Getting Started with Kubernetes on Docker Desktop

Course Index

Docker Desktop is the easiest way to run Kubernetes on your local machine - it gives you a fully certified Kubernetes cluster and manages all the components for you.

In this lab you'll learn how to set up Kubernetes on Docker Desktop and run a simple demo app. You'll gain experience of working with Kubernetes and comparing the app definition syntax to Docker Compose.

1. Install Docker Desktop

Docker Desktop is freely available in a community edition, for Windows and Mac. Start by downloading and installing the right version for you:

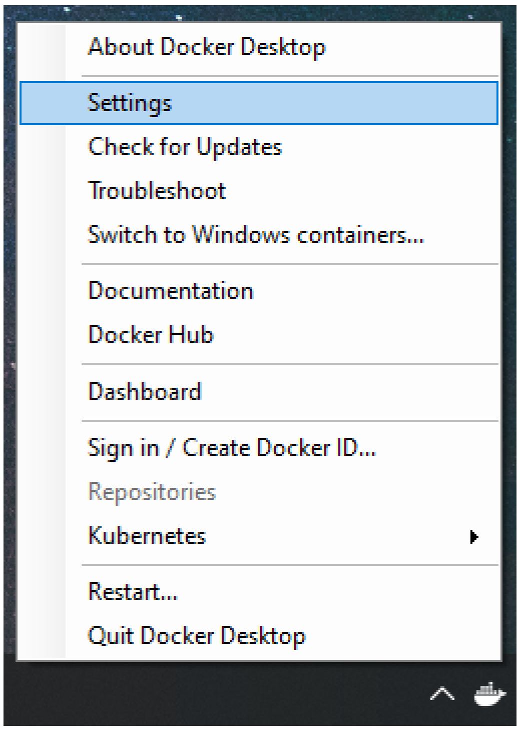
- Windows 10 (Professional or Enterprise)
- Mac OS X (Sierra 10.12 minimum)

  Older operating systems can't use Docker Desktop :(You can use Docker Toolbox instead, but that doesn't come with Kubernetes - so you'll need to run Kubernetes in Docker.

