

# Windows Docker Container - Einführung



**Tobias Fenster**  
**Microsoft MVP Business Applications**  
**CTO Axians Infoma**

# Vorstellung



- ▶ Tobias Fenster  
CTO bei Axians Infoma  
Microsoft MVP für Business Solutions

[@tobiasfenster](#)

<https://navblog.axians-infoma.com>

<https://github.com/tfenster>

- ▶ Ziel für heute: Eine **Basiseinführung** in Docker auf Windows

# Was ist docker ?

Führende cross-platform Software-Container-Umgebung

- ▶ Docker **Image**: **Template** mit dem notwendigen **Minimum** an **Betriebssystem**, **Bibliotheken** und **Anwendungsdateien**
  - Versioniert über **Tags**
- ▶ Docker **Container**: Die **laufende Instanz** eines Image mit einer unveränderbaren Basis und als Schicht darauf allen Änderungen
  - **Keine VM**, insbesondere keine GUI für z.B. RDP-Zugriffe o.ä.
- ▶ Docker **Host**: Virtueller/physischer Host, auf dem **Container laufen**
- ▶ Docker **Registry**: System zur **Image-Verwaltung** (public/private)
- ▶ Docker **Repository**: “Ordner” in einer Registry

# Was ist docker ?

Die Grundideen: Wie komme ich zu einem laufenden Container?

Passendes  
Image wählen

hello-world:nanoserver-1809

Hinweis: Eigentlich anzugeben ist  
`<registry>/<repository>/<image>:<tag>`  
aber Docker macht Annahmen, so dass z.B.  
`hello-world:nanoserver-1809`  
automatisch ergänzt wird zu  
`docker.io/library/hello-world:nanoserver-1809`

Microsoft-Images im Docker Hub gelistet, aber  
Download (pull) von `mcr.Microsoft.com`

# Was ist docker ?

Die Grundideen: Wie komme ich zu einem laufenden Container?

Passendes  
Image wählen

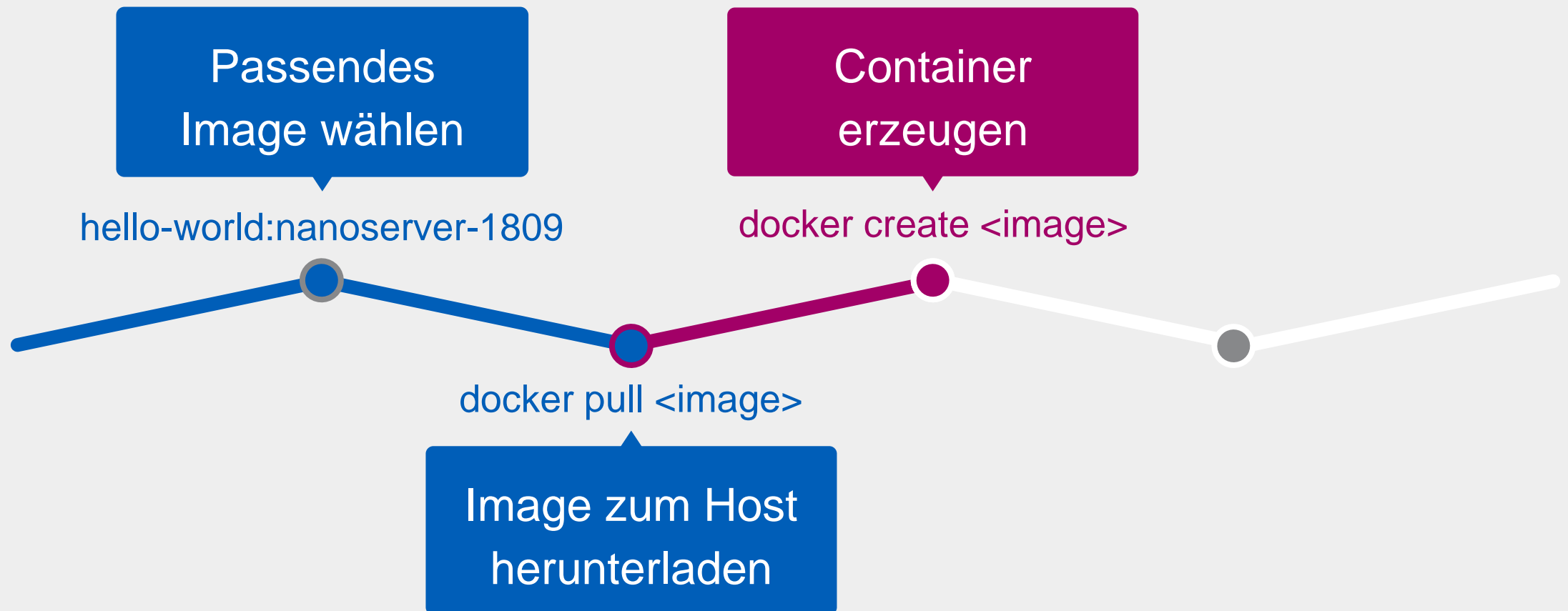
hello-world:nanoserver-1809

docker pull <image>

Image zum Host  
herunterladen

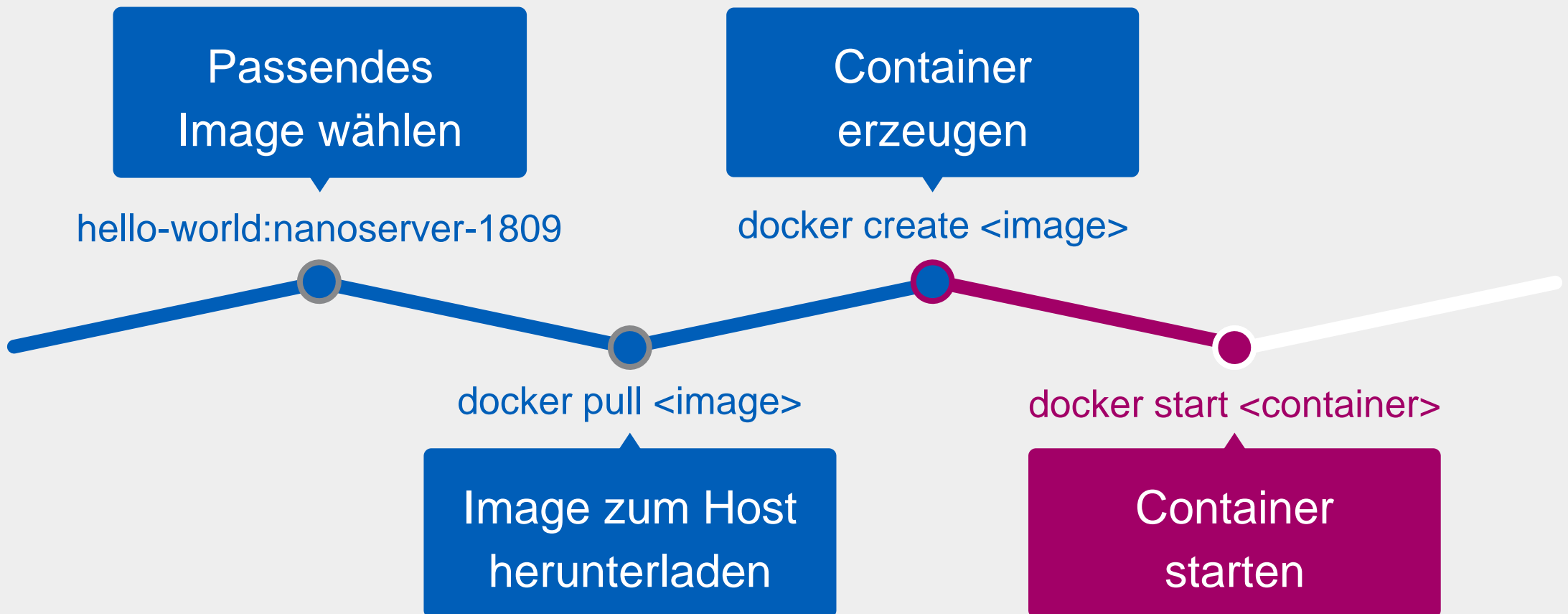
# Was ist docker ?

Die Grundideen: Wie komme ich zu einem laufenden Container?



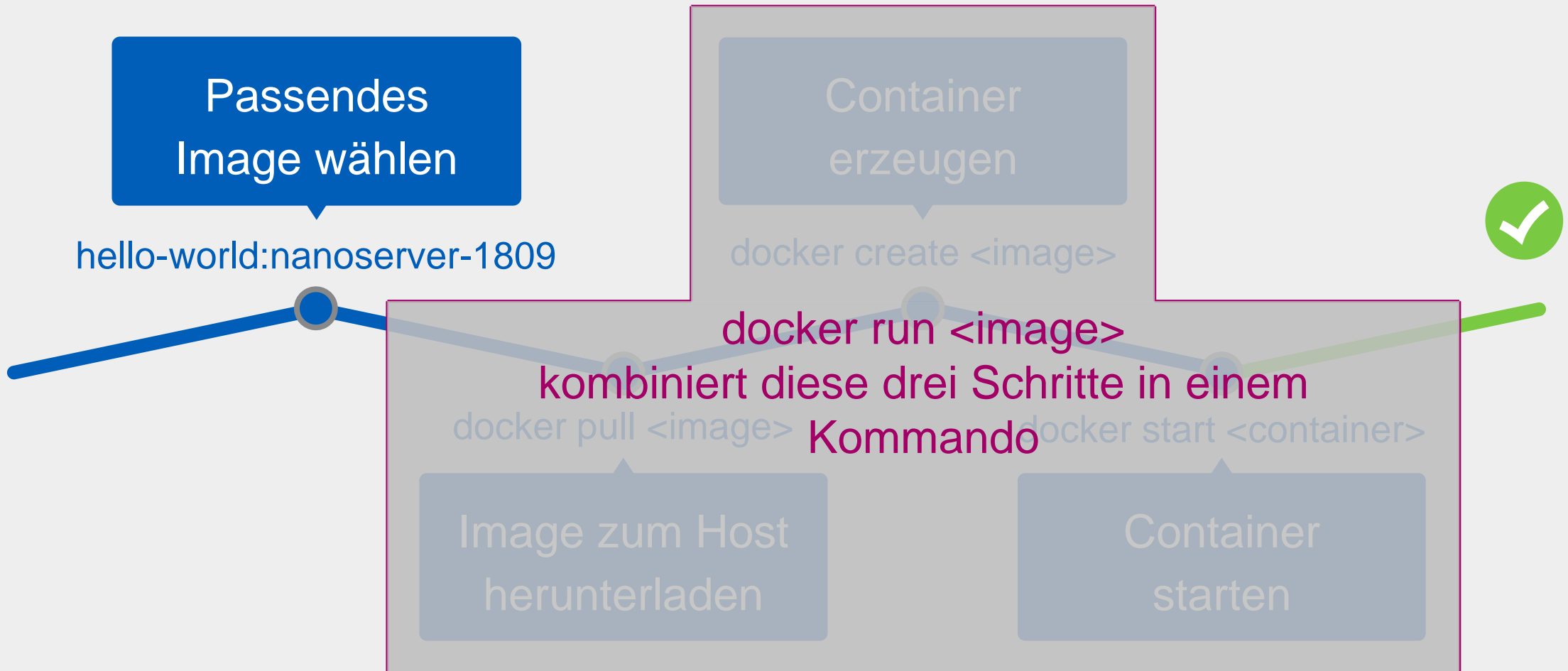
# Was ist docker ?

Die Grundideen: Wie komme ich zu einem laufenden Container?



# Was ist docker ?

Die Grundideen: Wie komme ich zu einem laufenden Container?





