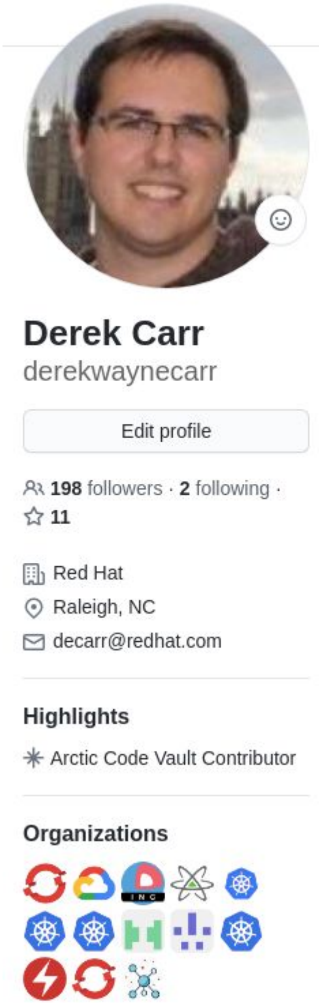
Red Hat Developer | DevNation

Pods in the Kubernetes and OpenShift Cosmos



Derek Carr
derekwaynecarr

198 followers · 2 following ·
☆ 11

Red Hat
Raleigh, NC
decarr@redhat.com

Highlights

Arctic Code Vault Contributor

Organizations

Kubernetes

- Steering Committee Member
- Co-Chair
  - SIG Architecture
  - SIG Node
  - WG Resource Management (Emeritus)

OpenShift

- Distinguished Engineer @Red Hat
- Member of OpenShift Architecture Team

Application Requirements

1. Developer
2. Imagination
3. Energy

Application Requirements

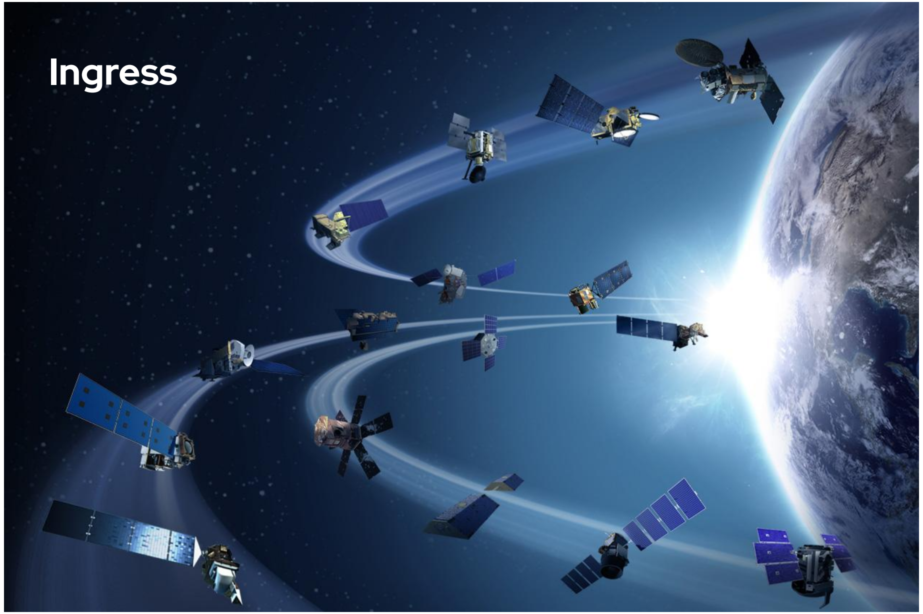
1. Developer
2. Imagination
3. Energy
4. Hybrid Cloud

Application Requirements

1. Developer
2. Imagination
3. Energy
4. Hybrid Cloud
    a. Operations

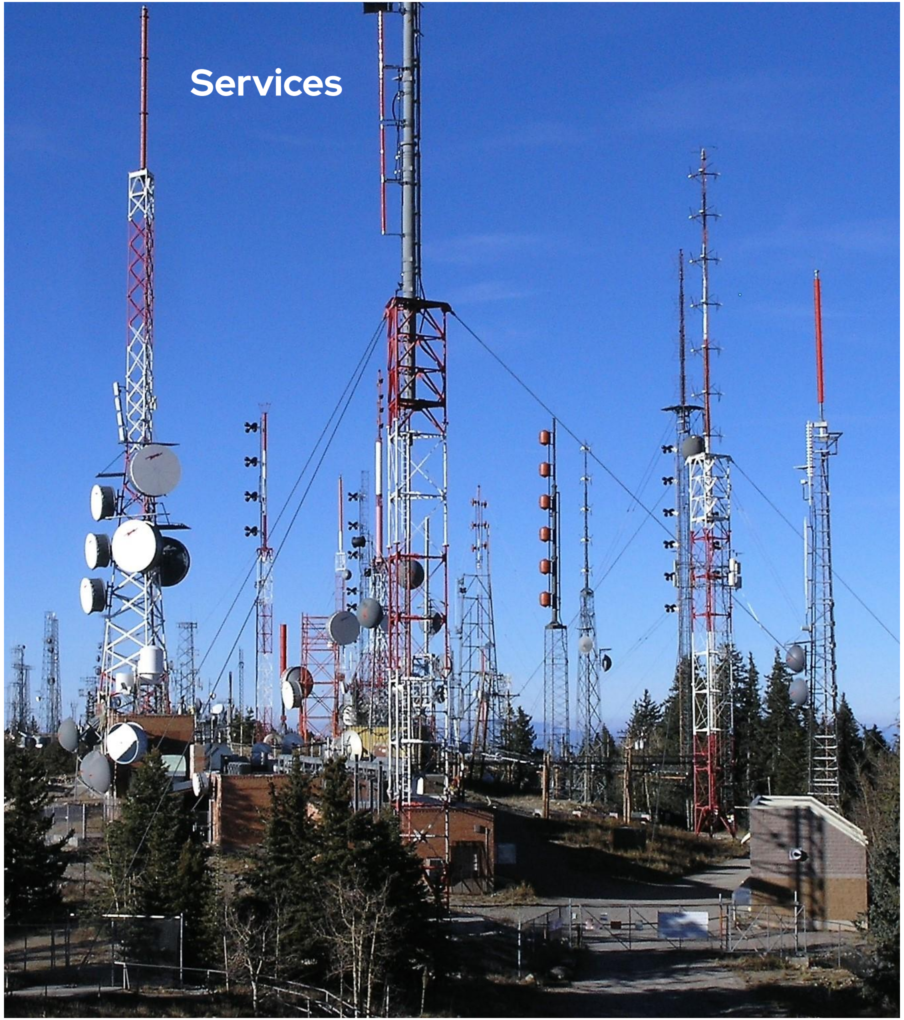
Application Requirements

1. Developer
2. Imagination
3. Energy
4. Hybrid Cloud
    1. Operations
    2. Cluster



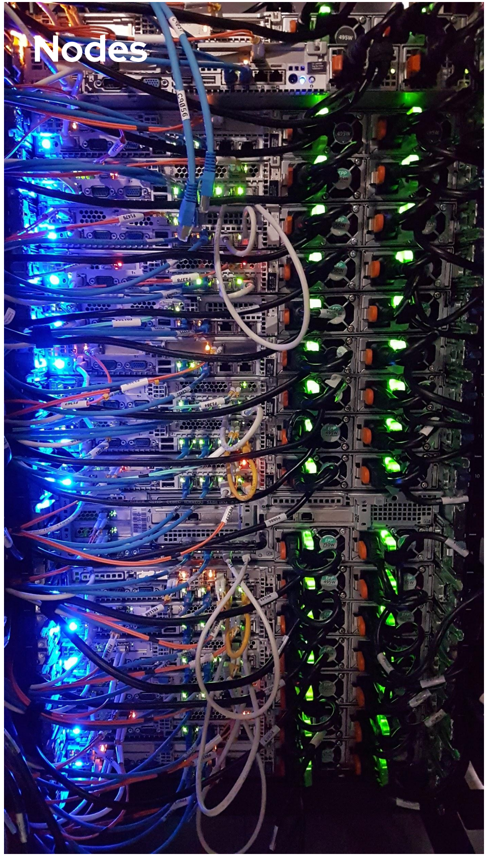
Application Requirements

1. Developer
2. Imagination
3. Energy
4. Hybrid Cloud
    a. Operations
    b. Cluster
    c. Ingress



Application Requirements

1. Developer
2. Imagination
3. Energy
4. Hybrid Cloud
    a. Operations
    b. Cluster (Location)
    c. Ingress
    d. Services



Application Requirements

1. Developer
2. Imagination
3. Energy
4. Hybrid Cloud
   a. Operations
   b. Cluster (Location)
   c. Ingress
   d. Services
   e. Nodes

