Kubernetes Section

Kubernetes Overview

- ETCD overview
- CoreDNS
- Flannel
- Kubernetes Architecture
- Kubernetes Networking
- Understand the concepts of Kubernetes
- Vault overview

ETCD Overview

What is ETCD?

What is Key-Value Store?

How ETC Operates

RAFT Protocol

Best Practices on number of nodes

Helpful Links

Cluster DNS: CoreDNS vs Kube-DNS

CoreDNS

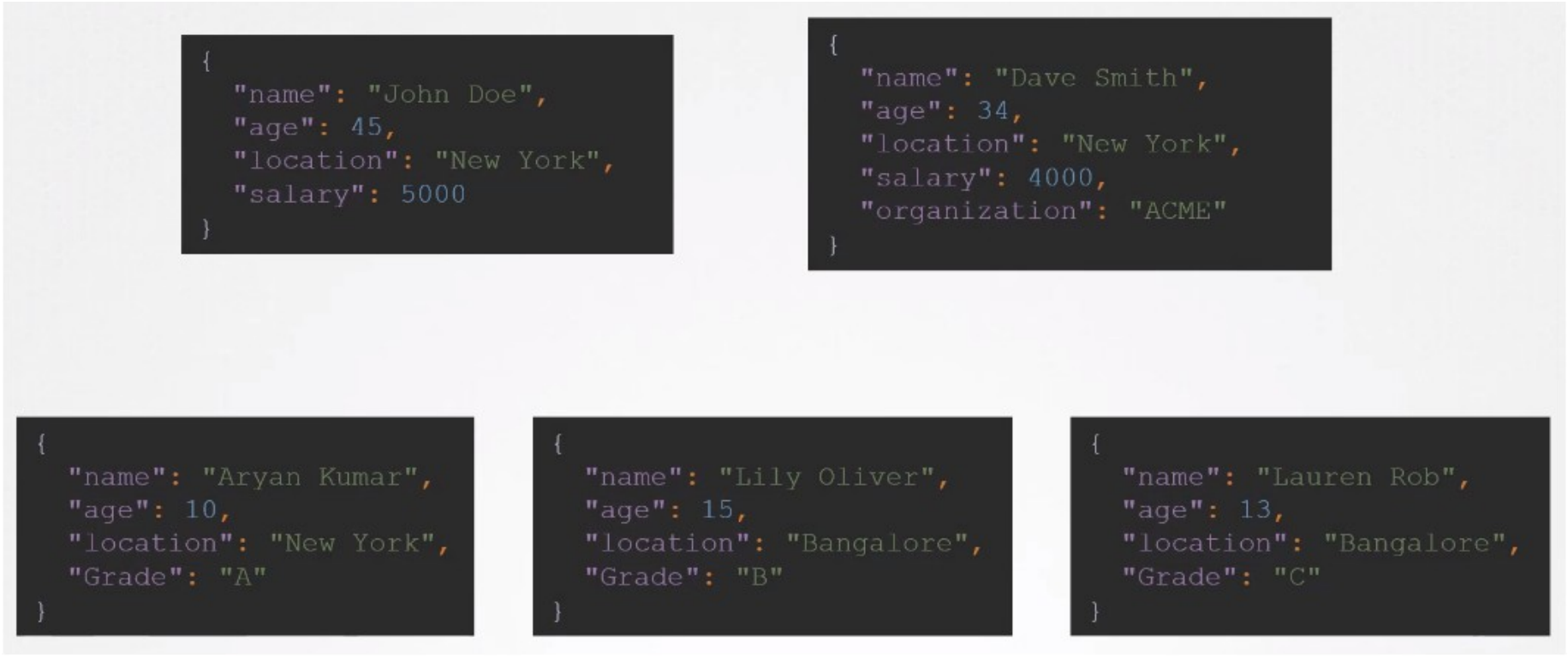
ETCD Overview

- **What does ETCD mean?**
  - "**Etcd**" **stands** for "/etc distributed"; it **is** meant to be a highly reliable configuration mechanism.

- **What is ETCD?**
  - **Etcd** is an open-source distributed key-value store that is simple, secure, fast and serves as the backbone of distributed systems by providing a canonical hub for cluster coordination and state management.

