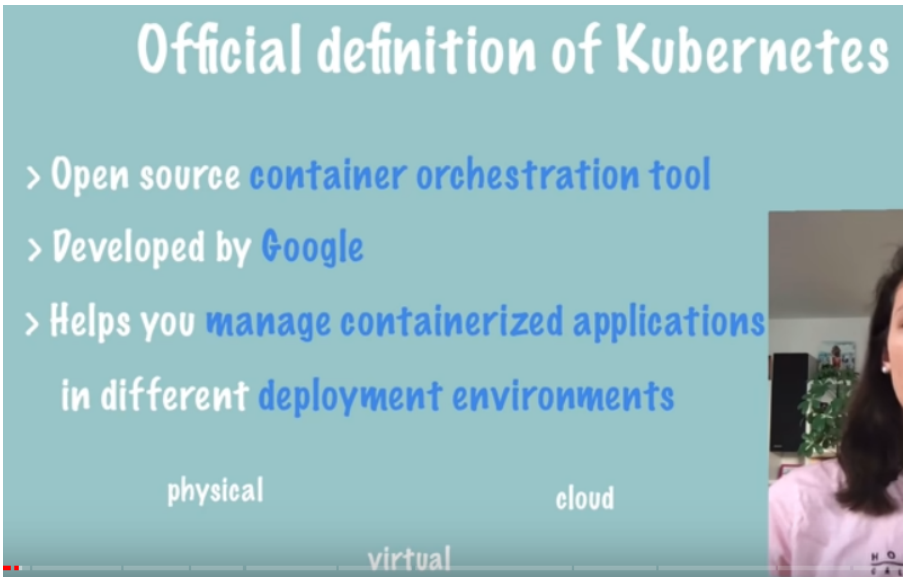
Kubernetes Tutorial for Beginners

Code: D:\workspace\docker_kubernetes\Kubernetes_Tutorial_for_Beginners

1. What it Kubernetes



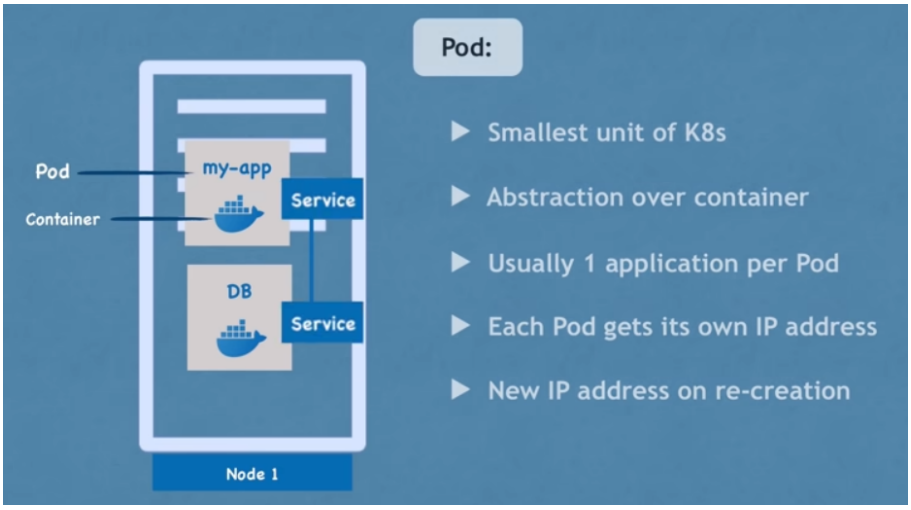
Trend from Monolith to Microservices increased usage of containers, which demands a proper way to manage those hundreds of containers.

1.1 What features do orchestration tools offer?

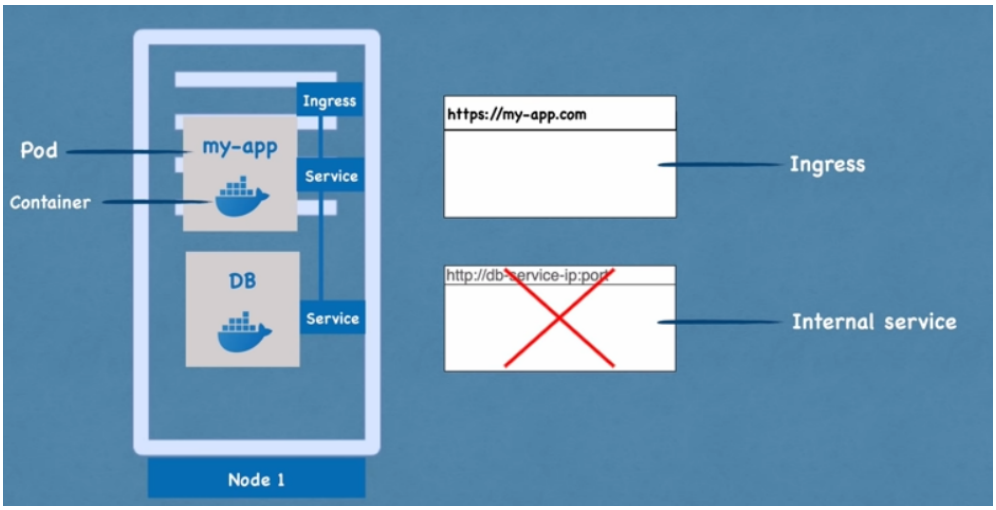
- High Availability or no downtime
- Scalability or high performance
- Disaster recovery - backup and restore

2. K8s Components

2.1 Node and Pod



2.2 Service and Ingress



Instead of Service,, the request goes to Ingress, and Ingress forwards to Service

2.3 ConfigMap and Secret

Pods communicate with each other using service.

Problem to resolve:

For example, in the above example, my-app needs to access DB service using url "mongo-db". The url us usually configured inside my-app using such as properties file. If the url changes, we need to change the url, rebuild and deploy the application.

The **solution** is the Kubernetes' component "**ConfigMap**" - External configuration of your application.

Don't put credentials in ConfigMap.

Solution for storing **credentials** - **Secret**

- A component used to store secret data
- base64 encoded

Note:

The build-in security mechanism is not enabled by default.

2.4 Volumes

Data storage - E.g. for the Db container inside the Pod, if the data is stored in the Pod and, when the **pod is restarted**, the DB **data is gone**.

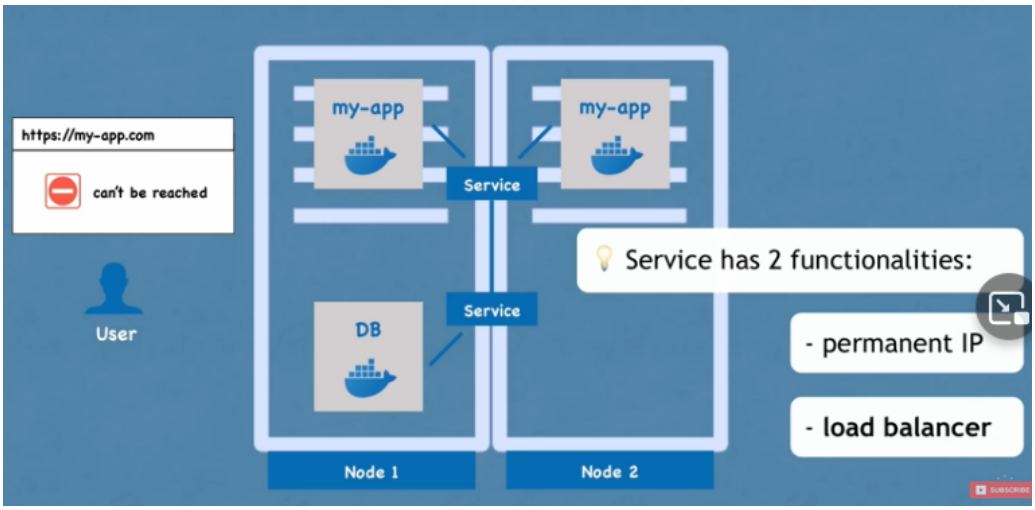
Solution - **Volumes** - Attach a physical hard drive to your POD. The volumes could be on local (inside the k8s cluster) machine, or remote (outside of the K8s cluster).

