KubernetesクラスタのHA構成

森　克彦

2018年11月7日

VA LINUX SYSTEMS JAPAN

VA Linux Systems Japan株式会社

自己紹介

以前はWebアプリケーション開発

現在は

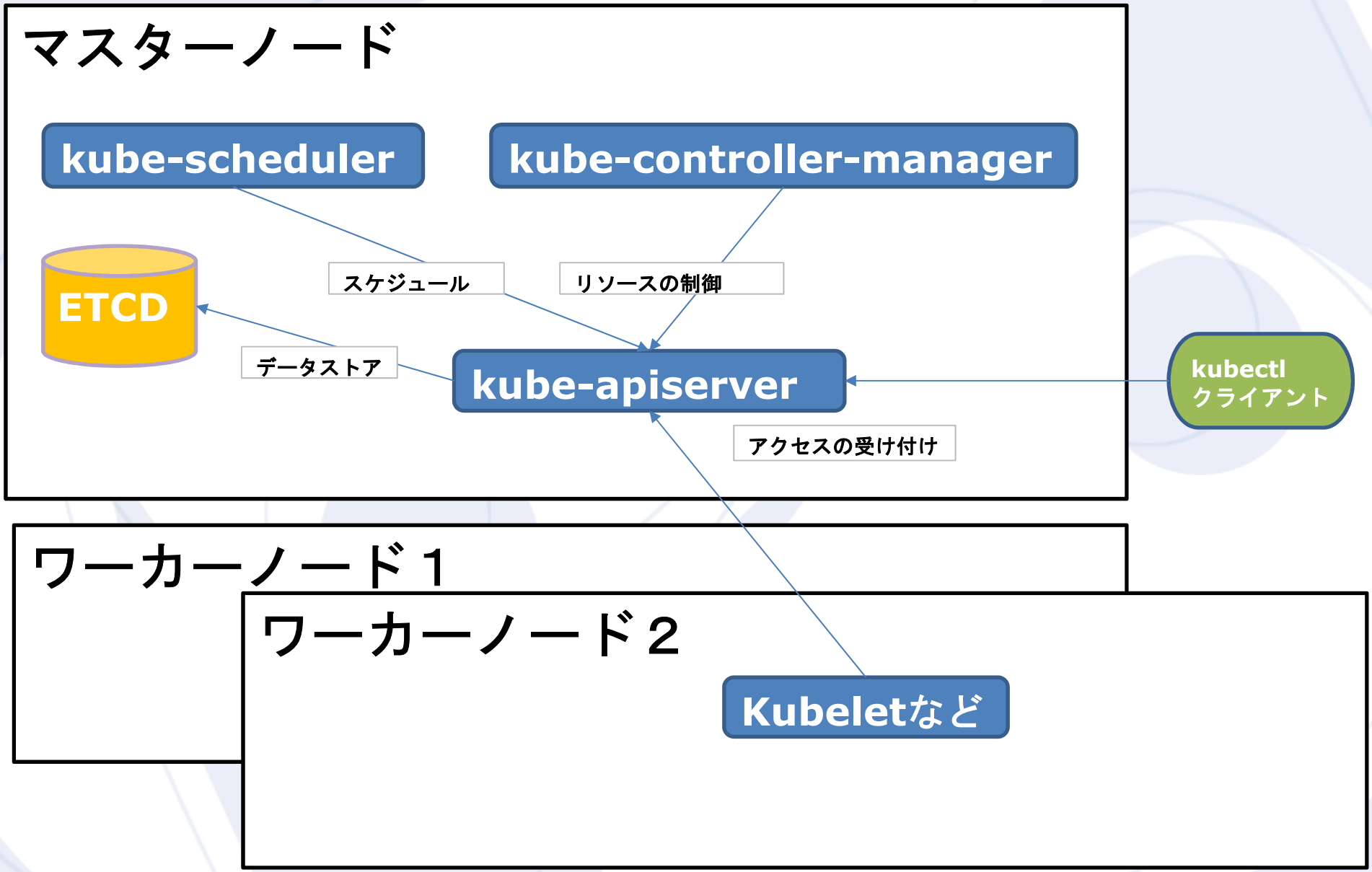
- Openstack
- docker
- Kubernetes

など２年半ほど

得意な言語：

- python

通常のk8sクラスタ構成（非HA）



通常のクラスタ構成（非HA)の問題点とその解決方法

通常のクラスタ（非HA）：

K8sのリソースの冗長化が可能

通常のクラスタの問題点：

マスターノードが１つのみ

マスターノードが落ちるとクラスタは管理機能を失う

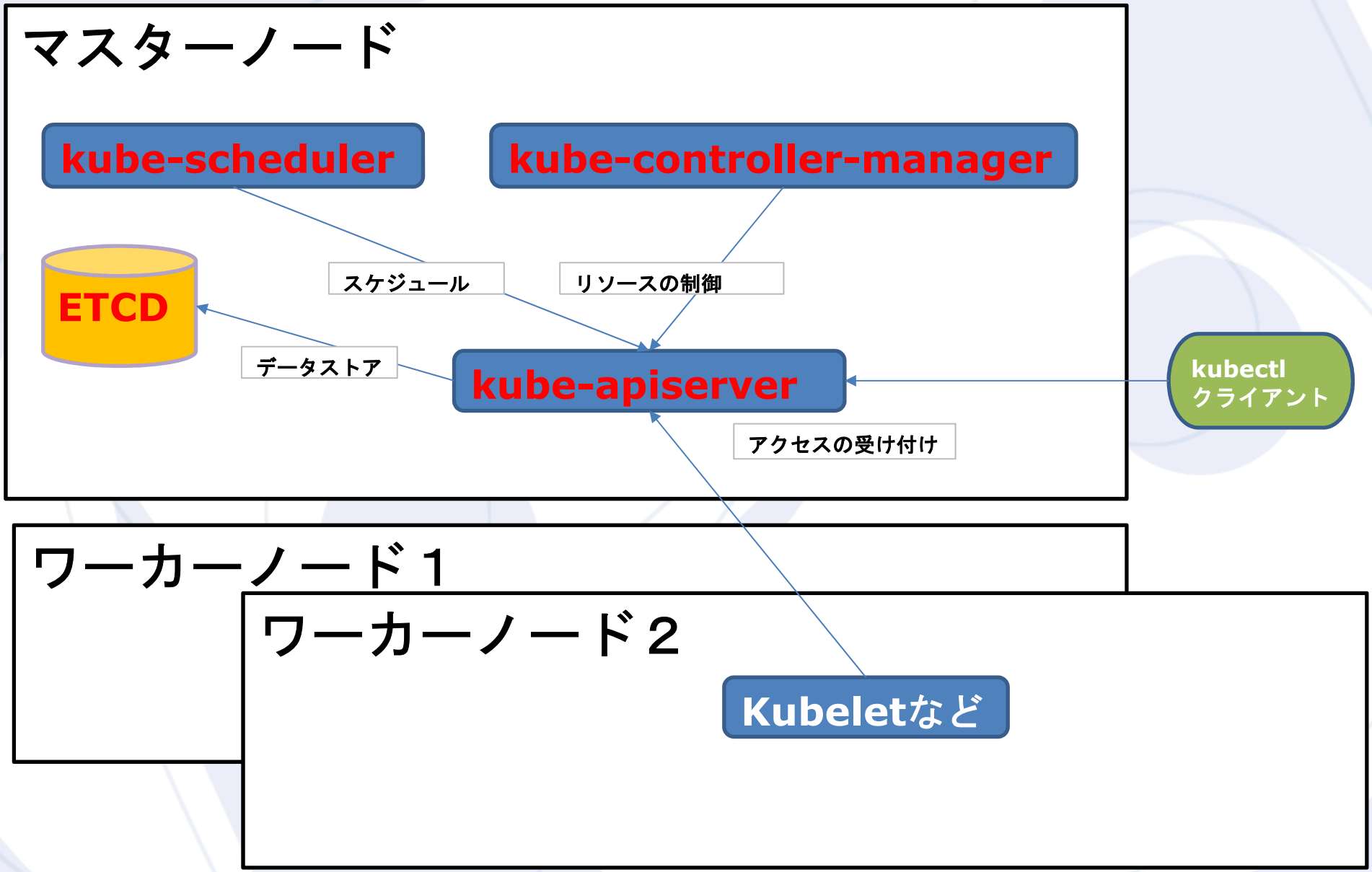
解決方法：

マスターノードを複数台用意

1台落ちてもクラスタの機能が保たれる

ここでは、マスターノードの冗長化のやり方を示す

冗長化が必要なもの（赤色の字）



冗長化の方法

kube-apiserver

- ロードバランサー配下に複数動作

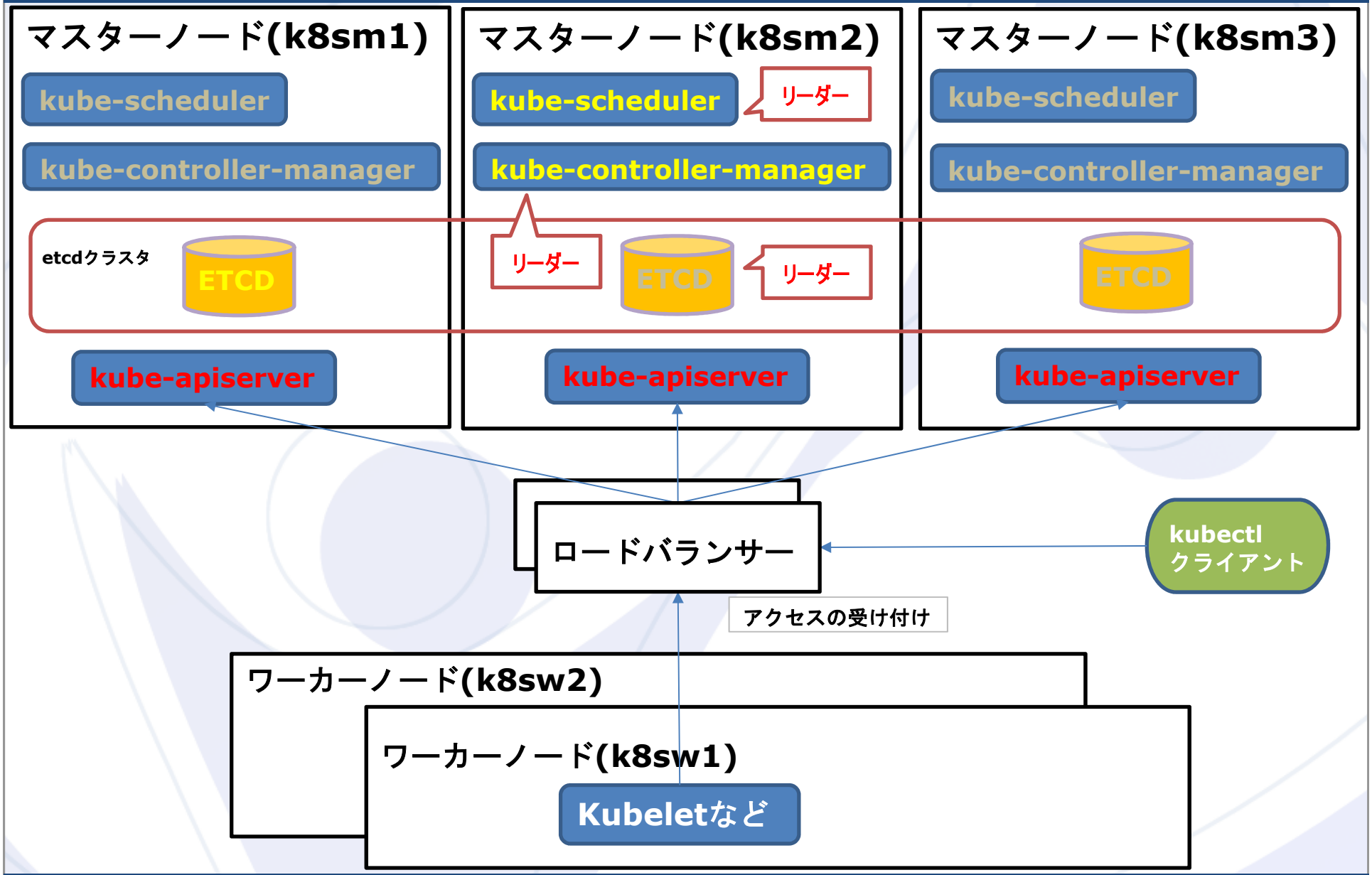
kube-schedulerとkube-controller-manager

- リーダー選出の仕組みあり
- Active-standby
- 特に設定する必要なし

etcd

- 奇数の数のノードで冗長化
- リーダー選出の仕組みあり

k8sクラスタのHA構成（シンプルな構成）



誰かリーダーかを確認（etcd)

K8sm2のetcdがリーダー

```
ubuntu@k8sm1:~$ sudo ETCDCTL_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt
+------------------+---------+-------+---------------------------+---------------------------+
|        ID        | STATUS  | NAME  |        PEER ADDRS         |       CLIENT ADDRS        |
+------------------+---------+-------+---------------------------+---------------------------+
|  8f7e7277bca9e21 | started | k8sm3 |  https://192.168.0.3:2380 |  https://192.168.0.3:2379 |
|  9fbed3041fa6d4b | started | k8sm2 | https://192.168.0.21:2380 | https://192.168.0.21:2379 |
| aaaacf0529a32f2f | started | k8sm1 |  https://192.168.0.9:2380 |  https://192.168.0.9:2379 |
+------------------+---------+-------+---------------------------+---------------------------+
ubuntu@k8sm1:~$ sudo ETCDCTL_API=3 etcdctl --endpoints=https://k8sm1:2379 --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=
+------------------------+------------------+---------+---------+-----------+-----------+------------+
