Westfälische Wilhelms-Universität Münster

ERCIS
European Research Center for Information Systems

Neue Möglichkeiten der Virtualisierung für KMU

Herbert Kuchen, Vincent von Hof, Andreas Fuchs

Department of Information Systems, Westfälische Wilhelms-Universität Münster

Agenda

1. Motivation
2. Hardware- und Software-Virtualisierung
3. Zusammenfassung der Studie
4. Handlungsempfehlungen
   - Vorgehensmodell zur Einführung von Containern
   - Technologieauswahl: 10 Schritte zur Container Architektur
   - Überlegungen zum Virtualisierungs-Sicherheitslevel
5. Zusammenfassung

Container Virtualisierung

Große Konzerne bestimmen die Diskussion

NETFLIX

amazon.com

Google

facebook

ebay

OTTO

zalando

Spotify®

Container Virtualisierung

Große Konzerne bestimmen die Diskussion

NETFLIX amazon.com Google

facebook ebay

OTTO zalando Spotify®

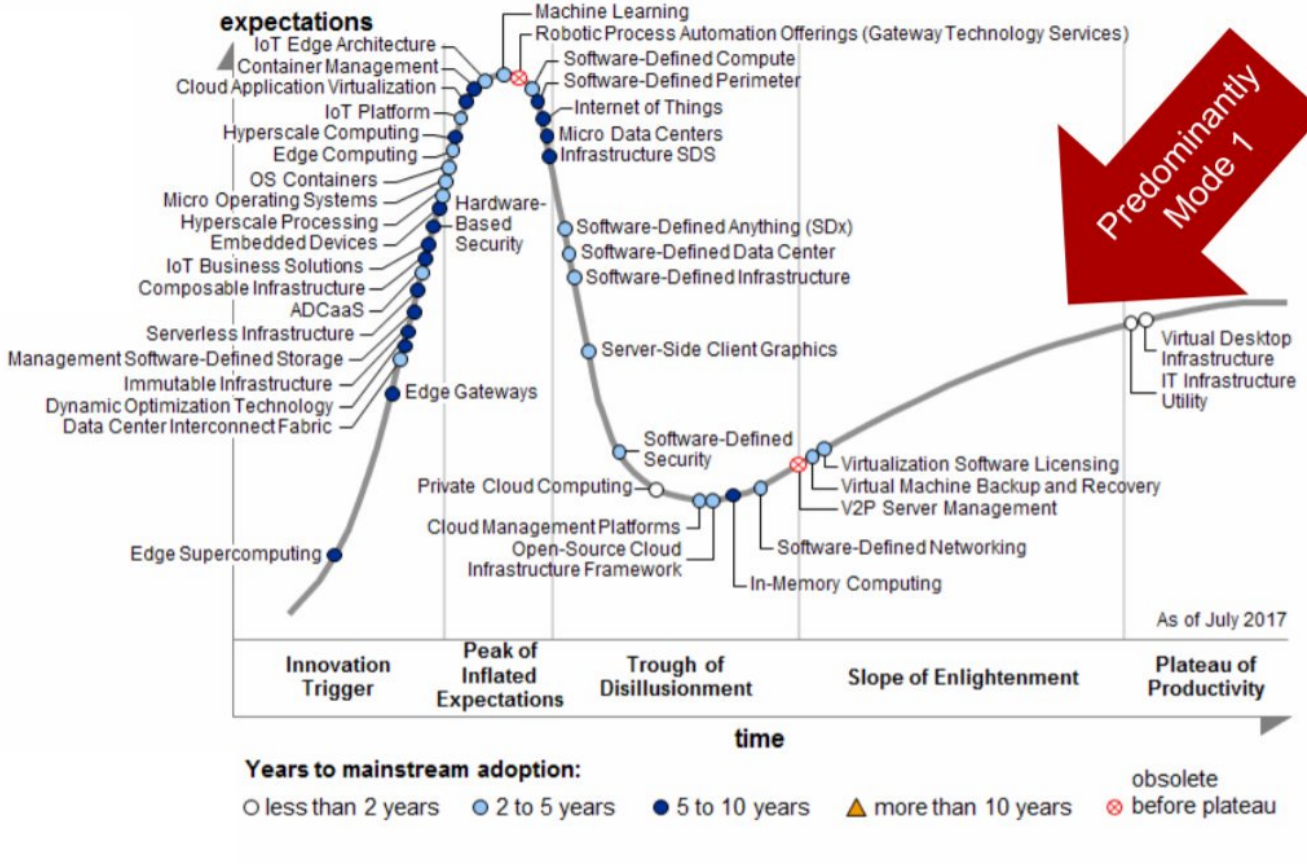
Was ist mit kleinen und mittleren Unternehmen (KMU)?

- 99.5% der Unternehmen in Deutschland sind KMU.
- 58% der Beschäftigten, 82% der Auszubildenden.
- 35% des Umsatzes wird in KMU erwirtschaftet.

(IfM Bonn, 2016)

ERCIS

Infrastructure Strategies' Hype Cycle 2017



Gartner

Container

Wo ist das Problem?

Gartner

“Viele Unternehmen [...] entsetzt (sind), wenn sie ihre ersten Cloud-Rechnungen bekommen, da diese weit höher sind als veranschlagt”

(Gartner, 2018)

- Nutzen von Skalierbarkeit hängt stark vom Business Modell ab.
- Sicherheit und Datenschutz Bedenken.
- Fachkräftemangel.

Container

Wo ist das Problem?

Gartner

“Viele Unternehmen [...] entsetzt (sind), wenn sie ihre ersten Cloud-Rechnungen bekommen, da diese weit höher sind als veranschlagt”

(Gartner, 2018)

- Nutzen von Skalierbarkeit hängt stark vom Business Modell ab.
- Sicherheit und Datenschutz Bedenken.
- Fachkräftemangel.

Container

- Hohe Geschwindigkeit der Entwicklung & Support Dauer unklar.
- Vorteil von Containern abseits der Cloud unklar.
  - DevOps war für viele Firmen schwer umsetzbar.

Agenda

1. Motivation
2. Hardware- und Software-Virtualisierung
3. Zusammenfassung der Studie
4. Handlungsempfehlungen
   - Vorgehensmodell zur Einführung von Containern
   - Technologieauswahl: 10 Schritte zur Container Architektur
   - Überlegungen zum Virtualisierungs-Sicherheitslevel
5. Zusammenfassung

Virtualisierung

- Erzeugung von *n* virtuellen Ressource aus *m* physischen Ressourcen.
  - z.B. mehrere virtuelle Prozessoren durch time sharing.
  - Mehrere Festplatten durch Partitioning.
  - Arbeitsspeicher in Memory Pools.
  - Emulation anderer Systeme.
- Abstrakte Ebene zur Entkoppelung von physischer Hardware.

Virtualisierung und Container

Definition

Virtualisierung

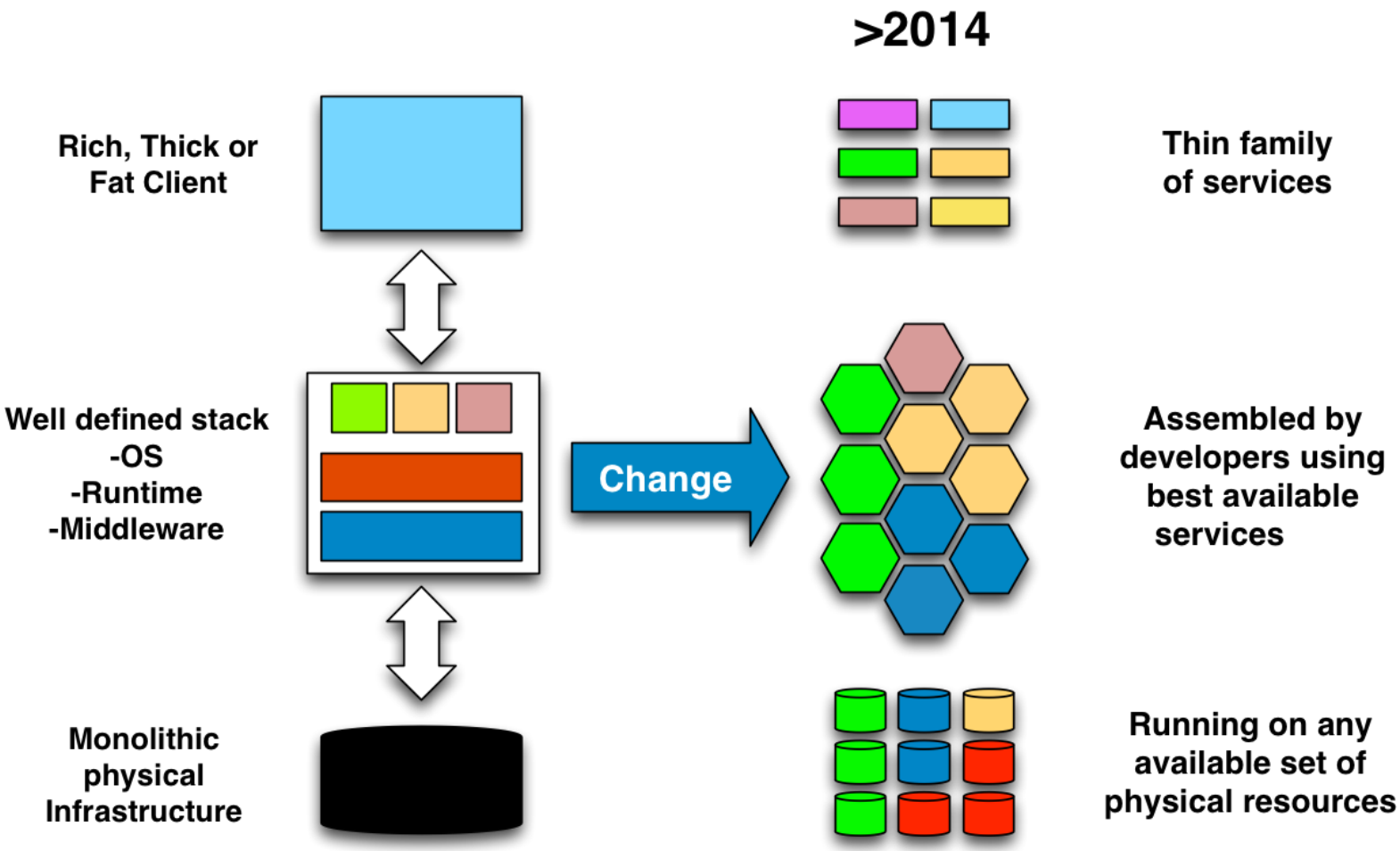
- Erzeugung von *n* virtuellen Ressource aus *m* physischen Ressourcen.
  - z.B. mehrere virtuelle Prozessoren durch time sharing.
  - Mehrere Festplatten durch Partitioning.
  - Arbeitsspeicher in Memory Pools.
  - Emulation anderer Systeme.
- Abstrakte Ebene zur Entkoppelung von physischer Hardware.

Container

- Architektur Entwurfsmuster
- Anwendung ist in kleine, lose gekoppelte Services aufgeteilt.
- Ziele: Resilience, Skalierbarkeit, Wartbarkeit, Modularität.