Note:

> K8s doesn't manage data persistence! - It's your responsibility to backup, replica and manage the data.

2.5 Deployment and Stateful Set

Deployment:



We need to define blueprints for pods: How many pods are needed. This **blueprint for my-app pods** are **called Deployment**

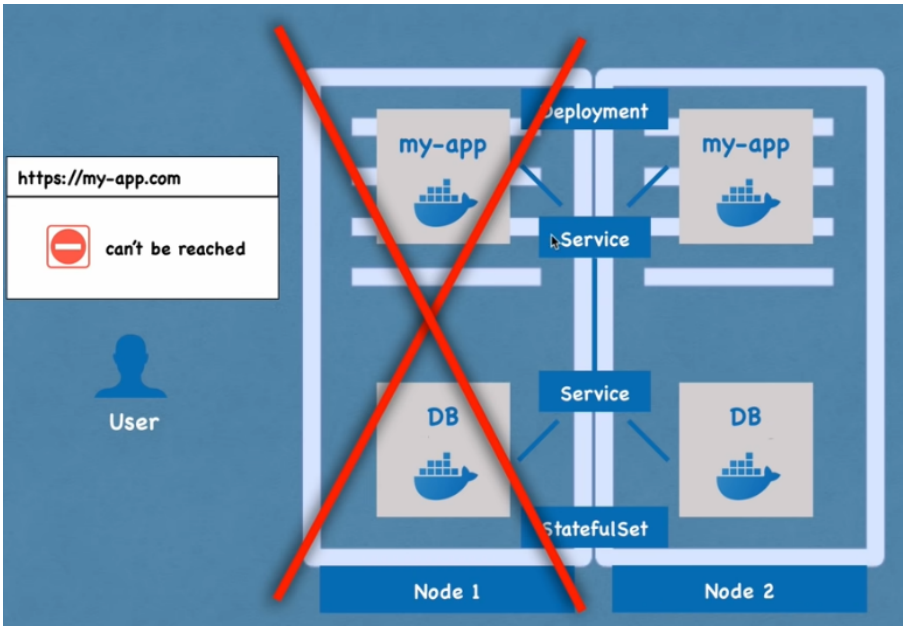
In practice, you **don't work with PODs**, you **actually** creates **Deployments**, where you can specify how many PODs are needed.

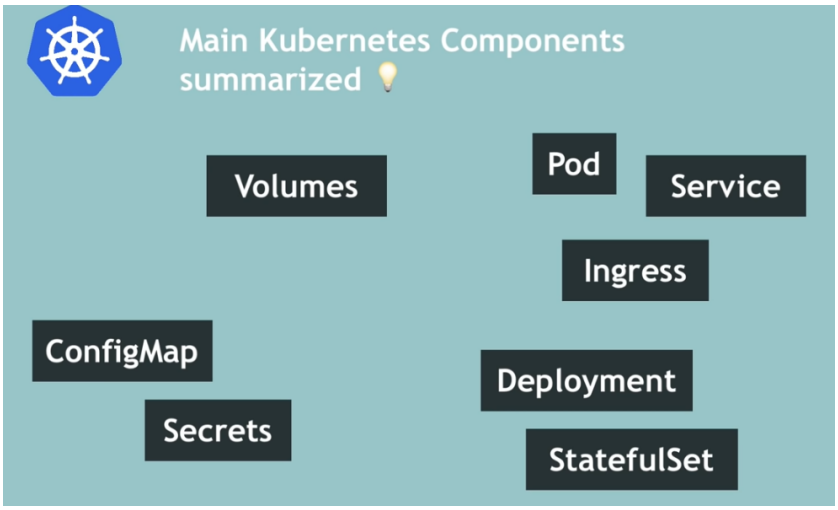
StatefulSet:

But, for **DB** - **DB can't be replicated via Deployment!** - Because database has State which is its data.

Solution - Another K8s component - **StatefulSet** - For STATEFUL apps such as MongoDB, ElasticSearch, etc, which can guarantee data consistency

Deploying StatefulSet is NOT EASY! - So DB are often hosted outside of K8s cluster.

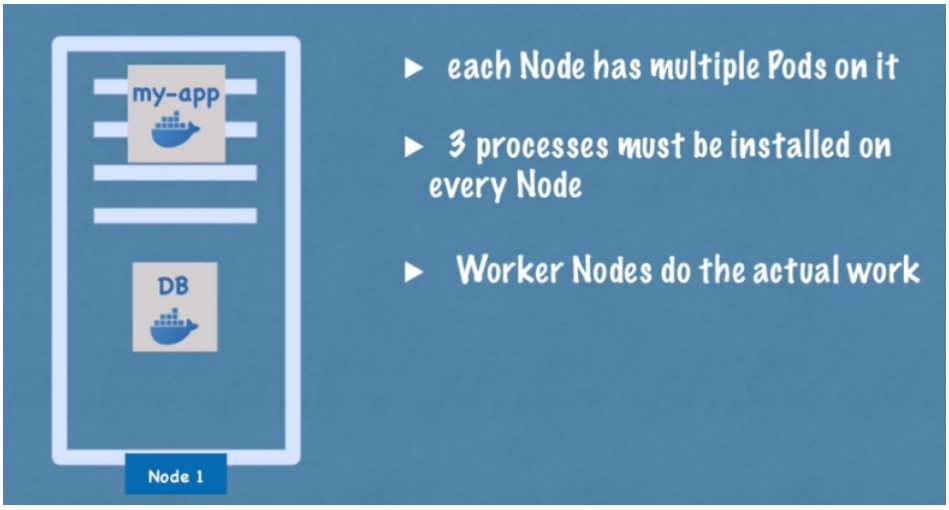




3. Kubernetes Architecture

3.1 Node Processes

Worker machine in K8s cluser



3 processes on every Worker Node:

- **Container runtime**
- **Kubelet**
  - Interacts with both the Container and Node
  - Starts the pod with a container inside
- **Kube proxy** - Forward the requests