|        ENDPOINT        |        ID        | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
+------------------------+------------------+---------+---------+-----------+-----------+------------+
| https://k8sm1:2379 | aaaacf0529a32f2f |  3.2.24 |  3.7 MB |     false |         8 |     188055 |
+------------------------+------------------+---------+---------+-----------+-----------+------------+
ubuntu@k8sm1:~$ sudo ETCDCTL_API=3 etcdctl --endpoints=https://k8sm2:2379 --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=
+------------------------+------------------+---------+---------+-----------+-----------+------------+
|        ENDPOINT        |        ID        | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
+------------------------+------------------+---------+---------+-----------+-----------+------------+
| https://k8sm2:2379 | 9fbed3041fa6d4b |  3.2.24 |  3.3 MB |      true |         8 |     188059 |
+------------------------+------------------+---------+---------+-----------+-----------+------------+
ubuntu@k8sm1:~$ sudo ETCDCTL_API=3 etcdctl --endpoints=https://k8sm3:2379 --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key=
+------------------------+------------------+---------+---------+-----------+-----------+------------+
|        ENDPOINT        |        ID        | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
+------------------------+------------------+---------+---------+-----------+-----------+------------+
| https://k8sm3:2379 | 8f7e7277bca9e21 |  3.2.24 |  3.3 MB |     false |         8 |     188071 |
+------------------------+------------------+---------+---------+-----------+-----------+------------+
ubuntu@k8sm1:~$
```