# Was ist docker ?

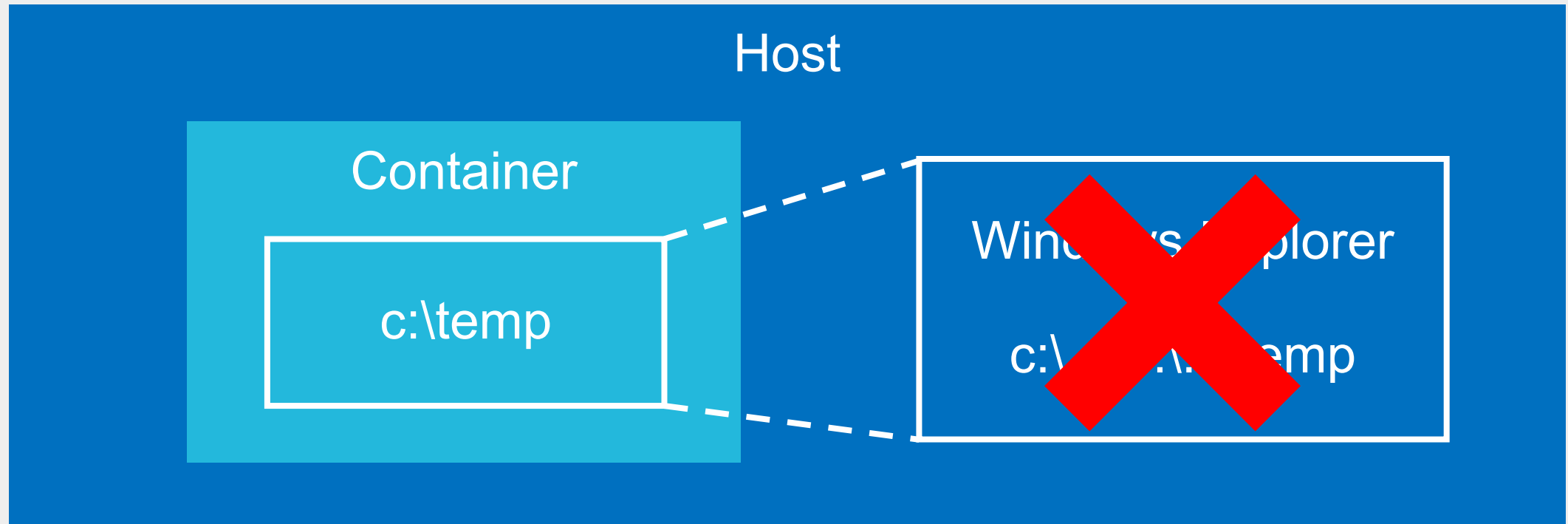
Die Grundideen: Wie komme ich zu einem laufenden Container?

- ▶ Laufende und alle Container anzeigen
- ▶ Container interaktiv starten
- ▶ Eine PowerShell-Session in einem laufenden Container öffnen
- ▶ Ressourcen-Nutzung und Logs
- ▶ Container stoppen und entfernen
- ▶ Image entfernen



# Was ist docker ?

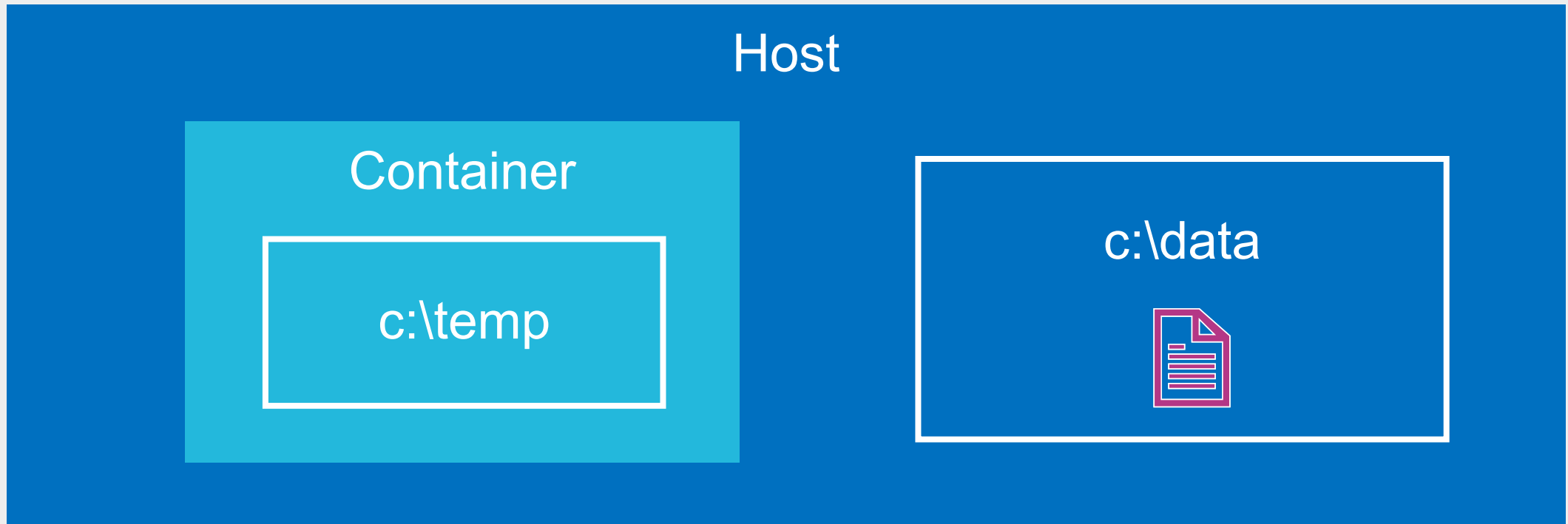
Die Grundideen: Umgang mit Dateien



Standard-Dateisystemeinrichtung: nichts konfiguriert

# Was ist docker ?

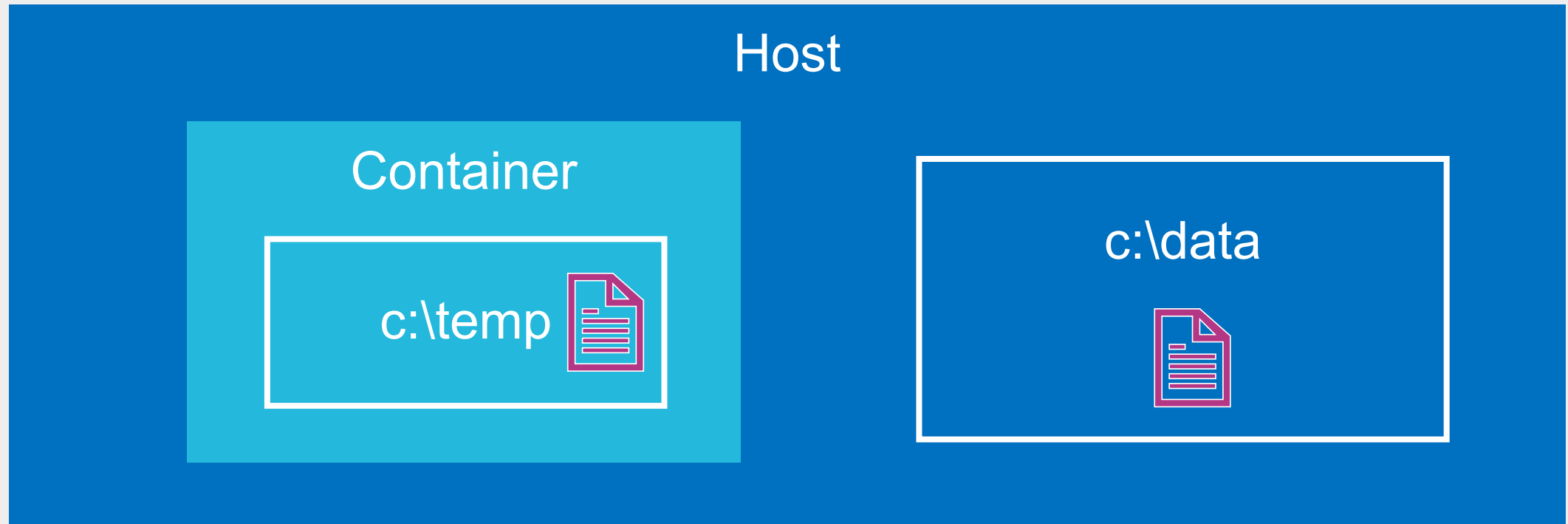
Die Grundideen: Umgang mit Dateien



Standard-Dateisystemeinrichtung: nichts konfiguriert. Nutzung von `docker cp` zum Kopieren von Dateien

# Was ist docker ?

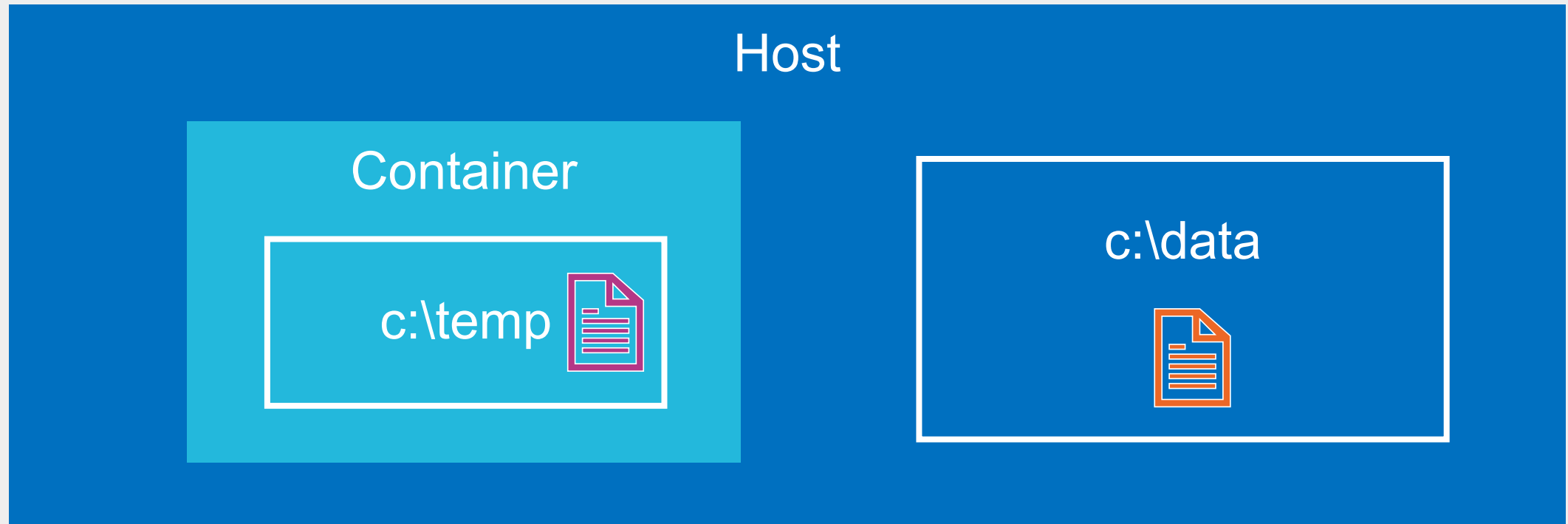
Die Grundideen: Umgang mit Dateien



Standard-Dateisystemeinrichtung: nichts konfiguriert. Nutzung von `docker cp` zum Kopieren von Dateien

