Kubernetes HA Cluster Build

Kubernetes HA Cluster Build

ETCD Cluster

Etcd install

Docker

Docker 1.12.x ~ 1.13.1 Kubenetes Cluster

Docker install

Kubernetes TLS

kube-apiserver HTTP Kubernetes TLS TLS

CA

ca-config.json

```
{
  "signing": {
    "default": {
      "expiry": "87600h"
    },
    "profiles": {
      "frognew": {
        "usages": [
            "signing",
            "key encipherment",
            "server auth",
            "client auth"
        ],
        "expiry": "87600h"
      }
    }
  }
}
```

CA

ca-csr.json

```
{
  "CN": "kubernetes",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "k8s",
      "OU": "cloudnative"
    }
  ]
}
```

- `CN` Common Namekube-apiserver
- `O` Organizationkube-apiserver

cfssCA

```
$ cfssl gencert -initca ca-csr.json | cfssljson -bare ca
```

kube-apiserver

kube-apiserver

apiserver-csr.json

```
{
    "CN": "kubernetes",
    "hosts": [
      "127.0.0.1",
      "192.166.1.12",
      "192.166.1.2",
      "192.166.1.13",
      "10.96.0.1",
      "kubernetes",
      "kubernetes.default",
      "kubernetes.default.svc",
      "kubernetes.default.svc.cluster",
      "kubernetes.default.svc.cluster.local"
    ],
    "key": {
        "algo": "rsa",
        "size": 2048
    },
    "names": [
        {
            "C": "CN",
            "ST": "BeiJing",
            "L": "BeiJing",
            "O": "k8s",
            "OU": "cloudnative"
        }
    ]
}
```

hosts IP Kubernetes Master IP hostname kube-apiserver IP `10.96.0.1` (kube-apiserver-`service-cluster-ip-range=10.96.0.0/12` IP) VIP

kube-apiserver

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=frognew apiserver-csr.json |
cfssljson -bare apiserver
$ ls apiserver*
apiserver.csr  apiserver-csr.json  apiserver-key.pem  apiserver.pem
```

kubernetes-admin

admin

admin-csr.json

```
{
  "CN": "kubernetes-admin",
  "hosts": [
            "192.166.1.12",
      "192.166.1.2",
      "192.166.1.13"
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "system:masters",
      "OU": "cloudnative"
    }
  ]
}
```

kube-apiserver `CN` `kubernetes-admin` `O` `system:master` kube-apiserver RBAC ClusterRoleBindings cluster-admin system:masters ClusterRole cluster-admin cluster-admin kube-apiserver `kubernetes-admin`

`kubernetes-admin`

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=frognew admin-csr.json |
cfssljson -bare admin
$ ls admin*
admin.csr  admin-csr.json  admin-key.pem  admin.pem
```

kube-controller-manager

kube-controller-manager ApiServer controller-manager

controller-manager-csr.json

```
{
  "CN": "system:kube-controller-manager",
  "hosts": [
            "192.168.1.12",
      "192.168.1.2",
      "192.168.1.13"
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "system:kube-controller-manager",
      "OU": "cloudnative"
    }
  ]
}
```

kube-apiserver `CN` `system:kube-controller-manager` kube-apiserver RBAC ClusterRoleBindings `system:kube-controller-manager` `system:kube-controller-manager` ClusterRole `system:kube-controller-manager`

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=frognew controller-manager-csr.
json | cfssljson -bare controller-manager
$ ls controller-manager*
controller-manager.csr        controller-manager-csr.json  controller-manager-key.pem  controller-manager.pem
```

kube-scheduler

kube-schedulerApiServer scheduler

scheduler-csr.json

```
{
  "CN": "system:kube-scheduler",
  "hosts": [
            "192.166.1.12",
      "192.166.1.2",
      "192.166.1.13"
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "system:kube-scheduler",
      "OU": "cloudnative"
    }
  ]
}
```

kube-scheduler `CN` `system:kube-scheduler` kube-apiserver RBAC ClusterRoleBindings `system:kube-scheduler` `system:kube-scheduler` ClusterRole `system:kube-scheduler`

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=frognew scheduler-csr.json |
cfssljson -bare scheduler
$ ls scheduler*
scheduler.csr  scheduler-csr.json  scheduler-key.pem  scheduler.pem
```

Kubernetes Master

Master Node1, Node2, Node3 kube-apiserver,kube-controller-manager,kube-scheduler kube-apiserver 3 kube-apiserve r kube-controller-manager kube-scheduler 31

ca.pem, apiserver-key.pem, apiserver.pem, admin.pem, admin-key.pem, controller-manager.pem, controller-manager-key.pem, scheduler-key.pem, scheduler.pem /etc/kubernetes/pki

Node

```
$ mkdir -p /etc/kubernetes/pki
$ scp ./{ca-key.pem,ca.pem,apiserver-key.pem,apiserver.pem,admin.pem,admin-key.pem,controller-manager.pem,
controller-manager-key.pem,scheduler-key.pem,scheduler.pem} node02:/etc/kubernetes/pki
```

Kubernetes Kubernetes Download Packet

Kubernetes kubernetes/server/bin kube-apiserver,kube-controller-manager,kube-scheduler,kubectl,kube-proxy,kubelet /usr/local/bin

```
$ tar zxf kubernetes-server-linux-amd64.tar.gz && cp kubernetes/server/bin/{kube-apiserver,kube-controller-
manager,kube-scheduler,kubectl,kube-proxy,kubelet} /usr/local/bin/
```

