» KOMMUNALE IT - VON MENSCH ZU MENSCH «

Kommunales Rechenzentrum Niederrhein (KRZN)
Friedrich-Heinrich-Allee 130
47475 Kamp-Lintfort
Internet: www.krzn.de

Vortrag: Christoph Thoma

eGovernment Entwicklung

Vom Monolithen zu Microservices

Der Change in der Softwarearchitektur

Agenda

Notwendigkeit
Warum eine Umstellung der Entwicklung?

Infrastruktur
Welche Systeme setzen wir ein? Warum? Wie?

Architektur
Wie sind die Komponenten aufgebaut?

Mitarbeiter
Wie soll ein Entwickler arbeiten?

Fazit
Wo stehen wir? Was ändert sich?

Leitkriterien

1. Klein
2. Schnell
3. Vernetzt
4. Automatisiert
5. Robust

Notwendigkeit

Warum eine Umstellung der Entwicklung?

Globale Sicht

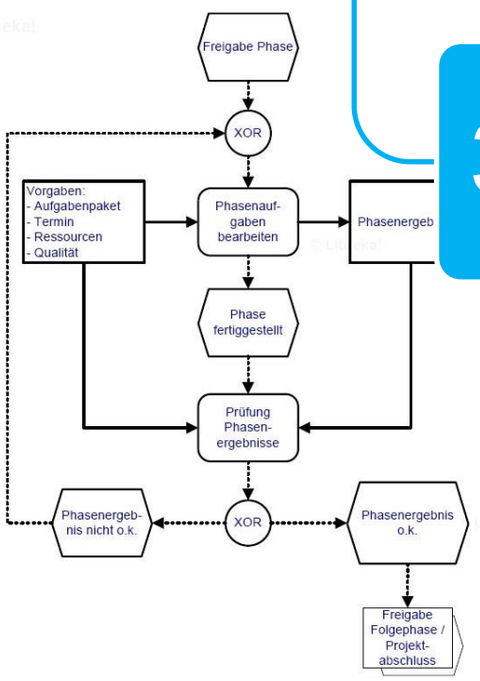
Digitale Revolution

- Digitaltechnik
- Computer
- Prozesse
- Logik

3te Revolution

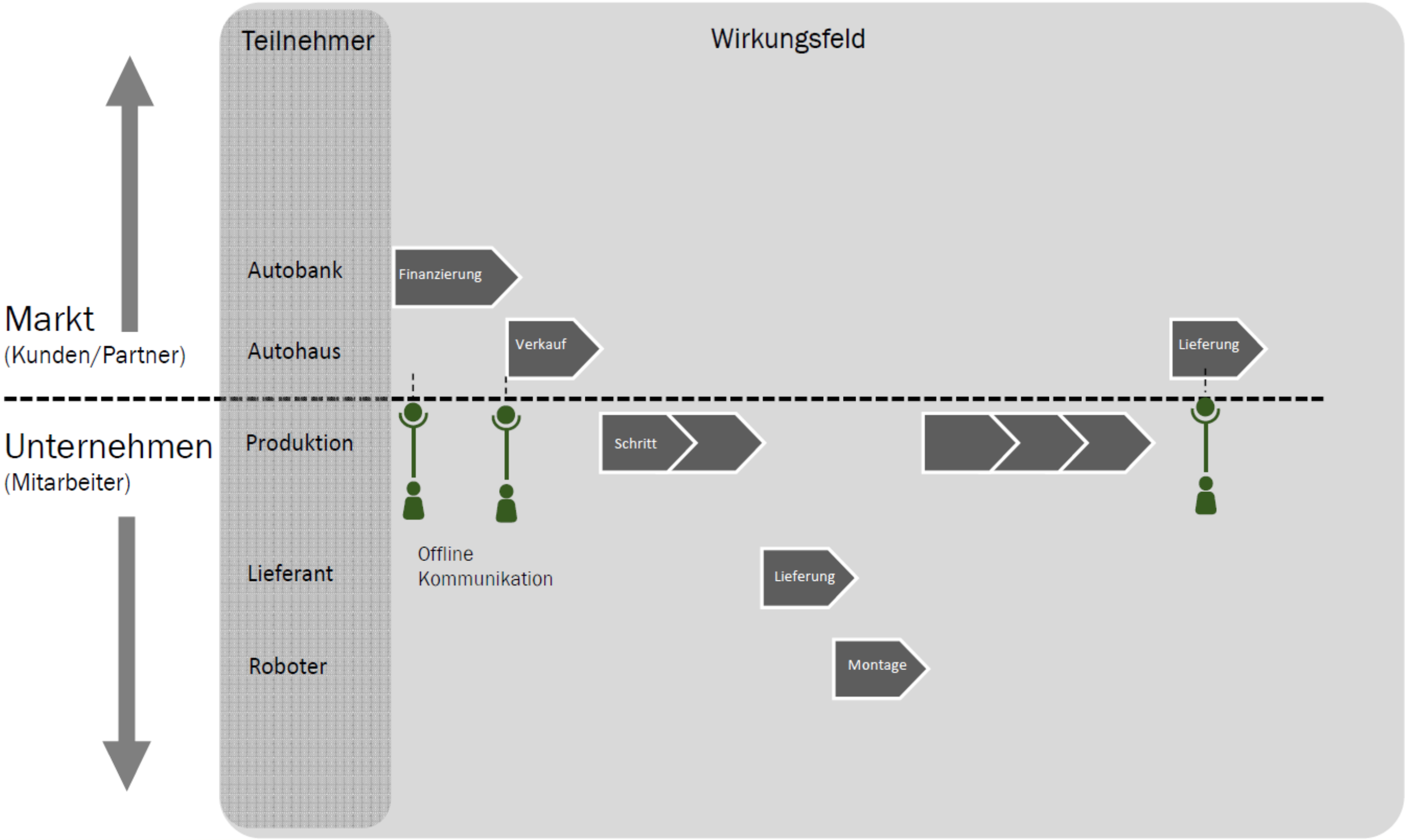
4te Revolution

- Internet
- Cloud
- Smart Devices
- Services

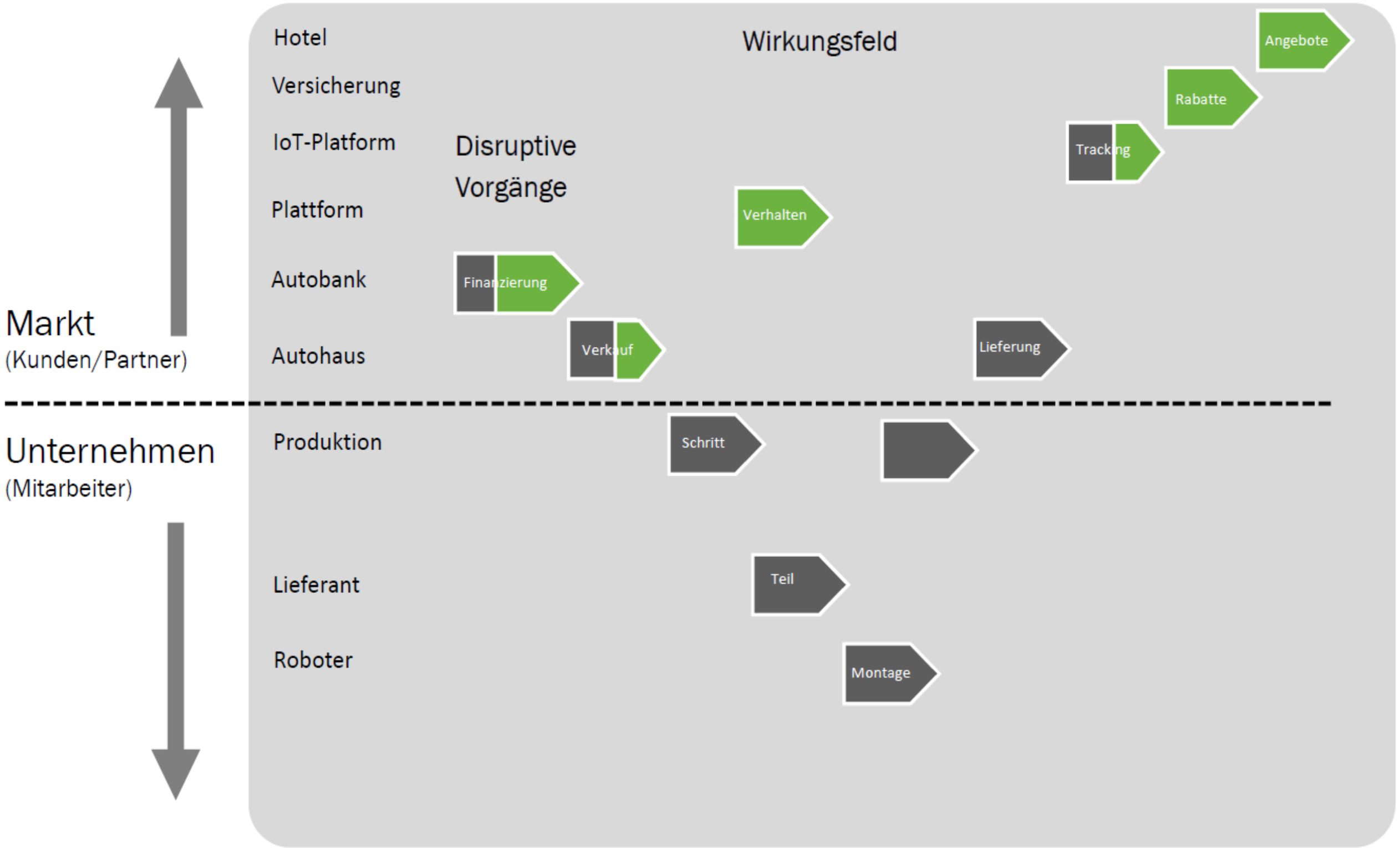


Digitale Transformation

Prozess Produkt Fokus



Prozess Service Fokus



Notwendigkeit

Warum eine Umstellung der Entwicklung?

Technische Sicht

System-Technologie-Fokus

eGovernment Suite < > eGovernment Integration

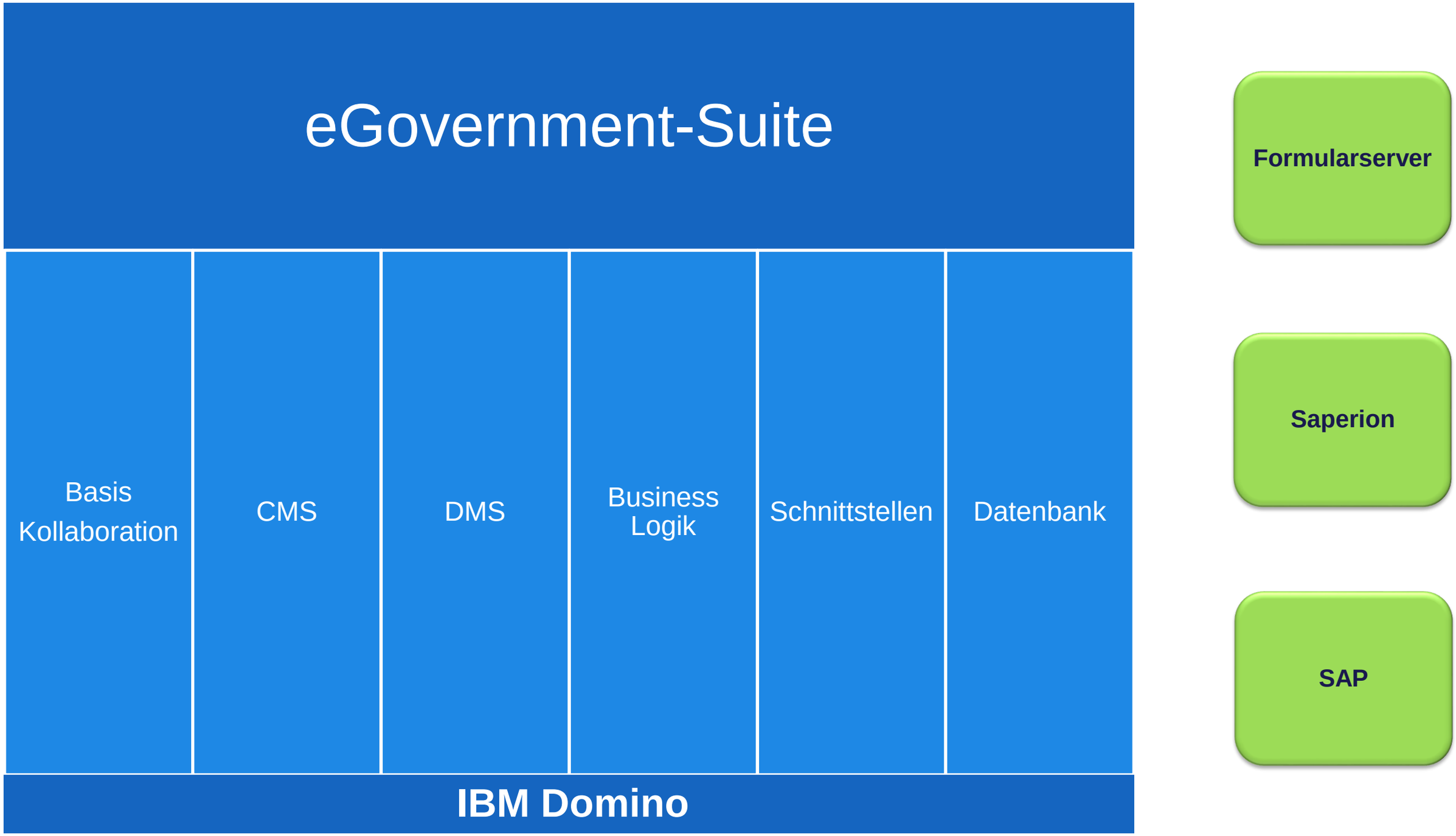
Notwendigkeit

Warum eine Umstellung der Entwicklung?

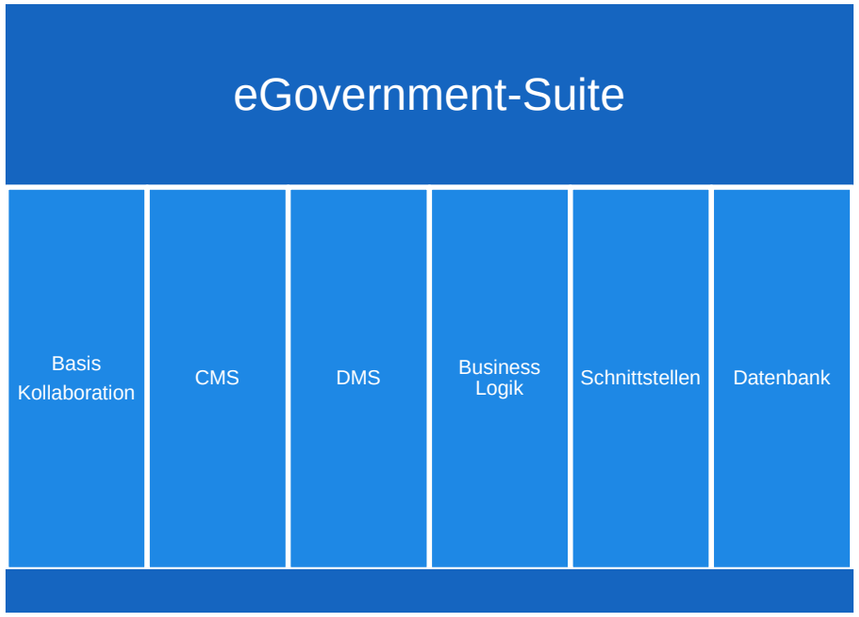
KRZN Sicht

Monolith

Aktuelle Struktur



Einzelne Verfahren!



