Continuous Integration

für ein gutes Zusammenspiel

Kevin Kessenich
Software Architekt @CofinproAG

Kevin Kessenich

Software Architekt

> Ne echte Kölsche Jung

COFINPRO

Management-, Fach- und Technologieberatung für Deutschlands führende Banken und Kapitalverwaltungsgesellschaften. Als Experten für Kredit und Wertpapier begleiten und navigieren wir unsere Kunden durch die Herausforderungen von Digitalisierung, neuen Marktanforderungen und Regulatorik.

Warum

Wie wir Software entwickeln

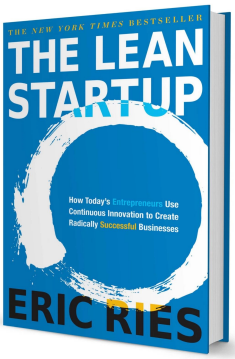
Business Value

Schnell an den Markt anpassen

Build
Ideen in Produkte umwandeln

Measure
Reaktionen der Kunden messen

Learn
Aus Kundenverhalten lernen



https://sniederm.files.wordpress.com/2018/06/eric-ries-lean-startup.jpeg

Öfter und schneller releasen

Das richtige bauen und schnell an den Markt / Kundenwünsche anpassen

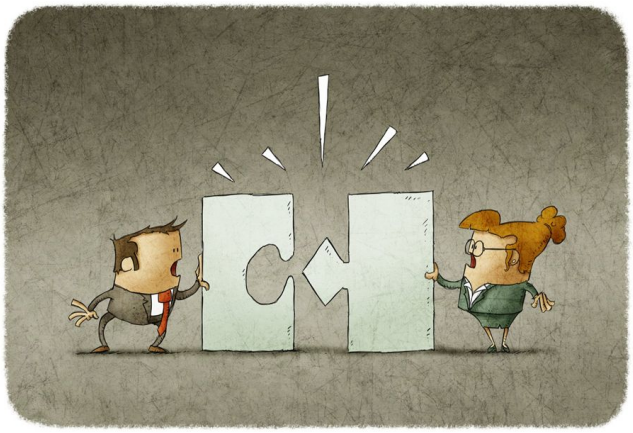
Im Team Software erstellen

Verschiedene Rahmenbedingungen

- SCRUM, XP,...
- Cross-Functional
- Mehere Teams
- Verteilt
- Mehrsprachig

Software ist ein großes Puzzle

Die Puzzlestücke müssen zusammen passen



https://www.allbusiness.com/asset/2016/10/Puzzles-dont-fit..jpg

Integrations-Hölle
Entwickler, die in Teams arbeiten, werden bestätigen, dass der zeitraubendste Task bei der gemeinsamen Erstellung einer Software in der Zusammenführung der einzelnen Code-Komponenten, geliefert von unterschiedlichen Entwicklern, besteht. Dabei wird der Task immer schwieriger, je größer das Team ist, und noch größer, wenn sogar mehrere Teams beteiligt sind. In der Vergangenheit wurde daher mit strikten Deadlines und zwischengeschalteten Meilensteinen gearbeitet, ab deren Erreichung nur noch zusammengeführt und Fehler beseitigt wurden.

https://t3n.de/news/software-entwicklung-vorteile-1246651/

Agile Manifest

Prinzipien hinter dem Agilen Manifest

Unsere höchste Priorität ist es, den Kunden durch frühe und **kontinuierliche Auslieferung wertvoller Software** zufrieden zu stellen.

Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen **Veränderungen** zum Wettbewerbsvorteil des Kunden.

Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.

Fachexperten und Entwickler müssen während des Projektes täglich **zusammenarbeiten**.

Funktionierende Software ist das wichtigste Fortschrittsmaß.

