RX-M Cloud Native Consulting

Advanced Kubernetes

Lab 2 – Kubelet and advanced pod specification

The `kubelet` is the primary "node agent" that runs on each Node in a Kubernetes cluster. The `kubelet` works in terms of PodSpecs. A PodSpec is a YAML or JSON object that describes a pod. The `kubelet` takes a set of PodSpecs that are provided through various mechanisms (primarily through the kube-apiserver), and ensures that the containers described in those PodSpecs are running and healthy.

Other than a PodSpec from the kube-apiserver, there are three additional ways that a pod manifest can be provided to the `kubelet`:

- **File** - The `staticPodPath` in the kubelet's config file or the `--pod-manifest-path` switch can be used to pass a path containing pods to run on startup. This path is rechecked every 20 seconds (configurable)
  - The switch is deprecated, replaced by the setting in the config file, but still works
  - `--file-check-frequency=20s` - duration between checking config files for new data
- **HTTP endpoint** - HTTP endpoint passed as a parameter on the command line, checked every 20 seconds (configurable)
  - `--http-check-frequency duration` - duration between checking http for new data
- **HTTP server** - The `kubelet` can also listen for HTTP manifest posts
- **--runonce[=false]** - If true, exit after spawning pods from local manifests or remote urls (can not be used with `--api-servers` and/or `--enable-server`)

The `staticPodPath` is typically used to tell the kubelet to start other kubernetes components, like the kube-proxy on worker nodes and/or the kube-apiserver on master nodes.

1. Stop running cluster components

To experiment with the `kubelet` independently, stop the Kubernetes components (kube-apiserver, kubelet, and etcd) you may have running by typing ^C (or kill -SIGINT) in their TTYs.

```
ubuntu@nodea:~$ sudo kill -SIGINT $(pidof kube-apiserver kubelet etcd)

ubuntu@nodea:~$
```

Before you restart your cluster, clear the kube-apiserver cluster state by removing the `etcd` backing store:

```
ubuntu@nodea:~$ rm -Rf ~/default.etcd/

ubuntu@nodea:~$
```

The `kubelet` also caches its state on disk. You can eliminate the kubelet's cached state by stopping kubelet and then removing the kubelet's backing store as well:

```
ubuntu@nodea:~$ sudo rm -Rf /var/lib/kubelet/

ubuntu@nodea:~$
```

This is a good remedy for components that will not restart due to preexisting state that is out of synch with the rest of the cluster.

Recheck that all of the Kubernetes services are stopped.

```
ubuntu@nodea:~$ pidof etcd kube-apiserver kubelet

ubuntu@nodea:~$
```

In addition to k8s components, we need to clean up what Docker is currently running.

```
ubuntu@nodea:~$ docker container rm $(docker container stop $(docker container ls -qa))

...
ubuntu@nodea:~$
```

2. Using files

When you run the `kubelet` you can supply a single manifest file (or several in a directory) as a command line argument. On `kubelet` startup these manifests

will start prior to the manifests supplied by the API server. If no kube-apiserver is supplied the `kubelet` will simply run these manifests independently.

We will try running the `kubelet` stand alone with a simple pod config. First create a working directory for your configuration files:

```
ubuntu@nodea:~$ cd

ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ mkdir kubelet

ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ cd kubelet/

ubuntu@nodea:~/kubelet$
```

Now create a simple pod to test:

```
ubuntu@nodea:~/kubelet$ vim pod.yaml
ubuntu@nodea:~/kubelet$ cat pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx-startup
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80
ubuntu@nodea:~/kubelet$
```

