partie, vous allez restaurer votre application à une version spécifique.

1) Listez l'historique des versions de votre **Deployment** grâce à la commande suivante :

```
kubectl rollout history deployment webapp
```

2) Modifiez l'image du générateur de votre application à « ilkiformation/funjokegen :v1 » grâce à la commande suivante :

```
kubectl set image deployment webapp generator=ilkiformation/funjokegen:v1 --record=True
```

3) Listez de nouveau l'historique de votre **Deployment**. Que remarquez-vous ?

Lors d'une modification de la déclaration d'un **Deployment**, il est possible de renseigner le flag « --record=True ». Cela permet d'enregistrer la ligne de commande responsable de la modification de l'état du **Deployment**. Il est également possible d'utiliser ce flag avec les commandes « kubectl create » et « kubectl apply ».

Si ce flag n'est pas renseigné lors d'une modification de déploiement, alors le champ « CHANGE-CAUSE » du « rollout history » reste à « <none> ».

À chaque modification du déploiement, une nouvelle « révision » est enregistrée. Vous pouvez restaurer votre **Deployment** à une version antérieure spécifique grâce à la commande suivante :

```
kubectl rollout undo deployment [DEPLOYMENT_NAME] --to-revision=[NUM_REVISION]
```

4) Procédez à une restauration de votre **Deployment** à sa première version grâce à la commande suivante :

```
kubectl rollout undo deployment webapp --to-revision=1
```

5) Vérifiez que votre application **webapp** est bien revenue à sa version initiale.
    a. Elle affichait du texte généré par « fortunegen »)

4. Notions avancées

Vous avez pu constater, lors d'une mise à jour ou d'un rollback, le workfow suivant :

- Le déploiement crée un nouveau **ReplicaSet** gérant l'apparition de pods dans la nouvelle version d'application désirée.
- Un nouveau pod est créé, et lorsqu'il est prêt, un ancien pod disparait, et ce jusqu'à ce qu'il n'existe plus d'anciens pods.
- Une fois tous les anciens pods supprimés, l'ancien **ReplicaSet** qui les contrôlait est également supprimé.

Ce mécanisme permet d'avoir une continuité de service lors des « Rolling Update/Rolling Back ».

Kubernetes vous permet de personnaliser ce workflow au moyen d'une stratégie de « rolling update » déclarée dans les « Spec » du **Deployment**.

Cette stratégie couvre 2 propriétés qui sont :

- **maxSurge** : Détermine, lors d'un « Rolling Update/Rolling Back », combien de pods peuvent exister en plus (par rapport au nombre de pods défini dans le déploiement). Cette limite peut être spécifiée en pourcentage ou en nombre entier. La valeur par défaut est de 25%.
    - ***Exemple :*** Si on place ce paramètre à 30%, alors dès le début du « Rolling Update/Rolling Back », le nouveau **ReplicaSet** peut immédiatement scale-up tant que le nombre de pod n'excède pas 130% du nombre de pods désirés. Dès que des anciens pods sont supprimés, le nouveau **ReplicaSet** peut à nouveau scale-up, mais en faisant toujours attention à ne pas dépasser les 130% du nombre de pods désiré.
    - Ce paramètre permet de mieux gérer la vitesse de « Rolling Update/Rolling Back », mais il faut faire attention à la consommation de ressources engendrée par un trop grand nombre de pods !

- **maxUnavailable** : Détermine, lors du « Rolling Update/Rolling Back », combien de pods peuvent être non disponibles (par rapport au nombre de pods défini dans le **Deployment**). Cette limite peut être spécifiée en pourcentage ou en nombre entier. La valeur par défaut est 25%.
    - ***Exemple :*** Si on place ce paramètre à 30% alors dès le début du « Rolling Update/Rolling Back », le nombre de pods gérés par l'ancien **ReplicaSet** va passer à 70% de la valeur désirée. Au fur et à mesure que le nouveau **ReplicaSet** va scale-up et que ses pods vont être prêts, l'ancien **ReplicaSet** va scale-down, mais en s'assurant toujours qu'il y a au moins 70% des pods qui sont prêts tout au long du processus.

- Cela permet de d'assurer une continuité de service minimale lors des « Rolling Update/Rolling Back ».

Dans l'exercice ci-dessous, vous allez modifier le nombre de répliquas de votre **Deployment** à 10, puis vous allez spécifier un **maxSurge** à 30% et un **maxUnavailable** à 30%.

Vous pourrez ensuite visualiser l'influence qu'ont ces 2 paramètres lors des « Rolling Update/Rolling Back ».

1) Commencez par éditer votre fichier « deployment.yaml » comme suit :

```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: webapp
spec:
  strategy:
    rollingUpdate:
      maxSurge: 30%
      maxUnavailable: 30%
    type: RollingUpdate
  replicas: 10
  template:
    metadata:
      name: webapp
      labels:
        app: app
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
        - containerPort: 80
        - containerPort: 22
      - name: generator
        image: ilkiformation/fortunegen:v1
```

2) Lancez la mise à jour de votre **Deployment** via la commande :

```
kubectl apply -f deployment.yaml
```

3) Vérifiez que votre **Deployment** dispose bien de 10 répliquas.

4) Modifiez l'image du container « generator » par « ilkiformation/chuckgen:v1 »

5) Lancez une « rolling update » de votre **Deployment**.

6) Visualisez l'avancement de votre « rolling update » via la commande :

```
kubectl rollout status deployment webapp
```

Comme vous avez pu le constater, dès le début de la « rolling update » le nombre total de pods de votre **Deployment** est passé à 13, puis au fur et à mesure de l'avancement les anciens pods ont été supprimés et les nouveaux ont été créés jusqu'à ce qu'il ne reste que 10 nouveaux pods.

Tout au long de la « rolling update », au moins 7 pods étaient en fonctionnement ce qui a permis une continuité de service.

7) Supprimez votre **Deployment** et votre service **webapp.**