MISE À NIVEAU D'UN CLUSTER KUBERNETES (KUBEADM)

Conseils, informations et prérequis

Dans ce chapitre nous allons **étudier la mise à niveau des nœuds maîtres et des nœuds de travail de votre cluster kubernetes**. Nous allons pour cela **mettre à jour l'outil kubeadm et kubelet**.

Pour ce Lab, nous allons passer de la version 1.16 vers la version 1.17. Mais, avant toutes choses assurez-vous du respect des prérequis/conseils suivants :

- Le swap doit être désactivé .
- Assurez-vous de lire attentivement les notes de version kubernetes dans le but de connaître les nouveaux changements que vous allez mettre en place.
- Assurez-vous de sauvegarder tous les composants importants (voir mon article sur la sauvegarde et restauration d'un cluster kubernetes)
- Tous les conteneurs sont redémarrés après la mise à niveau.

Les versions kubernetes

Comprendre les versions kubernetes

Description

Les versions de Kubernetes sont exprimées en `x.y.z` , où `x` est la version principale, `y` est la version mineure et `z` est la version du correctif. Nous avons

ainsi :

- **La version principale** : modifications majeures de l'API.
- **La version mineure** : ajout de nouvelles fonctionnalités rétrocompatibles.
- **La version du correctif** : corrections de bugs/failles des fonctionnalités ajoutées par la version mineure.

Cycle de vie

Il n'y a pas de calendrier de mise à niveau obligatoire pour les versions majeures. Cependant, les versions mineures se produisent environ tous les 3 mois et sont prises en charge pendant environ 9 mois.

Les versions de correctifs sont disponibles chaque semaine et sont destinées aux corrections de bugs critiques de la dernière version mineure, telles que la correction des vulnérabilités de sécurité, la résolution des problèmes affectant un grand nombre d'utilisateurs, les problèmes graves sans solution de contournement et les bloqueurs pour les produits basés sur Kubernetes. Aucune incompatibilité ne doit être introduite entre les versions de correctif de la même version mineure.

Informations supplémentaires

Les dépendances, telles que Docker, ne devraient pas être modifiées sauf en cas d'absolue nécessité, et également uniquement pour corriger des bugs critiques.

Vous n'êtes pas obligé d'exécuter approximativement la dernière version de correctif d'une version mineure donnée. Néanmoins, l'équipe kubernetes inclue souvent des corrections de bugs critiques dans les versions de correctifs , et encouragent ses utilisateurs à mettre à niveau dès que possible.

Les versions et les mises à jour

Avec un cycle de publication aussi agressif, si jamais vous gérez plusieurs clusters Kubernetes, il peut être très avantageux d'automatiser autant que possible le processus de mise à niveau. Heureusement que Kubernetes est conçu pour être mis à niveau de manière continue :

- Les nœuds worker peuvent posséder jusqu'à deux versions mineures inférieures au nœud master.
- Les nœuds worker ne doivent pas être dans une version plus récente que le nœud master.
- Un client peut posséder jusqu'à une seule version mineure supérieure ou inférieure à celle du nœud master.

Par exemple : un nœud maître v1.3 peut fonctionner avec les nœuds en version v1.1, v1.2 et v1.3 et peut fonctionner avec les clients v1.2, v1.3 et v1.4.

Les mises à niveau de versions mineures sont généralement sûres. En d'autres termes, la mise à niveau d'une version mineure vers la suivante devrait fonctionner sans temps d'arrêt ou états incohérents. Cependant, le projet Kubernetes recommande la **mise à niveau incrémental par version mineure**, c'est-à-dire que vous pouvez uniquement passer d'une version mineure à la prochaine version mineure, ou entre des versions de correctif du même mineur.

Autrement dit, vous ne pouvez pas ignorer les versions mineures lors de la mise à niveau. Par exemple, vous pouvez passer de 1.y à 1.y + 1, mais pas de 1.y à 1.y + 2.

Upgrade de notre master

Quand il s'agit d'une montée de version d'un cluster kubernetes, on doit **tout d'abord commencer par mettre à jour notre nœud master** , et par la suite **mettre à jour les autres nœuds de travail un par un**.

