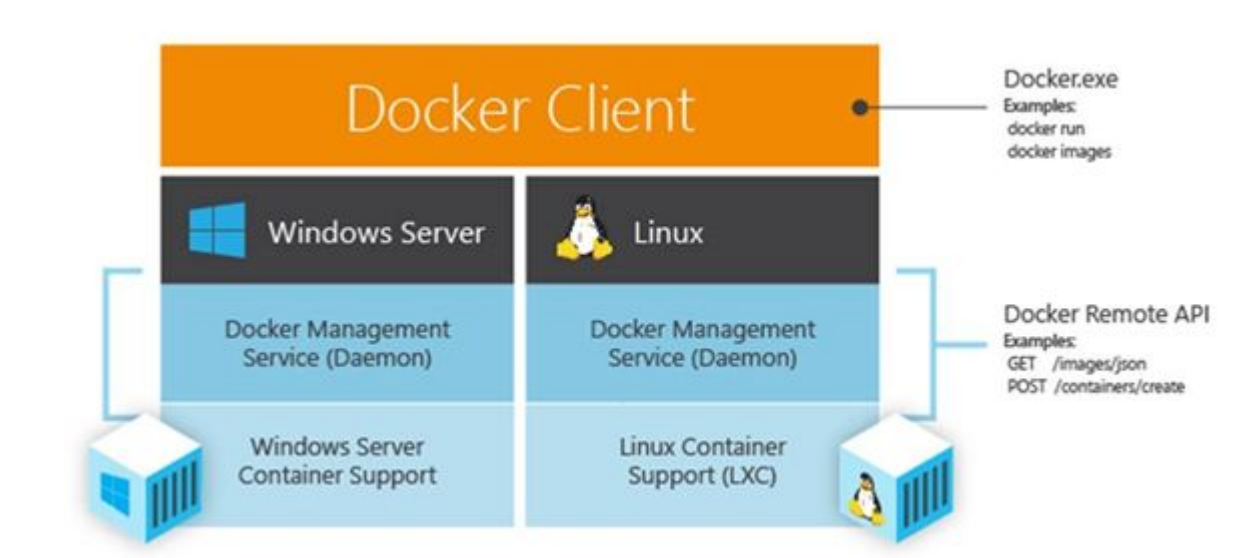
МДК 02.02 Организация администрирования компьютерных сетей

Контейнеры Docker



Для групп специальности СПО 09.02.02 Компьютерные сети

Docker – это программная платформа для быстрой сборки, отладки и развертывания приложений. Docker упаковывает ПО в стандартизованные блоки, которые называются контейнерами.

Контейнеры – это способ виртуализации операционной системы, позволяющий запускать приложение и его зависимости в виде процессов с изолированными ресурсами. Контейнеры позволяют легко упаковывать код приложений, их настройки и зависимости в простые в использовании компоновочные блоки, обеспечивающие единообразную среду, эффективность и производительность труда разработчиков, а также контроль версий.

Каждый **контейнер** включает все необходимое для работы приложения: библиотеки, системные инструменты, код и среду исполнения. Благодаря Docker пользователи могут быстро развертывать и масштабировать свои приложения в любой среде и сохранять уверенность в том, что код будет работать.

Docker — это инструмент, предоставляющий удобный интерфейс для работы с LXC.

LXC (Linux Containers) — механизм виртуализации на уровне операционной системы, позволяющий исполнять множество изолированных Linux-систем (контейнеров) в одной системе.

Ядро Linux может изолировать ресурсы (процессор, память, ввод/вывод, сеть и так далее) при помощи cgroups, не прибегая для этого к использованию виртуальных машин. Посредством cgroups изолируются так же деревья процессов, сеть, пользователи и файловые системы.

LXC комбинирует cgroups и пространства имён (namespace).

На данный момент LXC использует следующие возможности ядра:

- Kernel namespaces (ipc, uts, mount, pid, network and user)
- Apparmor and SELinux profiles
- Seccomp policies
- Chroots (using pivot_root)
- Kernel capabilities
- CGroups (control groups)

cgroups (англ. *control group*) — механизм ядра Linux, который ограничивает и изолирует вычислительные ресурсы (процессорные, сетевые, ресурсы памяти, ресурсы ввода-вывода) для групп процессов

Установка

1. Инсталлируем сразу все необходимые пакеты:

```
sudo apt-get install lxc debootstrap bridge-utils
```

2. Монтирование cgroup. Пакет lxc зависит от пакета cgroup-lite, который монтирует каждую cgroup подсистему отдельно в `/sys/fs/cgroup/` (в Debian и Ubuntu cgroup вручную монтировать не нужно), но если cgroup все же не смонтирован, то:

добавляем строку в `/etc/fstab`:

```
cgroup  /sys/fs/cgroup  cgroup  defaults  0   0
```

монтируем:

```
sudo mount /sys/fs/cgroup
```

В место создания точки монтирования - можно создать любой каталог (например, sudo mkdir /cgroup) и сохранить соответствующую запись в `/etc/fstab`:

```
cgroup  /cgroup   cgroup  defaults  0   0
```

Проверка правильности установки:

```
sudo lxc-checkconfig
```

При проверке - выводе команды не должно присутствовать сообщений об ошибках:

Команды

lxc-create

создать новый контейнер LXC

lxc-start

запустить контейнер LXC

lxc-console

подключиться к консоли указанного контейнера

lxc-attach

запустить указанную программу внутри контейнера - (NOT SUPPORTED) Run a command in a running container

lxc-destroy

уничтожения контейнера

lxc-stop

остановить процесс, работающий внутри контейнера

lxc-execute

выполнить указанную команду внутри контейнера (в чём отличие от lxc-attach?)

lxc-monitor

мониторить состояние контейнеров

lxc-wait

ждать определённого состояния контейнера; завершаться, когда состояние достигнуто

lxc-cgroup

управление cgroup-группами контейнера

lxc-ls

показать список контейнеров в системе

lxc-ps

показать список процессов внутри определённого контейнера

lxc-info

показать информацию о заданном контейнере

lxc-freeze

заморозить все процессы указанного контейнера

lxc-unfreeze

разморозить все процессы указанного контейнера

Способы связи контейнеров Docker.

Технология Docker позволяет связать два контейнера и даёт механизм ссылок (Docker links).

1. Определение имя контейнеру при запуске: `docker run -d --name db training/postgres`.

Теперь можно ссылаться на этот контейнер по имени `db`.

2. Запускаем второй контейнер, связывая его с первым:

```
docker run -d -P --name web --link db:db training/webapp python app.py.
```

`--link name:alias.name` — имя контейнера,

`alias` — имя, под которым этот контейнер будет известен запускаемому.

В результате:

во-первых, в контейнере `web` появится набор переменных окружения, указывающих на контейнер `db`,

во-вторых в `/etc/hosts` контейнера `web` появится имя `db` указывающий на ip, на котором мы запустили контейнер с базой данных. Набор переменных окружения, которые будут доступны в контейнере `web` вот такой:

```
DB_NAME=/web/db
DB_PORT=tcp://172.17.0.5:5432
DB_PORT_5432_TCP=tcp://172.17.0.5:5432
DB_PORT_5432_TCP_PROTO=tcp
DB_PORT_5432_TCP_PORT=5432
DB_PORT_5432_TCP_ADDR=172.17.0.5
```