Deployment

In this lecture, we will discuss about Kubernetes deployment. For a minute, let us forget about pods, and ReplicaSets, and other Kubernetes concepts, and talk about how you might want to deploy your application in a production environment.

Say for example, you have a web server that needs to be deployed in a production environment. You need not one, but many such instances of the web server running for obvious reasons.

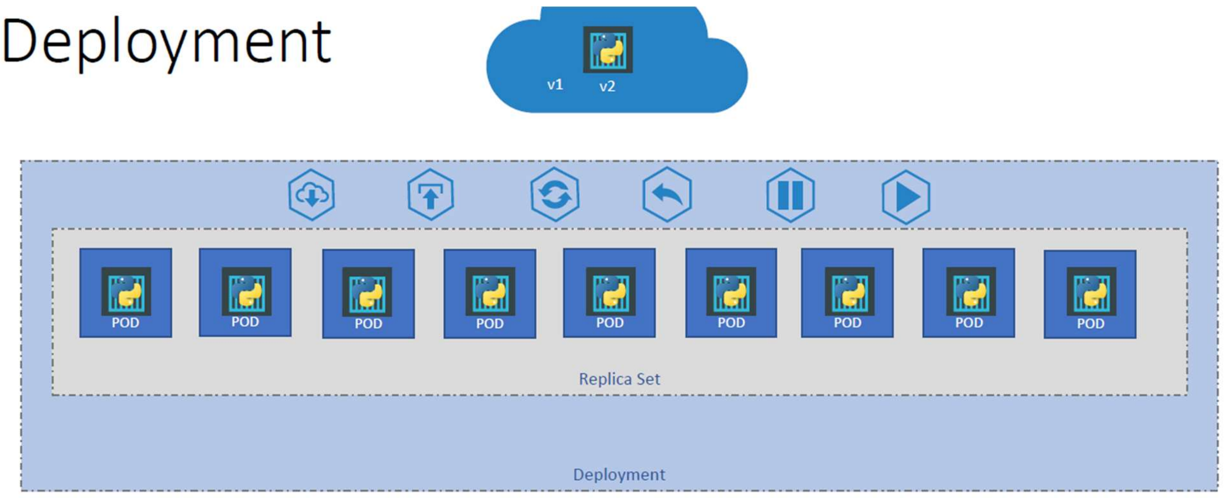
Secondly, whenever newer versions of application bills become available on the Docker Registry, you would like to upgrade your Docker instances seamlessly.

However, when you upgrade your instances, you do not want to upgrade all of them at once as we just did. This may impact users accessing our applications so you might want to upgrade them one after the other. And that kind of upgrade is known as rolling updates.

Suppose one of the upgrades you performed resulted in an unexpected error and you're asked to undo the recent change. You would like to be able to roll back the changes that were recently carried out.

Finally, say for example, you would like to make multiple changes to your environment, such as, upgrading the underlying web server versions, as well as scaling your environment and also modifying the resource allocations, et cetera. You do not want to apply each change immediately after the command is run. Instead, you would like to apply a pause to your environment, make the changes, and then resume so that all the changes are rolled out together. All of these capabilities are available with the Kubernetes deployments.

Deployment



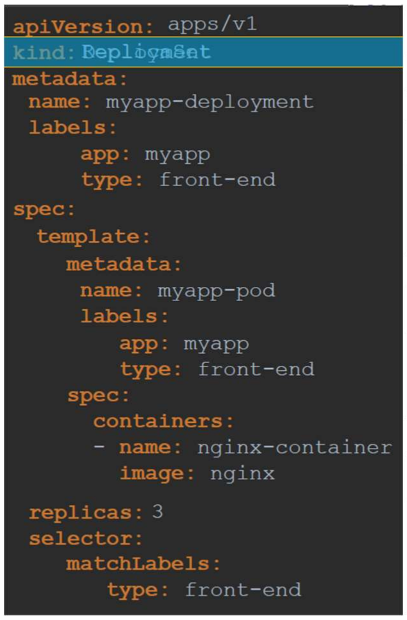
So far in this course, we discussed about pods, which deploy single instances of our application, such as the web application in this case. Each container is encapsulated in pods. Multiple such pods are deployed, using replication controllers or ReplicaSets. And then comes deployment which is a Kubernetes object that comes higher in the hierarchy. **The deployment provides us with the capability to upgrade the underlying instances seamlessly using rolling updates, undo changes, and pause, and resume changes as required**.

