Systemy operacyjne 12

Docker

1. Instalacja dockera

1. **Pobranie hasha do instalacji**

   ```
   curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
   ```

2. **Dodanie pakietu**

   ```
   add-apt-repository \
     "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
     $(lsb_release -cs) \
     stable"
   ```

3. **Aktualizacja pakietów**

   ```
   apt update
   ```

4. **Instalacja**

   ```
   apt install -yf docker-ce docker-ce-cli containerd.io
   ```

5. **Dodanie użytkownika do grupy docker**

   ```
   usermod -aG docker nazwa_użytkownika
   ```

2. Instalacja docker-compose

Composer to przydatne narzędzie, które pozwala na definiowanie i uruchamianie wielokontenerowych aplikacji Dockera. W łatwy sposób możemy ustawić wszystkie potrzebne parametry, volumeny oraz relacje pomiędzy poszczególnymi kontenerami naszej aplikacji. Następnie przy użyciu jednej komendy uruchomimy wszystkie serwisy, które zdefiniowaliśmy w pliku konfiguracyjnym. Instalacja Composera dla Ubuntu jest bardzo prosta. Wystarczy nam jedna komenda:

1. **Pobranie pakietu**

   ```
   curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
   ```

2. **Ustawienie globalnego dostępu**

   ```
   sudo chmod +x /usr/local/bin/docker-compose
   ```

3. Co należy wiedzieć przed rozpoczęciem

Każde narzędzie jakiego używamy wymaga od nas pewnego minimum wiedzy. W przypadku Dockera, musimy nauczyć się, czym jest kontener, volumen, obraz i zmienne środowiskowe.

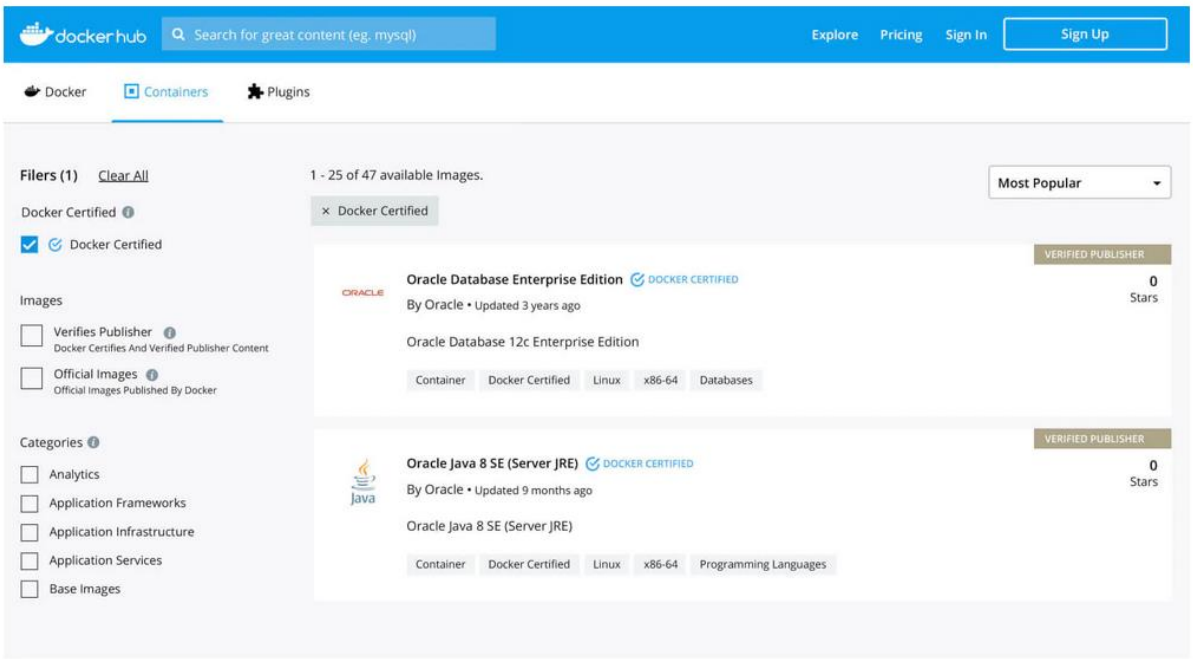
- **obraz (image)** – to nic innego, jak lekki i samodzielny pakiet potrzebnych do uruchomienia oprogramowania bibliotek, zmiennych, kodu oraz plików konfiguracyjnych. Przykładem, może być tutaj „phpdockerio/nginx:latest", który posiada wszystko, co jest niezbędne do uruchomienia Nginxa,
- **kontener (container)** – jest to uruchomiona instancja obrazu. Działa on całkowicie niezależnie od środowiska hosta, mając dostęp jedynie do plików oraz portów,
- **volumen (volume)** – służy do współdzielenia plików pomiędzy hostem a kontenerem (oraz pomiędzy kontenerami). Używa się go także do trzymania poza kontenerem, w momencie jego usuwania,
- **zmienne środowiskowe (environment)** – są to zmienne, jakie przekazujemy do kontenera w momencie jego budowy np. ustawienie danych do bazy danych MySQL

