



GitOps in Kubernetes

Lukas Buchner

lukas.buchner@qaware.de



QA|WARE

Recap

Continuous Delivery - Definition



QA|WARE

ContinuousDelivery



Martin Fowler

30 May 2013

Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.

martinfowler.com

Continuous delivery

From Wikipedia, the free encyclopedia

Continuous delivery (CD) is a [software engineering](#) approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time.^[1] It aims at building, testing, and releasing software faster and more frequently. The approach helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production. A straightforward and repeatable deployment process is important for continuous delivery.

Abgrenzung zu Continuous X



Continuous Integration (CI)

- Alle Änderungen werden sofort in den aktuellen Entwicklungsstand integriert und getestet.
- Dadurch wird kontinuierlich getestet, ob eine Änderung kompatibel mit anderen Änderungen ist.

Continuous Delivery (CD)

- Der Code *kann* zu jeder Zeit deployed werden.
- Er muss aber nicht immer deployed werden.
- D.h. der Code muss (möglichst) zu jedem Zeitpunkt bauen, getestet und ge-debugged sein.

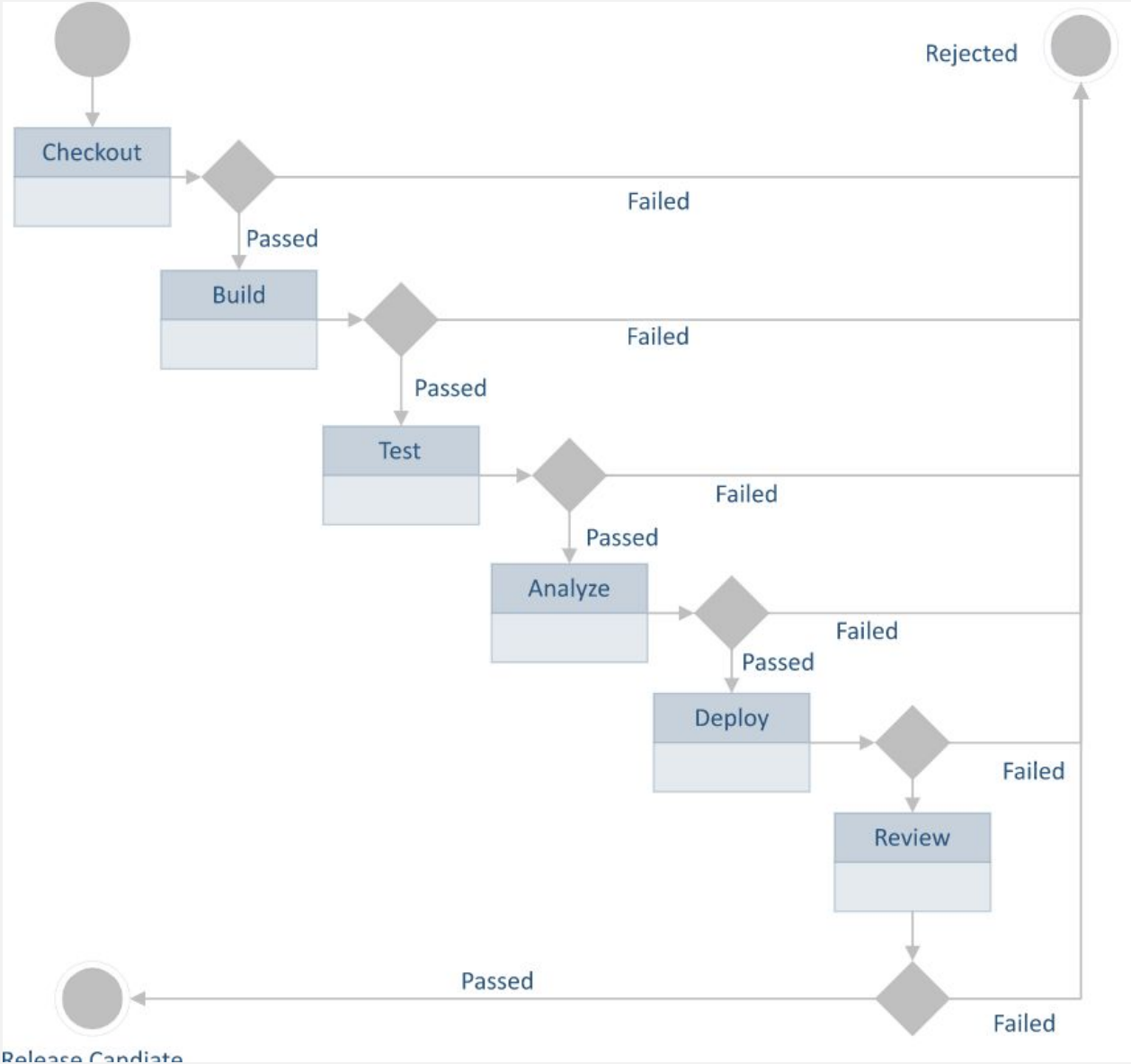
Continuous Deployment

- Jede stabile Änderung wird in Produktion deployed.
- Ein Teil der Qualitätstests finden dadurch in Produktion statt.