So, how do we create a deployment? As with the previous components, we first create a deployment definition file. The contents of the deployment definition file are exactly similar to the ReplicaSet definition file, except for the kind which is now going to be deployment.

If we walk through the contents of the file, it has an API version, which is apps/v1, metadata, which has name and labels, and a spec that has template, replicas, and selector. The template has a pod definition inside it.

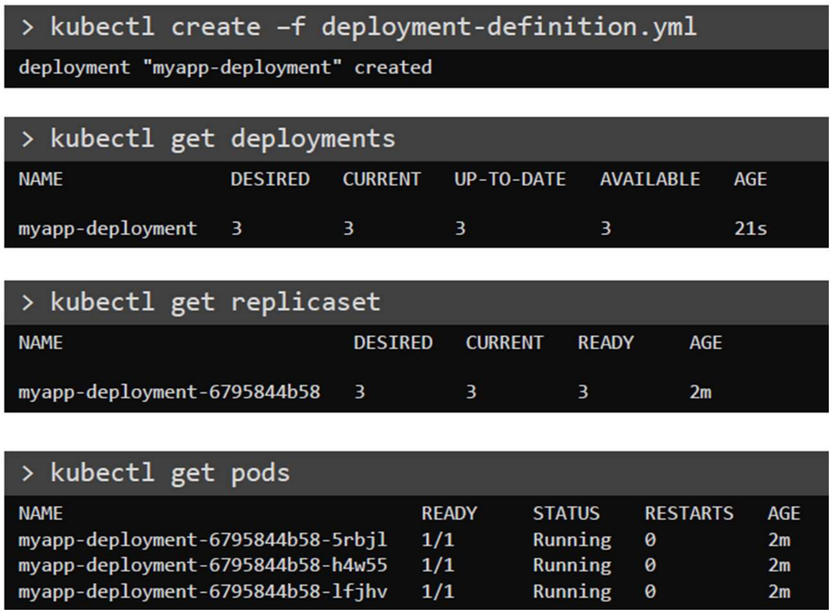
Once the file is ready, run the kubectl create command, and specify the deployment definition file. Then, run the cube control get deployment command to see the newly created deployment.

The deployment automatically creates a ReplicaSet so if you run the cube control, get ReplicaSet command, you will be able to see a new ReplicaSet in the name of the deployment. The ReplicaSets ultimately create parts, so if you run the cube control get parts command you will be able to see the pods with the name of the deployment and the ReplicaSet.



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
      app: myapp
      type: front-end
spec:
  template:
    metadata:
      name: myapp-pod
      labels:
         app: myapp
         type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx

  replicas: 3
  selector:
     matchLabels:
        type: front-end
```



```
> kubectl create -f deployment-definition.yml
deployment "myapp-deployment" created
```

```
> kubectl get deployments
NAME              DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE

myapp-deployment  3         3         3            3           21s
```

```
> kubectl get replicaset
NAME                         DESIRED   CURRENT   READY     AGE

myapp-deployment-6795844b58  3         3         3         2m
```

```
> kubectl get pods
NAME                               READY     STATUS    RESTARTS   AGE
myapp-deployment-6795844b58-5rbjl  1/1       Running   0          2m
myapp-deployment-6795844b58-h4w55  1/1       Running   0          2m
myapp-deployment-6795844b58-lfjhv  1/1       Running   0          2m
```

So far, there hasn't been much of a difference between ReplicaSet and deployments, except for the fact that deployment created a new Kubernetes object called deployments. We will see how to take advantage of the deployment using the use cases we discussed in the previous slide in the upcoming lectures.