CMS
Drupal

DMS
Fabasoft

Workflow
Fabasoft
Camunda

Datenbank

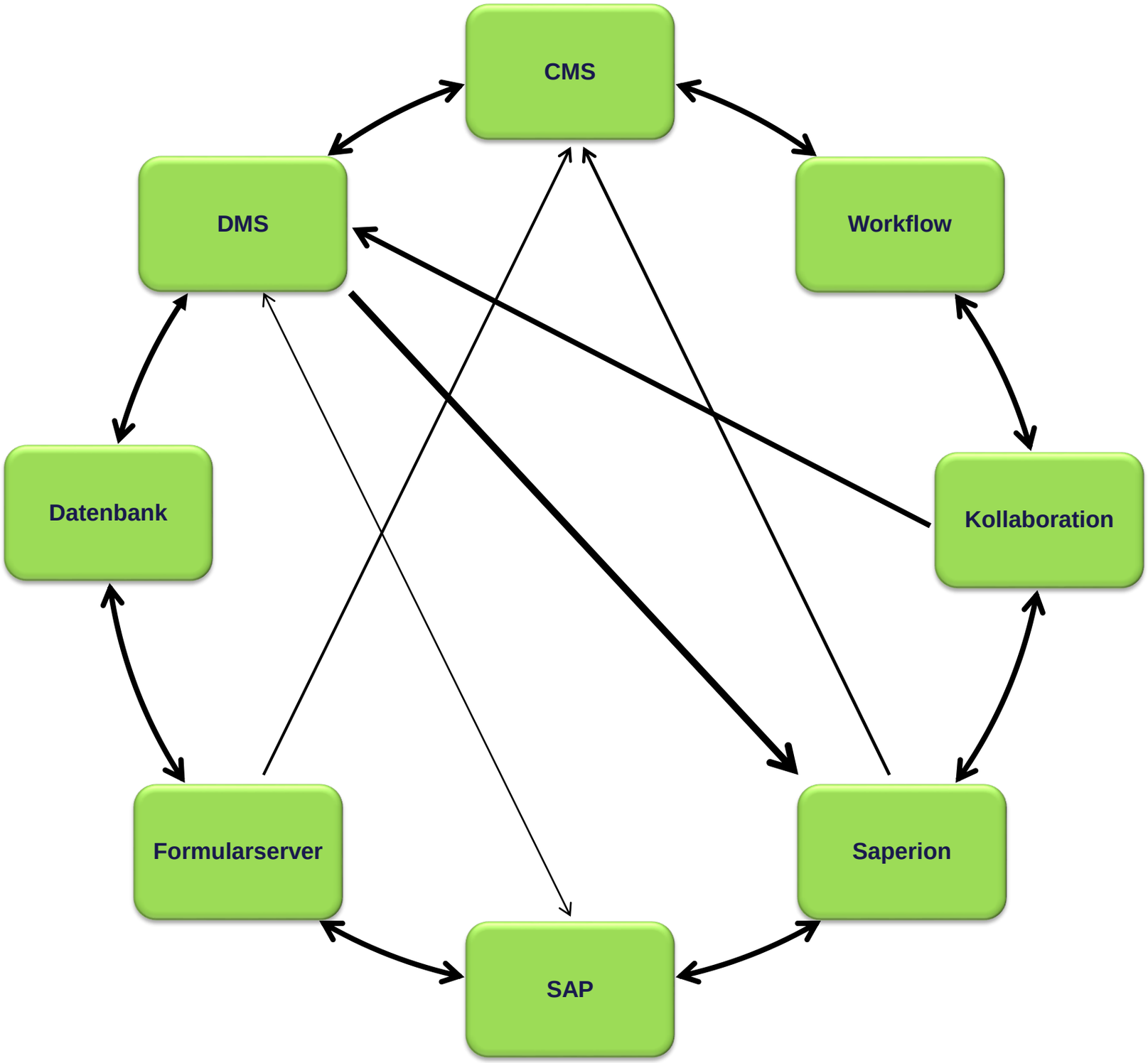
Kollaboration

Formularserver

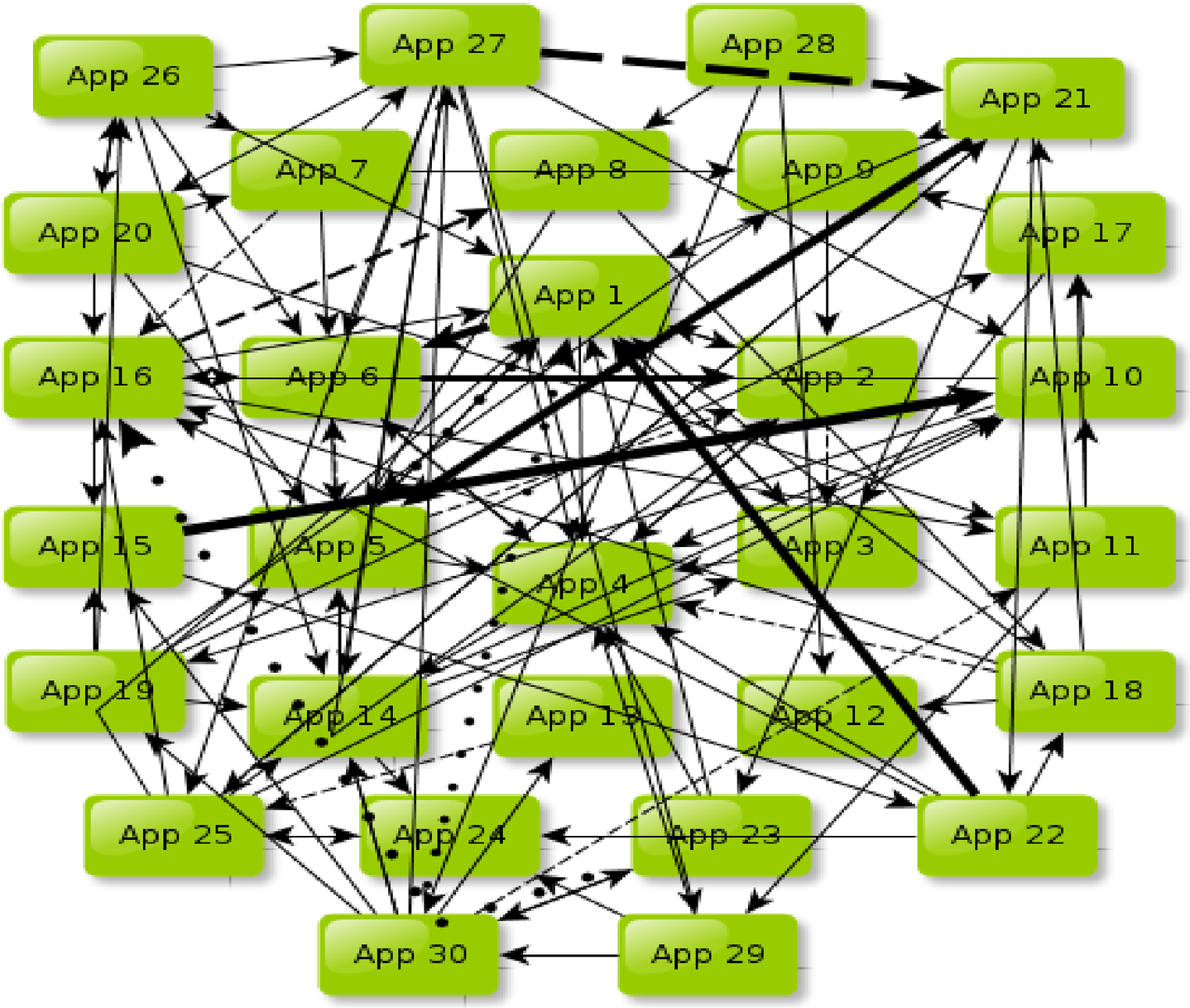
Saperion

SAP

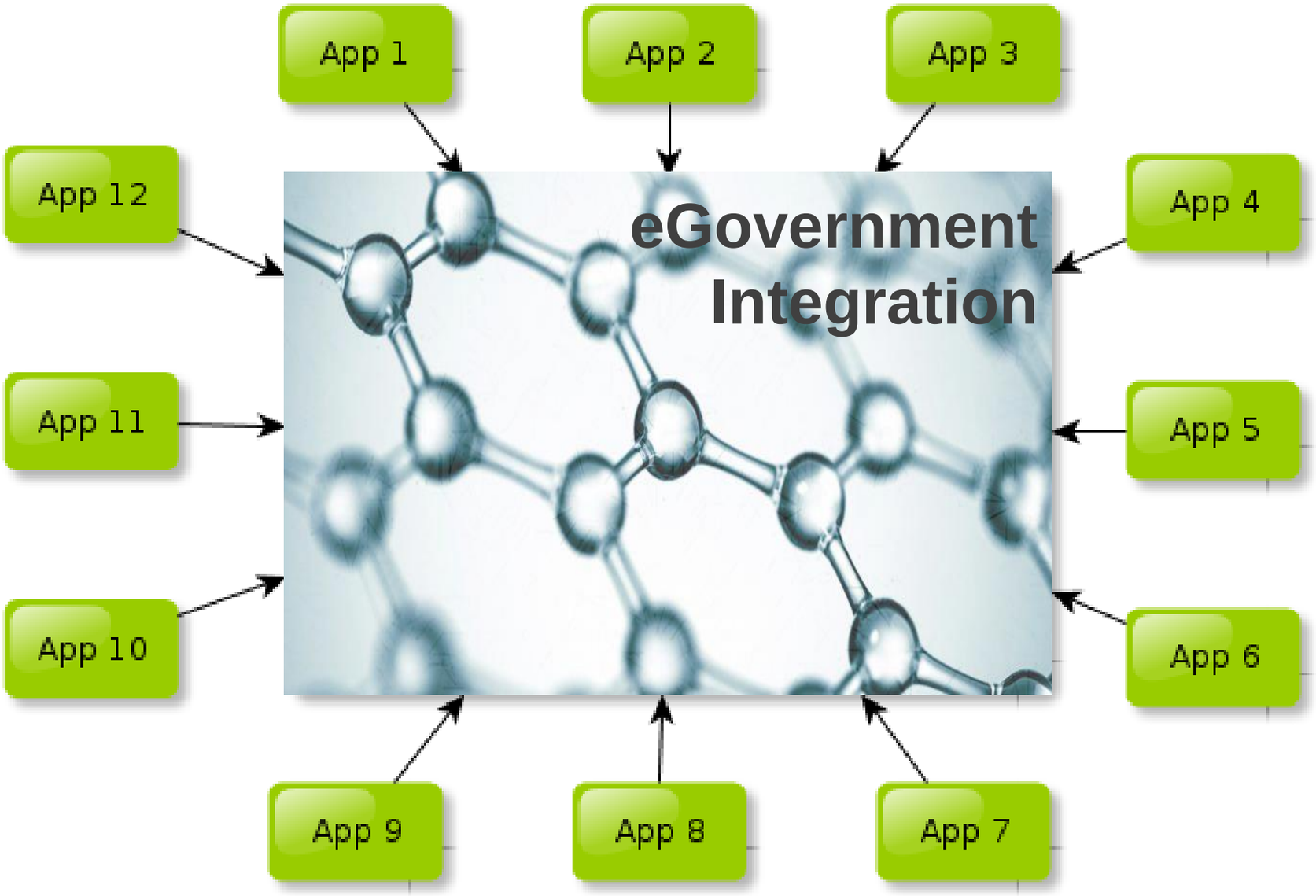
Was wird gefordert?



Worauf läuft es hinaus?



Wie wollen wir das lösen?



Auswirkungen auf Leitkriterien – Wir müssen:

1 **Klein**
Abhängigkeiten minimieren, um schnelles Bereitstellen zu ermöglichen

2 **Schnell**
Anpassung an veränderte Services und Verfahren umsetzen

3 **Vernetzt**
Services und Verfahren verbinden

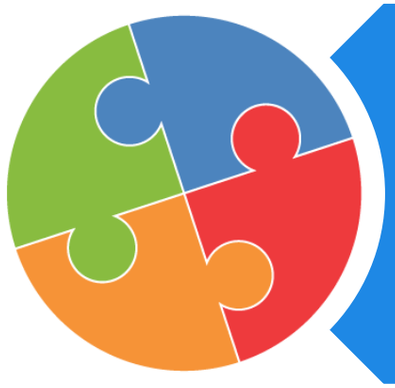
4 **Automatisiert**
Die Anpassungen nach definierten Abläufen (Zertifizierung) bereitstellen

5 **Robust**
Es muss funktionieren

Auswirkungen

Systeme

Infrastruktur der Server
Welche Systeme benutzen wir?



Architektur

Aufbau der Komponenten
Wie hängen die Systeme zusammen?

Mitarbeiter

Arbeitsweisen der Entwicklung
Wie benutzen wir die Systeme?

Was wollen wir erreichen

Chance nutzen
.komplett neu aufzustellen

Vernetzung
.lose Koppelung der Systeme