...

https://agilemanifesto.org/iso/de/principles.html

Unter anderem: Kent Beck, Alistair Cockburn, Martin Fowler, Brian Marick, Robert C. Martin, Ken Schwaber, Jeff Sutherland
Im Jahr **2001**

1. Unsere höchste Priorität ist es, den Kunden durch frühe und kontinuierliche Auslieferung wertvoller Software zufrieden zu stellen.
2. Heisse Anforderungsänderungen selbst spät in der Entwicklung willkommen. Agile Prozesse nutzen Veränderungen zum Wettbewerbsvorteil des Kunden.
3. Liefere funktionierende Software regelmäßig innerhalb weniger Wochen oder Monate und bevorzuge dabei die kürzere Zeitspanne.
4. Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
5. Errichte Projekte rund um motivierte Individuen. Gib ihnen das Umfeld und die Unterstützung, die sie benötigen und vertraue darauf, dass sie die Aufgabe erledigen.
6. Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
7. Funktionierende Software ist das wichtigste Fortschrittsmaß.
8. Agile Prozesse fördern nachhaltige Entwicklung. Die Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.

1. Ständiges Augenmerk auf technische Exzellenz und gutes Design fördert Agilität.
2. Einfachheit -- die Kunst, die Menge nicht getaner Arbeit zu maximieren -- ist essenziell.
3. Die besten Architekturen, Anforderungen und Entwürfe entstehen durch selbstorganisierte Teams.
4. In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.

Was

Was CI und CD ist

Continuous X

Definitionen

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*

Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time. - *Martin Fowler*

Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day. - *Martin Fowler*

https://martinfowler.com/articles/continuousIntegration.html

https://martinfowler.com/bliki/ContinuousDelivery.html

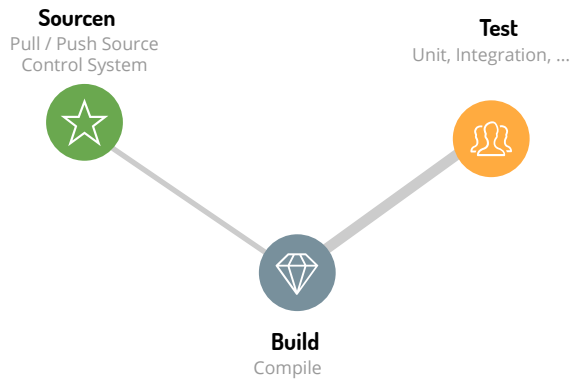
Continuous Delivery just means that you are **able to do frequent deployments** but may choose not to do it, usually due to businesses preferring a slower rate of deployment

fast, automated **feedback** on the production readiness of your applications every time there is a change - to code, infrastructure, or configuration

- Your software is deployable throughout its lifecycle
- Your team prioritizes keeping the software deployable over working on new features
- Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them
- You can perform push-button deployments of any version of the software to any environment on demand

Continuous Integration Pipeline

Merge-Konflikte vermeiden



https://www.redhat.com/de/topics/devops/what-is-ci-cd

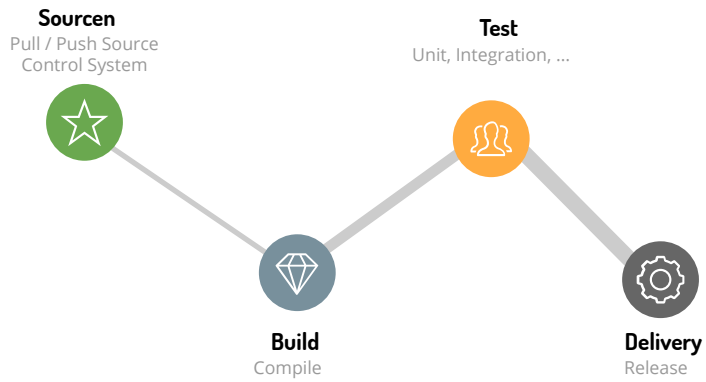
- Entwickler führen mindestens einmal am Tag einen **Check-In** ihres Codes durch
- Der Code wird bei jedem Check-In gebaut
- Code wird bei jedem Check-In automatisch mit **Unit Tests** getestet
- Jeder hat Zugriff auf den Build und die Testreports
- Der Build ist **schnell**, sodass Entwickler schnell Feedback bekommen
- Tests werden in einer herunterskalierten Version der Produktionsumgebung ausgeführt
- **Build-Artefakte** werden in einem versionskontrollierten Artefakt-Repository abgelegt
- Build-Artefakte werden nach jedem erfolgreichen Build automatisch auf eine Testumgebung deployt

