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 `--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 with a flag)
- **HTTP endpoint** - HTTP endpoint passed as a parameter on the command line, checked every 20 seconds (configurable)
- **HTTP server** - The `kubelet` can also listen for HTTP manifest posts

The `--pod-manifest-path` is typically used to tell the kubelet to start other kubernetes components, like the kube-proxy on worker nodes and 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.

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

user@nodea:~$
```

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

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

user@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:

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

user@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.

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

user@nodea:~$
```

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

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

...
user@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.

- **--pod-manifest-path=""** - Path to the pod spec file or directory of files
- **--file-check-frequency=20s** - Duration between checking config files for new data
- **--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`)

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

```
user@nodea:~$ cd

user@nodea:~$
```

```
user@nodea:~$ mkdir kubelet

user@nodea:~$
```

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

user@nodea:~/kubelet$
```

Now create a simple pod to test:

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

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

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

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

I0829 11:19:35.144861   42647 feature_gate.go:144] feature gates: map[]
W0829 11:19:35.145244   42647 server.go:496] No API client: no api servers specified
I0829 11:19:35.146049   42647 client.go:72] Connecting to docker on unix:///var/run/docker.sock
I0829 11:19:35.146069   42647 client.go:92] Start docker client with request timeout=2m0s
W0829 11:19:35.150012   42647 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 11:19:35.163655   42647 manager.go:143] cAdvisor running in container: "/user.slice"
W0829 11:19:35.184143   42647 manager.go:151] unable to connect to Rkt api service: rkt: cannot tcp Dial rkt api
service: dial tcp [::1]:15441: getsockopt: connection refused
I0829 11:19:35.193833   42647 fs.go:117] Filesystem partitions: map[/dev/sda1:{mountpoint:/var/lib/docker/aufs
major:8 minor:1 fsType:ext4 blockSize:0}]
I0829 11:19:35.197290   42647 manager.go:198] Machine: {NumCores:2 CpuFrequency:2711681 MemoryCapacity:4124880896
MachineID:6e883acc04fc7db3713776be57a3dac9 SystemUUID:F2564D56-3460-443D-57E3-836F703215A2 BootID:e21a0ef9-85e0-
4ef1-b0b9-f85c262ea596 Filesystems:[{Device:/dev/sda1 DeviceMajor:8 DeviceMinor:1 Capacity:18889830400 Type:vfs
Inodes:1179648 HasInodes:true}] DiskMap:map[8:0:{Name:sda Major:8 Minor:0 Size:21474836480 Scheduler:deadline}]
NetworkDevices:[{Name:ens33 MacAddress:00:0c:29:32:15:a2 Speed:1000 Mtu:1500}] Topology:[{Id:0 Memory:4124880896
Cores:[{Id:0 Threads:[0] Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144
Type:Unified Level:2}]}] Caches:[{Size:8388608 Type:Unified Level:3}]} {Id:2 Memory:0 Cores:[{Id:0 Threads:[1]
Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144 Type:Unified Level:2}]}]
Caches:[{Size:8388608 Type:Unified Level:3}]}] CloudProvider:Unknown InstanceType:Unknown InstanceID:None}
I0829 11:19:35.198294   42647 manager.go:204] Version: {KernelVersion:4.4.0-93-generic ContainerOsVersion:Ubuntu
16.04.1 LTS DockerVersion:17.06.1-ce DockerAPIVersion:1.30 CadvisorVersion: CadvisorRevision:}
W0829 11:19:35.200357   42647 server.go:356] No api server defined - no events will be sent to API server.
I0829 11:19:35.200373   42647 server.go:536] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
W0829 11:19:35.202322   42647 container_manager_linux.go:216] Running with swap on is not supported, please
disable swap! This will be a fatal error by default starting in K8s v1.6! In the meantime, you can opt-in to
making this a fatal error by enabling --experimental-fail-swap-on.
I0829 11:19:35.202384   42647 container_manager_linux.go:246] container manager verified user specified cgroup-
root exists: /
I0829 11:19:35.202399   42647 container_manager_linux.go:251] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs 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>}]} ExperimentalQOSReserved:map[]}
