Continuous Deployment

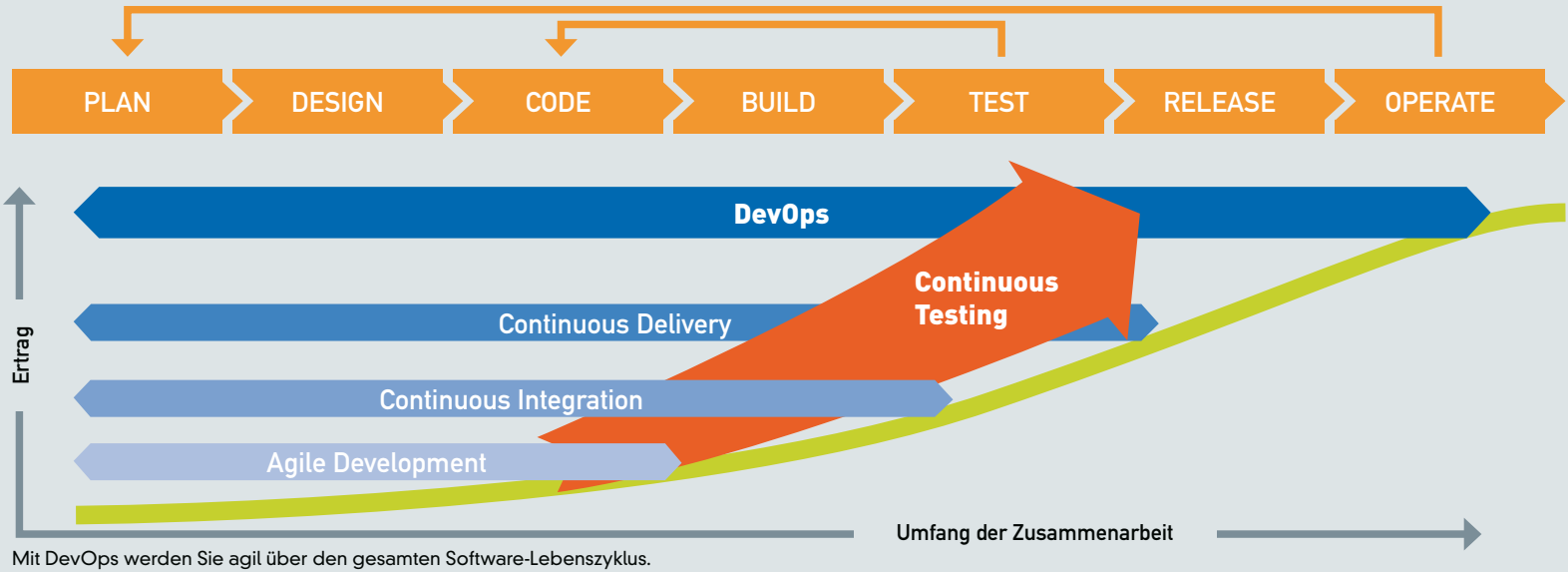
Generell führt Continuous Deployment zu vielen stabilen Releases:

1. Automatisierte Installation in Test- und Produktionsumgebungen durch Infrastructure-as-Code mit domänenspezifischen Sprachen zur Konfiguration und Installation, inkl. Versionierung dieses Infrastruktur-Codes → z. B. mit Puppet, RedHat Ansible und OpenMake Release Engineer
2. Entwicklungsmuster wie Feature-Toggles helfen dabei, Code früh an ausgewählte Nutzer zu liefern, auch wenn dieser noch nicht zur Verwendung durch alle gedacht ist. → z. B. mit Togglz
3. Fehler, die in der Deployment-Pipeline auftreten, dürfen nicht toleriert werden, führen zum Abbruch der Installation und müssen berichtet werden. → z. B. mit LambadaCD

Entwicklungs-Umgebung	Continuous Integration		Betriebsnahe Testumgebung	Life-System
Unit-Tests	Unit-Tests	Integrations und Akzeptanz-Tests	Nutzer-Tests	Monitoring
Debugging	Statische Analyse (Syntax Checks, Fehlermuster)		Lasttests	Fehlerdiagnose
Profiling	Dynamische Analyse (Lasttests, Benchmarks)		Monitoring	Elastizität

Versionsmanagement (Anwendungscode, Testdaten, Konfigurationsskripte, Infrastructure-as-Code, etc.)

Continuous Testing



Mit DevOps werden Sie agil über den gesamten Software-Lebenszyklus.