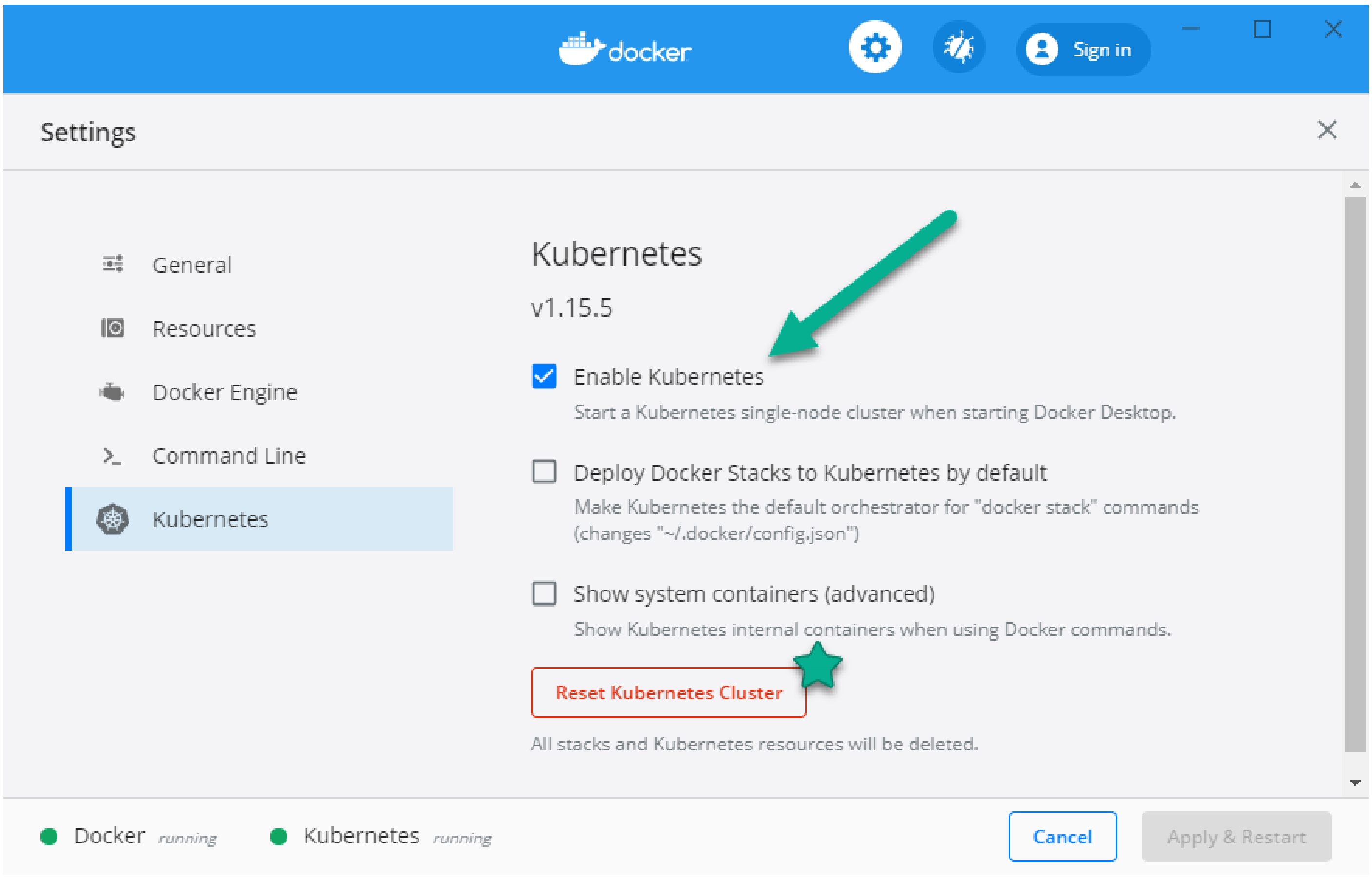
2. Enable Kubernetes

Kubernetes itself runs in containers. When you deploy a Kubenetes cluster you first install Docker (or another container runtime like containerd) and then use tools like kubeadm which starts all the Kubernetes components in containers. Docker Desktop does all that for you.

Make sure you have Docker Desktop running - in the taskbar in Windows and the menu bar on the Mac you'll see Docker's whale logo. Click the whale and select *Settings*:



A new screen opens with all of Docker Desktop's configuration options. Click on *Kubernetes* and check the *Enable Kubernetes* checkbox:


Settings

Kubernetes

v1.15.5

[✓] Enable Kubernetes
Start a Kubernetes single-node cluster when starting Docker Desktop.

[ ] Deploy Docker Stacks to Kubernetes by default
Make Kubernetes the default orchestrator for "docker stack" commands (changes "~/.docker/config.json")

[ ] Show system containers (advanced)
Show Kubernetes internal containers when using Docker commands.

Reset Kubernetes Cluster

All stacks and Kubernetes resources will be deleted.

● Docker *running* ● Kubernetes *running*

That's it! Docker Desktop will download all the Kubernetes images in the background and get everything started up. When it's ready you'll see two green lights in the bottom of the settings screen saying *Docker running* and *Kubernetes running*.

> The star in the screenshot shows the *Reset Kubernetes Cluster* button, which is one of the reasons why Docker Desktop is the best of the local Kubernetes options. Click that and it will reset your cluster back to a fresh install of Kubernetes.

3. Verify your Kubernetes cluster

If you've worked with Docker before, you're used to managing containers with the `docker` and `docker-compose` command lines. Kubernetes uses a different tool called `kubectl` to manage apps - Docker Desktop installs `kubectl` for you too.

