



# Docker под капотом

## Виктор Попов

Техлид DevOps команды дирекции  
больших данных в X5 Retail Group

Поддерживаем и развиваем k8s кластера  
as-a-service для 20+ продуктовых команд. А  
так же кучу сопутствующих инструментов

Чиню коммуникации между dev и ops  
командами. Рассказываю пользователям,  
чем вообще занимаются инженеры



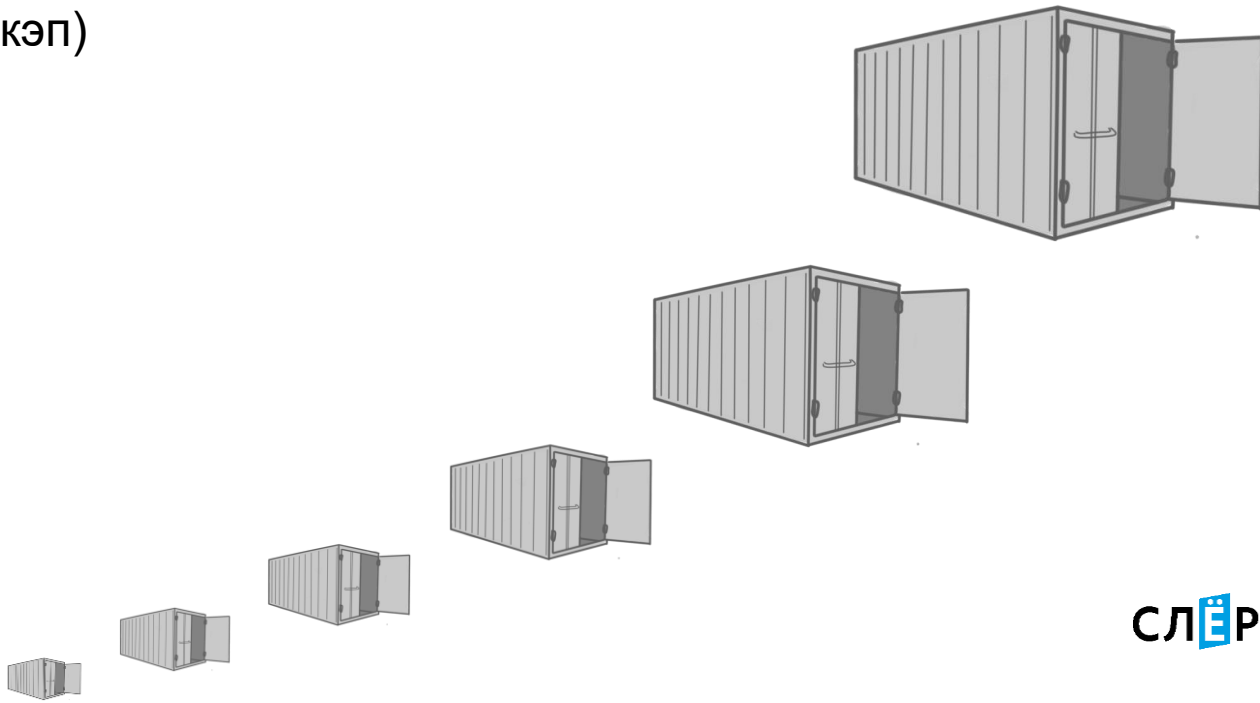
**Tg:** @IvanovIvanIvanovich1  
**mail:** slider2k4@gmail.com

**СЛЁРМ**



# Что такое контейнер?

Это процесс (ваш кэп)