Now start the `kubelet` with the new pod spec file supplied as a `--pod-manifest-path` parameter:

```
ubuntu@nodea:~/kubelet$ sudo $HOME/k8s/_output/bin/kubelet --pod-manifest-path=/home/ubuntu/kubelet/pod.yaml

Flag --pod-manifest-path has been deprecated, This parameter should be set via the config file specified by the
Kubelet's --config flag. See https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/ for more
information.
I0330 03:57:57.749438    4625 server.go:417] Version: v1.14.0
I0330 03:57:57.749621    4625 plugins.go:103] No cloud provider specified.
W0330 03:57:57.749639    4625 server.go:556] standalone mode, no API client
W0330 03:57:57.793899    4625 server.go:474] No api server defined - no events will be sent to API server.
I0330 03:57:57.793923    4625 server.go:625] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
I0330 03:57:57.794303    4625 container_manager_linux.go:261] container manager verified user specified cgroup-
root exists: []
I0330 03:57:57.794323    4625 container_manager_linux.go:266] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs KubeletRootDir:/var/lib/kubelet ProtectKernelDefaults:false
NodeAllocatableConfig:{KubeReservedCgroupName: SystemReservedCgroupName: EnforceNodeAllocatable:map[pods:{}]
KubeReserved:map[] SystemReserved:map[] HardEvictionThresholds:[{Signal:memory.available Operator:LessThan Value:
{Quantity:100Mi Percentage:0} GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.available Operator:LessThan Value:
{Quantity:<nil> Percentage:0.1} GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.inodesFree Operator:LessThan
Value:{Quantity:<nil> Percentage:0.05} GracePeriod:0s MinReclaim:<nil>} {Signal:imagefs.available
Operator:LessThan Value:{Quantity:<nil> Percentage:0.15} GracePeriod:0s MinReclaim:<nil>}]} QOSReserved:map[]
ExperimentalCPUManagerPolicy:none ExperimentalCPUManagerReconcilePeriod:10s ExperimentalPodPidsLimit:-1
EnforceCPULimits:true CPUCFSQuotaPeriod:100ms}
I0330 03:57:57.794472    4625 container_manager_linux.go:286] Creating device plugin manager: true
I0330 03:57:57.794624    4625 state_mem.go:36] [cpumanager] initializing new in-memory state store
I0330 03:57:57.797262    4625 kubelet.go:279] Adding pod path: /home/ubuntu/kubelet/pod.yaml
I0330 03:57:57.799579    4625 client.go:75] Connecting to docker on unix:///var/run/docker.sock
I0330 03:57:57.799618    4625 client.go:104] Start docker client with request timeout=2m0s
W0330 03:57:57.800696    4625 docker_service.go:561] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0330 03:57:57.800752    4625 docker_service.go:238] Hairpin mode set to "hairpin-veth"
W0330 03:57:57.800852    4625 cni.go:213] Unable to update cni config: No networks found in /etc/cni/net.d
W0330 03:57:57.802288    4625 hostport_manager.go:68] The binary conntrack is not installed, this can cause
failures in network connection cleanup.
I0330 03:57:57.803358    4625 docker_service.go:253] Docker cri networking managed by kubernetes.io/no-op
I0330 03:57:57.820478    4625 docker_service.go:258] Docker Info: &
{ID:TQ5X:T6PX:K2WX:LMX6:W44V:RPMB:DME6:AQJP:AMOD:45JW:HCD2:RCXA Containers:0 ContainersRunning:0
ContainersPaused:0 ContainersStopped:0 Images:3 Driver:overlay2 DriverStatus:[[Backing Filesystem extfs] [Supports
d_type true] [Native Overlay Diff true]] SystemStatus:[] Plugins:{Volume:[local] Network:[bridge host macvlan null
overlay] Authorization:[] Log:[awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog]}
MemoryLimit:true SwapLimit:false KernelMemory:true CPUCfsPeriod:true CPUCfsQuota:true CPUShares:true CPUSet:true
```

```
IPv4Forwarding:true BridgeNfIptables:true BridgeNfIP6tables:true Debug:false NFd:22 OomKillDisable:true
NGoroutines:38 SystemTime:2019-03-30T03:57:57.804036176Z LoggingDriver:json-file CgroupDriver:cgroupfs
NEventsListener:0 KernelVersion:4.4.0-1075-aws OperatingSystem:Ubuntu 16.04.5 LTS OSType:linux Architecture:x86_64
IndexServerAddress:https://index.docker.io/v1/ RegistryConfig:0xc0007bb2d0 NCPU:2 MemTotal:8369913856
GenericResources:[] DockerRootDir:/var/lib/docker HTTPProxy: HTTPSProxy: NoProxy: Name:nodea Labels:[]
ExperimentalBuild:false ServerVersion:18.09.3 ClusterStore: ClusterAdvertise: Runtimes:map[runc:{Path:runc Args:
[]}] DefaultRuntime:runc Swarm:{NodeID: NodeAddr: LocalNodeState:inactive ControlAvailable:false Error:
RemoteManagers:[] Nodes:0 Managers:0 Cluster:<nil>} LiveRestoreEnabled:false Isolation: InitBinary:docker-init
ContainerdCommit:{ID:e6b3f5632f50dbc4e9cb6288d911bf4f5e95b18e Expected:e6b3f5632f50dbc4e9cb6288d911bf4f5e95b18e}
RuncCommit:{ID:6635b4f0c6af3810594d2770f662f34ddc15b40d Expected:6635b4f0c6af3810594d2770f662f34ddc15b40d}
InitCommit:{ID:fec3683 Expected:fec3683} SecurityOptions:[name=apparmor name=seccomp,profile=default]}
I0330 03:57:57.820566    4625 docker_service.go:271] Setting cgroupDriver to cgroupfs
I0330 03:57:57.838006    4625 remote_runtime.go:62] parsed scheme: ""
I0330 03:57:57.838028    4625 remote_runtime.go:62] scheme "" not registered, fallback to default scheme
I0330 03:57:57.838153    4625 remote_image.go:50] parsed scheme: ""
I0330 03:57:57.838169    4625 remote_image.go:50] scheme "" not registered, fallback to default scheme
I0330 03:57:57.838311    4625 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 03:57:57.838328    4625 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 03:57:57.838349    4625 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 03:57:57.838360    4625 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 03:57:57.838378    4625 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000841a40, CONNECTING
I0330 03:57:57.838388    4625 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0008e0870, CONNECTING
I0330 03:57:57.838502    4625 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000841a40, READY
I0330 03:57:57.838503    4625 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0008e0870, READY
I0330 03:57:57.839903    4625 kuberuntime_manager.go:210] Container runtime docker initialized, version: 18.09.3,
apiVersion: 1.39.0
W0330 03:57:57.840368    4625 csi_plugin.go:218] kubernetes.io/csi: kubeclient not set, assuming standalone
kubelet
I0330 03:57:57.843996    4625 server.go:1037] Started kubelet
E0330 03:57:57.844114    4625 kubelet.go:1282] Image garbage collection failed once. Stats initialization may not
have completed yet: failed to get imageFs info: unable to find data in memory cache
W0330 03:57:57.844144    4625 kubelet.go:1387] No api server defined - no node status update will be sent.
I0330 03:57:57.844621    4625 fs_resource_analyzer.go:64] Starting FS ResourceAnalyzer
I0330 03:57:57.844653    4625 status_manager.go:148] Kubernetes client is nil, not starting status manager.
I0330 03:57:57.844672    4625 kubelet.go:1806] Starting kubelet main sync loop.
I0330 03:57:57.844692    4625 kubelet.go:1823] skipping pod synchronization - [container runtime status check may
not have completed yet., PLEG is not healthy: pleg has yet to be successful.]
```

```
I0330 03:57:57.844757    4625 server.go:141] Starting to listen on 0.0.0.0:10250
I0330 03:57:57.845301    4625 server.go:343] Adding debug handlers to kubelet server.
I0330 03:57:57.846695    4625 volume_manager.go:248] Starting Kubelet Volume Manager
E0330 03:57:57.847627    4625 runtime.go:69] Observed a panic: "invalid memory address or nil pointer dereference"
(runtime error: invalid memory address or nil pointer dereference)
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:76
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:65
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:51
/usr/local/go/src/runtime/panic.go:522
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:189
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:214
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:125
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:152
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:153
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:88
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:124
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:54
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:71
/usr/local/go/src/runtime/asm_amd64.s:1337
I0330 03:57:57.848720    4625 desired_state_of_world_populator.go:130] Desired state populator starts to run
I0330 03:57:57.866355    4625 clientconn.go:440] parsed scheme: "unix"
I0330 03:57:57.866369    4625 clientconn.go:440] scheme "unix" not registered, fallback to default scheme
I0330 03:57:57.866398    4625 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{unix:///run/containerd/containerd.sock 0  <nil>}]
I0330 03:57:57.866408    4625 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 03:57:57.866437    4625 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000246a00, CONNECTING
I0330 03:57:57.866537    4625 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000246a00, READY
I0330 03:57:57.943776    4625 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 03:57:57.944908    4625 kubelet.go:1823] skipping pod synchronization - container runtime status check may
not have completed yet.
I0330 03:57:57.945531    4625 cpu_manager.go:155] [cpumanager] starting with none policy
I0330 03:57:57.945549    4625 cpu_manager.go:156] [cpumanager] reconciling every 10s
I0330 03:57:57.945559    4625 policy_none.go:42] [cpumanager] none policy: Start
W0330 03:57:57.946094    4625 manager.go:538] Failed to retrieve checkpoint for "kubelet_internal_checkpoint":
checkpoint is not found
W0330 03:57:57.946578    4625 container_manager_linux.go:818] CPUAccounting not enabled for pid: 4625
W0330 03:57:57.946594    4625 container_manager_linux.go:821] MemoryAccounting not enabled for pid: 4625
I0330 03:57:57.946685    4625 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
E0330 03:57:57.956104    4625 summary_sys_containers.go:47] Failed to get system container stats for
```

```
"/user.slice/user-1000.slice/session-119.scope": failed to get cgroup stats for "/user.slice/user-
1000.slice/session-119.scope": failed to get container info for "/user.slice/user-1000.slice/session-119.scope":
unknown container "/user.slice/user-1000.slice/session-119.scope"
I0330 03:57:58.145494    4625 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 03:57:58.153962    4625 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 03:57:58.250292    4625 reconciler.go:154] Reconciler: start to sync state

...
```

