Realisierung einer CI/CD-Pipeline für container-basierte verteilte Anwendungen

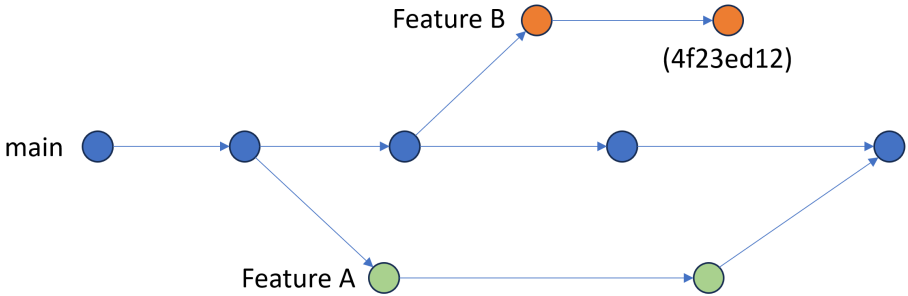
Status Quo, Herausforderungen, Realisierung am konkreten Beispiel

Dr.-Ing. Stephan Schneider

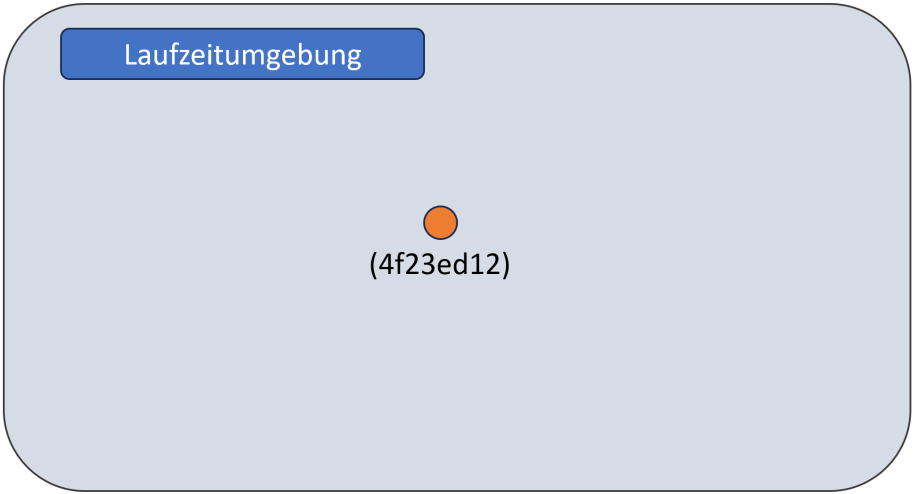
Dezember 18, 2024

Agenda

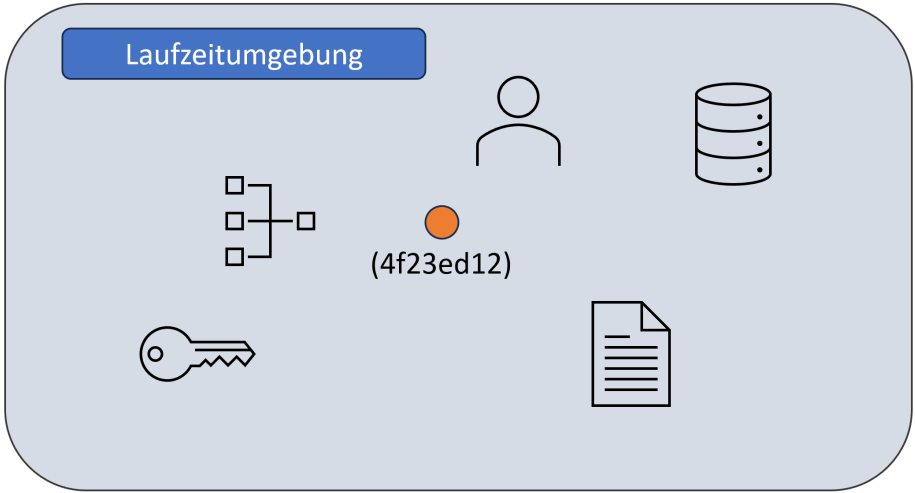
Beispiel



Beispiel



Beispiel



Was sind Container?

Definition (Container)

Eine Laufzeitumgebung einer Anwendung, mit allen notwendigen Abhängigkeiten.

Definition (Image)

Ein Image stellt die statische Abbildung des Dateisystems dar.

Container-basierte Anwendungen

Was sind Container?