# Was ist docker ?

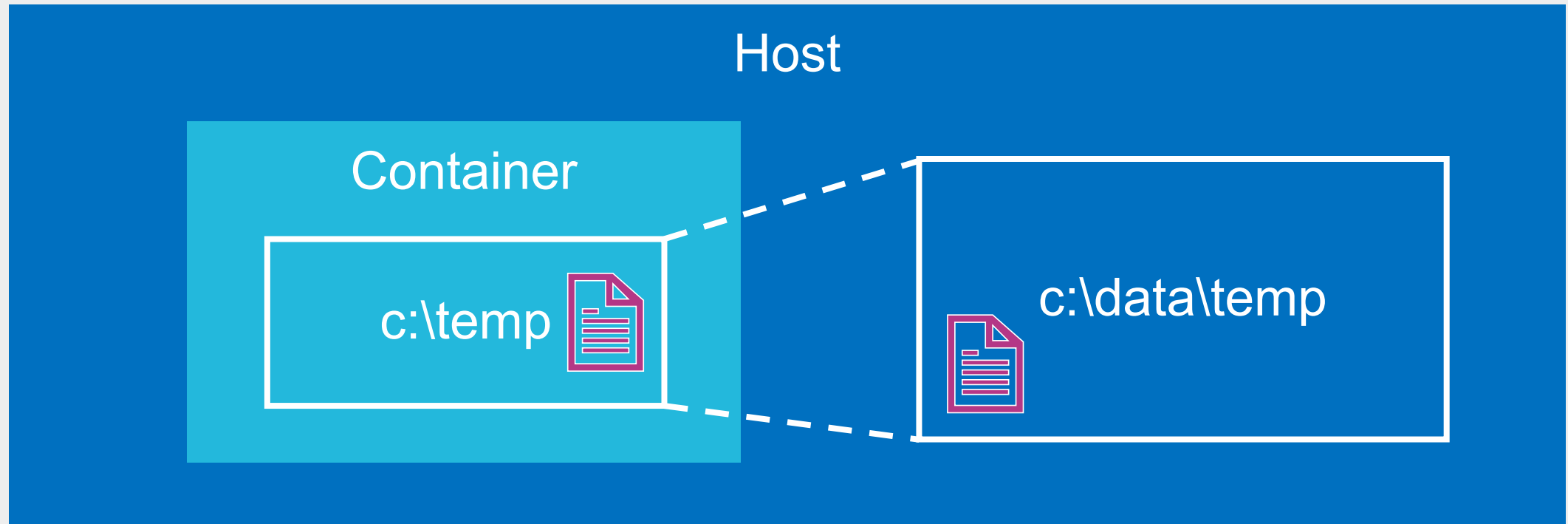
Die Grundideen: Umgang mit Dateien



Standard-Dateisystemeinrichtung: nichts konfiguriert. Nutzung von `docker cp` zum Kopieren von Dateien

# Was ist docker ?

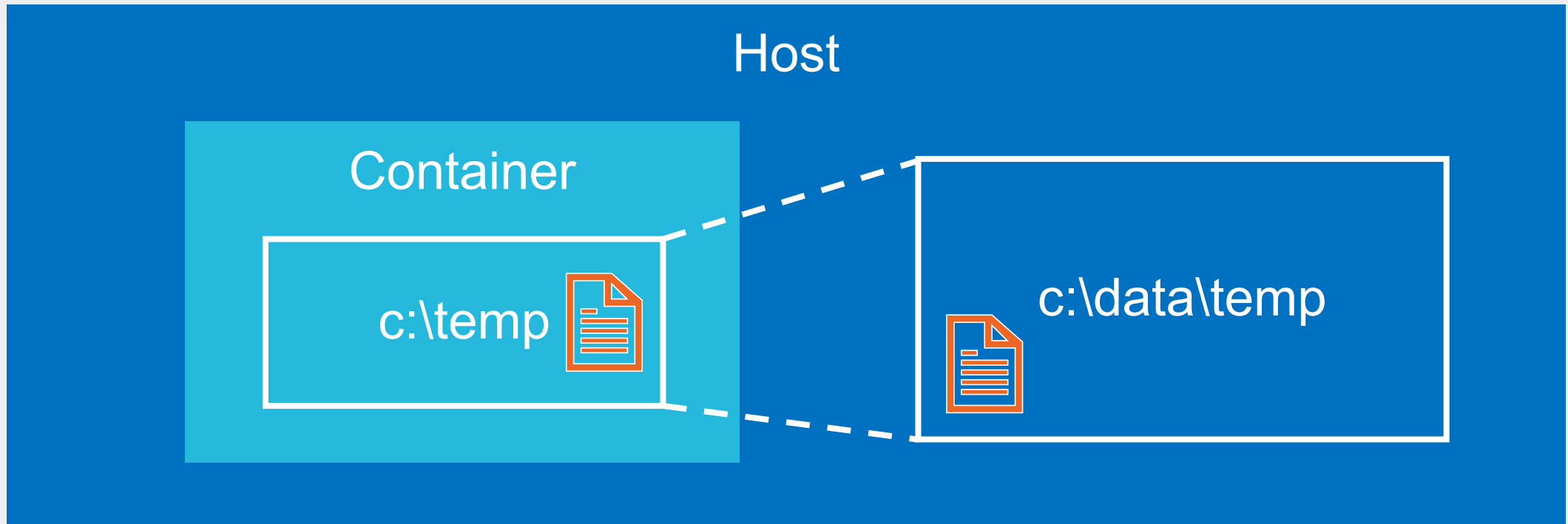
Die Grundideen: Umgang mit Dateien



Dateisystemeinrichtung mit einem **Volume**, z.B. `-v c:\data\temp:c:\temp`

# Was ist docker ?

Die Grundideen: Umgang mit Dateien



Dateisystemeinrichtung mit einem **Volume**, z.B. `-v c:\data\temp:c:\temp`

# Was ist docker ?

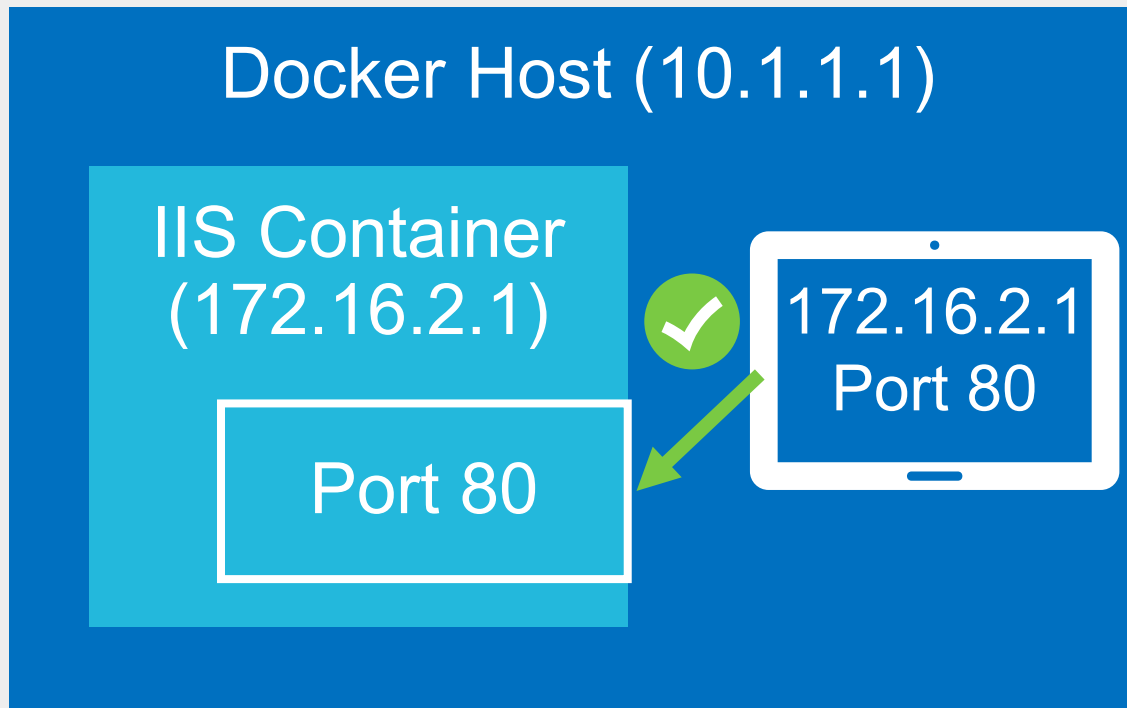
## Die Grundideen: Umgang mit Dateien

- ▶ Zwei Möglichkeiten für **Dateiaustausch** zwischen Host und Container:
  - **Kopieren von Dateien** mit **docker cp** führt zu zwei identischen, aber nicht verknüpften Dateien  
→ funktioniert **immer**
  - Parameter **-v** erzeugt ein **Volume**, das **geteilte Ordner und Dateien** zwischen Host und Container erlaubt (bis Server 2016 muss der Zielordner leer sein)  
→ kann **nur bei Containerstart** eingerichtet werden  
→ Plugins für die direkte Anbindung von Storage-Lösungen



