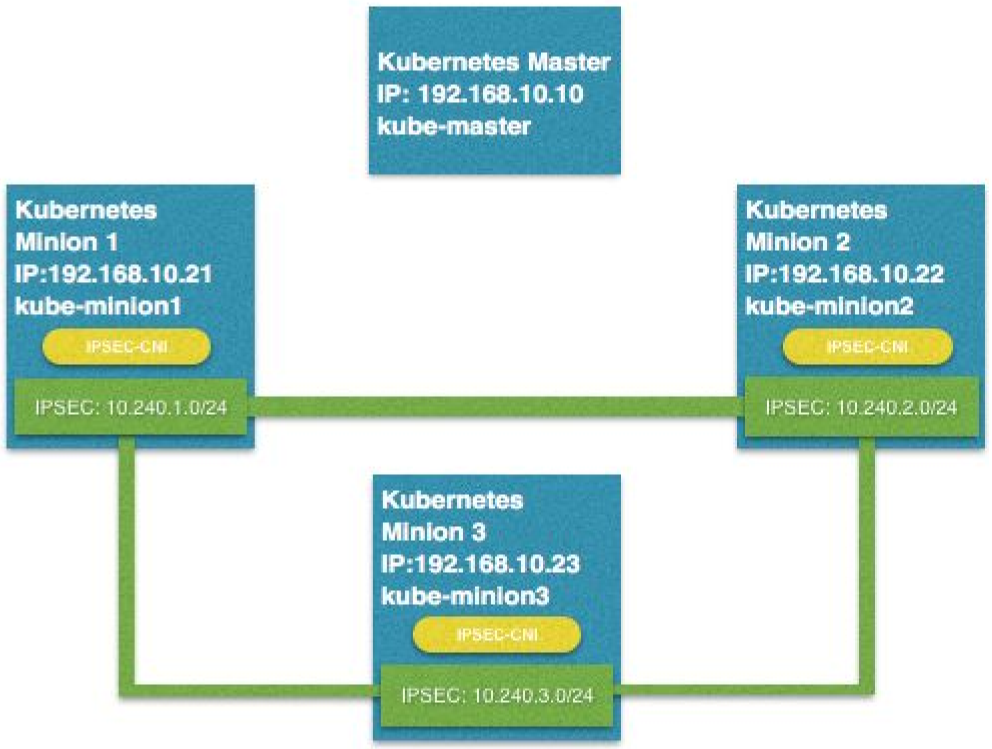
Kubernetes CNI Project



The above diagram gives the basic layout of the setup.
Primary IPs
Master : 192.168.10.10
Minion1 : 192.168.10.21
Minion2 : 192.168.10.22
Minion3 : 192.168.10.23

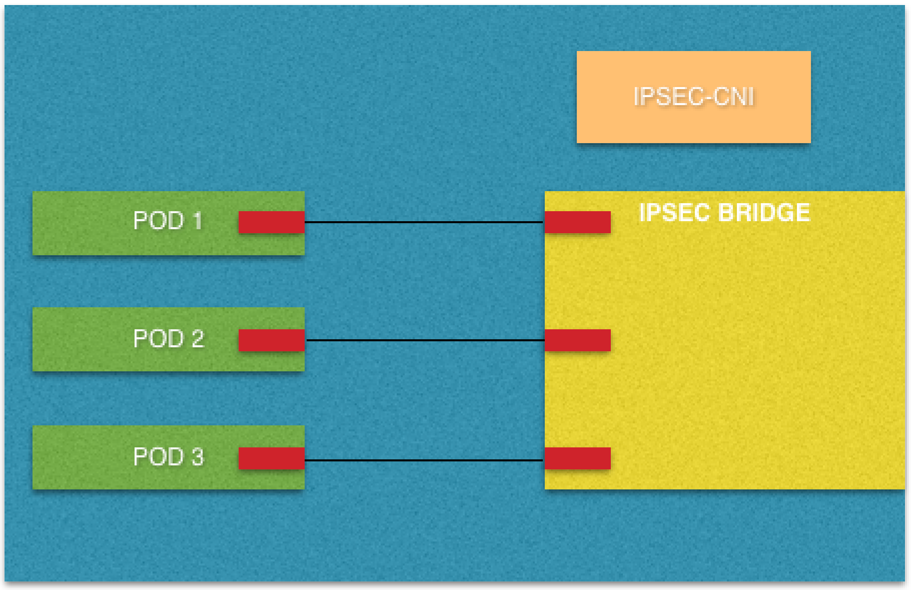
Service Cluster range : 10.30.0.0/24

IP range for POD: 10.240.0.0/16
This range will be split as follows between the nodes via vagrant(shell script) and will be used as input for ip assignment by the plugin. Further version can work similar to flannel to keep that data in etcd and make the split configurable and dynamic.
Minion1 : 10.240.1.0/24
Minion2 : 10.240.2.0/24
Minion3 : 10.240.3.0/24
These same ranges will be used for StrongSwan IPSEC communication.

- The kubernetes master will have the following services running

    - etcd
    - api server
    - scheduler
    - controller manager
    - Appropriate certs and token configuration will be required
- The kubernetes minion will have the following services running
    - docker
    - kubelet
        - cni configs will also be done here
    - kube-proxy
    - Strong swan
        - The server certs and client certs used will be common but we can different certs for different servers.
        - Every node will have configuration for the other two nodes like a host-to-host configuration.

