Generating a Root CA, Server, and Client Certs using CFSSL

Jump to bottom

Jason Riddle edited this page on May 7, 2016 · 38 revisions

> Modified from https://github.com/kelseyhightower/intro-to-kubernetes-workshop

Clone this directory

```
$ git clone https://github.com/jason-riddle/generating-certs.git
```

Create the Root CA (1-generate-root-ca)

```
$ cd 1-generate-root-ca
```

```
$ curl -O https://storage.googleapis.com/configs.kuar.io/ca-csr.json
```

```
$ cat ca-csr.json
{
  "CN": "Kubernetes CA",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "Kubernetes",
      "OU": "CA",
      "ST": "Oregon"
    }
  ]
}
```

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

```
$ ls
# ca-csr.json    ca-key.pem     ca.csr         ca.pem
```

Prepare for next section.

```
$ curl -O https://storage.googleapis.com/configs.kuar.io/ca-config.json
```

```
$ cat ca-config.json
{
  "signing": {
    "default": {
      "expiry": "168h"
    },
    "profiles": {
      "server": {
        "expiry": "8760h",
        "usages": [
          "signing",
          "key encipherment",
          "server auth"
        ]
      },
      "client": {
        "expiry": "8760h",
        "usages": [
          "signing",
          "key encipherment",
          "client auth"
        ]
      },
      "client-server": {
        "expiry": "8760h",
        "usages": [
          "signing",
          "key encipherment",
          "server auth",
          "client auth"
        ]
      }
    }
  }
}
```

```
$ ls
# ca-config.json ca-csr.json    ca-key.pem     ca.csr         ca.pem
```

Create the Server Certificate (2-generate-server-cert)

```
$ cd 2-generate-server-cert
```

```
$ curl -O https://storage.googleapis.com/configs.kuar.io/apiserver-csr.json
```

```
$ cat apiserver-csr.json
{
  "CN": "*.c.PROJECT_ID.internal",
  "hosts": [
    "127.0.0.1",
    "EXTERNAL_IP",
    "*.c.PROJECT_ID.internal"
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "Kubernetes",
      "OU": "API Server",
      "ST": "Oregon"
    }
  ]
}
```

```
$ cfssl gencert \
    -ca=../1-generate-root-ca/ca.pem \
    -ca-key=../1-generate-root-ca/ca-key.pem \
    -config=../1-generate-root-ca/ca-config.json \
    -profile=server \
    apiserver-csr.json | cfssljson -bare apiserver
```

```
$ ls
# apiserver-csr.json apiserver-key.pem  apiserver.csr      apiserver.pem
```

Generate the Client Certificate (3-generate-client-cert)

```
$ cd 3-generate-client-cert
```

```
$ curl -O https://storage.googleapis.com/configs.kuar.io/admin-csr.json
```

```
$ cat admin-csr.json
{
  "CN": "admin",
  "hosts": [
    ""
```

```
  ],
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "Kubernetes",
      "OU": "Cluster Admins",
      "ST": "Oregon"
    }
  ]
}
```

```
cfssl gencert \
    -ca=../1-generate-root-ca/ca.pem \
    -ca-key=../1-generate-root-ca/ca-key.pem  \
    -config=../1-generate-root-ca/ca-config.json \
    -profile=client \
    admin-csr.json | cfssljson -bare admin
```

```
$ ls
# admin-csr.json admin-key.pem  admin.csr      admin.pem
```

Validating your certs

See Validating Certificates.

▼ Pages 12

Find a Page...

Home

Accessing (Hashicorp) Vault Secrets using Chef

Achieving High Availability for the Chef Server on AWS

Architecture Diagram

Bootstrapping a Chef Server and Chef Vault together

Generating a Root CA, Server, and Client Certs using CFSSL

Migrating from an older Chef Server to another

Securing Communication for ...

Using (Hashicorp) Vault to Issue Intermediate Certs through Roles