https://www.scrum.de/unterschiede-zwischen-continuous-integration-continuous-delivery-und-continuous-deployment/

https://dzone.com/articles/learn-how-to-setup-a-cicd-pipeline-from-scratch

Continuous Delivery Pipeline

Automatisches Release in Repository



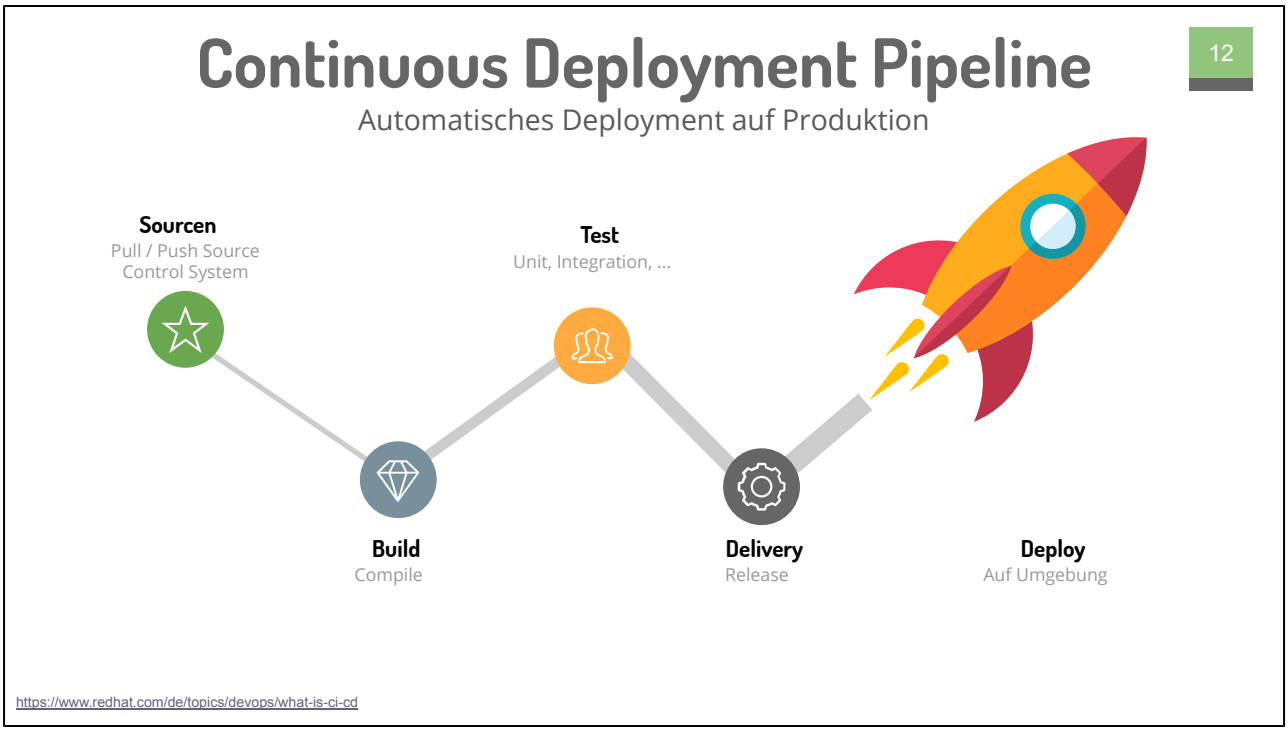
https://www.redhat.com/de/topics/devops/what-is-ci-cd

- Deine Software ist den gesamten Lebenszyklus hindurch **deploybar**
- Dein Team priorisiert das Sicherstellen von auslieferungsfähiger Software über das Umsetzen neuer Features
- Jeder erhält schnelles, automatisiertes **Feedback** über den Auslieferungszustand Deiner Systeme, jedes Mal, wenn jemand eine Änderung daran vornimmt
- Sie führen bei Bedarf Deployments jeder beliebigen Version der Software per Knopfdruck durch
- Es gibt eine enge, kollaborative Arbeitsbeziehung zwischen allen, die am Auslieferungsprozess beteiligt sind (häufig als **DevOps-Kultur** bezeichnet)
- Umfassende **Automatisierung aller möglichen Bereiche** des Auslieferungsprozesses sind umgesetzt, üblicherweise mit einer Deployment Pipeline.[1]

