Kubernetes Cluster Upgrade

Allgemeines

Das Upgrade eines Kubernetes-Clusters beinhaltet mehrere Schritte. Von diesen Schritten sind einige für alle Nodes gleich, wohingegen sich andere für Control und Data Plane unterscheiden. Allen gemein ist, dass die Nodes vor einem Update mit kubectl drain von den Work- loads befreit und aus dem Scheduling genommen werden. Ebenfalls gilt für alle Nodes, dass diese vor ihrem Update auf ihre ordnungsgemäße Konfiguration wie Funktion über- prüft werden, damit Komplikationen im Update-Prozess vermieden werden.

Für das Upgrade der Master Nodes in einem HA-Cluster wird zunächst das Upgrade auf dem ersten Master Node durchgeführt. Für die weiteren Master Nodes ist der Upgrade- Prozess anders, da diese dann die Upgrade-Parameter direkt aus der Configmap des Clusters über den Master1 erhalten.

Upgrade der Control Plane

Upgrade des Master1

Den Master1 „drainen“

Damit der Master1 ein Versions-Upgrade (oder auch Downgrade) erhalten kann, muss er zunächst mit `kubectl drain` von seiner Workload befreit und aus dem Scheduler genommen werden. Zwar laufen in der Control Plane keine Workloads im typischen Sinne wie auf den Workern, jedoch beispielsweise der `coredns`-Pod:

```
$ kubectl get pods -n kube-system -o wide --field-selector
```

NAME	READY	STATUS	RESTARTS	
calico-node-vdm6g	1/1	Running	1	105d
coredns-74ff55c5b-6k7rg	1/1	Running	0	24h
etcd-master1	1/1	Running	2	25h
ha-ip-apiservices-master1	2/2	Running	2	99d
kube-apiserver-master1	1/1	Running	2	25h
kube-controller-manager-master1	1/1	Running	2	25h
kube-proxy-hxw6g	1/1	Running	0	25h
kube-scheduler-master1	1/1	Running	2	25h

Status und Konfiguration überprüfen

Mit `kubectl drain` werden die Workloads entfernt und der Node aus dem Scheduling genommen. Die Option `–ignore-daemonsets` ist notwendig, da die Pods, die von ei- nem DaemonSet stammen (hier `calico-node-vdm6g` und `kube-proxy-hxw6g`) nicht mit `drain` entfernt werden können:

```
$ kubectl drain master1 --ignore-daemonsets
node/master1 already cordoned
WARNING: ignoring DaemonSet-managed Pods:
    kube-system/calico-node-vdm6g, kube-system/kube-proxy-prhqg
evicting pod kube-system/coredns-6f5c7bbdfb-jg4rk
pod/coredns-6f5c7bbdfb-jg4rk evicted
node/master1 evicted
```

Nachdem der Master1 keine Workloads (außer den Status Pods) mehr hat und aus dem Scheduling genommen wurde, sollten Sie noch einmal die Konfiguration und den Status des `etcd` in Ihrem Cluster überprüfen:

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	
master1	Ready,SchedulingDisabled	control-plane,master	106d	v1.19.2
master2	Ready	control-plane,master	106d	v1.19.2
master3	Ready	control-plane,master	106d	v1.19.2
worker1	Ready	<none>	106d	v1.19.2
worker2	Ready	<none>	106d	v1.19.2

Die Version bezieht sich hier auf die Version des Kubelets auf den Nodes. Mit der Op- tion `-o wide` können Sie sich zusätzlich zu diesen Daten auch Informationen über das Betriebssystem, die Kernel-Version und die verwendete Container Runtime auf den jeweiligen Nodes anzeigen lassen. Sollte der etcd-Cluster fehlerhaft sein und es kommt bei dem Upgrade zu einem Ausfall des Nodes, so kann das Quorum zwischen den etcd-Clustereinheiten nicht ausgeführt werden, wodurch der gesamte etcd-Cluster dysfunktional wird. Um den Status des etcd- Clusters zu erfragen, verwenden Sie die Option –`selector` (`-l`) unter Angabe des Se- lectors `component` mit dem Wert `etcd`:

```
$ kubectl get pod -n kube-system --selector component=etcd
```