kube-apiserver

kube-apiserver systemd unit , INTERNAL_IP

```
$ mkdir -p /var/log/kubernetes
$ export INTERNAL_IP=192.166.1.12
$ cat > /usr/lib/systemd/system/kube-apiserver.service <<EOF
[Unit]
Description=kube-apiserver
After=network.target
After=etcd.service

[Service]
EnvironmentFile=-/etc/kubernetes/apiserver
ExecStart=/usr/local/bin/kube-apiserver \
            --logtostderr=true \
            --v=0 \
            --advertise-address=${INTERNAL_IP} \
            --bind-address=${INTERNAL_IP} \
            --secure-port=6443 \
            --insecure-port=0 \
            --allow-privileged=true \
            --etcd-servers=https://192.166.1.12:2379,https://192.166.1.2:2379,https://192.166.1.13:2379 \
            --etcd-cafile=/etc/etcd/ssl/ca.pem \
            --etcd-certfile=/etc/etcd/ssl/etcd.pem \
            --etcd-keyfile=/etc/etcd/ssl/etcd-key.pem \
            --storage-backend=etcd3 \
        --service-node-port-range=79-39999 \
            --service-cluster-ip-range=10.96.0.0/12 \
            --tls-cert-file=/etc/kubernetes/pki/apiserver.pem \
            --tls-private-key-file=/etc/kubernetes/pki/apiserver-key.pem \
            --client-ca-file=/etc/kubernetes/pki/ca.pem \
            --service-account-key-file=/etc/kubernetes/pki/ca-key.pem \
            --experimental-bootstrap-token-auth=true \
            --apiserver-count=3 \
            --enable-swagger-ui=true \
            --admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,PersistentVolumeLabel,
DefaultStorageClass,ResourceQuota,DefaultTolerationSeconds \
            --authorization-mode=RBAC \
            --audit-log-maxage=30 \
            --audit-log-maxbackup=3 \
            --audit-log-maxsize=100 \
            --audit-log-path=/var/log/kubernetes/audit.log
Restart=on-failure
Type=notify
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF
```

/usr/lib/systemd/system/kube-apiserver.service

```
[Unit]
Description=kube-apiserver
After=network.target
After=etcd.service

[Service]
EnvironmentFile=-/etc/kubernetes/apiserver
ExecStart=/usr/local/bin/kube-apiserver     --logtostderr=true     --v=0     --advertise-address=192.
166.1.12     --bind-address=192.166.1.12     --secure-port=6443     --insecure-port=0     --allow-
privileged=true     --etcd-servers=https://192.166.1.12:2379,https://192.166.1.2:2379,https://192.166.1.13:
2379     --etcd-cafile=/etc/etcd/ssl/ca.pem     --etcd-certfile=/etc/etcd/ssl/etcd.pem     --etcd-keyfile=
/etc/etcd/ssl/etcd-key.pem     --storage-backend=etcd3     --service-cluster-ip-range=10.96.0.0/12     --tls-
cert-file=/etc/kubernetes/pki/apiserver.pem     --tls-private-key-file=/etc/kubernetes/pki/apiserver-key.
pem     --client-ca-file=/etc/kubernetes/pki/ca.pem     --service-account-key-file=/etc/kubernetes/pki/ca-
key.pem     --experimental-bootstrap-token-auth=true     --apiserver-count=3     --enable-swagger-
ui=true     --admission-control=NamespaceLifecycle,LimitRanger,ServiceAccount,PersistentVolumeLabel,
DefaultStorageClass,ResourceQuota,DefaultTolerationSeconds     --authorization-mode=RBAC     --audit-log-
maxage=30     --audit-log-maxbackup=3     --audit-log-maxsize=100     --audit-log-path=/var/log/kubernetes
/audit.log
Restart=on-failure
Type=notify
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
```

- `--insecure-port:` http
- `--secure-port:` https kube-schedulerkube-controller-managerkubeletkube-proxykubectl ApiServer ()
- `--authorization-mode=RBAC:` RBAC kube-schedulerkube-controller-managerkubeletkube-proxykubectl kubeconfig UserGroup RBAC
- `--admission-control: NamespaceLifecycle,LimitRanger,ServiceAccount,PersistentVolumeLabel, DefaultStorageClass,ResourceQuota,DefaultTolerationSeconds`
- `--service-cluster-ip-range:` Service Cluster IP Kubernetes Service IP

kube-apiserve

```
$ systemctl daemon-reload
$ systemctl enable kube-apiserver
$ systemctl start kube-apiserver
$ systemctl status kube-apiserver
```

kubectl apiserver

kube-apiserver kubernetes-admin ApiServer

```
$ kubectl --server=https://192.166.1.12:6443 \
--certificate-authority=/etc/kubernetes/pki/ca.pem  \
--client-certificate=/etc/kubernetes/pki/admin.pem \
--client-key=/etc/kubernetes/pki/admin-key.pem \
get componentstatuses
NAME                 STATUS
MESSAGE                                                                                     ERROR
scheduler            Unhealthy   Get http://127.0.0.1:10251/healthz: dial tcp 127.0.0.1:10251: getsockopt:
connection refused
controller-manager   Unhealthy   Get http://127.0.0.1:10252/healthz: dial tcp 127.0.0.1:10252: getsockopt:
connection refused
etcd-1               Healthy     {"health": "true"}
etcd-2               Healthy     {"health": "true"}
etcd-0               Healthy     {"health": "true"}
```

kubectl Kubernetes kube-scheduler controller-manager

kubectl ApiServer kubernetes-admin kubeconfig admin.conf

```
$ cd /etc/kubernetes
$ export KUBE_APISERVER="https://192.166.1.12:6443"
```

cluster

```
$ kubectl config set-cluster kubernetes \
  --certificate-authority=/etc/kubernetes/pki/ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=admin.conf
```

credentials

```
$ kubectl config set-credentials kubernetes-admin \
  --client-certificate=/etc/kubernetes/pki/admin.pem \
  --embed-certs=true \
  --client-key=/etc/kubernetes/pki/admin-key.pem \
  --kubeconfig=admin.conf
```

context

```
$ kubectl config set-context kubernetes-admin@kubernetes \
  --cluster=kubernetes \
  --user=kubernetes-admin \
  --kubeconfig=admin.conf
```

default context

```
$ kubectl config use-context kubernetes-admin@kubernetes --kubeconfig=admin.conf
```

kubectl apiserver admin.conf $HOME/.kube config

```
$ cp /etc/kubernetes/admin.conf ~/.kube/config
```

kubectl apiserver CA

```
$ kubectl get cs
NAME                 STATUS
MESSAGE                                                                                     ERROR
scheduler            Unhealthy   Get http://127.0.0.1:10251/healthz: dial tcp 127.0.0.1:10251: getsockopt:
connection refused
controller-manager   Unhealthy   Get http://127.0.0.1:10252/healthz: dial tcp 127.0.0.1:10252: getsockopt:
connection refused
etcd-2               Healthy     {"health": "true"}
etcd-0               Healthy     {"health": "true"}
etcd-1               Healthy     {"health": "true"}
```


kubernetes-admin admin.conf /etc/kubernetes/pki admin.pem admin-key.pem

```
$ cd /etc/kubernetes/pki
$ rm -f admin.pem admin-key.pem
```

