Docker & Kubernetes

Demo Guide

SE4458 - System Architecture for Large Scale Systems

Step-by-Step Implementation Guide

Prepared by:

Ali Haktan SIGIN - 21070001004
Ulku Bartu SERBEST - 19070001034

Table of Contents

1. Prerequisites

Before starting the demo, make sure Docker and Kubernetes are running:

✓ Docker Desktop must be open and running

✓ Kubernetes must be enabled through Docker Desktop

Verification Commands:

```
docker info
```

Purpose: Checks if Docker daemon is running.

```
kubectl cluster-info
```

Purpose: Shows Kubernetes cluster information and verifies connection.

2. Building Docker Image

Step 2.1: Image Build

Let's build the Docker image for our demo application:

```
docker build -t demo-app:latest ./demo-app
```

What does this command do?

• **docker build**: Docker image creation command
• **-t demo-app:latest**: Names the image 'demo-app' with tag 'latest'
• **./demo-app**: Directory containing the Dockerfile (build context)

Step 2.2: Image Verification (Optional)

Let's list the created image:

```
docker images | grep demo-app
```

■ **Tip:** To check image contents:
`docker run --rm demo-app:latest ls -la /app`

3. Kubernetes Deployment

Step 3.1: Apply Deployment

Let's create the Kubernetes deployment and service:

```
kubectl apply -f deployment.yaml
```

What does this command do?

• **kubectl apply**: Creates or updates Kubernetes resources
• **-f deployment.yaml**: Applies definitions from YAML file
• **Deployment**: Defines how pods should run (replica count, image, etc.)
• **Service**: Provides external access to pods (NodePort type)

Step 3.2: Check Pod Status

Let's see if pods have started:

```
kubectl get pods -l app=demo-app
```

Expected Output: You should see 3 pods (READY: 1/1, STATUS: Running)

Step 3.3: Wait for Pods to be Ready

```
kubectl wait --for=condition=ready pod -l app=demo-app --timeout=60s
```

This command waits until all pods are ready (maximum 60 seconds).

4. Getting Service Port

Let's find out which NodePort the Kubernetes Service is accessible on:

```
kubectl get service demo-app
```

Example Output:

```
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
demo-app NodePort 10.96.123.45 <none> 8080:31234/TCP 2m
```

☞ You can access the application on port **31234** (may differ in your case)

Automatic Port Extraction:

```
kubectl get service demo-app -o jsonpath='{.spec.ports[0].nodePort}'
```

This command returns only the port number. You can use it like this:

```
NODE_PORT=$(kubectl get service demo-app -o jsonpath='{.spec.ports[0].nodePort}')
echo http://localhost:$NODE_PORT
```

5. Load Balancing Test

The Service automatically distributes requests among 3 pods. Let's test this:

Step 5.1: Browser Test

Open **http://localhost:[PORT]** in your browser.
You'll see different pod hostnames with each refresh (e.g., demo-app-7d7dfff488-f8zhz)

Step 5.2: Terminal Test

Send multiple requests to see different pods responding:

```
for i in {1..10}; do curl -s http://localhost:[PORT] | grep -o 'demo-app-[^<]*';
done
```

Expected Output: You should see 3 different pod names (random distribution)

■ **Demo Tip:** Show the class this:
1. Open 3-4 tabs in browser
2. Different pod names will appear in each tab
3. This demonstrates Kubernetes Service load balancing

6. Single Pod Forwarding (Port Forward)

Now let's bypass the Load Balancer and connect directly to a single pod:

Step 6.1: Get Pod Name

First, let's see the list of pods:

```
kubectl get pods -l app=demo-app
```

Example Output:

```
NAME READY STATUS RESTARTS AGE
demo-app-7d7dfff488-abc12 1/1 Running 0 5m
demo-app-7d7dfff488-def34 1/1 Running 0 5m
demo-app-7d7dfff488-ghi56 1/1 Running 0 5m
```

Step 6.2: Port Forward Connection

Choose a pod name and run the following command:

```
kubectl port-forward demo-app-7d7dfff488-abc12 31355:8080
```

What does this command do?

• **kubectl port-forward**: Forwards local port to pod
• **demo-app-7d7dfff488-abc12**: Target pod name (will differ in your case)
• **31355:8080**: Local port 31355 → Pod's port 8080
• Load Balancer is **bypassed**, goes only to this pod

Step 6.3: Test

Open **http://localhost:31355** in your browser.
Now no matter how many times you refresh, you'll see the **same pod name**!

■ **Demo Point:** This shows that port-forward bypasses the Load Balancer.
Service: Different pod each request (Load Balancing)
Port Forward: Same pod every request (Direct connection)

■ **Note:** To stop the port-forward command, press **Ctrl+C** in terminal.

7. Additional Commands and Tips

Scaling Pods Up/Down:

```
kubectl scale deployment demo-app --replicas=5
```

Scales replica count to 5 (more load balancing)

Watch Pods Live:

```
kubectl get pods -w
```

Monitors pod status in real-time (-w = watch)

View Logs:

```
kubectl logs -l app=demo-app --tail=50
```

Shows last 50 log lines from all demo-app pods

Specific Pod Logs:

```
kubectl logs demo-app-7d7dfff488-abc12 -f
```

Follows logs from specific pod in real-time (-f = follow)

Deployment Details:

```
kubectl describe deployment demo-app
```

Shows all deployment details and events

Service Details:

```
kubectl describe service demo-app
```

Shows service endpoints and configuration

■ **Pro Tip:** To see IP addresses of all pods:
`kubectl get pods -l app=demo-app -o wide`

8. Cleanup Operations

To clean up Kubernetes resources after the demo:

Delete Deployment and Service:

```
kubectl delete -f deployment.yaml
```

or manually:

```
kubectl delete deployment demo-app
kubectl delete service demo-app
```

Delete Docker Image (Optional):

```
docker rmi demo-app:latest
```

Check All Resources:

```
kubectl get all
```

Lists all resources in the namespace

✓ **Demo completed!** Good luck with your presentation.