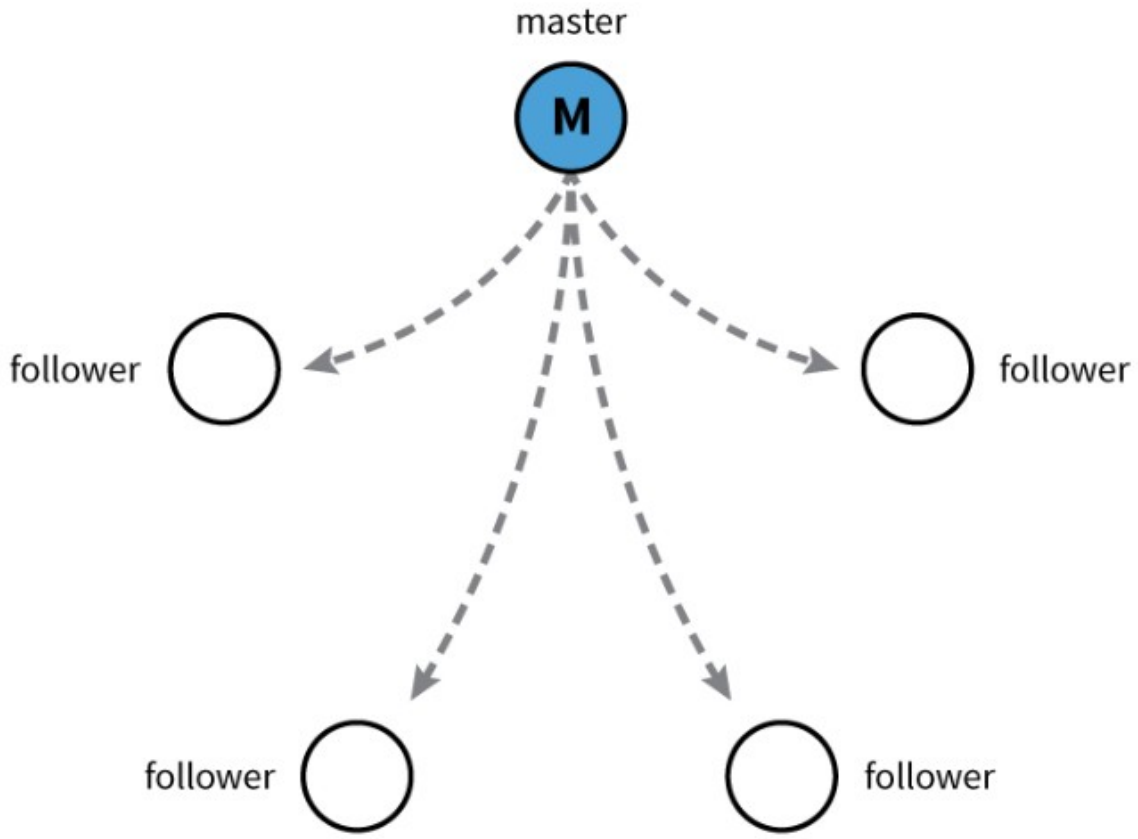
Key-Value Store

- **Data Format**
- Jason
- Yaml



ETCD Overview

- etcd is a distributed key/value store which is designed to have no single point of failure due to its multiple node architecture.



ETCD Overview

- ETCD acts as the cluster datastore; providing a strong, consistent and highly available key-value store used for persisting cluster state.
- When ETCD starts it starts a services that listens on port 2379 by default
- The etcd database uses a command line client to store and retrieve key value pair (./etcdctl get | set). We don’t use this command line client in Transformation Hub
- Store the following information
  - Nodes
  - PODs
  - Configs
  - Secrets
  - Accounts
  - Roles
  - Bindings
  - Others

ETCD Overview

- ETCD configuration file: /opt/arcsight/kubernetes/manifests/ etcd.yaml
- ETCD data is located at: /opt/arcsight/kubernetes/data/etcd/data/member
  - snap: The **snap file** format is a single compressed filesystem (based on squashfs format)
  - Wal: The write-ahead log or "**wal**" **file** is a roll-forward journal that records transactions that have been committed but not yet applied to the main database

- In a highly available cluster you will have multiple etcd instances spread across the master nodes.

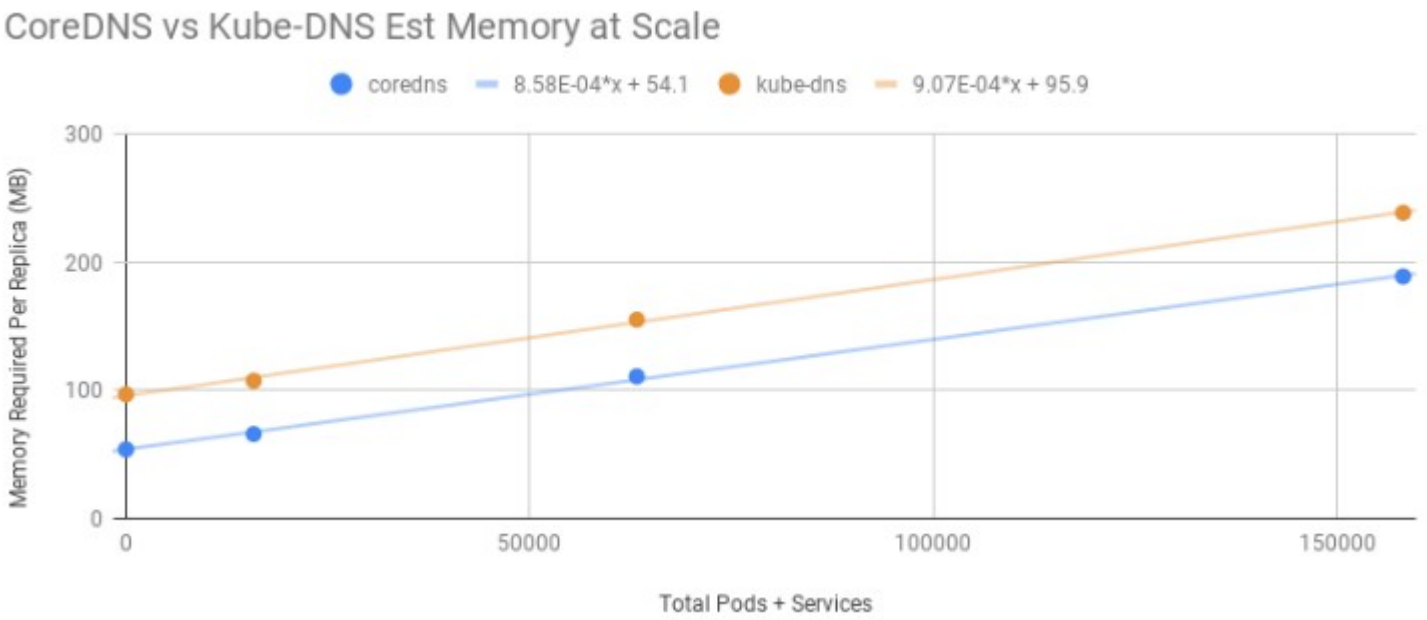
ETCD Overview

- Every time you execute the kubectl command you are communicating directly to the etcd server
- Every time there is a change in the cluster, such as a new pod, node, config, etc. the change is not complete until the change is recorded in the etcd database
- **Raft** is a distributed consensus algorithm. ... **Raft works** by electing a leader in the cluster. The leader is responsible for accepting client requests and managing the replication of the log to other servers. The data flows only in one direction: from leader to other servers