Kube-APIServer

api 6443 Haproxy Corosync + Pacemaker Haproxy install

Kube-controller-manager

controller-manager.pem controller-manage-key.pem controller-manager kubeconfig

controller-manager.conf

```
$ cd /etc/kubernetes
$ export KUBE_APISERVER="https://192.166.1.222:6443"
```

cluster

```
$ kubectl config set-cluster kubernetes \
  --certificate-authority=/etc/kubernetes/pki/ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=controller-manager.conf
```

credentials

```
$ kubectl config set-credentials system:kube-controller-manager \
  --client-certificate=/etc/kubernetes/pki/controller-manager.pem \
  --embed-certs=true \
  --client-key=/etc/kubernetes/pki/controller-manager-key.pem \
  --kubeconfig=controller-manager.conf
```

context

```
$ kubectl config set-context system:kube-controller-manager@kubernetes \
  --cluster=kubernetes \
  --user=system:kube-controller-manager \
  --kubeconfig=controller-manager.conf
```

default context

```
$ kubectl config use-context system:kube-controller-manager@kubernetes --kubeconfig=controller-manager.conf
```

controller-manager.conf Master /etc/kubernetes

```
$ scp controller-manager.conf node02:/etc/kubernetes
...
```

kube-controller-manager systemd unit

```
$ export KUBE_APISERVER="https://192.166.1.222:6443"
$ cat > /usr/lib/systemd/system/kube-controller-manager.service <<EOF
[Unit]
Description=kube-controller-manager
After=network.target
After=kube-apiserver.service

[Service]
EnvironmentFile=-/etc/kubernetes/controller-manager
ExecStart=/usr/local/bin/kube-controller-manager \
            --logtostderr=true \
            --v=0 \
            --master=${KUBE_APISERVER} \
            --kubeconfig=/etc/kubernetes/controller-manager.conf \
            --cluster-name=kubernetes \
            --cluster-signing-cert-file=/etc/kubernetes/pki/ca.pem \
            --cluster-signing-key-file=/etc/kubernetes/pki/ca-key.pem \
            --service-account-private-key-file=/etc/kubernetes/pki/ca-key.pem \
            --root-ca-file=/etc/kubernetes/pki/ca.pem \
            --insecure-experimental-approve-all-kubelet-csrs-for-group=system:bootstrappers \
            --use-service-account-credentials=true \
            --service-cluster-ip-range=10.96.0.0/12 \
            --cluster-cidr=10.244.0.0/16 \
            --allocate-node-cidrs=true \
            --leader-elect=true \
            --controllers=*,bootstrapsigner,tokencleaner
Restart=on-failure
Type=simple
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF
```

/usr/lib/systemd/system/kube-controller-manager.service

```
[Unit]
Description=kube-controller-manager
After=network.target
After=kube-apiserver.service

[Service]
EnvironmentFile=-/etc/kubernetes/controller-manager
ExecStart=/usr/local/bin/kube-controller-manager     --logtostderr=true     --v=0     --master=https://192.
166.1.222:6443     --kubeconfig=/etc/kubernetes/controller-manager.conf     --cluster-name=kubernetes     --
cluster-signing-cert-file=/etc/kubernetes/pki/ca.pem     --cluster-signing-key-file=/etc/kubernetes/pki/ca-
key.pem     --service-account-private-key-file=/etc/kubernetes/pki/ca-key.pem     --root-ca-file=/etc
/kubernetes/pki/ca.pem     --insecure-experimental-approve-all-kubelet-csrs-for-group=system:
bootstrappers     --use-service-account-credentials=true     --service-cluster-ip-range=10.96.0.0/12     --
cluster-cidr=10.244.0.0/16     --allocate-node-cidrs=true     --leader-elect=true     --controllers=*,
bootstrapsigner,tokencleaner
Restart=on-failure
Type=simple
LimitNOFILE=65536

[Install]
```

- `--service-cluster-ip-range` kube-api-server

:

```
$ systemctl daemon-reload
$ systemctl enable kube-controller-manager
$ systemctl start kube-controller-manager
$ systemctl status kube-controller-manager
```