NAME	READY	STATUS	RESTARTS	
etcd-master1	1/1	Running	4	29h
etcd-master2	1/1	Running	149	100d
etcd-master3	1/1	Running	188	100d

Da es nicht ohne Weiteres möglich ist, bei einem Upgrade des Clusters über mehrere Versionen hinweg zu springen, sollten Sie auf der folgenden Website nachschauen, auf welche Version ein Upgrade von Ihrer Version aus möglich ist:

https://kubernetes.io/docs/setup/release/version-skew-policy/

Die Version des verwendeten Apiservers können Sie mit folgendem curl-Aufruf erfragen:

```
$ curl https://<IP-Adress of MasterNode>:6443/version -k
{
"major": "1",
"minor": "19",
"gitVersion": "v1.19.2",
```

```
"gitCommit": "9f2892aab98fe339f3bd70e3c470144299398ace",
"gitTreeState": "clean",
"buildDate": "2020-08-13T16:04:18Z",
"goVersion": "go1.13.15",
"compiler": "gc",
"platform": "linux/amd64"
}%
```

Die Option `-k` wird hier verwendet, um das Überprüfen des TLS-Zertifikats zu unter- drücken. Nachdem alles überprüft ist, kann das eigentliche Upgrade vorgenommen werden.

kubeadm upgraden

Das Upgrade des Clusters wird mit `kubeadm` durchgeführt. Auch hier empfiehlt es sich, zunächst die Version zu überprüfen. Stellen Sie sicher, dass die aktuelle Version mit der, auf die das Upgrade durchgeführt werden soll, kompatible ist:

```
$ kubeadm version -o short
v1.19.2
```

Zum Abgleich der Kompatibilität siehe die o. g. Website. Nach der Prüfung muss das Paket `kubeadm` im Paketmanager `apt` auf `unhold` gesetzt werden, damit eine neue Version installiert werden kann:

```
# apt-mark unhold kubeadm
```

Anschließend installieren Sie die Version des Pakets, das der Kubernetes-Version ent- spricht, auf die Sie Ihren Cluster upgraden wollen:

```
# apt-get install kubeadm=1.20.1-00
```

Schließlich sollte das Paket im Paketmanager wieder auf hold gesetzt werden:

```
# apt-mark hold kubeadm
```

Alternativ können Sie das Paket auch in einem einzigen Befehl installieren:

```
# apt-get install -y --allow-change-held-packages kubeadm=1.20.x-00
```

Upgrade Master1

Ist die gewünschte und der zukünftigen Cluster-Version entsprechende Version von `kubeadm` installiert, können die letzten Vorbereitungen für das Update getroffen wer- den. Mit den Parametern `upgrade plan` für `kubeadm` überprüft dieses den Zustand des Clusters und zeigt Ihnen die ermittelten Möglichkeiten für ein Up- sowie ein Downgrade an:

```
# kubeadm upgrade plan
```

Images überprüfen

Sofern eine eigene Container-Registry verwendet wird, muss sichergestellt sein, dass die nötigen Images für die Static-Pods verfügbar sind. Dies überprüfen Sie mit `kubeadm` und den Parametern `config images list`:

```
# kubeadm config images list
k8s.gcr.io/kube-apiserver:v1.20.1
k8s.gcr.io/kube-controller-manager:v1.20.1
k8s.gcr.io/kube-scheduler:v1.20.1
k8s.gcr.io/kube-proxy:v1.20.1
k8s.gcr.io/pause:3.2
k8s.gcr.io/etcd:3.4.13-0
k8s.gcr.io/coredns:1.7.0
```

Zudem gibt es mit der Option `pull` die Möglichkeit, die Images bereits vor dem Upgrade auf den Master Node herunterzuladen:

```
# kubeadm config images pull
[config/images] Pulled k8s.gcr.io/kube-apiserver:v1.20.1
[config/images] Pulled k8s.gcr.io/kube-controller-manager:v1.20.1
[config/images] Pulled k8s.gcr.io/kube-scheduler:v1.20.1
[config/images] Pulled k8s.gcr.io/kube-proxy:v1.20.1
[config/images] Pulled k8s.gcr.io/pause:3.2
[config/images] Pulled k8s.gcr.io/etcd:3.4.13-0
[config/images] Pulled k8s.gcr.io/coredns:1.7.0
```

