

# **DEEPSHORE**

## **DEEPTALK**

**Von der Application  
zum Kubernetes-Deployment**

2020

## Agenda

Komponenten

Containerisierung

Kubernetes-Ressourcen

Zusammenfassung

## Recap

- Pods
- Services
- ReplicaSets
- Deployments
- Manifeste („yaml-Baupläne“)

→ Folge 2 auf YouTube:

<https://www.youtube.com/watch?v=2cK7K2uYtaI>

## Auswahl des Themas: Motivation

### **Wie kommt meine Applikation in den k8s-Cluster?**

- Vorstellung unserer Antwort, Best Practices und Einblick in unsere Arbeit bei Deepshore
- Andere motivieren: es sind einige Schritte bis in den k8s-Cluster - wenn man aber weiß, wie es geht, ist es nicht schwer
- Vorteile von k8s nutzbar machen: komfortables Skalieren, Verteilung der Last auf mehrere Nodes, effiziente Nutzung von Ressourcen, ...

## Komponenten



### Flask

- Leichtgewichtiges Framework für Web-Applikationen
- Python



### InfluxDB

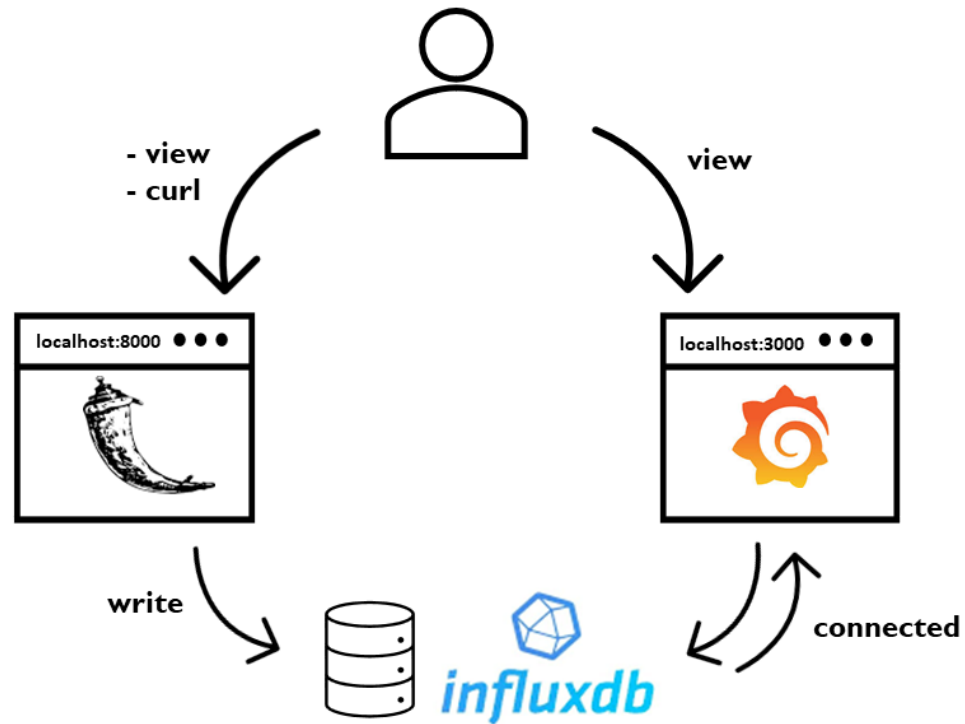
- Zeitreihenbasierte Datenbank (time series database)
- HTTP API für Client/Server-Kommunikation



### Grafana

- Plattformübergreifende Webanwendung
- Analyse und interaktive Visualisierung von (Monitoring-)Daten

