LU02a - Pipeline Phasen

In einer Build-Pipeline sind in der Regel verschiedene Phasen(Stages), die seriell oder parallel durchlaufen werden.

Stage	Beschreibung	Allfällige Umsysteme
Checkout	Kopieren des Codes in lokales Verzeichnis als Vorbereitung für die folgenden Phasen.	Code-Repository(Github, Gitlab, ...)
Abhängigkeiten installieren	`npm install`, `pip install -r requirements.txt`, `gradlew dependencies`	Artifactory(JFrog, ...)
Linting	Überprüfung, ob die vordefinierten Coderichtlinien eingehalten wurden. Je nach Konfiguration kann diese Stage fehlschlagen, ohne die Pipeline abzubrechen (z. B. als Warnung).	-
Build/Compile	Kompilieren/Transpilieren/Builden des Codes. Eine einfache Methode, um sicherzustellen, dass der Code zumindest installiert und gestartet werden kann	-
Unit Tests	Einfache Tests, um einzelne Teile der Logik zu kontrollieren.	-
Integration Tests	Tests, welche die Zusammenarbeit mehrerer Komponenten unter Einbezug externer Abhängigkeiten prüfen.	Datenbank, Dateisystem, Mock-Server, ...
E2E Tests	Tests, die den kompletten Stack testen, also von der Benutzeroberfläche bis zur Datenbank.	Test-System, Mock-Server, 3rd Parties(SauceLabs, ...), ...
Erweiterte Codeanalyse	Statische Codeanalyse alleine ist ähnlich wie Linting, jedoch gibt es Systeme wie SonarQube, welche noch viel mächtiger sind. Bei Auswertung der Test Coverage läuft diese Phase entsprechend erst nach allen Tests.	SonarQube
Package / Artifact bauen	Code-Ergebnis wird als Paket bereitgestellt, welches online verfügbar ist und weiterverwendet werden kann.	Artifactory(JFrog, ...)
Deploy auf Dev/Test	Code wird auf einem Entwicklungs- oder Testsystem installiert zur weiteren Validierung (manuell oder automatisch).	Dev/Testsystem
Deploy auf Prod	Code wird auf dem produktiven System ausgerollt und für die Benutzer zur Verfügung gestellt.	Prodsystem