CoreDNS

CoreDNS is a flexible, extensible DNS server that **can** serve as the Kubernetes cluster DNS. It serves as the default primary service discovery mechanism for Kubernetes

- CoreDNS was released On June 14, 2017, CoreDNS-008
- CoreDNS is a single container per instance VS kube-dns which uses three
- CoreDNS is multithreaded Go VS kube-dns uses dnsmasq which uses single threaded C.
- CoreDNS is a DNS server/forwarder meaning CoreDNS can also forward DNS queries for external DNS names to DNS servers outside of that network.



CoreDNS in Transformation Hub

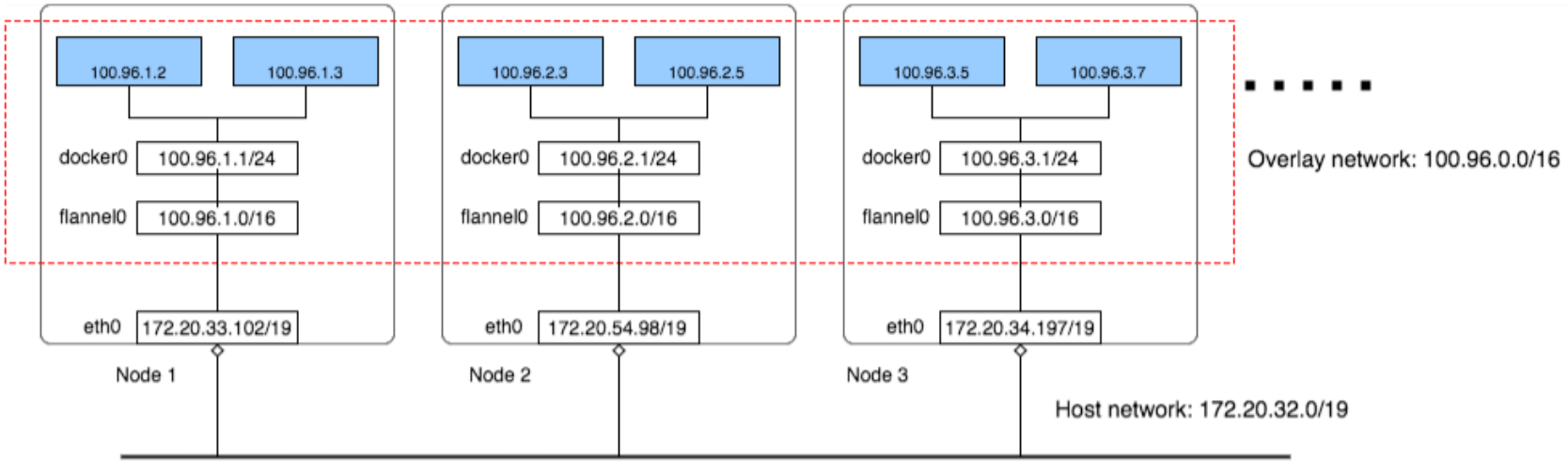
- We have one container CoreDNS instance per master node

```
kube-system   coredns-8lll9   1/1   Running   0   34d   172.16.64.2   thm2.svs.swinfra.net
kube-system   coredns-cnxcn   1/1   Running   0   34d   172.16.79.5   thm3.svs.swinfra.net
kube-system   coredns-lwhjc   1/1   Running   1   34d   172.16.4.20   thm1.svs.swinfra.net
```

- This is the configuration file: /opt/arcsight/kubernetes/objectdefs/coredns.yaml
- Activity of the coredns can be found in these logs: /var/log/containers/coredns-lwhjc_kube-system_coredns-############.log
- Or
- for p in $(kubectl get pods --namespace=kube-system -l k8s-app=kube-dns -o name); do kubectl logs --namespace=kube-system $p; done

