RX-M Cloud Native Consulting

Advanced Kubernetes

Lab 3 – Scheduler

Per the k8s reference documentation, the scheduler is described as follows:

> The Kubernetes scheduler is a policy-rich, topology-aware, workload-specific function that significantly impacts availability, performance, and capacity. The scheduler needs to take into account individual and collective resource requirements, quality of service requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, deadlines, and so on. Workload-specific requirements will be exposed through the API as necessary.

http://kubernetes.io/docs/admin/kube-scheduler/

In this lab we will see how the scheduler affects placement of pods.

1. Deploy a two node cluster

In order to demonstrate multi-node cluster operations and pod scheduling we will set up a second node called *nodeb* (The master node was configured in lab 1 and was called *nodea*).

2. Update IPs and hostnames

Set the host name for the new VM to *nodeb*:

```
laptop$ ssh -i k8s-adv-student.pem ubuntu@<external-ip>

...

ubuntu@nodeb:~$ sudo hostnamectl set-hostname nodeb
```

```
ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ hostname

nodeb
ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ cat /etc/hostname

nodeb
ubuntu@nodeb:~$
```

Now discover your IP address (typically eth0 or ens33):

```
ubuntu@nodeb:~$ ip a show eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 02:d8:c0:66:a6:b8 brd ff:ff:ff:ff:ff:ff
    inet 172.31.30.148/20 brd 172.31.31.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::d8:c0ff:fe66:a6b8/64 scope link
       valid_lft forever preferred_lft forever
ubuntu@nodeb:~$
```

Add your IP address and host name to `/etc/hosts`, also add *nodea*'s information and remove any references to the ubuntu hostname. In a more sophisticated setting, DNS could be used to perform hostname lookups.

```
ubuntu@nodeb:~$ sudo vim /etc/hosts
ubuntu@nodeb:~$ cat /etc/hosts

127.0.0.1       localhost
172.31.30.148 nodeb
172.31.28.198 nodea
```

```
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

You may need to exit the current shell and open a new shell for your prompt (PS1) to update to the new hostname.

Now go back to *nodea*, add *nodeb*'s IP information to the `/etc/hosts` file. Depending on the hypervisor and technique used your IPs may differ.

Finally, verify that you can reach the internet and both nodes by name with ping from both VMs:

```
ubuntu@nodeb:~$ ping -c 2 yahoo.com

PING yahoo.com (98.137.246.7) 56(84) bytes of data.
64 bytes from media-router-fp1.prod1.media.vip.gq1.yahoo.com (98.137.246.7): icmp_seq=1 ttl=43 time=11.7 ms
64 bytes from media-router-fp1.prod1.media.vip.gq1.yahoo.com (98.137.246.7): icmp_seq=2 ttl=43 time=11.8 ms

--- yahoo.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 11.741/11.776/11.811/0.035 ms
ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ ping -c 2 nodea

PING nodea (172.31.28.198) 56(84) bytes of data.
64 bytes from nodea (172.31.28.198): icmp_seq=1 ttl=64 time=0.370 ms
64 bytes from nodea (172.31.28.198): icmp_seq=2 ttl=64 time=0.391 ms

--- nodea ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.370/0.380/0.391/0.022 ms
ubuntu@nodeb:~$
```

```
ubuntu@nodea:~$ ping -c 2 nodeb

PING nodeb (172.31.30.148) 56(84) bytes of data.
64 bytes from nodeb (172.31.30.148): icmp_seq=1 ttl=64 time=0.451 ms
64 bytes from nodeb (172.31.30.148): icmp_seq=2 ttl=64 time=0.410 ms

--- nodeb ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.410/0.430/0.451/0.029 ms
ubuntu@nodea:~$
```

If you can not resolve public DNS names or reach the internet, debug your connectivity before continuing.

3. Install Docker

Every k8s node will need Docker installed. We have already installed Docker on *nodea*, now do the same for *nodeb*. We will use a short cut script supplied by docker:

> Note: if you get errors regarding dpkg your system is probably updating, wait a few minutes and try again.

```
ubuntu@nodeb:~$ sudo apt-get update

ubuntu@nodeb:~$ sudo apt-get -y install apt-transport-https ca-certificates curl

ubuntu@nodeb:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

ubuntu@nodeb:~$ sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

ubuntu@nodeb:~$ sudo apt-get update

ubuntu@nodeb:~$ sudo apt-get -y install docker-ce

ubuntu@nodeb:~$ sudo usermod -aG docker ubuntu

ubuntu@nodeb:~$ exit
```

```
laptop$
```

- Does this node need to have the same version of Docker as other nodes?

The answer is no; only the kubelet on that node talks to Docker so in theory every node could have a different version of Docker. In practice it is easier to manage and debug a cluster with the same version of Docker everywhere. Some upgrade Docker versions progressively (e.g. 10% of the nodes per day) to limit the impact of latent defects or incompatibilities.

4. Verify Docker operation

When the system comes back up login and check the version of all parts of the Docker platform with the `docker version` subcommand:

```
laptop$ ssh -i k8s-adv-student.pem ubuntu@<external-ip>

...

ubuntu@nodeb:~$ docker version

Client:
 Version:           18.09.4
 API version:       1.39
 Go version:        go1.10.8
 Git commit:        d14af54
 Built:             Wed Mar 27 18:34:51 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          18.09.4
  API version:      1.39 (minimum version 1.12)
  Go version:       go1.10.8
  Git commit:       d14af54
  Built:            Wed Mar 27 18:01:48 2019
  OS/Arch:          linux/amd64
  Experimental:     false
ubuntu@nodeb:~$
```

5. Pod placement without the scheduler

To begin, we need to restart all of the previously configured parts of the k8s cluster (etcd, kube-apiserver, kubelet).

On nodea

Stop all Kubernetes services and etcd (^C them as needed).

Clear the etcd and kubelet state caches, along with Docker containers:

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

ubuntu@nodea:~$
```

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

ubuntu@nodea:~$
```

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

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

Start a fresh etcd:

```
ubuntu@nodea:~$ etcd

2019-03-30 07:44:12.491555 I | etcdmain: etcd Version: 3.3.10
2019-03-30 07:44:12.491597 I | etcdmain: Git SHA: 27fc7e2
2019-03-30 07:44:12.491612 I | etcdmain: Go Version: go1.10.4
2019-03-30 07:44:12.491620 I | etcdmain: Go OS/Arch: linux/amd64
2019-03-30 07:44:12.491629 I | etcdmain: setting maximum number of CPUs to 2, total number of available CPUs is 2
2019-03-30 07:44:12.491640 W | etcdmain: no data-dir provided, using default data-dir ./default.etcd
2019-03-30 07:44:12.491901 I | embed: listening for peers on http://localhost:2380
```

```
2019-03-30 07:44:12.491953 I | embed: listening for client requests on localhost:2379
2019-03-30 07:44:12.495983 I | etcdserver: name = default
2019-03-30 07:44:12.495999 I | etcdserver: data dir = default.etcd
2019-03-30 07:44:12.496004 I | etcdserver: member dir = default.etcd/member
2019-03-30 07:44:12.496013 I | etcdserver: heartbeat = 100ms
2019-03-30 07:44:12.496019 I | etcdserver: election = 1000ms
2019-03-30 07:44:12.496027 I | etcdserver: snapshot count = 100000
2019-03-30 07:44:12.496043 I | etcdserver: advertise client URLs = http://localhost:2379
2019-03-30 07:44:12.496052 I | etcdserver: initial advertise peer URLs = http://localhost:2380
2019-03-30 07:44:12.496064 I | etcdserver: initial cluster = default=http://localhost:2380
2019-03-30 07:44:12.499317 I | etcdserver: starting member 8e9e05c52164694d in cluster cdf818194e3a8c32
2019-03-30 07:44:12.499345 I | raft: 8e9e05c52164694d became follower at term 0
2019-03-30 07:44:12.499359 I | raft: newRaft 8e9e05c52164694d [peers: [], term: 0, commit: 0, applied: 0,
lastindex: 0, lastterm: 0]
2019-03-30 07:44:12.499368 I | raft: 8e9e05c52164694d became follower at term 1
2019-03-30 07:44:12.503658 W | auth: simple token is not cryptographically signed
2019-03-30 07:44:12.506058 I | etcdserver: starting server... [version: 3.3.10, cluster version: to_be_decided]
2019-03-30 07:44:12.506788 I | etcdserver: 8e9e05c52164694d as single-node; fast-forwarding 9 ticks (election
ticks 10)
2019-03-30 07:44:12.507107 I | etcdserver/membership: added member 8e9e05c52164694d [http://localhost:2380] to
cluster cdf818194e3a8c32
2019-03-30 07:44:13.499644 I | raft: 8e9e05c52164694d is starting a new election at term 1
2019-03-30 07:44:13.499683 I | raft: 8e9e05c52164694d became candidate at term 2
2019-03-30 07:44:13.499710 I | raft: 8e9e05c52164694d received MsgVoteResp from 8e9e05c52164694d at term 2
2019-03-30 07:44:13.499729 I | raft: 8e9e05c52164694d became leader at term 2
2019-03-30 07:44:13.499741 I | raft: raft.node: 8e9e05c52164694d elected leader 8e9e05c52164694d at term 2
2019-03-30 07:44:13.500035 I | etcdserver: published {Name:default ClientURLs:[http://localhost:2379]} to cluster
cdf818194e3a8c32
2019-03-30 07:44:13.500162 I | etcdserver: setting up the initial cluster version to 3.3
2019-03-30 07:44:13.500208 I | embed: ready to serve client requests
2019-03-30 07:44:13.500253 E | etcdmain: forgot to set Type=notify in systemd service file?
2019-03-30 07:44:13.500750 N | embed: serving insecure client requests on 127.0.0.1:2379, this is strongly
discouraged!
2019-03-30 07:44:13.500838 N | etcdserver/membership: set the initial cluster version to 3.3
2019-03-30 07:44:13.500937 I | etcdserver/api: enabled capabilities for version 3.3
```

Restart the kube-apiserver:

```
ubuntu@nodea:~$ sudo $HOME/k8s/_output/bin/kube-apiserver \
--etcd-servers=http://localhost:2379 \
--allow-privileged=true \
--service-cluster-ip-range=10.0.0.0/16 \
--insecure-bind-address=0.0.0.0 \
--disable-admission-plugins=ServiceAccount

Flag --insecure-bind-address has been deprecated, This flag will be removed in a future version.
I0330 07:45:27.185457    7701 server.go:559] external host was not specified, using 172.31.28.198
W0330 07:45:27.185503    7701 authentication.go:415] AnonymousAuth is not allowed with the AlwaysAllow authorizer.
Resetting AnonymousAuth to false. You should use a different authorizer
I0330 07:45:27.185677    7701 server.go:146] Version: v1.14.0
I0330 07:45:27.635388    7701 plugins.go:158] Loaded 7 mutating admission controller(s) successfully in the
following order:
NamespaceLifecycle,LimitRanger,TaintNodesByCondition,Priority,DefaultTolerationSeconds,DefaultStorageClass,Mutatin
gAdmissionWebhook.
I0330 07:45:27.635414    7701 plugins.go:161] Loaded 5 validating admission controller(s) successfully in the
following order: LimitRanger,Priority,PersistentVolumeClaimResize,ValidatingAdmissionWebhook,ResourceQuota.
E0330 07:45:27.637043    7701 prometheus.go:138] failed to register depth metric admission_quota_controller:
duplicate metrics collector registration attempted
E0330 07:45:27.637077    7701 prometheus.go:150] failed to register adds metric admission_quota_controller:
duplicate metrics collector registration attempted
E0330 07:45:27.637115    7701 prometheus.go:162] failed to register latency metric admission_quota_controller:
duplicate metrics collector registration attempted
E0330 07:45:27.637150    7701 prometheus.go:174] failed to register work_duration metric
admission_quota_controller: duplicate metrics collector registration attempted
E0330 07:45:27.637201    7701 prometheus.go:189] failed to register unfinished_work_seconds metric
admission_quota_controller: duplicate metrics collector registration attempted
E0330 07:45:27.637243    7701 prometheus.go:202] failed to register longest_running_processor_microseconds metric
admission_quota_controller: duplicate metrics collector registration attempted
I0330 07:45:27.637264    7701 plugins.go:158] Loaded 7 mutating admission controller(s) successfully in the
following order:
NamespaceLifecycle,LimitRanger,TaintNodesByCondition,Priority,DefaultTolerationSeconds,DefaultStorageClass,Mutatin
gAdmissionWebhook.
I0330 07:45:27.637275    7701 plugins.go:161] Loaded 5 validating admission controller(s) successfully in the
following order: LimitRanger,Priority,PersistentVolumeClaimResize,ValidatingAdmissionWebhook,ResourceQuota.