I0829 11:19:35.202698   42647 kubelet.go:263] Adding manifest file: /home/user/kubelet/pod.yaml
W0829 11:19:35.206451   42647 kubelet_network.go:70] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0829 11:19:35.206475   42647 kubelet.go:508] Hairpin mode set to "hairpin-veth"
W0829 11:19:35.210581   42647 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 11:19:35.218624   42647 docker_service.go:208] Docker cri networking managed by kubernetes.io/no-op
I0829 11:19:35.231203   42647 docker_service.go:225] Setting cgroupDriver to cgroupfs
I0829 11:19:35.245809   42647 remote_runtime.go:42] Connecting to runtime service unix:///var/run/dockershim.sock
I0829 11:19:35.247785   42647 kuberuntime_manager.go:163] Container runtime docker initialized, version: 17.06.1-
ce, apiVersion: 1.30.0
I0829 11:19:35.248966   42647 server.go:943] Started kubelet v1.7.4+793658f2d7ca7
E0829 11:19:35.249278   42647 kubelet.go:1229] Image garbage collection failed once. Stats initialization may not
have completed yet: unable to find data for container /
W0829 11:19:35.249340   42647 kubelet.go:1313] No api server defined - no node status update will be sent.
I0829 11:19:35.249581   42647 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 11:19:35.250491   42647 server.go:132] Starting to listen on 0.0.0.0:10250
I0829 11:19:35.251336   42647 server.go:310] Adding debug handlers to kubelet server.
E0829 11:19:35.258398   42647 kubelet.go:1729] Failed to check if disk space is available for the runtime: failed
to get fs info for "runtime": unable to find data for container /
E0829 11:19:35.258705   42647 kubelet.go:1737] Failed to check if disk space is available on the root partition:
failed to get fs info for "root": unable to find data for container /
E0829 11:19:35.265539   42647 docker_sandbox.go:239] Failed to stop sandbox
"b2d1c612ee8b1e63c8b3ed856c5d6e1f3af5ed9aded5b055e915b589132e8f33": Error response from daemon: {"message":"No
such container: b2d1c612ee8b1e63c8b3ed856c5d6e1f3af5ed9aded5b055e915b589132e8f33"}
I0829 11:19:35.268110   42647 fs_resource_analyzer.go:66] Starting FS ResourceAnalyzer
I0829 11:19:35.268581   42647 status_manager.go:136] Kubernetes client is nil, not starting status manager.
I0829 11:19:35.272098   42647 kubelet.go:1809] Starting kubelet main sync loop.
I0829 11:19:35.272150   42647 kubelet.go:1820] skipping pod synchronization - [container runtime is down PLEG is
not healthy: pleg was last seen active 2562047h47m16.854775807s ago; threshold is 3m0s]
W0829 11:19:35.269002   42647 container_manager_linux.go:747] CPUAccounting not enabled for pid: 42647
W0829 11:19:35.272336   42647 container_manager_linux.go:750] MemoryAccounting not enabled for pid: 42647
I0829 11:19:35.269016   42647 volume_manager.go:245] Starting Kubelet Volume Manager
E0829 11:19:35.269045   42647 container_manager_linux.go:543] [ContainerManager]: Fail to get rootfs information
unable to find data for container /
I0829 11:19:35.286710   42647 factory.go:351] Registering Docker factory
W0829 11:19:35.286739   42647 manager.go:247] Registration of the rkt container factory failed: unable to
communicate with Rkt api service: rkt: cannot tcp Dial rkt api service: dial tcp [::1]:15441: getsockopt:
connection refused
I0829 11:19:35.286802   42647 factory.go:54] Registering systemd factory
```

```
I0829 11:19:35.287007   42647 factory.go:86] Registering Raw factory
I0829 11:19:35.287741   42647 manager.go:1121] Started watching for new ooms in manager
I0829 11:19:35.288750   42647 oomparser.go:185] oomparser using systemd
I0829 11:19:35.289194   42647 manager.go:288] Starting recovery of all containers
I0829 11:19:35.360916   42647 manager.go:293] Recovery completed
I0829 11:19:35.447129   42647 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
E0829 11:19:35.450250   42647 helpers.go:771] Could not find capacity information for resource
storage.kubernetes.io/scratch
W0829 11:19:35.450297   42647 helpers.go:782] eviction manager: no observation found for eviction signal
allocatableNodeFs.available
I0829 11:19:40.279325   42647 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 11:19:40.290495   42647 kuberuntime_manager.go:457] Container {Name:nginx Image:nginx:1.7.9 Command:[] Args:
[] WorkingDir: Ports:[{Name: HostPort:0 ContainerPort:80 Protocol:TCP HostIP:}] EnvFrom:[] Env:[] Resources:
{Limits:map[] Requests:map[]} VolumeMounts:[] LivenessProbe:nil ReadinessProbe:nil Lifecycle:nil
TerminationMessagePath:/dev/termination-log TerminationMessagePolicy:File ImagePullPolicy:IfNotPresent
SecurityContext:nil Stdin:false StdinOnce:false TTY:false} is dead, but RestartPolicy says that we should restart
it.
...
```

