# Continuous Integration

Sascha Becker

Hochschule Hannover
Fakultät IV – Wirtschaft und Informatik
Studiengang M. Sc. Angewandte Informatik

04. 11. 2016

# Geschrieben von

Sascha Becker
Schwarze-Dorn Str.6
30974 Wennigsen
s.becker@wertarbyte.com

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

Continuous Integration, kurz CI, ist ein Entwicklungsverfahren, welches Entwickler dazu drängt mehrmals am Tag bei einem gemeinsamen Repository beizutragen. Jeder Beitrag wird automatisch verifiziert. Dieser Vorgang erlaubt es dem Team Probleme früh zu erkennen. Durch einen regelmäßigen Beitrag der Entwickler können Fehler schnell erkannt und lokalisiert werden.

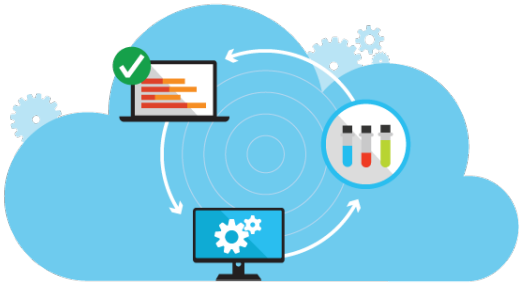

Abbildung 1.1: Continuous Integration Kreislauf [1].

## 1.1 Spät Änderungen integrieren

Ein spät erkannter Fehler führt zu einer schwereren Eingrenzung des Codebereiches. Durch einen regelmäßigen Beitrag zum Code und den damit verbunden automatischen Überprüfungen sind Fehler direkt mit zeitlich kleineren Arbeitseinheiten verknüpft. Dadurch können sie schneller gefunden werden.

> "Continuous Integration doesn't get rid of bugs, but it does make them dramatically easier to find and remove." - Martin Fowler, Chief Scientist, ThoughtWorks [7]

## 1.2 Kosten

Eine ständige Entwicklung mit verbundener Integration und Testphase ist teuer. Denken Sie hier langfristiger! Der angesprochene Zeitfaktor zum Finden ist kritisch. Früh erkannte Fehler sind einfach zu beheben und verbrauchen somit weniger Ressourcen. Wird nicht nach diesem Entwicklungsverfahren gelebt entstehen längere Zeitperioden in dem eine Integration statt findet. Eine solche Zeitspanne bedeutet eine exponentielle Steigerung der nötigen Ressourcen zum Finden und Beheben der Fehler. Sie fahren demnach deutlich besser, wenn Fehler möglichst früh erkannt und behoben werden.

## 1.3 Projektstabilität

Spät erkannte Fehler rauben Ressourcen, welche an anderer Stelle wichtig sind. Werden Mitarbeiter von der Entwicklung zur Fehlerbehebung geschoben ist der Zeitplan in Gefahr. Im schlimmsten Fall können

kritische Fehler das komplette Projekt zu Fall bringen. Mit CI sollten diese Gefahren in akzeptablen Größen auftauchen. Fehlerbehebungen sind gut zu planen und bringen nicht überraschend das Projekt in Gefahr.

## 1.4 Vorteile

Jedes Entwicklungsunternehmen profitiert von vielen Vorteilen gegenüber einer späten Integration dank CI.

- Die langen intensiven Integrationen entfallen
- Ein häufiger Beitrag erhöht die Sichtbarkeit der Codeänderungen für andere Mitarbeiter und fördert somit die Kommunikation
- Probleme werden schnell erfasst und behoben
- Es kann mehr Zeit in die eigentliche Entwicklung anstatt in die Fehlerbehebung investiert werden
- Aufbauende Änderungen basieren auf einem festen Fundament
- Direktes Feedback über Funktionsfähigkeit des neuen Codes
- Eine Reduzierung von Integrationsproblemen erlaubt eine häufigere Auslieferung der Software

# 2 Anforderungen

Continuous Integration muss, wie jedes Entwicklungsverfahren, gelebt werden. Dazu sind einige Anforderungen an das System und Team zu stellen.

