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* (using this template you can then add as many nodes as you like to your cluster). The master node was configured in lab 1 and was called *nodea*.

To create *nodeb*, open a second virtual machine using the lab system VM image (you copied it). Login to the VM with the same credentials (user/user).

2. Network configuration

In order for our servers to act as part of a single cluster they need to be able to reach each other over a network. Use the instructions in the following sections to configure a server network for your Kubernetes VMs (pick the section that applies to your hypervisor).

VMWare Player/Workstation/Fusion

No additional setup required.

VMWare desktop solutions create a virtual network and attach all running VMs to this network. Both VMs should be able to ping each other by IP address by default.

Virtual Box

To configure Virtual Box so that the two VMs share a virtual network go to the command line and run `VBoxManage` as follows:

```
VBoxManage natnetwork add --netname k8s --network "10.2.0.0/24" --enable --dhcp on
```

Now that we have created a network we need to attach each virtual machine as follows.

For both nodes:

- open menu *Devices->Network->Network Settings...*
- then under *Adapter 1*
- under *Attached to* select *NAT Network*
- select name *k8s*

For Virtual Box, make sure the *nodeb* mac address differs from *nodea*. You may need to halt the VM to update it in order to change it.

3. Update IPs and hostnames

Make sure to reboot the VMs to acquire a new ip address associated with the new subnet if you changed the network configuration.

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

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

```
user@ubuntu:~$ hostname
nodeb
user@ubuntu:~$
```

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

nodeb

user@ubuntu:~$
```

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

```
user@ubuntu:~$ ip a show ens33

2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:68:cd:9d brd ff:ff:ff:ff:ff:ff
    inet 172.16.151.204/24 brd 172.16.151.255 scope global ens33
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe68:cd9d/64 scope link
       valid_lft forever preferred_lft forever

user@ubuntu:~$
```

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.

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

127.0.0.1       localhost
#127.0.1.1      ubuntu
172.16.151.204 nodeb
172.16.151.203 nodea

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

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:

```
user@ubuntu:~$ ping -c 1 yahoo.com

PING yahoo.com (98.139.180.149) 56(84) bytes of data.
64 bytes from ir1.fp.vip.bf1.yahoo.com (98.139.180.149): icmp_seq=1 ttl=128 time=128 ms

--- yahoo.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 128.131/128.131/128.131/0.000 ms

user@ubuntu:~$
```

```
user@ubuntu:~$ ping -c 1 nodea

PING nodea (172.16.151.203) 56(84) bytes of data.
64 bytes from nodea (172.16.151.203): icmp_seq=1 ttl=64 time=0.362 ms

--- nodea ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.362/0.362/0.362/0.000 ms

user@ubuntu:~$
```

```
user@nodea:~$ ping -c 1 nodeb

PING nodeb (172.16.151.204) 56(84) bytes of data.
64 bytes from nodeb (172.16.151.204): icmp_seq=1 ttl=64 time=0.314 ms

--- nodeb ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.314/0.314/0.314/0.000 ms

user@nodea:~$
```

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

4. 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.

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

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

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

user@nodeb:~$ sudo apt-get update

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

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

user@nodeb:~$ sudo reboot
```

- 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.

5. 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:

```
user@nodeb:~$ docker version

Client:
 Version:      17.06.2-ce
 API version:  1.30
 Go version:   go1.8.3
 Git commit:   874a737
 Built:        Thu Aug 17 22:51:12 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.06.2-ce
 API version:  1.30 (minimum version 1.12)
 Go version:   go1.8.3
 Git commit:   874a737
 Built:        Thu Aug 17 22:50:04 2017
 OS/Arch:      linux/amd64
 Experimental: false

user@nodeb:~$
```

6. 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:

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

user@nodea:~$
```

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

```
user@nodea:~$
```

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

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

Start a fresh etcd:

```
user@nodea:~$ etcd

2017-08-29 12:54:17.486782 I | etcdmain: etcd Version: 3.2.6
2017-08-29 12:54:17.486976 I | etcdmain: Git SHA: 9d43462
2017-08-29 12:54:17.486999 I | etcdmain: Go Version: go1.8.3
2017-08-29 12:54:17.487006 I | etcdmain: Go OS/Arch: linux/amd64
2017-08-29 12:54:17.487013 I | etcdmain: setting maximum number of CPUs to 2, total number of available CPUs is 2
2017-08-29 12:54:17.488792 W | etcdmain: no data-dir provided, using default data-dir ./default.etcd
2017-08-29 12:54:17.491295 I | embed: listening for peers on http://localhost:2380
2017-08-29 12:54:17.495238 I | embed: listening for client requests on localhost:2379
2017-08-29 12:54:17.510643 I | etcdserver: name = default
2017-08-29 12:54:17.510688 I | etcdserver: data dir = default.etcd
2017-08-29 12:54:17.510694 I | etcdserver: member dir = default.etcd/member
2017-08-29 12:54:17.510697 I | etcdserver: heartbeat = 100ms
2017-08-29 12:54:17.510699 I | etcdserver: election = 1000ms
2017-08-29 12:54:17.510702 I | etcdserver: snapshot count = 100000
2017-08-29 12:54:17.510712 I | etcdserver: advertise client URLs = http://localhost:2379
2017-08-29 12:54:17.510715 I | etcdserver: initial advertise peer URLs = http://localhost:2380
2017-08-29 12:54:17.510723 I | etcdserver: initial cluster = default=http://localhost:2380
2017-08-29 12:54:17.521646 I | etcdserver: starting member 8e9e05c52164694d in cluster cdf818194e3a8c32
2017-08-29 12:54:17.523546 I | raft: 8e9e05c52164694d became follower at term 0
2017-08-29 12:54:17.523612 I | raft: newRaft 8e9e05c52164694d [peers: [], term: 0, commit: 0, applied: 0,
lastindex: 0, lastterm: 0]
2017-08-29 12:54:17.523621 I | raft: 8e9e05c52164694d became follower at term 1
2017-08-29 12:54:17.538680 W | auth: simple token is not cryptographically signed
2017-08-29 12:54:17.544073 I | etcdserver: starting server... [version: 3.2.6, cluster version: to_be_decided]
2017-08-29 12:54:17.561485 I | etcdserver/membership: added member 8e9e05c52164694d [http://localhost:2380] to
cluster cdf818194e3a8c32
2017-08-29 12:54:17.932232 I | raft: 8e9e05c52164694d is starting a new election at term 1
```