Arbeitsweise
.überdenken und verbessern

Technologiewechsel
.modern, aktuelle Entwicklung

Ablösung
.alte Zöpfe abschneiden

Skalierbarkeit
.horizontal und vertikal

Kosten
.Open Source wenn sinnvoll

Abhängigkeit minimieren
.austauschbare Komponenten

Zertifizierung
.Netze, Zugriffe, Deployment, Produktion

Synergieeffekte im KRZN
.wiederverwendbare Strukturen

Agenda

Notwendigkeit
Warum eine Umstellung der Entwicklung?

Infrastruktur
Welche Systeme setzen wir ein? Warum? Wie?

Architektur
Wie sind die Komponenten aufgebaut?

Mitarbeiter
Wie soll ein Entwickler arbeiten?

Fazit
Wo stehen wir? Was ändert sich?

Cloud Architektur

- Software, die der Endkunde unmittelbar einsetzen kann.
- Betrieb liegt vollständig beim Anbieter
- Wartung, Aktualisierung, Fehlerbeseitigung oder Weiterentwicklung liegt beim Anbieter

SaaS
Software as a Service

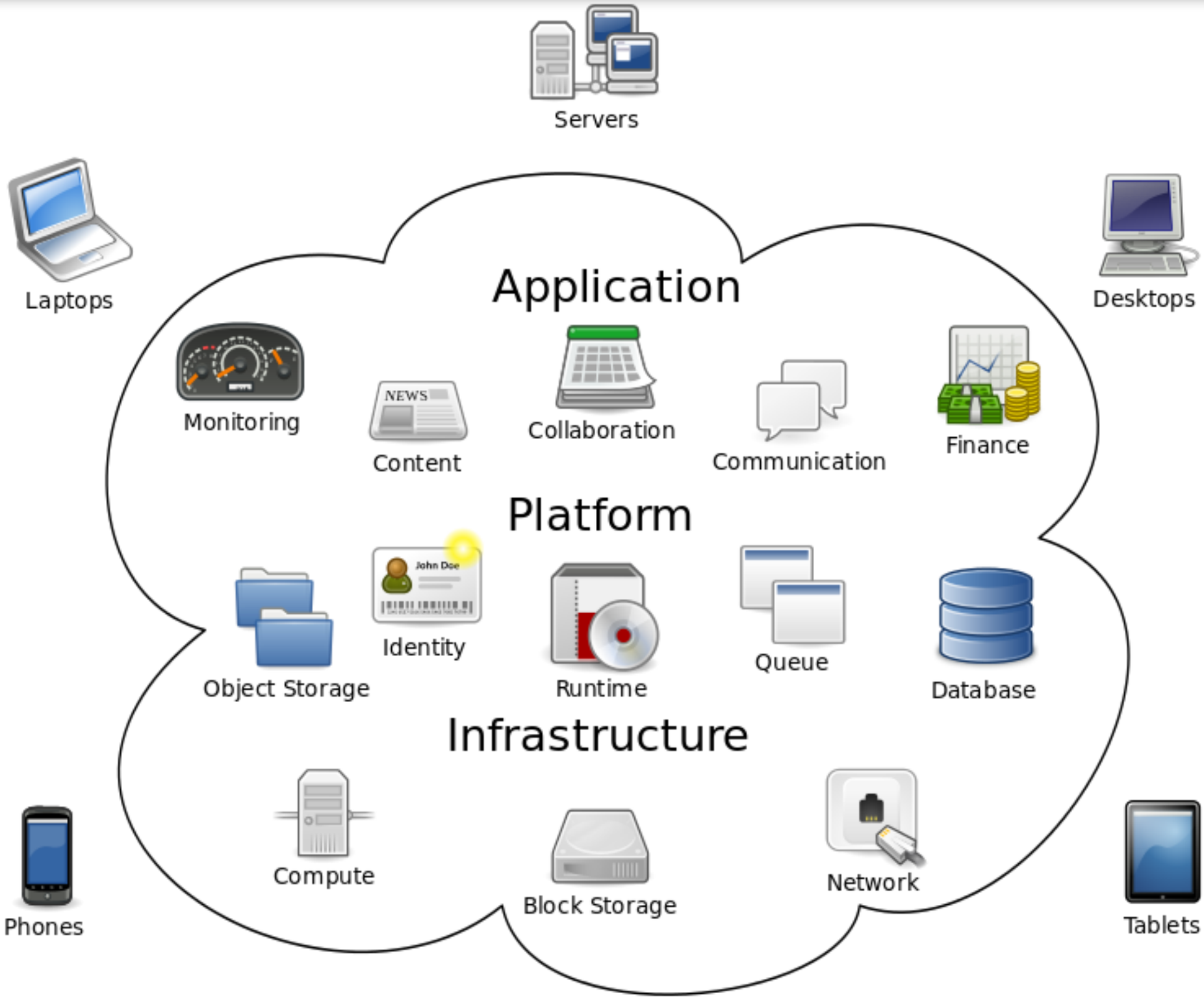
- Eigene Programme auf einer Plattform in der Cloud
- Rahmenbedingungen werden vorgegeben
  z. B. die Programmiersprache und Bibliotheken oder Schnittstellen.

