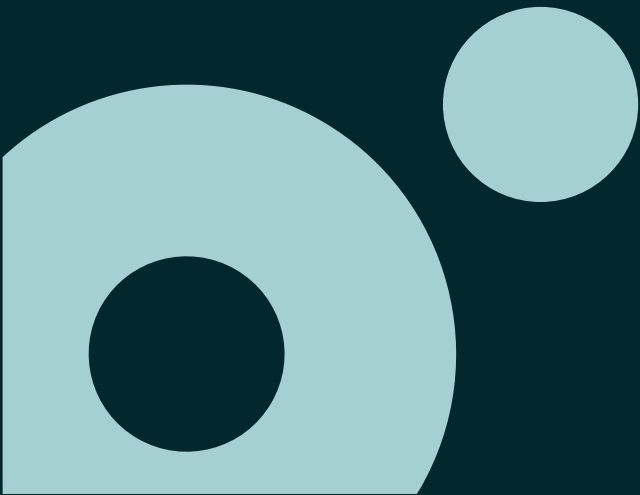


# Containers & Kubernetes

Session #09





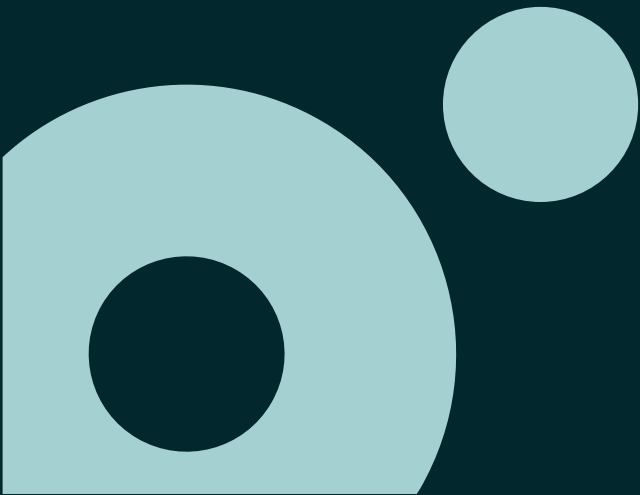
Monitoring and Operation

Kubernetes Dashboard

Auto-scaling: HPA

Lab

# Monitoring and Operation



# Best practices

## Monitoring & Operation

- Containers must write logs to STDOUT or STDERR
  - Log files can be lost when container is removed
  - Common monitoring platform automatically stream stdout and stderr
- Containers must implement only one service/process
- Use services for internal communication between pods
- Use ingress to allow communication from outside the cluster

# kubectl logs

## Monitoring & Operation

```
kubectl logs <pod> [-n <namespace>]
```

- Shows pod stdout and stderr
- Flag **-f** blocks the console and show new lines

# kubectl describe

## Monitoring & Operation

**kubectl describe pod <pod> [-n <namespace>]**

- Shows details about pod
  - Metadata
  - Network
- Lists all events occurred during pod lifecycle
- First place to go when pod don't have "Running" status

# kubectl describe

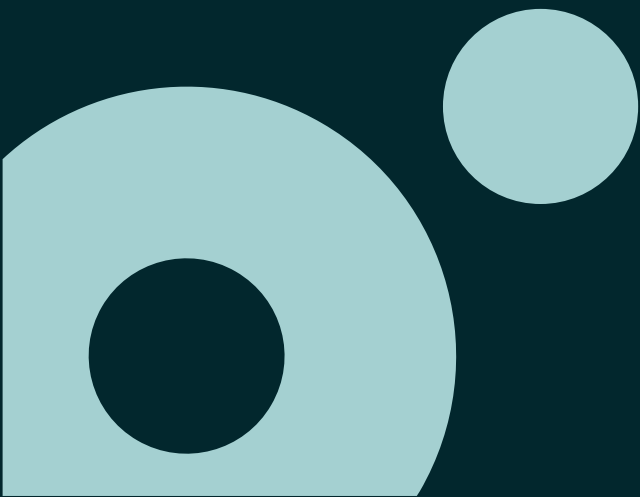
## Monitoring & Operation

```
kubectl port-forward pod <pod> [-n <ns>] hostport:podPort
```

```
kubectl port-forward svc <svc> [-n <ns>] hostport:podPort
```