## 2.1 System

Es gibt ein gemeinsames Quellverzeichnis in das alle Mitarbeiter ihre Änderungen hinterlegen. Dieses wird regelmäßig gebaut und mit hinterlegten Tests selbständig überprüft. Jede neue Änderung muss auf einem Integrationssystem einen erfolgreichen Schritt des Kompilers mit sich ziehen. Jeder Kompiliervorgang sollte kurz gehalten werden um ein direktes Feedback über die Qualität der Änderung zu erhalten. Es ist immer ratsam die Tests in einem direkten Klon der späteren Produktionsumgebung durchzuführen. Dies verhindert spätere Integrationsprobleme. Die getestete Software sollte für alle zur Verfügung stehen um einen Austausch über den aktuellen Stand zu gewährleisten.

Einen Schritt weiter sollte das Continuous Deployment angesetzt werden um es direkt ausliefern zu können.

### 2.1.1 Umsetzung

Soll CI intern genutzt werden sind einige Schritte nötig um die gewünschten Ergebnisse zu erzielen.

Es wird ein globales Quellverzeichnis angelegte, welches jeder Entwickler nutzen wird. Dazu dienen Git Dienste wie zum Beispiel Github oder Bitbucket. Lokale Änderungen werden getestet und nach Erfolg in das Verzeichnis übertragen. Der CI Server hört auf Änderungen in dem Quellverzeichnis und holt sich den aktuellen Stand. Dieser wird gebaut und mit Unit- und Integrationstests überprüft.

Nach Erfolg wird es dem Team mit einem Namen der Version zur Verfügung gestellt. Eine Benachrichtigung über den Zustand wird nachfolgend an das Team gesendet.

Ist jedoch ein Fehler während des Bauvorgangs aufgetreten wird das Entwicklerteam alarmiert diesen schnellst möglich zu beheben.

Dieser Prozess wird durch das komplette Projekt zyklisch gezogen.

## 2.2 Team

Auf menschlicher Ebene müssen natürlich auch einige Regeln eingehalten werden. Essentiell ist der regelmäßige Änderungsbeitrag jedes Entwicklers in das Quellverzeichnis. Änderungen mit Fehlern im Code gehören nicht den gemeinsamen Pool der Software, sondern sollten vorher lokal syntaktisch korrigiert werden. Code muss lokal immer lauffähig sein bevor darüber nachgedacht wird die Änderung zu veröffentlichen. In vielen Unternehmen wird deshalb die Arbeitszeit nach hinten gezogen bis dies geschehen ist.

Viele Teams entwickeln aus diesen Regeln ein tägliches Ritual. Dadurch korrigieren und kontrollieren sie sich selbst. Es ist nicht nötig weitere Regeln aus höherer Hierarchieebene geben zu müssen.

# 3 Software

Unterstützend gibt es natürlich für alle Prozessschritte bereits Software. Nachfolgende gehen wir auf drei Softwaregestützte Aspekte ein.

## 3.1 Vorlagen

Jedes Projekt muss irgendwo anfangen. Um die Konsistenz zwischen den Projekten innerhalb eines Unternehmens zu gewährleisten sind Vorlagen die ideale Lösung. Im Bereich der Webentwicklung bietet sich zum Beispiel Yeoman [2] an. Es handelt sich um eine Community, welche getestete Vorlagen zur Benutzung anbietet. Viele dieser Vorlagen erlaube sogar eine Selektierung bestimmter verwendeter Techniken.


Abbildung 3.1: Yeoman - THE WEB'S SCAFFOLDING TOOL FOR MODERN WEBAPPS [2].

## 3.2 Quellverzeichnis

Essentiell ist ein zentrales Quellverzeichnis in das alle Entwickler ihre Änderungen veröffentlichen. Dazu kann entweder ein internes git oder svn aufgesetzt werden oder ein online verfügbares und etabliertes System. Zu den größten zählen dabei Github [8] und Bitbucket [9]. Die Unterschiede belaufen sich lediglich auf den öffentlichen oder privaten Nutzen der Quellverzeichnisse.

