# CI - Continuous Integration

## SiW

Finn Welker

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

22-06-2020


# Gliederung (Agenda)

# Prinzip

- Continuous Integration –> Kontinuierliche Integration
- Bearbeitete Kopie ähnelt immer weniger dem Originalprojekt
- Beim Integrieren in Originalprojekt können Probleme auftreten
- Code muss an Originalprojekt angepasst werden –> Zeitintensiv

- Arbeitskopien **häufig** zu Ganzem integrieren

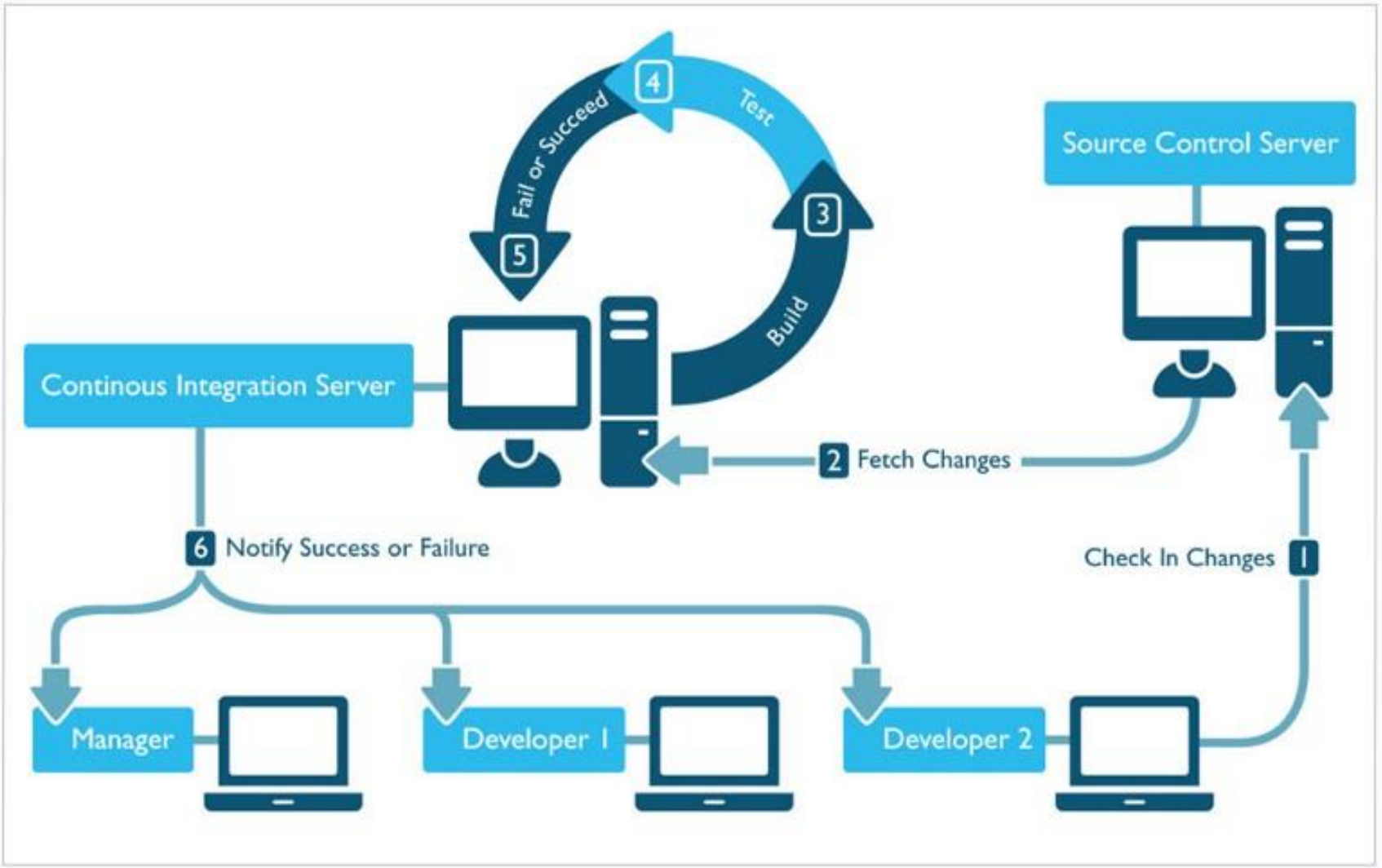
# Prinzip

- Nach jedem Hinzufügen (Commit) wird Projekt kompiliert und getestet
- So können Integrationsfehler aufgedeckt werden
- Automatisierung von Testen und Builden ist optimal

# Hintergrund

- Relativ Jung
- Erste Erwähnung in den 90-ern
- Erstes Tool wurde 2001 released (CruiseControl)

# Beispiel


Source Control Server
Continous Integration Server
Test
Build
Fail or Succeed
4
3
5
2 Fetch Changes
6 Notify Success or Failure
Check In Changes 1
Manager
Developer 1
Developer 2

# Methoden/Voraussetzungen

- Versionsverwaltungsprogramme (Source Control Tools)
- Automatisierte Builds
- Selbsttestender Code
- Builddauer niedrig wie möglich halten (Zeit/Ressourcen)

# Methoden/Voraussetzungen

- Hohe Frequenz von Commits (mind. einmal pro Tag)
- Jeder Commit gefolgt von einem Build
- Tests in ähnlichem Umfeld wie Originalprojekt (Unterschiede durch Hardware, Betriebssystem, etc. vermeiden)
- Änderungen sollten für alle Beteiligten sichtbar sein

# Vorteile

- Integrationsprobleme werden kontinuierlich entdeckt
- Probleme werden schnell behoben
- Lauffähiger Build für Demos oder Tests immer verfügbar
- Positive Auswirkung auf Entwickler (Disziplin)
