创建kubernetes 各组件TLS加密通信的证书和密钥

kubernetes 系统的各组件需要使用TLS证书对通信进行加密，这里使用 CloudFlare 的PKI工具集 cfssl 来生成 Certificate Authority (CA) 和其它证书

生成的CA证书和密钥文件如下：

- ca-key.pem
- ca.pem
- kubernetes-key.pem
- kubernetes.pem
- kube-proxy.pem
- kube-proxy-key.pem
- admin.pem
- admin-key.pem

使用证书的组件如下：

- etcd 使用：ca.pem 、kubernetes-key.pem、kubernetes.pem
- kube-apiserver ：使用 ca.pem、 kubernetes-key.pem、kubernetes.pem
- kubelet 使用：ca.pem
- kube-proxy 使用：ca.pem 、kube-proxy-key.pem 、kube-proxy.pem
- kubectl 使用：ca.pem、admin-key.pem、 admin.pem

kube-controller 、kube-scheduler 当前需要和kube-apiserver部署在同一台机器上且使用非安全端口通信，故不需要证书。

注意：以下操作都在 master 节点即 192.168.251.101 这台主机上执行，证书只需要创建一次即可，以后在向集群中添加新节点时只要将 /etc/kubernetes/ 目录下的证书拷贝到新节点上即可。

安装 CFSSL

直接使用二进制源码包安装

```
[root@vlnx251101 ~]# vim a.sh

mkdir -p local/bin

wget https://pkg.cfssl.org/R1.2/cfssl_linux-amd64
chmod +x cfssl_linux-amd64
mv cfssl_linux-amd64 /root/local/bin/cfssl


wget https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
chmod +x cfssljson_linux-amd64
mv cfssljson_linux-amd64 /root/local/bin/cfssljson


wget https://pkg.cfssl.org/R1.2/cfssl-certinfo_linux-amd64
chmod +x cfssl-certinfo_linux-amd64
mv cfssl-certinfo_linux-amd64 /root/local/bin/cfssl-
certinfo


[root@vlnx251101 ~]# bash a.sh
[root@vlnx251101 ~]# export PATH=/root/local/bin:$PATH
```

创建CA (Certificate Authority)

创建 CA 配置文件

```
[root@vlnx251101 ~]# mkdir /root/ssl
[root@vlnx251101 ~]# cd /root/ssl
[root@vlnx251101 ssl]# cfssl print-defaults config > config.json
[root@vlnx251101 ssl]# cfssl print-defaults csr > csr.json


# 根据config.json文件的格式创建如下的ca-config.json文件
# 过期时间设置成了 87600h
cat <<EOF > ca-config.json
{
     "signing": {
       "default": {
         "expiry": "8760h"
       },
       "profiles": {
         "kubernetes": {
           "usages": [
               "signing",
               "key encipherment",
               "server auth",
               "client auth"
           ],
           "expiry": "8760h"
         }
      }
```

```
    }
}
EOF
```

字段说明

- ca-config.json ：可以定义多个profiles，分别指定不同的过期时间、使用场景等参数；后续在签名证书时使用某个 profile；
- signing：表示该证书可用于签名其它证书；生成的 ca.pem 证书中CA=TRUE；
- server auth：表示 client 可以用该CA 对server提供的证书进行验证；
- client auth ：表示server可以用该 CA对 client 提供的证书进行验证；

创建CA证书签名请求

```
cat <<EOF > ca-csr.json
{
      "CN": "kubernetes",
      "key": {
        "algo": "rsa",
        "size": 2048
      },
"names": [ {
          "C": "CN",
          "ST": "BeiJing",
          "L": "BeiJing",
          "O": "k8s",
          "OU": "System"
        } ]
}
EOF
```

- "CN": Common Name,kube-apiserver从证书中提取该字段作为请求的用户名（User Name）；浏览器使用该字段验证网站是否合法；
- "O": Organization, kube-apiserver从证书中提取该字段作为请求用户所属的组（Group）；

生成CA证书和私钥

```
[root@vlnx251101 ssl]# cfssl gencert -initca ca-csr.json | cfssljson -bare ca

[root@vlnx251101 ssl]# ls ca*
ca-config.json  ca.csr  ca-csr.json  ca-key.pem  ca.pem
```

