Cloud Provider:AWS
Server:Ubuntu 18.04 x6
Server List: 2 Controller(Master), 2(Worker), 1 Nginx(Reverse Proxy), 1 Ubuntu 18.04(local)

Installing Client Tools

// In order to proceed with Kubernetes the Hard Way, there are some client tools that you need to install on your local workstation

// installing cfssl

```
$ wget -q --show-progress --https-only --timestamping
  https://pkg.cfssl.org/R1.2/cfssl_linux-amd64 \
  https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
$ chmod +x cfssl_linux-amd64 cfssljson_linux-amd64
$ sudo mv cfssl_linux-amd64 /usr/local/bin/cfssl
$ sudo mv cfssljson_linux-amd64 /usr/local/bin/cfssljson
```

// installing kubectl

```
$ wget
https://storage.googleapis.com/kubernetes-release/release/v1.10.2/bin/linux/amd64/kubectl
$ chmod +x kubectl
$ sudo mv kubectl /usr/local/bin/
$ kubectl version --client
```

Provisioning the Certificate Authority

```
$ cd ~/
$ mkdir kthw
$ cd kthw/
$ sudo curl -s -L -o /bin/cfssl https://pkg.cfssl.org/R1.2/cfssl_linux-amd64
$ sudo curl -s -L -o /bin/cfssljson https://pkg.cfssl.org/R1.2/cfssljson_linux-amd64
$ sudo curl -s -L -o /bin/cfssl-certinfo https://pkg.cfssl.org/R1.2/cfssl-certinfo_linux-amd64
$ sudo chmod +x /bin/cfssl*
```

// Use this command to generate the certificate authority. Include the opening and closing curly braces to run this entire block as a single command.

```
{

cat > ca-config.json << EOF
{
  "signing": {
```

```
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "kubernetes": {
        "usages": ["signing", "key encipherment", "server auth", "client auth"],
        "expiry": "8760h"
      }
    }
  }
}
EOF

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

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

}
```

Generating Client Certificates

```
$ 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(변수설정해주기:ex WORKEREX_HOST=ec2-13-125-251-85.ap-northeast-2.compute.amazonaws.com)

```
$ 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

}
```

Generating the Kubernetes API Server Certificate

```
$ cd ~/kthw

$ CERT_HOSTNAME=10.32.0.1,<controller node 1 Private IP>,<controller node 1 hostname>,<controller node 2 Private IP>,<controller node 2 hostname>,<API load balancer Private IP>,<API load balancer hostname>,127.0.0.1,localhost,kubernetes.default

{

cat > kubernetes-csr.json << EOF
{

  "CN": "kubernetes",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
   {
     "C": "US",
     "L": "Portland",
     "O": "Kubernetes",
     "OU": "Kubernetes The Hard Way",
     "ST": "Oregon"
   }
  ]
}
EOF

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

}
```

Generating the Service Account Key Pair

```
$ cd ~/kthw

{

cat > service-account-csr.json << EOF
{
  "CN": "service-accounts",
```

```
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "US",
      "L": "Portland",
      "O": "Kubernetes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Oregon"
    }
  ]
}
EOF

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

}
```

Distributing the Certificate Files

```
// Move certificate files to the worker nodes

$ scp ca.pem <worker 1 hostname>-key.pem <worker 1 hostname>.pem
cloud_user@<worker 1 public IP>:~/

$ scp ca.pem <worker 2 hostname>-key.pem <worker 2 hostname>.pem
cloud_user@<worker 2 public IP>:~/

// Move certificate files to the controller nodes

$ scp ca.pem ca-key.pem kubernetes-key.pem kubernetes.pem \
  service-account-key.pem service-account.pem cloud_user@<controller 1 public IP>:~/

$ scp ca.pem ca-key.pem kubernetes-key.pem kubernetes.pem \
  service-account-key.pem service-account.pem cloud_user@<controller 2 public IP>:~/
```