Das Upgrade durchführen

Sind alle gezeigten Vorbereitungen abgeschlossen und alle Komponenten vorhanden und richtig konfiguriert, führen Sie das Upgrade mit dem folgenden Befehl aus:

```
# kubeadm upgrade apply <version>
```

Nachdem das Upgrade erfolgreich durchgeführt wurde, erscheint eine Erfolgsmeldung und der Hinweis, das kubelet ebenfalls upzugraden:

```
[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.20.1".
    Enjoy!
[upgrade/kubelet] Now that your control plane is upgraded, please
    proceed with upgrading your kubelets if you haven't already done
    so.
```

kubelet und kubectl upgraden

Zuletzt werden die dem Cluster (Apiserver) entsprechenden Paketversionen von kubelet und kubectl

installiert:

```
# apt-mark unhold kubelet kubectl
# apt-get install kubelet=1.20.1-00 kubectl=1.20.1-00
# apt-mark hold "kube*"
```

Den Master1 wieder in das Scheduling aufnehmen

Nachdem Sie das Upgrade auf die neue Version vollzogen haben, nehmen Sie den Node mit dem Befehl `kubectl uncordon` wieder in das Scheduling auf:

```
$ kubectl uncordon master1
node/master1 uncordoned
```

Der Node ist nun wieder einsatzbereit und auf die neue Version aktualisiert:

> code bash>

$ kubectl get nodes </code>

NAME	STATUS	ROLES	AGE	
master1	Ready	control-plane,master	107d	v1.20.1
master2	Ready	control-plane,master	107d	v1.19.2
master3	Ready	control-plane,master	107d	v1.19.2
worker1	Ready	<none>	107d	v1.19.2
worker2	Ready	<none>	107d	v1.19.2

Der Master1 wird nun als Grundlage für das Upgrade der restlichen Control Plane die- nen.

Hinweis Erneuerung der Zertifikate des Clusters Im Zuge eines Cluster-Upgrades werden die Zertifikate des Clusters um ein Jahr ver- längert. Mit dem Befehl „kubeadm alpha certs check-expiration“ können Sie dies überprüfen und nachvollziehen.

Upgrade Master{2,. . . }

Das Vorgehen beim Upgrade weiterer Master in der Control Plane entspricht weitgehend dem Vorgehen des Upgrades beim Master1. Der einzige Unterschied sind die Parameter der upgrade-Option von kubeadm.

Upgrade

Master Nodes drainen

Zunächst müssen auch die weiteren Master Nodes mit `kubectl drain` von ihrer Work- load befreit und aus dem Scheduling genommen werden:

```
$ kubectl drain master2 --ignore-daemonsets
node/tmaster2 already cordoned
WARNING: ignoring DaemonSet-managed Pods:
    kube-system/calico-node-vz27p, kube-system/kube-proxy-sq2lm
node/master2 drained
$ kubectl get nodes master2
```

NAME	STATUS	ROLES	AGE	
master2	Ready,SchedulingDisabled	control-plane,master	109d	v1.19.2

Status und Konfiguration überprüfen

Auch vor den Upgrades weiterer Master Nodes ist es sinnvoll, den Status des etcd-Clusters sowie die Version der Api des Masters zu überprüfen. **Überprüfen des etcd:**

```
$ kubectl get pod -n kube-system --selector component=etcd
```

NAME	READY	STATUS	RESTARTS	
etcd-master1	1/1	Running	13	5d1h
etcd-master2	1/1	Running	158	103d
etcd-master3	1/1	Running	194	103d

Überprüfen der Api-Version mit curl:

```
curl https://<IP-Adress of MasterNode>:6443/version -k
{
"major": "1",
"minor": "19",
"gitVersion": "v1.19.2",
"gitCommit": "f5743093fd1c663cb0cbc89748f730662345d44d",
"gitTreeState": "clean",
"buildDate": "2020-09-16T13:32:58Z",
"goVersion": "go1.15",
"compiler": "gc",
"platform": "linux/amd64"
}%
```

Upgrade durchführen

Damit das Upgrade durchgeführt werden kann, muss zunächst auf dem jeweiligen Mas- ter Node die gewünschte kubeadm-Version installiert werden. Dafür setzen Sie zunächst das Paket auf unhold:

```
$ sudo apt-mark unhold kubeadm
Canceled hold on kubeadm
```

