docker kubernetes

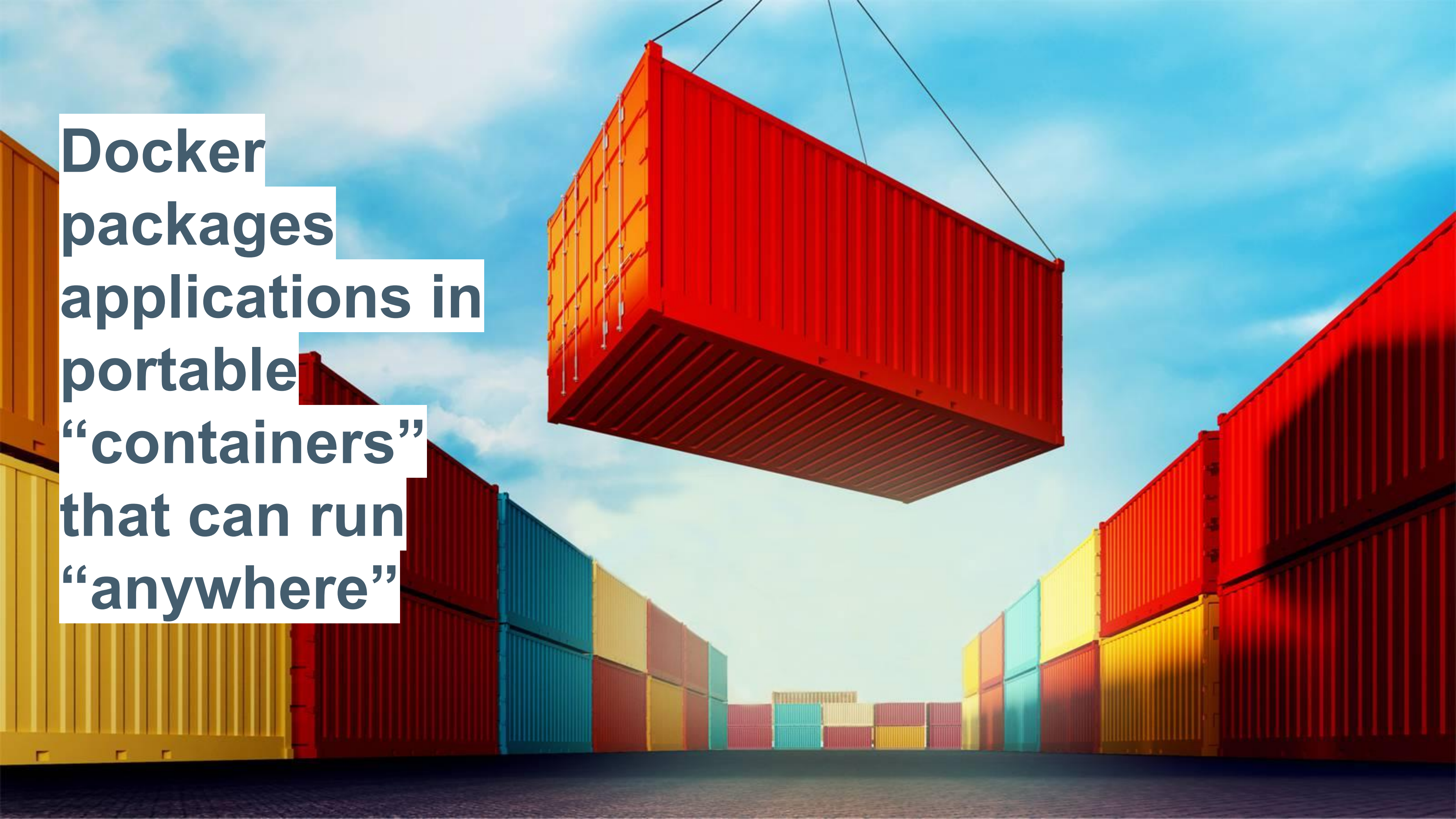
“It works on my computer!!!”

