Software erstellen

Nicht triviale Software wird im Allgemeinen mithilfe von automatisierten Build Systemen erstellt. Das Ziel eines automatisierten Builds ist die vollständige Automatisierung aller Schritte, die zum Erstellen des Produkts erforderlich sind. Dabei sollte das Ergebnis eines jeden Builds immer dasselbe sein – unabhängig von der lokalen Konfiguration des Entwicklers. „Wir wollen stabile Builds.“ Das automatisierte Build System ist für die automatische Ausführung aller zum Bau erforderlichen Schritte das Produkt verantwortlich. Eine Build-Automatisierung führt normalerweise die folgenden Aufgaben aus:

- Formatieren des Quellcodes
- Codegenerierung
- Quellcode-Kompilierung
- Verlinkungscode / Verpackungscode (falls erforderlich)
- Ausführen der Tests
- Ausführen statischer Analysewerkzeuge
- Bereitstellung auf dem Testsystem / Produktionssystem (en)
- Erstellen und Veröffentlichen von Dokumentation, Versionshinweisen, Webseiten

Es gibt folgende automatisierten Builds:

- **On-Demand** (z. B. von einem Entwickler)
- **Scheduled** oder **geplant** von einem Build-Server (z. B. jede Nacht)
- **Triggered** (z. B. bei jedem Commit an ein Versionskontrollsystem)

Automatisieren von Builds

Beispiele für (Open-Source) -Tools zum Automatisieren von Builds

- Apache Ant
- Apache Maven
- gradle (Groovy Based)
- RAKE (Ruby Make)
- sbt

Continuous Integration

Continuous Integration bedeutet im Grunde nur, dass Arbeitskopien des Entwicklers mehrmals täglich mit einer gemeinsamen Master-Branche synchronisiert werden. Dies wurde zuerst von Grady Booch benannt und vorgeschlagen.

- Ziel ist es, Integrationsprobleme zu vermeiden.
- Continuous Integration ist besonders nützlich in Kombination mit automatisierten Komponententests.
- In der Praxis wird ein spezieller Build-Server verwendet (z.B. Jenkins)
- Pflege eins Code-Repository
- Automatisieren des Builds

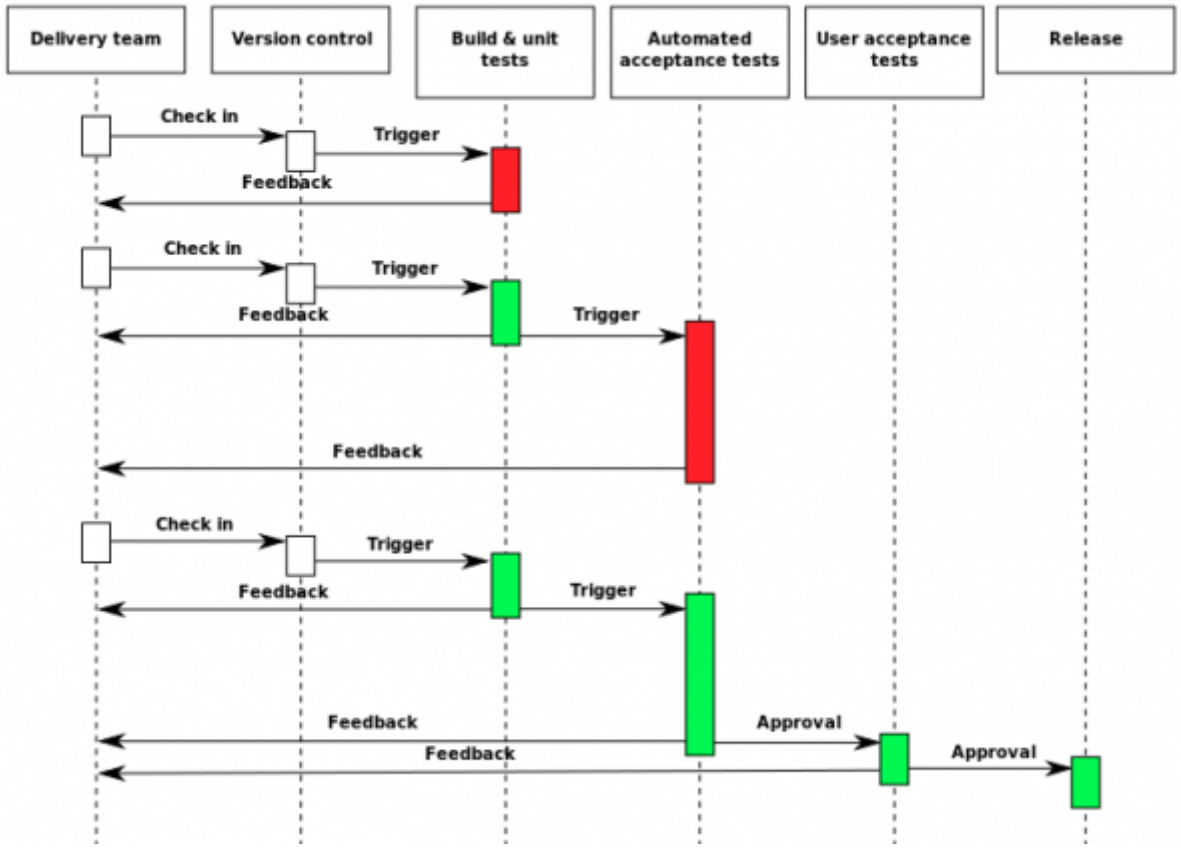
- automatisiertes Durchführen von Tests
- Commits werden an die Master-Branche durchgereicht
- jeder Commit wird dokumentiert
- Ein Commit – ein Feature
- Halten Sie den Build schnell
- einfaches Testen durch Klonen in die Produktionsumgebung
- einfach, die neuesten Entwicklungen zu sehen
- jeder kann die Ergebnisse des neuesten Builds sehen
- Automatisierung der Bereitstellung