Logical View of Node, POD and CNI Implementation



Notes for CNI Implementation

- The CNI Plugin will get triggered with env params (PATH, Container name, Container interface name, namespace and command)
- The CNI plugin as standard input will get details such as bridge name(ipsec-br01) and ip cidr for pod per node (populated via Vagrant)
- The CNI plugin will check if the bridge exists, if not it will create and set first ip as the bridge ip.
- Further on it will create a veth pair (veth-truncated-CONTAINERNAME and container interface name given from env variables)
- It will place the veth-truncated-CONTAINERNAME in the ipse bridge and the other end it will move to container namespace

- The logic will further require to maintain a dictionary of container names ,container name spaces ,container interface names and ip addresses assigned. It will also store this data in the file.

Sample Service Configurations

StrongSwan IPSEC Minion1 (appropriate changes for Minion2 and 3)

```
# /etc/ipsec.conf - strongSwan IPsec configuration file for Minion1

config setup
conn %default
  ikelifetime=60m
  keylife=20m
  rekeymargin=3m
  keyingtries=1
  keyexchange=ikev2
  authby=psk

conn minion2
  left=%defaultroute
  leftsubnet=10.240.1.0/24
  leftid=minion1
  leftfirewall=yes
  rightid=minion2
  right=192.168.10.22
  rightsubnet=10.240.2.0/24
  auto=add

conn minion3
  left=%defaultroute
  leftsubnet=10.240.1.0/24
  leftid=minion3
  leftfirewall=yes
  rightid=minion3
  right=192.168.10.23
  rightsubnet=10.240.3.0/24
  auto=add
```

ETCD

```
[Unit]
Description=etcd
Documentation=https://github.com/coreos

[Service]
ExecStart=/usr/bin/etcd --name $NODENAME \
  --cert-file=/etc/etcd/kubernetes.pem \
  --key-file=/etc/etcd/kubernetes-key.pem \
  --peer-cert-file=/etc/etcd/kubernetes.pem \
  --peer-key-file=/etc/etcd/kubernetes-key.pem \
  --trusted-ca-file=/etc/etcd/ca.pem \
  --peer-trusted-ca-file=/etc/etcd/ca.pem \
  --initial-advertise-peer-urls https://$IP:2380 \
  --listen-peer-urls https://$MASTER_IP:2380 \
  --listen-client-urls https://$MASTER_IP:2379,http://127.0.0.1:2379 \
  --advertise-client-urls https://$MASTER_IP:2379 \
  --initial-cluster-token etcd-cluster-0 \
  --initial-cluster $INITIAL_CLUSTER \
  --initial-cluster-state new \
  --data-dir=/var/lib/etcd
```

```
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

API Server

```
[Unit]
Description=Kubernetes API Server


[Service]
ExecStart=/usr/local/bin/kube-apiserver \
  --admission-control=NamespaceLifecycle,LimitRanger,SecurityContextDeny,ServiceAccount,ResourceQuota \
  --advertise-address=${IP} \
  --allow-privileged=true \
  --apiserver-count=3 \
  --authorization-mode=ABAC \
  --authorization-policy-file=/var/lib/kubernetes/authorization-policy.jsonl \
  --bind-address=0.0.0.0 \
  --enable-swagger-ui=true \
  --etcd-cafile=/var/lib/kubernetes/ca.pem \
  --insecure-bind-address=0.0.0.0 \
  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \
  --etcd-servers=${ETCD_CLIENT_ACCESS} \
  --service-account-key-file=/var/lib/kubernetes/kubernetes-key.pem \
  --service-cluster-ip-range=10.30.0.0/24 \
  --service-node-port-range=30000-32767 \
  --tls-cert-file=/var/lib/kubernetes/kubernetes.pem \
  --tls-private-key-file=/var/lib/kubernetes/kubernetes-key.pem \
  --token-auth-file=/var/lib/kubernetes/token.csv \
  --v=2
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Controller Manager

```
[Unit]
Description=Kubernetes Controller Manager

[Service]
ExecStart=/usr/local/bin/kube-controller-manager \
  --allocate-node-cidrs=true \
  --cluster-name=kubernetes \
  --leader-elect=true \
  —master=http://$MASTER_IP:8080 \
  --root-ca-file=/var/lib/kubernetes/ca.pem \
  --service-account-private-key-file=/var/lib/kubernetes/kubernetes-key.pem \
  --service-cluster-ip-range=10.30.0.0/16 \
  --v=2
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Scheduler

```
[Unit]
Description=Kubernetes Scheduler
```

```
[Service]
ExecStart=/usr/local/bin/kube-scheduler \
  --leader-elect=true \
  —master=http://$MASTER_IP:8080 \
  --v=2
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Docker

```
[Unit]
Description=Docker Application Container Engine
Documentation=http://docs.docker.io

[Service]
ExecStart=/usr/bin/docker daemon \
  --iptables=false \
  --ip-masq=false \
  --host=unix:///var/run/docker.sock \
  --log-level=error \
  --storage-driver=overlay
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

Kubelet

```
[Unit]
Description=Kubernetes Kubelet

After=docker.service
Requires=docker.service

[Service]
ExecStart=/usr/bin/kubelet \
  --allow-privileged=true \
  --api-servers=${API_SERVERS} \
  --cluster-dns=10.30.0.10 \
  --cluster-domain=cluster.local \
  --container-runtime=docker \
  —network-plugin=ipsec-cni \
  --kubeconfig=/var/lib/kubelet/kubeconfig \
  --serialize-image-pulls=false \
  --tls-cert-file=/var/lib/kubernetes/kubernetes.pem \
  --tls-private-key-file=/var/lib/kubernetes/kubernetes-key.pem \
  —v=2

Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

kube-proxy

```
[Unit]
Description=Kubernetes Kube Proxy
```

```
[Service]
ExecStart=/usr/bin/kube-proxy \
  --masquerade-all \
  —master=$MASTER_IP \
  --kubeconfig=/var/lib/kubelet/kubeconfig \
  --proxy-mode=iptables \
  --v=2

Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```