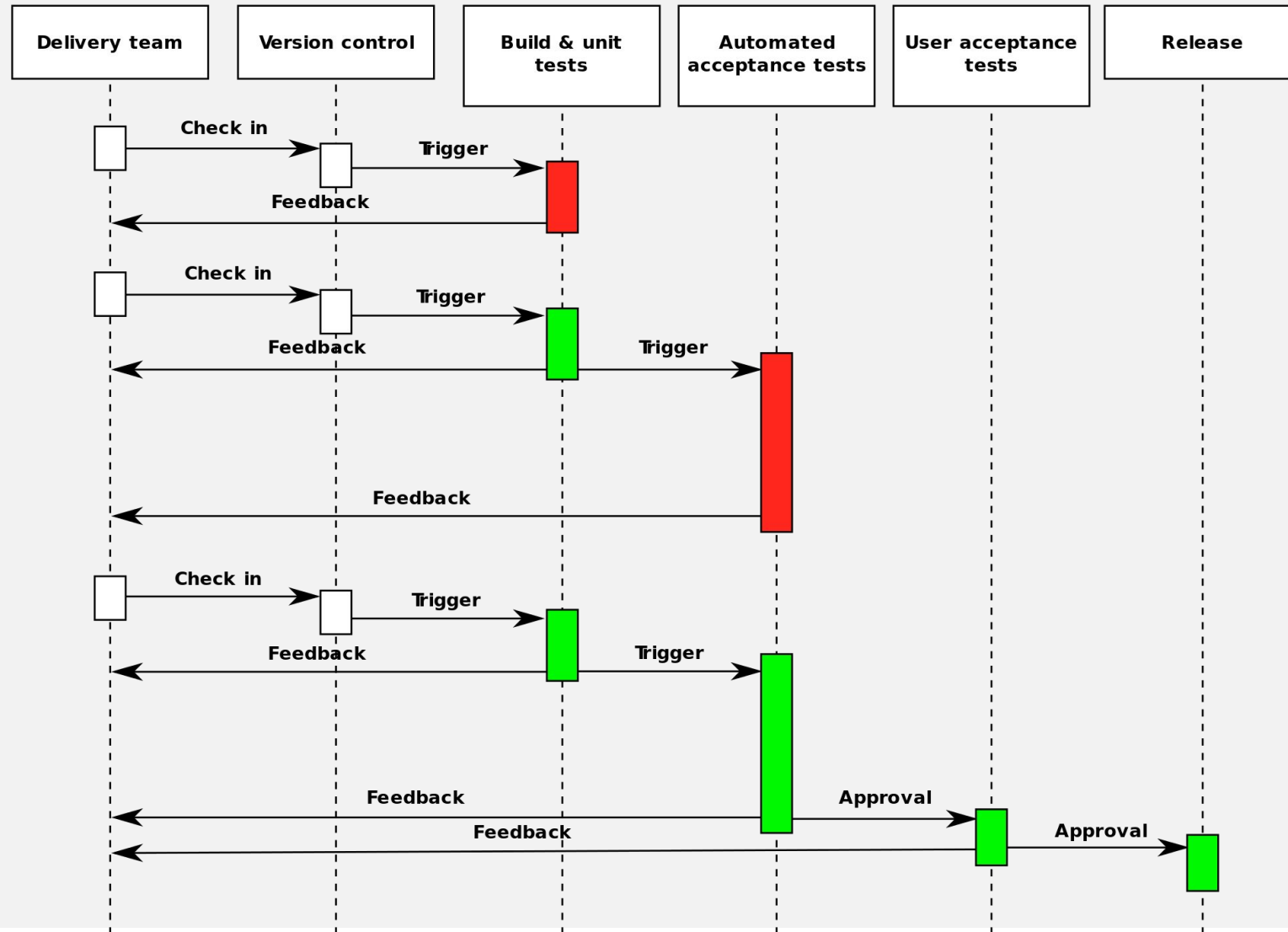
Beispiel: Continuous-Delivery-Pipeline



QA|WARE



Wichtig ist ein schnelles Feedback an das Entwicklerteam, damit Fehler zügig behoben werden