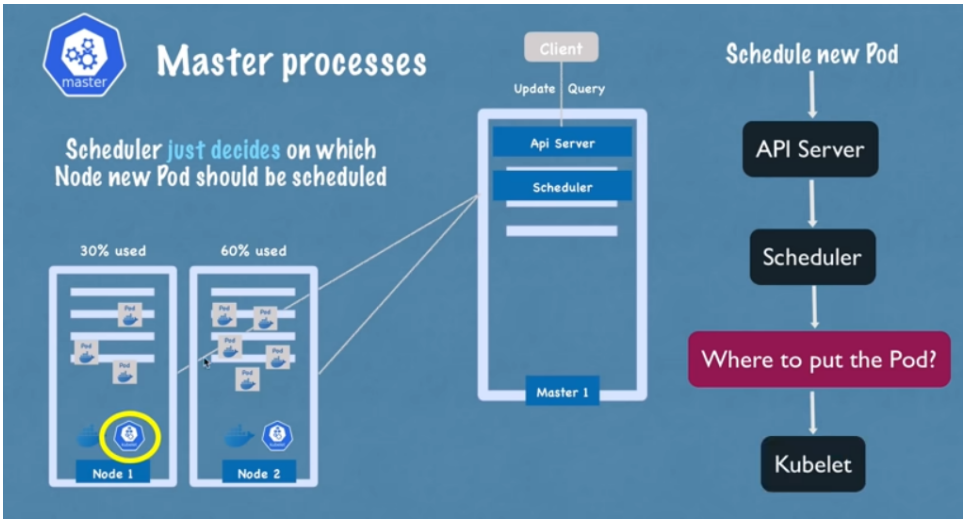
3.2 Master Node

- Schedule pod
- Monitor
- re-schedule / re-start pod
- Join a new Node

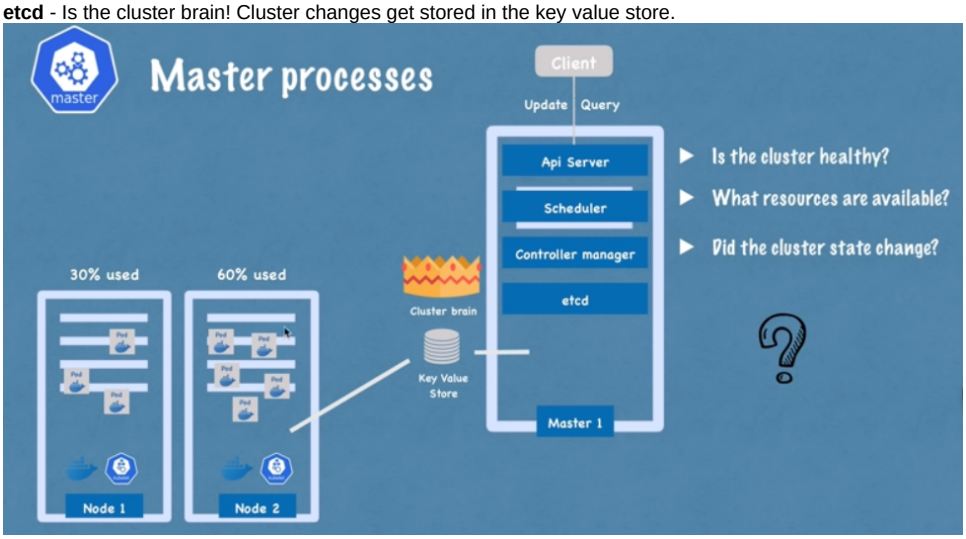
4 processes run on every Master Node:

- **API Server**
  - **Cluster gateway**, which gets the initial requests of update/query of the cluster.
  - Acts as a **gatekeeper** for authentication

- **Scheduler** - Get the request from the scheduler, and kubelet executes the request



- **Controller manager** -Detects cluster state changes
    - Controller Manager Scheduler Kubelet
- **etcd** - Is the cluster brain! Cluster changes get stored in the key value store.

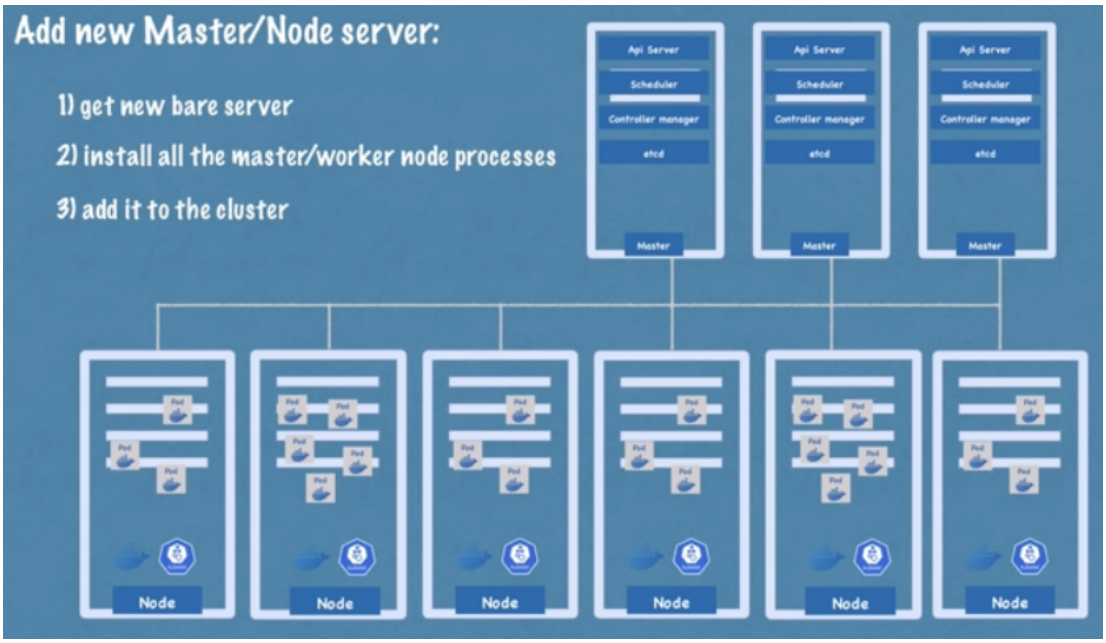


Application data IS NOT stored in **etcd**.

Usually **K8s cluster** runs **multiple Master Nodes**.

3.3 Example cluster setup

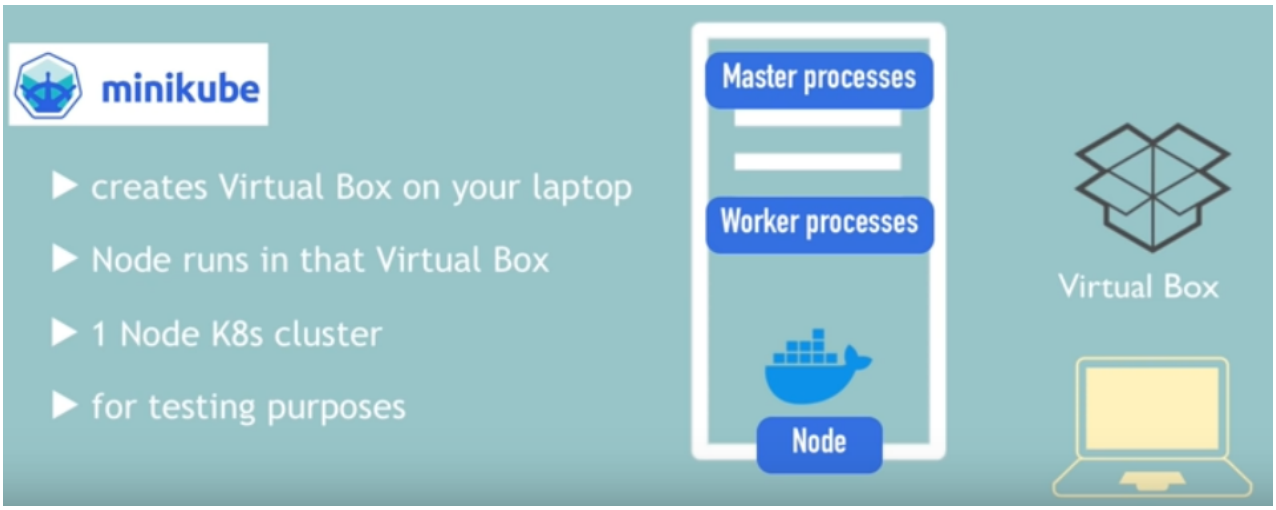
Usually **Master** Nodes need **LESS resources**, and **Work** Nodes need **MORE resources**.



