


Praktische Projekte

Die folgenden praktischen Aufgaben können parallel zu den jeweiligen Kapitel der udemy Kurse "[learn-docker](#)" und "[CKAD](#)" gemacht werden.

Voraussetzungen

Docker und clusterfähiges K8s auf Deinem lokalen Gerät

 Bonus - für gewisse Bonusaufgaben kannst du natürlich gern zu einem späteren Zeitpunkt zu einer anderen Aufgabe zurückkehren! Am Besten kopierst Du die untenstehende Tabelle gleich auf Deine persönliche Seite, damit Du auch bei Aktualisierungen immer die ursprüngliche Aufgabenstellung hast.

Bezug zu Udeym Kurs, Kapitel (Reihenfolge analog udemy Kurse)	Idee / Auftrag / Übung	Links
learn-docker , Creating a new Docker Image	<p>Paktetiere dein eigenes Image, das auf dem hier basiert:</p> <p>google (php-redis): gcr.io/google_samples/gb-frontend:v5</p> <ul style="list-style-type: none">• Das Tag von Deinem neu gebauten Image soll 'v5-<DeinKürzel>' heissen• Das neu gebaute Image, soll die aktuellsten OS Updates/Patches beinhalten. Denn gemäss trivy hat das tag gb-frontend:v5 ziemlich viele (1759!) Lücken: <pre>trivy image gcr.io/google_samples/gb-frontend:v5 Total: 1759 (UNKNOWN: 32, LOW: 808, MEDIUM: 417, HIGH: 428, CRITICAL: 74)</pre> <p>Hinweis: Vielleicht möchtest Du herausfinden, was dieses Image genau beinhaltet. Hier könnte dive oder auch docker inspect gute Dienste leisten.</p> <p>Bonus: Das resultierte Image soll kleiner werden, Ideen was Du tun könntest?</p>	https://docs.docker.com/engine/reference/commandline/build/
learn-docker , Docker Run learn-docker , Environment Variables	<ol style="list-style-type: none">1. Zeige den Inhalt der Environment Variable (ENV) '<i>PHP_INI_DIR</i>' von deinem neu gebauten Image an, ohne das Image in einem Container zu starten.2. starte einen Container aus Deinem Image und setze die ENV '<i>SET_MANUAL=true</i>' ; stoppe und lösche den Container wieder.3. setze eine neue Environment Variable '<i>PRAKTIKUM=true</i>', diesmal fix im gebauten Image (also via build des Image).4. starte einen Container aus Deinem Image und zeige den Inhalt vom '<i>PHP_INI_DIR</i>' sowie '<i>PRAKTIKUM</i>' im laufenden Container an.	https://docs.docker.com/engine/reference/commandline/run/

