How to build your own Empire with Kubernetes

Introduction

K8s - Intro
Intro - Who's that guy?

Lars Martin
Software Engineer / Container Whisperer

Docker / Kubernetes / DevOps / Backend
IoT / Machine Learning / Home Automation

SMB GmbH
lars.martin@smb-tec.com
@CrystalMethodLE
www.smb-tec.com

K8s - Intro
Intro - Who's that guy?

Passionate Java Developer (especially Spring)
Python, Go-Lang
Agile and Devops infected
Container enthusiast
berndfischer63@gmail.com
@berndfischer63

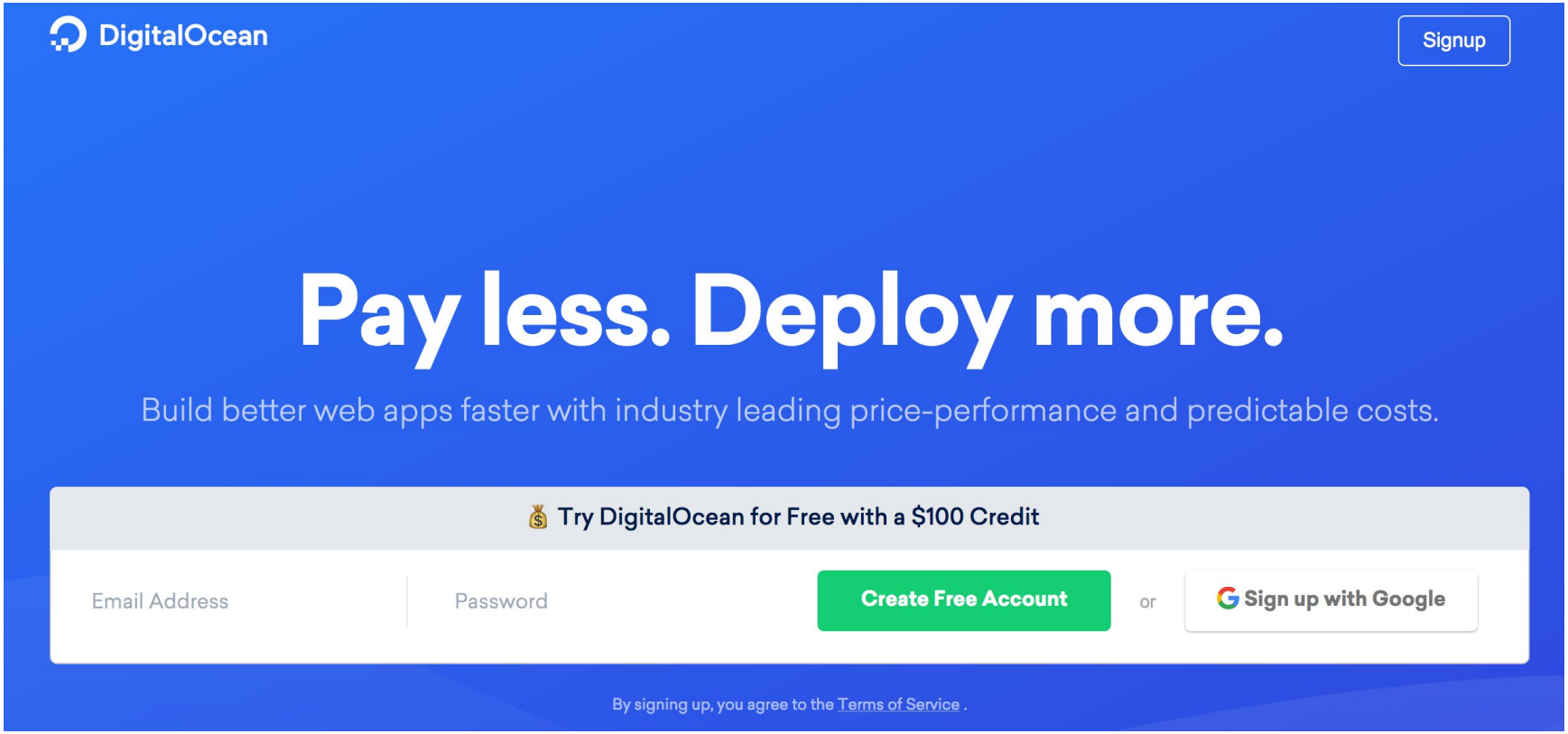
JUG Saxony e.V., Docker Community Dresden

CTO MindApproach GmbH, Dresden
bfischer@mindapproach.de

K8s - Intro

Infrastructure Sponsor

K8s - Intro
Infrastructure Sponsor



https://do.co/jugsaxony-serverless

K8s - Intro
Prerequisites

- Notebook
  - WLAN, USB
  - OS
  - SSH-Client, besser Bash-Terminal
- Some knowledge about
  - Linux,
  - How to work with Linux remotely (ssh, terminal, …)
  - Docker

K8s - Intro
What's Covered

- You get
  - "super speed dating" with basic Kubernetes feature
    - Architecture, Networking, ...
    - Labels,
    - Pods, ReplicaSets, Deployments
    - Service
    - State, Persistence, Volumes
    - Ingress
  - ready to use Kubernetes cluster to try examples by yourself

K8s - Intro
What's not Covered

- ❑ Installation
- ❑ Deep dive into internals
- ❑ High Availability
- ❑ Logging, monitoring, …
- ❑ Service mesh
- ❑ RBAC
- ❑ …
- ❑ !!!
- ❑ Non-productive environment and examples
- ❑ !!!

K8s - Intro
Links

- Source Code
  - https://gitlab.com/aemc/demo/demo-k8s-cluster
- Kubernetes
  - [Cncf01] Cloud Native Computing Foundation https://www.cncf.io/
  - [K8s01] Kubernetes Project https://kubernetes.io/
  - [K8s02] Kubernetes Documentation Home https://kubernetes.io/docs/home/
  - [K8s03] Kubernetes Documentation Concepts https://kubernetes.io/docs/concepts/
  - [K8s04] Kubernetes Documentation Tasks https://kubernetes.io/docs/tasks/
  - [K8s05] Kubernetes Documentation Tutorials https://kubernetes.io/docs/tutorials/

K8s - Intro

This is the last slide ...

How to build your own Empire with Kubernetes

Basics

K8s - Basics
What is it for?

Kubernetes is an open source system for

- managing containerized applications across multiple hosts,
  - stateless, stateful, jobs
- providing basic mechanisms for
  - deployment,
    - installation, configuration and update
  - maintenance and
  - scaling of applications.

https://kubernetes.io/docs/home/

K8s - Basics
Nodes

- Types
  - Manager (aka master)
  - Worker (formerly named “minion”)
- Container Runtime
  - Docker
  - Container Runtime Interface
    - ContainerD + RunC
    - ContainerD + rkt
  - CRI-O
  - Rkt
  - ...

K8s - Basics
Nodes - Master (Control Plane)

- cluster storage - (more or less) distributed
  - etcd (distributed reliable key-value-store)
    - https://coreos.com/etcd/
    - https://github.com/etcd-io/etcd
- kube-apiserver
- kube-scheduler
- kube-dns
- controller manager
  - replication controller
  - endpoint controller
  - ...

