Advanced Java

Agenda

Überblick zu Neuigkeiten in den letzten Java Versionen

Java 8 Lambda und Streams, Date/Time-Api

Java 9 Modulsystem (Jigsaw)

Java 10 var

Java 11 Support- und Lizenzpolitik von Oracle ab 2019

Überblick zu Enterprise Java

Java EE vs. Spring

Einführung in Spring-Boot

Praxis-Java Basics

Versionskontrolle (git)

Build-Tools (maven oder gradle)

Unit-Testing (JUnit 5 und Mockito)

Praxis-Java Advanced

Continuous Integration and delivery

Microservices

DevOps

Containerisierung

Java Versionen Legacy

Version	Release	EOL	Content
JDK 1.0	January 1996		First Version
JDK 1.1	February 1997		RMI, JDBC
J2SE 1.2	December 1998		Swing, Collections
J2SE 1.3	May 2000		HotSpotCompiler, JNDI
J2SE 1.4	February 2002		Assert, RegEx, ExceptionChain
J2SE 5.0	September 2004		Annotations, Generics,...
Java SE 6	December 2006		JDBC 4.0, Performance
Java SE 7	July 2011	April 2015	Nwe File I/O, dynamic languages

Java Versionen current

Version	Release	EOL	Content
Java SE 8 (LTS)	March 2014	January 2019 for Oracle (commercial) December 2020 for Oracle (personal use) September 2023 for AdoptOpenJDK	Lambdas, Streams, New Date/Time
Java SE 9	September 2017	March 2018 for OpenJDK	Modulsystem
Java SE 10	March 2018	September 2018 for OpenJDK	Local-variable type inference
Java SE 11 (LTS)	September 2018	At least September 2022 for AdoptOpenJDK	HTTP client (standard)
Java SE 12	March 2019	September 2019 for OpenJDK	Microbenchmark Suite

