1.1. Создание HA cluster

> Офдок: https://kubernetes.io/docs/setup/independent/high-availability/

Для этого дела нам понабится три машины. Выбранный режим кластера - **Stacked**, это значит, что **etcd** инстансы будут находиться на **API server** (мастер) нодах.

Кластер не должен быть инициализован. Сервис kubelet должен быть остановлен.

Файл **/etc/systemd/system/kubelet.service.d/10-kubeadm.conf** должен выглядеть так:

```
# Note: This dropin only works with kubeadm and kubelet v1.11+
[Service]
CPUAccounting=true
MemoryAccounting=true
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernete
s/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf
--cgroup-driver=systemd"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at
runtime, populating the KUBELET_KUBEADM_ARGS variable dynamically
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as
a last resort. Preferably, the user should use
# the .NodeRegistration.KubeletExtraArgs object in the configuration files
instead. KUBELET_EXTRA_ARGS should be sourced from this file.
EnvironmentFile=-/etc/sysconfig/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS
$KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
```

Настройка Control Pane 0

Создать файл **ha-cluster-m1.yaml:**

```
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
kubernetesVersion: v1.11.2
apiServerCertSANs:
- "kubernetes.vsk.ru"
api:
    controlPlaneEndpoint: "kubernetes.vsk.ru:6443"
etcd:
  local:
    extraArgs:
      listen-client-urls:
"https://127.0.0.1:2379,https://192.168.66.206:2379"
      advertise-client-urls: "https://192.168.66.206:2379"
      listen-peer-urls: "https://192.168.66.206:2380"
      initial-advertise-peer-urls: "https://192.168.66.206:2380"
      initial-cluster: "k8sm1=https://192.168.66.206:2380"
    serverCertSANs:
      - k8sm1.vsk.ru
      - 192.168.66.206
    peerCertSANs:
      - k8sm1.vsk.ru
      - 192.168.66.206
```

Выполнить инициализирующие команды (файл kubelet-setup-m1.sh):

```
sudo kubeadm init --config ha-cluster-m1.yaml
sudo su
./scopy-certs.sh
exit
sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
export kubever=$(kubectl version | base64 | tr -d '\n')
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
```

Подобный вывод команды **kubeadm init** нужно сохранить для последующего join worker node в наш кластер:

это команда для join в кластер доп worker'ов

```
kubeadm join 192.168.66.206:6443 --token msrvry.t7cd0zgoi9mwvmwt
--discovery-token-ca-cert-hash
sha256:a8b0ce2e19a6a972826391da4fcafb3e7f6b08488c440412873540d5615fa
e1c
```

Скопировать сертификаты на другие машины (файл scopy-certs.sh):

```
#!/usr/bin/env bash
eval $(ssh-agent)
ssh-add /home/user/.ssh/id_rsa
USER=user
CONTROL_PLANE_IPS="192.168.66.207 192.168.66.131"
for host in ${CONTROL_PLANE_IPS}; do
    scp /etc/kubernetes/pki/ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.pub "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/etcd/ca.crt "${USER}"@$host:etcd-ca.crt
    scp /etc/kubernetes/pki/etcd/ca.key "${USER}"@$host:etcd-ca.key
    scp /etc/kubernetes/admin.conf "${USER}"@$host:
    scp copy-certs.sh "${USER}"@$host:
done
scp kubelet-setup-m2.sh "${USER}"@192.168.66.207:
scp ha-cluster-m2.yaml  "${USER}"@192.168.66.207:
scp kubelet-setup-m3.sh "${USER}"@192.168.66.131:
scp ha-cluster-m3.yaml  "${USER}"@192.168.66.131:
```

Настройка Control Pane 1

Создать файл **ha-cluster-m2.yaml:**

```
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
kubernetesVersion: v1.11.2
apiServerCertSANs:
- "kubernetes.vsk.ru"
api:
    controlPlaneEndpoint: "kubernetes.vsk.ru:6443"
etcd:
  local:
    extraArgs:
      listen-client-urls:
"https://127.0.0.1:2379,https://192.168.66.207:2379"
      advertise-client-urls: "https://192.168.66.207:2379"
      listen-peer-urls: "https://192.168.66.207:2380"
      initial-advertise-peer-urls: "https://192.168.66.207:2380"
      initial-cluster:
"k8sm1=https://192.168.66.206:2380,k8sm2=https://192.168.66.207:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - k8sm2.vsk.ru
      - 192.168.66.207
    peerCertSANs:
      - k8sm2.vsk.ru
      - 192.168.66.207
```

