

Обзор Docker-Compose на примере настройки микро-сервисов для систем IBM POWER

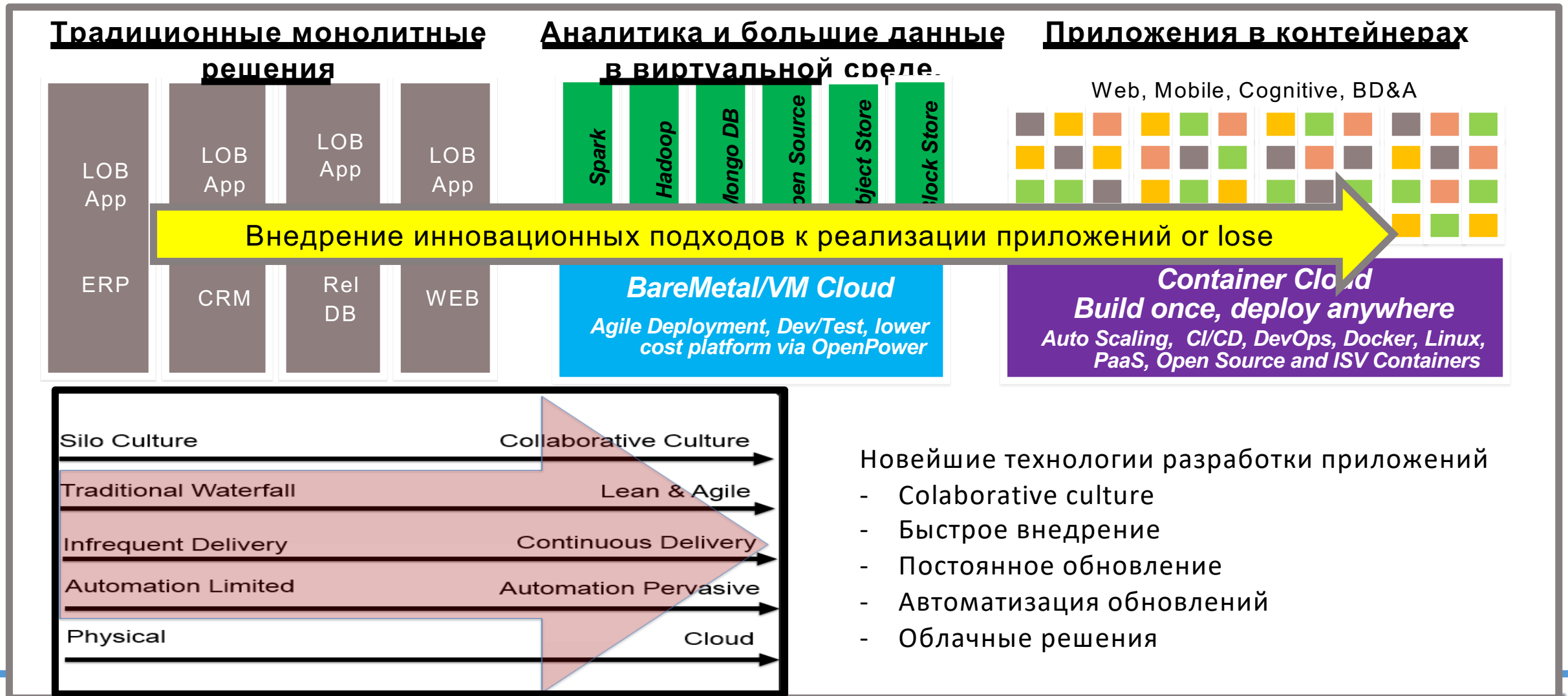


Валерий Груздев
IBM Power Moscow

vgruzdev@ru.ibm.com

<https://www.meetup.com/IBM-Power-Moscow/>

Облачные технологии заставляют двигаться от в сторону открытых стандартов.

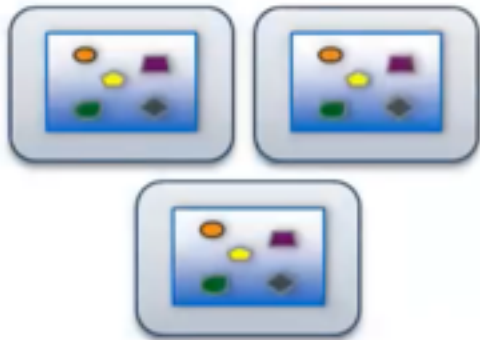


Развитие приложений – Микросервисы и Контейнеры.