- Last-Minute Chaos wird verhindert
- Beim Zurückkehren in Buglosen Zustand geht wenig verloren

# Nachteile

- Automatisieren kostet Ressourcen und Zeit
- Nicht unbedingt nötig für kleine Projekte
- In großen Teams können Warteschlangen entstehen
- Manche Branchen (Luftfahrt, etc.) setzen strikte Dokumentation und Reviews voraus die hier nicht im Fokus stehen

# Ähnliche Konzepte

- Extreme Programming
- Nightly Builds
- Continuous Delivery/Deployment

# Programme

- CruiseControl
- Teamcity (Für IDEs wie IntelliJ, Eclipse)
- DroneCI
- CircleCI
  - Drone/Circle unterstützen Cloud-based CI (Teilweise kostenfrei)
  - Aber auch private Server (einfache Installation über GitHub/Bitbucket)

# Zusammenfassung

- Konzept zur Verwaltung größerer Projekte
  - Zusammenfügen der Arbeitskopien zur Mainline (Commit and Build)
  - Dies mit hoher Frequenz
  - Aufdecken/Reparieren von Integration-Fehlern wird einfacher
  - Gut über GitHub, Drone und Circle möglich

# Nutzen für uns

- Weniger nützlich im Studium, aber sehr interessant in der Industrie
- Wirkungsvoller bei größeren Projekten
- Definitiv im Gedächtnis behalten, interessantes Konzept
- Implementation über GitHub ziemlich einfach (bei Interesse s. Quellen)

# Quellen


- martinfowler.com/articles/continuousIntegration.html
- en.wikipedia.org/wiki/Continuous_integration
- circleCI.com / drone.io
- github.com/features/actions
- Literatur: "Continuous Integration"by Paul Duvall, Steve Matyas, and Andrew Glover
- Quelle
  Bild: https://developers.redhat.com/blog/2017/09/06/continuous- integration-a-typical-process/

  - Quelle Code aus GitLab Beispiel:

  www.youtube.com/watch?v=Jav4vbUrqII