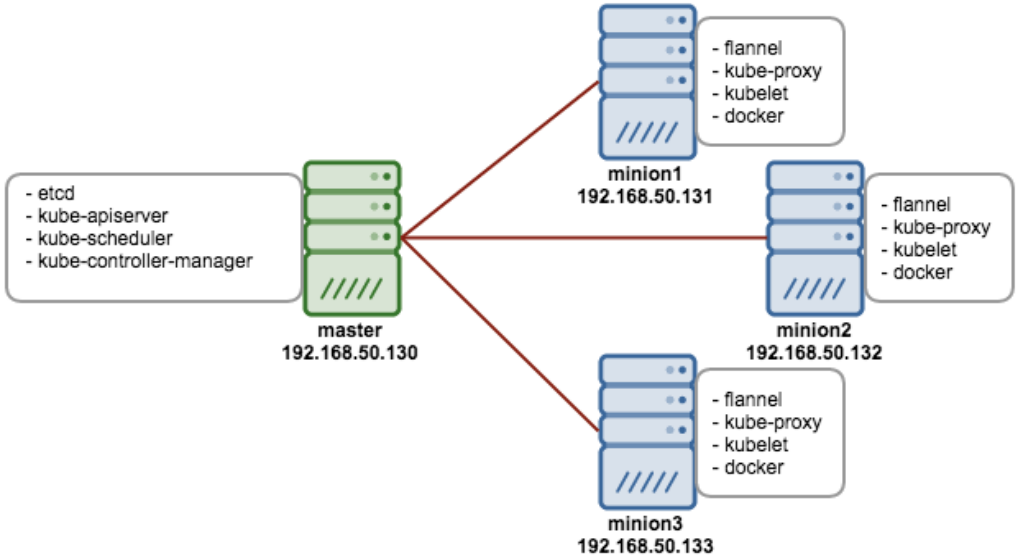
Flannel

- With Docker, each container is assigned an IP address that can be used to communicate with other containers on the *same* host. For communicating over a network, containers are tied to the IP addresses of the host machines and must rely on port-mapping to reach the desired container.
- Flannel is created by CoreOS for K8s. Flannel is a virtual network that gives a subnet to each host for use with container runtimes. Each address corresponds to a container so that all containers in a system may reside on different hosts.
- When the daemon instantiates the container, it assigns a unique network address, connecting it to a virtual Ethernet bridge for communication between containers.
- In the case of Docker, the bridge is called docker0. All containers in the system communicate with each other by directing packets to docker0, which then forwards those packets through the subnet automatically.



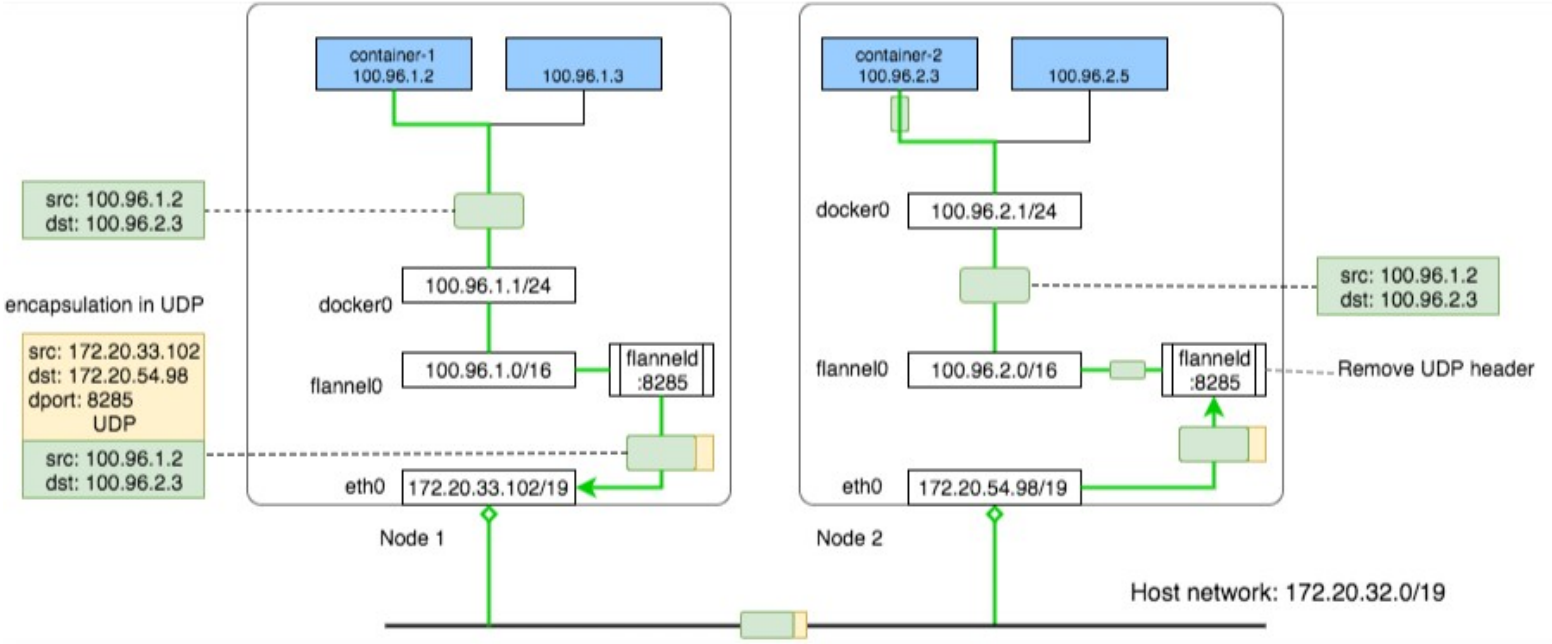
Flannel

- Flannel is responsible for providing a layer 3 IPv4 network between multiple nodes in a cluster. Flannel does not control how containers are networked to the host, only how the traffic is transported between hosts. However, flannel does provide a CNI plugin for Kubernetes and a guidance on integrating with Docker.
- Flannel uses the open source etcd key-value store to record the mappings between the addresses assigned to containers by their native hosts, and their addresses in the overlay network.



