KubeCon

North America 2017

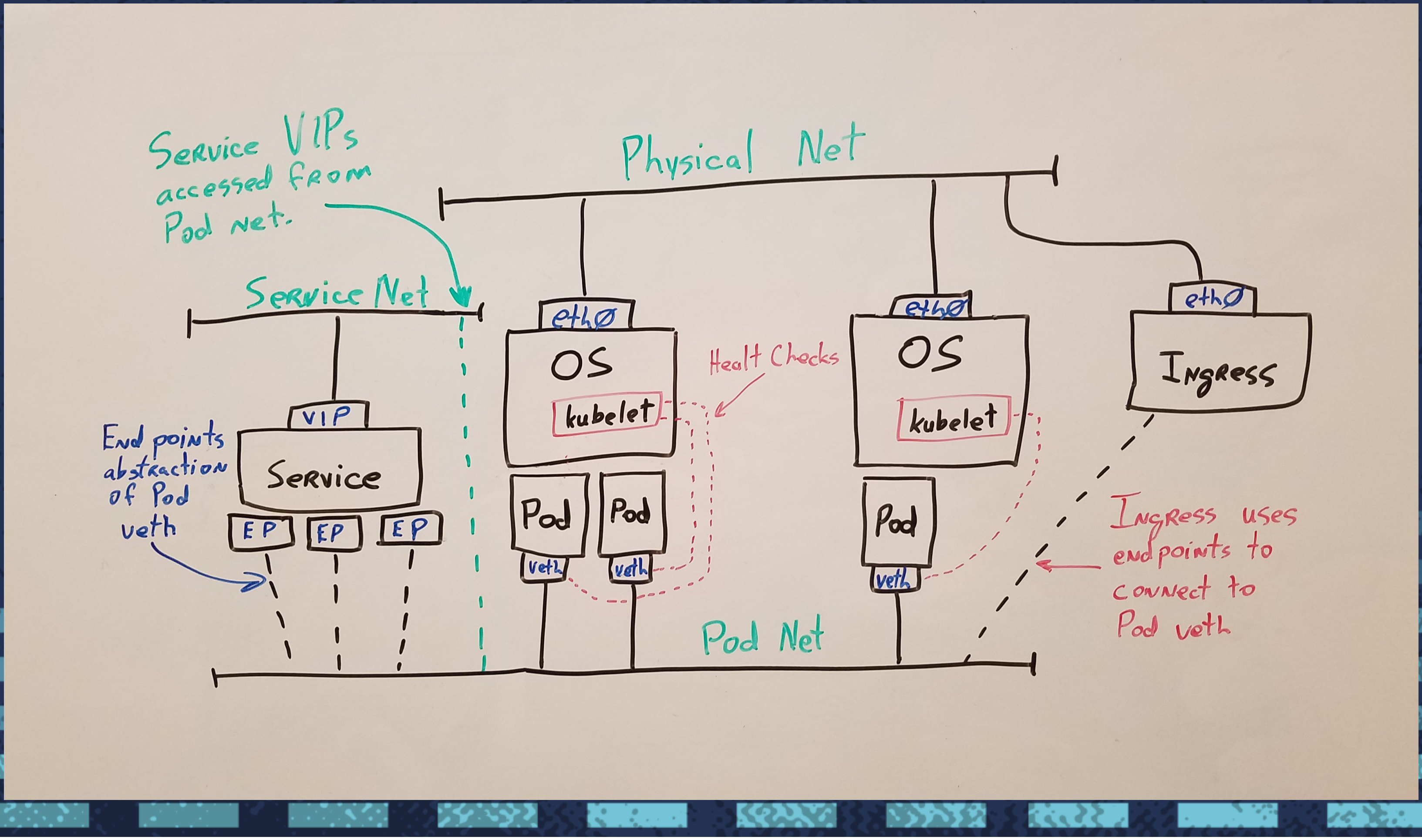
The Easy—Don't Drive Yourself Crazy—Way to Kubernetes Networking

Gerard Hickey, Principal Systems Engineer, Smartsheet

Accolades

- Too many people to thank directly
- Special thanks to Erik Stidham @ Tigera for helping me get the my first running network stack.