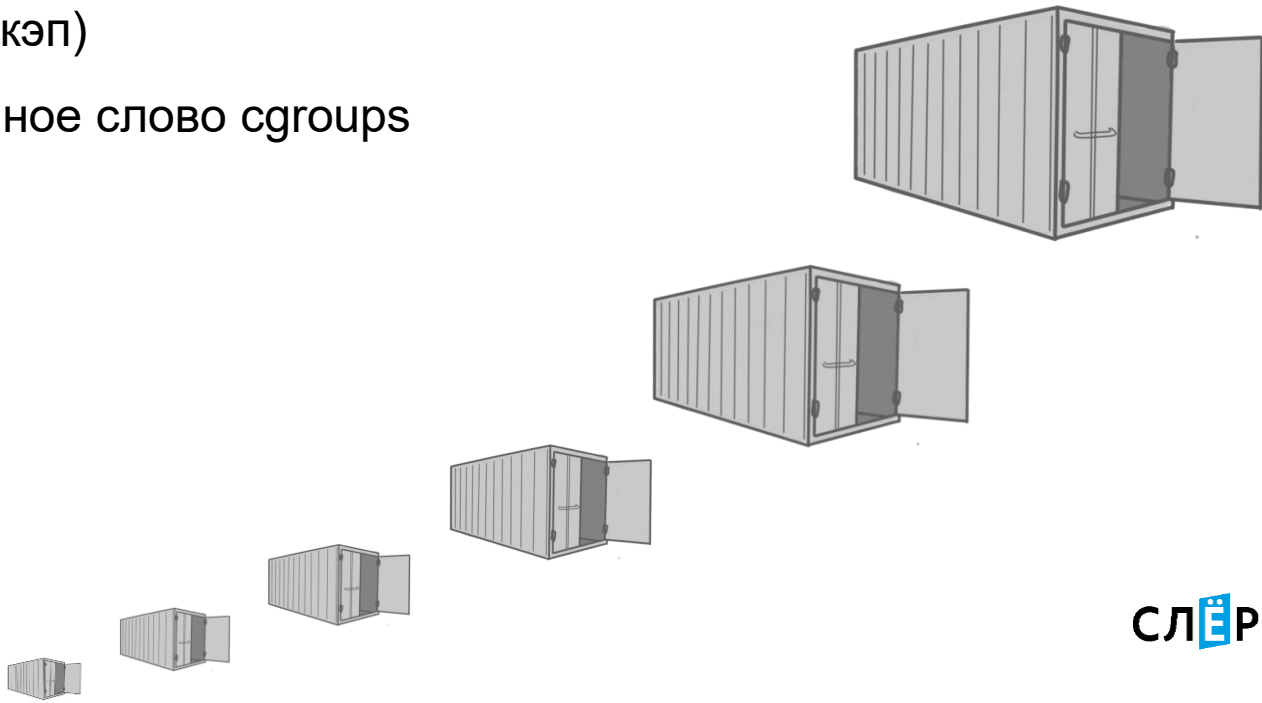




# Что такое контейнер?

Это процесс (ваш кэп)

