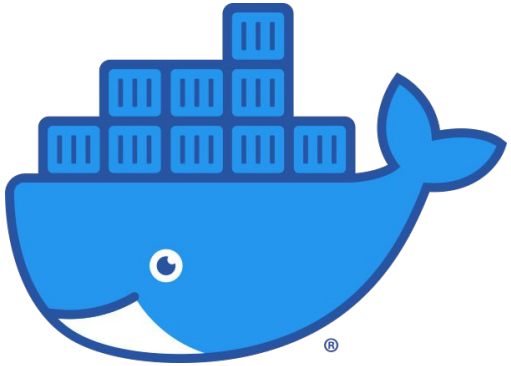
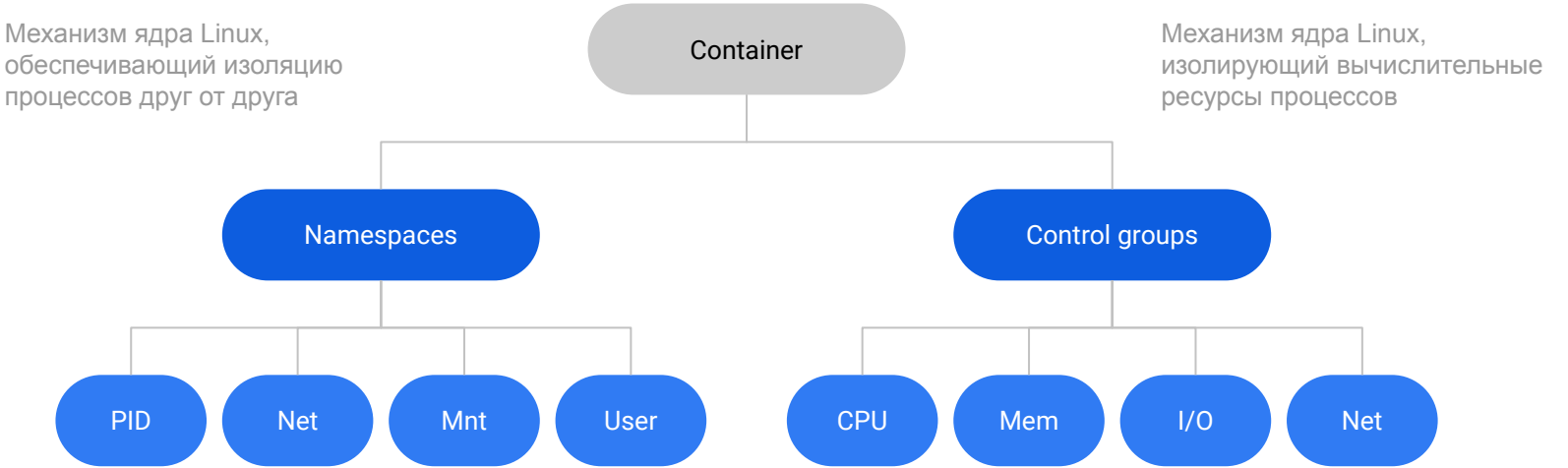


Введение в Docker

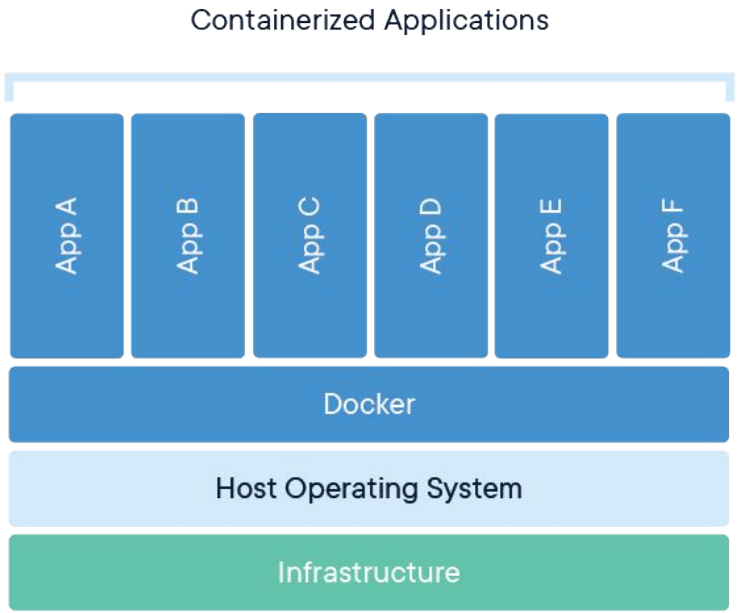
Докладчик: Крылов Илья
Дата: февраль 2020