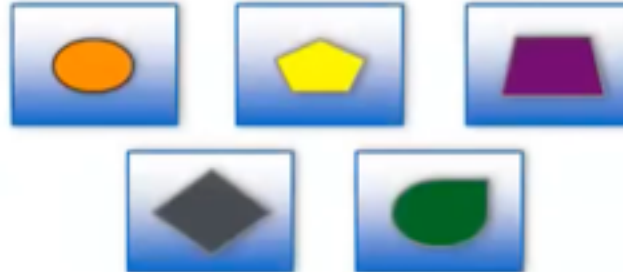
Monolythic



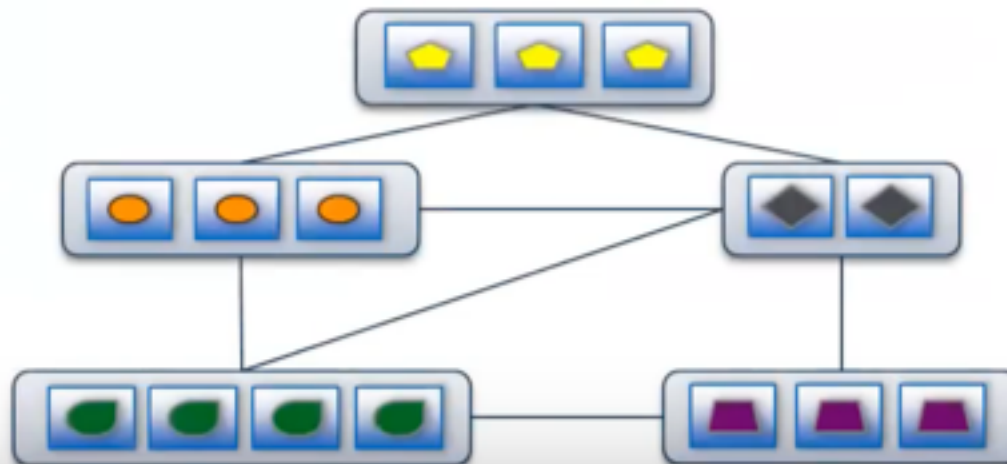
Scaling



Microservices



Scaling



Зачем ?


- Стандартизует API интерфейс
- Упрощает внесение изменений
- Ускоряет разработку и внедрение новых функций
- Повторяемость решений
- Строгая упорядоченность

Приложения в контейнерах



Все начинают с одного первого контейнера.

Приложения в контейнерах



**Но очень скоро контейнеров становится слишком много –
необходим механизм управления контейнерами.**

Kubernetes – оркестратор для контейнеров.



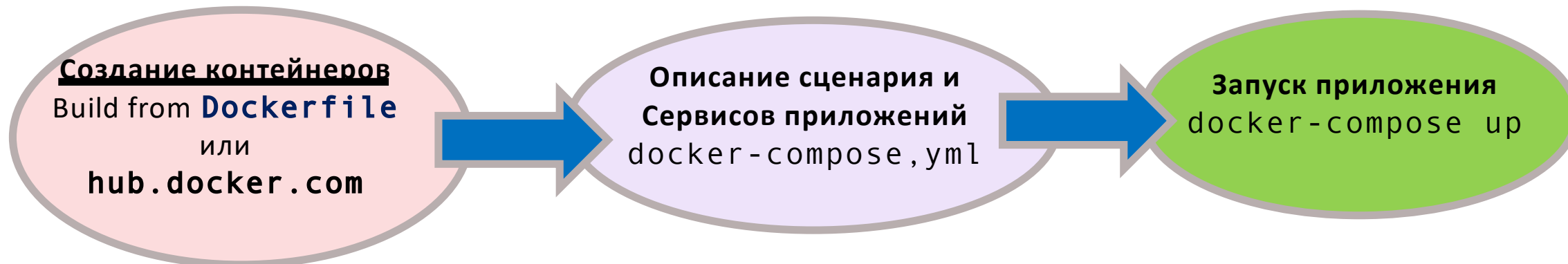
Docker-compose

Docker-compose очень простой в использовании,
но довольно мощный инструмент управления контейнерами