Kubernetes Network Topology



Useful Network Ranges

- Choose ranges for the Pod and Service CIDR blocks

- Generally any of the RFC-1918 ranges work well
  - 10.0.0.0/8
  - 172.0.0.0/11
  - 192.168.0.0/16

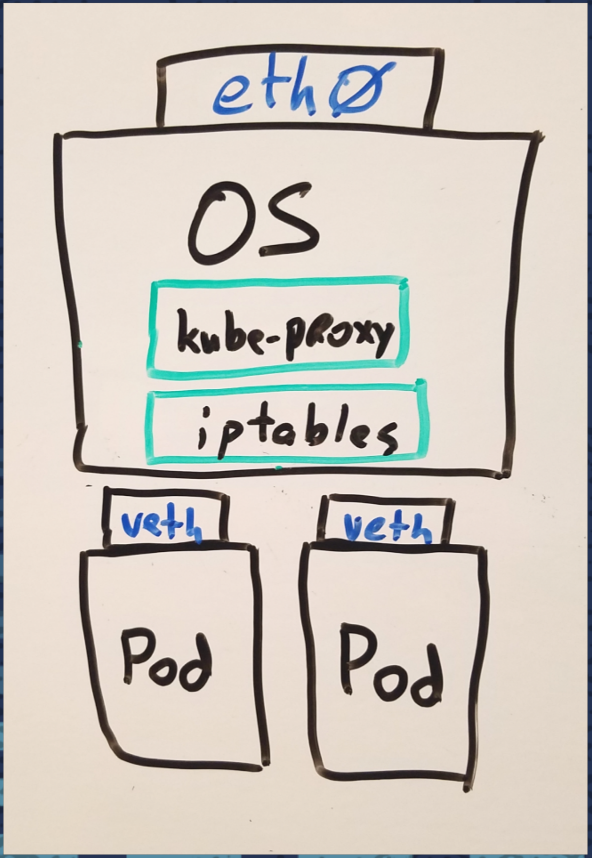
- Keep the network range simple, don’t be creative

Key Understanding #1

Every Pod can communicate directly with every other pod

Kubernetes Node

- A general purpose compute that has at least one interface
  - The host OS will have a real world IP for accessing the machine
  - Kubernetes Pods are given virtual interfaces connected to an internal
  - Each node has a running network stack
- Kube-proxy runs in the OS to control iptables
  - Services
  - NodePorts



Networking Substrate

- Most Kubernetes network stacks allocate subnets for each node
  - Network stack is responsible for arbitration of subnets and IPs
  - Network stack is also responsible for moving packets around the network
- Pods have a unique, routable IP on the Pod CIDR block
  - CIDR block is ***not*** accessed from outside the Kubernetes cluster
  - Magic of IP Tables allows the Pods to make outgoing connections
- Insure that Kubernetes has the correct Pod and Service CIDR blocks