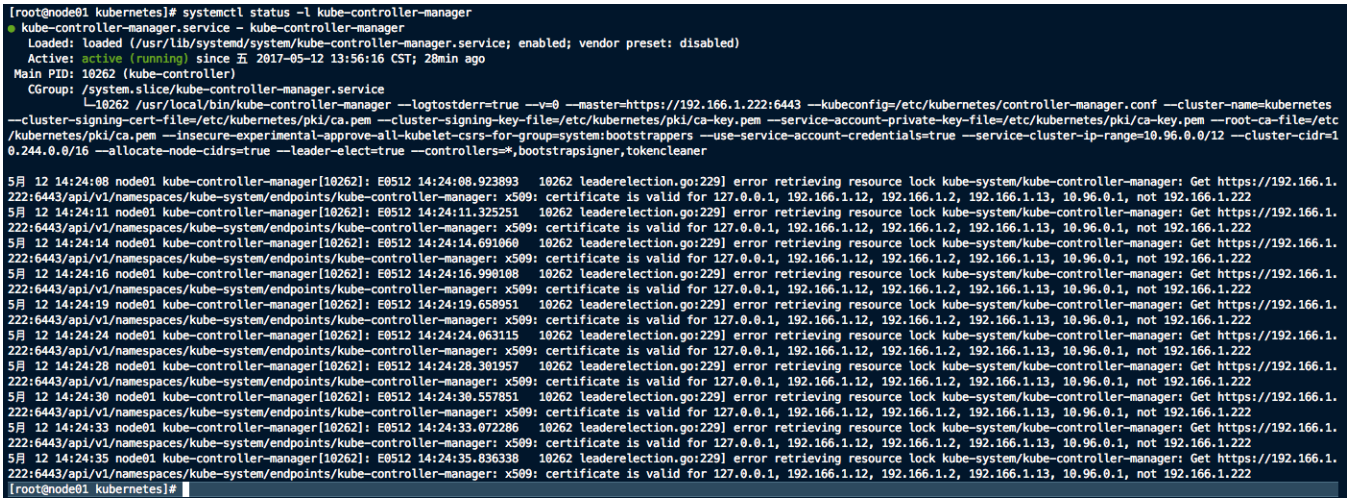
node1,node2,node3

```
$ kubectl get cs
NAME                 STATUS
MESSAGE                                                                                     ERROR
scheduler            Unhealthy   Get http://127.0.0.1:10251/healthz: dial tcp 127.0.0.1:10251: getsockopt:
connection refused
controller-manager   Healthy     ok
etcd-1               Healthy     {"health": "true"}
etcd-0               Healthy     {"health": "true"}
etcd-2               Healthy     {"health": "true"}
```

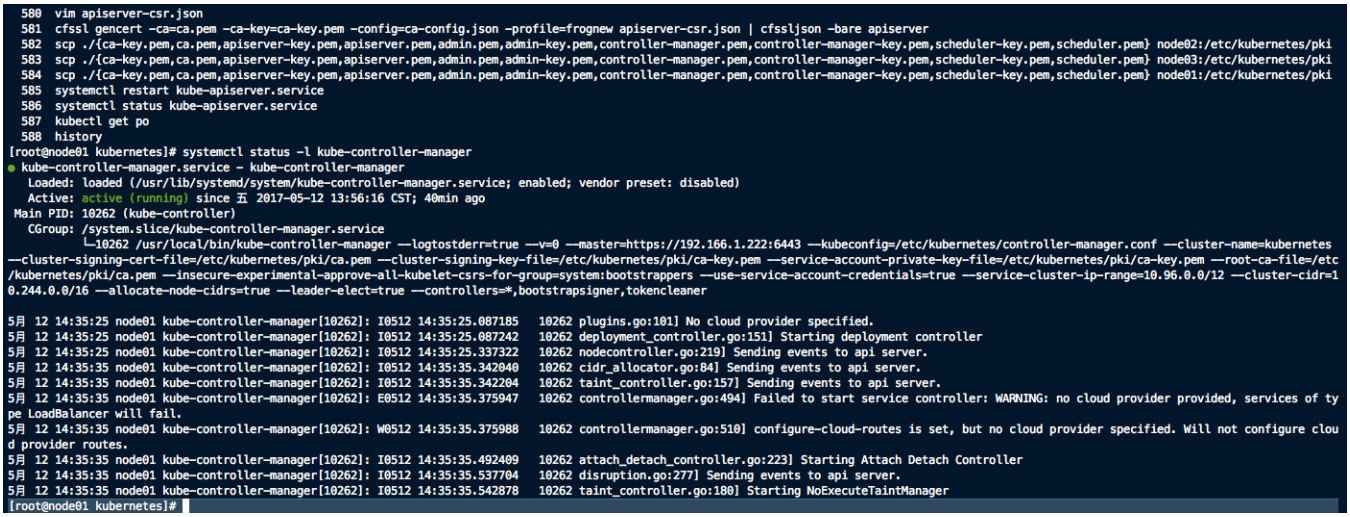
Master kube-controller-manager leader

```
$ systemctl status -l kube-controller-manager
```

node01



apiserver-csr.json IP VIP cfssl master /etc/kubernetes/pki kube-apiserver.service



