Kubernetes Workshop

The Basics

Ansgar Sachs

Senior Software Developer
CGM Software GmbH

ansgar.sa@gmail.com

What do you already know and what's your goal of this workshop?

JAY ESDOT

Knows about Docker but has never used Kubernetes

JAYLINA TEDOT

Has seen Docker a while ago but isn't really comfortable with it

Goal of this workshop

Build a small

Monolith

Database

Decompose the Monolith into

Services

Database

Move these Services into Containers and let them be orchestrated by

Kubernetes

Database

Database


YOU GET A CLUSTER AND YOU GET A CLUSTER
EVERYBODY GETS A CLUSTER
imgflip.com

Build Pairs

If you are already experienced, then search for someone with less knowledge about containers and vice versa

Check out:

https://github.com/ansgars/kubernetes-workshop

But before...

Here are some Spring fundamentals you should know

Spring Basics 1

```
@SpringBootApplication
public class MyAwesomeApp {

    public static void main(String[] args) {
        SpringApplication.run(MyAwesomeApp.class, args);
    }

    @Bean(name = "someService")
    public SomeService someService() {
        return new SomeService();
    }
}
```

Spring Basics 2

```
@RestController
@RequestMapping(value = "/iamroute")
public class SomeController {
  Logger log = LoggerFactory.getLogger(SomeController.class);

  @Resource(name = "someService")
  private SomeService someService;

  @PostMapping(produces = {"application/json"})
  public ResponseEntity<Sth> addSth(@RequestBody Blub blub) {
    // ...
    return new ResponseEntity<>(blob, HttpStatus.CREATED);
  }
```

Task 1

Build a monolithic spring app, that

- accepts new patients (id, firstName, lastName, age)
- accepts new prescriptions (id, name, dose, price)
- returns the cumulated costs of a patient's prescriptions
- use an in memory data store

Use a client such as Postman to validate your application **and** run the integration tests

```
cd integration-tests && mvn clean
verify
```

Served Routes

```
POST /patients {
 "firstName": "...",
 "lastName": "...",
 "age": 123
}

Returns 201 Created { "id":  "..." }
```

```
GET /patients/<id>/costs

Returns 200 OK {
  "patientId": "...",
  "costs": 45.27
}
Returns 404 Not found if patient does not
exist
```

```
POST /prescriptions {
 "patientId": "...",
 "name": "...",
 "price": 12.12
}

Returns 201 Created { "id": "..." }
Returns 422 if patient doesn't exist
```

```
GET /patients

Returns 200 OK {
  "patients": ["id1", "id2"]
}
```

Task 1: Result

If you could not finish your task

- checkout task1/final
- run the integration tests once

Questions

- Did you create different models for the write and the read model?
- How did you manage your singleton in memory state?
- How did you cluster your controllers and actions?