PaaS
Platform as a Service

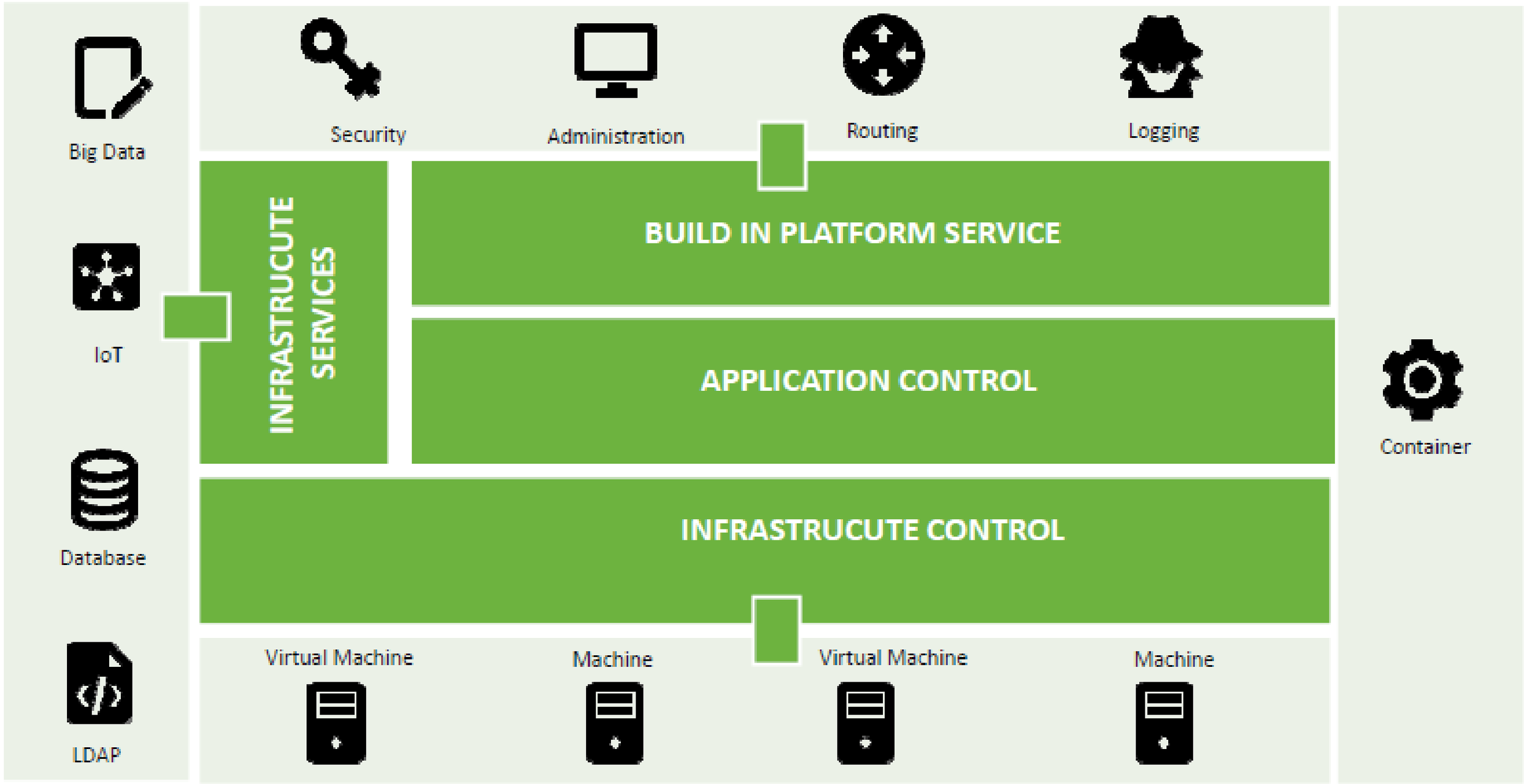
- Hardware oder Infrastrukturdienste in der Cloud
  z. B. Speicherplatz, Rechenleistung oder Netzwerkbandbreite.

IaaS
Infrastructure as a Service

Cloud Struktur



Platform as a Service (PaaS)



eGovernment-Integration

GitLab

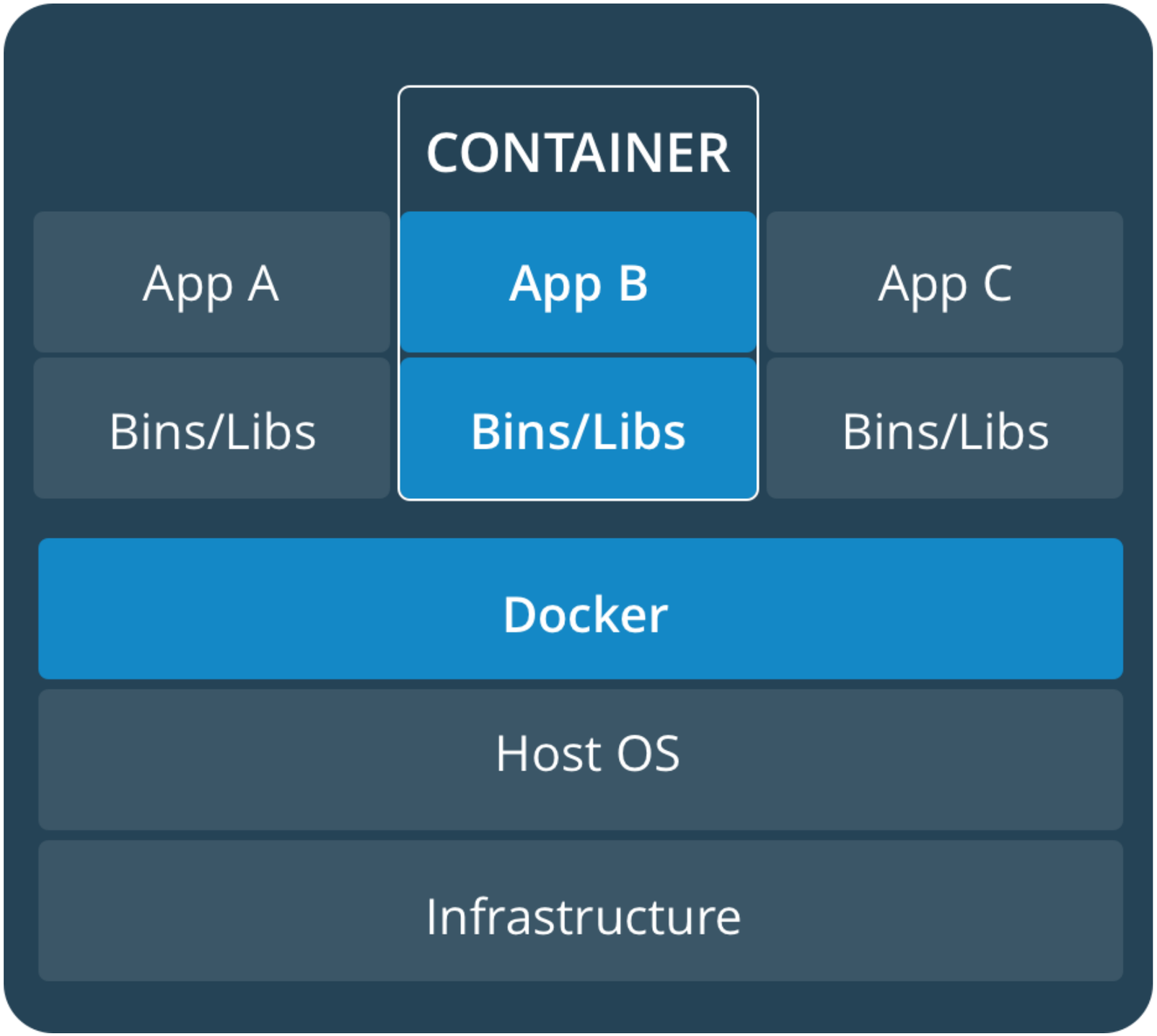
GitLab ist eine Webanwendung zur Versionsverwaltung für Softwareprojekte auf Basis von git.

