

# CI / CD

Continuous Integration und Continuous Delivery

ZooRoyal IT

Sebastian Knott

1/21/2019

# SITUATION

---

# QQ little bUGS IN THE CODE

## PROBLEMSTELLUNG

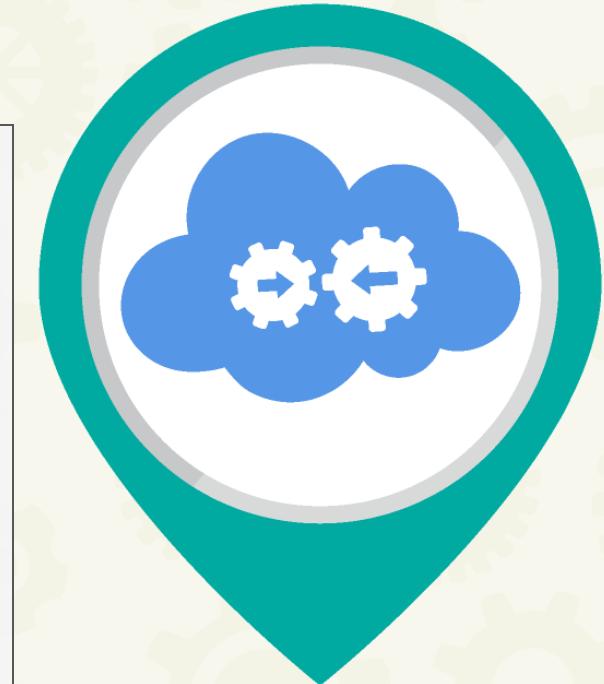
- ❖ Commits selten und/oder buggy
- ❖ Tests zeitaufwendig
- ❖ Unregelmäßige Releases
  - ❖ Großer Abstand zwischen Feature-Releases ...
  - ❖ ... gefolgt von vielen Hotfixes
  - ❖ Releases kaum planbar

# QQ little bUGS IN THE CODE



## INTEGRATION KOMPLEX

- ❖ Viele benötigte Komponenten
  - ❖ Fixtures (Datenbanken, Caches)
  - ❖ Services (Redis, MySQL, nginx)
  - ❖ Libraries (composer, npm)
- ❖ Gar nicht oder teilweise automatisiert
- ❖ Benötigt Expertenwissen



## ZUSAMMENARBEIT KOMPLEX

- ❖ Überschneidende Commits werden spät bemerkt
- ❖ Gemeinsame Abhängigkeiten schwer zu verwalten
- ❖ Regressionen

