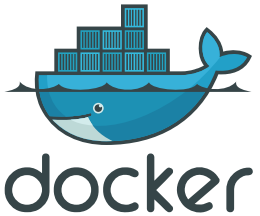


Docker без боли для PHP-разработчиков

Савченко Валерий
к.ф.м.н., разработчик, CPCS

1. Основные ошибки использования

docker container ≠ Virtual Machine

docker container → isollated process

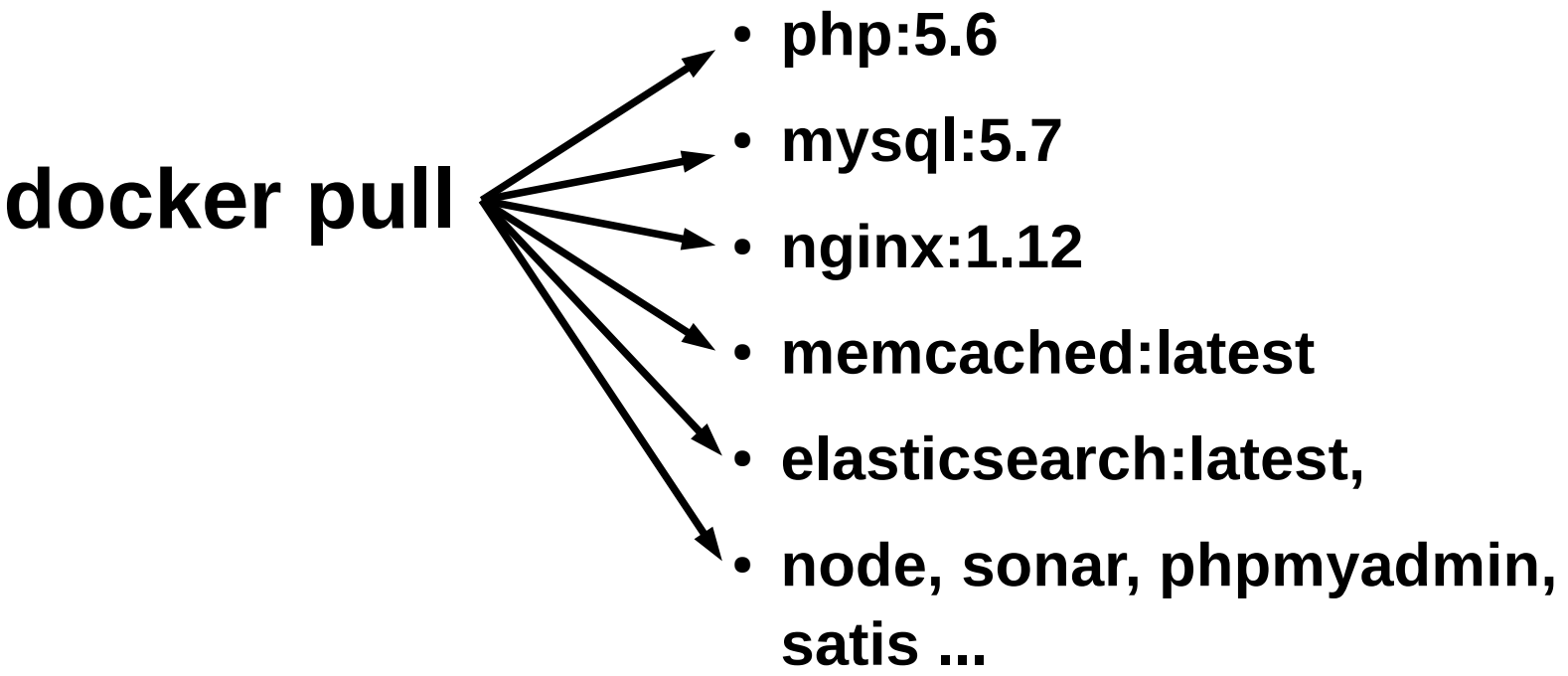
docker image ≠ docker container

docker image → template

docker containers → instances

real complex system → set of containers

2. Docker Engine - workflow

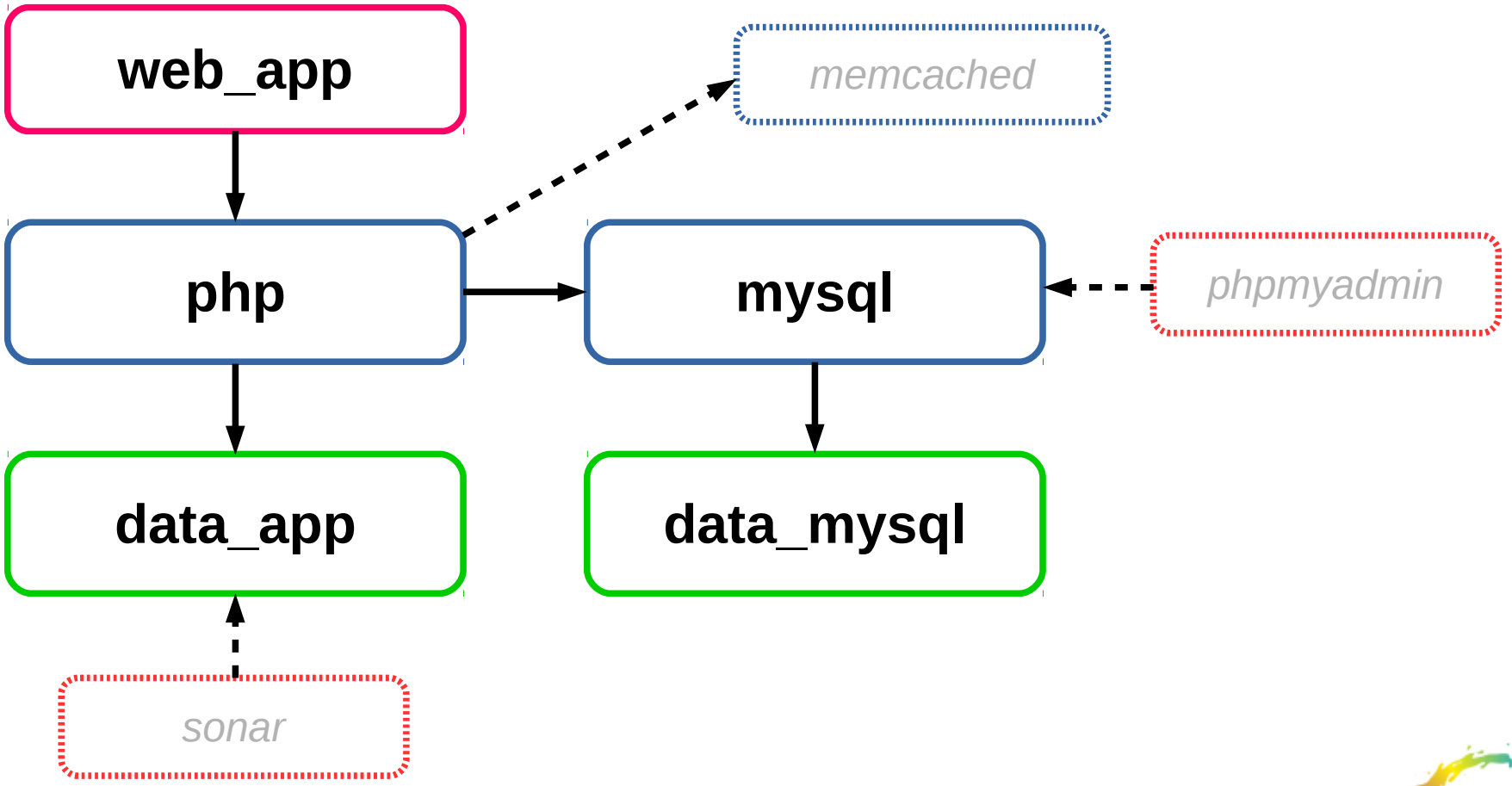


docker run -v ... -p ... -name ... -link ... -env...

3. Docker Engine - resume

- в реальной разработке обычно используется несколько контейнеров
- контейнеры связаны между собой, поэтому требуется правильно управлять очерёдностью загрузки
- необходимо использовать много технической информации при связывании (названия контейнеров, порты, зависимости и т.п.)
- при обновлении сценария необходимо самостоятельно обновить образы и перезапустить связанные контейнеры
- могут возникать проблемы при тонкой настройке образов и контейнеров
- возникает необходимость писать и отлаживать дополнительные скрипты запуска

4. Docker Compose — microservices



5. Docker Compose — skeleton

Корневой каталог разработки:

```
.
├── data
├── services
├── src
├── .env
└── docker-compose.yml
```

Каталог приложений:

```
src
└── www
    └── app1
        ├── …
        └── index.php
```