Key Understanding #2

Pod network is not seen on physical network

Making Setup Easier: CNI

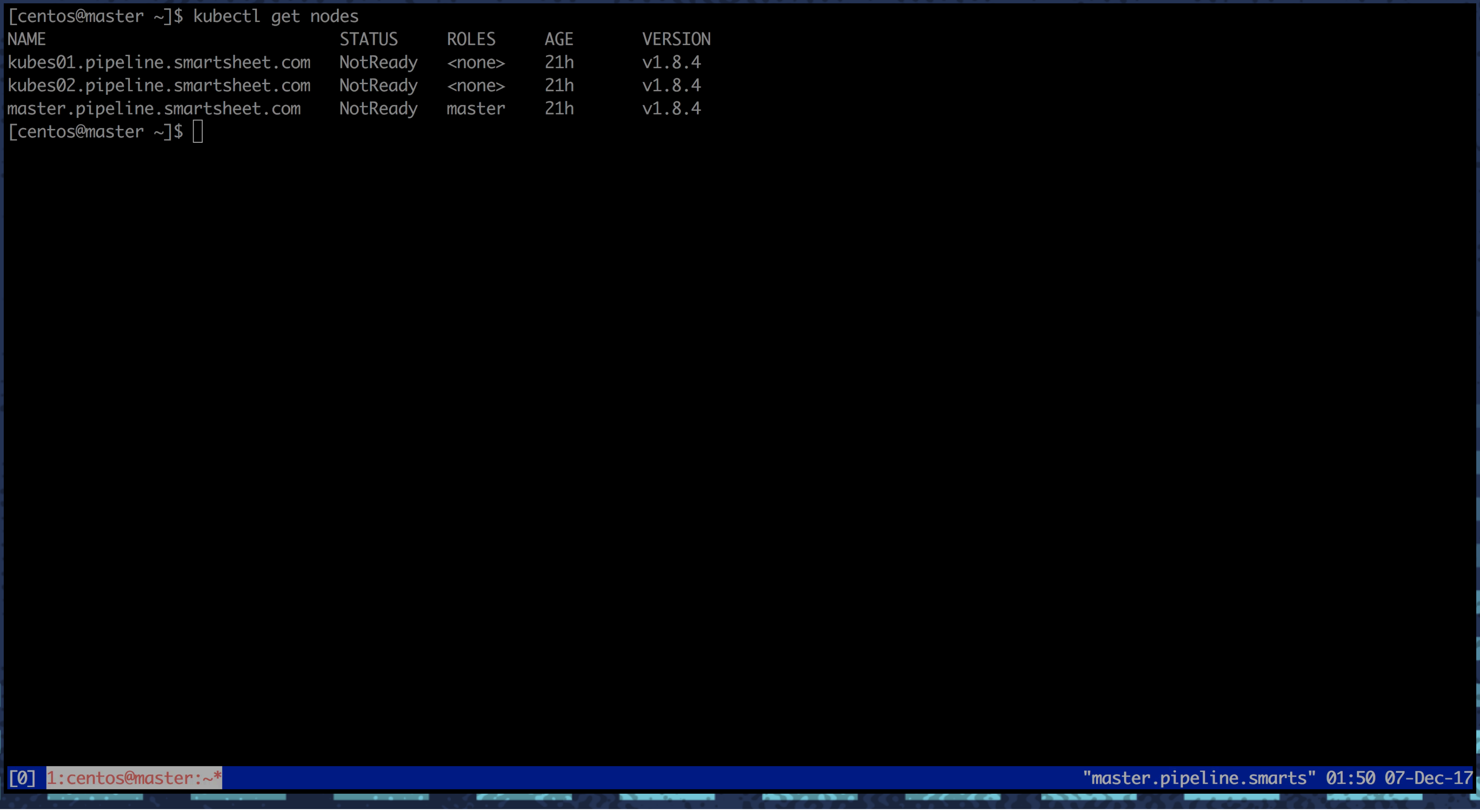
- Container Network Interface
- Relieves Kubernetes from having to have specific network configuration
- Activated by supplying `--network-plugin=cni, --cni-conf-dir, --cni-bin-dir` to kubelet
  - Typical configuration directory: `/etc/cni/net.d`
  - Typical bin directory: `/opt/cni/bin`
- Allows for multiple backends to be used: linux-bridge, macvlan, ipvlan, Open vSwitch, network stacks