4. Rejestr docker

Chociaż obrazy Docker są łatwe w budowie i programiści uwielbiają ich prostotę oraz przenośność, szybko odkryto, że zarządzanie tysiącami obrazów Docker jest bardzo trudnym zadaniem. Rejestr Docker jest odpowiedzią na to wyzwanie. Rejestr Docker to standardowy sposób przechowywania i rozpowszechniania obrazów Docker. Rejestr jest repozytorium open source korzystającym z otwartej licencji Apache.

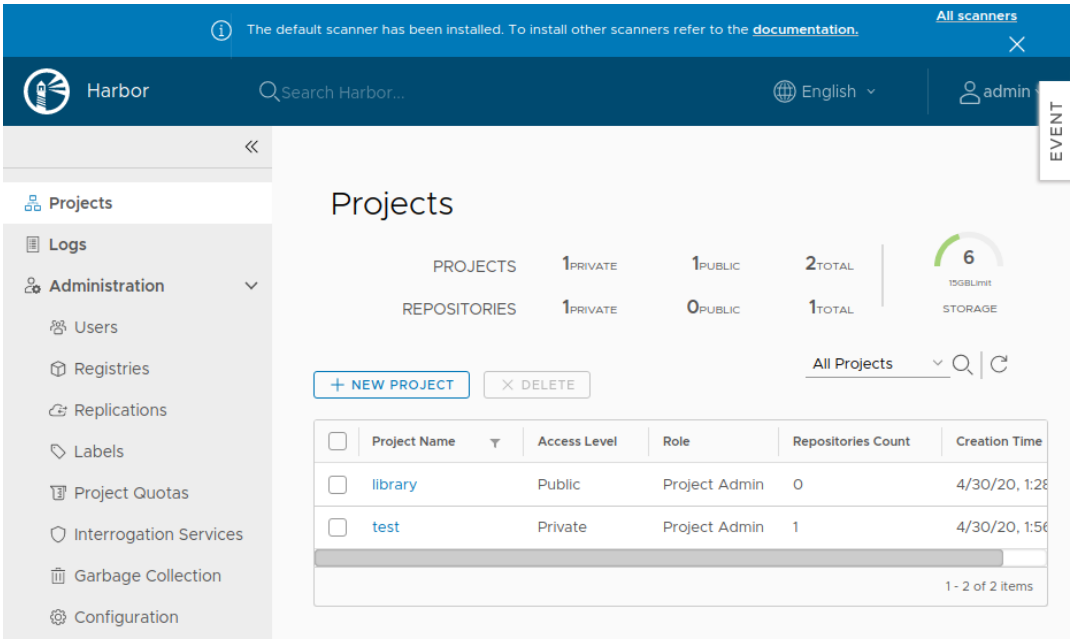
Rejestr Docker pomaga także usprawnić kontrolę dostępu oraz zwiększyć bezpieczeństwo obrazów Docker przechowywanych w repozytorium. Zarządza dystrybucją obrazów, a także może integrować się z procesami tworzenia aplikacji. Programiści mogą skonfigurować własny rejestr Docker lub skorzystać z hostowanych usług, takich jak Docker Hub, Oracle Container Registry, Azure Container Registry itp.