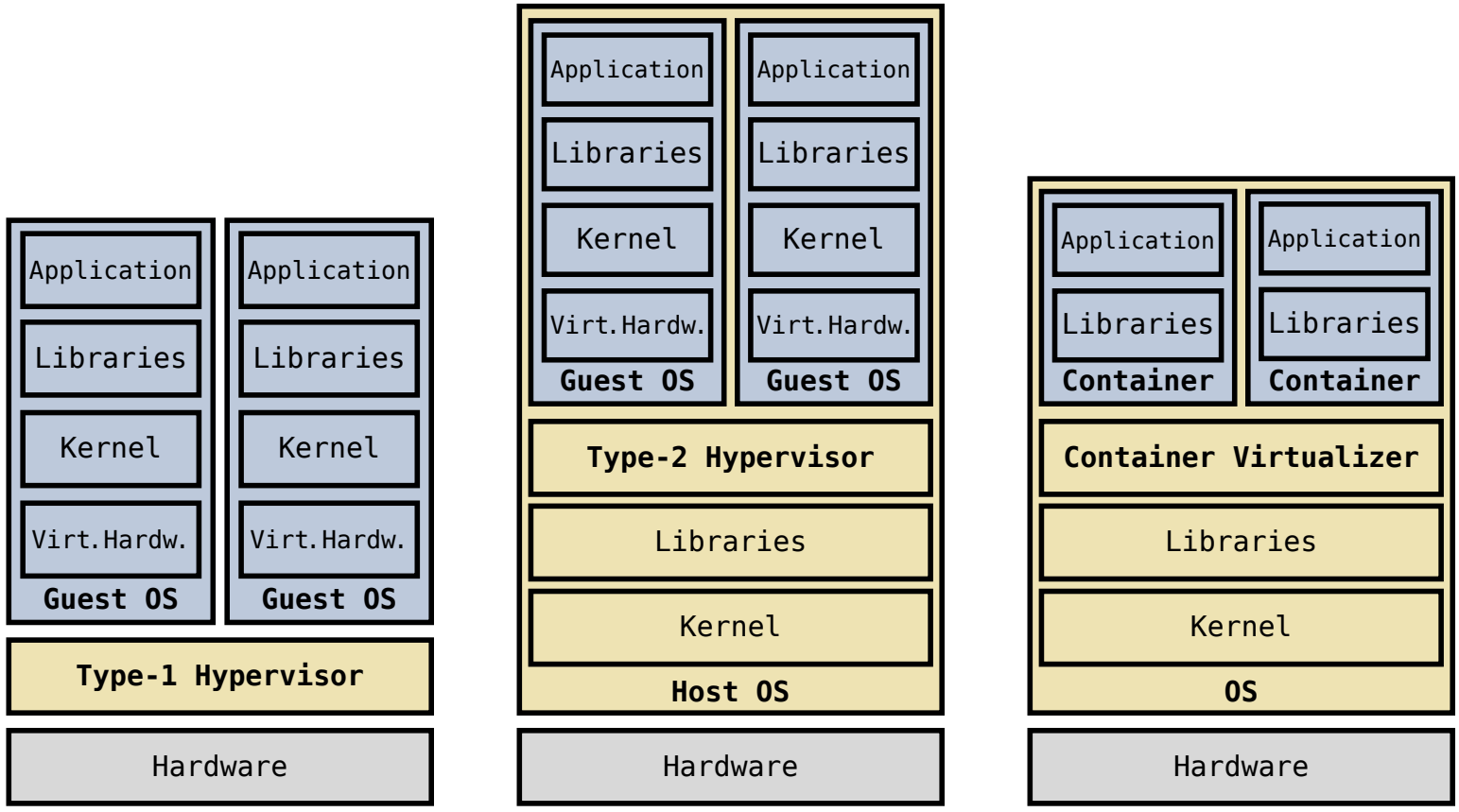
Container

Container Evolution



Container

Virtualisierungsansätze im Vergleich



Container

Virtualisierungsansätze im Vergleich

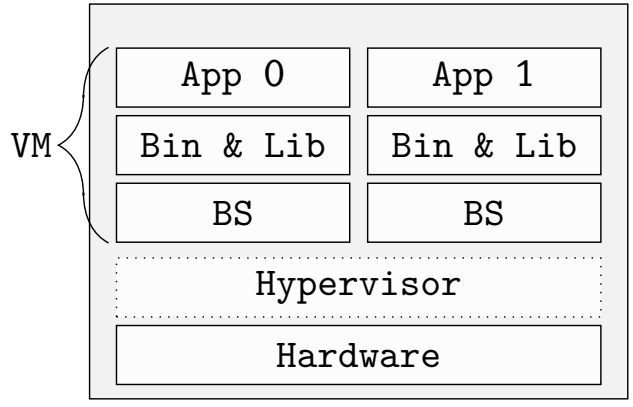
- Virtualisierte Ressource: CPU, RAM, Speicher, Netzwerk ...

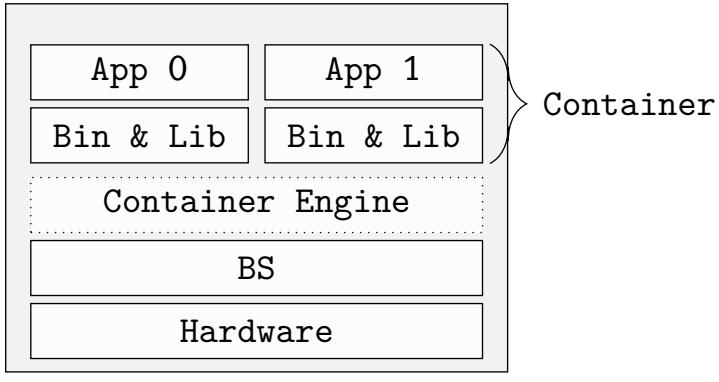
Hypervisor

- Hohes Level an isolation.
- Performance Overhead und Speicherverbrauch.

Container Virtualisierung

- Geringere, aber meist ausreichendes Level an isolation.
- Geringerer Overhead.





Microservice

- Erledigt eine Aufgabe.

Microservice

- Erledigt eine Aufgabe.
- In ≤ 1 Monat implementiert.

Microservice

- Erledigt eine Aufgabe.
- In ≤ 1 Monat implementiert.
- Über ein Netzwerk verknüpft.

Microservice

- Erledigt eine Aufgabe.
- In ≤ 1 Monat implementiert.
- Über ein Netzwerk verknüpft.
- Sprach-unabhängiges Interface (oft REST).

Microservice

- Erledigt eine Aufgabe.
- In ≤ 1 Monat implementiert.
- Über ein Netzwerk verknüpft.
- Sprach-unabhängiges Interface (oft REST).
- Einfach austauschbar (sogar im Betrieb).

Microservice

- Erledigt eine Aufgabe.
- In ≤ 1 Monat implementiert.
- Über ein Netzwerk verknüpft.
- Sprach-unabhängiges Interface (oft REST).
- Einfach austauschbar (sogar im Betrieb).
- Meinst als Container (Docker) oder VM ausgeliefert.

Microservice

- Erledigt eine Aufgabe.
- In ≤ 1 Monat implementiert.
- Über ein Netzwerk verknüpft.
- Sprach-unabhängiges Interface (oft REST).
- Einfach austauschbar (sogar im Betrieb).
- Meinst als Container (Docker) oder VM ausgeliefert.
- Meist zusammen mit logging und monitoring eingesetzt.

Microservice

- Erledigt eine Aufgabe.
- In ≤ 1 Monat implementiert.
- Über ein Netzwerk verknüpft.
- Sprach-unabhängiges Interface (oft REST).
- Einfach austauschbar (sogar im Betrieb).
- Meinst als Container (Docker) oder VM ausgeliefert.
- Meist zusammen mit logging und monitoring eingesetzt.
- Idealerweise stateless.

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.
  - Erfordert Implementierung, entsprechend werden Abhängigkeiten nicht zufällig hinzugefügt.

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.
  - Erfordert Implementierung, entsprechend werden Abhängigkeiten nicht zufällig hinzugefügt.
- Wenn eine Instanz ausfällt, sind andere davon nicht betroffen (→ resilience!).

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.
  - Erfordert Implementierung, entsprechend werden Abhängigkeiten nicht zufällig hinzugefügt.
- Wenn eine Instanz ausfällt, sind andere davon nicht betroffen ($\rightarrow$ resilience!).
- Microservices können unabhängig voneinander von verschieden Teams entwickelt werden.

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.
  - Erfordert Implementierung, entsprechend werden Abhängigkeiten nicht zufällig hinzugefügt.
- Wenn eine Instanz ausfällt, sind andere davon nicht betroffen ($\rightarrow$ resilience!).
- Microservices können unabhängig voneinander von verschieden Teams entwickelt werden.
- Microservices können unabhängig voneinander skaliert werden.

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.
  - Erfordert Implementierung, entsprechend werden Abhängigkeiten nicht zufällig hinzugefügt.
- Wenn eine Instanz ausfällt, sind andere davon nicht betroffen ($\rightarrow$ resilience!).
- Microservices können unabhängig voneinander von verschieden Teams entwickelt werden.
- Microservices können unabhängig voneinander skaliert werden.
- Jeder Service kann mit der passendsten Sprache implementiert werden.

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.
  - Erfordert Implementierung, entsprechend werden Abhängigkeiten nicht zufällig hinzugefügt.
- Wenn eine Instanz ausfällt, sind andere davon nicht betroffen ($\rightarrow$ resilience!).
- Microservices können unabhängig voneinander von verschieden Teams entwickelt werden.
- Microservices können unabhängig voneinander skaliert werden.
- Jeder Service kann mit der passendsten Sprache implementiert werden.
- Gradueller Wechsel auf neue Technologien möglich.

Vorteile von Microservices

- Stabile, reproduzierbare Umgebung
  - Neue Abhängigkeiten benötigen neue Services.
  - Erfordert Implementierung, entsprechend werden Abhängigkeiten nicht zufällig hinzugefügt.
- Wenn eine Instanz ausfällt, sind andere davon nicht betroffen ($\rightarrow$ resilience!).
- Microservices können unabhängig voneinander von verschieden Teams entwickelt werden.
- Microservices können unabhängig voneinander skaliert werden.
- Jeder Service kann mit der passendsten Sprache implementiert werden.
- Gradueller Wechsel auf neue Technologien möglich.
- Continous refactoring, delivery, und deployment möglich (DevOps).

Nachteile von Microservices

- Kommunikations-overhead.

Nachteile von Microservices

- Kommunikations-overhead.
- Zusätzliche Komplexität eines verteilten Systems:

Nachteile von Microservices

- Kommunikations-overhead.
- Zusätzliche Komplexität eines verteilten Systems:
  - Testing, Auslieferung, logging, monitoring.

Nachteile von Microservices

- Kommunikations-overhead.
- Zusätzliche Komplexität eines verteilten Systems:
  - Testing, Auslieferung, logging, monitoring.
- Komplexe Migration von monolithischen Systemen.

Nachteile von Microservices

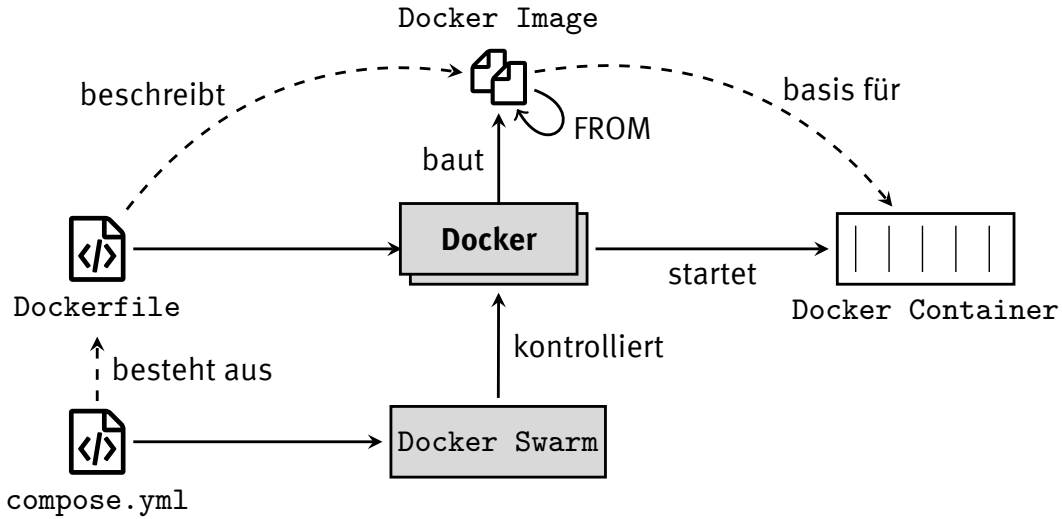
- Kommunikations-overhead.
- Zusätzliche Komplexität eines verteilten Systems:
  - Testing, Auslieferung, logging, monitoring.
- Komplexe Migration von monolithischen Systemen.
- Nanoservice anti-pattern: Kommunikations-overhead › Nutzen.

Docker

- Der erfolgreichste Container-Virtualisierungs Ansatz.
- Basiert auf Linux; insbesondere:
  - namespaces: bieten isolation
    (des z.B. Dateisystems, Netzwerks, users, groups, process lists, ...)
  - control groups (cgroups): bieten gemeinsame Ressource Nutzung
    (z.B. Festplatten quotas)

Docker

Docker Begrifflichkeiten



- image: unveränderlicher Dateisystem-Zustand.
- layer:
  - *images* sind eine Sammlung von *layers*.
  - ein layer stellt eine Menge von Änderungen am Dateisystem dar.
- dockerfile: Eine Reihenfolge von layers $\rightarrow$ image.
- registry: Speichert lokale und globale images.
- container: Laufzeitinstanz eines images.

Docker

Dockerfile Beispiel


```
1  # Benutze jboss/base-jdk:8 als Basis image
2  FROM jboss/base-jdk:8
3
4  # Setze unsere Umgebungsvariable
5  ENV JBOSS_HOME /opt/jboss/wildfly
6
7  # Lade Wildfly
8  RUN curl -O https://download.jboss.org/wildfly/$WILDFLY_VERSION/wildfly-$WILDFLY_VERSION
       .tar.gz
9  ... # unpack and install
10
11 # Spezifizere, welcher Port intern für den Service genutzt wird.
12 EXPOSE 8080
13
14 # Spezifiziere das Startkommando
15 CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0"]
```

Docker

Docker Compose

- Compose erlaubt es mehrere Services in eine Anwendungsumgebung zu kombinieren.
- Die 3 Compose Schritte:
  1. Definiere die Services durch Dockerfiles.
  2. Definiere die Interfaces der Services, sodass sie untereinander kommunizieren können.
  3. Starte mit docker-compose up

Docker

Docker Compose

- Compose erlaubt es mehrere Services in eine Anwendungsumgebung zu kombinieren.
- Die 3 Compose Schritte:
  1. Definiere die Services durch Dockerfiles.
  2. Definiere die Interfaces der Services, sodass sie untereinander kommunizieren können.
  3. Starte mit docker-compose up
- Skaliere Service mit docker-compose scale ‹service-name›=n

Docker

Docker Compose Beispiel


```
1    version: '2'
2    services:
3      web:
4        build: ./Dockerfiles/wildfly/.
5        depends_on:
6         - db
7      db:
8        build: ./Dockerfiles/mysql/.
```

- Es ist möglich image: ‹image› wiederzuverwenden oder images neu zu konstruieren durch build: ‹location›
- depends_on: - ‹service› bestimmt die Startreihenfolge.

