Università di Pisa

Dept. of Information Engineering

Course Wireless Networks - 2021/2022

Virtualization (LAB)

Alessandra Fais – PhD Student

email: alessandra.fais@phd.unipi.it
web page: for.unipi.it/alessandra_fais/

LAB organization

- ❑ **PART I (theoretical)**
  - ❑ Introduction to SDN, NFV, MEC * concepts
  - ❑ Cloud computing and service-based architectures

* SDN = Software Defined Networking,
NFV = Network Function Virtualization,
MEC = Multi-access Edge Computing

- ❑ **PART II**
  - ❑ OpenStack cloud computing platform
  - ❑ OpenStack and NFV
  - ❑ Live session: OpenStack platform of the DII CrossLab project

LAB organization

* VM = Virtual Machine

- **PART III**
  - Virtualization overview and different approaches
    - VMs* on hypervisors, containers, alternative solutions

- **PART IV**
  - Containers -> Docker
  - Orchestrators -> Kubernetes
  - Hands-on session: Docker, docker-compose, Kubernetes

PART IV

Outline of Part IV

1) Containers characteristics
2) Docker
    - Objects
    - Architecture
    - Deployment modes
        - Single host VS Cluster
3) Kubernetes

Outline of Part IV

Hands-on session:

Docker

- ○ Installation
- ○ Execution flow
- ○ Commands
- ○ Dockerfile
- ○ Docker Compose
- ○ Persistent storage management (volumes)

Kubernetes

- ○ Installation
- ○ Complete application

Cluster mode:
Docker and Kubernetes

Working on clusters with an orchestrator

- Many services cooperating together to implement a single application
  - Easy to scale and deploy single services (app components)
  - Model typically referred to as **microservice-based architecture**
- Orchestrators help managing these components running in containers
  - Run applications in a reliable way
  - Take advantage of existing Docker workloads and run them at scale

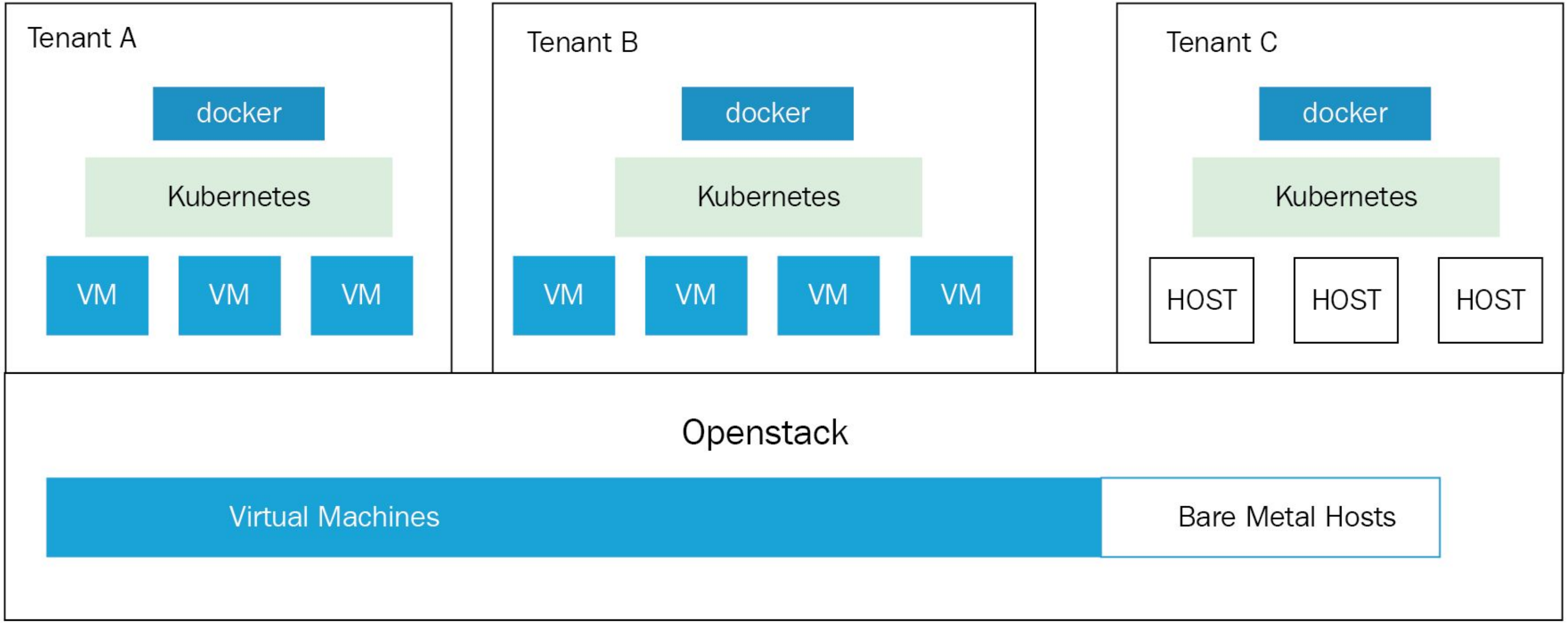
Kubernetes

Kubernetes: overview

- **Main idea**
    - deploy containerized applications to a cluster without tying them specifically to individual machines

- Distribution and scheduling of application containers is automated across a cluster in a more efficient way

Kubernetes on OpenStack (example)

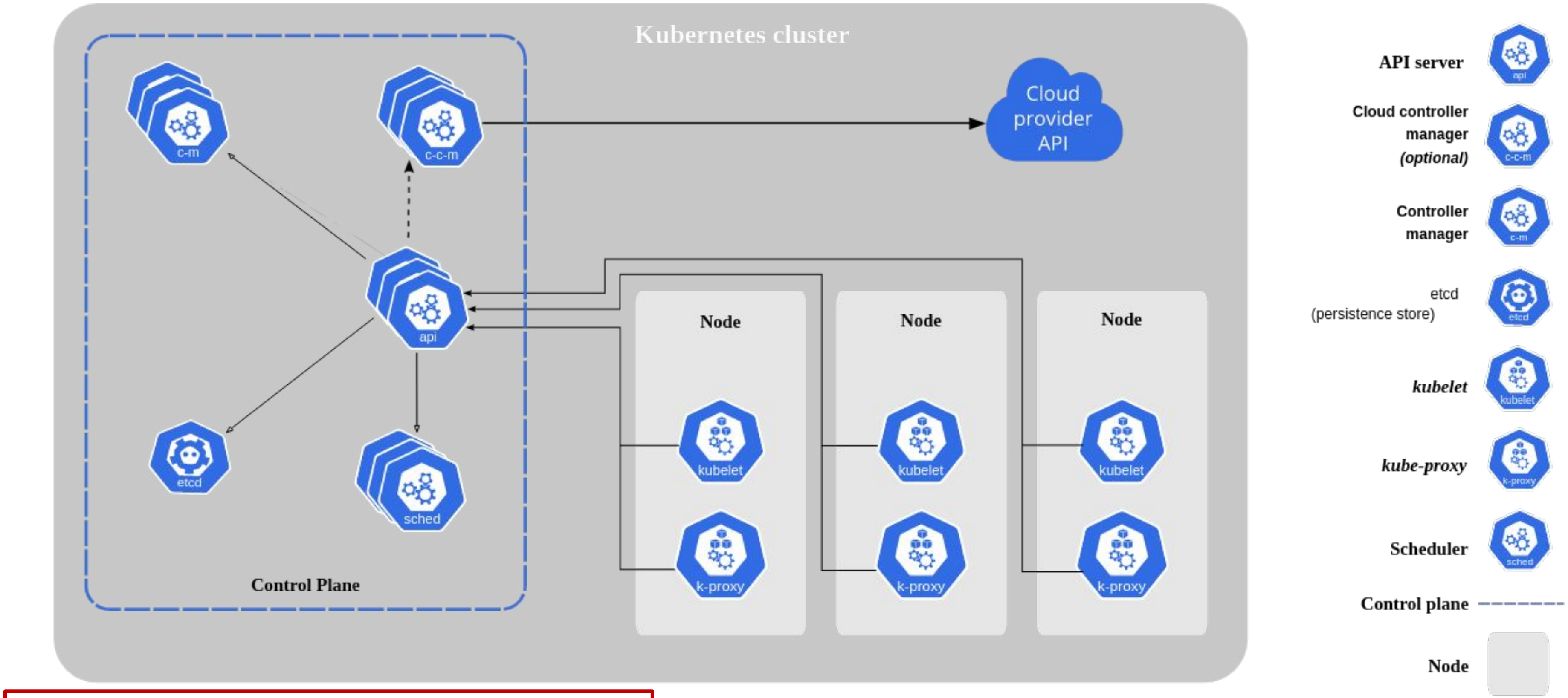


https://www.oreilly.com/library/view/openstack-for-architects/9781788624510/c86451e6-56cb-4949-8c7c-367e5ba01444.xhtml