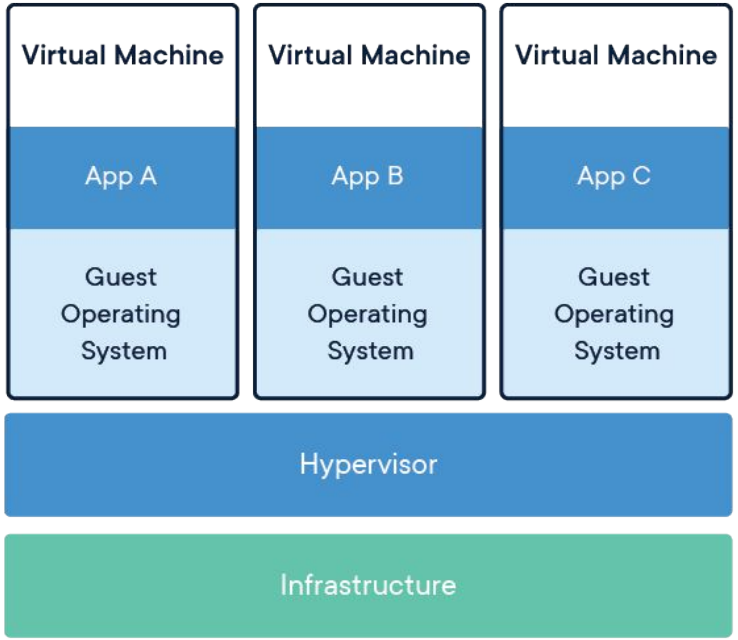
Что такое Docker?

Инструмент для запуска приложений в изолированных окружениях. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на любую Linux-систему с поддержкой **namespaces** и **control groups** в ядре, а также предоставляет среду по управлению контейнерами (Docker Engine).



Сравнение контейнеров с виртуальными машинами





Для чего нам нужен Docker?

- Изолированный запуск приложений в контейнерах
- Упрощение разработки, тестирования и деплоя приложений
- Отсутствие необходимости конфигурировать среду для запуска — она поставляется вместе с приложением — в контейнере
- Упрощает масштабируемость приложений и управление их работой с помощью систем оркестрации контейнеров (Kubernetes, Docker Swarm, Docker Compose)
- Хорошо подходит для микро-сервисной архитектуры (много маленьких сервисов, работающих совместно)

Реализация Docker для разных ОС

Linux

Нативная поддержка (контейнеры используют linux-ядро хост-сервера).

Windows

- Docker Desktop for Windows (для систем с поддержкой Hyper-V).
  Hyper-V не может использоваться совместно с “Virtual Box” (Vagrant перестанет работать!).
- Docker Toolbox (использует Docker Machine - виртуальную машину на базе “Virtual Box”).
  Legacy-решение с проблемами общих файловых каталогов.

Mac OS

- Docker Desktop for Mac. Использует виртуальную машину на базе “HyperKit”.
  Наблюдаются проблемы производительности совместно используемой файловой системы.
- Docker Toolbox (использует Docker Machine - виртуальную машину на базе “Virtual Box”).
  Legacy-решение с проблемами общих файловых каталогов.