Docker

Docker Compose Beispiel


```
1   version: '2'
2   services:
3     web:
4       build: ./Dockerfiles/wildfly/.
5       ports:
6        - "80:8080"
7       depends_on: -  db
8
9     db:
10      build: ./Dockerfiles/mysql/.
11      ports:
12        - "3307:3306"
```

- ports bindet container ports an ports der Netzwerks.

Docker

Docker Compose Beispiel


```
1   version: '2'
2   services:
3     web:
4       build: ./Dockerfiles/wildfly/.
5       ports: - 8080
6       depends_on: -  db
7     db:
8       build: ./Dockerfiles/mysql/.
9
10    loadbalancer:
11      image: dockercloud/haproxy
12      ports:
13       - 80:80
14      links:
15       - web
16      volumes:
17       - /var/run/docker.sock:/var/run/docker.sock
```

- loadbalancer

Docker Compose Networks


```
1   version: '2'
2   services:
3   webfrontend: #...
4     networks:
5      - front-tier
6   businesslogic: #...
7     networks:
8      - front-tier
9      - back-tier
10  database: #...
11    networks:
12     - back-tier
```

```
1   networks:
2     front-tier: driver: bridge
3     back-tier: driver: bridge
```

- networks: virtuelle Netzwerke $\rightarrow$ Netzwerk-Level Isolation.

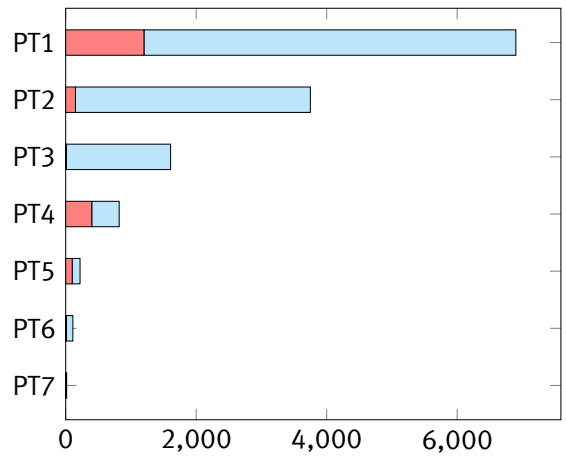
Agenda

1. Motivation
2. Hardware- und Software-Virtualisierung
3. Zusammenfassung der Studie
4. Handlungsempfehlungen
   - Vorgehensmodell zur Einführung von Containern
   - Technologieauswahl: 10 Schritte zur Container Architektur
   - Überlegungen zum Virtualisierungs-Sicherheitslevel
5. Zusammenfassung

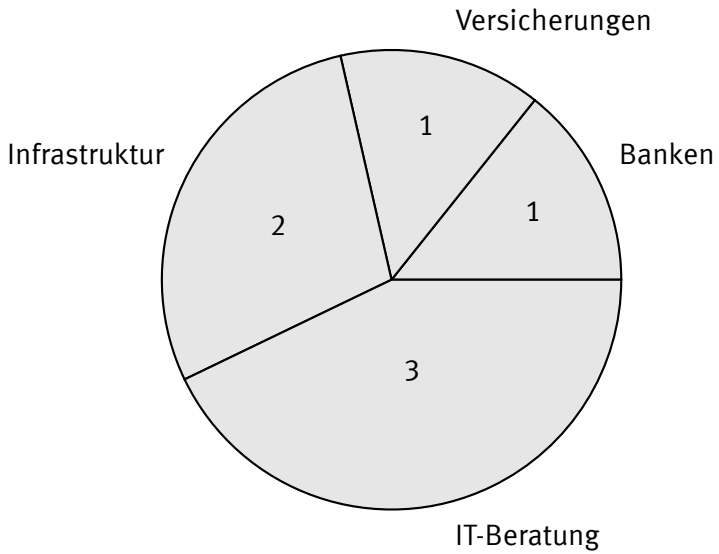
Studie

Datenerhebung bei IAI Mitgliedern

- Einladung an alle 30 Mitglieder des IAI versendet
- sieben Mitglieder sind Einladung gefolgt
  $\rightarrow$ Teilnahmequote von 23%
- Projektteilnehmer waren heterogene Gruppe



Anzahl der Mitarbeiter in den Unternehmen der Projektteilnehmer.



Verteilung der Branchen der Projektteilnehmer.

Studie

Untersuchungsmethode

- **Interviews** mit IT-Verantwortlichen der KMUs
- Interviews haben zwei wesentliche Prinzipien:
  - *Prinzip der Offenheit*
  - *Prinzip der Kommunikation*
- Entwicklung eines **Fragebogens** als Leitpfaden für die Interviews

Studie

Untersuchungsmethode

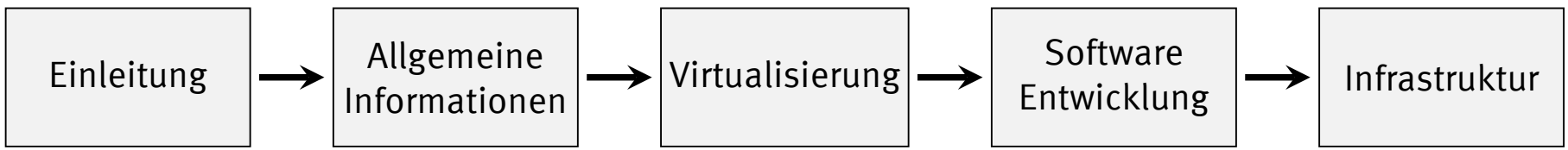
- **Interviews** mit IT-Verantwortlichen der KMUs
- Interviews haben zwei wesentliche Prinzipien:
  - *Prinzip der Offenheit*
  - *Prinzip der Kommunikation*
- Entwicklung eines **Fragebogens** als Leitpfaden für die Interviews

Forschungsfragen:

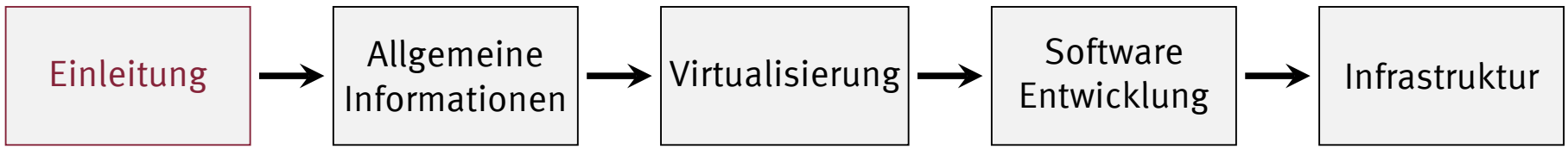
- **FF1:** Wie ist der aktuelle Stand der Einführung von Software-Virtualisierungen in KMUs?
- **FF2:** Was sind generelle Hindernisse bei der Einführung von Container-basierten Virtualisierungslösungen?

Problemzentriertes Interview im Projekt

- Interviews erlauben eine offene Interaktion.
- Problemzentriertes Interview wird auf Grundlage eines Leitfadens durchgeführt.

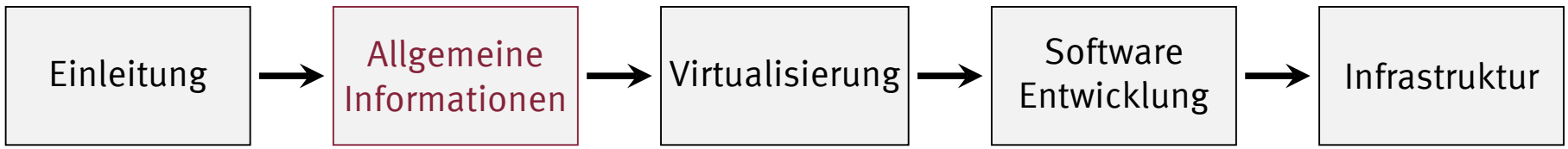


Problemzentriertes Interview im Projekt



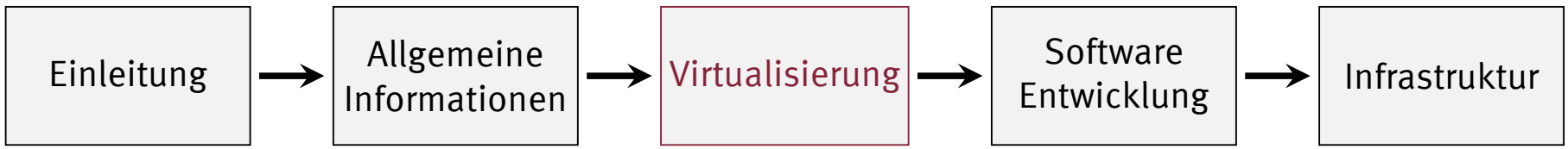
- Über Ziele des Interviews informieren.
- Aufzeichnung des Interviews mit Einverständnis des Interviewten.
- Anschließend: transkribieren der Aufzeichnungen für weitere Auswertung.

Problemzentriertes Interview im Projekt



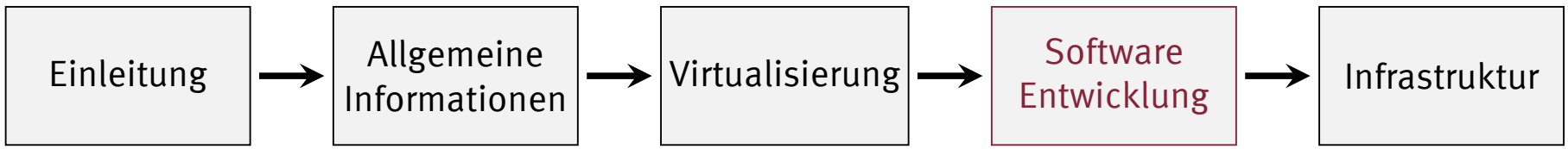
- Projektteilnehmer haben tendenziell Interesse an Virtualisierung.
- Allgemeine Informationen über Unternehmen identifizieren.
- Überblick über Projektpartner erhalten.
- Ist Virtualisierung nur für bestimmte KMU interessant?

Problemzentriertes Interview im Projekt



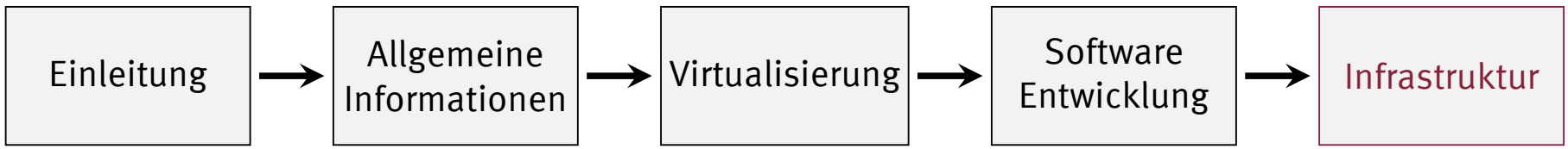
- Identifizierung von …
  - …Software-Artefakten, die bereits virtualisiert werden
  - …und Technologien, die dafür verwendet werden.
- Welche Herausforderungen, Probleme und Chancen sehen die IT-Verantwortlichen durch den Einsatz von Software-Virtualisierung (und im Speziellen Container-basierter Virtualisierung).

Problemzentriertes Interview im Projekt



- In wie weit spielt Virtualisierung bei der Software-Entwicklung im Unternehmen eine Rolle?
- Alternativer Blick auf Software-Virtualisierung.

Problemzentriertes Interview im Projekt



- Welche Komponenten sind Teil der technischen Infrastruktur?
  (Was setzt das Unternehmen ein? DB-Server, Web-Server, etc.)
- Gibt es ein Transformationsschema für Service-Virtualisierung im Unternehmen?
- Einschätzung der Chancen von Container-basierter Virtualisierung ermitteln – im Hinblick auf Skalierbarkeit und Ausfallsicherheit.

Kernbereiche der Studie

Gesprächsleitfaden hat drei Kernbereiche:

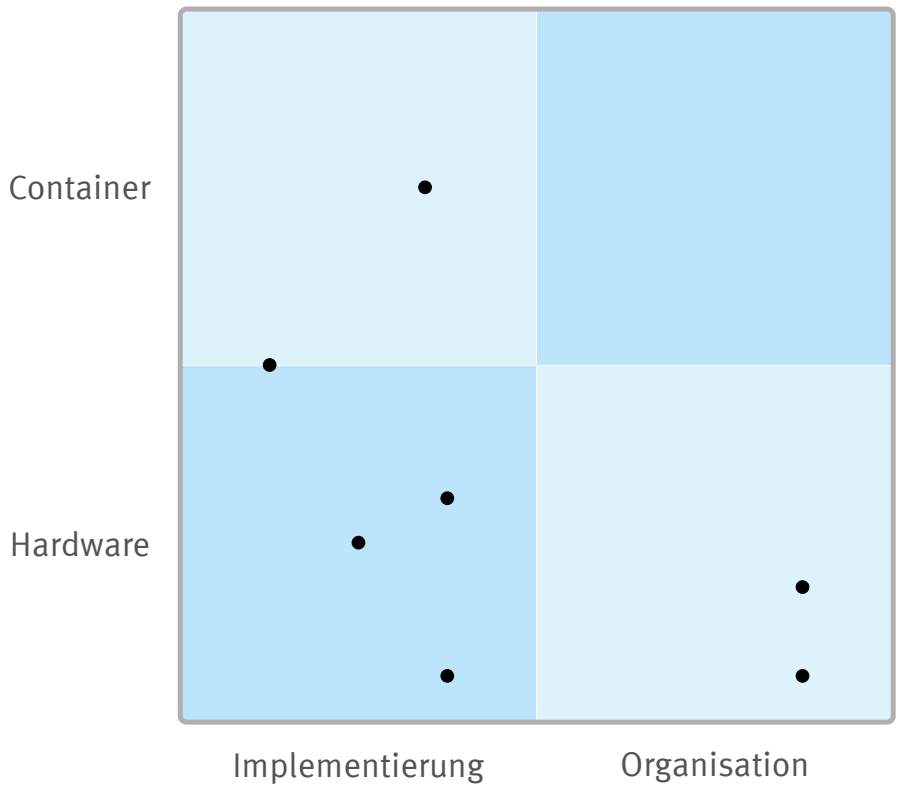
1. Allgemeine Informationen zum Unternehmen der Projektpartner identifizieren.
2. Kenntnisstand der Projektpartner im Bereich Software-Virtualisierung ermitteln.
3. Lösungsansätze und Vorgehensmodelle zur Einführung von Software-Virtualisierung im Unternehmen aufzeigen.