А еще там есть умное слово sgroups



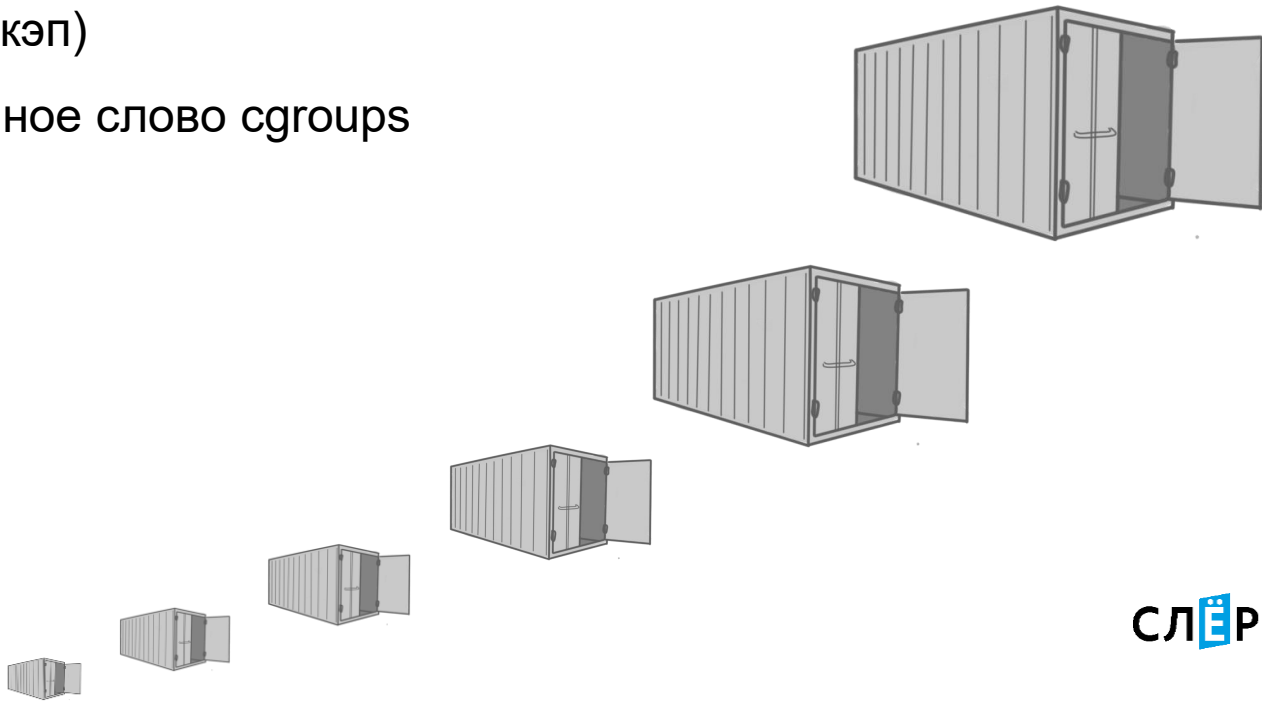


# Что такое контейнер?

Это процесс (ваш кэп)

А еще там есть умное слово cgroups

И namespaces





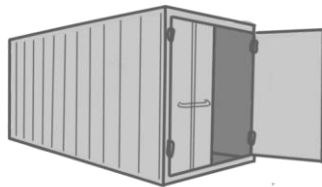
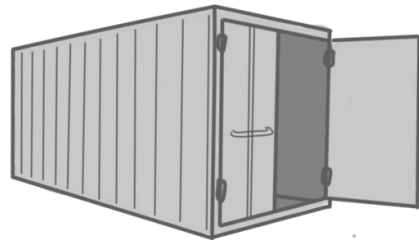
# Что такое контейнер?

Это процесс (ваш кэп)

А еще там есть умное слово cgroups

И namespaces

А еще capabilities



Давайте посмотрим на этот процесс!



**Контрольная группа** - это группа процессов в Linux, для которой механизмами ядра наложена изоляция и установлены ограничения на некоторые вычислительные ресурсы.  
(с) Википедия







# Cgroups. А зачем?

Ограничивать процессы по потреблению ресурсов:

Память

ЦПУ

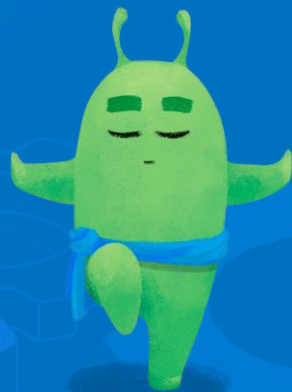
Диск

Сеть

И т.д.



# Запуск контейнеров с ограничениями по ресурсам





# Namespaces

Namespaces – абстракция над ресурсами операционной системы.

Ресурсы, находящиеся в одном NS делят соответствующие ресурсы





# Namespaces

**Namespaces – абстракция над ресурсами операционной системы.**

**Ресурсы, находящиеся в одном NS делят соответствующие ресурсы**

**Существует 7 пространств имён:**

Cgroups

IPC (InterProcessConnection)

Network

Mount

PID

User

UTS





# Capabilities

Разрешения процесса на выполнение определённых системных вызовов





# Capabilities

Разрешения процесса на выполнение определённых системных вызовов

Всего около 20 шт





# Capabilities

Разрешения процесса на выполнение определённых системных вызовов

Всего около 20 шт

Например:

`CAP_CHOWN` – разрешение на смену UID и GUID файла





# Capabilities

Разрешения процесса на выполнение определённых системных вызовов

Всего около 20 шт

Например:

- `CAP_CHOWN` – разрешение на смену UID и GUID файла

- `CAP_KILL` – разрешение на отправку сигналов (sigterm, sigkill и др)







# Capabilities

Разрешения процесса на выполнение определённых системных вызовов

Всего около 20 шт

Например:

- CAP\_CHOWN – разрешение на смену UID и GUID файла

- CAP\_KILL – разрешение на отправку сигналов (sigterm, sigkill и др)

- CAP\_NET\_BIND\_SERVICE – разрешение на использование портов с номером меньше 1024





# Capabilities

Разрешения процесса на выполнение определённых системных вызовов

Всего около 20 шт

Например:

- `CAP_CHOWN` – разрешение на смену UID и GUID файла

- `CAP_KILL` – разрешение на отправку сигналов (sigterm, sigkill и др)

- `CAP_NET_BIND_SERVICE` – разрешение на использование портов с номером меньше 1024

- И др.