Скопировать сертификаты, запустив файл copy-certs.sh

```
#!/usr/bin/env bash
sudo mkdir -p /etc/kubernetes/pki/etcd
sudo mv -v ca.crt /etc/kubernetes/pki/
sudo mv -v ca.key /etc/kubernetes/pki/
sudo mv -v sa.pub /etc/kubernetes/pki/
sudo mv -v sa.key /etc/kubernetes/pki/
sudo mv -v front-proxy-ca.crt /etc/kubernetes/pki/
sudo mv -v front-proxy-ca.key /etc/kubernetes/pki/
sudo mv -v etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
sudo mv -v etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
sudo mv -v admin.conf /etc/kubernetes/admin.conf
```

Выполнить инициализирующие команды (файл kubelet-setup-m2.sh):

```
export CONFIG="ha-cluster-m2.yaml"
export CP0_IP=192.168.66.206
export CP0_HOSTNAME=k8sm1
export CP1_IP=192.168.66.207
export CP1_HOSTNAME=k8sm2
export KUBECONFIG="/etc/kubernetes/admin.conf"

sudo kubeadm alpha phase certs all --config $CONFIG
sudo kubeadm alpha phase kubelet config write-to-disk --config $CONFIG
sudo kubeadm alpha phase kubelet write-env-file --config $CONFIG
sudo kubeadm alpha phase kubeconfig kubelet --config $CONFIG
sudo systemctl start kubelet


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
sudo kubeadm alpha phase etcd local --config $CONFIG
```

После этого кластер высыпется в нестабильное состояние на несколько минут, после чего восстановит целостность.

Проверка состояния кластера **etcd**:

```
$ sudo docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes
k8s.gcr.io/etcd-amd64:3.2.18 etcdctl --cert-file
/etc/kubernetes/pki/etcd/peer.crt --key-file
/etc/kubernetes/pki/etcd/peer.key--ca-file /etc/kubernetes/pki/etcd/ca.crt
--endpoints https://192.168.66.206:2379 cluster-health
member a48c168e19699a32 is healthy: got healthy result from
https://192.168.66.207:2379
member ab3a311853d08ce4 is healthy: got healthy result from
https://192.168.66.206:2379
cluster is healthy
$ sudo docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes
k8s.gcr.io/etcd-amd64:3.2.18 etcdctl --cert-file
/etc/kubernetes/pki/etcd/peer.crt --key-file
/etc/kubernetes/pki/etcd/peer.key--ca-file /etc/kubernetes/pki/etcd/ca.crt
--endpoints https://192.168.66.206:2379 member list
a48c168e19699a32: name=k8sm2 peerURLs=https://192.168.66.207:2380
clientURLs=https://192.168.66.207:2379 isLeader=false
ab3a311853d08ce4: name=k8sm1 peerURLs=https://192.168.66.206:2380
clientURLs=https://192.168.66.206:2379 isLeader=true
```

Завершить инициализацию контрольной ноды (файл kubelet-setup-m2.sh):

```
sudo kubeadm alpha phase kubeconfig all --config $CONFIG
sudo kubeadm alpha phase controlplane all --config $CONFIG
sudo kubeadm alpha phase mark-master --config $CONFIG
```