```
2017-08-29 12:54:17.934126 I | raft: 8e9e05c52164694d became candidate at term 2
2017-08-29 12:54:17.935138 I | raft: 8e9e05c52164694d received MsgVoteResp from 8e9e05c52164694d at term 2
2017-08-29 12:54:17.935562 I | raft: 8e9e05c52164694d became leader at term 2
2017-08-29 12:54:17.935948 I | raft: raft.node: 8e9e05c52164694d elected leader 8e9e05c52164694d at term 2
2017-08-29 12:54:17.942458 I | etcdserver: setting up the initial cluster version to 3.2
2017-08-29 12:54:17.946398 N | etcdserver/membership: set the initial cluster version to 3.2
2017-08-29 12:54:17.946484 I | etcdserver/api: enabled capabilities for version 3.2
2017-08-29 12:54:17.946524 I | etcdserver: published {Name:default ClientURLs:[http://localhost:2379]} to cluster
cdf818194e3a8c32
2017-08-29 12:54:17.946564 E | etcdmain: forgot to set Type=notify in systemd service file?
2017-08-29 12:54:17.946570 I | embed: ready to serve client requests
2017-08-29 12:54:17.975646 N | embed: serving insecure client requests on 127.0.0.1:2379, this is strongly
discouraged!
```

Restart the kube-apiserver and ask it to listen on all interfaces (so that nodeb can reach it):

```
user@nodea:~$ sudo ~user/k8s/_output/bin/kube-apiserver \
--etcd-servers=http://localhost:2379 \
--service-cluster-ip-range=10.0.0.0/16 \
--insecure-bind-address=0.0.0.0

I0829 12:55:12.111564   46640 server.go:112] Version: v1.7.8+793658f2d7ca7
W0829 12:55:12.129024   46640 authentication.go:368] AnonymousAuth is not allowed with the AllowAll authorizer.
Resetting AnonymousAuth to false. You should use a different authorizer
W0829 12:55:13.077812   46640 genericapiserver.go:325] Skipping API autoscaling/v2alpha1 because it has no
resources.
W0829 12:55:13.084670   46640 genericapiserver.go:325] Skipping API batch/v2alpha1 because it has no resources.
[restful] 2017/08/29 12:55:13 log.go:33: [restful/swagger] listing is available at
https://172.16.151.203:6443/swaggerapi
[restful] 2017/08/29 12:55:13 log.go:33: [restful/swagger] https://172.16.151.203:6443/swaggerui/ is mapped to
folder /swagger-ui/
I0829 12:55:29.434721   46640 insecure_handler.go:118] Serving insecurely on 0.0.0.0:8080
I0829 12:55:29.435378   46640 serve.go:85] Serving securely on 0.0.0.0:6443
I0829 12:55:29.438180   46640 apiservice_controller.go:113] Starting APIServiceRegistrationController
I0829 12:55:29.438513   46640 cache.go:32] Waiting for caches to sync for APIServiceRegistrationController
controller
I0829 12:55:29.451847   46640 available_controller.go:201] Starting AvailableConditionController
I0829 12:55:29.452447   46640 cache.go:32] Waiting for caches to sync for AvailableConditionController controller
I0829 12:55:29.458193   46640 tprregistration_controller.go:144] Starting tpr-autoregister controller
I0829 12:55:29.458223   46640 controller_utils.go:994] Waiting for caches to sync for tpr-autoregister controller
I0829 12:55:29.487010   46640 crd_finalizer.go:248] Starting CRDFinalizer
```