K8s - Basics
Nodes - Worker

- ❏ kubelet
- ❏ kube-proxy

K8s - Basics
Installation

- Docker4Mac/Docker4Win
- Minikube
- From scratch (aka “the hard way”)
  - https://github.com/kelseyhightower/kubernetes-the-hard-way
- Kubeadm, Kops, Kubespray, ...
- Many cloud-specific solutions
- Cloud-specific “ready-to-use” offers
  - Amazon (EKS), MS Azure (AKS), Google, Digital Ocean (beta), 1&1, …

K8s - Basics
Installation

- Products
  - RedHat OpenShift, Suse CaaS, Rancher 2.0 (RKE), …
- “Play with Kubernetes”
  - https://labs.play-with-k8s.com/
- KataCoda
  - https://www.katacoda.com/courses/kubernetes

K8s - Basics
Declarative vs. Imperative Way

- API is resource oriented (some kind of REST)
  - GET, POST, ...
- imperative
  - commands via CLI (kubectl)
  - translated into API calls
  - not as mighty as API itself
- declarative
  - resource description
  - yaml/(json) files (aka manifests)

K8s - Basics
Labels

- used as metadata to identify or group k8s-objects
- hint: they aren’t meant as "uid"

K8s - Basics
Labels

- key-value-pairs
- metadata to identify or group k8s-objects
- keys and values are “strings”
- `keys := 0{prefix/}1 + name`
- prefix := DNS subdomain and <= 253 characters
- name:= <= 63 characters; alphanumerical; "-" and "_" and "." are allowed except first and last character

Key	Value
acme.com/app-version	1.0.0
appVersion	1.0.0
kubernetes.io/cluster-service	true

K8s - Basics
Selectors

- similar to “SQL Where-Clause” to select (sub) group of k8s-objects based on labels

Operator	**Description**
`key = value`	
`key != value`	
`key in (value1, value2)`	
`key notin (...)`	
`key`	`key exists`
`!key`	`key doesn’t exist`

K8s - Basics

This is the last slide ...

How to build your own Empire with Kubernetes

Demo Environment

K8s - Demo Environment
Overview

- 3x VM/Droplets (Digital Ocean)
  - s-1vcpu-2gb, 50 GByte HD, FRA1
  - Ubuntu 18.04 LTS
- 1x Master
- 2x Worker

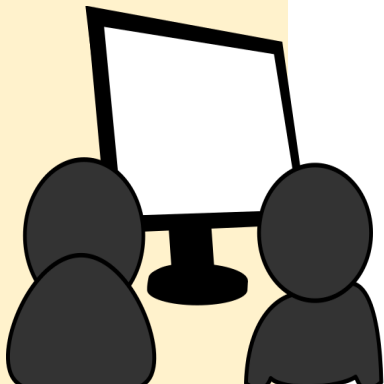
K8s - Demo Environment
Naming

- DNS domain
  - **`aemc.me`**
- Node names
  - **`[m01|w01|w02]-k8s-cluster<nr>`**

- **`!!! IMPORTANT !!!`**
  - **`Use your own cluster (by number)`**
  - **`Please ….`**

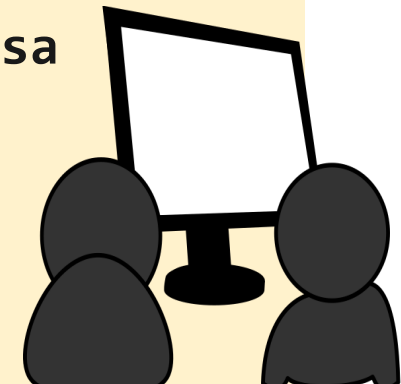
K8s - Demo Environment
Accessing the Cluster

```
Local:<somewhere> bf$

# download ssh private key
# care about formatting ... !!!
URL=https://gitlab.com/aemc/demo/demo-k8s-cluster/
uploads/6adb76148237d15722dfa156f44a0fe4/id_rsa_dmo

curl $URL --output id_rsa_dmo

# or use browser and download manually
open $URL
```



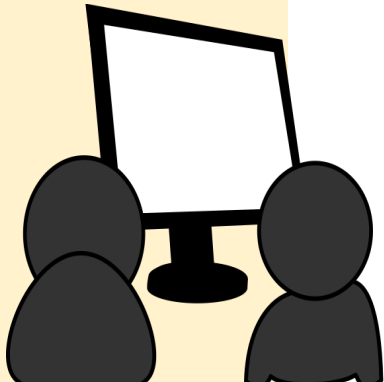
K8s - Demo Environment
Accessing the Cluster

```
Local:<somewhere> bf$

ls -al id_rsa_dmo
-rw-r--r--   1 bf  staff  29845 25 Okt 17:42 id_rsa

# set access rights
chmod 0600 id_rsa_dmo

ls -al id_rsa_dmo
-rw-------  1 bf  staff  29845 25 Okt 17:42 id_rsa
```



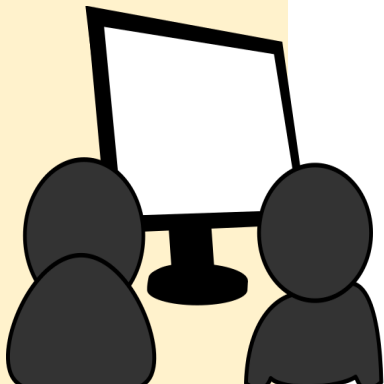
K8s - Demo Environment
Accessing the Cluster

```
Local:<somewhere> bf$

# hint: !!! use your cluster number !!! PLEASE !!!
ssh ubuntu@m01.k8s-cluster<nr>.aemc.me \
  -i id_rsa_dmo \
  -o UserKnownHostsFile=/dev/null \
  -o StrictHostKeyChecking=no

# !!!
# !!! replace <nr> with <your-number> !!!
# !!!
```



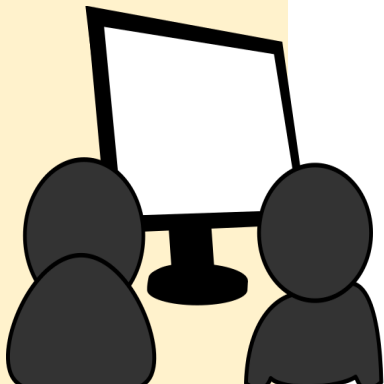
K8s - Demo Environment
Preparations

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ ls -al
$ cd workshop

$ git clone \
  https://gitlab.com/aemc/demo/demo-k8s-cluster.git

$ cd demo-k8s-cluster
$ ls -al

$ PRJ_DIR=$(pwd)
```



K8s - Demo Environment
Preparations

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl version -o json | jq
$ kubectl -v7 version
$ kubectl cluster-info
$ kubectl get componentstatus

$ kubectl get nodes
$ kubectl get nodes -o wide
$ kubectl get nodes -o yaml | less
$ kubectl get nodes -o json | less
```



K8s - Demo Environment
Demo Application

Wordpress

- Wordpress web application
- MySql as database

K8s - Demo Environment

This is the last slide ...

