Kubernetes安装之证书验证 (https://www.kubernetes.org.cn/1861.html)

2017-04-12 17:17 中文社区 (https://www.kubernetes.org.cn/author/edit) 分类：Kubernetes安装说明 (https://www.kubernetes.org.cn/course/install) 阅读(8985) 作者：Jimmy Song | 原文 (http://rootsongjc.github.io/blogs/kubernetes-tls-certificate/?from=timeline&isappinstalled=0) 评论(0)

前言

昨晚（Apr 9,2017）金山软件的opsnull (https://github.com/opsnull)发布了一个开源项目和我一步步部署kubernetes集群 (https://github.com/opsnull/follow-me-install-kubernetes-cluster)，下文是结合我之前部署kubernetes的过程 (http://rootsongjc.github.io/tags/kubernetes/)打造的kubernetes环境和opsnull的文章创建 kubernetes 各组件 TLS 加密通信的证书和秘钥 (https://github.com/opsnull/follow-me-install-kubernetes-cluster/blob/master/01-TLS%E8%AF%81%E4%B9%A6%E5%92%8C%E7%A7%98%E9%92%A5.md)的实践。之前安装过程中一直使用的是非加密方式，一直到后来使用Fluentd和ElasticSearch收集Kubernetes集群日志 (http://rootsongjc.github.io/blogs/kubernetes-fluentd-elasticsearch-installation/)时发现有权限验证问题，所以为了深入研究kubernentes。

Kubernentes中的身份验证

`kubernetes` 系统的各组件需要使用 `TLS` 证书对通信进行加密，本文档使用 `CloudFlare` 的 PKI 工具集 cfssl (https://github.com/cloudflare/cfssl) 来生成 Certificate Authority (CA) 和其它证书；

生成的 CA 证书和秘钥文件如下：

- 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。

安装 CFSSL

方式一：直接使用二进制源码包安装

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

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

$ export PATH=/root/local/bin:$PATH
```

方式二：使用go命令安装

我们的系统中安装了Go1.7.5，使用以下命令安装更快捷：

```
$go get -u github.com/cloudflare/cfssl/cmd/...
$echo $GOPATH
/usr/local
$ls /usr/local/bin/cfssl*
cfssl cfssl-bundle cfssl-certinfo cfssljson cfssl-newkey cfssl-scan
```

在 `$GOPATH/bin` 目录下得到以cfssl开头的几个命令。

创建 CA (Certificate Authority)

创建 CA 配置文件

```
$ mkdir /root/ssl
$ cd /root/ssl
$ cfssl print-defaults config > config.json
$ cfssl print-defaults csr > csr.json
$ cat ca-config.json
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "kubernetes": {
        "usages": [
            "signing",
            "key encipherment",
            "server auth",
            "client auth"
        ],
        "expiry": "8760h"
      }
    }
  }
}
```

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

- “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

```
$ cfssl gencert -initca ca-csr.json | cfssljson -bare ca
$ 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 kubernetes-csr.json
{
    "CN": "kubernetes",
    "hosts": [
      "127.0.0.1",
      "172.20.0.112",
      "172.20.0.113",
      "172.20.0.114",
      "172.20.0.115",
      "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"
        }
    ]
}
```

- 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 kue-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

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-conf  kubernetes 中文社区  s kubernetes-csr.json | cfssljson -bare kubernetes
$ ls kuberntes*
kubernetes.csr  kubernetes-csr.json  kubernetes-key.pem  kubernetes.pem
```

或者直接在命令行上指定相关参数:

```
$ echo '{"CN":"kubernetes","hosts":[""],"key":{"algo":"rsa","size":2048}}' | cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -pr
ofile=kubernetes -hostname="127.0.0.1,10.64.3.7,10.254.0.1,kubernetes,kubernetes.default" - | cfssljson -bare kubernetes
```

创建 Admin 证书

创建 admin 证书签名请求

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

- 后续 `kube-apiserver` 使用 `RBAC` 对客户端(如 `kubelet`、 `kube-proxy`、 `Pod`)请求进行授权;
- `kube-apiserver` 预定义了一些 `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`， `kubelet` 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

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes admin-csr.json | cfssljson -bare admin
$ 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 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"
    }
  ]
}
```

- 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 Role `system: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

```
$ cfssl gencert -ca=ca.pem -ca-key=ca-key.pem -config=ca-config.json -profile=kubernetes  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
```

校验证书

以 kubernetes 证书为例

使用 `Opsnssl` 命令