- Git - Von Linus Torvalds
- Git Standard im Web

Link - GitLab

https://gitlab.com/

Container Technologie



Laufzeitumgebung

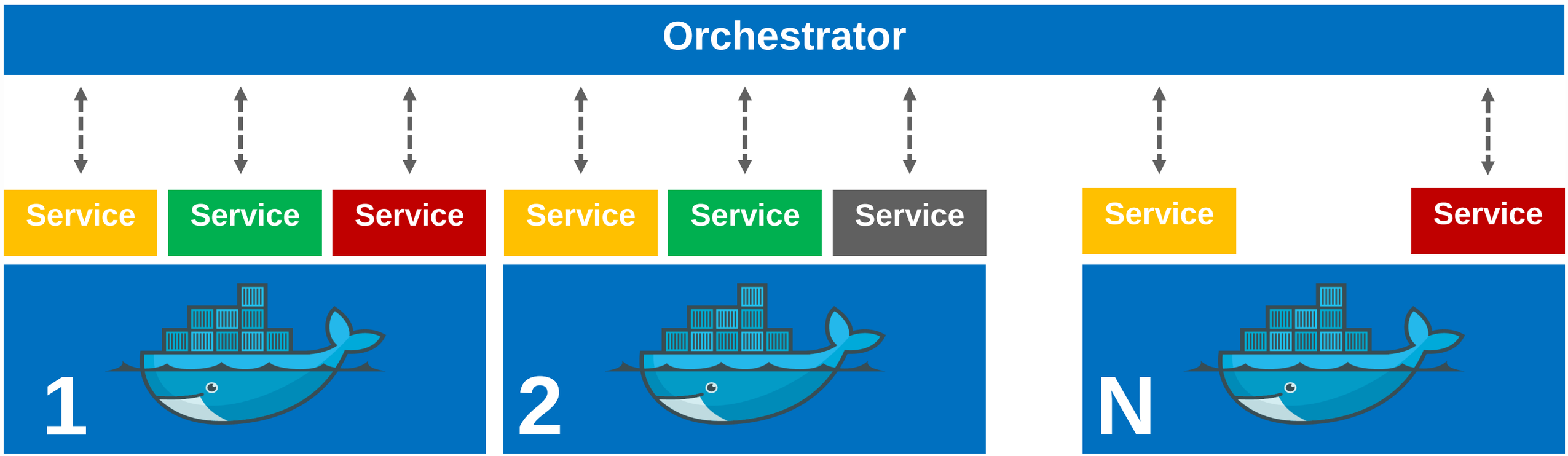
eGovernment-Integration

Docker

Universelle Laufzeitumgebung für Container.

» Basiert auf Linux

» Isoliert Anwendungen in Containern



eGovernment-Integration

Jenkins

Jenkins ist ein erweiterbares, webbasiertes Software-System zur kontinuierlichen Integration von Komponenten zu einem Anwendungsprogramm.

» Automatisierung

» Build-Tool, Deployment-Tool

Link - Jenkins

https://jenkins.io/

eGovernment-Integration

Jira

Jira ist eine Webanwendung zur Fehlerverwaltung, Problembehandlung und operativem Projektmanagement.

- » Workflow Management
- » Hoch integrierbar
- » Flexibles, modernes UI

Link - Jira

https://de.atlassian.com/software/jira

eGovernment-Integration

Maven - Nexus

Java-Programme standardisiert erstellen und verwalten.

- Zugriff auf Bibliotheken und Plug-Ins
- Abhängigkeiten von Frameworks

Link - Maven

https://www.sonatype.com/

Logging – ELK Stack

Suchen & Analysieren

elasticsearch

Sammeln & Transformieren

logstash

Visualisieren & Verwalten

kibana

Anwendungen **Benutzer**

Link – ELK Stack

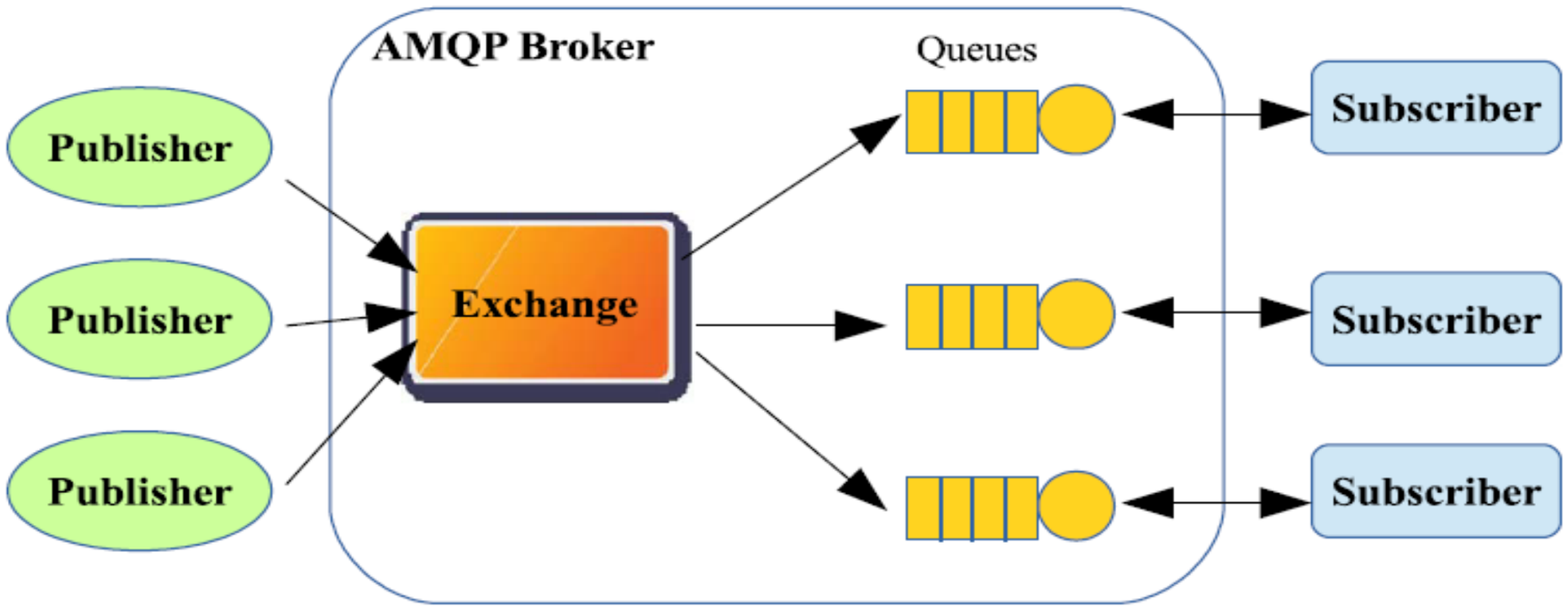
https://www.elastic.co/de/elk-stack

eGovernment-Integration

RabbitMQ

Verändert die Kommunikation zwischen Anwendungen.