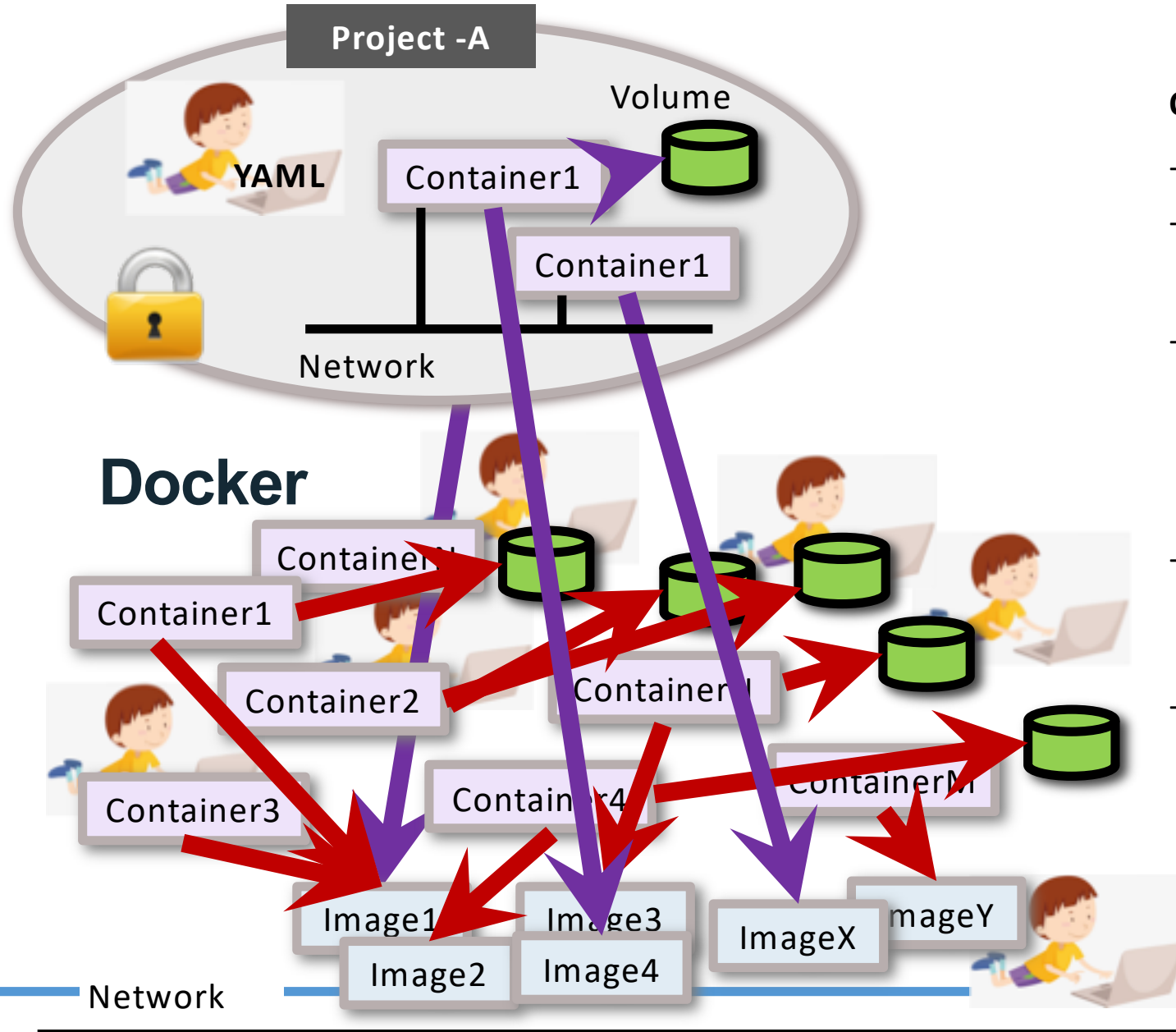
Может использоваться во всех возможных сценариях:



Любой проект строится за 3 шага



Docker-compose



Основные свойства COMPOSE

- Обеспечивает **изолированное окружение**.
- Compose использует **уникальное имя проекта**
 - «чужие» контейнеры не видны
- **Управление томами** для постоянных данных, а также управление сетями
 - Автоматически создает, подключает и сохраняет тома для размещения данных,
- **Отслеживает состояние контейнеров**. Использует уже собранные контейнеры, что ускоряет процесс разработки и сборки
- Широкое **использование переменных** в конфигурационном файле позволяет строить сложные логические цепочки.