...
```

Restart the nodea `kubelet`:

```
ubuntu@nodea:~$ sudo $HOME/k8s/_output/bin/kubelet \
--kubeconfig=nodea.conf \
--config=nodea.yaml \
--allow-privileged=true \
--runtime-cgroups=/systemd/machine.slice \
--kubelet-cgroups=/systemd/machine.slice \
--pod-infra-container-image=k8s.gcr.io/pause:3.1

Flag --allow-privileged has been deprecated, will be removed in a future version
Flag --kubelet-cgroups 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.
Flag --allow-privileged has been deprecated, will be removed in a future version
Flag --kubelet-cgroups 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 07:46:25.830948    7713 server.go:417] Version: v1.14.0
I0330 07:46:25.832351    7713 plugins.go:103] No cloud provider specified.
I0330 07:46:25.872819    7713 server.go:625] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
I0330 07:46:25.873090    7713 container_manager_linux.go:261] container manager verified user specified cgroup-
root exists: []
I0330 07:46:25.873111    7713 container_manager_linux.go:266] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName:/systemd/machine.slice SystemCgroupsName: KubeletCgroupsName:/systemd/machine.slice
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 07:46:25.873200    7713 container_manager_linux.go:286] Creating device plugin manager: true
I0330 07:46:25.873271    7713 state_mem.go:36] [cpumanager] initializing new in-memory state store
I0330 07:46:25.875736    7713 kubelet.go:304] Watching apiserver
I0330 07:46:25.878558    7713 client.go:75] Connecting to docker on unix:///var/run/docker.sock
I0330 07:46:25.878581    7713 client.go:104] Start docker client with request timeout=2m0s
W0330 07:46:25.880560    7713 docker_service.go:561] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0330 07:46:25.880584    7713 docker_service.go:238] Hairpin mode set to "hairpin-veth"
W0330 07:46:25.880739    7713 cni.go:213] Unable to update cni config: No networks found in /etc/cni/net.d
W0330 07:46:25.882214    7713 hostport_manager.go:68] The binary conntrack is not installed, this can cause
failures in network connection cleanup.
I0330 07:46:25.883177    7713 docker_service.go:253] Docker cri networking managed by kubernetes.io/no-op
```

```
I0330 07:46:25.899118    7713 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:2019-03-30T07:46:25.883814796Z 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:0xc00074ae00 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 07:46:25.899195    7713 docker_service.go:271] Setting cgroupDriver to cgroupfs
I0330 07:46:25.917301    7713 remote_runtime.go:62] parsed scheme: ""
I0330 07:46:25.917321    7713 remote_runtime.go:62] scheme "" not registered, fallback to default scheme
I0330 07:46:25.917350    7713 remote_image.go:50] parsed scheme: ""
I0330 07:46:25.917356    7713 remote_image.go:50] scheme "" not registered, fallback to default scheme
I0330 07:46:25.917545    7713 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 07:46:25.917546    7713 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 07:46:25.917559    7713 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 07:46:25.917568    7713 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 07:46:25.917594    7713 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0001fda10, CONNECTING
I0330 07:46:25.917599    7713 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc00023bc50, CONNECTING
I0330 07:46:25.918704    7713 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc00023bc50, READY
I0330 07:46:25.919464    7713 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0001fda10, READY
I0330 07:46:25.922629    7713 kuberuntime_manager.go:210] Container runtime docker initialized, version: 18.09.3,
apiVersion: 1.39.0
I0330 07:46:25.923918    7713 server.go:1037] Started kubelet
E0330 07:46:25.924215    7713 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 07:46:25.924716    7713 fs_resource_analyzer.go:64] Starting FS ResourceAnalyzer
I0330 07:46:25.924746    7713 status_manager.go:152] Starting to sync pod status with apiserver
```

```
I0330 07:46:25.924761    7713 kubelet.go:1806] Starting kubelet main sync loop.
I0330 07:46:25.924778    7713 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 07:46:25.924865    7713 server.go:141] Starting to listen on 0.0.0.0:10250
I0330 07:46:25.925483    7713 server.go:343] Adding debug handlers to kubelet server.
I0330 07:46:25.927000    7713 volume_manager.go:248] Starting Kubelet Volume Manager
I0330 07:46:25.928060    7713 desired_state_of_world_populator.go:130] Desired state populator starts to run
E0330 07:46:25.948811    7713 controller.go:194] failed to get node "nodea" when trying to set owner ref to the
node lease: nodes "nodea" not found
I0330 07:46:25.953124    7713 clientconn.go:440] parsed scheme: "unix"
I0330 07:46:25.953277    7713 clientconn.go:440] scheme "unix" not registered, fallback to default scheme
I0330 07:46:25.953312    7713 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{unix:///run/containerd/containerd.sock 0  <nil>}]
I0330 07:46:25.953327    7713 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 07:46:25.953363    7713 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000b14660, CONNECTING
I0330 07:46:25.953517    7713 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000b14660, READY
I0330 07:46:26.023125    7713 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 07:46:26.025001    7713 cpu_manager.go:155] [cpumanager] starting with none policy
I0330 07:46:26.025018    7713 cpu_manager.go:156] [cpumanager] reconciling every 10s
I0330 07:46:26.025034    7713 policy_none.go:42] [cpumanager] none policy: Start
W0330 07:46:26.025633    7713 manager.go:538] Failed to retrieve checkpoint for "kubelet_internal_checkpoint":
checkpoint is not found
W0330 07:46:26.026074    7713 container_manager_linux.go:818] CPUAccounting not enabled for pid: 7713
W0330 07:46:26.026090    7713 container_manager_linux.go:821] MemoryAccounting not enabled for pid: 7713
E0330 07:46:26.026520    7713 eviction_manager.go:247] eviction manager: failed to get summary stats: failed to
get node info: node "nodea" not found
I0330 07:46:26.038838    7713 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
W0330 07:46:26.039128    7713 pod_container_deletor.go:75] Container
"85176a97712ee4859924d45d113c4e25a8c8828a3a456409841df57633f8b8c4" not found in pod's containers
E0330 07:46:26.039165    7713 kubelet.go:2244] node "nodea" not found
I0330 07:46:26.040942    7713 kubelet_node_status.go:72] Attempting to register node nodea
I0330 07:46:26.043668    7713 kubelet_node_status.go:75] Successfully registered node nodea
I0330 07:46:26.138857    7713 reconciler.go:154] Reconciler: start to sync state

...
```

Verify the cluster (with one node so far). Before we can use the kubectl command we need to specify the cluster we want to interact with, again substitute your cluster master IP in the example below:

```
ubuntu@nodea:~$ kubectl config set-cluster local --server=http://nodea:8080

Cluster "local" set.
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl config set-context local --cluster=local

Context "local" created.
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl config use-context local

Switched to context "local".
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl get nodes

NAME    STATUS   ROLES    AGE   VERSION
nodea   Ready    <none>   81s   v1.14.0
ubuntu@nodea:~$
```

Now let recreate our simple Pod on *nodea* (from lab 1), as a reminder, this is the yaml:

```
ubuntu@nodea:~$ cat testpod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: nodea
  automountServiceAccountToken: false
  containers:
  - name: nginx
```

```
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /var/log/nginx
      name: nginx-logs
  - name: log-truncator
    image: busybox
    command:
    - /bin/sh
    args: [-c, 'while true; do cat /dev/null > /logdir/access.log; sleep 10; done']
    volumeMounts:
    - mountPath: /logdir
      name: nginx-logs
  volumes:
  - name: nginx-logs
    emptyDir: {}
ubuntu@nodea:~$
```

Deploy your pod via create subcommand.

```
ubuntu@nodea:~$ kubectl create -f testpod.yaml

pod/nginx created
ubuntu@nodea:~$
```

Confirm your pod has entered the *Running* state via `kubectl get pod`.

```
ubuntu@nodea:~$ kubectl get pods

NAME      READY     STATUS    RESTARTS   AGE
nginx     2/2       Running   0          18s
ubuntu@nodea:~$
```

We will now locate the node our pod has been deployed to (remember, we have not added *nodeb* to the cluster, yet).

```
ubuntu@nodea:~$ kubectl describe pod nginx | grep -E ^Node:
```

```
Node:         nodea/172.31.28.198
ubuntu@nodea:~$
```

or

```
ubuntu@nodea:~$ curl -s http://localhost:8080/api/v1/pods | jq .items[].spec.nodeName -r

nodea
ubuntu@nodea:~$
```

If you review our pods template, you will notice an entry *spec.nodeName*. This field is where we hardcoded the node where our pod was placed.

7. Run a pod without *nodeName*

Open copy `testpod.yaml`, change the pod name to `nginx-a` and remove the option *nodeName*; leave everything else the same as before.

```
ubuntu@nodea:~$ cp testpod.yaml testpod-a.yaml

ubuntu@nodea:~$ vim testpod-a.yaml

ubuntu@nodea:~$ cat testpod-a.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx-a
spec:
#  nodeName: nodea
  containers:

...

ubuntu@nodea:~$
```

Launch the pod again and monitor its status.

```
ubuntu@nodea:~$ kubectl create -f testpod-a.yaml

pod/nginx-a created
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl get pods

NAME      READY     STATUS    RESTARTS   AGE
nginx     2/2       Running   0          3m
nginx-a   0/2       Pending   0          9s
ubuntu@nodea:~$
```

Notice, the status is "Pending". Why?

The pod has no target host, which means that it must be scheduled to a node but we have no scheduler!

The pod will remain in the pending state until you either recreate the pod with a *nodeName* configured, or start the scheduler.

8. Add nodeb to the cluster

Before we start the scheduler let's add nodeb to the cluster. To do this, we need to install the `kubelet` services on *nodeb*. Since we have already compiled it on *nodea* we will simply copy it (and everything else) over.

On nodeb run the following commands:

```
laptop$ ssh -i k8s-adv-student.pem ubuntu@<external-ip>

...

ubuntu@nodeb:~$ mkdir kube-bin

ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ nc -l -p 7000 | tar xv -C kube-bin/
```

```
...
```

This puts netcat in listening mode, with tar decompressing into the `~/kube-bin` directory.

On nodea run (it will take a couple minutes to complete):

First add nodeb to the etc/hosts file:

```
ubuntu@nodea:~$ sudo vim /etc/hosts

ubuntu@nodea:~$ cat /etc/hosts

127.0.0.1 localhost
172.31.30.148 nodeb
172.31.28.198 nodea

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
ubuntu@nodea:~$
```

Now tar the Kubernetes binaries into the netcat tunnel (it will take a couple minutes to complete):

```
ubuntu@nodea:~$ tar -C ~/k8s/_output/local/bin/linux/amd64/ -cf - . | nc nodeb 7000

ubuntu@nodea:~$
```

This command uses netcat to funnel our tared data over to nodeb. This will copy our binaries from *nodea* to *nodeb* (you will see output on *nodeb*). Depending on what you compiled, your output may differ slightly on nodeb.