```
I0829 12:55:29.499592   46640 autoregister_controller.go:120] Starting autoregister controller
I0829 12:55:29.499856   46640 cache.go:32] Waiting for caches to sync for autoregister controller
I0829 12:55:29.524511   46640 customresource_discovery_controller.go:152] Starting DiscoveryController
I0829 12:55:29.530392   46640 naming_controller.go:284] Starting NamingConditionController
I0829 12:55:29.611801   46640 cache.go:39] Caches are synced for autoregister controller
I0829 12:55:29.642530   46640 cache.go:39] Caches are synced for APIServiceRegistrationController controller
I0829 12:55:29.671649   46640 controller_utils.go:1001] Caches are synced for tpr-autoregister controller
I0829 12:55:29.675762   46640 cache.go:39] Caches are synced for AvailableConditionController controller
E0829 12:55:29.829256   46640 autoregister_controller.go:167] v1. failed with : Operation cannot be fulfilled on
apiservices.apiregistration.k8s.io "v1.": the object has been modified; please apply your changes to the latest
version and try again
E0829 12:55:29.863784   46640 available_controller.go:234] v1beta1.apps failed with: Operation cannot be fulfilled
on apiservices.apiregistration.k8s.io "v1beta1.apps": the object has been modified; please apply your changes to
the latest version and try again
E0829 12:55:29.951529   46640 autoregister_controller.go:167] v1.authentication.k8s.io failed with : Operation
cannot be fulfilled on apiservices.apiregistration.k8s.io "v1.authentication.k8s.io": the object has been
modified; please apply your changes to the latest version and try again
...
```

Before restarting the kubelet, we will create a configuration file using the kubeconfig syntax. You can learn more here https://kubernetes.io/docs/user-guide/kubeconfig-file/.

Be sure to use your Kubernetes master hostname or VM IP in place of the hostname in the example below:

```
user@nodea:~$ vim nodea.kubeconfig
user@nodea:~$ cat nodea.kubeconfig

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

```
user@nodea:~$
```

Restart the nodea `kubelet` with following flags to use the kubeconfig:

```
user@nodea:~$ sudo ~user/k8s/_output/bin/kubelet \
--kubeconfig=nodea.kubeconfig \
--require-kubeconfig \
--allow-privileged=true

I0829 13:01:39.614861   46672 feature_gate.go:144] feature gates: map[]
I0829 13:01:39.663116   46672 client.go:72] Connecting to docker on unix:///var/run/docker.sock
I0829 13:01:39.663178   46672 client.go:92] Start docker client with request timeout=2m0s
W0829 13:01:39.667876   46672 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 13:01:39.701001   46672 manager.go:143] cAdvisor running in container: "/user.slice"
W0829 13:01:39.832881   46672 manager.go:151] unable to connect to Rkt api service: rkt: cannot tcp Dial rkt api
service: dial tcp [::1]:15441: getsockopt: connection refused
I0829 13:01:39.846075   46672 fs.go:117] Filesystem partitions: map[/dev/sda1:{mountpoint:/var/lib/docker/aufs
major:8 minor:1 fsType:ext4 blockSize:0}]
I0829 13:01:39.852421   46672 manager.go:198] Machine: {NumCores:2 CpuFrequency:2711681 MemoryCapacity:4124880896
MachineID:6e883acc04fc7db3713776be57a3dac9 SystemUUID:F2564D56-3460-443D-57E3-836F703215A2 BootID:e21a0ef9-85e0-
4ef1-b0b9-f85c262ea596 Filesystems:[{Device:/dev/sda1 DeviceMajor:8 DeviceMinor:1 Capacity:18889830400 Type:vfs
Inodes:1179648 HasInodes:true}] DiskMap:map[8:0:{Name:sda Major:8 Minor:0 Size:21474836480 Scheduler:deadline}]
NetworkDevices:[{Name:ens33 MacAddress:00:0c:29:32:15:a2 Speed:1000 Mtu:1500}] Topology:[{Id:0 Memory:4124880896
Cores:[{Id:0 Threads:[0] Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144
Type:Unified Level:2}]}] Caches:[{Size:8388608 Type:Unified Level:3}]} {Id:2 Memory:0 Cores:[{Id:0 Threads:[1]
Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144 Type:Unified Level:2}]}]
Caches:[{Size:8388608 Type:Unified Level:3}]}] CloudProvider:Unknown InstanceType:Unknown InstanceID:None}
I0829 13:01:39.853590   46672 manager.go:204] Version: {KernelVersion:4.4.0-93-generic ContainerOsVersion:Ubuntu
16.04.1 LTS DockerVersion:17.06.1-ce DockerAPIVersion:1.30 CadvisorVersion: CadvisorRevision:}
I0829 13:01:39.856422   46672 server.go:536] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
W0829 13:01:39.879173   46672 container_manager_linux.go:216] Running with swap on is not supported, please
disable swap! This will be a fatal error by default starting in K8s v1.6! In the meantime, you can opt-in to
making this a fatal error by enabling --experimental-fail-swap-on.
I0829 13:01:39.879245   46672 container_manager_linux.go:246] container manager verified user specified cgroup-
root exists: /
I0829 13:01:39.879282   46672 container_manager_linux.go:251] Creating Container Manager object based on Node
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs ProtectKernelDefaults:false NodeAllocatableConfig:{KubeReservedCgroupName:
SystemReservedCgroupName: EnforceNodeAllocatable:map[pods:{}] KubeReserved:map[] SystemReserved:map[]
HardEvictionThresholds:[{Signal:memory.available Operator:LessThan Value:{Quantity:100Mi Percentage:0}
GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.available Operator:LessThan Value:{Quantity:<nil> Percentage:0.1}
```

```
GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.inodesFree Operator:LessThan Value:{Quantity:<nil>
Percentage:0.05} GracePeriod:0s MinReclaim:<nil>}]} ExperimentalQOSReserved:map[]}
I0829 13:01:39.879732   46672 kubelet.go:273] Watching apiserver
W0829 13:01:39.927370   46672 kubelet_network.go:70] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0829 13:01:39.927400   46672 kubelet.go:508] Hairpin mode set to "hairpin-veth"
W0829 13:01:39.966888   46672 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 13:01:39.985742   46672 docker_service.go:208] Docker cri networking managed by kubernetes.io/no-op
I0829 13:01:40.030286   46672 docker_service.go:225] Setting cgroupDriver to cgroupfs
I0829 13:01:40.175617   46672 remote_runtime.go:42] Connecting to runtime service unix:///var/run/dockershim.sock
I0829 13:01:40.183658   46672 kuberuntime_manager.go:163] Container runtime docker initialized, version: 17.06.1-
ce, apiVersion: 1.30.0
I0829 13:01:40.200794   46672 server.go:943] Started kubelet v1.7.8+793658f2d7ca7
I0829 13:01:40.218516   46672 server.go:132] Starting to listen on 0.0.0.0:10250
E0829 13:01:40.221069   46672 kubelet.go:1229] Image garbage collection failed once. Stats initialization may not
have completed yet: unable to find data for container /
I0829 13:01:40.224247   46672 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 13:01:40.278897   46672 server.go:310] Adding debug handlers to kubelet server.
E0829 13:01:40.375989   46672 kubelet.go:1729] Failed to check if disk space is available for the runtime: failed
to get fs info for "runtime": unable to find data for container /
E0829 13:01:40.376015   46672 kubelet.go:1737] Failed to check if disk space is available on the root partition:
failed to get fs info for "root": unable to find data for container /
I0829 13:01:40.386001   46672 fs_resource_analyzer.go:66] Starting FS ResourceAnalyzer
I0829 13:01:40.386034   46672 status_manager.go:140] Starting to sync pod status with apiserver
I0829 13:01:40.386044   46672 kubelet.go:1809] Starting kubelet main sync loop.
I0829 13:01:40.386063   46672 kubelet.go:1820] skipping pod synchronization - [container runtime is down PLEG is
not healthy: pleg was last seen active 2562047h47m16.854775807s ago; threshold is 3m0s]
W0829 13:01:40.392655   46672 container_manager_linux.go:747] CPUAccounting not enabled for pid: 46672
W0829 13:01:40.392669   46672 container_manager_linux.go:750] MemoryAccounting not enabled for pid: 46672
E0829 13:01:40.394720   46672 container_manager_linux.go:543] [ContainerManager]: Fail to get rootfs information
unable to find data for container /
I0829 13:01:40.394822   46672 volume_manager.go:245] Starting Kubelet Volume Manager
E0829 13:01:40.400816   46672 docker_sandbox.go:239] Failed to stop sandbox
"17fdc80a4c01b2719157c5cee982c410b37efc21b394085c093c766e7a8bbb71": Error response from daemon: {"message":"No
such container: 17fdc80a4c01b2719157c5cee982c410b37efc21b394085c093c766e7a8bbb71"}
I0829 13:01:40.497175   46672 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
E0829 13:01:40.503661   46672 kubelet.go:1729] Failed to check if disk space is available for the runtime: failed
to get fs info for "runtime": unable to find data for container /
E0829 13:01:40.503704   46672 kubelet.go:1737] Failed to check if disk space is available on the root partition:
failed to get fs info for "root": unable to find data for container /
I0829 13:01:40.503731   46672 kubelet_node_status.go:82] Attempting to register node nodea
```

```
I0829 13:01:40.597668   46672 kubelet_node_status.go:85] Successfully registered node nodea
I0829 13:01:40.603399   46672 factory.go:351] Registering Docker factory
W0829 13:01:40.603425   46672 manager.go:247] Registration of the rkt container factory failed: unable to
communicate with Rkt api service: rkt: cannot tcp Dial rkt api service: dial tcp [::1]:15441: getsockopt:
connection refused
I0829 13:01:40.603439   46672 factory.go:54] Registering systemd factory
I0829 13:01:40.603586   46672 factory.go:86] Registering Raw factory
I0829 13:01:40.603728   46672 manager.go:1121] Started watching for new ooms in manager
E0829 13:01:40.607611   46672 kubelet.go:1729] Failed to check if disk space is available for the runtime: failed
to get fs info for "runtime": unable to find data for container /
E0829 13:01:40.607640   46672 kubelet.go:1737] Failed to check if disk space is available on the root partition:
failed to get fs info for "root": unable to find data for container /
I0829 13:01:40.624784   46672 oomparser.go:185] oomparser using systemd
I0829 13:01:40.629945   46672 manager.go:288] Starting recovery of all containers
I0829 13:01:41.404427   46672 manager.go:293] Recovery completed
W0829 13:01:45.386597   46672 pod_container_deletor.go:77] Container
"17fdc80a4c01b2719157c5cee982c410b37efc21b394085c093c766e7a8bbb71" not found in pod's containers
...
```

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:

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

Cluster "local" set.

user@nodea:~$
```

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

Context "local" created.

user@nodea:~$
```

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

```
Switched to context "local".

user@nodea:~$
```

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

NAME      STATUS    AGE       VERSION
nodea     Ready     2m        v1.7.8+793658f2d7ca7

user@nodea:~$
```

Now let recreate our simple Pod on *nodea* (from lab 1).

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

apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  nodeName: nodea
  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: {}
```

```
user@nodea:~$
```

Deploy your pod via create subcommand.

