ALPEN-ADRIA UNIVERSITÄT KLAGENFURT

Clustering with (µ)Kubernetes

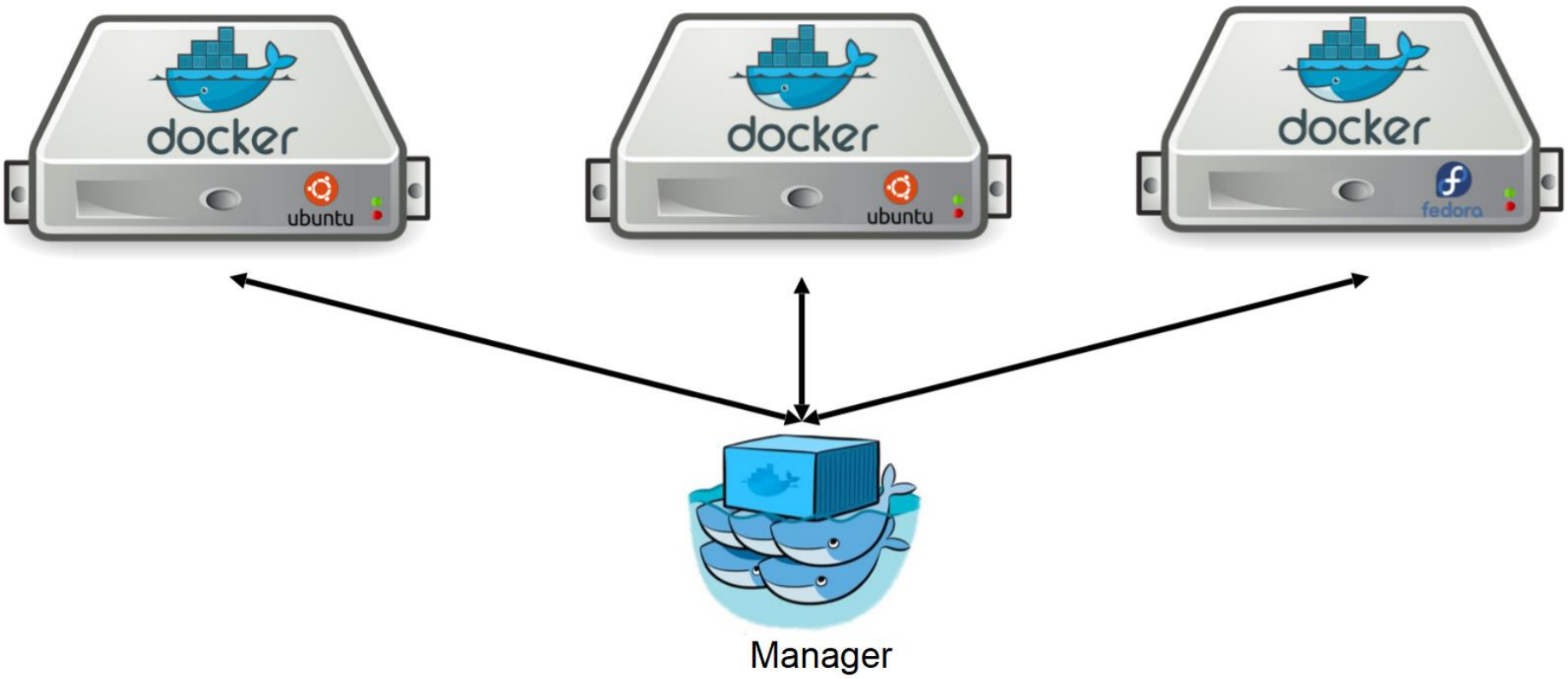
Narges Mehran, MSc.

Current Topics in Distributed Systems:
Internet of Things and Cloud Computing,

SS2021

23.06.2021

Container Orchestration



https://docs.docker.com/

Container Orchestration (cont.)

When you have large applications deployed on a lot of containers,

- you need a dedicated monitoring system.

Container orchestration refers to the process of organizing multiple-containerized applications,

- guides container deployment
- automates updates, health monitoring, and failover procedures.