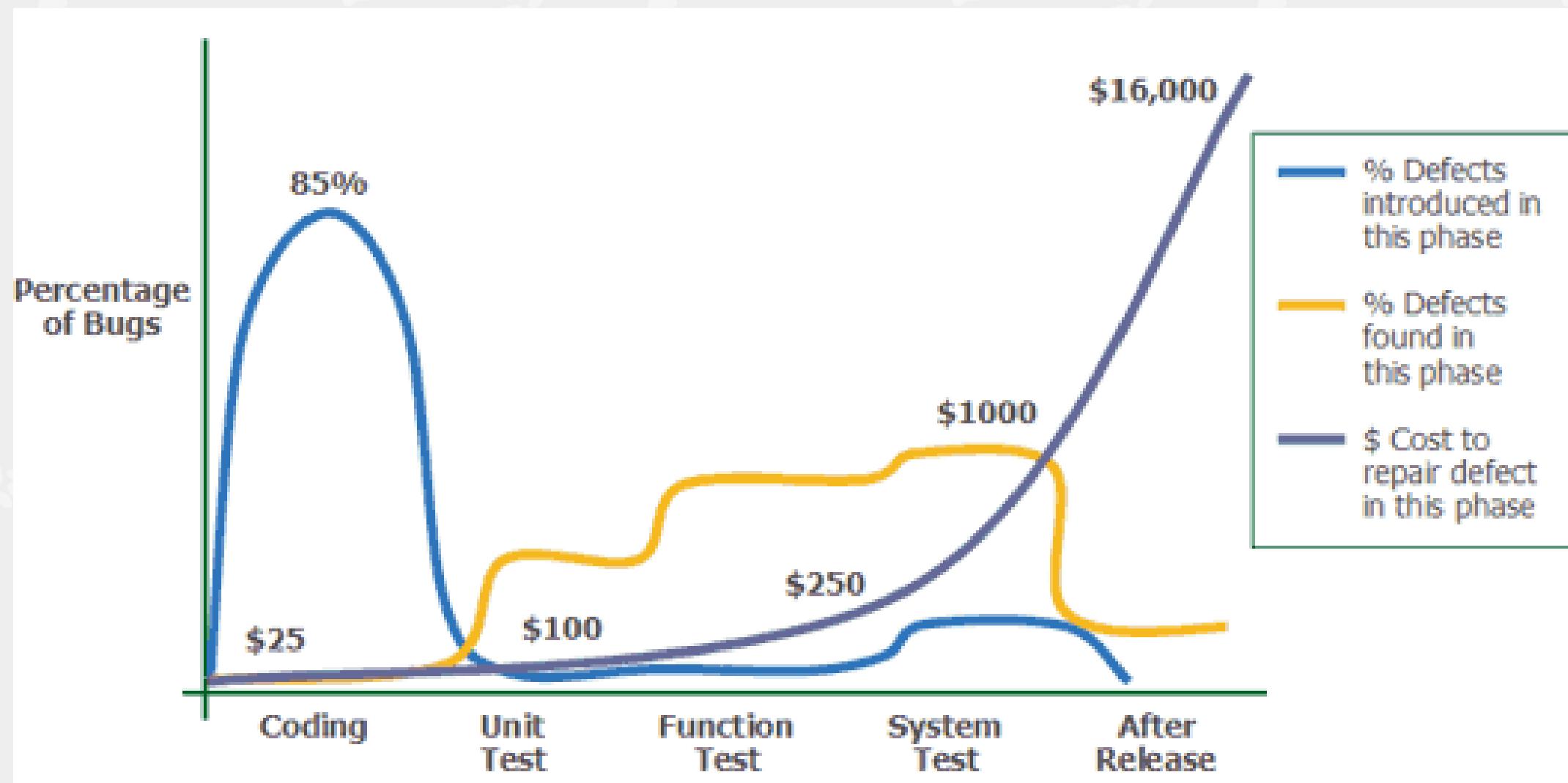
## FEHLENDE ÜBERSICHT

- ❖ Tests werden nicht genutzt
- ❖ Codemetriken punktuell
- ❖ Es gibt keinen aktuellen Stand

## KEIN LAUFFÄHIGER STAND

- ❖ Läuft nicht auf alle Systemen
- ❖ Test-/Abnahmebuilds extra angefertigt
- ❖ Spezialwissen für jeden Stand

## FEHLER WERDEN SPÄT ERKANNT



Source: Applied Software Measurement, Capers Jones, 1996

# FOLGEN

- Unzureichende Tests
- Lange Produktzyklen
- Fehler sind schwer zu beheben
- Verzögerungen
- Maintenance aufwendig
- Codebasis unflexibel
- Verschwendung von Entwicklerzeit
- Kein Trunk Based Development ↗ möglich

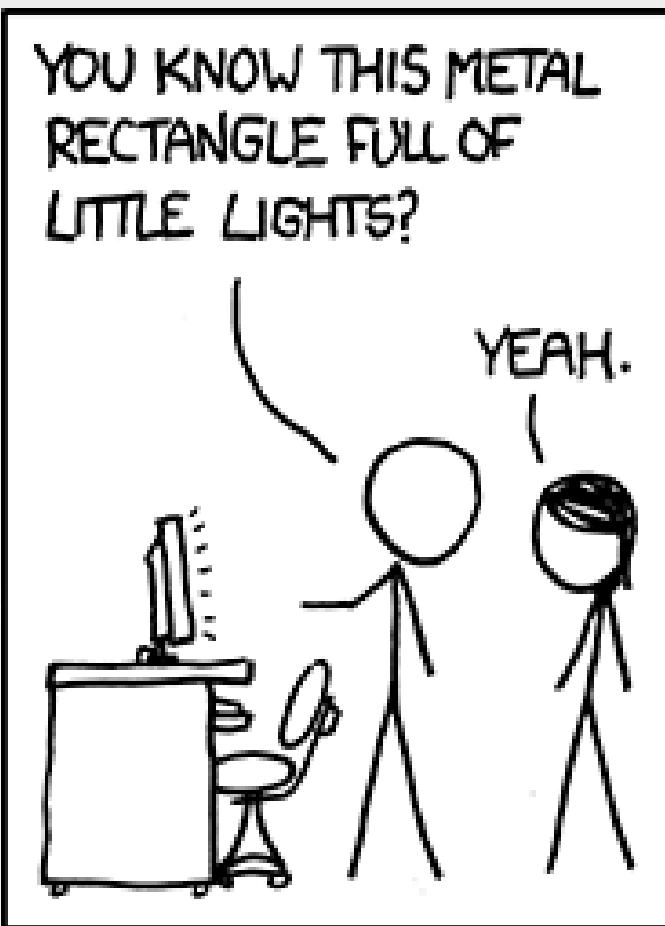
M T W T F S S

1						
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

## ZIELE

- ❖ Fehler früh finden
- ❖ Abläufe automatisieren
- ❖ Auf wichtige Sachen konzentrieren