How to build your own Empire with Kubernetes

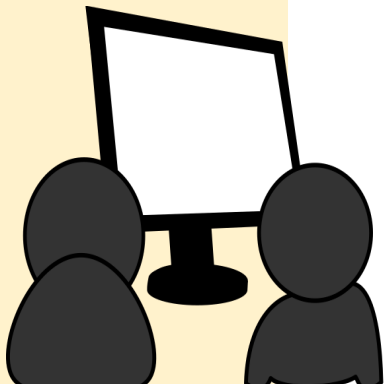
MySql

K8s - MySql
Pod - What is it?

- English:
  - “School”/Group of Whales
- Basic element of k8s (deployment artefakt)
- Group (1..N) of containers
- All containers of a pod share the same
  - node (they run on)
  - ip address, hostname and ipc
- Pods are more or less immutable

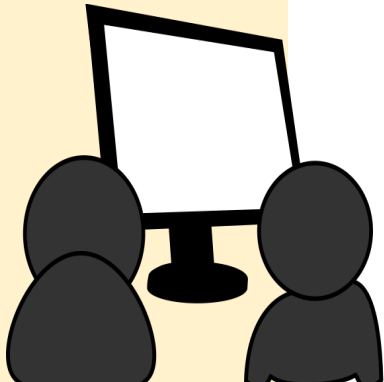
K8s - MySql
Pod - Manifest

```
# k8s-manifests/wordpress-01.yaml (truncated)

---
apiVersion: v1
kind: Pod
metadata:
  name: wordpress-mysql
spec:
  containers:
  - name: mysql
    image: mysql:8.0.3
    ports:
    - containerPort: 3306
```



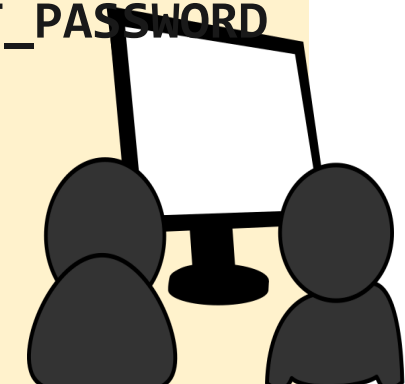
K8s - MySql
Pod - Creation

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-01.yaml
pod/wordpress-mysql created

$ kubectl get pods
NAME              READY STATUS             RESTARTS   AGE
wordpress-mysql 0/1   ContainerCreating 0          10s
wordpress-mysql 0/1   CrashLoopBackOff  6          7m8s

bf$ kubectl get pods --watch
...
```



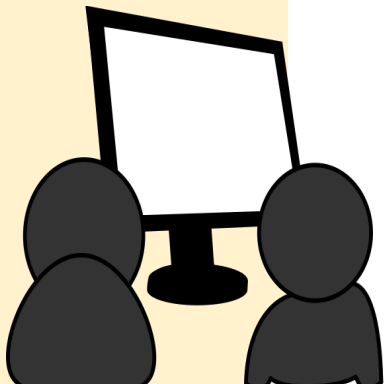
K8s - MySql
Pod - Trouble Shooting

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

bf$ kubectl describe pod wordpress-mysql
...

bf$ kubectl logs wordpress-mysql
error: database is uninitialized and password option is
not specified
  You need to specify one of MYSQL_ROOT_PASSWORD,
MYSQL_ALLOW_EMPTY_PASSWORD and MYSQL_RANDOM_ROOT_PASSWORD
```



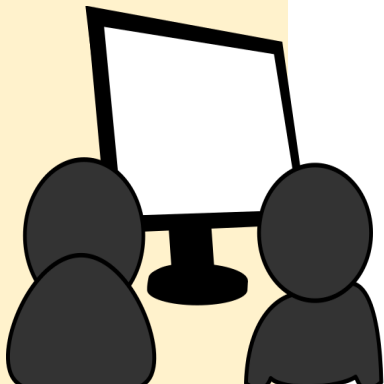
K8s - MySql
Pod - Deletion

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl delete -f k8s-manifests/wordpress-01.yaml
$ kubectl delete pod wordpress-mysql



# Delete a pod with minimal delay
$ kubectl delete pod wordpress-mysql --now

# Delete a pod with minimal delay
# Default: 30s
$ kubectl delete pod foo --grace-period=0
```



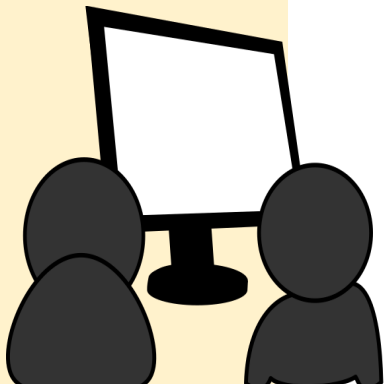
K8s - MySql
Pod - Environment

```
# k8s-manifests/wordpress-02.yaml (truncated)

apiVersion: v1
kind: Pod
metadata:
  name: wordpress-mysql
spec:
  containers:
  - name: mysql
    image: mysql:8.0.3
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "9876"
```



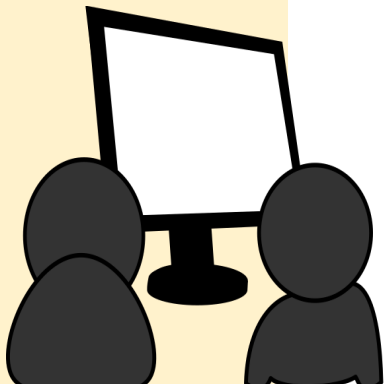
K8s - MySql
Pod - Environment

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-02.yaml
pod/wordpress-mysql created

bf$ kubectl get pods
NAME              READY STATUS  RESTARTS AGE
wordpress-mysql 1/1   Running 0        60s

$ kubectl get pods -o wide
...
# now knowing IP and node
```



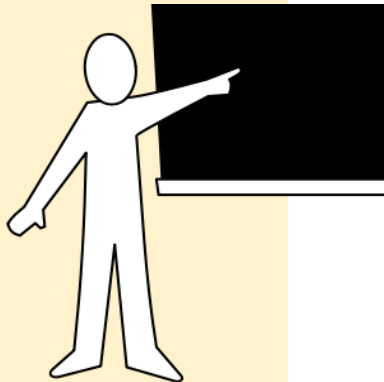
K8s - MySql
Pod - Connectivity

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ IP=$(kubectl get pod wordpress-mysql -o json |\
  jq -r .status.podIP)
$ echo $IP
192.168.1.4

$ nc $IP 3306
P
8.0.3-rc-log0,R
zuo������nS~Qgvimysql_native_password^C
```



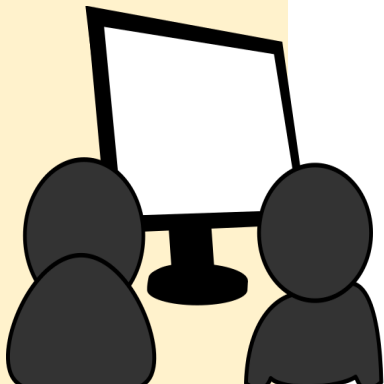
K8s - MySql
Pod - Connectivity from LocalDev