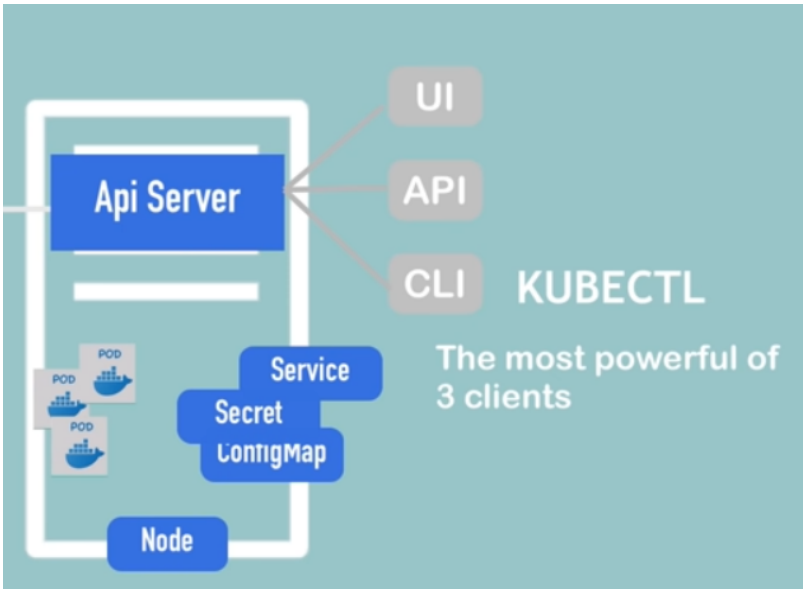
4. Minikube and kubectl - Local Setup

4.1 Minikube

Single node. **Master** Node and **Worker** Node run on the **same machine**, and **Docker** is also **installed**!



4.2 Kubectl - Command line tool for K8s cluster



4.3 Installation

Note: **Virtualization** on your machine is **needed**!

See Kubernetes for the Absolute Beginners

Install Virtualization and Minikube:

- Windows - **Kubectl** and **Hyper**-V or **VirtualBox**
- Linux - **Kubectl** and **KVM** or **VirtualBox**
- MacOS
    - brew install **hyperkit**
    - brew install **minikube** (this will install kubectl as well)

Start Kubernetes Cluster:

> minikube start --vm-driver=xxxxxx

- Windows - On my local, it starts using "hyper-v" vm-driver
- MacOS - --vm-driver=hyperkit

```
* minikube v1.25.2 on Microsoft Windows 10 Pro 10.0.19043 Build 19043
* Using the hyperv driver based on existing profile
* Starting control plane node minikube in cluster minikube
* hyperv "minikube" VM is missing, will recreate.
E0317 21:46:40.243776   40020 main.go:126] libmachine: [stderr =====>] : Hyper-V\Stop-VM : Windows PowerShell
is in NonInteractive mode. Read and Prompt functionality is not available.
At line:1 char:1
+ Hyper-V\Stop-VM minikube
+ ~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [Stop-VM], VirtualizationException
    + FullyQualifiedErrorId : InvalidOperation,Microsoft.HyperV.PowerShell.Commands.StopVM

* Creating hyperv VM (CPUs=2, Memory=6000MB, Disk=20000MB) ...
* Preparing Kubernetes v1.23.3 on Docker 20.10.12 ...
  - kubelet.housekeeping-interval=5m
  - Generating certificates and keys ...
  - Booting up control plane ...
  - Configuring RBAC rules ...
* Verifying Kubernetes components...
  - Using image kubernetesui/dashboard:v2.3.1
  - Using image kubernetesui/metrics-scraper:v1.0.7
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, dashboard, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Note: Docker is pre-installed inside the container

> kubectl get nodes

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   135m
```

> minikube status

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

5. Main Kubectl Commands

5.1 Kubectl Commands

> kubectl get nodes

List one node which is the Master Node

> kubectl get pod — Empty for now

> kubectl get services

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   135m
```

Create pod

Pod is the smallest unit in Kubernetes cluster, we **don't create POD** directly. Instead, we create **Deployment** - abstraction over Pods

> **kubectl create deployment nginx-depl --image=nginx**

```
deployment.apps/nginx-depl created
```

This will **download** the latest **nginx image** from **Docker Hub**

> kubectl get deployments

```
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
nginx-depl   1/1     1            1           52s
```

> kubectl get pods ---- Now there is **One POD** created

```
NAME                          READY   STATUS    RESTARTS   AGE
nginx-depl-5ddc44dd46-lg7df   1/1     Running   0          103s
```

> kubectl get replicaset

```
NAME                    DESIRED   CURRENT   READY   AGE
nginx-depl-5ddc44dd46   1         1         1       3m16s
```

- **Deployment manages Relicaset**
- **Replicaset manages the replicas of that Pod**
- **Pod is an abstraction of a Container**

> kubectl edit deployment nginx-depl

This opens the **deployment configuration file** (**kubectl-edit-4096125223.yaml**) in default text editor (Notepad)

We can edit it. For example, we can change nginx image version

```
...
    containers:
    - image: nginx
// Change to
    - image: nginx:1.16
```

Save and exit the editor.

```
deployment.apps/nginx-depl edited
```

> kubectl get pod

```
NAME                          READY   STATUS    RESTARTS   AGE
nginx-depl-7d459cf5c8-sk2d9   1/1     Running   0          118s
```

Note: The name is changed, which means the **OLD pod was gone**, and the **NEW pod was created**

> kubectl get replicaset

```
NAME                    DESIRED   CURRENT   READY   AGE
nginx-depl-5ddc44dd46   0         0         0       20m
nginx-depl-7d459cf5c8   1         1         1       7m21s
```

Note: The OLD ONE has no pod in it, and NEW ONE was created.

Debugging Pod

> kubectl logs *pod-name* ---- *Empty now*

Show what the application runs in the pod logged.

> kubectl create deployment mongo-depl --image=mongo

> kubectl get pod

```
NAME                          READY   STATUS    RESTARTS   AGE
mongo-depl-85ddc6d66-xmfl7    1/1     Running   0          20s
nginx-depl-7d459cf5c8-sk2d9   1/1     Running   0          12m
```

> kubectl describe pod mongo-depl-85ddc6d66-xmfl7

> kubectl logs mongo-depl-85ddc6d66-xmfl7

Enter the terminal of the MongoDB application container

> kubectl exec -it mongo-depl-85ddc6d66-xmfl7 – bin/bash

Delete deployment

> kubectl delete deployment mongo-depl

> kubectl get pod — The pod for mongodb is gone

> kubectl get replicaset — The replicaset for mongodb is gone

5.2 Kubernetes Configuration File

Create using configuration file

> kubectl apply -f xxxxx.yaml

Create a basic configuration file for a deployment - **nginx-deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16
        ports:
        - containerPort: 8080
```

> kubectl get deployment

```
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           70s
```