Ergebnisse der Studie.

(Auszug – gesamte Studie im Abschlussbericht vorhanden.)

Ausrichtung der Interviewpartner

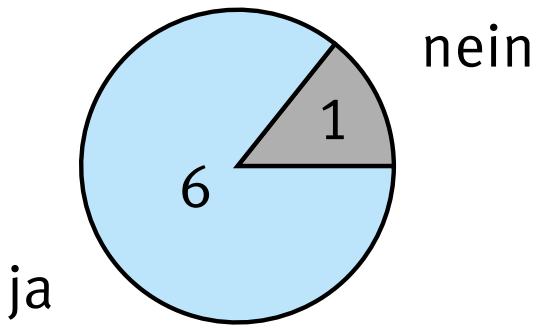
Wie schauen die Projektteilnehmer auf das Thema Virtualisierung?



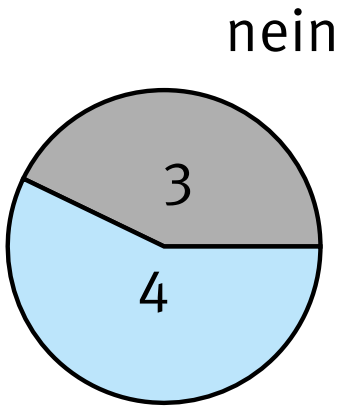
- Großteil der Teilnehmer ist aus technischer Sicht in das Thema Virtualisierung involviert.

Virtualisierung im Unternehmen

Wie verbreitet ist Virtualisierung im Unternehmen?



(a) Server Virtualisierung.

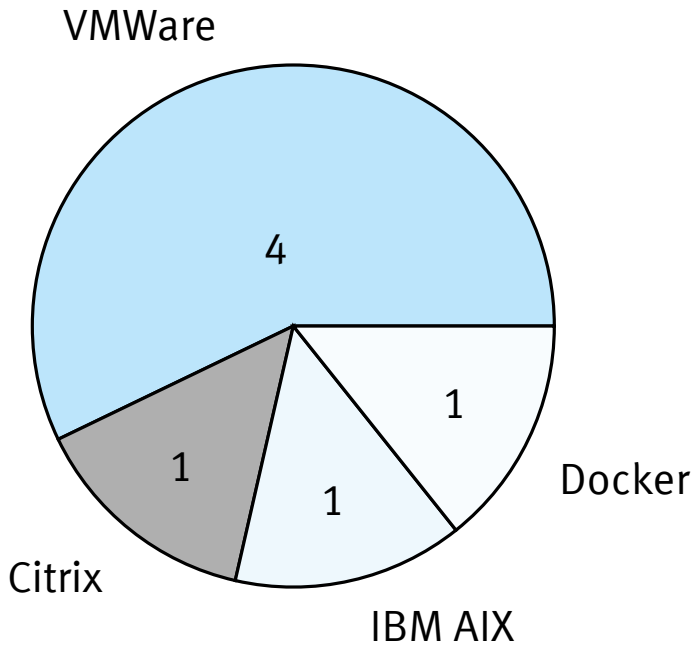


(b) Client Virtualisierung.

- 86% der KMU setzt auf Server-Virtualisierung (VMWare, etc.)
- 58% der KMU setzt auf Client-Virtualisierung (Citrix, etc.)

Eingesetzte Virtualisierungs-Technologien

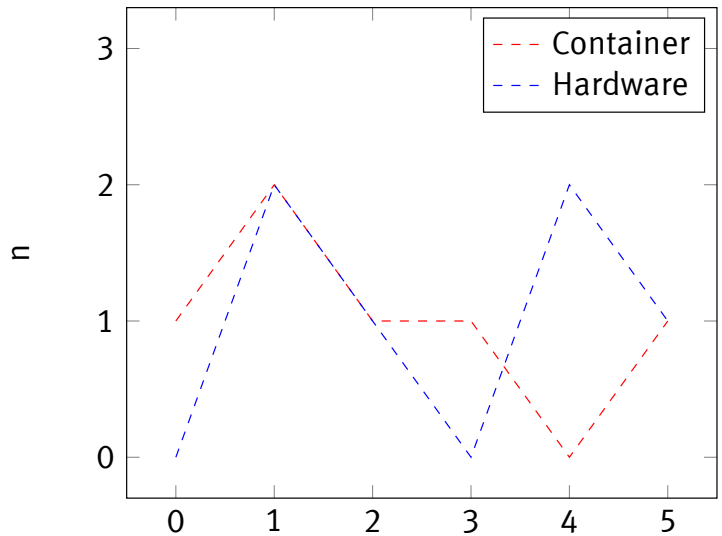
Welche Technologien werden primär eingesetzt?



- Großteil setzt VMWare als Virtualisierungs-Technologie ein.
- IBM AIX soll – mit Hilfe eines Transformationsprozesses – durch VMWare ersetzt werden.
- Nur ein Unternehmen besitzt ein Transformationsplan für Container-basierte Virtualisierung.

Technisches Wissen der IT-Verantwortlichen

Wie vertraut sind die IT-Verantwortlichen mit Hardware- und Container-basierter Virtualisierung?

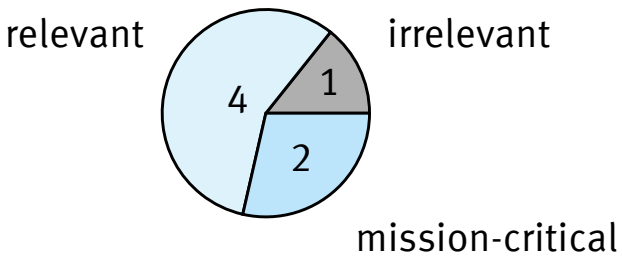


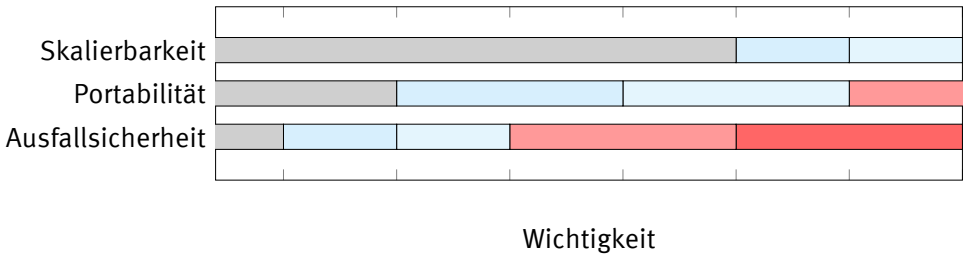
Skala: 0=wenig Wissen, 5=ausführliches Wissen

- Technisches Detailwissen weniger ausgeprägt, da nicht sehr relevant für das Alltagsgeschäft.
- Wenig Experten für Container-basierte Virtualisierung.
- Einige Experten für Hardware-basierte Virtualisierung.
  (Klassische Virtualisierung wird oft produktiv eingesetzt)

Wichtigkeit von Virtualisierung in den KMU

Welche Vor- und Nachteile werden von den IT-Verantwortlichen wahrgenommen?

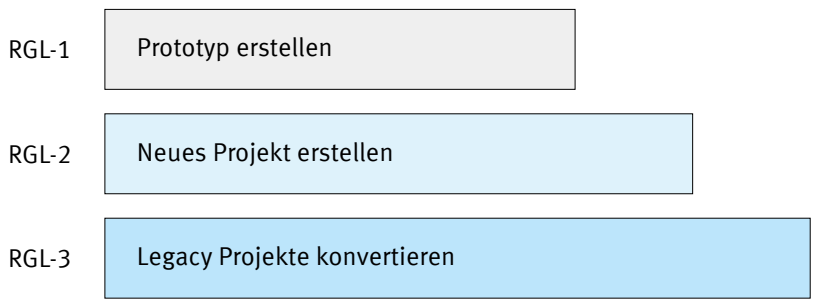


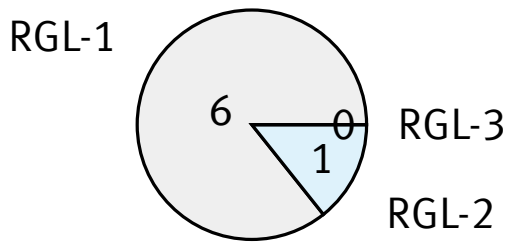


- Alle befragten Unternehmen setzten entweder Hardware- und/oder Software-Virtualisierung ein.
- Genannten Vorteile: Ausfallsicherheit, Portabilität und Skalierbarkeit.
- Portabilität: Fähigkeit, logische Systeme zwischen physischen Systeme bewegen zu können verbessert die Wartung der Hardware.
- Ausfallsicherheit: setzt häufig Portabilität voraus.
- Skalierbarkeit: spielt für die meisten Projektteilnehmer eine untergeordnete Rolle.

Reifegrade Level (RGL) der Service-Virtualisierung in KMU

Ist-Zustand von Container-basierter Virtualisierung in Reifegraden.



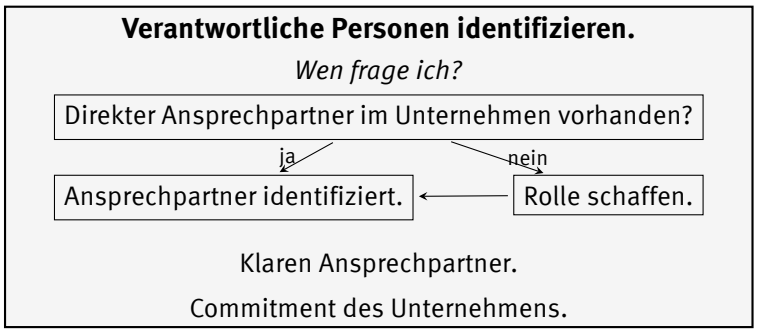


- RGL-1 Prototyp erstellen:
  - Sammlung von Erfahrungen mit (neuen) Virtualisierungstechniken.
  - Erfahrung soll in zukünftige Entscheidungen einfließen.
    (z.B. Zeitaufwand für Einarbeitung eines Mitarbeiters in die Technik)
- RGL-2 Neues Projekt von Beginn an mit Mirco-Service-Design erstellen.
- RGL-3 Konvertierung bestehender Projekte, bei zeitgleicher Beibehaltung der bestehenden Funktionalität.