Check the state of your Docker Desktop cluster:

```
kubectl get nodes
```

You should see a single node in the output called `docker-desktop`. That's a full Kubernetes cluster, with a single node that runs the Kubernetes API and your own applications.

The Kubernetes components are running in Docker containers, but Docker Desktop doesn't show them by default to keep things simple when you're running `docker` commands.

Try:

```
docker container ls
```

and you'll see zero containers (unless you have some of your own running).

But try:

```
docker info
```

And you'll see a whole bunch of containers in the running state (18 on my machine), which are the various parts of Kubernetes.

4. Run a familiar application

Let's run the classic Docker sample voting app! It's a distributed application which uses a Postgres database and Redis message queue, with application components running in Python, .NET and Node.js containers.

All the components of the app are published in public images on Docker Hub. All you need to run it is a Kubernetes manifest - a YAML files which describes all the components of the app. Here's the voting app definition in Kubernetes which you'll be deploying (compare it to the voting app definition in Docker Compose if you want to see how Kubernetes is different):

```
kubectl apply -f https://raw.githubusercontent.com/docker/docker-
birthday/master/resources/kubernetes-docker-desktop/vote.yaml
```

It'll take a couple of minutes for all the container images to download from Docker Hub and start up.

5. Check the app components

Kubernetes runs containers for you, so instead of explicitly running them with `docker container run`, you describe the desired outcome in a YAML file and when you run `kubectl apply` Kubernetes starts all the containers.

Containers in Kubernetes are wrapped in another object called a pod. Have a look at the pods for the voting app:

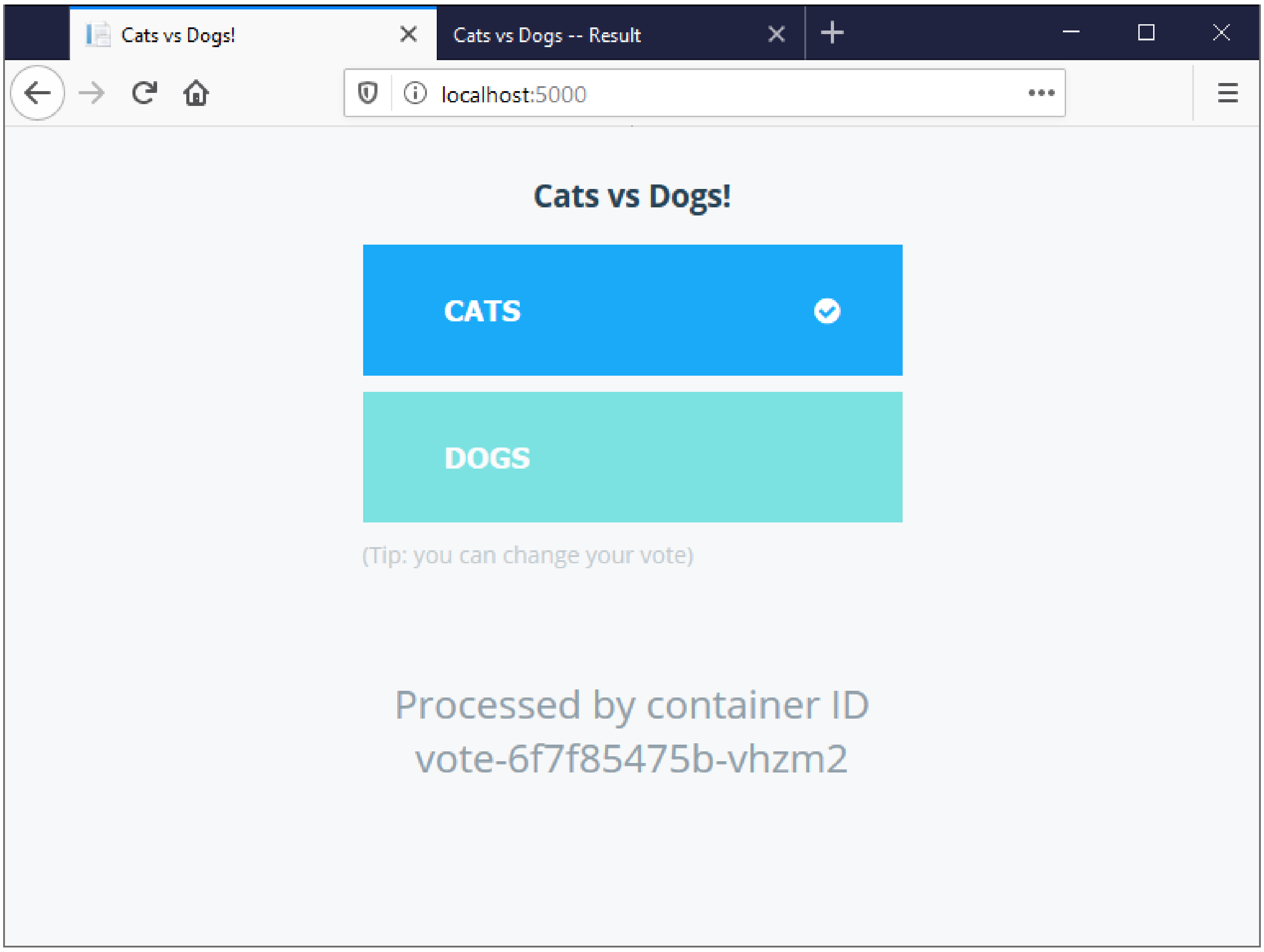
```
kubectl -n vote get pods
```

You should see lots of pods, with names starting `db-`, `redis-` etc. When the `READY` column says `1/1` for every pod, that means all the containers are running.

> You can have many containers in one pod in Kubernetes, and they share the same network and compute environment. That lets you do very cool things with the sidecar pattern.

6. Use the app

Browse to http://localhost:5000 and you should see the classic voting application:



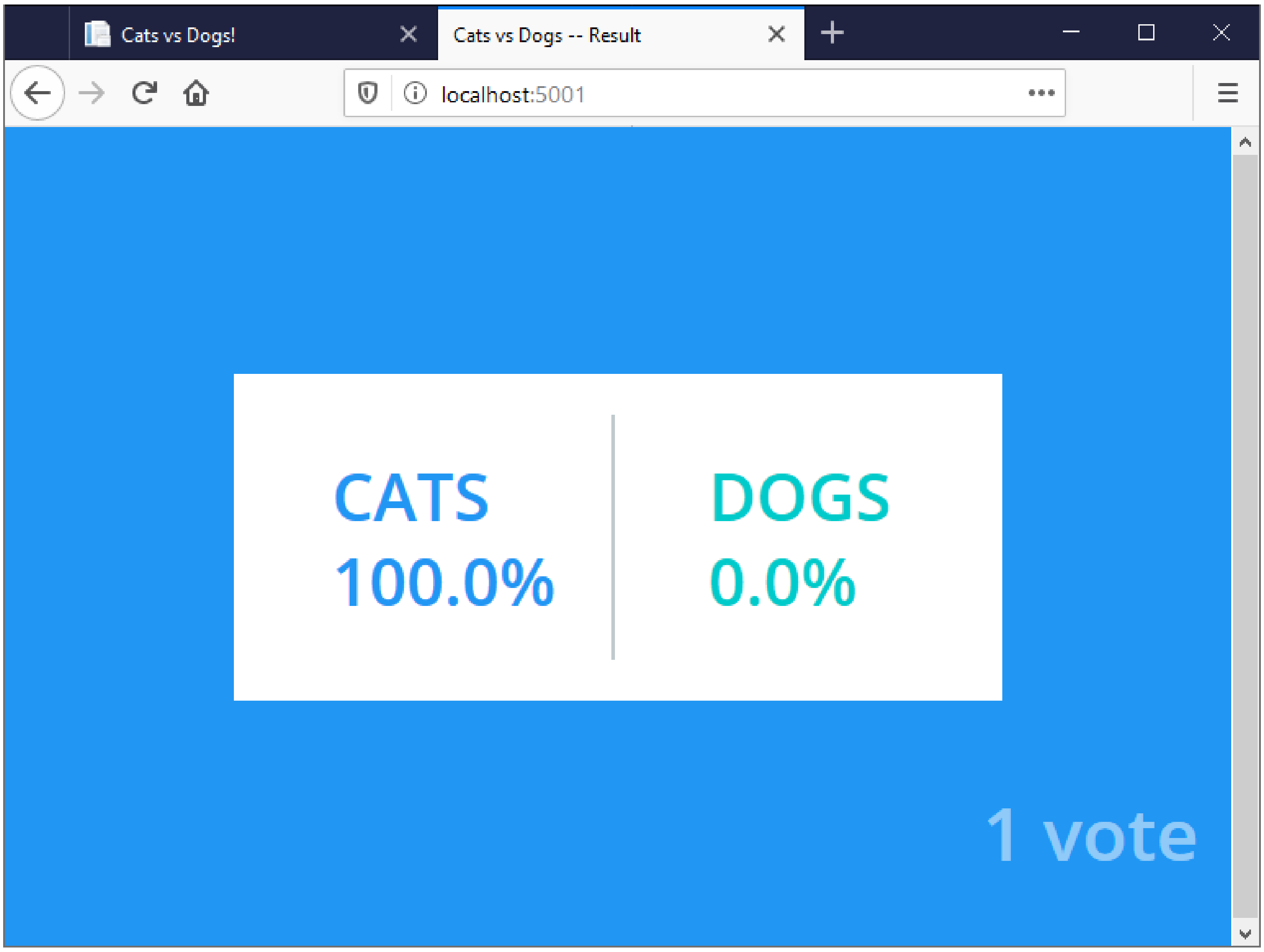
Cats vs Dogs!

