Pods in the Kubernetes and OpenShift Cosmos
**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
   a. Operations
   b. 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: Master(n)

apiserver

api-int.foo.com

Host: Worker(n)

LB

kubelet

unix://var/run/crio.sock

Reconcile <operation>

unix://var/run/crio.sock

Reconcile <operation>

Host: Worker

LB

cieset

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

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

**Files System:**
/var/lib/kubelet/

**Cgroup Controls:**
cpu, cpuacct, cpuset, memory, hugetlb, pids

**Cgroup Hierarchy**
- /kubepods.slice
- /kubepods-bestefort.slice
- /kubepods-burstable.slice
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 roots, 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

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: Master(n)
- apiserver
- api-int.foo.com
- LB
- HTTPS (WATCH)
- apiserver
- kubelet
- kubepods.slice
- kubepods-burstable.slice
- kubepods-besteffort.slice
- /etc-hosts
- /volumes

Host: Worker(n)
- kubelet
- Host Filesystem
  - /var/lib/kubelet/pods
    - /podA
    - /etc-hosts
    - /volumes

Host: Worker (Linux FS State)
- Cgroup Controllers:
  - cpu, cpuaacct, cpuset, memory, hugetlb, pids
- Cgroup Hierarchy
  - /kubepods.slice
    - /podA.slice
    - /kubepods-burstable.slice
    - /kubepods-besteffort.slice

Create pod local host resources

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: Master(n) api-int.foo.com Host: Worker(n)

apiserver LB kubelet

HTTPS (WATCH)

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

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

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

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

api-int.foo.com

Host: Worker (Linux FS State)

apiserver

LB

kubelet

Wait for volumes to attach and mount

Host: Worker (n)

LB

kubernetes
Kubelet sees pod - fetch pull secrets

Host: Master(n)  api-int.foo.com  Host: Worker(n)
apiserver           LB            kubelet

HTTPS GET /secret/...

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

Cgroup Controllers:
cpu, cpuacl, cgroup, memory, hugetlb, pids

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

Host: Worker (Linux FS State)

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

Host: Worker (n)
api-int.foo.com
LB
apiserver
kubelet

Secret
Name: secret-1

Fetch image pull secret for pod (if any)
Kubelet sees pod - sandbox

**Host: Master(n)**
- api-server
- **api-int.foo.com**
- LB
- HTTPS (WATCH)

**Host: Worker(n)**
- kubelet
- unix://var/run/crio.sock
- CreateSandboxRequest

**Host: Worker**
- cri-o
- unix://var/run/crio.sock
- CreateSandboxResponse

**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-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)

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.

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

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

Cgroup Controllers:
- cpu, cpuacl, cpuset, memory, hugetlb, pids

Cgroup Hierarchy
- /kubepods.slice
  - /podA.slice
    - Sandbox
      - Container 1 … N
    - /kubepods-besteffort.slice
    - /kubepods-burstable.slice
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: Master(n)      api-int.foo.com      Host: Worker(n)
apiserver            LB                   kubelet
  HTTPS (WATCH)

Host: Worker
kubelet                   cri-o

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: Master(n)       api-int.foo.com       Host: Worker(n)
apiserver            LB                      kubelet

Host: Worker
kubelet
unix://var/run/crio.sock
KillContainers <graceperiod>

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, cpucct, cgroup, 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)  api-int.foo.com  Host: Worker(n)
apiserver  LB  kubectl

Host: Worker
cri-o

Host: Worker (Linux FS State)

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

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

Cgroup Controllers:
cpu, cpuaclt, 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 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.
Kubelet - remove record from API server

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)

HTTPS (POST) ..../pod/exec
HTTPS POST <node IP> /exec/.../pod
unix://var/run/crio.sock
ExecRequest
Generate streaming URL & token
Proxy connection
unix://var/run/crio.sock
ExecResponse
Proxy connection

client
api.foo.com
Host: X
LB
apiserver
Host: Master(n)
kubelet
Host: Worker(n)
cri-o
Host: Worker

client
api.foo.com
Host: X
LB
apiserver
Host: Master(n)
kubelet
Host: Worker(n)
cri-o
Host: Worker
Network Flows - Logs

- Host: X
  - client
  - api.foo.com
  - lb

- Host: Master(n)
  - apiserver

- Host: Worker(n)
  - kubelet
  - listen ::10250

- Host: Worker
  - cri-o
  - unix://var/run/crio.sock
  - ContainerStatus
  - ContainerStatusResponse

Diagram:
- HTTPS POST .../pod/log from client to api.foo.com
- HTTPS POST /logs/.../pod from apiserver to kubelet
- POST /logs/.../pod from kubelet to cri-o
- Flush Logs Response from client to lb
- Flush Logs Response from kubelet to lb

Supporting services

Cluster Logging
Cluster Monitoring
Node - Log Collection

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

kubelet

Host: Worker
cri-o
unix://var/run/crio.sock
response <operation>

cri-o

Host: Worker
pod
/var/log/containers/pod-xyz.log
stdout/err

pod

Host: Worker
fluentd
Collect via hostpath

fluentd

Host: Worker
journald
Collect via hostpath

journald

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

/run/log/journal
/var/log

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

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

**Host: remote**
- **thirdparty**
  - Listens <somewhere:someip>
  - **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