- Automatisiere die Regressionstests,
- Deployment in Produktion automatisiert
- Es ist **DONE!** Kann direkt in Produktion
- Der Schritt in die Produktion ist eine Entscheidung der Businessseite.

https://www.scrum.de/unterschiede-zwischen-continuous-integration-continuous-delivery-und-continuous-deployment/

https://dzone.com/articles/learn-how-to-setup-a-cicd-pipeline-from-scratch



- Jede Änderung geht **direkt in Produktion**. Somit ist das Zeitfenster für eine verlorene Gelegenheit sehr klein.
- Feedback erfolgt noch schneller.
- Durch den Einsatz von **Feature Toggles** ist es möglich, nach Produktion zu deployen, ohne die neuen Funktionalitäten zu nutzen. Es sind also Teile, die noch unfertig sind, bereits in Produktion deployt. So kann bereits Feedback über das Deployment der neuen Teile gesammelt werden.
- Daneben ist es auch möglich, eine kleine Nutzergruppe eine neue Funktionalität bereits testen zu lassen, um auch so frühes Feedback zu bekommen.

https://www.scrum.de/unterschiede-zwischen-continuous-integration-continuous-delivery-und-continuous-deployment/

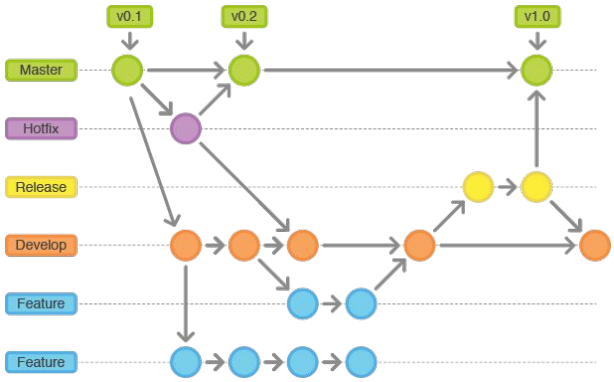
https://dzone.com/articles/learn-how-to-setup-a-cicd-pipeline-from-scratch

Was Genau

Ein Blick auf die einzelnen Steps

Source Control System

Zentraler Ablageort für Sourcen und Konfigurationen



git

SUBVERSION

mercurial

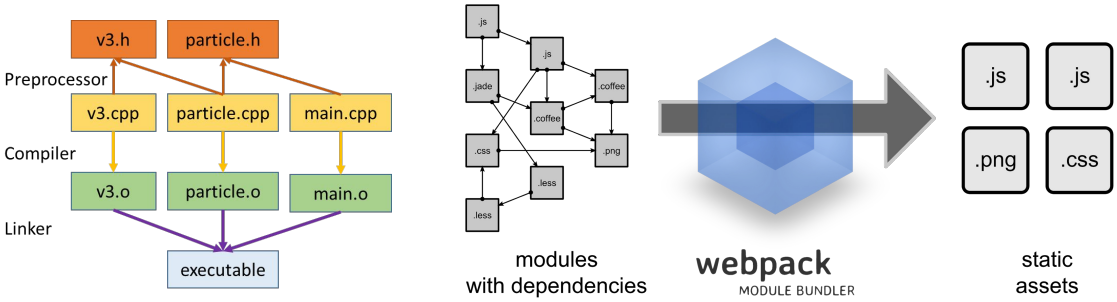
TFS

https://leanpub.com/git-flow/read

You must put everything required for a build in the source control system, however you may also put other stuff that people generally work with in there too.
IDE configurations are good to put in there because that way it's easy for people to share the same IDE setups.