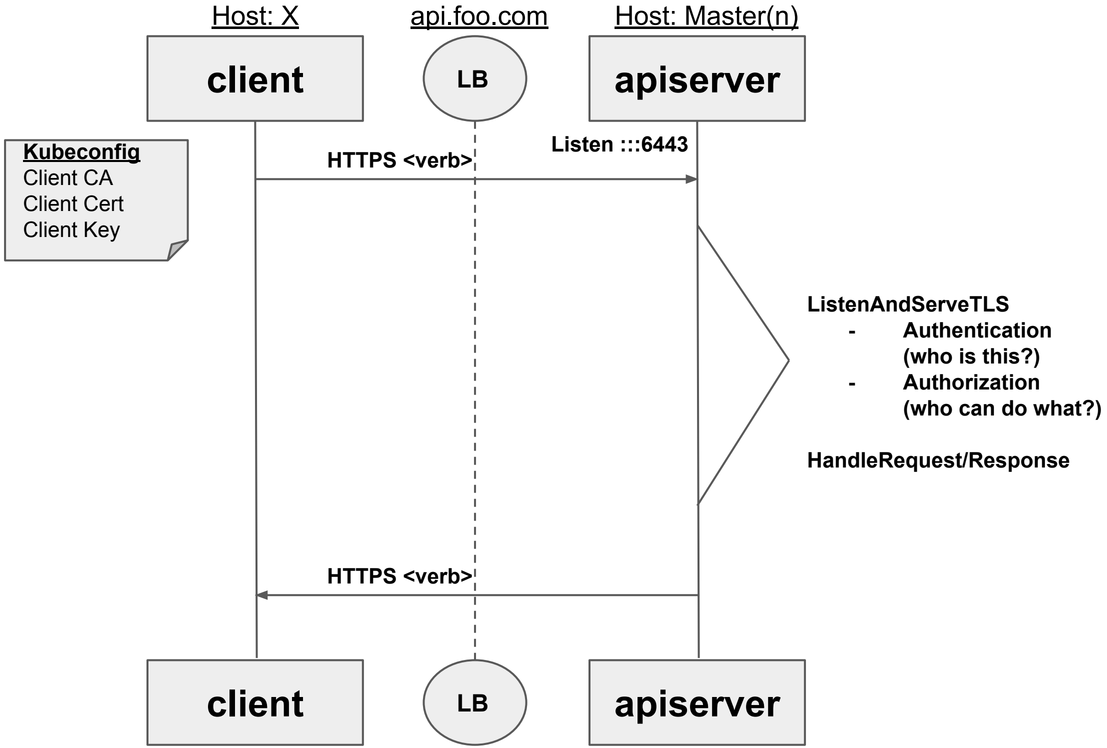
Application Requirements

1. Developer
2. Imagination
3. Energy
4. Hybrid Cloud
   a. Operations
   b. Cluster (Location)
   c. Ingress
   d. Services
   e. Nodes
   f. Pods

But how do Pods actually work?

```
$ kubectl run -i -t busybox  --image=busybox --restart=Never
```

Network Flow - Client to Control Plane



External Load Balancer

An external load balancer (api.foo.com) handles all traffic external to cluster.
It balances kube-apiserver traffic across N hosts.

All end-user API interaction is directed against the api-server component.

API Server Configuration

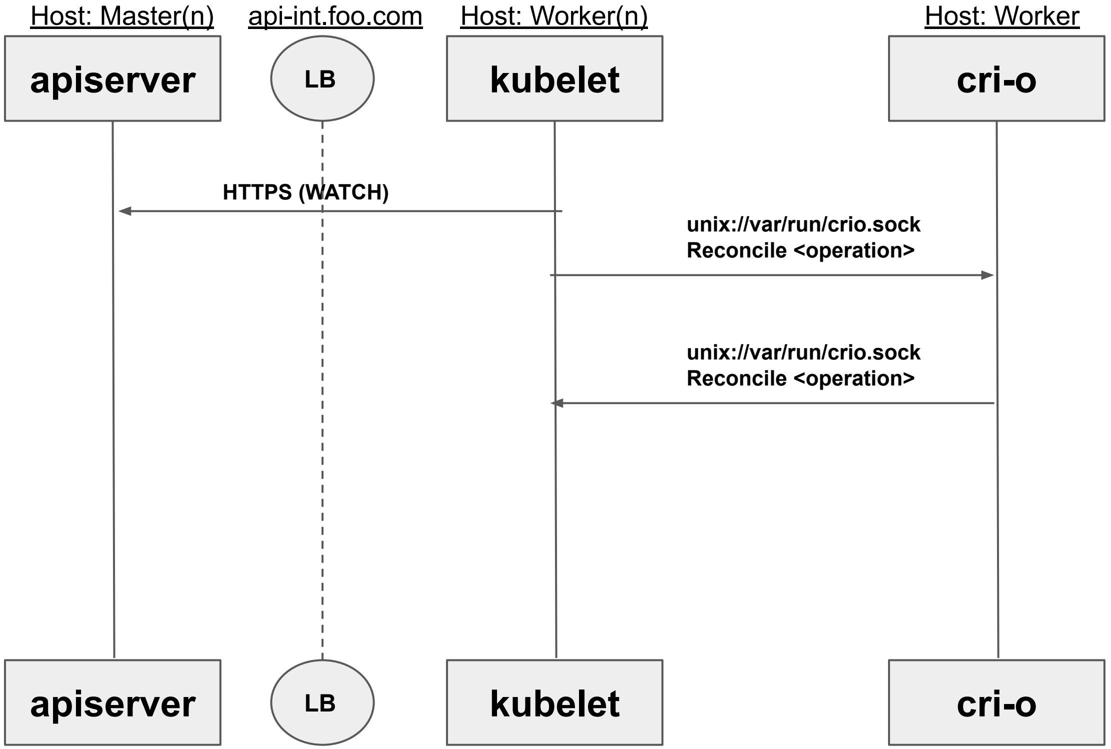
Client CA & Serving Certs - ca can be provided by admin, serving certs can be provided by admin and configured for SNI

TLS Security Profiles - Cipher Profiles for *old, intermediate, modern* per Mozilla, *custom* profile definition available for customer specific cipher lists. TLS 1.2 (by default), 1.3 (configurable)

Allowed CORS Origins - regex hosts allows access using CORS headers.

Encryption - Resources encrypted in data store layer

Node (no pods)



Host: Worker (Linux FS State)

Filesystem:
/var/lib/kubelet/

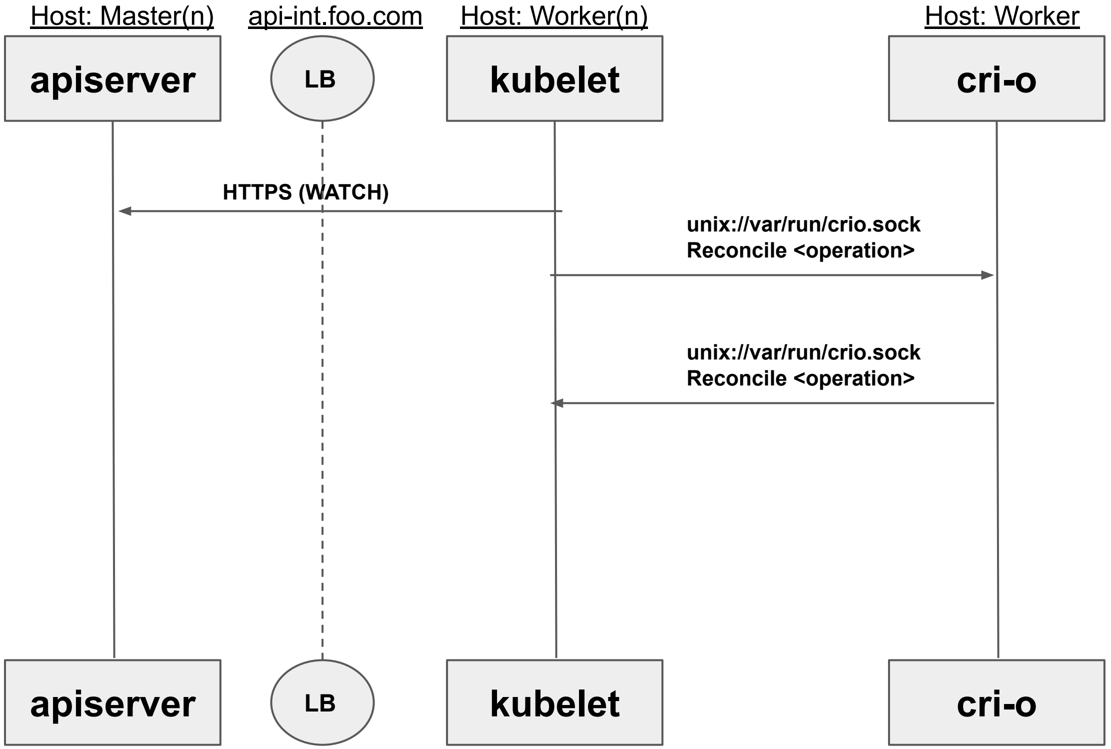
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

- /kubepods.slice
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet launches and creates cgroup hierarchy per quality of service tier. Kubelet watches API server for bound pods to run. It reconciles local node host to current state in a constant loop. It interacts with cri-o to determine status of running containers and image filesystem.