Проверка состояния кластера:

```
$ kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
k8sm1     Ready     master    41m       v1.11.2
k8sm2     Ready     master    17m       v1.11.2
$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                              READY     STATUS    RESTARTS
AGE       IP               NODE      NOMINATED NODE
kube-system   coredns-78fcdf6894-52gxd          1/1       Running   0
51m       10.32.0.2        k8sm1     <none>
kube-system   coredns-78fcdf6894-ghh6z          1/1       Running   0
51m       10.32.0.3        k8sm1     <none>
kube-system   etcd-k8sm1                        1/1       Running   1
50m       192.168.66.206   k8sm1     <none>
kube-system   etcd-k8sm2                        1/1       Running   0
24m       192.168.66.207   k8sm2     <none>
kube-system   kube-apiserver-k8sm1              1/1       Running   0
50m       192.168.66.206   k8sm1     <none>
kube-system   kube-apiserver-k8sm2              1/1       Running   0
12m       192.168.66.207   k8sm2     <none>
kube-system   kube-controller-manager-k8sm1     1/1       Running   1
50m       192.168.66.206   k8sm1     <none>
kube-system   kube-controller-manager-k8sm2     1/1       Running   0
12m       192.168.66.207   k8sm2     <none>
kube-system   kube-proxy-gwwxb                  1/1       Running   0
51m       192.168.66.206   k8sm1     <none>
kube-system   kube-proxy-mfjmk                  1/1       Running   0
27m       192.168.66.207   k8sm2     <none>
kube-system   kube-scheduler-k8sm1              1/1       Running   1
50m       192.168.66.206   k8sm1     <none>
kube-system   kube-scheduler-k8sm2              1/1       Running   0
12m       192.168.66.207   k8sm2     <none>
kube-system   weave-net-6rnqq                   2/2       Running   3
27m       192.168.66.207   k8sm2     <none>
kube-system   weave-net-dt9sg                   2/2       Running   0
45m       192.168.66.206   k8sm1     <none>
```

Настройка Control Pane 2

Создать файл **ha-cluster-m3.yaml:**

```
apiVersion: kubeadm.k8s.io/v1alpha2
kind: MasterConfiguration
kubernetesVersion: v1.11.2
apiServerCertSANs:
- "kubernetes.vsk.ru"
api:
    controlPlaneEndpoint: "kubernetes.vsk.ru:6443"
etcd:
  local:
    extraArgs:
      listen-client-urls:
"https://127.0.0.1:2379,https://192.168.66.131:2379"
      advertise-client-urls: "https://192.168.66.131:2379"
      listen-peer-urls: "https://192.168.66.131:2380"
      initial-advertise-peer-urls: "https://192.168.66.131:2380"
      initial-cluster:
"k8sm1=https://192.168.66.206:2380,k8sm2=https://192.168.66.207:2380,k8sm3
=https://192.168.66.131:2380"
      initial-cluster-state: existing
    serverCertSANs:
      - k8sm3.vsk.ru
      - 192.168.66.131
    peerCertSANs:
      - k8sm3.vsk.ru
      - 192.168.66.131
```

Скопировать сертификаты, запустив файл copy-certs.sh

```
#!/usr/bin/env bash
sudo mkdir -p /etc/kubernetes/pki/etcd
sudo mv -v ca.crt /etc/kubernetes/pki/
sudo mv -v ca.key /etc/kubernetes/pki/
sudo mv -v sa.pub /etc/kubernetes/pki/
sudo mv -v sa.key /etc/kubernetes/pki/
sudo mv -v front-proxy-ca.crt /etc/kubernetes/pki/
sudo mv -v front-proxy-ca.key /etc/kubernetes/pki/
sudo mv -v etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
sudo mv -v etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
sudo mv -v admin.conf /etc/kubernetes/admin.conf
```

Выполнить инициализирующие команды (файл kubelet-setup-m3.sh):

```
export CONFIG="ha-cluster-m3.yaml"
export CP0_IP=192.168.66.206
export CP0_HOSTNAME=k8sm1
export CP1_IP=192.168.66.131
export CP1_HOSTNAME=k8sm3
export KUBECONFIG="/etc/kubernetes/admin.conf"

sudo kubeadm alpha phase certs all --config $CONFIG
sudo kubeadm alpha phase kubelet config write-to-disk --config $CONFIG
sudo kubeadm alpha phase kubelet write-env-file --config $CONFIG
sudo kubeadm alpha phase kubeconfig kubelet --config $CONFIG
sudo systemctl start kubelet

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
sudo kubeadm alpha phase etcd local --config $CONFIG
```

После этого кластер высыпется в нестабильное состояние на несколько минут, после чего восстановит целостность.