Podczas uruchamiania kontenera, Docker domyślnie automatycznie ściągnie odpowiadający mu obraz z dostępnego publicznie repozytorium Docker Hub, jeśli nie jest on dostępny lokalnie. Ponadto można tworzyć własne obrazy i wysyłać je do Docker Hub do repozytorium publicznego lub prywatnego.



5. Prywatne repozytorium – harbor

Harbor to rejestr typu open source, który trzyma nasze zbudowane obrazy. Jest systemem w pełni zarządzanym tzn. możemy określić role użytkowników, tworzyć osobne projekty na obrazy dockerowe, automatycznie sprawdzać zabezpieczenia.



6. Przydatne polecenia

1. Pobieranie obrazu

   docker pull naza_obrazu:tag

2. Lista działających kontenerów

   docker ps

3. Lista wszystkich kontenerów

   docker ps –a

4. Sprawdzenie ustawień kontenera

   docker inspect nazwa_kontenera/id_kontenera

5. Lista obrazów

   docker images

6. Kasowanie obrazu

   docker rmi nazwa_obrazu/id_obrazu

7. Kasowanie kontenera

   docker rm nazwa_kontenera/id_kontenera

8. Kasowanie wszystkich obrazów

   docker rmi $(docker images)

9. Czyszczenie woluminów

   docker volume prune

10. Czyszczenie całego stacku dockerowego

    docker system prune

11. Uruchomienie z pliku docker-compose

    docker-compose up/ docker-compose up -d

12. Budowanie obrazu

    docker build -t nazwa_obrazu:tag .

13. Pushowanie obrazu do rejestru

    docker push nazwa_obrazu:tag

14. Commitówanie kontenerów

    docker commit id_kontenera nazwa_kontenera:tag

15. Wejście na kontener

    docker exec -it id_kontenera/nazwa_kontenera /bin/bash

16. Kopiowanie zawartości do kontenera

    docker cp lokalny_plik id_kontenera:~/lokazlizacj

7. Dockerfile

1. Klonowanie repo

   git clone https://github.com/docker/getting-started.git

2. Przygotowanie Dockerfile

```
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

3. Budowanie obrazu

   docker build -t getting-started .

4. Odpalenie obrazu

   docker run -dp 3000:3000 getting-started

8. Docker-compose

Przykład docker-compose.yml

```yaml
version: "3.8"
services:
  aplikacja:
    tty: true
    build:
      context: .
      dockerfile: ./Dockerfile.aplikacja
    volumes:
      - .:/usr/app/
      - /usr/app/node_modules
  szperacz:
    tty: true
    build:
      context: .
      dockerfile: ./Dockerfile.szperacz
    depends_on:
      - aplikacja
```

Uruchomienie bazy danych

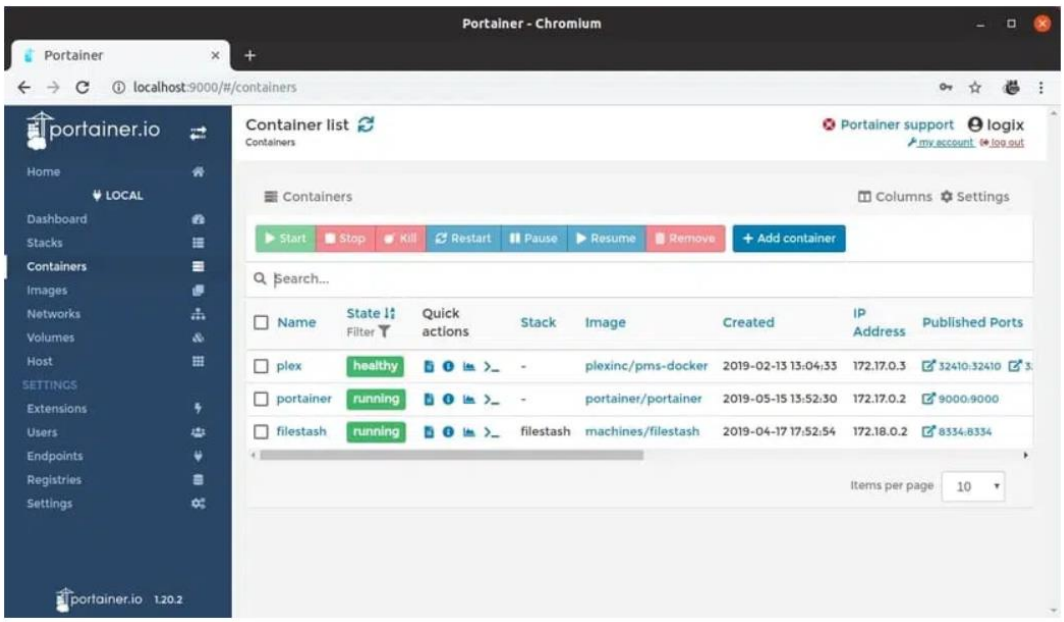
```
docker-compose up -d
```