```
ubuntu@nodeb:~$ nc -l -p 7000 | tar xv -C kube-bin/
```

```
./
./gendocs
./genman
./kube-apiserver
./genswaggertypedocs
./linkcheck
./conversion-gen
./teststale
./go-bindata
./defaulter-gen
./genyaml
./hyperkube
./kube-aggregator
./deepcopy-gen
./genfeddocs
./kubelet
./kube-proxy
./genkubedocs
./kubeadm
./kube-scheduler
./gke-certificates-controller
./kube-controller-manager
./mungedocs
./apiextensions-apiserver
./openapi-gen
./cloud-controller-manager
./ginkgo
./e2e.test
./kubemark
./kubectl
./e2e_node.test
./kubefed

ubuntu@nodeb:~$
```

Before running the kubelet on nodeb, create a kubeconfig file with information to connect to the nodea apiserver.

```
ubuntu@nodeb:~$ vim nodeb.conf
ubuntu@nodeb:~$ cat nodeb.conf

apiVersion: v1
clusters:
```

```
- cluster:
    server: http://nodea:8080
  name: local
contexts:
- context:
    cluster: local
    user: ""
  name: local
current-context: local
kind: Config
preferences: {}
users: []
ubuntu@nodeb:~$
```

Note that all our kubelet really needs to know is the URI of the API server.

```
ubuntu@nodeb:~$ vim nodeb.yaml
ubuntu@nodeb:~$ cat nodeb.yaml

apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
authentication:
  anonymous:
    enabled: true
cgroupDriver: cgroupfs
failSwapOn: true
ubuntu@nodeb:~$
```

On *nodeb*, you can start the `kubelet` process via:

```
ubuntu@nodeb:~$ sudo $HOME/kube-bin/kubelet \
--kubeconfig=nodeb.conf \
--config=nodeb.yaml \
--allow-privileged=true \
--runtime-cgroups=/systemd/machine.slice \
--kubelet-cgroups=/systemd/machine.slice \
--pod-infra-container-image=k8s.gcr.io/pause:3.1

Flag --allow-privileged has been deprecated, will be removed in a future version
Flag --kubelet-cgroups 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.
Flag --allow-privileged has been deprecated, will be removed in a future version
Flag --kubelet-cgroups 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 08:19:31.687354    4673 server.go:417] Version: v1.14.0
I0330 08:19:31.687718    4673 plugins.go:103] No cloud provider specified.
I0330 08:19:31.731090    4673 server.go:625] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
I0330 08:19:31.731380    4673 container_manager_linux.go:261] container manager verified user specified cgroup-
root exists: []
I0330 08:19:31.731401    4673 container_manager_linux.go:266] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName:/systemd/machine.slice SystemCgroupsName: KubeletCgroupsName:/systemd/machine.slice
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 08:19:31.731499    4673 container_manager_linux.go:286] Creating device plugin manager: true
I0330 08:19:31.731570    4673 state_mem.go:36] [cpumanager] initializing new in-memory state store
I0330 08:19:31.738047    4673 kubelet.go:304] Watching apiserver
I0330 08:19:31.740203    4673 client.go:75] Connecting to docker on unix:///var/run/docker.sock
I0330 08:19:31.740227    4673 client.go:104] Start docker client with request timeout=2m0s
W0330 08:19:31.741546    4673 docker_service.go:561] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0330 08:19:31.741569    4673 docker_service.go:238] Hairpin mode set to "hairpin-veth"
W0330 08:19:31.741662    4673 cni.go:213] Unable to update cni config: No networks found in /etc/cni/net.d
W0330 08:19:31.743077    4673 hostport_manager.go:68] The binary conntrack is not installed, this can cause
failures in network connection cleanup.
I0330 08:19:31.744096    4673 docker_service.go:253] Docker cri networking managed by kubernetes.io/no-op
I0330 08:19:31.761066    4673 docker_service.go:258] Docker Info: &
{ID:6TFI:7NRU:W6A5:B2NU:VOXK:OYQG:FJNE:4MSB:KG5T:P7OQ:SMJW:ATUN Containers:0 ContainersRunning:0
ContainersPaused:0 ContainersStopped:0 Images:0 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:37 SystemTime:2019-03-30T08:19:31.744762922Z 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:0xc0007838f0 NCPU:2 MemTotal:4142067712
GenericResources:[] DockerRootDir:/var/lib/docker HTTPProxy: HTTPSProxy: NoProxy: Name:nodeb Labels:[]
ExperimentalBuild:false ServerVersion:18.09.4 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:bb71b10fd8f58240ca47fbb579b9d1028eea7c84 Expected:bb71b10fd8f58240ca47fbb579b9d1028eea7c84}
RuncCommit:{ID:2b18fe1d885ee5083ef9f0838fee39b62d653e30 Expected:2b18fe1d885ee5083ef9f0838fee39b62d653e30}
InitCommit:{ID:fec3683 Expected:fec3683} SecurityOptions:[name=apparmor name=seccomp,profile=default]}
I0330 08:19:31.761152    4673 docker_service.go:271] Setting cgroupDriver to cgroupfs
I0330 08:19:31.778932    4673 remote_runtime.go:62] parsed scheme: ""
I0330 08:19:31.778956    4673 remote_runtime.go:62] scheme "" not registered, fallback to default scheme
I0330 08:19:31.778986    4673 remote_image.go:50] parsed scheme: ""
I0330 08:19:31.778995    4673 remote_image.go:50] scheme "" not registered, fallback to default scheme
I0330 08:19:31.779042    4673 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 08:19:31.779058    4673 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 08:19:31.779098    4673 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0001d3580, CONNECTING
I0330 08:19:31.779133    4673 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{/var/run/dockershim.sock 0  <nil>}]
I0330 08:19:31.779143    4673 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 08:19:31.779170    4673 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0001d8680, CONNECTING
I0330 08:19:31.779694    4673 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0001d3580, READY
I0330 08:19:31.781156    4673 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc0001d8680, READY
I0330 08:19:31.784413    4673 kuberuntime_manager.go:210] Container runtime docker initialized, version: 18.09.4,
apiVersion: 1.39.0
W0330 08:19:31.784610    4673 probe.go:268] Flexvolume plugin directory at /usr/libexec/kubernetes/kubelet-
plugins/volume/exec/ does not exist. Recreating.
I0330 08:19:31.786082    4673 server.go:1037] Started kubelet
E0330 08:19:31.786529    4673 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 08:19:31.787175    4673 fs_resource_analyzer.go:64] Starting FS ResourceAnalyzer
I0330 08:19:31.787264    4673 status_manager.go:152] Starting to sync pod status with apiserver
I0330 08:19:31.787356    4673 kubelet.go:1806] Starting kubelet main sync loop.
I0330 08:19:31.787436    4673 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 08:19:31.787581    4673 server.go:141] Starting to listen on 0.0.0.0:10250
I0330 08:19:31.788282    4673 server.go:343] Adding debug handlers to kubelet server.
I0330 08:19:31.789427    4673 volume_manager.go:248] Starting Kubelet Volume Manager
```

```
E0330 08:19:31.798544    4673 controller.go:194] failed to get node "nodeb" when trying to set owner ref to the
node lease: nodes "nodeb" not found
I0330 08:19:31.800275    4673 desired_state_of_world_populator.go:130] Desired state populator starts to run
I0330 08:19:31.824668    4673 clientconn.go:440] parsed scheme: "unix"
I0330 08:19:31.824822    4673 clientconn.go:440] scheme "unix" not registered, fallback to default scheme
I0330 08:19:31.824949    4673 asm_amd64.s:1337] ccResolverWrapper: sending new addresses to cc:
[{unix:///run/containerd/containerd.sock 0  <nil>}]
I0330 08:19:31.824967    4673 clientconn.go:796] ClientConn switching balancer to "pick_first"
I0330 08:19:31.825002    4673 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000235bf0, CONNECTING
I0330 08:19:31.825270    4673 balancer_conn_wrappers.go:131] pickfirstBalancer: HandleSubConnStateChange:
0xc000235bf0, READY
E0330 08:19:31.895713    4673 kubelet.go:2244] node "nodeb" not found
I0330 08:19:31.895740    4673 kubelet.go:1823] skipping pod synchronization - container runtime status check may
not have completed yet.
I0330 08:19:31.895768    4673 kubelet_node_status.go:283] Setting node annotation to enable volume controller
attach/detach
I0330 08:19:31.897665    4673 kubelet_node_status.go:72] Attempting to register node nodeb
I0330 08:19:31.901600    4673 kubelet_node_status.go:75] Successfully registered node nodeb
I0330 08:19:31.905029    4673 cpu_manager.go:155] [cpumanager] starting with none policy
I0330 08:19:31.905047    4673 cpu_manager.go:156] [cpumanager] reconciling every 10s
I0330 08:19:31.905065    4673 policy_none.go:42] [cpumanager] none policy: Start
W0330 08:19:31.917027    4673 manager.go:538] Failed to retrieve checkpoint for "kubelet_internal_checkpoint":
checkpoint is not found
W0330 08:19:31.920491    4673 container_manager_linux.go:818] CPUAccounting not enabled for pid: 4673
W0330 08:19:31.920509    4673 container_manager_linux.go:821] MemoryAccounting not enabled for pid: 4673
I0330 08:19:32.100782    4673 reconciler.go:154] Reconciler: start to sync state

...
```

To confirm the *nodeb* `kubelet` has connected to the kube-apiserver on *nodeb*, run the following commands on nodeb to configure the kubelet and get the cluster node list.

```
ubuntu@nodeb:~$ sudo cp ./kube-bin/kubectl /usr/bin/

ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ kubectl config set-cluster local --server=http://nodea:8080
```

```
Cluster "local" set.
ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ kubectl config set-context local --cluster=local

Context "local" created.
ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ kubectl config use-context local

Switched to context "local".
ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ kubectl get nodes

NAME    STATUS   ROLES    AGE    VERSION
nodea   Ready    <none>   35m    v1.14.0
nodeb   Ready    <none>   2m4s   v1.14.0
ubuntu@nodeb:~$
```

You can also use `curl` (with help from `jq`) directly against the API:

```
ubuntu@nodeb:~$ sudo apt-get -y install jq

...
ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ curl -s http://nodea:8080/api/v1/nodes | jq -r .items[].metadata.name

nodea
nodeb
ubuntu@nodeb:~$
```

or to see the full output:

```
ubuntu@nodeb:~$ curl -s http://nodea:8080/api/v1/nodes

{
  "kind": "NodeList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/nodes",
    "resourceVersion": "264"
  },
  "items": [
    {
      "metadata": {
        "name": "nodea",

...

ubuntu@nodeb:~$
```

9. Running a pod on *nodeb*

Back on *nodea*, copy and modify the `testpod.yaml` to include:

```
ubuntu@nodea:~$ cp testpod-a.yaml testpod-b.yaml

ubuntu@nodea:~$ vim testpod-b.yaml
ubuntu@nodea:~$ cat testpod-b.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx-b
spec:
  nodeName: nodeb
  containers:

...
```

Launch the pod, as you proceed check status with the following methods.

- via `kubectl`
- via `docker`
- via `curl`

If you see status "ContainerCreating", this typically indicates the node is pulling the container image. Recall that *nodeb* is a brand new Docker install and as of yet has no local images to work with. Each node must pull its own images.