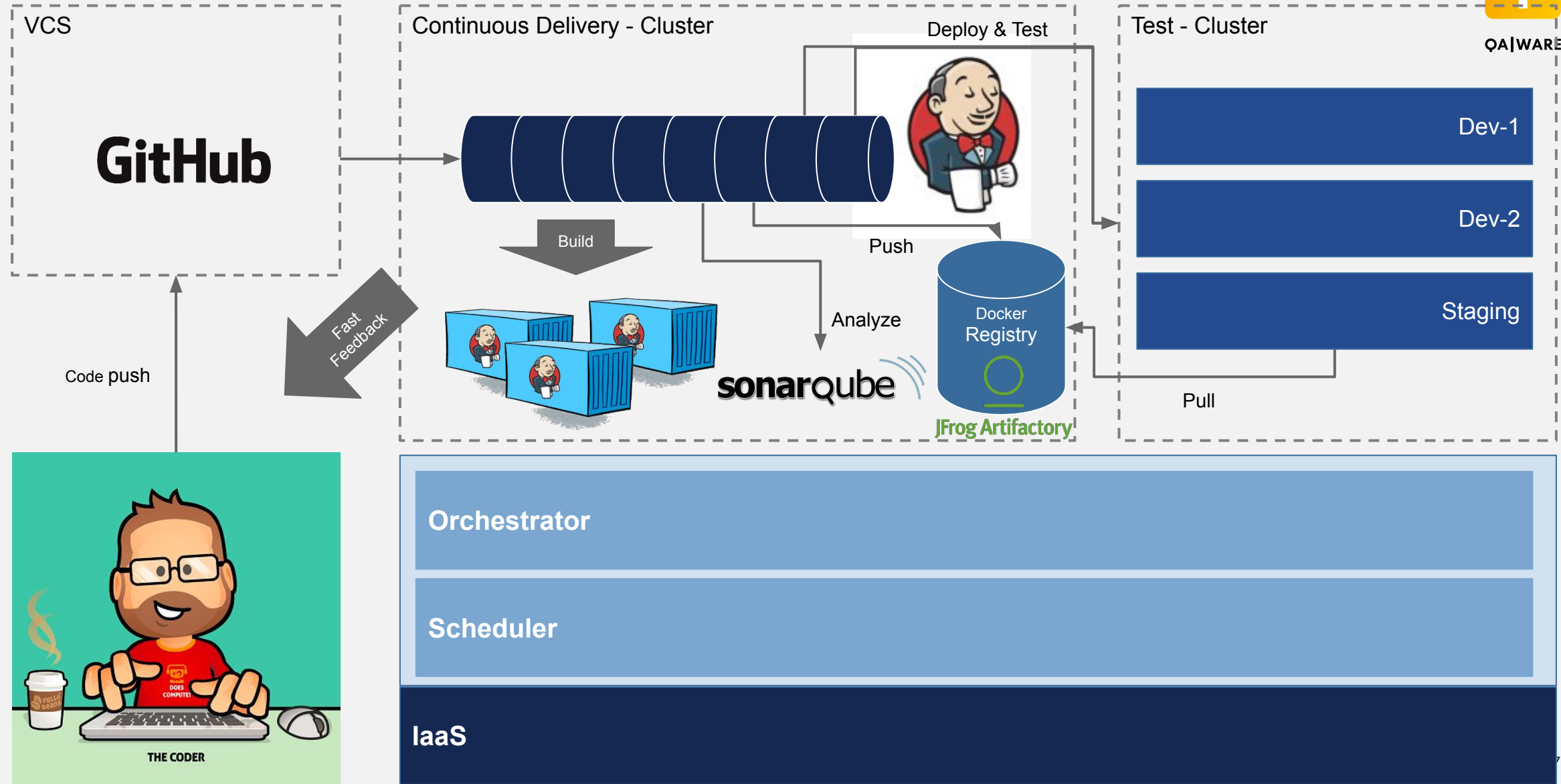


„Continuous delivery process
diagramm“
by Grégoire Détrez
Creative Commons 4.0

Beispiel einer Continuous Delivery Pipeline



QA|WARE



THE CODER



QA|WARE

GitOps

Die Idee



QA|WARE

1 Declarative

A system managed by GitOps must have its desired state expressed declaratively.

2 Versioned and Immutable

Desired state is stored in a way that enforces immutability, versioning and retains a complete version history.

3 Pulled automatically

Software agents automatically pull the desired state declarations from the source.

4 Continuously reconciled

Software agents continuously observe actual system state and attempt to apply the desired state.