- Leichtgewichtige Methoden zur Isolation von Dateisystemen, CPU- und Speicherressourcen, System- und Netzwerregeln
- Bieten jedoch eine geringere Isolation als virtuelle Systeme

Vorteile:

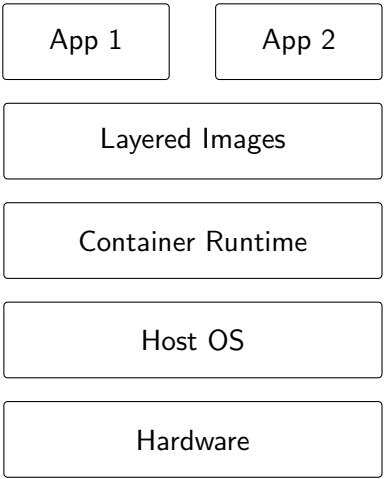
- Geringer Overhead im Vergleich zu VMs
- Keine Hardware-Emulation
- Direkter Zugriff auf Ressourcen des Hostsystems (GPU, Netzwerk, Storage)

Nachteile:

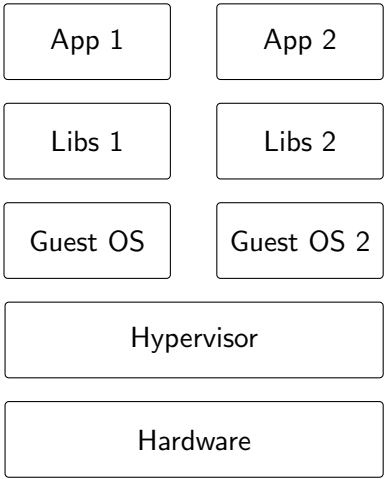
- Kein abweichendes OS möglich
- Keine Änderung der Kernelkonfiguration
- Schwache Isolation

Container vs. Virtuelle Maschine

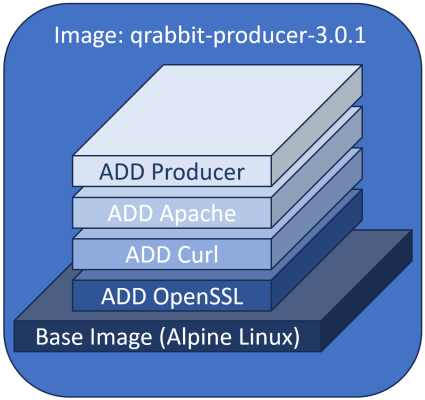
Container:

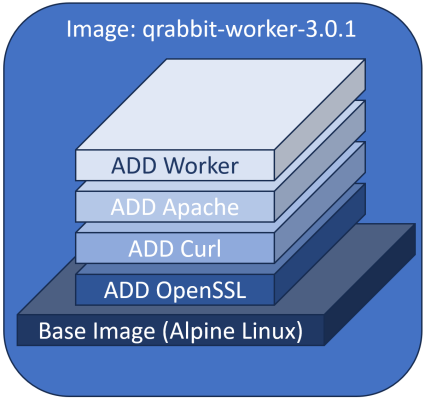


VM:



Layered Images





Docker

- **1979** UNIX : *chroot*
- **1999** BSD Jail
- **2000** Linux virtual environment
- **2004** Linux BSD Jail / Solaris Zones
- **2005** Linux OpenVZ
- **2008** Linux Containers (LXC)
- **2013** Docker (bis 2014 ein Wrapper für LXC)

Docker

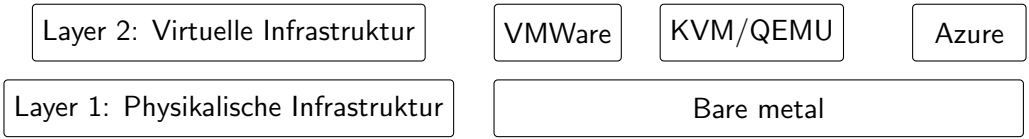
- **1979** UNIX : *chroot*
- **1999** BSD Jail
- **2000** Linux virtual environment
- **2004** Linux BSD Jail / Solaris Zones
- **2005** Linux OpenVZ
- **2008** Linux Containers (LXC)
- **2013** Docker (bis 2014 ein Wrapper für LXC)

Container sind keine neue Technologie und Docker ist nicht die einzige Container-Runtime.