Give Docker enough time to pull the nginx image, then list the running containers in a new shell.

```
ubuntu@nodea:~$ docker container ls --no-trunc --format "table {{.Image}}"

IMAGE
nginx@sha256:e3456c851a152494c3e4ff5fcc26f240206abac0c9d794affb40e0714846c451
k8s.gcr.io/pause:3.1
ubuntu@nodea:~$
```

Note the nginx image tag (in the YAML), it is version 1.7.9 as requested in the spec but Kubernetes and Docker 1.10 and above track images by the content addressable SHA hash. Imagine we would like to change the version. Instead of manipulating the Docker containers directly, we will update the pod configuration and let the `kubelet` redeploy the new version of nginx.

In a separate shell, update your config to request image tag 1.9.1, leaving your `kubelet` running:

```
ubuntu@nodea:~$ cd kubelet/

ubuntu@nodea:~/kubelet$
```

```
ubuntu@nodea:~/kubelet$ vim pod.yaml
ubuntu@nodea:~/kubelet$ cat pod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx-startup
```

```
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.9.1
    ports:
    - containerPort: 80
ubuntu@nodea:~/kubelet$
```

Shortly after you save your file changes you should notice the following log output in the `kubelet` log:

```
...

E0708 19:11:24.989605   40352 file_linux.go:61] Unable to read config path "/home/user/kubelet/pod.yaml": error
while processing inotify event ("/home/user/kubelet/pod.yaml": 0x400 == IN_DELETE_SELF): the watched path is
deleted
I0708 19:11:24.996046   40352 kubelet_node_status.go:269] Setting node annotation to enable volume controller
attach/detach
W0708 19:11:29.529114   40352 pod_container_deletor.go:75] Container
"d99aed7c64c24279287f70fa8ba1ae402ff025c5e2c4b53028232e949a627be4" not found in pod's containers
E0708 19:11:30.117230   40352 kuberuntime_container.go:65] Can't make a ref to pod "nginx-startup-
nodea_default(9d6e47038092eb6f3563ff648b593046)", container nginx: selfLink was empty, can't make reference
I0708 19:11:30.538543   40352 kubelet_node_status.go:269] Setting node annotation to enable volume controller
attach/detach
W0708 19:11:30.543534   40352 pod_container_deletor.go:75] Container
"98539533b6f25724223387e8faca0cbecd9ac4350f34e958388c39b33c9535d2" not found in pod's containers

...
```

It may take Docker some time to pull the image. You can see the pull status by issuing the appropriate `docker pull` command:

```
ubuntu@nodea:~/kubelet$ docker image pull nginx:1.9.1

1.9.1: Pulling from library/nginx
5641bf7f839b: Pull complete
a3ed95caeb02: Pull complete
d003dd0d7f8a: Pull complete
c5dd085dcc7c: Pull complete
d95a07673dd5: Pull complete
```

```
cec5c5855afe: Pull complete
b315c6f2ccf3: Pull complete
Digest: sha256:2f68b99bc0d6d25d0c56876b924ec20418544ff28e1fb89a4c27679a40da811b
Status: Downloaded newer image for nginx:1.9.1

ubuntu@nodea:~/kubelet$
```

If you list the running containers you will see that the kubelet has started the new nginx container.

```
ubuntu@nodea:~/kubelet$ docker container ls \
--no-trunc --format "table {{.Image}}\t{{.CreatedAt}}\t{{.Status}}"

IMAGE                                                                  CREATED AT                      STATUS
sha256:94ec7e53edfc793d6d8412b4748cd84270da290ce9256730eb428574f98f7c95   2018-04-30 19:10:12 -0700 PDT   Up 3
minutes
k8s.gcr.io/pause:3.1                                                   2018-04-30 19:10:12 -0700 PDT   Up 3
minutes

ubuntu@nodea:~/kubelet$
```

- Note that the new 1.9 nginx container has a different SHA hash than the 1.7 version.

While the above example is just an experiment, this is exactly the way a Kubernetes master is typically boot strapped. For example, the kubeadm installer, configures a kubelet with a pod manifest path and then adds pod specs to the directory for etcd, the api-server, the controller-manager, and the scheduler. In this way the kubelet is the only Kubernetes service actually running on the host, all of the other services run in containers.

The kubelet in turn is generally configured as a systemd service, so if the kubelet fails, systemd will restart it.

3. HTTP endpoint

The `kubelet` also has the ability load manifests from URL based resources. The following switches configure this feature:

- **staticPodURL / --manifest-url=""** - URL for accessing the container manifest (deprecated but still available)
- **staticPodURLHeader / --manifest-url-header=""** - HTTP header to use when accessing the manifest URL, with the key separated from the value with a ':', as in 'key:value'
- **httpCheckFrequency / --http-check-frequency=20s** - Duration between checking http for new data

Let's try running the `kubelet` using a config supplied by URL.

Stop the `kubelet` with ^C.

Clean up the running containers.

```
ubuntu@nodea:~/kubelet$ docker container rm $(docker container stop $(docker container ls -qa))

...
ubuntu@nodea:~/kubelet$
```

Clear all of the kubelet state:

```
ubuntu@nodea:~/kubelet$ sudo rm -Rf  /var/lib/kubelet

ubuntu@nodea:~/kubelet$
```

The Kubernetes GitHub repo has a sample pod we can use for this test:

```
ubuntu@nodea:~/kubelet$ curl https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.10/examples/pod

# Copy of pod.yaml without file extension for test
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80

ubuntu@nodea:~/kubelet$
```