誰かリーダーかを確認（K8sプロセス）

k8sm2のkube-schedulerがリーダー

```
ubuntu@k8sm1:~$ kubectl logs kube-scheduler-k8sm1 -n=kube-system
I1107 02:59:15.812260       1 server.go:128] Version: v1.12.2
W1107 02:59:15.877551       1 defaults.go:210] TaintNodesByCondition is enabled, PodToleratesNodeTaints predicate is mandatory
W1107 02:59:15.878821       1 authorization.go:47] Authorization is disabled
W1107 02:59:15.878894       1 authentication.go:55] Authentication is disabled
I1107 02:59:15.878938       1 deprecated_insecure_serving.go:48] Serving healthz insecurely on 127.0.0.1:10251
I1107 02:59:18.034442       1 controller_utils.go:1027] Waiting for caches to sync for scheduler controller
I1107 02:59:18.134608       1 controller_utils.go:1034] Caches are synced for scheduler controller
I1107 02:59:18.134883       1 leaderelection.go:187] attempting to acquire leader lease  kube-system/kube-scheduler...
ubuntu@k8sm1:~$ kubectl logs kube-scheduler-k8sm3 -n=kube-system
I1107 05:23:20.349013       1 server.go:128] Version: v1.12.2
W1107 05:23:20.398602       1 defaults.go:210] TaintNodesByCondition is enabled, PodToleratesNodeTaints predicate is mandatory
W1107 05:23:20.400217       1 authorization.go:47] Authorization is disabled
W1107 05:23:20.400523       1 authentication.go:55] Authentication is disabled
I1107 05:23:20.400627       1 deprecated_insecure_serving.go:48] Serving healthz insecurely on 127.0.0.1:10251
I1107 05:23:21.702190       1 controller_utils.go:1027] Waiting for caches to sync for scheduler controller
I1107 05:23:21.802304       1 controller_utils.go:1034] Caches are synced for scheduler controller
I1107 05:23:21.802337       1 leaderelection.go:187] attempting to acquire leader lease  kube-system/kube-scheduler...
ubuntu@k8sm1:~$ kubectl logs kube-scheduler-k8sm2 -n=kube-system
I1106 07:31:15.184728       1 server.go:128] Version: v1.12.2
W1106 07:31:15.185636       1 defaults.go:210] TaintNodesByCondition is enabled, PodToleratesNodeTaints predicate is mandatory
W1106 07:31:15.192603       1 authorization.go:47] Authorization is disabled
W1106 07:31:15.192675       1 authentication.go:55] Authentication is disabled
I1106 07:31:15.192755       1 deprecated_insecure_serving.go:48] Serving healthz insecurely on 127.0.0.1:10251
I1106 07:31:16.111178       1 controller_utils.go:1027] Waiting for caches to sync for scheduler controller
I1106 07:31:16.211475       1 controller_utils.go:1034] Caches are synced for scheduler controller
I1106 07:31:16.211715       1 leaderelection.go:187] attempting to acquire leader lease  kube-system/kube-scheduler...
I1106 07:31:36.936178       1 leaderelection.go:196] successfully acquired lease kube-system/kube-scheduler
```

K8sm2のkube-controller-manager がリーダー

```
ubuntu@k8sm1:~$ kubectl logs kube-controller-manager-k8sm2 -n=kube-system | grep leader
I1106 07:31:15.830909       1 leaderelection.go:187] attempting to acquire leader lease  kube-system/kube-controller-manager...
I1106 07:31:36.607196       1 leaderelection.go:196] successfully acquired lease kube-system/kube-controller-manager
I1106 07:31:36.607972       1 event.go:221] Event(v1.ObjectReference{Kind:"Endpoints", Namespace:"kube-system", Name:"kube-controller-ma
eason: 'LeaderElection' k8sm2_f1d2ec5b-e195-11e8-9c58-fa163ea6461? became leader
```

使用するソフトウェアとOS

Docker　17.12.1-ce
Kubernetes：1.12.2
OS:　ubuntu18.04

ここでは、kubeadmを使用して
KubernetesのHAクラスタを構築

HAクラスタの構築手順

1. ロードバランサーを作成
2. ノード共通の設定
3. マスターノードの設定（最初の1台）
4. マスターノードの設定（2，3台目）
5. ワーカーノードを登録とCNIの設定

１．ロードバランサーを作成

１．haproxyをインストール

２．/etc/haproxy/haproxy.cfgを設定

３．フロントエンドはポート6443をlisten

４．バックエンドはマスターノードのIP＋ポート(6443)に転送設定

２.ノード共通の設定（通常のクラスタ設定と同様）

１．macアドレスとproduct_uuidがユニークかを確認

２．swapを無効に設定

３．/etc/hostsの編集

４．必要なポートを開放

５．dockerのインストール

６．kubeadm kubeletのインストール

参照URL　https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/

３．マスターノードの設定（最初の1台）

１．Kubeadm用のYAMLファイルを作成

３つの要点

・使うKubernetesのバージョンを指定

・ロードバランサーをアクセスエンドポイントに設定

・新規のetcdクラスタ作成を指示

２．作成したYAMLファイルでkubeadm init を実行

３．/etc/kubernetes/admin.confとpki証明書をコピーをその他のマスターノードにコピー

参照URL　https://kubernetes.io/docs/setup/independent/high-availability/

４．マスターノードの設定（2、3台目）

１．Kubeadm用のYAMLファイルを作成
　　3つの要点
　　・使うKubernetesのバージョンを指定
　　・アクセスエンドポイントをロードバランサーに設定
　　・既存のetcdクラスタにメンバーとして登録

２．Kubeadmで設定

３．etcdを設定

４．マスターノードとして追加

参照URL　https://kubernetes.io/docs/setup/independent/high-availability/

５．ワーカーノードを登録とCNIの設定

１．kubeadm joinを実行してワーカーノードを登録

```
ubuntu@k8sw1:~$ sudo kubeadm join k8slb:6443 --token 65ivj1.hxj7xvhbyd53r4x3 --discovery-token-ca-cert-hash sha256:eb748b20c648028abd7690911c61bb14eb9f6c6ef0631e35fc3c660e73c19792
```

```
ubuntu@k8sw2:~$ sudo  kubeadm join k8slb:6443 --token 65ivj1.hxj7xvhbyd53r4x3 --discovery-token-ca-cert-hash sha256:eb748b20c648028abd7690911c61bb14eb9f6c6ef0631e35fc3c660e73c19792
```

２．CalicoYAMLファイルを使用してCNIを設定

```
ubuntu@k8sm1:~$ kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/rbac-kdd.yaml
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
ubuntu@k8sm1:~$ kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installation/hosted/kubernetes-datastore/calico-networking/1.7/calico.yaml
configmap/calico-config created
service/calico-typha created
deployment.apps/calico-typha created
poddisruptionbudget.policy/calico-typha created
daemonset.extensions/calico-node created
serviceaccount/calico-node created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
ubuntu@k8sm1:~$
```