Flannel under the hood

- Kubernetes does not provide any default network implementation, rather it only defines the model and leaves to other tools to implement it. There are many implementations nowadays, flannel is one of them and one of the simplest.
- By this design, each container has it's own IP address, all fall into the overlay subnet 100.96.0.0/16. The containers inside the same host can communicate with each other by the docker bridge docker0, which is simple so I will skip in this article. To communicate across hosts with other containers in the overlay network, flannel uses kernel route table and UDP encapsulation to achieve it



Flannel in Transformation Hub

kubernetes

- **# ps -aux | grep flanneld**
- 27992 0.1 0.1 493192 19560 ?   Sl   Jan25 44:35 /opt/bin/flanneld
- -etcd-endpoints https://thm1.svs.swinfra.net:4001
- -etcd-cafile /etc/flannel/ssl/ca.crt
- -etcd-certfile /etc/flannel/ssl/server.crt
- -etcd-keyfile /etc/flannel/ssl/server.key
- -etcd-prefix /coreos.com/network

- **# netstat -atup | grep flanneld**
- tcp   0   0 thm1.svs.swinfra.:43810 thm1.svs.swinfra:newoak ESTABLISHED 27992/flanneld
- tcp   0   0 thm1.svs.swinfra.:58202 thm1.svs.swinfra:newoak ESTABLISHED 27992/flanneld

Kube-API server Overview

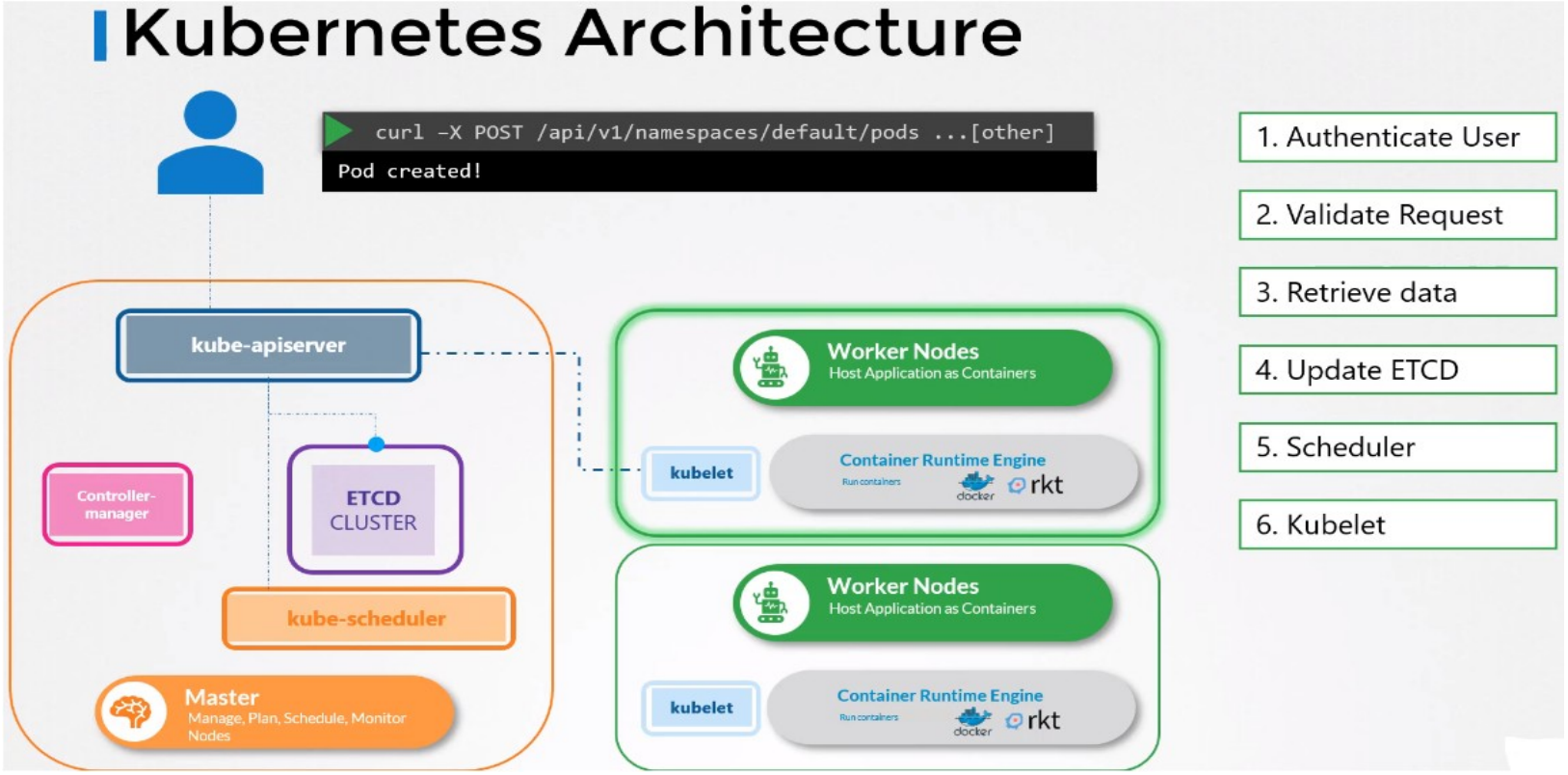
kubernetes

- What is kube-API server?