Who never said that?

“Docker enables true independence between **applications** and **infrastructure**”

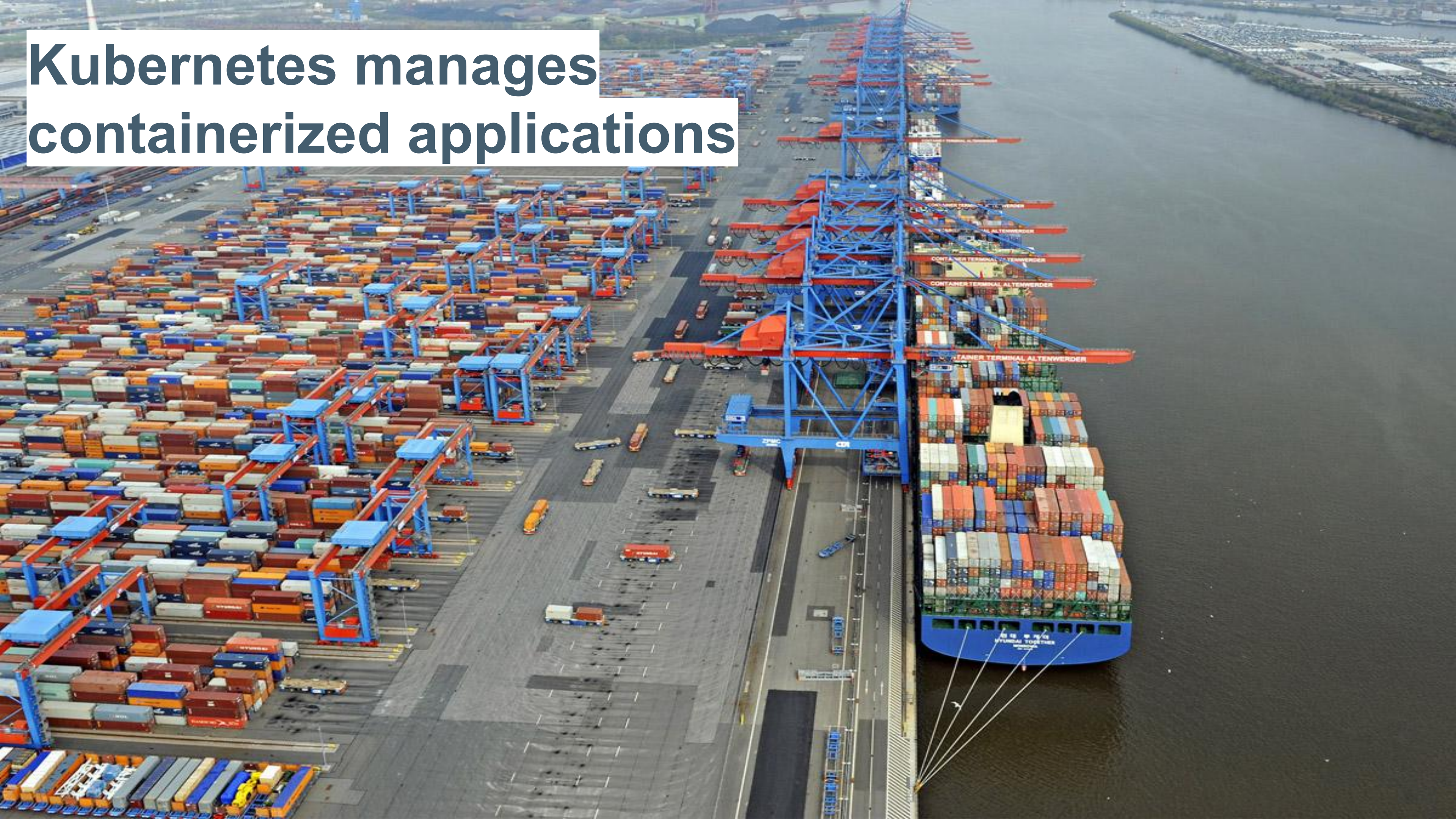
https://www.docker.com/what-docker

(hint)