/etc/kubernetes log

```
[root@node01 kubernetes]# systemctl status kube-controller-manager
● kube-controller-manager.service - kube-controller-manager
   Loaded: loaded (/usr/lib/systemd/system/kube-controller-manager.service; enabled; vendor preset: disabled)
   Active: active (running) since 五 2017-05-12 14:46:27 CST; 10s ago
 Main PID: 21080 (kube-controller)
   CGroup: /system.slice/kube-controller-manager.service
           └─21080 /usr/local/bin/kube-controller-manager --logtostderr=true --v=0 --master=https://192.166.1.222:6443 --kubeconfig=/etc/kubernetes/controller-manager.conf --cluster-name=kubernet...

5月 12 14:46:29 node01 kube-controller-manager[21080]: E0512 14:46:29.797575   21080 util.go:45] Metric for serviceaccount_controller already registered
5月 12 14:46:29 node01 kube-controller-manager[21080]: I0512 14:46:29.797639   21080 controllermanager.go:437] Started "serviceaccount"
5月 12 14:46:29 node01 kube-controller-manager[21080]: I0512 14:46:29.797696   21080 serviceaccounts_controller.go:122] Starting ServiceAccount controller
5月 12 14:46:29 node01 kube-controller-manager[21080]: I0512 14:46:29.947781   21080 controllermanager.go:437] Started "daemonset"
5月 12 14:46:29 node01 kube-controller-manager[21080]: I0512 14:46:29.947869   21080 daemoncontroller.go:199] Starting Daemon Sets controller manager
5月 12 14:46:30 node01 kube-controller-manager[21080]: I0512 14:46:30.098269   21080 controllermanager.go:437] Started "job"
5月 12 14:46:30 node01 kube-controller-manager[21080]: I0512 14:46:30.247487   21080 controllermanager.go:437] Started "deployment"
5月 12 14:46:30 node01 kube-controller-manager[21080]: I0512 14:46:30.247517   21080 plugins.go:101] No cloud provider specified.
5月 12 14:46:30 node01 kube-controller-manager[21080]: I0512 14:46:30.247576   21080 deployment_controller.go:151] Starting deployment controller
5月 12 14:46:30 node01 kube-controller-manager[21080]: I0512 14:46:30.397321   21080 nodecontroller.go:219] Sending events to api server.
[root@node01 kubernetes]#
```

```
[root@node02 ~]# systemctl status -l kube-controller-manager
● kube-controller-manager.service - kube-controller-manager
   Loaded: loaded (/usr/lib/systemd/system/kube-controller-manager.service; enabled; vendor preset: disabled)
   Active: failed (Result: start-limit) since Fri 2017-05-12 07:56:20 EET; 16min ago
  Process: 9923 ExecStart=/usr/local/bin/kube-controller-manager --logtostderr=true --v=0 --master=https://192.166.1.222:6443 --kubeconfig=/etc/kubernetes/controller-manager.conf --cluster-name=kube
rnetes --cluster-signing-cert-file=/etc/kubernetes/pki/ca.pem --cluster-signing-key-file=/etc/kubernetes/pki/ca-key.pem --service-account-private-key-file=/etc/kubernetes/pki/ca-key.pem --root-ca-fi
le=/etc/kubernetes/pki/ca.pem --insecure-experimental-approve-all-kubelet-csrs-for-group=system:bootstrappers --use-service-account-credentials=true --service-cluster-ip-range=10.96.0.0/12 --cluster
-cidr=10.244.0.0/16 --allocate-node-cidrs=true --leader-elect=true --controllers=*,bootstrapsigner,tokencleaner (code=exited, status=1/FAILURE)
 Main PID: 9923 (code=exited, status=1/FAILURE)

May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service: main process exited, code=exited, status=1/FAILURE
May 12 07:56:20 node02 systemd[1]: Unit kube-controller-manager.service entered failed state.
May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service failed.
May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service holdoff time over, scheduling restart.
May 12 07:56:20 node02 systemd[1]: start request repeated too quickly for kube-controller-manager.service
May 12 07:56:20 node02 systemd[1]: Failed to start kube-controller-manager.
May 12 07:56:20 node02 systemd[1]: Unit kube-controller-manager.service entered failed state.
May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service failed.
[root@node02 ~]# systemctl restart haproxy.service
[root@node02 ~]# systemctl status -l kube-controller-manager
● kube-controller-manager.service - kube-controller-manager
   Loaded: loaded (/usr/lib/systemd/system/kube-controller-manager.service; enabled; vendor preset: disabled)
   Active: failed (Result: start-limit) since Fri 2017-05-12 07:56:20 EET; 29min ago
  Process: 9923 ExecStart=/usr/local/bin/kube-controller-manager --logtostderr=true --v=0 --master=https://192.166.1.222:6443 --kubeconfig=/etc/kubernetes/controller-manager.conf --cluster-name=kube
rnetes --cluster-signing-cert-file=/etc/kubernetes/pki/ca.pem --cluster-signing-key-file=/etc/kubernetes/pki/ca-key.pem --service-account-private-key-file=/etc/kubernetes/pki/ca-key.pem --root-ca-fi
le=/etc/kubernetes/pki/ca.pem --insecure-experimental-approve-all-kubelet-csrs-for-group=system:bootstrappers --use-service-account-credentials=true --service-cluster-ip-range=10.96.0.0/12 --cluster
-cidr=10.244.0.0/16 --allocate-node-cidrs=true --leader-elect=true --controllers=*,bootstrapsigner,tokencleaner (code=exited, status=1/FAILURE)
 Main PID: 9923 (code=exited, status=1/FAILURE)

May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service: main process exited, code=exited, status=1/FAILURE
May 12 07:56:20 node02 systemd[1]: Unit kube-controller-manager.service entered failed state.
May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service failed.
May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service holdoff time over, scheduling restart.
May 12 07:56:20 node02 systemd[1]: start request repeated too quickly for kube-controller-manager.service
May 12 07:56:20 node02 systemd[1]: Failed to start kube-controller-manager.
May 12 07:56:20 node02 systemd[1]: Unit kube-controller-manager.service entered failed state.
May 12 07:56:20 node02 systemd[1]: kube-controller-manager.service failed.
[root@node02 ~]#
```

kube-scheduler

kube-schedulerkubeconfig

scheduler.conf

```
$ cd /etc/kubernetes
$ export KUBE_APISERVER="https://192.166.1.222:6443"
```

cluster

```
$ kubectl config set-cluster kubernetes \
  --certificate-authority=/etc/kubernetes/pki/ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=scheduler.conf
```

credentials

```
$ kubectl config set-credentials system:kube-scheduler \
  --client-certificate=/etc/kubernetes/pki/scheduler.pem \
  --embed-certs=true \
  --client-key=/etc/kubernetes/pki/scheduler-key.pem \
  --kubeconfig=scheduler.conf
```

context

```
$ kubectl config set-context system:kube-scheduler@kubernetes \
  --cluster=kubernetes \
  --user=system:kube-scheduler \
  --kubeconfig=scheduler.conf
```

default context

```
$ kubectl config use-context system:kube-scheduler@kubernetes --kubeconfig=scheduler.conf
```

scheduler.conf Master /etc/kubernetes

```
$ scp scheduler.conf node02:/etc/kubernetes/
...
```

kube-scheduler systemd unit

```
$ export KUBE_APISERVER="https://192.166.1.222:6443"
$ cat > /usr/lib/systemd/system/kube-scheduler.service <<EOF
[Unit]
Description=kube-scheduler
After=network.target
After=kube-apiserver.service

[Service]
EnvironmentFile=-/etc/kubernetes/scheduler
ExecStart=/usr/local/bin/kube-scheduler \
            --logtostderr=true \
            --v=0 \
            --master=${KUBE_APISERVER} \
            --kubeconfig=/etc/kubernetes/scheduler.conf \
            --leader-elect=true
Restart=on-failure
Type=simple
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
EOF
```