Die Vorteile



QA|WARE

- Ermöglicht im Idealfall einen beliebigen Systemzustand in der Historie wiederherzustellen, e.g. einfacher Rollback
- Forciert Pipelines
- Bietet Transparenz von Änderungen, im Falle von Git ist auch ersichtlich wer etwas geändert hat
- Stellt sicher, dass der Systemzustand nicht vom Zielzustand abweicht

GitOps im K8s Umfeld



QA|WARE

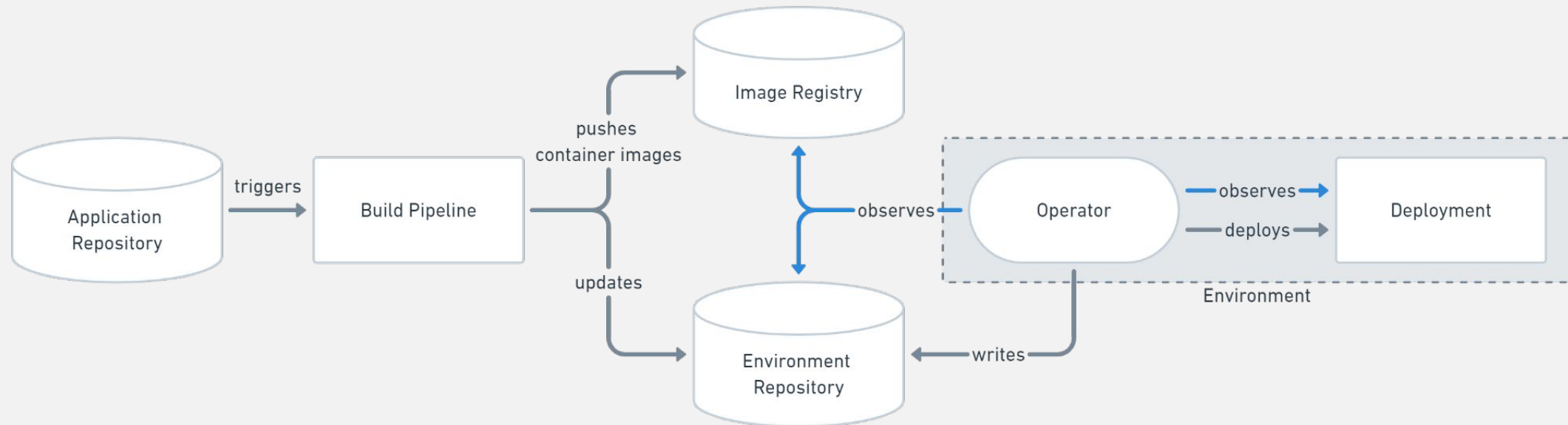


ArgoCD

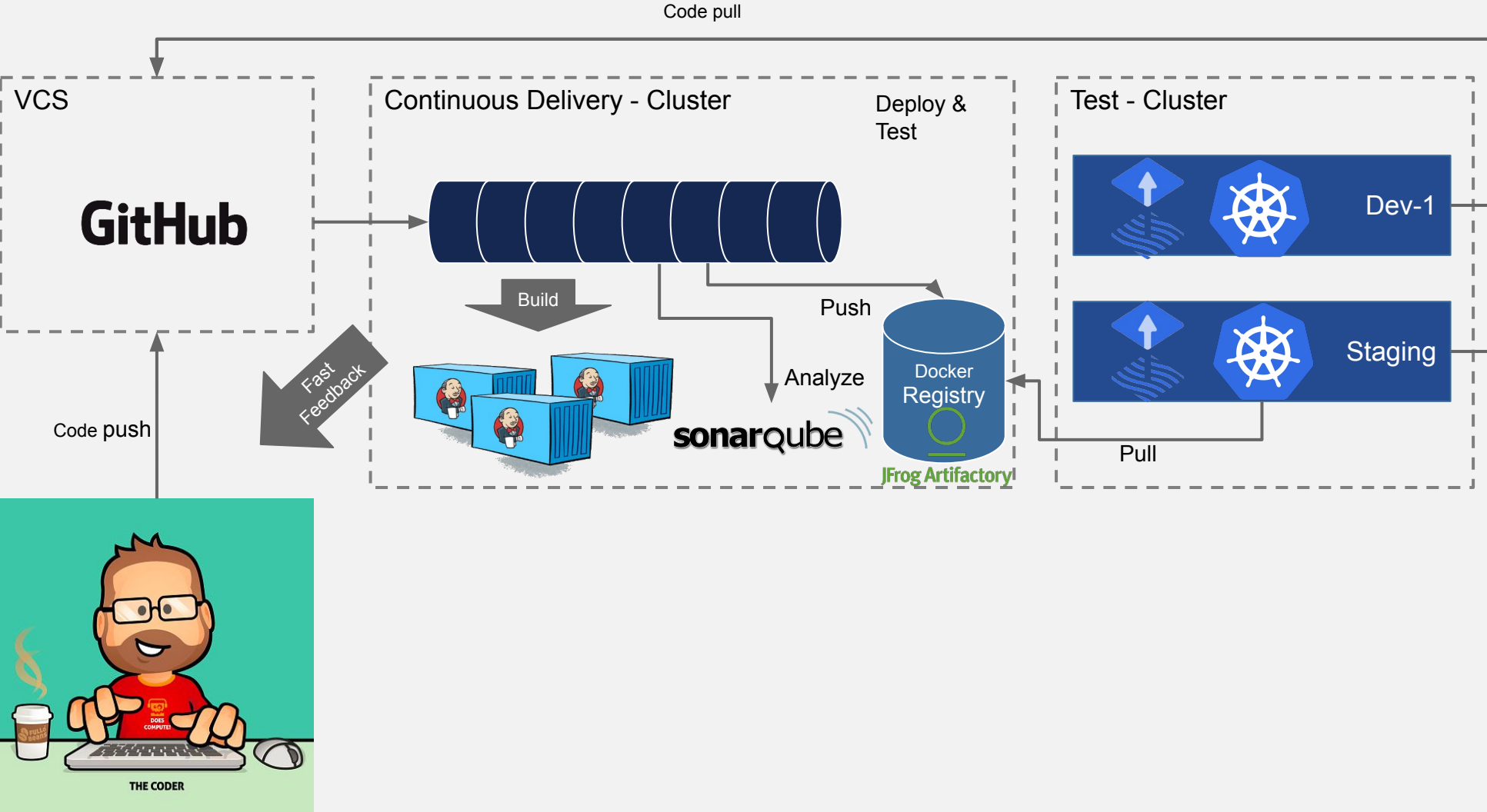


Flux

Pull-based deployments mit GitOps



GitOps mit Flux



Unterschiede in der Architektur

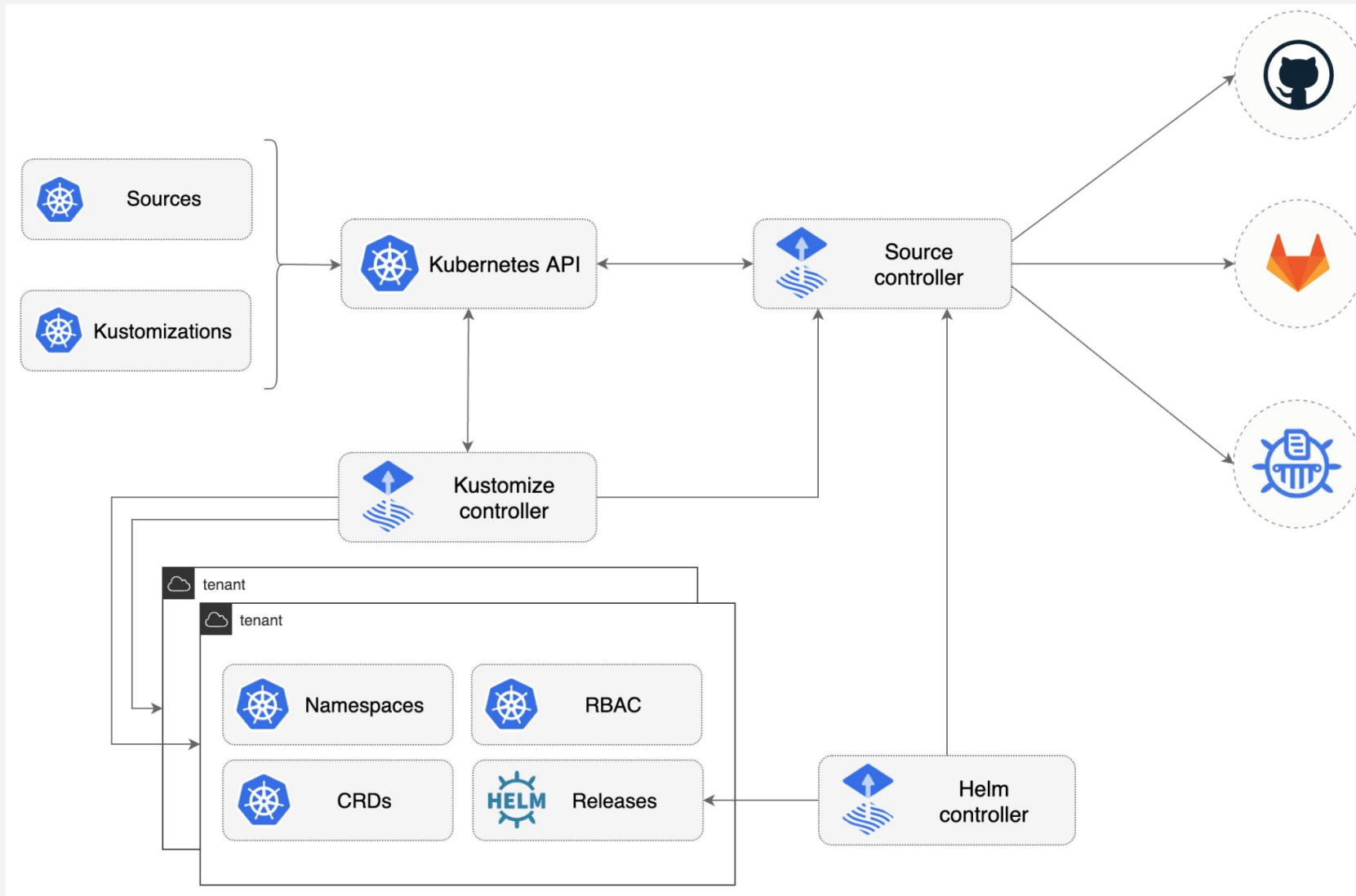


- CI / CD Pipeline braucht keinen cluster Zugriff mehr
 - => verbesserte Security, da für die Pipeline kein high privilege credential mehr notwendig ist
- Cluster pullen jetzt aktiv Source-Repos, welche den Zielzustand enthalten
- Cluster reconcilen ihrer Zustand, d.h. es wird der Ist-Zustand mit dem Ziel-Zustand abgeglichen und versucht den Ziel-Zustand zu erreichen.