Build

Komilieren der Sourcen



https://devblogs.nvidia.com/separate-compilation-linking-cuda-device-code/

http://webpack.github.io/

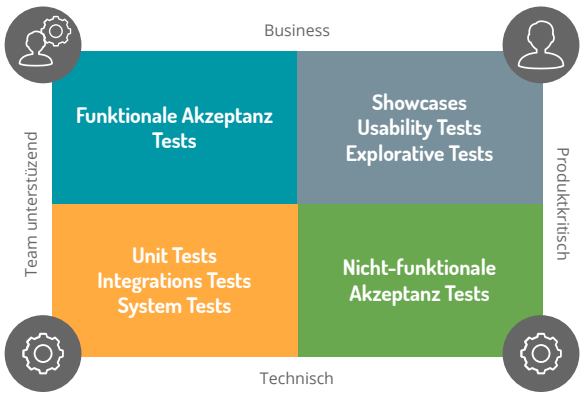
What is a Build? - Compile vs. CI Build

A build is much more than a compile. A build may consist of the compilation, testing, inspection, and deployment – among other things.

A build acts the process for putting source code together and verifying that the software works as a cohesive unit.

Test

Brian Marick's Test Quadrant



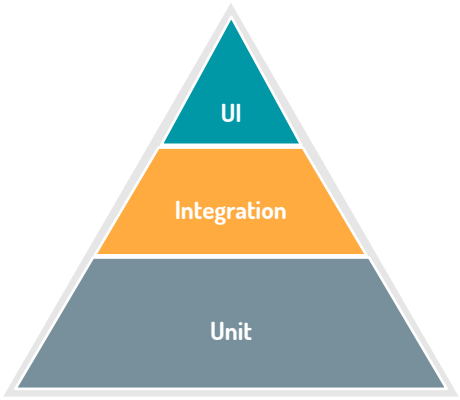
https://lisacrispin.com/2011/11/08/using-the-agile-testing-quadrants

Test

Test Pyramide

Langsam

Schnell



€€€ Teuer

€ Günstig

https://martinfowler.com/bliki/TestPyramid.html

Test

Code Analyse

sonarqube

DEPENDENCY-CHECK

ESLint

stylelint

Delivery

Paketierung und Release

- Paketierung der Anwendung und allen benötigten Dateien für das Deployment
- Configuration Management; jegliche Konfiguration der Umgebung, die benötigt wird. Bsp.: Datenbank Scripte
- Release in ein Repository / zentrale Stelle

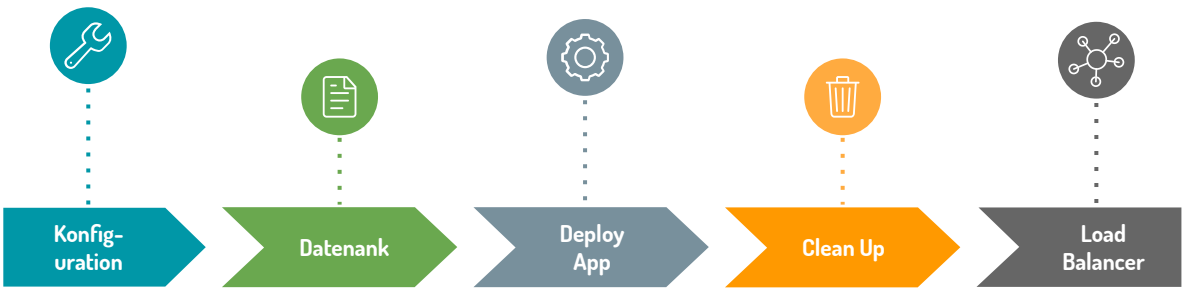
https://martinfowler.com/bliki/ContinuousDelivery.html

Make it Easy for Anyone to Get the Latest Executable

One of the most difficult parts of software development is making sure that you build the right software. We've found that it's very hard to specify what you want in advance and be correct; people find it much easier to see something that's not quite right and say how it needs to be changed. Agile development processes explicitly expect and take advantage of this part of human behavior.