>> Synchrone Kommunikation zu asynchrone Kommunikation <<



Link – RabbitMQ

https://www.rabbitmq.com/

eGovernment-Integration

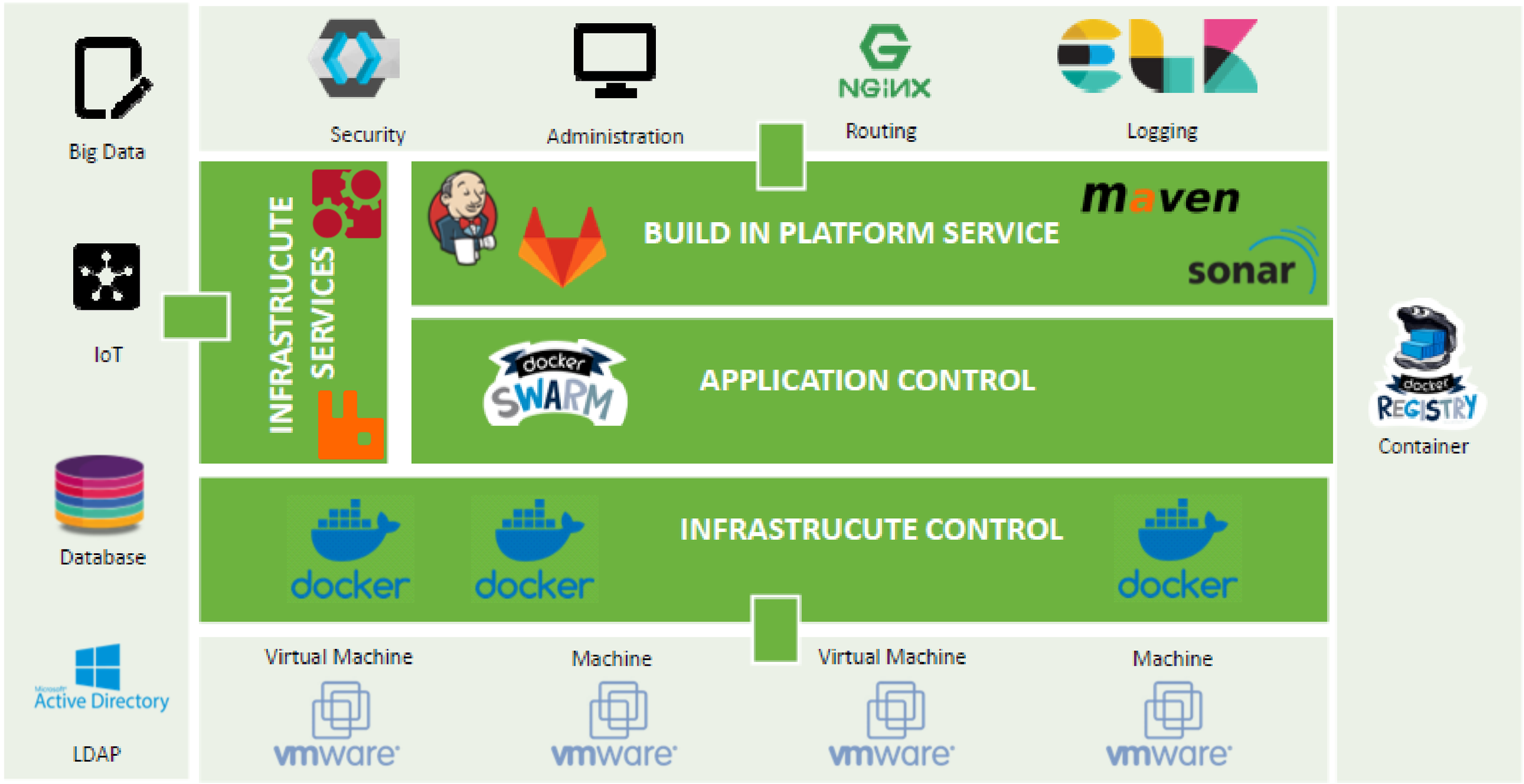
SonarQube

- Technischen Qualität von Sourcecode
- Analysiert den Sourcecode hinsichtlich verschiedener Qualitätsbereiche
- Unterstützung verschiedener Programmiersprachen
- Standardisierung gleiche Regeln für alle Entwickler.
- Hilfe und Sicherheit für Entwickler
- Robustere Anwendungen

Link - SonarQube

https://www.sonarqube.org/

Allgemein - PaaS



Agenda

Notwendigkeit
Warum eine Umstellung der Entwicklung?

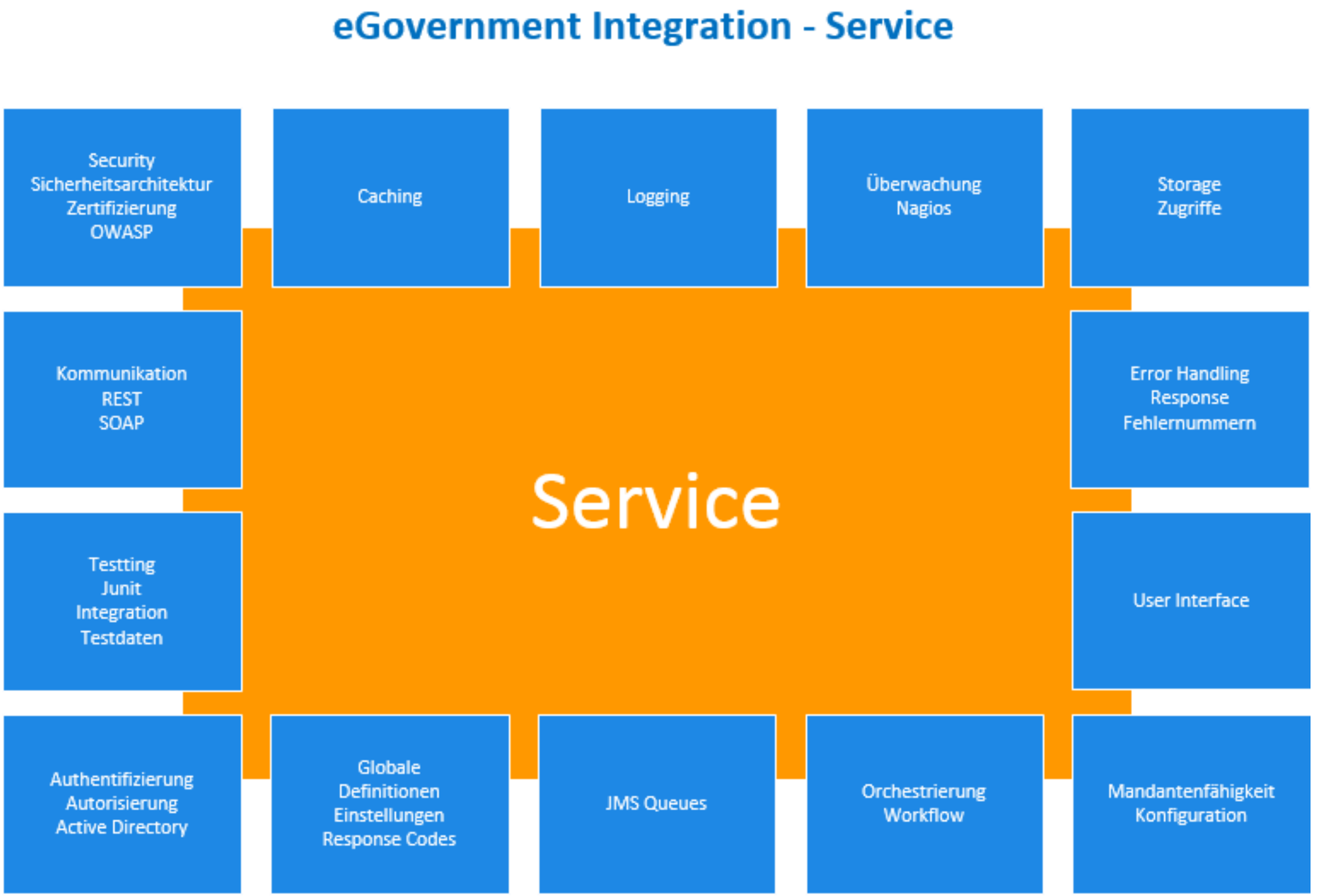
Infrastruktur
Welche Systeme setzen wir ein? Warum? Wie?

Architektur
Wie sind die Komponenten aufgebaut?

Mitarbeiter
Wie soll ein Entwickler arbeiten?

Fazit
Wo stehen wir? Was ändert sich?