 - => Cluster konvergieren automatisch gegen den Zielzustand

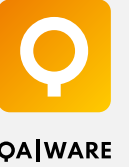
Flux im Detail



QA|WARE



Flux im Detail



- Source-Controller
 - bindet verschiedene Quellen an z.B. Git-Repos, OCI-Repos etc.
 - stellt die heruntergeladenen Pakete den anderen Controllern zur Verfügung
- Helm-Controller
 - bietet eine Helm Integration für Flux. In Verbindung mit dem Source-Controller lassen sich Helm-Charts automatisch herunterladen und installieren/updaten
- Kustomize-Controller
 - nutzt die Git-Repos des Source-Controllers um kustomize auszuführen und die konfigurierten Ressourcen im Cluster zu applien. Alle Ressourcen werden getrackt und können auch automatisiert wieder garbage collected werden.
- Notification-Controller
 - bietet insights in flux events z.B. Erfolge oder aufgetretene Fehler.
 - bietet Integrationen in Chat Systeme

Sources konfigurieren



QA|WARE

```
apiVersion: source.toolkit.fluxcd.io/v1
kind: GitRepository
metadata:
  name: podinfo
  namespace: default
spec:
  interval: 5m0s
  url: https://github.com/stefanprodan/podinfo
  ref:
    branch: master
```

- konfiguriert ein **GitRepository** als Quelle im Source-Controller
- klonst den **master** branch von **url** und stellt ihn den anderen controllern als tar.gz zur Verfügung
- pollt die **url** alle 5 Minuten und prüft, ob es Änderungen gibt

Kustomizations konfigurieren



QA|WARE

```
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: podinfo
  namespace: default
spec:
  interval: 10m
  targetNamespace: default
  sourceRef:
    kind: GitRepository
    name: podinfo
  path: "./kustomize"
  prune: true
  timeout: 1m
```

- erstellt eine Kustomization im Kustomize-Controller, die über **sourceRef** das GitRepository vom Source-Controller referenziert
- der kustomize-Controller holt sich den Source-Code vom Source-Controller und applied mit kustomize (k8s-Tool) alles unter **path**



QAWARE

Flux in action!



