ILKI
YOUR CLOUD DESIGNER

Formation Kubernetes

-

Pratique

kubernetes

Atelier 5

-

ReplicaSets

ILKI
YOUR CLOUD DESIGNER

Table des matières

1. Les ReplicaSets

Un **ReplicaSet** est un composant **Kubernetes** permettant de répliquer des pods et de les réordonnancer en cas de défaillance ou de panne d'une node.

Généralement, vous ne créez pas les **ReplicaSets** directement, mais plutôt en association avec une ressource de type **Deployment** (qui sera traitée dans les prochains ateliers).

Un **ReplicaSet** utilise les label selectors pour surveiller et gérer les pods sous sa responsabilité. En effet, il s'assure que le nombre actuel de pods, avec un certain label, correspond au nombre désiré :

- Si trop peu de pods sont lancés, il créé un nouveau répliqua basé sur la template spécifiée,
- Si trop de pods sont lancés, il supprime les répliquas supplémentaires.

1.1. Création et manipulations des ReplicaSets

1.1.1. Création d'un ReplicaSet mono-pod

Dans cette partie vous allez créer un **ReplicaSet** qui aura comme Template le pod « fortune » constitué des deux containers suivants :

- 1 container « web », qui affiche une page HTML.
- 1 container « generator » qui génère une page HTML toutes les 5 secondes et qui l'upload sur le container « web ».

Vous pourrez accéder à votre page web via l'IP de votre pod.

1) Créez le fichier « ReplicaSet1.yaml » suivant :

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: fortune
spec:
  selector:
    matchLabels:
      app: fortune
  template:
    metadata:
      labels:
        app: fortune
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
          - containerPort: 80
          - containerPort: 22
      - name: generator
        image: ilkiformation/fortunegen:v1
```

2) Placez-vous dans le dossier qui contient votre fichier « ReplicaSet1.yaml » puis déployer votre **ReplicaSet** via la commande :

```
kubectl create -f ReplicaSet1.yaml
```

3) Vérifiez que votre **ReplicaSet** est bien créé et visualisez sur quel nœud **Kubernetes** il s'exécute :

```
kubectl get pods -o wide
```

Note : Votre pod doit avoir un nom en « fortune-xxxxx »

4) Via GCP, connectez-vous sur le nœud qui exécute votre pod et affichez le contenu de votre site web via la commande « curl IP_POD »

À ce stade, vous pouvez constater que votre **ReplicaSet** a bien déployé un pod basé sur le template défini lors de sa création.

5) Détruisez le pod « fortune-xxxxx » généré via votre **ReplicaSet**.

6) Listez de nouveau vos pods via la commande « kubectl get pods ». Que constatez-vous ?

Comme vous pouvez le constater, la suppression du premier pod entraine la création d'un second pod.

1.1.2. ReplicaSet & Scaling

Dans cette partie vous allez « Scale-up » et « Scale-down » votre application « fortune ».

Scale-up

Il est possible d'augmenter le nombre de répliquas gérés par votre **ReplicaSet** de deux façons :

- Via une ligne de commande.
- Via un fichier déclaration.

1) Commencez par augmenter le nombre de répliquas de votre application « fortune » à 3 en ligne de commande :

```
kubectl scale --replicas=3 replicaset fortune
```

2) Visualisez l'augmentation du nombre de pods à 3 via la commande :

```
kubectl get pod
```

3) Modifiez maintenant votre fichier « ReplicaSet1.yaml » pour y déclarer un nombre de répliquas à 5, comme suit :

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: fortune
spec:
  replicas: 5
  selector:
    matchLabels:
      app: fortune
  template:
    metadata:
     labels:
       app: fortune
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
          - containerPort: 80
          - containerPort: 22
      - name: generator
        image: ilkiformation/fortunegen:v1
```

4) Mettez à jour votre **ReplicaSet** grâce à la commande suivante :

```
kubectl apply -f ReplicaSet1.yaml
```

5) Visualisez l'augmentation du nombre de pods à 5 via la commande :

```
kubectl get pod
```

À ce stade, votre application fortune doit posséder 5 répliquas.

Scale-down

Il est possible de diminuer le nombre de répliquas gérés par votre **ReplicaSet** de deux façons :

- Via une ligne de commande.
- Via un fichier déclaration.