**Weniger verwalten. Mehr Umsetzen.**



# CONTINUOUS INTEGRATION

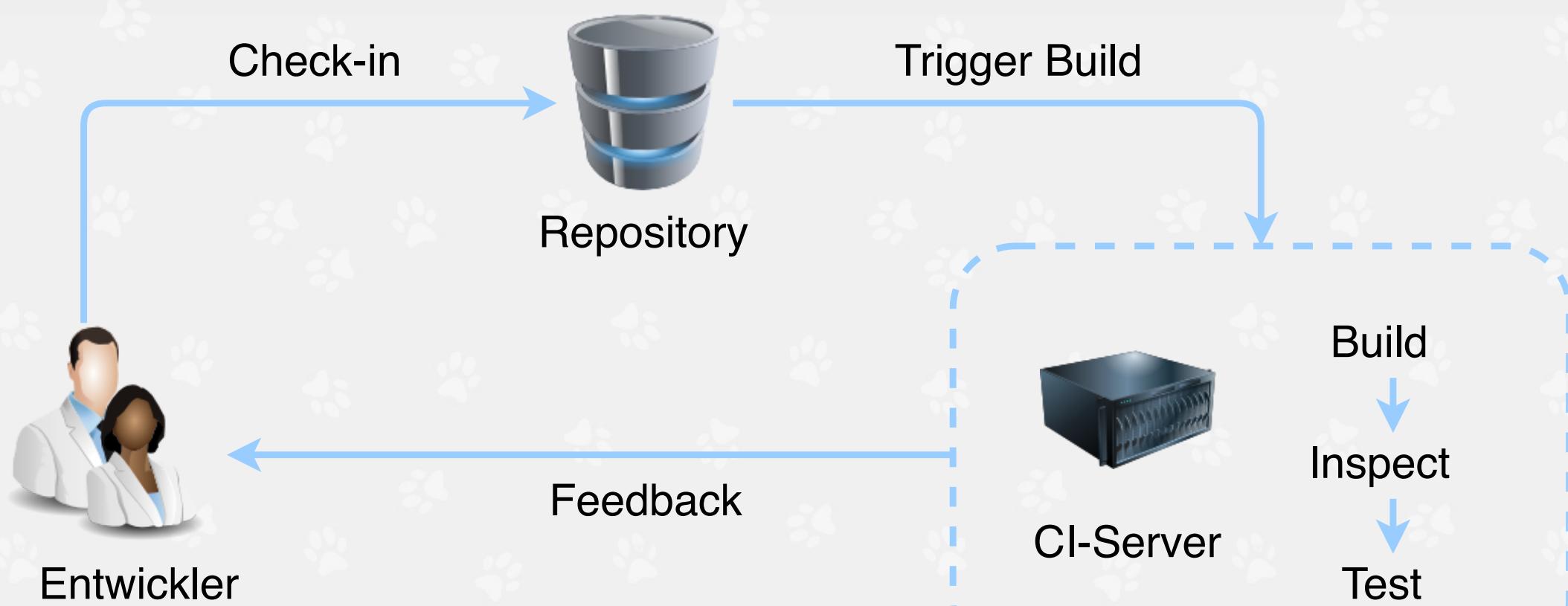
---

# WAS IST CONTINUOUS INTEGRATION

”

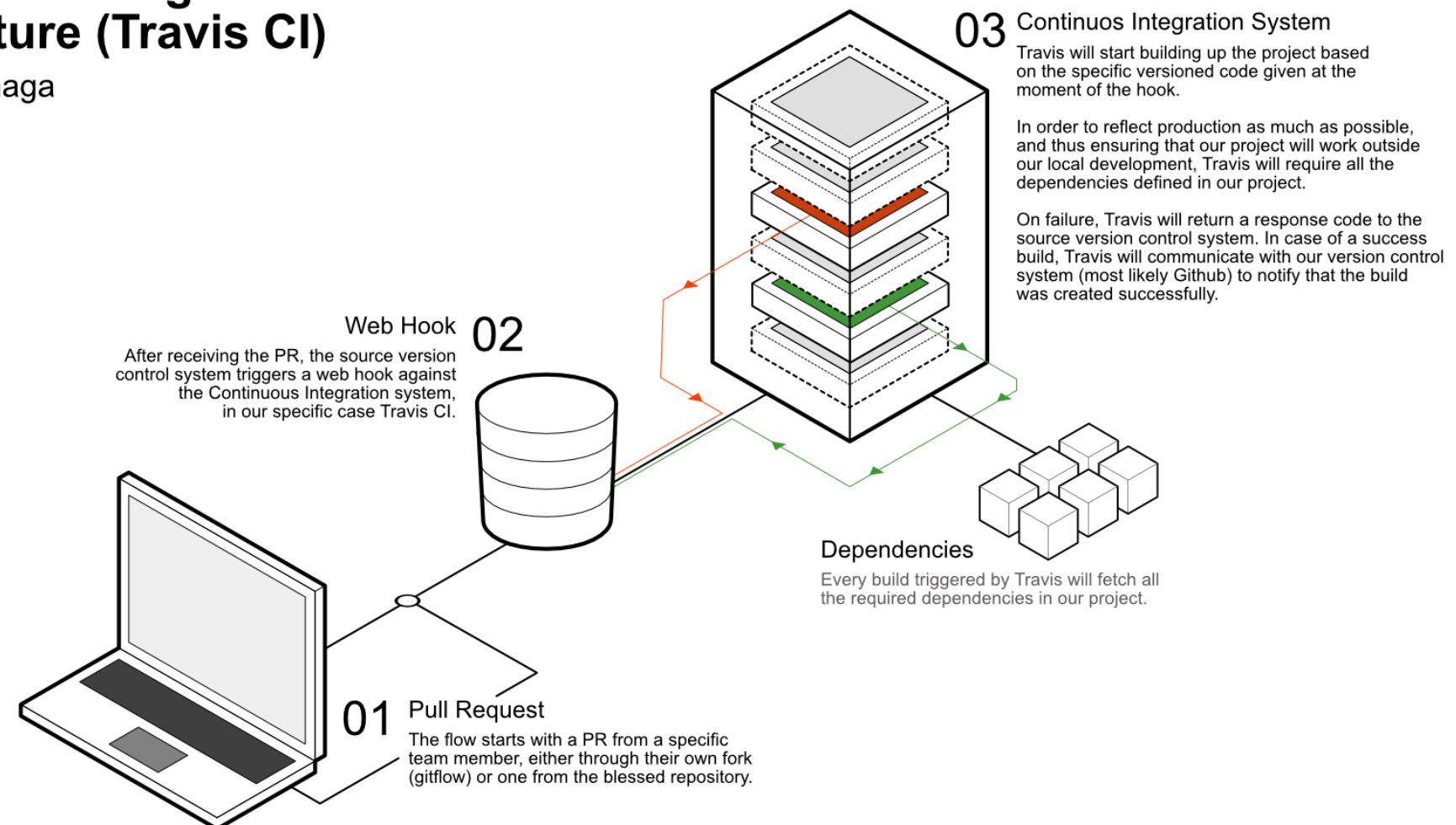
*Continuous Integration is a **software development practice** where members of a team **integrate their work frequently**, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is **verified by an automated build** (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly **reduced integration problems** and allows a team to develop cohesive software more rapidly.*