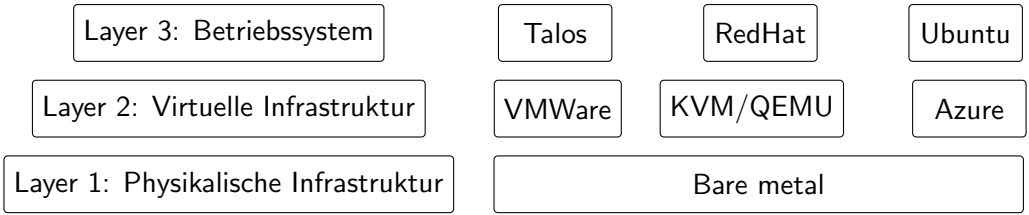
Multilayer-Architektur

Layer 1: Physikalische Infrastruktur	Bare metal

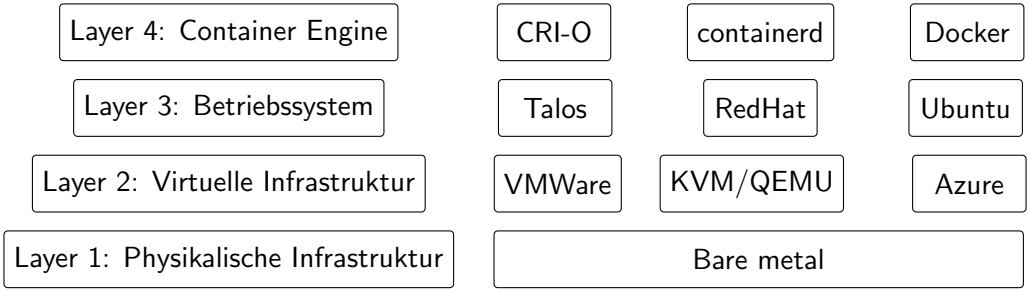
Multilayer-Architektur



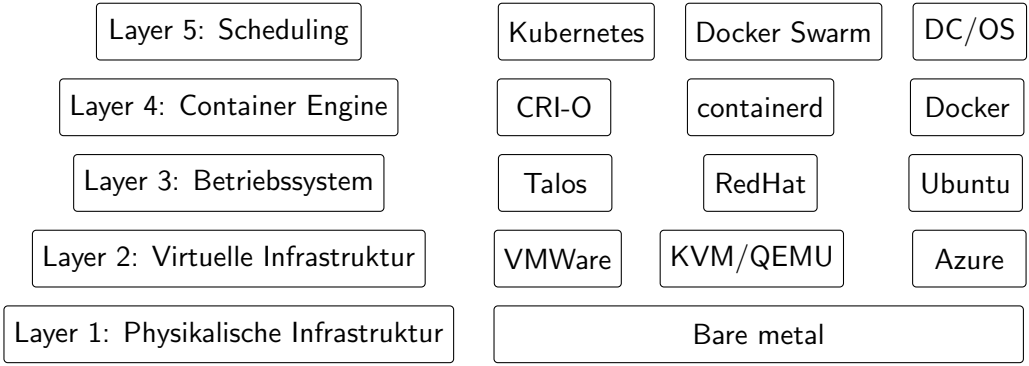
Multilayer-Architektur



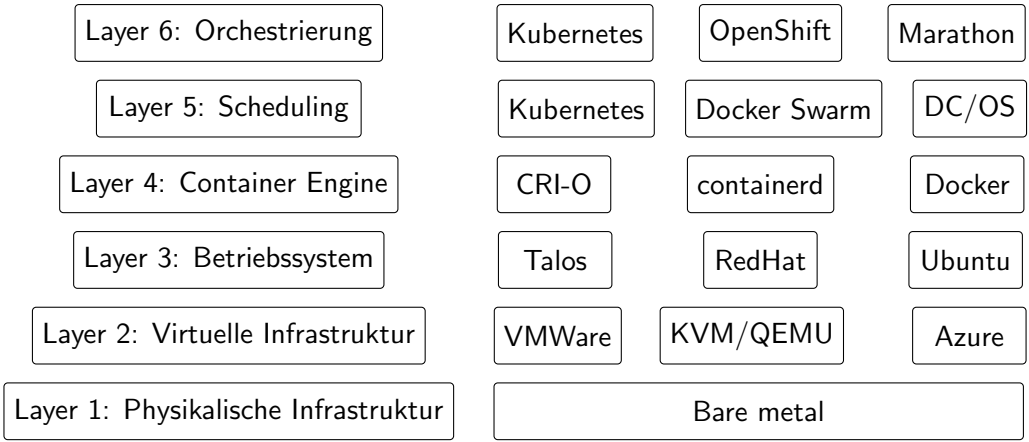
Multilayer-Architektur



Multilayer-Architektur



Multilayer-Architektur



CI/CD

Definition

CI : engl. *Continious Integration*
CD : engl. *Continious Delivery* oder *Continious Deploy*

CI/CD

Definition

CI : engl. *Continious Integration*
CD : engl. *Continious Delivery* oder *Continious Deploy*

Was soll kontinuierlich integriert werden?