  - It is the primary management component in K8s
  - The kubectl command reaches out to the kube-api server
  - The kube-api authenticates and validates it, then retrieves the data from the etcd cluster and response back with the information
  - The kube-api configuration file is found in /opt/arcsight/kubernetes/manifests/kube-apiserver.yaml

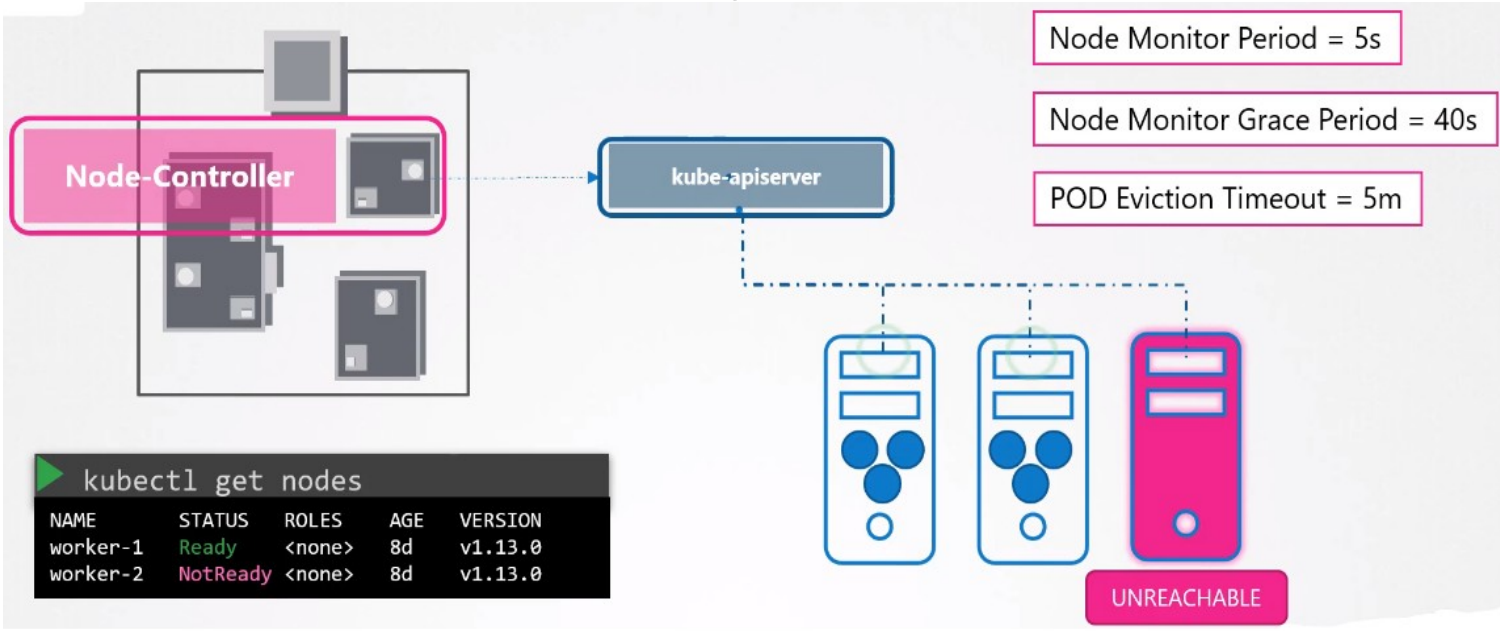
Kube-API server Overview

- How the kube-api works?



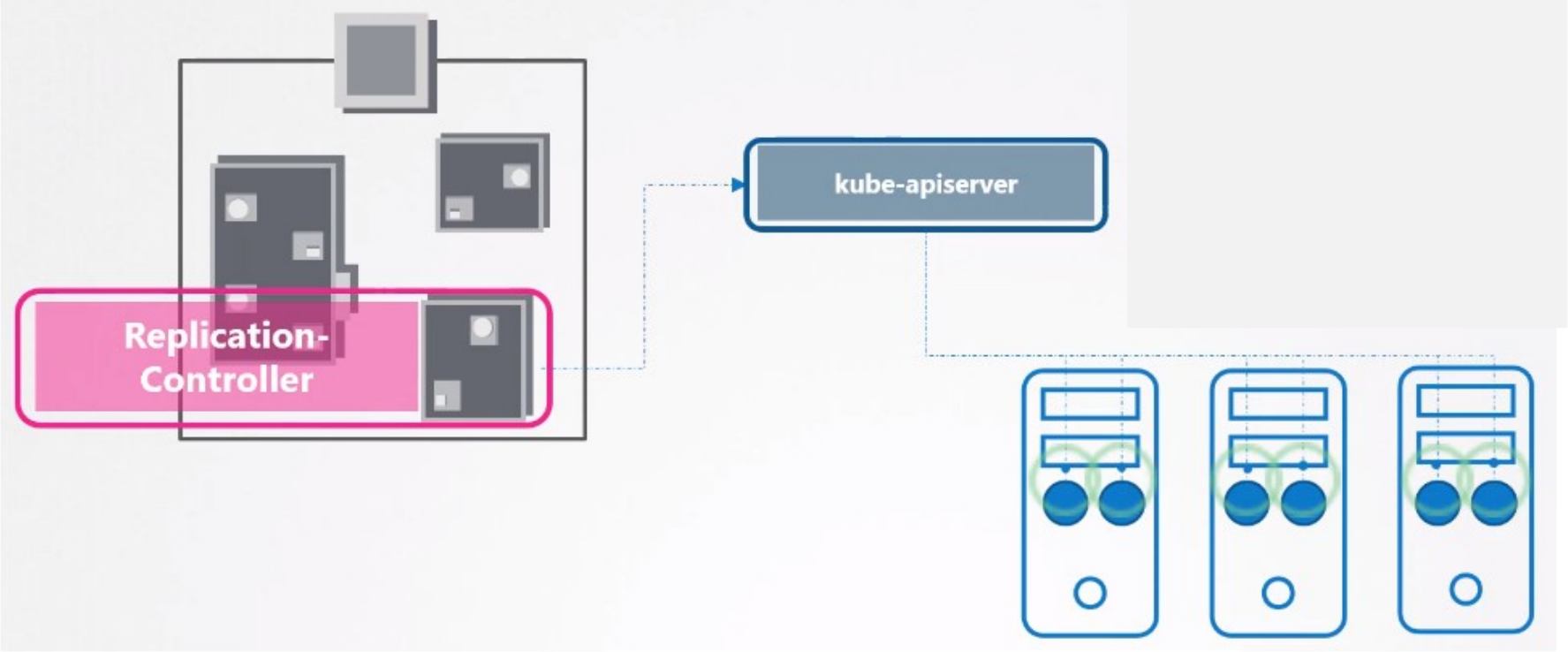
Kube Controller Manager Overview

- What is the Kube Controller Manager?
  - Process that monitors the state of different components
  - You can check the process by running # ps -aux | grep controller-manager
- Node-Controller
  - Checks the status of the nodes every 5 seconds