— Martin Fowler, 2006 



# Continuous Integration Architecture (Travis CI)

By Jose Aguinaga



# NICHT BENÖTIGT

- ❖ Continuous Delivery
- ❖ Agile, Scrum, Kanban, Wasserfall
- ❖ Infrastructure as Code

# VORAUSSETZUNGEN

- ❖ Version Control System
  - ❖ (Git, SVN, CVS)
- ❖ Automatisierte Builds
  - ❖ Builden auf allen Rechnern/Servern
- ❖ Automatisierte Tests
  - ❖ Automatische Ausführung
  - ❖ Laufen auf allen Rechnern
- ❖ Build Server / CI-Tool
  - ❖ (Jenkins, Bamboo, GitLab)

# AUTOMATISIERTE BUILDS

## 1. Installation

- Code wird bezogen aus VCS
- Libraries und Tools werden installiert

## 2. Codeanalyse

- Lint
- Style
- Potentielle Fehler aufdecken

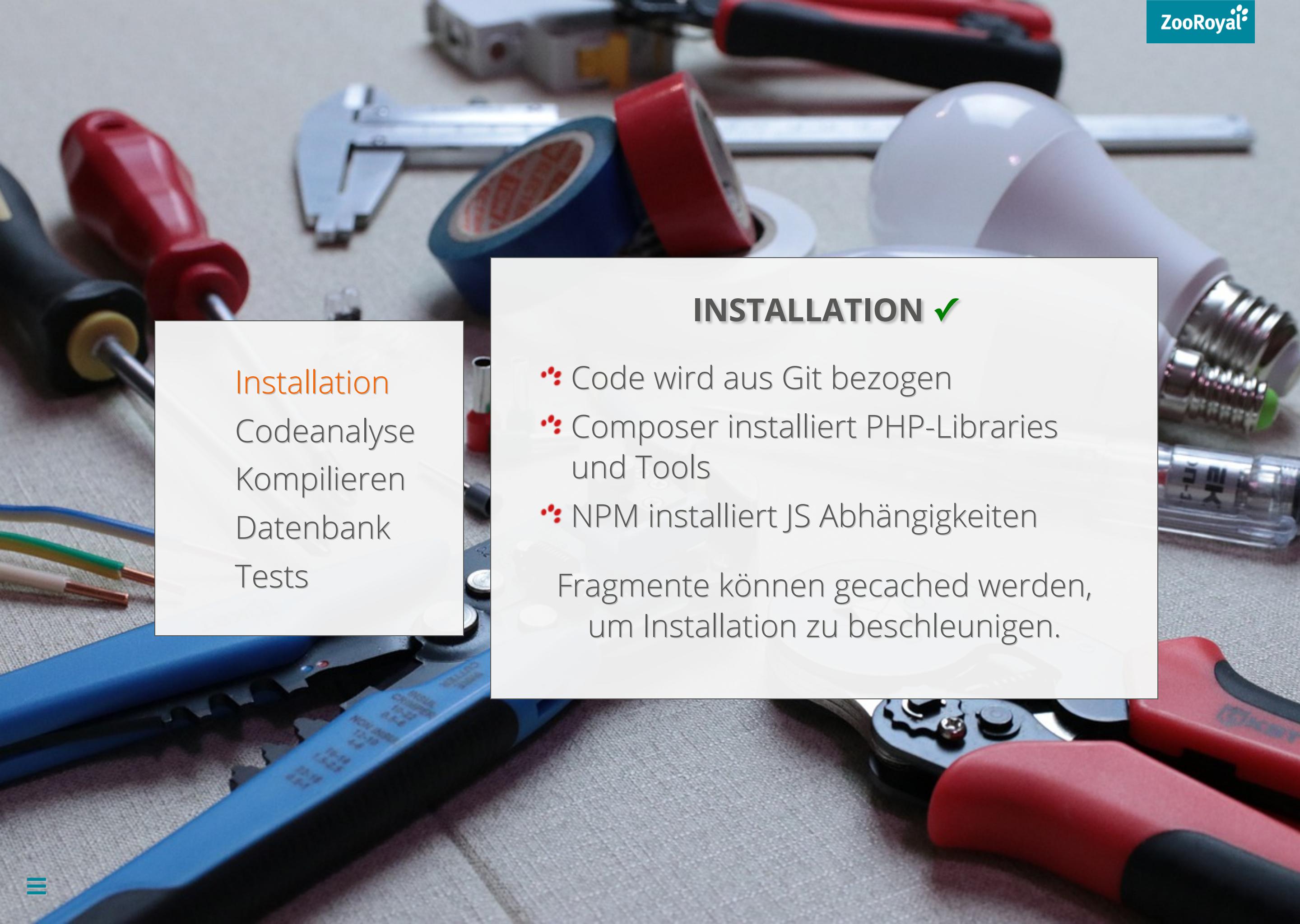
## 3. Kompilieren

- Z.B. Less -> CSS

## 4. Datenbank

- Migrationen ausführen
- Fixtures nutzen um Datenbank mit Daten zu befüllen

## 5. Automatische Tests



Installation  
Codeanalyse  
Kompilieren  
Datenbank  
Tests

## INSTALLATION ✓

- ❖ Code wird aus Git bezogen
- ❖ Composer installiert PHP-Libraries und Tools
- ❖ NPM installiert JS Abhängigkeiten

Fragmente können gecached werden, um Installation zu beschleunigen.