```
ubuntu@nodea:~$ kubectl create -f testpod-b.yaml

pod/nginx-b created
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl get pods

NAME      READY     STATUS              RESTARTS   AGE
nginx     2/2       Running             0          18m
nginx-a   0/2       Pending             0          15m
nginx-b   0/2       ContainerCreating   0          7s
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl get pods

NAME      READY     STATUS    RESTARTS   AGE
nginx     2/2       Running   0          5m
nginx-a   0/2       Pending   0          56s
nginx-b   2/2       Running   0          25s
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl get pod nginx-b \
-o=custom-columns=Name:.metadata.name,hostIP:.status.hostIP
```

```
Name      hostIP
nginx-b   172.31.30.148
ubuntu@nodea:~$
```

On nodeb:

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

CONTAINER ID        IMAGE                          COMMAND                  CREATED              STATUS
PORTS               NAMES
885652b2eeb1        busybox                        "/bin/sh -c 'while t…"   About a minute ago   Up About a minute
k8s_log-truncator_nginx_default_bea2bed7-4cef-11e8-8645-000c29473113_0
d6b0bc401ad8        nginx                          "nginx -g 'daemon of…"   2 minutes ago        Up 2 minutes
k8s_nginx_nginx_default_bea2bed7-4cef-11e8-8645-000c29473113_0
b319ff91f9c4        k8s.gcr.io/pause-amd64:3.1     "/pause"                 3 minutes ago        Up 3 minutes
k8s_POD_nginx_default_bea2bed7-4cef-11e8-8645-000c29473113_0

ubuntu@nodeb:~$
```

```
ubuntu@nodeb:~$ curl -s http://nodea:8080/api/v1/pods

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
    "resourceVersion": "213"
  },
  "items": [
    {
      "metadata": {
        "name": "nginx",
        "namespace": "default",
        "selfLink": "/api/v1/namespaces/default/pods/nginx",
        "uid": "4c12afc0-fbd8-11e7-bf93-000c290928aa",
        "resourceVersion": "194",
        "creationTimestamp": "2018-01-17T22:46:46Z"
      },
...

ubuntu@nodeb:~$
```

10. Start the scheduler

To run the scheduler we can simply execute the binary with a switch pointing it to the api server. **In a new terminal** on nodea:

```
laptop$ ssh -i k8s-adv-student.pem ubuntu@<external-ip>

...

ubuntu@nodea:~$ $HOME/k8s/_output/bin/kube-scheduler --kubeconfig=nodea.conf

I0330 08:31:14.877734    9479 serving.go:319] Generated self-signed cert in-memory
W0330 08:31:15.494975    9479 authentication.go:249] No authentication-kubeconfig provided in order to lookup
client-ca-file in configmap/extension-apiserver-authentication in kube-system, so client certificate
authentication won't work.
W0330 08:31:15.495002    9479 authentication.go:252] No authentication-kubeconfig provided in order to lookup
requestheader-client-ca-file in configmap/extension-apiserver-authentication in kube-system, so request-header
client certificate authentication won't work.
W0330 08:31:15.495015    9479 authorization.go:146] No authorization-kubeconfig provided, so SubjectAccessReview
of authorization tokens won't work.
I0330 08:31:15.497506    9479 server.go:142] Version: v1.14.0
I0330 08:31:15.497574    9479 defaults.go:87] TaintNodesByCondition is enabled, PodToleratesNodeTaints predicate
is mandatory
W0330 08:31:15.499062    9479 authorization.go:47] Authorization is disabled
W0330 08:31:15.499078    9479 authentication.go:55] Authentication is disabled
I0330 08:31:15.499092    9479 deprecated_insecure_serving.go:49] Serving healthz insecurely on [::]:10251
I0330 08:31:15.499503    9479 secure_serving.go:116] Serving securely on [::]:10259
I0330 08:31:16.401540    9479 controller_utils.go:1027] Waiting for caches to sync for scheduler controller
I0330 08:31:16.501749    9479 controller_utils.go:1034] Caches are synced for scheduler controller
I0330 08:31:16.501828    9479 leaderelection.go:217] attempting to acquire leader lease  kube-system/kube-
scheduler...
I0330 08:31:16.506049    9479 leaderelection.go:227] successfully acquired lease kube-system/kube-scheduler
```

Note that while many API servers can run in parallel (etcd ensures state is always consistent) only one scheduler may run within the cluster to avoid scheduling conflicts. For this reason the scheduler causes an election using etcd to determine which of the possibly several schedulers running will become the leader. All other schedulers simply monitor the leader for failure. If the leader fails, the remaining schedulers elect an new leader.

11. Pod placement via the scheduler

Now that we have built and started the scheduler, check in with our "pending" pods:

```
ubuntu@nodea:~$ kubectl get pods

NAME      READY   STATUS    RESTARTS   AGE
nginx     2/2     Running   0          52m
nginx-a   0/2     Pending   0          45m
nginx-b   2/2     Running   0          17m
ubuntu@nodea:~$
```

What happened?

In Kubernetes terms, the nodes are tainted. A taint consists of a *key*, a *value*, and an *effect*. The effect must be *NoSchedule*, *PreferNoSchedule* or *NoExecute*. You can view the taints on your node with the `kubectl` command. Use the `kubectl describe` subcommand to see details for one of your nodes:

```
ubuntu@nodea:~$ kubectl describe node nodea | grep Taints

Taints:             node.kubernetes.io/not-ready:NoSchedule
ubuntu@nodea:~$
```

This means the `kube-scheduler` can not place pods on this node. To remove this taint we can use the `kubectl taint` subcommand.

NOTE The command below removes ("-") the taint from all (--all) nodes in the cluster. **Do not forget the trailing** `-` The `-` is what tells Kubernetes to remove the taint!

> We know what you're thinking and we agree, "taint" is an awful name for this feature and a trailing dash with no space is an equally wacky way to remove something.

```
ubuntu@nodea:~$ kubectl taint nodes --all node.kubernetes.io/not-ready-
```

```
node/nodea untainted
node/nodeb untainted

ubuntu@nodea:~$ kubectl describe node nodea | grep Taints

Taints:             <none>
ubuntu@nodea:~$
```

Check in with our "pending" pods once more:

```
ubuntu@nodea:~$ kubectl get pods

NAME      READY   STATUS    RESTARTS   AGE
nginx     2/2     Running   0          70m
nginx-a   2/2     Running   0          64m
nginx-b   2/2     Running   0          36m
ubuntu@nodea:~$
```

Where does the pod land?

```
ubuntu@nodea:~$ kubectl get pod nginx-a \
-o=custom-columns=Name:.metadata.name,hostIP:.status.hostIP

Name      hostIP
nginx-a   172.31.28.198
ubuntu@nodea:~$
```

12. Launch additional pods

Create another pod with a random name.

```
ubuntu@nodea:~$ sed -e '/nodeName/d' testpod-a.yaml \
-e "s/name: nginx/name: nginx-$RANDOM/g" | kubectl create -f -
```

```
pod/nginx-29538-a created
ubuntu@nodea:~$
```

Determine which Node is our new pod running on:

```
ubuntu@nodea:~$ kubectl get pods -o wide

NAME            READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE   READINESS GATES
nginx           2/2     Running   0          71m   172.17.0.2   nodea   <none>           <none>
nginx-12095-a   2/2     Running   0          10s   172.17.0.3   nodeb   <none>           <none>
nginx-a         2/2     Running   0          65m   172.17.0.3   nodea   <none>           <none>
nginx-b         2/2     Running   0          36m   172.17.0.2   nodeb   <none>           <none>
ubuntu@nodea:~$
```

Create several more pods and view which node a pod is placed on. The default scheduler will spread pods across the nodes.

```
ubuntu@nodea:~$ kubectl get pods -o wide

NAME            READY   STATUS    RESTARTS   AGE   IP           NODE    NOMINATED NODE   READINESS GATES
nginx           2/2     Running   0          72m   172.17.0.2   nodea   <none>           <none>
nginx-11271-a   2/2     Running   0          17s   172.17.0.4   nodea   <none>           <none>
nginx-11427-a   2/2     Running   0          16s   172.17.0.5   nodeb   <none>           <none>
nginx-12095-a   2/2     Running   0          58s   172.17.0.3   nodeb   <none>           <none>
nginx-12751-a   2/2     Running   0          19s   172.17.0.4   nodeb   <none>           <none>
nginx-a         2/2     Running   0          65m   172.17.0.3   nodea   <none>           <none>
nginx-b         2/2     Running   0          37m   172.17.0.2   nodeb   <none>           <none>
ubuntu@nodea:~$
```

Try to remove all the pods (**note: your terminal should hang at this command**):

```
ubuntu@nodea:~$ kubectl get pod -o go-template \
--template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}' | xargs kubectl delete pod

pod "nginx" deleted
pod "nginx-23152-a" deleted
pod "nginx-25624-a" deleted
pod "nginx-25979-a" deleted
pod "nginx-29538-a" deleted
pod "nginx-9358-a" deleted
pod "nginx-a" deleted
pod "nginx-b" deleted
```

What's going on here?

Some of the Pod clean up tasks are performed by the contoller manager, notice how the pods hang on deletion. By running the controller manager we can resolve the pod shutdown.

13. Starting the controller manager

The old approach to running non-APIServer Kubernetes services was to provide them with the IRI of the API Server on the command line. The kube config approach is the go forward way to centralize cluster configuration for all of the Kubernetes services on a given node. Command line switches can still be used for many features (and are still required in some cases) but for basic operation the kubeconfig should suffice.

Run the controller manager and use the kubeconfig created earlier to point the controller manager at the appropriate API server; as a reminder, this is what the kubeconfig looks like:

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

apiVersion: v1
clusters:
- cluster:
    server: http://nodea:8080
  name: local
contexts:
- context:
    cluster: local
    user: ""
  name: local
current-context: local
kind: Config
preferences: {}
users: []
ubuntu@nodea:~$
```

In a new terminal start the controller manager:

```
ubuntu@nodea:~$ $HOME/k8s/_output/bin/kube-controller-manager --kubeconfig=nodea.conf

I0330 19:57:07.651811   13668 serving.go:319] Generated self-signed cert in-memory
W0330 19:57:08.117032   13668 authentication.go:249] No authentication-kubeconfig provided in order to lookup
client-ca-file in configmap/extension-apiserver-authentication in kube-system, so client certificate
authentication won't work.
W0330 19:57:08.117061   13668 authentication.go:252] No authentication-kubeconfig provided in order to lookup
requestheader-client-ca-file in configmap/extension-apiserver-authentication in kube-system, so request-header
client certificate authentication won't work.
W0330 19:57:08.117075   13668 authorization.go:146] No authorization-kubeconfig provided, so SubjectAccessReview
of authorization tokens won't work.
I0330 19:57:08.117116   13668 controllermanager.go:155] Version: v1.14.0
I0330 19:57:08.117556   13668 secure_serving.go:116] Serving securely on [::]:10257
I0330 19:57:08.117982   13668 deprecated_insecure_serving.go:51] Serving insecurely on [::]:10252
I0330 19:57:08.118091   13668 leaderelection.go:217] attempting to acquire leader lease  kube-system/kube-
controller-manager...
I0330 19:57:08.122732   13668 leaderelection.go:227] successfully acquired lease kube-system/kube-controller-
manager
I0330 19:57:08.123112   13668 event.go:209] Event(v1.ObjectReference{Kind:"Endpoints", Namespace:"kube-system",
Name:"kube-controller-manager", UID:"ffc05c46-5325-11e9-92a6-02ef63d53bbe", APIVersion:"v1",
ResourceVersion:"785", FieldPath:""}): type: 'Normal' reason: 'LeaderElection' nodea_ffc005c8-5325-11e9-bf69-
02ef63d53bbe became leader
I0330 19:57:08.334487   13668 plugins.go:103] No cloud provider specified.
W0330 19:57:08.334531   13668 controllermanager.go:517] "serviceaccount-token" is disabled because there is no
private key
I0330 19:57:08.335193   13668 controllermanager.go:497] Started "disruption"
I0330 19:57:08.335411   13668 node_lifecycle_controller.go:292] Sending events to api server.
I0330 19:57:08.335488   13668 disruption.go:286] Starting disruption controller
I0330 19:57:08.335559   13668 node_lifecycle_controller.go:325] Controller is using taint based evictions.
I0330 19:57:08.335566   13668 controller_utils.go:1027] Waiting for caches to sync for disruption controller
I0330 19:57:08.335613   13668 taint_manager.go:175] Sending events to api server.
I0330 19:57:08.335996   13668 node_lifecycle_controller.go:390] Controller will reconcile labels.
I0330 19:57:08.336021   13668 node_lifecycle_controller.go:403] Controller will taint node by condition.
I0330 19:57:08.336083   13668 controllermanager.go:497] Started "nodelifecycle"
I0330 19:57:08.336485   13668 controllermanager.go:497] Started "persistentvolume-expander"
I0330 19:57:08.336745   13668 node_lifecycle_controller.go:427] Starting node controller
I0330 19:57:08.336778   13668 controller_utils.go:1027] Waiting for caches to sync for taint controller
I0330 19:57:08.336811   13668 expand_controller.go:153] Starting expand controller
I0330 19:57:08.336829   13668 controller_utils.go:1027] Waiting for caches to sync for expand controller
I0330 19:57:08.336882   13668 controllermanager.go:497] Started "pvc-protection"
I0330 19:57:08.336890   13668 pvc_protection_controller.go:99] Starting PVC protection controller
I0330 19:57:08.336920   13668 controller_utils.go:1027] Waiting for caches to sync for PVC protection controller
```