CNI Configuration

- CNI is configured through a JSON file
- CNI generic parameters shown
- Plugins are allowed to have their own specific parameters
- Kubelet will use the configuration and call the plugin before each container starts

```
{
   "cniVersion":  "0.2.0",
   "name":  "mybridge",
   "type":  "bridge",
   "bridge":  "cni_bridge0",
   "isGateway": true,
   "ipMasq":  true,
   "ipam": {
      "type": "host-local",
      "subnet":  "10.15.20.0/24",
      "routes": [
         { "dst":  "0.0.0.0/0" },
         { "dst":  "1.1.1.1/32", "gw":"10.15.20.1"}
      ]
   }
}
```

Demonstration



```
[centos@master ~]$ kubectl get nodes
NAME                             STATUS     ROLES     AGE       VERSION
kubes01.pipeline.smartsheet.com  NotReady   <none>    21h       v1.8.4
kubes02.pipeline.smartsheet.com  NotReady   <none>    21h       v1.8.4
master.pipeline.smartsheet.com   NotReady   master    21h       v1.8.4
[centos@master ~]$ 
```

```
[0] 1:centos@master:~*                                                                      "master.pipeline.smarts" 01:50 07-Dec-17
```

Key Understanding #3

Services are crucial
for service discovery and
distributing traffic to Pods

Kubernetes Services

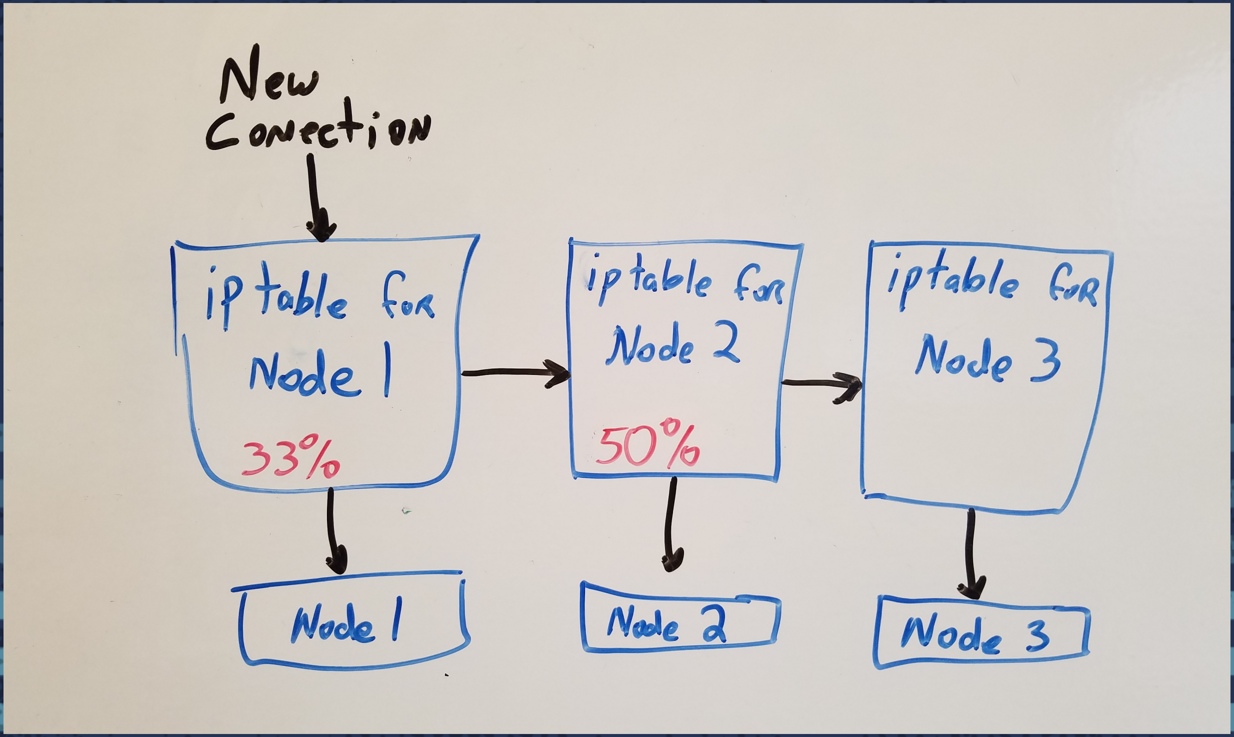
- Services act as simple internal load balancers with VIPs
  - No access controls
  - No traffic controls
- IP Tables magically route to virtual IPs
- Internally Services can are used as inter-Pod service discovery
  - Kube-DNS publishes DNS record (i.e. nginx.default.svc.cluster.local)
- Services can be exposed three different ways
  - ClusterIP, LoadBalancer, NodePort