Версии Docker и установка для Ubuntu

- Docker Community Edition (Docker CE)
- Docker Enterprise Edition (Docker EE)

```
$ sudo apt-get update

$ sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent \
    software-properties-common

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo apt-key fingerprint 0EBFCD88

$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) stable"

$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Запуск контейнера

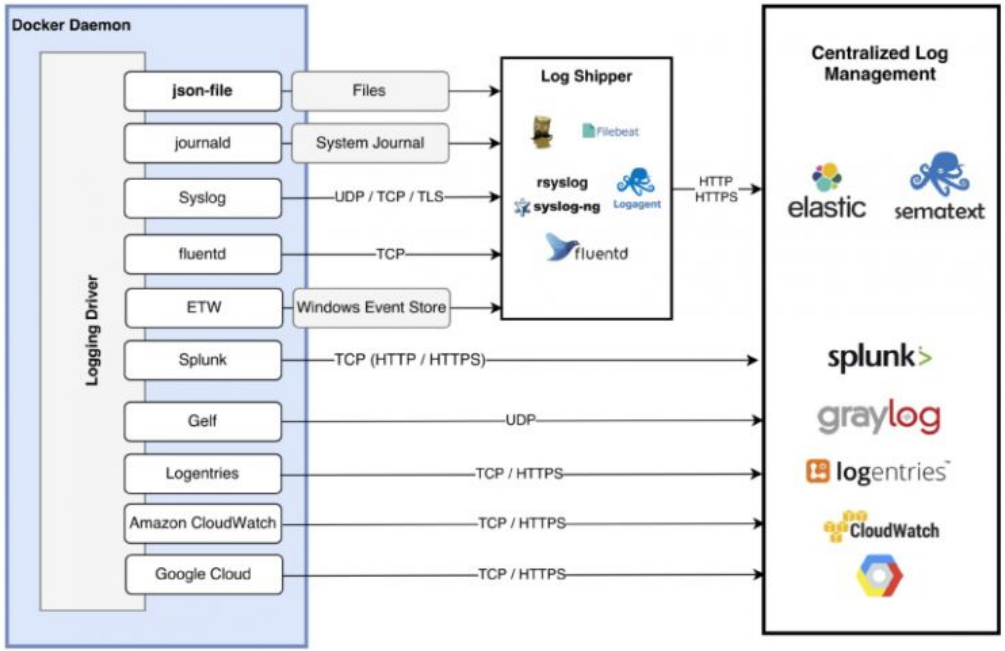
```
$ docker run -d -p 8080:80 nginx:1.13
```

1. docker pull nginx:1.13
2. docker create nginx:1.13
3. docker start <containerID>

```
$ docker ps
$ curl 127.0.0.1:8080 -i
$ docker logs <containerID>
$ docker rm -f <containerID>
```

Логирование контейнеров

Приложение должно писать все логи в STDOUT/STDERR. Далее их утилизацией занимается Docker.



Основные команды для работы с контейнером

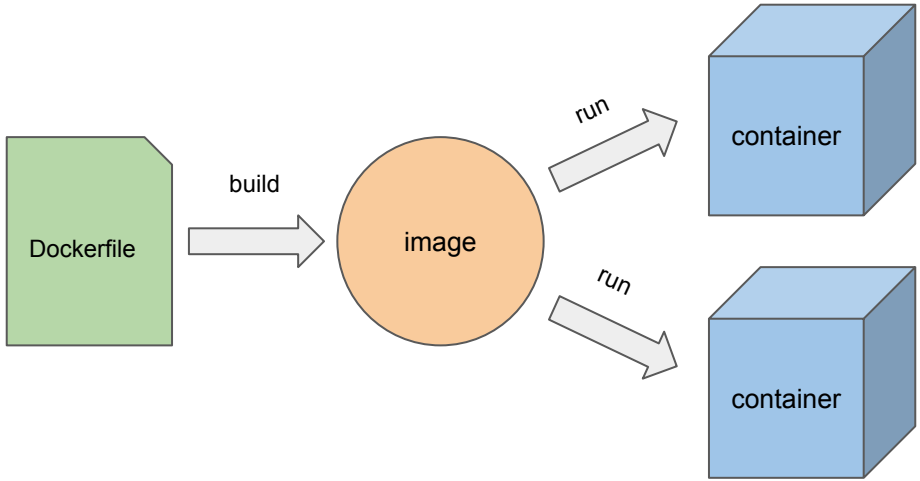
Запуск контейнера	`$ docker run <image>`
Просмотр логов контейнера	`$ docker logs <containerID>`
Запуск команды внутри контейнера	`$ docker exec <containerID> <command>`
Остановка контейнера	`$ docker stop <containerID>`
Удаление контейнера	`$ docker rm <containerID>`
Список всех образов в системе	`$ docker image ls`
Сохранение изменений в образе	`$ docker commit <containerID> <repo:tag>`
Отправка изменений образа в Registry	`$ docker push <repo:tag>`

Запуск контейнера со своими настройками