確認

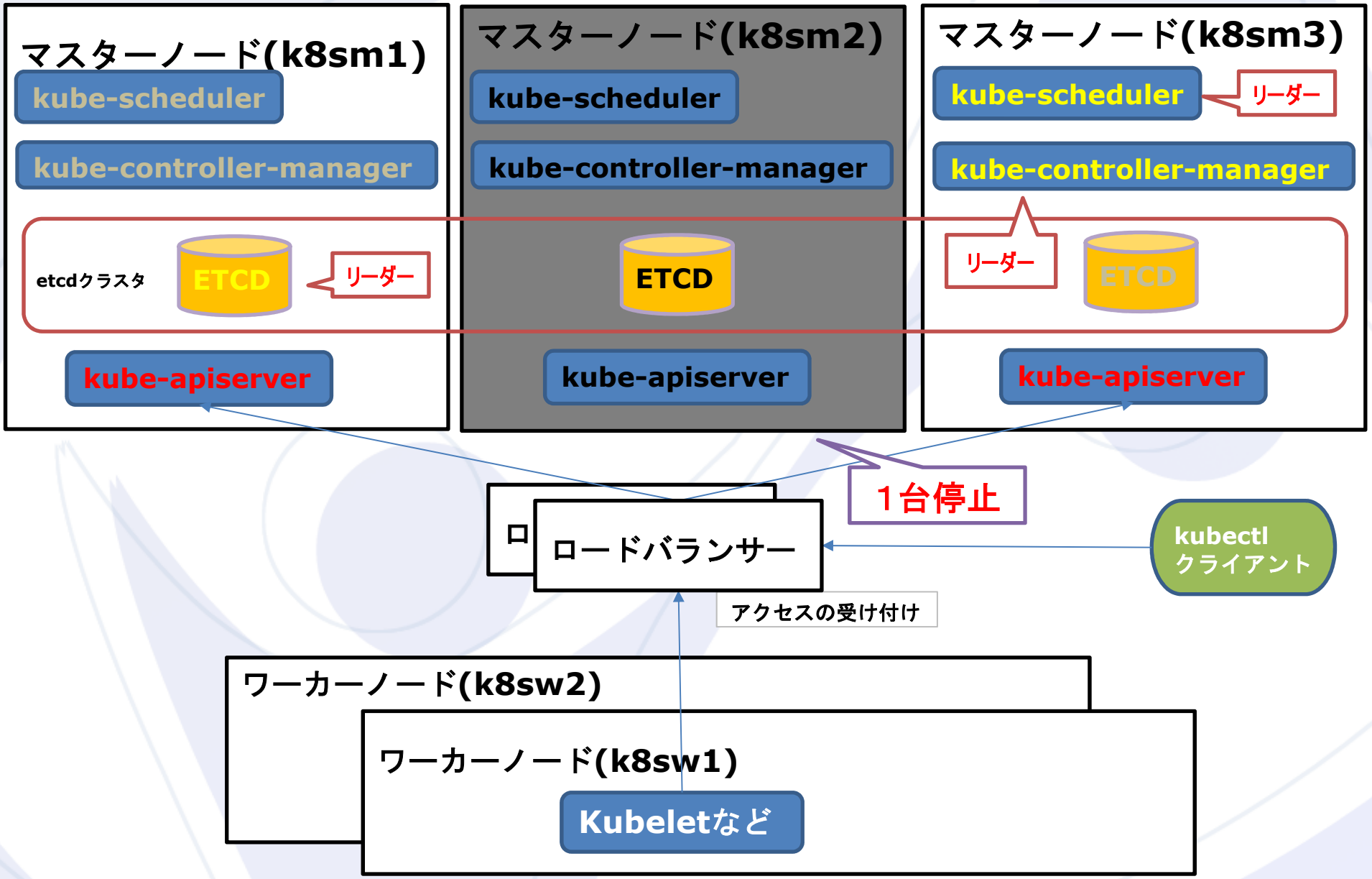
```
ubuntu@k8sm1:~$ kubectl get pods -n=kube-system
NAME                           READY   STATUS    RESTARTS   AGE
calico-node-49s7t              2/2     Running   0          3m21s
calico-node-5zwm5              2/2     Running   0          12m
calico-node-h8jhs              2/2     Running   0          12m
calico-node-rdplk              2/2     Running   0          14m
calico-node-wnlmp              2/2     Running   0          5m23s
coredns-576cbf47c7-dgwlx       1/1     Running   0          15m
coredns-576cbf47c7-lclmp       1/1     Running   0          15m
etcd-k8sm1                     1/1     Running   0          14m
etcd-k8sm2                     1/1     Running   0          11m
etcd-k8sm3                     1/1     Running   0          12m
kube-apiserver-k8sm1           1/1     Running   0          14m
kube-apiserver-k8sm2           1/1     Running   0          11m
kube-apiserver-k8sm3           1/1     Running   0          12m
kube-controller-manager-k8sm1  1/1     Running   1          14m
kube-controller-manager-k8sm2  1/1     Running   0          12m
kube-controller-manager-k8sm3  1/1     Running   0          12m
kube-proxy-8dgsk               1/1     Running   0          15m
kube-proxy-fb82h               1/1     Running   0          12m
kube-proxy-gkm5f               1/1     Running   0          5m23s
kube-proxy-k42mw               1/1     Running   0          3m21s
kube-proxy-ntslj               1/1     Running   0          12m
kube-scheduler-k8sm1           1/1     Running   1          14m
kube-scheduler-k8sm2           1/1     Running   0          11m
kube-scheduler-k8sm3           1/1     Running   0          12m
ubuntu@k8sm1:~$ kubectl get nodes -o wide
NAME    STATUS   ROLES    AGE     VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
k8sm1   Ready    master   16m     v1.12.2   192.168.0.9    <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sm2   Ready    master   13m     v1.12.2   192.168.0.21   <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sm3   Ready    master   13m     v1.12.2   192.168.0.3    <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sw1   Ready    <none>   5m42s   v1.12.2   192.168.0.22   <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sw2   Ready    <none>   3m40s   v1.12.2   192.168.0.29   <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
```

Deploymentで確認

```
ubuntu@k8sm1:~$ cat de.yaml
        apiVersion: apps/v1beta1
        kind: Deployment
        metadata:
          name: nginx-deployment3
          labels:
            app: nginx
        spec:
          replicas: 5
          selector:
            matchLabels:
              app: nginx
          template:
            metadata:
              labels:
                app: nginx
            spec:
              containers:
              - name: nginx
                image: nginx:1.12.1
                ports:
                - containerPort: 10080

ubuntu@k8sm1:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment3-6759fc9475-4kslh   1/1     Running   0          12m
nginx-deployment3-6759fc9475-bc7c4   1/1     Running   0          12m
nginx-deployment3-6759fc9475-dw7mh   1/1     Running   0          12m
nginx-deployment3-6759fc9475-h5nfr   1/1     Running   0          12m
nginx-deployment3-6759fc9475-p6qmt   1/1     Running   0          12m
```