```
$ openssl x509  -noout -text -in  kubernetes.pem
...
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=CN, ST=BeiJing, L=BeiJing, O=k8s, OU=System, CN=Kubernetes
        Validity
            Not Before: Apr  5 05:36:00 2017 GMT
            Not After : Apr  5 05:36:00 2018 GMT
        Subject: C=CN, ST=BeiJing, L=BeiJing, O=k8s, OU=System, CN=kubernetes
...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                DD:52:04:43:10:13:A9:29:24:17:3A:0E:D7:14:DB:36:F8:6C:E0:E0
            X509v3 Authority Key Identifier:
                keyid:44:04:3B:60:BD:69:78:14:68:AF:A0:41:13:F6:17:07:13:63:58:CD

            X509v3 Subject Alternative Name:
                DNS:kubernetes, DNS:kubernetes.default, DNS:kubernetes.default.svc, DNS:kubernetes.default.svc.cluster, DNS:kubernetes.default.svc.
cluster.local, IP Address:127.0.0.1, IP Address:172.20.0.112, IP Address:172.20.0.113, IP Address:172.20.0.114, IP Address:172.20.0.115, IP Address
:10.254.0.1
...
```

- 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

```
$ cfssl-certinfo -cert kubernetes.pem
...
{
  "subject": {
    "common_name": "kubernetes",
    "country": "CN",
    "organization": "k8s",
    "organizational_unit": "System",
    "locality": "BeiJing",
    "province": "BeiJing",
    "names": [
      "CN",
      "BeiJing",
      "BeiJing",
      "k8s",
      "System",
      "kubernetes"
    ]
  },
  "issuer": {
    "common_name": "Kubernetes",
    "country": "CN",
    "organization": "k8s",
    "organizational_unit": "System",
    "locality": "BeiJing",
    "province": "BeiJing",
    "names": [
      "CN",
      "BeiJing",
      "BeiJing",
      "k8s",
      "System",
      "Kubernetes"
    ]
  },
  "serial_number": "174360492872423263473151971632292895707129022309",
  "sans": [
    "kubernetes",
    "kubernetes.default",
    "kubernetes.default.svc",
    "kubernetes.default.svc.cluster",
    "kubernetes.default.svc.cluster.local",
    "127.0.0.1",
    "10.64.3.7",
    "10.254.0.1"
  ],
  "not_before": "2017-04-05T05:36:00Z",
  "not_after": "2018-04-05T05:36:00Z",
  "sigalg": "SHA256WithRSA",
...
```

分发证书

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

```
$ sudo mkdir -p /etc/kubernetes/ssl
$ sudo cp *.pem /etc/kubernetes/ssl
```

参考

- Generate self-signed certificates (https://coreos.com/os/docs/latest/generate-self-signed-certificates.html)
- Setting up a Certificate Authority and Creating TLS Certificates (https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/02-certificate-authority.md)
- Client Certificates V/s Server Certificates (https://blogs.msdn.microsoft.com/kaushal/2012/02/17/client-certificates-vs-server-certificates/)
- 数字证书及 CA 的扫盲介绍 (http://blog.jobbole.com/104919/)

上一篇: 容器编排之Kubernetes1.6.1安装与配置 (https://www.kubernetes.org.cn/1860.html)

下一篇: Kubernetes高可用Master节点安装 (https://www.kubernetes.org.cn/1865.html)

相关推荐

- kubeadm HA master(v1.14.0)离线包 + 自动化脚本 + 常用插件 For Centos/Fedora (https://www.kubernetes.org.cn/5213.html)
- 二进制部署Kubernetes v1.13.4 HA可选 (https://www.kubernetes.org.cn/5163.html)
- kubernetes1.13.1+etcd3.3.10+flanneld0.10集群部署 (https://www.kubernetes.org.cn/5025.html)
- CentOS 使用二进制部署 Kubernetes 1.13集群 (https://www.kubernetes.org.cn/4963.html)
- Kubernetes-部署API网关Kong (https://www.kubernetes.org.cn/4952.html)
- 使用kubeadm安装Kubernetes 1.13 (https://www.kubernetes.org.cn/4956.html)
- kubeadm HA master(v1.13.0)离线包 + 自动化脚本 + 常用插件 For Centos/Fedora (https://www.kubernetes.org.cn/4948.html)
- kubeadm HA master(v1.12.1)集群搭建指南(离线包 + 自动化脚本 + ipvs + keepalived + calico + helm) For Centos/Fedora (https://www.kubernetes.org.cn/4725.html)

评论 抢沙发

社区交流

8 3 5 5 4

提交评论

昵称 昵称 (必填)

邮箱 邮箱 (必填)

网址 网址