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

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

IMAGE
nginx@sha256:e3456c851a152494c3e4ff5fcc26f240206abac0c9d794affb40e0714846c451
gcr.io/google_containers/pause-amd64:3.0

user@nodea:~$
```

Note the nginx image tag (in the YAML), it is version 1.7.9 as requested in the spec but Kubernetes and Docker 17.06 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:

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

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

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

user@nodea:~/kubelet$
```

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

```
...
E0829 11:29:32.360602   42647 file.go:72] unable to read config path "/home/user/kubelet/pod.yaml": error while
processing event ("/home/user/kubelet/pod.yaml": 0x400 == IN_DELETE_SELF): the watched path is deleted

W0829 11:29:35.273487   42647 container_manager_linux.go:747] CPUAccounting not enabled for pid: 42647
W0829 11:29:35.274890   42647 container_manager_linux.go:750] MemoryAccounting not enabled for pid: 42647
I0829 11:29:37.413328   42647 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 11:29:47.444459   42647 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 11:29:52.363954   42647 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 11:29:52.371401   42647 kuberuntime_manager.go:457] Container {Name:nginx Image:nginx:1.9.1 Command:[] Args:
[] WorkingDir: Ports:[{Name: HostPort:0 ContainerPort:80 Protocol:TCP HostIP:}] EnvFrom:[] Env:[] Resources:
{Limits:map[] Requests:map[]} VolumeMounts:[] LivenessProbe:nil ReadinessProbe:nil Lifecycle:nil
TerminationMessagePath:/dev/termination-log TerminationMessagePolicy:File ImagePullPolicy:IfNotPresent
SecurityContext:nil Stdin:false StdinOnce:false TTY:false} is dead, but RestartPolicy says that we should restart
```

```
it.
E0829 11:29:53.281663   42647 kuberuntime_container.go:59] Can't make a ref to pod "nginx-startup-
nodea_default(89beb3e314b1eb83a626cea6dcd92be7)", container nginx: selfLink was empty, can't make reference
W0829 11:29:53.291901   42647 pod_container_deletor.go:77] Container
"7a356a64bcc4e23b6e5e109e76b586775b871614632eef2ec0bed94270556f73" 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:

```
user@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

user@nodea:~/kubelet$
```

or

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

1.9.1: Pulling from library/nginx
5641bf7f839b: Already exists
a3ed95caeb02: Already exists
d003dd0d7f8a: Already exists
c5dd085dcc7c: Already exists
d95a07673dd5: Already exists
cec5c5855afe: Already exists
b315c6f2ccf3: Already exists
Digest: sha256:2f68b99bc0d6d25d0c56876b924ec20418544ff28e1fb89a4c27679a40da811b
Status: Image is up to date for nginx:1.9.1
```

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

or a mixture of the previous two outputs.

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

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

IMAGE                                                                               CREATED AT
STATUS
nginx@sha256:2f68b99bc0d6d25d0c56876b924ec20418544ff28e1fb89a4c27679a40da811b   2017-08-29 11:30:03 -0700 PDT   Up
4 minutes
gcr.io/google_containers/pause-amd64:3.0                                            2017-08-29 11:29:52 -0700 PDT   Up
4 minutes

user@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:

- **--manifest-url=""** - URL for accessing the container manifest
- **--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'
- **--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.