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

pod "nginx" created

user@nodea:~$
```

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

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

NAME      READY     STATUS    RESTARTS   AGE
nginx     2/2       Running   0          8s

user@nodea:~$
```

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

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

Node:           nodea/172.16.151.203

user@nodea:~$
```

or

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

nodea

user@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.

Lets remove the pod.

```
user@nodea:~$ kubectl delete pod nginx

pod "nginx" deleted

user@nodea:~$
```

7. Run a pod without *nodeName*

Open `testpod.yaml` and remove the option *nodeName*.

```
user@nodea:~$ vim testpod.yaml
user@nodea:~$ cat testpod.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
#  nodeName: nodea
  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: {}

user@nodea:~$
```

Launch the pod again and monitor its status.

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

pod "nginx" created

user@nodea:~$
```

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

NAME      READY     STATUS    RESTARTS   AGE
nginx     0/2       Pending   0          4s

user@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. For now, delete the "Pending" Pod.

```
user@nodea:~$ kubectl delete pod nginx

pod "nginx" deleted

user@nodea:~$
```

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:

```
user@nodeb:~$ mkdir kube-bin

user@nodeb:~$
```

```
user@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):

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

user@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.

```
user@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

 user@nodeb:~$
```

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

```
user@nodeb:~$ vim nodeb.kubeconfig
user@nodeb:~$ cat nodeb.kubeconfig

apiVersion: v1
clusters:
- cluster:
    server: http://nodea:8080
  name: local
contexts:
- context:
    cluster: local
    user: ""
  name: local
current-context: local
kind: Config
```

```
preferences: {}
users: []

user@nodeb:~$
```

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

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

```
user@nodeb:~$ sudo ~user/kube-bin/kubelet \
--kubeconfig=nodeb.kubeconfig \
--require-kubeconfig \
--allow-privileged=true

I0829 13:18:56.864789    2199 feature_gate.go:144] feature gates: map[]
I0829 13:18:57.051599    2199 client.go:72] Connecting to docker on unix:///var/run/docker.sock
I0829 13:18:57.052052    2199 client.go:92] Start docker client with request timeout=2m0s
W0829 13:18:57.054805    2199 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 13:18:57.067813    2199 manager.go:143] cAdvisor running in container: "/user.slice"
W0829 13:18:57.103535    2199 manager.go:151] unable to connect to Rkt api service: rkt: cannot tcp Dial rkt api
service: dial tcp [::1]:15441: getsockopt: connection refused
I0829 13:18:57.130348    2199 fs.go:117] Filesystem partitions: map[/dev/sda1:{mountpoint:/var/lib/docker/aufs
major:8 minor:1 fsType:ext4 blockSize:0}]
I0829 13:18:57.135599    2199 manager.go:198] Machine: {NumCores:2 CpuFrequency:2711681 MemoryCapacity:4124880896
MachineID:6e883acc04fc7db3713776be57a3dac9 SystemUUID:D95C4D56-F205-449D-CF79-2DD35C68CD9D BootID:25488d59-1f64-
4ed4-98ff-27259f42938b Filesystems:[{Device:/dev/sda1 DeviceMajor:8 DeviceMinor:1 Capacity:18889830400 Type:vfs
Inodes:1179648 HasInodes:true}] DiskMap:map[8:0:{Name:sda Major:8 Minor:0 Size:21474836480 Scheduler:deadline}]
NetworkDevices:[{Name:ens33 MacAddress:00:0c:29:68:cd:9d Speed:1000 Mtu:1500}] Topology:[{Id:0 Memory:4124880896
Cores:[{Id:0 Threads:[0] Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144
Type:Unified Level:2}]}] Caches:[{Size:8388608 Type:Unified Level:3}]} {Id:2 Memory:0 Cores:[{Id:0 Threads:[1]
Caches:[{Size:32768 Type:Data Level:1} {Size:32768 Type:Instruction Level:1} {Size:262144 Type:Unified Level:2}]}]
Caches:[{Size:8388608 Type:Unified Level:3}]}] CloudProvider:Unknown InstanceType:Unknown InstanceID:None}
I0829 13:18:57.136499    2199 manager.go:204] Version: {KernelVersion:4.4.0-93-generic ContainerOsVersion:Ubuntu
16.04.1 LTS DockerVersion:17.06.1-ce DockerAPIVersion:1.30 CadvisorVersion: CadvisorRevision:}
I0829 13:18:57.137067    2199 server.go:536] --cgroups-per-qos enabled, but --cgroup-root was not specified.
defaulting to /
W0829 13:18:57.140529    2199 container_manager_linux.go:216] Running with swap on is not supported, please
disable swap! This will be a fatal error by default starting in K8s v1.6! In the meantime, you can opt-in to
making this a fatal error by enabling --experimental-fail-swap-on.
I0829 13:18:57.140848    2199 container_manager_linux.go:246] container manager verified user specified cgroup-
root exists: /
I0829 13:18:57.141051    2199 container_manager_linux.go:251] Creating Container Manager object based on Node
```