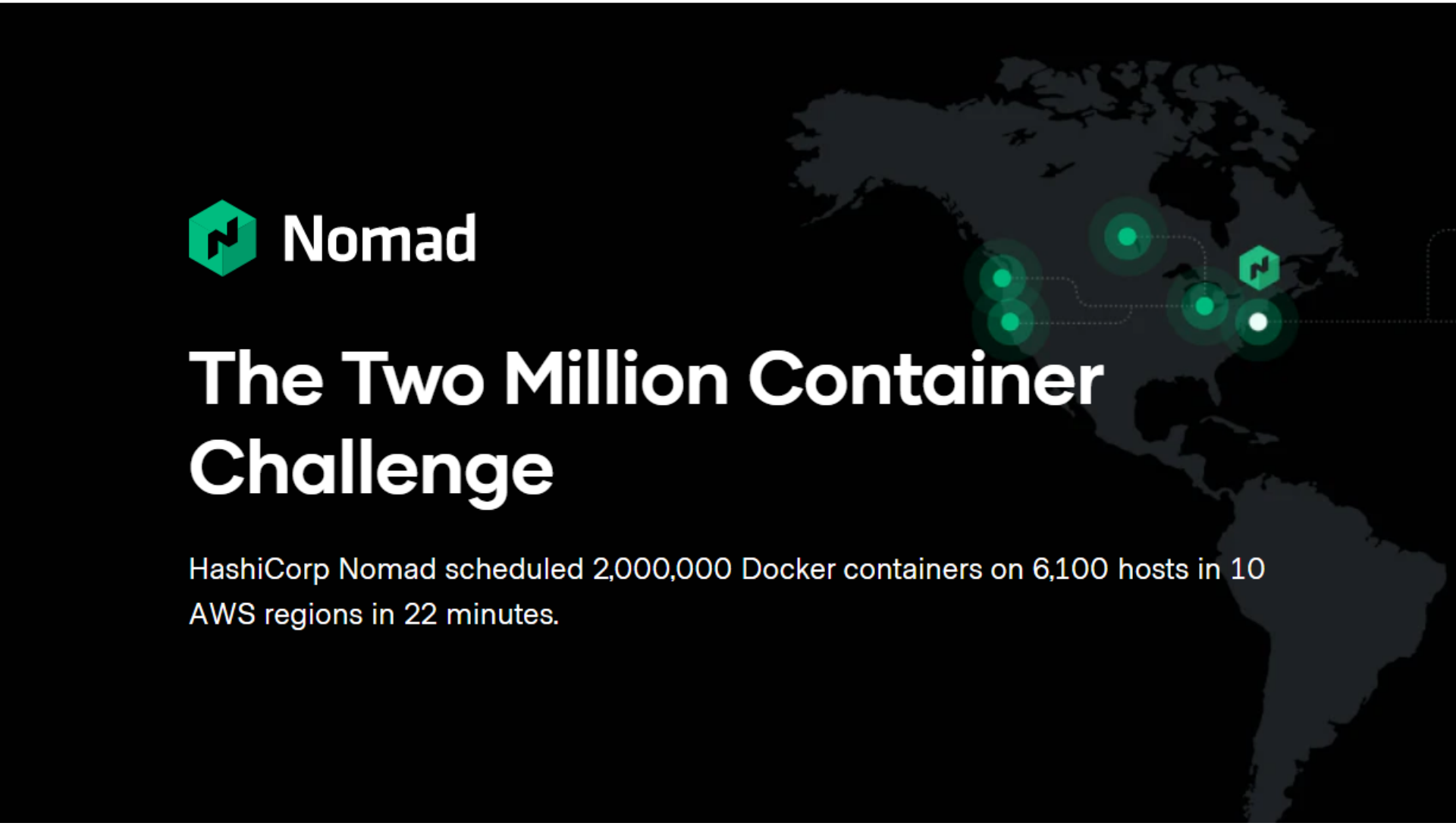
```
docker service create --replicas=1000 node
```

Platforms such as:

