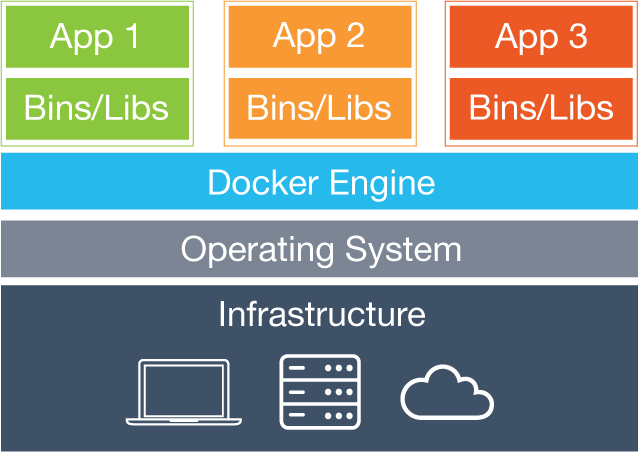
Rozproszone systemy operacyjne

Projekt

Grupa G

**Opis rozwiązania Docker**

Docker to zestaw narzędzi umożliwiających pracę z kontenerami w systemie operacyjnym Linux udostępniany na zasadach Open-Source. Kontenery są dostępne w jądrze Linuxa już od pewnego czasu, ale dopiero za sprawą Dockera, zaczęły być popularne oraz wykorzystywane przez programistów, w szczególności do wygodnego udostępniania aplikacji, wraz z jej wszystkimi zależnościami (wszelkimi bibliotekami i narzędziami z których korzysta). Można za jego pomocą tworzyć lekkie środowiska wirtualne, które nie posiadają części odpowiedzialnej za wirtualizację sprzętu.



Docker składa się z narzędzi:

* Docker Engine – służy do tworzenia, zarządzania i uruchamiania kontenerów.
* Docker Hub – służy do dzielenia się obrazami kontenerów między ich użytkownikami (obrazy mogą być dostępne zarówno publicznie jak i tylko dla ograniczonej grupy odbiorów)

Za pomocą narzędzia Docker Engine możemy przygotować obrazy kontenerów, które następnie będą uruchamiane na konkretnych kontenerach. Obraz przygotowuje się wykorzystując specjalny plik Dockerfile:

FROM python:2.7

ADD . /code

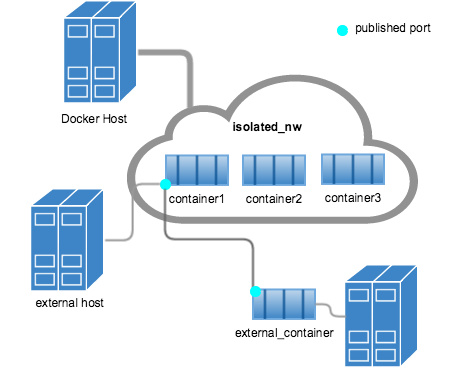
WORKDIR /code

RUN pip install -r requirements.txt

CMD python app.py

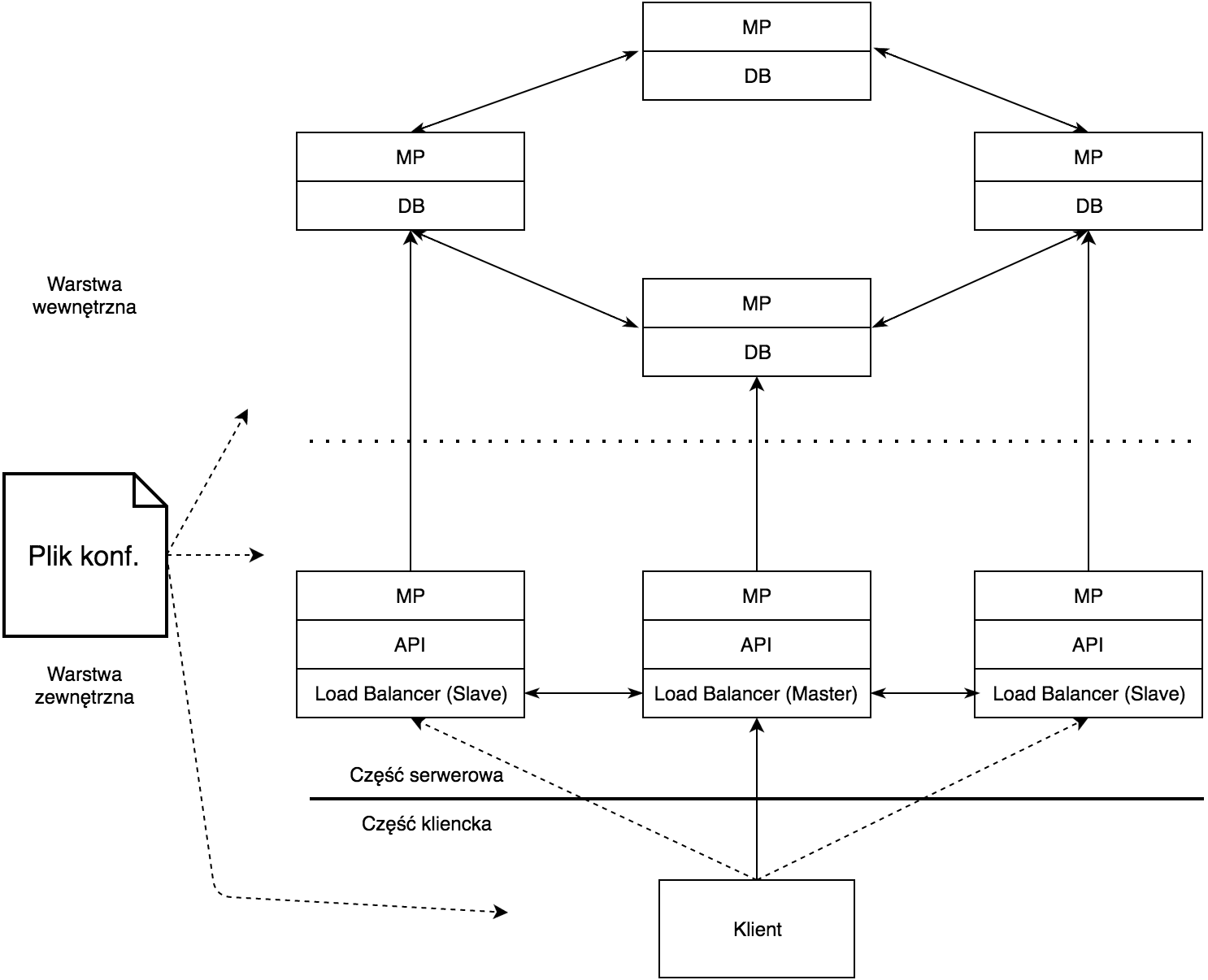
W pliku tym określamy na podstawie jakiego obrazu budujemy nasz obraz oraz mamy do dyspozycji różne instrukcje, dodające do naszego obrazu jakieś pliki, bądź wywołujące polecenia. Każda instrukcja stanowi tak zwaną warstwę, po wywołaniu której Docker Engine zapamiętuje jakie zmiany zaszły w obrazie. Dzięki temu po dokonaniu zmiany w naszym obrazie, do naszych współpracowników bądź na platformę DockerHub musimy wysłać tylko paczkę zawierającą tą zmianę, co znacznie przyśpiesza pracę z obrazami kontenerów.

Przygotowane obrazy można następnie uruchamiać na kontenerach. Kontenery działają w odseparowanym środowisku i można się z nimi komunikować na przykład wykorzystując Sockety. Domyślnie wszystkie uruchomione kontenery znajdują się w jednej wirtualnej sieci, ale mamy możliwość zmapowania portu konkretnego kontenera, z portem fizycznej maszyny na której kontenery są uruchamiane:



**Narzedzie Docker Compose**

**Koncepcja i architektura rozwiązania**



Rysunek - Architektura systemu

Główne elementy aplikacji:

* Część kliencka:
  + Aplikacja kliencka
* Część serwerowa:
  + Warstwa zewnętrzna:
    - Moduł Load Balancer
    - Moduł obsługi zapytań do interfejsu programistycznego (API)
  + Warstwa wewnętrzna:
    - Moduł rozproszonej bazy danych (DB)
  + Elementy wspólne dla warstwy wewnętrznej i zewnętrznej:
    - Moduł nadzorcy węzłów (MP)
* Elementy wspólne dla części serwerowej i klienckiej:
  + Moduł interpretujący wspólny plik konfiguracyjny

**Wymagania funkcjonalne i niefunkcjonalne**

**Harmonogram realizacji projektu**

**Organizacja środowiska programistycznego**

**Zadania do wykonania:**

**Aplikacja kliencka**

**Obsługa zapytań od klientów**

**-Równoważenie obciążenia (Load Balancing)**

**-Buforowanie i nadzór nad wykonaniem zapytań**

**Moduł obsługi zapytań do interfejsu programistycznego (API)**

**Moduł rozproszonej bazy danych klucz-wartość (DB)**

**-Obsługa interfejsu programistycznego bazy danych**

**-Nadzorca spójności danych**

**Moduł nadzorcy węzłów (MP)**

**Moduł interpretujący wspólny plik konfiguracyjny**

**Mechanizm uwierzytelniania węzłów w części serwerowej**

**Zestaw testów**

**Skrypt prezentujący działanie aplikacji**

**Konfiguracja narzędzi Docker i Docker Compose**