フェールオーバー（1台落ちたー＞機能する）



フェールオーバーの確認

ノードとK8s用Podの状態を確認

```
ubuntu@k8sm1:~$ kubectl get nodes -o wide
NAME    STATUS     ROLES    AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
k8sm1   Ready      master   22h   v1.12.2   192.168.0.9    <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sm2   NotReady   master   22h   v1.12.2   192.168.0.21   <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sm3   Ready      master   22h   v1.12.2   192.168.0.3    <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sw1   Ready      <none>   22h   v1.12.2   192.168.0.22   <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
k8sw2   Ready      <none>   22h   v1.12.2   192.168.0.29   <none>        Ubuntu 18.04.1 LTS   4.15.0-38-generic   docker://17.12.1-ce
ubuntu@k8sm1:~$ kubectl get pods -n=kube-system
NAME                              READY   STATUS     RESTARTS   AGE
calico-node-49s7t                 1/2     Running    0          22h
calico-node-5zwm5                 1/2     Running    2          22h
calico-node-h8jhs                 2/2     NodeLost   0          22h
calico-node-rdplk                 1/2     Running    2          22h
calico-node-wnlmp                 1/2     Running    0          22h
coredns-576cbf47c7-9bs4m          1/1     Running    0          3h26m
coredns-576cbf47c7-lspfp          1/1     Running    0          3h26m
etcd-k8sm1                        1/1     Running    1          178m
etcd-k8sm2                        1/1     Unknown    0          22h
etcd-k8sm3                        1/1     Running    1          34m
kube-apiserver-k8sm1              1/1     Running    1          178m
kube-apiserver-k8sm2              1/1     Unknown    0          22h
kube-apiserver-k8sm3              1/1     Running    1          34m
kube-controller-manager-k8sm1     1/1     Running    2          178m
kube-controller-manager-k8sm2     1/1     Unknown    0          22h
kube-controller-manager-k8sm3     1/1     Running    1          34m
kube-proxy-8dgsk                  1/1     Running    1          22h
kube-proxy-fb82h                  1/1     NodeLost   0          22h
kube-proxy-gkm5f                  1/1     Running    0          22h
kube-proxy-k42mw                  1/1     Running    0          22h
kube-proxy-ntslj                  1/1     Running    1          22h
kube-scheduler-k8sm1              1/1     Running    2          178m
kube-scheduler-k8sm2              1/1     Unknown    0          22h
kube-scheduler-k8sm3              1/1     Running    1          34m
ubuntu@k8sm1:~$
```

フェールオーバーの確認 (リーダーの確認)

K8sm1のetcdがリーダーになった

```
ubuntu@k8sm1:~$ sudo ETCDCTL_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt
+------------------+---------+------+-------------------------+-------------------------+
|        ID        | STATUS  | NAME |        PEER ADDRS       |       CLIENT ADDRS      |
+------------------+---------+------+-------------------------+-------------------------+
|  8f7e7277bca9e21 | started | k8sm3 |  https://192.168.0.3:2380 |  https://192.168.0.3:2379 |
|  9fbed3041fa6d4b | started | k8sm2 | https://192.168.0.21:2380 | https://192.168.0.21:2379 |
| aaaacf0529a32f2f | started | k8sm1 |  https://192.168.0.9:2380 |  https://192.168.0.9:2379 |
+------------------+---------+------+-------------------------+-------------------------+
ubuntu@k8sm1:~$ sudo ETCDCTL_API=3 etcdctl --endpoints=https://k8sm1:2379 --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key
+-------------------+------------------+---------+---------+-----------+-----------+------------+
|      ENDPOINT     |        ID        | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
+-------------------+------------------+---------+---------+-----------+-----------+------------+
| https://k8sm1:2379 | aaaacf0529a32f2f |  3.2.24 |  3.7 MB |      true |         9 |     190012 |
+-------------------+------------------+---------+---------+-----------+-----------+------------+
ubuntu@k8sm1:~$ sudo ETCDCTL_API=3 etcdctl --endpoints=https://k8sm3:2379 --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt --key
+-------------------+------------------+---------+---------+-----------+-----------+------------+
|      ENDPOINT     |        ID        | VERSION | DB SIZE | IS LEADER | RAFT TERM | RAFT INDEX |
+-------------------+------------------+---------+---------+-----------+-----------+------------+
| https://k8sm3:2379 | 8f7e7277bca9e21 |  3.2.24 |  3.3 MB |     false |         9 |     190038 |
+-------------------+------------------+---------+---------+-----------+-----------+------------+
ubuntu@k8sm1:~$
```

フェールオーバーの確認 (リーダーの確認)

K8sm3のkube-schedulerがリーダーになった

```
ubuntu@k8sm1:~$ kubectl logs kube-scheduler-k8sm3 -n=kube-system
I1107 05:23:20.349013       1 server.go:126] Version: v1.12.2
W1107 05:23:20.398602       1 defaults.go:210] TaintNodesByCondition is enabled, PodToleratesNodeTaints predicate is mandatory
W1107 05:23:20.400217       1 authorization.go:47] Authorization is disabled
W1107 05:23:20.400523       1 authentication.go:55] Authentication is disabled
I1107 05:23:20.400627       1 deprecated_insecure_serving.go:48] Serving healthz insecurely on 127.0.0.1:10251
I1107 05:23:21.702190       1 controller_utils.go:1027] Waiting for caches to sync for scheduler controller
I1107 05:23:21.802304       1 controller_utils.go:1034] Caches are synced for scheduler controller
I1107 05:23:21.802337       1 leaderelection.go:187] attempting to acquire leader lease  kube-system/kube-scheduler...
E1107 05:45:01.610248       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.610377       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.610644       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.611021       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.611273       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.611302       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.611321       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.611339       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.611361       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
E1107 05:45:01.611375       1 streamwatcher.go:109] Unable to decode an event from the watch stream: http2: server sent GOAWAY and clos
W1107 05:45:01.618202       1 reflector.go:270] k8s.io/client-go/informers/factory.go:131: watch of *v1.ReplicationController ended wit
W1107 05:45:01.618502       1 reflector.go:270] k8s.io/client-go/informers/factory.go:131: watch of *v1.Service ended with: too old res
W1107 05:45:01.618583       1 reflector.go:270] k8s.io/client-go/informers/factory.go:131: watch of *v1beta1.PodDisruptionBudget ended
W1107 05:45:01.668593       1 reflector.go:270] k8s.io/client-go/informers/factory.go:131: watch of *v1.PersistentVolumeClaim ended wit
W1107 05:45:01.668925       1 reflector.go:270] k8s.io/client-go/informers/factory.go:131: watch of *v1.StorageClass ended with: too ol
W1107 05:45:01.669057       1 reflector.go:270] k8s.io/client-go/informers/factory.go:131: watch of *v1.StatefulSet ended with: too old
W1107 05:45:01.669202       1 reflector.go:270] k8s.io/client-go/informers/factory.go:131: watch of *v1.PersistentVolume ended with: to
I1107 05:45:17.470653       1 leaderelection.go:196] successfully acquired lease kube-system/kube-scheduler
```

K8sm3のkube-controller-managerがリーダーになった

```
ubuntu@k8sm1:~$ kubectl logs kube-controller-manager-k8sm3 -n=kube-system | grep leader
I1107 05:23:27.818980       1 leaderelection.go:187] attempting to acquire leader lease  kube-system/kube-controller-manager...
I1107 05:45:15.584784       1 leaderelection.go:196] successfully acquired lease kube-system/kube-controller-manager
I1107 05:45:15.585953       1 event.go:221] Event(v1.ObjectReference{Kind:"Endpoints", Namespace:"kube-system", Name:"kube-controller-m
' reason: 'LeaderElection' k8sm3_41bee86b-e24d-11e8-9e1f-fa163eeb62ee became leader
```

