Kurs DevOps

Lista 3

29 i 30 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.
Zapoznaj się z `docker network`. Utwórz sieć dockera typu `bridge` i podłącz do niej 2 kontenery. Pokaż, że kontenery te są w stanie się komunikować. Czy są w stanie połączyć się z Internetem? Jak to zablokować?

Zadanie 2.
Przygotuj obraz dockerowy, uruchamiający `miniserve`[1], dostępny poprzez adres IP hosta i wybrany przez Ciebie numer portu. Pliki udostępniane przez serwer powinny pochodzić z dwóch źródeł a) z nieulotnego wolumenu b) katalogu współdzielonego z hostem. Zademonstruj działanie Twojego kontenera.

Zadanie 3.
Postaraj się zademonstrować, w jaki sposób z użyciem dockera można zaatakować hosta. W tym celu na maszynie wirtualnej stwórz plik z flagą (jak w CTF-ach), który będzie należał do roota i z uprawnieniami `0400`, natomiast zwykłego użytkownika dodaj do grupy `docker` (przyznając mu tym samym uprawnienia do używania dockera). Stwórz i uruchom obraz w dockerze w ten sposób, by odczytać utworzony przez Ciebie plik.

Zadanie 4.
W systemie przetwarzającym pewne dane (archiwum z plikami zamieszczone jest na SKOS-ie) działała wersja X oprogramowania. Następnie przygotowano aktualizację do wersji Y, jednakże okazało się, że drugie uruchomienie kontenera zakończyło się błędem. W związku z tym uruchomiono z powrotem dockera używającego starszego obrazu (wersja X). Jednakże pomimo tego, że wcześniej wersja X działała, to teraz zwraca analogiczny błąd jak wersja Y. Wyjaśnij, co się stało.

- `docker build -f Dockerfile-x -t lista3zadA:x .`
- `docker build -f Dockerfile-y -t lista3zadA:y .`
- `docker run -v lista3zA:/root/katalog -it --rm lista3zadA:x` OK
- `docker run -v lista3zA:/root/katalog -it --rm lista3zadA:x` OK
- `docker run -v lista3zA:/root/katalog -it --rm lista3zadA:x` OK
- `docker run -v lista3zA:/root/katalog -it --rm lista3zadA:y` OK
- `docker run -v lista3zA:/root/katalog -it --rm lista3zadA:y` BŁĄD
- `docker run -v lista3zA:/root/katalog -it --rm lista3zadA:x` BŁĄD

Zadanie 5.
W jaki sposób sprawdzić, czy cgrupy są dostępne na Twoim komputerze i w jakiej wersji? Dla wybranego procesu sprawdź, do jakiej cgrupy należy poprzez procfs. Następnie w sysfs sprawdź jakie są ograniczenia nałożone na tę grupę. Jaki serwis utworzył tę grupę i przypisał do niej proces?

[1] `https://github.com/svenstaro/miniserve`

Przy pomocy sysfs i procfs: stwórz własną cgrupę, przypisz tam wybrany proces i zademonstruj konfigurację ograniczeń zużycia zasobów. Pokaż i omów zawartość przykładowego pliku `cgroup.stat`.

Zadanie 6.
Czym jest kontroler w cgroup? Przedstaw krótkie omówienie najciekawszych Twoim zdaniem ograniczeń, które można nałożyć z użyciem kontrolerów.

Z czego wynika różnica w zawartości: `/sys/fs/cgroup/cgroup.controllers` oraz `/sys/fs/cgroup/user.slice/user-1000.slice/cgroup.controllers` (lub jego odpowiednika)?

Zadanie 7.
Czym są slice w systemd? Zademonstruj jak utworzyć przykładowy slice, dodać do niego dwa procesy i następnie ustawić ograniczenie na zużycie pamięci i obciążenie CPU. Co pokazują polecenia `systemd-cgtop` i `systemd-cgls`?

Zadanie 8.
Pokaż, do jakich cgroup należy uruchomiony przez Ciebie kontener. Ogranicz zasoby dla danej cgrupy (np. czas procesora) i pokaż, że rzeczywiście jest to respektowane. Czym jest cgroup driver zwracany przez `docker info`?

Uruchom instancję obrazu dockera z limitem pamięci (`docker run --memory`/`docker run --memory-reservation`) i zademonstruj, co się stanie, gdy spróbujesz go przekroczyć. Czym się różni limit twardy od miękkiego? W jaki sposób aplikowane są te limity?