> kubectl get pod

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-7956bd8bb9-kckwt   1/1     Running   0          80s
nginx-deployment-7956bd8bb9-mnvvq   1/1     Running   0          80s
```

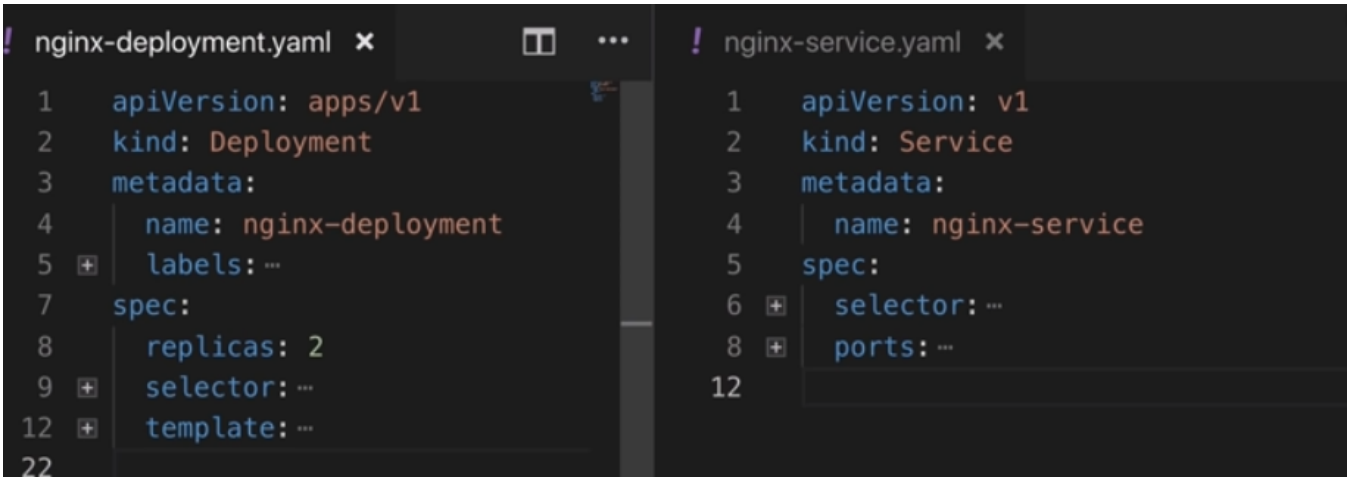
If we want to change, just update **nginx-deployment.yaml** and apply again.

Delete using configuration file

> kubectl delete -f d:\workspace\docker_kubernetes\Kubernetes_Tutorial_for_Beginners\nginx-deployment.yaml

5.3 K8S YAML Configuration File

5.3.1 The 3 parts of configuration file



5.3.1.1 metadata

5.3.1.2 specification

Attributes of "**spec**" are **specific** to the "**kind**"!

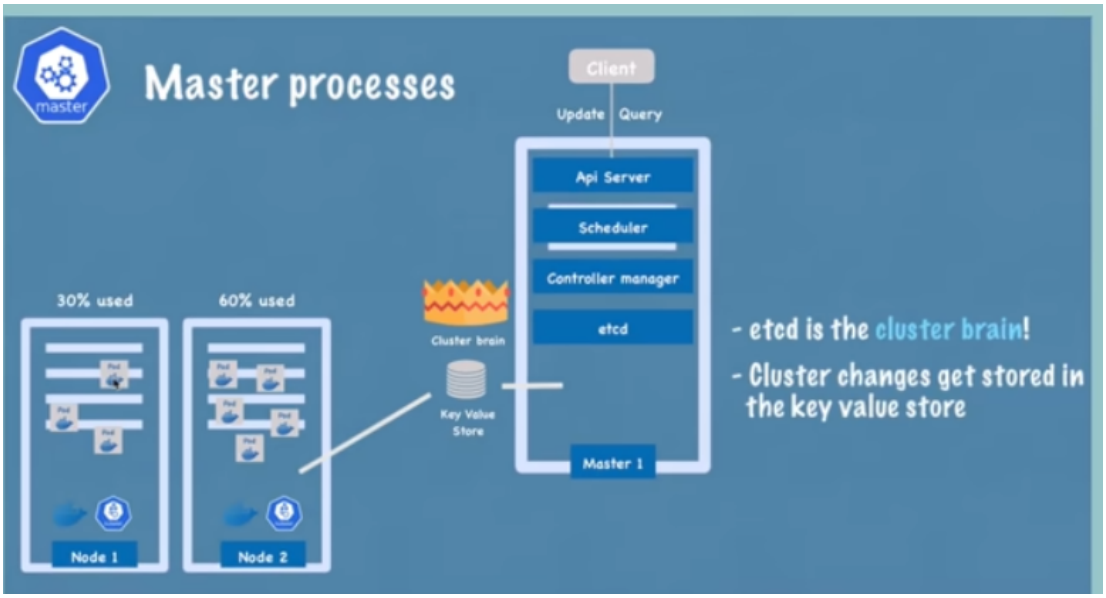
5.3.1.3 status

Automatically generated and added by Kubernetes!

K8s update state continuasly!

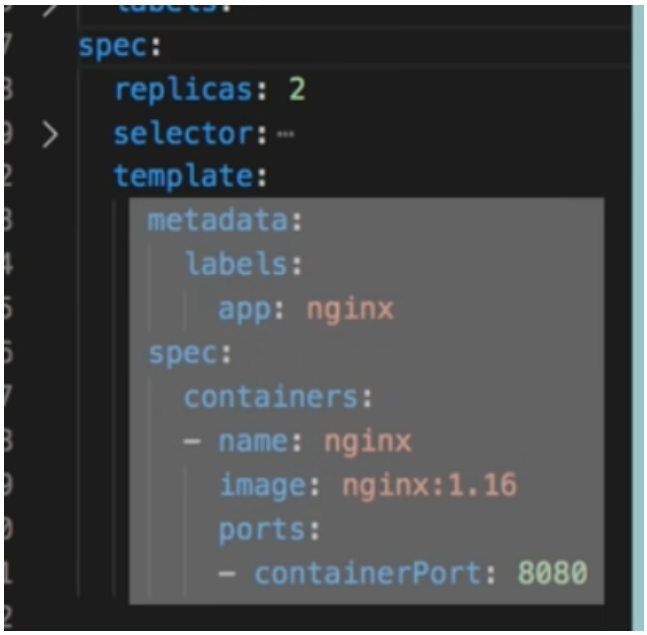
```
status:
  availableReplicas: 1
  conditions:
  - lastTransitionTime: "2020-01-24T10:54:59Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2020-01-24T10:54:56Z"
    lastUpdateTime: "2020-01-24T10:54:59Z"
    message: ReplicaSet "nginx-deployment-7d64f4b
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 1
  readyReplicas: 1