## Architektur



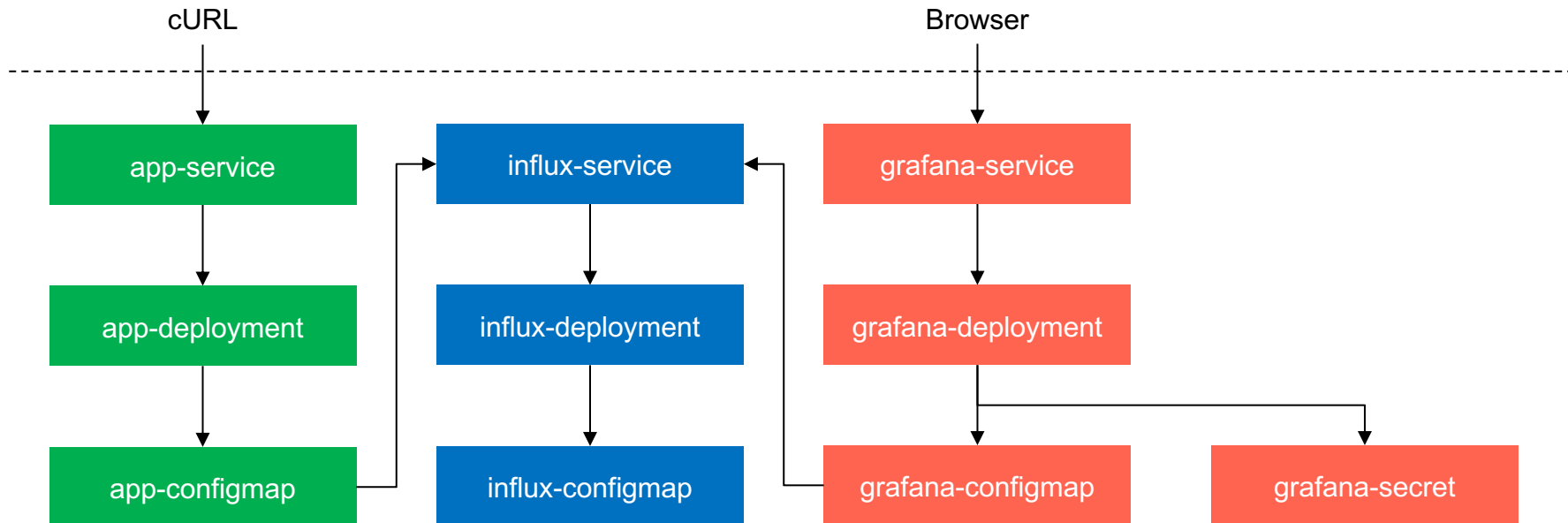
## Vorbereitung für k8s

Voraussetzung: containerisierte Anwendung ist für den Kubernetes-Cluster bzw. Docker auf den Nodes zugänglich

Bereitstellung von Images über Registries:

- public: z.B. Docker Hub (default), Docker Cloud
- private: z.B. Registries in Gitlab

## Overview: k8s-Resources





## ConfigMaps und Secrets

### ConfigMaps

- „decouple environment-specific configuration from your [container images](#)“
- Key-Value-Paare
- Umgebungsvariablen, CLI-Argumente, Konfigurationsdateien

### Secrets

- „store and manage sensitive information“
- Key-Value-Paare (Value: base64-encoded)
- Passwörter, OAuth-Tokens, SSH-Keys

## Zusammenhang zwischen k8s-Objekten einer Komponente

Beispiel: deeptalk-app

### **Services**

- LabelSelector → Pods

### **Deployment**

- LabelSelector → Pods
- LabelSelector des Deployments entspricht LabelSelector des Services
- Name → ConfigMap

### **ConfigMap**

- Name → influxdb-service (INFLUX\_HOST)

## Effektives Erzeugen der Manifeste

### Deployments

```
kubectl create deployment nginx --image=nginx --dry-run -o yaml > deployment.yaml
```

### Services

```
kubectl expose deployment nginx --port=80 --dry-run -o yaml > service.yaml
```

(wenn Deployment existiert)

### ConfigMaps

```
kubectl create configmap user --from-literal=USER=user --dry-run -o yaml > cm.yaml
```

## Installation

1. Secret für Grafana erzeugen
2. `kubectl create -f .`
3. Funktionalität checken

## Takeaways

- Schritte in den k8s-Cluster:
  - Architektur der App muss geeignet sein (Microservices)
  - Containerisierung
  - Bereitstellung der Container
  - Ressourcen definieren
- Best Practices vereinfachen Entwicklung von Manifesten
- Grafana kann komfortabel konfiguriert werden (ConfigMap)

## Misc

### Materialien zum Talk

<https://github.com/grothesk/deeptalk>

### Weiterführendes

Celery → Asynchron in Python: <https://docs.celeryproject.org>

Django → Alternative zu Flask: <https://www.djangoproject.com>

Skaffold → Entwicklung von k8s-Anwendungen: <https://skaffold.dev>

### Feedback , Anregungen, Themenvorschläge

[florian.boldt@deepshore.de](mailto:florian.boldt@deepshore.de)

[malte.groth@deepshore.de](mailto:malte.groth@deepshore.de)

[frederic.born@deepshore.de](mailto:frederic.born@deepshore.de)

# DEEPSHORE

Vielen Dank.