# Was ist docker ?

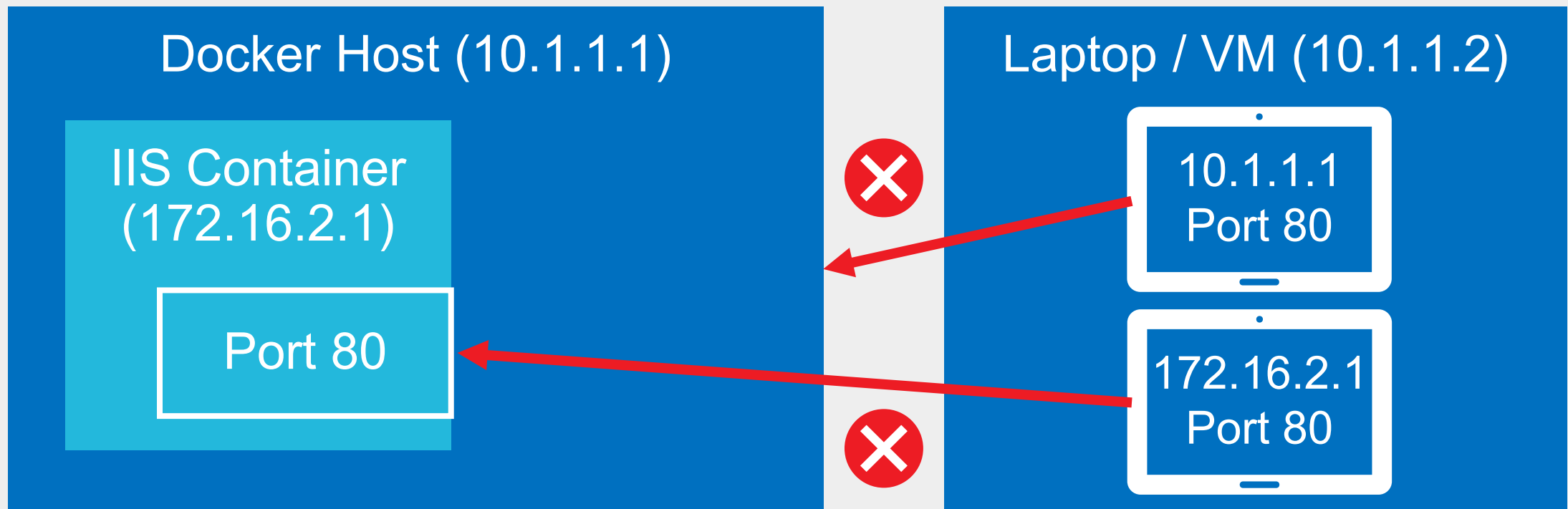
Die Grundideen: Arbeiten im Netzwerk



Standard-Netzwerkeinrichtung: NAT

# Was ist docker ?

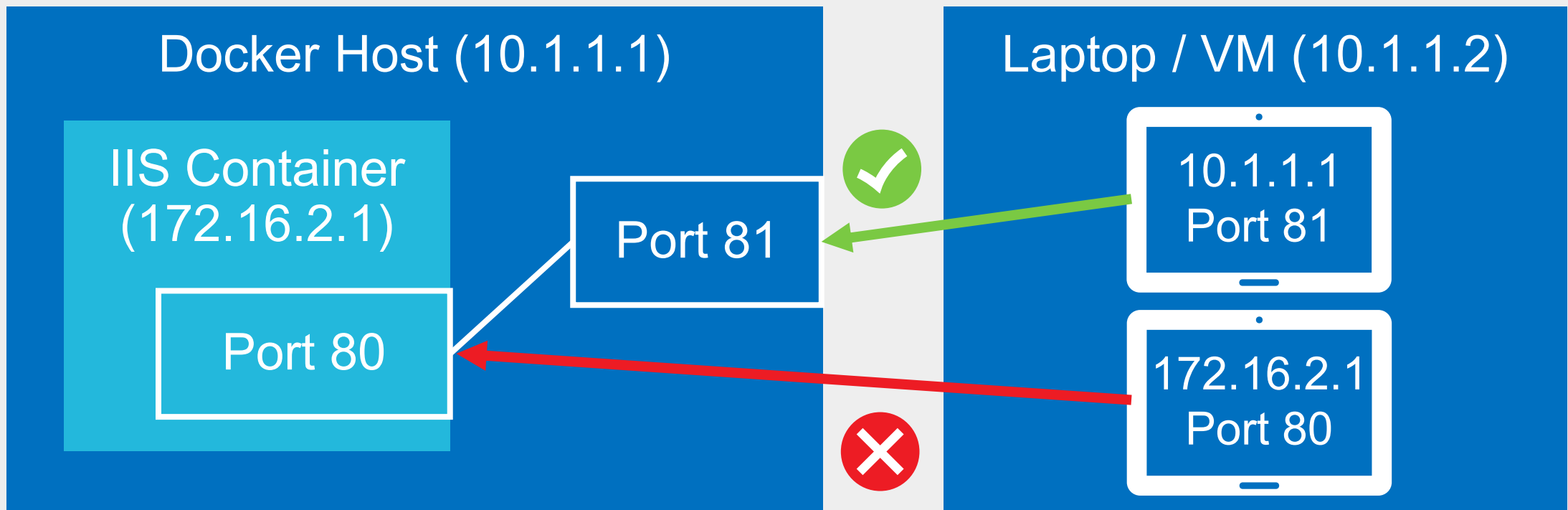
Die Grundideen: Arbeiten im Netzwerk



Standard-Netzwerkeinrichtung: NAT

# Was ist docker ?

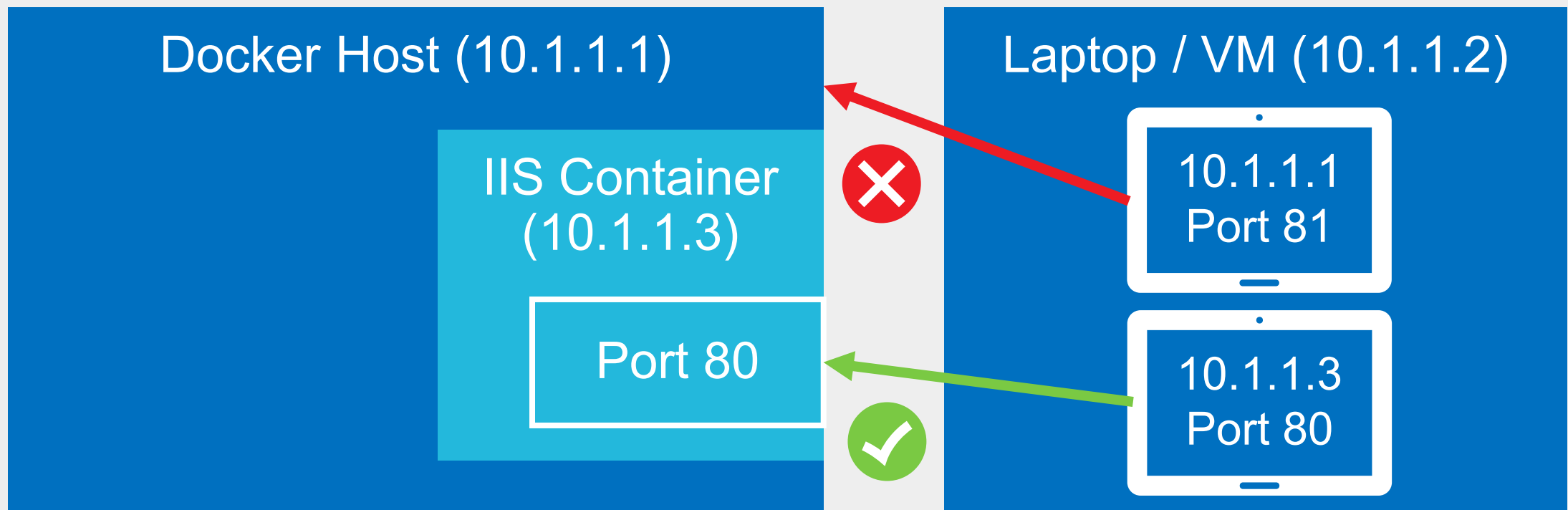
Die Grundideen: Arbeiten im Netzwerk



Standard-Netzwerkeinrichtung mit **Port Mapping**, z.B. -p 81:80

# Was ist docker ?

Die Grundideen: Arbeiten im Netzwerk



Einrichtung mit transparentem Netzwerk: “Geteilter” Netzwerkadapter

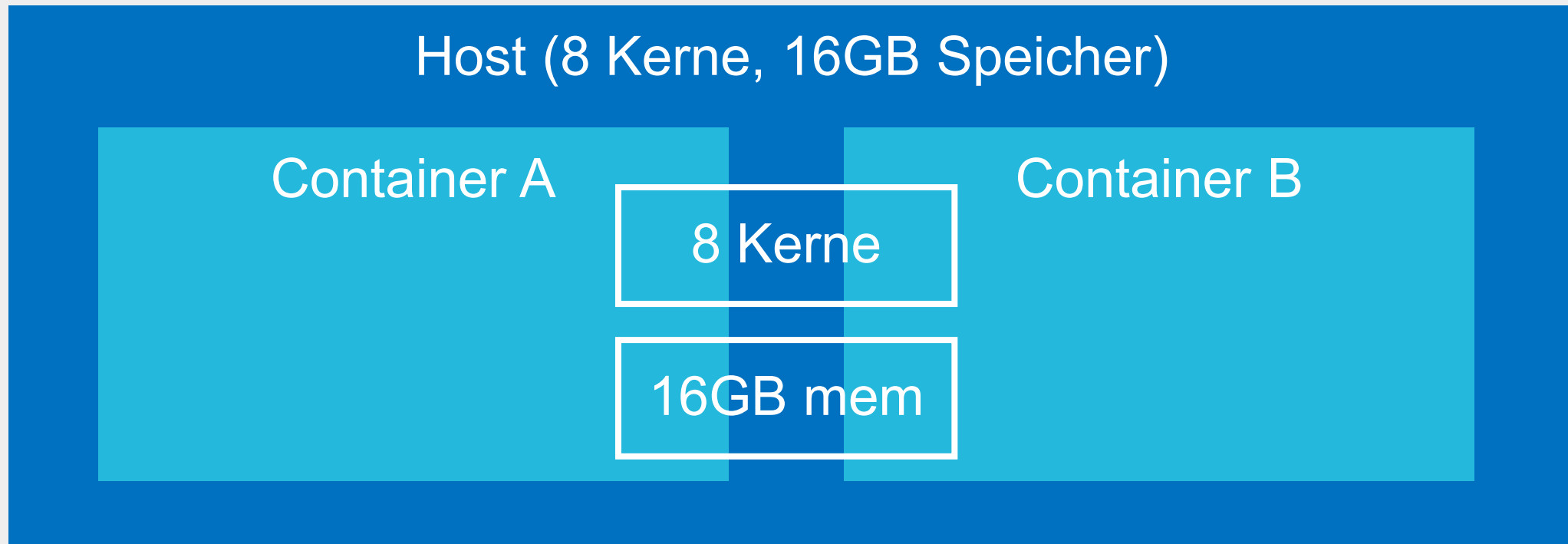
# Was ist docker ?

## Die Grundideen: Arbeiten im Netzwerk

- ▶ Drei Möglichkeiten für **Netzwerkverbindungen** zum Container:
  - Standardeinrichtung: **NAT-Netzwerk** erlaubt nur Verbindungen vom Host zum Container
  - **Port Mapping** von 1-n Ports im Container auf 1-n ggf. abweichende Ports im Host
    - Host-Firewall beachten
  - Geteilter Netzwerkadapter über **transparente Einrichtung** bringt eine dedizierte IP (statisch oder dynamisch) für jeden Container und macht ihn im Netzwerk erreichbar
    - Abhängig von allgemeiner Netzwerkeinrichtung
    - Benötigt MAC-Address-Spoofing

# Was ist docker ?

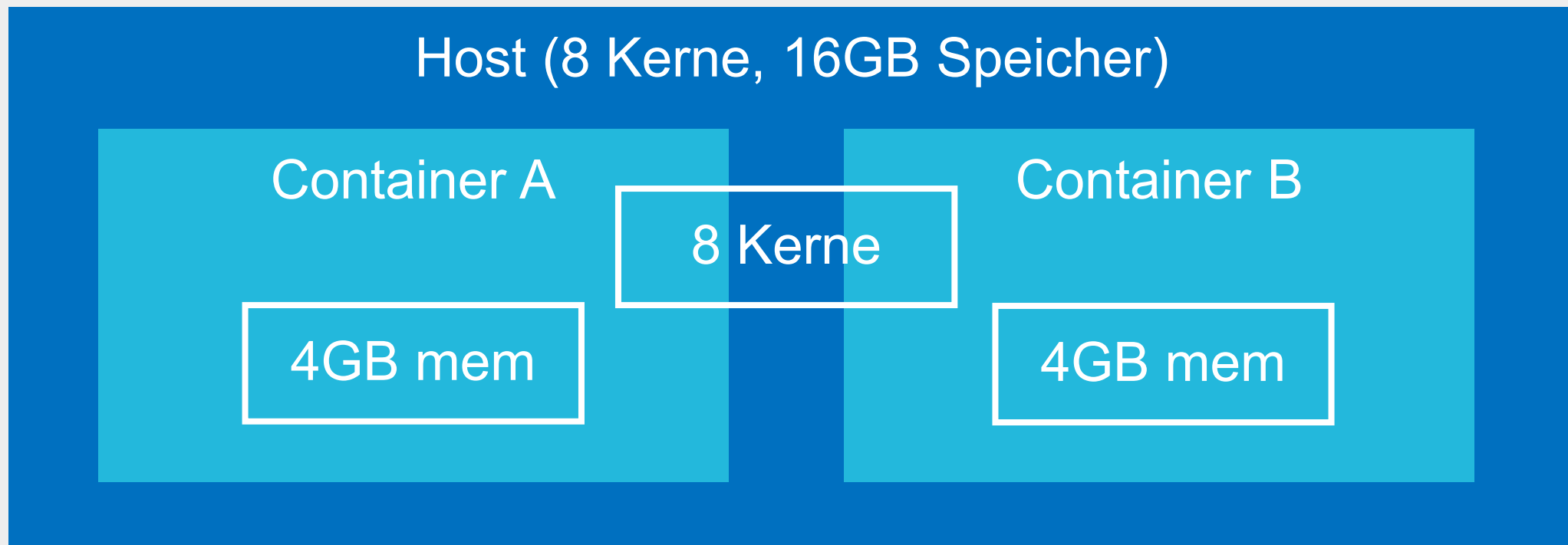
Die Grundideen: Ressourcenbeschränkungen



Standard-Ressourceneinrichtung: nichts konfiguriert

# Was ist docker ?

Die Grundideen: Ressourcenbeschränkungen



Aktiviert Ressourcenbeschränkung: z.B. **Speichergrenze** -m 4g

# Was ist docker ?

## Die Grundideen: Ressourcenbeschränkungen

- ▶ Diverse Beschränkungsmöglichkeiten für CPU, Speicher und E/A
  - `docker run --help` gibt einen guten Überblick
- ▶ Kann nur bei Containerstart eingerichtet werden

