MONOREPO

Ein Repo – viele Teams

Für die Erstellung dieser Präsentation wurden keine KI-Modelle gequält* oder KI-Slop generiert.

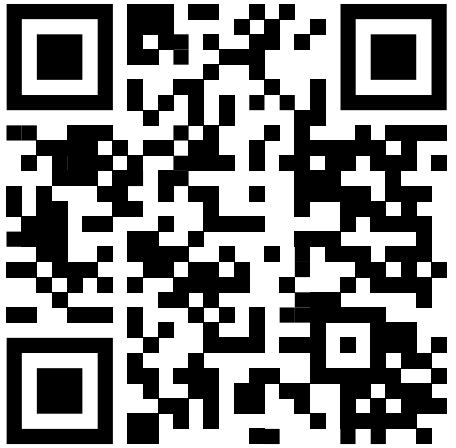
* Recherche ausgenommen.

ABOUT ME

- Software Engineer
  - Seit 11 Jahren selbständig
  - DevCon5 GmbH → Mitglied der Coalist Genossenschaft

  - Java
  - SW-Architektur & Organisation

- Brugg/Windisch (Aargau)



https://www.linkedin.com/in/gerald-muecke/

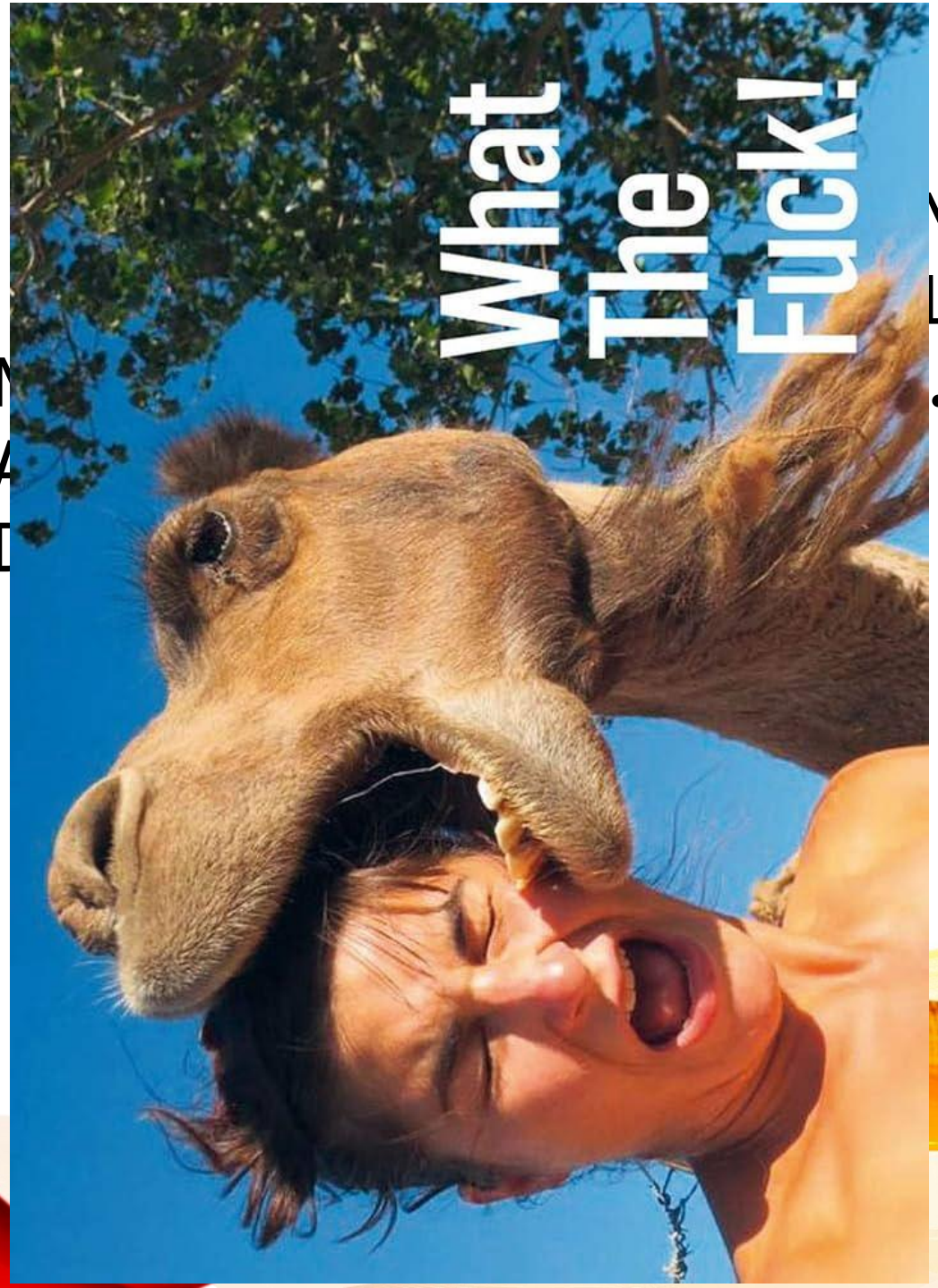
„DAS HATTEN WIR NIE, DAS BRAUCHEN WIR NICHT“

Architekt

(Hat das Unternehmen nach 6 Monaten verlassen)


TRIGGER
WARNING
EXPLICIT CONTENT

“ >GENERIERE ND STETIG
WACHSENDER LOGIESEKTOR
EXPLODIEREN .
MÖGLICHST A GRAUEM
PUDDING UND ”



KOMPLEXITÄT

Im Entwicklungsprozess

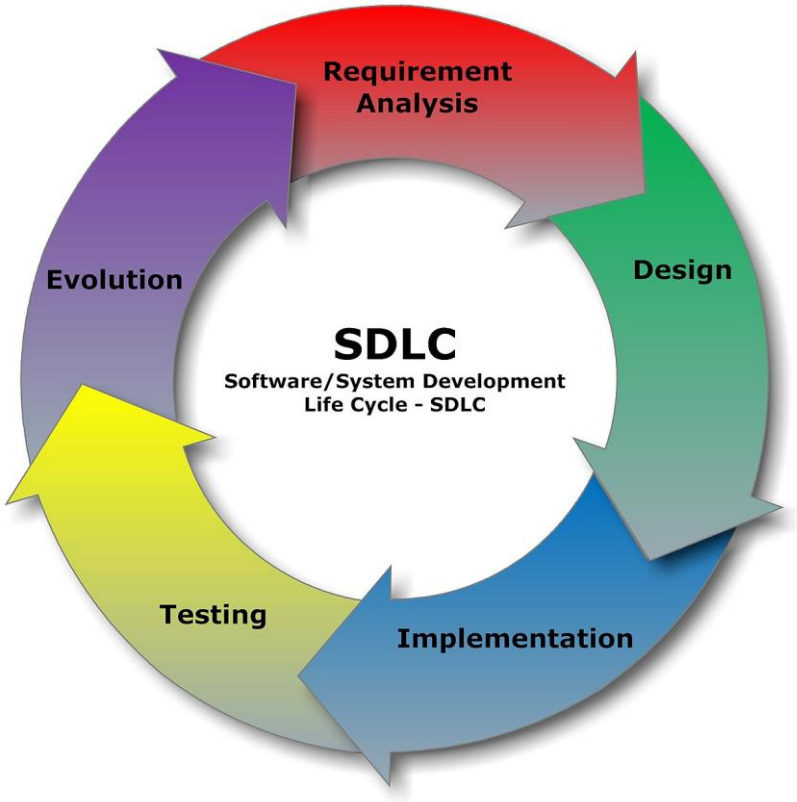


“EVERY APPLICATION HAS AN **INHERENT AMOUNT OF COMPLEXITY** THAT **CANNOT** BE **REMOVED** OR HIDDEN. INSTEAD, IT MUST BE DEALT WITH, EITHER IN PRODUCT DEVELOPMENT OR IN USER INTERACTION”