```
Config: {RuntimeCgroupsName: SystemCgroupsName: KubeletCgroupsName: ContainerRuntime:docker CgroupsPerQOS:true
CgroupRoot:/ CgroupDriver:cgroupfs ProtectKernelDefaults:false NodeAllocatableConfig:{KubeReservedCgroupName:
SystemReservedCgroupName: EnforceNodeAllocatable:map[pods:{}] KubeReserved:map[] SystemReserved:map[]
HardEvictionThresholds:[{Signal:memory.available Operator:LessThan Value:{Quantity:100Mi Percentage:0}
GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.available Operator:LessThan Value:{Quantity:<nil> Percentage:0.1}
GracePeriod:0s MinReclaim:<nil>} {Signal:nodefs.inodesFree Operator:LessThan Value:{Quantity:<nil>
Percentage:0.05} GracePeriod:0s MinReclaim:<nil>}]} ExperimentalQOSReserved:map[]}
I0829 13:18:57.141628    2199 kubelet.go:273] Watching apiserver
W0829 13:18:57.146422    2199 kubelet_network.go:70] Hairpin mode set to "promiscuous-bridge" but kubenet is not
enabled, falling back to "hairpin-veth"
I0829 13:18:57.146452    2199 kubelet.go:508] Hairpin mode set to "hairpin-veth"
W0829 13:18:57.149432    2199 cni.go:189] Unable to update cni config: No networks found in /etc/cni/net.d
I0829 13:18:57.157952    2199 docker_service.go:208] Docker cri networking managed by kubernetes.io/no-op
I0829 13:18:57.167330    2199 docker_service.go:225] Setting cgroupDriver to cgroupfs
I0829 13:18:57.179614    2199 remote_runtime.go:42] Connecting to runtime service unix:///var/run/dockershim.sock
I0829 13:18:57.181236    2199 kuberuntime_manager.go:163] Container runtime docker initialized, version: 17.06.1-
ce, apiVersion: 1.30.0
I0829 13:18:57.182841    2199 server.go:943] Started kubelet v1.7.8+793658f2d7ca7
E0829 13:18:57.183023    2199 kubelet.go:1229] Image garbage collection failed once. Stats initialization may not
have completed yet: unable to find data for container /
I0829 13:18:57.183254    2199 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 13:18:57.183488    2199 server.go:132] Starting to listen on 0.0.0.0:10250
I0829 13:18:57.185993    2199 server.go:310] Adding debug handlers to kubelet server.
E0829 13:18:57.193605    2199 kubelet.go:1729] Failed to check if disk space is available for the runtime: failed
to get fs info for "runtime": unable to find data for container /
E0829 13:18:57.193649    2199 kubelet.go:1737] Failed to check if disk space is available on the root partition:
failed to get fs info for "root": unable to find data for container /
I0829 13:18:57.208870    2199 fs_resource_analyzer.go:66] Starting FS ResourceAnalyzer
I0829 13:18:57.208953    2199 status_manager.go:140] Starting to sync pod status with apiserver
I0829 13:18:57.208969    2199 kubelet.go:1809] Starting kubelet main sync loop.
I0829 13:18:57.208995    2199 kubelet.go:1820] skipping pod synchronization - [container runtime is down PLEG is
not healthy: pleg was last seen active 2562047h47m16.854775807s ago; threshold is 3m0s]
E0829 13:18:57.209189    2199 container_manager_linux.go:543] [ContainerManager]: Fail to get rootfs information
unable to find data for container /
I0829 13:18:57.209213    2199 volume_manager.go:245] Starting Kubelet Volume Manager
W0829 13:18:57.209554    2199 container_manager_linux.go:747] CPUAccounting not enabled for pid: 2199
W0829 13:18:57.209589    2199 container_manager_linux.go:750] MemoryAccounting not enabled for pid: 2199
I0829 13:18:57.240324    2199 factory.go:351] Registering Docker factory
W0829 13:18:57.240358    2199 manager.go:247] Registration of the rkt container factory failed: unable to
communicate with Rkt api service: rkt: cannot tcp Dial rkt api service: dial tcp [::1]:15441: getsockopt:
connection refused
I0829 13:18:57.240372    2199 factory.go:54] Registering systemd factory
```

```
I0829 13:18:57.240491    2199 factory.go:86] Registering Raw factory
I0829 13:18:57.240809    2199 manager.go:1121] Started watching for new ooms in manager
I0829 13:18:57.241320    2199 oomparser.go:185] oomparser using systemd
I0829 13:18:57.241600    2199 manager.go:288] Starting recovery of all containers
I0829 13:18:57.309415    2199 kubelet_node_status.go:247] Setting node annotation to enable volume controller
attach/detach
I0829 13:18:57.310925    2199 kubelet_node_status.go:82] Attempting to register node nodeb
I0829 13:18:57.317724    2199 kubelet_node_status.go:85] Successfully registered node nodeb
I0829 13:18:57.357376    2199 manager.go:293] Recovery completed
E0829 13:18:57.447104    2199 helpers.go:771] Could not find capacity information for resource
storage.kubernetes.io/scratch
W0829 13:18:57.447159    2199 helpers.go:782] eviction manager: no observation found for eviction signal
allocatableNodeFs.available
...
```

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.

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

user@nodeb:~$
```

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

Cluster "local" set.

user@nodeb:~$
```

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

Context "local" created.

user@nodeb:~$
```

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

Switched to context "local".

user@nodeb:~$
```

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

NAME      STATUS    AGE       VERSION
nodea     Ready     19m       v1.7.8+793658f2d7ca7
nodeb     Ready     1m        v1.7.8+793658f2d7ca7

user@nodeb:~$
```

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

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

...
user@nodeb:~$
```

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

nodea
nodeb

user@nodeb:~$
```

or to see the full output:

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

{
```

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

...

user@nodeb:~$
```

9. Running a pod on *nodeb*

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

```
...
spec:
  nodeName: nodeb
...
```

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.