Java 8

Lambdas und Streams

Date/Time-Api

Java Modularisierung

Packages

Jar mit maven/gradle

Osgi

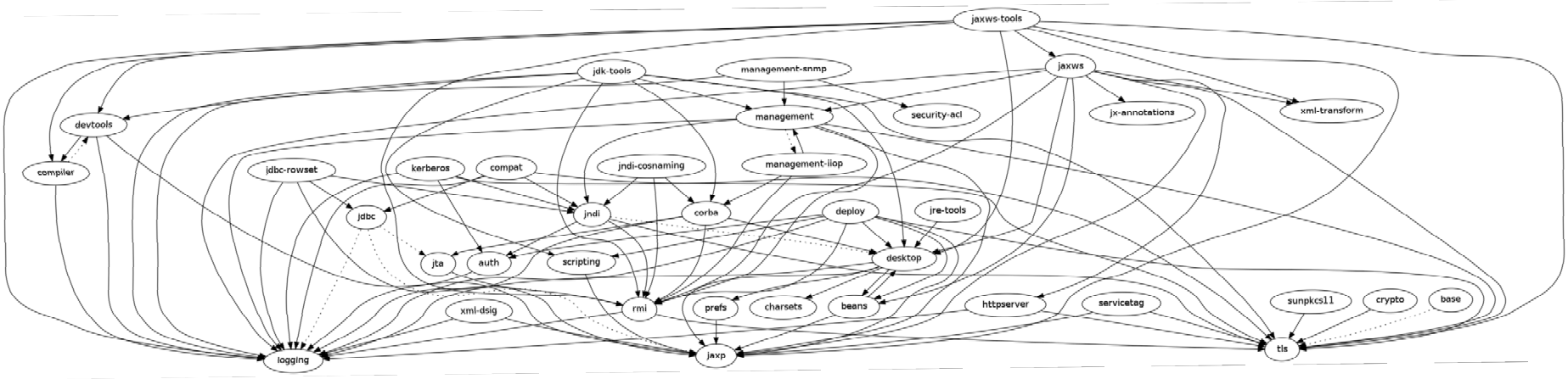
Jigsaw

https://www.baeldung.com/java-9-modularity

https://github.com/eugenp/tutorials/tree/master/core-java-9

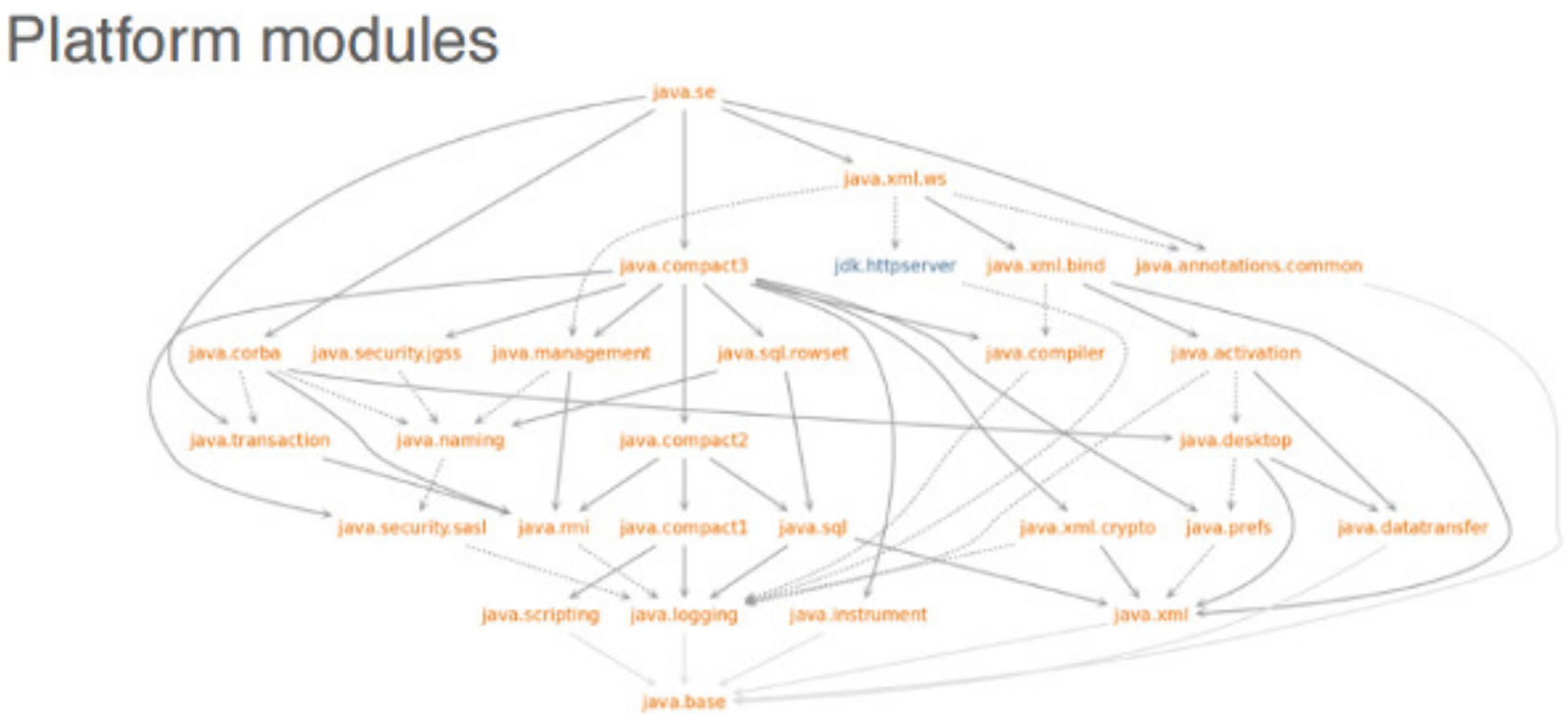
Java 9 - Modulsystem

Das JDK – Vorher (jdk7)



Java 9 - Modulsystem

Das JDK – Jetzt (jdk9)



Java 10

var

Java 11 - Lizenz

- Keine Unterscheidung JRE und JDK mehr
- Oracle JDK nur noch für Entwicklung/Privat kostenlos
- Kommerziell OpenJdk oder Derivate (AdoptOpenJdk, Coretto, etc.)
- Nicht-LTS nur noch bis zur nächsten Version Support
- LTS nur 6 Monate kostenlos, danach kostenpflichtig
- OpenJdk länger (für 11 bis 2022)

Agenda

Überblick zu Neuigkeiten in den letzten Java Versionen

Java 8 Lambda und Streams, Date/Time-Api

Java 9 Modulsystem (Jigsaw)

Java 10 var

Java 11 Support- und Lizenzpolitik von Oracle ab 2019

Überblick zu Enterprise Java

Java EE vs. Spring

Einführung in Spring-Boot

Praxis-Java Basics

Versionskontrolle (git)

Build-Tools (maven oder gradle)

Unit-Testing (JUnit 5 und Mockito)