  replicas: 1
  updatedReplicas: 1
```

K8s get the **status from etcd**. Etcd holds the current status of any K8s component!



Usually we **store** the **config file with** your **application code**!

5.3.1.4 Blueprint for pods (Template)

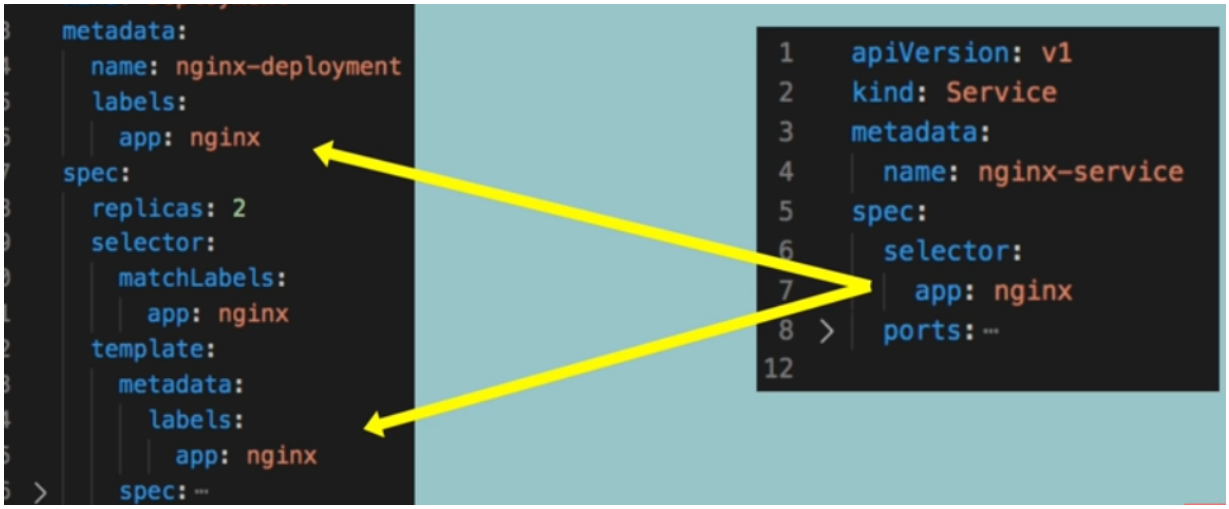


5.3.2 Connecting components (Labels & Selectors & Ports)

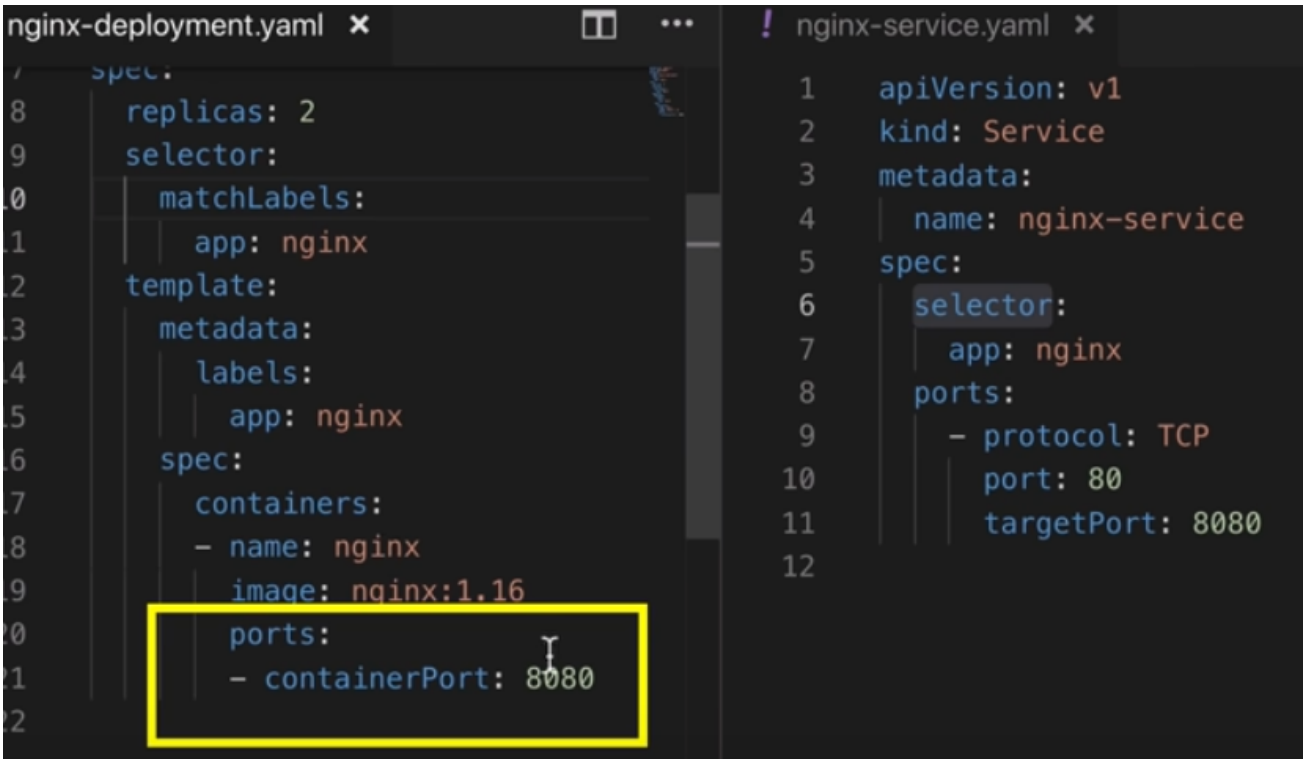
Connecting Deployment to Pods

- **Pods** get the **label** through the **template** blueprint
- This **label** is **matched by the selector**

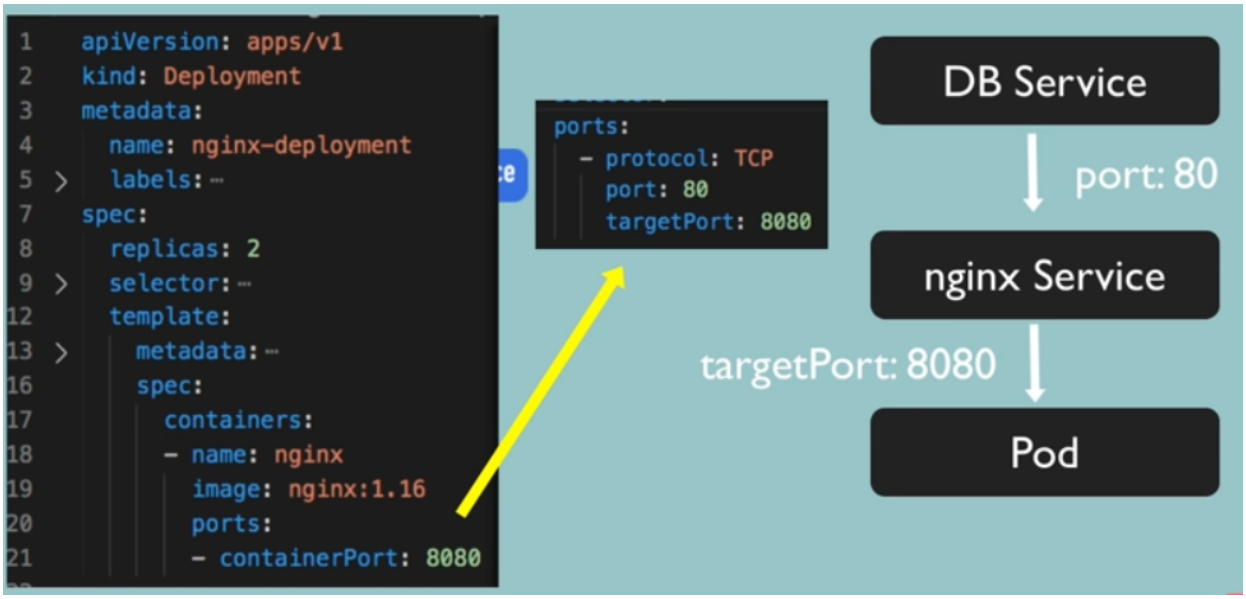
Connecting Services to Deployments



Ports in Service and Pod



When other service sends request to nginx service, they send it on port 80, and the nginx service needs to know to which port it should forward the request and which port is for listening. That is the **targetPort**!



ngnix-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```

> kubectl apply -f d:\workspace\docker_kubernetes\Kubernetes_Tutorial_for_Beginners\nginx-deployment.yaml

> kubectl apply -f d:\workspace\docker_kubernetes\Kubernetes_Tutorial_for_Beginners\nginx-service.yaml

> kubectl get pod