1) Commencez par diminuer le nombre de répliquas de votre application « fortune » à 3 en ligne de commande :

```
kubectl scale --replicas=3 replicaset fortune
```

2) Visualisez la diminution du nombre de pods à 3 via la commande :

```
kubectl get pod
```

Note : Vous pouvez remarquer que l'algorithme de suppression des pods est de type : « Last-In/First-Out » (LIFO)

3) Modifiez maintenant votre fichier « ReplicaSet1.yaml » pour y déclarer un nombre de répliquas à 2, comme suit :

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: fortune
spec:
  replicas: 2
  selector:
    matchLabels:
      app: fortune
  template:
    metadata:
     labels:
        app: fortune
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
          - containerPort: 80
          - containerPort: 22
      - name: generator
        image: ilkiformation/fortunegen:v1
```

4) Mettez à jour votre **ReplicaSet** grâce à la commande suivante :

```
kubectl apply -f ReplicaSet1.yaml
```

5) Visualisez la diminution du nombre de pods à 2 via la commande :

```
kubectl get pod
```

À ce stade, votre application fortune doit posséder 2 répliquas.

1.2.Labels et ReplicaSets

1.2.1. La gestion des labels

La gestion des labels est un point essentiel au bon fonctionnement des **ReplicaSets**.

En effet, un **ReplicaSet** identifie les pods qu'il manage grâce à leurs labels. Lors de la création du **ReplicatSet** de l'exercice précédent, vous avez spécifié dans la rubrique « template » que les nouveaux pods qui seront déployés par le **ReplicaSet** auront pour label « app = fortune » (1).

Vous avez également indiqué, grâce au « selector », que le **ReplicaSet** « surveille » les pods qui ont pour label « app = fortune ». (2).

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: fortune
spec:
  replicas: 2
  selector:
    matchLabels:
      app: fortune
  template:
    metadata:
      labels:
        app: fortune
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
          - containerPort: 80
          - containerPort: 22
      - name: generator
        image: ilkiformation/fortunegen:v1
```

Dans cette partie vous procéderez à des modifications sur les labels des pods afin de comprendre comment fonctionne un **ReplicaSet**.

1) Vérifiez que vous disposez toujours des 2 pods « fortune-xxxxx » résultants de l'exercice précédent et listez leurs labels :

```
kubectl get pods --show-labels
```

2) Modifiez le label « app = fortune » d'un de ces pods à « app = debug ».

```
kubectl label pod fortune-9nmx2 app=debug --overwrite
```

3) Listez de nouveau vos pods « fortune-xxxxx » et leurs labels.

   Que remarquez-vous ?

Comme vous pouvez le remarquer, suite à la modification du label « app=fortune » de votre pod en « app=debug », le **ReplicaSet** provisionne un nouveau pod, basé sur son Template et ayant pour label « app=fortune ».

Ce comportement est normal car dans votre cas, le **ReplicaSet** surveille qu'il y a bien 2 pods avec le label « app=fortune ». Si ce nombre de répliquas n'est pas respecté, le **ReplicaSet** prend les mesures nécessaires.

Dans votre cas, après modification du label « app », le nombre de répliquas avec le champ label « app=fortune » est descendu à 1, et donc le **ReplicaSet** a provisionné un nouveau pod avec le label « app=fortune » pour revenir à un état à 2 pods avec le label « app=fortune ».

4) Supprimez le pod « fortune-xxxxx » qui a le label « app=debug ».

   Que remarquez-vous ?

Vous pouvez remarquer que la suppression du pod « fortune-xxxxx » avec un label à « app=debug » n'entraine pas la création d'un nouveau pod « fortune ». En effet, pour le **ReplicaSet**, le nombre de pods avec un label « app=fortune » est toujours égale à 2 et, de fait, aucune action n'est requise.

5) Créez le fichier « pod.yaml » suivant :

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: addpod
  labels:
    app: fortune
spec:
  containers:
  - name: web
    image: ilkiformation/webapache:v1
    ports:
      - containerPort: 80
      - containerPort: 22
  - name: generator
    image: ilkiformation/fortunegen:v1
```

1) Lancez le déploiement de votre pod puis exécutez la commande suivante. Que remarquez-vous ?

```
kubectl get pods --show-labels
```

En ajoutant un nouveau pod avec un label « app=fortune », le nombre de pod avec ce label est passé à 3. Le **ReplicaSet** voit alors qu'il y a un pod en trop et le supprime en respectant l'algorithme LIFO. C'est pour cela que votre pod s'est supprimé !

1.2.2. Les label selectors plus expressifs

Dans la partie précédente, vous avez utilisé le simple *matchLabels* selector, mais il est important de savoir qu'il est possible de définir des expressions plus avancées par le biais de la propriété *matchExpressions*. Ci-dessous un exemple :