Dans mon cas, je souhaite upgrade mon master de la version 1.16 vers la version 1.17. Pour ce faire, nous entamerons une phase de recherche afin de **trouver la dernière version stable de kubeadm et kubelet** en v1.17. Comme je suis sous la distribution Ubuntu, je vais utiliser l'outil `apt` afin de rechercher mes nouveaux paquets :

```
apt-get update && \
apt-cache policy kubeadm
```

Résultat :

```
kubeadm:
Installed: 1.16.0-00
Candidate: 1.17.0-00
...
```

```
apt-cache policy kubelet
```

Résultat :

```
kubelet:
Installed: 1.16.0-00
Candidate: 1.17.0-00
...
```

Le résultat nous signale que la version 1.17 de kubeadm et kubelet sont bien disponibles. Nous allons d'abord nous attaquer directement par l'installation du nouveau paquet kubeadm :

```
apt-get upgrade -y kubeadm=1.17.0-00
```

Ensuite, nous vérifions que le téléchargement fonctionne et possède la version attendue :

```
kubeadm version
```

Résultat :

```
kubeadm version: &version.Info{Major:"1", mineur:"17", GitVersion:"v1.17.0" ...}
```

À présent, nous allons interroger l'outil kubeadm afin de vérifier les versions disponibles pour la mise à niveau et valider si notre cluster actuel peut être mis à niveau vers notre nouvelle version :

```
kubeadm upgrade plan
```

Résultat :

```
Components that must be upgraded manually after you have upgraded the control plane wi
COMPONENT   CURRENT       AVAILABLE
Kubelet     4 x v1.16.0   v1.17.0

Upgrade to the latest stable version:

COMPONENT            CURRENT   AVAILABLE
API Server           v1.16.0   v1.17.0
Controller Manager   v1.16.0   v1.17.0
Scheduler            v1.16.0   v1.17.0
Kube Proxy           v1.16.0   v1.17.0
CoreDNS              1.6.2     1.6.5
Etcd                 3.3.15    3.4.3-0

You can now apply the upgrade by executing the following command:

kubeadm upgrade apply v1.17.0
```

L'étape suivante consiste à **rendre notre nœud master unschedulable**, c'est-à-dire que le nœud ne pourra plus accepter de nouveaux pods et tous les pods existants dans le nœud seront déplacés vers un autre nœud :

```
kubectl drain master --ignore-daemonsets
```

Enfin, nous allons appliquer les nouveaux changement grâce la commande suivante :

```
kubeadm upgrade apply v1.17.0
```

Résultat :

```
[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.17.0". Enjoy!
```

Nous devons par la suite installer la version stable de kubelet en v1.17.0 et redémarrer le service de manière à prendre en compte sa nouvelle version :

```
apt-get upgrade -y kubelet=1.17.0-00 && \
systemctl restart kubelet
```

Enfin, on n'oublie pas de rendre notre nœud à nouveau schedulable :

```
kubectl uncordon master
```

En lançant la commande ci-dessous, on peut remarquer que nos nœuds workers ne possèdent pas la même version que notre nœud master:

```
kubectl get nodes
```

Résultat :

```
NAME     STATUS   ROLES    AGE   VERSION
master   Ready    master   33m   v1.17.0
node01   Ready    <none>   32m   v1.16.0
```

Comme vu auparavant sur la section des versions, on peut d'ores et déjà s'arrêter ici, car il est toléré d'avoir au minimum deux version mineurs des nœuds workers inférieurs à celle du nœud master. Mais il est de mon devoir de vous montrer comment procéder à un upgrade des nœuds de travail.

Upgrade de notre worker

Comme dit antérieurement, nous devons upgrade nos workers un par un. La démarche reste tout de même similaire à celle du master à quelques différences près.

Premièrement, nous allons **rendre notre nœud de travail unschedulable** depuis notre nœud master :

```
kubectl drain node01 --ignore-daemonsets
```

Ensuite il faut être connecté sur un worker et installer les dernières versions stables de kubeadm et kubelet depuis notre nœud de travail :

```
ssh root@node01 "apt-get update && apt-get upgrade -y kubeadm=1.17.0-00 kubelet=1.17.0
```

Depuis notre master nous allons mettre à niveau notre worker :

```
kubeadm upgrade node01 config --kubelet-version v1.17.0
```

Ensuite, il faut redémarrer le service kubelet :

```
ssh root@node01 "systemctl restart kubelet"
```

Enfin, on n'oublie pas de rendre notre nœud à nouveau schedulable :

```
kubectl uncordon node01
```

En vérifiant les nœuds disponibles de note cluster, on peut se rendre compte que notre worker et master sont au même niveau de version.

```
kubectl get nodes
```

Résultat :

```
NAME     STATUS   ROLES    AGE   VERSION
master   Ready    master   55m   v1.17.0
node01   Ready    <none>   53m   v1.17.0
```