Kubelet to CRI-O communication



Host: Worker (Linux FS State)

Filesystem:
/var/lib/kubelet/

Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

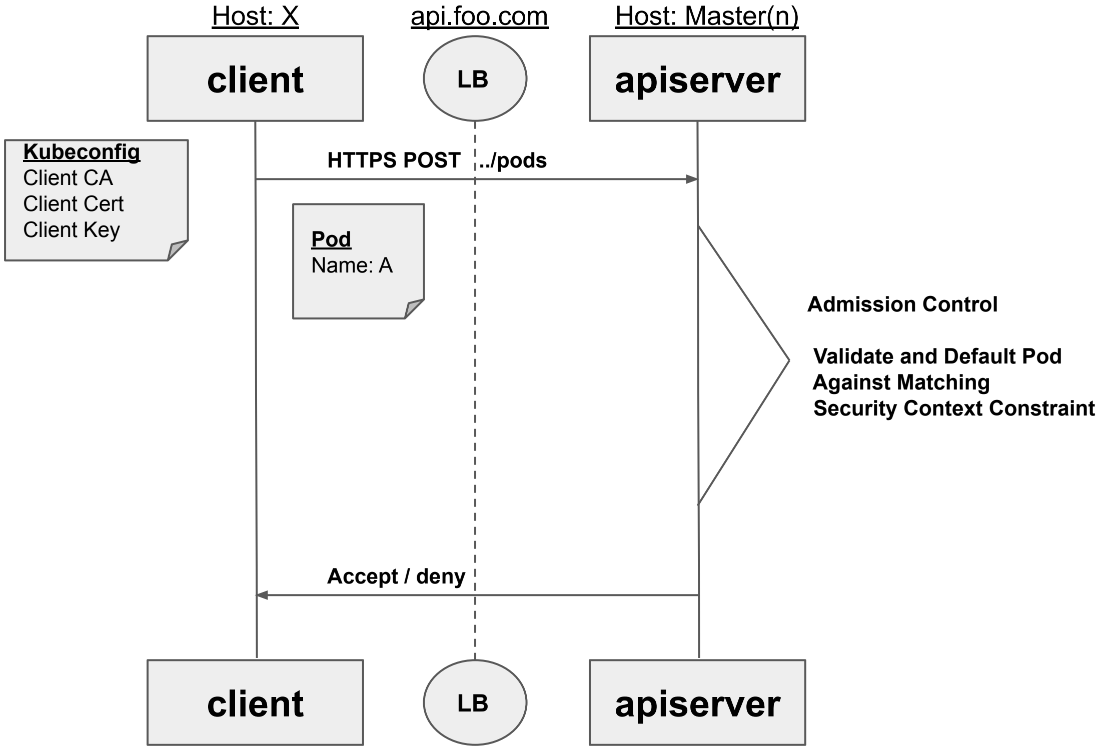
- /kubepods.slice
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet communicates with cri-o runtime over unix socket.

IMPORTANT
The crio.sock is protected by a SELinux label that is NOT accessible by default SELinux context for end-user containers.

SELinux labels for key processes and sockets:
kubelet *system_u:system_r:unconfined_service_t:s0*
crio *system_u:system_r:container_runtime_t:s0*
crio.sock *system_u:object_r:container_var_run_t:s0*
<example user container processes>
system_u:system_r:container_t:s0:c14,c24

Pod - Restricting access by default



OpenShift security feature:

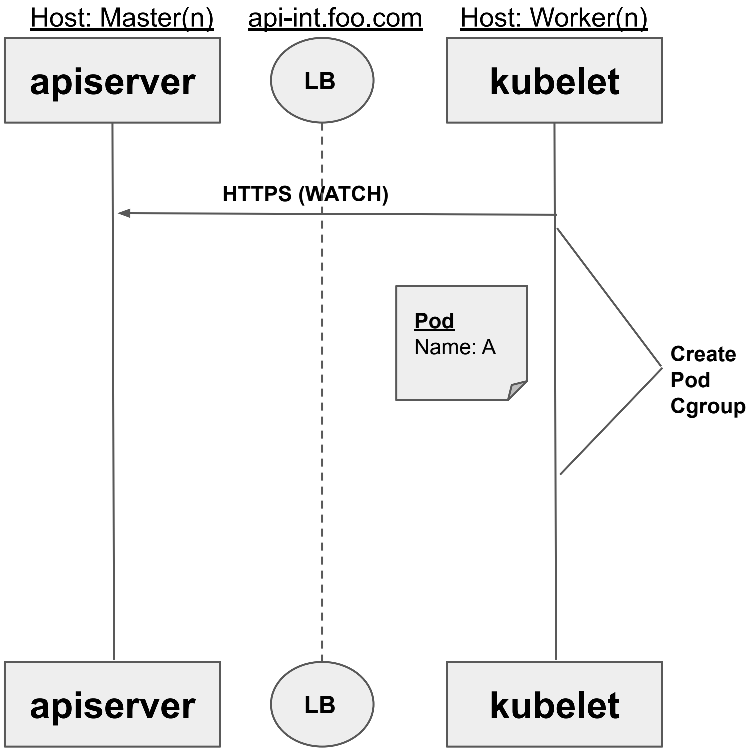
Security Context Constraints (SCC)
Each pod is validated prior to persistence against a set of constraints that control ability for pod to run privileged, add/remove capabilities, selinux modes, run as user restrictions, fs group restrictions, supplemental group restrictions, readonly rootfs, and volumes allowed for use.

Available out of box (custom profiles are supported)
Anyuid
Hostaccess
Hostmount-anyuid
Hostnetwork
Node-exporter
Nonroot
Privileged
Restricted* (default)

The default SCC denies access to all host features and requires pods to be run with a UID, and SELinux context that are allocated to the pod's namespace in OpenShift. This is the most restrictive policy and is used by default for authenticated users by default.

<scheduler magic>

Kubelet sees pod - create cgroups



Host: Worker (Linux FS State)

Host Filesystem:
/var/lib/kubelet/

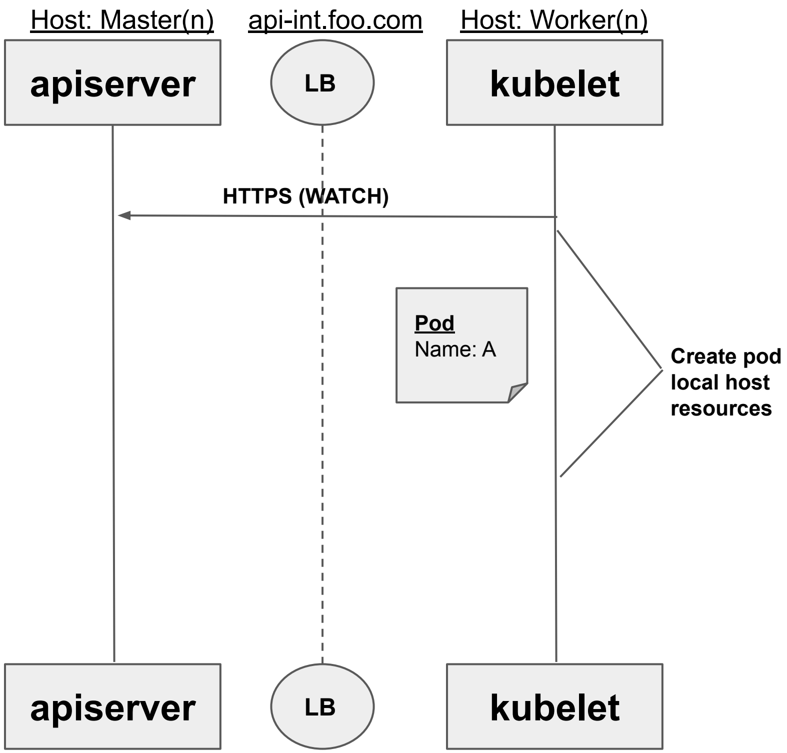
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

- /kubepods.slice
  - ***/podA.slice***
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet sees it should run Pod A. Kubelet creates cgroup for Pod A under QoS subtree for required resources

Kubelet sees pod - create host resource



Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

- /podA
  - ***/etc-hosts***
  - ***/volumes***

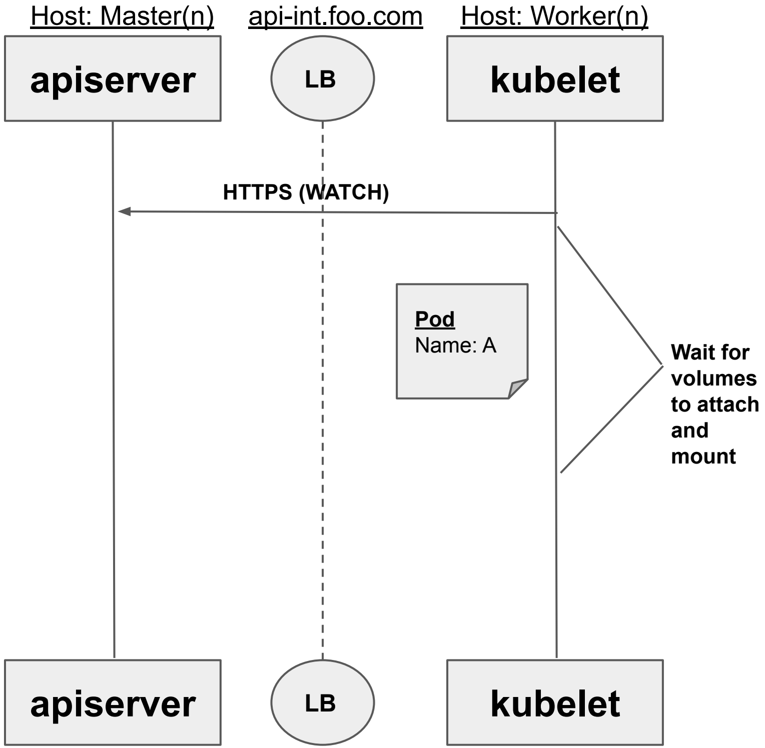
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

- /kubepods.slice
  - /podA.slice
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet creates pod A etc-hosts file for DNS configuration
Kubelet creates pod A data directories on local host (emptyDir, etc.)