Tesler‘s Law - Law of Conservation of Complexity



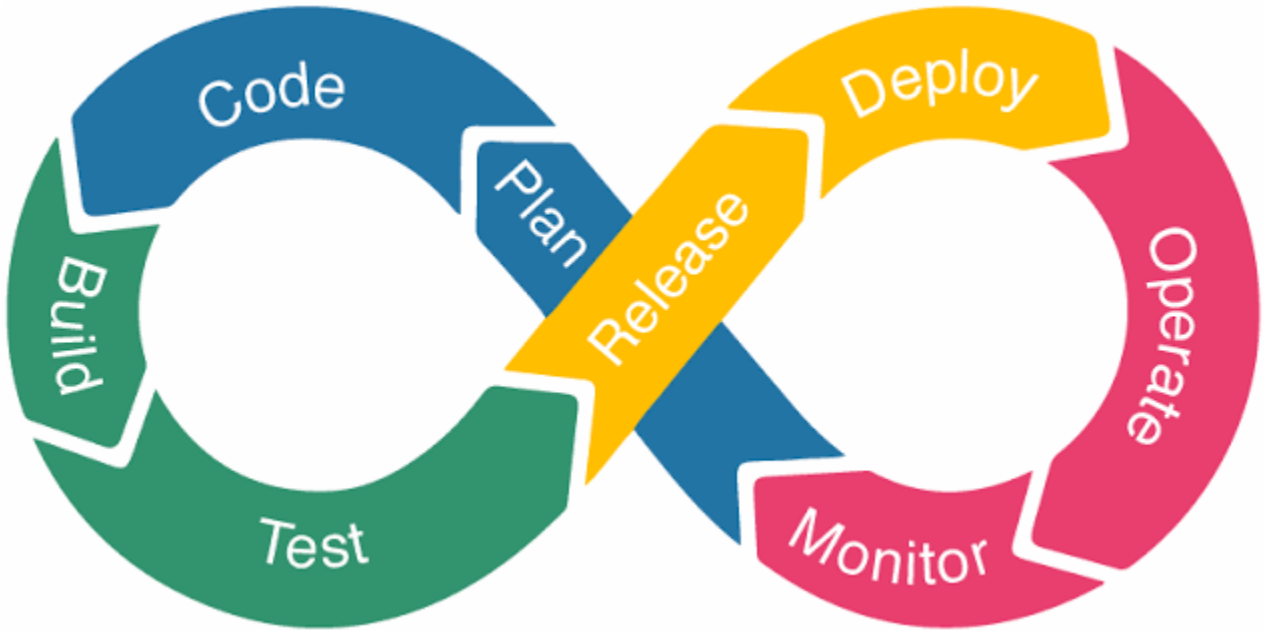
SOFTWARE DEVELOPMENT LIFE CYCLE SDLC





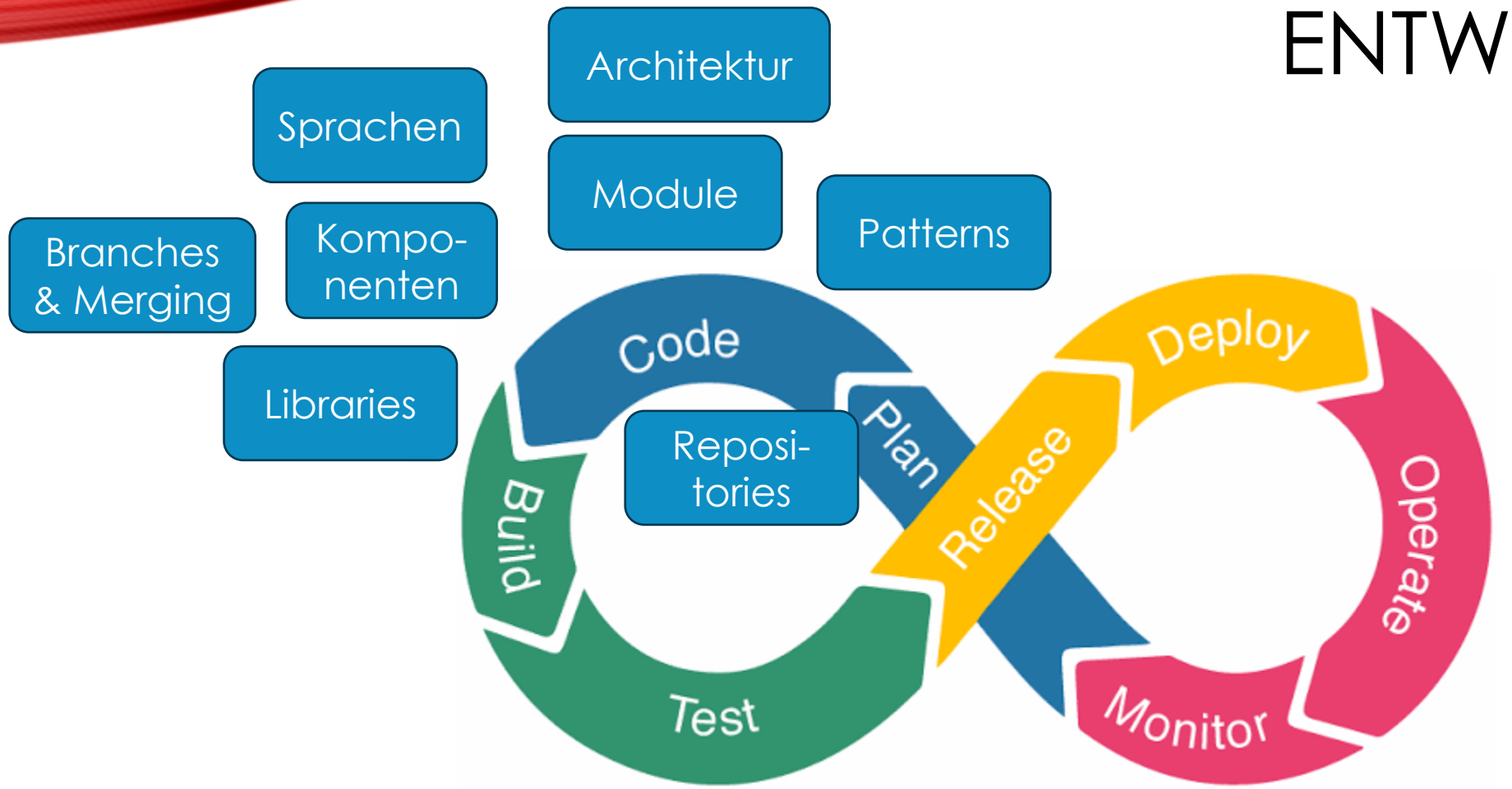
https://de.wikipedia.org/wiki/Software-Lebenszyklus

DEVOPS LIFECYCLE

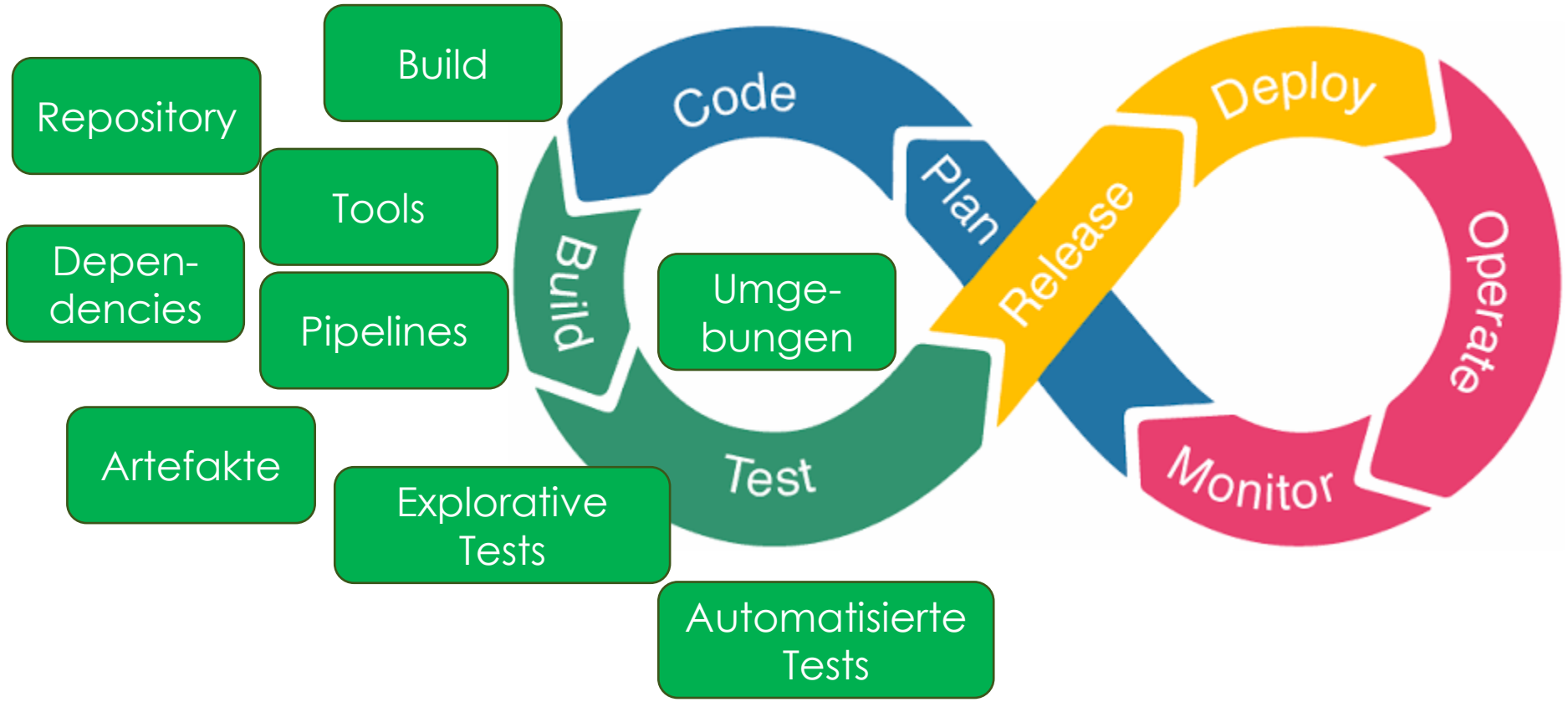


https://medium.com/t%C3%BCrk-telekom-bulut-teknolojileri/devops-lifecycle-continuous-integration-and-development-e7851a9c059d

