Deploying Stateful Applications without Persisting State

In this lesson, we will deploy Jenkins, a stateful application, without persisting its state.

WE'LL COVER THE FOLLOWING

We'll start the exploration by deploying a stateful application without any mechanism to persist its state. That will give us a better insight into benefits behind some of the Kubernetes concepts and resources we'll use in this chapter.

Deploying Jenkins

We already deployed Jenkins a few times. Since it is a stateful application, it is an excellent candidate to serve as a playground.

Looking into the Definition

Let's take a look at a definition stored in the `pv/jenkins-no-pv.yml` file.

```
cat pv/jenkins-no-pv.yml
```

The YAML defines the `jenkins` Namespace, an Ingress controller, and a Service. We're already familiar with those types of resources so we'll skip explaining them and jump straight to the Deployment definition.

The output of the `cat` command, limited to the `jenkins` Deployment, is as follows.

```
...
apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  namespace: jenkins
spec:
  selector:
    matchLabels:
      app: jenkins
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
      - name: jenkins
        image: vfarcic/jenkins
        env:
        - name: JENKINS_OPTS
          value: --prefix=/jenkins
        - name: SECRETS_DIR
          value: /etc/secrets
        volumeMounts:
        - name: jenkins-creds
          mountPath: /etc/secrets
        resources:
          limits:
            memory: 2Gi
            cpu: 1
          requests:
            memory: 1Gi
            cpu: 0.5
      volumes:
      - name: jenkins-creds
        secret:
          secretName: jenkins-creds
```

There's nothing special about this Deployment. We already used a very similar one. Besides, by now, you're an expert at Deployment controllers.

The only thing worth mentioning is that there is only one volume mount and it references a secret we're using to provide Jenkins with the initial administrative user. Jenkins is persisting its state in `/var/jenkins_home`, and we are not mounting that directory.

Creating the Resources

Let's create the resources defined in `pv/jenkins-no-pv.yml`.

```
kubectl create \
    -f pv/jenkins-no-pv.yml \
    --record --save-config
```

The **output** is as follows.

```
namespace "jenkins" created
ingress "jenkins" created
service "jenkins" created
deployment "jenkins" created
```

We'll take a quick look at the events as a way to check that everything was deployed successfully.

```
kubectl --namespace jenkins \
    get events
```

The **output**, limited to relevant parts, is as follows.

```
...
2018-03-14 22:36:26 +0100 CET   2018-03-14 22:35:54 +0100 CET   7         jenkins-8768d486-lm
...
```

Creating the Secret

We can see that the setup of the only volume failed since it could not find the secret referenced as `jenkins-creds`. Let's create it.

```
kubectl --namespace jenkins \
    create secret \
    generic jenkins-creds \
    --from-literal=jenkins-user=jdoe \
    --from-literal=jenkins-pass=incognito
```

Verification

Now, with the secret `jenkins-creds` created in the `jenkins` Namespace, we can confirm that the rollout of the Deployment was successful.

```
kubectl --namespace jenkins \
    rollout status \
    deployment jenkins
```

We can see, from the output, that the `deployment "jenkins"` was `successfully rolled out`.

In the next lesson, we will analyze the failure of the Jenkins deployment due to the lack of persisting state.