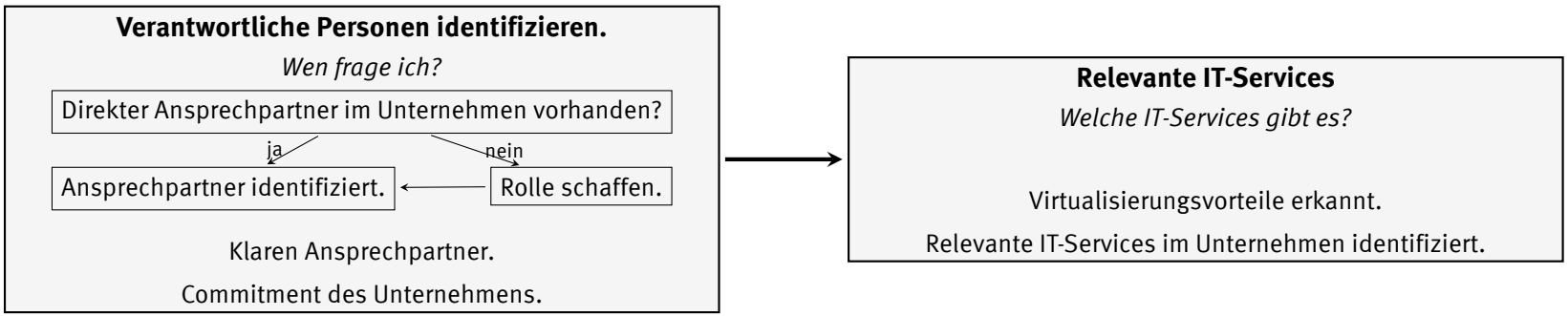
Agenda

1. Motivation
2. Hardware- und Software-Virtualisierung
3. Zusammenfassung der Studie
4. Handlungsempfehlungen
   - Vorgehensmodell zur Einführung von Containern
   - Technologieauswahl: 10 Schritte zur Container Architektur
   - Überlegungen zum Virtualisierungs-Sicherheitslevel
5. Zusammenfassung

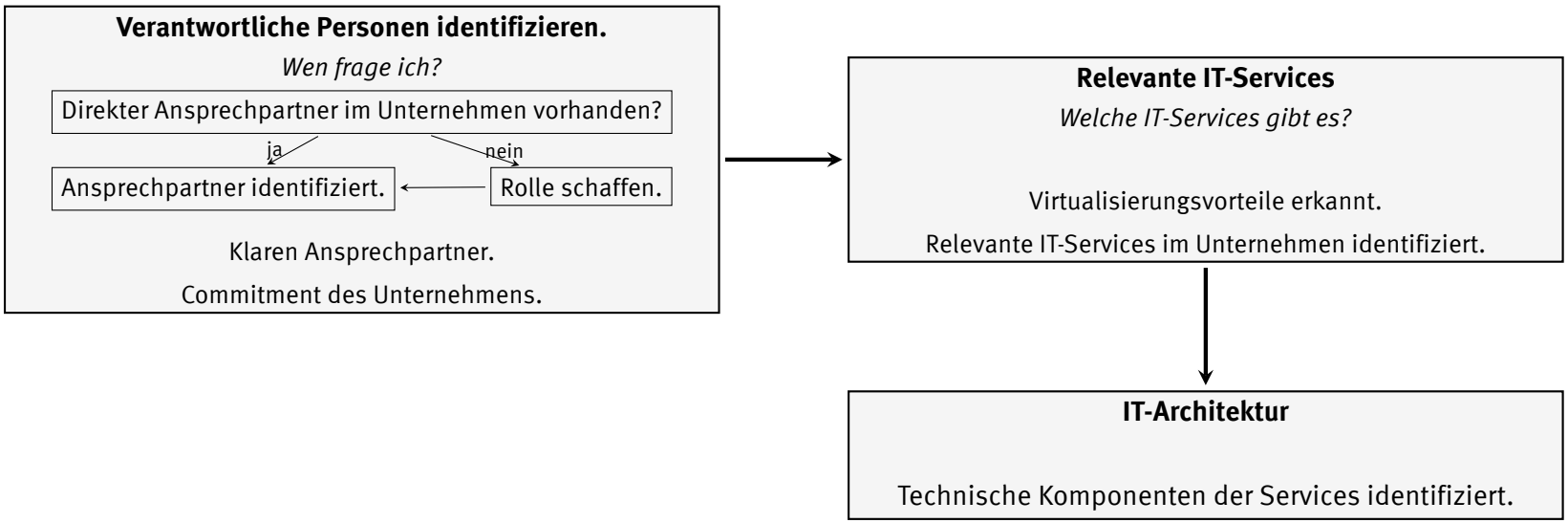
Vorgehensmodell zur Einführung von Virtualisierung.



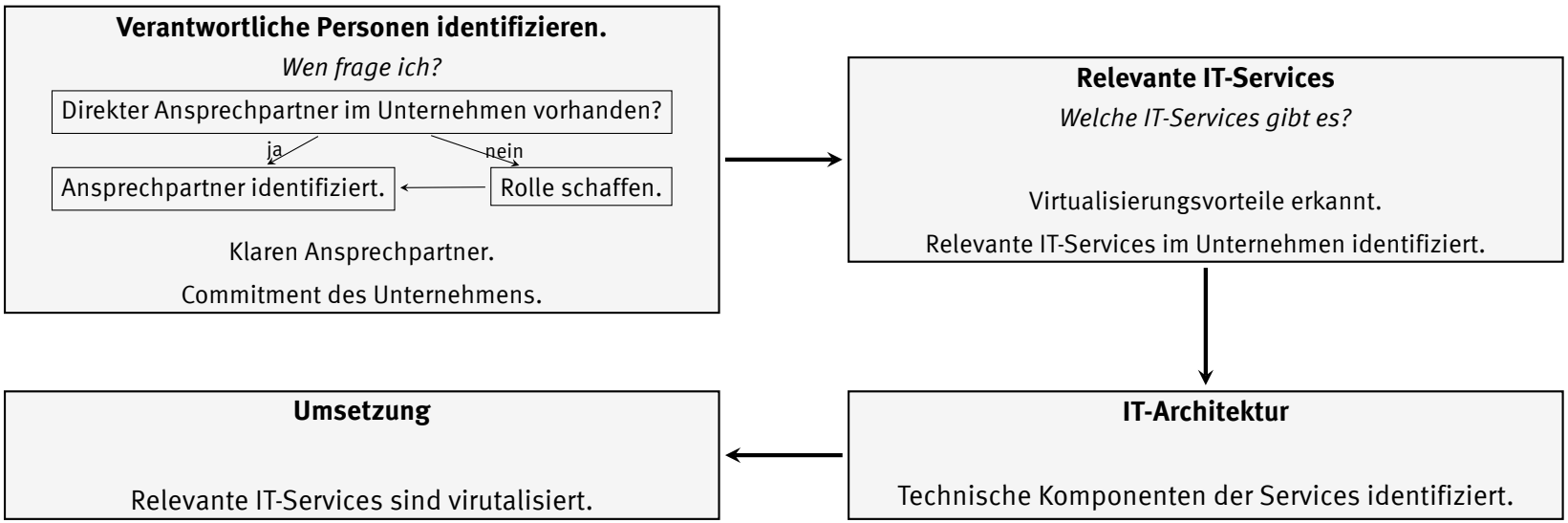
Vorgehensmodell zur Einführung von Virtualisierung.



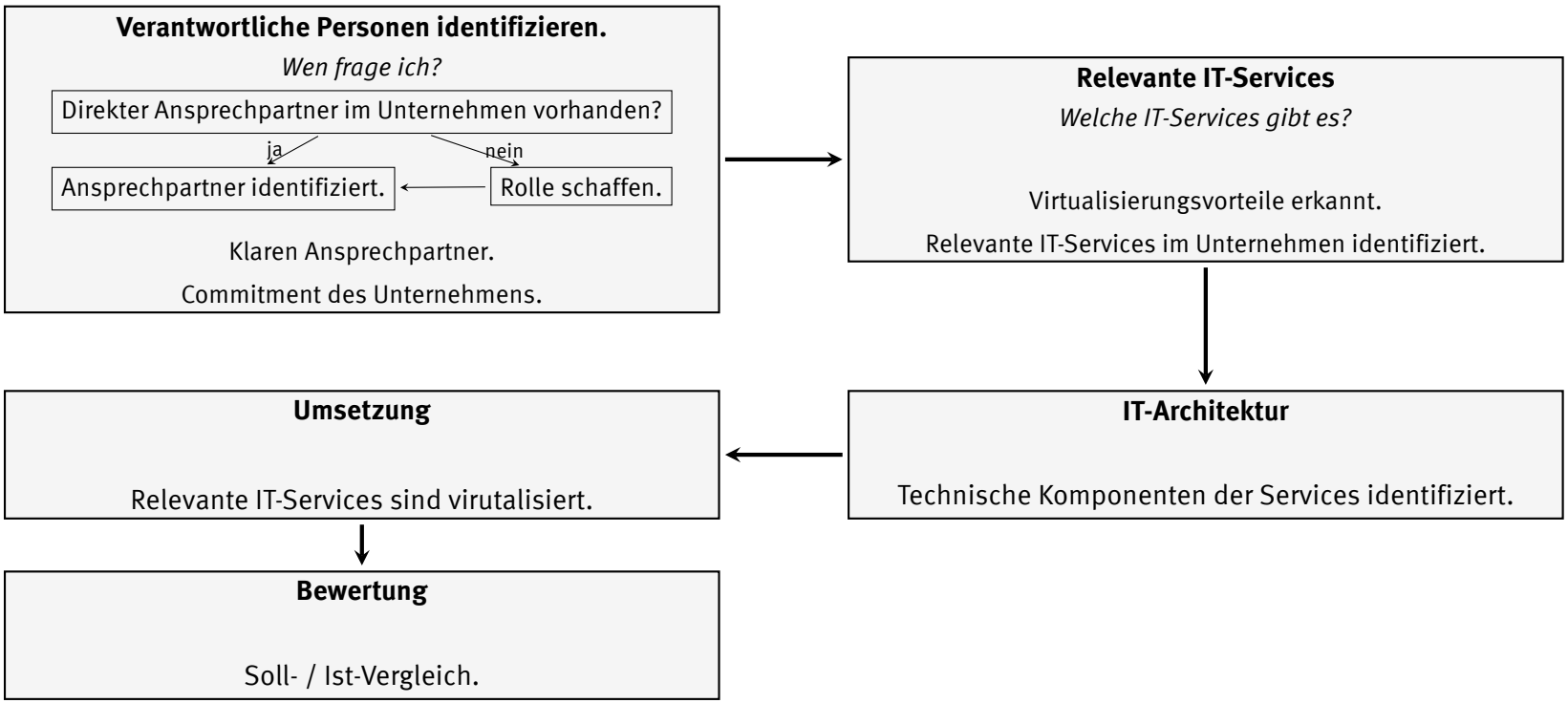
Vorgehensmodell zur Einführung von Virtualisierung.



Vorgehensmodell zur Einführung von Virtualisierung.

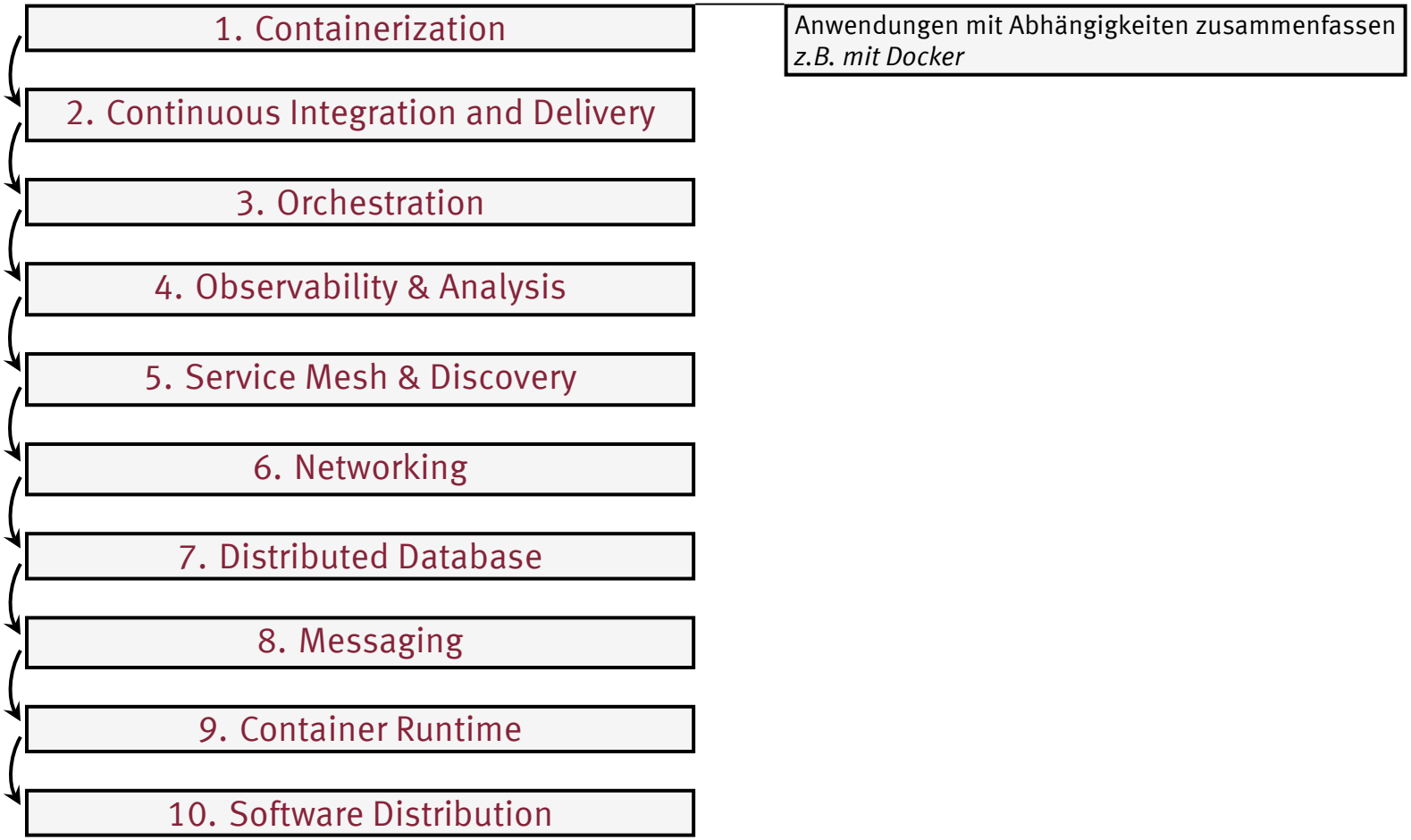


Vorgehensmodell zur Einführung von Virtualisierung.