Installation  
Codeanalyse  
Kompilieren  
Datenbank  
Tests

## CODEANALYSE ✓

- PHP Parallel Lint
- PHP Code Sniffer
- PHP Mess Detector
- Copy Paste Detector
- ESLint
- StyleLint

Nur geänderte Teile der Codebasis  
prüfen.

Installation  
Codeanalyse  
**Kompilieren**  
Datenbank  
Tests

## KOMPILIEREN ✓



Nur geänderte Quellen kompilieren.

Installation  
Codeanalyse  
Kompilieren  
**Datenbank**  
Tests

## DATENBANK X

- ❖ Migration
  - ❖ (Doctrine, Phinx , ??? )
- ❖ Keine einheitliche Infrastrukturen und Prozesse
- ❖ Fixtures einspielen

Installation  
Codeanalyse  
Kompilieren  
Datenbank  
**Tests**

## AUTOMATISCHE TESTS

- ❖ Tests mit PHPUnit
  - ❖ Unit Tests
  - ❖ Integration Tests
  - ❖ System Tests
- ❖ Frontendtests
  - ❖ Z.B. mit Selenium



## WANN BUILDEN?

- ❖ Bei jedem Pullrequest
- ❖ Bei jedem Commit?

# FEEDBACK

... sollte möglichst ...

- ❖ ... schnell zur Verfügung stehen.
- ❖ ... einfach zu erreichen sein.
- ❖ ... unaufgefordert verteilt werden.

## ENTWICKELN MIT CI

- Commit Early, Commit often
- Keinen kaputten Stand commiten
- Fehler im Build sofort beseitigen
- "Fail fast" → "Learn fast"
- Gegen alle Umgebungen testen

# CONTINUOUS DELIVERY

---

# WARUM CONTINUOUS DELIVERY?

## ANFORDERUNGEN DES BUSINESS

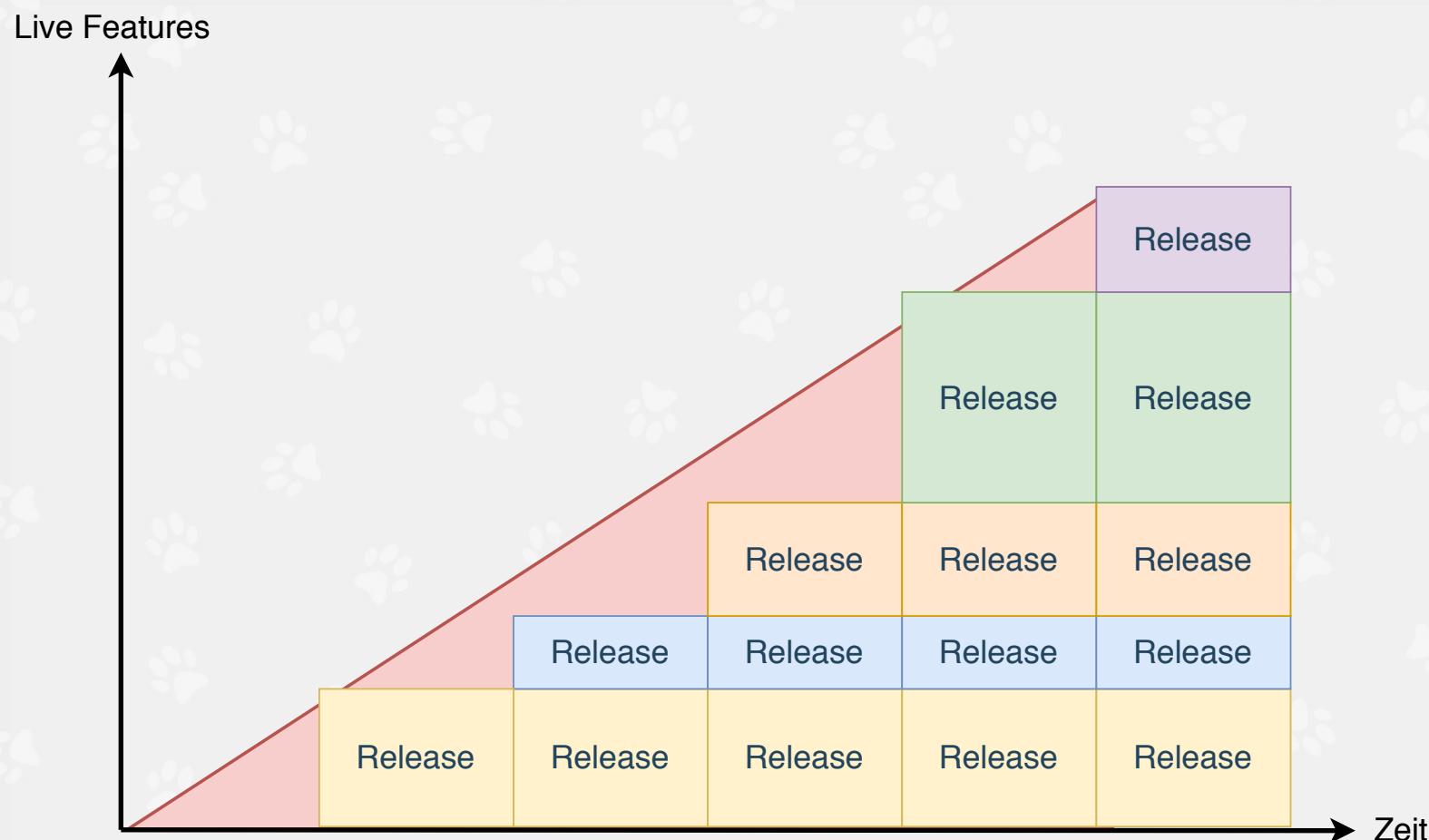
- ❖ Kurze Entwicklungszeit
- ❖ Kostengünstig
- ❖ Kurze Korrekturschleifen
- ❖ Kurze Time-to-Value