Praxis-Java Advanced

Continuous Integration and delivery

Microservices

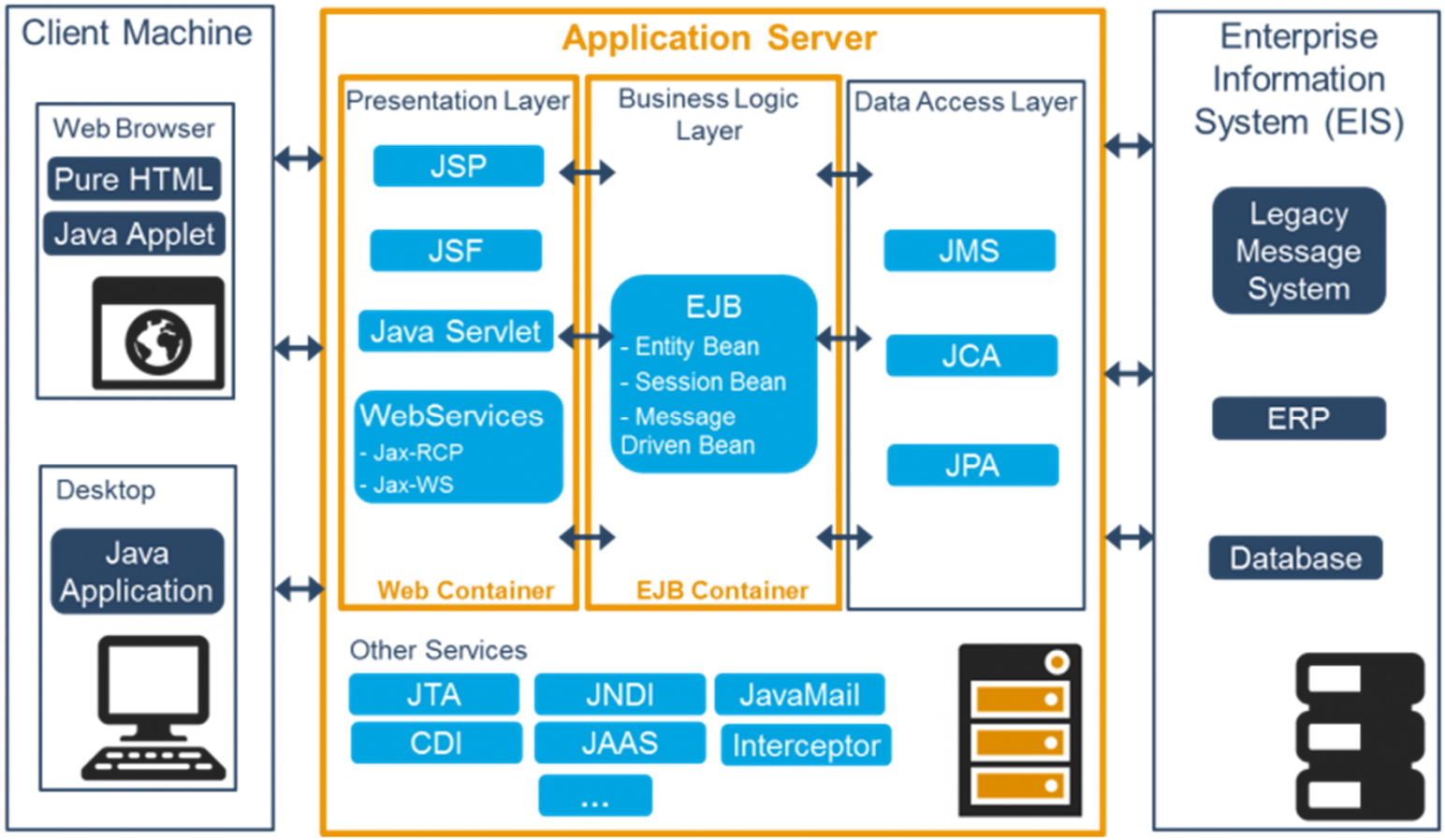
DevOps

Containerisierung

Java „Enterprise“

- Security
- Transaktionsmanagement
- Persistenzdienste
- Namens- und Verzeichnisdienste
- Kommunikation zwischen den Komponenten
- Instanziierung von Objekten
- Deploymentunterstützung

Was ist JEE?


Client Machine
Web Browser
Pure HTML
Java Applet
Desktop
Java Application
Application Server
Presentation Layer
JSP
JSF
Java Servlet
WebServices
- Jax-RCP
- Jax-WS
Web Container
Business Logic Layer
EJB
- Entity Bean
- Session Bean
- Message Driven Bean
EJB Container
Data Access Layer
JMS
JCA
JPA
Other Services
JTA
JNDI
JavaMail
CDI
JAAS
Interceptor
...
Enterprise Information System (EIS)
Legacy Message System
ERP
Database

JEE Spezifikationen

Specifications [edit]

Java EE includes several specifications that serve different purposes, like generating web pages, reading and writing from a database in a transactional way, managing distributed queues.

The Java EE APIs include several technologies that extend the functionality of the base Java SE APIs, such as Enterprise JavaBeans, connectors, servlets, JavaServer Pages and several web service technologies.

Web specifications [edit]

- Servlet: defines how to manage HTTP requests, in a synchronous or asynchronous way. It is low level and other Java EE specifications rely on it;
- WebSocket: The Java API for WebSocket specification defines a set of APIs to service WebSocket connections;
- Java Server Faces: a technology for constructing user interfaces out of components;
- Unified Expression Language (*EL*) is a simple language originally designed to satisfy the specific needs of web application developers. It is used specifically in Java Server Faces to bind components to (backing) beans and in Contexts and Dependency Injection to name beans, but can be used throughout the entire platform.

Web service specifications [edit]

- Java API for RESTful Web Services provides support in creating web services according to the Representational State Transfer (REST) architectural pattern;
- Java API for JSON Processing is a set of specifications to manage information encoded in JSON format;
- Java API for JSON Binding provides specifications to convert JSON information into or from Java classes;
- Java Architecture for XML Binding allows mapping XML into Java objects;
- Java API for XML Web Services can be used to create SOAP web services.

Enterprise specifications [edit]

- Contexts and Dependency Injection is a specification to provide a dependency injection container, as in Spring;
- Enterprise JavaBean (*EJB*) specification defines a set of lightweight APIs that an object container (the EJB container) will support in order to provide transactions (using JTA), remote procedure calls (using RMI or RMI-IIOP), concurrency control, dependency injection and access control for business objects. This package contains the Enterprise JavaBeans classes and interfaces that define the contracts between the enterprise bean and its clients and between the enterprise bean and the ejb container.
- Java Persistence API are specifications about object-relational mapping between relation database tables and Java classes.
- Java Transaction API contains the interfaces and annotations to interact with the transaction support offered by Java EE. Even though this API abstracts from the really low-level details, the interfaces are also considered somewhat low-level and the average application developer in Java EE is either assumed to be relying on transparent handling of transactions by the higher level EJB abstractions, or using the annotations provided by this API in combination with CDI managed beans.
- Java Message Service provides a common way for Java programs to create, send, receive and read an enterprise messaging system's messages.

Other specifications [edit]

- Validation: This package contains the annotations and interfaces for the declarative validation support offered by the Bean Validation API. Bean Validation provides a unified way to provide constraints on beans (e.g. JPA model classes) that can be enforced cross-layer. In Java EE, JPA honors bean validation constraints in the persistence layer, while JSF does so in the view layer.
- Batch Applications provides the means to run long running background tasks that possibly involve a large volume of data and which may need to be periodically executed.
- Java EE Connector Architecture is a Java-based technology solution for connecting application servers and enterprise information systems (*EIS*) as part of enterprise application integration (*EAI*) solutions. This is a low-level API aimed at vendors that the average application developer typically does not come in contact with.

JEE Versionen

Version	Release
J2EE 1.2	December 12, 1999
J2EE 1.3	September 24, 2001
J2EE 1.4	November 11, 2003
Java EE 5	May 11, 2006
Java EE 6	December 10, 2009
Java EE 7	May 28, 2013
Java EE 8	August 31, 2017