Dann wird unter der Angabe der Version kubeadm installiert:

```
$ sudo apt-get install kubeadm=1.20.1-00
```

Jetzt wird der Node mit dem Parameter node für die Option upgrade von kubeadm auf den Versionsstand des Master1 gebracht:

```
$ sudo kubeadm upgrade node
[...]
[upgrade] The configuration for this node was successfully updated!
[upgrade] Now you should go ahead and upgrade the kubelet package
    using your package manager.
```

Damit ist das Upgrade auf dem Node durchgeführt. Jetzt müssen das kubelet und kubectl auf die neue Version gebracht werden, bevor Sie den Node wieder „uncordo- nen“.

kubectl und kubelet upgraden

Um die Binaries auf die neue Version zu upgraden, müssen diese im Paketmanager auf unhold gesetzt werden:

```
$ sudo apt-mark unhold kubelet kubectl
Canceled hold on kubelet
Canceled hold on kubectl
```

Anschließend werden die Pakete mit Angabe der Version installiert:

```
$ sudo apt-get install kubelet=1.20.1-00 kubectl=1.20.1-00
```

Schließlich werden alle Pakete wieder auf hold gesetzt:

```
$ sudo apt-mark hold "kube*"
kubernetes-cni was already set on hold.
kubeadm set on hold
kubelet set on hold
kubectl set on hold
```

Node „uncordonen“

Im letzten Schritt wird der Node wieder in das Scheduling aufgenommen. Damit ist das Upgrade abgeschlossen:

```
$ kubectl uncordon master2
node/master2 uncordoned
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	
master1	Ready	control-plane,master	110d	v1.20.1
master2	Ready	control-plane,master	109d	v1.20.1
master3	Ready	control-plane,master	109d	v1.19.2

Für alle weiteren Master Nodes verfahren Sie nach in der gleichen Art.

Upgrade Data Plane

Das Upgrade der Data Plane besteht im Prinzip ausschließlich aus dem Angleichen der Version des kubelets der Worker Nodes auf die entsprechende Version der Master No- des in der Control Plane. Dabei wird der Worker Node aus dem Scheduling entfernt und von Workloads bereinigt, bevor im Anschluss die neue Version des kubelets installiert und er wieder ins Scheduling aufgenommen wird.

Upgrade Worker Node

Node drainen

Führen Sie `kubectl drain` für den entsprechenden Worker aus:

```
$ kubectl drain worker1 --ignore-daemonsets
node/worker1 cordoned
WARNING: ignoring DaemonSet-managed Pods:
    kube-system/calico-node-2s4lv, kube-system/kube-proxy-k54gg
evicting pod kube-system/calico-kube-controllers-866f6f96b5-smldm
evicting pod kube-system/coredns-74ff55c5b-j8fkn
pod/coredns-74ff55c5b-j8fkn evicted
pod/calico-kube-controllers-866f6f96b5-smldm evicted
node/worker1 evicted
```

Upgrade des kubelet

Das Upgrade des `kubelet` wird mit Hilfe des Paketmanagers unter Angabe der genauen Version vorgenommen. Optional können auch die Pakete `kubeadm` und `kubectl` in der neuen Version installiert werden. Pakete im Paketmanager auf `unhold` setzen:

```
$ sudo apt-mark unhold kubelet kubeadm kubectl
Canceled hold on kubelet
Canceled hold on kubeadm
Canceled hold on kubectl
```

Anschließend werden die neuen Pakete installiert und dann wieder auf `hold` gesetzt: <coe bash> $ sudo apt-get install kubelet=1.20.1-00 kubectl=1.20.1-00

```
  kubeadm=1.20.1-00
```

$ sudo apt-mark hold kubeadm kubelet kubectl kubeadm set on hold kubelet set on hold kubectl set on hold </code>

Nun wird der Node wieder „uncordoned“, sodass auf ihm wieder Workloads platziert werden können:

```
$ kubectl uncordon worker1
node/worker1 uncordoned
```

Damit ist das Upgrade für den Worker abgeschlossen:

```
$ kubectl get nodes worker1
```

NAME	STATUS	ROLES	AGE
worker1	Ready	<none>	109d

Verfahren Sie für alle weiteren Worker Nodes in Ihrer Data Plane auf dieselbe Weise.