- Apache Mesos,
- Google Kubernetes,
- Docker Swarm(kit),
- Nomad

hashicorp.com/c2m

HashiCorp Products Solutions Company Partners



Nomad

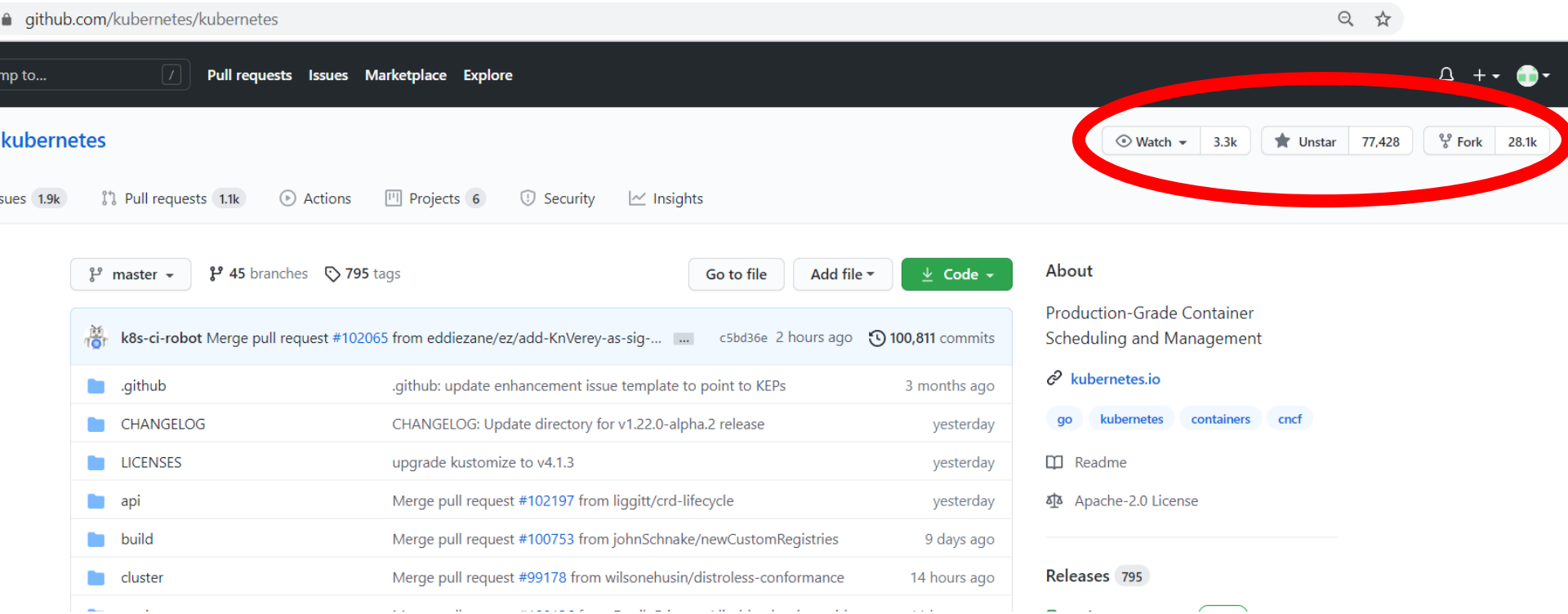
The Two Million Container Challenge

HashiCorp Nomad scheduled 2,000,000 Docker containers on 6,100 hosts in 10 AWS regions in 22 minutes.

Container Orchestration (cont.)

Platforms such as:

- Google Kubernetes → one of the top-ranked projects in GitHub



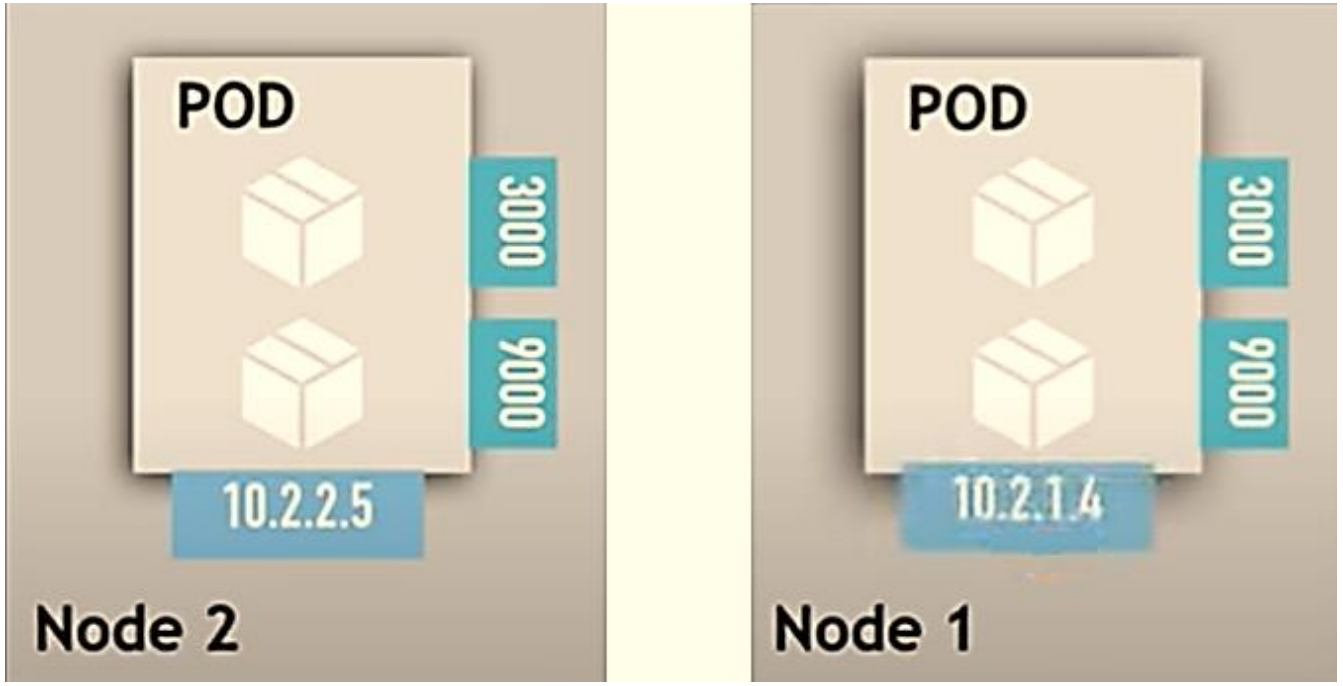
Kubernetes

Kubernetes is an open-source orchestration system for automating the management, placement, scaling and routing of containers that has become popular with developers and IT operations teams in recent years.
It was first developed by Google and contributed to Open Source in 2014 and is now maintained by the Cloud Native Computing Foundation.
There is an active Kubernetes community and ecosystem developing around Kubernetes with thousands of contributors and dozens of certified partners.

