Now that you have provisioned a certificate authority for the Kubernetes cluster, you are ready to begin generating certificates. The first set of certificates you will need to generate consists of the client certificates used by various Kubernetes components. In this lesson, we will generate the following client certificates: `admin`, `kubelet` (one for each worker node), `kube-controller-manager`, `kube-proxy`, and `kube-scheduler`. After completing this lesson, you will have the client certificate files which you will need later to set up the cluster.

Here are the commands used in the demo. The command blocks surrounded by curly braces can be entered as a single command:

```
cd ~/kthw
```

Admin Client certificate:

```
{

cat > admin-csr.json << EOF
{
  "CN": "admin",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:masters",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  admin-csr.json | cfssljson -bare admin

}
```

Kubelet Client certificates. Be sure to enter your actual cloud server values for all four of the variables at the top:

```
WORKER0_HOST=<Public hostname of your first worker node cloud server>
WORKER0_IP=<Private IP of your first worker node cloud server>
WORKER1_HOST=<Public hostname of your second worker node cloud server>
WORKER1_IP=<Private IP of your second worker node cloud server>

{
cat > ${WORKER0_HOST}-csr.json << EOF
{
  "CN": "system:node:${WORKER0_HOST}",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:nodes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=${WORKER0_IP},${WORKER0_HOST} \
  -profile=kubernetes \
  ${WORKER0_HOST}-csr.json | cfssljson -bare ${WORKER0_HOST}

cat > ${WORKER1_HOST}-csr.json << EOF
{
  "CN": "system:node:${WORKER1_HOST}",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:nodes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=${WORKER1_IP},${WORKER1_HOST} \
  -profile=kubernetes \
  ${WORKER1_HOST}-csr.json | cfssljson -bare ${WORKER1_HOST}

}
```

Controller Manager Client certificate:

```
{

cat > kube-controller-manager-csr.json << EOF
{
  "CN": "system:kube-controller-manager",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:kube-controller-manager",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  kube-controller-manager-csr.json | cfssljson -bare kube-controller-manager

}
```

Kube Proxy Client certificate:

```
{

cat > kube-proxy-csr.json << EOF
{
  "CN": "system:kube-proxy",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:node-proxier",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  kube-proxy-csr.json | cfssljson -bare kube-proxy

}
```

Kube Scheduler Client Certificate:

```
{

cat > kube-scheduler-csr.json << EOF
{
  "CN": "system:kube-scheduler",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "system:kube-scheduler",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  kube-scheduler-csr.json | cfssljson -bare kube-scheduler

}
```