```
nginx-deployment-599bdddccc-2tgz6   1/1     Running   0          113s
nginx-deployment-599bdddccc-zpv5f   1/1     Running   0          113s
```

> kubectl get service — see "**nginx-service**"

```
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes      ClusterIP   10.96.0.1      <none>        443/TCP   2d2h
nginx-service   ClusterIP   10.104.96.98   <none>        80/TCP    70s
```

How to valid the port is correct?

> kubectl describe service nginx-service

```
...
Selector:          app=nginx
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.104.96.98
IPs:               10.104.96.98
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         172.17.0.4:8080,172.17.0.6:8080
...
```

> kubectl get pod -o wide

```
NAME                                READY   STATUS    RESTARTS   AGE     IP           NODE       NOMINATED
NODE   READINESS GATES
nginx-deployment-599bdddccc-2tgz6   1/1     Running   0          5m14s   172.17.0.6   minikube
<none>           <none>
nginx-deployment-599bdddccc-zpv5f   1/1     Running   0          5m14s   172.17.0.4   minikube
<none>           <none>
```

Status

> kubectl get deployment nginx-deployment -o yaml > .....\nginx-deploymen t-result.yaml

This command gets the YAML configuration.

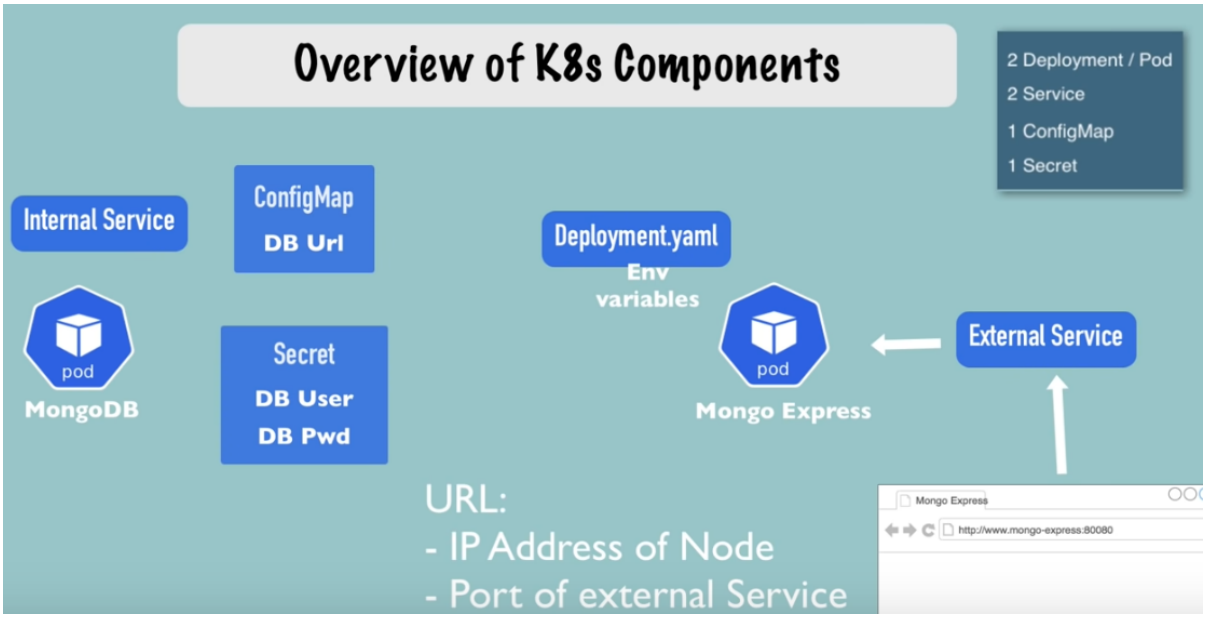
```
......
status:
  availableReplicas: 2
  conditions:
  - lastTransitionTime: "2022-03-20T04:23:08Z"
    lastUpdateTime: "2022-03-20T04:23:08Z"
    message: Deployment has minimum availability.
    reason: MinimumReplicasAvailable
    status: "True"
    type: Available
  - lastTransitionTime: "2022-03-20T04:23:06Z"
    lastUpdateTime: "2022-03-20T04:23:08Z"
    message: ReplicaSet "nginx-deployment-599bdddccc" has successfully progressed.
    reason: NewReplicaSetAvailable
    status: "True"
    type: Progressing
  observedGeneration: 1
  readyReplicas: 2
  replicas: 2
  updatedReplicas: 2
```

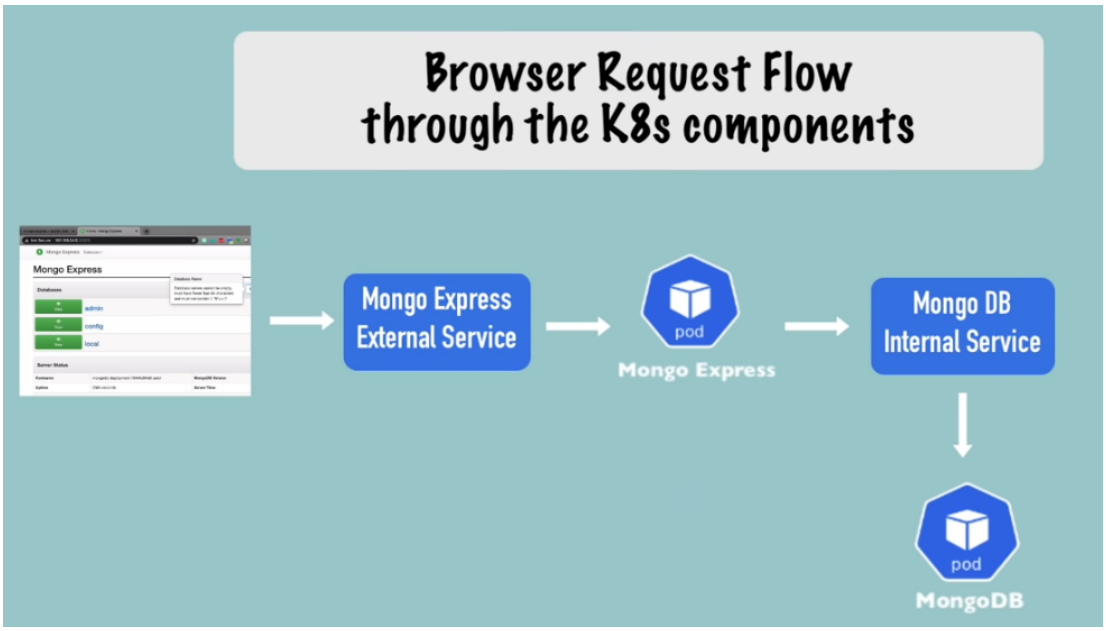
Delete the deployment and service

5.3.3 Complete Demo Project

Complete Application Setup with Kubernetes Components

- mongoDB: https://hub.docker.com/_/mongo
- mongo-express: is a web-based MongoDB admin interface written in Node.js, Express.js, and Bootstrap3. For details, see https://hub.docker.com/_/mongo-express





5.3.3.1 Create MongoDB Deployment & Service & Secret

1. Create MongoDB Deployment

mongo.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
```

```
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  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
```

