

Continuous Integration

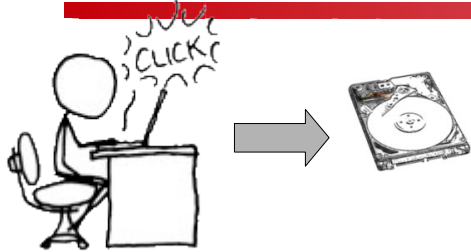
Der Build-Server sieht dich

Entwicklungszyklus



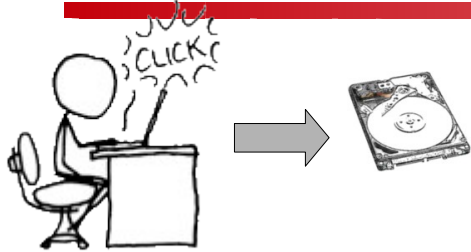
Entwickler

Entwicklungszyklus



Entwickler

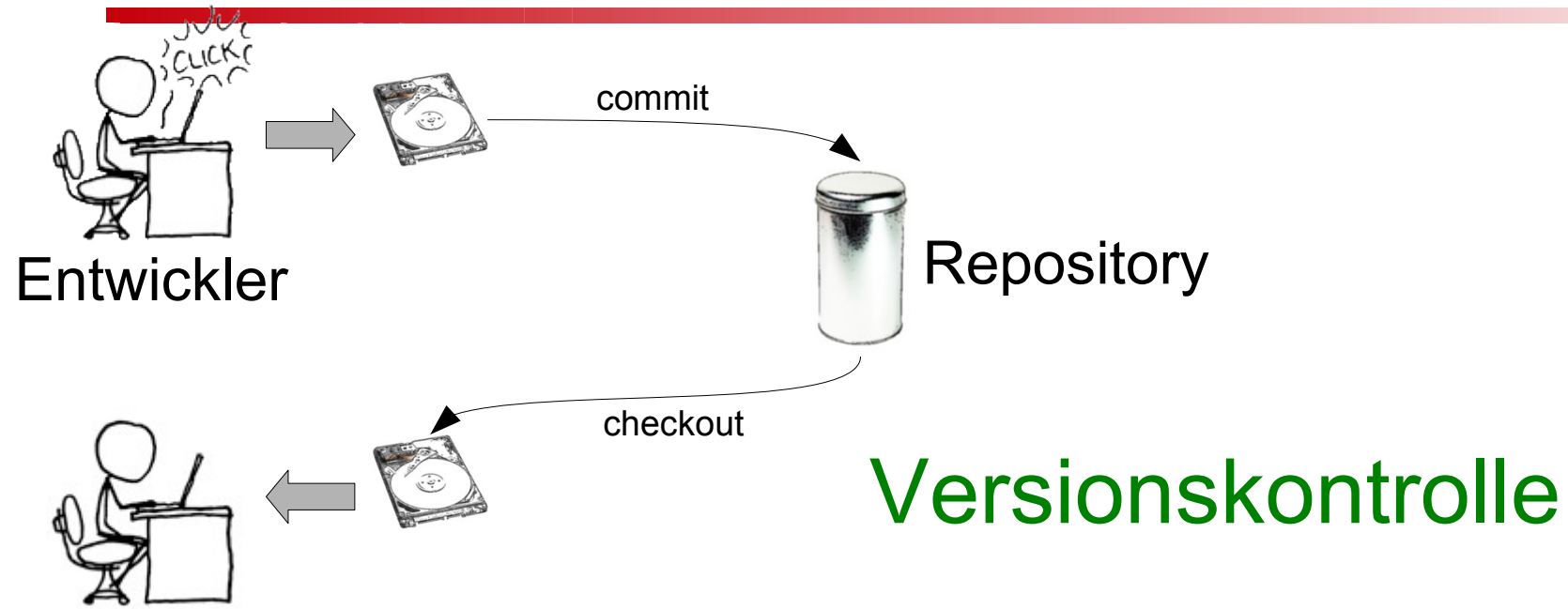
Entwicklungszyklus



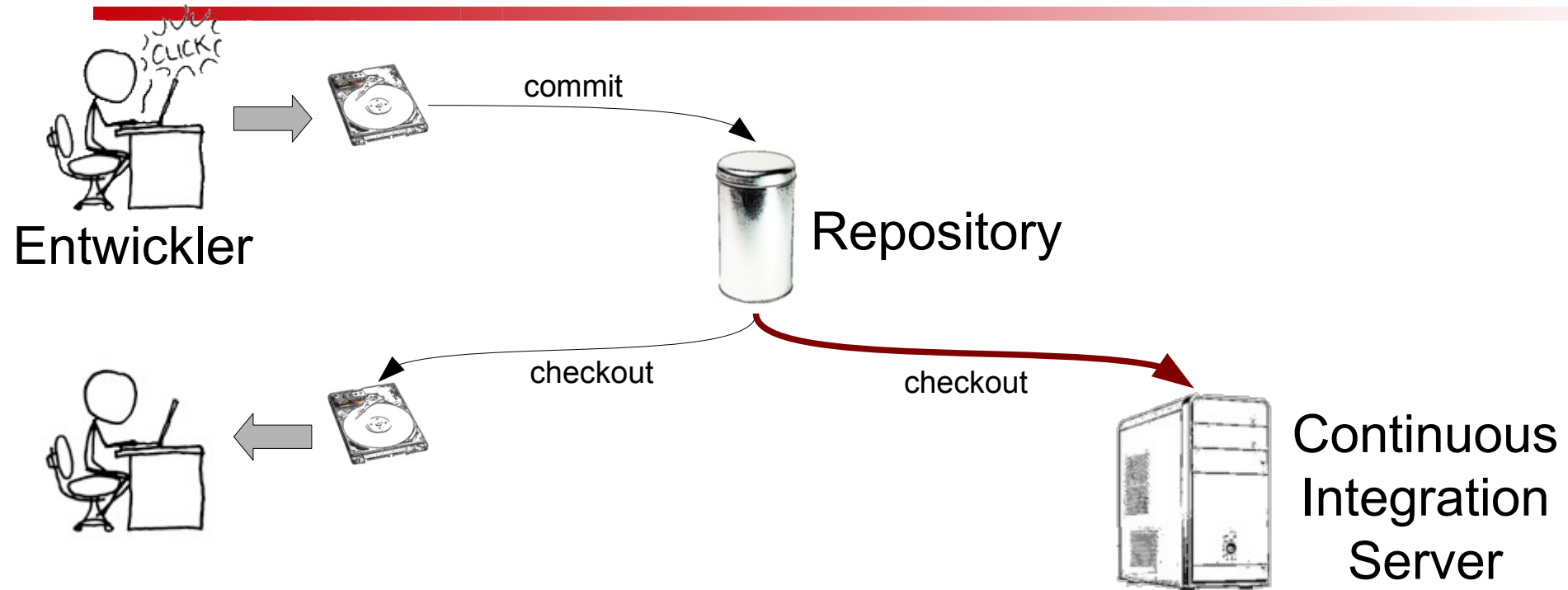
Entwickler



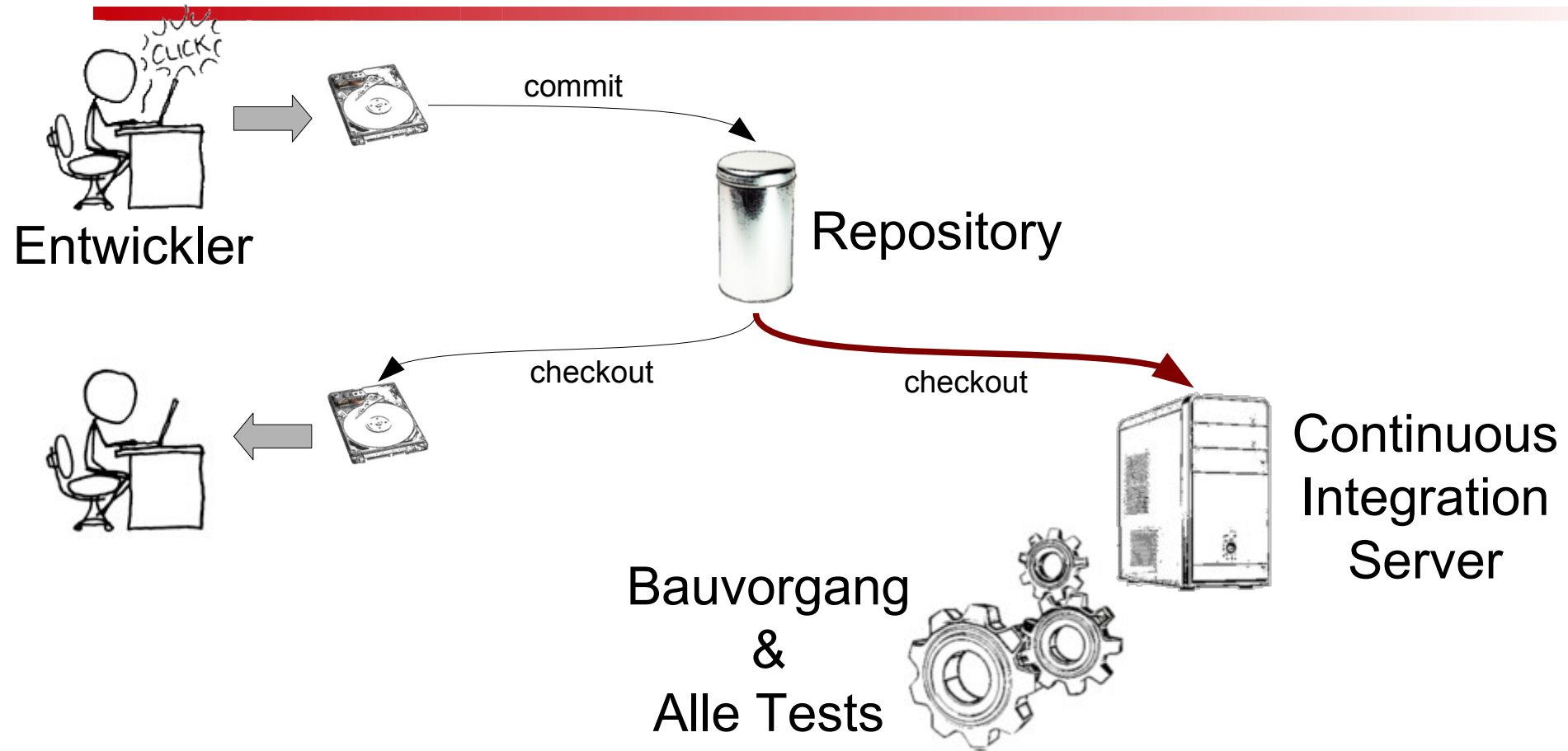
