Deployments are an important tool if you want to take full advantage of the automation capabilities provided by Kubernetes. In this lesson, we will discuss what deployments are and briefly mention some common use cases for Kubernetes deployments. We will also create a simple deployment in our cluster and explore how we can interact with it.Here are the commands used in this lesson:

Create a deployment:

```
cat <<EOF | kubectl create -f -apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
app: nginx
spec:
replicas: 2
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: nginx
image: nginx:1.15.4
ports: - containerPort: 80
EOF
```

Get a list of deployments:

```
kubectl get deployments
```

Get more information about a deployment:

```
kubectl describe deployment nginx-deployment
```

Get a list of pods:

```
kubectl get pods
```

You should see two pods created by the deployment.

While deployments provide a great way to automate the management of your pods, you need a way to easily communicate with the dynamic set of replicas managed by a deployment. This is where services come in. In this lesson, we will discuss what services are in Kubernetes, demonstrate how to create a simple service, and explore that service in our own cluster.

Here are the commands used in the demonstration:

- Create a NodePort service on top of your nginx pods:

```
cat << EOF | kubectl create -f -
kind: Service
apiVersion: v1
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 30080
  type: NodePort
EOF
```

- Get a list of services in the cluster.

```
kubectl get svc
```

You should see your service called `nginx-service`.

- Since this is a NodePort service, you should be able to access it using port 30080 on any of your cluster's servers. You can test this with the command:

```
curl localhost:30080
```

You should get an HTML response from nginx!