Need the following Environment Variables

- **MONGO_INITDB_ROOT_USERNAME** - **Value** will be **stored in** a "**Secret**"
- **MONGO_INITDB_ROOT_PASSWORD** - **Value** will be **stored in** a "**Secret**"

sceretKeyRef.name is referencing metadata.data.name in the following secret.yaml

2. Create MongoDB Secret

secret.yaml - Secret must be created before the deployment

```
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
type: Opaque
data:
  mongo-root-username: LW4gJ3VzZXJuYW1lMTIzJyANCg==
  mongo-root-password: LW4gJ3Bhc3N3b3JkMTIzJyANCg==
```

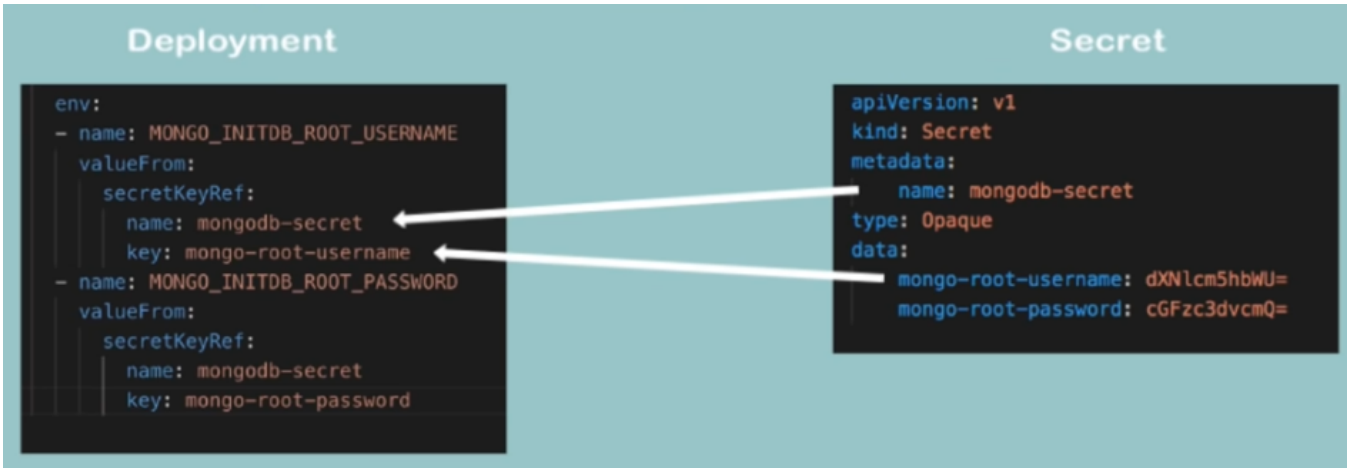
type: Opaque: A default type which is basically key-value pair.

data: The actual contents in **key-value** pairs

The **mongo-root-username** and **mongo-root-password cannot** be **plain text,** must be **base64 encoded**

Generate the the encoded username and password:

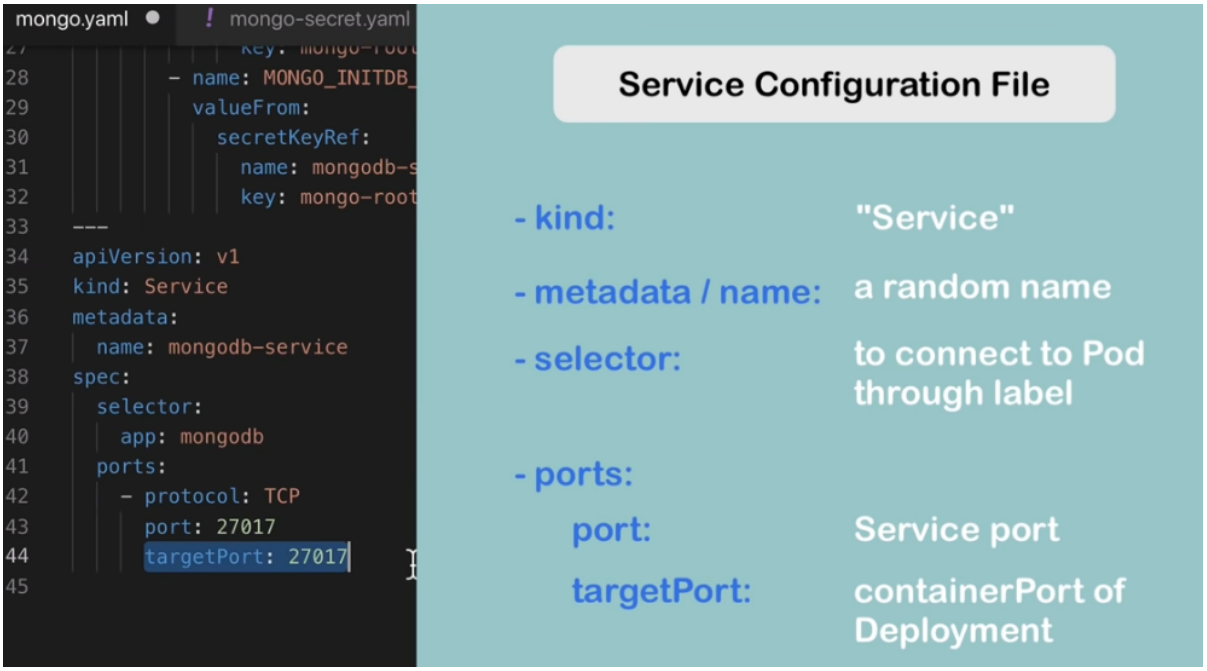
```
> echo -n 'username123' | base64
LW4gJ3VzZXJuYW1lMTIzJyANCg==
> echo -n 'password123' | base64
LW4gJ3Bhc3N3b3JkMTIzJyANCg==
```




```
Create secret first:
> kubectl apply -f D:
\workspace\docker_kubernetes\Kubernetes_Tutorial_for_Beginners\complete_demo_configurations\mongo-secret.yaml
> kubectl get secret
Create deployment
> kubectl apply -f D:
\workspace\docker_kubernetes\Kubernetes_Tutorial_for_Beginners\complete_demo_configurations\mongo.yaml
> kubectl get all
```

3. Create Internal Service for MongoDB

Put it in the same YAML (mongo.yaml) as deployment configuration. Use "---" to separate multiple documents in YAML.



Note: port can be different from targetPort

> kubectl apply -f D:\workspace\docker_kubernetes\Kubernetes_Tutorial_for_Beginners\complete_demo_configurations\mongo.yaml

To check if the service is attached to the correct port

> kubectl describe service mongodb-service

> kubectl get pod -o wide

```
NAME                                  READY   STATUS             RESTARTS      AGE     IP
mongodb-deployment-7bb6c6c4c7-dw5h5   0/1     CrashLoopBackOff   6 (54s ago)   6m53s   172.17.0.6
```

Now we have he IP address: 172.17.0.6

5.3.3.2 Create Mongo Express Deployment & Service & ConfigMap