```
Bernds-MBP:demo-k8s-cluster bf$

bf$ kubectl port-forward pod/wordpress-mysql 43306:3306
Forwarding from 127.0.0.1:43306 -> 3306
Forwarding from [::1]:43306 -> 3306
# stays open


# now using IntelliJ Database Tools ...
# jdbc:mysql://localhost:43306
```



K8s - MySql
Pod - Deletion

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl delete -f k8s-manifests/wordpress-02.yaml
```



K8s - MySql
Current State

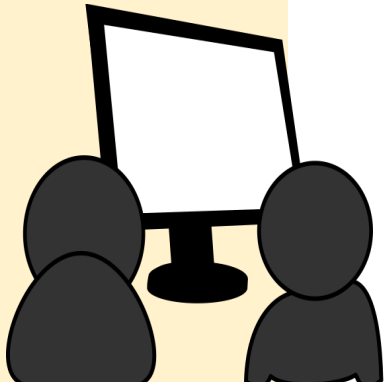
- Pro:
  - create containerized processes (pods)
- Con:
  - not reliable
  - not scalable
  - update ?
- Nice to have
  - redundancy
  - some kind of “self healing”
  - no down time

K8s - MySql
Deployment - What is it?

- Replication-Sets
  - a group of pods based on a "template"
  - paired with some kind of "guaranties"
    - desired state, current state, reconciliation loop
  - scalability
  - designed for stateless architecture
- on top of replication-sets deployments manage updates or rollouts
  - changes to the image field of pod template
  - different strategies to achieve its goals
    - recreate, rollingUpdate

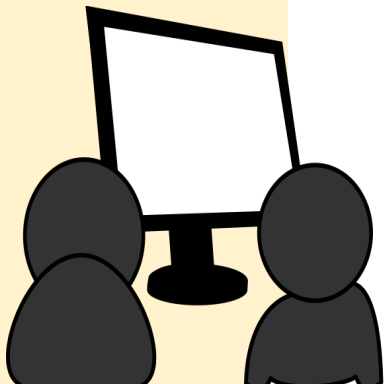
K8s - MySql
Deployment - Manifest

```
# k8s-manifests/wordpress-03.yaml (truncated)

apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
spec:
  selector:
    matchLabels:
      app: wordpress
  template:                    # from here: pod
    metadata:
      labels:
        app: wordpress
```



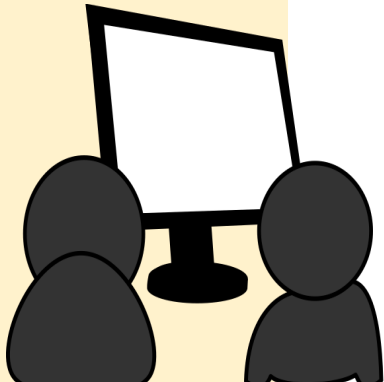
K8s - MySql
Deployment - Creation

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-03.yaml
deployment.apps/wordpress-mysql created
```



K8s - MySql
Deployment - Hierarchy and Naming

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl get deploy,rs,po  # (truncated)

manual         deployment/wordpress-mysql
generated      replicaset/wordpress-mysql-868d5bc85f
generated            pod/wordpress-mysql-868d5bc85f-bwbqk
```

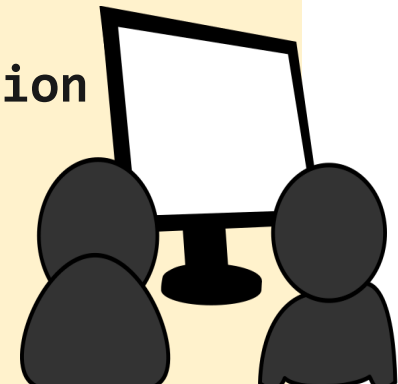


K8s - MySql
Deployment - Informations

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl get deploy wordpress-mysql
NAME              DESIRED CURRENT UP-TO-DATE AVAILABLE
wordpress-mysql 1       1       1          1

# desired    - shall, desired state
# current    - how many replicas are currently running
# up-to-date - number of replicas that have been updated
               to achieve the desired state
# available  - how many replicas of the application
               are available to your users
```

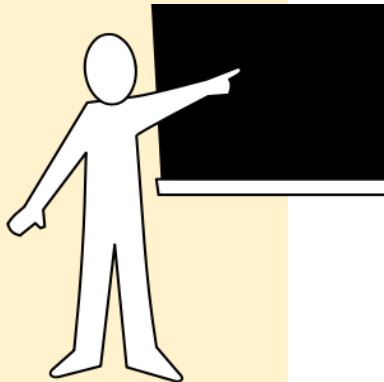
From Deploy

From RS



K8s - MySql
Deployment - Delete

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl delete -f k8s-manifests/wordpress-03.yaml
```



K8s - MySql
Current State

- Pro:
  - create containerized processes (pods)
  - reliable, redundant
  - scalable
  - (update)
- Con
  - state ???
  - persistence ???

K8s - MySql
Volume

- container are ephemeral and after container crash (re-creation), “files” are lost
- sharing files between container inside a pod

The Kubernetes Volume abstraction solves both of these problems.

https://kubernetes.io/docs/concepts/storage/volumes

K8s - MySql
Volume

- Types / k8s objects
  - StorageClass
  - PersistentVolume
  - PersistentVolumeClaim

- Lifecycle
  - Static
  - Dynamic

K8s - MySql
Volume - HostPath

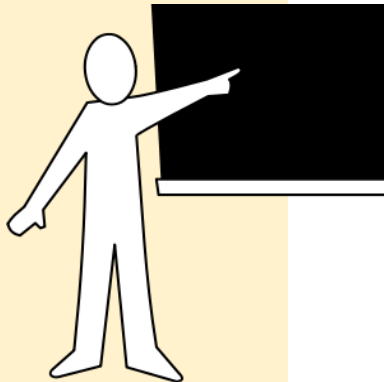
- Binding host dir / file
- aka "mount --bind ..."
- Types
  - Directory / File (resource must exist before)
  - DirectoryOrCreate / FileOrCreate (created if resource doesn’t exist)

K8s - MySql
Volume - Local Storage

- binding host dir / file (see hostPath) plus “binding” to nodes
- automatic scheduling of pods to node the volume lives on
- host dir / file must exists
- beta (not all features implemented)
  - especially static (manual) provisioning only

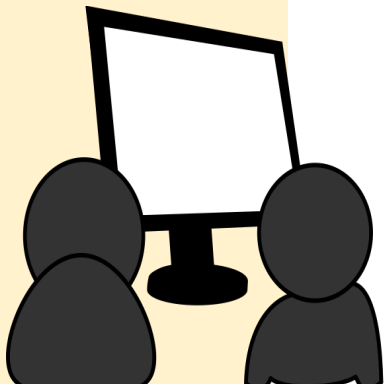
K8s - MySql
Volume - Local Storage - SC

```
# k8s-manifests/cluster-init.yaml

...
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: local-storage
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
...
```