## ANFORDERUNGEN DER ENTWICKLER

- ❖ Vorhersagbare ...
- ❖ Wiederholbare ...
- ❖ Verlässliche Deployments
- ❖ Wartungsarme Software

## TIME-TO-VALUE

Kurze Time-To-Value-Zeiten erfordern Kurze Releasezyklen auch und gerade bei großen Änderungen.

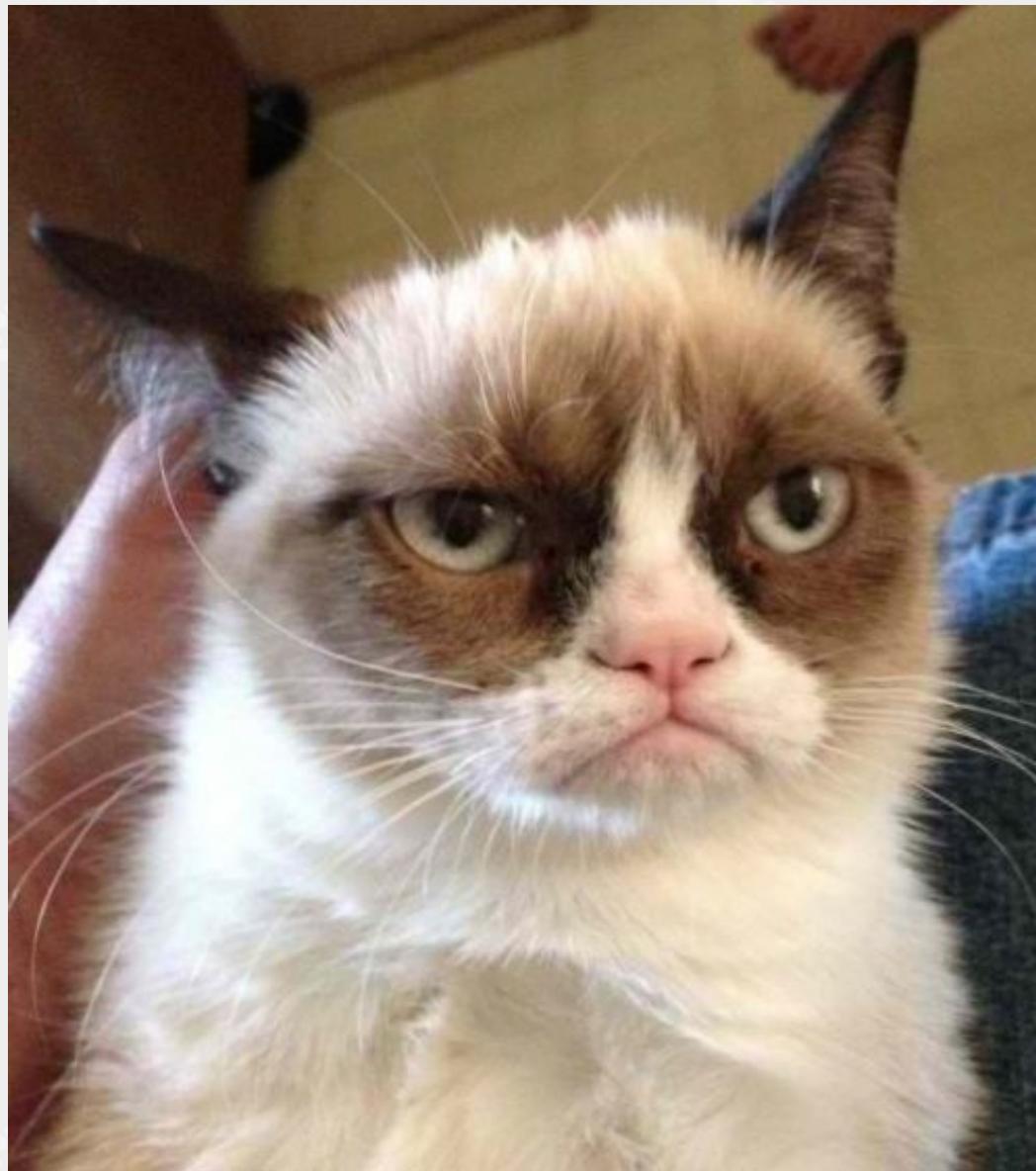




## BUZWORD BINGO

- ❖ CD fügt sich ein in LEAN Product Development ↗
- ❖ Build → Measure → Learn → Repeat
- ❖ CD beschleunigt den Build-Teil

