Seção 8 - Comunicação entre containers

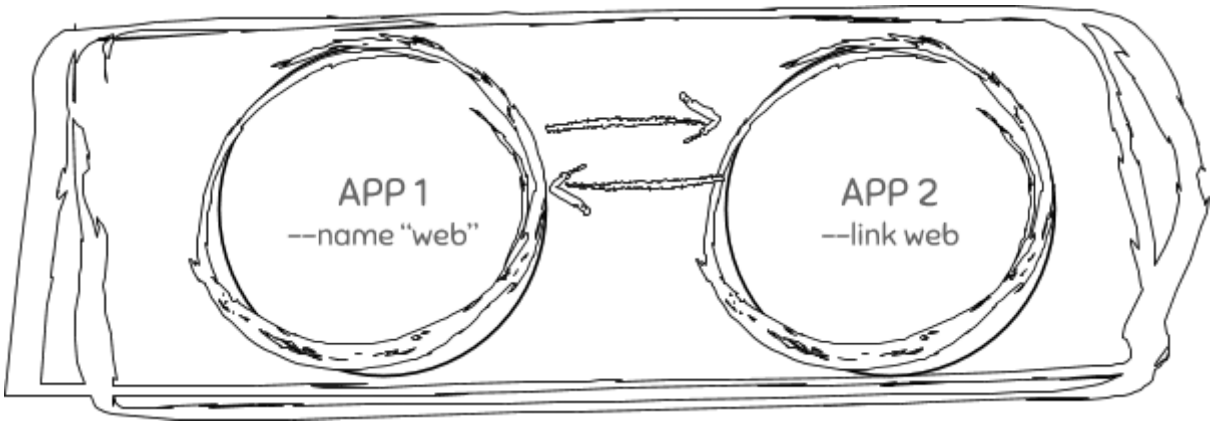
Comunicação entre containers

- Usado para, de maneira segura, transferir dados de um para outro

Métodos

- Linking (Legacy)
- User-defined networks (Docker Network)

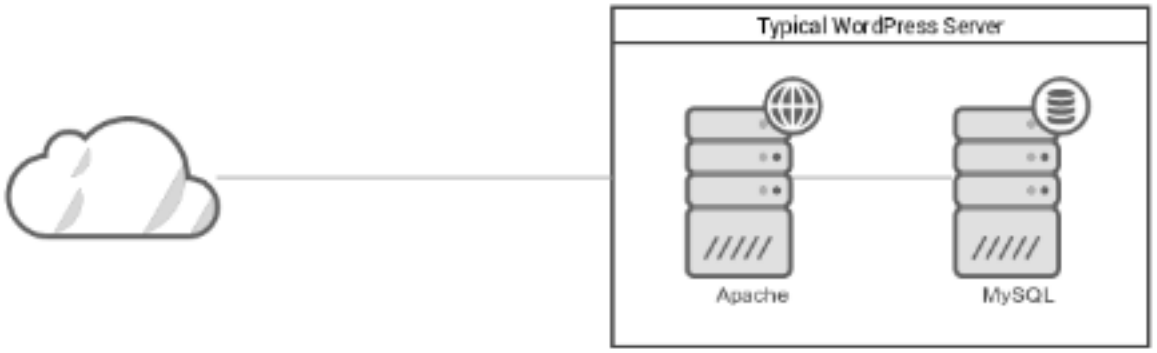
1. Linking



+ **Simples**
- **Obsoleto - novas versões docker podem não mais suportar**

- Container de origem tem acesso aos dados do container de destino;
- "Permite conexão entre containers através dos seus respectivos nomes;

- Essencial para disponibilizar microserviços:
    - Container com servidor web Apache;
    - Container com base de dados MySQL;