01

Baseline

Let's create something to iterate on

02

Docker Fundamentals

Overview of the most important components

03

Kubernetes 101

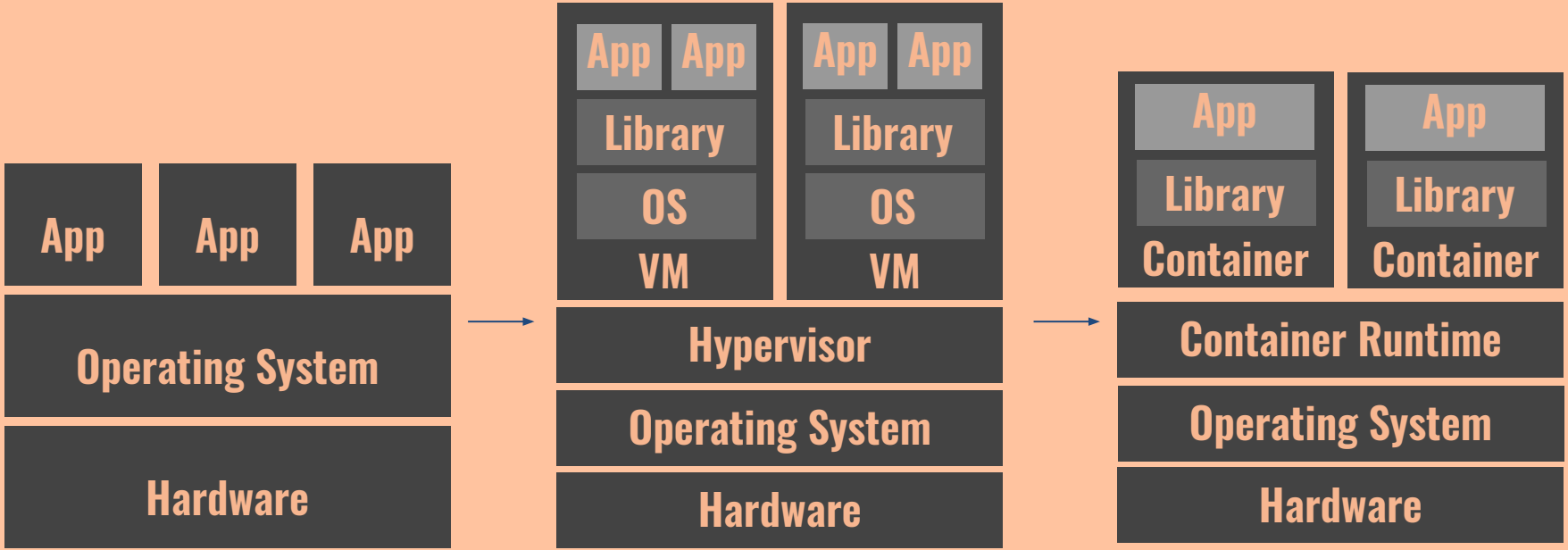
What does it take to create a mini application?

04

Discussion

I understand that this topic is complex. What are the next steps?

A brief History of Deployments



Traditional Deployment

Virtualized Deployment

Container Deployment

Docker

A platform to separate your application from your infrastructure

- package and run applications in a loosely isolated environment (container)
- The container becomes the unit for distributing and testing your application
- The isolation and security allow you to run many containers simultaneously on a given host
- A registry keeps track of your application

Container, Docker and everything else...

Dockerfile

```
FROM SomeBaseImage

RUN someShellCommand

ENV SOME_ENV SOME_VALUE

ADD someFile
Copy someOtherFile

EXPOSE 9999

CMD [“Some Start Command”]
```

Container, Docker and everything else...

Docker Client

`docker build -t maintainer/imageName:tag .`	Build a Docker image
`docker pull maintainer/imageName:tag`	Pull a Docker image
`docker run -d -p 8000:8000 --name=portainer \ -v host:container maintainer/image`	Create and run a Docker Container

docker-compose.yml

```yaml
version: "3.7"

services:
 some-service:
   build:
     context: ./service-directory
   ports:
   - "9999:9999"
   networks:
     - a-nerwork
   volumes:
     - a_volume:/container/path
   environment:
     - POSTGRES_USER=cgmuser
...
```

docker-compose.yml

```yaml
...

volumes:
 a_volume:

networks:
 some-network:
   driver: bridge
```

Container, Docker and everything else...

Containers, Docker and everything else...

	Docker Compose Client
Build a Docker Image	docker-compose build service-name
Pull a Docker Image	docker-compose pull service-name
Create and run a Docker Container	docker-compose up -d service-name
Shutdown composed cluster	docker-compose down

Task 2

Split up the monolithic spring app and containerize it in a composed platform

- first spring app accepts new patients and prescriptions
- second spring app returns a list of all patients or a list of all prescriptions for a given patient
- all apps are connected to a postgres database

Note: Merge **task2/starter** into your current branch

Use a client such as Postman to validate your application or run the integration tests

Inside Kubernetes 03.

A brief story about the Architecture

Kubernetes does the things that the very best system administrator would do: automation, failover, centralized logging, monitoring. It takes what we've learned in the devops community and makes it the default, out of the box.

— **KELSEY HIGHTOWER**

Why Kubernetes?