Kubernetes: architectural components

- **Kubernetes cluster service (control plane)**
  - Coordinates the cluster and exposes an API
    - accepts a **yaml** configuration file describing the desired app management on the infrastructure
  - Deploys app configuration on the infrastructure
    - checks that the **yaml** configuration is running correctly at any point in time on the workers
- **Workers (nodes: VMs or physical machines)**
  - Basically container hosts (they run applications)
  - Communicate with the cluster service through the API

Kubernetes: architectural components

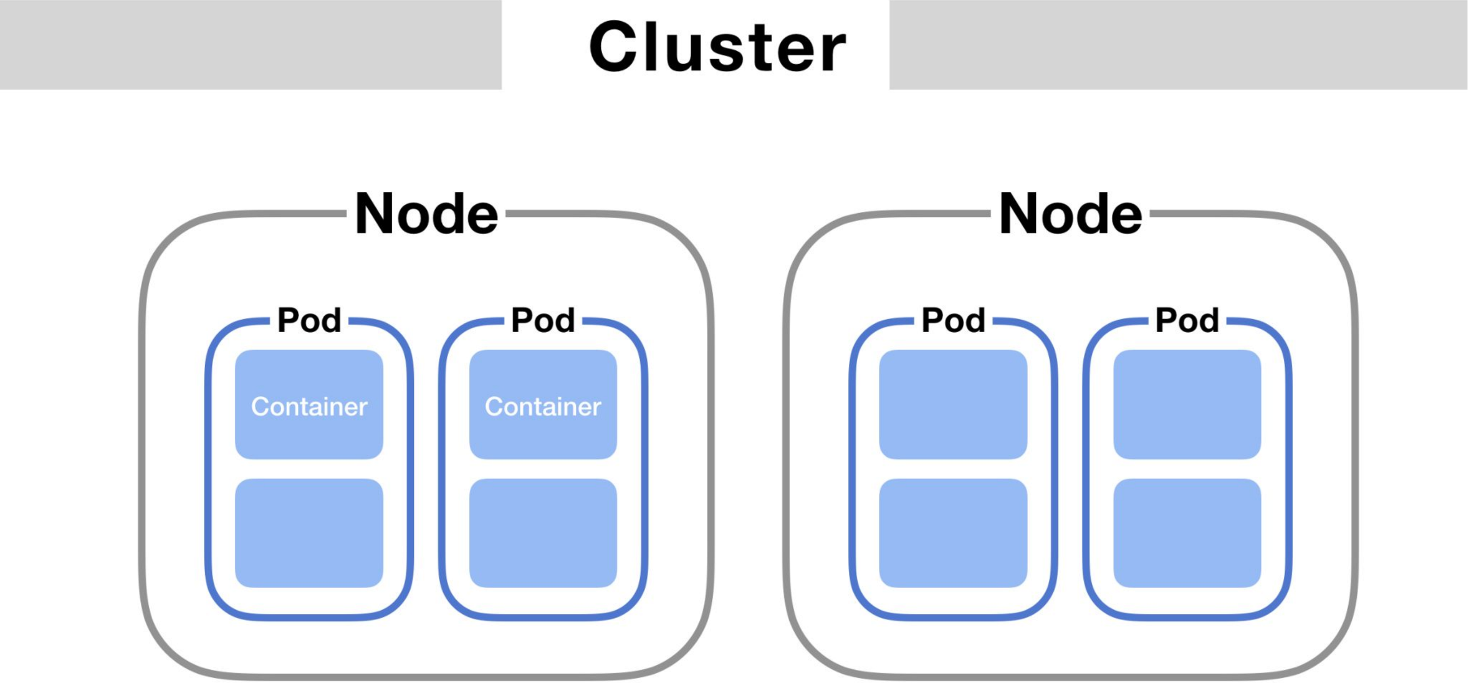


https://kubernetes.io/docs/concepts/overview/components/

Kubernetes: architectural components

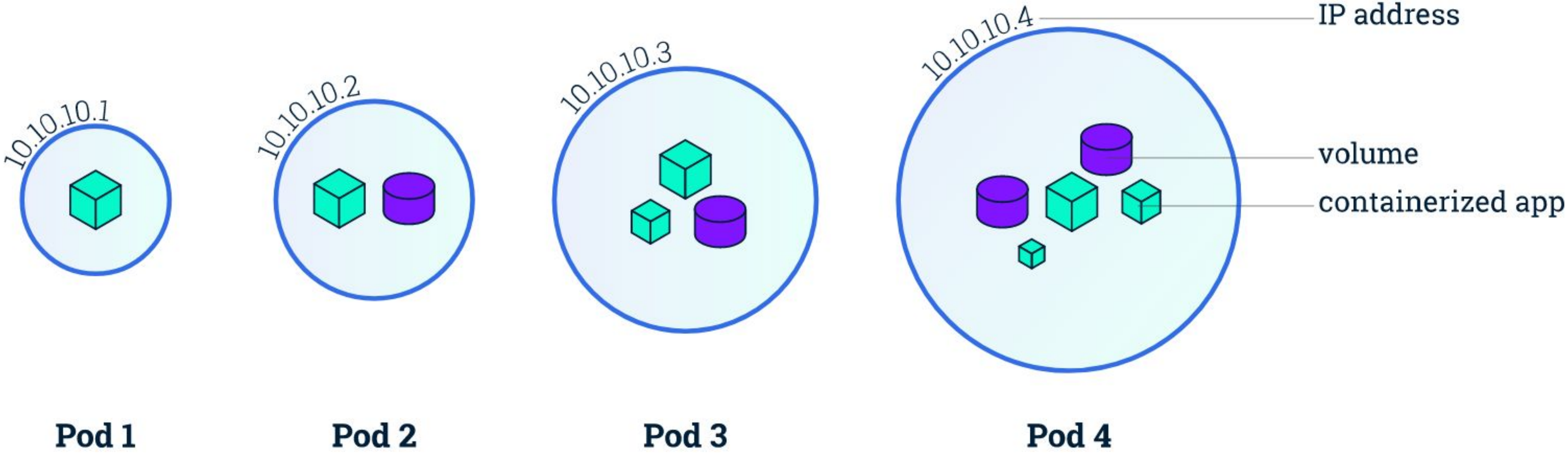
- **yaml** file contains configuration information useful for the application deployment on the infrastructure
- **Pod**
  - Run on the nodes/workers
    - scheduled by the **master** across the nodes in the cluster
  - Main entity and smallest unit of deployment
    - can run one or more containers (from images)
  - **Replicas**: how many of these pods I want to run?

Kubernetes: architecture overview



https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-networking-guide-beginners.html