## 3.3 Automatisches Bauen

Ist eine Zentrale Verwaltung für den Code vorhanden muss nun noch ein System auf Änderungen horchen und etwas damit durchführen. Für öffentliche Projekte bietet sich Travis CI [3] an. Dort können Github Quellverzeichnisse verknüpft und bei Änderungen gebaut werden. Sind es private Angelegenheiten kann ein lokales Jenkins [4] aufgesetzt und genutzt werden. Es bietet ebenfalls das automatische Bauen ist grafisch jedoch nicht so hübsch heraus geputzt wie Travis CI.

(a) Travis CI (b) Jenkins

Abbildung 3.2: Travis CI [3] und Jenkins [4]

## 3.4 Codequalität und Tests

Ist ein eventbasiertes System einsatzbereit kann es mit beliebigen weiteren Tools verknüpft werden. Der größte Fokus liegt oftmals bei Tests und Codequalität. Travis als auch Jenkins unterstützen Unit testing und liefern als Ergebnis aufbereitete Grafiken. Wie viel mit den Tests an Code abgedeckt wird und wie sich sonst um die Qualität des Codes bemüht wurde kann mit z.B. SonarQube [5] oder Code Climate [6] geprüft werden. Diese Erweiterungen können als böser Finger gesehen werden und weisen die Entwickler auf schlechte Praktiken hin.

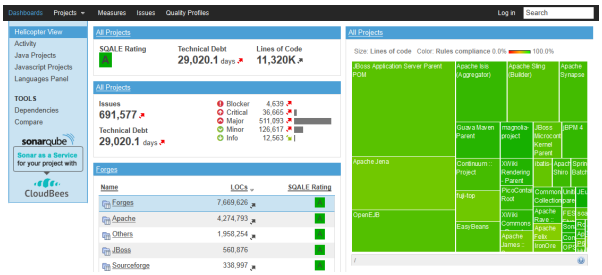


Abbildung 3.3: SonarQube Oberfläche [5].

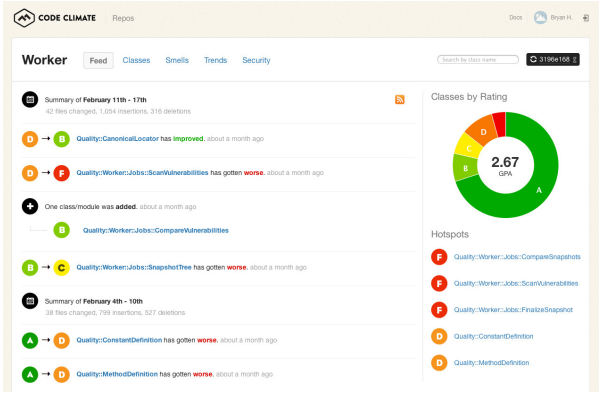


Abbildung 3.4: Code Climate Oberfläche [6].

# Literaturverzeichnis


[1] Matt: *How to use continuous integration to improve your workflow*, Oktober 2016. http://fluent.software/how-to-use-continuous-integration-to-improve-your-workflow/.

[2] The Yeoman Team: *The web's scaffolding tool for modern webapps*, Oktober 2016. http://yeoman.io/.

[3] Travis CI, GmbH: *Test and deploy with confidence*, Oktober 2016. https://travis-ci.org/.

[4] Jenkins: *Jenkins build great things at any scale*, Oktober 2016. https://jenkins.io/.

[5] SonarSource S.A: *Sonarqube open source quality management platform*, Oktober 2016. http://www.sonarqube.org/.

[6] codeclimate: *Healthy code ships faster.*, Oktober 2016. https://codeclimate.com/.

[7] ThoughtWorks, Inc.: *Continuous integration eliminate blind spots so you can build and deliver software more rapidly.*, Oktober 2016. https://www.thoughtworks.com/de/continuous-integration.

[8] GitHub, Inc.: *How people build software*, Oktober 2016. https://github.com/.

[9] Atlassian: *Code, manage, collaborate bitbucket is the git solution for professional teams*, Oktober 2016. https://bitbucket.org/.