Now rerun the `kubelet`, providing it the URL from the Kubernetes repo.

```
ubuntu@nodea:~/kubelet$ sudo $HOME/k8s/_output/bin/kubelet \
--manifest-url=https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.10/examples/pod

Flag --manifest-url has been deprecated, This parameter should be set via the config file specified by the
Kubelet's --config flag. See https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/ for more
information.
I0330 04:06:47.648756    5750 server.go:417] Version: v1.14.0
I0330 04:06:47.648932    5750 plugins.go:103] No cloud provider specified.
W0330 04:06:47.648949    5750 server.go:556] standalone mode, no API client
W0330 04:06:47.689169    5750 server.go:474] No api server defined - no events will be sent to API server.
I0330 04:06:47.689192    5750 server.go:625] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
I0330 04:06:47.689481    5750 container_manager_linux.go:261] container manager verified user specified cgroup-
root exists: []
I0330 04:06:47.689501    5750 container_manager_linux.go:266] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs KubeletRootDir:/var/lib/kubelet ProtectKernelDefaults:false
NodeAllocatableConfig:{KubeReservedCgroupName: SystemReservedCgroupName: EnforceNodeAllocatable:map[pods:{}]
KubeReserved:map[] SystemReserved:map[] HardEvictionThresholds:[{Signal:nodefs.inodesFree Operator:LessThan Value:
{Quantity:<nil> Percentage:0.05} GracePeriod:0s MinReclaim:<nil>} {Signal:imagefs.available Operator:LessThan
Value:{Quantity:<nil> Percentage:0.15} GracePeriod:0s MinReclaim:<nil>} {Signal:memory.available Operator:LessThan
Value:{Quantity:100Mi Percentage:0} GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.available Operator:LessThan
Value:{Quantity:<nil> Percentage:0.1} GracePeriod:0s MinReclaim:<nil>}]} QOSReserved:map[]
ExperimentalCPUManagerPolicy:none ExperimentalCPUManagerReconcilePeriod:10s ExperimentalPodPidsLimit:-1
EnforceCPULimits:true CPUCFSQuotaPeriod:100ms}
I0330 04:06:47.689599    5750 container_manager_linux.go:286] Creating device plugin manager: true
I0330 04:06:47.689667    5750 state_mem.go:36] [cpumanager] initializing new in-memory state store
I0330 04:06:47.692121    5750 kubelet.go:285] Adding pod url
"https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.10/examples/pod" with HTTP header map[]
I0330 04:06:47.693610    5750 client.go:75] Connecting to docker on unix:///var/run/docker.sock
I0330 04:06:47.693630    5750 client.go:104] Start docker client with request timeout=2m0s
W0330 04:06:47.694612    5750 docker_service.go:561] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0330 04:06:47.694633    5750 docker_service.go:238] Hairpin mode set to "hairpin-veth"
W0330 04:06:47.694719    5750 cni.go:213] Unable to update cni config: No networks found in /etc/cni/net.d
W0330 04:06:47.696137    5750 hostport_manager.go:68] The binary conntrack is not installed, this can cause
failures in network connection cleanup.
I0330 04:06:47.697092    5750 docker_service.go:253] Docker cri networking managed by kubernetes.io/no-op
I0330 04:06:47.714555    5750 docker_service.go:258] Docker Info: &
{ID:TQ5X:T6PX:K2WX:LMX6:W44V:RPMB:DME6:AQJP:AMOD:45JW:HCD2:RCXA Containers:0 ContainersRunning:0
ContainersPaused:0 ContainersStopped:0 Images:5 Driver:overlay2 DriverStatus:[[Backing Filesystem extfs] [Supports
d_type true] [Native Overlay Diff true]] SystemStatus:[] Plugins:{Volume:[local] Network:[bridge host macvlan null
```

```
overlay] Authorization:[] Log:[awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog]}
MemoryLimit:true SwapLimit:false KernelMemory:true CPUCfsPeriod:true CPUCfsQuota:true CPUShares:true CPUSet:true
IPv4Forwarding:true BridgeNfIptables:true BridgeNfIP6tables:true Debug:false NFd:22 OomKillDisable:true
NGoroutines:38 SystemTime:2019-03-30T04:06:47.697759333Z LoggingDriver:json-file CgroupDriver:cgroupfs
NEventsListener:0 KernelVersion:4.4.0-1075-aws OperatingSystem:Ubuntu 16.04.5 LTS OSType:linux Architecture:x86_64
IndexServerAddress:https://index.docker.io/v1/ RegistryConfig:0xc0007a0620 NCPU:2 MemTotal:8369913856
GenericResources:[] DockerRootDir:/var/lib/docker HTTPProxy: HTTPSProxy: NoProxy: Name:nodea Labels:[]
ExperimentalBuild:false ServerVersion:18.09.3 ClusterStore: ClusterAdvertise: Runtimes:map[runc:{Path:runc Args:
[]}] DefaultRuntime:runc Swarm:{NodeID: NodeAddr: LocalNodeState:inactive ControlAvailable:false Error:
RemoteManagers:[] Nodes:0 Managers:0 Cluster:<nil>} LiveRestoreEnabled:false Isolation: InitBinary:docker-init
ContainerdCommit:{ID:e6b3f5632f50dbc4e9cb6288d911bf4f5e95b18e Expected:e6b3f5632f50dbc4e9cb6288d911bf4f5e95b18e}
RuncCommit:{ID:6635b4f0c6af3810594d2770f662f34ddc15b40d Expected:6635b4f0c6af3810594d2770f662f34ddc15b40d}
InitCommit:{ID:fec3683 Expected:fec3683} SecurityOptions:[name=apparmor name=seccomp,profile=default]}
I0330 04:06:47.714890    5750 docker_service.go:271] Setting cgroupDriver to cgroupfs
I0330 04:06:47.737120    5750 remote_runtime.go:62] parsed scheme: ""
I0330 04:06:47.737135    5750 remote_runtime.go:62] scheme "" not registered, fallback to default scheme
I0330 04:06:47.737154    5750 remote_image.go:50] parsed scheme: ""
I0330 04:06:47.737160    5750 remote_image.go:50] scheme "" not registered, fallback to default scheme
I0330 04:06:47.737294    5750 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 04:06:47.737306    5750 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 04:06:47.737341    5750 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000bde910, CONNECTING
I0330 04:06:47.737448    5750 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000bde910, READY
I0330 04:06:47.737464    5750 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 04:06:47.737471    5750 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 04:06:47.737494    5750 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000bdea60, CONNECTING
I0330 04:06:47.737599    5750 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000bdea60, READY
I0330 04:06:47.738606    5750 kuberuntime_manager.go:210] Container runtime docker initialized, version: 18.09.3,
apiVersion: 1.39.0
W0330 04:06:47.738886    5750 csi_plugin.go:218] kubernetes.io/csi: kubeclient not set, assuming standalone
kubelet
I0330 04:06:47.742925    5750 server.go:1037] Started kubelet
E0330 04:06:47.743046    5750 kubelet.go:1282] Image garbage collection failed once. Stats initialization may not
have completed yet: failed to get imageFs info: unable to find data in memory cache
W0330 04:06:47.743075    5750 kubelet.go:1387] No api server defined - no node status update will be sent.
I0330 04:06:47.743531    5750 fs_resource_analyzer.go:64] Starting FS ResourceAnalyzer
I0330 04:06:47.743558    5750 status_manager.go:148] Kubernetes client is nil, not starting status manager.
I0330 04:06:47.743578    5750 kubelet.go:1806] Starting kubelet main sync loop.
```

```
I0330 04:06:47.743598    5750 kubelet.go:1823] skipping pod synchronization - [container runtime status check may
not have completed yet., PLEG is not healthy: pleg has yet to be successful.]
I0330 04:06:47.743663    5750 server.go:141] Starting to listen on 0.0.0.0:10250
I0330 04:06:47.744182    5750 server.go:343] Adding debug handlers to kubelet server.
I0330 04:06:47.748713    5750 volume_manager.go:248] Starting Kubelet Volume Manager
E0330 04:06:47.750501    5750 runtime.go:69] Observed a panic: "invalid memory address or nil pointer dereference"
(runtime error: invalid memory address or nil pointer dereference)
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:76
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:65
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:51
/usr/local/go/src/runtime/panic.go:522
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:189
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:214
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:125
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:152
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:153
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:88
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:124
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:54
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:71
/usr/local/go/src/runtime/asm_amd64.s:1337
I0330 04:06:47.750539    5750 desired_state_of_world_populator.go:130] Desired state populator starts to run
I0330 04:06:47.787387    5750 clientconn.go:440] parsed scheme: "unix"
I0330 04:06:47.787405    5750 clientconn.go:440] scheme "unix" not registered, fallback to default scheme
I0330 04:06:47.787675    5750 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{unix:///run/containerd/containerd.sock 0  <nil>}]
I0330 04:06:47.787695    5750 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 04:06:47.788150    5750 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000641ef0, CONNECTING
I0330 04:06:47.788384    5750 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000641ef0, READY
I0330 04:06:47.846589    5750 kubelet.go:1823] skipping pod synchronization - container runtime status check may
not have completed yet.
I0330 04:06:47.865132    5750 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 04:06:47.866902    5750 cpu_manager.go:155] [cpumanager] starting with none policy
I0330 04:06:47.866919    5750 cpu_manager.go:156] [cpumanager] reconciling every 10s
I0330 04:06:47.866932    5750 policy_none.go:42] [cpumanager] none policy: Start
W0330 04:06:47.867591    5750 manager.go:538] Failed to retrieve checkpoint for "kubelet_internal_checkpoint":
checkpoint is not found
I0330 04:06:47.867868    5750 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
W0330 04:06:47.868066    5750 container_manager_linux.go:818] CPUAccounting not enabled for pid: 5750
```

```
W0330 04:06:47.868082    5750 container_manager_linux.go:821] MemoryAccounting not enabled for pid: 5750
E0330 04:06:47.877761    5750 summary_sys_containers.go:47] Failed to get system container stats for
"/user.slice/user-1000.slice/session-119.scope": failed to get cgroup stats for "/user.slice/user-
1000.slice/session-119.scope": failed to get container info for "/user.slice/user-1000.slice/session-119.scope":
unknown container "/user.slice/user-1000.slice/session-119.scope"
W0330 04:06:48.046970    5750 pod_container_deletor.go:75] Container
"8c303af855c11761594b827c598b99b3a66a846041af3641d9529ac556a59969" not found in pod's containers
I0330 04:06:48.047091    5750 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 04:06:48.055750    5750 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 04:06:48.156231    5750 reconciler.go:154] Reconciler: start to sync state

...
```