Kubernetes: architecture overview



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

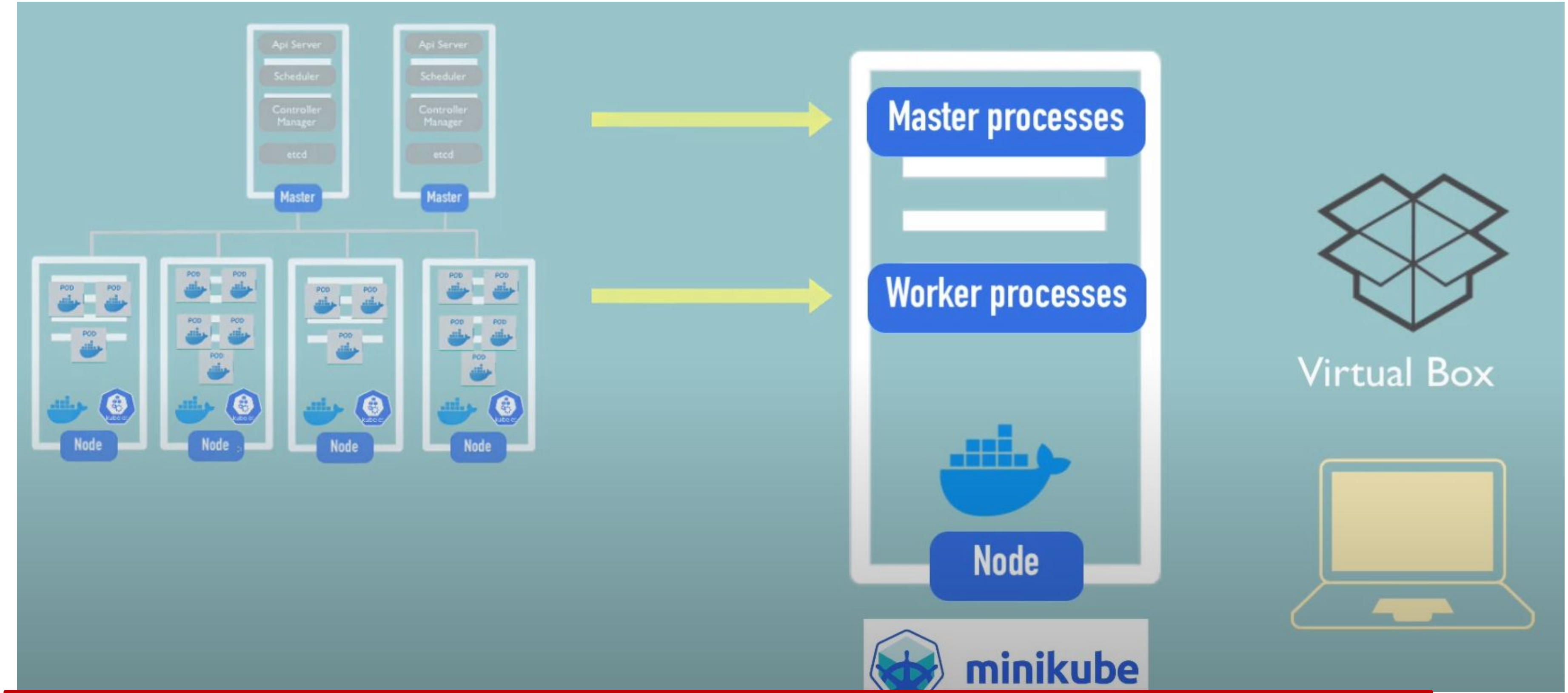
Hands-on with Kubernetes

minikube, kubectl and Minikube cluster

Local cluster setup with minikube

- **minikube environment**
  - One-node Kubernetes cluster where master and worker processes run on one node
    - the same machine or VM
  - On our laptop it can run on an hypervisor
    - E.g. VirtualBox
  - Docker runtime is pre-installed
  - Useful for testing purposes!

Test: local cluster setup with minikube

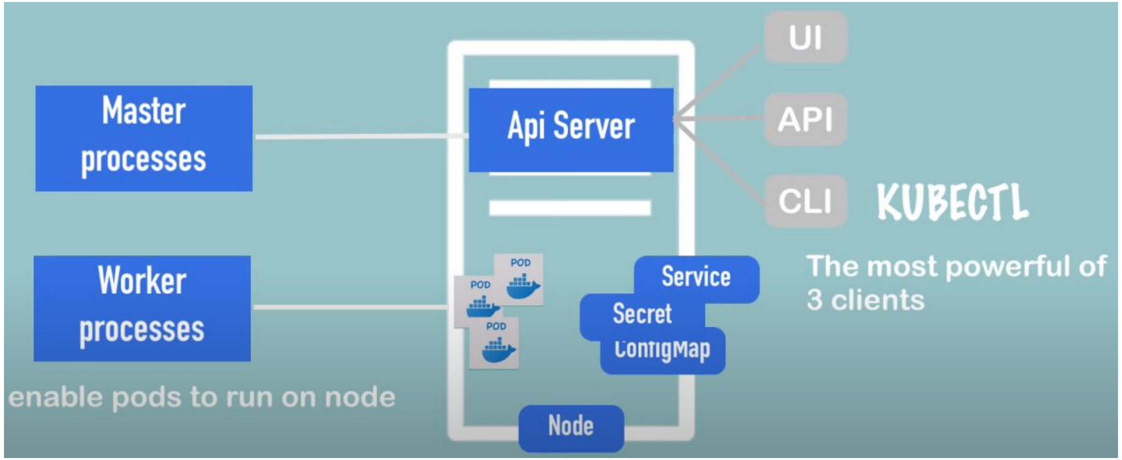


TechWorld with Nana - Minikube and Kubectl explained | Setup for Beginners: https://www.youtube.com/watch?v=E2pP1MOfo3g

Interaction with the cluster with kubectl

- A way to interact with **any kind of K8 cluster** (local/hybrid/cloud)
  - In our case, the local cluster created with minikube
    - Create pods and/or other K8 components

- The API Server is the main entrypoint in the K8 cluster
- Interaction with cluster
  - UI - dashboard
  - API
  - CLI with kubectl

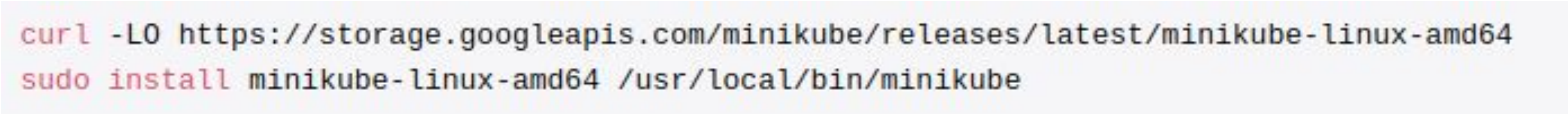


TechWorld with Nana - Minikube and Kubectl explained | Setup for Beginners: https://www.youtube.com/watch?v=E2pP1MOfo3g

Install minikube and kubectl

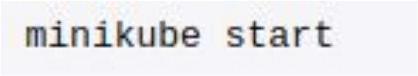
Official Documentation:
https://minikube.sigs.k8s.io/docs/start/

- Installation of **minikube**



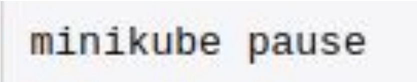
```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

- Start the cluster

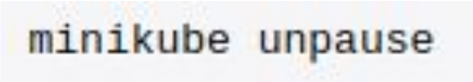


```
minikube start
```

- Pause/Un-pause Kubernetes without impacting deployed applications

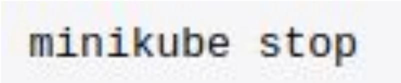


```
minikube pause
```



```
minikube unpause
```

- Stop the cluster

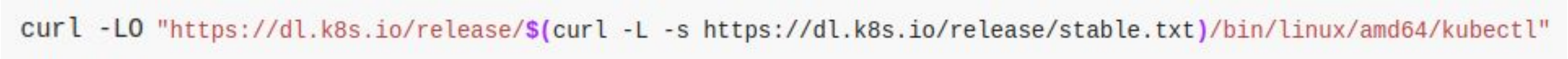


```
minikube stop
```

Install minikube and kubectl

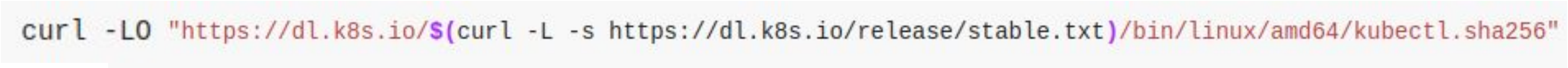
Official Documentation:
https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/

- Download the latest release of **kubectl** for Linux

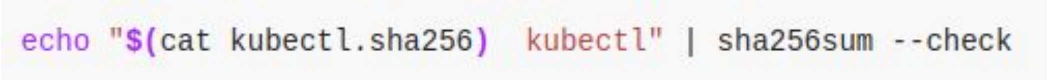


```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

- Validate the binary (first download the checksum file)

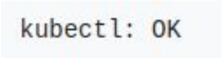


```
curl -LO "https://dl.k8s.io/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
```



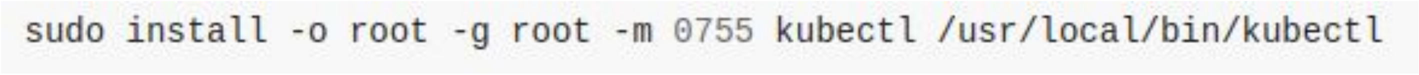
```
echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
```

  - If valid the output is:



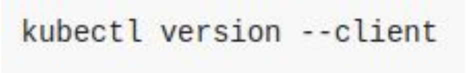
```
kubectl: OK
```

- Install kubectl on Linux



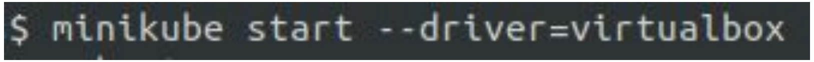
```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

  - Check the installed version:

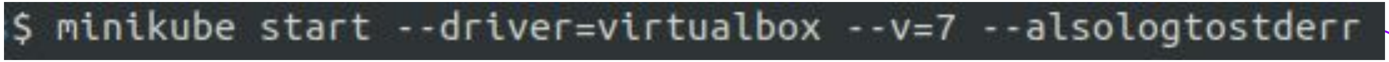


```
kubectl version --client
```

Manage and configure the cluster

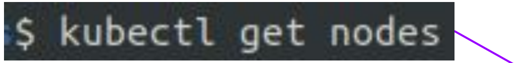
- 

```
$ minikube start --driver=virtualbox
```

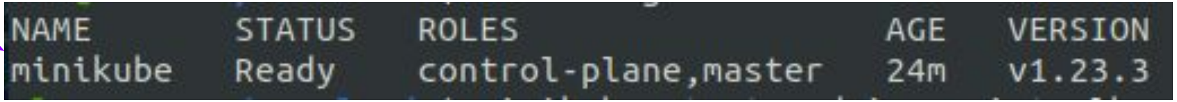


```
$ minikube start --driver=virtualbox --v=7 --alsologtostderr
```

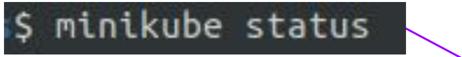
Debug mode

- 

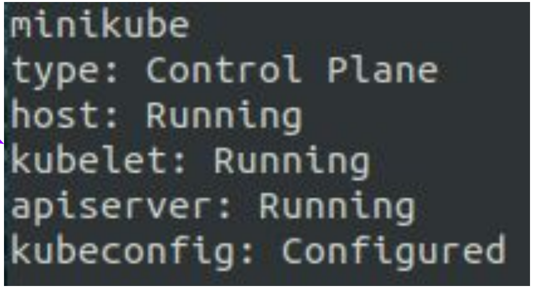
```
$ kubectl get nodes
```



```
NAME       STATUS   ROLES                  AGE   VERSION
minikube   Ready    control-plane,master   24m   v1.23.3
```

- 

```
$ minikube status
```

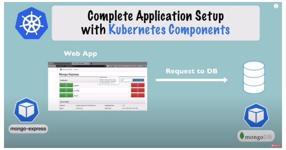


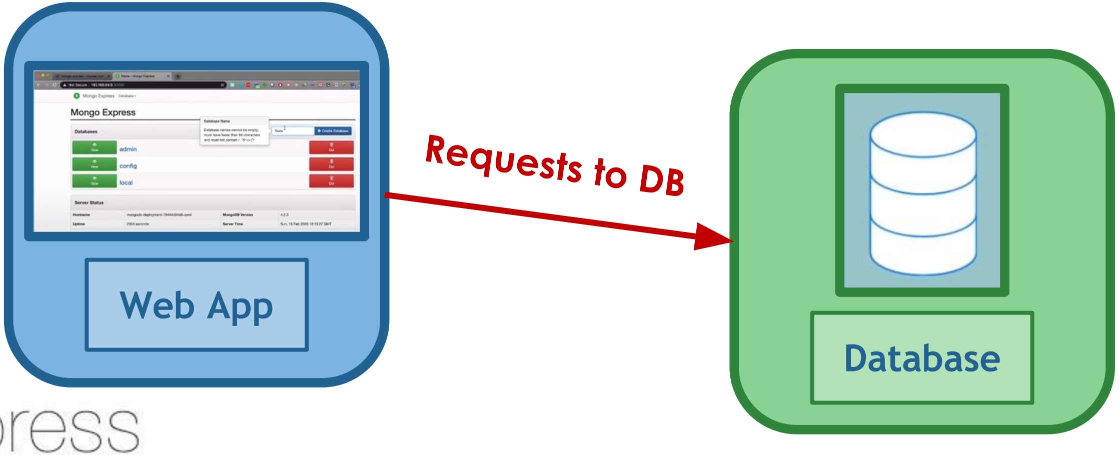
```
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

First example (local cluster on single host)

A complete application deployment using Kubernetes

Hands-on with Kubernetes: application



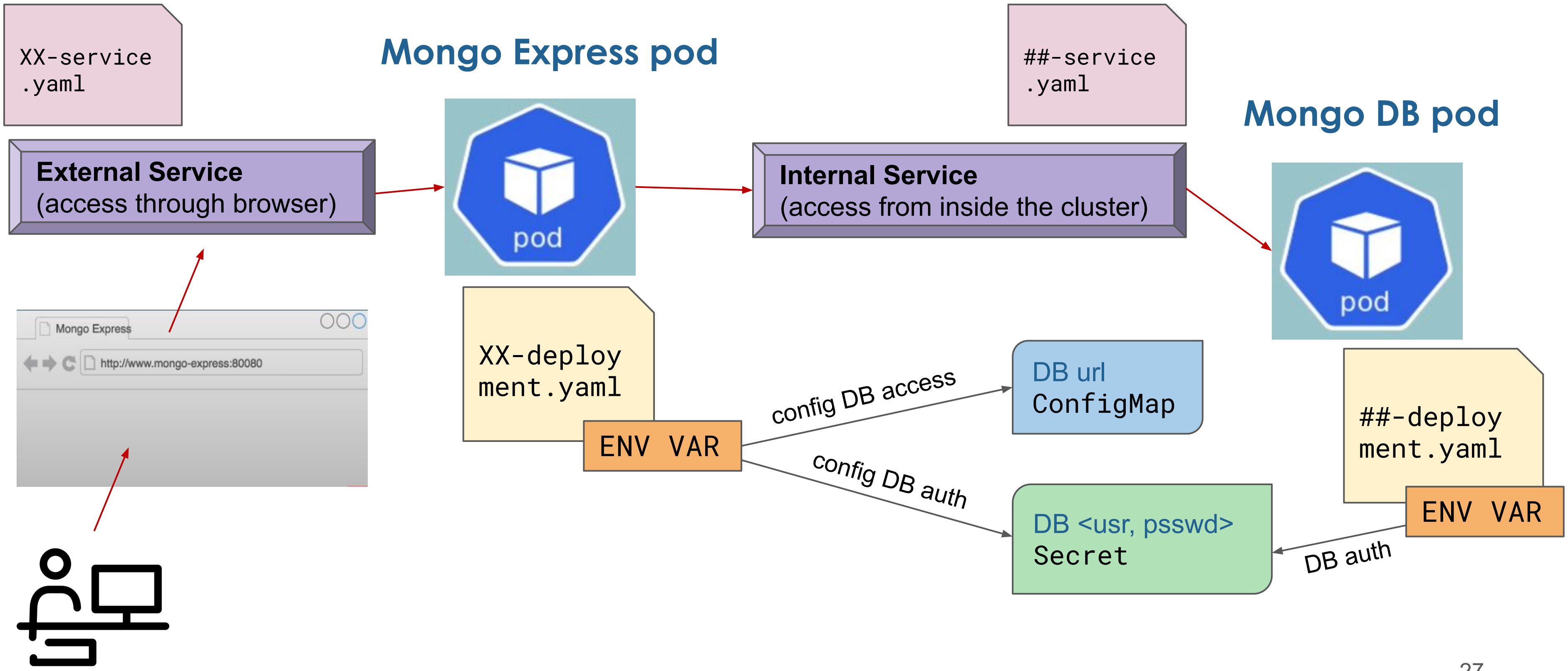


Interactive lightweight Web-Based Administrative Tool to effectively manage MongoDB Databases

A MongoDB Database

TechWorld with Nana - Complete Application Deployment using Kubernetes Components: https://youtu.be/EQNO_kM96Mo

Create the database deployment



XX-service .yaml

External Service
(access through browser)

Mongo Express

http://www.mongo-express:80080

Mongo Express pod

pod

XX-deployment.yaml

ENV VAR

##-service .yaml

Internal Service
(access from inside the cluster)