Kubelet sees pod - volumes



Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

- /podA
  - /etc-hosts
  - /volumes
    - **secret-1 (tmpfs)**
    - **configMap-1**
    - **...**

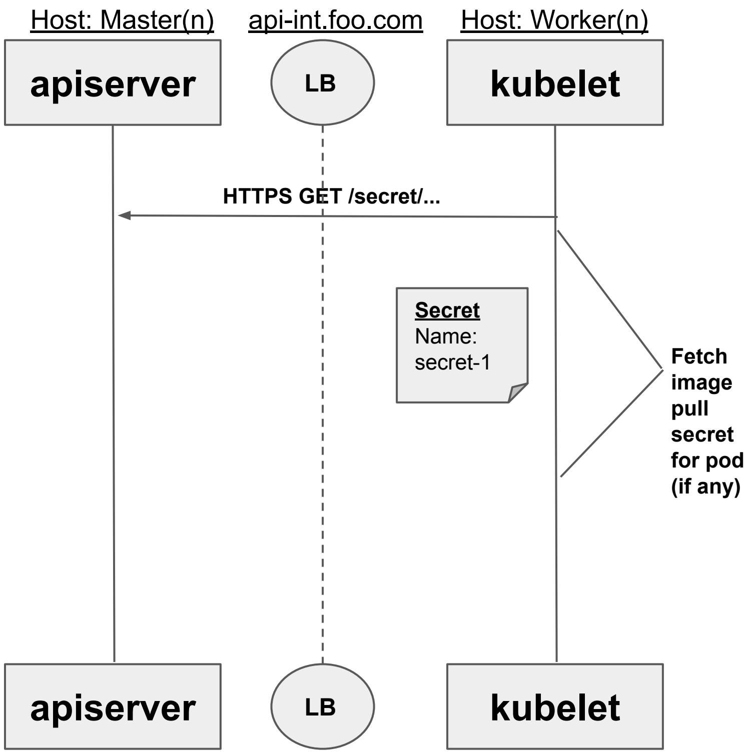
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

- /kubepods.slice
  - /podA.slice
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet waits for volumes to attach and mount for pod from its spec

Kubelet sees pod - fetch pull secrets



Host: Master(n)

api-int.foo.com

Host: Worker(n)

apiserver

LB

kubelet

HTTPS GET /secret/...

Secret
Name:
secret-1

Fetch image pull secret for pod (if any)

apiserver

LB

kubelet

Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

- /podA
    - /etc-hosts
    - /volumes
        - **secret-1 (tmpfs)**
        - **configMap-1**
        - **...**

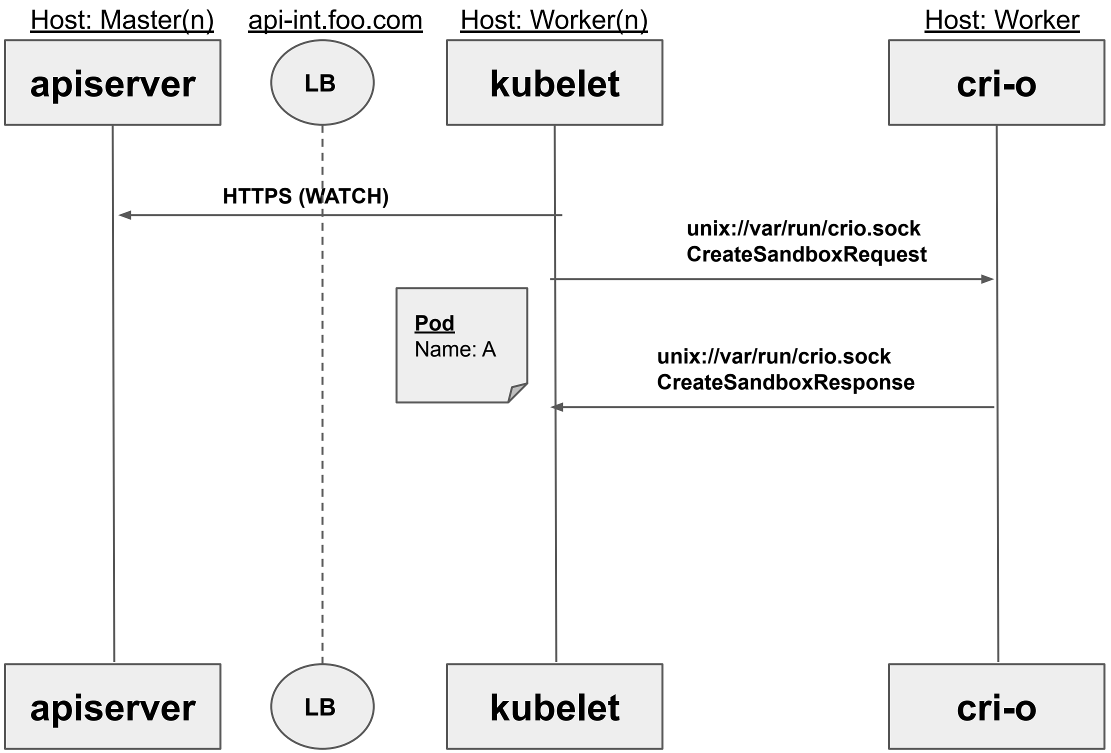
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

- /kubepods.slice
    - /podA.slice
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet fetches pull secret associated with pod (if any) used to pull its container images

Kubelet sees pod - sandbox



Host: Master(n)

apiserver

api-int.foo.com

LB

Host: Worker(n)

kubelet

Host: Worker

cri-o

HTTPS (WATCH)

unix://var/run/crio.sock
CreateSandboxRequest

Pod
Name: A

unix://var/run/crio.sock
CreateSandboxResponse

apiserver

LB

kubelet

cri-o

Host: Worker (Linux FS State)

Host Filesystem

/var/lib/kubelet/pods

- /podA
  - /etc-hosts
  - /volumes
    - secret-1 (tmpfs)
    - configMap-1
    - ...

Containers

- podA sandbox (podIP) (image=pause)

Cgroup Controllers:

cpu, cpuacct, cpuset, memory, hugetlb, pids

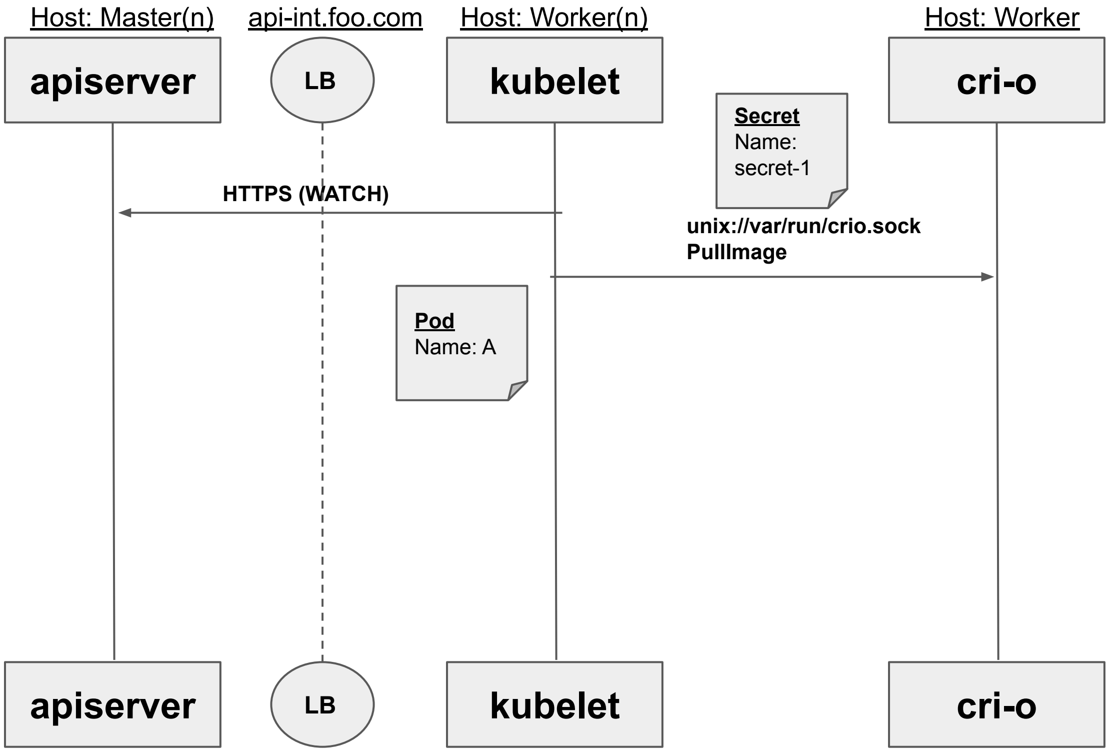
Cgroup Hierarchy

- /kubepods.slice
  - /podA.slice
    - **sandbox**
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet request create sandbox for pod under pod cgroup

Sandbox is a container that holds Linux namespace and IP for all other containers in the pod, it is often referred to as **pause** container

Kubelet sees pod - pull image(s)



Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

- /podA
    - /etc-hosts
    - /volumes
        - secret-1 (tmpfs)
        - configMap-1
        - ...