Notice the informational log output indicating that the `kubelet` is adding the desired URL to its manifest list.

> I0330 04:06:47.692121 5750 kubelet.go:285] Adding pod url "https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.10/examples/pod" with HTTP header map[]

Examine the containers running through Docker.

```
ubuntu@nodea:~/kubelet$ docker container ls \
--no-trunc --format "table {{.Image}}\t{{.CreatedAt}}\t{{.Status}}"

IMAGE                                                                     CREATED AT
STATUS
nginx@sha256:a65beb8c90a08b22a9ff6a219c2f363e16c477b6d610da28fe9cba37c2c3a2ac   2018-07-08 19:15:47 -0700 PDT   Up
About a minute
k8s.gcr.io/pause:3.1                                                      2018-07-08 19:15:45 -0700 PDT   Up
About a minute
ubuntu@nodea:~/kubelet$
```

Stop the `kubelet` again with ^C, then list the running containers.

```
ubuntu@nodea:~/kubelet$ docker container ls \
--no-trunc --format "table {{.Image}}\t{{.CreatedAt}}\t{{.Status}}"
```

```
IMAGE                                                                                 CREATED AT
STATUS
nginx@sha256:a65beb8c90a08b22a9ff6a219c2f363e16c477b6d610da28fe9cba37c2c3a2ac   2018-07-08 19:15:47 -0700 PDT   Up
About a minute
k8s.gcr.io/pause:3.1                                                                  2018-04-30 19:24:45 -0700 PDT   Up
3 minutes

ubuntu@nodea:~/kubelet$
```

As you can see the nginx pod started by the `kubelet` is still running.

Display the labels associated with the nginx container:

```
ubuntu@nodea:~/kubelet$ docker container inspect \
$(docker container ls --filter=ancestor=nginx -q) | jq -r '.[].Config.Labels'
```


```
{
  "annotation.io.kubernetes.container.hash": "d31f99e0",
  "annotation.io.kubernetes.container.ports": "[{\"containerPort\":80,\"protocol\":\"TCP\"}]",
  "annotation.io.kubernetes.container.restartCount": "0",
  "annotation.io.kubernetes.container.terminationMessagePath": "/dev/termination-log",
  "annotation.io.kubernetes.container.terminationMessagePolicy": "File",
  "annotation.io.kubernetes.pod.terminationGracePeriod": "30",
  "io.kubernetes.container.logpath": "/var/log/pods/7fd56f7a4dadfbe6b6e30602fb1e0deb/nginx/0.log",
  "io.kubernetes.container.name": "nginx",
  "io.kubernetes.docker.type": "container",
  "io.kubernetes.pod.name": "nginx-nodea",
  "io.kubernetes.pod.namespace": "default",
  "io.kubernetes.pod.uid": "7fd56f7a4dadfbe6b6e30602fb1e0deb",
  "io.kubernetes.sandbox.id": "1927e8277b0bb8b32a38d1afa23d870da378ca2e3012fb6e4a0a00d6f969a219",
  "maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"
}
```


```
ubuntu@nodea:~/kubelet$
```

As you can see, it is easy for the Kubelet to identify its own containers.