Entwicklungszyklus



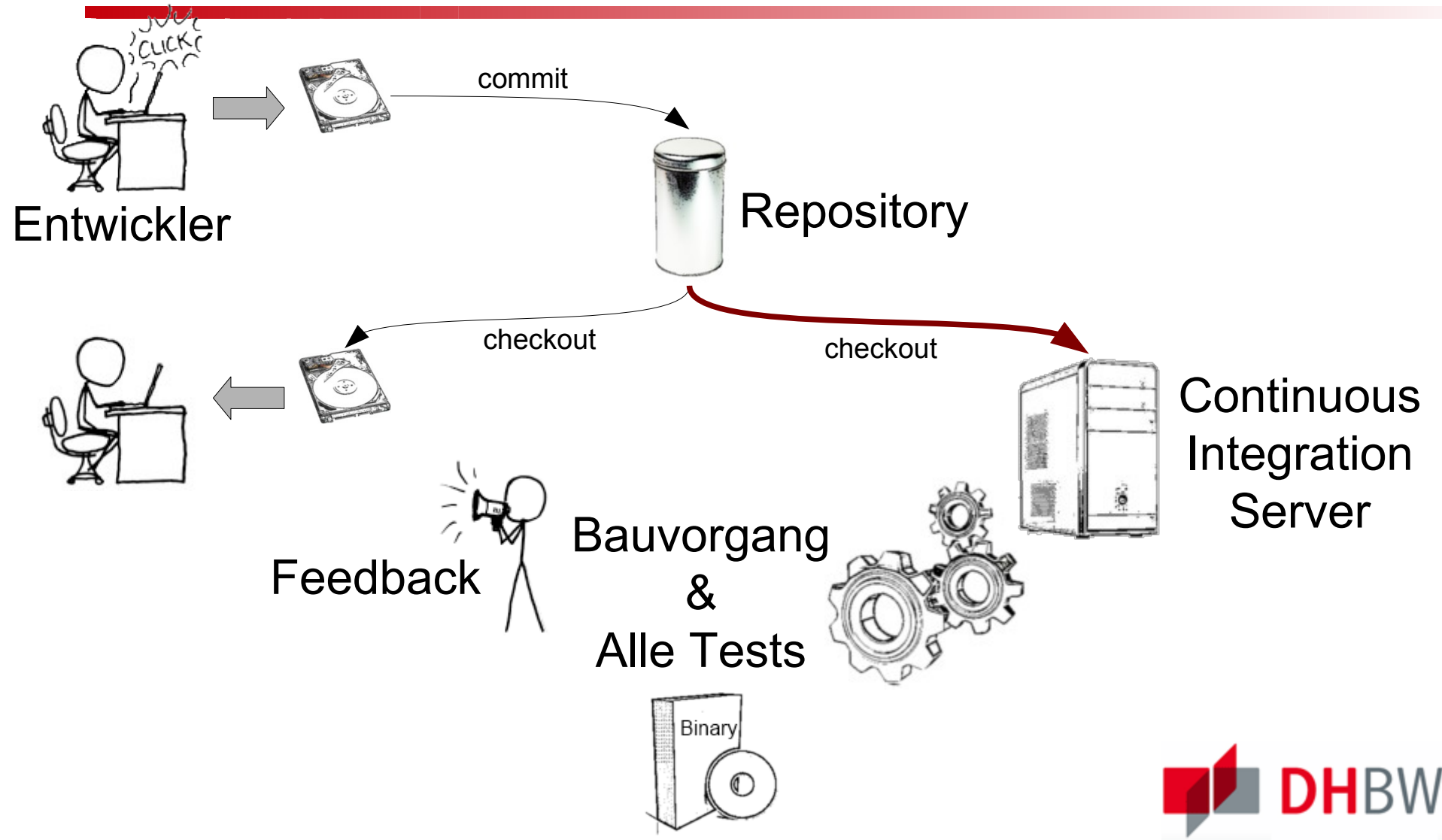
Entwicklungszyklus



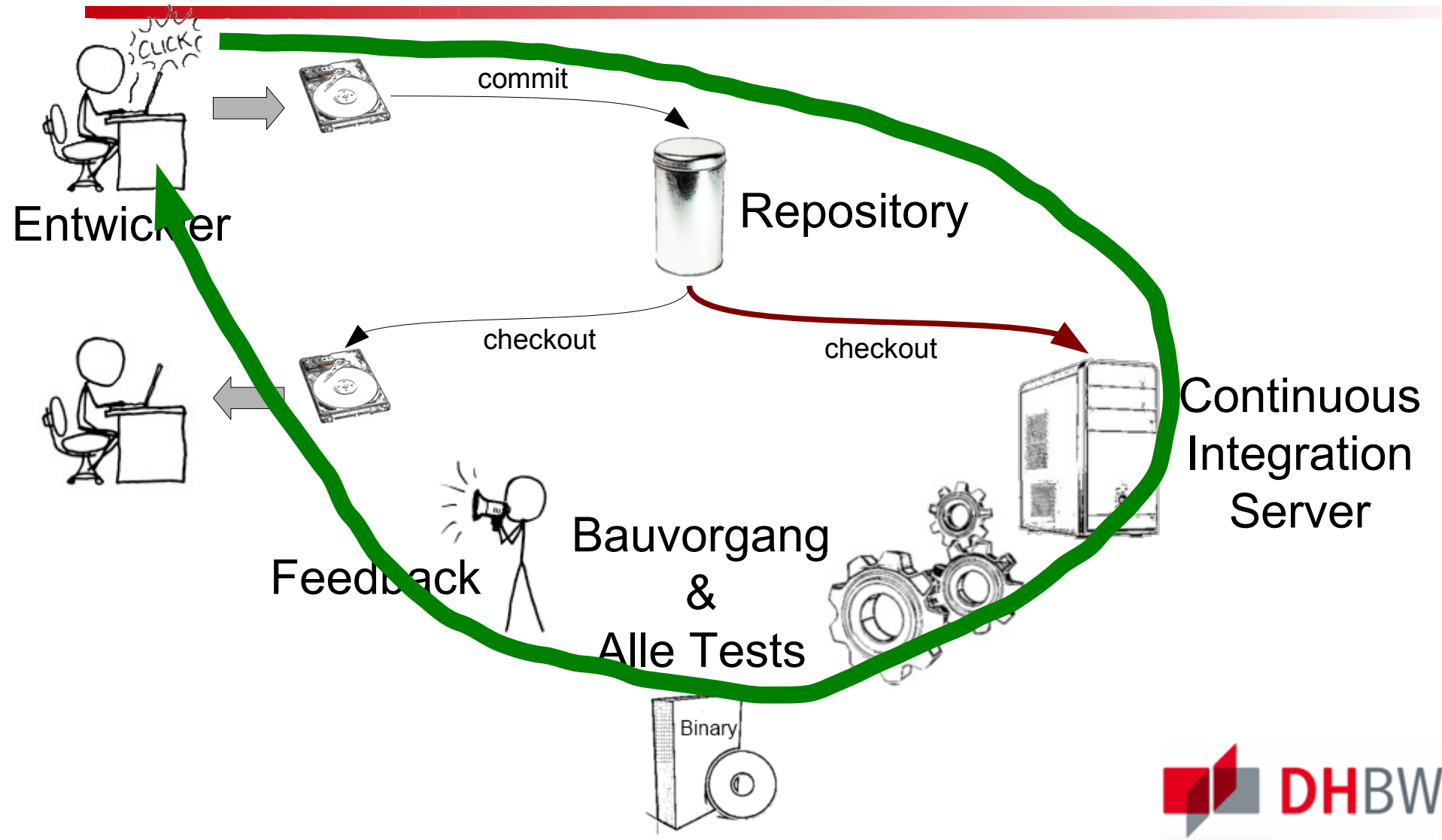
Entwicklungszyklus



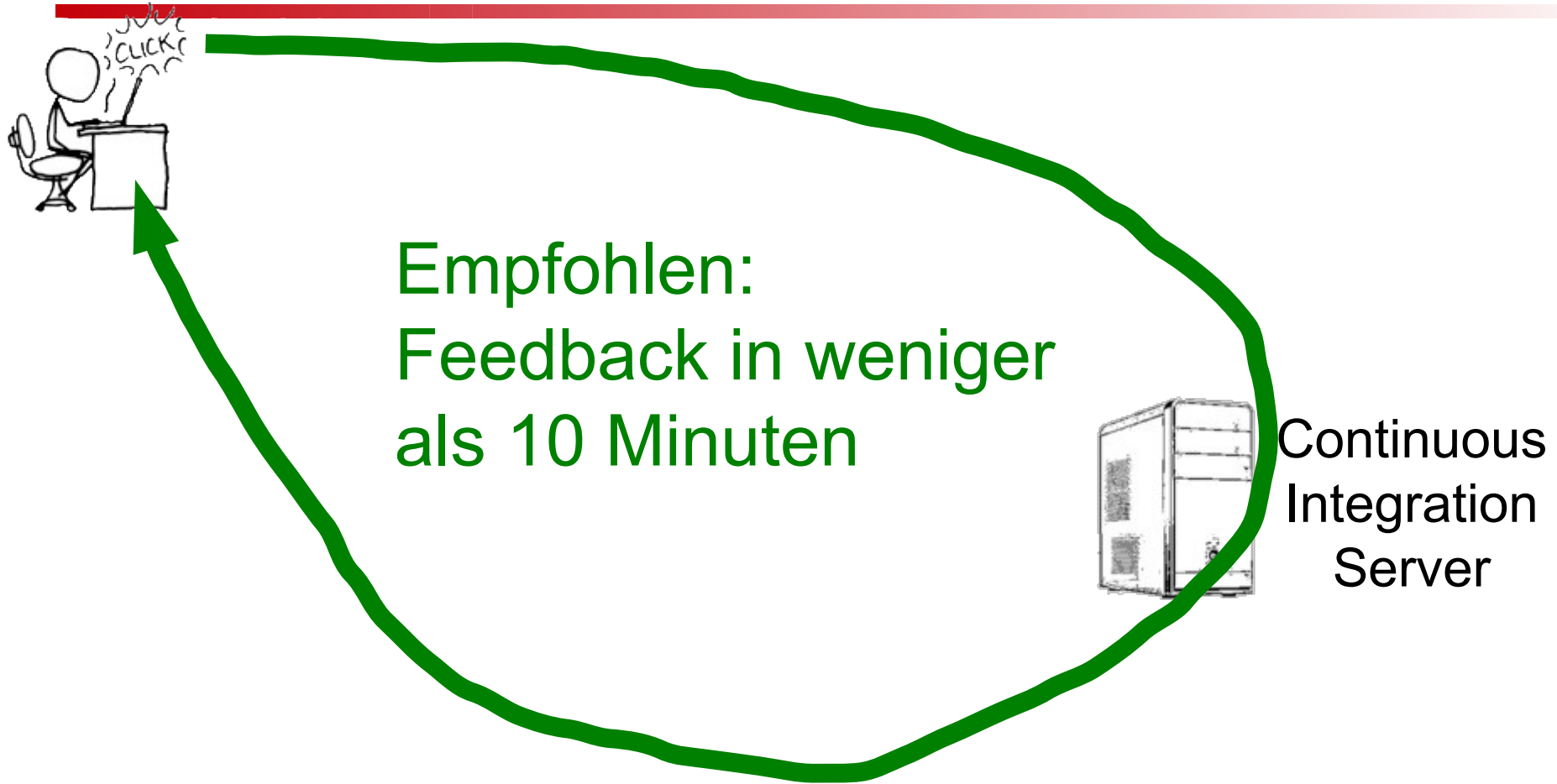
Entwicklungszyklus



Feedback-Schleife



Feedback-Schleife



Prinzipien des Continuous Integration

Verwende ein Repository

Automatisiere den Bauprozess

Lasse den Bau sich selbst überprüfen

Halte den Bauprozess schnell

Prinzipien des Continuous Integration

Jeder committed täglich ins Repository

Jeder Commit löst einen Bauvorgang aus

Jeder kann das Ergebnis des letzten
Bauvorgangs einsehen

Prinzipien des Continuous Integration

Teste in einem Abbild des Echtsystems

Möglichst einfacher Zugriff auf das
neueste Produktpaket („deliverable“)