リカバリー方法（マスターノード1台だけ落ちた場合）

クラスタの設定が無事の場合：

マスターノードを再起動すれば正常動作

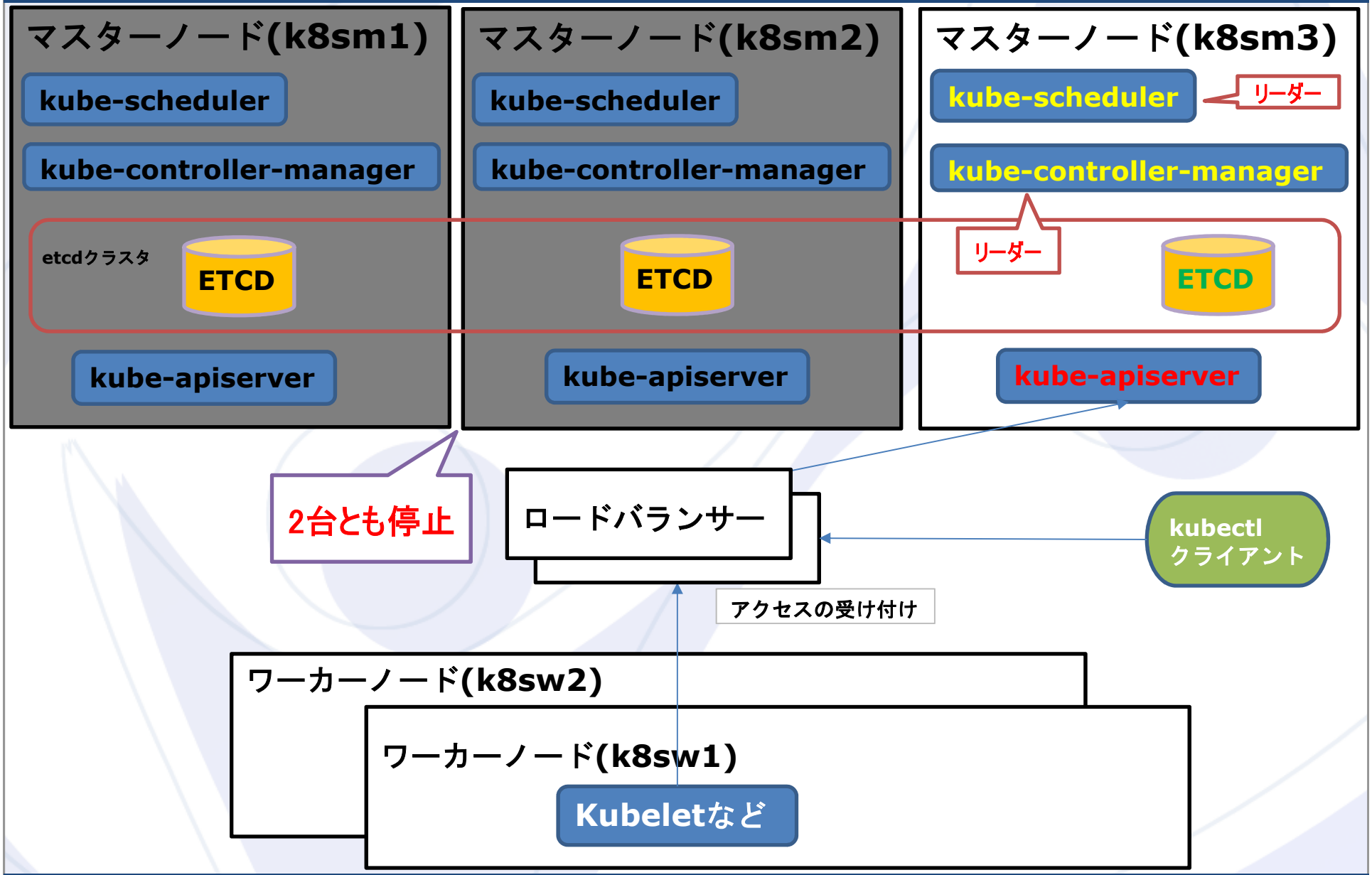
マスターノードが起動不可の場合：

新規のマスターノードを作成して追加する必要がある

＜手順＞

1. ロードバランサーの設定を更新
2. Etcdのクラスタから死んだマスターノードを削除
3. 3台目のマスターノード追加作業と同じ作業をする
4. Kubectl delete nodeで古いマスターノードを削除