Deploy

Deployment besteht aus mehreren Schritten



Deployment kann auf Dev-, Test-, und Produktivumgebung sein

Reihenfolge und Steps können variieren
Meistens ist es nicht nur ein Deploy, sondern auch Konfiguration und Datenbank Änderungen
Am Ende gibt es oft ein Umstellen von Server A auf Server B im Load Balancer
Load Balancer auch für A/B Testing nutzen
Wichtig: Sicherstellung eines Rollback

Wie

Wie eine Build Pipeline definiert ist

Workflow / Pipeline

Aufbau des Build Workflow

Workflow / Pipeline

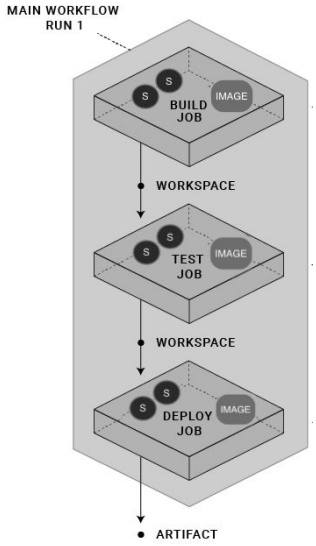
- Beschreibt den gesamten Build Prozess
- Besteht aus Jobs/Stages die sequentiell, parallel, zeitgesteuert oder “manuell” laufen

Jobs / Stages

- Sammlung von Steps
- Beispiele: Build, Test, Deploy

Steps

- Scripte (sh) / Befehle die ausgeführt werden



https://circleci.com/docs/2.0/concepts/#section=getting-started

https://jenkins.io/doc/book/pipeline/#pipeline-concepts

Image / Node

An image is a packaged system that has the instructions for creating a running container. This is where commands are executed for jobs using the Docker or machine executor.

A node is a machine which is part of the Jenkins environment and is capable of executing a Pipeline.

Datenhaltung

Storage Varianten

Workspace

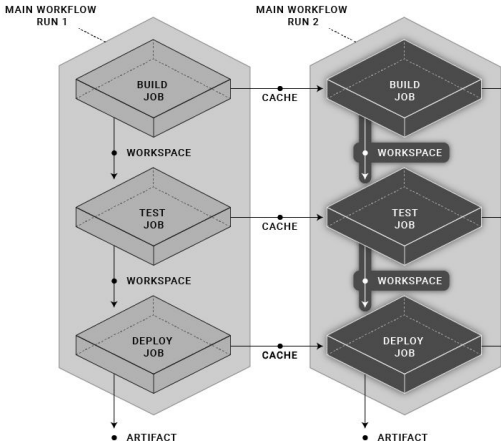
- Dauer: 1 Build
- Daten für den jeweiligen Build

Cache

- Dauer: Monate
- Dependencies, etc.

Artifact

- Dauer: Monate
- Output vom Build Prozess (Logs, Exec,..)



