Understanding Deployments

Relevant Documentation

- Deployment

Exam Tips

- A Deployment actively manages a desired state for a set of replica Pods.
- The Pod template provides the Pod configuration that the Deployment will use to create new Pods.
- The replicas field sets the number of replicas. You can scale up or down by changing this value.

Lesson Reference

Log in to the **control plane node**.

Create a Deployment.

```
vi nginx-deployment.yml
```

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

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

Check the status of the Deployment.

```
kubectl get deployments
```

Check the Deployment's replica Pods.

```
kubectl get pods
```

Scale the Deployment.

```
kubectl scale deployment/nginx-deployment --replicas=4
```

Check the Deployment's status and replica Pods.

```
kubectl get deployment nginx-deployment

kubectl get pods
```

Scale by editing the Deployment spec.

```
kubectl edit deployment nginx-deployment
```

```
spec:
  ...
  replicas: 2
```

Check the Deployment's status and replica Pods.

```
kubectl get deployment nginx-deployment

kubectl get pods
```

Copy the full name of one of the active Deployment Pods. Delete that Pod.

```
kubectl delete pod <Pod name> --force
```

List the Pods again. The Deployment will maintain desired state by creating a new replica to replace the deleted Pod.

```
kubectl get pods
```