フェールオーバー（残り１台ー＞機能しない）



フェールオーバーの確認

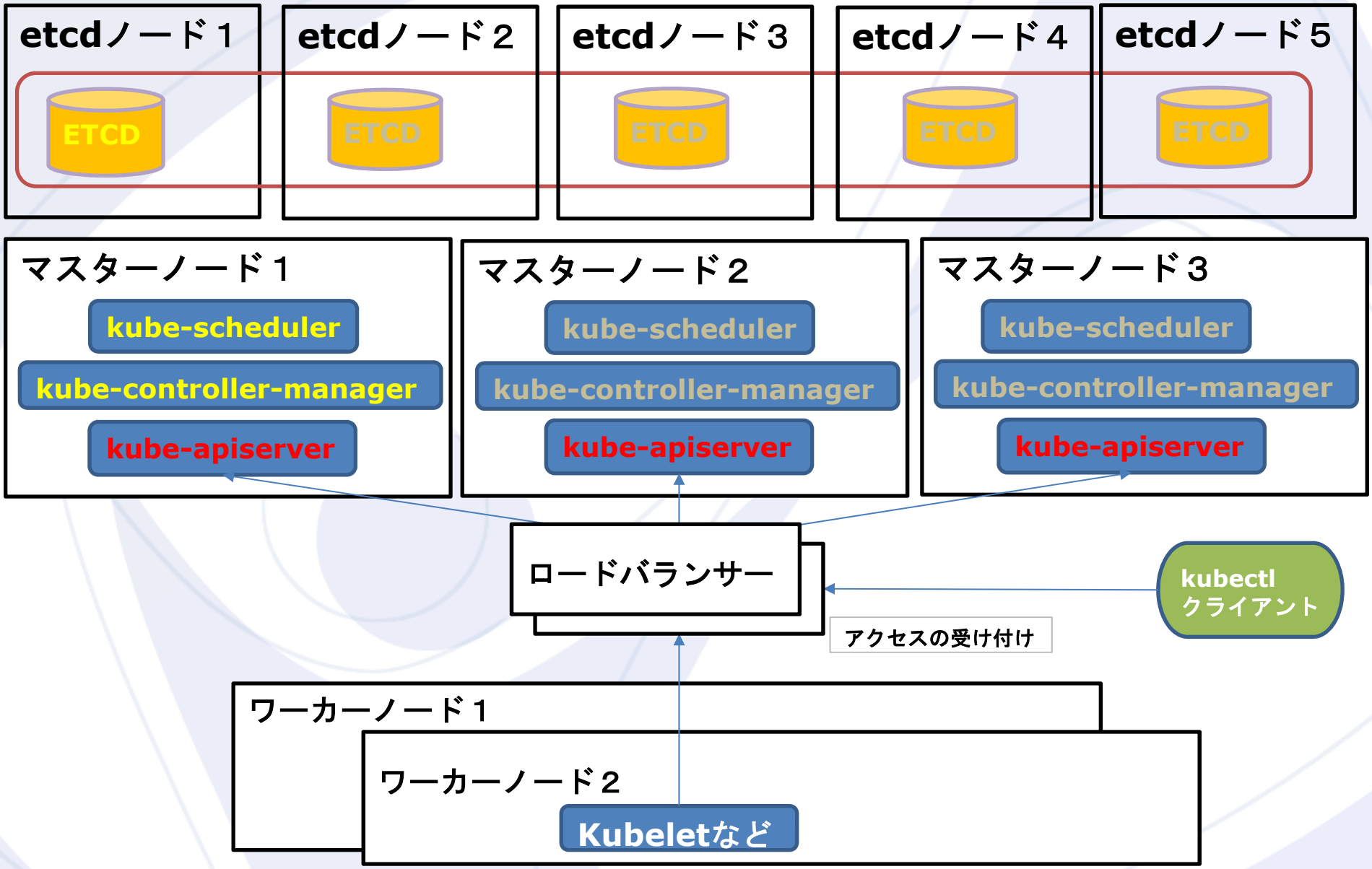
K8sm3のetcdが応答しない

```
2018-11-07 07:47:51.463360 W | rafthttp: health check for peer 8f7e7277bca9e21 could not connect: dial tcp 192.168.0.3:2380: getsockopt: no
2018-11-07 07:47:51.467466 W | rafthttp: health check for peer 9fbed3041fa6d4b could not connect: dial tcp 192.168.0.21:2380: i/o timeout
2018-11-07 07:47:51.927800 I | raft: aaaacf0529a32f2f is starting a new election at term 524
2018-11-07 07:47:51.927869 I | raft: aaaacf0529a32f2f became candidate at term 525
2018-11-07 07:47:51.927887 I | raft: aaaacf0529a32f2f received MsgVoteResp from aaaacf0529a32f2f at term 525
2018-11-07 07:47:51.927899 I | raft: aaaacf0529a32f2f [logterm: 9, index: 202778] sent MsgVote request to 8f7e7277bca9e21 at term 525
2018-11-07 07:47:51.927910 I | raft: aaaacf0529a32f2f [logterm: 9, index: 202778] sent MsgVote request to 9fbed3041fa6d4b at term 525
2018-11-07 07:47:52.468181 E | etcdserver: publish error: etcdserver: request timed out
2018-11-07 07:47:53.027819 I | raft: aaaacf0529a32f2f is starting a new election at term 525
2018-11-07 07:47:53.027894 I | raft: aaaacf0529a32f2f became candidate at term 526
2018-11-07 07:47:53.027943 I | raft: aaaacf0529a32f2f received MsgVoteResp from aaaacf0529a32f2f at term 526
2018-11-07 07:47:53.028150 I | raft: aaaacf0529a32f2f [logterm: 9, index: 202778] sent MsgVote request to 9fbed3041fa6d4b at term 526
2018-11-07 07:47:53.028179 I | raft: aaaacf0529a32f2f [logterm: 9, index: 202778] sent MsgVote request to 8f7e7277bca9e21 at term 526
2018-11-07 07:47:54.027865 I | raft: aaaacf0529a32f2f is starting a new election at term 526
2018-11-07 07:47:54.027935 I | raft: aaaacf0529a32f2f became candidate at term 527
2018-11-07 07:47:54.027953 I | raft: aaaacf0529a32f2f received MsgVoteResp from aaaacf0529a32f2f at term 527
2018-11-07 07:47:54.027967 I | raft: aaaacf0529a32f2f [logterm: 9, index: 202778] sent MsgVote request to 8f7e7277bca9e21 at term 527
2018-11-07 07:47:54.027978 I | raft: aaaacf0529a32f2f [logterm: 9, index: 202778] sent MsgVote request to 9fbed3041fa6d4b at term 527
```

kube-apiserverも応答しない―＞kubectlで何もできない

```
ubuntu@k8sm1:~$ kubectl get pods
Unable to connect to the server: EOF
ubuntu@k8sm1:~$
```

k8sクラスタのHA構成例（外部のetcdクラスタを利用）



作業詳細手順１（ロードバランサーの作成）

```
ubuntu@k8slb:~$ cat /etc/haproxy/haproxy.cfg
global
    log         127.0.0.1 local2 info
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     256
    user        haproxy
    group       haproxy
    daemon

defaults
    mode                    tcp
    log                     global
    option                  tcplog
    timeout connect         10s
    timeout client          30s
    timeout server          30s

frontend k8s
    bind *:6443
    mode                    tcp
    default_backend         k8s_backend

backend k8s_backend
    balance                 roundrobin
    server                  k8sm1 192.168.0.9:6443 check
    server                  k8sm2 192.168.0.20:6443 check
    server                  k8sm3 192.168.0.21:6443 check
ubuntu@k8slb:~$
```

作業詳細手順２（ノード共通の設定）

```
1. インストール後の作業

    1) rootでログインして、sudoを設定する
        # visudo

        1.1) root ALL=(ALL) ALLを検索して、その下に以下の内容を追加（auserはシステムにある一般ユーザ名に置換）。
            auser  ALL=(ALL) ALL

        1.2) ファイルを閉じてログアウトする。

        1.3) auserでログインし、sudoができること確認する。以降の作業はこのauserで行う。


    4) 以下コマンドを実行して、すべてのノードのhostnameが正しく設定されていることを確認する
        $ sudo cat /etc/hostname

    5) 以下コマンドを実行して、すべてのノードのmacアドレスとproduct_uuidがユニークであることを確認する
        $ sudo ip l
        $ sudo cat /sys/class/dmi/id/product_uuid

2. Dockerのインストール

        1) sudo apt-get install -y docker.io

        2) sudo gpasswd -a $USER docker

    5) dockerはデフォルトでdocker0というブリッジを作成する。そのCIDRが
      172.17.0.1/16で、この設定でよければ次の6)に進む。
       変更したい場合は以下のように作業が必要である。

        5.1) dockerのサービス設定を編集
            $ sudo vi /lib/systemd/system/docker.service

        5.2) ExecStartを検索して以下のように変更する、--bip=CIDRで追加
            前) ExecStart=/usr/bin/dockerd

            後) ExecStart=/usr/bin/dockerd ¥
                        --bip=10.0.0.1/16

        5.3) deamonを再読み込み
            $ sudo systemctl daemon-reload
            $ sudo systemctl restart docker


3. Kubeadmのインストール
    1) リポジトリを追加とパッケージのインストール

apt-get update && apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
apt-get update
apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl

    6) kubeletサービスを有効にして、起動する
        sudo systemctl daemon-reload
        sudo systemctl restart kubelet
        sudo systemctl enable kubelet
```

最初のマスターノードのyamlファイル

```
ubuntu@k8sm1:~$ cat kubeadm-config.yaml
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: 1.12.2
apiServerCertSANs:
- "k8slb"
controlPlaneEndpoint: "k8slb:6443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://192.168.0.9:2379"
      advertise-client-urls: "https://192.168.0.9:2379"
      listen-peer-urls: "https://192.168.0.9:2380"
      initial-advertise-peer-urls: "https://192.168.0.9:2380"
      initial-cluster: "k8sm1=https://192.168.0.9:2380"
    serverCertSANs:
      - k8sm1
      - 192.168.0.9
    peerCertSANs:
      - k8sm1
      - 192.168.0.9
networking:
    # This CIDR is a Calico default. Substitute or remove for your CNI provider.
    podSubnet: "192.168.0.0/16"
```