Нет!  
Но с ним удобнее

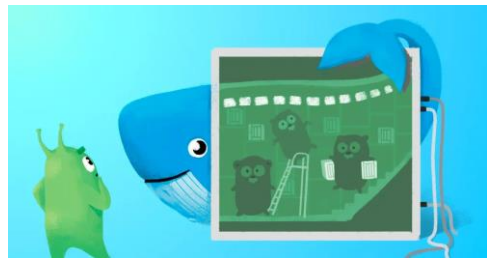




# Так что же такое Docker?

## 5 основных компонентов:

**Dockerd** - Демон докера, собирает контейнеры, управляет сетью и вольюмами, логированием и прочими высокоуровневыми вещами;

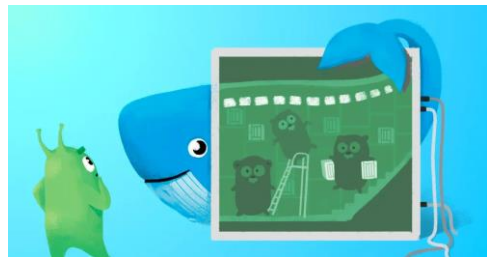




# Так что же такое Docker?

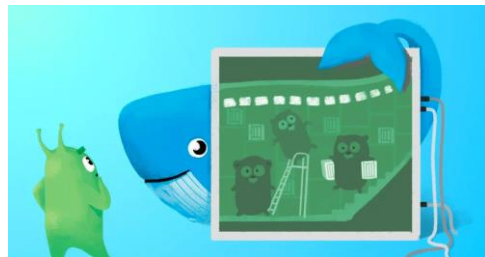
## 5 основных компонентов:

- **Dockerd** - Демон докера, собирает контейнеры, управляет сетью и вольюмами, логированием и прочими высокоуровневыми вещами;
- **Containerd** – Управляет жизненным циклом контейнеров, запуском, сетью на уровне драйвера;





# Так что же такое Docker?

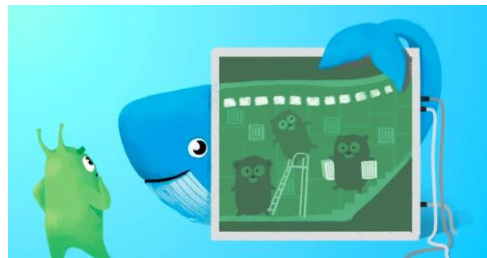


## 5 основных компонентов:

- **Dockerd** - Демон докера, собирает контейнеры, управляет сетью и вольюмами, логированием и прочими высокоуровневыми вещами;
- **Containerd** – Управляет жизненным циклом контейнеров, запуском, сетью на уровне драйвера;
- **Runc** – Собирает контейнер и запускает его, отдаёт события жизненного цикла контейнеров;



# Так что же такое Docker?

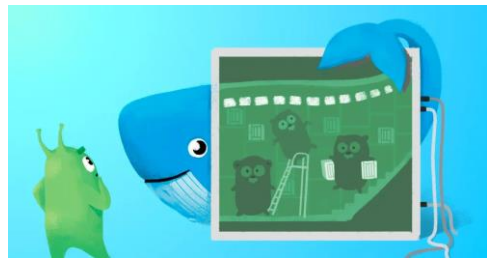


## 5 основных компонентов:

- **Dockerd** - Демон докера, собирает контейнеры, управляет сетью и вольюмами, логированием и прочими высокоуровневыми вещами;
- **Containerd** – Управляет жизненным циклом контейнеров, запуском, сетью на уровне драйвера;
- **Runc** – Собирает контейнер и запускает его, отдаёт события жизненного цикла контейнеров;
- **Docker-containerd-shim** – Передаёт файловые дескрипторы контейнера (stdin/out);