kube-proxy

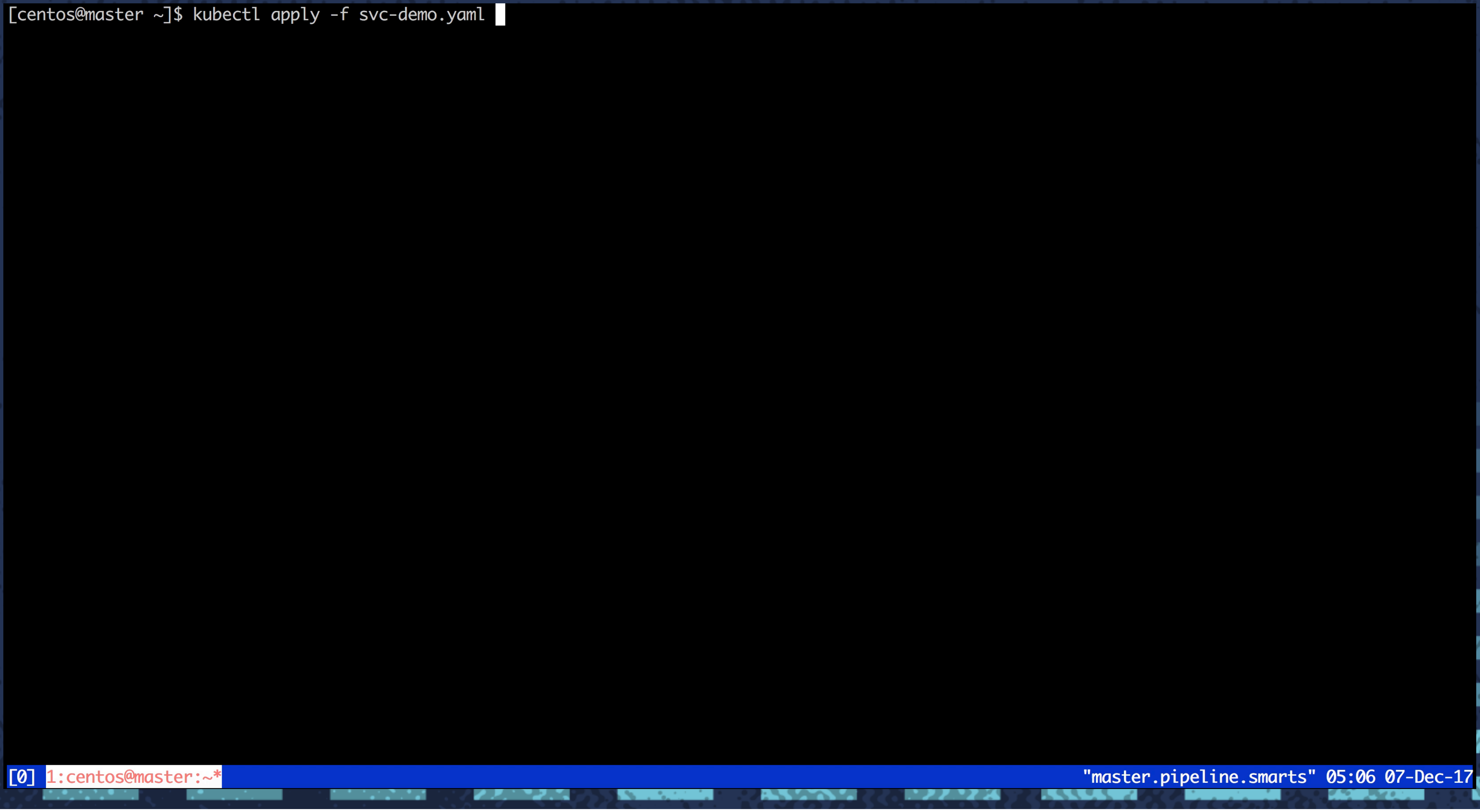
- Each Kubernetes node in the cluster runs a kube-proxy
- Two modes: userspace and iptables
  - iptables much more performant – userspace should no longer be used
- kube-proxy has the task of configuring iptables to expose each Kubernetes service
  - iptables rules distributes traffic randomly across the endpoints