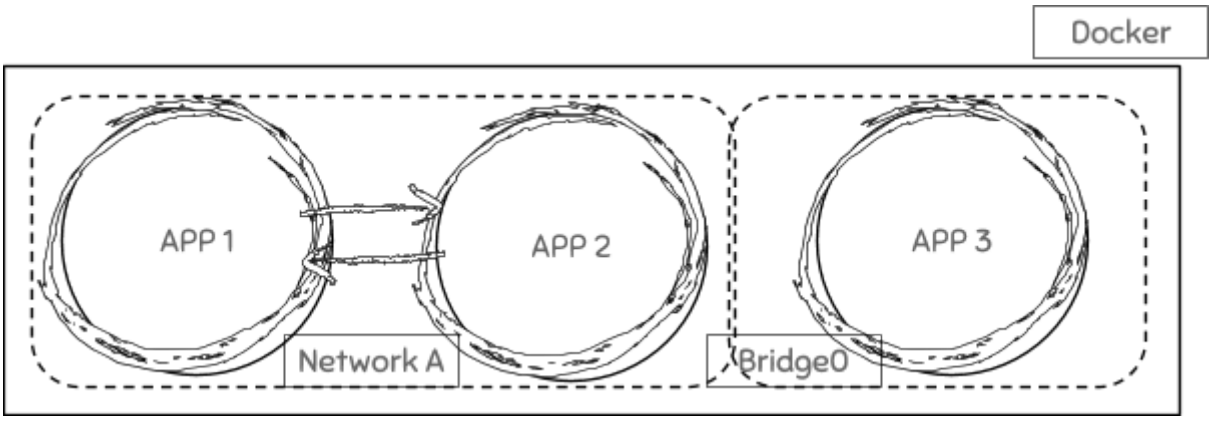
1.1 Criar o container de destino

```
# docker run -d --name database -e
MYSQL_ROOT_PASSWORD=123 -e MYSQL_DATABASE=teste
-e MYSQL_USER=user -e MYSQL_PASSWORD=pass mysql:5.5
```

1.2 Criar o container de origem usando o parâmetro --link

```
# docker run -d -p 8080:80 --name php --link database:db
-v /home/ricardo/site:/var/www/html php:5.6-apache
```

2. User-defined networks (Docker Network)



+ **Seguro - rede “isolada” para comunicação dos containers**
- **“Complexo” - uso de redes bridges criadas pelo próprio usuário**

- “Sucessor” do método “linking”;
- Usuário pode criar diversas “redes próprias”;
- Permite a comunicação entre containers através dos seus respectivos nomes;

2.1 Criar a nova rede docker

```
# docker network create redeA
```

2.2 Criar o container usando o parâmetro --network

```
# docker run -itd --name ubuntu --network=redeA ubuntu
# docker run -itd --name ubuntu2 --network=redeA ubuntu
```

"Múltiplos containers conectados"

Criar um container ubuntu com nome ubuntu1:

sudo docker run -itd --name ubuntu1 ubuntu:14.04

** Observe que esse container rodará em segundo plano*

Criar um segundo container fazendo 'linking' com o primeiro:

sudo docker run -itd --name ubuntu2 --link ubuntu1:cont1 ubuntu:14.04

** Observe que o container de destino foi "apelidado" de cont1.*

Acessar o terminal do segundo container

sudo docker exec -it ubuntu2 bash

ping cont1
exit

** Observe que o comando 'ping' deve ser executado dentro do container. E 'cont1' representa o container de nome ubuntu1.*

Verifique os arquivo hosts do container de origem (ubuntu2):

sudo docker exec ubuntu2 cat /etc/hosts

** Observe que o resultado do comando exibirá o ip do container ubuntu1 associado ao nome 'cont1' - seu apelido*

Crie uma rede bridge "própria":

sudo docker network create minharede

** Ao criar esse rede você poderá colocar todos os containers que desejar manter conexões seguras*

Visualize a criação da rede:

sudo docker network ls
sudo docker network inspect minharede

** Veja que o comando network ls exibe todas as redes existentes e o comando network inspect detalhe as informações da rede criada*

Crie dois containers de imagens debian e coloque eles na rede criada:

docker run -itd --name debian1 --network=minharede debian
docker run -itd --name debian2 --network=minharede debian

Acesse o container de nome ubuntu1 e teste uma conexão com o container debian2:

docker exec -it ubuntu1 bash

ping debian2
exit

** Observe que você não conseguiu êxito no ping. Porque o container ubuntu1 não faz parte da rede criada*

Acesse o container de nome debian1 e teste uma conexão com o container debian2:

docker exec -it debian1 bash

ping debian2
exit

** Observe que dessa vez conseguiu êxito no ping. Porque o container debian1 faz parte da rede criada*

Remova a rede e containers criados

sudo docker stop ubuntu1
sudo docker stop ubuntu2
sudo docker stop debian1

```
sudo docker stop debian2
sudo docker rm ubuntu1
sudo docker rm ubuntu2
sudo docker rm debian1
sudo docker rm debian2
sudo docker network rm minharede
```

Referências

- https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/
- https://docs.docker.com/engine/userguide/networking/work-with-networks/

- https://docs.docker.com/engine/userguide/networking/