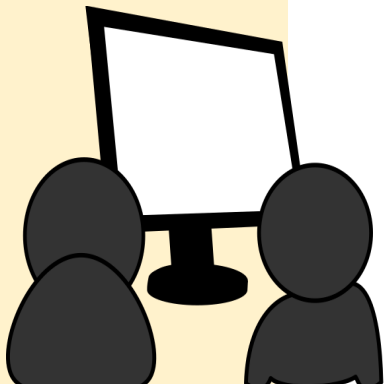
K8s - MySql
Volume - Local Storage - PV

```
# k8s-manifests/wordpress-04.yaml (truncated)

kind: PersistentVolume
apiVersion: v1
metadata:
  name: wp-mysq-data
spec:
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: /mnt/wp-mysql-data
  nodeAffinity:
...
```



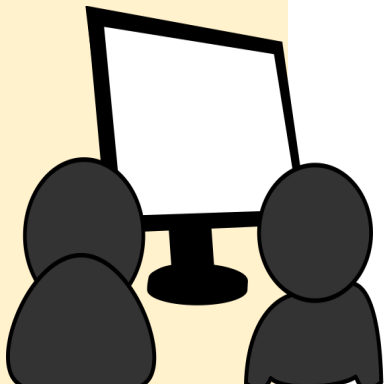
K8s - MySql
Volume - Local Storage - PVC

```
# k8s-manifests/wordpress-04.yaml (truncated)

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: wp-mysql-data
spec:
  storageClassName: local-storage
  resources:
    requests:
      storage: 5Gi
  selector:
    matchLabels:
      pv-name: wp-mysql-data
```



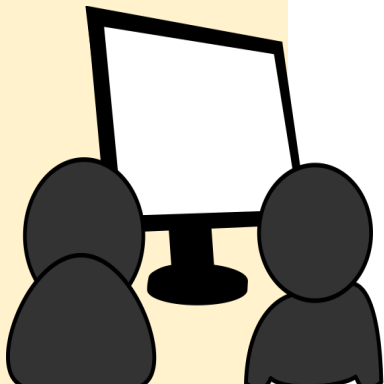
K8s - MySql
Volume - Local Storage - Pod

```
# k8s-manifests/wordpress-04.yaml (truncated)

kind: Deployment
spec:
  template:
    spec:
      containers:
      - name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: wp-mysql-data
```



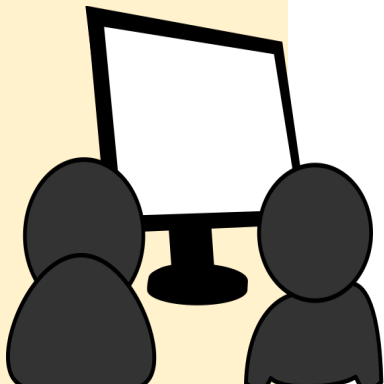
K8s - MySql
Volume - Local Storage - Hack

```
# k8s-manifests/wordpress-04.yaml (truncated)

kind: Deployment
spec:
  template:
    spec:
      containers:
      initContainers:
       ...
      volumes:
      - name: init-data
        hostPath:
          type: DirectoryOrCreate
          path: /mnt/wp-mysql-data
```



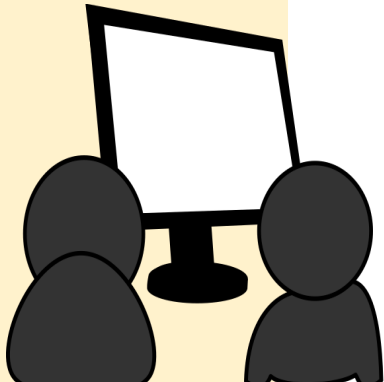
K8s - MySql
Volume - Create

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-04.yaml
persistentvolume/wp-mysq-data created
persistentvolumeclaim/wp-mysql-data created
deployment.apps/wordpress-mysql created

$ kubectl get pv
$ kubectl get pvc
$ kubectl get deploy
```



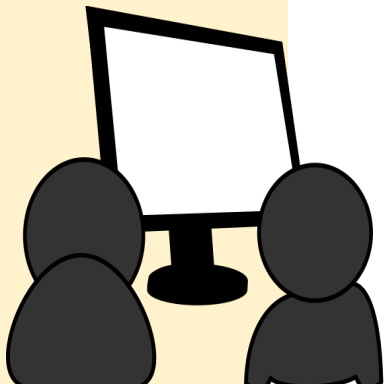
K8s - MySql
Volume - Check

```
ubuntu@w02-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ ls -al /mnt/wp-mysql-data
total 163836
drwxr-xr-x 6  999  999 4096 Oct 26 10:00 .
drwxr-xr-x 3 root root 4096 Oct 25 20:54 ..
-rw-r----- 1  999  999   56 Oct 25 20:54 auto.cnf
-rw------- 1  999  999 1675 Oct 25 20:54 ca-key.pem
-rw-r--r-- 1  999  999 1078 Oct 25 20:54 ca.pem
...
```



K8s - MySql
Volume - Delete

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl delete -f k8s-manifests/wordpress-04.yaml
persistentvolume "wp-mysq-data" deleted
persistentvolumeclaim "wp-mysql-data" deleted
deployment.apps "wordpress-mysql" deleted

# data in host dir still present / available
```