Cloud runtimes

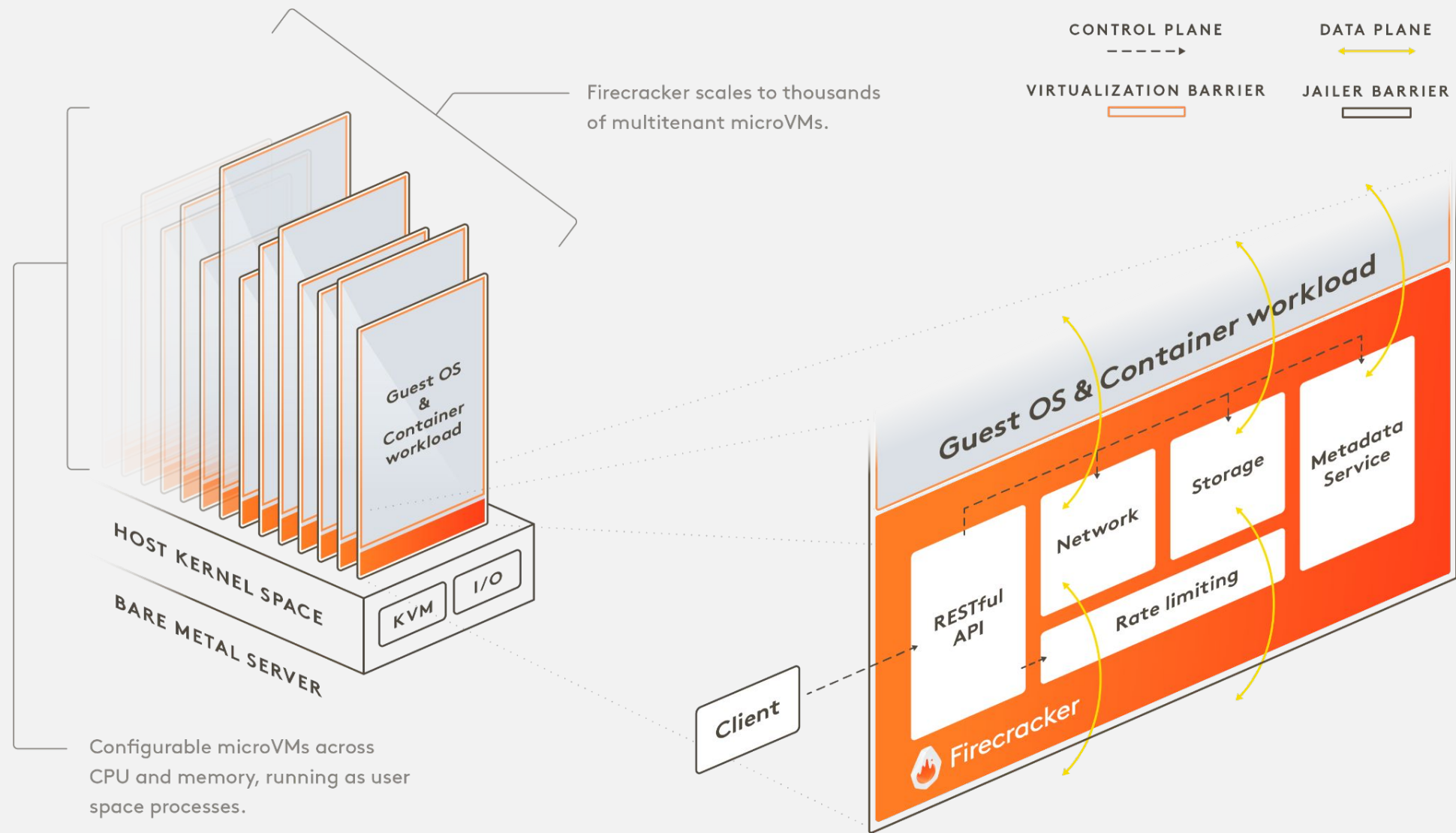
Lukas Buchner

lukas.buchner@qaware.de

Firecracker - back to VMs



QAWARE



Configurable microVMs across CPU and memory, running as user space processes.

Firecracker



QAWARE

- microVMs statt containern
- minimaler Overhead, wesentlich bessere Isolation
- schnelle Startup times
- wird von Amazon für AWS lambda und Fargate verwendet

Hashicorp Nomad als Orchestrator



Vorteile:

- einfach zu bedienender Workload Orchestrator
 - container
 - VMs
 - binaries
- einfacher strukturiert als Kubernetes
 - es muss nur ein binary deployed werden - gleiches binary für server und client

Nachteile:

- Ökosystem ist wesentlich schwächer
 - kein helm
 - kein gitops etc.
- Braucht für features Service Discovery und Secrets Management weitere Produkte (Consul und Vault)
- Features für größere Firmen erfordern Lizenzen

Webassembly - the runtime of the future?