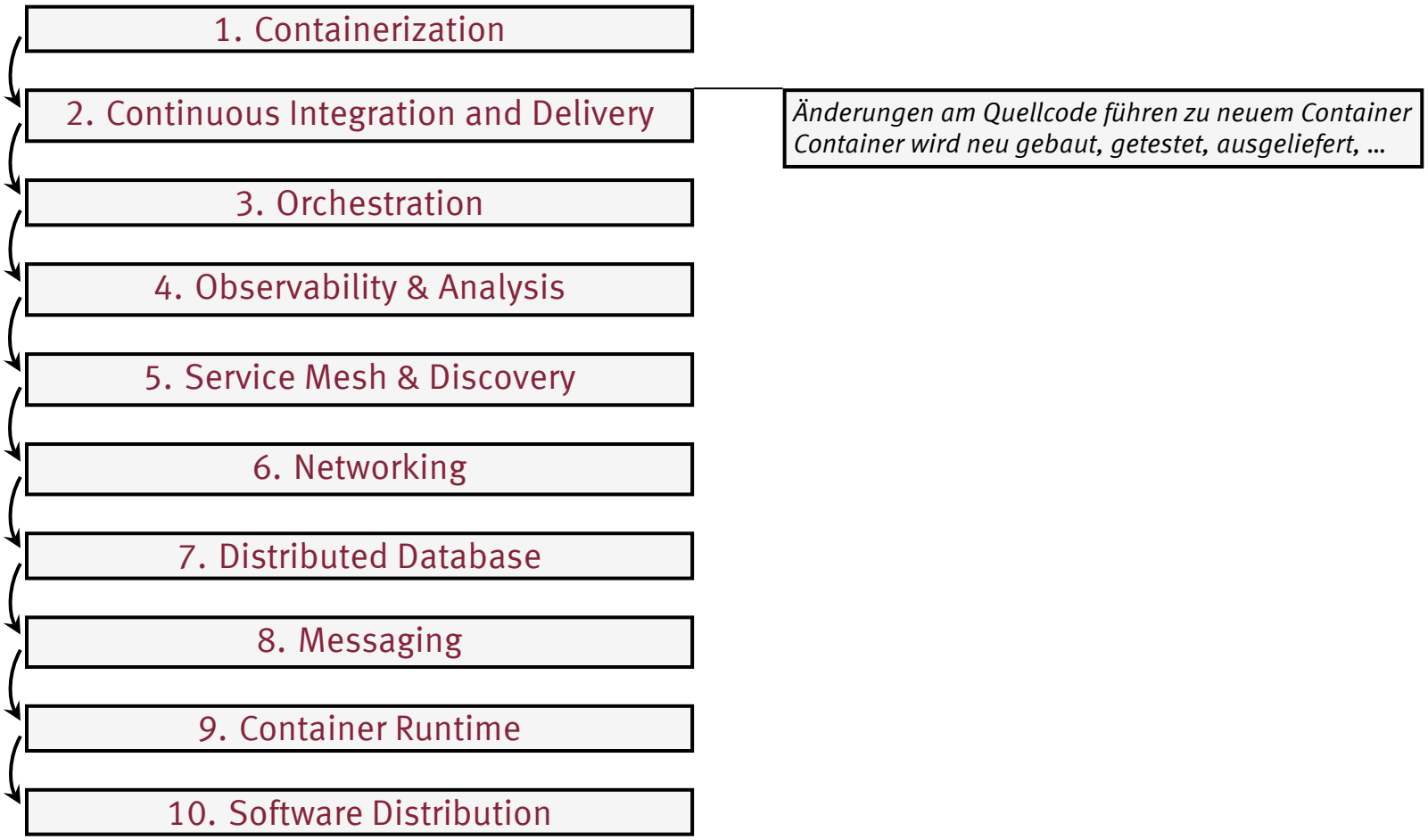
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



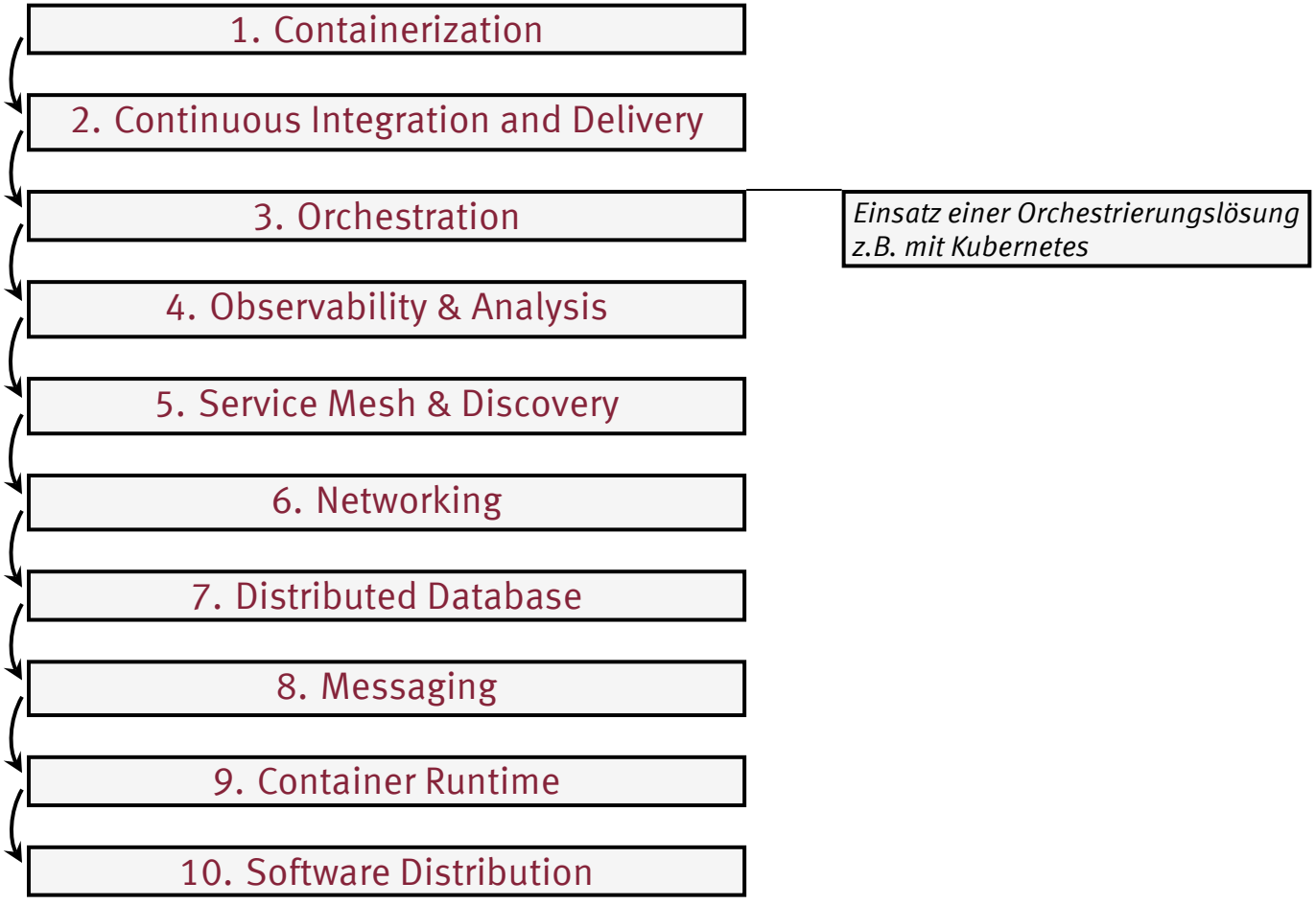
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



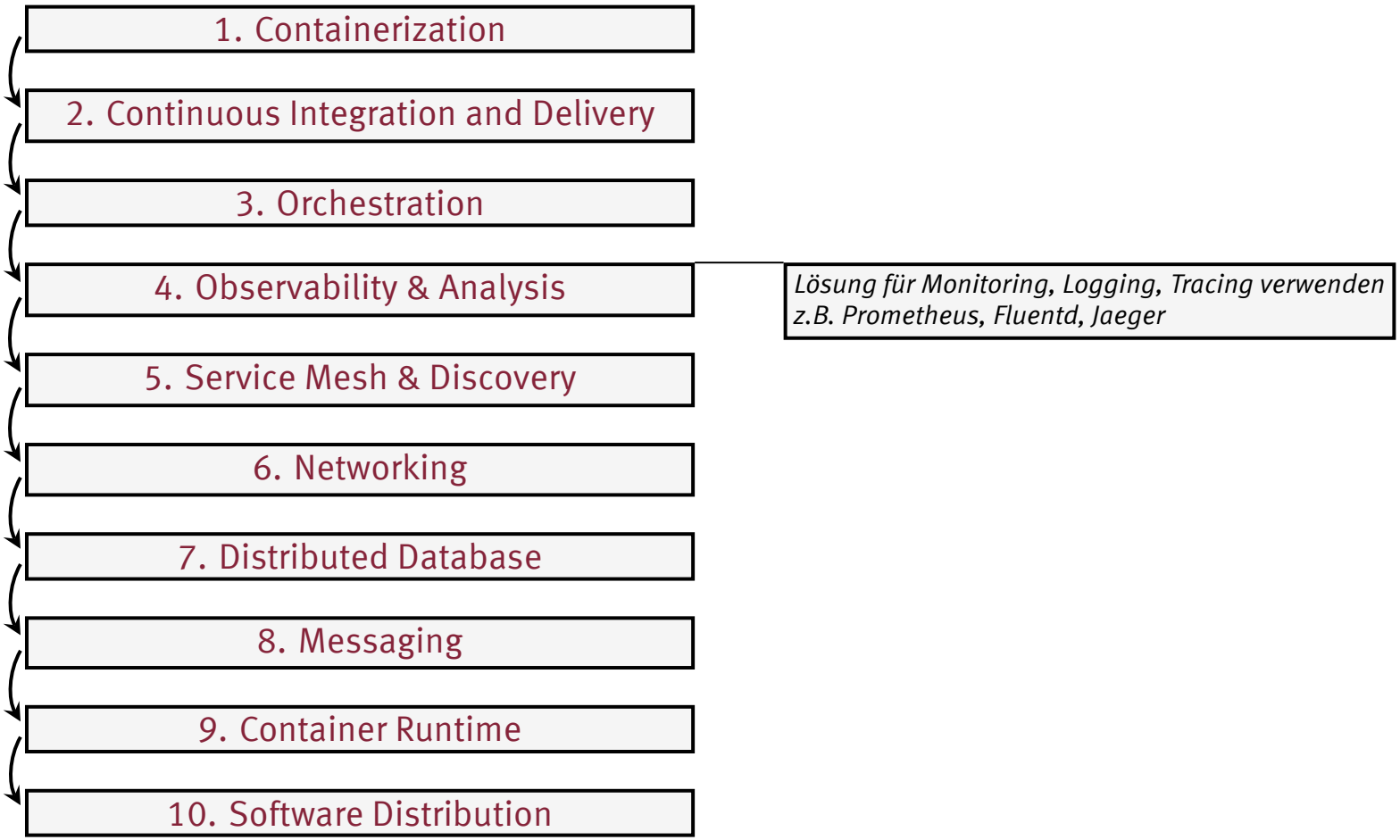
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



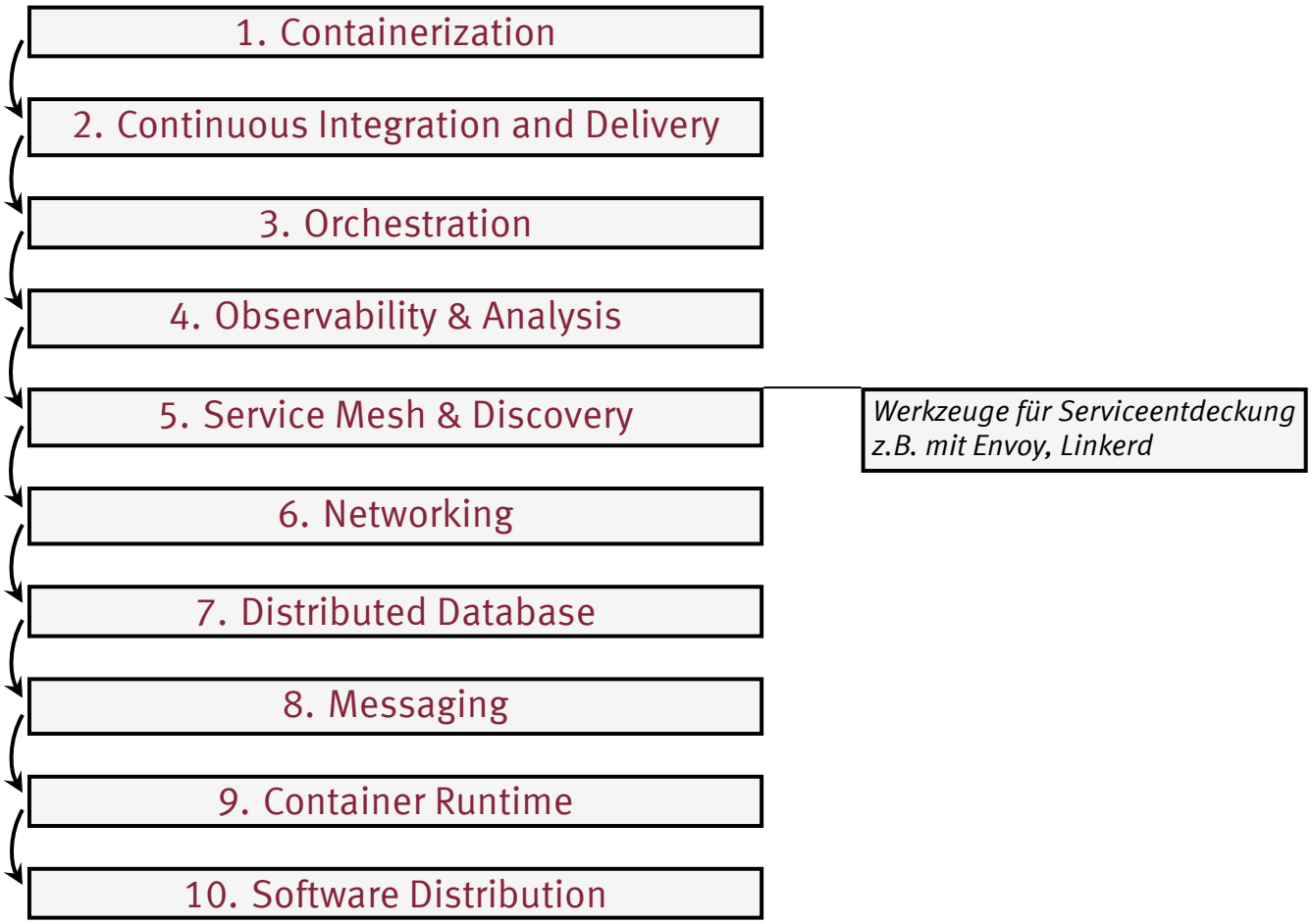
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