## DAS IST DOCH HIPPIEKRAM ...

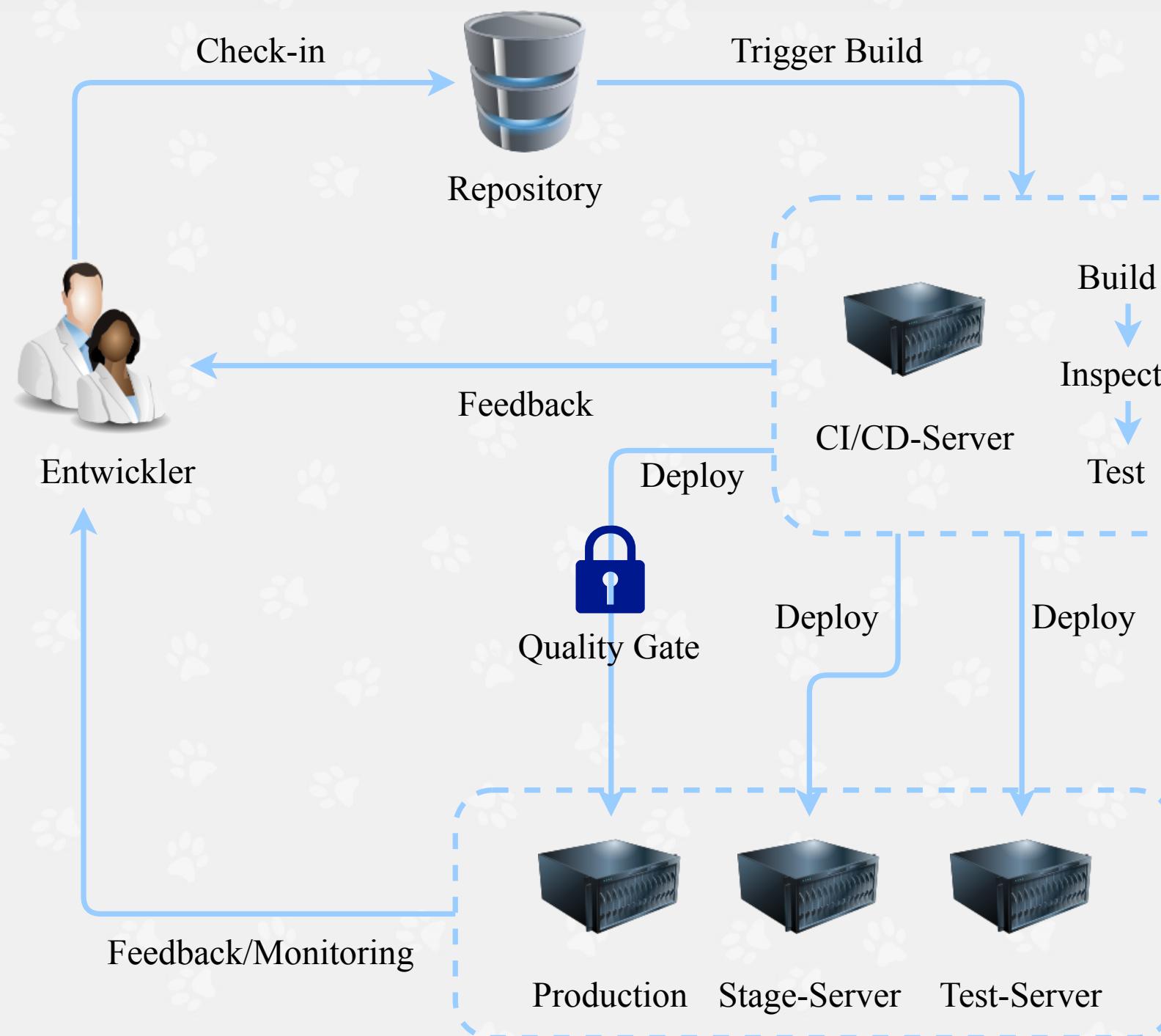


- ❖ Weiter Verbreitung – auch bei kleineren Unternehmen (z.B. IBM, Microsoft, Adobe, Google, Tesla, Trivago, Sparhandy, FriendScout24)
- ❖ Ausführliche Analyse in *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation* (2011)<sup>↗</sup>  
von Jez Humble und David Farley