Containers

- podA sandbox (podIP) (image=pause)

Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

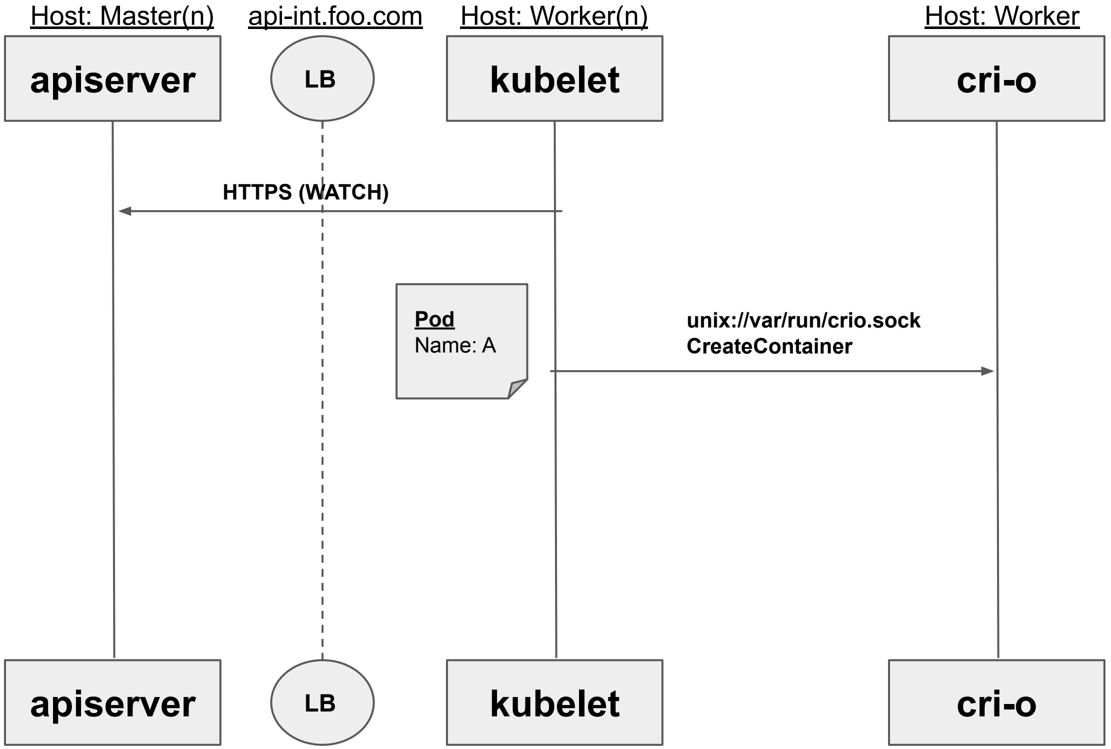
Cgroup Hierarchy

- /kubepods.slice
    - /podA.slice
        - Sandbox
        - **Container 1 … N**
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet requests runtime to pull images.
Image pull policy protects access to image content.
Policy Options: Always, Never, IfNotPresent
Note: Always ensures rights to use image based on pull secret credentials.

Red Hat Developer | DevNation

Kubelet - create container(s)



Container Configuration (Container 1)

Command, Args (to run in the container)
WorkingDir
Envs (env vars for container)
Mounts (mounts available to container)
Devices
LogPath (where logs are stored and rotated)
Stdin/Tty
Resources (derived from pod spec and calculated per container)

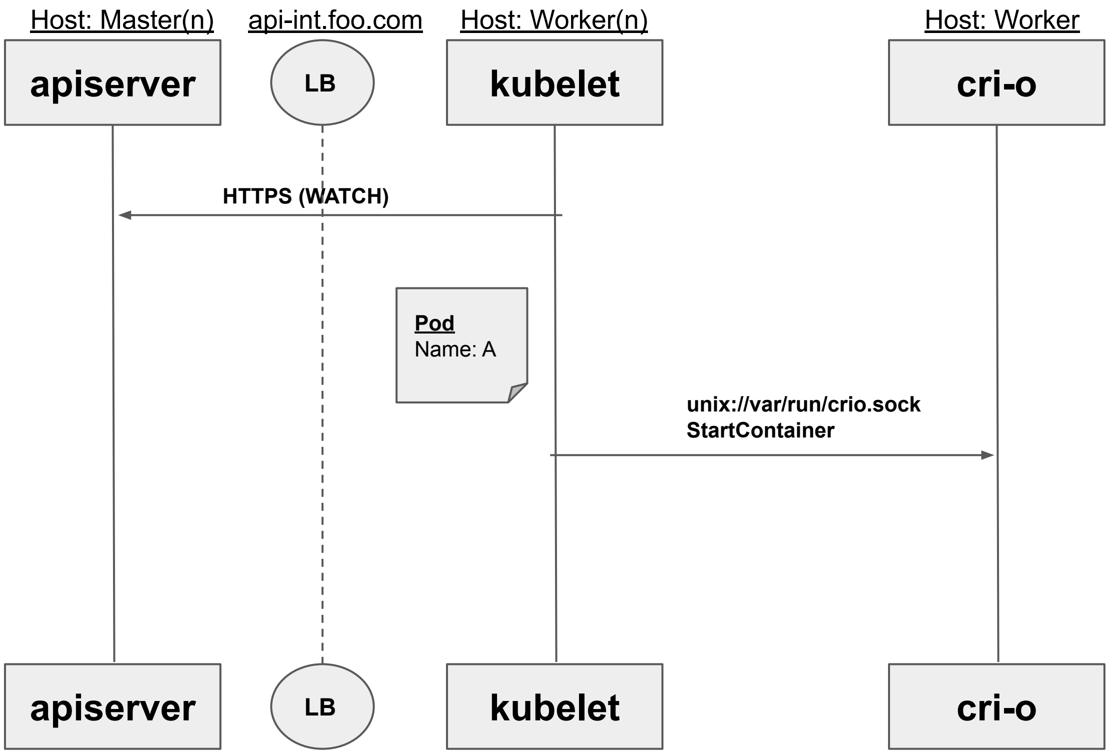
- CPU period, quota, shares
- CPUset (cpu, memory)
- Memory limit (bytes)
- HugePage limits (bytes per page size)
- Oom score

Security Context (derived from pod spec)

- Capabilities (Add / drop)
- Privileged (bool)
- Namespace Options
- SelinuxOptions (User, Role, Type, Level)
- RunAsUser, RunAsGroup (uid/gid to run process)
- RunAsUsername (/etc/passwd in image if used)
- ReadonlyRootfs (bool)
- SupplementalGroups
- Seccomp Profile Path (full path to profile file on host)
- NoNewPrivs (bool)
- MaskedPaths (slice of paths masked by runtime)
- ReadonlyPaths (slice of paths masked as readonly)

Each container has a configuration that tells runtime how to isolate based on pod spec.

Kubelet - start container(s)



Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods
- /podA
  - /etc-hosts
  - /volumes
    - secret-1 (tmpfs)
    - configMap-1
    - **...**

Containers
- podA sandbox (podIP) (image=pause)
- **Container 1… N**

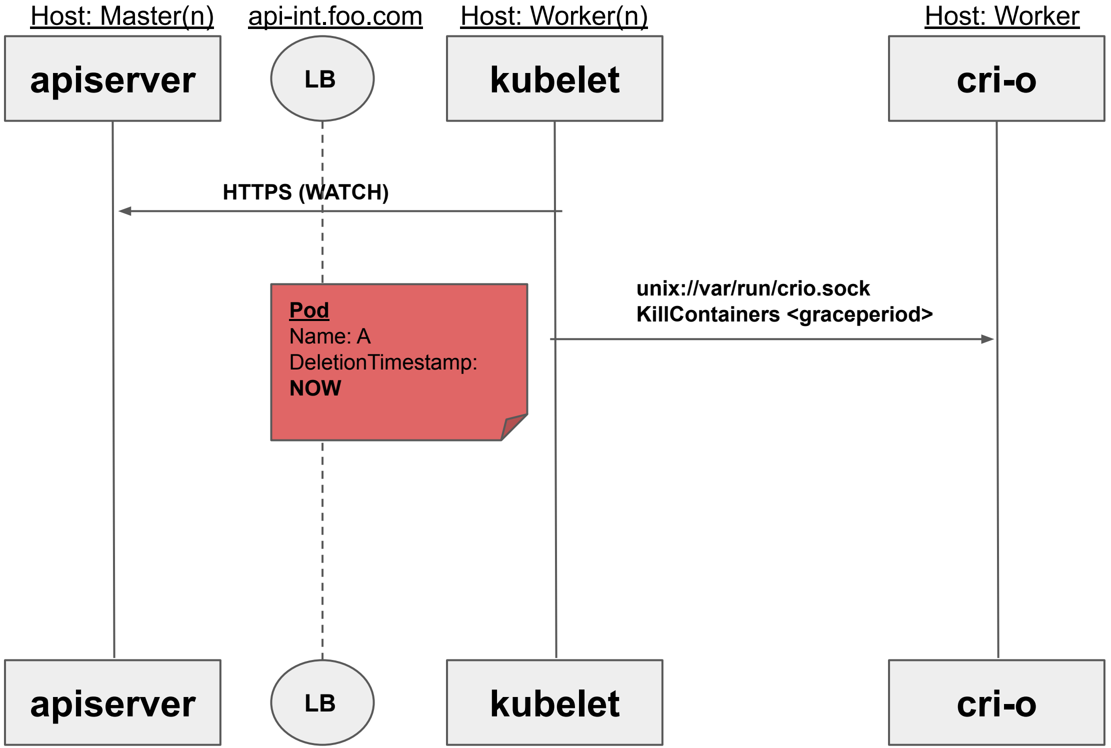
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy
- /kubepods.slice
  - /podA.slice
    - Sandbox
    - **Container 1 … N**
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet requests runtime to start container.
Each container has a cgroup nested under pod cgroup.
Container is launched based on OCI config provided earlier.

```
$ kubectl delete pods <foo>
```

Kubelet pod deletion - kill containers



Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

- /podA
  - /etc-hosts
  - /volumes
    - secret-1 (tmpfs)
    - configMap-1
    - ...