Restart the `kubelet` with no arguments:

```
ubuntu@nodea:~/kubelet$ sudo $HOME/k8s/_output/bin/kubelet

I0330 04:09:46.656964    6048 server.go:417] Version: v1.14.0
I0330 04:09:46.657221    6048 plugins.go:103] No cloud provider specified.
W0330 04:09:46.657320    6048 server.go:556] standalone mode, no API client
W0330 04:09:46.697396    6048 server.go:474] No api server defined - no events will be sent to API server.
I0330 04:09:46.697417    6048 server.go:625] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
I0330 04:09:46.697691    6048 container_manager_linux.go:261] container manager verified user specified cgroup-
root exists: []
I0330 04:09:46.697705    6048 container_manager_linux.go:266] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs KubeletRootDir:/var/lib/kubelet ProtectKernelDefaults:false
NodeAllocatableConfig:{KubeReservedCgroupName: SystemReservedCgroupName: EnforceNodeAllocatable:map[pods:{}]
KubeReserved:map[] SystemReserved:map[] HardEvictionThresholds:[{Signal:imagefs.available Operator:LessThan Value:
{Quantity:<nil> Percentage:0.15} GracePeriod:0s MinReclaim:<nil>} {Signal:memory.available Operator:LessThan
Value:{Quantity:100Mi Percentage:0} GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.available Operator:LessThan
Value:{Quantity:<nil> Percentage:0.1} GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.inodesFree Operator:LessThan
Value:{Quantity:<nil> Percentage:0.05} GracePeriod:0s MinReclaim:<nil>}]} QOSReserved:map[]
ExperimentalCPUManagerPolicy:none ExperimentalCPUManagerReconcilePeriod:10s ExperimentalPodPidsLimit:-1
EnforceCPULimits:true CPUCFSQuotaPeriod:100ms}
I0330 04:09:46.697796    6048 container_manager_linux.go:286] Creating device plugin manager: true
I0330 04:09:46.697820    6048 state_mem.go:36] [cpumanager] initializing new in-memory state store
I0330 04:09:46.697931    6048 state_mem.go:84] [cpumanager] updated default cpuset: ""
I0330 04:09:46.697946    6048 state_mem.go:92] [cpumanager] updated cpuset assignments: "map[]"
I0330 04:09:46.699443    6048 client.go:75] Connecting to docker on unix:///var/run/docker.sock
I0330 04:09:46.699462    6048 client.go:104] Start docker client with request timeout=2m0s
W0330 04:09:46.700451    6048 docker_service.go:561] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0330 04:09:46.700472    6048 docker_service.go:238] Hairpin mode set to "hairpin-veth"
W0330 04:09:46.700558    6048 cni.go:213] Unable to update cni config: No networks found in /etc/cni/net.d
W0330 04:09:46.701861    6048 hostport_manager.go:68] The binary conntrack is not installed, this can cause
failures in network connection cleanup.
I0330 04:09:46.702828    6048 docker_service.go:253] Docker cri networking managed by kubernetes.io/no-op
I0330 04:09:46.718906    6048 docker_service.go:258] Docker Info: &
{ID:TQ5X:T6PX:K2WX:LMX6:W44V:RPMB:DME6:AQJP:AMOD:45JW:HCD2:RCXA Containers:2 ContainersRunning:2
ContainersPaused:0 ContainersStopped:0 Images:5 Driver:overlay2 DriverStatus:[[Backing Filesystem extfs] [Supports
d_type true] [Native Overlay Diff true]] SystemStatus:[] Plugins:{Volume:[local] Network:[bridge host macvlan null
```

```
overlay] Authorization:[] Log:[awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog]}
MemoryLimit:true SwapLimit:false KernelMemory:true CPUCfsPeriod:true CPUCfsQuota:true CPUShares:true CPUSet:true
IPv4Forwarding:true BridgeNfIptables:true BridgeNfIP6tables:true Debug:false NFd:33 OomKillDisable:true
NGoroutines:46 SystemTime:2019-03-30T04:09:46.703486208Z LoggingDriver:json-file CgroupDriver:cgroupfs
NEventsListener:0 KernelVersion:4.4.0-1075-aws OperatingSystem:Ubuntu 16.04.5 LTS OSType:linux Architecture:x86_64
IndexServerAddress:https://index.docker.io/v1/ RegistryConfig:0xc0004579d0 NCPU:2 MemTotal:8369913856
GenericResources:[] DockerRootDir:/var/lib/docker HTTPProxy: HTTPSProxy: NoProxy: Name:nodea Labels:[]
ExperimentalBuild:false ServerVersion:18.09.3 ClusterStore: ClusterAdvertise: Runtimes:map[runc:{Path:runc Args:
[]}] DefaultRuntime:runc Swarm:{NodeID: NodeAddr: LocalNodeState:inactive ControlAvailable:false Error:
RemoteManagers:[] Nodes:0 Managers:0 Cluster:<nil>} LiveRestoreEnabled:false Isolation: InitBinary:docker-init
ContainerdCommit:{ID:e6b3f5632f50dbc4e9cb6288d911bf4f5e95b18e Expected:e6b3f5632f50dbc4e9cb6288d911bf4f5e95b18e}
RuncCommit:{ID:6635b4f0c6af3810594d2770f662f34ddc15b40d Expected:6635b4f0c6af3810594d2770f662f34ddc15b40d}
InitCommit:{ID:fec3683 Expected:fec3683} SecurityOptions:[name=apparmor name=seccomp,profile=default]}
I0330 04:09:46.718987    6048 docker_service.go:271] Setting cgroupDriver to cgroupfs
I0330 04:09:46.735407    6048 remote_runtime.go:62] parsed scheme: ""
I0330 04:09:46.735426    6048 remote_runtime.go:62] scheme "" not registered, fallback to default scheme
I0330 04:09:46.735453    6048 remote_image.go:50] parsed scheme: ""
I0330 04:09:46.735461    6048 remote_image.go:50] scheme "" not registered, fallback to default scheme
I0330 04:09:46.735515    6048 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 04:09:46.735530    6048 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 04:09:46.735582    6048 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc00038bf30, CONNECTING
I0330 04:09:46.735610    6048 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 04:09:46.735621    6048 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 04:09:46.735656    6048 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0004c8ad0, CONNECTING
I0330 04:09:46.735713    6048 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc00038bf30, READY
I0330 04:09:46.735755    6048 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0004c8ad0, READY
I0330 04:09:46.739877    6048 kuberuntime_manager.go:210] Container runtime docker initialized, version: 18.09.3,
apiVersion: 1.39.0
W0330 04:09:46.740093    6048 csi_plugin.go:218] kubernetes.io/csi: kubeclient not set, assuming standalone
kubelet
I0330 04:09:46.740776    6048 server.go:1037] Started kubelet
W0330 04:09:46.740810    6048 kubelet.go:1387] No api server defined - no node status update will be sent.
E0330 04:09:46.740878    6048 kubelet.go:1282] Image garbage collection failed once. Stats initialization may not
have completed yet: failed to get imageFs info: unable to find data in memory cache
I0330 04:09:46.741236    6048 fs_resource_analyzer.go:64] Starting FS ResourceAnalyzer
I0330 04:09:46.741260    6048 status_manager.go:148] Kubernetes client is nil, not starting status manager.
I0330 04:09:46.741279    6048 kubelet.go:1806] Starting kubelet main sync loop.
```

```
I0330 04:09:46.741299    6048 kubelet.go:1823] skipping pod synchronization - [container runtime status check may
not have completed yet., PLEG is not healthy: pleg has yet to be successful.]
I0330 04:09:46.741379    6048 server.go:141] Starting to listen on 0.0.0.0:10250
I0330 04:09:46.741882    6048 server.go:343] Adding debug handlers to kubelet server.
I0330 04:09:46.743226    6048 volume_manager.go:248] Starting Kubelet Volume Manager
E0330 04:09:46.748060    6048 runtime.go:69] Observed a panic: "invalid memory address or nil pointer dereference"
(runtime error: invalid memory address or nil pointer dereference)
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:76
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:65
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:51
/usr/local/go/src/runtime/panic.go:522
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:189
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:214
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:125
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:152
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:153
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:88
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:124
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:54
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:71
/usr/local/go/src/runtime/asm_amd64.s:1337
I0330 04:09:46.748131    6048 desired_state_of_world_populator.go:130] Desired state populator starts to run
I0330 04:09:46.769199    6048 clientconn.go:440] parsed scheme: "unix"
I0330 04:09:46.769222    6048 clientconn.go:440] scheme "unix" not registered, fallback to default scheme
I0330 04:09:46.769251    6048 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{unix:///run/containerd/containerd.sock 0  <nil>}]
I0330 04:09:46.769265    6048 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 04:09:46.769295    6048 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000b304e0, CONNECTING
I0330 04:09:46.769402    6048 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000b304e0, READY
I0330 04:09:46.841475    6048 kubelet.go:1823] skipping pod synchronization - container runtime status check may
not have completed yet.
I0330 04:09:46.848620    6048 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 04:09:46.850252    6048 cpu_manager.go:155] [cpumanager] starting with none policy
I0330 04:09:46.850315    6048 cpu_manager.go:156] [cpumanager] reconciling every 10s
I0330 04:09:46.850372    6048 policy_none.go:42] [cpumanager] none policy: Start
W0330 04:09:46.851409    6048 container_manager_linux.go:818] CPUAccounting not enabled for pid: 6048
W0330 04:09:46.851425    6048 container_manager_linux.go:821] MemoryAccounting not enabled for pid: 6048
I0330 04:09:46.851484    6048 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
E0330 04:09:46.868063    6048 summary_sys_containers.go:47] Failed to get system container stats for
```

```
"/user.slice/user-1000.slice/session-119.scope": failed to get cgroup stats for "/user.slice/user-
1000.slice/session-119.scope": failed to get container info for "/user.slice/user-1000.slice/session-119.scope":
unknown container "/user.slice/user-1000.slice/session-119.scope"
I0330 04:09:46.957172    6048 reconciler.go:154] Reconciler: start to sync state
E0330 04:09:47.748458    6048 runtime.go:69] Observed a panic: "invalid memory address or nil pointer dereference"
(runtime error: invalid memory address or nil pointer dereference)
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:76
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:65
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/runtime/runtime.go:51
/usr/local/go/src/runtime/panic.go:522
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:189
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:214
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:125
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:152
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:153
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:88
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/client-go/tools/cache/reflector.go:124
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:54
/home/ubuntu/k8s/_output/local/go/src/k8s.io/kubernetes/vendor/k8s.io/apimachinery/pkg/util/wait/wait.go:71
/usr/local/go/src/runtime/asm_amd64.s:1337
E0330 04:09:48.747784    6048 kuberuntime_container.go:71] Can't make a ref to pod "nginx-
nodea_default(a99a44614791402b058c084253a9e75f)", container nginx: selfLink was empty, can't make reference

...
```