Docker packages applications in portable “containers” that can run “anywhere”

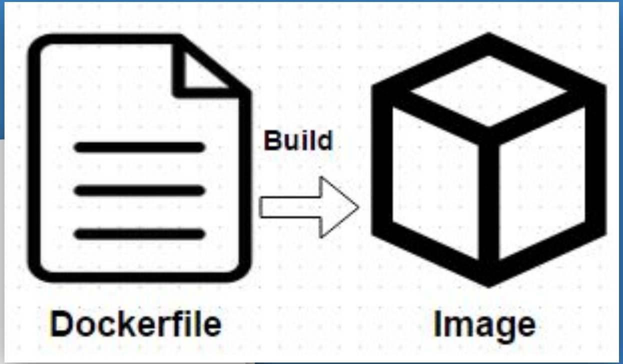
Kubernetes manages containerized applications



```
$ docker images
```

https://hub.docker.com/search/?q=python

```
$ docker pull IMAGE
$ docker run IMAGE
$ docker run IMAGE CMD
$ docker ps
$ docker build -f FILE -t IMAGE DIR
```



```
FROM python:alpine

CMD python --version
```

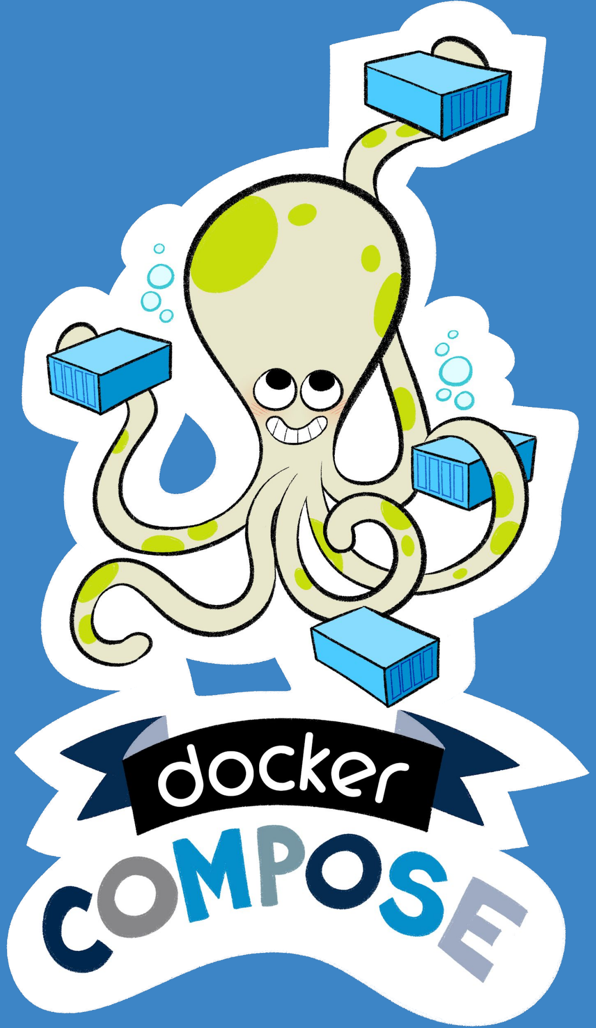
Dockerfile

```yaml
version: "3"
services:
  app-name:
    build: .
```