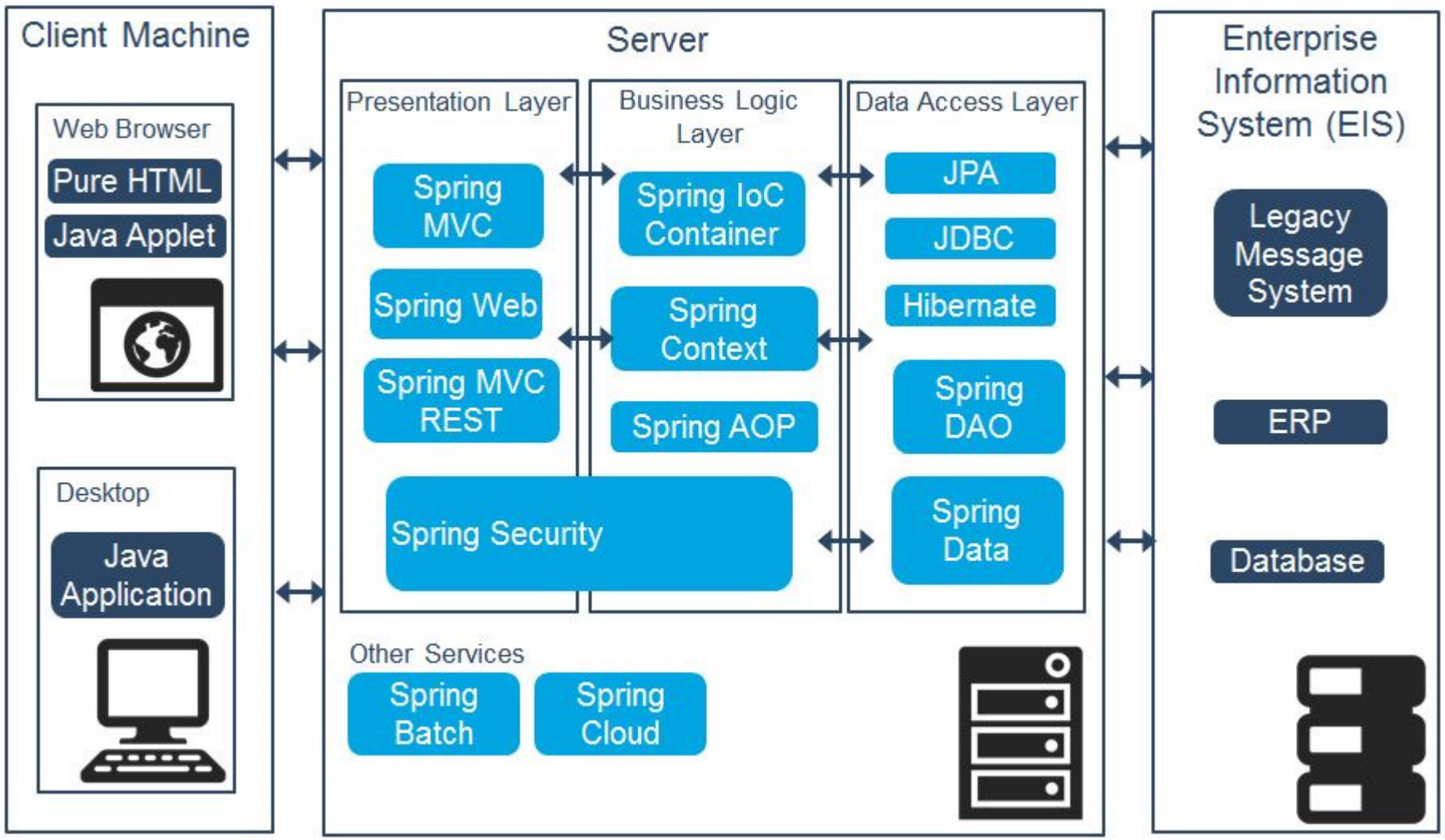
JEE Zukunft

Oracle macht nicht weiter

Eclipse Community hat übernommen – Jakarta EE

2019 soll Jakarta EE 8 erscheinen (ohne Neuerungen)

Spring


Client Machine
Web Browser
Pure HTML
Java Applet
Desktop
Java Application
Server
Presentation Layer
Spring MVC
Spring Web
Spring MVC REST
Business Logic Layer
Spring IoC Container
Spring Context
Spring AOP
Data Access Layer
JPA
JDBC
Hibernate
Spring DAO
Spring Data
Spring Security
Other Services
Spring Batch
Spring Cloud
Enterprise Information System (EIS)
Legacy Message System
ERP
Database

Vergleich Spring vs. JEE

(–) Einarbeitung + Konfiguration AppServer	(+) Keine Einarbeitung in AppServer
(–) Aufwändige Einrichtung eines AppServers bei den Entwicklern	(+) Out-of-the-box-Funktion
(–) nutzen Kunden versch. AppServer muss jeder konfiguriert werden	(+) Applikation kann auf x-beliebigem Webserver Out-of-the-box laufen
(+) AppServer bringt Funktionalitäten mit wie Clustering, TimerService	(–) Eigenimplementierung Funktionalitäten oder Web Server nutzen
(–) Testing über AppServer (Arquillian)	(+) Einfacheres Testing
(+) versch. Implementierungen zur Auswahl	(+) Komponenten austauschbar
(+) Standardisiert. Wechsel möglich.	
(–) Ergänzung neue Features langsamer	(+) Ergänzung neue Features schneller

Spring Boot

https://start.spring.io/

Agenda

Überblick zu Neuigkeiten in den letzten Java Versionen

Java 8 Lambda und Streams, Date/Time-Api

Java 9 Modulsystem (Jigsaw)

Java 10 var

Java 11 Support- und Lizenzpolitik von Oracle ab 2019

Überblick zu Enterprise Java

Java EE vs. Spring

Einführung in Spring-Boot

Praxis-Java Basics

Versionskontrolle (git)

Build-Tools (maven oder gradle)

Unit-Testing (JUnit 5 und Mockito)