Automatisiere die Installation

Nachteile von Continuous Integration

Benötigt initialen Einrichtungsaufwand

Benötigt eine gute Testabdeckung, um die Vorteile des Testens zu erlangen

Die Buildbox kann durchaus teuer sein

Vorteile von Continuous Integration

Der letzte funktionierende Zustand ist genau bekannt und im Zugriff

Direkte Meldung von

- Integrationsproblemen (nicht erst zum letztmöglichen Zeitpunkt)
- unvollständigem oder inkompatiblen Code
- konfliktreichen Änderungen

Vorteile von Continuous Integration

Garantiertes Durchtesten jeder Änderung

Immer eine stabile Version verfügbar,
bspw. für Tests oder Demos

Zeitnahes Feedback an die Entwickler,
was Qualität und Funktionalität angeht

Noch mehr Vorteile

Häufiges Einchecken fördert einfachere Lösungen und Modularisierung

Software-Metriken können als Bestandteil des Bauprozesses ermittelt und aufbereitet werden. Qualitäts-Feedback.

Fazit

Continuous Integration ist ein grundlegender Bestandteil moderner Softwareentwicklung

Macht Projektverlauf sichtbar und nachvollziehbar

Erhöht die Qualität eines Projekts bereits durch bloße Anwesenheit
→ Hawthorne-Effekt

Praktische Übung

Wir setzen einen Jenkins CI-Server auf
Schnappen uns ein Projekt von Github
Legen einen CI-Job dafür an
Editieren das Projekt lokal
Und lassen so den CI-Server arbeiten