创建kubernetes证书

创建 kubernetes证书签名请求

```
cat <<EOF > kubernetes-csr.json
{
    "CN": "kubernetes",
    "hosts": [
      "127.0.0.1",
      "192.168.251.101",
      "192.168.251.102",
      "192.168.251.103",
      "192.168.251.104",
      "10.254.0.1",
      "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": "System"
    } ]
}
EOF
```

- 如果hosts字段不为空则需要指定授权使用该证书的**IP或域名列表**，由于该证书后续被 etcd 集群和 kubernetes master 集群使用，所以上面分别指定了etcd集群、kubernetes master集群的主机IP和**kubernetes服务的服务IP**（一般是 kube-apiserver指定的 service-cluster-ip-range 网段的第一个IP，如 10.254.0.1。

生成kubernetes证书和私钥

```
[root@vlnx251101 ssl]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes kubernetes-csr.json | cfssljson -bare kubernetes

[root@vlnx251101 ssl]# ls kubernetes*
kubernetes.csr  kubernetes-csr.json  kubernetes-key.pem kubernetes.pem
```

创建admin证书

创建admin证书签名请求

```
cat <<EOF >  admin-csr.json
{
      "CN": "admin",
      "hosts": [],
      "key": {
        "algo": "rsa",
        "size": 2048
      },
      "names": [ {
          "C": "CN",
          "ST": "BeiJing",
          "L": "BeiJing",
          "O": "system:masters",
          "OU": "System"
      } ]
}
EOF
```

- 后续 kube-apiserver使用 RBAC 对客户端（如 kubelet、 kube-proxy 、Pod ）请求进行授权；
- kube-apiserver 预定了一些RBAC 使用的RoleBindings，如cluster-admin将 Group system:masters与 Role cluster-admin 绑定，该Role授予了调用 kube-apiserver 的所有**API** 的权限；
- OU指定该证书的 Group 为system:masters ，kubelets使用该证书访问 kube-apiserver 时，由于证书被CA签名，所以认证通过，同时由于证书用户组为经过预授权的 system:masters ，所以被授予访问所有 API的权限

生成 admin证书和私钥

```
[root@vlnx251101 ssl]# cfssl gencert -ca=ca.pem -ca-
key=ca-key.pem -config=ca-config.json -profile=kubernetes
admin-csr.json | cfssljson -bare admin

[root@vlnx251101 ssl]# ls admin*
admin.csr  admin-csr.json  admin-key.pem  admin.pem
```

创建kube-proxy证书

创建kube-proxy证书签名请求

```
cat <<EOF > kube-proxy-csr.json
{
     "CN": "system:kube-proxy",
     "hosts": [],
     "key": {
       "algo": "rsa",
       "size": 2048
     },
    "names": [ {
          "C": "CN",
          "ST": "BeiJing",
          "L": "BeiJing",
          "O": "k8s",
          "OU": "System"
    } ]
}
EOF
```

- CN指定该证书的 User 为system:kube-proxy；
- kube-apiserver预定义的 RoleBinding cluster-admin 将User system:kube-proxy与 Rolesystem:node-proxier绑定，该 Role 授予了调用kube-apiserver Proxy 相关API的权限

生成kube-proxy客户端证书和私钥

```
[root@vlnx251101 ssl]# cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes kube-proxy-csr.json | cfssljson -bare kube-proxy

[root@vlnx251101 ssl]# ls kube-proxy*
kube-proxy.csr  kube-proxy-csr.json  kube-proxy-key.pem  kube-proxy.pem
```

校验证书

以kubernetes证书为例

使用opsnssl命令

```
[root@vlnx251101 ssl]# openssl x509  -noout -text -in kubernetes.pem
```

- 确认Issuer字段的内容和 ca-csr.json一致
- 确认Subject 字段的内容和 kubernetes-csr.json一致
- 确认X509v3 Subject Alternative Name 字段的内容和 kubernetes-csr.json 一致
- 确认X509v3 Key Usage Extended Key Usage 字段的内容和ca-config.json 中 kubernetes profile一致

使用cfssl-certinfo命令

```
[root@vlnx251101 ssl]# cfssl-certinfo -cert kubernetes.pem
```

分发证书

将生成的证书和密钥文件（后缀名为.pem ）**拷贝到所有机器的 /etc/kubernetes/ssl** 目录下备用；

```
[root@vlnx251101 ssl]# mkdir -p /etc/kubernetes/ssl
[root@vlnx251101 ssl]# scp *.pem
192.168.251.101:/etc/kubernetes/ssl/
```