Praxis-Java Advanced

Continuous Integration and delivery

Microservices

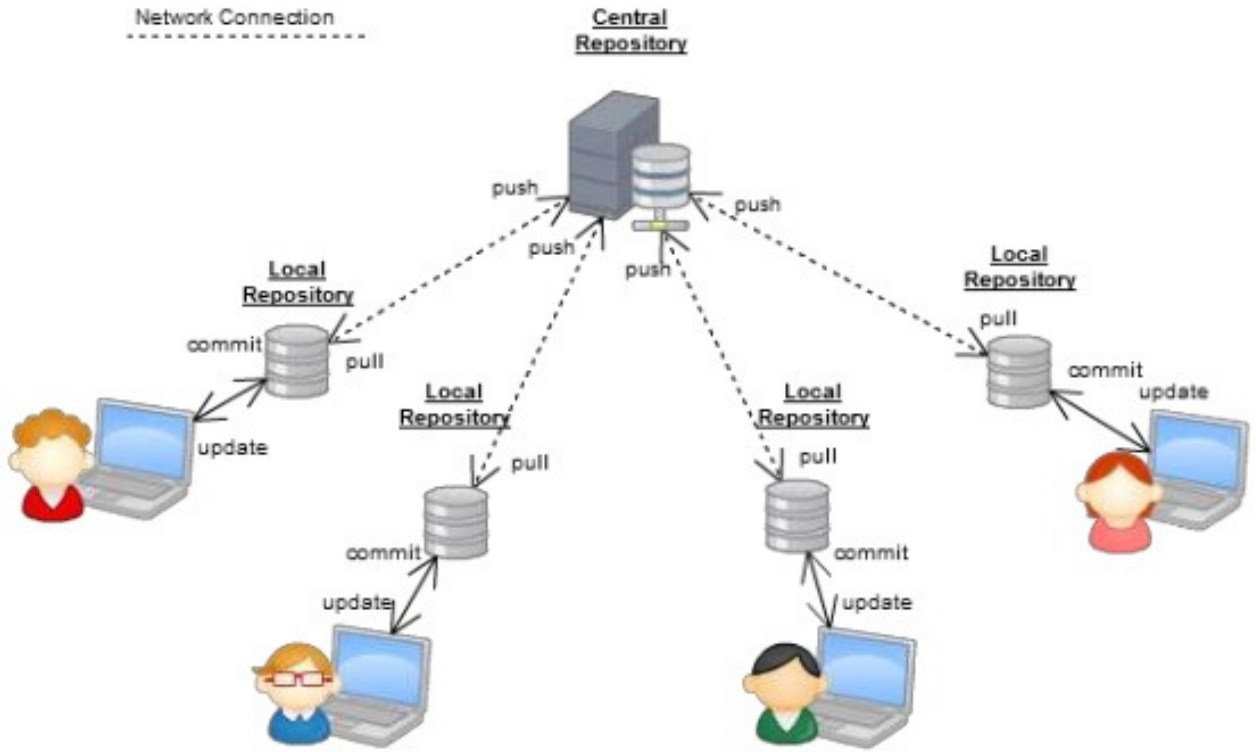
DevOps

Containerisierung

Verteilte Versionskontrolle

Hochschule Reutlingen
Reutlingen University

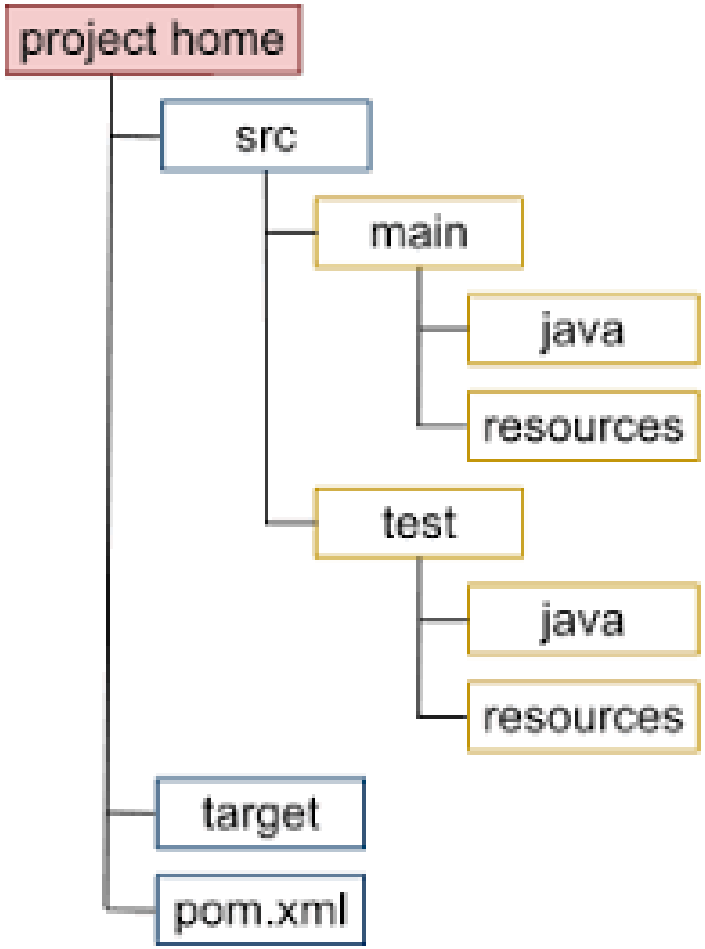
Verteilte Versionsverwaltung



Git

- Vom „Macher“ von Linux: Linus Torvalds
- Jeder hat das komplette Repo samt History lokal verfügbar
- Sehr effiziente Speicherung der Daten
- Branch und Merge integraler Bestandteil des Werkzeugs
- Wichtigste Befehle:
  - Clone: (Remote-)Projekt auschecken
  - Status: Unterschied zwischen lokal und remote feststellen
  - Fetch: Änderungen von remote holen ohne Verarbeitung
  - Pull/Rebase: Aktualisieren des lokalen repos von remote
  - Add: markieren von Änderungen für das commit
  - Commit: Einchecken ins lokale repo
  - Push: Hochladen ins remote repo

Maven Struktur

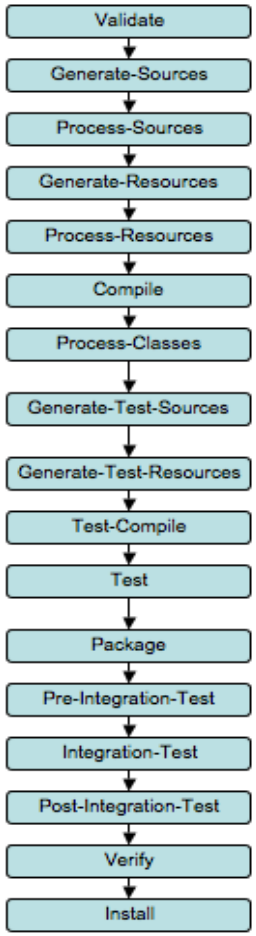


- M2
- Artifact repository
- Internet

Maven Lifecycle

Clean (eigener Befehl)