Mongo DB pod

pod

config DB access

DB url
ConfigMap

config DB auth

DB <usr, psswd>
Secret

##-deployment.yaml

ENV VAR

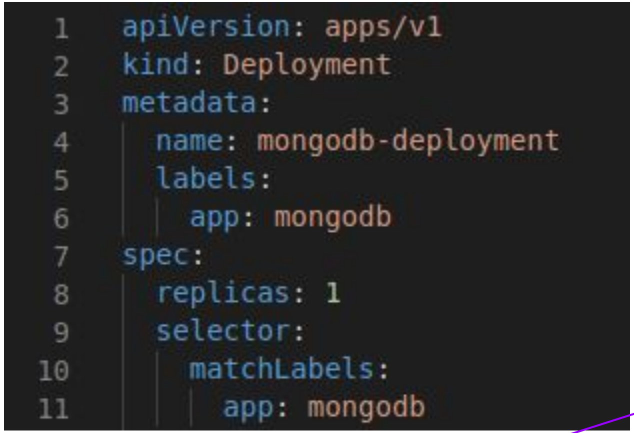
DB auth

Create the database deployment

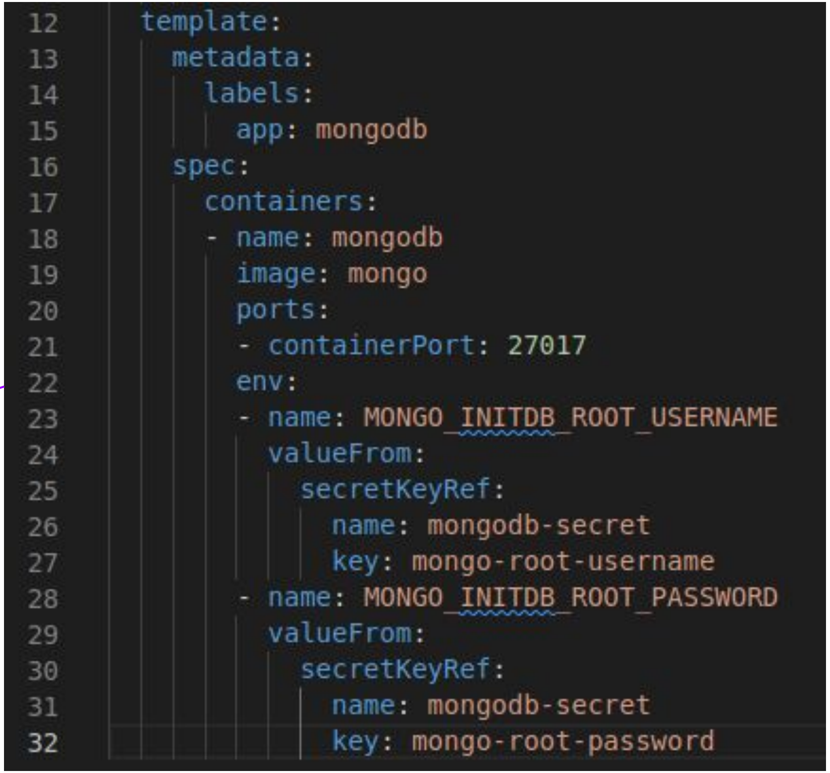
- Create the **Mongo DB pod**
  - Create internal service to talk with the DB
  - No external requests can reach the pod, only requests from components in the same cluster are allowed
- Create the **Mongo Express pod** that needs
  - Database URL of Mongo DB to connect to it
    - Create a ConfigMap for the URL (cluster-shared config object)
  - Credentials of the Mongo DB (username, password) to authenticate to it
    - Create a Secret for the credentials (cluster-shared config object)
  - All is passed in *-deployment.yaml configuration files through ENV variables which reference the needed information
  - Create external service to access Mongo Express through browser

Mongo DB

Config: mongodb-deployment.yaml



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
```

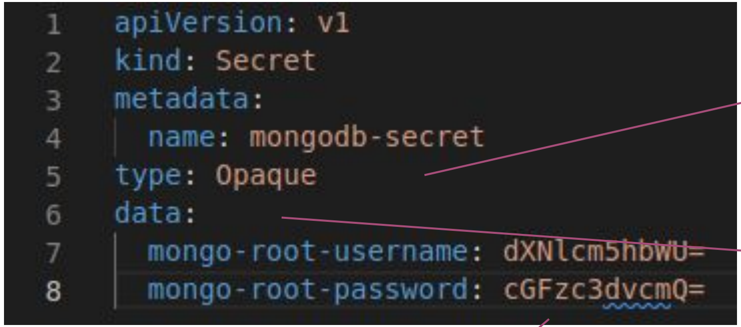


```yaml
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo
        ports:
        - containerPort: 27017
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-username
        - name: MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-password
```

Defines blueprints for the pods that this deployment will create

Docker image configuration for `mongodb` container: https://hub.docker.com/_/mongo

Secret: DB username and password information

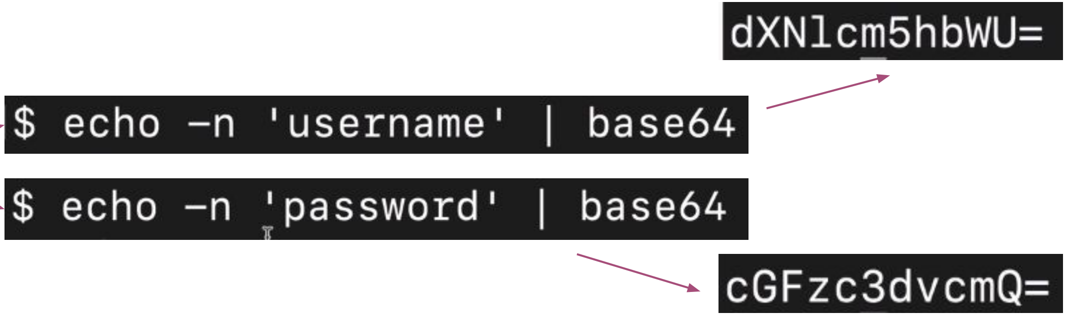


```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
type: Opaque
data:
  mongo-root-username: dXNlcm5hbWU=
  mongo-root-password: cGFzc3dvcmQ=
```

`Opaque` type is the default one for storing key-value arbitrary pairs

`data` is the label for the actual key-value pair content

`Values` in key-value pairs are not in plain text but they must be **base64 encoded**



```
dXNlcm5hbWU=
```

```
$ echo -n 'username' | base64
```

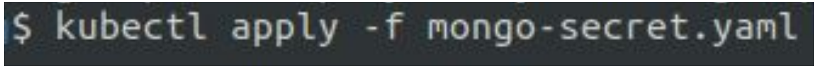
```
$ echo -n 'password' | base64
```

```
cGFzc3dvcmQ=
```

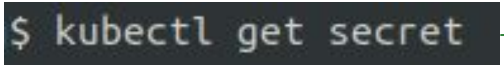
Apply configuration

Official Documentation:
https://docs.docker.com/get-started/kube-deploy/

- Create `Secret` configuration

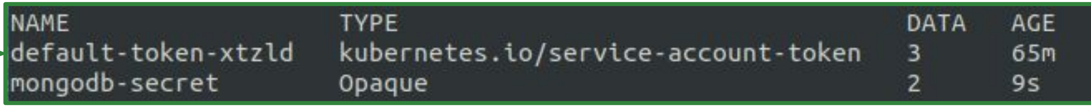


```
$ kubectl apply -f mongo-secret.yaml
```



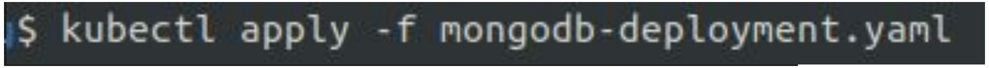
```
$ kubectl get secret
```

output

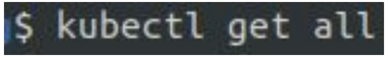


```
NAME                  TYPE                                  DATA   AGE
default-token-xtzld   kubernetes.io/service-account-token   3      65m
mongodb-secret        Opaque                                2      9s
```

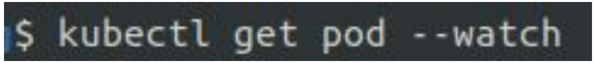
- From now on it can be referenced with no errors from inside the deployment file `mongodb-deployment.yaml`



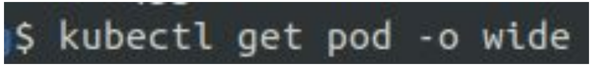
```
$ kubectl apply -f mongodb-deployment.yaml
```