/usr/lib/systemd/system/kube-scheduler.service

```
[Unit]
Description=kube-scheduler
After=network.target
After=kube-apiserver.service

[Service]
EnvironmentFile=-/etc/kubernetes/scheduler
ExecStart=/usr/local/bin/kube-scheduler     --logtostderr=true     --v=0     --master=https://192.166.1.222:
6443     --kubeconfig=/etc/kubernetes/scheduler.conf     --leader-elect=true
Restart=on-failure
Type=simple
LimitNOFILE=65536

[Install]
WantedBy=multi-user.target
```

```
$ systemctl daemon-reload
$ systemctl enable kube-scheduler
$ systemctl start kube-scheduler
$ systemctl status kube-scheduler
```

Master kube-schedule r leader

Kubernetes Master

```
$ kubectl get cs
NAME                 STATUS    MESSAGE             ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-2               Healthy   {"health": "true"}
etcd-1               Healthy   {"health": "true"}
etcd-0               Healthy   {"health": "true"}
```

Kubernetes Node

KubernetesNode

- Docker
- kubelet
- kube-proxy

node1

CNI

```
$ wget https://github.com/containernetworking/cni/releases/download/v0.5.2/cni-amd64-v0.5.2.tgz
$ mkdir -p /opt/cni/bin
$ tar -zxvf cni-amd64-v0.5.2.tgz -C /opt/cni/bin
$ ls /opt/cni/bin/
bridge  cnitool  dhcp  flannel  host-local  ipvlan  loopback  macvlan  noop  ptp  tuning
```

kubelet

kubelet /usr/local/bin

```
$ cp kubernetes/server/bin/kubelet /usr/local/bin
```

kubelet

```
$ mkdir -p /var/lib/kubelet
```

```
$ yum install ebtables socat util-linux conntrack-tools
```

ApiServer

kubelet-csr.json

```
{
  "CN": "system:node:node2",
  "hosts": [
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "system:nodes",
      "OU": "cloudnative"
    }
  ]
}
```

- CN `system:node:<node-name>`
- OKubernetes RBAC ClusterRoleBinding Group system:nodes ClusterRole system:node

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=frognew kubelet-csr.json |
cfssljson -bare kubelet
$ ls kubelet*
kubelet.csr  kubelet-csr.json  kubelet-key.pem  kubelet.pem
```

/etc/kubernetes/pki/

```
$ scp ./{kubelet.csr,kubelet-key.pem,kubelet.pem} node02:/etc/kubernetes/pki/
...
```

node2 kubeconfig

```
$ cd /etc/kubernetes
$ export KUBE_APISERVER="https://192.166.1.222:6443"
```

cluster

```
$ kubectl config set-cluster kubernetes \
  --certificate-authority=/etc/kubernetes/pki/ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=kubelet.conf
```

credentials

```
$ kubectl config set-credentials system:node:node2 \
  --client-certificate=/etc/kubernetes/pki/kubelet.pem \
  --embed-certs=true \
  --client-key=/etc/kubernetes/pki/kubelet-key.pem \
  --kubeconfig=kubelet.conf
```

context

```
$ kubectl config set-context system:node:node2@kubernetes \
  --cluster=kubernetes \
  --user=system:node:node2 \
  --kubeconfig=kubelet.conf
```

default context

```
$ kubectl config use-context system:node:node2@kubernetes --kubeconfig=kubelet.conf
```

kubelet systemd unit service `NodeIP`

```
$ export KUBE_APISERVER="https://192.166.1.222:6443"
$ export NodeIP="192.166.1.2"
$ cat > /usr/lib/systemd/system/kubelet.service <<EOF
[Unit]
Description=kubelet
After=docker.service
Requires=docker.service

[Service]
WorkingDirectory=/var/lib/kubelet
EnvironmentFile=-/etc/kubernetes/kubelet
ExecStart=/usr/local/bin/kubelet \
        --logtostderr=true \
        --v=0 \
        --address=${NodeIP} \
        --api-servers==${KUBE_APISERVER} \
        --cluster-dns=10.96.0.10 \
        --cluster-domain=cluster.local \
        --kubeconfig=/etc/kubernetes/kubelet.conf \
        --require-kubeconfig=true \
        --pod-manifest-path=/etc/kubernetes/manifests \
        --allow-privileged=true \
        --authorization-mode=Webhook \
        --client-ca-file=/etc/kubernetes/pki/ca.pem \
        --network-plugin=cni \
        --cni-conf-dir=/etc/cni/net.d \
        --cni-bin-dir=/opt/cni/bin \
        --cgroup-driver=systemd
Restart=on-failure

[Install]
WantedBy=multi-user.target
EOF
```

- `--pod-manifest-path=/etc/kubernetes/manifests` Pod

  ```
  $ mkdir -p /etc/kubernetes/manifests
  ```

/usr/lib/systemd/system/kubelet.service

```
[Unit]
Description=kubelet
After=docker.service
Requires=docker.service

[Service]
WorkingDirectory=/var/lib/kubelet
EnvironmentFile=-/etc/kubernetes/kubelet
ExecStart=/usr/local/bin/kubelet         --logtostderr=true         --v=0         --address=192.
166.1.2         --api-servers==https://192.166.1.222:6443         --cluster-dns=10.96.0.10         --cluster-
domain=cluster.local         --kubeconfig=/etc/kubernetes/kubelet.conf         --require-
kubeconfig=true         --pod-manifest-path=/etc/kubernetes/manifests         --allow-
privileged=true         --authorization-mode=Webhook         --client-ca-file=/etc/kubernetes/pki/ca.
pem         --network-plugin=cni         --cni-conf-dir=/etc/cni/net.d         --cni-bin-dir=/opt/cni
/bin     --cgroup-driver=systemd
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

kubelet

```
$ systemctl daemon-reload
$ systemctl enable kubelet
$ systemctl start kubelet
$ systemctl status kubelet
```