Validate
Compile
Test
Package
Verify
Install
Deploy



Unit Testing

Unit-Testing (JUnit 5 und Mockito)

https://github.com/eugenp/tutorials/tree/master/testing-modules/junit-5

Agenda

Überblick zu Neuigkeiten in den letzten Java Versionen

Java 8 Lambda und Streams, Date/Time-Api

Java 9 Modulsystem (Jigsaw)

Java 10 var

Java 11 Support- und Lizenzpolitik von Oracle ab 2019

Überblick zu Enterprise Java

Java EE vs. Spring

Einführung in Spring-Boot

Praxis-Java Basics

Versionskontrolle (git)

Build-Tools (maven oder gradle)

Unit-Testing (JUnit 5 und Mockito)

Praxis-Java Advanced

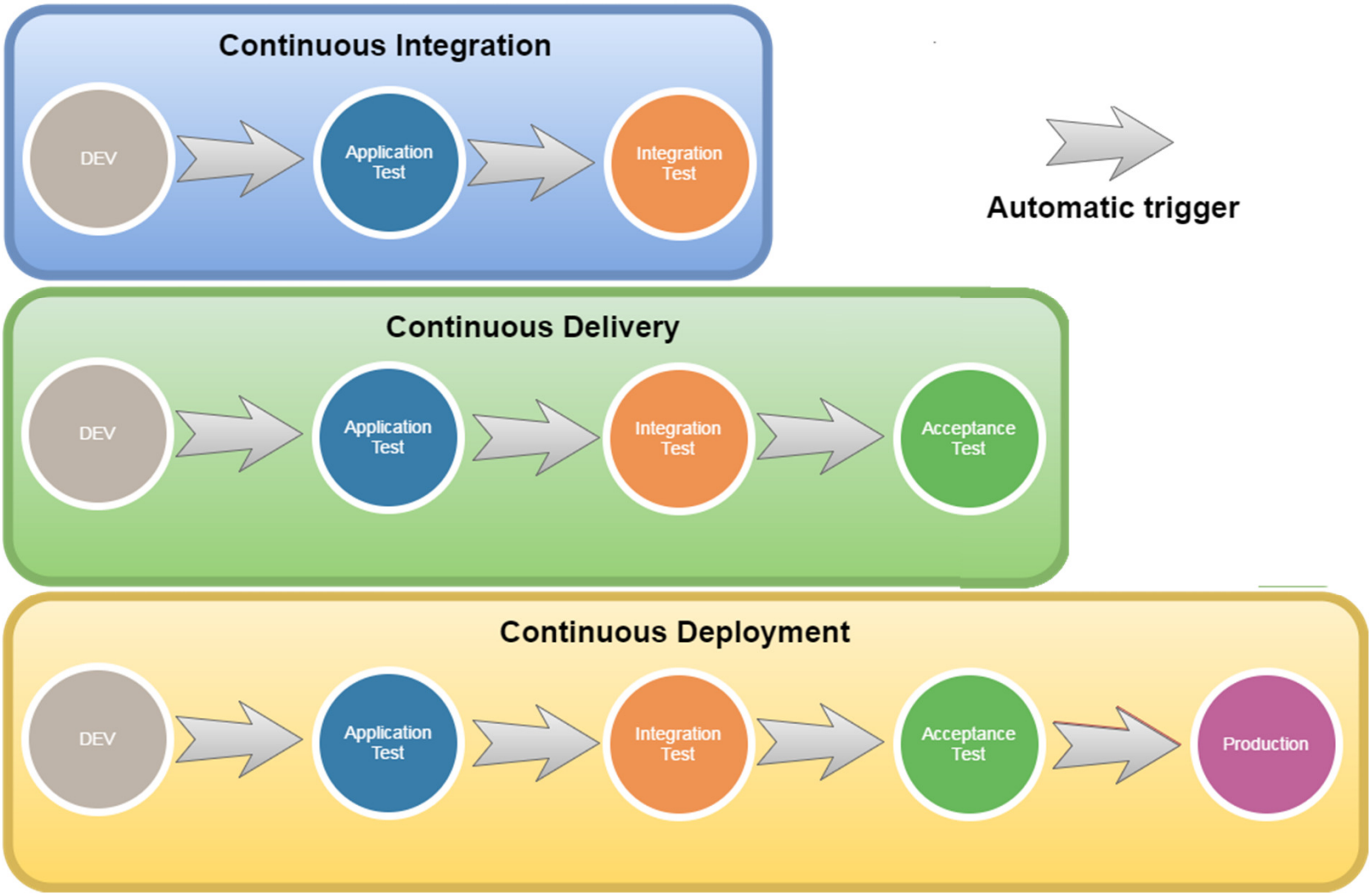
Continuous Integration and delivery

Microservices

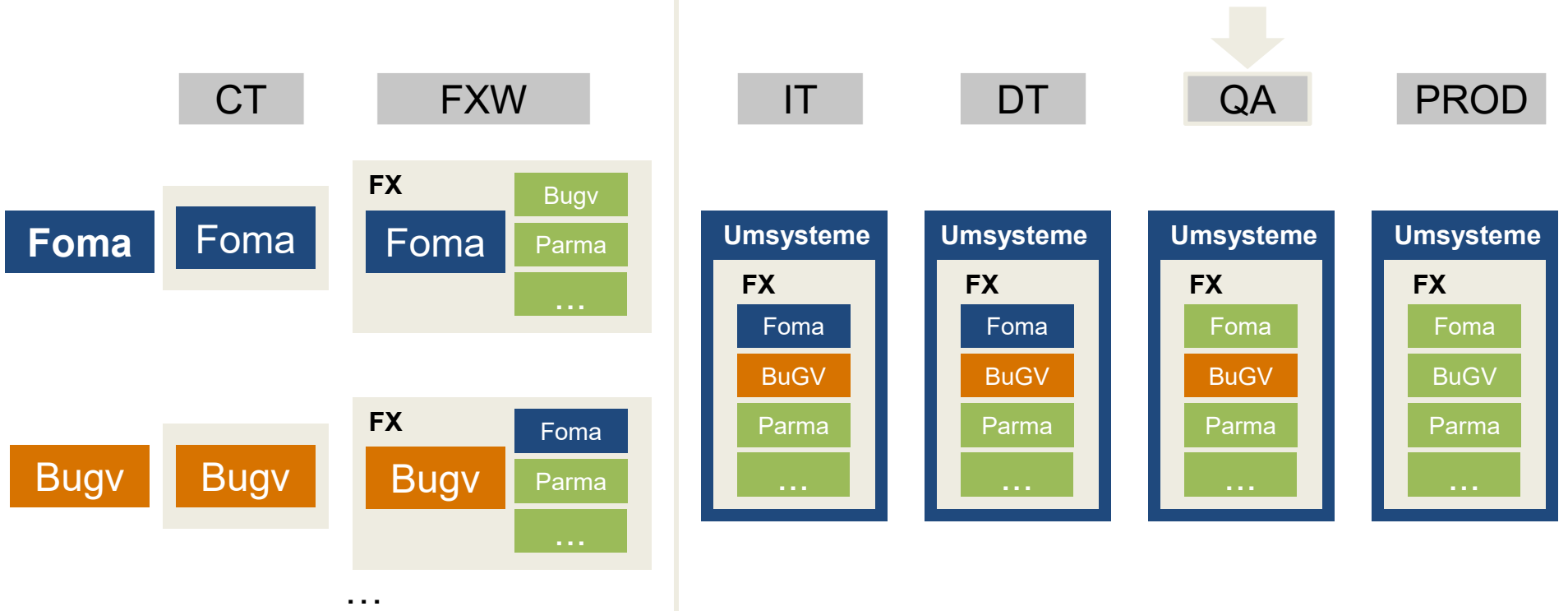
DevOps

Containerisierung

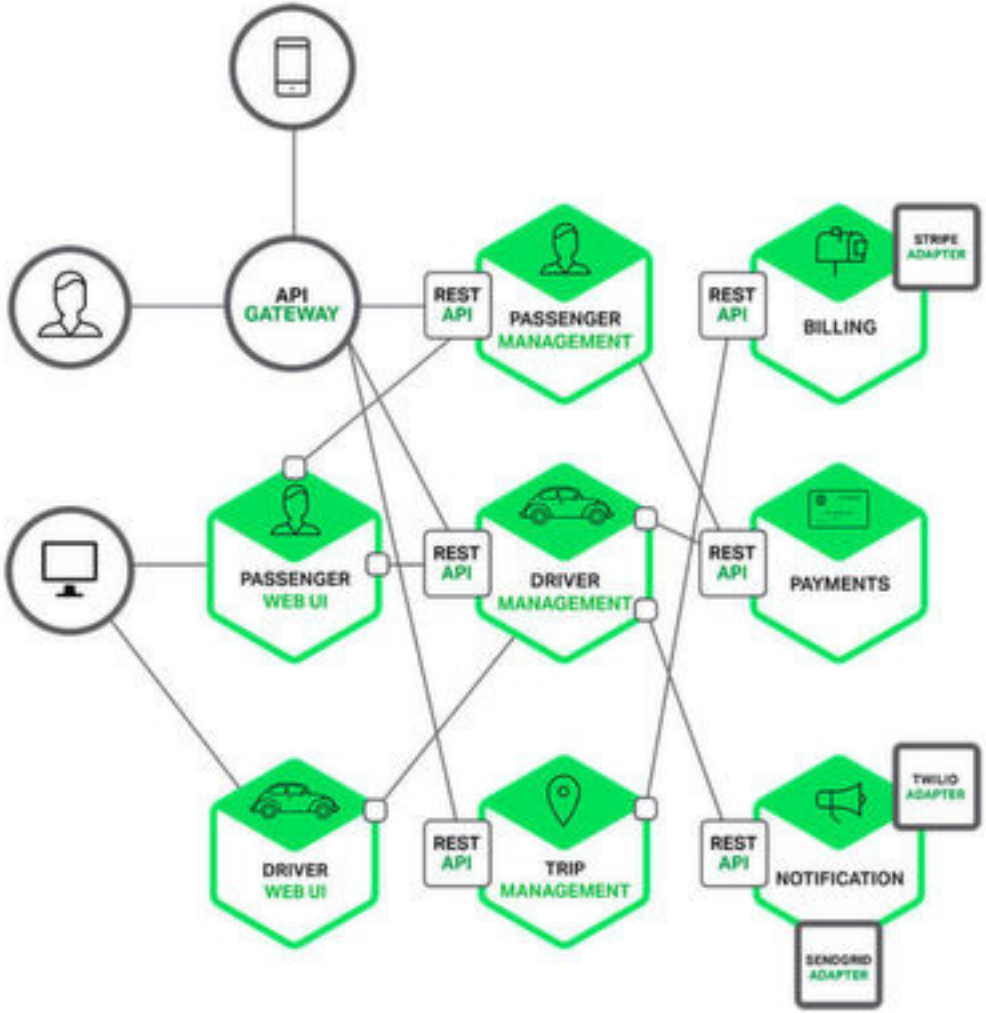
CI & CD


Continuous Integration
DEV
Application Test
Integration Test
Automatic trigger
Continuous Delivery
DEV
Application Test
Integration Test
Acceptance Test
Continuous Deployment
DEV
Application Test
Integration Test
Acceptance Test
Production