Pod

- Pod is the smallest and simplest Kubernetes object.
- Pod represents a set of running containers on a cluster.
- Pod is typically set up to run a single primary container.
- Pods are commonly managed by a Deployment.
- The containers belong to the same Pod expose a single private IP address to the rest of Kubernetes system.
- They can communicate with various IP addresses.

Pod (cont.)



Pod (cont.)

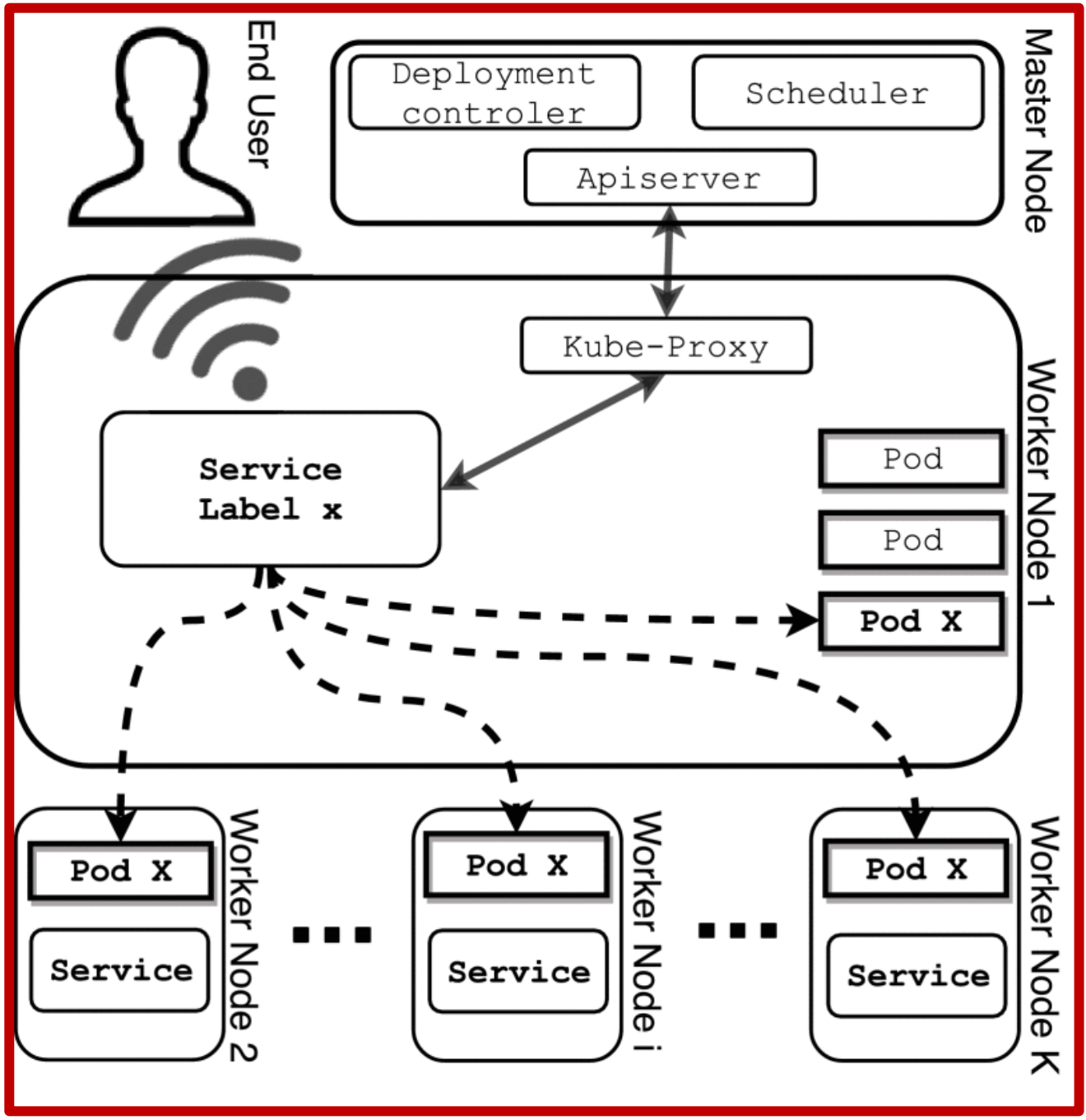
Containers in a pod share a common network interface and each container has access to all storage volumes assigned to the pod.

Pods are not usually managed directly by developers.