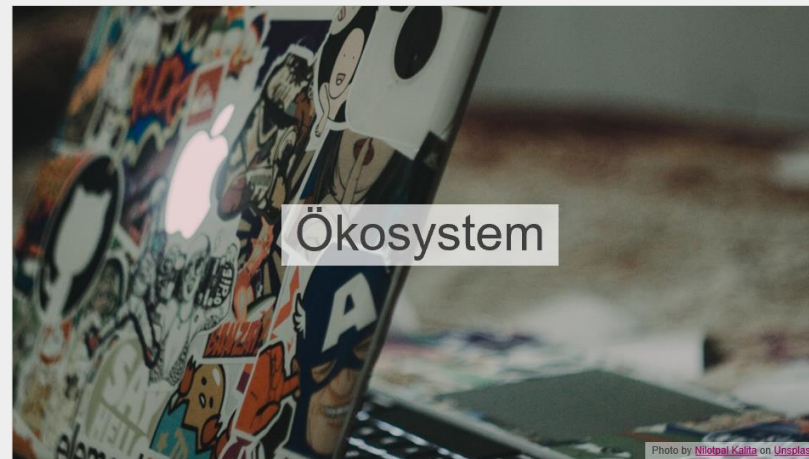
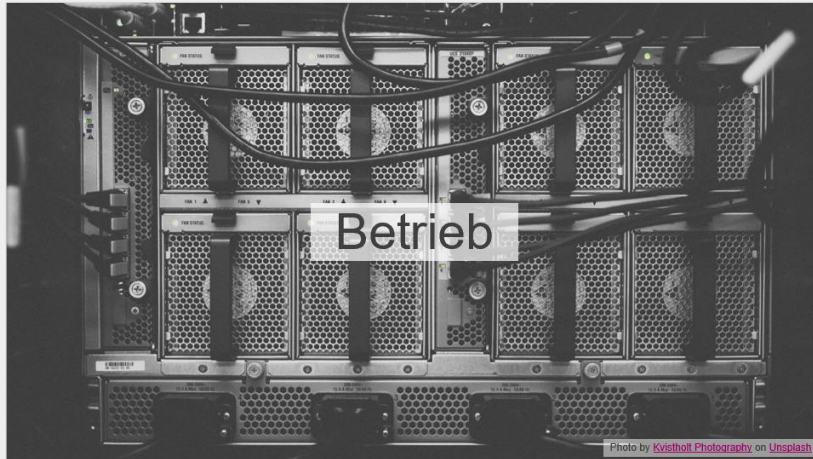


# Weshalb docker. ?

## Entwicklung / Betrieb / Ökosystem

- ▶ Grundidee: **Einheitliches** Format für die **Auslieferung bzw. Übertragung** von **kompletten Anwendungen**
  - .dll / .jar / ... zu wenig, nicht einheitlich
  - VMs zu viel
- ▶ Wer 20 Minuten Zeit hat: [Why we built Docker](#), erste öffentliche Session zu Docker bei der PyCon 2013, von Solomon Hykes, einem der Gründer des OpenSource-Projekts Docker und der daraus entstandenen Firma Docker Inc.

# Weshalb docker ?





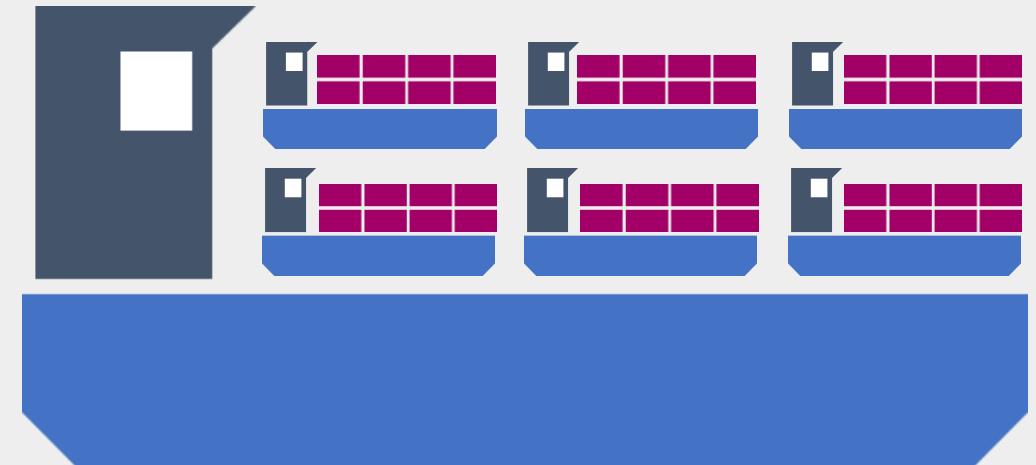
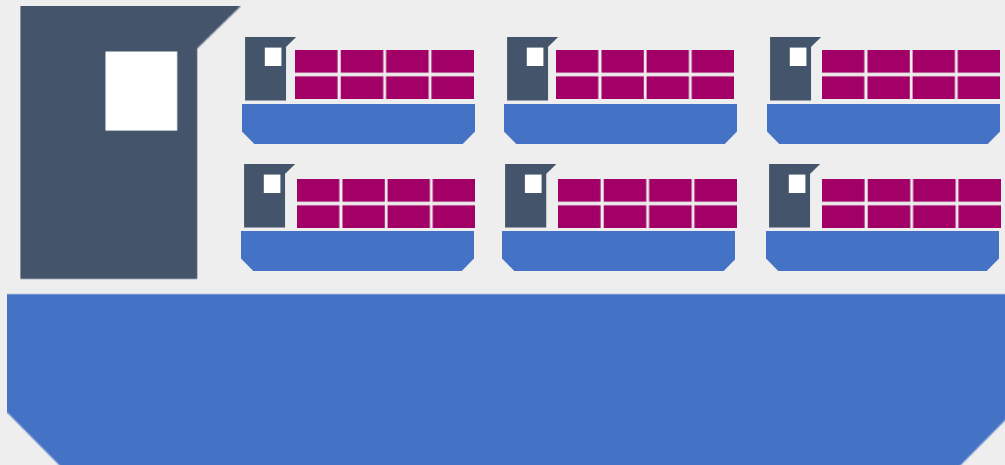
A black and white photograph of a server rack. The rack is filled with server units, each featuring a perforated metal front panel. Several cables are plugged into the top of the rack. A semi-transparent white box with the word "Betrieb" in a large, bold, sans-serif font is centered over the middle of the image. The background is dark, and the lighting highlights the metallic textures of the server components.

# Betrieb



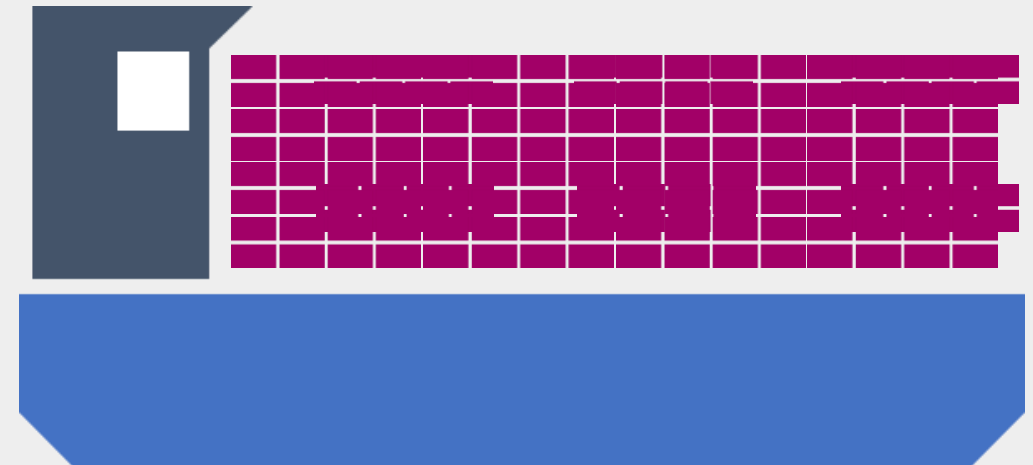
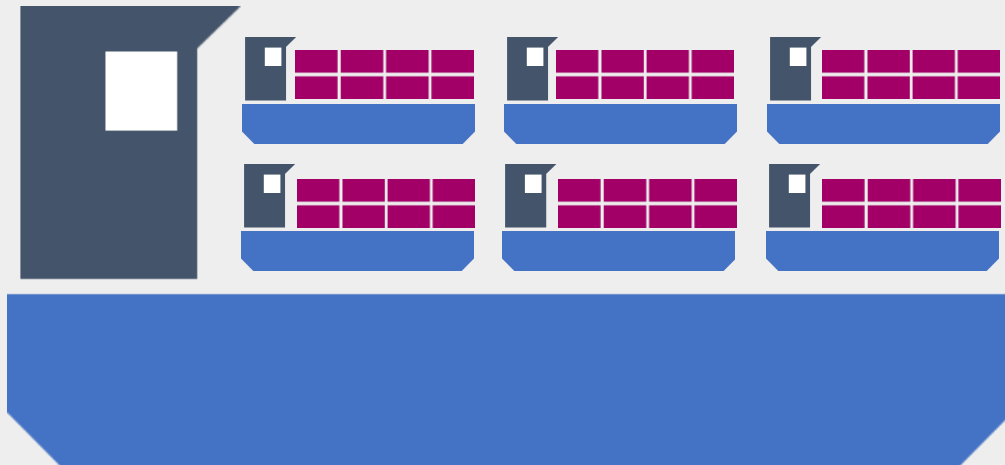
# Betrieb