```
$ kubectl get all
```

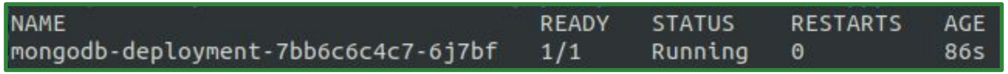


```
$ kubectl get pod --watch
```

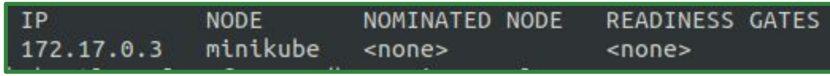


```
$ kubectl get pod -o wide
```

output



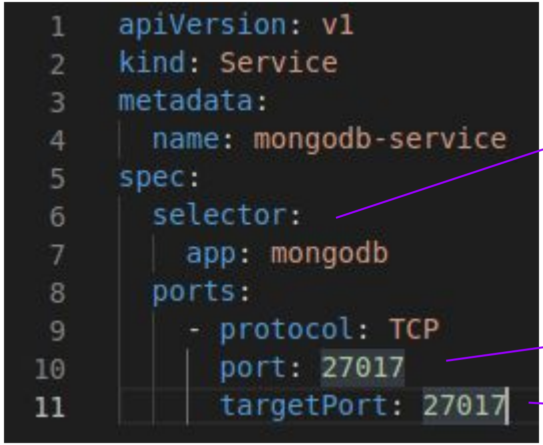
```
NAME                                  READY   STATUS    RESTARTS   AGE
mongodb-deployment-7bb6c6c4c7-6j7bf   1/1     Running   0          86s
```



```
IP           NODE       NOMINATED NODE   READINESS GATES
172.17.0.3   minikube   <none>           <none>
```

- Now the internal service must be created so that other components in the K8 cluster can talk to this one

Config & create service: mongodb-service.yaml



```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

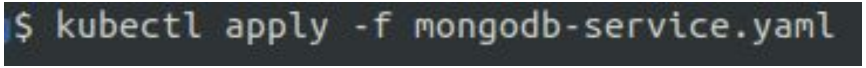
`selector` defines to which pod this service needs to connect

-> it matches the `label` of the pod (deployment)

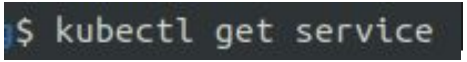
`port` defines the exposed service port

`targetPort` matches the `containerPort` pod port (used by the mongodb container)

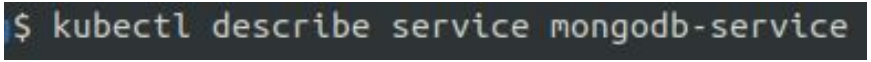
- Create internal service (check service `endpoint:port` with the one of the pod)



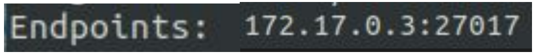
```
$ kubectl apply -f mongodb-service.yaml
```



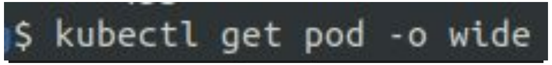
```
$ kubectl get service
```



```
$ kubectl describe service mongodb-service
```



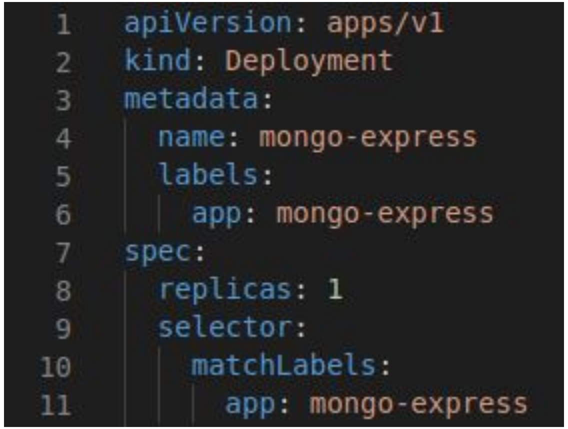
```
Endpoints:  172.17.0.3:27017
```



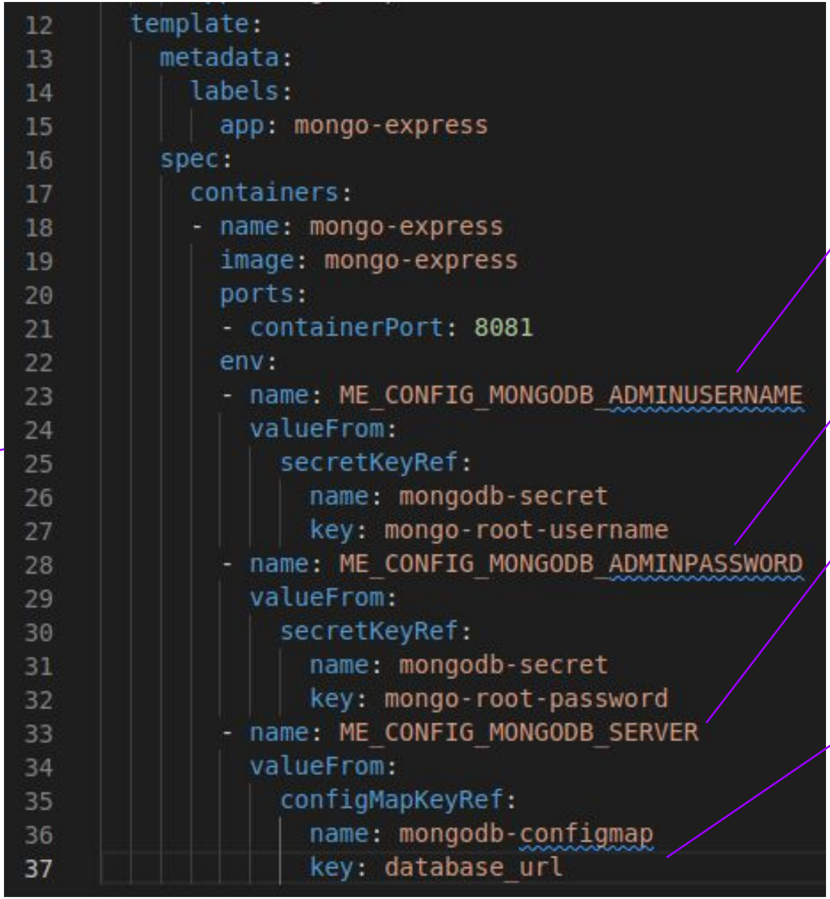
```
$ kubectl get pod -o wide
```

Mongo Express

Config: mongoexpress-deployment.yaml



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-express
  labels:
    app: mongo-express
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo-express
```



```yaml
  template:
    metadata:
      labels:
        app: mongo-express
    spec:
      containers:
      - name: mongo-express
        image: mongo-express
        ports:
        - containerPort: 8081
        env:
        - name: ME_CONFIG_MONGODB_ADMINUSERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-username
        - name: ME_CONFIG_MONGODB_ADMINPASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-password
        - name: ME_CONFIG_MONGODB_SERVER
          valueFrom:
            configMapKeyRef:
              name: mongodb-configmap
              key: database_url
```

Defines blueprints for the pods that this deployment will create

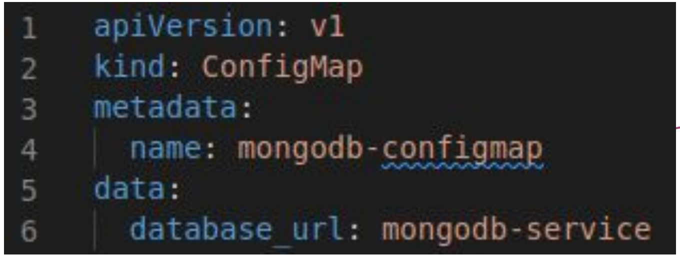
Credentials are the ones stored in the `Secret`

This is the ENV VAR for the DB server URL

-> refers the config key-value pair in the `ConfigMap`

Docker image configuration for `mongo-express` container: https://hub.docker.com/_/mongo-express

ConfigMap: contains DB server URL



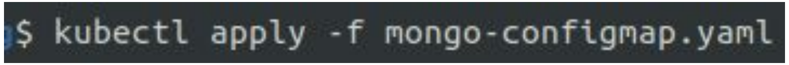
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-configmap
data:
  database_url: mongodb-service
```

It's useful to create a ConfigMap to have a **centralized configuration** that can be shared among many components in the cluster

- centralized config
- if the config changes, you need to update only this element

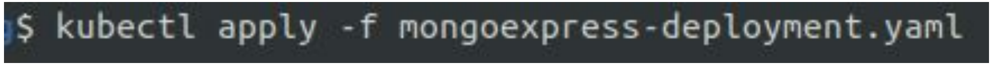
The name of the mongodb Service

- Create `ConfigMap` configuration in the cluster

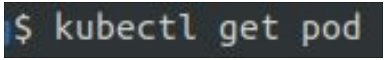


```
$ kubectl apply -f mongo-configmap.yaml
```