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

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

Clear all of the kubelet state:

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

user@nodea:~/kubelet$
```

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

```
user@nodea:~/kubelet$ curl https://raw.githubusercontent.com/kubernetes/kubernetes/master/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

user@nodea:~/kubelet$
```

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

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

I0829 11:40:29.274971   44070 feature_gate.go:144] feature gates: map[]
```

```
W0829 11:40:29.275107   44070 server.go:496] No API client: no api servers specified
I0829 11:40:29.275152   44070 client.go:72] Connecting to docker on unix:///var/run/docker.sock
I0829 11:40:29.275161   44070 client.go:92] Start docker client with request timeout=2m0s
W0829 11:40:29.277792   44070 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 11:40:29.283748   44070 manager.go:143] cAdvisor running in container: "/user.slice"
W0829 11:40:29.296253   44070 manager.go:151] unable to connect to Rkt api service: rkt: cannot tcp Dial rkt api
service: dial tcp [::1]:15441: getsockopt: connection refused
I0829 11:40:29.308595   44070 fs.go:117] Filesystem partitions: map[/dev/sda1:{mountpoint:/var/lib/docker/aufs
major:8 minor:1 fsType:ext4 blockSize:0}]
I0829 11:40:29.309828   44070 manager.go:198] Machine: {NumCores:2 CpuFrequency:2711681 MemoryCapacity:4124880896
MachineID:6e883acc04fc7db3713776be57a3dac9 SystemUUID:F2564D56-3460-443D-57E3-836F703215A2 BootID:e21a0ef9-85e0-
4ef1-b0b9-f85c262ea596 Filesystems:[{Device:/dev/sda1 DeviceMajor:8 DeviceMinor:1 Capacity:18889830400 Type:vfs
Inodes:1179648 HasInodes:true}] DiskMap:map[8:0:{Name:sda Major:8 Minor:0 Size:21474836480 Scheduler:deadline}]
NetworkDevices:[{Name:ens33 MacAddress:00:0c:29:32:15:a2 Speed:1000 Mtu:1500}] Topology:[{Id:0 Memory:4124880896
Cores:[{Id:0 Threads:[0] Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144
Type:Unified Level:2}]}] Caches:[{Size:8388608 Type:Unified Level:3}]} {Id:2 Memory:0 Cores:[{Id:0 Threads:[1]
Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144 Type:Unified Level:2}]}]
Caches:[{Size:8388608 Type:Unified Level:3}]}] CloudProvider:Unknown InstanceType:Unknown InstanceID:None}
I0829 11:40:29.310868   44070 manager.go:204] Version: {KernelVersion:4.4.0-93-generic ContainerOsVersion:Ubuntu
16.04.1 LTS DockerVersion:17.06.1-ce DockerAPIVersion:1.30 CadvisorVersion: CadvisorRevision:}
W0829 11:40:29.311508   44070 server.go:356] No api server defined - no events will be sent to API server.
I0829 11:40:29.312010   44070 server.go:536] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
W0829 11:40:29.313661   44070 container_manager_linux.go:216] Running with swap on is not supported, please
disable swap! This will be a fatal error by default starting in K8s v1.6! In the meantime, you can opt-in to
making this a fatal error by enabling --experimental-fail-swap-on.
I0829 11:40:29.313724   44070 container_manager_linux.go:246] container manager verified user specified cgroup-
root exists: /
I0829 11:40:29.313744   44070 container_manager_linux.go:251] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs 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>}]} ExperimentalQOSReserved:map[]}
I0829 11:40:29.314036   44070 kubelet.go:269] Adding manifest url
"https://raw.githubusercontent.com/kubernetes/kubernetes/master/examples/pod" with HTTP header map[]
W0829 11:40:29.316698   44070 kubelet_network.go:70] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0829 11:40:29.316792   44070 kubelet.go:508] Hairpin mode set to "hairpin-veth"
W0829 11:40:29.319984   44070 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 11:40:29.328644   44070 docker_service.go:208] Docker cri networking managed by kubernetes.io/no-op
```

```
I0829 11:40:29.341116   44070 docker_service.go:225] Setting cgroupDriver to cgroupfs
I0829 11:40:29.355454   44070 remote_runtime.go:42] Connecting to runtime service unix:///var/run/dockershim.sock
I0829 11:40:29.357024   44070 kuberuntime_manager.go:163] Container runtime docker initialized, version: 17.06.1-
