Docker dla zdesperowanych

Ten dokument jest przeznaczony dla osób, które nie mają pojęcia o Dockerze, a czeka ich zaliczenie przedmiotu 'Biopython'.

> "Do not go where the path may lead, go instead where there is no path and leave a trail."
>
> — Ralph Waldo Emerson

Komendy Dockerowe

Ogólne

- `docker --version` - pokazuje wersję Dockera zainstalowaną na komputerze
- `docker info` - pokazuje informacje o Dockerze
- `docker help` - wypisuje wszystkie dostępne komendy Dockera (pomoc)

Obrazy

- `docker images` - wypisuje wszystkie dostępne obrazy Dockera na komputerze
- `docker pull <nazwa_obrazu>` - ściąga obraz z Docker Huba
- `docker build -t <nazwa_obrazu>:<tag> .` - buduje obraz z pliku Dockerfile w aktualnym katalogu. Tag jest opcjonalny i służy do oznaczenia wersji obrazu, jeśli nie podamy tagu, Docker sam nadaje mu wartość 'latest' (najnowsza wersja) oraz jeśli chcesz zbudować obraz z innego katalogu, to zamiast kropki na końcu ścieżki podaj ścieżkę do katalogu, np. `docker build -t <nazwa_obrazu>:<tag> /ścieżka/do/katalogu`
- `docker rmi <nazwa_obrazu>` - usuwa obraz z komputera

Kontenery

- `docker ps` - wypisuje wszystkie uruchomione kontenery
- `docker ps -a` - wypisuje wszystkie kontenery (uruchomione i zatrzymane)
- `docker run <nazwa_obrazu>` - uruchamia kontener na podstawie obrazu
- `docker run -it <nazwa_obrazu> bash` - uruchamia kontener z interaktywnym trybem i dostępem do terminala
- `docker start <id_kontenera>` - uruchamia zatrzymany kontener
- `docker stop <id_kontenera>` - zatrzymuje uruchomiony kontener
- `docker restart <id_kontenera>` - restartuje kontener
- `docker rm <id_kontenera>` - usuwa zatrzymany kontener

- `docker logs <id_kontenera>` - wypisuje logi kontenera
- `docker exec -it <id_kontenera> <komenda>` - wykonuje komendę w kontenerze
- `docker attach <id_kontenera>` - podłącza terminal do kontenera
- `docker cp <ścieżka/do/lokalnego/pliku> <id_kontenera>:<ścieżka/do/pliku/w/kontenerze>` - kopiowanie pliku z lokalnego komputera do kontenera
- `docker rename <stara_nazwa> <nowa_nazwa>` - zmienia nazwę kontenera

Volumes (Wolumeny)

- `docker volume ls` - wypisuje wszystkie dostępne wolumeny
- `docker volume create <nazwa_wolumenu>` - tworzy nowy wolumen
- `docker run -v <nazwa_wolumenu>:<ścieżka/w/kontenerze> <nazwa_obrazu>` - uruchamia kontener z wolumenem
- `docker volume rm <nazwa_wolumenu>` - usuwa wolumen

Sieć

- `docker network ls` - wypisuje wszystkie dostępne sieci
- `docker network create <nazwa_sieci>` - tworzy nową sieć
- `docker network connect <nazwa_sieci> <id_kontenera>` - łączy kontener z siecią
- `docker network disconnect <nazwa_sieci> <id_kontenera>` - rozłącza kontener z siecią
- `docker inspect <id_kontenera>` - pokazuje szczegółowe informacje o kontenerze

Docker Compose

- `docker-compose up` - uruchamia kontenery zdefiniowane w pliku docker-compose.yml
- `docker-compose down` - zatrzymuje kontenery zdefiniowane w pliku docker-compose.yml
- `docker-compose ps` - wypisuje wszystkie kontenery zdefiniowane w pliku docker-compose.yml

Czyszczenie

- `docker system prune` - usuwa wszystkie zatrzymane kontenery, wolumeny i sieci
- `docker container prune` - usuwa wszystkie zatrzymane kontenery
- `docker volume prune` - usuwa wszystkie wolumeny
- `docker image prune` - usuwa wszystkie obrazy, które nie są używane

Zapisywanie i wczytywanie obrazów

- `docker save -o <ścieżka/do/pliku>.tar <nazwa_obrazu>` - zapisuje obraz do pliku
- `docker load -i <ścieżka/do/pliku>.tar` - wczytuje obraz z pliku
- `docker export -o <ścieżka/do/pliku>.tar <id_kontenera>` - zapisuje kontener do pliku tar
- `docker import <ścieżka/do/pliku>.tar <nowy_obraz>` - wczytuje obraz z pliku tar

Publikowanie obrazów

- `docker login` - logowanie do Docker Huba
- `docker tag <nazwa_obrazu> <repozytorium>:<tag>` - tagowanie obrazu
- `docker push <repozytorium>:<tag>` - publikowanie obrazu na Docker Hubie

Dockerfile

Dockerfile to plik, który zawiera instrukcje do zbudowania obrazu Dockera. Poniżej znajduje się przykładowy plik Dockerfile:

```
# website::tag::1:: Build a simple Docker image that contains a text
file with the contents "Hello, World!"
FROM ubuntu:18.04
RUN echo 'Hello, World!' > /test.txt
```

Opiszę teraz najbardziej popularne instrukcje używane w pliku Dockerfile:

- `FROM <bazowy_obraz>` - określa obraz bazowy, na którym będziemy budować nasz obraz
- `RUN <komenda>` - wykonuje komendę w obrazie podczas budowania. Przykład: `RUN apt-get update && apt-get install -y wget` - aktualizuje listę pakietów i instaluje wget
- `COPY <lokalny_plik> <ścieżka_w_obrazie>` - kopiuje plik z lokalnego komputera do obrazu
- `ADD <plik> <ścieżka>` - kopiuje plik z lokalnego komputera do obrazu, a także pozwala na kopiowanie plików z adresów URL np. `ADD http://example.com/file.txt /file.txt`
- `WORKDIR <ścieżka>` - ustawia katalog roboczy dla kolejnych instrukcji
- `CMD ["komenda", "argument1", "argument2"]` - określa komendę, która ma zostać wykonana po uruchomieniu kontenera. Można podać argumenty jako tablicę
- `ENTRYPOINT ["komenda", "argument1", "argument2"]` - określa komendę, która ma zostać wykonana po uruchomieniu kontenera. Można podać argumenty jako tablicę, ale można je nadpisać podając

argumenty przy uruchamianiu kontenera. Ta komenda zawsze zostanie wykonana przy starcie. Można łączyć z CMD.

Przykład:

```
ENTRYPOINT ["python3"]
CMD ["app.py"]
```

- `ENV <zmienna>=<wartość>` - ustawia zmienną środowiskową w obrazie
- `EXPOSE <port>` - określa port, który ma być eksponowany z kontenera na zewnątrz
- `VOLUME <ścieżka>` - tworzy punkt montowania w kontenerze
- `ARG <nazwa>` - definiuje argument, który może być przekazany podczas budowania obrazu
- `USER <użytkownik>` - ustawia użytkownika, który ma zostać użyty podczas uruchamiania kontenera
- `LABEL <klucz>=<wartość>` - dodaje metadane do obrazu np. wersja, autor, opis itp.
- `SHELL ["shell", "-c"]` - określa powłokę, która ma zostać użyta do wykonywania poleceń w RUN, CMD, ENTRYPOINT

Inne, więcej informacji

Więcej informacji na temat Dockera można znaleźć na oficjalnej stronie: https://docs.docker.com/