(Tip: you can change your vote)

Processed by container ID vote-6f7f85475b-vhzm2

That's a Python application running in a Docker container, being managed by Kubernetes. Click on *Cats* (or *Dogs*) and the app sends a message to the Redis message queue. That message gets picked up by a .NET worker application, which updates a Postgres database.

Browse to http://localhost:5001 and you'll see the results:



That's a Node.js app which reads the data from Postgres. Everything is running in containers through Kubernetes.

7. Check the resilience

Kubernetes makes sure containers are running to keep your app at the service level you requested in the YAML file. They're all Docker containers which you can also manage with the `docker` command line.

Print the container ID for the result app:

```
docker container ls -f name='k8s_result*' --format '{{.ID}}'
```

And now remove that container:

```
docker container rm -f $(docker container ls -f name='k8s_result*' --format
'{{.ID}}')
```

Check back on the result app in your browser at http://localhost:5001 and you'll see it's still working. Kubernetes saw that the container had been removed and started a replacement straight away.

Print the result container ID again:

```
docker container ls -f name='k8s_result*' --format '{{.ID}}'
```

And you'll see it's a new container. Kubernetes makes sure the running app always matches the desired state in the application YAML file.

This was a simple introduction to Kubernetes, and there's an awful lot more to learn. The example app is a good place to explore, you can read up on services, volumes and namespaces - they're all used in this app.

If you want to learn more, check out:

- The Kubernetes Book by Docker Captain Nigel Poulton
- the Pluralsight course Kubernetes for Developers: Core Concepts by Docker Captain Dan Wahlin.

8. Teardown your environment

Docker Desktop has a feature to tear down your whole Kubernetes cluster and reset it to the original state. Open the settings from the Docker whale icon and click *Kubernetes*. Click the *Reset Kubernetes Cluster* button and the demo app will be gone.

Quiz

Please answer the following questions.

Question 0

What does Kubernetes do? Select only one option

-
-
-

Question 1

How do you work with the Kubernetes API? Select only one option

-
-
-

Question 2

How does Kubernetes group containers together? Select only one option

-
-

-