CI/CD

Definition

CI : engl. *Continious Integration*
CD : engl. *Continious Delivery* oder *Continious Deploy*

Was soll kontinuierlich integriert werden? **Softwareanpassungen!**

CI/CD

Definition

CI : engl. *Continious Integration*
CD : engl. *Continious Delivery* oder *Continious Deploy*

Was soll kontinuierlich integriert werden? **Softwareanpassungen!**
Wo liegt der Unterschied zwischen *Delivery* und *Deploy*?

CI/CD

Definition

CI : engl. *Continious Integration*
CD : engl. *Continious Delivery* oder *Continious Deploy*

Was soll kontinuierlich integriert werden? **Softwareanpassungen!**
Wo liegt der Unterschied zwischen *Delivery* und *Deploy*?
Beim Continious Delivery findet die Aktivierung im Produktivsystem manuell statt.

Verfügbare CI/CD-Lösungen

- Jenkins
- Gitlab
- Travis-CI
- TeamCity
- CircleCI
- (uvm.)

Jede dieser Anwendungen besitze Stärken und Schwächen die im jeweiligen Projektkontext bewertet werden müssen.
Die nachfolgenden Beispiele wurden mit GitLab und dessen CI/CD-Möglichkeiten erstellt.

Beispielanwendung

- Die Beispielanwendung besteht aus drei Java-Microservices

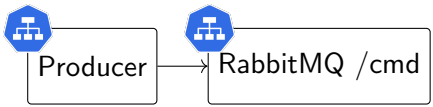
Beispielanwendung

- Die Beispielanwendung besteht aus drei Java-Microservices
- **Producer** erstellt Arbeitsaufträge und sendet diese an eine asynchrone Warteschlange.
  - **WAIT**: Kommando bewirkt einen Wartezyklus von 5 Sekunden im Worker
  - **NOOP**: Kommando bewirkt eine Ausführung eines Nullbefehls
  - **FIBONACCI**: Kommando bewirkt die Bereichnung einer Fibonacci-Zahl im Worker
  - **KILL**: Kommando bewirkt, dass der Worker beendet wird.

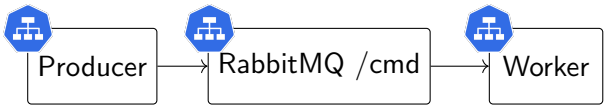
Beispielanwendung

- Die Beispielanwendung besteht aus drei Java-Microservices
- **Producer** erstellt Arbeitsaufträge und sendet diese an eine asynchrone Warteschlange.
  - **WAIT**: Kommando bewirkt einen Wartezyklus von 5 Sekunden im Worker
  - **NOOP**: Kommando bewirkt eine Ausführung eines Nullbefehls
  - **FIBONACCI**: Kommando bewirkt die Bereichnung einer Fibonacci-Zahl im Worker
  - **KILL**: Kommando bewirkt, dass der Worker beendet wird.
- **Worker** nimmt die Kommandos aus der Warteschlange und bearbeitet diese
  - Abgearbeitete Kommandos werden in die Warteschlage **results** eingestellt
- **Writer** Der Prozess nimmt die Ergebnisse aus der Warteschlange **results** und persistiert diese.