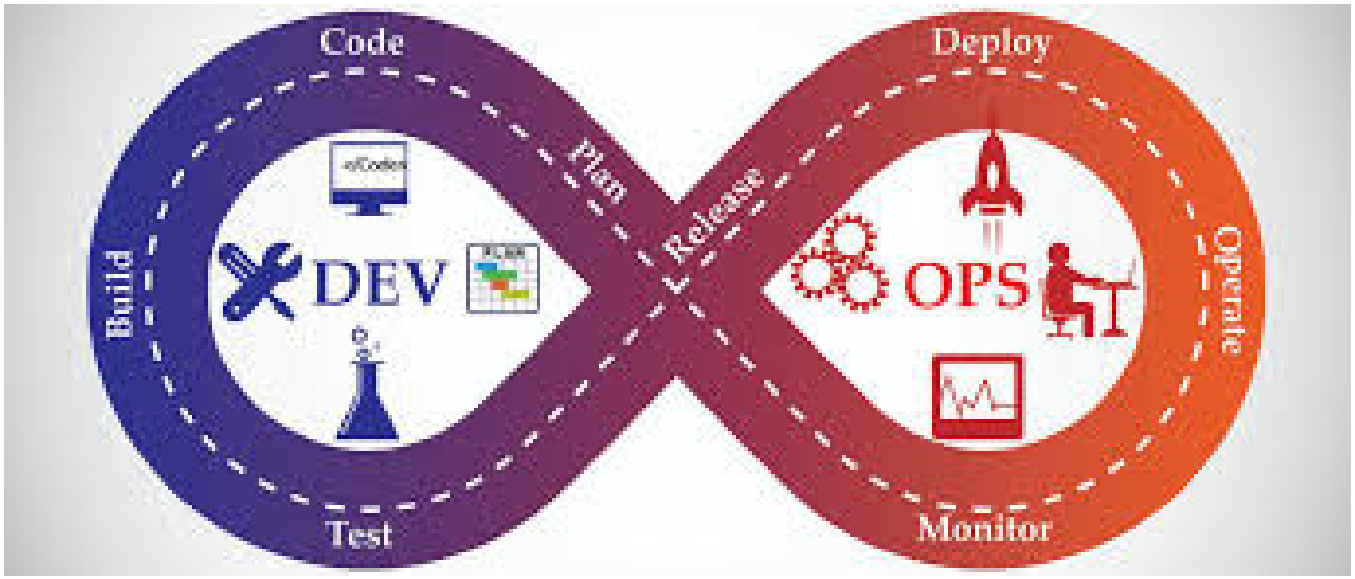
Ein Beispiel



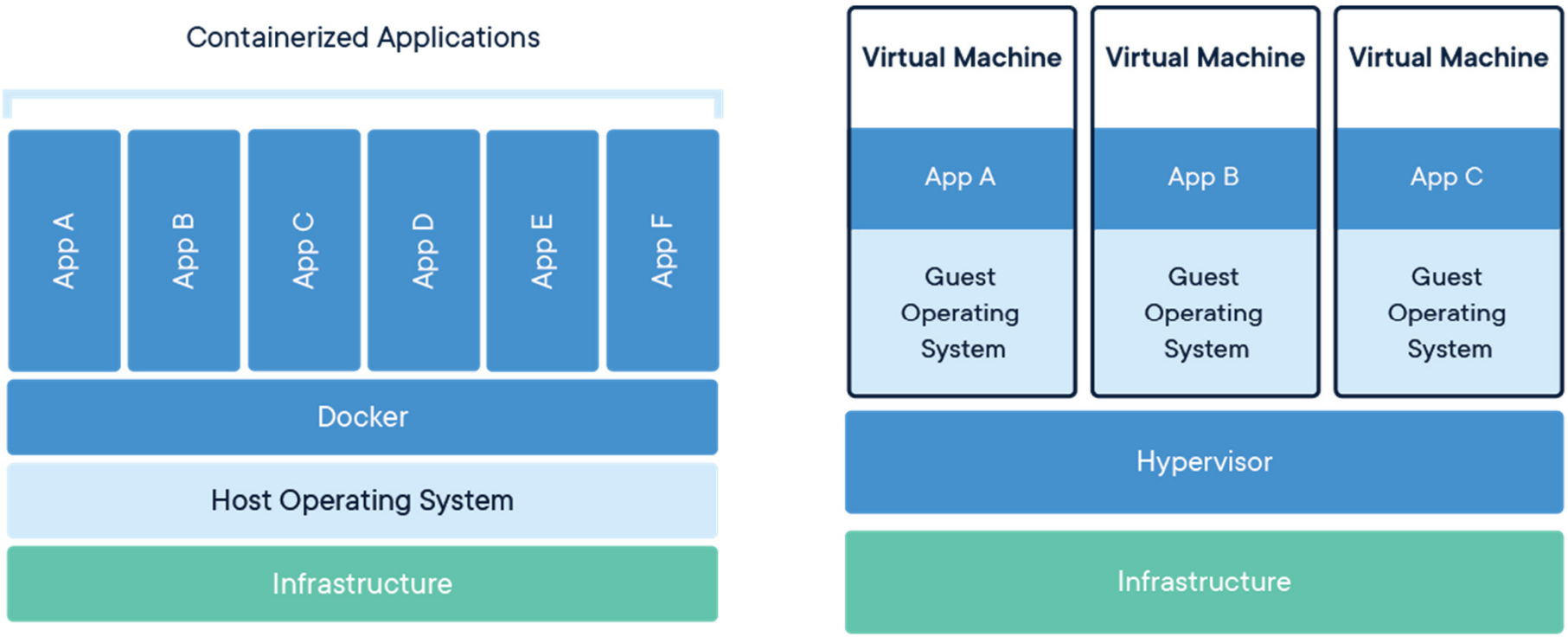
Microservices



DevOps


Code
Plan
Build
Test
DEV
Release
Deploy
Operate
Monitor
OPS

Container


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure
Virtual Machine
App A
Guest Operating System
Virtual Machine
App B
Guest Operating System
Virtual Machine
App C
Guest Operating System
Hypervisor
Infrastructure