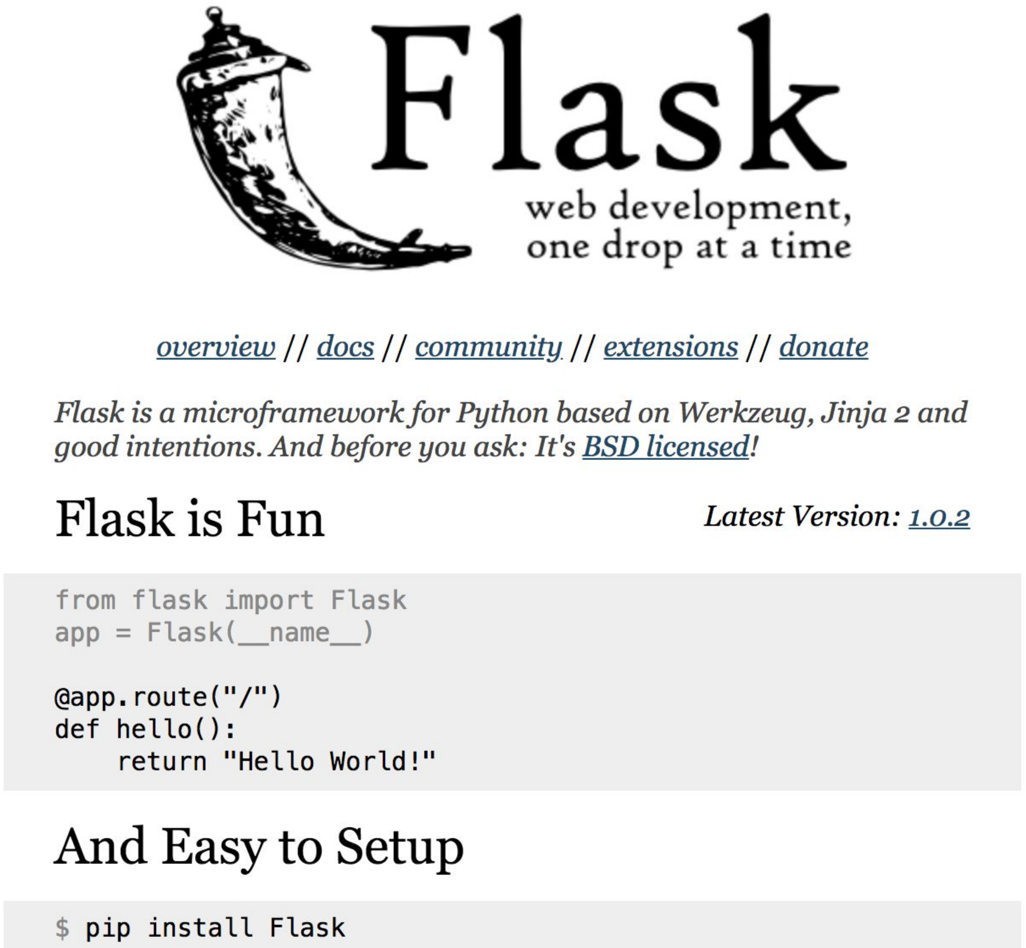
docker-compose.yml

"Compose is a tool for **defining** and **running** multi-container Docker applications"

https://docs.docker.com/compose/overview/



End of Part 1 (of 2)



overview // docs // community // extensions // donate

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's BSD licensed!