# Так что же такое Docker?



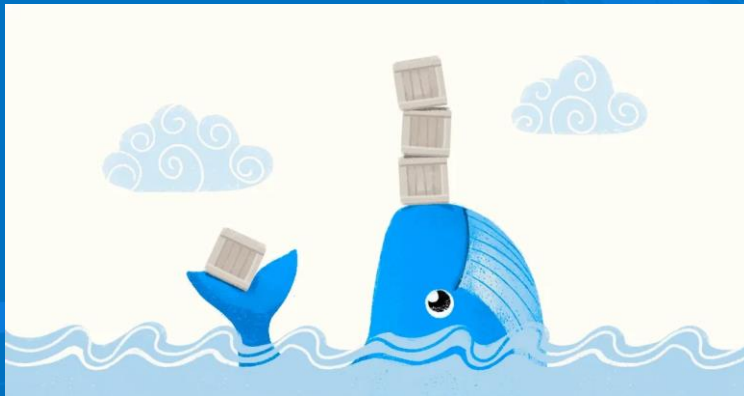
## 5 основных компонентов:

- **Dockerd** - Демон докера, собирает контейнеры, управляет сетью и вольюмами, логированием и прочими высокоуровневыми вещами;
- **Containerd** – Управляет жизненным циклом контейнеров, запуском, сетью на уровне драйвера;
- **Runc** – Собирает контейнер и запускает его, отдаёт события жизненного цикла контейнеров;
- **Docker-containerd-shim** – Передаёт файловые дескрипторы контейнера (stdin/out);
- **Docker-proxy** – Отвечает за nat между контейнером и хостовой системой (если он есть).



# Так что же такое Docker?

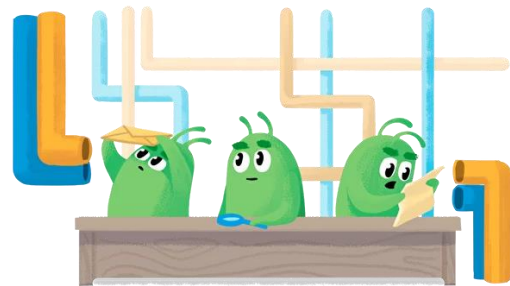
Удобный интерфейс над `linux`  
Плюс немного шашечек





# Docker run nginx

Что происходит?

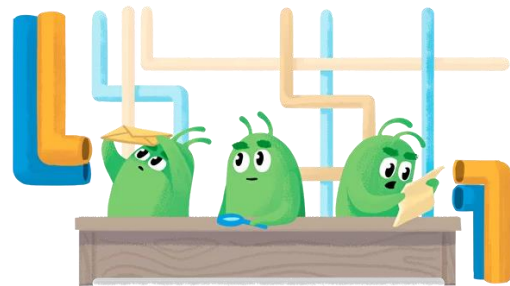




# Docker run nginx

## Что происходит?

**Dockerd** ищет image локально и если его нет пулит его из registry;

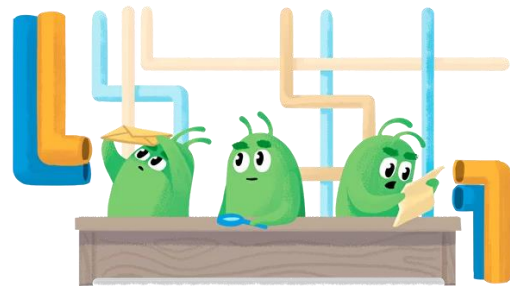




# Docker run nginx

## Что происходит?

- **Dockerd** ищет image локально и если его нет пулит его из registry;
- **Dockerd** обращается к containerd и просит его запустить контейнер;

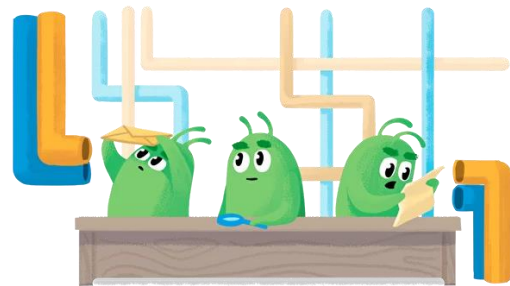




# Docker run nginx

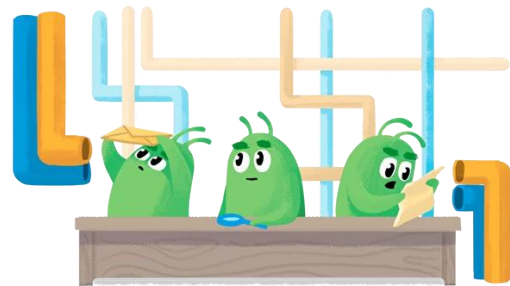
## Что происходит?