ce, apiVersion: 1.30.0
I0829 11:40:29.358345   44070 server.go:943] Started kubelet v1.7.4+793658f2d7ca7
E0829 11:40:29.358549   44070 kubelet.go:1229] Image garbage collection failed once. Stats initialization may not
have completed yet: unable to find data for container /
W0829 11:40:29.358620   44070 kubelet.go:1313] No api server defined - no node status update will be sent.
I0829 11:40:29.358824   44070 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 11:40:29.359053   44070 server.go:132] Starting to listen on 0.0.0.0:10250
I0829 11:40:29.359851   44070 server.go:310] Adding debug handlers to kubelet server.
E0829 11:40:29.364089   44070 kubelet.go:1729] Failed to check if disk space is available for the runtime: failed
to get fs info for "runtime": unable to find data for container /
E0829 11:40:29.366864   44070 kubelet.go:1737] Failed to check if disk space is available on the root partition:
failed to get fs info for "root": unable to find data for container /
I0829 11:40:29.367563   44070 fs_resource_analyzer.go:66] Starting FS ResourceAnalyzer
I0829 11:40:29.367595   44070 status_manager.go:136] Kubernetes client is nil, not starting status manager.
I0829 11:40:29.367604   44070 kubelet.go:1809] Starting kubelet main sync loop.
I0829 11:40:29.367627   44070 kubelet.go:1820] skipping pod synchronization - [container runtime is down PLEG is
not healthy: pleg was last seen active 2562047h47m16.854775807s ago; threshold is 3m0s]
W0829 11:40:29.368160   44070 container_manager_linux.go:747] CPUAccounting not enabled for pid: 44070
W0829 11:40:29.368750   44070 container_manager_linux.go:750] MemoryAccounting not enabled for pid: 44070
E0829 11:40:29.368357   44070 container_manager_linux.go:543] [ContainerManager]: Fail to get rootfs information
unable to find data for container /
I0829 11:40:29.368386   44070 volume_manager.go:245] Starting Kubelet Volume Manager
E0829 11:40:29.376221   44070 docker_sandbox.go:239] Failed to stop sandbox
"41ac78587353d8123c00e20352c4191e9c609e2fc3306142b5c0da07e9076f13": Error response from daemon: {"message":"No
such container: 41ac78587353d8123c00e20352c4191e9c609e2fc3306142b5c0da07e9076f13"}
I0829 11:40:29.390103   44070 factory.go:351] Registering Docker factory
W0829 11:40:29.390359   44070 manager.go:247] Registration of the rkt container factory failed: unable to
communicate with Rkt api service: rkt: cannot tcp Dial rkt api service: dial tcp [::1]:15441: getsockopt:
connection refused
I0829 11:40:29.390541   44070 factory.go:54] Registering systemd factory
I0829 11:40:29.390962   44070 factory.go:86] Registering Raw factory
I0829 11:40:29.391326   44070 manager.go:1121] Started watching for new ooms in manager
I0829 11:40:29.391732   44070 oomparser.go:185] oomparser using systemd
I0829 11:40:29.392138   44070 manager.go:288] Starting recovery of all containers
I0829 11:40:29.465837   44070 manager.go:293] Recovery completed
I0829 11:40:29.558781   44070 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
E0829 11:40:29.561975   44070 helpers.go:771] Could not find capacity information for resource
storage.kubernetes.io/scratch
```

```
W0829 11:40:29.562025   44070 helpers.go:782] eviction manager: no observation found for eviction signal
allocatableNodeFs.available
I0829 11:40:34.369250   44070 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
W0829 11:40:34.371578   44070 pod_container_deletor.go:77] Container
"41ac78587353d8123c00e20352c4191e9c609e2fc3306142b5c0da07e9076f13" not found in pod's containers
I0829 11:40:34.377291   44070 kuberuntime_manager.go:457] Container {Name:nginx Image:nginx Command:[] Args:[]
WorkingDir: Ports:[{Name: HostPort:0 ContainerPort:80 Protocol:TCP HostIP:}] EnvFrom:[] Env:[] Resources:
{Limits:map[] Requests:map[]} VolumeMounts:[] LivenessProbe:nil ReadinessProbe:nil Lifecycle:nil
TerminationMessagePath:/dev/termination-log TerminationMessagePolicy:File ImagePullPolicy:Always
SecurityContext:nil Stdin:false StdinOnce:false TTY:false} is dead, but RestartPolicy says that we should restart
it.
I0829 11:40:39.592182   44070 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
...
```