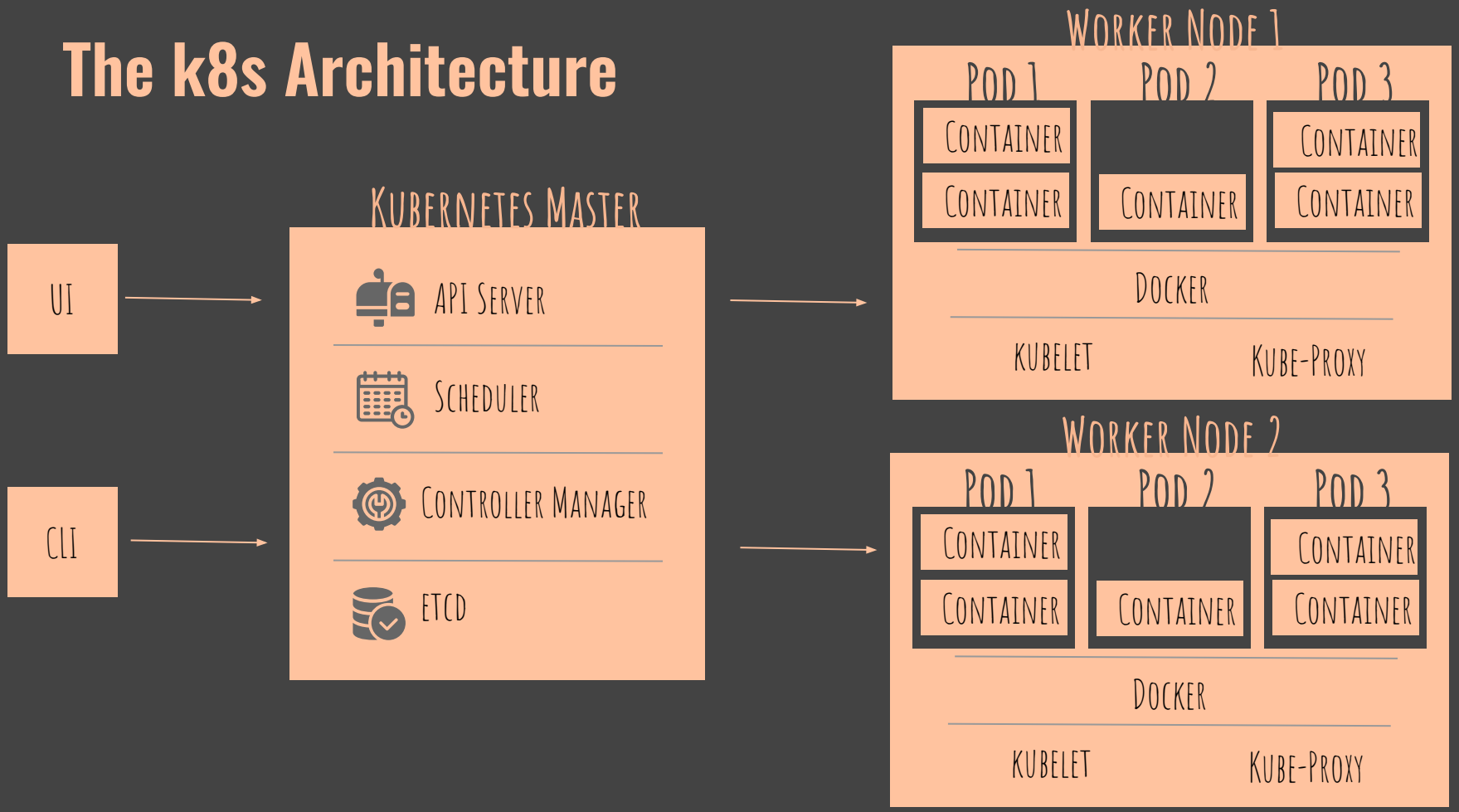
Organizations that operate on a massive scale require:

- an architecture that scales with the usage
- an architecture that scales with development teams
- an architecture that supports ongoing operations
- an architecture that supports automated development cycles

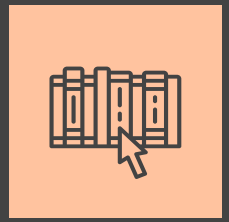
" More than 75% of global organizations will be running containerized applications in production by 2022 "

- *see Gartner*

The k8s Architecture



The two big players



Kubernetes Master

Responsible for exposing the application program interface (API)
Scheduled the deployments
Manages the overall cluster