```
I0330 19:57:08.337482   13668 controllermanager.go:497] Started "horizontalpodautoscaling"
I0330 19:57:08.337657   13668 horizontal.go:156] Starting HPA controller
I0330 19:57:08.337678   13668 controller_utils.go:1027] Waiting for caches to sync for HPA controller
I0330 19:57:08.337884   13668 controllermanager.go:497] Started "job"
W0330 19:57:08.337902   13668 controllermanager.go:476] "bootstrapsigner" is disabled
I0330 19:57:08.337912   13668 job_controller.go:143] Starting job controller
I0330 19:57:08.337933   13668 controller_utils.go:1027] Waiting for caches to sync for job controller
I0330 19:57:08.361029   13668 controllermanager.go:497] Started "namespace"
I0330 19:57:08.361119   13668 namespace_controller.go:186] Starting namespace controller
I0330 19:57:08.361180   13668 controller_utils.go:1027] Waiting for caches to sync for namespace controller
I0330 19:57:08.361472   13668 controllermanager.go:497] Started "deployment"
W0330 19:57:08.361488   13668 controllermanager.go:476] "tokencleaner" is disabled
I0330 19:57:08.361809   13668 deployment_controller.go:152] Starting deployment controller
I0330 19:57:08.361875   13668 controller_utils.go:1027] Waiting for caches to sync for deployment controller
I0330 19:57:08.361827   13668 controllermanager.go:497] Started "clusterrole-aggregation"
I0330 19:57:08.361834   13668 clusterroleaggregation_controller.go:148] Starting ClusterRoleAggregator
I0330 19:57:08.362238   13668 controller_utils.go:1027] Waiting for caches to sync for ClusterRoleAggregator
controller
I0330 19:57:08.767918   13668 controllermanager.go:497] Started "garbagecollector"
W0330 19:57:08.767967   13668 controllermanager.go:489] Skipping "csrsigning"
I0330 19:57:08.768383   13668 controllermanager.go:497] Started "ttl"
W0330 19:57:08.768402   13668 controllermanager.go:489] Skipping "ttl-after-finished"
I0330 19:57:08.768883   13668 controllermanager.go:497] Started "statefulset"
I0330 19:57:08.769144   13668 controllermanager.go:497] Started "cronjob"
I0330 19:57:08.769572   13668 controllermanager.go:497] Started "csrapproving"
I0330 19:57:08.770288   13668 controllermanager.go:497] Started "persistentvolume-binder"
I0330 19:57:08.767921   13668 garbagecollector.go:130] Starting garbage collector controller
I0330 19:57:08.770473   13668 controller_utils.go:1027] Waiting for caches to sync for garbage collector
controller
I0330 19:57:08.770496   13668 ttl_controller.go:116] Starting TTL controller
I0330 19:57:08.770508   13668 controller_utils.go:1027] Waiting for caches to sync for TTL controller
I0330 19:57:08.770525   13668 stateful_set.go:151] Starting stateful set controller
I0330 19:57:08.770552   13668 controller_utils.go:1027] Waiting for caches to sync for stateful set controller
I0330 19:57:08.770569   13668 graph_builder.go:308] GraphBuilder running
I0330 19:57:08.770584   13668 cronjob_controller.go:94] Starting CronJob Manager
I0330 19:57:08.770747   13668 certificate_controller.go:113] Starting certificate controller
I0330 19:57:08.770766   13668 controller_utils.go:1027] Waiting for caches to sync for certificate controller
I0330 19:57:08.770787   13668 pv_controller_base.go:270] Starting persistent volume controller
I0330 19:57:08.770801   13668 controller_utils.go:1027] Waiting for caches to sync for persistent volume
controller
I0330 19:57:08.771101   13668 controllermanager.go:497] Started "pv-protection"
I0330 19:57:08.771228   13668 pv_protection_controller.go:81] Starting PV protection controller
I0330 19:57:08.771812   13668 controller_utils.go:1027] Waiting for caches to sync for PV protection controller
```

```
I0330 19:57:08.772111   13668 controllermanager.go:497] Started "podgc"
I0330 19:57:08.772254   13668 gc_controller.go:76] Starting GC controller
I0330 19:57:08.772272   13668 controller_utils.go:1027] Waiting for caches to sync for GC controller
I0330 19:57:08.772982   13668 controllermanager.go:497] Started "serviceaccount"
I0330 19:57:08.773006   13668 serviceaccounts_controller.go:115] Starting service account controller
I0330 19:57:08.773049   13668 controller_utils.go:1027] Waiting for caches to sync for service account controller
I0330 19:57:08.773249   13668 controllermanager.go:497] Started "csrcleaner"
W0330 19:57:08.773263   13668 controllermanager.go:489] Skipping "nodeipam"
I0330 19:57:08.773281   13668 cleaner.go:81] Starting CSR cleaner controller
I0330 19:57:08.775263   13668 controllermanager.go:497] Started "replicationcontroller"
I0330 19:57:08.776190   13668 controllermanager.go:497] Started "replicaset"
I0330 19:57:08.776217   13668 core.go:171] Will not configure cloud provider routes for allocate-node-cidrs:
false, configure-cloud-routes: true.
W0330 19:57:08.776229   13668 controllermanager.go:489] Skipping "route"
I0330 19:57:08.776848   13668 node_lifecycle_controller.go:77] Sending events to api server
E0330 19:57:08.776911   13668 core.go:161] failed to start cloud node lifecycle controller: no cloud provider
provided
W0330 19:57:08.776942   13668 controllermanager.go:489] Skipping "cloud-node-lifecycle"
I0330 19:57:08.778554   13668 controllermanager.go:497] Started "attachdetach"
I0330 19:57:08.780283   13668 replica_set.go:182] Starting replicaset controller
I0330 19:57:08.780311   13668 controller_utils.go:1027] Waiting for caches to sync for ReplicaSet controller
I0330 19:57:08.780429   13668 controllermanager.go:497] Started "endpoint"
I0330 19:57:08.780941   13668 controllermanager.go:497] Started "daemonset"
E0330 19:57:08.781348   13668 core.go:77] Failed to start service controller: WARNING: no cloud provider provided,
services of type LoadBalancer will fail
W0330 19:57:08.781367   13668 controllermanager.go:489] Skipping "service"
W0330 19:57:08.781409   13668 controllermanager.go:489] Skipping "root-ca-cert-publisher"
I0330 19:57:08.782137   13668 replica_set.go:182] Starting replicationcontroller controller
I0330 19:57:08.782189   13668 controller_utils.go:1027] Waiting for caches to sync for ReplicationController
controller
I0330 19:57:08.782240   13668 endpoints_controller.go:166] Starting endpoint controller
I0330 19:57:08.782278   13668 controller_utils.go:1027] Waiting for caches to sync for endpoint controller
I0330 19:57:08.782324   13668 daemon_controller.go:267] Starting daemon sets controller
I0330 19:57:08.782360   13668 controller_utils.go:1027] Waiting for caches to sync for daemon sets controller
I0330 19:57:08.786116   13668 attach_detach_controller.go:323] Starting attach detach controller
I0330 19:57:08.786131   13668 controller_utils.go:1027] Waiting for caches to sync for attach detach controller
I0330 19:57:08.983474   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
endpoints
I0330 19:57:08.983556   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
replicasets.apps
I0330 19:57:08.983598   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
cronjobs.batch
I0330 19:57:08.983629   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
```

```
daemonsets.extensions
I0330 19:57:08.983667   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
replicasets.extensions
I0330 19:57:08.983708   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
horizontalpodautoscalers.autoscaling
I0330 19:57:08.983744   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
poddisruptionbudgets.policy
I0330 19:57:08.983777   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
leases.coordination.k8s.io
I0330 19:57:08.983896   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
ingresses.extensions
I0330 19:57:08.983949   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
controllerrevisions.apps
I0330 19:57:08.983987   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
events.events.k8s.io
I0330 19:57:08.984021   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
ingresses.networking.k8s.io
I0330 19:57:08.984080   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
limitranges
I0330 19:57:08.984145   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
daemonsets.apps
I0330 19:57:08.984234   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
statefulsets.apps
I0330 19:57:08.984281   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
networkpolicies.networking.k8s.io
I0330 19:57:08.984424   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
rolebindings.rbac.authorization.k8s.io
I0330 19:57:08.984475   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
serviceaccounts
I0330 19:57:08.984522   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
podtemplates
I0330 19:57:08.984600   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
deployments.extensions
I0330 19:57:08.984638   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
deployments.apps
I0330 19:57:08.984671   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
jobs.batch
I0330 19:57:08.984728   13668 resource_quota_monitor.go:228] QuotaMonitor created object count evaluator for
roles.rbac.authorization.k8s.io
E0330 19:57:08.984860   13668 resource_quota_controller.go:171] initial monitor sync has error: couldn't start
monitor for resource "extensions/v1beta1, Resource=networkpolicies": unable to monitor quota for resource
"extensions/v1beta1, Resource=networkpolicies"
I0330 19:57:08.984879   13668 controllermanager.go:497] Started "resourcequota"
```