- Maps a port on machine with pod port
- Allow to make direct requests
- When using service, maps directly to only one container (no load balancing)

# Kubernetes Dashboard





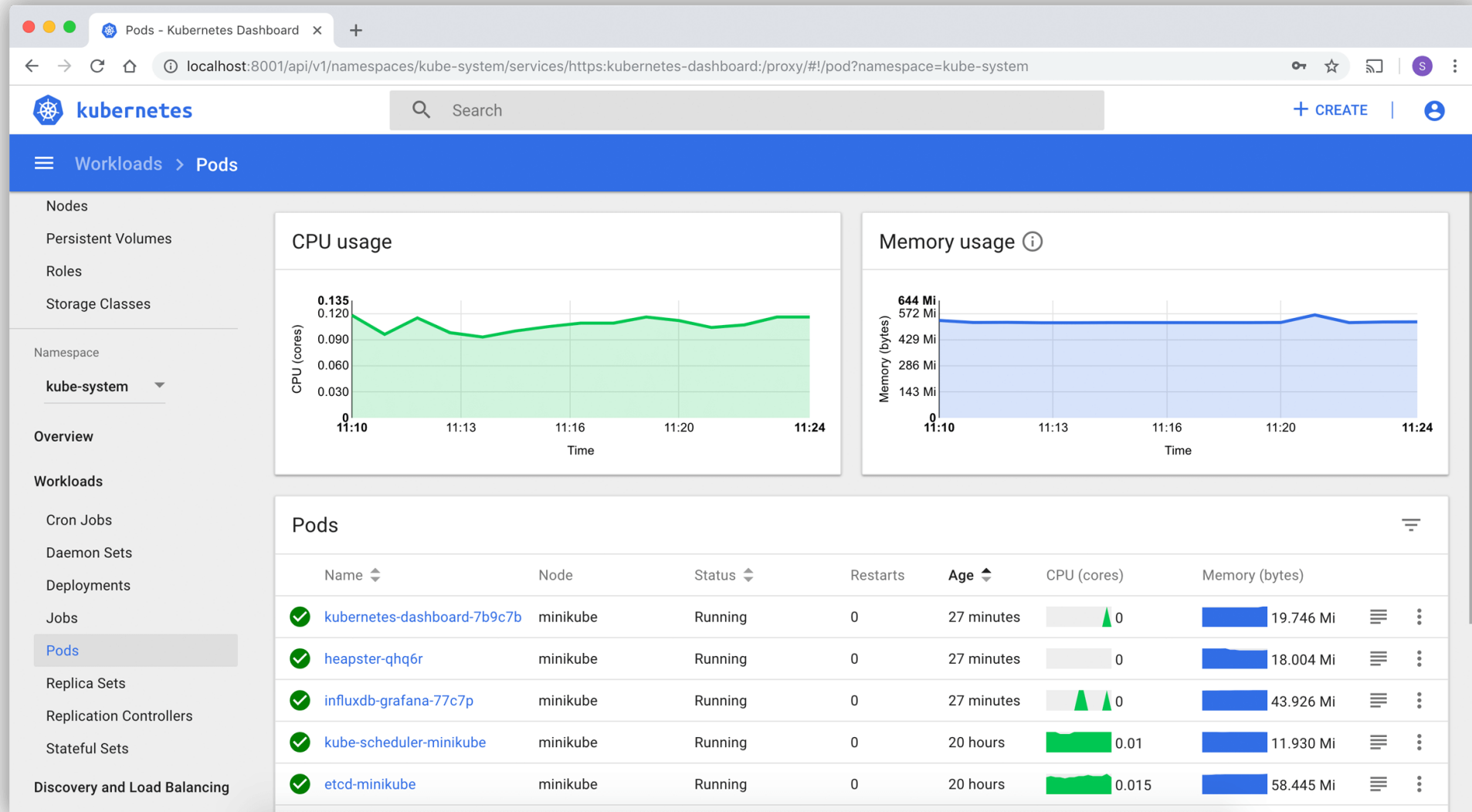
# Best practices

## Kubernetes Dashboard

- Web-based Kubernetes user interface.
- You can use Dashboard to deploy containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster resources.
- You can use Dashboard to get an overview of applications running on your cluster, as well as for creating or modifying individual Kubernetes resources. For example, you can scale a Deployment, initiate a rolling update, restart a pod or deploy new applications using a deploy wizard.
- Dashboard also provides information on the state of Kubernetes resources in your cluster and on any errors that may have occurred.