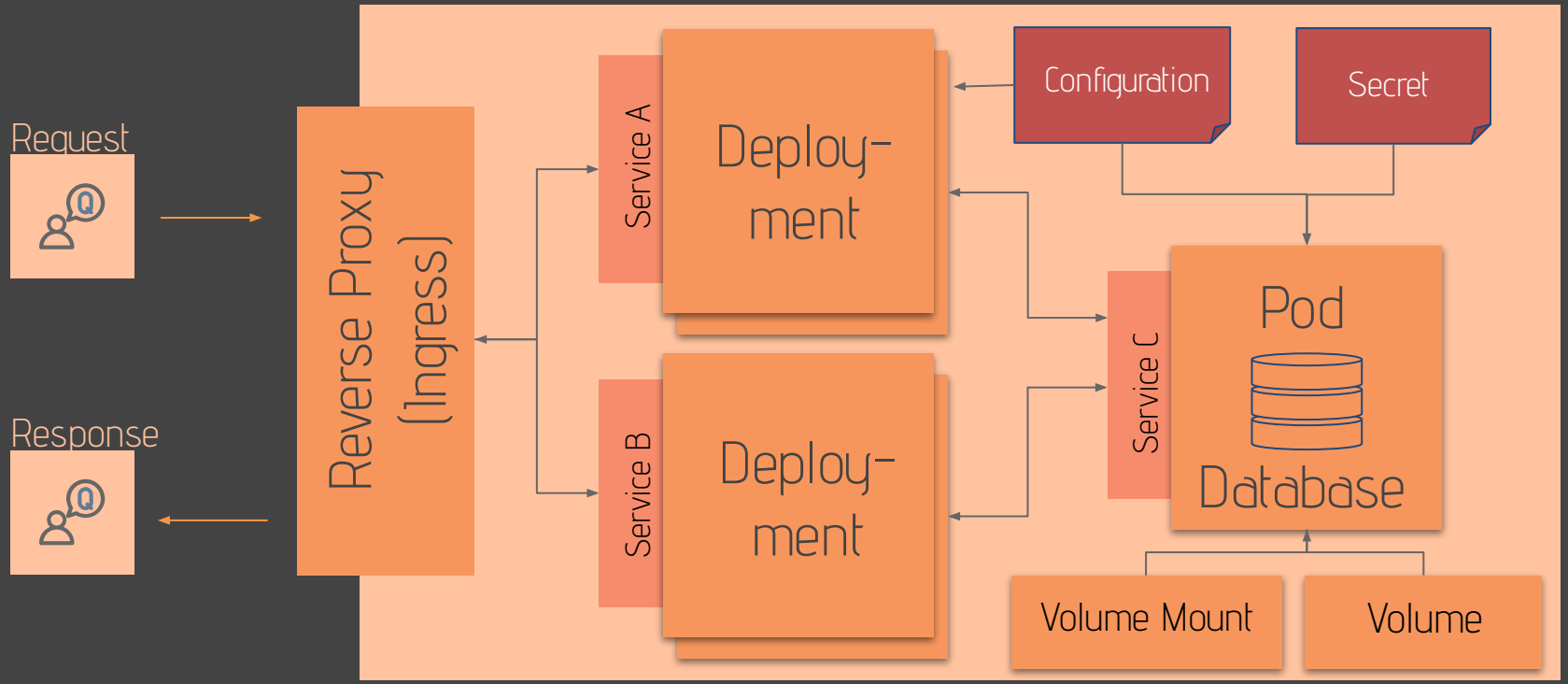
Node

The Workhorses of a Kubernetes Cluster
Expose compute, networking and storage resources to applications
Communicates via an agent with the Master

Kubernetes Components

A rush through the ecosystem

Let's start with an example



```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
 name: test-ingress
 annotations:
  nginx.ingress.kubernetes.io/rewrite-target:/$2
spec:
 rules:
   - host: some-url.info
     http:
       paths:
         - path: /testpath(/|$)(.*)
           backend:
             serviceName: test
             servicePort: 9999
```

Ingress

An API object that manages external access to the services in a cluster, typically HTTP

Further Reading:
https://kubernetes.io/docs/concepts/services-networking/ingress/

```yaml
apiVersion: v1
kind: Service
metadata:
  name: a-service-name
  namespace: default
  labels:
    app: the-service-label
spec:
  ports:
    - port: 9999
      protocol: TCP
  selector:
    app: the-pod-label
```

Service

An abstract way to expose an application running on a set of Pods as a network service.