```yaml
# Use root/example as user/password credentials
version: '3.1'

services:

  db:
    image: mysql
    # NOTE: use of "mysql_native_password" is not recommended: https://dev.mysql.com/doc/refman/8.0
    # (this is just an example, not intended to be a production configuration)
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: example

  adminer:
    image: adminer
    restart: always
    ports:
      - 8080:8080
```

9. Oprogramowanie do zarządzania dockerami

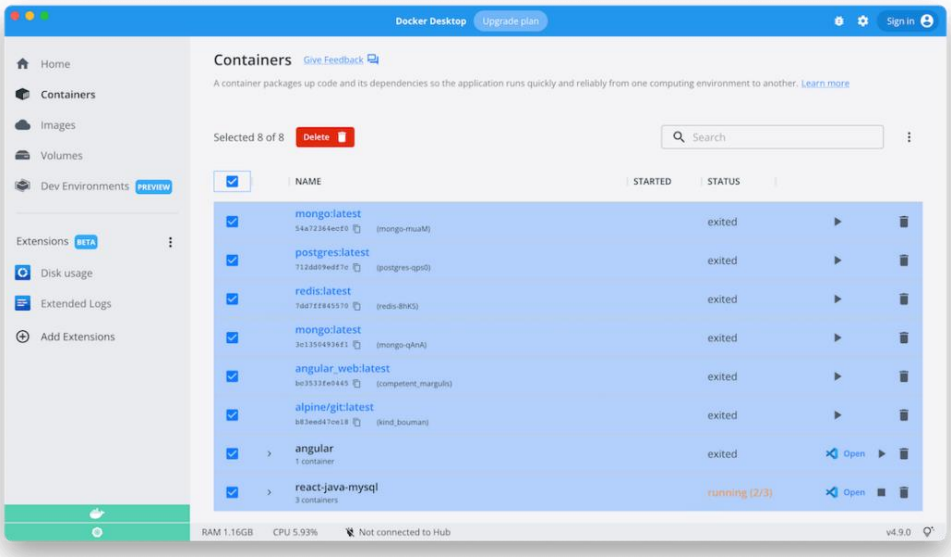
1. Portainer

   Jest to interfejs graficzny, z którego można zarządzać i pracować z kontenerami i Dockerem w znacznie bardziej intuicyjny i prosty sposób. Ten GUI jest oparty na sieci, więc może być używany na wielu platformach.



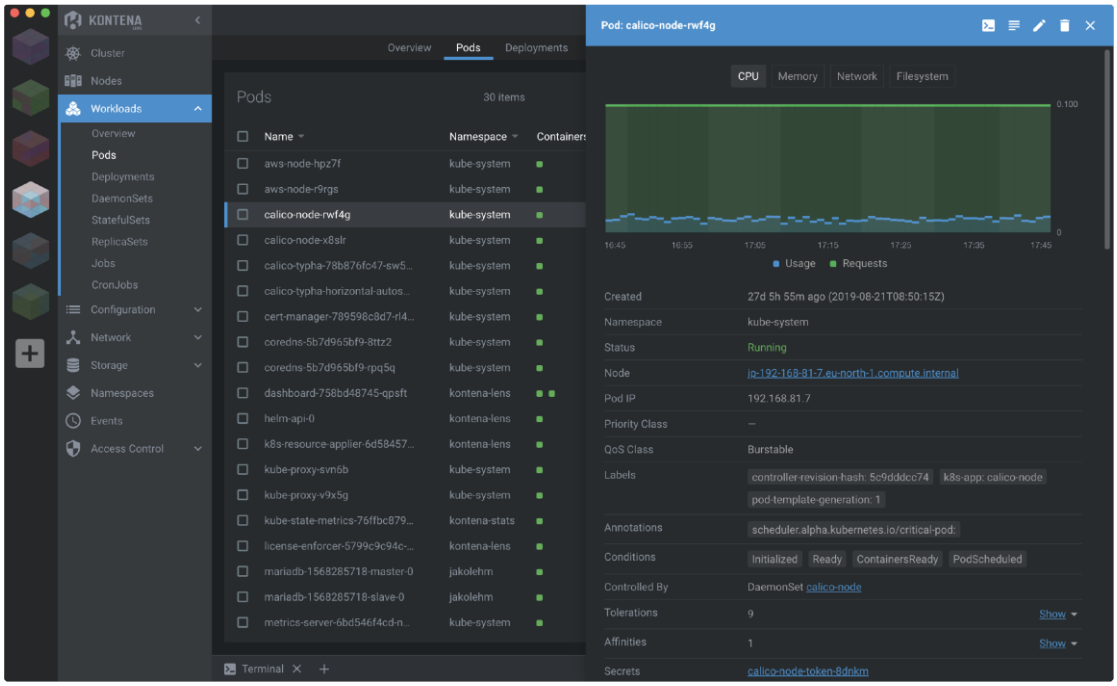
2. Docker-desktop