Проект NEXTCLOUD

https://hub.docker.com/_/nextcloud/

Nextcloud - Домашняя облачная хранилка (фото, видео, документы, и т.д.)

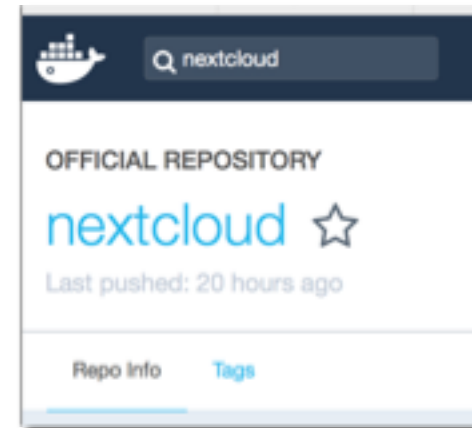
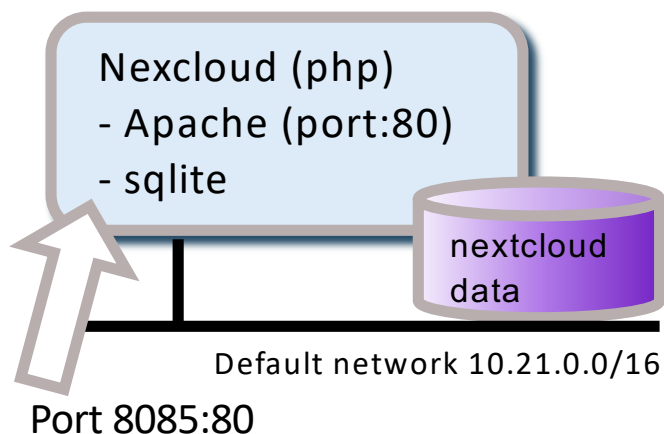
Простейший способ запуска данного сервиса на любом компьютера, где установлен docker:

```
$ docker run -d -p 8080:80 nextcloud
```

... и nextcloud будет доступен по адресу <http://localhost:8080/>

Но это не наш метод ...

Создадим проект **Compose**:



```
ibmuser@GPU-dockers:~/nextcloud-1$ cat docker-compose.yml
version: '3'
```

```
services:
  app:
    image: nextcloud
    restart: always
    ports:
      - 8085:80
    volumes:
      - nextcloud:/var/www/html
      - data:/var/www/html/data
```

```
volumes
  nextcloud:
  data:
```

```
networks:
  default:
```

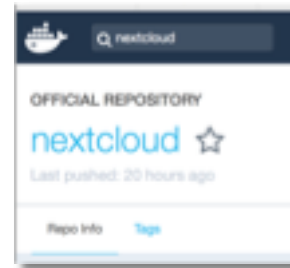
NEXTCLOUD – apache, sqlite

```
ibmuser@GPU-dockers:~/nextcloud-1$ docker-compose pull
Pulling app ... done

ibmuser@GPU-dockers:~/nextcloud-1$ docker-compose up -d
Creating network "nextcloud-1_default" with driver "bridge"
Creating volume "nextcloud-1_data" with default driver
Creating volume "nextcloud-1_nextcloud" with default driver
Creating nextcloud-1_app_1 ... Done

ibmuser@GPU-dockers:~/nextcloud-1$ docker-compose ps
      Name                        Command              State      Ports
-----
nextcloud-1_app_1  /entrypoint.sh apache..  Up        0.0.0.0:8085->80
```