```yaml
selector:
  matchExpressions:
    - key: rel
      operator: In
      values:
        - stable
```

Chaque expression doit contenir une **clé**, un **opérateur**, et éventuellement (en fonction de l'opérateur) une liste de **valeurs**. Ci-dessous, les 4 opérateurs valides :

- **In –** La valeur contenue dans le label doit correspondre à une des valeurs spécifiées.

- **NotIn -** La valeur contenue dans le label NE doit PAS correspondre à une des valeurs spécifiées.
- **Exists –** Le pod doit contenir un label avec la clé spécifiée (la valeur n'est pas importante). Quand vous utilisez cet opérateur, vous ne devez pas spécifier de valeurs.
- **DoesNotExist** – Le pod NE doit PAS contenir un label avec la clé spécifiée (la valeur n'est pas importante).

Si vous spécifiez plusieurs expressions, toutes celles-ci doivent être validées pourque le selector sélectionne le pod.

Si vous spécifiez à la fois des *matchLabels* et des *matchExpressions*, l'ensemble doit être validé pour que le pod soit sélectionné.

1.3.Mise à jour d'un ReplicaSet

Il est possible de mettre à jour la déclaration des **ReplicaSets**.

Dans cette partie, vous procéderez à la mise à jour du Template du **ReplicaSet** « fortune ». La mise à jour consistera à changer l'image du générateur de page HTML afin que notre application affiche des « Chuck Norris Facts ».

1) Modifiez votre fichier « ReplicaSet1.yaml » comme suit :

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: fortune
spec:
  replicas: 2
  selector:
    matchLabels:
      app: fortune
  template:
    metadata:
     labels:
       app: fortune
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
          - containerPort: 80
          - containerPort: 22
      - name: generator
        image: ilkiformation/chuckgen:v1
```

2) Procédez à la mise à jour de votre **ReplicaSet** grâce à la commande suivante :

```
kubectl apply -f ReplicaSet1.yaml
```

3) Vérifiez la présence vos pods « fortune-xxxxx ». Vous devriez en voir 2.

4) L'image utilisée par le container « generator » de vos 2 pods a-t-elle changée ? Vous pouvez par exemple utiliser les commandes suivantes :

```
kubectl describe pod fortune-cvwxf | grep Image:
```

Ou

```
kubectl describe pod [nomDuPod]
```

Comme vous pouvez le constater, la mise à jour du **ReplicaSet** n'a pas entrainée une mise à jour automatique des pods déjà déployés. En effet, en modifiant le Template utilisé par le **ReplicaSet**, vous avez modifié la déclaration des futurs pods qui seront déployés, mais cela n'affecte aucunement les pods déjà déployés.

5) Supprimez un de vos pods « fortune-xxxxx ».

6) Listez les images utilisées par le nouveau pod « fortune-xxxxx » déployé par votre **ReplicaSet**. Que remarquez-vous ?

7) Via GCP, connectez-vous sur le nœud qui exécute votre pod et affichez le contenu du site web de votre nouveau pod « fortune-xxxxx » via la commande « curl IP_POD ».

2. Les DaemonSets

Les **ReplicaSets** sont utilisés pour faire fonctionner un nombre spécifique de pods, déployés n'importe où dans le cluster **Kubernetes**.

Seulement, dans certains cas, vous pouvez avoir besoin de déployer un pod sur l'ensemble des nodes du cluster. Généralement, ce besoin apparaît dans le cas de pods liés à l'infrastructure, qui doivent effectuer des opérations au niveau système.

Par exemple, si vous souhaitez déployer un collecteur de logs et un outil de supervision de ressources sur chaque node. Un autre bon exemple est celui du **kube-proxy**, qui a besoin d'être exécuté sur toutes les nodes pour que le service soit fonctionnel.

Pour rendre cela possible, il vous faut utiliser les **DaemonSets**, qui sont très similaires aux **ReplicaSets**, à la différence que les pods créés ont directement une node cible spécifiée. Ils évitent donc le passage par le **Scheduler Kubernetes.**

Dans ce cas, vous aurez donc évidemment compris que vous retrouverez autant de pods que de nodes. Si une node tombe en panne, aucun pod ne sera recréé. Par contre, si une nouvelle node est ajoutée au cluster, un pod du **DaemonSet** sera automatiquement déployé.

À noter qu'il est également possible de spécifier un sous-ensemble de nodes sur lesquelles déployer les pods, grâce à la propriété *nodeSelector*.

Vous allez illustrer cette notion par le biais d'un exemple se basant sur la même template de pod que celle utilisée dans les exercices précédents.

1) Pour commencer, supprimez les ressources créées précédemment

```
kubectl delete rs fortune
```

2) Créez un fichier « daemonset.yaml » suivant :

```yaml
apiVersion: apps/v1beta2
kind: DaemonSet
metadata:
  name: fortune
spec:
  selector:
    matchLabels:
      app: fortune
  template:
    metadata:
      labels:
        app: fortune
    spec:
      containers:
      - name: web
        image: ilkiformation/webapache:v1
        ports:
          - containerPort: 80
          - containerPort: 22
      - name: generator
        image: ilkiformation/fortunegen:v1
```

Note :

- Attention, les **DaemonSets** ne sont disponibles pour le moment que dans l'API *apps/v1beta2*
- Vous remarquerez au passage qu'il n'y a aucune notion de nombre de répliquas

3) Listez les pods créés. Que remarquez-vous ?

```
kubectl get po -o wide
```

4) Supprimez le **DaemonSet** créé.