# Best practices

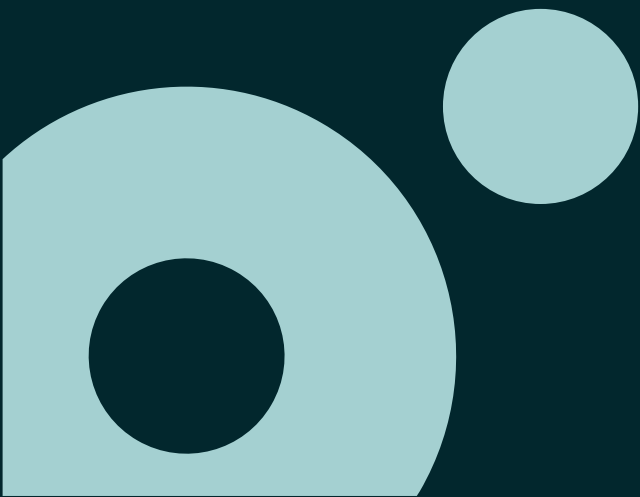
## Kubernetes Dashboard





Demo: Kubernetes Dashboard

# Auto-Scaling: HPA



# Motivation

## HPA

- Kubernetes can handle several replicas of the same pods
  - ReplicaSets handle replication
  - Services handle load balancing between them
- However, if the demand of a service starts to grow, the number of replicas deployed may be not sufficient to handle requests
- Number of replicas can be changed manually but it's not scalable
- Kubernetes have a HorizontalPodAutoscaler (HPA) object to handle scalability of a Deployment automatically

# HorizontalPodAutoscaler

## HPA

- Horizontal scaling means that the response to increased load is to deploy more Pods
- HPA defines a minimum and maximum number of replicas
- If the load increases, and the number of Pods is below the configured maximum, the HPA instructs the Deployment to scale up
- If the load decreases, and the number of Pods is above the configured minimum, the HPA instructs the workload resource to scale down
- HPA uses an interval (default is 15 seconds) to check if some change is needed

# Metrics

## HPA

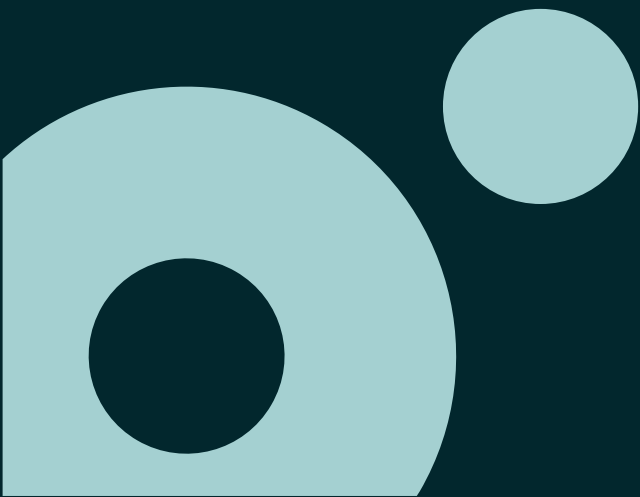
- To make the decision about scaling, HPA uses metrics about pods resources (CPU, Memory) utilization
- Metric target can be set as a percentage or an average value (preferable)
- Value used for check need to scale up/down is the average utilization of all pods
- Is mandatory that pods have resources (limits) defined

$$\text{desiredReplicas} = \text{ceil}[\text{currentReplicas} * (\text{currentMetricValue} / \text{desiredMetricValue})]$$

moOngy. | Demo: HPA



# Lab



# Lab 9: Monitoring and Operation

Github

[Lab 09 - Monitoring and Operation | docker-kubernetes-training  
\(tasb.github.io\)](https://github.com/tasb/docker-kubernetes-training)



● Rua Sousa Martins, nº 10  
1050-218 Lisboa | Portugal