At the bottom of the display. The `kubelet` is discovering containers running that it has no manifests for.

In another terminal display the running containers:

```
ubuntu@nodea:~$ docker container ls

CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
NAMES

ubuntu@nodea:~$
```

Any pods or containers running that the `kubelet` can not reconcile with the manifests it has been assigned are stopped and removed. In any version, if you run ad hoc containers using docker commands they will not have the Kubelet specific labels and the Kubelet will ignore them.

4. HTTP server

The `kubelet` has its own REST API and can be run as a standalone server when appropriate. You can request information including pod details (`/pods`) and overall node status (`/healthz`).

The REST endpoint on the `kubelet` is enabled by default but you can disable it with the *--enable-server=false* switch.

- **--enable-server=[true]** - Enable the kubelet's server

Try curling a list of pods from the Kubelet.

```
ubuntu@nodea:~/kubelet$ curl -s --insecure https://localhost:10250/pods | jq .
```

```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {},
  "items": null
}
```

```
ubuntu@nodea:~/kubelet$
```

Stop & rerun the `kubelet` with the previous IRI based PodSpec.

```
ubuntu@nodea:~/kubelet$ sudo $HOME/k8s/_output/bin/kubelet \
--manifest-url=https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.10/examples/pod

...
```

Try the pod listing again.

```
ubuntu@nodea:~/kubelet$ curl -s --insecure https://localhost:10250/pods | jq .items[].spec
```

```
{
  "containers": [
    {
      "name": "nginx",
      "image": "nginx",
      "ports": [
        {
          "containerPort": 80,
          "protocol": "TCP"
        }
      ],
      "resources": {},
      "terminationMessagePath": "/dev/termination-log",
      "terminationMessagePolicy": "File",
      "imagePullPolicy": "Always"
    }
  ],
  "restartPolicy": "Always",
  "terminationGracePeriodSeconds": 30,
  "dnsPolicy": "ClusterFirst",
  "nodeName": "nodea",
  "securityContext": {},
  "schedulerName": "default-scheduler",
  "enableServiceLinks": true
}
```

```
ubuntu@nodea:~/kubelet$
```

Next stop the kubelet and rerun it disabling the HTTP server with `--enable-server=false`.

```
ubuntu@nodea:~$ sudo $HOME/k8s/_output/bin/kubelet \
--manifest-url=https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.10/examples/pod \
--enable-server=false

...
```