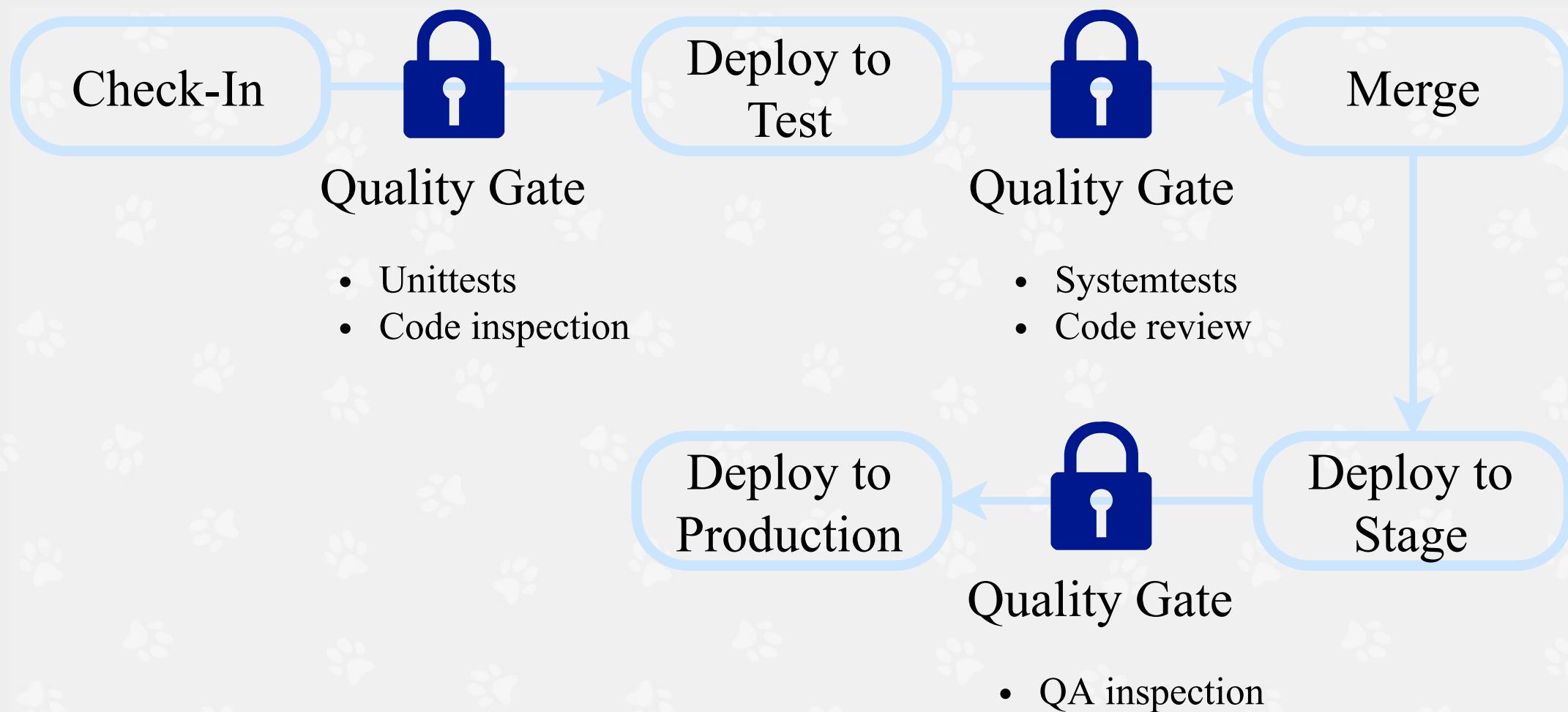
## WIE FUNKTIONIERT CD

- ❖ Automatisch auslösbarer Deployments
- ❖ Deployments vollständig automatisiert
- ❖ Deployments funktionieren in *allen* Umgebungen
- ❖ Alle Änderungen im CVS
- ❖ Jedem Entwickler zugänglich
- ❖ Jeder Schritt transparent für jeden Entwickler

## CD STRUKTUR



## CD ABLAUF BEISPIEL



# FAZIT

Continuous Integration

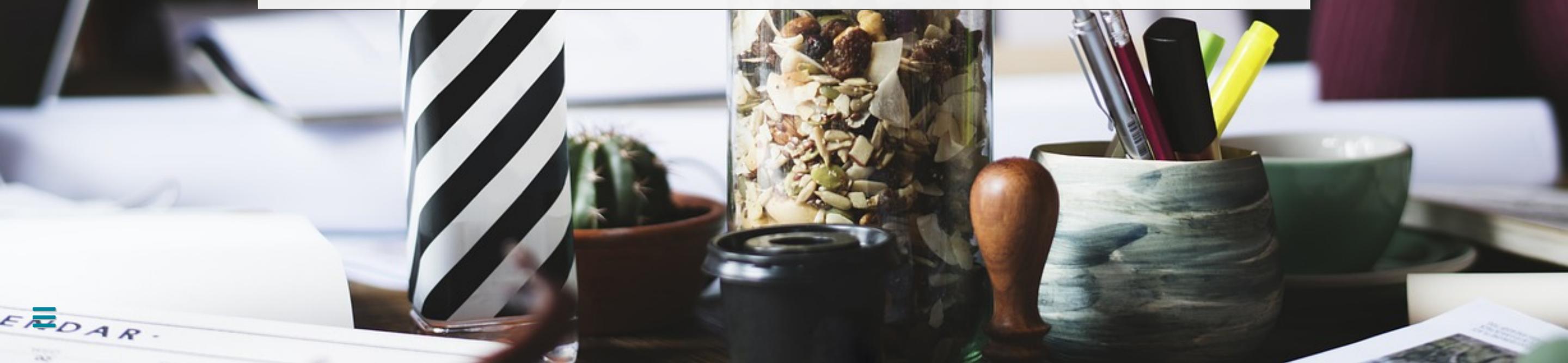


Fehler früh erkennen

Continuous Delivery



Software schnell ausliefern





leichen  
Danke!