Application developers are expected to create a Deployment Controller which oversees the creation and management of a set of identical pods providing the expected functionality.

It can dynamically add, and remove pods to/from the set, for example to adjust the processing capacity according to workload variations or to deal with user’s mobility.

Organization of a Kubernetes service.



Fahs, Ali, and Guillaume Pierre. "Proximity-aware traffic routing in distributed fog computing platforms." CCGrid 2019.

Kubernetes command-line tool

The Kubernetes command-line tool for controlling Kubernetes clusters, kubectl

You can use this to create and inspect objects in your Kubernetes cluster.

https://kubernetes.io/docs/reference/kubectl/overview/

Some features of *kubectl*

With Docker, you were able to run a single instance of an app, but with Kubernetes you can run a large number or even scale it up:

- kubectl run --replica=1000 my-web-server
- kubectl scale --replica=2000 my-web-server

It can be done automatically based on the users' load:

- kubectl rolling-update my-web-server --image=web-server:2
- kubectl rolling-update my-web-server --rollback

Kubernetes Control Plane

The Kubernetes control plane consists of a collection of processes running on a cluster:

The **Kubernetes Master** is a collection of three processes that run on a single node in a cluster, which is designated as the master node,

Master node

Those processes are:

- kube-apiserver
  - The Kubernetes API server validates and configures data for the API objects which include pods, services, replication-controllers, and others. The API server provides the frontend to the cluster's shared state through which all other components interact.
- kube-controller-manager
  - Control plane component that runs controller processes.
- kube-scheduler
  - It distributes workloads across multiple nodes.

Non-master node

A non-master node in a cluster executes two processes:

- kubelet: which communicates with the Kubernetes Master.
- kube-proxy: which is a network proxy and runs Kubernetes networking services on each node.

Getting started with Kubernetes

There are different options to set up, provision and run Kubernetes.

You can deploy a Kubernetes cluster on

- a local machine,
- Cloud and Fog computing environment,
- on-prem datacenter,
- or choose a managed Kubernetes cluster.

You can also create custom solutions across a wide range of cloud providers, or bare metal environments.

Interactive shell provided by K8s

https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/

Build Kubernetes-ready applications on your desktop

Play with Kubernetes for free

Minikube

A tool for running Kubernetes locally.

Minikube runs a single-node cluster inside a VM on your computer:

- curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube_latest_amd64.deb
- sudo dpkg -i minikube_latest_amd64.deb

- minikube start

MicroK8s

MicroK8s is

- a small, fast, single-package Kubernetes for developers, IoT and edge,

in other words,

- it is a minimal, lightweight Kubernetes you can run and use on practically any machine.
- it works on 42 flavors of Linux.

https://microk8s.io/docs

```
$ sudo snap install microk8s --channel=1.18 --classic
```

MicroK8s (cont.)

MicroK8s only installs the basics of a usable Kubernetes features:

- api-server
- controller-manager
- scheduler
- Kubelet
- kube-proxy

Adding a node

To create a cluster out of two or more already-running MicroK8s instances, use:

- $ microk8s add-node

If you run this command on a MicroK8s instance, it will be the master of the cluster and will host the Kubernetes control plane.

add-node command prints a *microk8s join* command which should be executed on the MicroK8s instance that you wish to join to the cluster:

- $ microk8s join ip-172-31-20-243:25000/ArvwtYMnHJBQkJrFzFsSmSPNDTEojrlS

Several commands of (μ)Kubernetes

$ microk8s.

- kubectl get services # List all services in the namespace
- kubectl get pods -o wide # List all pods in the current namespace, with more details
- kubectl get deployment my-dep # List a particular deployment
- kubectl get pods # List all pods in the namespace
- kubectl get pod my-pod -o yaml # Get a pod's YAML

- $ microk8s kubectl get pods

```
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          11m
```