Containers

- podA sandbox (podIP) (image=pause)
- **Container 1… N**

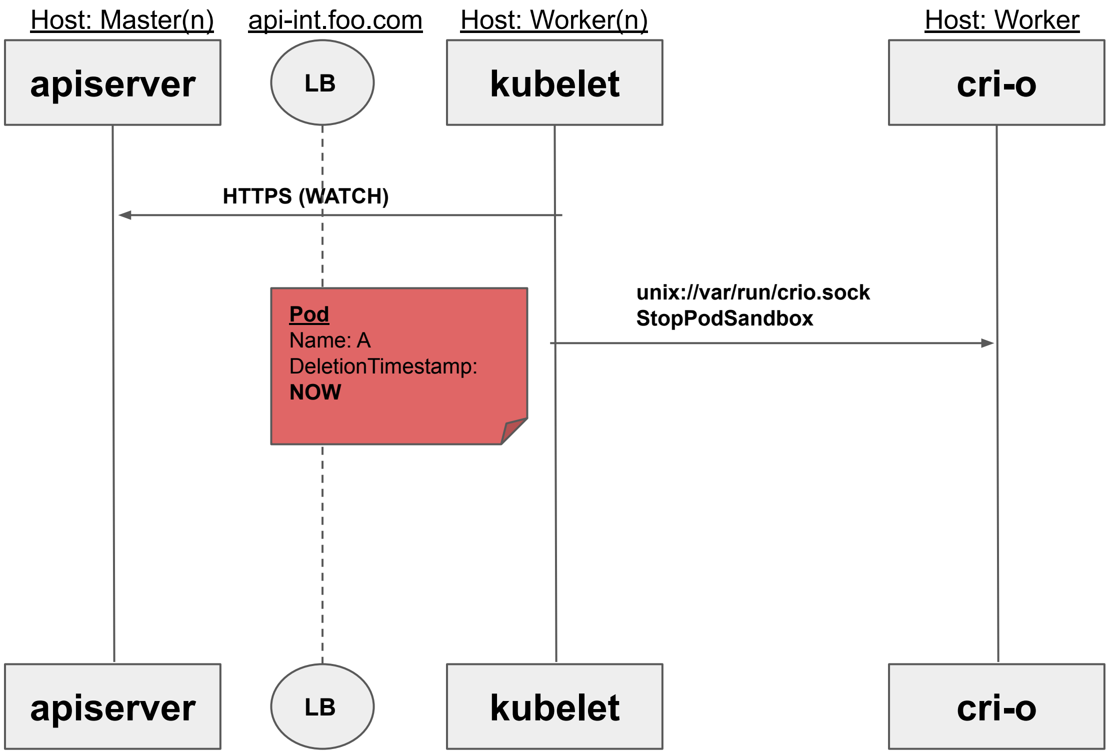
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

- /kubepods.slice
  - /podA.slice
    - Sandbox
    - **Container 1 … N**
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet observes desired state to terminate pod

Kubelet pod deletion - stop pod sandbox



Host: Master(n)

apiserver

api-int.foo.com

LB

Host: Worker(n)

kubelet

Host: Worker

cri-o

HTTPS (WATCH)

Pod
Name: A
DeletionTimestamp:
NOW

unix://var/run/crio.sock
StopPodSandbox

apiserver

LB

kubelet

cri-o

Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

- /podA
  - /etc-hosts
  - /volumes
    - secret-1 (tmpfs)
    - configMap-1
    - ...

Containers

- **podA sandbox (podIP) (image=pause)**

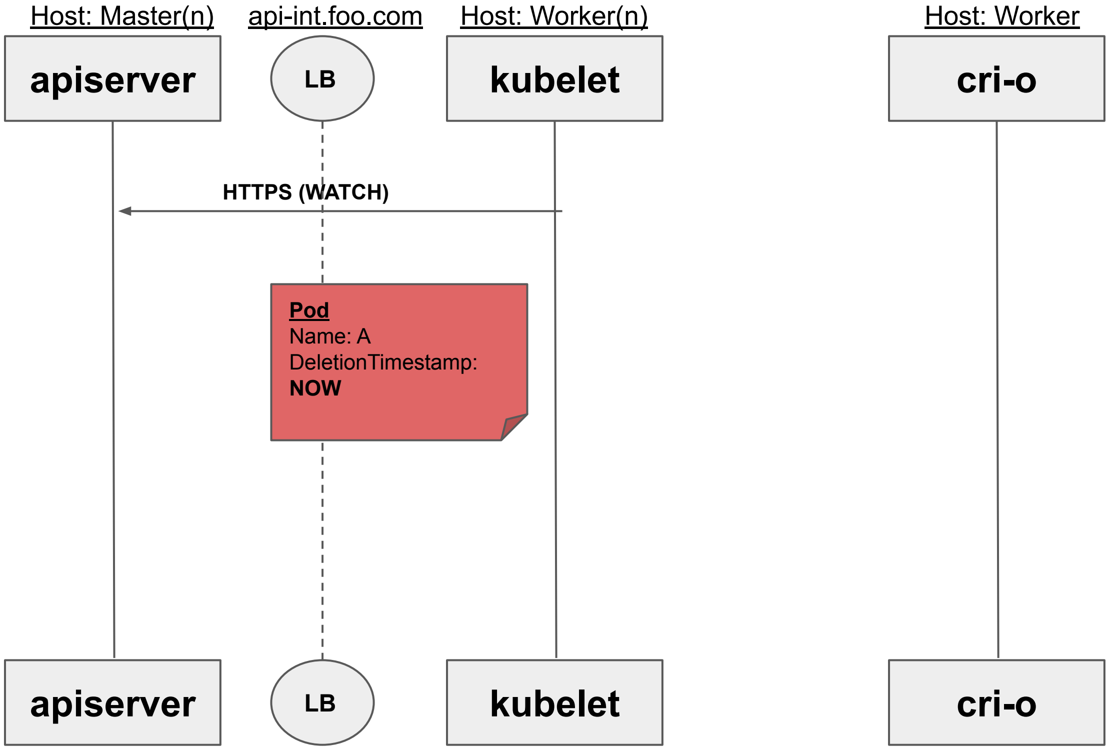
Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy

- /kubepods.slice
  - /podA.slice
    - **Sandbox**
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet instructs runtime to remove pods sandbox once containers terminated.

Kubelet - purge pod cgroup



Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

Containers

Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

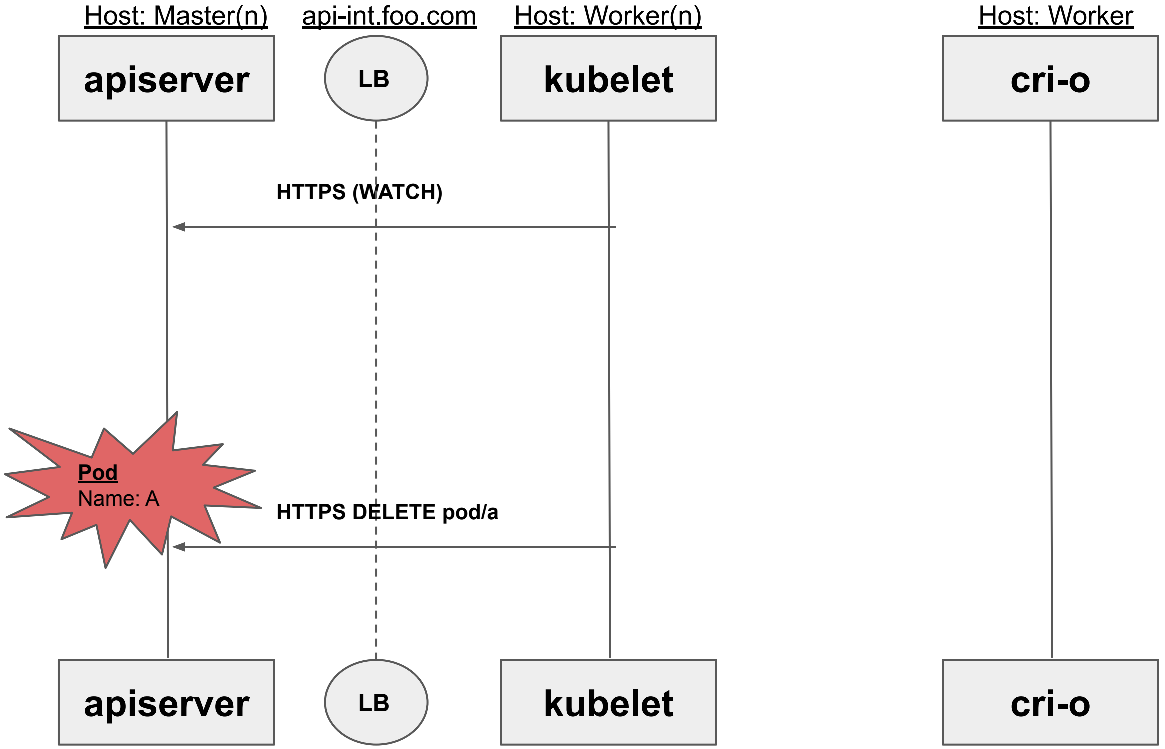
Cgroup Hierarchy

- /kubepods.slice
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

Kubelet cleans up pod cgroup.