Node

```
$ kubectl get node
NAME      STATUS     AGE       VERSION
node02    NotReady   2m        v1.6.2
```

Proxy DNS Notready

kube-proxy

kuber-proxy /usr/local/bin kubelet

```
$ mkdir -p /var/lib/kube-proxy
```

ApiServer

kube-proxy-csr.json

```
{
  "CN": "system:kube-proxy",
  "hosts": [
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CN",
      "ST": "BeiJing",
      "L": "BeiJing",
      "O": "system:kube-proxy",
      "OU": "cloudnative"
    }
  ]
}
```

- CN User system:kube-proxyKubernetes RBACClusterRoleBindingsystem:kube-proxysystem:node-proxier system:node-proxierkube-proxy ApiServer

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=frognew kube-proxy-csr.json |
cfssljson -bare kube-proxy
$ ls kube-proxy*
kube-proxy.csr        kube-proxy-csr.json  kube-proxy-key.pem  kube-proxy.pem
```

/etc/kubernetes/pki/

```
$ scp ./{kube-proxy.csr,kube-proxy-key.pem,kube-proxy.pem} node02:/etc/kubernetes/pki/
...
```

node2 kubeconfig

kube-proxy.conf

```
$ cd /etc/kubernetes
$ export KUBE_APISERVER="https://192.166.1.222:6443"
```

cluster

```
$ kubectl config set-cluster kubernetes \
  --certificate-authority=/etc/kubernetes/pki/ca.pem \
  --embed-certs=true \
  --server=${KUBE_APISERVER} \
  --kubeconfig=kube-proxy.conf
```

credentials

```
$ kubectl config set-credentials system:kube-proxy \
  --client-certificate=/etc/kubernetes/pki/kube-proxy.pem \
  --embed-certs=true \
  --client-key=/etc/kubernetes/pki/kube-proxy-key.pem \
  --kubeconfig=kube-proxy.conf
```

context

```
$ kubectl config set-context system:kube-proxy@kubernetes \
  --cluster=kubernetes \
  --user=system:kube-proxy \
  --kubeconfig=kube-proxy.conf
```

default context

```
$ kubectl config use-context system:kube-proxy@kubernetes --kubeconfig=kube-proxy.conf
```

kube-proxy systemd unit service `NodeIP`

```
$ export KUBE_APISERVER="https://192.166.1.222:6443"
$ export NodeIP="192.166.1.22"
$ cat > /usr/lib/systemd/system/kube-proxy.service <<EOF
[Unit]
Description=kube-proxy
After=network.target

[Service]
WorkingDirectory=/var/lib/kube-proxy
EnvironmentFile=-/etc/kubernetes/kube-proxy
ExecStart=/usr/local/bin/kube-proxy \
        --logtostderr=true \
        --v=0 \
        --bind-address=${NodeIP} \
        --kubeconfig=/etc/kubernetes/kube-proxy.conf \
        --cluster-cidr=10.244.0.0/16

Restart=on-failure

[Install]
WantedBy=multi-user.target
EOF
```

kubelet-proxy

```
$ systemctl daemon-reload
$ systemctl enable kube-proxy
$ systemctl start kube-proxy
$ systemctl status -l kube-proxy
```

Pod Network Flannel

flannel DaemonSet Kubernetes etcd TLS flannel etcd etcd TLS Kubernetes Secret

```
$ kubectl create secret generic etcd-tls-secret  --from-file=/etc/etcd/ssl/etcd.pem --from-file=/etc/etcd/ssl
/etcd-key.pem  --from-file=/etc/etcd/ssl/ca.pem -n kube-system
$ kubectl describe secret etcd-tls-secret -n kube-system
Name:                etcd-tls-secret
Namespace:        kube-system
Labels:                <none>
Annotations:        <none>

Type:        Opaque

Data
====
ca.pem:                1322 bytes
etcd-key.pem:        1675 bytes
etcd.pem:        1476 bytes
```

YAML

```
$ mkdir -p ~/k8s/flannel
$ cd ~/k8s/flannel
$ wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel-rbac.yml
$ wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

