## Storage Classes

Storage for EDCOP tools will be broken into three parts. Containers can use these three storage areas depending on their needs. Users must configure these three storage classes. The storage types include:

1. local-fast: This storage area is of type HostPath and will be stored in /EDCOP/fast. The purpose of this storage is for users to be able to configure a storage array using faster storage (such as high end SSDs).
2. local-bulk: This storage area is of type HostPath and will be stored in /EDCOP/bulk. The purpose of this storage is for users to be able to configure a storage array containing a large amount data. Generally this will be slower storage.
3. Shared: Users will require a form of shared storage technology using volumes. This can include CEPH, GlusterFS, NFS etc. The storage will use volumes managed by Kubernetes. This storage will be shared by all hosts.

All three of these storage techniques are required to be defined. It is possible that users may not have two separate volumes for local-fast and local-bulk. In this scenario they will still need to define these as storage and simply mount the volume as a single volume with two directories inside of them.

## Networking Classes

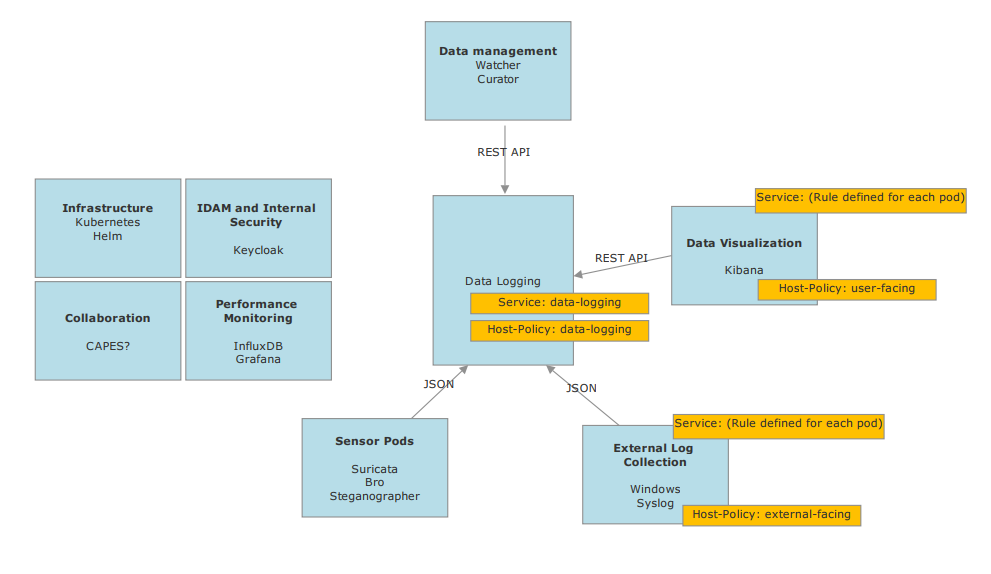
EDCOP Tools requires the use of Multus. Multus allows for containers to utilize multiple networks.

<https://github.com/Intel-Corp/multus-cni>

1. cluster: This will reference a networking technology which allows containers to talk to one another. Currently Calico is used is recommended an tested. The networking should support network policies which allow rules to be defined on how containers and pods can communicate with one another.
2. inline-1: A single container can connect to this to provide inline like capability, at some point it will be desired to allow for chaining inline tools together. The networking technology must pass Layer 2 traffic. Recommended is either MacVLAN to a physical NIC or using SR-IOV.
3. inline-2: This will be the other side of inline tools passing Layer-2 traffic.
4. passive-1: This will allow tools to connect and receive passive traffic. This must be a layer 2 technology that allows promiscuous mode to be configured.
5. Data Ingest Pod Design

## Pod Design

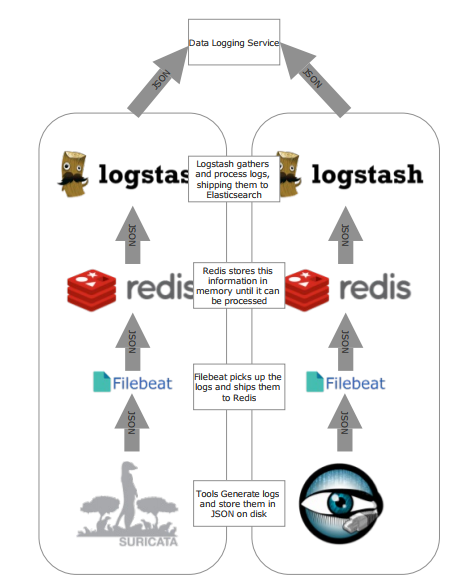
Pods should aim to be as self-sufficient as possible. The idea is to be able to provide a set of pods which are selectable by a user and can be deployed. Data Ingest pods can be inline tools, passive tools, external log collection. Pods will be one of a few categories with some rules defined how each should communicate.



Pods can have multiple containers but when traffic leaves the pod, they must follow the standard provided above. This will allow for components to be swapped in and out as desired.

Pods live on a single node and generally containers inside of a pod will communicate through localhost on a port. This means that containers will only listen internally on localhost by default.

TBD: Do we want to require encryption internal to localhost? The benefit I see is less confidentiality and more authentication.



Pods which require other pods to communicate with them must define a service in order to allow for communication inbound. Services which are externally accessible outside of the cluster must define an ingress or a nodePort to allow for inbound network communication to the cluster. Generally, all user facing services (i.e. Kibana) should utilize an ingress and any non-user facing services should utilize a node port.

There are two methods of deploying pods:

Daemonset – When a daemonset is created any nodes that have a particular tag or all nodes if no tag is specified will have the logs deployed to them.

Deployment – This allows for the cluster to deploy a specified number of pods. The pods can be manually scaled up and down if desired. Traffic needs to be load balanced to multiple hosts if this is required.

## Container design

Edcop has a few rules for containers that all containers must follow to be accepted as part of the platform.

1. Long running containers must not run as root with the exception of infrastructure containers and init containers. These containers with will live in the kube-system namespace. All containers must run as a limited user.
2. Containers must not run with privileged! Some containers may need additional privileges, in which case these must be specified individually using capabilities built into Linux. See: <http://man7.org/linux/man-pages/man7/capabilities.7.html>
3. Container debug and audit log output should output to STDOUT which is a Docker best practice. Containers will die which means the logs are lost. If logs are outputted to STDOUT it is possible to pick up these logs with a stack driver and save them to the data backend. It also makes troubleshooting easier as kubectl logs will display the output. It is possible to create symlinks to /dev/stout and /dev/stderr when application do not directly support this. See here for logging driver options: https://docs.docker.com/engine/admin/logging/overview/#configure-the-default-logging-driver
4. Containers should not run multiple processes. Each process should live in its own world. Kubernetes monitors these processes health and has mechanisms to restart containers as needed. When multiple processes are needed Kubernetes has a number of design patterns that can be used documented here: <http://blog.kubernetes.io/2015/06/the-distributed-system-toolkit-patterns.html>

We are evaluating at Clear Containers as a method of isolating “high risk” containers such as sensors.

<https://clearlinux.org/containers>

## Host Policies

Host policies in Kubernetes are controlled by tags. It is possible to define what containers are able to communicate with one another.

TODO: Define tags for

## Namespace

Todo: General gist of it is that we have kube-system which is required and default. Right now I am leaning toward having:

Kube-system

Data

Sensors and logging infrastructure.

The high level goal is that we should be able to have multiple tenants with the same set of sensors. This will all go into different indices in Elasticsearch and then an ACL will be applied.