https://circleci.com/docs/2.0/concepts/#section=getting-started

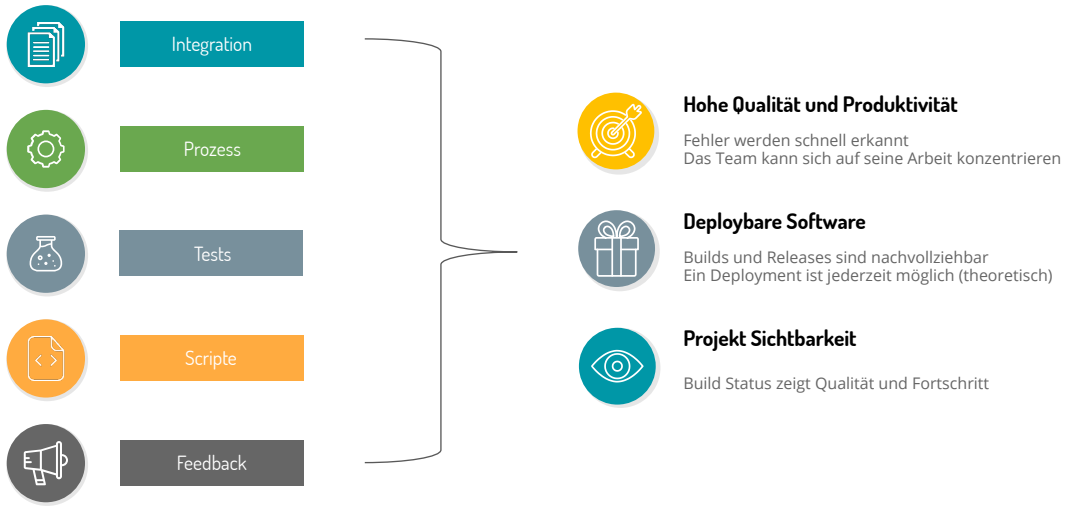
Gitlab und Jenkins
Github und CircleCI

Wie Richtig

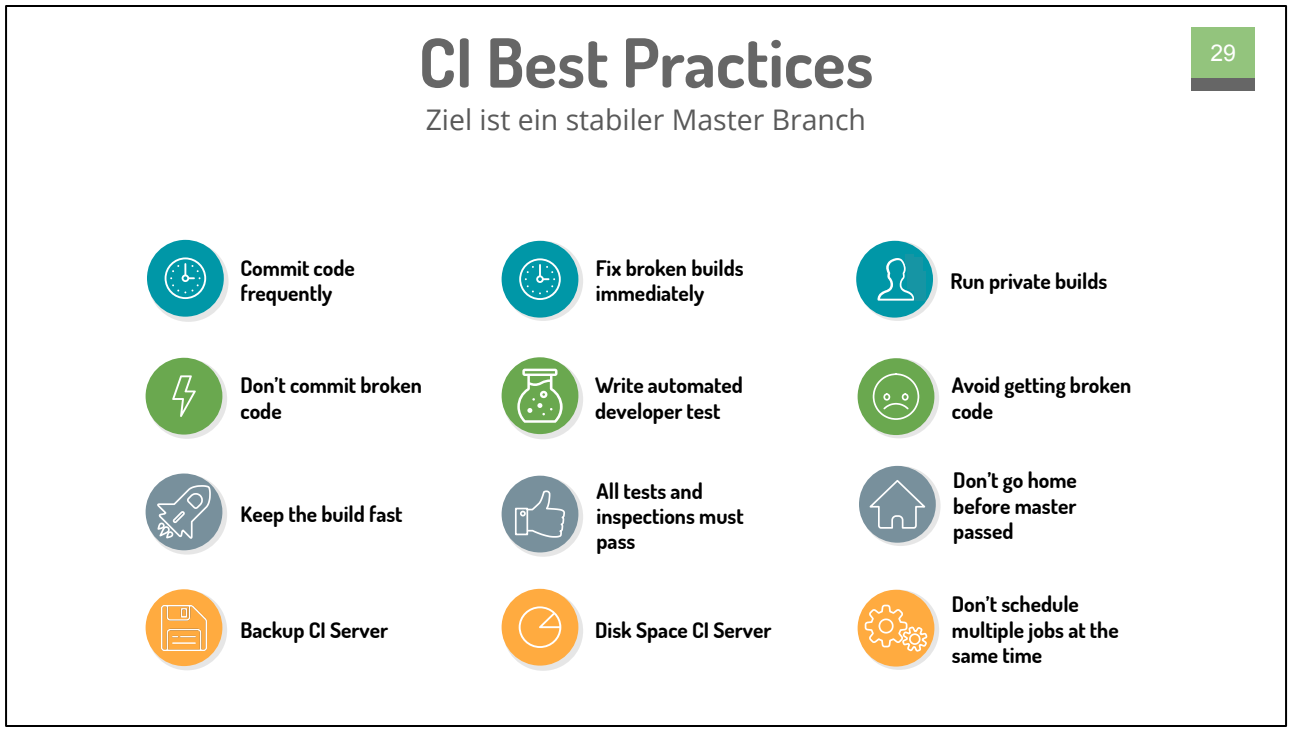
Vorteile und Best Practices

Vorteile von CI/CD

Schnell qualitative passende Software liefern



Integration - Automatische Integration durch Push/Pull
Prozess - Reproduzierbarer und automatisierter Prozess
Tests - Automatische Testdurchführung und Code Standard Checks
Scripte - Ausführung von Scripten abhängig zu Anforderungen
Feedback - Schnelle Feedback, Build Status



Commit code frequently
Commit code to your version control repository at least once a day.
=> Diff Debugging; The more frequently you commit, the less places you have to look for conflict errors, and the more rapidly you fix conflicts.
=> See conflicts as soon as possible

Don't commit broken code
Don't commit code that does not compile with other code or fails a test.