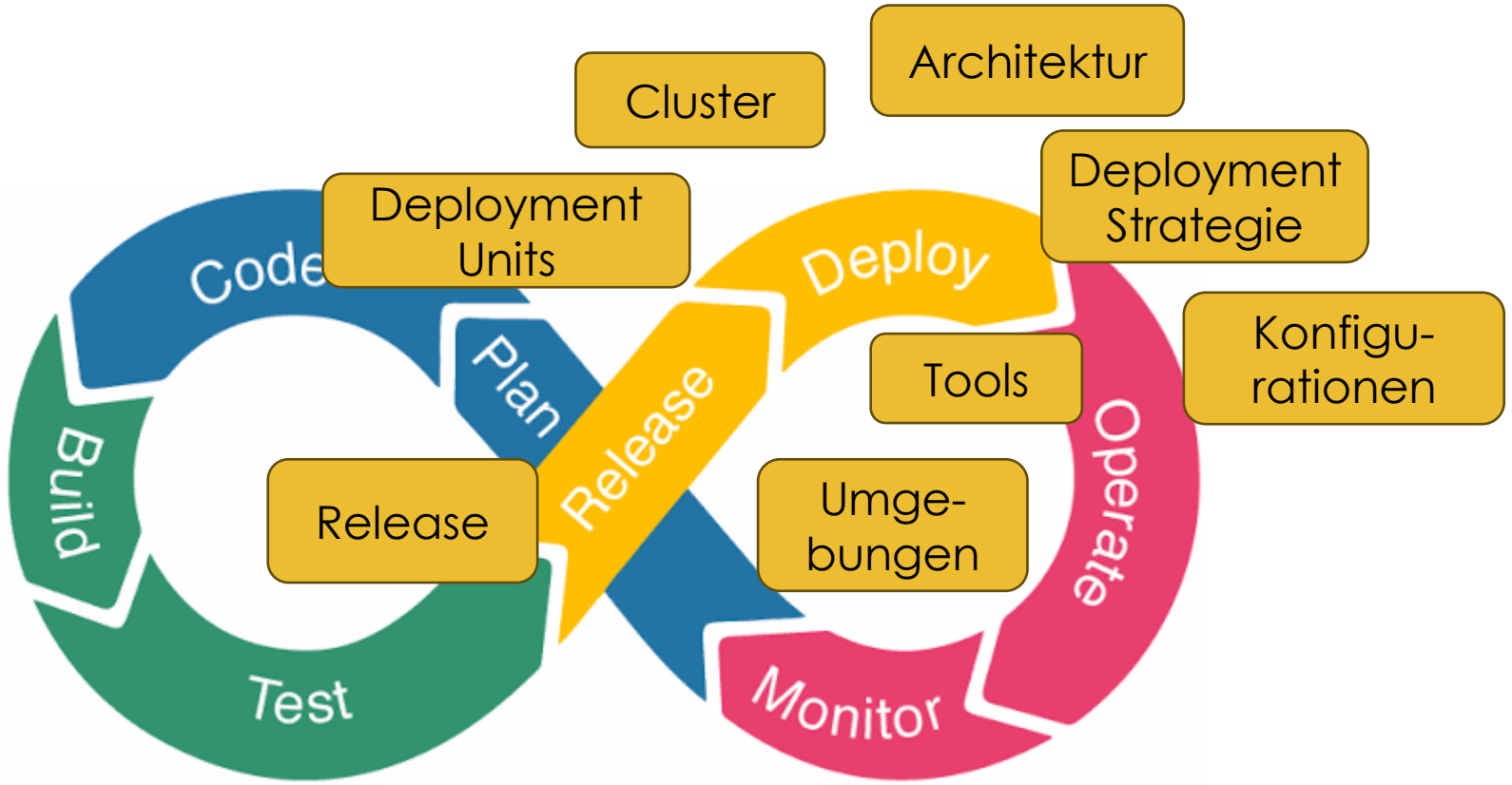
ENTWICKLUNG



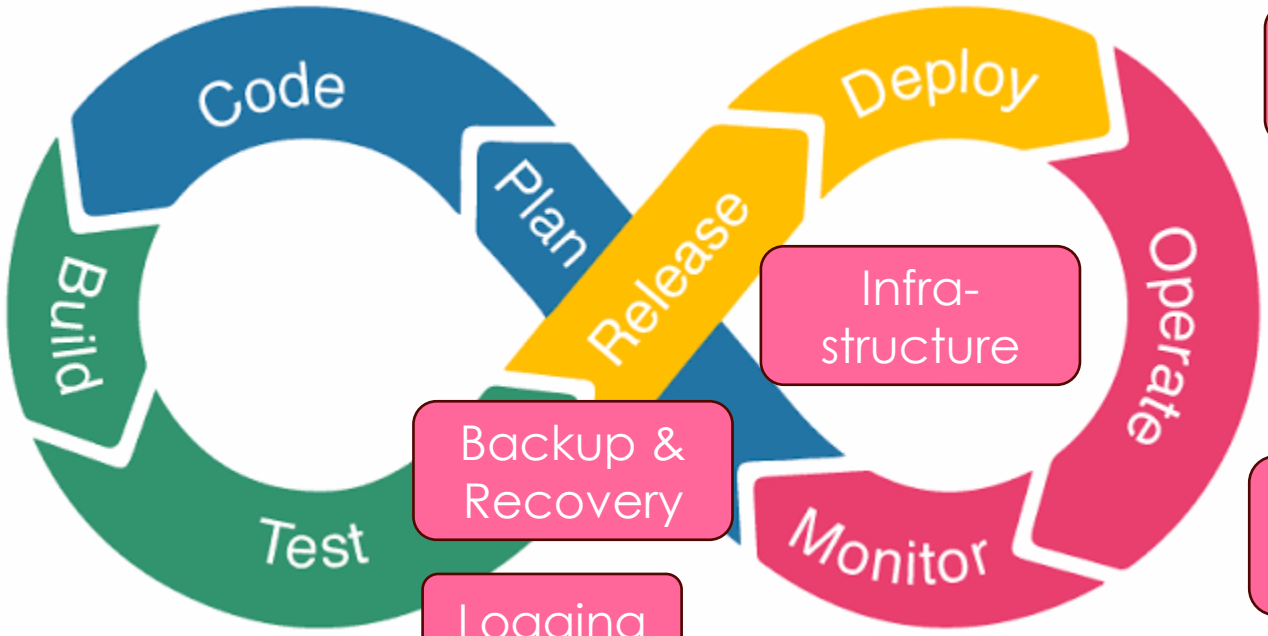
BUILD


Build
Repository
Tools
Depen-
dencies
Pipelines
Artefakte
Explorative
Tests
Automatisierte
Tests
Umge-
bungen
Code
Plan
Build
Test
Release
Deploy
Operate
Monitor