QA|WARE



WEBASSEMBLY

Webassembly - Overview



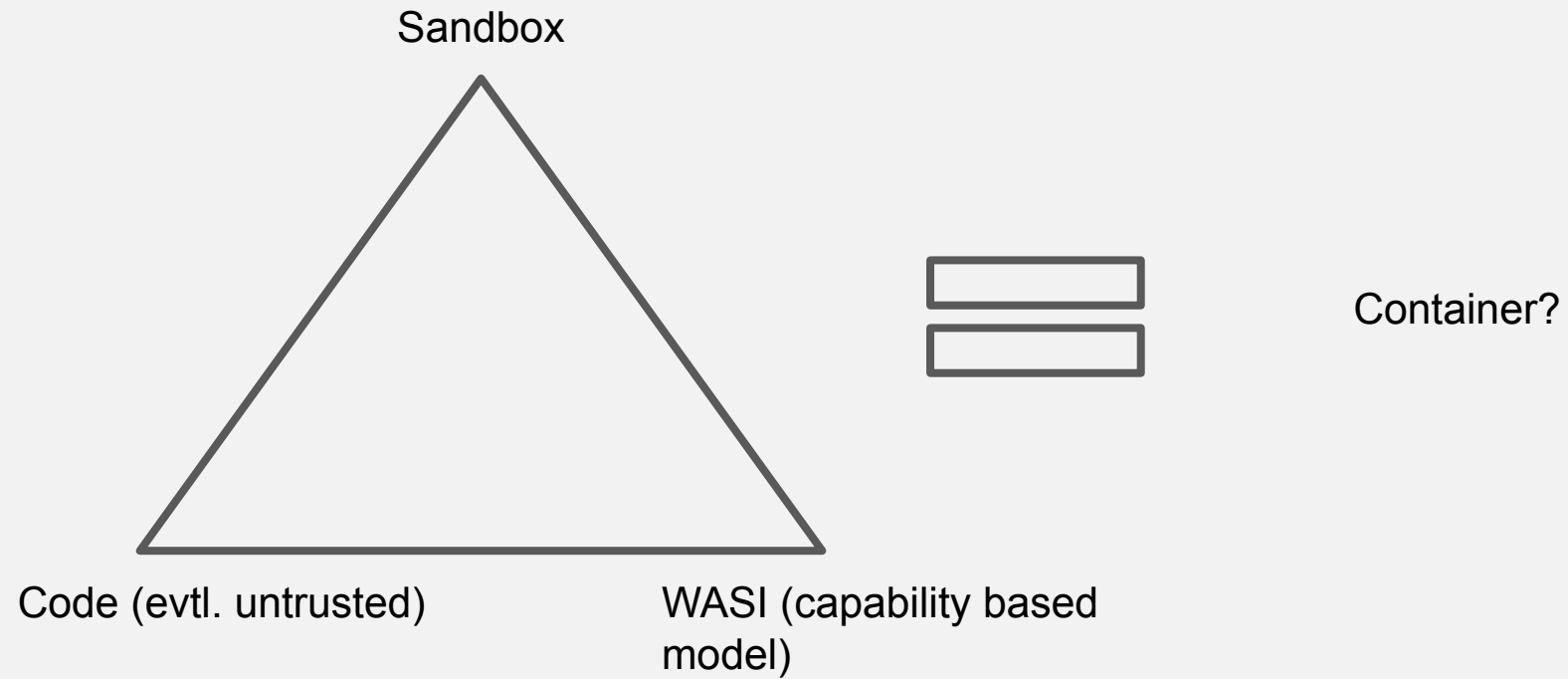
QA|WARE

- Webassembly ist eine Sammlung an Standards
 - Core Specification - Beschreibung des Bytecode Formats etc.
 - Embedding Interfaces - Beschreibung wie Webassembly in verschiedene Technologien eingebettet werden kann
 - WASI - System Interface Beschreibung. Webassembly Programme laufen in einer Sandbox. Mit WASI kann feingranular Zugriff auf System APIs gesteuert werden, z.B. file oder network access.
- Wurde ursprünglich für das Web entwickelt und ist im Browser lauffähig
- Webassembly ist als Compilation-Target verschiedenster Sprachen verfügbar
- Läuft per default in einer Sandbox => großer Unterschied zu vorherigen Versuchen einer generalisierten Plattform wie CLR oder JVM.

Webassembly - im Backend?



QA|WARE



Webassembly auf Kubernetes?



QA|WARE

- es wird mittlerweile aktiv versucht, dass container runtimes WASM Module direkt supporten
 - containerd wird erweitert und kann dann WASM als “normale” Kubernetes workload ausführen
- dedizierte Projekte (viele davon sind allerdings verwaist)
 - z.B. [wasmcloud](#)