Cheatsheet Docker M347

# Lösungen für Probleme der Softwareentwicklung mit Docker

1. **Umgebungsinkonsistenzen:**

* Docker bietet eine konsistente Entwicklungsumgebung unabhängig von der zugrunde liegenden Infrastruktur.

1. **Abhängigkeitsmanagement:**

* Docker ermöglicht die Definition und Verwaltung von Anwendungsabhängigkeiten in Dockerfiles für konsistente Umgebungen.

1. **Skalierung und Bereitstellung:**

* Docker erleichtert die Skalierung und Bereitstellung von Anwendungen durch Container-Orchestrierungsplattformen wie Kubernetes oder Docker Swarm.

1. **Testing und CI/CD:**

* Docker vereinfacht das Testen und die Integration in CI/CD-Pipelines durch die Definition von Testumgebungen in Containern.

1. **Ressourcenmanagement:**

* Docker ermöglicht die effiziente Nutzung von Ressourcen durch die Isolierung von Containern und die gemeinsame Nutzung des Host-Betriebssystemkerns.

# Container

Ein Docker-Container ist eine isolierte Ausführungseinheit für Anwendungen und ihre Abhängigkeiten, die auf einem Docker-Image basiert. Er teilt den Kernel des Host-Betriebssystems, ist aber in Bezug auf das Dateisystem, Prozesse und Netzwerk isoliert. Container sind portabel, konsistent und ermöglichen eine effiziente Nutzung von Ressourcen.

# Unterschiede VM und Docker Container

1. **Virtual Machines (VMs):**

* Emulieren eine vollständige physische Maschine.
* Benötigen mehr Ressourcen und haben längere Startzeiten.
* Bieten höhere Isolation und Sicherheit.

1. **Docker Container:**

* Teilen den Host-Betriebssystemkern und sind leichtgewichtiger.
* Starten schnell und benötigen weniger Ressourcen.
* Sind portabler und einfach zu verwalten, aber bieten weniger Isolation als VMs.

# Virtualisierungsumgebungen

|  |  |
| --- | --- |
| 1. **Hypervisoren:**  * VMware vSphere/ESXi * Microsoft Hyper-V * KVM (Kernel-based Virtual Machine) * Xen | 1. **Containerisierungsplattformen:**  * Docker * Kubernetes * Docker Swarm * OpenShift |

# Unterschiede gutes und schlechtes Image

1. **Gutes Docker-Image:**

* Sicher: Verwendet vertrauenswürdige Quellen, durch Sicherheitsbewertungen überprüft.
* Effizient: Schlank und optimiert für minimale Größe und Ladezeiten.
* Wartbar: Regelmäßige Aktualisierung, transparente Konfiguration und Dokumentation.
* Portabel: Plattformunabhängig und konsistent in verschiedenen Umgebungen.

1. **Schlechtes Docker-Image:**

* Unsicher: Verwendet veraltete oder unsichere Komponenten, fehlende Sicherheitsmaßnahmen.
* Ineffizient: Übermäßig groß, enthält unnötige Ressourcen.
* Schlecht wartbar: Nicht gepflegt, unklare Konfiguration und Dokumentation.
* Unportabel: An eine bestimmte Umgebung gebunden, mangelnde Anpassungsfähigkeit.

# Docker Architektur

Skript S. **6** / 24

|  |  |
| --- | --- |
| Komponenten von Docker (Client-Server-Modell) Docker overview | Docker Docs | Image Schichten From Dockerfile to layers |