<http://172.19.12.50:8085/>



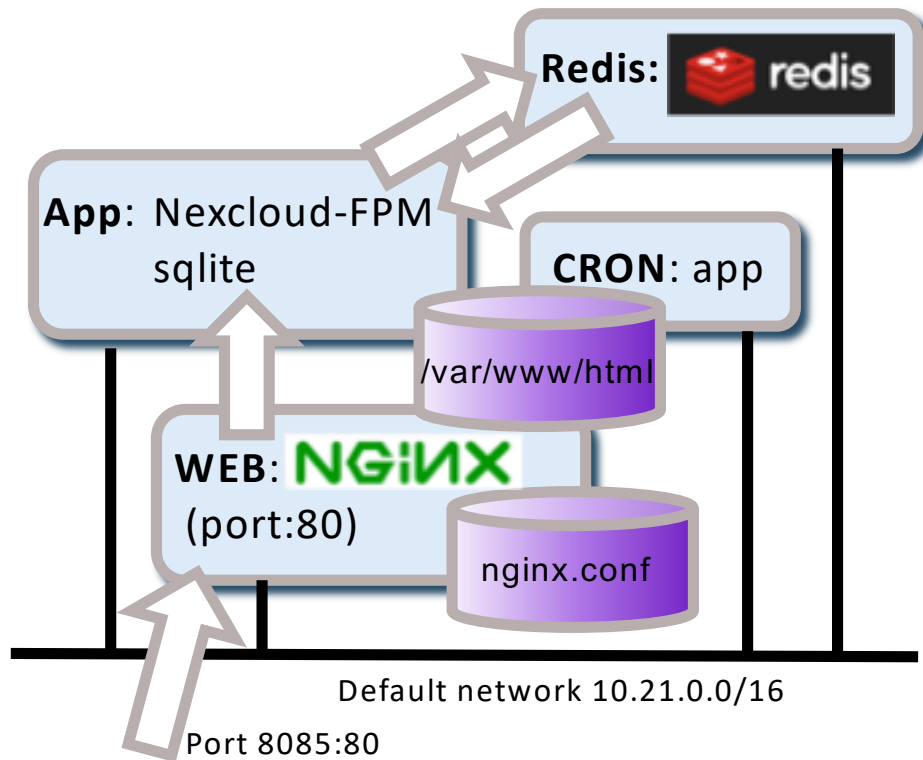
Для одного пользователя вполне достаточно !

Попробуем улучшить работу приложения.

- Используем версию nextcloud-fpm (FastCGI PHP Manager)
- Добавим кеш-DB 
- В качестве web-server используем 

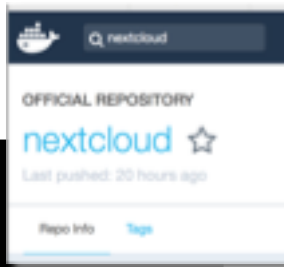


NEXTCLOUD – FPM, NGINX, Redis



```
ibmuser@GPU-dockers:~/nextcloud-2$ cat app/Dockerfile
FROM nextcloud:13.0.1-fpm
```