Once you have confirmed your pod is running on *nodeb*, delete the pod.

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

```
pod "nginx" created

user@nodea:~$
```

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

NAME      READY     STATUS              RESTARTS   AGE
nginx     0/2       ContainerCreating   0          6s

user@nodea:~$
```

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

NAME      READY     STATUS    RESTARTS   AGE
nginx     2/2       Running   0          15s

user@nodea:~$
```

```
user@nodea:~$ kubectl describe pod nginx | grep Node:

Node:           nodeb/172.16.151.204

user@nodea:~$
```

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

CONTAINER ID        IMAGE                                          COMMAND                  CREATED             STATUS
PORTS               NAMES
f8de34c3b3cb        busybox                                        "/bin/sh -c 'while..."   32 seconds ago      Up 31
seconds                         k8s_log-truncator_nginx_default_0bdf232e-8cf8-11e7-af42-000c293215a2_0
c4cab97625e1        nginx                                          "nginx -g 'daemon ..."   34 seconds ago      Up 33
seconds                         k8s_nginx_nginx_default_0bdf232e-8cf8-11e7-af42-000c293215a2_0
f76086b38d65        gcr.io/google_containers/pause-amd64:3.0   "/pause"                 42 seconds ago      Up 41
seconds                         k8s_POD_nginx_default_0bdf232e-8cf8-11e7-af42-000c293215a2_0
```

```
user@nodeb:~$
```

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

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/pods",
    "resourceVersion": "349"
  },
  "items": [...]
}

user@nodeb:~$
```

```
user@nodeb:~$ kubectl delete pod nginx

pod "nginx" deleted

user@nodeb:~$
```

10. Start the scheduler

To run the scheduler we can simply execute the binary with a switch pointing it to the api server:

```
user@nodea:~$ ~user/k8s/_output/bin/kube-scheduler --master=http://nodea:8080

I0829 13:29:06.968289   47515 controller_utils.go:994] Waiting for caches to sync for scheduler controller
I0829 13:29:07.068738   47515 controller_utils.go:1001] Caches are synced for scheduler controller
I0829 13:29:07.069153   47515 leaderelection.go:179] attempting to acquire leader lease...
I0829 13:29:07.072525   47515 leaderelection.go:189] successfully acquired lease kube-system/kube-scheduler
I0829 13:29:07.072879   47515 event.go:218] Event(v1.ObjectReference{Kind:"Endpoints", Namespace:"kube-system",
Name:"kube-scheduler", UID:"b4c58ab0-8cf8-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"385",
FieldPath:""}): type: 'Normal' reason: 'LeaderElection' nodea became leader
...
```

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, let's submit some pods and see where they land.

Remove the nodeName configuration from your testpod spec and relaunch the pod.

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

apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  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: {}

user@nodea:~$
```

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

pod "nginx" created

user@nodea:~$
```

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

NAME      READY     STATUS    RESTARTS   AGE
nginx     2/2       Running   0          11s

user@nodea:~$
```

Where does the pod land?

```
user@nodea:~$ kubectl describe pod nginx | grep ^Node:

Node:           nodeb/172.16.151.204

user@nodea:~$
```

12. Launch additional pods

Create a second pod with a random name.

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

pod "nginx-19686" created

user@nodea:~$
```

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

```
NAME          READY     STATUS    RESTARTS   AGE
nginx         2/2       Running   0          59s
nginx-19686   2/2       Running   0          11s

user@nodea:~$
```

Determine which Node is our new pod running on (make sure to replace the pod name in the example below with one from your cluster)

```
user@nodea:~$ kubectl describe pod nginx-19686 | grep ^Node:

Node:           nodea/172.16.151.203

user@nodea:~$
```

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

```
...
I0829 13:30:03.810756   47515 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default",
Name:"nginx", UID:"d693a9d1-8cf8-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"427", FieldPath:""}):
type: 'Normal' reason: 'Scheduled' Successfully assigned nginx to nodeb
I0829 13:30:51.557096   47515 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
19686", UID:"f3099804-8cf8-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"487", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-19686 to nodea
I0829 13:32:02.710555   47515 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
13575", UID:"1d75698d-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"565", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-13575 to nodeb
I0829 13:32:07.156289   47515 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
19952", UID:"201b60db-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"591", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-19952 to nodea
I0829 13:32:10.366732   47515 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
24957", UID:"2205bf20-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"603", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-24957 to nodeb
...
```

Remove all the pods.