Further Reading:
https://kubernetes.io/docs/concepts/services-networking/service/

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deployment-name
  namespace: default
  labels:
    app: a-label-I-can-select
spec:
  replicas: 1
  selector:
    matchLabels:
      app: the-selectable-label
  template:
    metadata:
      labels:
        app: give-me-the-label-again
```

Pod

Pods are the smallest deployable units of computing that can be created and managed in Kubernetes.

Deployment

A Deployment provides declarative updates for Pods and ReplicaSets.

Replica Set

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time.

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

```yaml
        app: give-me-the-label-again
    spec:
      containers:
        - name: how-i-call-the-pod
          image: docker-image-name:<tag>
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 9999
          env:
            - name: SUPER_SECRET_ENV
              valueFrom:
                secretKeyRef:
                  name: a-secret-name
                  key: key-in-secret
```

Pod

Pods are the smallest deployable units of computing that can be created and managed in Kubernetes.

Deployment

A Deployment provides declarative updates for Pods and ReplicaSets.

Replica Set

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time.

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

```yaml
          volumeMounts:
            - mountPath: /some/path
              name: volume-name
            - mountPath: some-file.txt
              name: other-volume-name
              readOnly: true
      volumes:
        - name: volume-name
          persistentVolumeClaim:
            claimName: volume-claim-name
        - name: other-volume-name
          configMap:
            name: configmap-name
```

Pod

Pods are the smallest deployable units of computing that can be created and managed in Kubernetes.

Deployment

A Deployment provides declarative updates for Pods and ReplicaSets.

Replica Set

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time.

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

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: some-configmap-name
  namespace: default
data:
  somefile.sql: |
    \connect someDatabaseName;

    CREATE TABLE blub (
        id          varchar(255) PRIMARY KEY,
        yolo         varchar(255) NOT NULL
    );
```

ConfigMap

A ConfigMap is an API object used to store non-confidential data in key-value pairs. Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume.

Further Reading:
https://kubernetes.io/docs/concepts/configuration/configmap/

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: some-secret-name
type: Opaque
data:
  dbname: cGF0aWVudGRi
  username: Y2dtdXNlcg==
  password: Y2dtcGFzc3dvcmQ=
```

Values are Base64 encoded

```
# Encode in Base64
echo -n 'username' | base64
# Decode from Base64
echo -n 'dXNlcm5hbWU=' | base64 --decode
```

Secret

Kubernetes Secrets let you store and manage sensitive information, such as passwords, OAuth tokens, and ssh keys.

Further Reading:
https://kubernetes.io/docs/concepts/configuration/secret/

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: some-volume-claim-name
  labels:
    app: some-claim-label
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1337Gi
```

There are many, many more types of volumes

Volumes

At its core, a volume is just a directory, possibly with some data in it, which is accessible to the Containers in a Pod.

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

Kubernetes 101

Kubernetes Client

Deploy Files	kubectl apply -f /path/to/directory
Undeploy Files	kubectl delete -f /path/to/directory
Get deployed Files	kubectl get deployments services ingress pvc pods secrets configmaps
Inspect Files	kubectl describe deployment <name> ....
Get logs	kubectl logs -f <pod-name>

Task 3

Deploy all your previous built docker images
in Kubernetes

- Add content to your kubernetes files in the k8s directory
- Recreate your database with a deployment, a volume, a secret, a configMap and a service
- Recreate your services with a deployment and a service file and the previous used secret file
- Use an ingress to route your traffic
- Update all application properties to the new routings

Note: Merge **task3/starter** into your current branch

Use a client such as Postman to validate your application and run the integration tests

Discussion 04.

The State of Play

SUGGESTIONS

JAY ESDOT

Which parts of this workshop can be reused for my work environment?
Did this workshop support me for future projects?

JAYLINA TEDOT

Where can I continue learning and am I now able to create a production ready cluster?

THANKS!

Does anyone have any questions?

ansgarsa@gmail.com
https://www.cgm.com/de/index.de.jsp

RESOURCES

The content of this presentation was stolen from

- https://kubernetes.io/docs/concepts/
- https://docs.docker.com/get-started/overview/
- https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html
- Thousands of youtube tech talks
- my brain

CREDITS

The following awesome resources have been used to create this presentation

- Presentation template by Slidesgo
- Icons by Flaticon
- Images & infographics by Freepik
- Author introduction slide photo created by Freepik
- Text & Image slide photo created by Freepik.com
- Big image slide photo created by Freepik.com