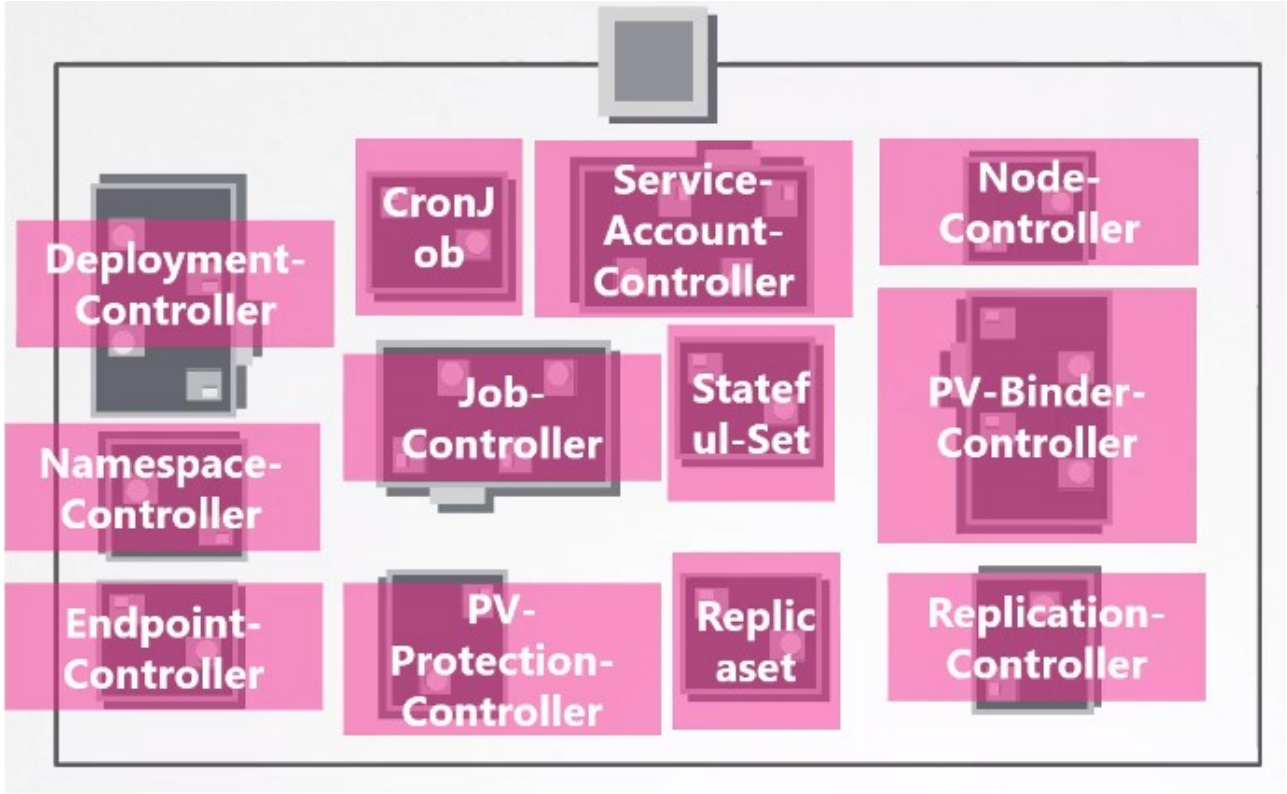
Kube Replication Controller Overview

- What is the Kube Replication Controller?
  - Process that monitors the right number of replicas sets
  - If a pod dies it creates another one



Kube Controller Manager Overview

- What is the Kube Replication Controller?
  - Process that monitors the right number of replicas sets
  - If a pod dies it creates another one



ETCD Overview

kubernetes

- **What is kube-scheduler?**
  - It is a process responsible to scheduling pods in the nodes. It does not place the pod in the node
  - It filters out the nodes based on the cpu number
  - Ranks the nodes based on resources after placing the pod in a node
- The configuration file is found in **/opt/arcsight/kubernetes/manifests/kube-scheduler.yaml**
- **# ps -aux | grep scheduler**

  thinsta+ 22455 1.3 0.3 949776 61676 ? Ssl Jan25 551:29 /hyperkube scheduler
  --address=127.0.0.1
  --bind-address=127.0.0.1
  --master=https://thm1.svs.swinfra.net:8443
  --leader-elect=true
  --v=1
  --logtostderr=true
  --kubeconfig=/etc/kubernetes/ssl/kubeconfig
  --authentication-kubeconfig=/etc/kubernetes/ssl/kubeconfig
  --authorization-kubeconfig=/etc/kubernetes/ssl/kubeconfig

Kubelet Overview

kubernetes

What is a kubelet?