```
I0330 19:57:08.985212   13668 resource_quota_controller.go:276] Starting resource quota controller
I0330 19:57:08.985234   13668 controller_utils.go:1027] Waiting for caches to sync for resource quota controller
I0330 19:57:09.000180   13668 resource_quota_monitor.go:301] QuotaMonitor running
W0330 19:57:09.008060   13668 actual_state_of_world.go:503] Failed to update statusUpdateNeeded field in actual
state of world: Failed to set statusUpdateNeeded to needed true, because nodeName="nodea" does not exist
I0330 19:57:09.037886   13668 controller_utils.go:1034] Caches are synced for HPA controller
I0330 19:57:09.038094   13668 controller_utils.go:1034] Caches are synced for job controller
I0330 19:57:09.061441   13668 controller_utils.go:1034] Caches are synced for namespace controller
I0330 19:57:09.070670   13668 controller_utils.go:1034] Caches are synced for TTL controller
I0330 19:57:09.070887   13668 controller_utils.go:1034] Caches are synced for certificate controller
I0330 19:57:09.072045   13668 controller_utils.go:1034] Caches are synced for PV protection controller
I0330 19:57:09.072381   13668 controller_utils.go:1034] Caches are synced for GC controller
I0330 19:57:09.073692   13668 controller_utils.go:1034] Caches are synced for service account controller
I0330 19:57:09.082410   13668 controller_utils.go:1034] Caches are synced for daemon sets controller
I0330 19:57:09.082503   13668 controller_utils.go:1034] Caches are synced for ReplicationController controller
I0330 19:57:09.086296   13668 controller_utils.go:1034] Caches are synced for attach detach controller
I0330 19:57:09.136943   13668 controller_utils.go:1034] Caches are synced for expand controller
I0330 19:57:09.137202   13668 controller_utils.go:1034] Caches are synced for PVC protection controller
I0330 19:57:09.162369   13668 controller_utils.go:1034] Caches are synced for ClusterRoleAggregator controller
I0330 19:57:09.437029   13668 controller_utils.go:1034] Caches are synced for taint controller
I0330 19:57:09.437092   13668 node_lifecycle_controller.go:1159] Initializing eviction metric for zone:
W0330 19:57:09.437143   13668 node_lifecycle_controller.go:833] Missing timestamp for Node nodea. Assuming now as
a timestamp.
I0330 19:57:09.437176   13668 node_lifecycle_controller.go:1059] Controller detected that zone  is now in state
Normal.
I0330 19:57:09.437712   13668 taint_manager.go:198] Starting NoExecuteTaintManager
I0330 19:57:09.438138   13668 event.go:209] Event(v1.ObjectReference{Kind:"Node", Namespace:"", Name:"nodea",
UID:"f17942e8-5322-11e9-a6c5-02ef63d53bbe", APIVersion:"", ResourceVersion:"", FieldPath:""}): type: 'Normal'
reason: 'RegisteredNode' Node nodea event: Registered Node nodea in Controller
I0330 19:57:09.462156   13668 controller_utils.go:1034] Caches are synced for deployment controller
I0330 19:57:09.480489   13668 controller_utils.go:1034] Caches are synced for ReplicaSet controller
I0330 19:57:09.635860   13668 controller_utils.go:1034] Caches are synced for disruption controller
I0330 19:57:09.635884   13668 disruption.go:294] Sending events to api server.
I0330 19:57:09.670678   13668 controller_utils.go:1034] Caches are synced for stateful set controller
I0330 19:57:09.670964   13668 controller_utils.go:1034] Caches are synced for persistent volume controller
I0330 19:57:09.782420   13668 controller_utils.go:1034] Caches are synced for endpoint controller
I0330 19:57:09.785420   13668 controller_utils.go:1034] Caches are synced for resource quota controller
I0330 19:57:09.871514   13668 controller_utils.go:1034] Caches are synced for garbage collector controller
I0330 19:57:09.871537   13668 garbagecollector.go:139] Garbage collector: all resource monitors have synced.
Proceeding to collect garbage
I0330 19:57:10.263511   13668 controller_utils.go:1027] Waiting for caches to sync for garbage collector
controller
I0330 19:57:10.363817   13668 controller_utils.go:1034] Caches are synced for garbage collector controller
```

```
E0330 19:57:10.683383   13668 resource_quota_controller.go:437] failed to sync resource monitors: couldn't start
monitor for resource "extensions/v1beta1, Resource=networkpolicies": unable to monitor quota for resource
"extensions/v1beta1, Resource=networkpolicies"
```

This provides us with a lot of output.

A key takeaway:

> I0330 19:57:08.122732 13668 leaderelection.go:227] successfully acquired lease kube-system/kube-controller-manager

Like the scheduler, there can only be one active controller manager in a cluster. Anytime a new controller manager starts it forces an election. Given that this is the first controller manager it becomes the leader and will actively begin managing Deployments, ReplicaSets and Replication Controllers.

The Controller Manager manages many other resources however. Look over the log output and identify the various resource types reported.

You should be able to find at least:

```
 - ReplicationController
 - DaemonSet
 - Job
 - Deployment
 - ReplicaSet
 - HorizontalPodAutoscaler
 - StatefulSet
```

After running the controller manager, our pods should successfully terminate and give back control over our terminal:

```
pod "nginx" deleted
pod "nginx-23152-a" deleted
pod "nginx-25624-a" deleted
pod "nginx-25979-a" deleted
pod "nginx-29538-a" deleted
pod "nginx-9358-a" deleted
pod "nginx-a" deleted
pod "nginx-b" deleted
```

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

14. Running deployments without a controller manager

To get an even clearer picture of our cluster's function with and without a Controller Manager, let's build a test deployment and and see what happens when we create it without a Controller Manager. Perform the following steps on nodea:

Stop the controller manager with Ctrl+C (^C):

```
...
I0330 20:00:20.223408   15191 controller_utils.go:1034] Caches are synced for attach detach controller
I0330 20:00:20.292498   15191 controller_utils.go:1034] Caches are synced for service account controller
I0330 20:00:20.320529   15191 controller_utils.go:1034] Caches are synced for namespace controller
I0330 20:00:20.406418   15191 controller_utils.go:1034] Caches are synced for ReplicationController controller
I0330 20:00:20.406613   15191 controller_utils.go:1034] Caches are synced for disruption controller
I0330 20:00:20.406693   15191 disruption.go:294] Sending events to api server.
I0330 20:00:20.697002   15191 controller_utils.go:1034] Caches are synced for certificate controller
I0330 20:00:20.796654   15191 controller_utils.go:1034] Caches are synced for garbage collector controller
I0330 20:00:20.796677   15191 garbagecollector.go:139] Garbage collector: all resource monitors have synced.
Proceeding to collect garbage
I0330 20:00:20.809309   15191 controller_utils.go:1034] Caches are synced for resource quota controller
I0330 20:00:21.192971   15191 controller_utils.go:1027] Waiting for caches to sync for garbage collector
controller
I0330 20:00:21.293201   15191 controller_utils.go:1034] Caches are synced for garbage collector controller
E0330 20:00:21.607165   15191 resource_quota_controller.go:437] failed to sync resource monitors: couldn't start
monitor for resource "extensions/v1beta1, Resource=networkpolicies": unable to monitor quota for resource
"extensions/v1beta1, Resource=networkpolicies

^C
ubuntu@nodea:~$
```

Now build the test deployment:

```
ubuntu@nodea:~$ vim testdep.yaml
ubuntu@nodea:~$ cat testdep.yaml

apiVersion: apps/v1
kind: Deployment
```

```
metadata:
  name: nginx-deployment
  labels:
    name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl create -f testdep.yaml

deployment.apps/nginx-deployment created
ubuntu@nodea:~$
```

Note that the message 'deployment "nginx-deployment" created' is from the API Server and indicates nothing other than that the API server added your desired state to the etcd store (this will fail in only the most dire circumstances).

Try listing the pods in your cluster:

```
ubuntu@nodea:~$ kubectl get pods

No resources found.
ubuntu@nodea:~$
```

This is a bad sign. Why didn't the cluster create the 2 pods requested?

List the other resources that should be associated with your deployment

```
ubuntu@nodea:~$ kubectl get rs

No resources found.
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl get deployment

NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   0/2     0            0           18s
ubuntu@nodea:~$
```

So our deployment was added to the cluster target state but nothing else was. Let's look deeper:

```
ubuntu@nodea:~$ kubectl describe deploy nginx-deployment

Name:                   nginx-deployment
Namespace:              default
CreationTimestamp:      Sat, 30 Mar 2019 20:01:59 +0000
Labels:                 <none>
Annotations:            <none>
Selector:               app=nginx
Replicas:               2 desired | 0 updated | 0 total | 0 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:        nginx:1.7.9
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
OldReplicaSets:   <none>
```

```
NewReplicaSet:    <none>
Events:           <none>
ubuntu@nodea:~$
```

Our deployment has no ReplicaSets. As you have probably guessed this is because the Kubernetes component that acts on deployments and creates ReplicaSets is the Controller Manager and we stopped it.

Start Controller Manager once more:

```
ubuntu@nodea:~$ $HOME/k8s/_output/bin/kube-controller-manager --kubeconfig=nodea.conf

...

I0330 20:03:37.097757   15757 controller_utils.go:1034] Caches are synced for GC controller
I0330 20:03:37.099389   15757 controller_utils.go:1034] Caches are synced for PV protection controller
I0330 20:03:37.099570   15757 controller_utils.go:1034] Caches are synced for job controller
I0330 20:03:37.108021   15757 controller_utils.go:1034] Caches are synced for deployment controller
I0330 20:03:37.111620   15757 event.go:209] Event(v1.ObjectReference{Kind:"Deployment", Namespace:"default",
Name:"nginx-deployment", UID:"ad28fb15-5326-11e9-92a6-02ef63d53bbe", APIVersion:"apps/v1", ResourceVersion:"1225",
FieldPath:""}): type: 'Normal' reason: 'ScalingReplicaSet' Scaled up replica set nginx-deployment-6dd86d77d to 2
I0330 20:03:37.115271   15757 controller_utils.go:1034] Caches are synced for ClusterRoleAggregator controller
I0330 20:03:37.130847   15757 controller_utils.go:1034] Caches are synced for namespace controller
I0330 20:03:37.131296   15757 controller_utils.go:1034] Caches are synced for service account controller
I0330 20:03:37.131476   15757 controller_utils.go:1034] Caches are synced for ReplicaSet controller
I0330 20:03:37.134585   15757 controller_utils.go:1034] Caches are synced for certificate controller
I0330 20:03:37.134616   15757 controller_utils.go:1034] Caches are synced for disruption controller
I0330 20:03:37.134651   15757 disruption.go:294] Sending events to api server.
I0330 20:03:37.134938   15757 event.go:209] Event(v1.ObjectReference{Kind:"ReplicaSet", Namespace:"default",
Name:"nginx-deployment-6dd86d77d", UID:"e79b48d1-5326-11e9-92a6-02ef63d53bbe", APIVersion:"apps/v1",
ResourceVersion:"1299", FieldPath:""}): type: 'Normal' reason: 'SuccessfulCreate' Created pod: nginx-deployment-
6dd86d77d-kcsdc
I0330 20:03:37.137283   15757 event.go:209] Event(v1.ObjectReference{Kind:"ReplicaSet", Namespace:"default",
Name:"nginx-deployment-6dd86d77d", UID:"e79b48d1-5326-11e9-92a6-02ef63d53bbe", APIVersion:"apps/v1",
ResourceVersion:"1299", FieldPath:""}): type: 'Normal' reason: 'SuccessfulCreate' Created pod: nginx-deployment-
6dd86d77d-z6tdf
I0330 20:03:37.147939   15757 controller_utils.go:1034] Caches are synced for expand controller
I0330 20:03:37.148035   15757 controller_utils.go:1034] Caches are synced for endpoint controller
I0330 20:03:37.148098   15757 controller_utils.go:1034] Caches are synced for PVC protection controller
I0330 20:03:37.148746   15757 controller_utils.go:1034] Caches are synced for stateful set controller
I0330 20:03:37.150774   15757 controller_utils.go:1034] Caches are synced for ReplicationController controller
I0330 20:03:37.510259   15757 controller_utils.go:1034] Caches are synced for HPA controller
```