## Virtuelle Maschinen vs. Container



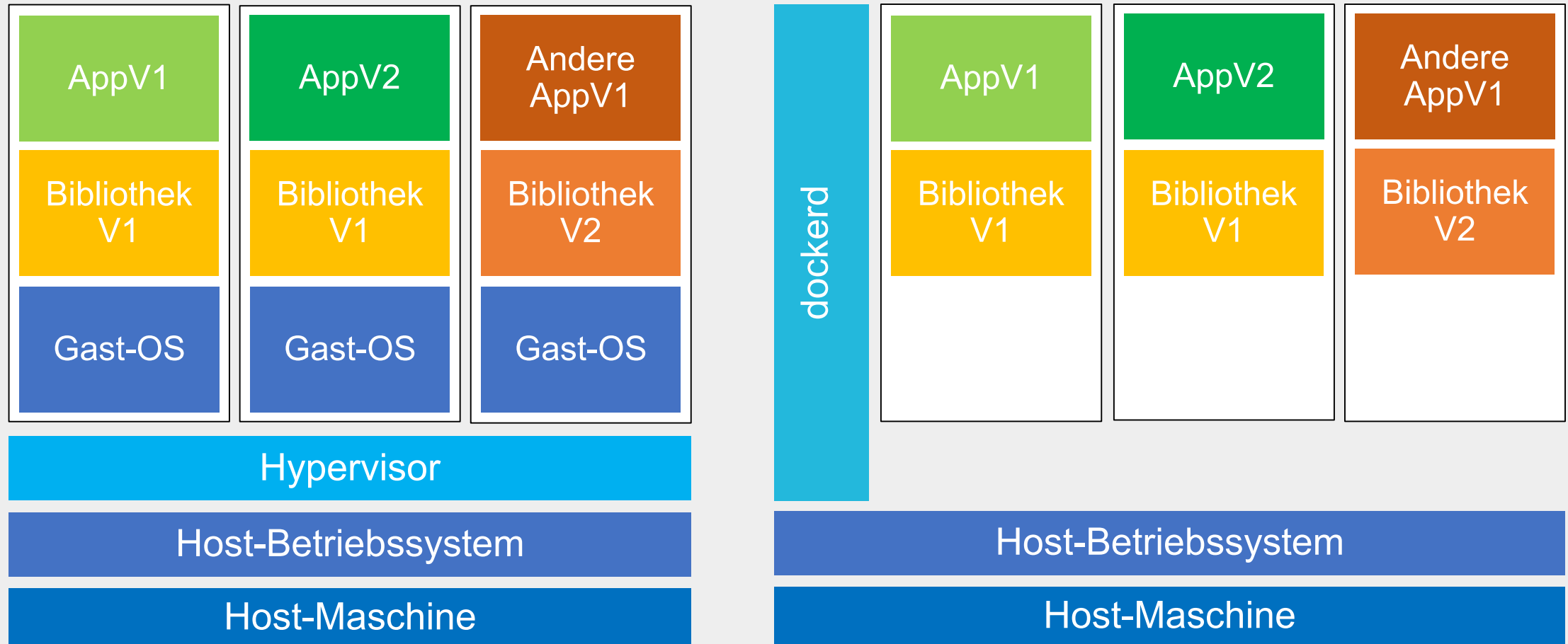
# Betrieb

## Virtuelle Maschinen vs. Container



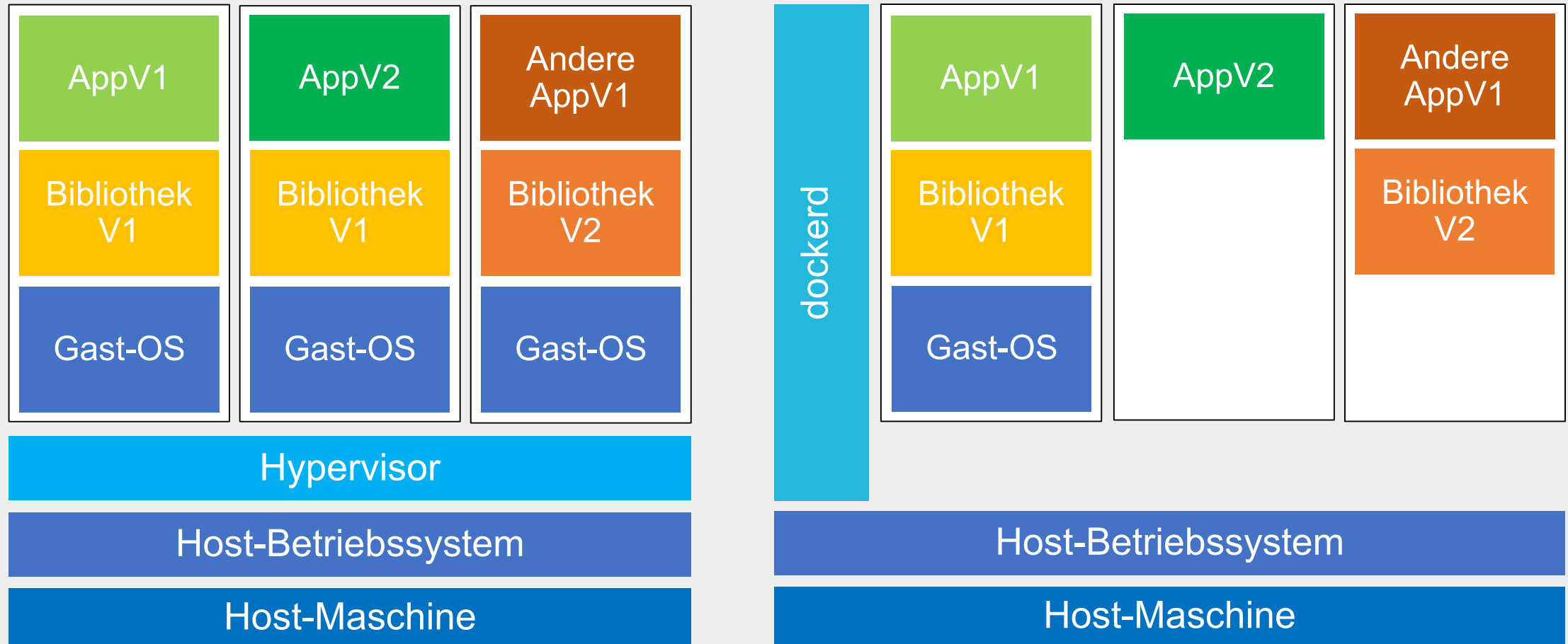
# Betrieb

## Virtuelle Maschinen vs. Container - zur Laufzeit



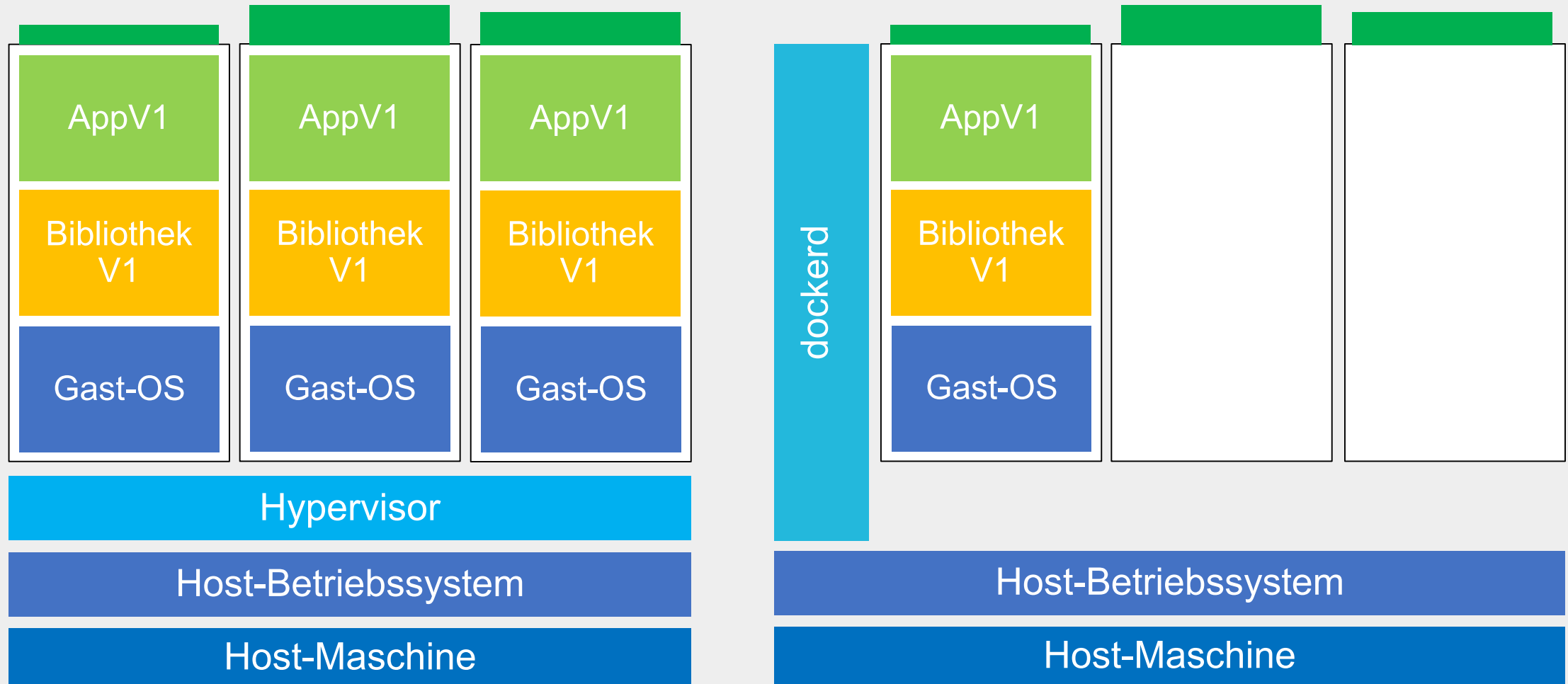
# Betrieb

## Virtuelle Maschinen vs. Container - Speicherbedarf Images



# Betrieb

## Virtuelle Maschinen vs. Container - Speicherbedarf Container





# Dockerfiles

# Steuerung mehrerer Container

## Vorteile durch Docker

- ▶ Mit Docker Compose über sehr einfache Definitionen möglich
- ▶ Skalierung in beschränktem Rahmen
- ▶ Für komplexere Orchestrierung Werkzeuge wie Docker Swarm oder Kubernetes



# Docker Compose

# Betrieb

## Vorteile durch Docker

- ▶ Deutlich **weniger Overhead** als in VMs, da kein Gast-Betriebssystem notwendig ist
- ▶ **Optimierte Ressourcennutzung** durch Layer-Technologie bei der Ablage von Images und im laufenden Betrieb
- ▶ **Alle** für den Betrieb **notwendigen Informationen** im Dockerfile bzw. Docker Image enthalten
  - Stabil **reproduzierbar** und einfach **aktualisierbar** auf allen Zielsystemen
- ▶ Möglichkeit für **Ressourcenbeschränkung**, auch wenn die Anwendung das nicht nativ unterstützt

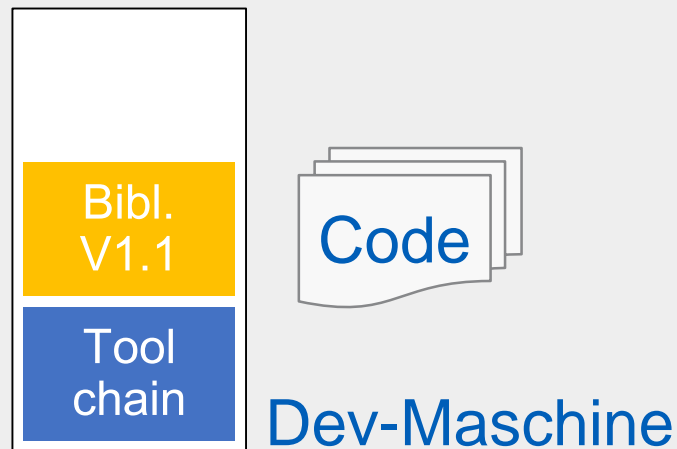


# Entwicklung

# Anwendungsentwicklung

## Ohne Docker

- Entwicklung und Probe-Build in lokaler Umgebung, Codeänderungen werden in ein Repository übertragen. Kontinuierlicher Build nimmt sie auf und erzeugt die Anwendung

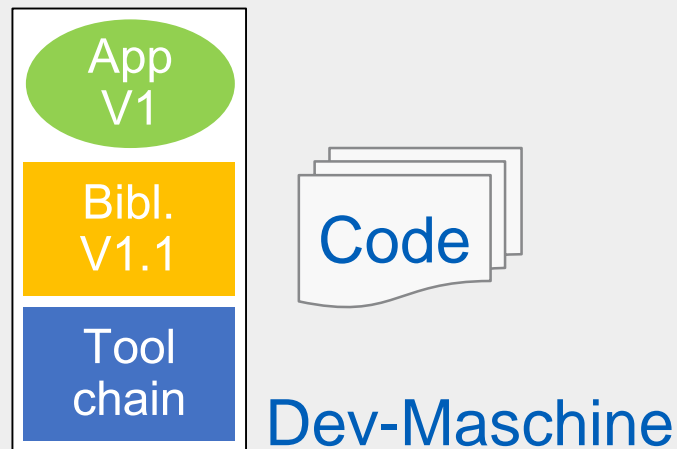




# Anwendungsentwicklung

## Ohne Docker

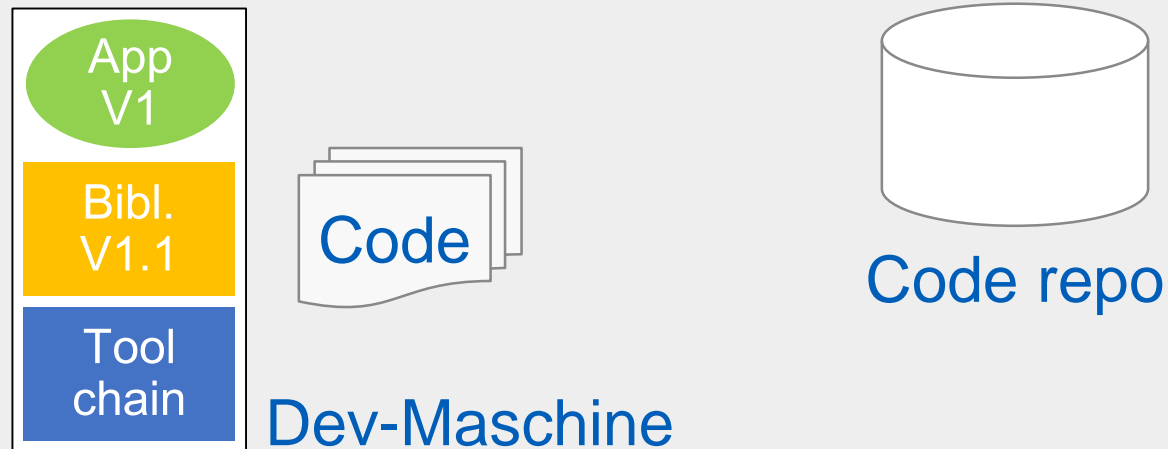
- Entwicklung und Probe-Build in lokaler Umgebung, Codeänderungen werden in ein Repository übertragen. Kontinuierlicher Build nimmt sie auf und erzeugt die Anwendung