RELEASE & DEPLYOMENT



OPERATIONS



Availability

Bare Metal

Virtualisierung

Container-isierung

Infra-structure

Backup & Recovery

Fleet Management

Logging

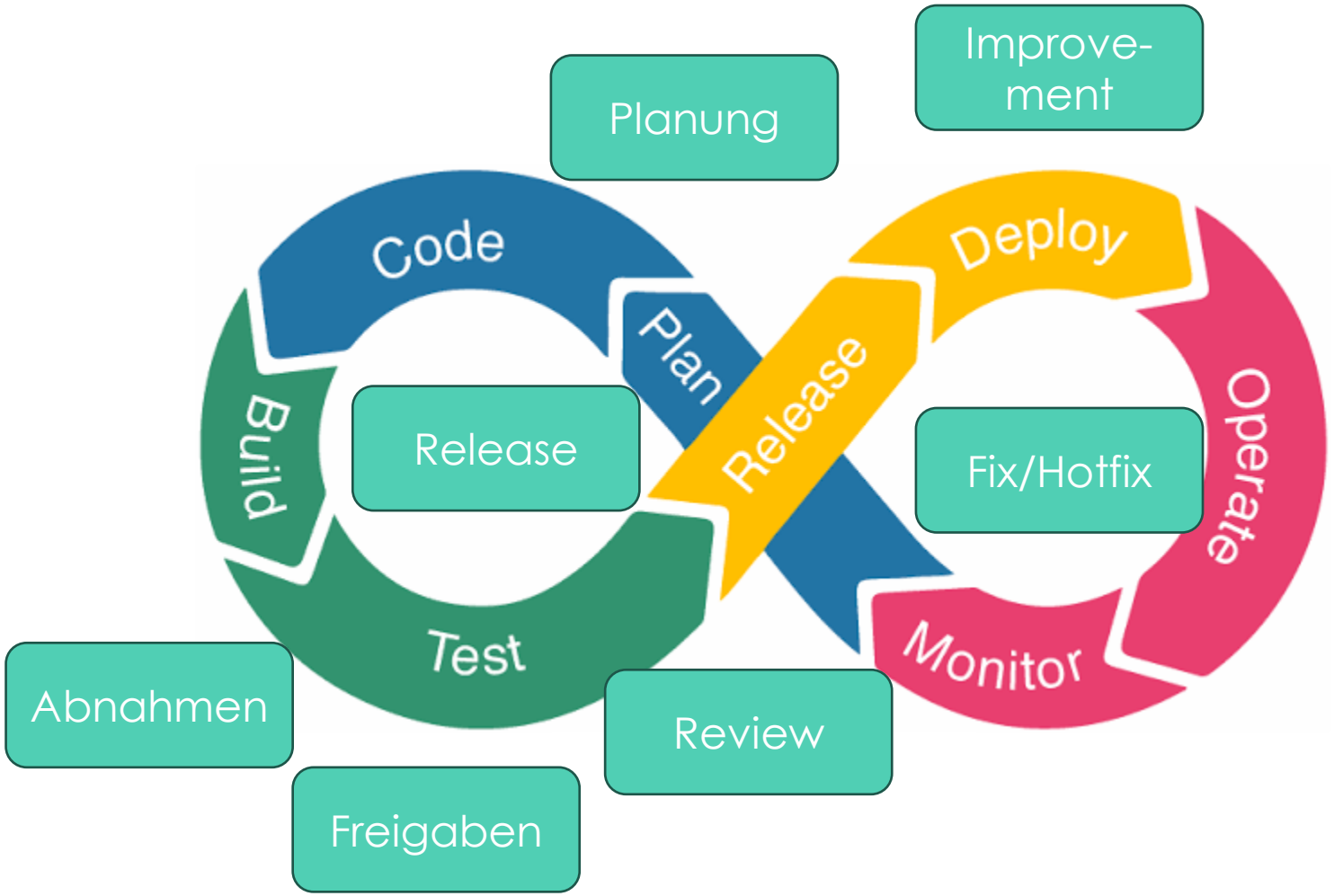
Monitoring

Security

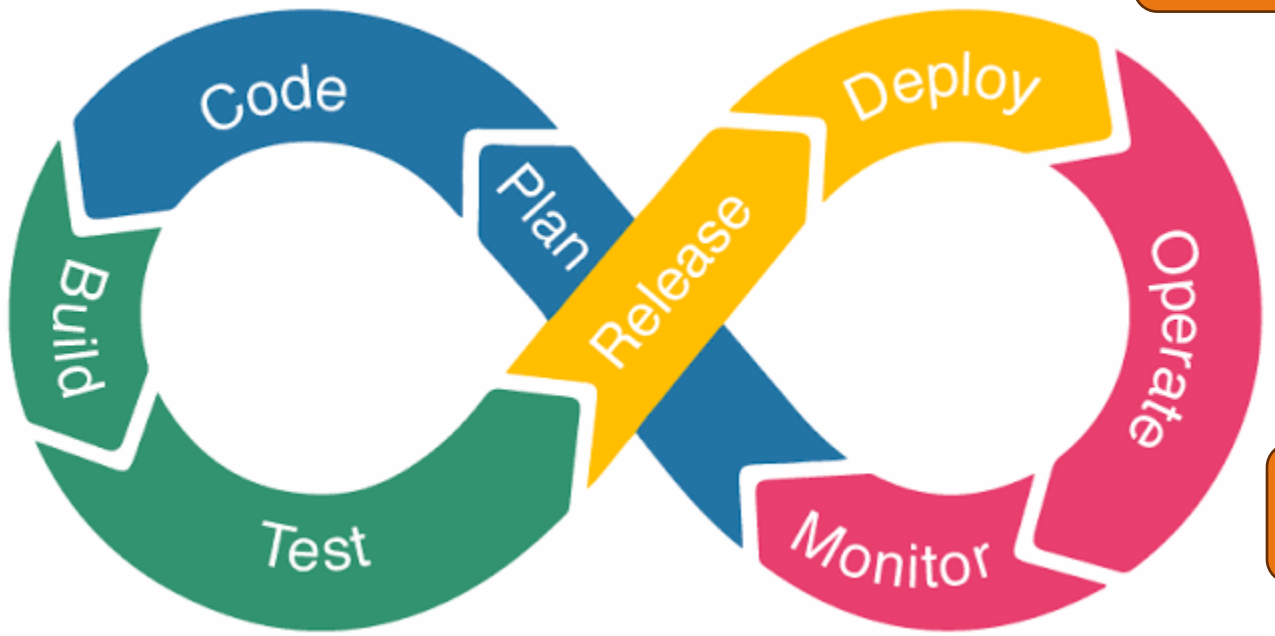
Tracing

Metrics

PROZESSE


Improve-
ment
Planung
Code
Deploy
Plan
Release
Release
Build
Operate
Fix/Hotfix
Test
Monitor
Abnahmen
Review
Freigaben

FAKTOR MENSCH



Meetings

Koordi-nation

Kommuni-kation

Change Management

Zuständig-keiten

Verantwort lichkeiten

Fluktuation

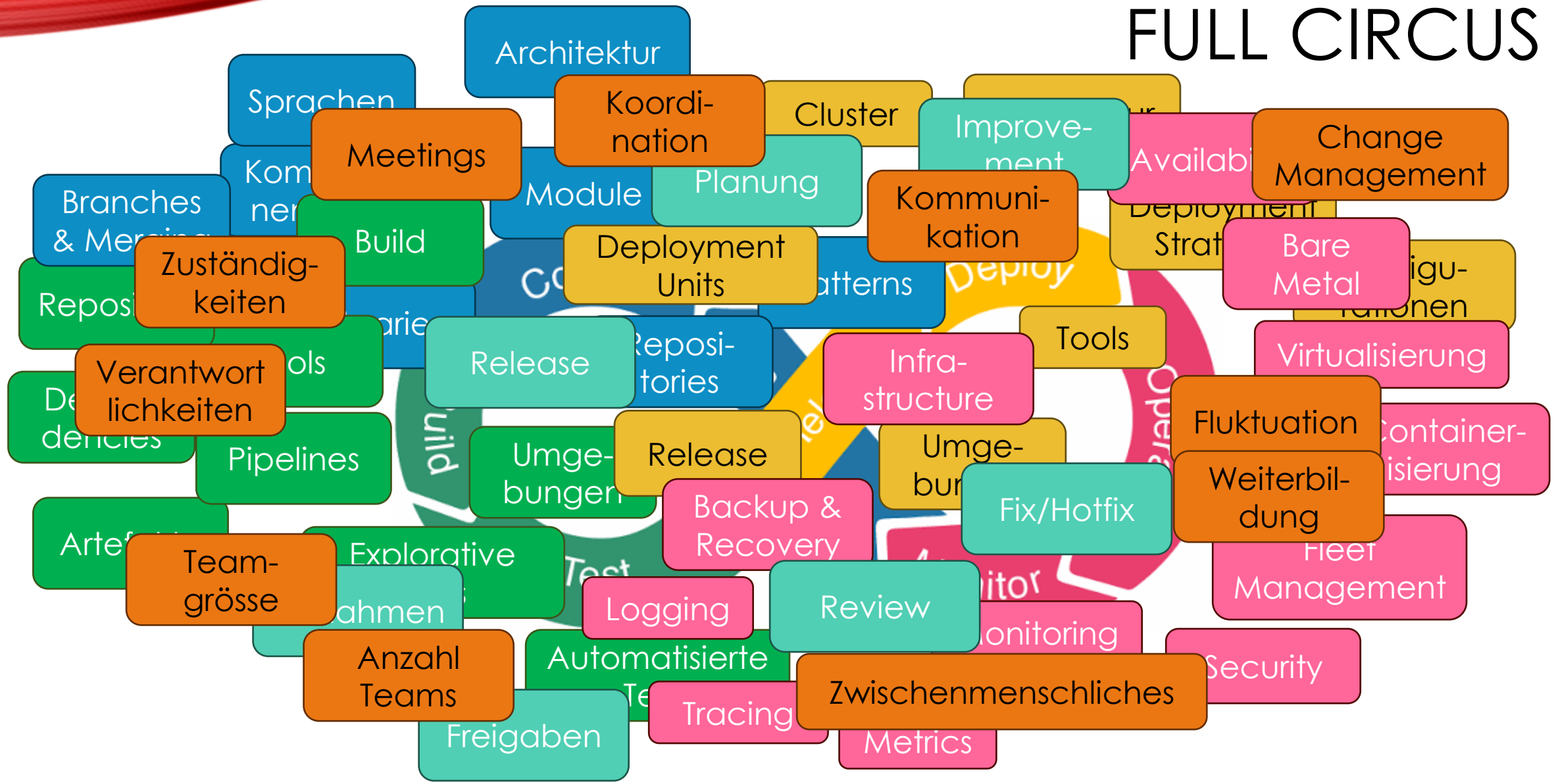
Team-grösse

Weiterbil-dung

Anzahl Teams

Zwischenmenschliches

FULL CIRCUS



Architektur
Sprachen
Koordi-nation
Cluster
Improve-ment
Change Management
Meetings
Module
Planung
Kommuni-kation
Branches & Mer
Build
Deployment Units
Bare Metal
Zuständig-keiten
Release
Reposi-tories
Infra-structure
Tools
Virtualisierung
Verantwort lichkeiten
Fluktuation
Pipelines
Umge-bungen
Release
Umge-bun
Weiterbil-dung
Fix/Hotfix
Backup & Recovery
Team-grösse
Explorative
Logging
Review
Fleet Management
Anzahl Teams
Automatisierte
Security
Zwischenmenschliches
Tracing
Freigaben
Metrics

AI

„mach mal alles einfach“

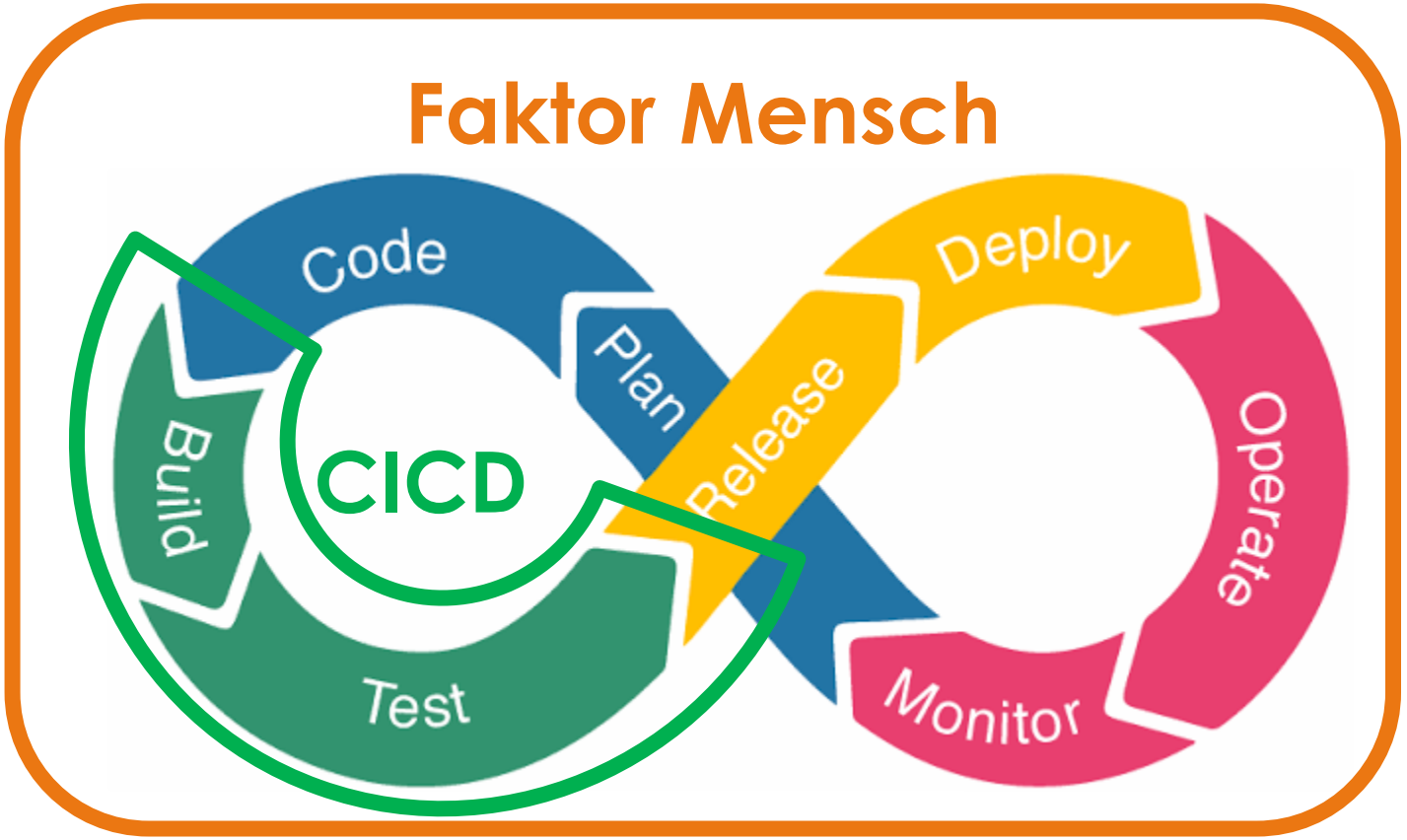


“WIE KANN ICH MIT DER KOMPLEXITÄT AM EINFACHSTEN UMGEHEN?”