The `kubelet` is running and our pod is started but the REST endpoint is down.

```
ubuntu@nodea:~/kubelet$ curl -svk https://localhost:10250/pods

*   Trying ::1...
* connect to ::1 port 10250 failed: Connection refused
*   Trying 127.0.0.1...
* connect to 127.0.0.1 port 10250 failed: Connection refused
* Failed to connect to localhost port 10250: Connection refused
* Closing connection 0
ubuntu@nodea:~/kubelet$
```

```
ubuntu@nodea:~/kubelet$ docker container ls

CONTAINER ID        IMAGE                       COMMAND                  CREATED              STATUS
PORTS               NAMES
8e5c5fe2ced9        nginx                       "nginx -g 'daemon of…"   About a minute ago   Up About a minute
k8s_nginx_nginx-nodea_default_7fd56f7a4dadfbe6b6e30602fb1e0deb_0
5e99265d933f        k8s.gcr.io/pause-amd64:3.1   "/pause"                 2 minutes ago        Up 2 minutes
k8s_POD_nginx-nodea_default_7fd56f7a4dadfbe6b6e30602fb1e0deb_0
ubuntu@nodea:~/kubelet$
```

Restart the `kubelet` with HTTP enabled (remove the `--enable-server` or set it to *true*)

```
ubuntu@nodea:~/kubelet$ sudo $HOME/k8s/_output/bin/kubelet \
--manifest-url=https://raw.githubusercontent.com/kubernetes/kubernetes/release-1.10/examples/pod \
--enable-server=true
...
```

Try to stop the nginx container owned by the kubelet via Docker.

```
ubuntu@nodea:~/kubelet$ docker container ls

CONTAINER ID        IMAGE                         COMMAND                  CREATED              STATUS
PORTS               NAMES
```

```
7a0b1acd0a95        nginx                           "nginx -g 'daemon ..."   2 minutes ago       Up 2 minutes
k8s_nginx_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_0
17fdc80a4c01        k8s.gcr.io/pause-amd64:3.1      "/pause"                 2 minutes ago       Up 2 minutes
k8s_POD_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_0
ubuntu@nodea:~/kubelet$
```

```
ubuntu@nodea:~/kubelet$ docker container kill $(docker container ls --filter=ancestor=nginx -q)

7a0b1acd0a95
ubuntu@nodea:~/kubelet$
```

```
ubuntu@nodea:~/kubelet$ docker container ls

CONTAINER ID        IMAGE                           COMMAND                  CREATED             STATUS
PORTS               NAMES
da140e7f5d1b        nginx                           "nginx -g 'daemon ..."   3 seconds ago       Up 2 seconds
k8s_nginx_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_1
17fdc80a4c01        k8s.gcr.io/pause-amd64:3.1      "/pause"                 2 minutes ago       Up 2 minutes
k8s_POD_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_0
ubuntu@nodea:~/kubelet$
```

- What happened?

Docker reports that it killed the container in question. However a new `docker container ls` shows the same nginx image running. However, if you look carefully, you will see that it is *not* the same container. You killed one container (7a0b1acd0a95 in the example) and the `kubelet` started a new copy of the image (container da140e7f5d1b in the example). The `kubelet` will *never* restart a container, it will only run new copies of the image when an old container fails.

Look at the `kubelet` log output for clues.

When the container fails, the `kubelet` checks the backoff time and if it has expired the `kubelet` tries to recreate the container. The back off ensures that the `kubelet` will not try to restart the container more than once in the backoff time window.

This behavior is consistent with the general Kubernetes philosophy, users supply the desired state and Kubernetes ensures that it is enforced as the actual state. As long as this `kubelet` has the podspec for nginx, it will make sure nginx is running.

5. Health check

The `kubelet` offers a basic health check endpoint which is used to verify reachability and liveness of the `kubelet`.

The `/healthz` path can be curled easily, try it:

```
ubuntu@nodea:~/kubelet$ curl -v 127.0.0.1:10248/healthz && echo

*   Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 10248 (#0)
> GET /healthz HTTP/1.1
> Host: 127.0.0.1:10248
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Tue, 29 Aug 2017 18:51:34 GMT
< Content-Length: 2
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host 127.0.0.1 left intact
ok
ubuntu@nodea:~/kubelet$
```

Be advised that this is a very primitive health check, it only tells you that the `kubelet` is running. You can stop the Docker daemon (crash all pods) and the `kubelet` will still return ok. This only tells you that the kubelet is ok, it says nothing about the rest of the node.

6. Spec

You can use the spec endpoint to retrieve general information about this kubelet's node.

Try it:

```
ubuntu@nodea:~/kubelet$ curl -sL 127.0.0.1:10255/spec | jq .
```

```
{
  "num_cores": 2,
  "cpu_frequency_khz": 2300062,
  "memory_capacity": 8369913856,
```

```
  "hugepages": [
    {
      "page_size": 2048,
      "num_pages": 0
    }
  ],
  "machine_id": "e53d14d788454608be05a016cbffebf6",
  "system_uuid": "EC203559-73E9-971D-B8A9-50080CBED047",
  "boot_id": "fe03b0b8-f5d2-490d-a949-faef5f3f1211",
  "filesystems": [
    {
      "device": "tmpfs",
      "capacity": 836993024,
      "type": "vfs",
      "inodes": 1021718,
      "has_inodes": true
    },
    {
      "device": "/dev/xvda1",
      "capacity": 31158935552,
      "type": "vfs",
      "inodes": 3840000,
      "has_inodes": true
    },
    {
      "device": "shm",
      "capacity": 67108864,
      "type": "vfs",
      "inodes": 1021718,
      "has_inodes": true
    }
  ],
  "disk_map": {
    "202:0": {
      "name": "xvda",
      "major": 202,
      "minor": 0,
      "size": 32212254720,
      "scheduler": "deadline"
    }
  },
  "network_devices": [
    {
```

```
      "name": "eth0",
      "mac_address": "02:ef:63:d5:3b:be",
      "speed": 0,
      "mtu": 9001
    }
  ],
  "topology": [
    {
      "node_id": 0,
      "memory": 8369913856,
      "cores": [
        {
          "core_id": 0,
          "thread_ids": [
            0
          ],
          "caches": [
            {
              "size": 32768,
              "type": "Data",
              "level": 1
            },
            {
              "size": 32768,
              "type": "Instruction",
              "level": 1
            },
            {
              "size": 262144,
              "type": "Unified",
              "level": 2
            }
          ]
        },
        {
          "core_id": 1,
          "thread_ids": [
            1
          ],
          "caches": [
            {
              "size": 32768,
              "type": "Data",
```

```
              "level": 1
            },
            {
              "size": 32768,
              "type": "Instruction",
              "level": 1
            },
            {
              "size": 262144,
              "type": "Unified",
              "level": 2
            }
          ]
        }
      ],
      "caches": [
        {
          "size": 47185920,
          "type": "Unified",
          "level": 3
        }
      ]
    }
  ],
  "cloud_provider": "AWS",
  "instance_type": "t2.large",
  "instance_id": "i-0ebfee6563638eef6"
}
```


```
ubuntu@nodea:~/kubelet$
```

Congratulations you have successfully completed the lab!