Namespaces in Kubernetes

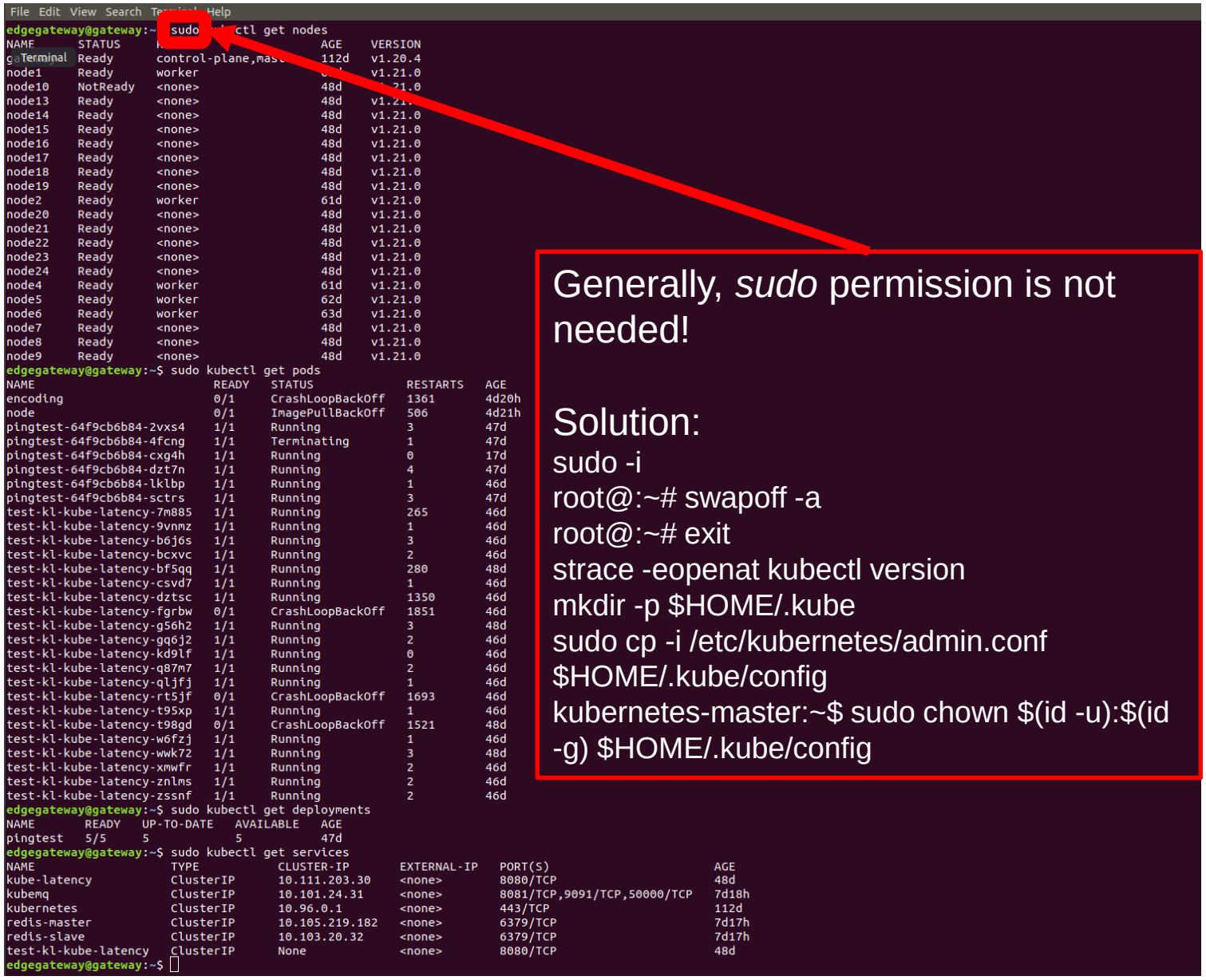
- Namespaces are intended for use in environments with many users spread across multiple teams, or projects.

- Namespaces are a way to divide cluster resources between multiple users (via resource quota).
  - ✓ https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

$ microk8s.

- ➢ kubectl get namespace # List the current namespaces in a cluster
- ➢ kubectl get all --all-namespaces # List all pods, services, deployments
- ➢ kubectl get pods --all-namespaces # List all pods in all namespaces with more details

Several commands of Kubernetes



Check the status

MicroK8s has a built-in command to display its status, by which you can check the list of available addons.

During installation, you can use the --wait-ready flag to wait for the Kubernetes services to get initialized:

- $ microk8s status --wait-ready

Deploy an app

To deploy apps and services, use the kubectl command. Try installing a demo app:

- $ microk8s kubectl create -f deployment.yaml

Run deployment if already created

- $ microk8s kubectl apply -f deployment.yaml

It may take a minute or two to install, but you can check the status:

- $ microk8s kubectl get pods

Starting and Stopping MicroK8s

MicroK8s will continue running until you decide to stop it. You can stop and start MicroK8s with these simple commands:

- $ microk8s stop

… will stop MicroK8s and its services. You can start it again by running:

- $ microk8s start

Bulletin-board

node-bulletin-board project is a simple bulletin board application, written in Node.js

https://github.com/dockersamples/node-bulletin-board

A Kubernetes YAML file of two objects

apiVersion
indicates the Kubernetes API that parses this object

kind
indicates the type of object

metadata
applies some names to your objects

spec
specifies all the parameters and configurations of your object.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bb-demo
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      bb: web
  template:
    metadata:
      labels:
        bb: web
    spec:
      containers:
      - name: bb-site
        image: bulletinboard:1.0
---
apiVersion: v1
kind: Service
metadata:
  name: bb-entrypoint
  namespace: default
spec:
  type: NodePort
  selector:
    bb: web
  ports:
  - port: 8080
    targetPort: 8080
    nodePort: 30001
```

A Kubernetes YAML file of two objects

Deployment
describes a scalable group of identical Pods. Here, Pod (under the template: key) has just one container based on your bulletinboard:1.0 image.

Service: NodePort type
routes traffic from port 30001 on your host to port 8080 of pods, and allows a customer to browse your bulletin board service.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bb-demo
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      bb: web
  template:
    metadata:
      labels:
        bb: web
    spec:
      containers:
      - name: bb-site
        image: bulletinboard:1.0
---
apiVersion: v1
kind: Service
metadata:
  name: bb-entrypoint
  namespace: default
spec:
  type: NodePort
  selector:
    bb: web
  ports:
  - port: 8080
    targetPort: 8080
    nodePort: 30001
```