kube-flannel.yml

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  ......
spec:
  template:
    metadata:
      ......
    spec:
      ......
      containers:
      - name: kube-flannel
        image: quay.io/coreos/flannel:v0.7.1-amd64
        command: [
             "/opt/bin/flanneld",
             "--ip-masq",
             "--kube-subnet-mgr",
             "-etcd-endpoints=https://192.166.1.12:2379,https://192.166.1.2:2379,https://192.166.1.13:2379",
             "-etcd-cafile=/etc/etcd/ssl/ca.pem",
             "--etcd-certfile=/etc/etcd/ssl/etcd.pem",
             "-etcd-keyfile=/etcd/etcd/ssl/etcd-key.pem",
             "--iface=etn0" ]
        securityContext:
          privileged: true
        ......
        volumeMounts:
        ......
        - name: etcd-tls-secret
          readOnly: true
          mountPath: /etc/etcd/ssl/
      ......
      volumes:
        ......
        - name: etcd-tls-secret
          secret:
            secretName: etcd-tls-secret
```

flanneld :

- `-etcd-endpoints:` etcd
- `-etcd-cafile:` etcdCA,/etc/etcd/ssl/ca.pemetcd-tls-secretSecret
- `--etcd-certfile:` etcd,/etc/etcd/ssl/etcd.pemetcd-tls-secretSecret
- `--etcd-keyfile:` etcd,/etc/etcd/ssl/etcd-key.pemetcd-tls-secretSecret
- `--iface:` Node eth0!

kube-flannel.yml

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: flannel
  namespace: kube-system
---
kind: ConfigMap
apiVersion: v1
```

```
metadata:
  name: kube-flannel-cfg
  namespace: kube-system
  labels:
    tier: node
    app: flannel
data:
  cni-conf.json: |
    {
      "name": "cbr0",
      "type": "flannel",
      "delegate": {
        "isDefaultGateway": true
      }
    }
  net-conf.json: |
    {
      "Network": "10.244.0.0/16",
      "Backend": {
        "Type": "vxlan"
      }
    }
---
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: kube-flannel-ds
  namespace: kube-system
  labels:
    tier: node
    app: flannel
spec:
  template:
    metadata:
      labels:
        tier: node
        app: flannel
    spec:
      hostNetwork: true
      nodeSelector:
        beta.kubernetes.io/arch: amd64
      tolerations:
      - key: node-role.kubernetes.io/master
        operator: Exists
        effect: NoSchedule
      serviceAccountName: flannel
      containers:
      - name: kube-flannel
        image: quay.io/coreos/flannel:v0.7.1-amd64
        command: [
             "/opt/bin/flanneld",
             "--ip-masq",
             "--kube-subnet-mgr",
             "-etcd-endpoints=https://192.166.1.12:2379,https://192.166.1.2:2379,https://192.166.1.13:2379",
             "-etcd-cafile=/etc/etcd/ssl/ca.pem",
             "--etcd-certfile=/etc/etcd/ssl/etcd.pem",
             "-etcd-keyfile=/etcd/etcd/ssl/etcd-key.pem",
             "--iface=etn0" ]
        securityContext:
          privileged: true
        env:
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        volumeMounts:
        - name: run
```

```
          mountPath: /run
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
        - name: etcd-tls-secret
          readOnly: true
          mountPath: /etc/etcd/ssl/
      - name: install-cni
        image: quay.io/coreos/flannel:v0.7.1-amd64
        command: [ "/bin/sh", "-c", "set -e -x; cp -f /etc/kube-flannel/cni-conf.json /etc/cni/net.d/10-
flannel.conf; while true; do sleep 3600; done" ]
        volumeMounts:
        - name: cni
          mountPath: /etc/cni/net.d
        - name: flannel-cfg
          mountPath: /etc/kube-flannel/
      volumes:
        - name: run
          hostPath:
            path: /run
        - name: cni
          hostPath:
            path: /etc/cni/net.d
        - name: flannel-cfg
          configMap:
            name: kube-flannel-cfg
        - name: etcd-tls-secret
          secret:
            secretName: etcd-tls-secret
```

flannel

```
$ kubectl create -f kube-flannel-rbac.yml
$ kubectl create -f kube-flannel.yml
```

```
$ kubectl get pods --all-namespaces
NAMESPACE     NAME                    READY     STATUS    RESTARTS   AGE
kube-system   kube-flannel-ds-1b1vv   2/2       Running   0          14s
$ ifconfig flannel.1
flannel.1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1450
        inet 10.244.0.0  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::b81f:4eff:fe06:71ce  prefixlen 64  scopeid 0x20<link>
        ether ba:1f:4e:06:71:ce  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 8 overruns 0  carrier 0  collisions 0
```

kube-dns

Kubernetes kube-dns Cluster Add-On Kubernetes DNS Pod Service

```
$ mkdir -p ~/k8s/kube-dns
$ cd ~/k8s/kube-dns
$ wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/cluster/addons/dns/kubedns-cm.yaml
$ wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/cluster/addons/dns/kubedns-sa.yaml
$ wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/cluster/addons/dns/kubedns-svc.yaml.
base
$ wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/cluster/addons/dns/kubedns-controller.
yaml.base
$ wget https://raw.githubusercontent.com/kubernetes/kubernetes/master/cluster/addons/dns/transforms2sed.sed
```

transforms2sed.sed

```
$ cat transforms2sed.sed
s/__PILLAR__DNS__SERVER__/$DNS_SERVER_IP/g
s/__PILLAR__DNS__DOMAIN__/$DNS_DOMAIN/g
s/__MACHINE_GENERATED_WARNING__/Warning: This is a file generated from the base underscore template file:
__SOURCE_FILENAME__/g
```

$DNS_SERVER_IP 10.96.0.10 DNS_DOMAIN cluster.local. $DNS_SERVER_IP kubelet --cluster-dns

transforms2sed.sed

```
s/__PILLAR__DNS__SERVER__/10.96.0.10/g
s/__PILLAR__DNS__DOMAIN__/cluster.local/g
s/__MACHINE_GENERATED_WARNING__/Warning: This is a file generated from the base underscore template file:
__SOURCE_FILENAME__/g
```

```
$ cd ~/k8s/kube-dns
$ sed -f transforms2sed.sed kubedns-svc.yaml.base > kubedns-svc.yaml
$ sed -f transforms2sed.sed kubedns-controller.yaml.base > kubedns-controller.yaml
```

DNS_SERVER kubelet --cluster-dns

```
$ kubectl create -f kubedns-cm.yaml
$ kubectl create -f kubedns-sa.yaml
$ kubectl create -f kubedns-svc.yaml
$ kubectl create -f kubedns-controller.yaml
```

kube-dns Pod Pod Running

```
$ kubectl get pods --all-namespaces
NAMESPACE     NAME                          READY     STATUS    RESTARTS   AGE
kube-system   kube-dns-1759312207-wd7m2   3/3       Running   0          1m
kube-system   kube-flannel-ds-1b1vv       2/2       Running   0          6m
```

```
$ kubectl get nodes
NAME      STATUS    AGE       VERSION
node02    Ready     54m       v1.6.2
```

DNS

```
$ kubectl run bosybox --image=registry.aliyuncs.com/slzcc/busybox -i --tty sh

$ nslookup kubernetes.default
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      kubernetes
Address 1: 10.96.0.1 kubernetes.default.svc.cluster.local
```

kube-dns Kubernetes DNS Pod kube-dns

- `kubectl --namespace=kube-system scale deployment kube-dns --replicas=<NUM_YOU_WANT>`

Node

1.7.2 Node

- kubelet ApiServer kubeconfig CN `system:node:<node-name>` Node
- kube-proxy ApiServer Node kubeconfig kube-proxy.conf

```
$ kubectl get nodes
NAME      STATUS    AGE       VERSION
node02    Ready     54m       v1.6.2
...
```