**Erklärung Container**

Ein Container ist eine Art virtuelle Maschine, die Anwendungen in einer isolierten Umgebung ausführt. Container teilen sich den Kernel des Host-Betriebssystems, was bedeutet, dass sie weniger Ressourcen benötigen als traditionelle virtuelle Maschinen und schneller gestartet werden können. Container sind eine wichtige Technologie in der Anwendungsbereitstellung und DevOps-Infrastruktur, da sie die Portabilität und Skalierbarkeit von Anwendungen verbessern.

**Erklärung Docker-Image**

Ein Docker-Image ist ein Paket mit allem, was benötigt wird, um eine Anwendung in einem Container auszuführen. Es enthält den Anwendungscode, alle Abhängigkeiten und Konfigurationsdateien. Docker-Images werden mit einem Dockerfile erstellt, das die Anweisungen enthält, die Docker verwenden soll, um das Image zu erstellen.

**Erklärung Dockerfile**

Ein Dockerfile ist eine Textdatei, die die Anweisungen enthält, die Docker verwenden soll, um ein Docker-Image zu erstellen. Es enthält Anweisungen wie "FROM", "RUN", "COPY" und "CMD", die Docker dazu anweisen, Abhängigkeiten herunterzuladen, Anwendungscode hinzuzufügen und den Container zu konfigurieren.

**Erklärung Docker-Container**

Ein Docker-Container ist eine laufende Instanz eines Docker-Images. Ein Docker-Container enthält den Anwendungscode und alle Abhängigkeiten, die im Docker-Image definiert sind. Docker-Container sind in der Regel kurzlebig, was bedeutet, dass sie erstellt, ausgeführt und dann verworfen werden können.

**Erklärung Docker-Registry**

Eine Docker-Registry ist ein zentraler Speicherort für Docker-Images. Es ist ein Ort, an dem Entwickler Docker-Images hochladen und herunterladen können. Die am häufigsten verwendete Docker-Registry ist Docker Hub, aber es gibt auch andere öffentliche und private Registries, die verwendet werden können.

**Unterschied Virtualisierung und Cloud**

Virtualisierung ist das Bereitstellen von Leistung eines physischen Servers zu mehreren virtuellen Servern. (Virtuelle Maschinen)

Cloud-Computing umfasst auch Dienste wie Software as a Service (SaaS), Platform as a Service (PaaS) und Infrastructure as a Service (IaaS) und der physische Server steht beim Anbieter. (Speicher, Rechenleistung, Anwendungen)

**Erklärung Portweiterleitung und Docker-Container verwenden**

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

**Erklärung Netzwerk**

* **Bridge-Netzwerk:** Standardmäßig werden Docker-Container in einem Bridge-Netzwerk erstellt, das es den Containern ermöglicht, miteinander zu kommunizieren, ohne dass sie öffentlich zugänglich sind. Der Host-System wird auch in dieses Netzwerk einbezogen. Jeder Container erhält eine IP-Adresse im Bridge-Netzwerk und kann auf andere Container im Netzwerk zugreifen, indem er die IP-Adresse oder den Container-Namen verwendet.
* **Host-Netzwerk:** Docker bietet auch die Option, Container direkt an das Host-Netzwerk anzuschließen, wodurch sie dieselbe Netzwerkumgebung wie das Host-System haben. Container, die dem Host-Netzwerk angeschlossen sind, haben Zugriff auf alle Netzwerkressourcen, die auf dem Host verfügbar sind.
* **Overlay-Netzwerk:** Overlay-Netzwerke ermöglichen es Containern, über mehrere Hosts hinweg miteinander zu kommunizieren. Dies ist nützlich für Anwendungen, die auf mehrere Hosts verteilt sind und die Interaktion zwischen Containern erfordern.
* **Custom-Netzwerk:** Docker ermöglicht es auch, benutzerdefinierte Netzwerke zu erstellen, die spezifisch für eine Anwendung oder Gruppe von Anwendungen sind. Diese Netzwerke können so konfiguriert werden, dass sie nur den erforderlichen Datenverkehr zulassen und die Sicherheit erhöhen.

**Wichtigste Befehle**

**Platzbedarf von Images und Containern ermitteln:**

"docker images -a" zeigt eine Liste aller Images an, die auf dem Host-System gespeichert sind, zusammen mit ihrer Größe.

"docker ps -a" zeigt eine Liste aller Container an, die auf dem Host-System ausgeführt werden, zusammen mit ihrem Status und ihrer Größe.

**Container und Images löschen:**

"docker rm" löscht einen oder mehrere Container von dem Host-System. Zum Beispiel: "docker rm container\_name".

"docker rmi" löscht ein oder mehrere Images von dem Host-System. Zum Beispiel: "docker rmi image\_name".

**Volumes verwalten:**

"docker volume ls" zeigt eine Liste aller Volumes an, die auf dem Host-System erstellt wurden.

"docker volume create" erstellt ein neues Volume.

"docker volume rm" löscht ein oder mehrere Volumes. Zum Beispiel: "docker volume rm volume\_name".

**Gesamtüberblick erhalten:**

"docker stats" zeigt Echtzeitinformationen zu CPU, Speicher und Netzwerkverbrauch von allen laufenden Containern an.

"docker system df" zeigt Informationen zu allen Images, Containern und Volumes an, die auf dem Host-System gespeichert sind.

**Ungenutzten Speicher freigeben:**

"docker system prune" entfernt alle nicht verwendeten Images, Container und Volumes von dem Host-System.

"docker image prune" entfernt alle nicht verwendeten Images von dem Host-System.

"docker container prune" entfernt alle nicht verwendeten Container von dem Host-System.

"docker volume prune" entfernt alle nicht verwendeten Volumes von dem Host-System.

docker network create Network\_Name

docker run -d (Dämon) --name (Custom Containername) --network (definierter netzwerkname) -e (Benötigte Enviroment Variablen)

-v myvolume:/var/lib/mysql (Pfad zu Speicherort) mariadb (Containername)

-p 8080:80 (Portweiterleitung)