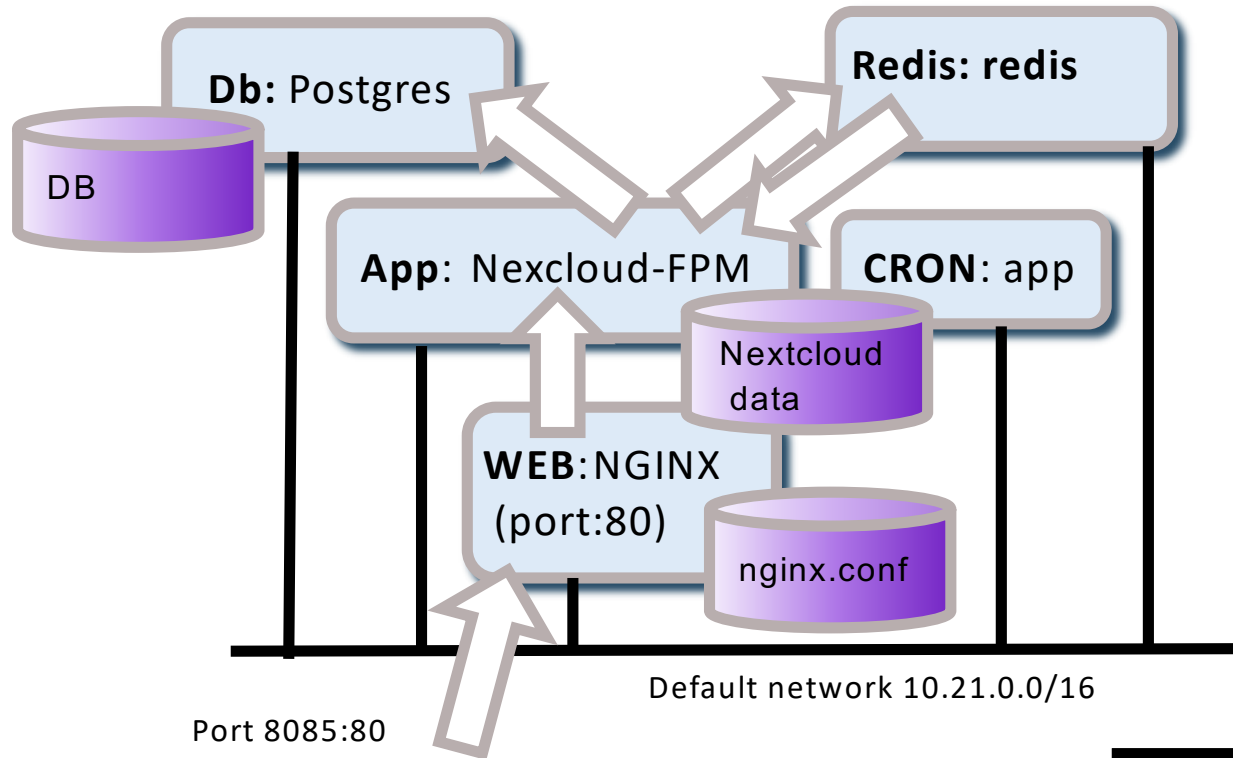
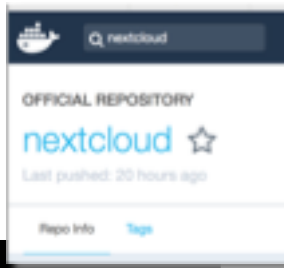
```
COPY redis.config.php
/usr/src/nextcloud/config/redis.config.php
```



```
version: '3'

services:
  redis:
    image: redis
    restart: always
  app:
    build: ./app
    restart: always
    volumes:
      - nextcloud:/var/www/html
      - data:/var/www/html/data
    depends_on:
      - redis
  web:
    image: nginx
    restart: always
    volumes:
      - nextcloud:/var/www/html:ro
      - ./web/nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - app
    ports:
      - 8085:80
    networks:
      - default
  . . . . .
```