learn-docker , Command vs Entrypoint	<p>Schau dir das gb-frontend Image nochmal genauer an. Fokussiere dich dabei auf Entrypoint und Cmd.</p> <ol style="list-style-type: none"> 1. Was ist generell der Unterschied zwischen den beiden Operationen. Wann eignet sich welche? 2. Starte einen Container und überschreibe den Befehl welcher dabei ausgeführt wird. Welche Möglichkeiten gibt es hier und was stellst du fest? 3. Erstelle ein neues Image, z.B. basierend auf Debian:latest. Adde zunächst einfach eine ENTRYPOINT Instruction (<i>echo eignet sich für die Visualisierung der Unterschiede</i>) <ol style="list-style-type: none"> a. Was wenn du den Befehl nun versuchst zu überschreiben beim Starten eines Containers? b. Was geschieht, wenn du ENTRYPOINT und CMD setzt? c. Setze jetzt mal nur CMD. Was stellst du fest? 4. Betrachtet du die verwendete Option beim gb-frontend als sinnvoll? Warum /Warum nicht? 	<p>https://docs.docker.com/engine/reference/builder/#entrypoint</p> <p>https://docs.docker.com/engine/reference/builder/#cmd</p>
learn-docker , Registry	<p>Starte lokal eine Registry und pushe dein selbst gebautes Image darauf.</p> <p>Diese Registry (resp. das Image da drauf) wird in zukünftigen Lektionen immer wieder genutzt werden.</p> <hr/> <p>Bonus: Erstelle für die Registry ein Docker-Compose file</p>	<p>https://docs.docker.com/registry/</p> <p>bzw.</p> <p>https://docs.docker.com/registry/deploying/</p>
learn-docker , Docker Compose	<ol style="list-style-type: none"> 1. Erstelle eine Docker-Compose Datei, die sowohl <u>dein Image</u> als auch Redis beinhaltet. (Redis soll mindestens aus zwei Komponenten bestehen: einem Leader (Image: docker.io/redis:6.0.5) und einen Follower (Image: gcr.io/google_samples/gb-redis-follower:v2). 2. Sorge dafür das dein Image (das modifizierte gb-frontend) sauber startet und lokal via Browser aufgerufen werden kann. <p>Bonus:</p> <ol style="list-style-type: none"> 1. verifiziere auch, dass redis aus deinem gb-frontend heraus ansprechbar ist. 	<p>https://docs.docker.com/compose/compose-file/compose-file-v3/</p>
learn-docker , Storage	<ol style="list-style-type: none"> 1. starte deinen Frontend Container und nimm eine Änderung im index.html vor 2. rufe deine App auf. Kannst du deine Änderung sehen? 3. Start den Container neu, kannst du deine Änderung nun sehen? <p>Deine Änderungen sind nach dem Restart weg. Im nächsten abschnitt geht es darum wie wir solche Änderungen dauerhaft machen können</p> <ol style="list-style-type: none"> 1. kopiere die <u>index.html</u> aus dem gb-frontend:v5 auf deine Workstation 2. mache eine Anpassung (z.B. am Titel oder den Farben) 3. starte einen neuen Container und mounte die angepasste Datei nach /var/www/html/... 4. Kannst du deine Änderungen in der App sehen? 5. Passe dasselbe File direkt im Container an und schau was mit der lokalen Version passiert 6. Verhindere, dass zur Runtime des Containers Änderungen an dem gemounteten index.html vorgenommen werden können <hr/> <p>Bonus: tue dasselbe im docker-compose</p> <ol style="list-style-type: none"> 1. Definiere einen eigenen Pfad für die persistenten Daten vom Redis Container 	<p>https://docs.docker.com/storage/volumes/</p>

learn-docker , Docker Networking	<p>Vorbereitung:</p> <p>Erstelle ein docker-compose in dem das Frontend mit dem redis "spricht". Das heisst: wenn Du in der App einen Guestbook Eintrag erstellst, sollte der in redis gespeichert werden (und im Browser entsprechend angezeigt werden).</p> <p>Da das Debuggen der App hier nicht die Hauptaufgabe ist 😊 Hinweis:</p> <p>Die Redis Hosts müssen bestimmte Namen haben, damit die App automatisch darauf connecten kann:</p> <p>Der Redis Leader host muss "redis-leader" heissen und der Follower muss "redis-follower" heissen. Wenn die Guestbook Einträge nicht gespeichert werden können, liegt es vermutlich daran, dass Du die services im docker-compose nicht so genannt hast.</p> <p>mehrere "Netze":</p> <ol style="list-style-type: none"> 1. Erstelle 2 unterschiedliche Subnetzen 2. Das Frontend soll aus Subnetz 1 erreichbar sein und Redis darf nur im Subnetz 2 sein 3. Stell sicher, dass das Frontend Redis erreicht. 	
Security	<p>Nimm dir 3 Stunden Zeit um dich in potentielle Sicherheitsaspekte im Zusammenhang mit Docker Container einzulesen. Notiere dir im Anschluss deine Erkenntnisse.</p>	
Docker Milestone → zeige das bisher erarbeitete Deinen Lehrlingsbetreuer.		
CKAD -> noch in Arbeit...		