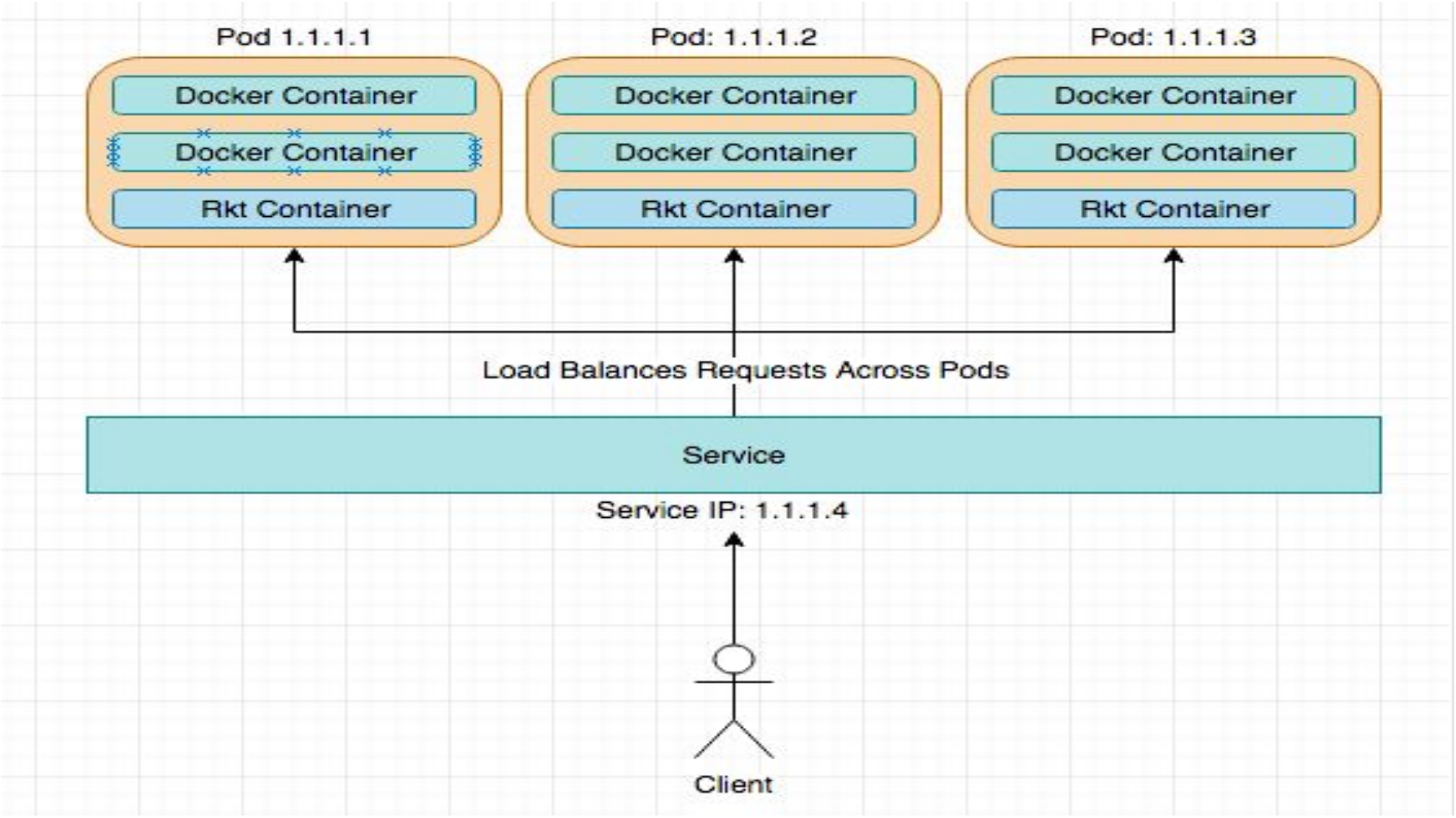
K8s - MySql
Current State

- Pro:
  - create containerized processes (pods)
  - reliable, redundant
  - scalable
  - (update)
  - persistence
- Con
  - stable access to pods ???
    - changes to DNS are slowly distributed ...
  - load balancing ???

K8s - MySql
Service - What is it?

- k8s-object
- more or less metadata to control behavior of k8s - not "backed" by "real" object like pods
- mutable
- frontend for pods
  - stable IP and **internal** DNS name
  - **internal** load balancer

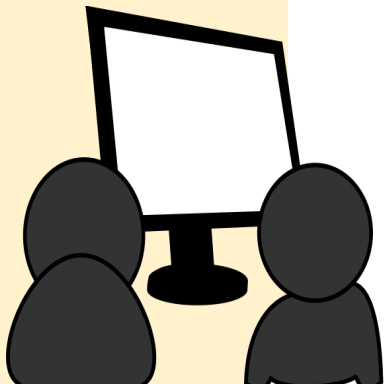
K8s - MySql
Service - Relation to Pods



https://cdn-images-1.medium.com/max/1143/1*dR_WPjgb81puXsyeErk_GA.png

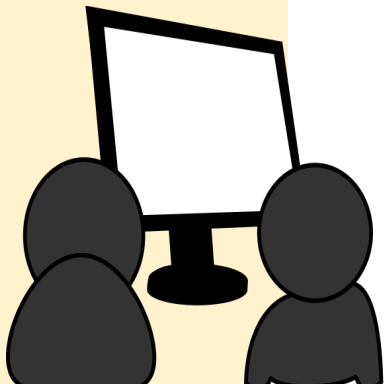
K8s - MySql
Service - Manifest

```
# k8s-manifests/wordpress-05.yaml

apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
  - port: 3306
  selector:
    app: wordpress
    tier: mysql
```



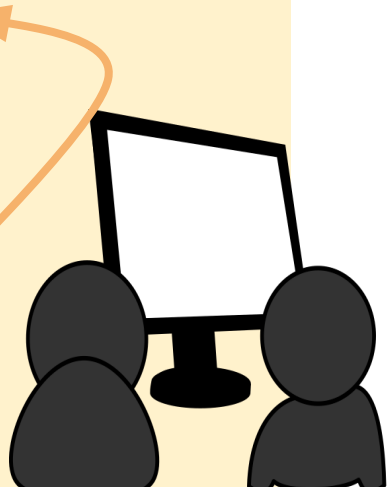
K8s - MySql
Service - Creation

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-05.yaml
```



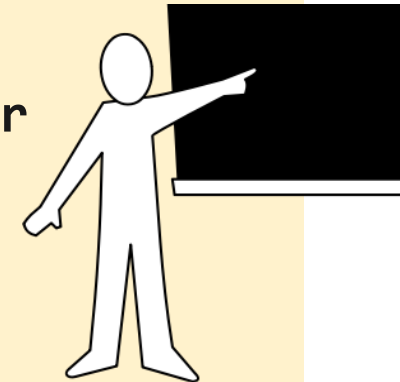
K8s - MySql
Service - Pod Selection

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

#selector from manifest:
#    app: wordpress
#    tier: mysql

$ kubectl get pods -L tier -L app
NAME                READY STATUS  TIER  APP
wordpress-mysql... 1/1   Running mysql wordpress

kubectl get svc wordpress-mysql -o wide
NAME              ... SELECTOR
wordpress-mysql ... app=wordpress,tier=mysql
```



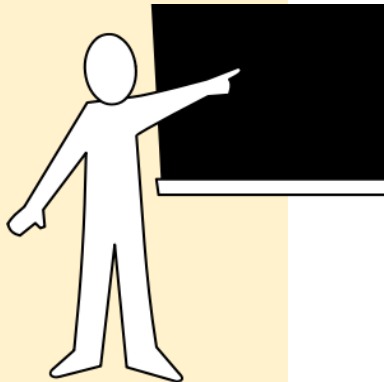
K8s - MySql
Service - Connectivity

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ nc 10.96.45.228 3306
P
8.0.3-rc-logf|3| D�����
jVm:"Qmmysql_native_password^C

nc wordpress-mysql 3306
... Temporary failure in name resolution

=> Service Discovery doesn’t work outside cluster
```



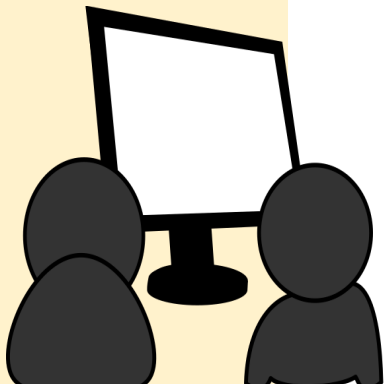
K8s - MySql
Service - Connectivity

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl run -it --rm --image=alpine --restart=Never \
  debug sh
If you don't see a command prompt, try pressing enter.
/ #
/ # nc wordpress-mysql 3306
P
8.0.3-rc-lo"zSXK0������we<
TYvpIi.Zmysql_native_password^Cpunt!
/ # exit

=> Service Discovery works inside cluster
```