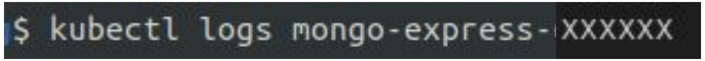
- From now on it can be referenced with no errors from inside the deployment file `mongoexpress-deployment.yaml`



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

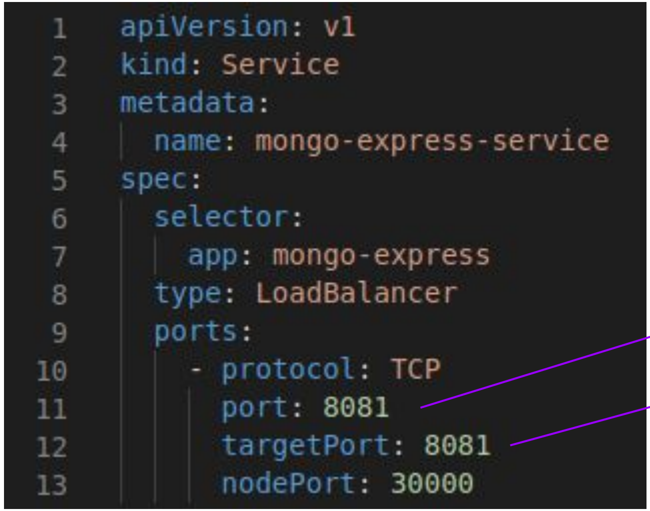


```
$ kubectl get pod
```



```
$ kubectl logs mongo-express-XXXXXX
```

Config & create service: mongoexpress-service.yaml



```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongo-express-service
spec:
  selector:
    app: mongo-express
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8081
      targetPort: 8081
      nodePort: 30000
```

Needed to access mongo-express from the outside of the cluster, via browser

`port` defines the exposed service port

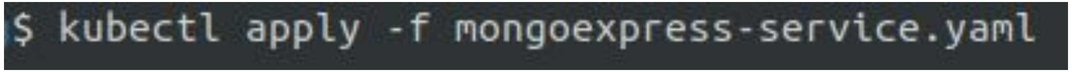
`targetPort` matches the `containerPort` pod port (where the mongo-express container is listening)

- Make this an external service
  - need `type` in `spec` section (`LoadBalancer` accepts external requests by assigning the service an external IP address)
  - need a third port called `nodePort` which is the port where the external IP address will be open (you must select port in range `30000-32767`)

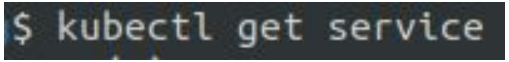
Apply configuration

Official Documentation:
https://docs.docker.com/get-started/kube-deploy/

- Create external service
  - check service `endpoint:port` with the one of the pod

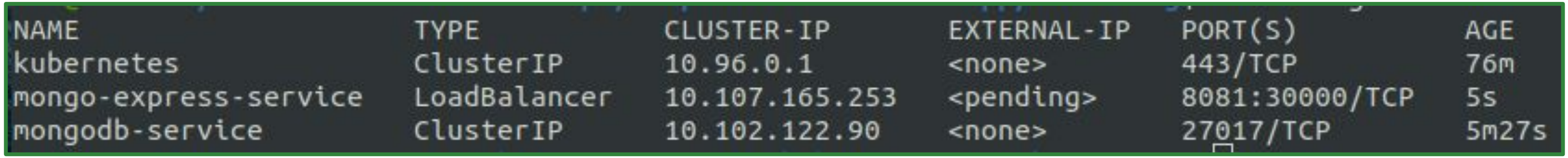


```
$ kubectl apply -f mongoexpress-service.yaml
```

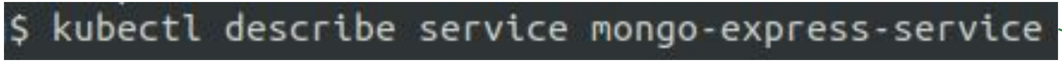


```
$ kubectl get service
```

output

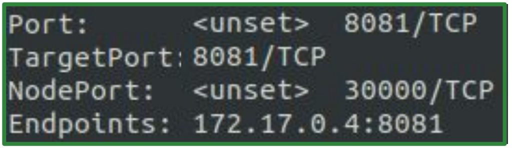


```
NAME                    TYPE          CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes              ClusterIP     10.96.0.1        <none>        443/TCP          76m
mongo-express-service   LoadBalancer  10.107.165.253   <pending>     8081:30000/TCP   5s
mongodb-service         ClusterIP     10.102.122.90    <none>        27017/TCP        5m27s
```

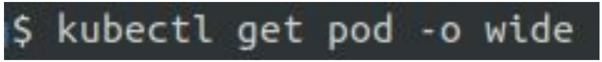


```
$ kubectl describe service mongo-express-service
```

output

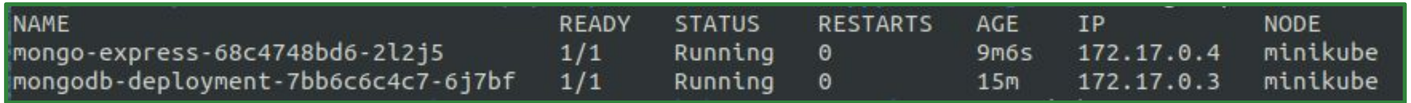


```
Port:       <unset>  8081/TCP
TargetPort: 8081/TCP
NodePort:   <unset>  30000/TCP
Endpoints:  172.17.0.4:8081
```



```
$ kubectl get pod -o wide
```

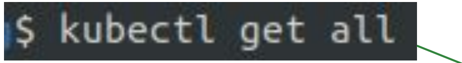
output



```
NAME                                  READY   STATUS    RESTARTS   AGE    IP           NODE
mongo-express-68c4748bd6-2l2j5        1/1     Running   0          9m6s   172.17.0.4   minikube
mongodb-deployment-7bb6c6c4c7-6j7bf   1/1     Running   0          15m    172.17.0.3   minikube
```

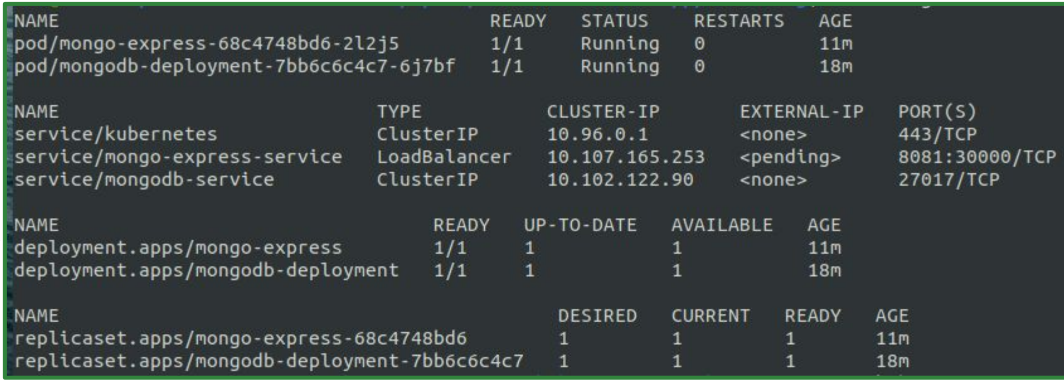
Apply configuration (last steps)

Official Documentation:
https://docs.docker.com/get-started/kube-deploy/



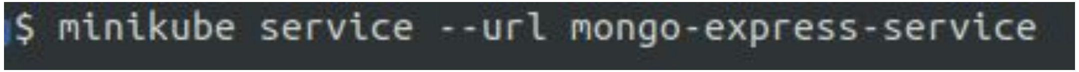
```
$ kubectl get all
```

output

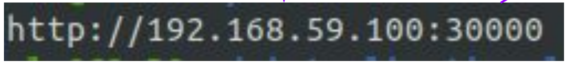


```
NAME                                       READY   STATUS    RESTARTS   AGE
pod/mongo-express-68c4748bd6-2l2j5         1/1     Running   0          11m
pod/mongodb-deployment-7bb6c6c4c7-6j7bf    1/1     Running   0          18m

NAME                            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
service/kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP
service/mongo-express-service   LoadBalancer   10.107.165.253   <pending>     8081:30000/TCP
service/mongodb-service         ClusterIP      10.102.122.90    <none>        27017/TCP

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongo-express        1/1     1            1           11m
deployment.apps/mongodb-deployment   1/1     1            1           18m

NAME                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/mongo-express-68c4748bd6        1         1         1       11m
replicaset.apps/mongodb-deployment-7bb6c6c4c7   1         1         1       18m
```