… und was hat das mit Monorepos zu tun?



MONOREPO IMPACT



BEGRIFFSKLÄRUNGEN

„Ich mag keine Definitionen. Denn Definitionen schränken ein.“

Miles Davis, Jazz Musiker

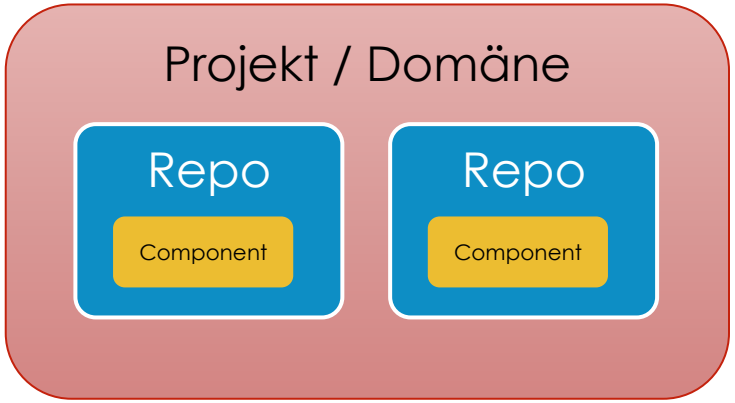


MULTIREPO / POLYREPO

- «A multi-repo is a software development approach where different projects or components of a larger application are stored and managed as separate repositories.»

https://www.thoughtworks.com/insights/blog/agile-engineering-practices/monorepo-vs-multirepo

- Synonym: „Poly-Repo“, „Multi-Repo“, „Many-Repo“
- Beispiel
  - Komponente
  - Microservice (bzw. Deployment Unit)
  - Dokumentation
  - Konfiguration
  - Tools

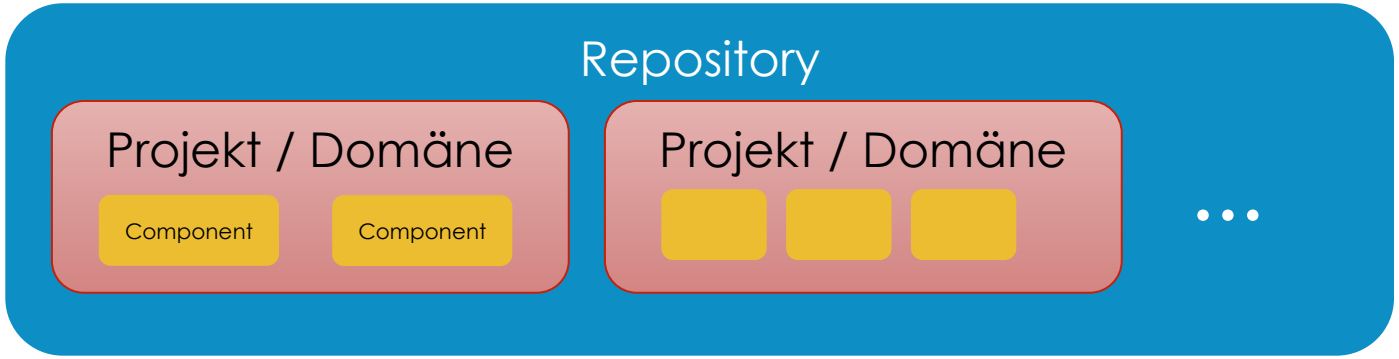


MONOREPO

- „... is a software-development strategy in which the code for a number of projects is stored in the same repository.»

  https://en.wikipedia.org/wiki/Monorepo

- Ist *kein* Monolith
- Ist *nicht* auf «eins pro Unternehmen» beschränkt
- Ist *nicht* mehr als eine Sammlung von Dateien und Verzeichnissen





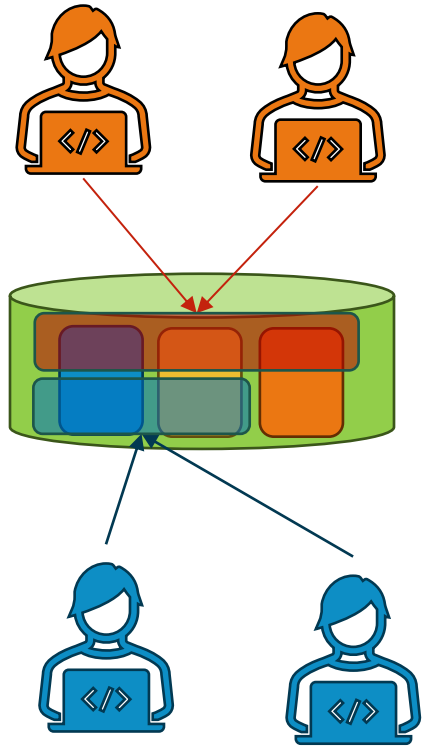
VORTEILE

„To hell with the rules. If it sounds right, then it is.”
Eddie Van Halen

https://commons.wikimedia.org/wiki/File:Edward_Van_Halen_live_2004-08-11.jpg

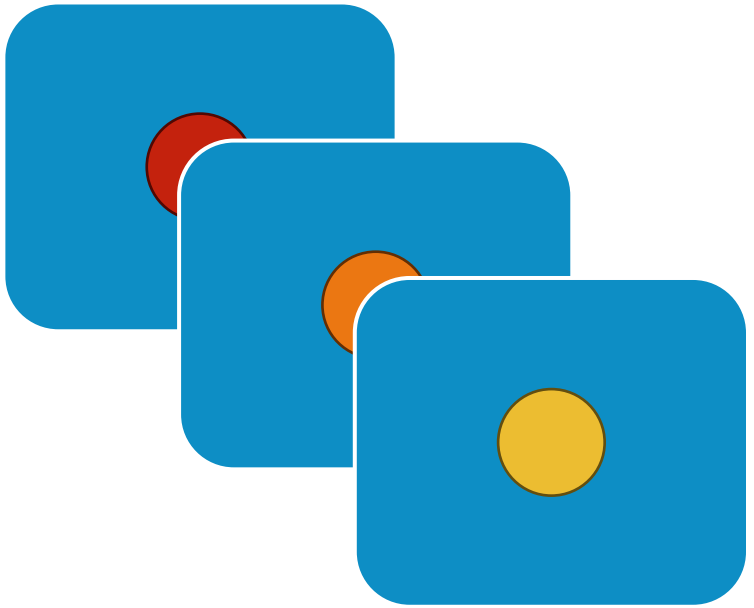
CROSS-TEAM COLABORATION

- Geteilte Verantwortung
  - Jeder ist verantwortlich die Code Basis sauber zu halten
  - Jeder *kann* sie sauber halten
- Erleichtert Team Veränderungen
  - Codebasis ist bereits vertraut
  - Workspace bleibt gleich
  - Keine zusätzlichen Berechtigungen erforderlich
  - Code anderer Teams ist jederzeit zugänglich
- Task Ownership vor Component Ownership
  - Teams verantworten Änderungen End2End
  - Komponentengrenzen sind nicht unüberwindbar



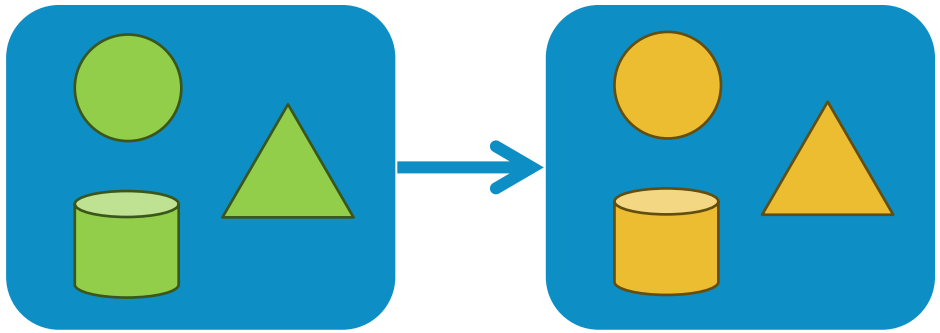
SINGLE SOURCE OF TRUTH

- Gemeinsame Domänendefinition
  - Geschäftslogik
  - Validierungslogik
  - Datenmodelle
  - APIs / Contracts
  - Testdaten
- Durchgehende Entwicklungsstandards
  - Architekturregeln
  - Coding Style & Konventionen



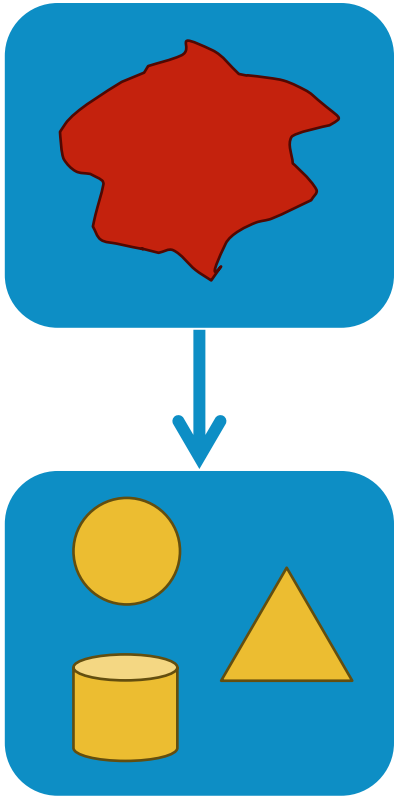
ATOMIC COMMITS

- Commit überführen das Gesamtsystem von einem konsistenten Zustand zum nächsten:
  - Domänenlogik
  - Dokumentation (Architektur-, Requirements-, Developer-, Benutzer-)
  - CICD Prozesse
  - End2End Tests
- Commits können auch Team übergreifend sein
  - Collaborativer Branch + Squash