K8s - MySql
Service - Deletion

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl delete -f k8s-manifests/wordpress-05.yaml
```



K8s - MySql

This is the last slide ...

How to build your own Empire with Kubernetes

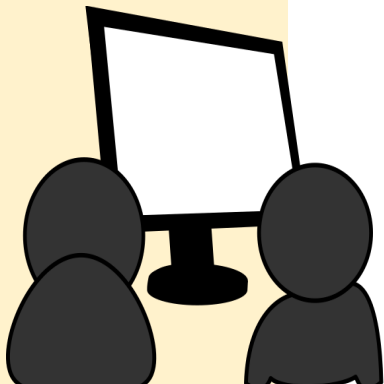
Wordpress Webapp

K8s - Webapp
Deployment

- Docker Image
  - https://hub.docker.com/r/library/wordpress/tags/
  - **`docker pull wordpress:4.9`**
  - but better use Docker image digest
  - **`wordpress@sha256:18970d888d5395e1ba5c23dea a9f23c35cfbc241078d2dbba5db4177414cc9d2`**
- Environment variables
  - `WORDPRESS_DB_HOST`
  - `WORDPRESS_DB_USER`
  - `WORDPRESS_DB_PASSWORD`
  - `WORDPRESS_DB_NAME`

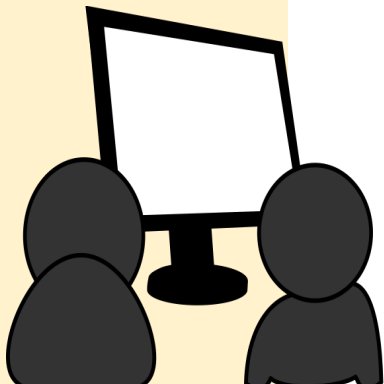
K8s - Webapp
Deployment - Manifest

```
# k8s-manifests/wordpress-06.yaml (truncated)

kind: Deployment
spec:
  selector:
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.9
        name: wordpress
        env:
```



K8s - Webapp
Deployment - Creation

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-06.yaml
deployment.apps/wordpress-mysql unchanged
persistentvolumeclaim/wp-mysql-data unchanged
persistentvolume/wp-mysq-data unchanged
service/wordpress-mysql unchanged
deployment.apps/wordpress created

kubectl get deploy
kubectl get rs
kubectl get pods
```

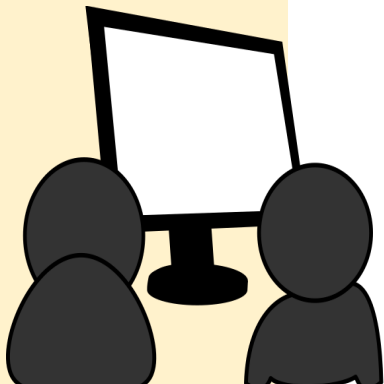


K8s - Webapp
Volume

- StorageClass “local-host” (already present)
- PersistentVolume
- PersistentVolumeClaim
- Mount to Pod
  - /var/www/html

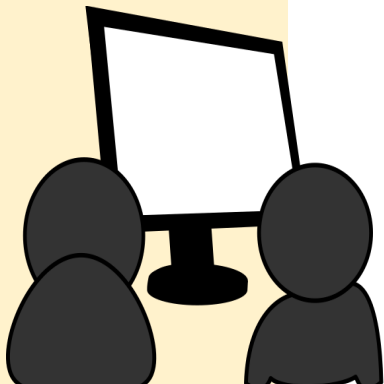
K8s - Webapp
Volume - Manifest

```
# k8s-manifests/wordpress-07.yaml (truncated)

kind: PersistentVolume
...
spec:
  local:
    path: /mnt/wp-www-data
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - w01-k8s-cluster01
```



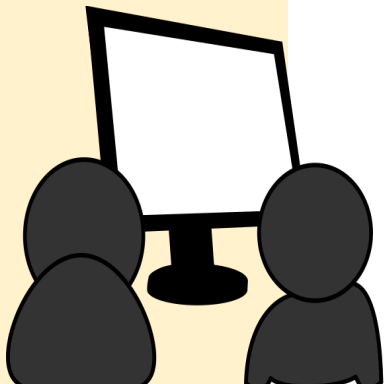
K8s - Webapp
Volume - Manifest

```
# k8s-manifests/wordpress-07.yaml (truncated)

kind: PersistentVolumeClaim
metadata:
  name: wp-www-data
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: local-storage
  resources:
    requests:
      storage: 5Gi
  selector:
    matchLabels:
      pv-name: wp-www-data
```



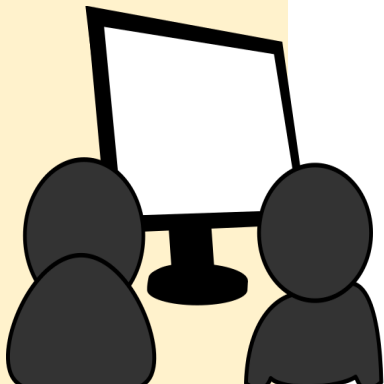
K8s - Webapp
Volume - Manifest

```
# k8s-manifests/wordpress-07.yaml (truncated)

      containers:
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      initContainers:
      volumes:
      - name: init-data
        hostPath:
          type: DirectoryOrCreate
          path: /mnt/wp-www-data
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-www-data
```



K8s - Webapp
Volume - Creation

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-07.yaml
...
deployment.apps/wordpress configured
persistentvolumeclaim/wp-www-data created
persistentvolume/wp-www-data created

kubectl get deploy
kubectl get rs
kubectl get pods
kubectl get pv
kubectl get pvc
```

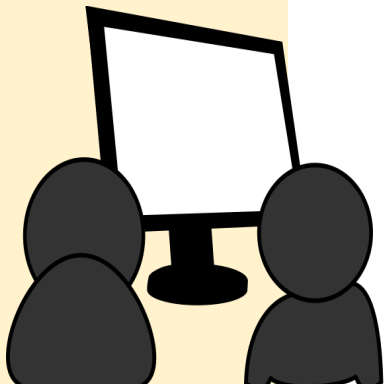


K8s - Webapp
Service - Nodeport

- publication of service at a port on all cluster nodes
  - can be chosen from defined ranged
    - 30000-32767
  - automatically by cluster
  - manual (manifest)

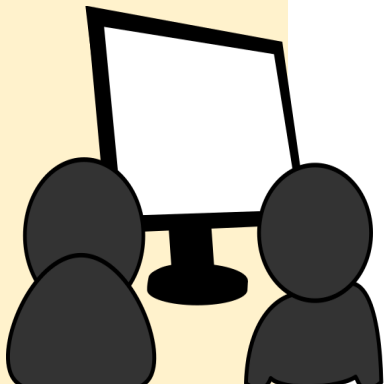
K8s - Webapp
Service - Manifest