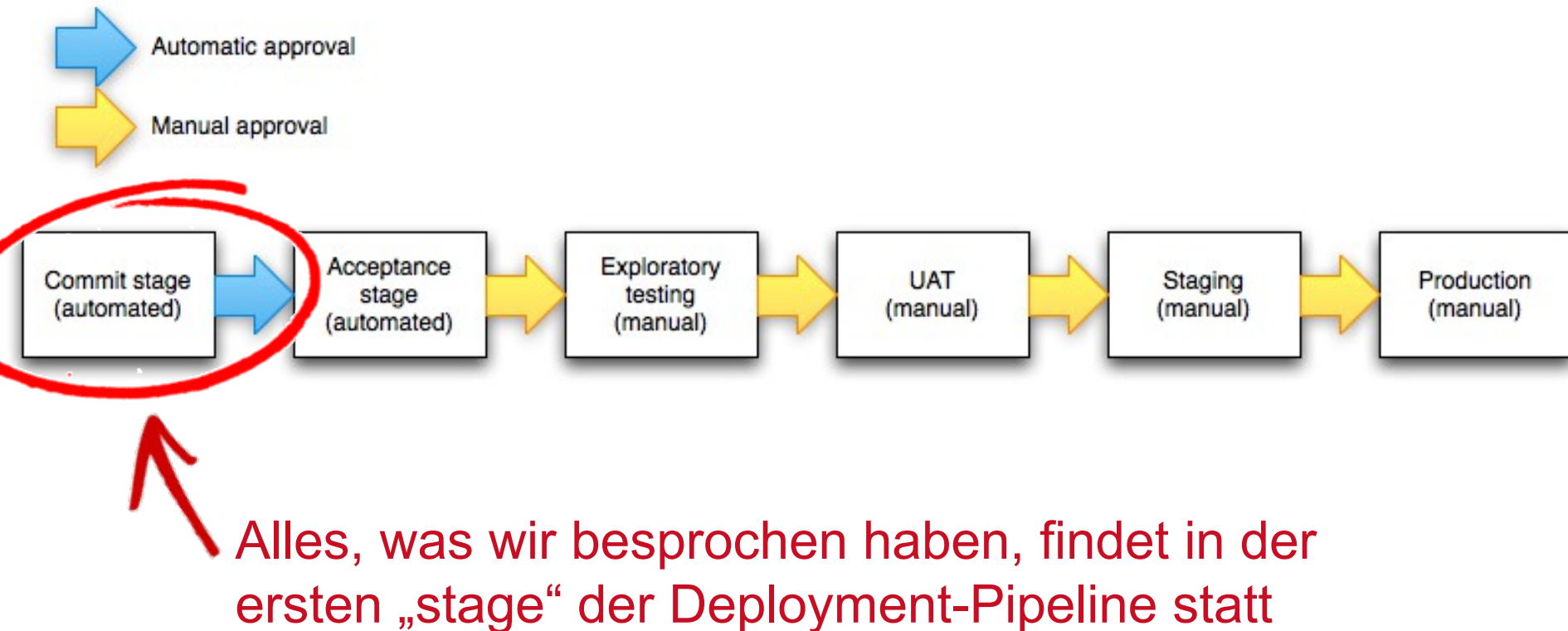
CI weitergedacht: Staging



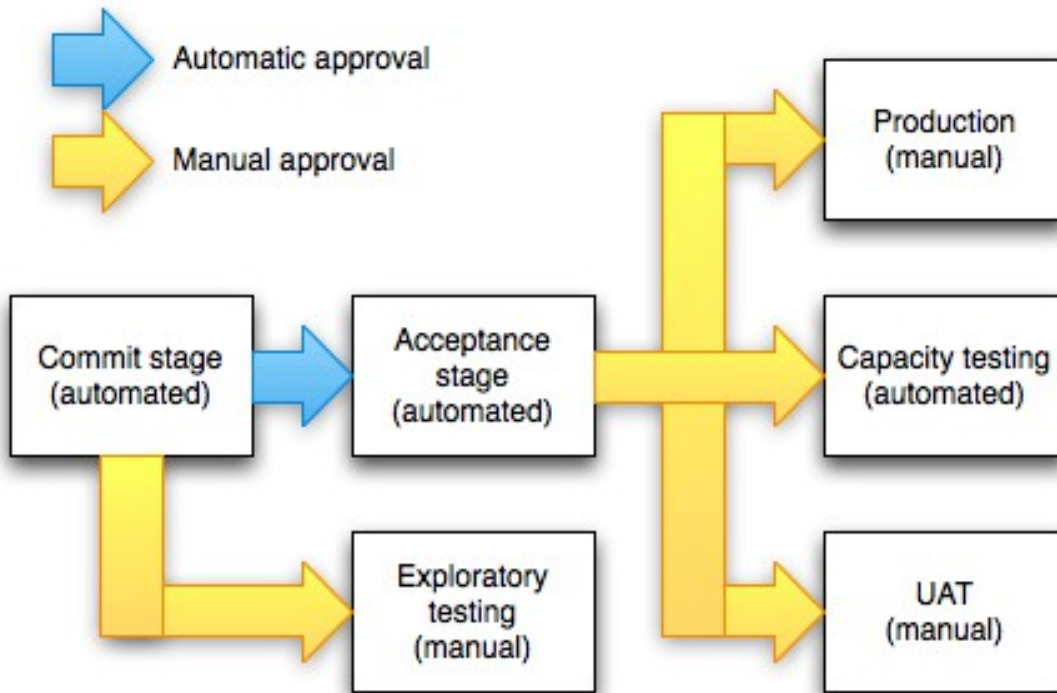
Deployment Pipeline:

Jedes Release durchläuft definierte Stufen (stages), bevor es veröffentlicht wird

CI weitergedacht: Staging



CI weitergedacht: Staging



Die Pipeline muss nicht seriell ablaufen

Parallele stages ermöglichen flexibleren Prozess

Internet-Ressourcen

[http://www.martinfowler.com/articles/
continuousIntegration.html](http://www.martinfowler.com/articles/continuousIntegration.html)

Guter Grundlagen-Überblick

<http://www.jenkins-ci.org>

Bekannter freier CI-Server

Literatur zum Thema