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

> I0829 11:40:29.314036 44070 kubelet.go:269] Adding manifest url "https://raw.githubusercontent.com/kubernetes/kubernetes/master/examples/pod" with HTTP header map[]

Examine the containers running through Docker.

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

IMAGE                                                                           CREATED AT
STATUS
nginx@sha256:788fa27763db6d69ad3444e8ba72f947df9e7e163bad7c1f5614f8fd27a311c3   2017-08-29 11:40:35 -0700 PDT   Up
2 minutes
gcr.io/google_containers/pause-amd64:3.0                                        2017-08-29 11:40:34 -0700 PDT   Up
2 minutes

user@nodea:~/kubelet$
```

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

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

```
IMAGE                                                                            CREATED AT
STATUS
nginx@sha256:788fa27763db6d69ad3444e8ba72f947df9e7e163bad7c1f5614f8fd27a311c3   2017-08-29 11:40:35 -0700 PDT   Up
3 minutes
gcr.io/google_containers/pause-amd64:3.0                                         2017-08-29 11:40:34 -0700 PDT   Up
3 minutes

user@nodea:~/kubelet$
```

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

Display the labels associated with the nginx container:

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

{
  "annotation.io.kubernetes.container.hash": "30f2a656",
  "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/ab4f45926942575bebaa13c947218fce/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": "ab4f45926942575bebaa13c947218fce",
  "io.kubernetes.sandbox.id": "ce2d0a2474457925ae2d4d28b1fc273954bc19faa59ab776e34365877a2a2e17"
}
user@nodea:~/kubelet$
```

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

Restart the `kubelet` with no arguments:

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

```
I0829 11:44:57.624360   44351 feature_gate.go:144] feature gates: map[]
W0829 11:44:57.624721   44351 server.go:496] No API client: no api servers specified
I0829 11:44:57.625019   44351 client.go:72] Connecting to docker on unix:///var/run/docker.sock
I0829 11:44:57.625280   44351 client.go:92] Start docker client with request timeout=2m0s
W0829 11:44:57.626978   44351 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 11:44:57.641173   44351 manager.go:143] cAdvisor running in container: "/user.slice"
W0829 11:44:57.655334   44351 manager.go:151] unable to connect to Rkt api service: rkt: cannot tcp Dial rkt api
service: dial tcp [::1]:15441: getsockopt: connection refused
I0829 11:44:57.669240   44351 fs.go:117] Filesystem partitions: map[/dev/sda1:{mountpoint:/var/lib/docker/aufs
major:8 minor:1 fsType:ext4 blockSize:0}]
I0829 11:44:57.672457   44351 manager.go:198] Machine: {NumCores:2 CpuFrequency:2711681 MemoryCapacity:4124880896
MachineID:6e883acc04fc7db3713776be57a3dac9 SystemUUID:F2564D56-3460-443D-57E3-836F703215A2 BootID:e21a0ef9-85e0-
4ef1-b0b9-f85c262ea596 Filesystems:[{Device:/dev/sda1 DeviceMajor:8 DeviceMinor:1 Capacity:18889830400 Type:vfs
Inodes:1179648 HasInodes:true}] DiskMap:map[8:0:{Name:sda Major:8 Minor:0 Size:21474836480 Scheduler:deadline}]
NetworkDevices:[{Name:ens33 MacAddress:00:0c:29:32:15:a2 Speed:1000 Mtu:1500}] Topology:[{Id:0 Memory:4124880896
Cores:[{Id:0 Threads:[0] Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144
Type:Unified Level:2}]}] Caches:[{Size:8388608 Type:Unified Level:3}]} {Id:2 Memory:0 Cores:[{Id:0 Threads:[1]
Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144 Type:Unified Level:2}]}]
Caches:[{Size:8388608 Type:Unified Level:3}]}] CloudProvider:Unknown InstanceType:Unknown InstanceID:None}
I0829 11:44:57.673578   44351 manager.go:204] Version: {KernelVersion:4.4.0-93-generic ContainerOsVersion:Ubuntu
16.04.1 LTS DockerVersion:17.06.1-ce DockerAPIVersion:1.30 CadvisorVersion: CadvisorRevision:}
W0829 11:44:57.674146   44351 server.go:356] No api server defined - no events will be sent to API server.
I0829 11:44:57.674176   44351 server.go:536] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
W0829 11:44:57.675237   44351 container_manager_linux.go:216] Running with swap on is not supported, please
disable swap! This will be a fatal error by default starting in K8s v1.6! In the meantime, you can opt-in to
making this a fatal error by enabling --experimental-fail-swap-on.
I0829 11:44:57.675396   44351 container_manager_linux.go:246] container manager verified user specified cgroup-
root exists: /
I0829 11:44:57.675491   44351 container_manager_linux.go:251] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs 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>}]} ExperimentalQOSReserved:map[]}
W0829 11:44:57.677961   44351 kubelet_network.go:70] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0829 11:44:57.678012   44351 kubelet.go:508] Hairpin mode set to "hairpin-veth"
W0829 11:44:57.680324   44351 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 11:44:57.689597   44351 docker_service.go:208] Docker cri networking managed by kubernetes.io/no-op
I0829 11:44:57.707612   44351 docker_service.go:225] Setting cgroupDriver to cgroupfs
```

```
I0829 11:44:57.727897   44351 remote_runtime.go:42] Connecting to runtime service unix:///var/run/dockershim.sock
I0829 11:44:57.729464   44351 kuberuntime_manager.go:163] Container runtime docker initialized, version: 17.06.1-
ce, apiVersion: 1.30.0
I0829 11:44:57.730939   44351 server.go:943] Started kubelet v1.7.4+793658f2d7ca7
E0829 11:44:57.730957   44351 kubelet.go:1229] Image garbage collection failed once. Stats initialization may not
have completed yet: unable to find data for container /
W0829 11:44:57.731025   44351 kubelet.go:1313] No api server defined - no node status update will be sent.
I0829 11:44:57.731102   44351 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 11:44:57.731239   44351 server.go:132] Starting to listen on 0.0.0.0:10250
I0829 11:44:57.731827   44351 server.go:310] Adding debug handlers to kubelet server.
E0829 11:44:57.735584   44351 kubelet.go:1729] Failed to check if disk space is available for the runtime: failed
to get fs info for "runtime": unable to find data for container /
E0829 11:44:57.735614   44351 kubelet.go:1737] Failed to check if disk space is available on the root partition:
failed to get fs info for "root": unable to find data for container /
I0829 11:44:57.736439   44351 fs_resource_analyzer.go:66] Starting FS ResourceAnalyzer
I0829 11:44:57.736470   44351 status_manager.go:136] Kubernetes client is nil, not starting status manager.
I0829 11:44:57.736477   44351 kubelet.go:1809] Starting kubelet main sync loop.
I0829 11:44:57.736505   44351 kubelet.go:1820] skipping pod synchronization - [container runtime is down PLEG is
not healthy: pleg was last seen active 2562047h47m16.854775807s ago; threshold is 3m0s]
I0829 11:44:57.736708   44351 volume_manager.go:245] Starting Kubelet Volume Manager
W0829 11:44:57.737327   44351 container_manager_linux.go:747] CPUAccounting not enabled for pid: 44351
W0829 11:44:57.737509   44351 container_manager_linux.go:750] MemoryAccounting not enabled for pid: 44351
E0829 11:44:57.737776   44351 container_manager_linux.go:543] [ContainerManager]: Fail to get rootfs information
unable to find data for container /
I0829 11:44:57.756019   44351 factory.go:351] Registering Docker factory
W0829 11:44:57.756298   44351 manager.go:247] Registration of the rkt container factory failed: unable to
communicate with Rkt api service: rkt: cannot tcp Dial rkt api service: dial tcp [::1]:15441: getsockopt:
connection refused
I0829 11:44:57.756487   44351 factory.go:54] Registering systemd factory
I0829 11:44:57.756996   44351 factory.go:86] Registering Raw factory
I0829 11:44:57.757476   44351 manager.go:1121] Started watching for new ooms in manager
I0829 11:44:57.759187   44351 oomparser.go:185] oomparser using systemd
I0829 11:44:57.759771   44351 manager.go:288] Starting recovery of all containers
I0829 11:44:57.821291   44351 manager.go:293] Recovery completed
I0829 11:44:57.918714   44351 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
E0829 11:44:57.921871   44351 helpers.go:771] Could not find capacity information for resource
storage.kubernetes.io/scratch
W0829 11:44:57.921945   44351 helpers.go:782] eviction manager: no observation found for eviction signal
allocatableNodeFs.available
E0829 11:45:02.852311   44351 kuberuntime_container.go:59] Can't make a ref to pod "nginx-
nodea_default(ab4f45926942575bebaa13c947218fce)", container nginx: selfLink was empty, can't make reference
```

```
W0829 11:45:02.952758   44351 docker_sandbox.go:342] failed to read pod IP from plugin/docker: Couldn't find
network status for default/nginx-nodea through plugin: invalid network status for
E0829 11:45:03.742881   44351 kuberuntime_container.go:59] Can't make a ref to pod "nginx-
nodea_default(ab4f45926942575bebaa13c947218fce)", container nginx: selfLink was empty, can't make reference
W0829 11:45:03.809963   44351 pod_container_deletor.go:77] Container
"ce2d0a2474457925ae2d4d28b1fc273954bc19faa59ab776e34365877a2a2e17" not found in pod's containers
I0829 11:45:07.948608   44351 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
...
```