Fix broken builds immediately
Although it's the team's responsibility, the developer Who recently committed code must be involved in fixing the failed build.
Kent Beck using "nobody has a higher priority task than fixing the build"

Write automated developer tests
Verify that your software works using automated developer tests. Run these tests With your automated build and run them often With CI.

All tests and inspections must pass
Not 90% or 95% of tests, but all tests must pass prior to committing code to the version control

repository.

Run private builds
To prevent integration failures, get changes from other developers by getting the latest changes from the repository and run a full integration build locally, known as a private system build.

Avoid getting broken code
If the build has failed, you will lose time if you get code from the repository. Wait for the change or help the developer(s) fix the build failure and then get the latest code.

Keep the build fast (https://martinfowler.com/articles/continuousIntegration.html)
For most projects, however, the XP guideline of a ten minute build is perfectly within reason. Most of our modern projects achieve this. It's worth putting in concentrated effort to make it happen, because every minute you reduce off the build time is a minute saved for each developer every time they commit. Since CI demands frequent commits, this adds up to a lot of time.

you shouldn't go home until the mainline build has passed with any commits you've added late in the day

Backup the home directory of the CI server regularly as it contains archived builds and other artifacts too, which may be useful in troubleshooting.
Make sure the CI server has enough free disk space available as it stores a lot of build-related details.
Do not schedule multiple jobs to start at the same time, or use a master-slave concept, where specific jobs are assigned to slave instances so that multiple build jobs can be executed at the same time.

Source: Continuous Integration: Improving Software Quality and Reducing Risk - Paul M. Duvall - 2007

Ganzheitlicher Ansatz

28

Gemeinsam Richtung Ziel

> To successfully implement continuous delivery, you need to change the culture of how an entire organization views software development efforts.
>
> Tommy Tynjä

https://dzone.com/articles/my-18-favorite-quotes-on-agile-devops-and-continuo

CI is not a practice that can be handed off to a project's "build master" and forgotten about. It **affects every person** on the software development team, so we discuss CI in terms of what all team members must practice to implement it.
Source: Continuous Integration: Improving Software Quality and Reducing Risk - Paul M. Duvall - 2007

Continues Noise
I'm a big fan of continuous integration. I've used it and recommended it for many years with great success. But it has a dark side as well. The larger and/or more complex the project, the higher the chance that it devolves into what I call "Continuous Noise." In this case, you get **notified every 10 minutes** or so (depending on how much building and testing is going on) that the build and/or test suite is still failing. It may be for a different reason every time, but it doesn't matter. It is tough to make progress when the mainline is unstable most of the time. **This problem is not caused by CI, but it is exposed by CI.** I have found that everything your organization needs to do in order to produce the best possible development organization can be entirely derived from the **patterns and practices at the individual level**.
https://www.drdobbs.com/tools/multi-stage-continuous-integration/212201506

CI / CD

THANK YOU

Link zur Präsentation

https://github.com/kessenich/webdev-2020

Die Beispiel App gibt es hier:

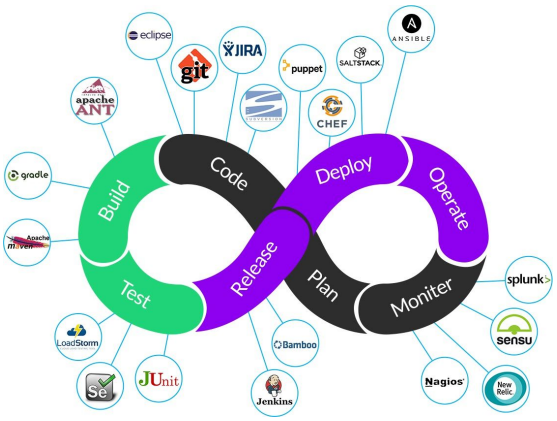
https://github.com/kessenich/webdev-2020-app

CI / CD

Back Up

DevOps

DevOps Cycle



https://evontech.com/component/easyblog/why-is-devops-the-future-of-software-production-cycles.html?Itemid=159