Travis CI

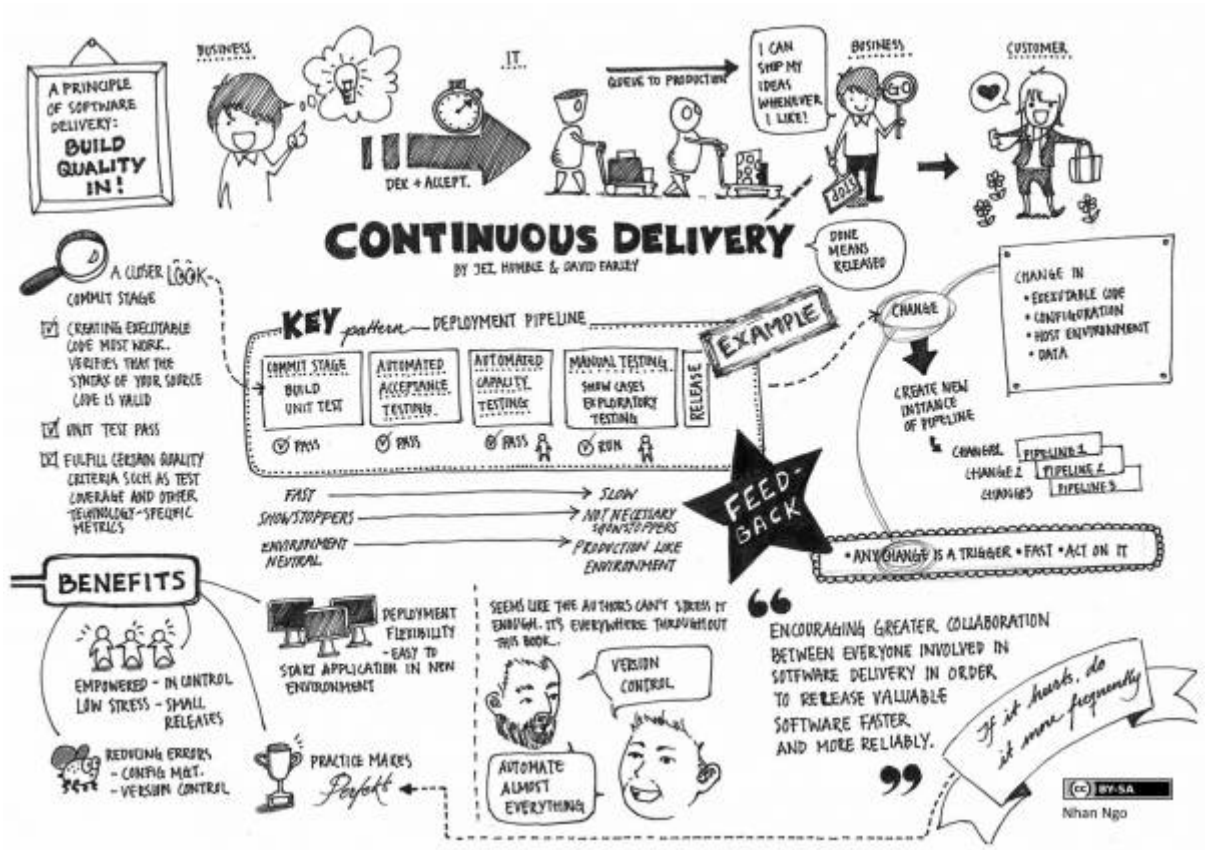
Ein gehosteter Continuous Integration Service für Open Source und private Projekte.

Continuous Delivery

- Immer in der Lage sein, ein Produkt in Produktion zu bringen
- Unit- / Abnahmetests
- Codeabdeckung und statische Analyse
- Bereitstellung in einer Integrationsumgebung
- Integrationstests
- Bereitstellungen in einer Testumgebung
- Leistungstests
- Benachrichtigungen, Berichte und Versionshinweise werden gesendet
- Bereitstellung zur Freigabe des Repositorys



Quelle: Wikipedia



Quelle: continuousdelivery.com

Stellen Sie Ihr Produkt automatisch der Produktion bereit, wenn es die Qualitätssicherung erfolgreich durchlaufen hat. Dadurch liegt der Veröffentlichungszeitplan in der Hand der IT-Abteilung des Unternehmens.

Achtung: Manchmal wird der Begriff „Continuous Deployment“ auch verwendet, wenn Sie die kontinuierliche Bereitstellung auf einem Testsystem durchführen können.

Projekte werden mithilfe von Build-Tools erstellt, dabei kümmert sich ein Build-Skript um alle Schritte, die zum Erstellen des Projekts erforderlich sind. Im Falle einer Anwendung bedeutet Erstellen, eine ausführbare Anwendung zu erstellen, z.B. für Windows eine .exe Datei.

From:
https://wiki.haberland.it/ - **haberland.it**

Permanent link:
https://wiki.haberland.it/doku.php?id=projekte.haberland.it:software-engineering:software-erstellen

Last update: **2020/05/12 11:45**