- **Dockerd** ищет image локально и если его нет пулит его из registry;
- **Dockerd** обращается к containerd и просит его запустить контейнер;
- **ContainerD** берёт image и создает из него OCI bundle, который runc может запустить;





# Docker run nginx

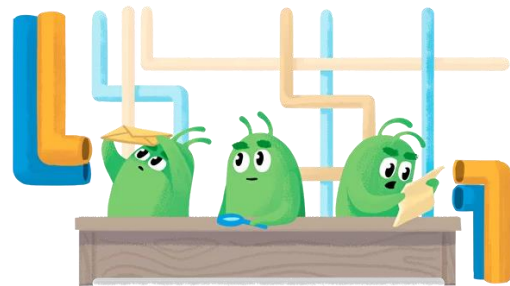


## Что происходит?

- **Dockerd** ищет image локально и если его нет пулит его из registry;
- **Dockerd** обращается к containerd и просит его запустить контейнер;
- **ContainerD** берёт image и создает из него OCI bundle, который runc может запустить;
- Затем вызывается **docker-containerd-shim**, который уже вызывает runc непосредственно для запуска контейнера;



# Docker run nginx



## Что происходит?

- **Dockerd** ищет image локально и если его нет пулит его из registry;
- **Dockerd** обращается к containerd и просит его запустить контейнер;
- **ContainerD** берёт image и создает из него OCI bundle, который runc может запустить;
- Затем вызывается **docker-containerd-shim**, который уже вызывает runc непосредственно для запуска контейнера;
- **Runc** не остаётся запущенным, родителем процесса-контейнера является docker-containerd-shim.



# Что еще за OCI?

**Open Container Initiative** – проект стандарта хранения и запуска контейнеров под крылом Linux Foundation







# Что еще за OCI?

**Open Container Initiative** – проект стандарта хранения и запуска контейнеров под крылом Linux Foundation

## Определяет 2 спецификации:

- **Runtime Specification** – всё про запуск контейнеров

- **Image Specification** – всё про формат контейнера и их хранение



Runс: дно кроличьей дыры

Давайте запустим контейнер именно так, как  
это делает docker, но без посредников!



# На уровень выше: containerD

А теперь к чуть более реальным инструментам.  
Запустите наш любимый nginx через containerD



# Практика





# На уровень выше: containerD

**Ctrl** – cli управления containerd

## Старт контейнера:

**Ctrl run -d docker.io/library/name:tag name**

Нам нужно назвать контейнер **ctrnginx**, чтобы его увидел мониторинг 😊

**Ctrl image pull** – аналог docker pull





# На уровень выше: containerD

А точно работает?

**Ctrl** – cli управления containerd

## Старт контейнера:

**Ctrl run -d docker.io/library/name:tag name**

Нам нужно назвать контейнер **ctrnginx**, чтобы его увидел мониторинг 😊

**Ctrl image pull** – аналог docker pull

Давайте сделаем **curl** внутри контейнера и посмотрим:

**Ctrl tasks exec --exec-id PID NAME CMD**





## На уровень выше: containerD. Ответы

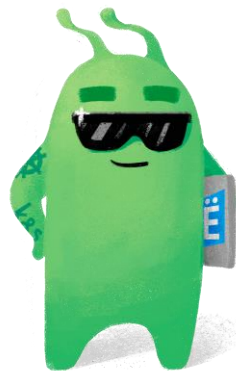
**Containerd** не пулит образ автоматически, если его нет – сделать это нужно руками.

```
ctr image pull docker.io/library/nginx:latest
```

```
ctr run -d docker.io/library/nginx:latest ctrnginx
```

```
ctr tasks exec --exec-id 1006 ctrnginx curl 127.0.0.1
```

**docker container ls** - не видим наши контейнеры!





southbridge.io

Спасибо!

СЛЁРМ

slurm.io