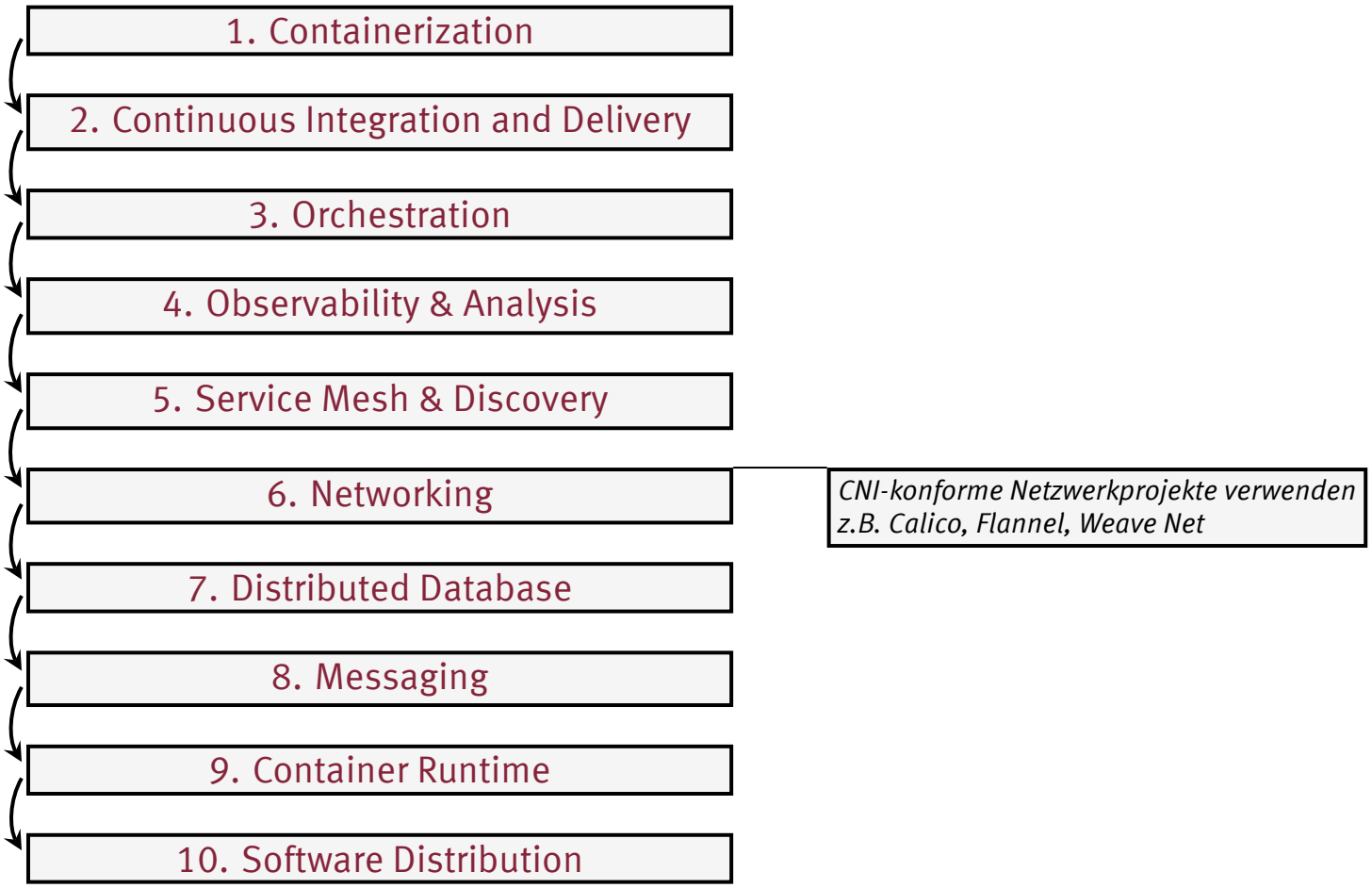
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



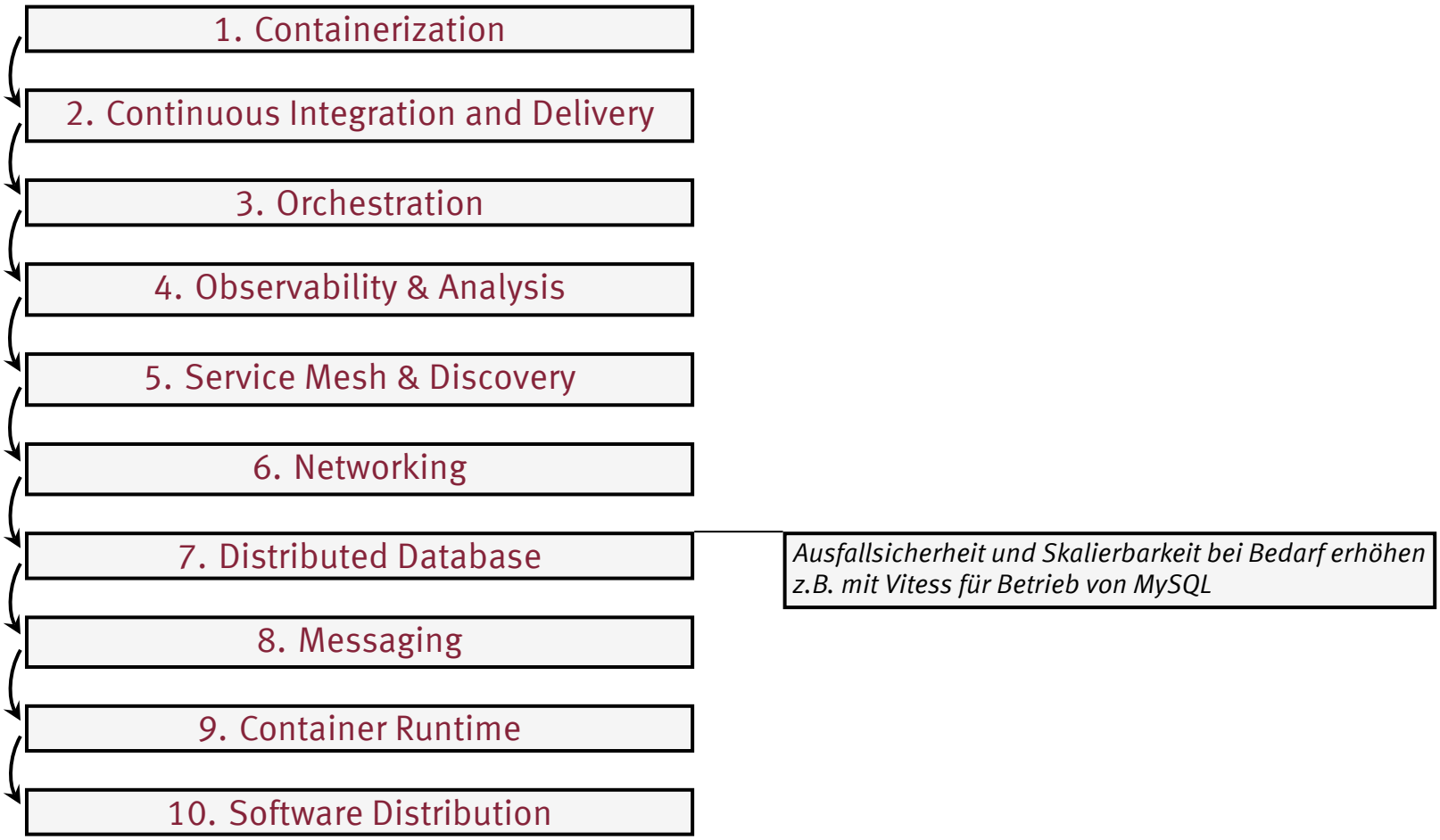
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



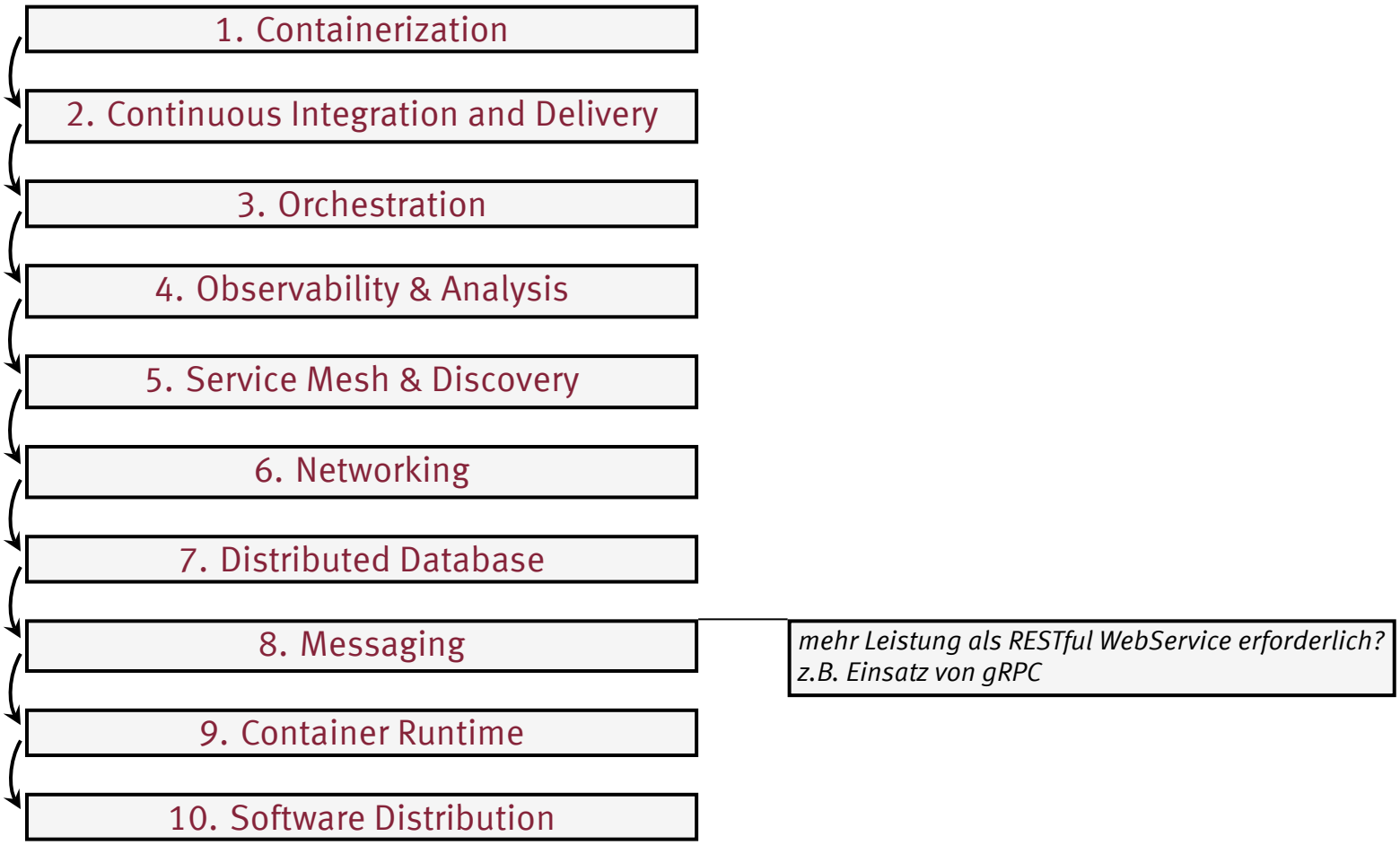
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