- The **kubelet** is responsible for maintaining a set of pods, which are composed of one or more containers, on a local system. Within a Kubernetes cluster, the **kubelet** functions as a local agent that watches for pod specs via the Kubernetes API server.
- **Additional Info**
- # ps -aux |grep kubelet (process running in every cluster node)
- # systemctl status kubelet (service)
- # ls -las /opt/arcsight/kubernetes/data/kubelet
  - -rw------- 1 root root 62 Jan 18 00:55 cpu_manager_state
  - drwxr-xr-x 2 root root 4096 Jan 18 00:55 pki
  - drwx------ 2 root root 4096 Jan 18 00:55 plugin-containers
  - drwxr-x--- 2 root root 4096 Jan 18 00:55 plugins
  - drwxr-x--- 2 root root 4096 Jan 18 00:55 plugins_registry
  - drwxr-x--- 2 root root 4096 Jan 18 00:55 pod-resources
  - drwxr-x--- 22 root root 4096 Feb 20 10:09 pods

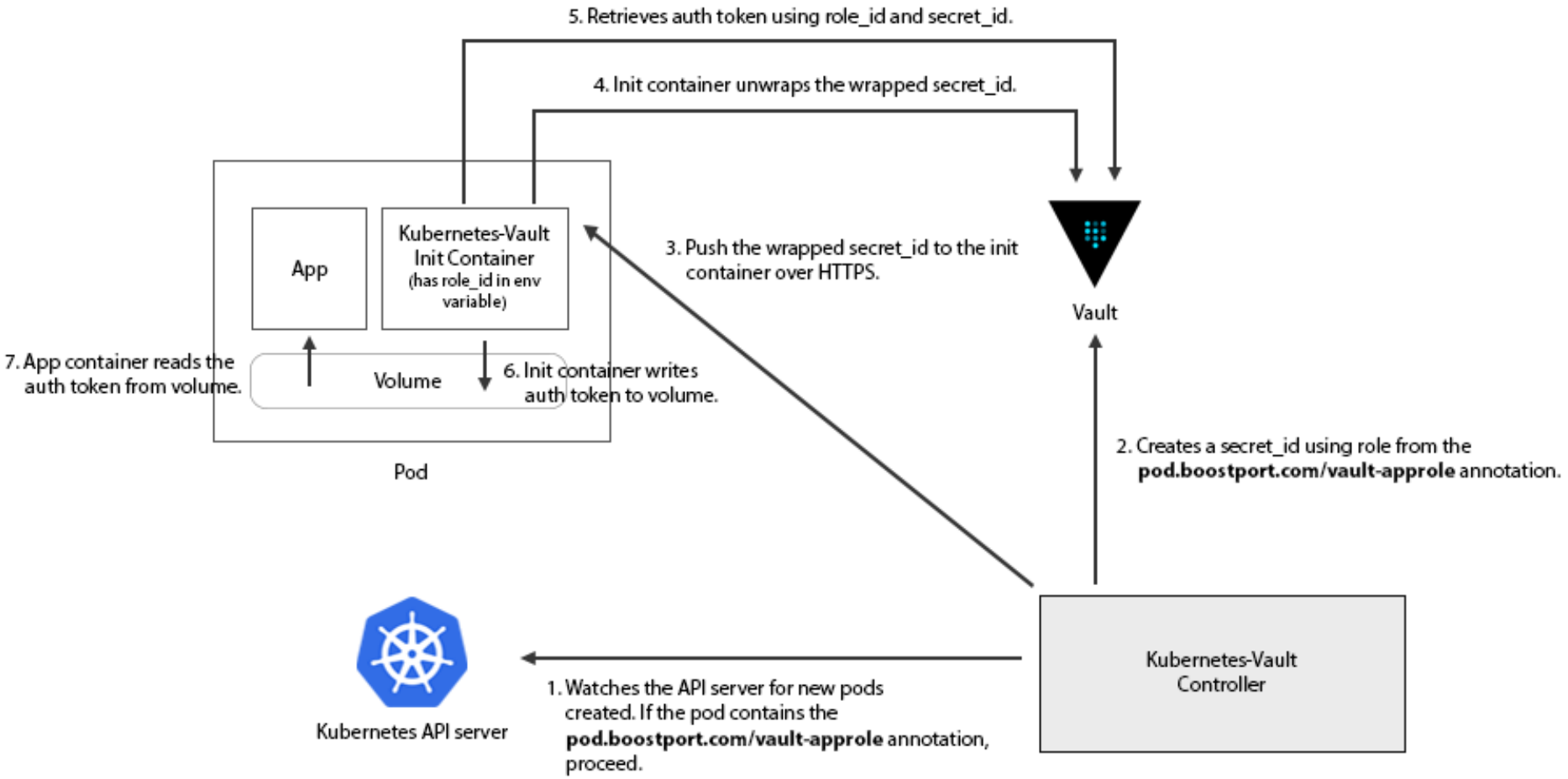
- # /opt/arcsight/kubernetes/bin/kubelet (Kubelet logs)

Vault and Kubernetes-vault

- The ITOM platform uses Hashicorp's Vault solution to provide:
  - Secure storage of sensitive configuration data
  - Generation of cluster-internal certificates
  - PKI backends Vault backends to generate cluster internal certificates
  - Vault Approle authentication mechanism for services to access secure configuration data
  - The Vault root key and unseal shards are stored in ETCD

- The benefits provide:
  - Generic secret Vault backends to store configuration data
  - The data is secured at rest and in-flight
  - Vault to component communication is secured using certificates

Vault and Kubernetes-vault



Thank You.


MICRO®
FOCUS