kube-proxy Randomizer

- iptable rule created for each endpoint listed in a service
- Random number generated for each connection and used for routing to a specific node
- Last iptable rule accepts all traffic and routes to node



Demonstration



```
[centos@master ~]$ kubectl apply -f svc-demo.yaml
```

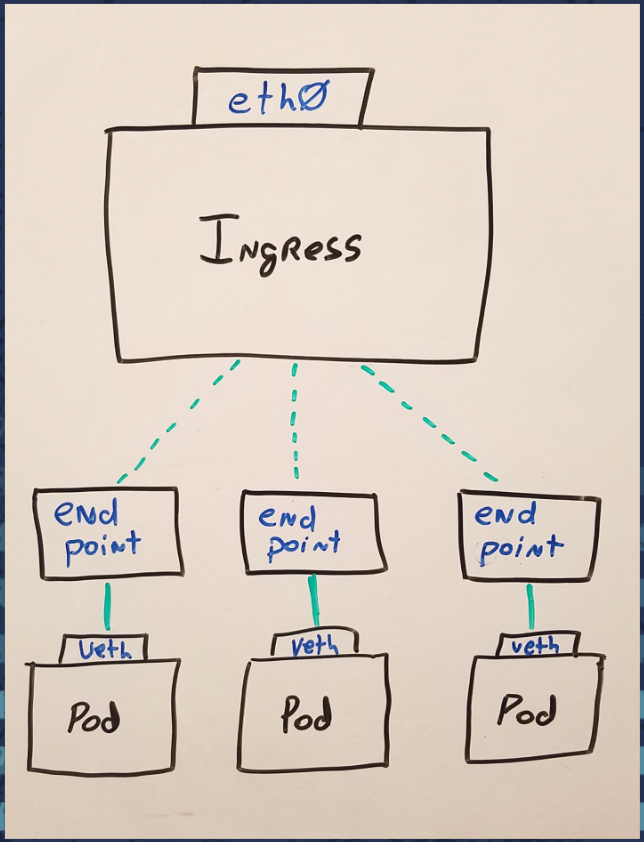
```
[0] 1:centos@master:~*                                   "master.pipeline.smarts" 05:06 07-Dec-17
```

Key Understanding #4

Ingresses are entry points into the Kubernetes network

Kubernetes Ingress

- Exposes Services outside the Kubernetes network
- Most Ingresses are layer 7 load balancers (i.e. HTTP/HTTPS)
  - NGINX, Traefik, haproxy, vulcand, cloud provider load balancers
  - F5 Container Connector
- A few layer 4 load balancers available but no standard yet
  - NGINX



Network Stack Choices

- Flannel
  - Most popular because it is simple and easy to use
- Weave Net
  - A bit more complex, scales better than Flannel
- Project Calico
  - Similar to Weave Net (may scale better), but one of the few that provide egress rules
- Romana
  - Tailored a bit more to security and is able to expose Services as real world VIPs

Summary of Key Understandings

- Every Pod can communicate directly with every other pod
- Pod network is not seen on physical network
- Services are crucial for service discovery and distributing traffic to Pods
- Ingresses are entry points into the Kubernetes network