Exercise 3.3: Finish Cluster Setup

1. View the available nodes of the cluster. It can take a minute or two for the status to change from `NotReady` to `Ready`. The `NAME` field can be used to look at the details. Your node name may be different, use YOUR `control-plane` name in future commands, if different than the book.

```
student@cp:~$ kubectl get node
```

```
NAME      STATUS   ROLES                  AGE     VERSION
cp        Ready    control-plane          28m     v1.32.1
worker    Ready    <none>                 50s     v1.32.1
```

2. Look at the details of the node. Work line by line to view the resources and their current status. Notice the status of `Taints`. The cp won't allow non-infrastructure pods by default for security and resource contention reasons. Take a moment to read each line of output, some appear to be an error until you notice the status shows `False`.

```
student@cp:~$ kubectl describe node cp
```

```
Name:               cp
Roles:              control-plane
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=cp
                    kubernetes.io/os=linux
                    node-role.kubernetes.io/control-plane=
                    node.kubernetes.io/exclude-from-external-load-balancers=
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket:
↪   unix:///var/run/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
<output_omitted>
```

3. Allow the cp server to run non-infrastructure pods. The cp node begins tainted for security and performance reasons. We will allow usage of the node in the training environment, but this step may be skipped in a production environment. Note the **minus sign (-)** at the end, which is the syntax to remove a taint. As the second node does not have the taint you will get a `not found` error. There may be more than one taint. Keep checking and removing them until all are removed.

```
student@cp:~$ kubectl describe node | grep -i taint
```

```
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
Taints:             <none>
```

```
student@cp:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

```
node/cp untainted
error: taint "node-role.kubernetes.io/control-plane" not found
```

```
student@cp:~$ kubectl describe node | grep -i taint
```

```
Taints:             <none>
Taints:             <none>
```

4. Determine if the DNS and Cilium pods are ready for use. They should all show a status of `Running`. It may take a minute or two to transition from `Pending`.

```
student@cp:~$ kubectl get pods --all-namespaces
```

```
NAMESPACE     NAME                               READY   STATUS    RESTARTS    AGE
kube-system   cilium-operator-788c7d7585-tnsph   1/1     Running   0           95m
kube-system   cilium-swjsj                       1/1     Running   0           95m
kube-system   coredns-5d78c9869d-dwds8           1/1     Running   0           100m
kube-system   coredns-5d78c9869d-t24p5           1/1     Running   0           100m

<output_omitted>
```

5. **Only if** you notice the `coredns-` pods are stuck in `ContainerCreating` status you may have to delete them, causing new ones to be generated. Delete both pods and check to see they show a `Running` state. Your pod names will be different.

```
student@cp:~$ kubectl get pods --all-namespaces
```

```
NAMESPACE     NAME                       READY   STATUS            RESTARTS    AGE
kube-system   cilium-swjsj               2/2     Running           0           12m
kube-system   coredns-576cbf47c7-rn6v4   0/1     ContainerCreating 0           3s
kube-system   coredns-576cbf47c7-vq5dz   0/1     ContainerCreating 0           94m
<output_omitted>
```

```
student@cp:~$ kubectl -n kube-system delete \
    pod coredns-576cbf47c7-vq5dz coredns-576cbf47c7-rn6v4
```

```
pod "coredns-576cbf47c7-vq5dz" deleted
pod "coredns-576cbf47c7-rn6v4" deleted
```

6. When it finished you should see more interfaces will be created . It may take up to a minute to be created. You will notice interfaces such as `cilium` interfaces when you deploy pods, as shown in the output below.

```
student@cp:~$ ip a
```

```
<output_omitted>
3: cilium_net@cilium_host: <BROADCAST,MULTICAST,NOARP,UP,LOWER_UP> mtu 1460 qdisc noqueue state UP
↪   group default qlen 1000
    link/ether be:19:22:da:62:ac brd ff:ff:ff:ff:ff:ff
    inet6 fe80::bc19:22ff:feda:62ac/64 scope link
       valid_lft forever preferred_lft forever
5: cilium_vxlan: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc noqueue state UNKNOWN group
↪   default qlen 1000
    link/ether ca:22:e7:23:42:89 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::c822:e7ff:fe23:4289/64 scope link
       valid_lft forever preferred_lft forever
<output_omitted>
```

7. Containerd may still be using an out of date notation for the `runtime-endpoint`. You may see errors about an undeclared resource type such as `unix//:`. We will update the **crictl** configuration. There are many possible configuration

options. We will set one, and view the configuration file that is created. We will also set this configuration on worker node as well for our convenience.

```
student@cp:˜$ sudo crictl config --set \
runtime-endpoint=unix:///run/containerd/containerd.sock \
--set image-endpoint=unix:///run/containerd/containerd.sock

student@worker:˜$ sudo crictl config --set \
runtime-endpoint=unix:///run/containerd/containerd.sock \
--set image-endpoint=unix:///run/containerd/containerd.sock

student@cp:˜$ sudo cat /etc/crictl.yaml
```

```
runtime-endpoint: "unix:///run/containerd/containerd.sock"
image-endpoint: "unix:///run/containerd/containerd.sock"
timeout: 0
debug: false
pull-image-on-create: false
disable-pull-on-run: false
```