```
# k8s-manifests/wordpress-08.yaml

kind: Service
spec:
  selector:
    app: wordpress
    tier: frontend
  type: NodePort
  ports:
  - targetPort: 80
    port: 80
    nodePort: 30080
    protocol: TCP
```



K8s - Webapp
Service - Create

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-08.yaml
...
service/wordpress created

$ kubectl get deploy
```

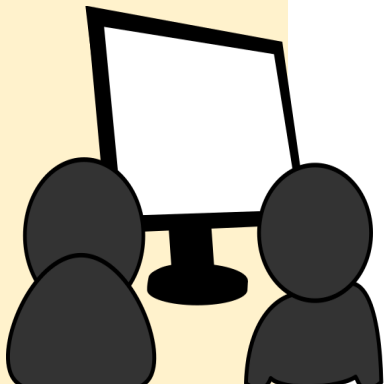


K8s - Webapp
Service - Create

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl get svc
NAME        TYPE       CLUSTER-IP       ... PORT(S)
wordpress   NodePort   10.110.128.245   ... 80:30080/TCP

$ open http://m01.k8s-cluster01.aemc.me:30080
$ open http://w01.k8s-cluster01.aemc.me:30080
$ open http://w02.k8s-cluster01.aemc.me:30080
```

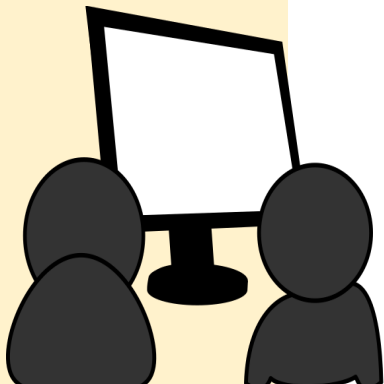
Change it

```
# !!! DO NOT INSTALL WORDPRESS NOW !!!
```



K8s - Webapp
Service - Delete

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl delete svc wordpress
```



K8s - Webapp
Current State

- Pro
  - create containerized processes (pods)
  - reliable, redundant
  - scalable
  - (update)
  - persistence
  - stable internal access to pods, (load balancing)
- Con
  - stable external access
  - Routing on url’s

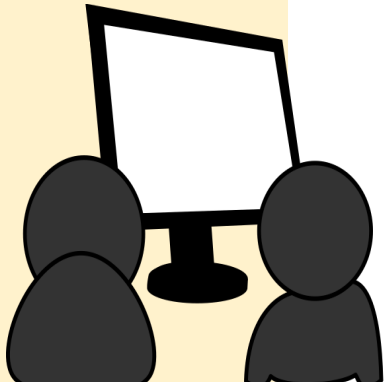
K8s - Webapp
Ingress - What is it?

- k8s-object, but not any "internal" implementation (necessary to install one)
  - examples: **Traefik**, Nginx, HA-Proxy, Istio/Envoy, LinkerD, ...
- frontend for services, some kind of “reverse proxy”
- Ingress Controller, Ingress Rules
- feature set depend on chosen implementation

K8s - Webapp
Ingress - Manifest

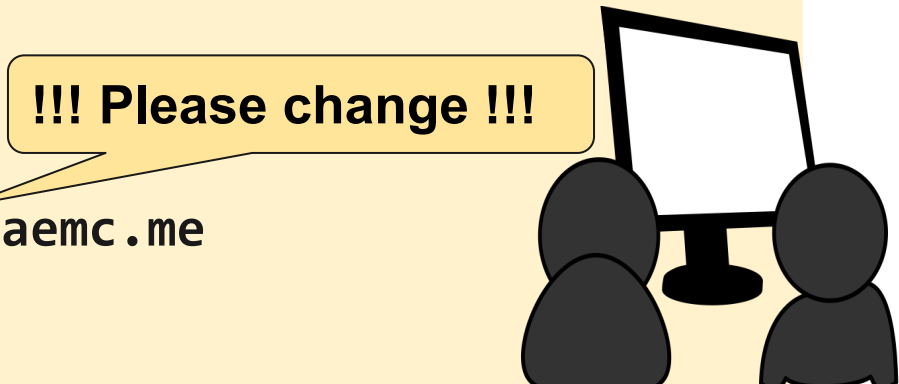
```
# k8s-manifests/wordpress-09.yaml (truncated)

kind: Ingress
metadata:
  name: wordpress-ui
  annotations: kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: wp.k8s-cluster01.aemc.me
    http:
      paths:
      - path: /
        backend:
          serviceName: wordpress
          servicePort: 80
```

!!! Please change !!!



K8s - Webapp
Ingress - Create

```
ubuntu@m01-k8s-cluster01:~/workshop/demo-k8s-cluster$

$ kubectl apply -f k8s-manifests/wordpress-09.yaml
...
ingress.extensions/wordpress-ui created

$ kubectl get ingresses
NAME         HOSTS                        PORTS
wordpress-ui wp.k8s-cluster01.aemc.me 80
```

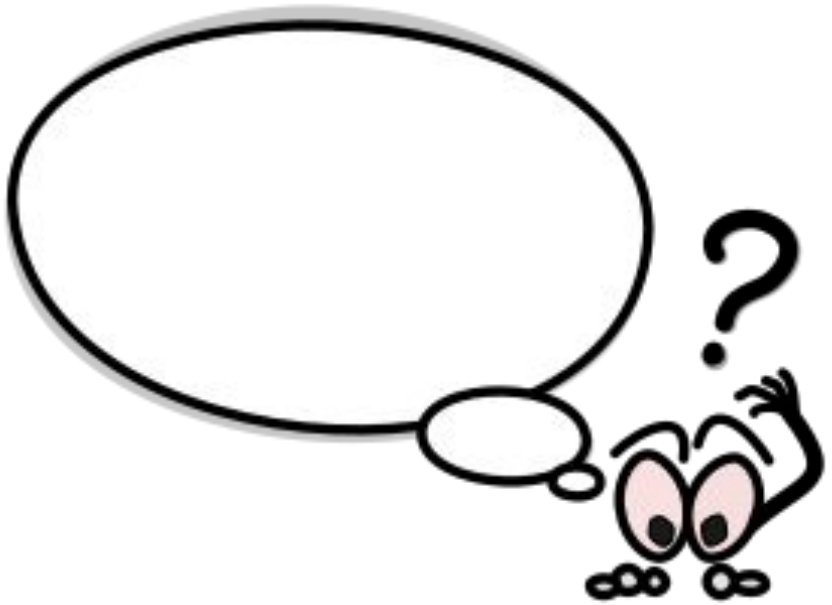



```
$ open http://wp.k8s-cluster01.aemc.me
```

K8s - Webapp

K8s - Webapp

https://cdn.pixabay.com/photo/2015/01/08/18/24/children-593313_960_720.jpg



Questions …?

K8s - Webapp

This is the last slide ...