```
W0330 20:03:37.803298   15757 actual_state_of_world.go:503] Failed to update statusUpdateNeeded field in actual
state of world: Failed to set statusUpdateNeeded to needed true, because nodeName="nodea" does not exist
I0330 20:03:37.848329   15757 controller_utils.go:1034] Caches are synced for TTL controller
I0330 20:03:37.849448   15757 controller_utils.go:1034] Caches are synced for taint controller
I0330 20:03:37.849502   15757 node_lifecycle_controller.go:1159] Initializing eviction metric for zone:
W0330 20:03:37.849556   15757 node_lifecycle_controller.go:833] Missing timestamp for Node nodea. Assuming now as
a timestamp.
I0330 20:03:37.849598   15757 node_lifecycle_controller.go:1059] Controller detected that zone  is now in state
Normal.
I0330 20:03:37.849631   15757 event.go:209] Event(v1.ObjectReference{Kind:"Node", Namespace:"", Name:"nodea",
UID:"f17942e8-5322-11e9-a6c5-02ef63d53bbe", APIVersion:"", ResourceVersion:"", FieldPath:""}): type: 'Normal'
reason: 'RegisteredNode' Node nodea event: Registered Node nodea in Controller
I0330 20:03:37.849655   15757 taint_manager.go:198] Starting NoExecuteTaintManager
I0330 20:03:37.896796   15757 controller_utils.go:1034] Caches are synced for daemon sets controller
I0330 20:03:37.898246   15757 controller_utils.go:1034] Caches are synced for persistent volume controller
I0330 20:03:37.899307   15757 controller_utils.go:1034] Caches are synced for attach detach controller
I0330 20:03:37.919088   15757 controller_utils.go:1034] Caches are synced for garbage collector controller
I0330 20:03:37.919108   15757 garbagecollector.go:139] Garbage collector: all resource monitors have synced.
Proceeding to collect garbage
I0330 20:03:37.947852   15757 controller_utils.go:1034] Caches are synced for resource quota controller
I0330 20:03:38.316138   15757 controller_utils.go:1027] Waiting for caches to sync for garbage collector
controller
I0330 20:03:38.416371   15757 controller_utils.go:1034] Caches are synced for garbage collector controller
E0330 20:03:38.746731   15757 resource_quota_controller.go:437] failed to sync resource monitors: couldn't start
monitor for resource "extensions/v1beta1, Resource=networkpolicies": unable to monitor quota for resource
"extensions/v1beta1, Resource=networkpolicies"
```

Once the Controller Manager is up and running, toward the end of the log output you will see it discover your deployment.

> I0330 20:03:37.111620 15757 event.go:209] Event(v1.ObjectReference{Kind:"Deployment", Namespace:"default", Name:"nginx-deployment", UID:"ad28fb15-5326-11e9-92a6-02ef63d53bbe", APIVersion:"apps/v1", ResourceVersion:"1225", FieldPath:""}): type: 'Normal' reason: 'ScalingReplicaSet' Scaled up replica set nginx-deployment-6dd86d77d to 2

This is immediately followed by events reporting the actions take by the Controller Manager to bring the cluster in line with your wishes:

> I0330 20:03:37.134938 15757 event.go:209] Event(v1.ObjectReference{Kind:"ReplicaSet", Namespace:"default", Name:"nginx-deployment-6dd86d77d", UID:"e79b48d1-5326-11e9-92a6-02ef63d53bbe", APIVersion:"apps/v1", ResourceVersion:"1299", FieldPath:""}): type: 'Normal' reason: 'SuccessfulCreate' Created pod: nginx-deployment-6dd86d77d-kcsdc

> I0330 20:03:37.137283 15757 event.go:209] Event(v1.ObjectReference{Kind:"ReplicaSet", Namespace:"default", Name:"nginx-deployment-6dd86d77d", UID:"e79b48d1-5326-11e9-92a6-02ef63d53bbe", APIVersion:"apps/v1", ResourceVersion:"1299", FieldPath:""}): type: 'Normal' reason: 'SuccessfulCreate' Created pod: nginx-deployment-6dd86d77d-z6tdf

Now try displaying the active pods:

```
ubuntu@nodea:~$ kubectl get pods

NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-6dd86d77d-jxxx7   1/1     Running   0          101s
nginx-deployment-6dd86d77d-qz8nl   1/1     Running   0          101s
ubuntu@nodea:~$
```

As advertised, the Controller manager has created the two pods required. You can examine the system events to see the progression of work involved in launching your two pods:

```
ubuntu@nodea:~$ kubectl get events | grep $(kubectl get pod -o name |tail -1)

3m49s       Normal    Scheduled                 pod/nginx-deployment-6dd86d77d-qz8nl    Successfully assigned
default/nginx-deployment-6dd86d77d-qz8nl to nodea
6s          Warning   MissingClusterDNS         pod/nginx-deployment-6dd86d77d-qz8nl    pod: "nginx-deployment-
6dd86d77d-qz8nl_default(231337e4-5328-11e9-92a6-02ef63d53bbe)". kubelet does not have ClusterDNS IP configured and
cannot create Pod using "ClusterFirst" policy. Falling back to "Default" policy.
3m48s       Normal    Pulled                    pod/nginx-deployment-6dd86d77d-qz8nl    Container image
"nginx:1.7.9" already present on machine
3m48s       Normal    Created                   pod/nginx-deployment-6dd86d77d-qz8nl    Created container nginx
3m48s       Normal    Started                   pod/nginx-deployment-6dd86d77d-qz8nl    Started container nginx
ubuntu@nodea:~$
```

After creating the Deployment's ReplicaSet the Controller Manager creates two Pods from the template and submits them to the cluster. The Scheduler is exclusively responsible for scheduling Pods to Kubelets when the Pods are not pinned to a particular host.

As soon as the Pods are scheduled the Kubelets begin pulling images and launching Pods. When the Pods are up the ReplicaSet reports successful Pod creation and when all Pods are up the Deployment reports successful scaling (from 0 to 2 as the Controller Manager logs stated).

Note that all Pods are automatically registered in the Kubernetes Cluster DNS if one is configured. We see Warnings because we have not yet setup Cluster DNS. We will take care of that in a future lab.

Relist your running Pods, ReplicaSets, and Deployments:

```
ubuntu@nodea:~$ kubectl get deploy,rs,po

NAME                                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/nginx-deployment     2/2     2            2           6m29s

NAME                                                  DESIRED   CURRENT   READY   AGE
replicaset.extensions/nginx-deployment-6dd86d77d     2         2         2       6m29s

NAME                                     READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-6dd86d77d-jxxx7     1/1     Running   0          6m29s
pod/nginx-deployment-6dd86d77d-qz8nl     1/1     Running   0          6m29s
ubuntu@nodea:~$
```

Everything looks healthy! Delete your deployment and all pods.

[OPTIONAL] Customizing the scheduler

The Kubernetes scheduler makes use of predicates to identify nodes to which a pod may be scheduled. The scheduler then uses policies to rank the nodes that pass the predicate tests. By default, Kubernetes provides built-in predicates and priority policies documented in `scheduler_algorithm.md`. The predicates and priorities code are defined in `plugin/pkg/scheduler/algorithm/predicates/predicates.go` and `plugin/pkg/scheduler/algorithm/priorities`, respectively.

The policies that are applied when scheduling can be selected in one of two ways. The default policies used are selected by the functions `defaultPredicates()` and `defaultPriorities()` in `plugin/pkg/scheduler/algorithmprovider/defaults/defaults.go`. However, the choice of policies can be overridden by passing the command-line flag `--policy-config-file` to the scheduler, pointing to a JSON file specifying which scheduling policies to use. See `examples/scheduler-policy-config.json` for an example config file. Note that the config file format is versioned; the API is defined in `plugin/pkg/scheduler/api`. To add a new scheduling policy, you should modify `plugin/pkg/scheduler/algorithm/predicates/predicates.go` or add to the directory `plugin/pkg/scheduler/algorithm/priorities`, and either register the policy in `defaultPredicates()` or `defaultPriorities()`, or use a policy config file.

To experiment with scheduling configuration, create a custom policy file and restart the scheduler with it.

```
ubuntu@nodea:~$ vim custom.json
ubuntu@nodea:~$ cat custom.json
```

```
{
"kind" : "Policy",
"apiVersion" : "v1",
"predicates" : [
        {"name" : "PodFitsHostPorts", "order": 1},
        {"name" : "PodFitsResources", "order": 3},
        {"name" : "NoDiskConflict",  "order": 4},
        {"name" : "NoVolumeZoneConflict",  "order": 6},
        {"name" : "MatchNodeSelector",  "order": 5},
        {"name" : "HostName",  "order": 2}
        ],
"priorities" : [
        {"name" : "LeastRequestedPriority", "weight" : 1},
        {"name" : "BalancedResourceAllocation", "weight" : 1},
        {"name" : "ServiceSpreadingPriority", "weight" : 1},
        {"name" : "EqualPriority", "weight" : 1}
        ]
}
```

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

Now restart the scheduler with the new policy:

```
ubuntu@nodea:~$ $HOME/k8s/_output/bin/kube-scheduler --kubeconfig=nodea.conf \
--policy-config-file=custom.json --v=2

I0330 20:23:05.568546   17416 flags.go:33] FLAG: --address="0.0.0.0"
I0330 20:23:05.568605   17416 flags.go:33] FLAG: --algorithm-provider=""
I0330 20:23:05.568611   17416 flags.go:33] FLAG: --alsologtostderr="false"
I0330 20:23:05.568616   17416 flags.go:33] FLAG: --authentication-kubeconfig=""
I0330 20:23:05.568640   17416 flags.go:33] FLAG: --authentication-skip-lookup="false"
I0330 20:23:05.568647   17416 flags.go:33] FLAG: --authentication-token-webhook-cache-ttl="10s"
I0330 20:23:05.568653   17416 flags.go:33] FLAG: --authentication-tolerate-lookup-failure="true"
I0330 20:23:05.568658   17416 flags.go:33] FLAG: --authorization-always-allow-paths="[/healthz]"
```