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

pod "nginx" deleted
pod "nginx-13575" deleted
pod "nginx-19686" deleted
pod "nginx-19952" deleted
pod "nginx-24957" deleted

user@nodea:~$
```

13. 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.

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

{
"kind" : "Policy",
"apiVersion" : "v1",
"predicates" : [
        {"name" : "PodFitsHostPorts"},
        {"name" : "PodFitsResources"},
        {"name" : "NoDiskConflict"},
        {"name" : "NoVolumeZoneConflict"},
```

```
        {"name" : "MatchNodeSelector"},
        {"name" : "HostName"}
        ],
"priorities" : [
        {"name" : "LeastRequestedPriority", "weight" : 1},
        {"name" : "BalancedResourceAllocation", "weight" : 1},
        {"name" : "ServiceSpreadingPriority", "weight" : 1},
        {"name" : "EqualPriority", "weight" : 1}
        ]
}

user@nodea:~$
```

Now restart the scheduler with the new policy:

```
user@nodea:~$ ~user/k8s/_output/bin/kube-scheduler \
--master=http://nodea:8080 \
--policy-config-file=custom.json \
--v=2

I0829 13:34:25.408319   48215 factory.go:351] 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}
I0829 13:34:25.408393   48215 factory.go:360] Registering predicate: PodFitsHostPorts
I0829 13:34:25.408400   48215 plugins.go:145] Predicate type PodFitsHostPorts already registered, reusing.
I0829 13:34:25.408404   48215 factory.go:360] Registering predicate: PodFitsResources
I0829 13:34:25.408407   48215 plugins.go:145] Predicate type PodFitsResources already registered, reusing.
I0829 13:34:25.408409   48215 factory.go:360] Registering predicate: NoDiskConflict
I0829 13:34:25.408412   48215 plugins.go:145] Predicate type NoDiskConflict already registered, reusing.
I0829 13:34:25.408415   48215 factory.go:360] Registering predicate: NoVolumeZoneConflict
I0829 13:34:25.408418   48215 plugins.go:145] Predicate type NoVolumeZoneConflict already registered, reusing.
I0829 13:34:25.408421   48215 factory.go:360] Registering predicate: MatchNodeSelector
I0829 13:34:25.408423   48215 plugins.go:145] Predicate type MatchNodeSelector already registered, reusing.
I0829 13:34:25.408426   48215 factory.go:360] Registering predicate: HostName
I0829 13:34:25.408429   48215 plugins.go:145] Predicate type HostName already registered, reusing.
I0829 13:34:25.408432   48215 factory.go:366] Registering priority: LeastRequestedPriority
I0829 13:34:25.408436   48215 plugins.go:245] Priority type LeastRequestedPriority already registered, reusing.
I0829 13:34:25.408441   48215 factory.go:366] Registering priority: BalancedResourceAllocation
I0829 13:34:25.408444   48215 plugins.go:245] Priority type BalancedResourceAllocation already registered,
reusing.
I0829 13:34:25.408451   48215 factory.go:366] Registering priority: ServiceSpreadingPriority
```

```
I0829 13:34:25.408454   48215 plugins.go:245] Priority type ServiceSpreadingPriority already registered, reusing.
I0829 13:34:25.408459   48215 factory.go:366] Registering priority: EqualPriority
I0829 13:34:25.408463   48215 plugins.go:245] Priority type EqualPriority already registered, reusing.
I0829 13:34:25.408468   48215 factory.go:401] Creating scheduler with fit predicates 'map[MatchNodeSelector:{}
HostName:{} PodFitsHostPorts:{} PodFitsResources:{} NoDiskConflict:{} NoVolumeZoneConflict:{}]' and priority
functions 'map[LeastRequestedPriority:{} BalancedResourceAllocation:{} ServiceSpreadingPriority:{} EqualPriority:
{}]
I0829 13:34:26.111696   48215 controller_utils.go:994] Waiting for caches to sync for scheduler controller
I0829 13:34:26.211918   48215 controller_utils.go:1001] Caches are synced for scheduler controller
I0829 13:34:26.212016   48215 leaderelection.go:179] attempting to acquire leader lease...
I0829 13:34:26.216145   48215 leaderelection.go:189] successfully acquired lease kube-system/kube-scheduler
I0829 13:34:26.216223   48215 event.go:218] Event(v1.ObjectReference{Kind:"Endpoints", Namespace:"kube-system",
Name:"kube-scheduler", UID:"b4c58ab0-8cf8-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"737",
FieldPath:""}): type: 'Normal' reason: 'LeaderElection' nodea became leader
...
```

Launch several pods and adjust the values above to see how it effects placement.

```
...
I0829 13:34:52.369608   48215 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
13119", UID:"8294e8c8-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"757", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-13119 to nodeb
I0829 13:34:53.441372   48215 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
16863", UID:"8338e344-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"768", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-16863 to nodea
I0829 13:34:54.296568   48215 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
5157", UID:"83bb1d0d-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"779", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-5157 to nodeb
I0829 13:34:54.992291   48215 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
17870", UID:"84243ce9-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"795", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-17870 to nodea
I0829 13:34:56.926779   48215 event.go:218] Event(v1.ObjectReference{Kind:"Pod", Namespace:"default", Name:"nginx-
32247", UID:"854cd00d-8cf9-11e7-af42-000c293215a2", APIVersion:"v1", ResourceVersion:"820", FieldPath:""}): type:
'Normal' reason: 'Scheduled' Successfully assigned nginx-32247 to nodeb
...
```

You can review the functions here:

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

and here:

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

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

```
user@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

user@nodea:~$
```

Congratulations you have successfully completed the scheduler lab!