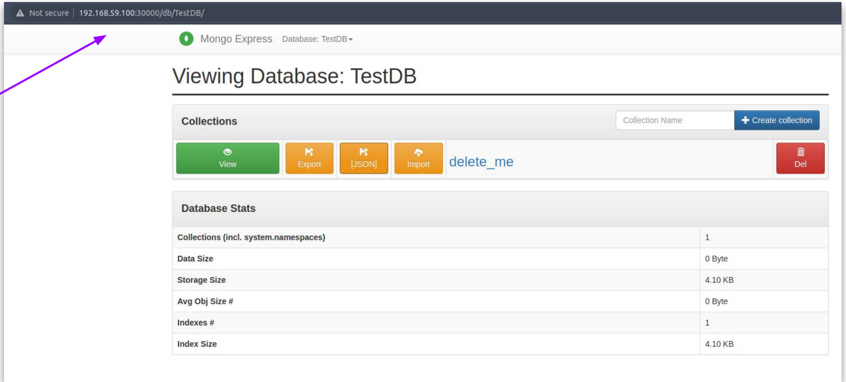
- Assign an external IP address



```
$ minikube service --url mongo-express-service
```



```
http://192.168.59.100:30000
```



Viewing Database: TestDB

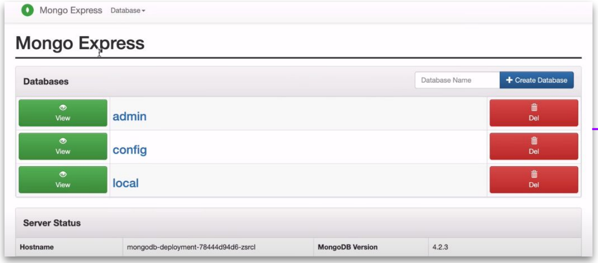
Collections

delete_me

Database Stats

Collections (incl. system.namespaces)	1
Data Size	0 Byte
Storage Size	4.10 KB
Avg Obj Size #	0 Byte
Indexes #	1
Index Size	4.10 KB

Taking incoming requests…



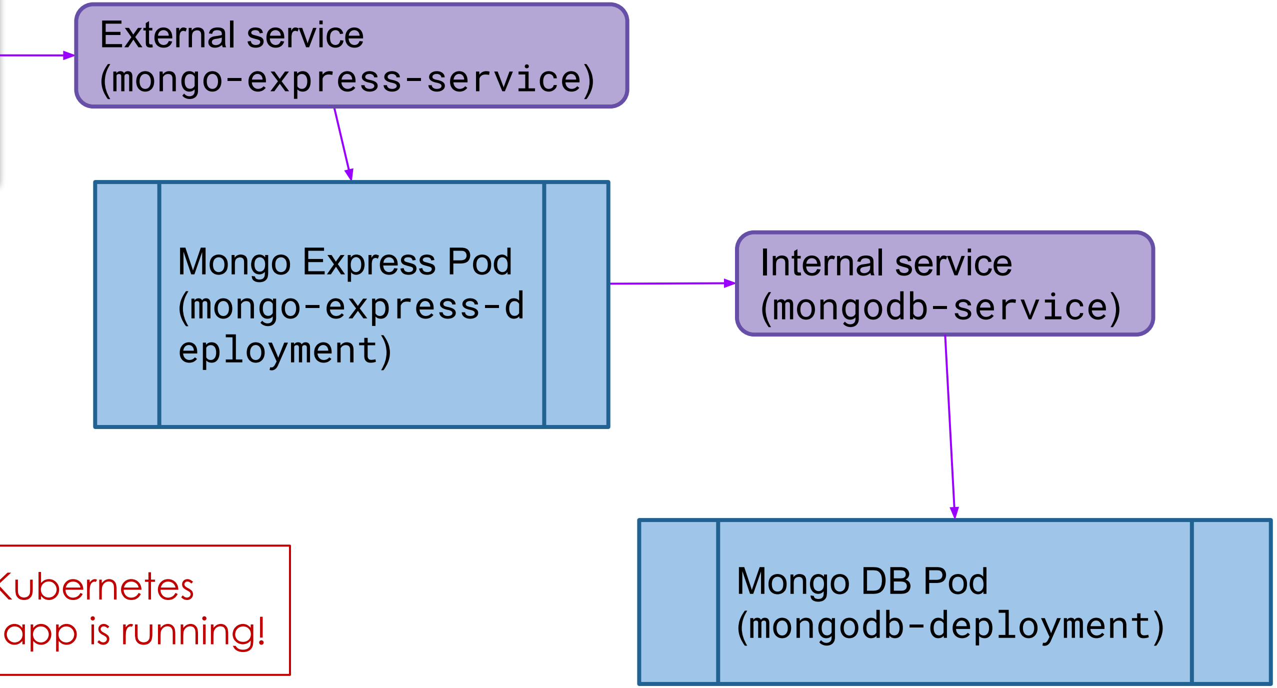
Mongo Express

Databases

View	admin	Del
View	config	Del
View	local	Del

Server Status

Hostname	mongodb-deployment-78444d94d6-zsrcl	**MongoDB Version**	4.2.3



External service
(`mongo-express-service`)

Mongo Express Pod
(`mongo-express-deployment`)

Internal service
(`mongodb-service`)

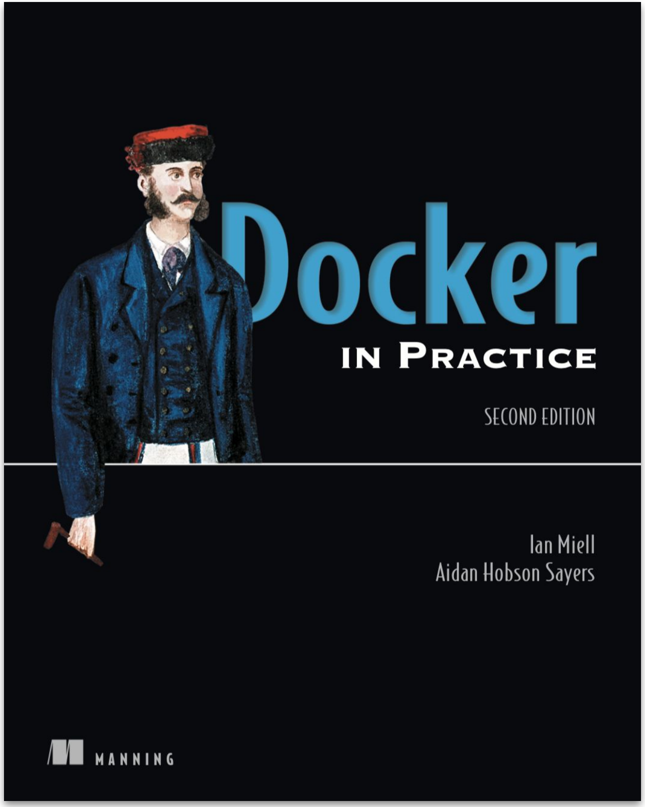
Mongo DB Pod
(`mongodb-deployment`)

Our application setup in the Kubernetes cluster is completed and the app is running!

TechWorld with Nana - Complete Application Deployment using Kubernetes Components: https://youtu.be/EQNO_kM96Mo

Useful references

Useful reference book



Link

Link

Useful references on Kubernetes

- Starting with clusters management using Kubernetes
  - https://docs.docker.com/get-started/kube-deploy/
- Learn Kubernetes basics: using Minikube to create a Cluster
  - https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/
- Learn Kubernetes basics: small tutorial
  - https://kubernetes.io/docs/tutorials/hello-minikube/
- Learn Kubernetes basics: Pods and Nodes
  - https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/
- Kubernetes guide for beginners
  - https://matthewpalmer.net/kubernetes-app-developer/articles/kubernetes-networking-guide-beginners.html
- TechWorld with Nana: Kubernetes Crash Course for Absolute Beginners
  - https://youtu.be/s_o8dwzRlu4
- TechWorld with Nana: Kubernetes explained in 15 minutes
  - https://youtu.be/VnvRFRk_51k
- TechWorld with Nana: Kubernetes YAML files explained in 15 minutes
  - https://youtu.be/qmDzcu5uY1I