Kurs DevOps

Lista 2

22 i 23 października 2025

Jako że zazwyczaj w dużych grupach znajduje się co najmniej jeden użytkownik Windowsa, to informuje się, iż zadania należy wykonywać z linii poleceń, o ile nie powiedziano inaczej w treści.

Zadanie 1.
Pokaż, w jaki sposób skonfigurować proxy pełniące rolę cache dla `pacman`/`apt`. Czemu chcielibyśmy to robić? Pomyśl o sytuacji, gdy uruchamiasz CI bazujący na obrazie maszyny wirtualnej, w którym jednym z kroków jest aktualizacja paczek systemowych do najnowszej wersji.

Zademonstruj, że Twoja konfiguracja działa (np. z użyciem maszyny wirtualnej). Niech Twój host robi za cache/proxy, a maszyna niech spróbuje zaktualizować pakiety. Pokaż, że pakiety brakujące w cache hosta są ściągane, zapisywane i udostępniane maszynie wirtualnej bez błędów.

Zadanie 2.
Stwórz obraz dockera w którym:

- Zainstalujesz przydatne Ci programy oraz serwer `ssh` i `iproute2`
- Dodasz swój klucz publiczny ssh, tak by można było się z jego pomocą logować na root-a
- Włączysz logowanie po `ssh` na root
- Automatycznie przy uruchomieniu kontenera uruchomisz serwer ssh

Po uruchomieniu kontenera, powinno być możliwe zalogowanie się z hosta po ssh, bez wykonywania dodatkowych czynności.

Następnie uruchom docker w trybie interaktywnym i pokaż, że jesteś w stanie używać basha wewnątrz dockera. Wyjdź z kontenera. Co się z nim stało? Czy został usunięty? Co pokazuje `docker ps`? A `docker ps --all`?

Zadanie 3.
Zademonstruj użycie multistage builds z dockera rozdzielając środowisko, w którym budowana jest aplikacja C/C++ od środowiska, w którym jest wykonywana.

Zadanie 4.
Zapoznaj się z `docker compose` i przygotuj jego krótkie omówienie. Następnie wykorzystaj go do uruchomienia kilku (2-4) instancji tego samego obrazu na raz.

Zadanie 5.
Przeanalizuj poniższe fragmenty Dockerfile i wyjaśnij, jakie nieoczekiwane zachowania mogą w nich wystąpić.

```
FROM ubuntu:22.04
RUN apt-get update
RUN apt-get install -y curl
```

```
FROM alpine
ENV ADMIN_USER="mark"
RUN echo $ADMIN_USER > ./mark
RUN unset ADMIN_USER
```

Zadanie 6.
Pokaż w jaki sposób podczas budowania obrazu dockerowego, bezpiecznie wygenerować certyfikat kryptograficzny i go podpisać przy użyciu klucza prywatnego znajdującego się na hoście.

Zadanie 7.

- W jakich sytuacjach możemy chcieć używać `--cpuset-cpus` w dockerze?
- Wyjaśnij jak działa w dockerze cache budowania obrazów, a następnie zademonstruj i wyjaśnij działanie poleceń: `docker image ls --all`, `docker image history`, `docker image inspect`

Zadanie 8.
Zapoznaj się i przedstaw podsystem `bake` dockera[1]. Z jakiego powodu moglibyśmy chcieć go wykorzystywać? Zademonstruj, w jaki sposób zbudować przykładowy obraz z pomocą tego podsystemu.

Zadanie 9.
Poniżej dołączono dockerfile opisujący obraz do zbudowania oraz skrypt `smok.sh` używany w tym pliku. Kontener utworzony na podstawie powstałego obrazu powinien:

- Uruchomić skrypt z katalogu `/home/smok` (i wyświetlić jako wynik liczby od 1 do 10).
- Działać jako użytkownik `smok`.

Jednakże coś nie działa. Pokaż, co dokładnie jest problemem i zademonstruj jak poprawić dockerfile.

```
FROM ubuntu:latest

RUN groupadd -r smok && useradd --no-log-init -r -g smok smok
COPY smok.sh /home/smok/smok.sh

USER smok
CMD ./smok.sh
```

```
-------------------------------------
```

```
#!/bin/bash
for i in {1..10}; do
  echo $i;
  sleep 0.5;
done
```

Zadanie 10.
Jakie mamy możliwości optymalizacji rozmiaru obrazu Dockera? Co się stanie z rozmiarem obrazu, jeśli dodasz bardzo duży plik, a następnie go usuniesz? Jak sobie z tym poradzić?

```
RUN dd if=/dev/urandom of=duzyPlik bs=1M count=100
RUN rm duzyPlik
```

[1] `https://docs.docker.com/build/bake/`