Notice the Warning log output

> "nginx-nodea_default(ab4f45926942575bebaa13c947218fce)", 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:

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

CONTAINER ID        IMAGE          COMMAND        CREATED        STATUS          PORTS           NAMES

user@nodea:~$
```

The `kubelet` takes ownership of the node, seriously, any pods running that the `kubelet` can not reconcile with the manifests it has been assigned are stopped and removed. 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.

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

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {},
  "items": null
}
user@nodea:~/kubelet$
```

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

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

Try the pod listing again.

```
user@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"
}

user@nodea:~/kubelet$
```

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

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

...
```

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

```
user@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

user@nodea:~/kubelet$
```

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

CONTAINER ID        IMAGE                                      COMMAND                  CREATED
STATUS              PORTS               NAMES
7a0b1acd0a95        nginx                                      "nginx -g 'daemon ..."   About a minute ago   Up
About a minute                          k8s_nginx_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_0
17fdc80a4c01        gcr.io/google_containers/pause-amd64:3.0   "/pause"                 About a minute ago   Up
About a minute                          k8s_POD_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_0
```

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

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

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

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

```
user@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        gcr.io/google_containers/pause-amd64:3.0   "/pause"                 2 minutes ago       Up 2
minutes                                k8s_POD_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_0

user@nodea:~/kubelet$
```

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

7a0b1acd0a95

user@nodea:~/kubelet$
```

```
user@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        gcr.io/google_containers/pause-amd64:3.0   "/pause"                 2 minutes ago       Up 2
minutes                              k8s_POD_nginx-nodea_default_ab4f45926942575bebaa13c947218fce_0

user@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:

```
user@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

user@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:

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

{
  "num_cores": 2,
  "cpu_frequency_khz": 2711681,
  "memory_capacity": 4124880896,
  "machine_id": "6e883acc04fc7db3713776be57a3dac9",
  "system_uuid": "F2564D56-3460-443D-57E3-836F703215A2",
  "boot_id": "e21a0ef9-85e0-4ef1-b0b9-f85c262ea596",
  "filesystems": [
    {
      "device": "/dev/sda1",
      "capacity": 18889830400,
      "type": "vfs",
      "inodes": 1179648,
      "has_inodes": true
    }
  ],
  "disk_map": {
    "8:0": {
      "name": "sda",
      "major": 8,
      "minor": 0,
      "size": 21474836480,
      "scheduler": "deadline"
    }
  },
  "network_devices": [
    {
      "name": "ens33",
      "mac_address": "00:0c:29:32:15:a2",
      "speed": 1000,
      "mtu": 1500
    }
  ],
  "topology": [
    {
      "node_id": 0,
      "memory": 4124880896,
      "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
            }
          ]
        }
      ],
      "caches": [
        {
          "size": 8388608,
          "type": "Unified",
          "level": 3
        }
      ]
    },
    {
      "node_id": 2,
      "memory": 0,
      "cores": [
        {
          "core_id": 0,
          "thread_ids": [
            1
          ],
          "caches": [
            {
              "size": 32768,
              "type": "Data",
              "level": 1
            },
            {
              "size": 32768,
              "type": "Instruction",
              "level": 1
            },
            {
              "size": 262144,
              "type": "Unified",
              "level": 2
            }
          ]
        }
      ],
      "caches": [
        {
          "size": 8388608,
          "type": "Unified",
          "level": 3
        }
      ]
    }
  ],
  "cloud_provider": "Unknown",
  "instance_type": "Unknown",
  "instance_id": "None"
}
```

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

Congratulations you have successfully completed the kubelet lab!