NEXTCLOUD – FPM, NGINX, Redis, Cron, Postgres

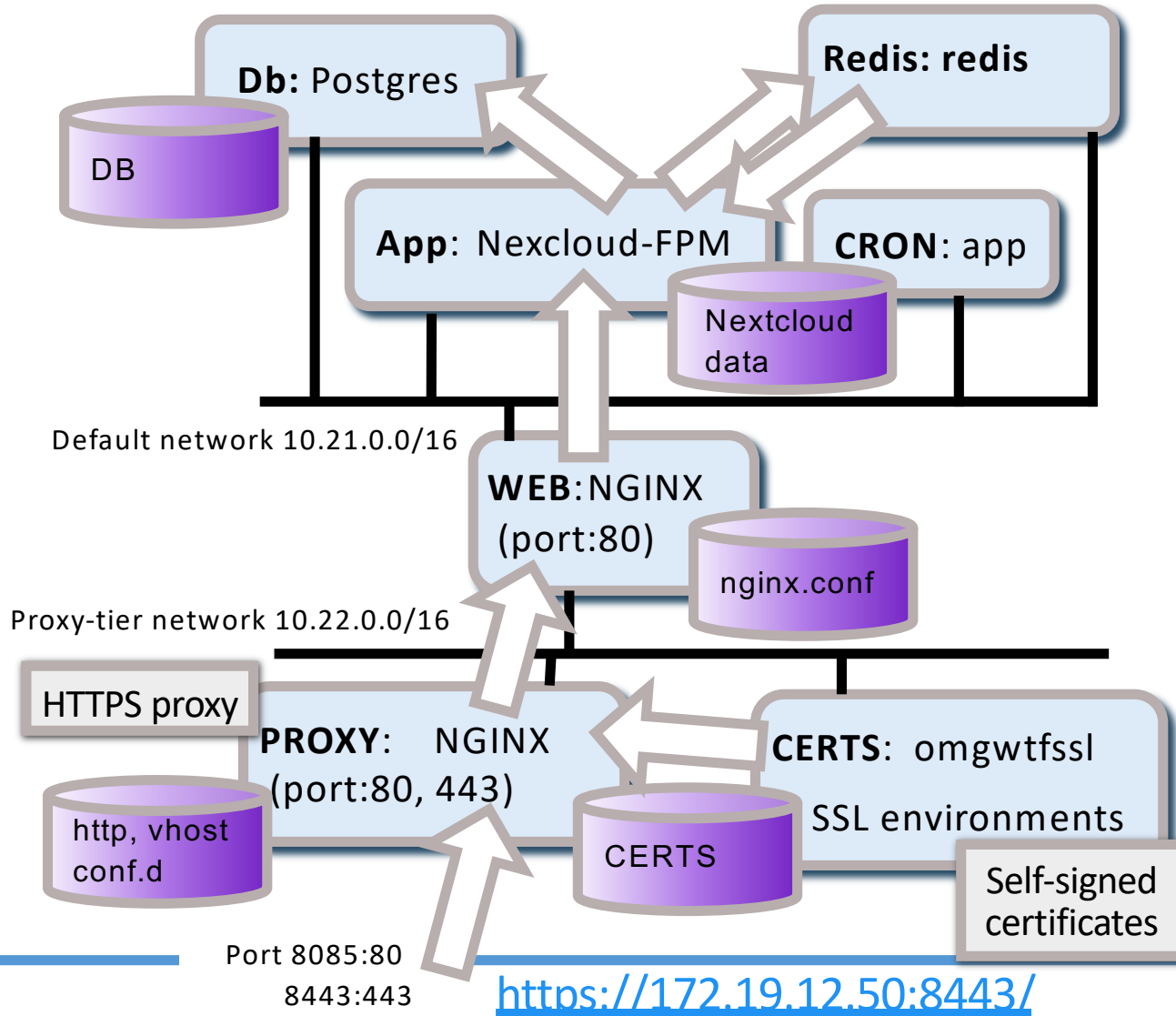


```
version: '3'

services:
  db:
    image: postgres
    restart: always
    volumes:
      - db:/var/lib/postgresql/data
    env_file:
      - postgres.env
```

```
ibmuser@GPU-dockers:~/nextcloud-3$ cat postgres.env
POSTGRES_PASSWORD=nextpasswd
POSTGRES_DB=nextcloud
POSTGRES_USER=nextcloud
POSTGRES_HOST=db
```

NEXTCLOUD – FPM, NGINX, Redis, Cron, Postgres, Nginx-proxy, Certificates



```
ibmuser@GPU-dockers:~/nextcloud-4$ cat nginx-  
proxy/Dockerfile  
#FROM nginx:1.13  
FROM nginx  
  
# Install wget and install/updates certificates  
RUN apt-get update \  
&& apt-get install -y -q --no-install-recommends \  
ca-certificates \  
dnsutils \  
# apt-utils \  
# busybox \  
&& apt-get clean \  
&& rm -r /var/lib/apt/lists/* \  
&& mkdir -p /etc/nginx/dhparam \  
&& mkdir /app  
  
COPY ./* /app/  
  
WORKDIR /app/  
  
ENTRYPOINT ["/app/entrypoint.sh"]  
CMD ["nginx", "-g", "daemon off;"]
```

Команды Docker-compose



`docker-compose [-f yml-file] COMMAND ...`

`docker-compose pull`

скачать образы контейнеров

`docker-compose build [--pull]`

построить все сервисы docker-compose согласно сценарию YAML

`docker-compose up -d [SERVICE]`

запустить все или указанные сервисы

`docker-compose ps`

статус сервисов в текущем окружении

`docker-compose start | pause | unpause | stop | restart | rm [SERVICE]`

`docker-compose logs [-f] [SERVICE]`

`docker-compose top`

`docker-compose exec [--user USER] SERVICE COMMAND [ARGS...]`

`docker-compose down [-v]`

Все примеры можно найти: <https://github.com/vgrusdev/nextcloud-ibm-power>