作業詳細手順３（最初のマスターノード）

```
ubuntu@k8sm1:~$ sudo kubeadm init --config kubeadm-config.yaml
[init] using Kubernetes version: v1.12.2
[preflight] running pre-flight checks
        [WARNING Service-Docker]: docker service is not enabled, please run 'systemctl enable docker.service'
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 17.12.1-ce. Latest validated version: 18.06
[preflight/images] Pulling images required for setting up a Kubernetes cluster
[preflight/images] This might take a minute or two, depending on the speed of your internet connection
[preflight/images] You can also perform this action in beforehand using 'kubeadm config images pull'
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[preflight] Activating the kubelet service
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
[certificates] etcd/server serving cert is signed for DNS names [k8sm1 localhost k8sm1] and IPs [127.0.0.1 ::1 192.168.0.9]
[certificates] Generated etcd/peer certificate and key.
[certificates] etcd/peer serving cert is signed for DNS names [k8sm1 localhost k8sm1] and IPs [192.168.0.9 127.0.0.1 ::1 192.168.0.9]
[certificates] Generated etcd/healthcheck-client certificate and key.
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
[certificates] Generated ca certificate and key.
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [k8sm1 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local k8slb k8slb] and IPs [10.96.0.1 192.168.0.9]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] valid certificates and keys now exist in "/etc/kubernetes/pki"
[certificates] Generated sa key and public key.
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[controlplane] wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/manifests/kube-apiserver.yaml"
[controlplane] wrote Static Pod manifest for component kube-controller-manager to "/etc/kubernetes/manifests/kube-controller-manager.yaml"
[controlplane] wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/manifests/kube-scheduler.yaml"
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/manifests/etcd.yaml"
[init] waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/kubernetes/manifests"
[init] this might take a minute or longer if the control plane images have to be pulled
[apiclient] All control plane components are healthy after 30.006665 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.12" in namespace kube-system with the configuration for the kubelets in the cluster
[markmaster] Marking the node k8sm1 as master by adding the label "node-role.kubernetes.io/master=''"
[markmaster] Marking the node k8sm1 as master by adding the taints [node-role.kubernetes.io/master:NoSchedule]
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "k8sm1" as an annotation
[bootstraptoken] using token: 0yap4x.epgg4d9jcvlppp3b
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join k8slb:6443 --token 0yap4x.epgg4d9jcvlppp3b --discovery-token-ca-cert-hash sha256:b9a471919592699974e105d1a210fcc71db128dba26f9619b0a7d3275e68ead9e
```

確認（最初のマスターノード）

```
ubuntu@k8sm1:~$ kubectl get nodes
NAME    STATUS     ROLES    AGE   VERSION
k8sm1   NotReady   master   22m   v1.12.2
ubuntu@k8sm1:~$ kubectl get pods -n=kube-system
NAME                              READY   STATUS    RESTARTS   AGE
coredns-576cbf47c7-m7zcm          0/1     Pending   0          21m
coredns-576cbf47c7-s56j8          0/1     Pending   0          21m
etcd-k8sm1                        1/1     Running   0          21m
kube-apiserver-k8sm1              1/1     Running   0          21m
kube-controller-manager-k8sm1     1/1     Running   0          21m
kube-proxy-xmxtn                  1/1     Running   0          21m
kube-scheduler-k8sm1              1/1     Running   0          21m
```

その他マスターノードのyamlファイル

```
ubuntu@k8sm2:~$ cat kubeadm-config.yaml
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: 1.12.2
apiServerCertSANs:
- "k8slb"
controlPlaneEndpoint: "k8slb:6443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://192.168.0.21:2379"
      advertise-client-urls: "https://192.168.0.21:2379"
      listen-peer-urls: "https://192.168.0.21:2380"
      initial-advertise-peer-urls: "https://192.168.0.21:2380"
      initial-cluster: "k8sm1=https://192.168.0.9:2380,k8sm2=https://192.168.0.21:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - k8sm2
      - 192.168.0.21
    peerCertSANs:
      - k8sm2
      - 192.168.0.21
networking:
    # This CIDR is a calico default. Substitute or remove for your CNI provider.
    podSubnet: "192.168.0.0/16"
```

```
ubuntu@k8sm3:~$ cat kubeadm-config.yaml
apiVersion: kubeadm.k8s.io/v1alpha3
kind: ClusterConfiguration
kubernetesVersion: 1.12.2
apiServerCertSANs:
- "k8slb"
controlPlaneEndpoint: "k8slb:6443"
etcd:
  local:
    extraArgs:
      listen-client-urls: "https://127.0.0.1:2379,https://192.168.0.3:2379"
      advertise-client-urls: "https://192.168.0.3:2379"
      listen-peer-urls: "https://192.168.0.3:2380"
      initial-advertise-peer-urls: "https://192.168.0.3:2380"
      initial-cluster: "k8sm1=https://192.168.0.9:2380,k8sm2=https://192.168.0.21:2380,k8sm3=https://192.168.0.3:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - k8sm3
      - 192.168.0.3
    peerCertSANs:
      - k8sm3
      - 192.168.0.3
networking:
    # This CIDR is a calico default. Substitute or remove for your CNI provider.
    podSubnet: "192.168.0.0/16"
```

作業詳細手順４（２，３台目マスターノード）

```
ubuntu@k8sm2:~$ cat run.sh
export CP0_IP=192.168.0.9
export CP0_HOSTNAME=k8sm1
export CP1_IP=192.168.0.21
export CP1_HOSTNAME=k8sm2

export KUBECONFIG=/etc/kubernetes/admin.conf
kubectl exec -n kube-system etcd-${CP0_HOSTNAME} -- etcdctl --ca-file /etc/kubernetes/pki/etcd/ca.crt --cert-file /etc/kubernetes/pki/etcd/peer.crt --key-file /etc/kubernetes
/pki/etcd/peer.key --endpoints=https://${CP0_IP}:2379 member add ${CP1_HOSTNAME} https://${CP1_IP}:2380
kubeadm alpha phase etcd local --config kubeadm-config.yaml
kubeadm alpha phase kubeconfig all --config kubeadm-config.yaml
kubeadm alpha phase controlplane all --config kubeadm-config.yaml
kubeadm alpha phase kubelet config annotate-cri --config kubeadm-config.yaml
kubeadm alpha phase mark-master --config kubeadm-config.yaml
```

```
ubuntu@k8sm3:~$ cat run.sh
kubeadm alpha phase certs all --config kubeadm-config.yaml
kubeadm alpha phase kubelet config write-to-disk --config kubeadm-config.yaml
kubeadm alpha phase kubelet write-env-file --config kubeadm-config.yaml
kubeadm alpha phase kubeconfig kubelet --config kubeadm-config.yaml
systemctl start kubelet

export CP0_IP=192.168.0.9
export CP0_HOSTNAME=k8sm1
export CP2_IP=192.168.0.3
export CP2_HOSTNAME=k8sm3

export KUBECONFIG=/etc/kubernetes/admin.conf
kubectl exec -n kube-system etcd-${CP0_HOSTNAME} -- etcdctl --ca-file /etc/kubernetes/pki/etcd/ca.crt --cert-file /etc/kubernetes/pki/etcd/peer.crt --key-file /etc/kubernetes/pki/etcd/peer.key --endpoint
s=https://${CP0_IP}:2379 member add ${CP2_HOSTNAME} https://${CP2_IP}:2380
kubeadm alpha phase etcd local --config kubeadm-config.yaml

kubeadm alpha phase kubeconfig all --config kubeadm-config.yaml
kubeadm alpha phase controlplane all --config kubeadm-config.yaml
kubeadm alpha phase kubelet config annotate-cri --config kubeadm-config.yaml
kubeadm alpha phase mark-master --config kubeadm-config.yaml
```

以上ありがとうございました。