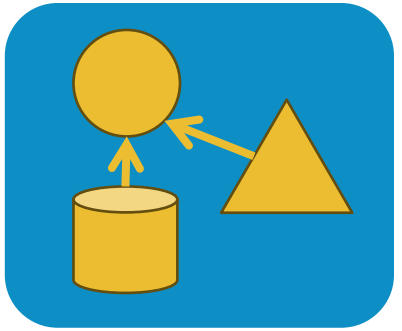
LARGE SCALE REFACTORING

- Refactorings über das Gesamtsystem in einem Schritt
  - Domain spezifische Änderungen
  - Architektur-Refactorings (Org/Team Refactorings)
- Volle Transparenz bei Impact Analyse
  - Wer ruft wen auf? Wer benutzt was?
- Regelbasierte Migrationen des Gesamtsystems
  - Structural Search & Replace oder OpenRewrite
  - Z.B. Java, Spring oder Angular Migrationen
- Höhere Architektur Konsistenz
  - Gesamtsystem ist immer auf dem gleichen Level
  - Kann als Ganzes getestet werden



CODE REUSE

- Discoverability & Honest Reuse
  - Alle Möglichkeiten sind jederzeit sichtbar („Gibt es das bereits?“)
  - Beispiele an anderen Code-Stellen
  - Code ist leichter zum Re-Use herauszulösen
  - Dead Code wird leichter sichtbar
- Shared Library oder Cross-Linking
  - Build Order & Dependencies (z.B. Maven, NPM)
  - Direct-Import (Javascript)



DEVELOPER EXPERIENCE

- Schnelleres Onboarding
  - Setup ist einfacher – Single Checkout
  - Alles an einem Ort
- Weniger Kontext-Wechsel
  - alles in einem Workspace
  - alles in einer History
- Niedrige Hürden für Experimente
  - Alles notwendige liegt direkt zur Hand
- Ein Tooling für alles
  - Helper, Scripts
  - CodeStyle & Rules
  - Local Setup
  - Dokumentation
  - Infrastruktur





NACHTEILE

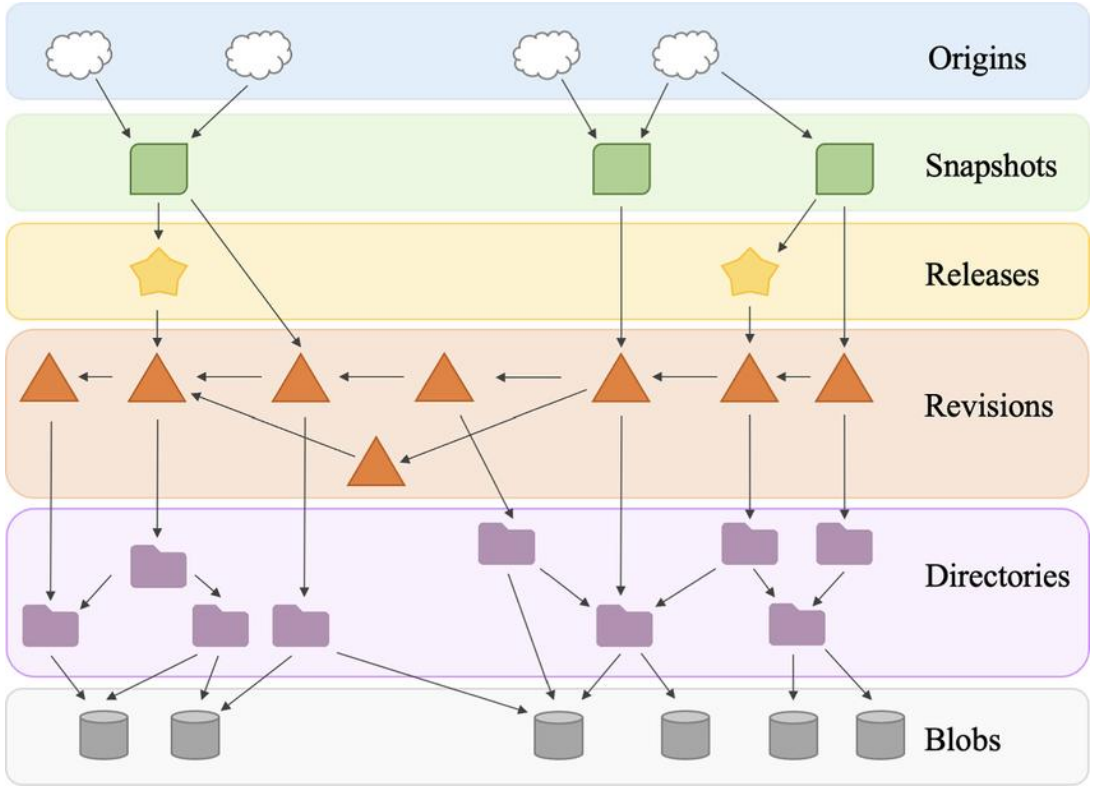
„I'd rather be hated for who I am, than loved for who I am not. “

Kurt Cobain (Nirvana)

https://commons.wikimedia.org/wiki/File:Nirvana_around_1992.jpg

GIT PERFORMANCE

- Git Performance issues at scale
- Wurde für Individuelle Projekte und nicht Millionen von Files & Commits gemacht
- Algorithmischen Kosten auf dem DAG:
  - Git log
  - Git blame
  - Git merge-base
  - Git rebase
- Git Status Tracking wird teurer



https://www.researchgate.net/figure/The-SWH-data-model-stored-as-a-Merkle-DAG_fig1_379642241

BUILD PERFORMANCE

- Drei Skalierungsdimensionen
  - Mehr User → Mehr Changes
  - Longer Builds → Queueing for Executors
  - More Data → Network Bandwith Limitations

- Folge: Längere Feedback Loops



ACCESS CONTROL

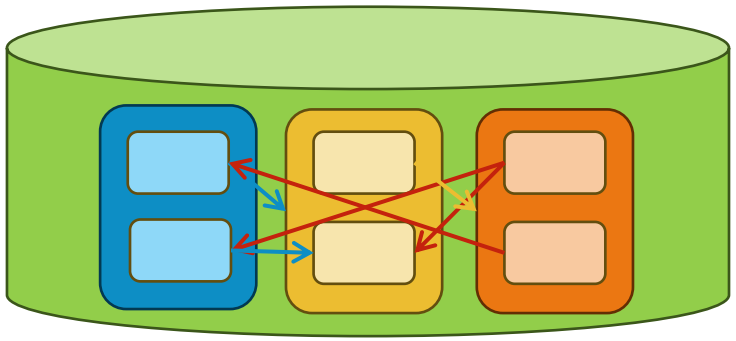
- Jeder hat immer vollen Zugriff auf das gesamte Repository

- Folge: Nicht geeignet für «Need-to-Know», «Confidential», «Firmengeheimnis»
  … oder eingecheckte Credentials



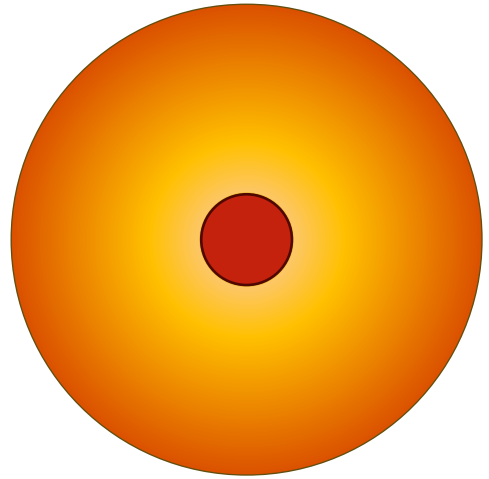
OWNERSHIP BOUNDARIES

- Grenzen zwischen Komponenten und Zuständigkeiten sind weniger strikt
- Jeder kann grundsätzlich überall Code ändern
- Forcierung von Code-Ownership erfordert mehr Disziplin und Zusammenarbeit



DEPENDENCY MANAGEMENT & BLAST RADIUS

- «one version»
  - → everybodes moves at once or none at all
  - → einzelnes «broken» package kann kompletten Upgrade verzögern
- Grösserer Blast Radius zentraler Komponenten
  - Ein Build Break lähmt die gesamte Organisation
  - Ein Bug kann die ganze Produktion lahmlegen
  - →Je zentraler eine Komponte, desto wichtiger werden Code Qualität, Tests und Review Prozesse





ORGANISATION

„Die Krankenversicherung, die mich nehmen würde, muss noch erfunden werden.“
Lemmy Kilmister (Motörhead)

https://commons.wikimedia.org/wiki/File:Lemmy-02.jpg

ORGANISATORISCHE ENTSCHEIDUNG

- Die Entscheidung zu einem Monorepo ist eine *fundamentale organisatorische Entscheidung* – keine technische
- Sie beeinflusst
  - Wie Teams miteinander Arbeiten
    - → Enge Zusammenarbeit
  - Wie Entscheidungen getroffen werden
    - → Konsensfähigkeit
  - Wie die Organisation (und die Architektur) sich entwickeln
    - → Offenheit für Veränderungen