   Docker Desktop to aplikacja instalowana jednym kliknięciem dla środowiska Mac, Linux lub Windows, która umożliwia tworzenie i udostępnianie kontenerowych aplikacji i mikrousług.



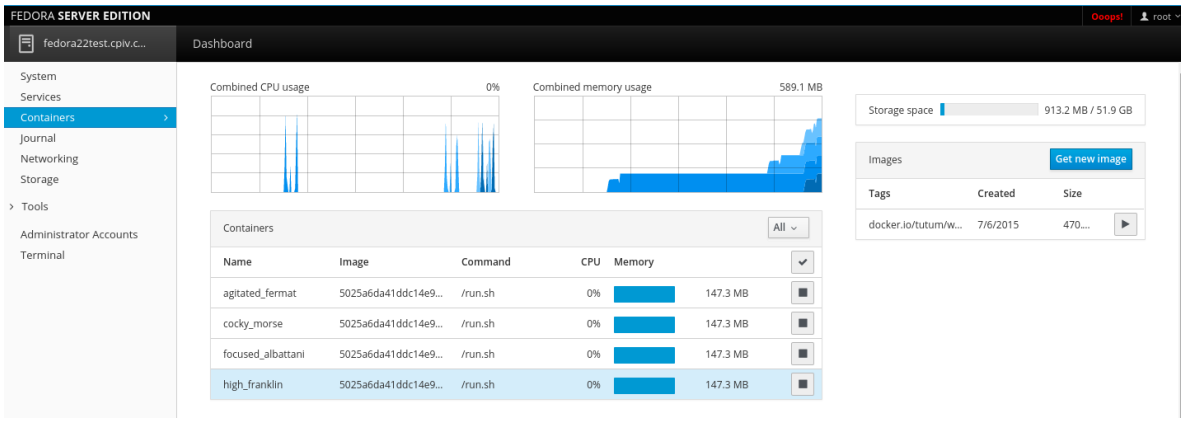
3. Lens

Lens zapewnia swoim użytkownikom prosty graficzny interfejs użytkownika, który zapewnia niezbędne narzędzia i zasoby do łatwego zarządzania i monitorowania wielu klastrów Kubernetes, aplikacji i wszystkich znajdujących się w nich zasobów.



4. Cocpit

Cockpit ułatwia administrowanie serwerami Linux za pośrednictwem przeglądarki internetowej ale także po zainstalowaniu odpowiedniej wtyczki cocpit-docker możliwe jest zarządzanie dockerami.



Zadania do wykonania

1. Przygotuj plik docker-compose.yml ze wstępną konfiguracją bazy
2. Uruchom dockera z bazą danych zrób kopie bazę danych, skasuj bazę i wgraj kopie bazy danych, którą wcześniej utowrzyłeś
3. ENTRYPOINT vs CMD
4. ADD vs COPY