```
I0330 20:23:05.568672   17416 flags.go:33] FLAG: --authorization-kubeconfig=""
I0330 20:23:05.568680   17416 flags.go:33] FLAG: --authorization-webhook-cache-authorized-ttl="10s"
I0330 20:23:05.568688   17416 flags.go:33] FLAG: --authorization-webhook-cache-unauthorized-ttl="10s"
I0330 20:23:05.568697   17416 flags.go:33] FLAG: --bind-address="0.0.0.0"
I0330 20:23:05.568706   17416 flags.go:33] FLAG: --cert-dir=""
I0330 20:23:05.568714   17416 flags.go:33] FLAG: --client-ca-file=""
I0330 20:23:05.568722   17416 flags.go:33] FLAG: --config=""
I0330 20:23:05.568730   17416 flags.go:33] FLAG: --contention-profiling="false"
I0330 20:23:05.568738   17416 flags.go:33] FLAG: --failure-domains="kubernetes.io/hostname,failure-
domain.beta.kubernetes.io/zone,failure-domain.beta.kubernetes.io/region"
I0330 20:23:05.568750   17416 flags.go:33] FLAG: --feature-gates=""
I0330 20:23:05.568760   17416 flags.go:33] FLAG: --hard-pod-affinity-symmetric-weight="1"
I0330 20:23:05.568765   17416 flags.go:33] FLAG: --help="false"
I0330 20:23:05.568769   17416 flags.go:33] FLAG: --http2-max-streams-per-connection="0"
I0330 20:23:05.568774   17416 flags.go:33] FLAG: --kube-api-burst="100"
I0330 20:23:05.568779   17416 flags.go:33] FLAG: --kube-api-content-type="application/vnd.kubernetes.protobuf"
I0330 20:23:05.568783   17416 flags.go:33] FLAG: --kube-api-qps="50"
I0330 20:23:05.568789   17416 flags.go:33] FLAG: --kubeconfig="nodea.conf"
I0330 20:23:05.568797   17416 flags.go:33] FLAG: --leader-elect="true"
I0330 20:23:05.568806   17416 flags.go:33] FLAG: --leader-elect-lease-duration="15s"
I0330 20:23:05.568814   17416 flags.go:33] FLAG: --leader-elect-renew-deadline="10s"
I0330 20:23:05.568822   17416 flags.go:33] FLAG: --leader-elect-resource-lock="endpoints"
I0330 20:23:05.568830   17416 flags.go:33] FLAG: --leader-elect-retry-period="2s"
I0330 20:23:05.568834   17416 flags.go:33] FLAG: --lock-object-name="kube-scheduler"
I0330 20:23:05.568842   17416 flags.go:33] FLAG: --lock-object-namespace="kube-system"
I0330 20:23:05.568878   17416 flags.go:33] FLAG: --log-backtrace-at=":0"
I0330 20:23:05.568890   17416 flags.go:33] FLAG: --log-dir=""
I0330 20:23:05.568898   17416 flags.go:33] FLAG: --log-file=""
I0330 20:23:05.568906   17416 flags.go:33] FLAG: --log-flush-frequency="5s"
I0330 20:23:05.568914   17416 flags.go:33] FLAG: --logtostderr="true"
I0330 20:23:05.568923   17416 flags.go:33] FLAG: --master=""
I0330 20:23:05.568927   17416 flags.go:33] FLAG: --policy-config-file="custom.json"
I0330 20:23:05.568935   17416 flags.go:33] FLAG: --policy-configmap=""
I0330 20:23:05.568943   17416 flags.go:33] FLAG: --policy-configmap-namespace="kube-system"
I0330 20:23:05.568950   17416 flags.go:33] FLAG: --port="10251"
I0330 20:23:05.568955   17416 flags.go:33] FLAG: --profiling="false"
I0330 20:23:05.568963   17416 flags.go:33] FLAG: --requestheader-allowed-names="[]"
I0330 20:23:05.568988   17416 flags.go:33] FLAG: --requestheader-client-ca-file=""
I0330 20:23:05.568996   17416 flags.go:33] FLAG: --requestheader-extra-headers-prefix="[x-remote-extra-]"
I0330 20:23:05.569008   17416 flags.go:33] FLAG: --requestheader-group-headers="[x-remote-group]"
I0330 20:23:05.569017   17416 flags.go:33] FLAG: --requestheader-username-headers="[x-remote-user]"
I0330 20:23:05.569027   17416 flags.go:33] FLAG: --scheduler-name="default-scheduler"
I0330 20:23:05.569039   17416 flags.go:33] FLAG: --secure-port="10259"
```

```
I0330 20:23:05.569045   17416 flags.go:33] FLAG: --skip-headers="false"
I0330 20:23:05.569053   17416 flags.go:33] FLAG: --stderrthreshold="2"
I0330 20:23:05.569075   17416 flags.go:33] FLAG: --tls-cert-file=""
I0330 20:23:05.569083   17416 flags.go:33] FLAG: --tls-cipher-suites="[]"
I0330 20:23:05.569088   17416 flags.go:33] FLAG: --tls-min-version=""
I0330 20:23:05.569096   17416 flags.go:33] FLAG: --tls-private-key-file=""
I0330 20:23:05.569103   17416 flags.go:33] FLAG: --tls-sni-cert-key="[]"
I0330 20:23:05.569113   17416 flags.go:33] FLAG: --use-legacy-policy-config="false"
I0330 20:23:05.569121   17416 flags.go:33] FLAG: --v="2"
I0330 20:23:05.569141   17416 flags.go:33] FLAG: --version="false"
I0330 20:23:05.569152   17416 flags.go:33] FLAG: --vmodule=""
I0330 20:23:05.569160   17416 flags.go:33] FLAG: --write-config-to=""
I0330 20:23:06.083603   17416 serving.go:319] Generated self-signed cert in-memory
W0330 20:23:06.726996   17416 authentication.go:249] No authentication-kubeconfig provided in order to lookup
client-ca-file in configmap/extension-apiserver-authentication in kube-system, so client certificate
authentication won't work.
W0330 20:23:06.727022   17416 authentication.go:252] No authentication-kubeconfig provided in order to lookup
requestheader-client-ca-file in configmap/extension-apiserver-authentication in kube-system, so request-header
client certificate authentication won't work.
W0330 20:23:06.727034   17416 authorization.go:146] No authorization-kubeconfig provided, so SubjectAccessReview
of authorization tokens won't work.
I0330 20:23:06.728618   17416 server.go:142] Version: v1.14.0
I0330 20:23:06.728663   17416 defaults.go:87] TaintNodesByCondition is enabled, PodToleratesNodeTaints predicate
is mandatory
I0330 20:23:06.728679   17416 server.go:161] Starting Kubernetes Scheduler version v1.14.0
I0330 20:23:06.729579   17416 factory.go:341] Creating scheduler from configuration: {{ } [{PodFitsHostPorts
<nil>} {PodFitsResources <nil>} {NoDiskConflict <nil>} {NoVolumeZoneConflict <nil>} {MatchNodeSelector <nil>}
{HostName <nil>}] [{LeastRequestedPriority 1 <nil>} {BalancedResourceAllocation 1 <nil>} {ServiceSpreadingPriority
1 <nil>} {EqualPriority 1 <nil>}] [] 0 false}
I0330 20:23:06.729631   17416 factory.go:358] Registering predicate: PodFitsHostPorts
I0330 20:23:06.729648   17416 plugins.go:236] Predicate type PodFitsHostPorts already registered, reusing.
I0330 20:23:06.729659   17416 factory.go:358] Registering predicate: PodFitsResources
I0330 20:23:06.729667   17416 plugins.go:236] Predicate type PodFitsResources already registered, reusing.
I0330 20:23:06.729676   17416 factory.go:358] Registering predicate: NoDiskConflict
I0330 20:23:06.729684   17416 plugins.go:236] Predicate type NoDiskConflict already registered, reusing.
I0330 20:23:06.729692   17416 factory.go:358] Registering predicate: NoVolumeZoneConflict
I0330 20:23:06.729701   17416 plugins.go:236] Predicate type NoVolumeZoneConflict already registered, reusing.
I0330 20:23:06.729709   17416 factory.go:358] Registering predicate: MatchNodeSelector
I0330 20:23:06.729726   17416 plugins.go:236] Predicate type MatchNodeSelector already registered, reusing.
I0330 20:23:06.729734   17416 factory.go:358] Registering predicate: HostName
I0330 20:23:06.729743   17416 plugins.go:236] Predicate type HostName already registered, reusing.
I0330 20:23:06.729757   17416 factory.go:373] Registering priority: LeastRequestedPriority
I0330 20:23:06.729767   17416 plugins.go:348] Priority type LeastRequestedPriority already registered, reusing.
```

```
I0330 20:23:06.729779   17416 factory.go:373] Registering priority: BalancedResourceAllocation
I0330 20:23:06.729788   17416 plugins.go:348] Priority type BalancedResourceAllocation already registered,
reusing.
I0330 20:23:06.729797   17416 factory.go:373] Registering priority: ServiceSpreadingPriority
I0330 20:23:06.729806   17416 plugins.go:348] Priority type ServiceSpreadingPriority already registered, reusing.
I0330 20:23:06.729816   17416 factory.go:373] Registering priority: EqualPriority
I0330 20:23:06.729824   17416 plugins.go:348] Priority type EqualPriority already registered, reusing.
I0330 20:23:06.729835   17416 factory.go:412] Creating scheduler with fit predicates 'map[HostName:{}
MatchNodeSelector:{} NoDiskConflict:{} NoVolumeZoneConflict:{} PodFitsHostPorts:{} PodFitsResources:{}]' and
priority functions 'map[BalancedResourceAllocation:{} EqualPriority:{} LeastRequestedPriority:{}
ServiceSpreadingPriority:{}]'
W0330 20:23:06.730501   17416 authorization.go:47] Authorization is disabled
W0330 20:23:06.730517   17416 authentication.go:55] Authentication is disabled
I0330 20:23:06.730530   17416 deprecated_insecure_serving.go:49] Serving healthz insecurely on [::]:10251
I0330 20:23:06.731220   17416 secure_serving.go:116] Serving securely on [::]:10259
I0330 20:23:07.633103   17416 controller_utils.go:1027] Waiting for caches to sync for scheduler controller
I0330 20:23:07.733332   17416 controller_utils.go:1034] Caches are synced for scheduler controller
I0330 20:23:07.733406   17416 leaderelection.go:217] attempting to acquire leader lease  kube-system/kube-
scheduler...
I0330 20:23:25.141662   17416 leaderelection.go:227] successfully acquired lease kube-system/kube-scheduler

...
```

Launch several pods and adjust the values above to see how your changes effect pod placement.

```
ubuntu@nodea:~$ sed -e '/nodeName/d' testpod-a.yaml \
-e "s/name: nginx/name: nginx-$RANDOM/g" | kubectl create -f -

pod/nginx-708-a created
ubuntu@nodea:~$
```

```
ubuntu@nodea:~$ kubectl get events -w --sort-by=LASTSEEN

25m         Normal    RegisteredNode           node/nodea                             Node nodea event:
Registered Node nodea in Controller
22m         Normal    RegisteredNode           node/nodea                             Node nodea event:
Registered Node nodea in Controller
0s          Warning   MissingClusterDNS        pod/nginx-18049-a                      pod: "nginx-18049-
a_default(fbdd1e4c-5329-11e9-92a6-02ef63d53bbe)". kubelet does not have ClusterDNS IP configured and cannot create
```

```
Pod using "ClusterFirst" policy. Falling back to "Default" policy.
0s          Warning   MissingClusterDNS        pod/nginx-2616-a                        pod: "nginx-2616-
a_default(f9a6deb0-5329-11e9-92a6-02ef63d53bbe)". kubelet does not have ClusterDNS IP configured and cannot create
Pod using "ClusterFirst" policy. Falling back to "Default" policy.
0s          Warning   MissingClusterDNS        node/nodea                              kubelet does not have
ClusterDNS IP configured and cannot create Pod using "ClusterFirst" policy. Falling back to "Default" policy.
0s          Warning   MissingClusterDNS        pod/nginx-708-a                         pod: "nginx-708-
a_default(f1d83762-5329-11e9-92a6-02ef63d53bbe)". kubelet does not have ClusterDNS IP configured and cannot create
Pod using "ClusterFirst" policy. Falling back to "Default" policy.
0s          Warning   MissingClusterDNS        pod/nginx-18049-a                       pod: "nginx-18049-
a_default(fbdd1e4c-5329-11e9-92a6-02ef63d53bbe)". kubelet does not have ClusterDNS IP configured and cannot create
Pod using "ClusterFirst" policy. Falling back to "Default" policy.
^[0s          Warning   MissingClusterDNS        pod/nginx-26007-a                       pod: "nginx-26007-
a_default(fa3ba3a2-5329-11e9-92a6-02ef63d53bbe)". kubelet does not have ClusterDNS IP configured and cannot create
Pod using "ClusterFirst" policy. Falling back to "Default" policy.

...
```

You can review the functions here:

- https://github.com/kubernetes/kubernetes/blob/master/pkg/scheduler/algorithm/predicates/predicates.go

and here:

- https://github.com/kubernetes/kubernetes/tree/master/pkg/scheduler/algorithm/priorities

Delete all your existing pods and related resources but leave your kubelets, Scheduler, API Server and etcd running.

```
ubuntu@nodea:~$ kubectl get pod -o go-template \
--template '{{range .items}}{{.metadata.name}}{{"\n"}}{{end}}' | xargs kubectl delete pod

pod "nginx-13119" deleted
pod "nginx-16863" deleted
pod "nginx-17870" deleted
pod "nginx-32247" deleted
pod "nginx-5157" deleted

ubuntu@nodea:~$
```

Congratulations you have successfully completed the lab!