Сервисы приложения:

```
services
├── data_app
│   └── Dockerfile
├── mysql
│   └── configs
├── nginx
│   ├── configs
│   └── Dockerfile
└── php
    ├── configs
    ├── scripts
    └── Dockerfile
```

6. Docker Compose — настройка .env

Подготовка параметров для образов и контейнеров

```
# Project vars
COMPOSE_PROJECT_NAME=sample

PREFIX_IMAGE=local_
PREFIX_CONTAINER=dev_

# id $USER
DOCKER_GROUP_ID=1000
DOCKER_USER_ID=1000
```

```
#Port web app on host machine
WEB_APP_PORT=8888

# MYSQL vars
MYSQL_VERSION=5.6
MYSQL_PORT=33306
MYSQL_ROOT_PASSWORD=root
```

7. Создание нового приложения

1. Поднимаем все контейнеры:
   docker-compose up -d

2. Копируем исходники в папку проекта:
   git clone git://... src/www/**app1**

3. Создаем файл конфигурации **services/nginx/configs/sites-enabled/app1.conf**:
   ```
   server_name    app1;
   root /var/www/app1;
   ...
   fastcgi_pass  dev_php:9000;
   ```

4. Добавлем на хост машине хост app1 в /etc/hosts:
   127.0.0.1 app1

5. Загружаем новые конфигурации в nginx:
   docker exec -it dev_web_app nginx -s reload

6. Проверяем работу приложения в браузере:
   http://app1:8888/

8. Выводы

- Данная методика дает возможность перенастроить стандартные образы под задачи разработчика
- Этот способ позволяет удобно добавлять необходимые сервисы на базе готовых стандартных образов
- Использование в контейнерах ID-пользователей хост машины исключает ряд ошибок, связанных с нарушением прав
- Использование отдельных томов данных исключает случайное удаление данных при удалении контейнера
- «Прокидывание» настроек внутрь контейнера позволяет обновлять контейнеры без дополнительной перезагрузки
- Использование стандартных образов с одной базой позволяет значительно экономить дисковое пространство

A1. Docker Compose — Data Volume (I)

docker-compose.yml

```
data_app:
    env_file: .env
    build:
        context: .
        dockerfile: ./services/data_app/Dockerfile
        args:
            - DOCKER_GROUP_ID
            - DOCKER_USER_ID
    image:          "${PREFIX_IMAGE}data_app"
    container_name: "${PREFIX_CONTAINER}data_app"
    hostname:       "${PREFIX_CONTAINER}data_app"
    volumes:
        - ./src/www:/var/www
    working_dir: /var/www
    user: web-user
```

A2. Docker Compose — Data Volume (II)

services/data_app/Dockerfile

```
FROM debian:wheezy

...

ARG DOCKER_GROUP_ID
ARG DOCKER_USER_ID

USER 0

RUN groupadd --gid ${DOCKER_GROUP_ID} web-user
RUN useradd  --gid ${DOCKER_GROUP_ID} --uid \
    ${DOCKER_USER_ID} --create-home --shell /bin/bash web-user
…
```

A3. Docker Compose — PHP service

docker-compose.yml

```
php:
    env_file: .env
    build:
        dockerfile: services/php/Dockerfile
        …
    image:          "${PREFIX_IMAGE}php"
    container_name: "${PREFIX_CONTAINER}php"
    hostname:       "${PREFIX_CONTAINER}php"
    depends_on:
        - data_app
        - mysql
    links:
        - mysql
    volumes_from:
        - data_app
    volumes:
        - ./services/php/configs/custom.ini:/usr/local/etc/php/…
        - ./services/php/scripts/entrypoint.sh:/entrypoint.sh
    user: web-user
    working_dir: /var/www
```

A4. Docker Compose — APP service

docker-compose.yml

```
web_app:
    …
    image:          "${PREFIX_IMAGE}web_app"
    container_name: "${PREFIX_CONTAINER}web_app"
    hostname:       "${PREFIX_CONTAINER}web_app"
    ports:
      - ${WEB_APP_PORT}:80
    depends_on:
      - data_app
      - data_mysql
      - php
      - mysql
    links:
      - php
      - mysql
    volumes_from:
      - data_app
    volumes:
      - ./data/logs/nginx:/var/log/nginx
      - ./services/nginx/configs/nginx.conf:/etc/nginx/nginx.conf
      - ./services/nginx/configs/sites-enabled:/etc/nginx/sites-enabled
    working_dir: /var/www/
```

Спасибо за внимание.

Материалы доклада:

https://github.com/wellic/conf-2016-docker-php.git