Red Hat Developer | DevNation

Kubelet - remove record from API server



Host: Worker (Linux FS State)

Host Filesystem
/var/lib/kubelet/pods

Containers

Cgroup Controllers:
cpu, cpuacct, cpuset, memory, hugetlb, pids

Cgroup Hierarchy
- /kubepods.slice
- /kubepods-besteffort.slice
- /kubepods-burstable.slice

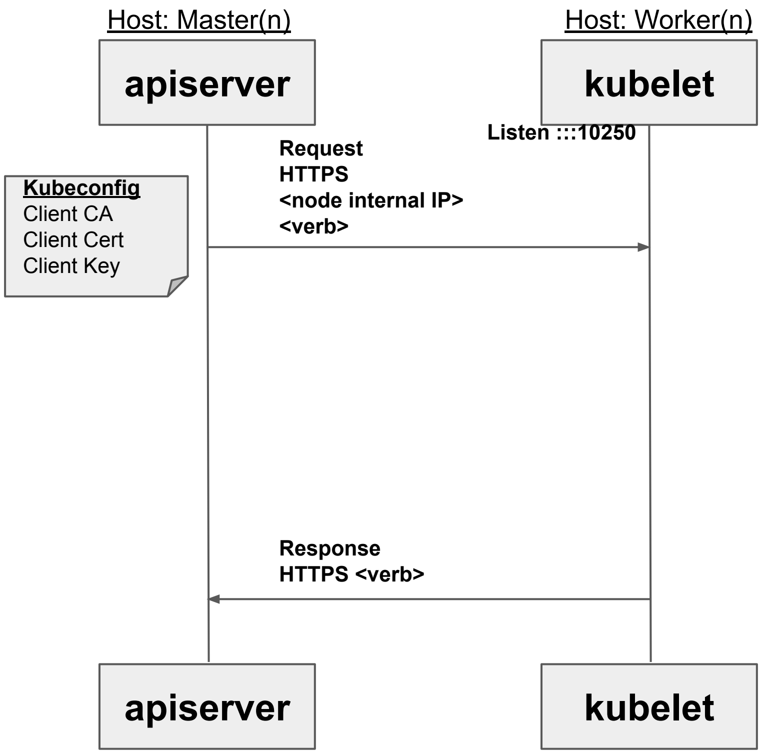
Kubelet sends delete to API server to remove record.

```
$ kubectl exec mypod
```

OR

```
$ kubectl logs mypod
```

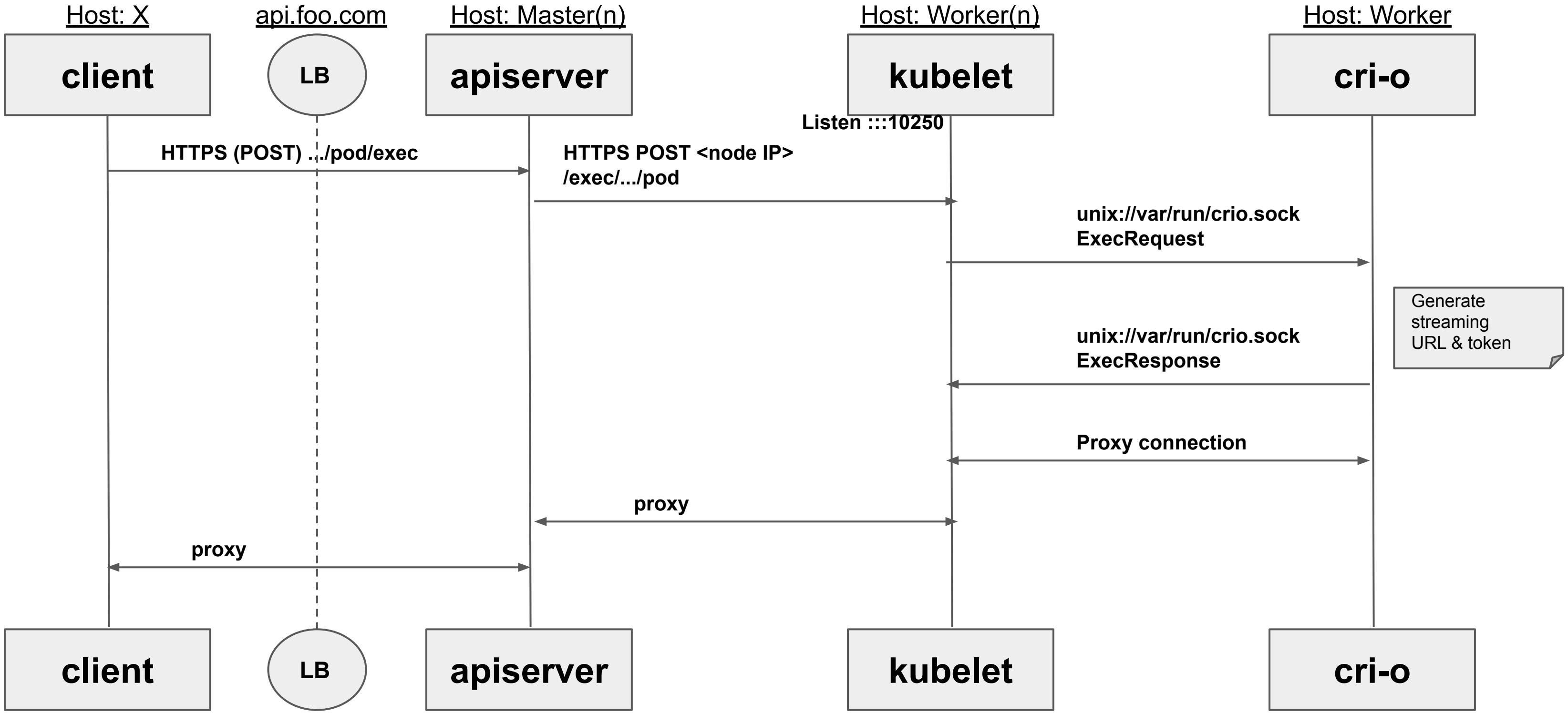
Network Flow - Control Plane to Worker



Network
API server connects over default interface to node advertised internal IP address. Kubelet serves from :::10250

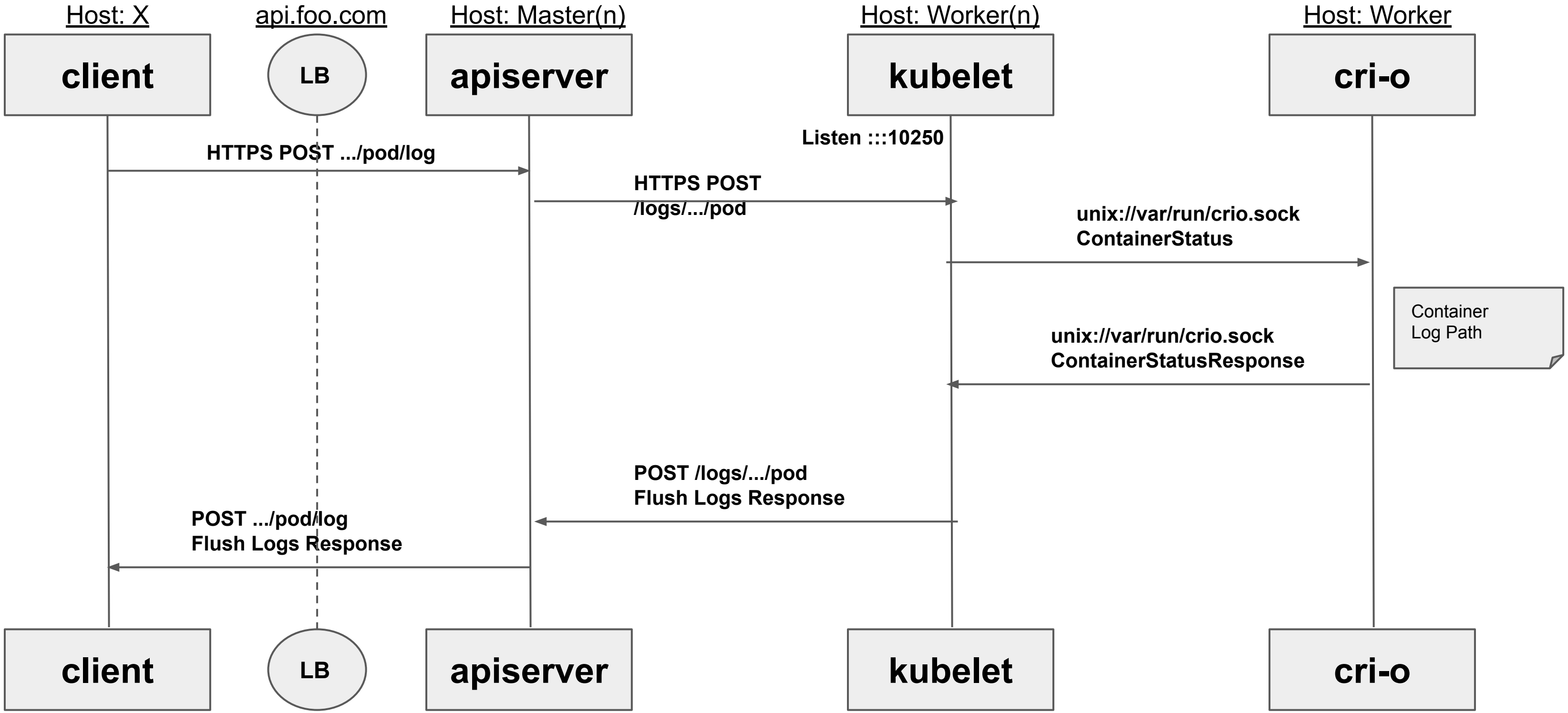
Kubelet Certificate Rotation
Client and Serving certificates are rotated automatically. Serving certificates are validated during rotation.