Проверка состояния кластера **etcd**:

```
$ sudo docker run --rm -it --net host -v /etc/kubernetes:/etc/kubernetes
k8s.gcr.io/etcd-amd64:3.2.18 etcdctl --cert-file
/etc/kubernetes/pki/etcd/peer.crt --key-file
/etc/kubernetes/pki/etcd/peer.key--ca-file /etc/kubernetes/pki/etcd/ca.crt
--endpoints https://192.168.66.206:2379 member list
30018431eee15c74: name=k8sm3 peerURLs=https://192.168.66.131:2380
clientURLs=https://192.168.66.131:2379 isLeader=false
a48c168e19699a32: name=k8sm2 peerURLs=https://192.168.66.207:2380
clientURLs=https://192.168.66.207:2379 isLeader=false
ab3a311853d08ce4: name=k8sm1 peerURLs=https://192.168.66.206:2380
clientURLs=https://192.168.66.206:2379 isLeader=true
```

Завершить инициализацию контрольной ноды (файл kubelet-setup-m3.sh):

```
sudo kubeadm alpha phase kubeconfig all --config $CONFIG
sudo kubeadm alpha phase controlplane all --config $CONFIG
sudo kubeadm alpha phase mark-master --config $CONFIG
```

Проверка состояния кластера:

```
$ kubectl get nodes
NAME      STATUS    ROLES     AGE       VERSION
k8sm1     Ready     master    1h        v1.11.2
k8sm2     Ready     master    41m       v1.11.2
k8sm3     Ready     master    6m        v1.11.2
$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                              READY     STATUS    RESTARTS
AGE       IP               NODE      NOMINATED NODE
kube-system   coredns-78fcdf6894-52gxd          1/1       Running   0
1h        10.32.0.2        k8sm1     <none>
kube-system   coredns-78fcdf6894-ghh6z          1/1       Running   0
1h        10.32.0.3        k8sm1     <none>
kube-system   etcd-k8sm1                        1/1       Running   1
1h        192.168.66.206   k8sm1     <none>
kube-system   etcd-k8sm2                        1/1       Running   0
38m       192.168.66.207   k8sm2     <none>
kube-system   etcd-k8sm3                        1/1       Running   0
4m        192.168.66.131   k8sm3     <none>
kube-system   kube-apiserver-k8sm1              1/1       Running   0
1h        192.168.66.206   k8sm1     <none>
kube-system   kube-apiserver-k8sm2              1/1       Running   0
26m       192.168.66.207   k8sm2     <none>
kube-system   kube-apiserver-k8sm3              1/1       Running   0
2m        192.168.66.131   k8sm3     <none>
kube-system   kube-controller-manager-k8sm1     1/1       Running   1
1h        192.168.66.206   k8sm1     <none>
kube-system   kube-controller-manager-k8sm2     1/1       Running   0
26m       192.168.66.207   k8sm2     <none>
kube-system   kube-controller-manager-k8sm3     1/1       Running   0
2m        192.168.66.131   k8sm3     <none>
kube-system   kube-proxy-cp89v                  1/1       Running   0
6m        192.168.66.131   k8sm3     <none>
kube-system   kube-proxy-gwwxb                  1/1       Running   0
1h        192.168.66.206   k8sm1     <none>
kube-system   kube-proxy-mfjmk                  1/1       Running   0
41m       192.168.66.207   k8sm2     <none>
kube-system   kube-scheduler-k8sm1              1/1       Running   1
1h        192.168.66.206   k8sm1     <none>
kube-system   kube-scheduler-k8sm2              1/1       Running   0
26m       192.168.66.207   k8sm2     <none>
kube-system   kube-scheduler-k8sm3              1/1       Running   0
2m        192.168.66.131   k8sm3     <none>
kube-system   weave-net-6rnqq                   2/2       Running   3
41m       192.168.66.207   k8sm2     <none>
kube-system   weave-net-dt9sg                   2/2       Running   0
59m       192.168.66.206   k8sm1     <none>
kube-system   weave-net-rdbvs                   2/2       Running   0
6m        192.168.66.131   k8sm3     <none>
```