# Anwendungsentwicklung

## Ohne Docker

- Entwicklung und Probe-Build in lokaler Umgebung, Codeänderungen werden in ein Repository übertragen. Kontinuierlicher Build nimmt sie auf und erzeugt die Anwendung

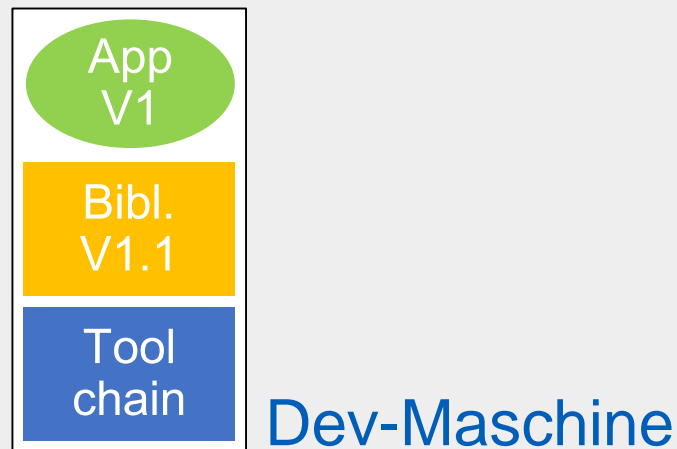




# Anwendungsentwicklung

## Ohne Docker

- Entwicklung und Probe-Build in lokaler Umgebung, Codeänderungen werden in ein Repository übertragen. Kontinuierlicher Build nimmt sie auf und erzeugt die Anwendung



# Anwendungsentwicklung

## Ohne Docker

- Entwicklung und Probe-Build in lokaler Umgebung, Codeänderungen werden in ein Repository übertragen. Kontinuierlicher Build nimmt sie auf und erzeugt die Anwendung



# Anwendungsentwicklung

## Ohne Docker

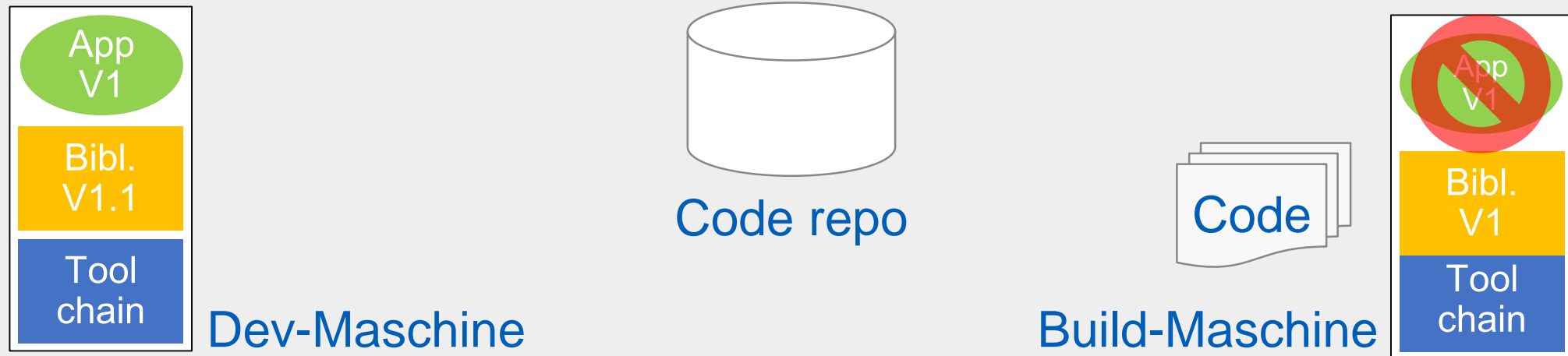
- Entwicklung und Probe-Build in lokaler Umgebung, Codeänderungen werden in ein Repository übertragen. Kontinuierlicher Build nimmt sie auf und erzeugt die Anwendung



# Anwendungsentwicklung

## Ohne Docker

- Entwicklung und Probe-Build in lokaler Umgebung, Codeänderungen werden in ein Repository übertragen. Kontinuierlicher Build nimmt sie auf und erzeugt die Anwendung

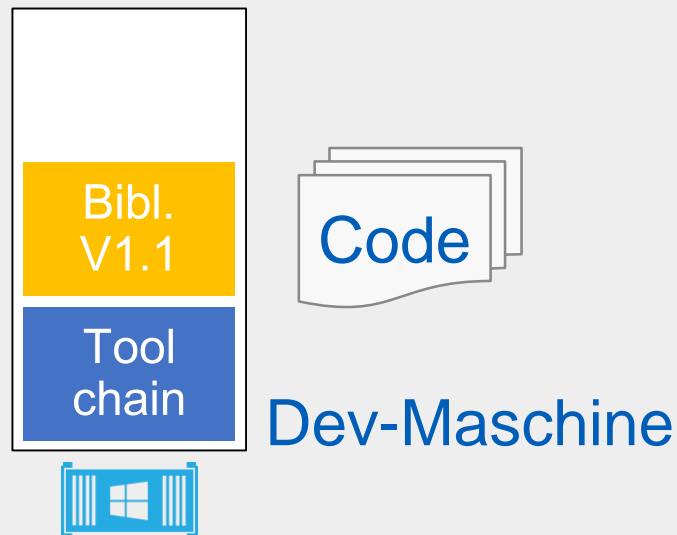


# Anwendungsentwicklung

## Mit Docker



- ▶ Dev- und Build-Maschine erzeugen die Anwendung in Containern, in denen alles identisch ist und daher identische Ergebnisse erzielen

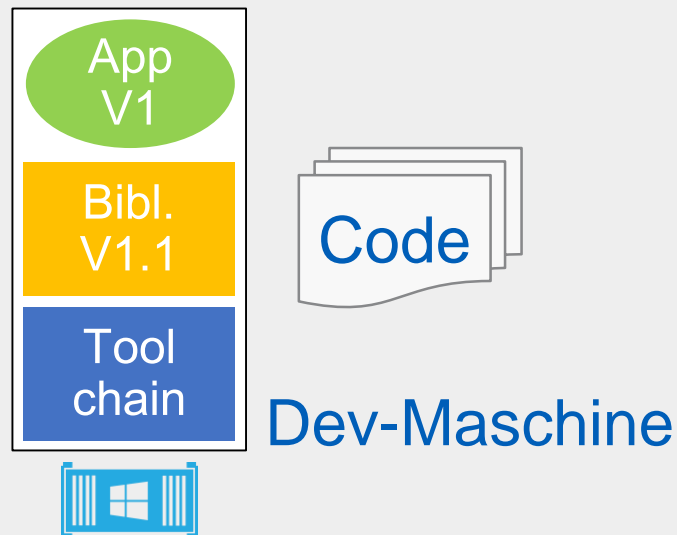


# Anwendungsentwicklung

## Mit Docker



- ▶ Dev- und Build-Maschine erzeugen die Anwendung in Containern, in denen alles identisch ist und daher identische Ergebnisse erzielen

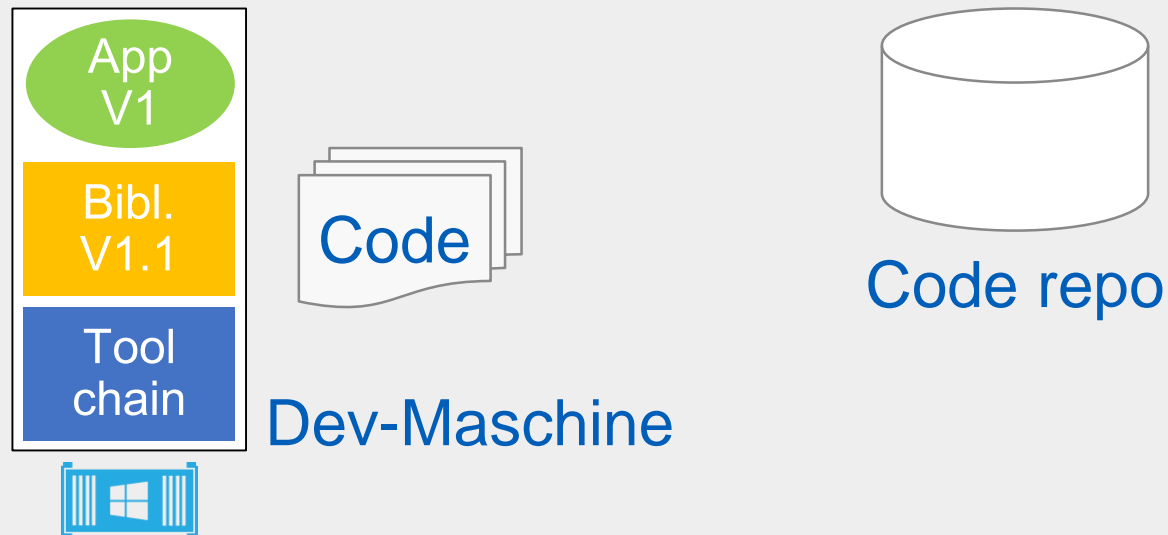


# Anwendungsentwicklung

## Mit Docker



- ▶ Dev- und Build-Maschine erzeugen die Anwendung in Containern, in denen alles identisch ist und daher identische Ergebnisse erzielen

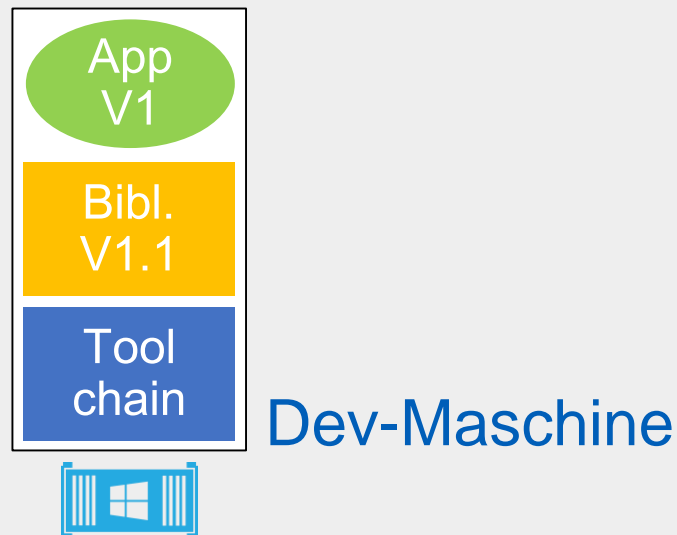


# Anwendungsentwicklung

## Mit Docker



- ▶ Dev- und Build-Maschine erzeugen die Anwendung in Containern, in denen alles identisch ist und daher identische Ergebnisse erzielen