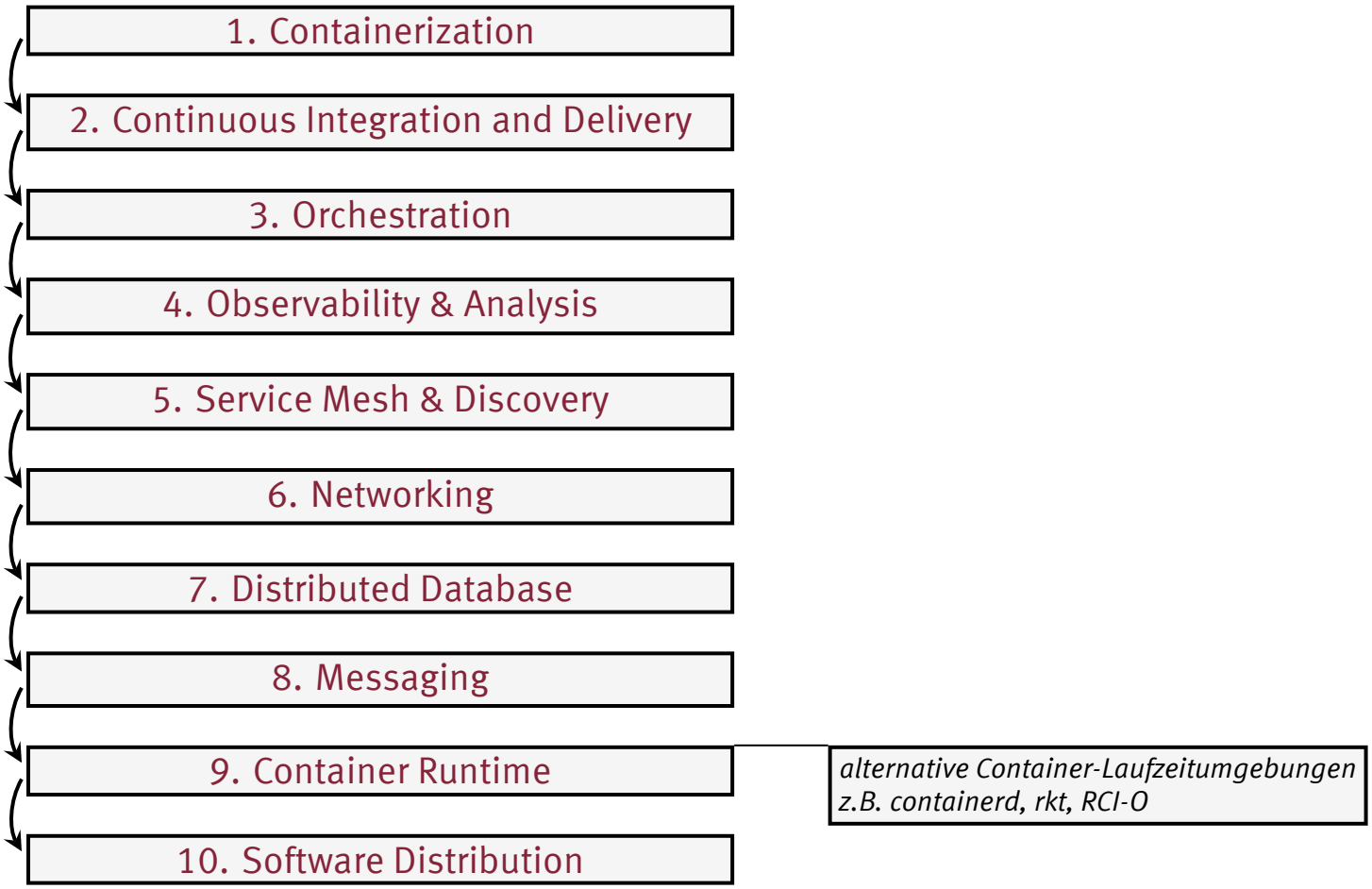
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



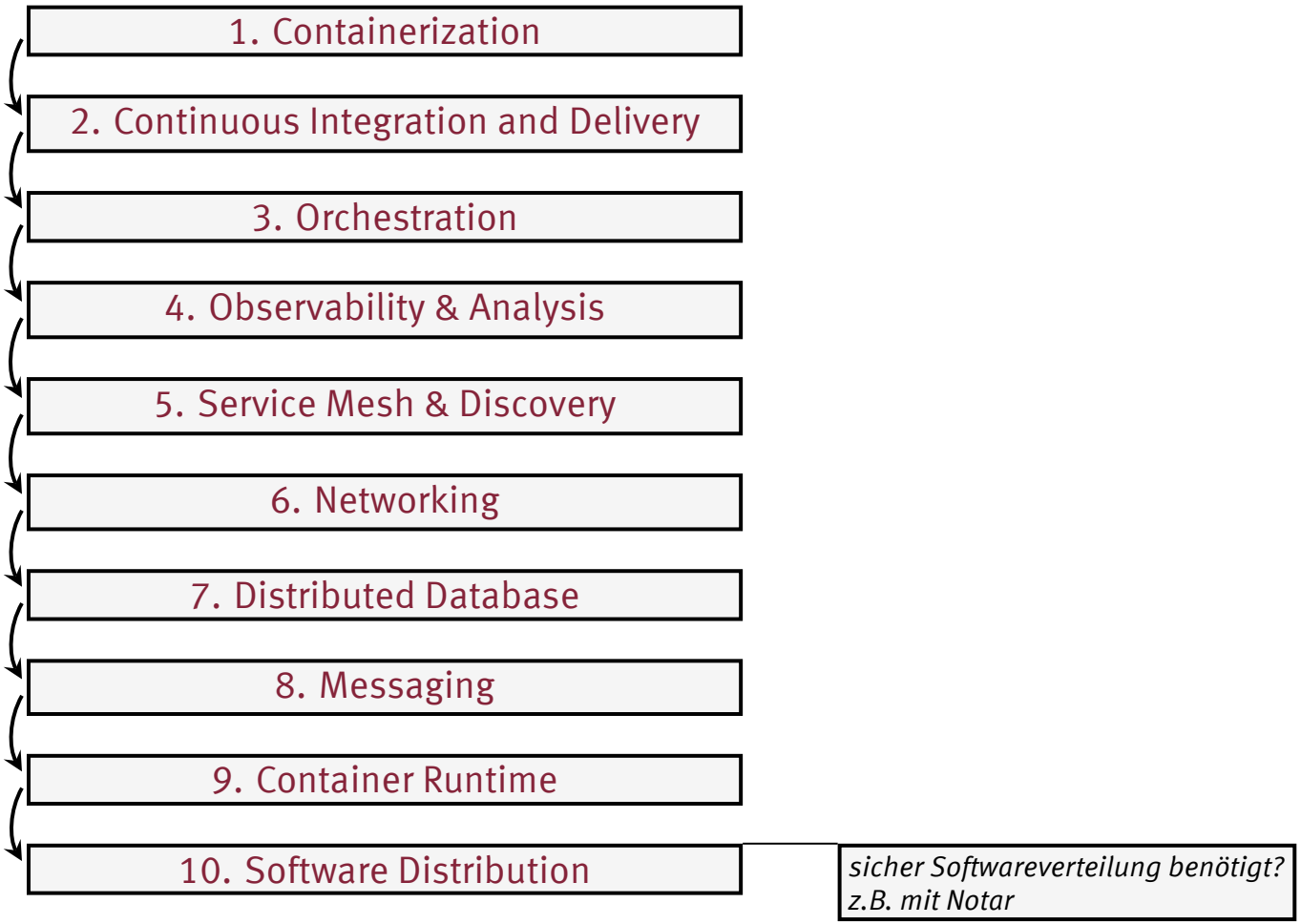
Container Foundation

10 Schritte zur Container-basierten Virtualisierung



Container Foundation

10 Schritte zur Container-basierten Virtualisierung



Container und Sicherheit

Sicherheits-Spektrum

- Container sind gut darin, Anwendungen zu *kapseln*.
  - Vorteile: Skalierbarkeit, Zugriffsregelung auf Ressourcen.

Container und Sicherheit

Sicherheits-Spektrum

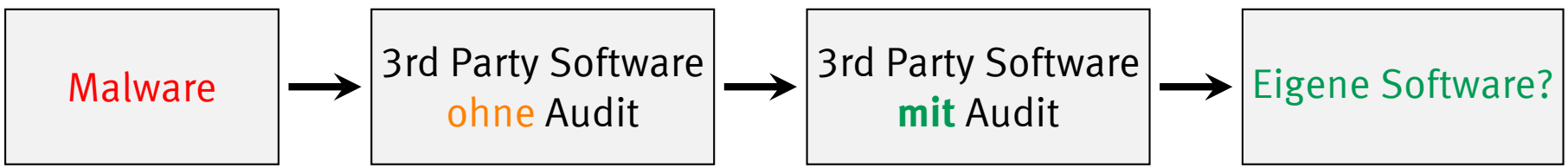
- Container sind gut darin, Anwendungen zu *kapseln*.
  - Vorteile: Skalierbarkeit, Zugriffsregelung auf Ressourcen.
  - Vorsicht: **Sicherheit** nicht Fokus von Container.

Container und Sicherheit

Sicherheits-Spektrum

- Container sind gut darin, Anwendungen zu *kapseln*.
  - Vorteile: Skalierbarkeit, Zugriffsregelung auf Ressourcen.
  - Vorsicht: **Sicherheit** nicht Fokus von Container.

- Gefährlichkeit von Software wird meist nicht binär bewertet, sondern existiert auf ein Spektrum.

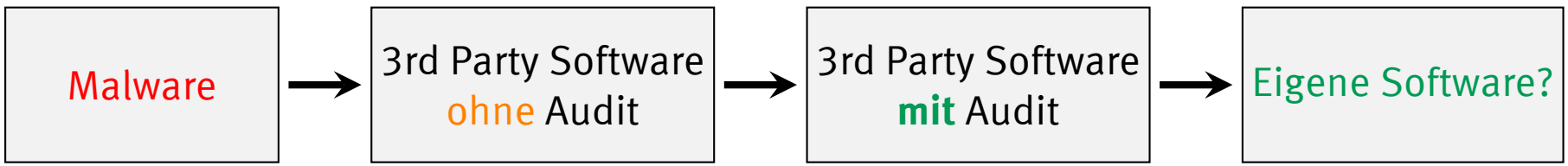


- **Ziel**: Verhinderung von Beeinflussung.

Container und Sicherheit

Sicherheits-Relativität

- **Ziel**: Verhinderung von Beeinflussung.

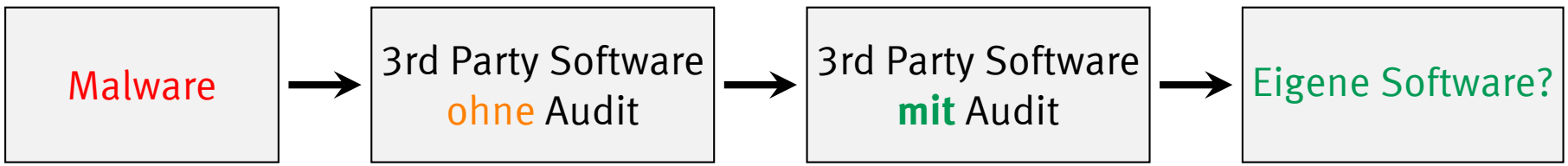


- Klassische Lösung: unsichere Software $\rightarrow$ virtuelle Maschine.
  - Bare Metal (Type1) oder Kernel-basierter Hypervisor (Type2).

Container und Sicherheit

Sicherheits-Relativität

- **Ziel**: Verhinderung von Beeinflussung.



- Klassische Lösung: unsichere Software $\rightarrow$ virtuelle Maschine.
  - Bare Metal (Type1) oder Kernel-basierter Hypervisor (Type2).
- Container-basierte Virtualisierung statt Type2 denkbar für mittelmäßig unsichere Software.
  - Geringere Angriffsfläche bezüglich Betriebssystem Angriffsvektoren.
  - *Kubernetes* mit verschiedenen Isolationsstufen **Pod**, **Node**, **Cluster**, **Project** mit unterschiedlicher kernel-, Daten- und Netzwerk-Isolierung.

Agenda

1. Motivation
2. Hardware- und Software-Virtualisierung
3. Zusammenfassung der Studie
4. Handlungsempfehlungen
   - Vorgehensmodell zur Einführung von Containern
   - Technologieauswahl: 10 Schritte zur Container Architektur
   - Überlegungen zum Virtualisierungs-Sicherheitslevel
5. Zusammenfassung

Zusammenfassung

- Nach dem „Tal der Tränen“ (Gartner) stellt sich heraus, wo Container für KMU nützlich sind:
  - Für DevOps → schnellere Releases.
  - Neue Mechanismen für Ausfallsicherheit und Erprobung der Resilience (Chaos Monkeys).

▸ Download

Zusammenfassung

- Nach dem „Tal der Tränen“ (Gartner) stellt sich heraus, wo Container für KMU nützlich sind:
  - Für DevOps → schnellere Releases.
  - Neue Mechanismen für Ausfallsicherheit und Erprobung der Resilience (Chaos Monkeys).
- Herausforderungen auch von KMU lösbar.
  - Know-How zu Containern (z.B. Docker) und Container-basierter Infrastruktur (z.B. mit Kubernetes) sollte langfristig aufgebaut werden.
  - Klassische Virtualisierung ist bewährt und wird auf mittelfristige Sicht nicht verschwinden, da …
  - …es aus sicherheitstechnischer Sicht auf unabsehbare Zeit relevant bleibt.

▸ Download

Thank You

ERCIS Leonardo-Campus 3, 48149 Münster, Germany

THE IS RESEARCH NETWORK

www.ercis.org