Flask is Fun

Latest Version: 1.0.2

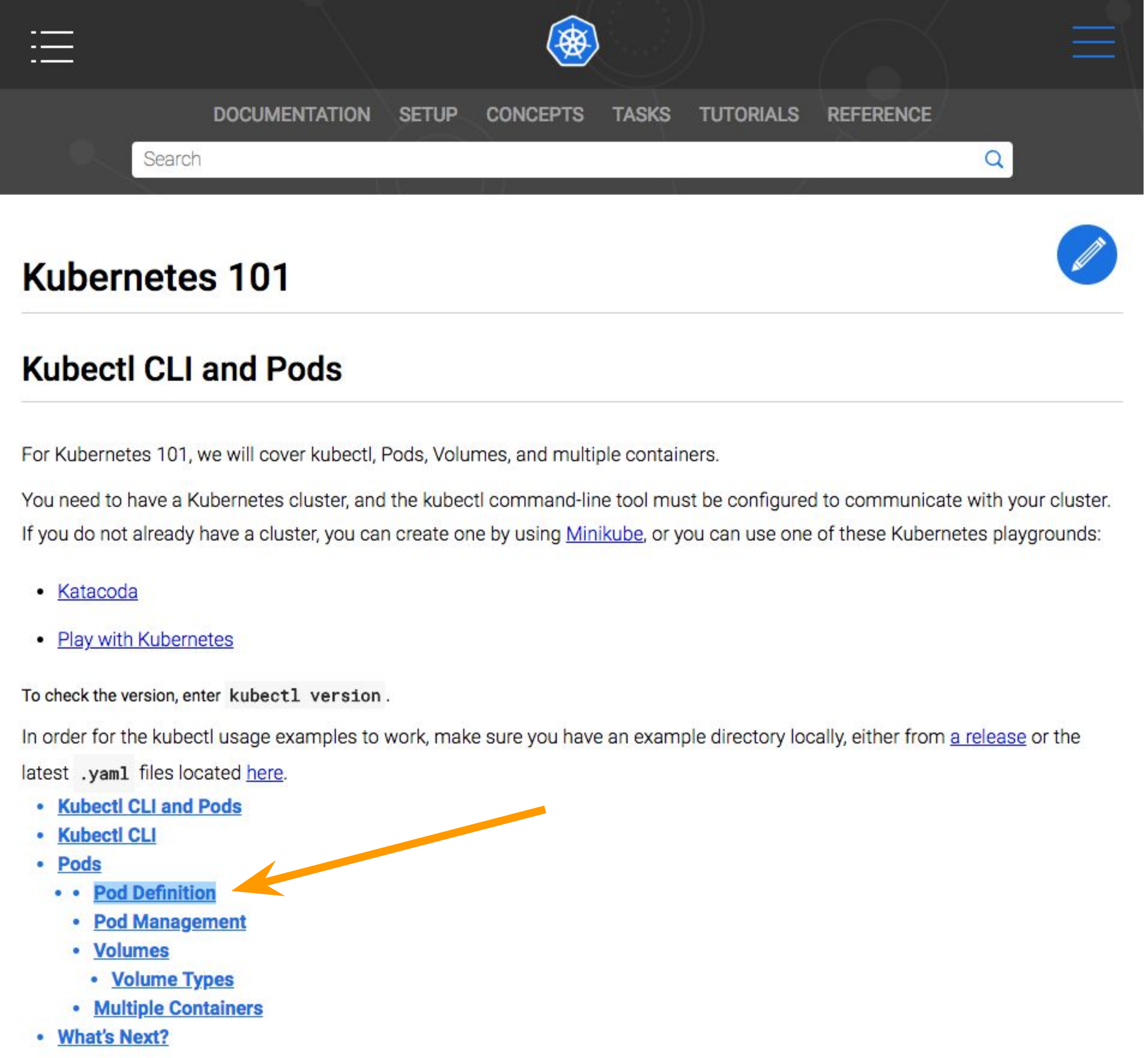
```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

And Easy to Setup