Network Flow - Streaming Requests (ex: exec)



Host: X
api.foo.com
Host: Master(n)
Host: Worker(n)
Host: Worker
client
LB
apiserver
kubelet
cri-o
Listen :::10250
HTTPS (POST) .../pod/exec
HTTPS POST <node IP> /exec/.../pod
unix://var/run/crio.sock ExecRequest
Generate streaming URL & token
unix://var/run/crio.sock ExecResponse
Proxy connection
proxy
proxy
client
LB
apiserver
kubelet
cri-o

Network Flows - Logs

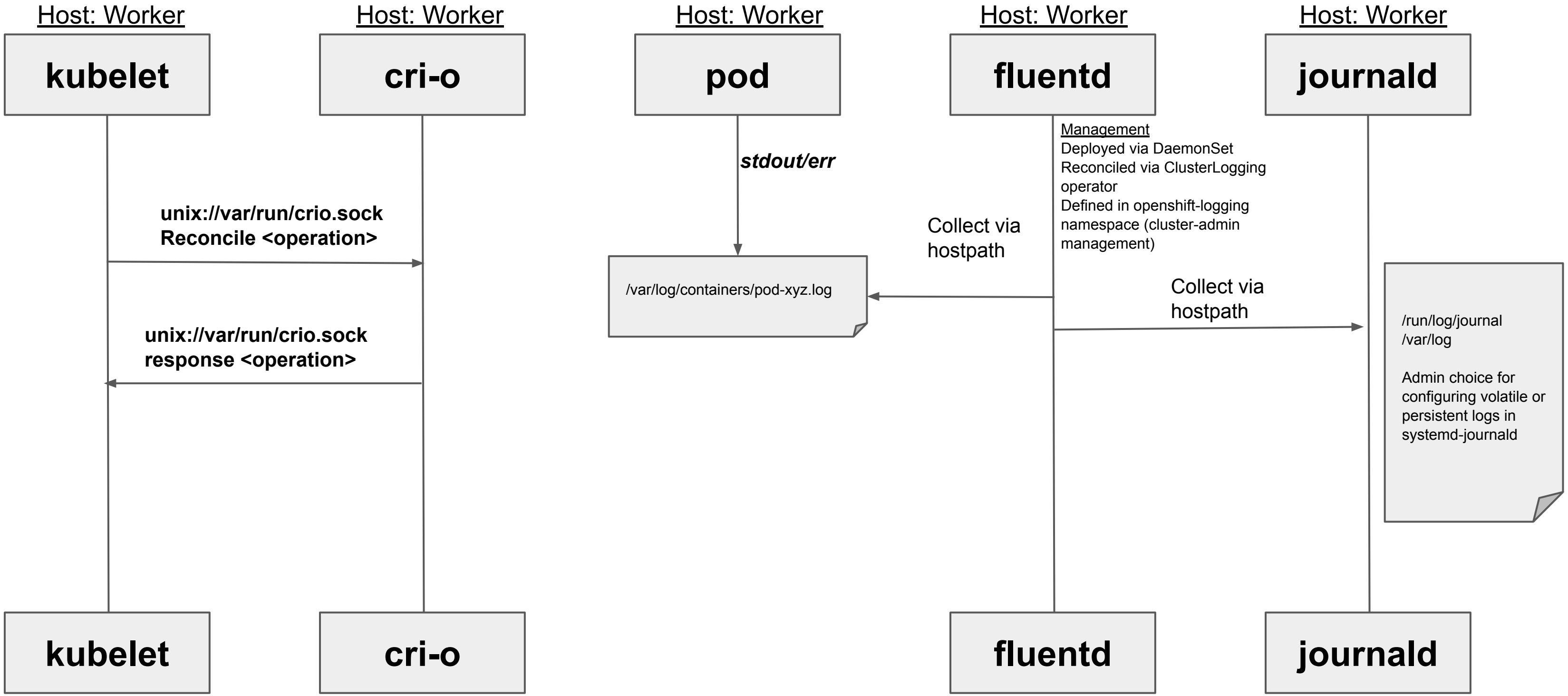


Supporting services

Cluster Logging

Cluster Monitoring

Node - Log Collection



Host: Worker

kubelet

Host: Worker

cri-o

Host: Worker

pod

Host: Worker

fluentd

Host: Worker

journald

unix://var/run/crio.sock
Reconcile <operation>

unix://var/run/crio.sock
response <operation>

stdout/err

/var/log/containers/pod-xyz.log

Collect via
hostpath

Management
Deployed via DaemonSet
Reconciled via ClusterLogging operator
Defined in openshift-logging namespace (cluster-admin management)

Collect via
hostpath

/run/log/journal
/var/log

Admin choice for configuring volatile or persistent logs in systemd-journald

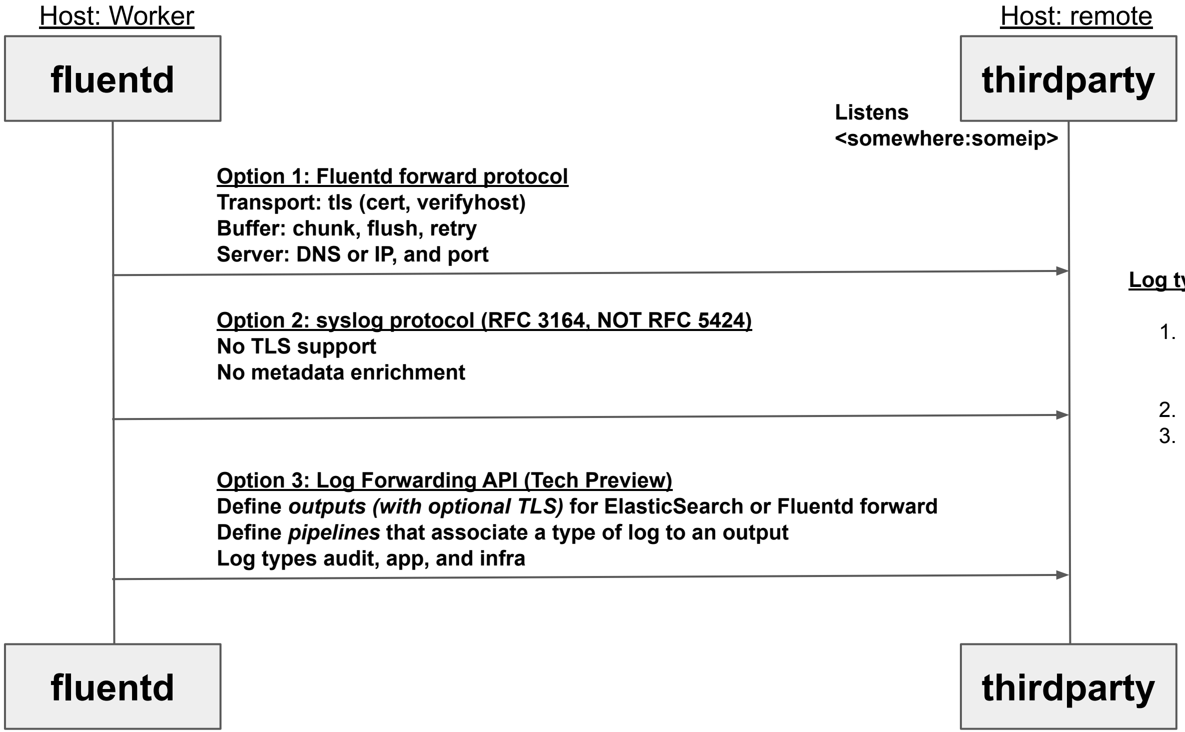
kubelet

cri-o

fluentd

journald

Log Collection - Forwarding



Option 1: Fluentd forward protocol
Transport: tls (cert, verifyhost)
Buffer: chunk, flush, retry
Server: DNS or IP, and port

Option 2: syslog protocol (RFC 3164, NOT RFC 5424)
No TLS support
No metadata enrichment

Option 3: Log Forwarding API (Tech Preview)
Define *outputs (with optional TLS)* for ElasticSearch or Fluentd forward
Define *pipelines* that associate a type of log to an output
Log types audit, app, and infra

Log types

1. Audit - The audit logs recording access to Kubernetes control plane and the *auditd* logs recorded on each node.
2. App - Normal end user pods
3. Infra - Logs from pods in openshift-*, kube*, and default namespaces that require elevated cluster-admin privilege for management