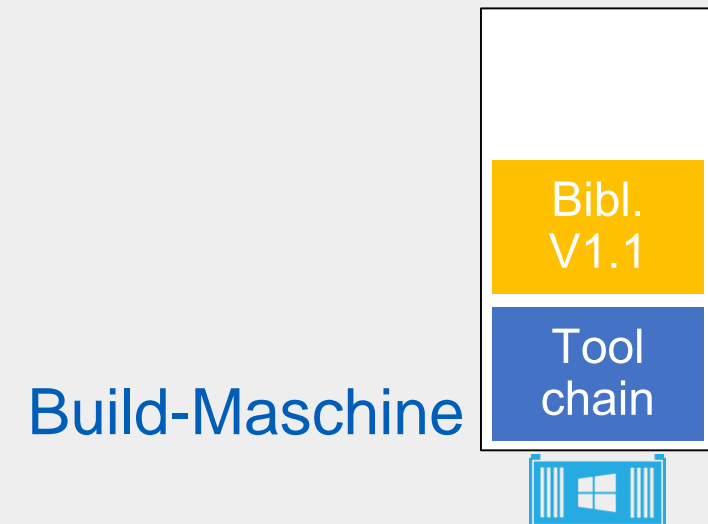
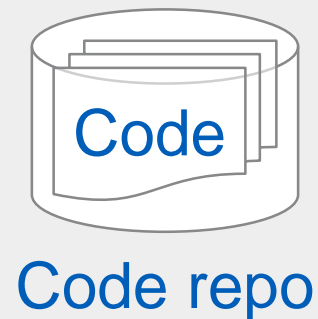
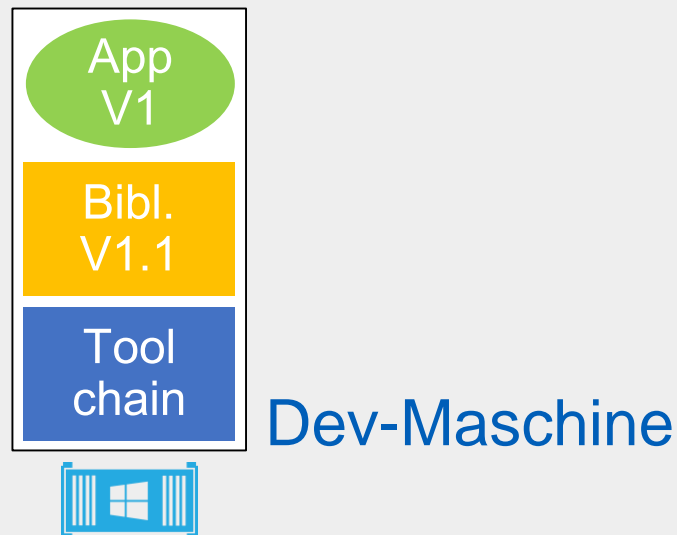


# Anwendungsentwicklung

## Mit Docker



- Dev- und Build-Maschine erzeugen die Anwendung in Containern, in denen alles identisch ist und daher identische Ergebnisse erzielen

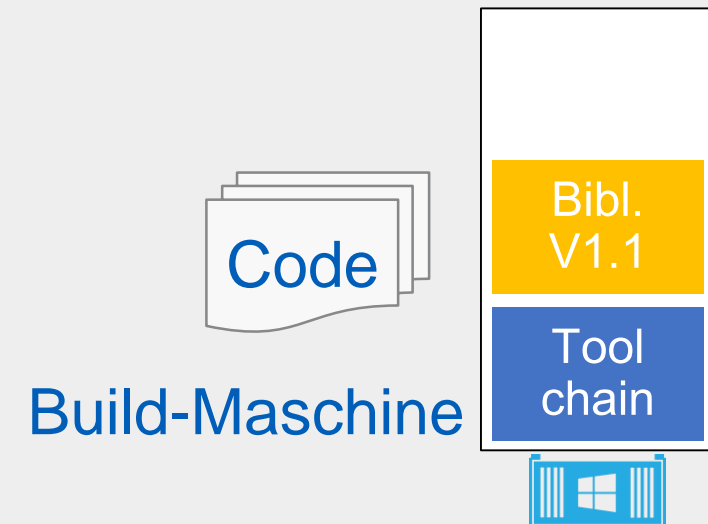
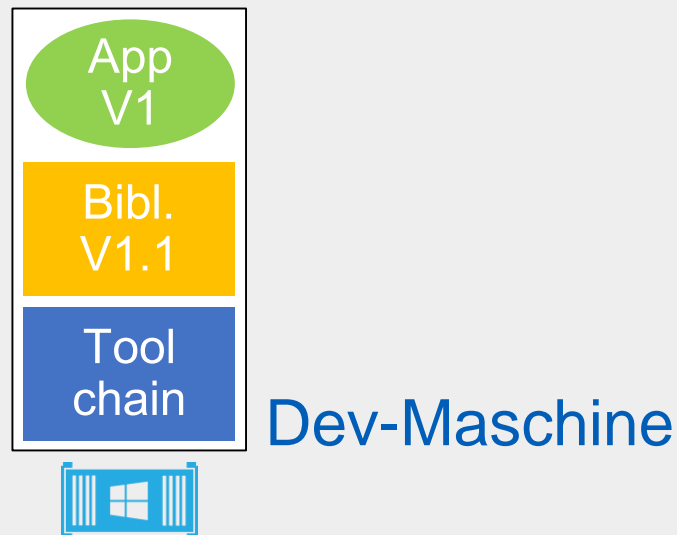


# Anwendungsentwicklung

## Mit Docker



- Dev- und Build-Maschine erzeugen die Anwendung in Containern, in denen alles identisch ist und daher identische Ergebnisse erzielen

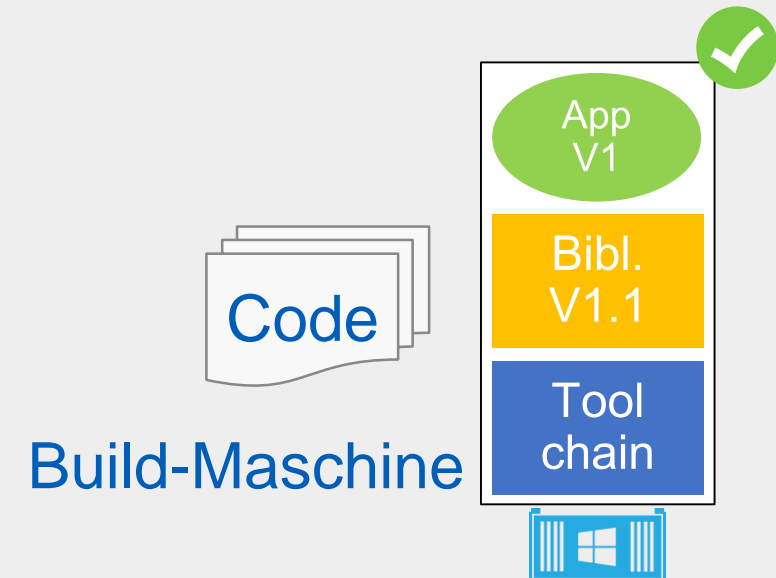
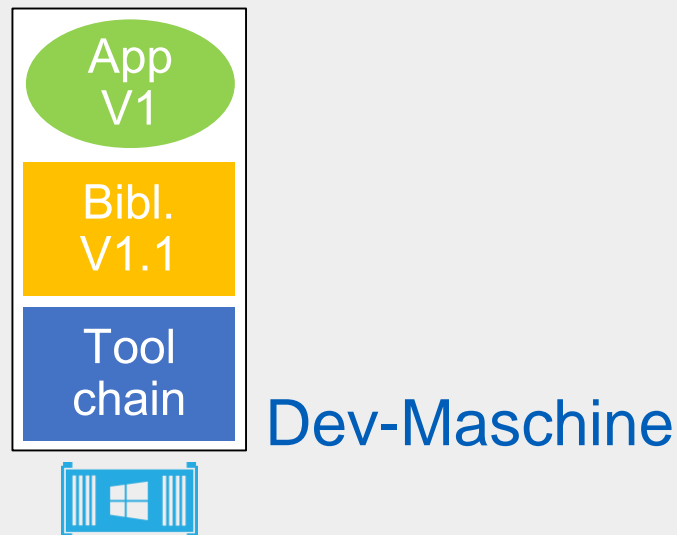


# Anwendungsentwicklung

## Mit Docker



- Dev- und Build-Maschine erzeugen die Anwendung in Containern, in denen alles identisch ist und daher identische Ergebnisse erzielen



# Dockerfiles




# Anwendungsentwicklung

## Vorteile durch Docker

- ▶ **Alle** für das Erzeugen einer Anwendung **notwendigen Informationen** und Tools im Dockerfile bzw. Docker Image enthalten
  - Stabil **reproduzierbar** und einfach **aktualisierbar** auf allen Entwicklungssystemen
  - Kein “bei mir funktioniert es”
- ▶ **Schneller, sauberer Wechsel** zwischen verschiedenen Toolsets, Versionen etc. (und zurück)
- ▶ Übergabe an Betriebs-Team deutlich **vereinfacht**





# Ökosystem



# Ökosystem

## Rund um Docker

- ▶ Mehr als **180.000 Images**, direkt zum Download (pull) verfügbar
  - Microsoft Container Registry für Microsoft-Images
- ▶ Sehr aktive **OpenSource Tools-Community** um Docker, auch kommerzieller Support
- ▶ Nutzen, anpassen, veröffentlichen
- ▶ Serverless **Ausführungsumgebung**, z.B. Azure Container Instances
- ▶ **Orchestrierungslösungen** wie Docker Swarm oder Kubernetes
  - Sehr großer Bereich mit extremem Potenzial für **hochverfügbare, dynamisch skalierende Anwendungen**

# Interessante Links und Personen

## Für Docker auf Windows

- ▶ <https://docs.microsoft.com/en-us/virtualization/windowscontainers/>
- ▶ <https://docs.docker.com/>
- ▶ [@Docker](#)
- ▶ [@EltonStoneman](#) (Dev Advocate bei Docker / Microsoft MVP)
- ▶ [@stefscherer](#) (Engineer bei Docker / Microsoft MVP)



Danke für die Aufmerksamkeit!

Welche Fragen darf ich beantworten?

tobias.fenster@axians-infoma.de  
+49 731 1551-964  
@tobiasfenster

# Danke an unsere Partner!

---

## Platinum Sponsor



## Gold Sponsoren



# Danke an unsere Partner!

---

## Gold Sponsoren

The logo for TAROX, featuring the word "TAROX" in a bold, dark blue, sans-serif font.The logo for Veeam, featuring the word "veeam" in a green, lowercase, sans-serif font.The logo for Lenovo, featuring the word "Lenovo" in white, bold, sans-serif font on a red rectangular background.The logo for Dell EMC, featuring the word "DELL" in blue and "EMC" in grey, with a stylized "E" in Dell's signature font.The logo for interxion, featuring the word "interxion" in a lowercase, black, sans-serif font.The logo for Mellanox Technologies, featuring a stylized white "M" on a blue background with the text "Mellanox TECHNOLOGIES" below it.The logo for SecureGuard, featuring the word "SecureGuard" in a stylized, italicized, dark blue font.

## Silber Sponsoren

The logo for Azure Stack Alliance, featuring a blue and orange icon followed by the text "Azure Stack ALLIANCE" in blue.The logo for PURE STORAGE, featuring an orange icon followed by the text "PURE STORAGE" in black.The logo for SquaredUp, featuring an orange and blue icon followed by the text "SquaredUp" in orange.