Deploying the *bb* application

- kubectl apply -f bb.yaml

  deployment.apps/bb-demo created
  service/bb-entrypoint created

- kubectl get deployments

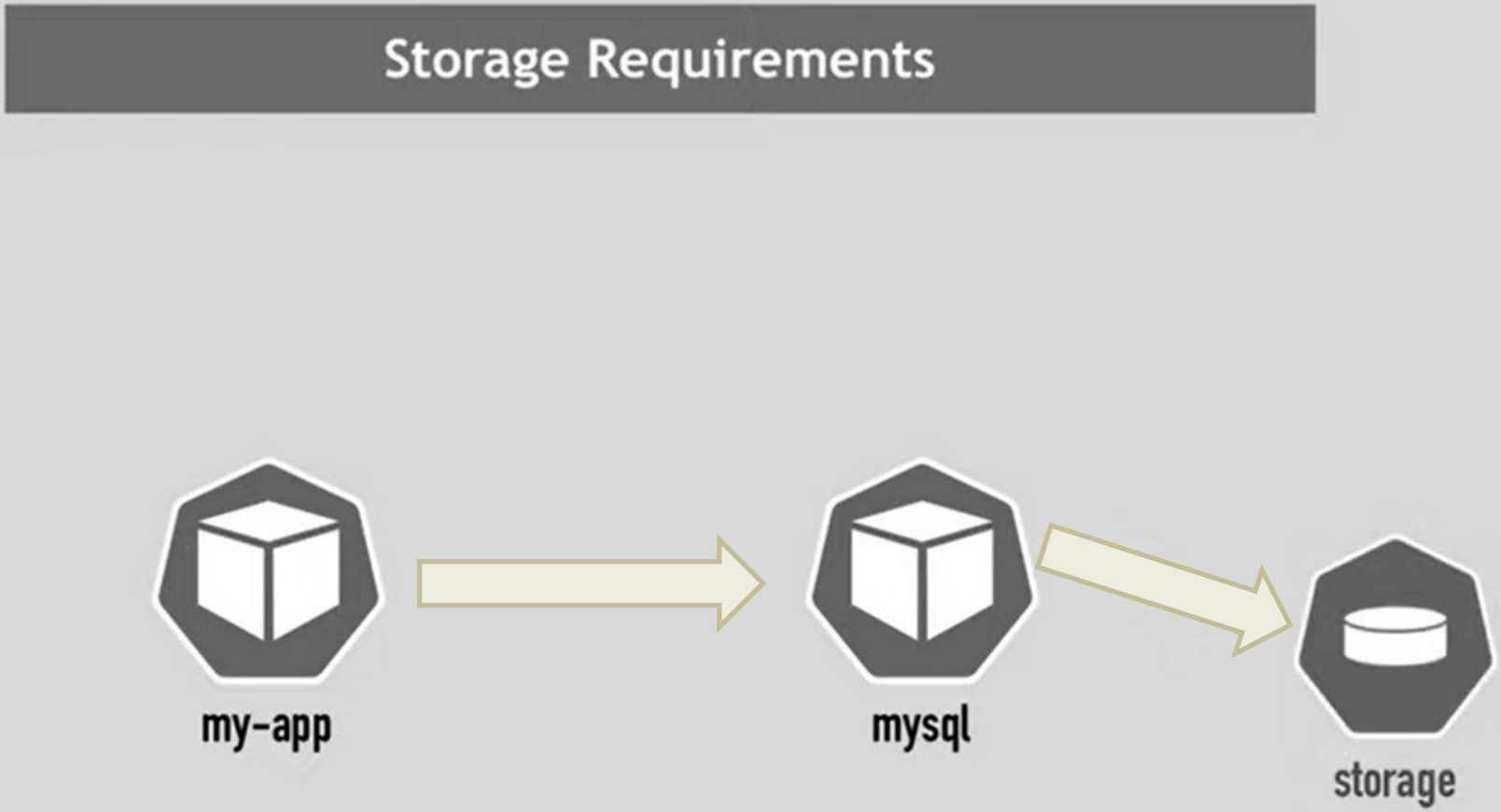
```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
bb-demo   1         1         1            1           48s
```

