Safely Draining a K8s Node

Relevant Documentation

- Draining a Node

Lesson Reference

Begin by creating some objects. We will examine how these objects are affected by the drain process.

First, create a pod.

```
vi pod.yml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
  restartPolicy: OnFailure
```

```
kubectl apply -f pod.yml
```

Create a deployment with two replicas.

```
vi deployment.yml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  labels:
    app: my-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-deployment
  template:
    metadata:
      labels:
        app: my-deployment
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

```
kubectl apply -f deployment.yml
```

Get a list of pods. You should see the pods you just created (including the two replicas from the deployment). Take node of which node these pods are running on.

```
kubectl get pods -o wide
```

Drain the node which the `my-pod` pod is running.

```
kubectl drain <node name> --ignore-daemonsets --force
```

Check your list of pods again. You should see the deployment replica pods being moved to the remaining node. The regular pod will be deleted.

```
kubectl get pods -o wide
```

Uncordon the node to allow new pods to be scheduled there again.

```
kubectl uncordon <node name>
```

Clean Up

Delete the deployment created for this lesson.

```
kubectl delete deployment my-deployment
```