```
$ pip install Flask
```

http://flask.pocoo.org/

(google “flask python”)



Kubernetes 101

Kubectl CLI and Pods

For Kubernetes 101, we will cover kubectl, Pods, Volumes, and multiple containers.

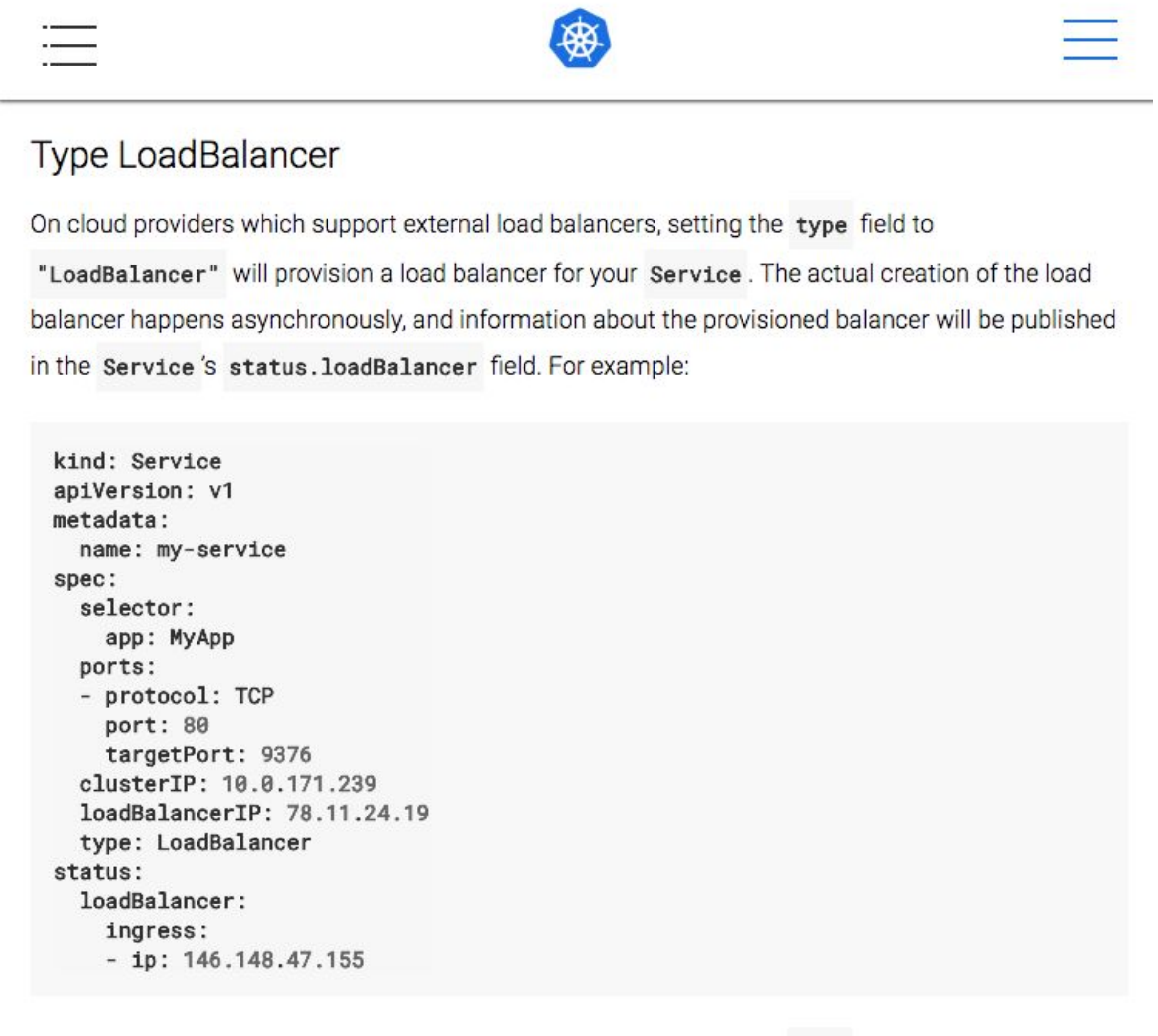
You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. If you do not already have a cluster, you can create one by using Minikube, or you can use one of these Kubernetes playgrounds:

- Katacoda
- Play with Kubernetes

To check the version, enter `kubectl version`.

In order for the kubectl usage examples to work, make sure you have an example directory locally, either from a release or the latest `.yaml` files located here.

(google "k8s 101")



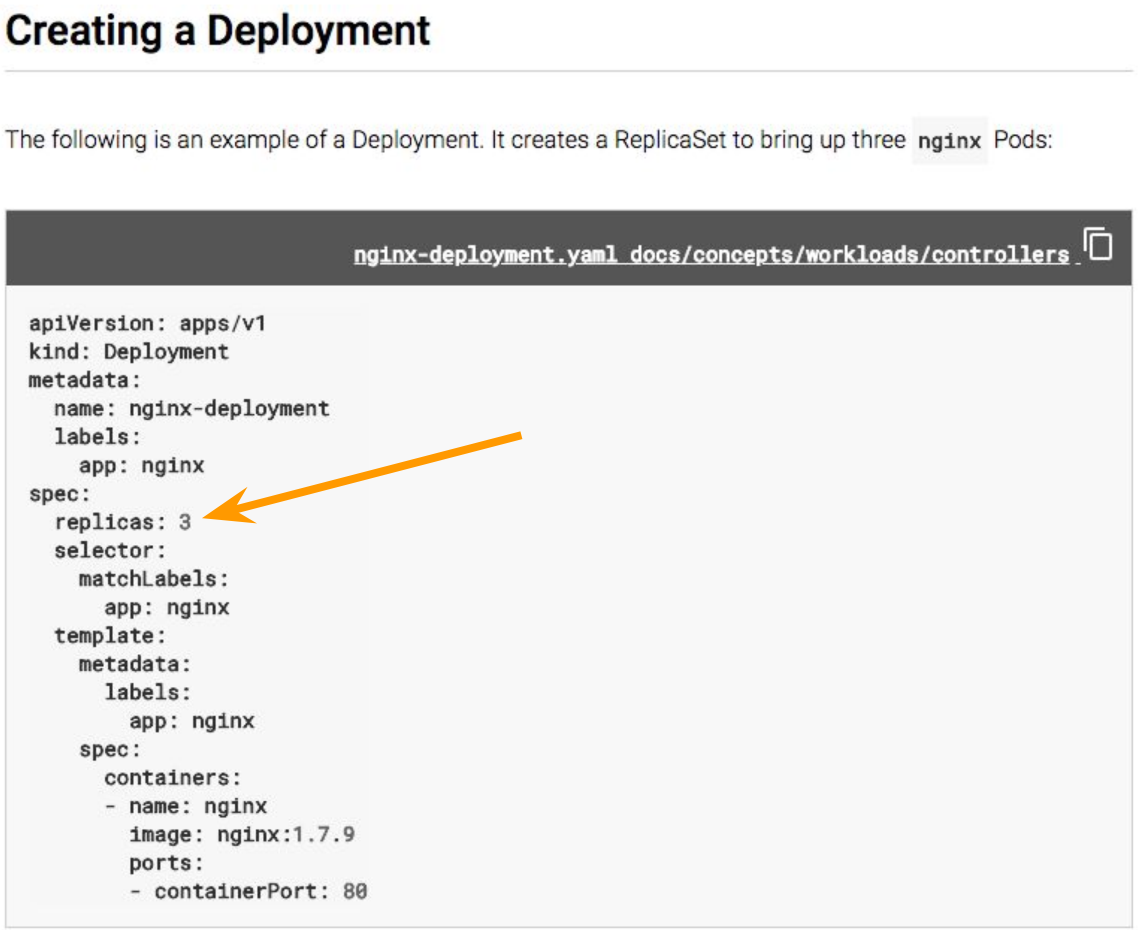
Type LoadBalancer

On cloud providers which support external load balancers, setting the `type` field to `"LoadBalancer"` will provision a load balancer for your `Service`. The actual creation of the load balancer happens asynchronously, and information about the provisioned balancer will be published in the `Service`'s `status.loadBalancer` field. For example:

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  clusterIP: 10.0.171.239
  loadBalancerIP: 78.11.24.19
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 146.148.47.155
```

https://kubernetes.io/docs/concepts/services-networking/service/#type-loadbalancer

(google "k8s services")



Creating a Deployment

The following is an example of a Deployment. It creates a ReplicaSet to bring up three `nginx` Pods:

nginx-deployment.yaml docs/concepts/workloads/controllers

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

(google "k8s deployments")

Thank you for listening!

vitorenesduarte@gmail.com
@vitorenesduarte