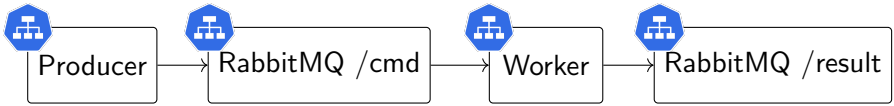
Beispielanwendung



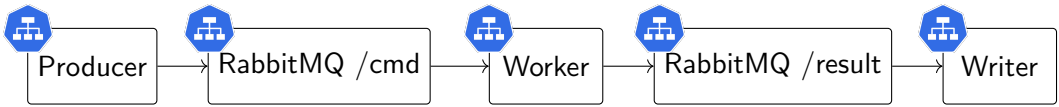
Beispielanwendung



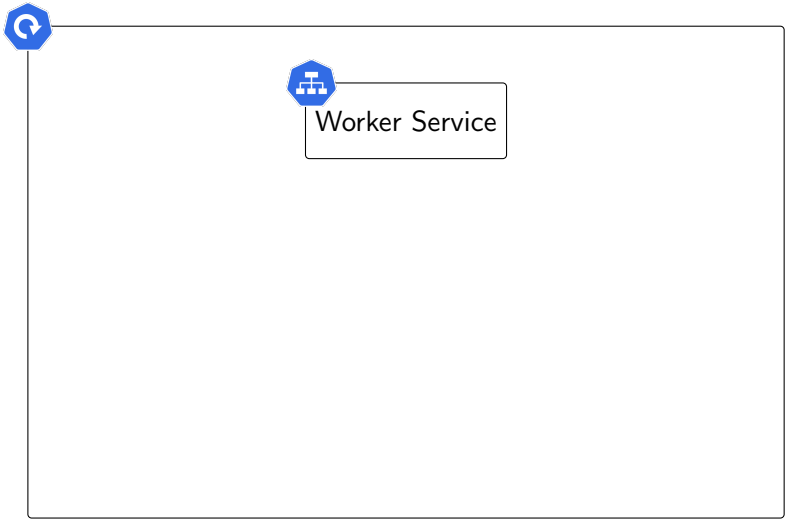
Beispielanwendung



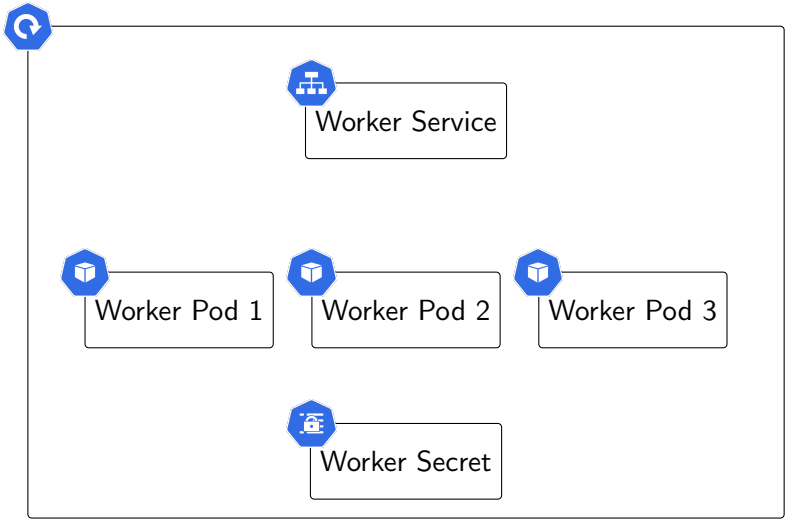
Beispielanwendung



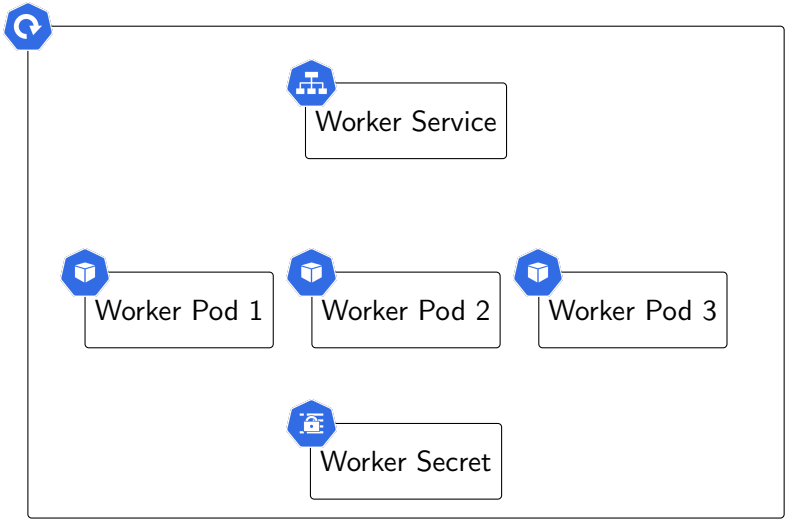
Kubernetes Deployment



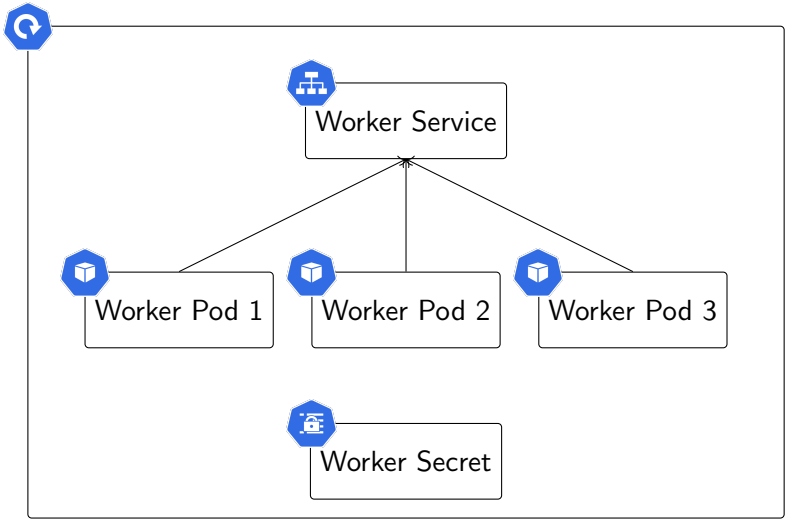
Kubernetes Deployment



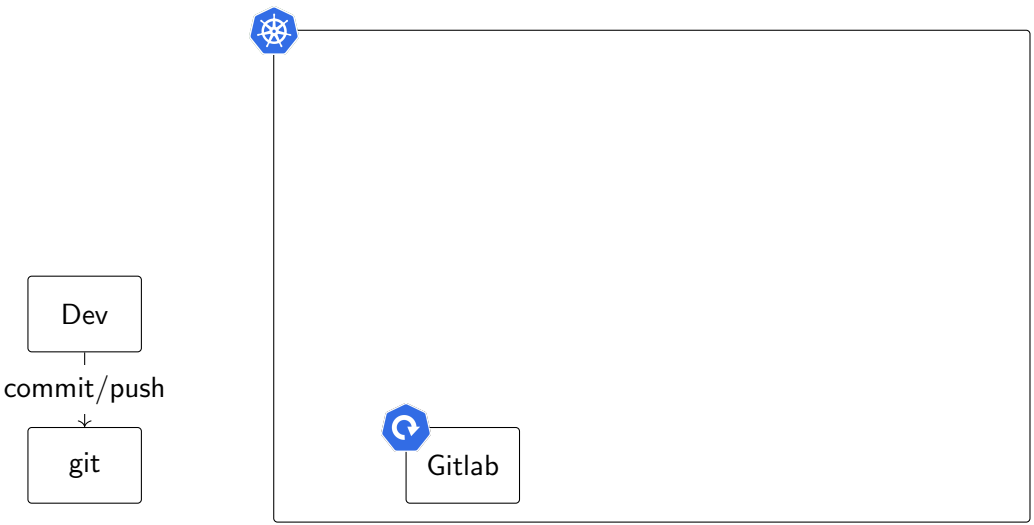
Kubernetes Deployment



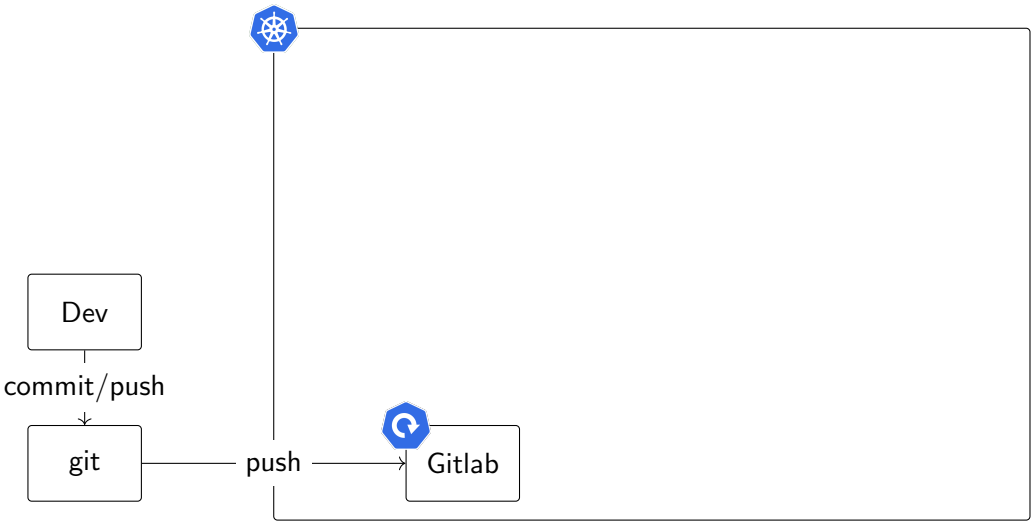
Kubernetes Deployment



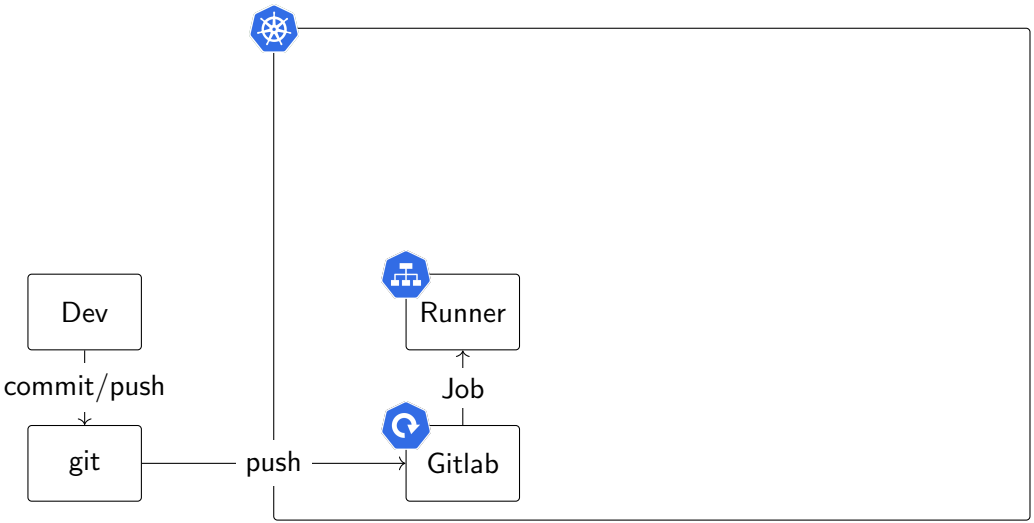