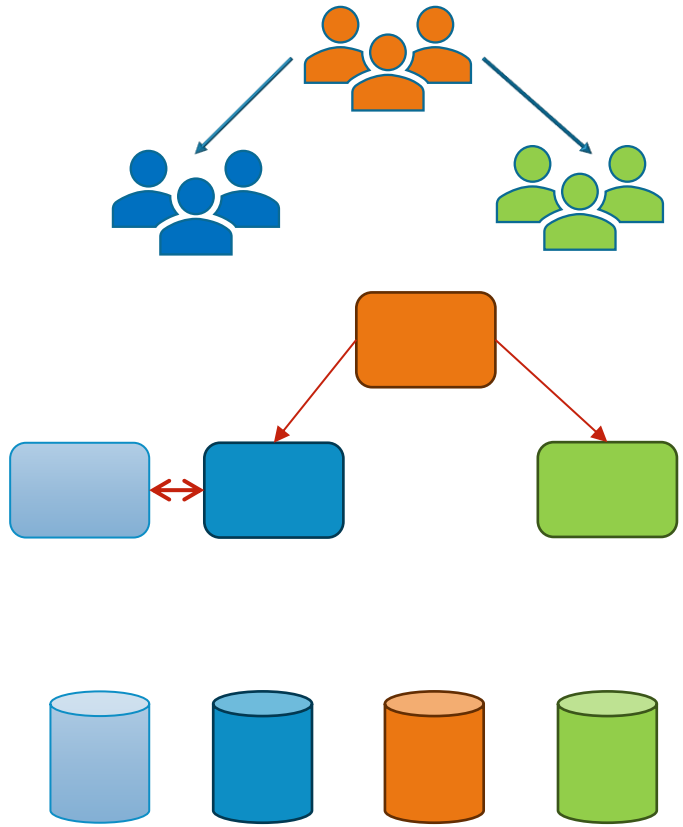
CONWAYS LAW

"Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure.«

Architektur spiegelt Kommunikationsstruktur der Organisation

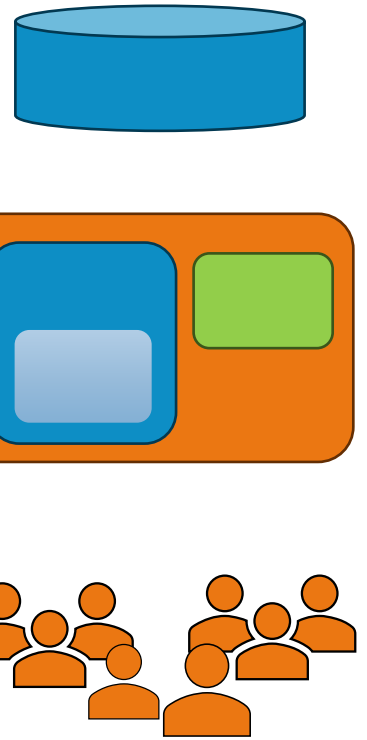
POLYREPO & CONWAYS LAW

- Typische Reihenfolge
  - 1. Organisation (Teams, etc)
  - 2. Architektur (Komponenten, etc.)
  - 3. Repositories (pro Komponente)
- Folge
  - Klare Ownership
  - Unabhängige Systeme
  - Änderungen an Repositories bedeuten
    - → Anpassungen an der Architektur
    - → Anpasssungen an der Organisation
    - → sehr hohe Barriere
- Struktur folgt der Organisation (Conways Law)



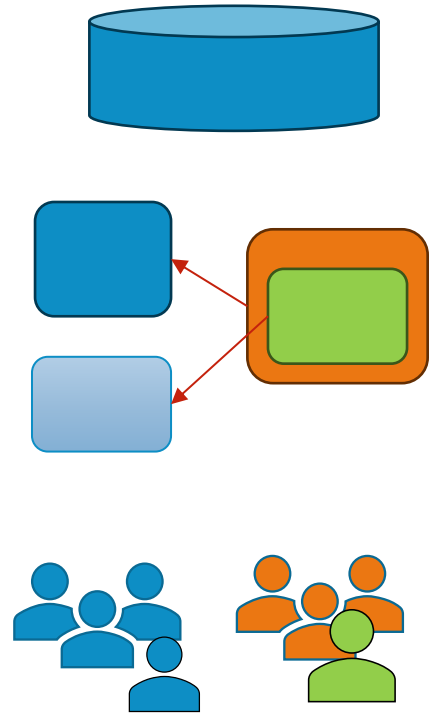
MONOREPO & CONWAYS LAW

- Typische Reihenfolge
  - 1. Repository anlegen
  - 2. Architektur bestimmen
  - 3. Organisation an Architektur ausrichten
- Folge
  - Shared Ownership
  - Koordinierte Systeme
  - Veränderungen von Architektur und Organisation
    - Nur „ein Commit“
    - Können sich fortlaufend weiterentwickeln (fliessend)
    - Leichter revidierbar.



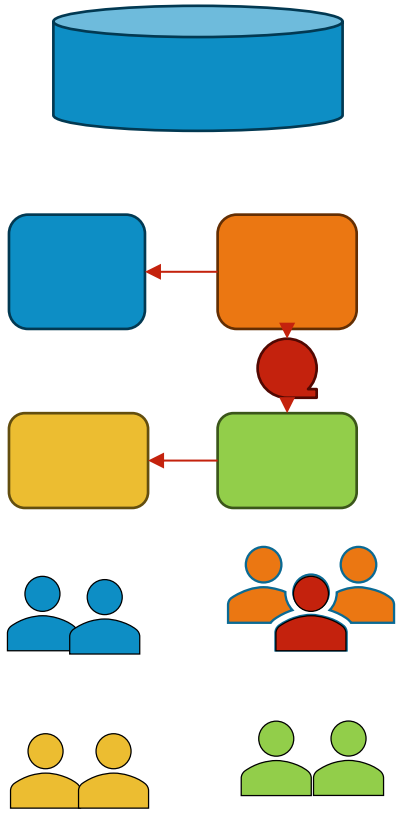
MONOREPO & CONWAYS LAW

- Typische Reihenfolge
  - 1. Repository anlegen
  - 2. Architektur bestimmen
  - 3. Organisation an Architektur ausrichten
- Folge
  - Shared Ownership
  - Koordinierte Systeme
  - Veränderungen von Architektur und Organisation
    - Nur „ein Commit“
    - Können sich fortlaufend weiterentwickeln (fliessend)
    - Leichter revidierbar.



MONOREPO & CONWAYS LAW

- Typische Reihenfolge
  - 1. Repository anlegen
  - 2. Architektur bestimmen
  - 3. Organisation an Architektur ausrichten
- Folge
  - Shared Ownership
  - Koordinierte Systeme
  - Veränderungen von Architektur und Organisation
    - Nur „ein Commit“
    - Können sich fortlaufend weiterentwickeln (fliessend)
    - Leichter revidierbar.
- Organisation folgt der Struktur (Reverse Conways Law)



GOOD PRACTICES

- Klare Verantwortlichkeiten & Zuständigkeiten definieren
  - CODEOWNERs* & Automatisches PR-Routing
  - Branch Protection
- Gemeinsame Regeln, Standards & Policies definieren & durschsetzen
  - Package Boundaries setzen und enforcen (z.B. ArchUnit)
  - Review Policies für Shared Libraries
- Kontinuierliches Investment in Build & CI Infrastruktur
  - → Platform Team

* Nicht unterstützt von Bitbucket Datacenter

PLATFORM TEAM

- Build Infrastruktur
  - CICD Pipeline Management
  - Build Tools
  - Caching
  - Workers
- Performance Monitoring
  - Build Time Tracking
  - Clone Size Monitoring
  - CI Queue Size Metrics
  - Developer Feedback Loops

- Developer Experience
  - Onboarding Process
  - Developer Tooling
  - Documentation & Knowledge Sharing
  - Training & Enablement
- Goveranance & Policy
  - Code Ownership
  - Branch Protection
  - Security Scanning
  - Compliance Enforcement



AI

„Out of everything I've lost, I miss my mind the most!"

Ozzy Osbourne

https://de.wikipedia.org/wiki/Ozzy_Osbourne#/media/Datei:Blizzcon_2009_11.jpg

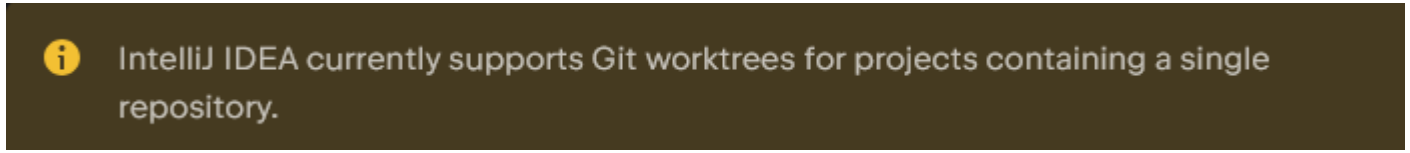
WENIGER CONTEXTUAL BLOAT

Monorepo

- Single History, Commits, PRs
- Keine Repository Wechsel
- Worktrees werden für Multi-Agents workflows essentiell
- Und nur für Monorepos gut unterstütz

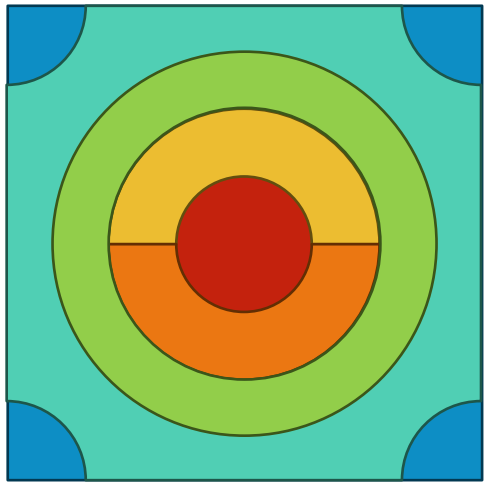
Polyrepo

- Multiple Git Histories, Commits, PRs
- Wechsel zwischen Repositories erzeugt zusätzliche Kosten
- Cross-Repo Features werden mit Worktrees zum „Workforest“



KONTEXT-REPOSITORY

- Code
- Produkt-Intention
- Anforderungen und deren Ratio
- Architektur-Entscheidungen und Begründungen
- Rahmenbedingungen
- Dokumentation