Services


eGovernment Integration - Service
Security
Sicherheitsarchitektur
Zertifizierung
OWASP
Caching
Logging
Überwachung
Nagios
Storage
Zugriffe
Kommunikation
REST
SOAP
Error Handling
Response
Fehlernummern
Service
Testting
Junit
Integration
Testdaten
User Interface
Authentifizierung
Autorisierung
Active Directory
Globale
Definitionen
Einstellungen
Response Codes
JMS Queues
Orchestrierung
Workflow
Mandantenfähigkeit
Konfiguration

eGovernment-Integration

Services

- Datenübergabe Zeiterfassung
- Abrechnung Dienstreisekosten
- Gehaltsabrechnung Online
- Revisionssichere Archivierung
- Registrierung, Änderung
- Buchung von Rechnungen
- Geolocation (Standortbestimmung)
- Straßen Service (Wahlen)
- Formularserver (Schnittstelle CMS)
- usw.

Agenda

Notwendigkeit
Warum eine Umstellung der Entwicklung?

Infrastruktur
Welche Systeme setzen wir ein? Warum? Wie?

Architektur
Wie sind die Komponenten aufgebaut?

Mitarbeiter
Wie soll ein Entwickler arbeiten?

Fazit
Wo stehen wir? Was ändert sich?

eGovernment-Team

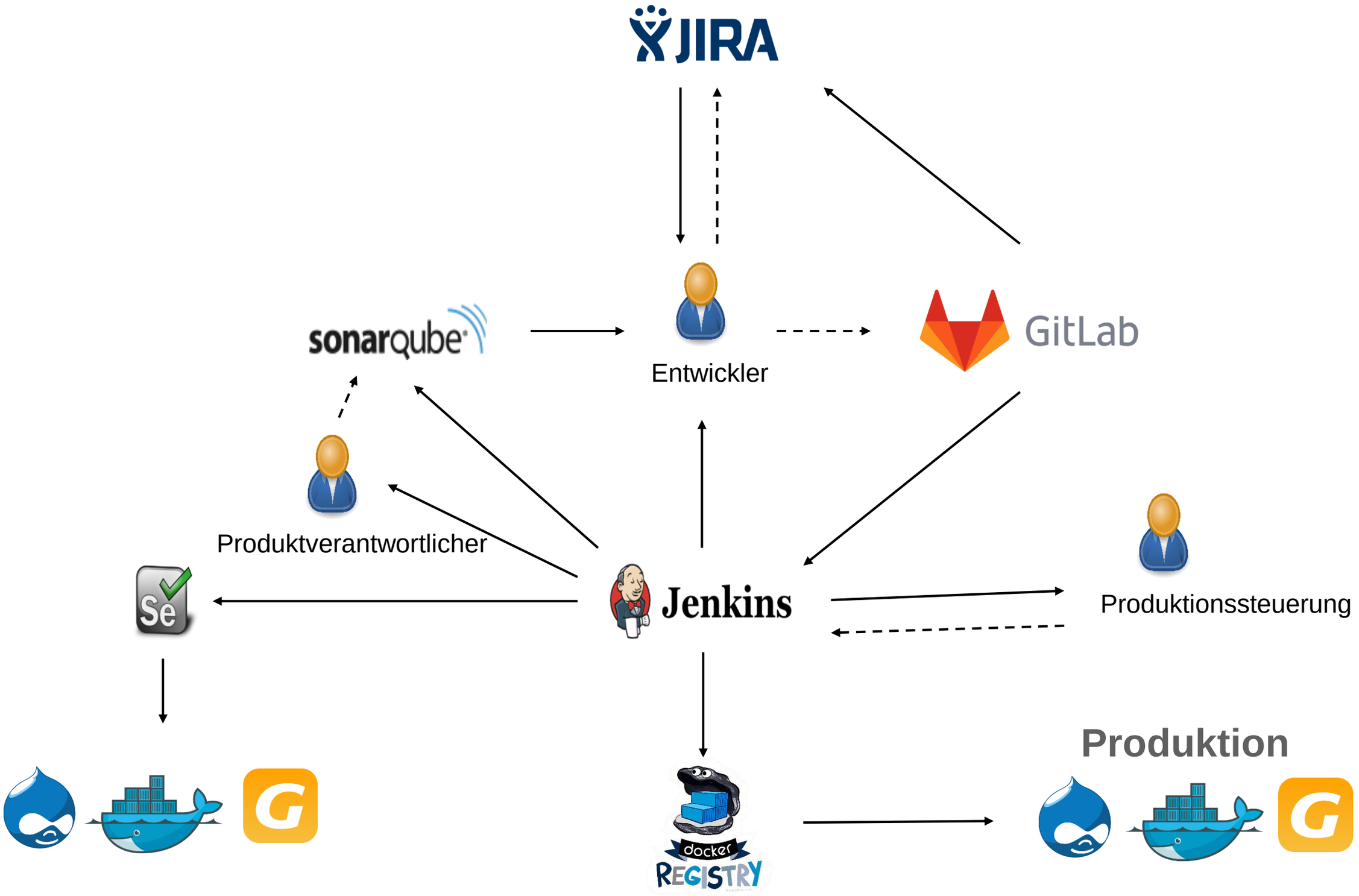
Neue Anforderungen

- Leitkriterien fordern uns auf anders zu arbeiten
- Lernen, lernen, lernen
- Neuer Entwicklertyp „DevOps“

Entwicklung ändert sich,

somit müssen wir uns auch ändern.

Arbeitsweise der Entwickler



eGovernment-Team

Unser Verständnis von Agilität

Um schnell reagieren zu können und in der Entwicklung aktuell zu bleiben, müssen:

Technische Voraussetzungen	Abläufe Prozesse	Entwicklung
• Automatisierung Zusammenspiel der Komponenten • Lose Koppelung Unabhängigkeit: Komponenten und Services (austauschbar) • Flexible Programmierung (Container)	• Im Entwicklerteam • Im KRZN (Abstimmung)	• Arbeitsweise (zentrale Entwicklungsabläufe) • Umgang mit Anforderungen (Aufteilung in Inkremente) • Meetingstrukturen X-Teams • Entwicklertyp Knowhow „DevOps“

Auswirkungen auf Leitkriterien - Wir möchten:

1 Klein
..kontinuierlich ausliefern können.

2 Schnell
..uns auf kleine Bereiche fokussieren können.

3 Vernetzt
..in X-Teams arbeiten.

4 Automatisiert
..dass alle Entwickler die gleichen zentralen Strukturen verwendet.

5 Robust
.. Stabilität durch QS Standards erreichen.

Agenda

Notwendigkeit
Warum eine Umstellung der Entwicklung?

Infrastruktur
Welche Systeme setzen wir ein? Warum? Wie?

Architektur
Wie sind die Komponenten aufgebaut?

Mitarbeiter
Wie soll ein Entwickler arbeiten?

Fazit
Wo stehen wir? Was ändert sich?

Was ändert sich konkret

1. Zuständigkeit	Systemtechnik: Installation von Linux und Docker Entwicklung: Zuständig für alle Tools, die für die Anwendung notwendig sind.
2. Aktualisierung	Systemtechnik: VM Ware, Betriebssystem, Docker Entwicklung ist zuständig für die Anwendung. (Für alles was im Container ist)
3. Server	Viele Server mit kleinen Minianwendungen nicht notwendig
4. Ausfallsicherheit	Keine Cluster für jede Minianwendung.
5. Skalierung	Über die Docker Infrastruktur
6. Verfügbarkeit	Docker/Swarm startet automatisiert den Container Neu.
7. Prozesse	Einheitliche und automatisierte Prozesse für die Produktion. Protokollierte Prozesse für die Zertifizierung.
8. Stabilität	Automatisierte Tests und Deployment.
9. Produktionssteuerung	Inbetriebnahme und Rollback der Services möglich.
10. Zyklen	Wir möchten kontinuierliche Integration.

Am Ende sind wir …

1 **Klein**
Weil wir Microservice-Strukturen verwenden.

2 **Schnell**
Weil wir auf Services und API Management setzen.

3 **Vernetzt**
Weil wir Kommunikation mit Services und Verfahren realisieren.

4 **Automatisiert**
Weil wir Build-Tools und definierte Abläufe haben.

5 **Robust**
Weil wir Punkt 1-4 beachten.

Vielen Dank

Christoph Thoma