Übersicht Kubernetes



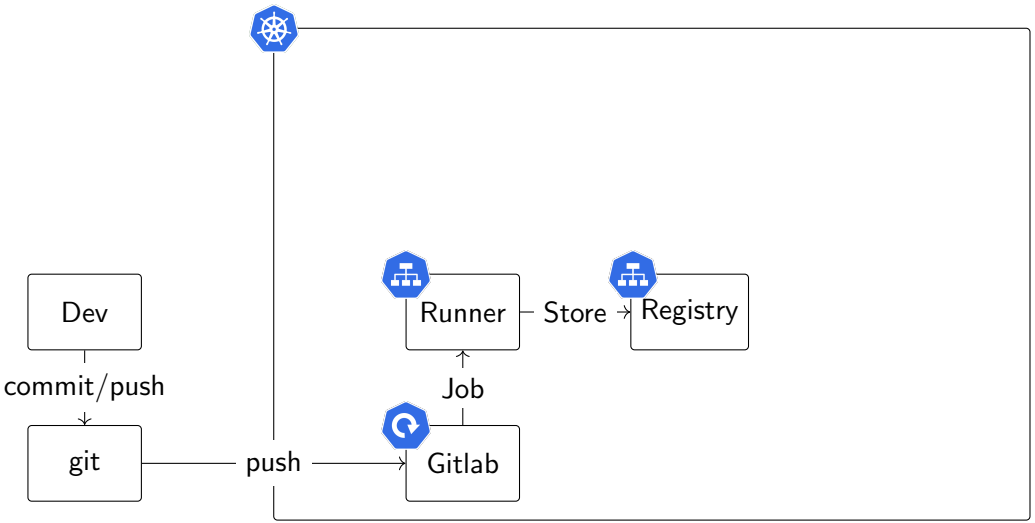
Übersicht Kubernetes



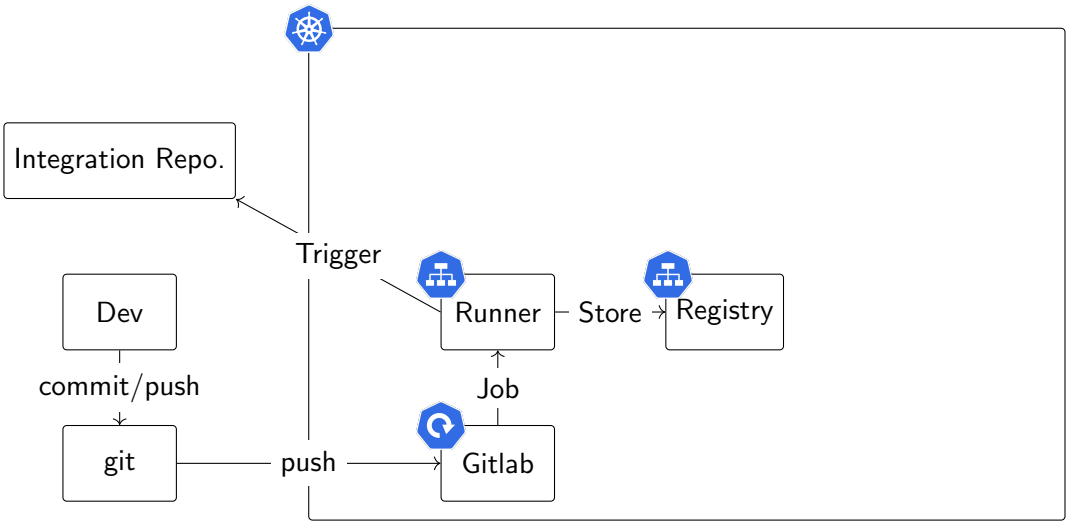
Übersicht Kubernetes



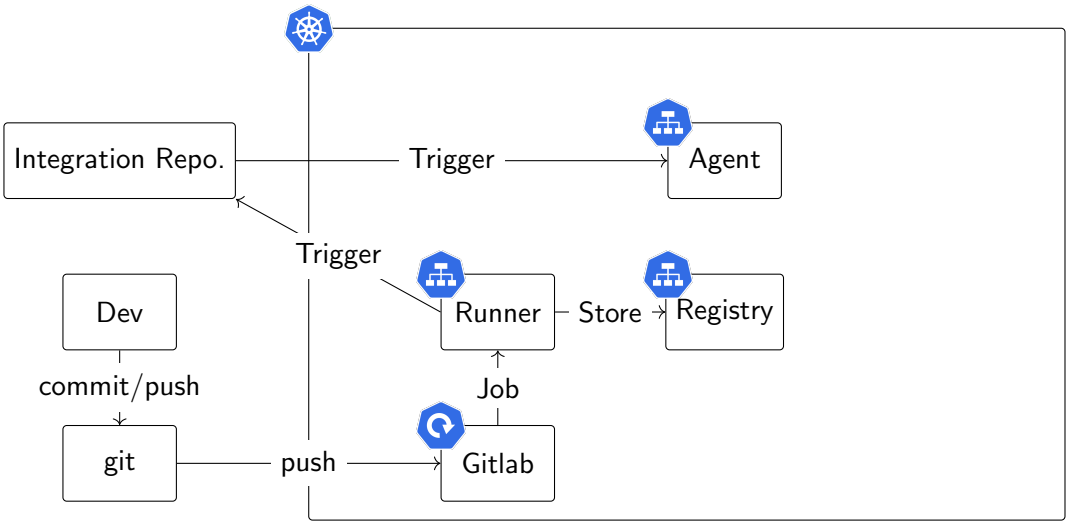
Übersicht Kubernetes



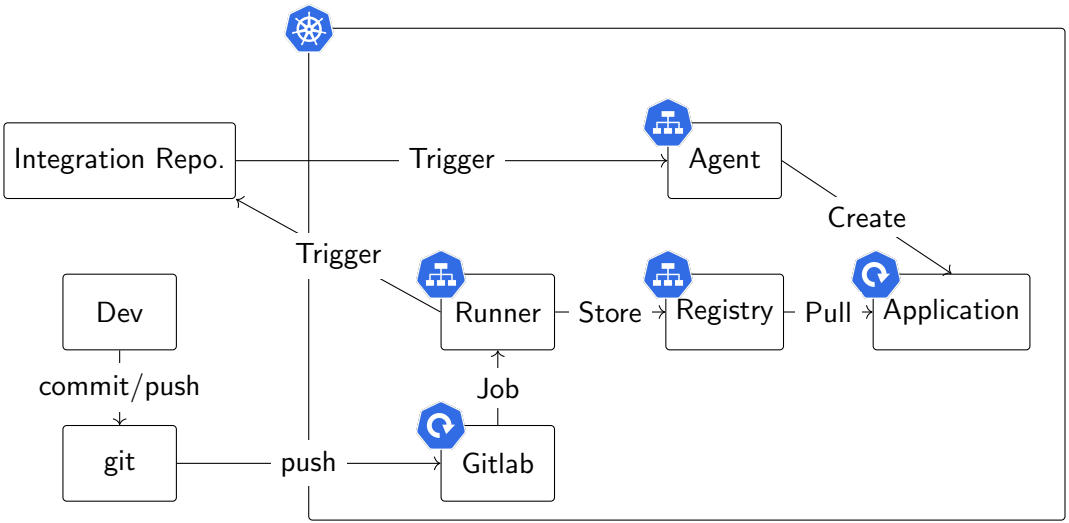
Übersicht Kubernetes



Übersicht Kubernetes



Übersicht Kubernetes



Summary

- Using CI/CD-Pipelines is highly recommended to ensure nessecary software quality.

Summary

- Using CI/CD-Pipelines is highly recommended to ensure nessecary software quality.
- Modern container solutions provide

Summary

- Using CI/CD-Pipelines is highly recommended to ensure nessecary software quality.
- Modern container solutions provide
  - Process isolation via kernel namespaces

Summary

- Using CI/CD-Pipelines is highly recommended to ensure nessecary software quality.
- Modern container solutions provide
  - Process isolation via kernel namespaces
  - Ressource constraints using kernel control groups

Summary

- Using CI/CD-Pipelines is highly recommended to ensure nessecary software quality.
- Modern container solutions provide
  - Process isolation via kernel namespaces
  - Ressource constraints using kernel control groups
- Implementing a reliable CI/CD operation many "moving" parts must align perfectly

Summary

- Using CI/CD-Pipelines is highly recommended to ensure nessecary software quality.
- Modern container solutions provide
  - Process isolation via kernel namespaces
  - Ressource constraints using kernel control groups
- Implementing a reliable CI/CD operation many "moving" parts must align perfectly

Thank you for your attention!

GitHub: `https://github.com/p2k-ko/cicd_pipeline`

References I