- kubectl get services

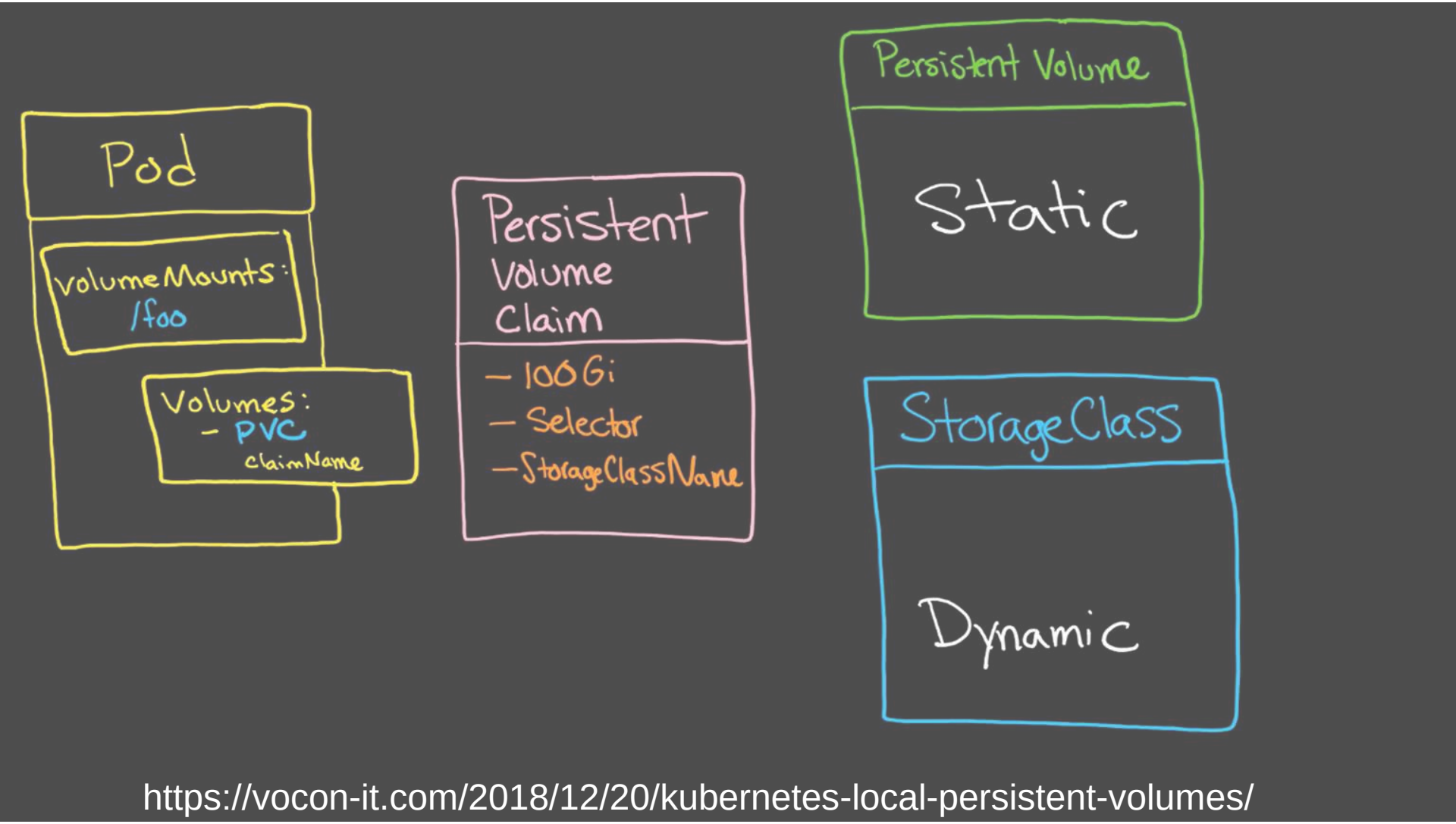
```
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
bb-entrypoint   NodePort    10.106.145.116  <none>        8080:30001/TCP   53s
kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP          138d
```

- Open a browser and visit your bulletin board at localhost:30001; you should see your bulletin board, the same as when we ran it as a stand-alone container in the previous step of this tutorial.

Persistent Storage



Persistent Storage



https://vocon-it.com/2018/12/20/kubernetes-local-persistent-volumes/

Prometheus Software

- Prometheus is a free software application used for event monitoring and alerting.
- It records real-time metrics in a time series database built using an HTTP pull model, with flexible queries and real-time alerting.
- It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if a condition is observed to be true.

Ref

- https://kubernetes.io/docs/tutorials/
- https://kompose.io/
- https://phoenixnap.com/kb/install-kubernetes-on-ubuntu
- https://minikube.sigs.k8s.io/docs/start/
- https://github.com/ubuntu/microk8s
- https://docs.docker.com/get-started/part2/
- https://docs.docker.com/get-started/kube-deploy/
- https://golang.org/doc/code.html