SHARED SETUP

- Skills
  - für Projekt
  - Codebasis
  - Standardisierte Workflows & Methoden
- Agents
  - Standard-Agents
  - projekt-spezifische Tasks
- MCP Server
  - vorkonfiguriert für das Projekt







END2END TASKS/GOALS

- Requirements Spec
- Architecture documentation
- Developer documentation
- User Documentation
- Design / UX
- Implementation
- Tests, einschliesslich end2end Tests
- Up to deployment (commit, push, verify, logs)



WEITERE MÖGLICHKEITEN

- Analysen
  - Systemweiter Datenfluss
  - Blast-Radius / Impact von Änderungen
- Erkennung
  - Patterns und Antipatterns
  - Inkonsistenzen
  - Impliziten und indirekten Abhängigkeiten
  - Spec & Config-Drift
- Automatisierung
  - Upgrades von Dependencies
  - Fehlerbehebung (Log Analyse, Fixes, Tests, PRs, Verify)
  - Dokumentationsupdates
- Vermeidung von „Shadow Tech Debt“



TOOLING

"Wozu ist das?" - "Das ist blaues Licht!" –
"Und was macht es?" - "Es leuchtet blau!„

Sylvester Stalone (Rambo III)

https://www.youtube.com/watch?v=R6kgPe2bSkA

BUILD TOOLING

- JVM
  - Maven
    - Build Cache Extension
    - --projects --also-make |--also-make-dependents
  - Gradle
- Polyglott
  - Gradle, NX, Bazel, Buck2, Pants
- Task Orchestration
  - Gradle, NX, Turborepo, Moon
- Workspace/Package Manager
  - Maven (.m2)
  - npm workspace
  - pnpm, yarn berry, rush

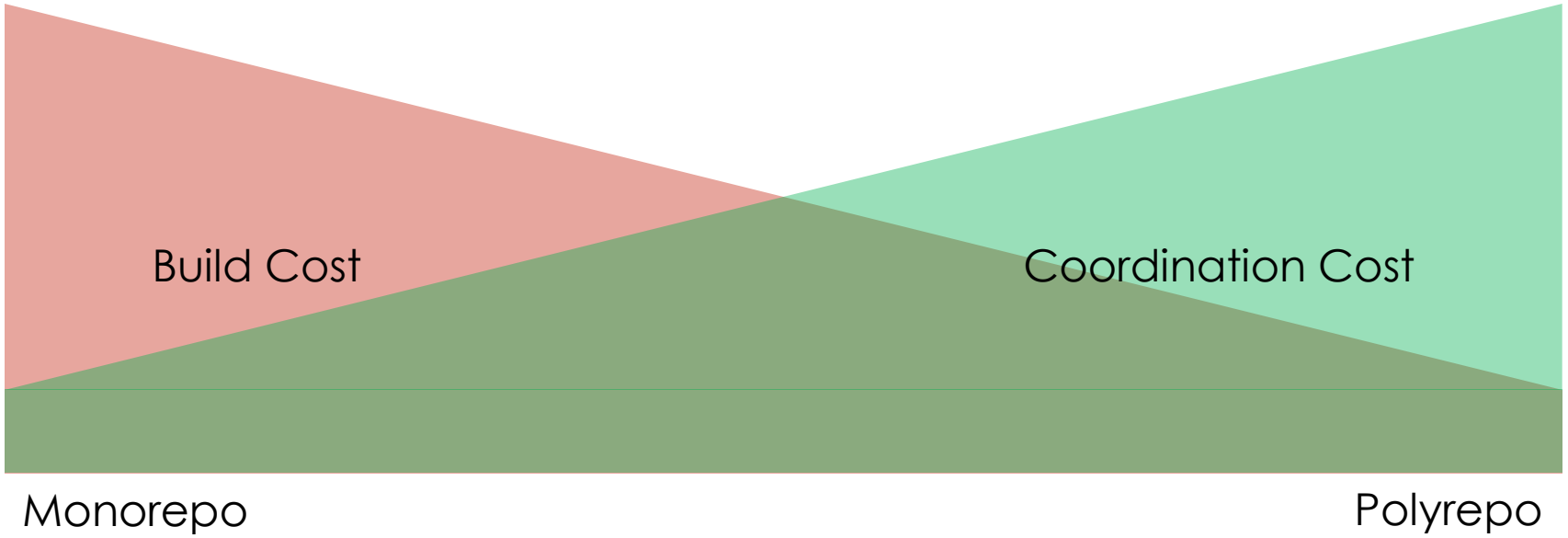
SCM TOOLING

- Git Default
  - Fetch = code + history + branches (+ tags)
  - Checkout – Kompletter Branch
  - One Worktree
- Speed up
  - Shallow Clone – keine/kürzere History
  - Sparse Checkout - nur bestimmte Folder
  - Multiple Worktrees - Parallelisierung
- Wenn das Repository wächst
  - Git Scalar – ab 2.38
  - Sapling (Meta) – Git compatible

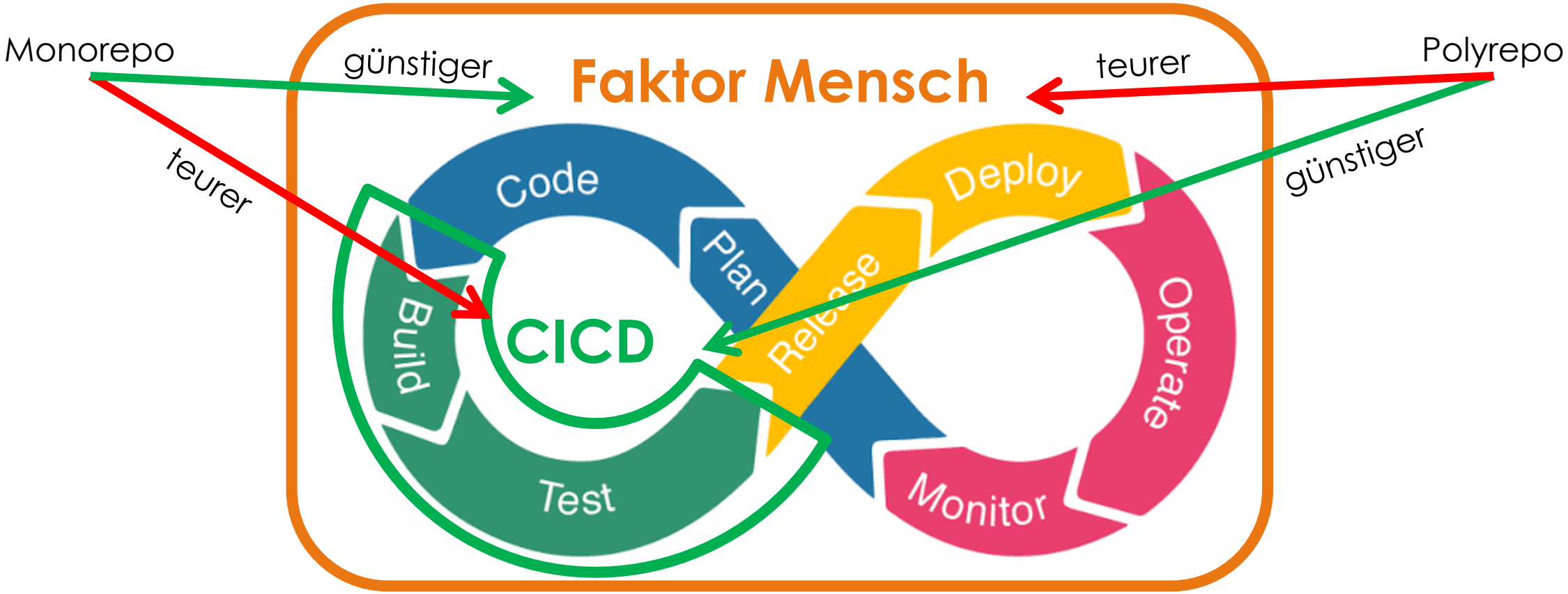
FAZIT



TRADEOFFS IN A NUTSHELL



MONOREPO IMPACT



ENTSCHEIDUNGSHILFE

Monorepo	Polyrepo
Interagierende Team	Autonome Teams
Project Interdependencies (Services & Shared Libraries)	Isolates Services No Sharing
Investment in Build & CI	Low Cost Build & CI
Knowledge Sharing	Security / Access Control / Need to Know
Agentic Coding	AI assisted Coding

VIELEN DANK!



LITERATUR

- https://monorepo-book.github.io/
- https://monorepo.tools/
- https://sourcegraph.com/blog/monorepo-vs-polyrepo
- https://cachely.dev/blog/why-monorepo
- https://www.datacamp.com/tutorial/monorepo
- https://www.thoughtworks.com/en-us/insights/blog/agile-engineering-practices/monorepo-vs-multirepo
- https://woliveiras.com/posts/advantages-and-disadvantages-of-using-monorepos
- https://circleci.com/blog/monorepo-dev-practices/
- https://mergify.com/learn/monorepo
- https://www.atlassian.com/git/tutorials/monorepos
- https://deviera.dev/blog/monorepo-vs-multi-repo-ci-coordination
- https://www.atlassian.com/git/tutorials/what-is-version-control
- https://dropbox.tech/infrastructure/reducing-our-monorepo-size-to-improve-developer-velocity
- https://kevingoslar.medium.com/monorepos-make-inner-source-come-to-life-bd1592b0cadf
- https://bubble.ro/2026/07/31/the-monorepo-trap-what-nobody-tells-you-before-you-migrate/
- https://www.codexical.com/posts/2026-05-14-monorepo-hidden-coordination-costs
- https://www.javacodegeeks.com/2026/07/the-misunderstood-monorepo-architectural-trade-offs-teams-discover-only-after-they-commit.html
- https://buildkite.com/resources/blog/scaling-ci-for-monorepos-challenges-and-how-to-overcome-them/
- https://github.blog/open-source/git/the-story-of-scalar/
- https://engineering.fb.com/2022/11/15/open-source/sapling-source-control-scalable/
- https://cacm.acm.org/research/why-google-stores-billions-of-lines-of-code-in-a-single-repository/