```
$ cat default.conf
server {
  listen 80 default_server;
  server_name _;
  location / {
    return 200 'Hello world!\n';
  }
}

$ docker run -d -p 8080:80 \
  -v $PWD/default.conf:/etc/nginx/conf.d/default.conf \
  nginx:1.13

$ curl 127.0.0.1:8080

$ docker rm -f <containerID>
```

Важно! Внутри контейнера не копия файла, а именно этот файл.

Docker image



Образ представляет из себя файловую систему с параметрами используемыми при запуске.
Не имеет состояния, не меняется.

Состав image:

1. Операционная система образа
2. Необходимые приложению пакеты (программы)
3. Само приложение и его зависимости

Dockerfile

Тестовый файл со списком шагов для создания образа.
Каждый “шаг” создает отдельный read-only слой в итоговом образе.

Самостоятельная сборка образа

```
$ cat Dockerfile
FROM alpine:3.11
RUN apk add php
CMD [ "php", "-a" ]

$ docker build -t php-cli .
$ docker image ls

$ docker run --rm -it php-cli
php > echo phpversion();
php > exit
```

Основные директивы Dockerfile

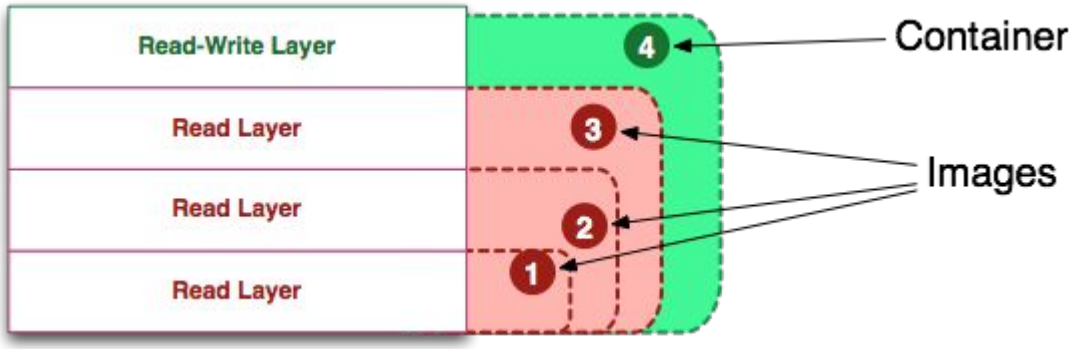
FROM	Указывает базовый (родительский) образ.
LABEL	Описывает метаданные. Например - сведения о том, кто создал и поддерживает образ.
ENV	Устанавливает постоянные переменные среды.
RUN	Выполняет произвольную команду. Обычно используется для установки пакетов.
COPY	Копирует в контейнер файлы и папки.
ADD	Копирует файлы и папки в контейнер, может распаковывать локальные .tar-файлы.
CMD	Описывает команду с аргументами, которую нужно выполнить когда контейнер будет запущен. Аргументы могут быть переопределены при запуске контейнера. В файле может присутствовать лишь одна инструкция CMD.
WORKDIR	Указывает рабочую директорию для следующей инструкции.
ENTRYPOINT	Предоставляет команду с аргументами для вызова во время выполнения контейнера. Аргументы не переопределяются.
EXPOSE	Указывает на необходимость открыть порт.

Контейнер

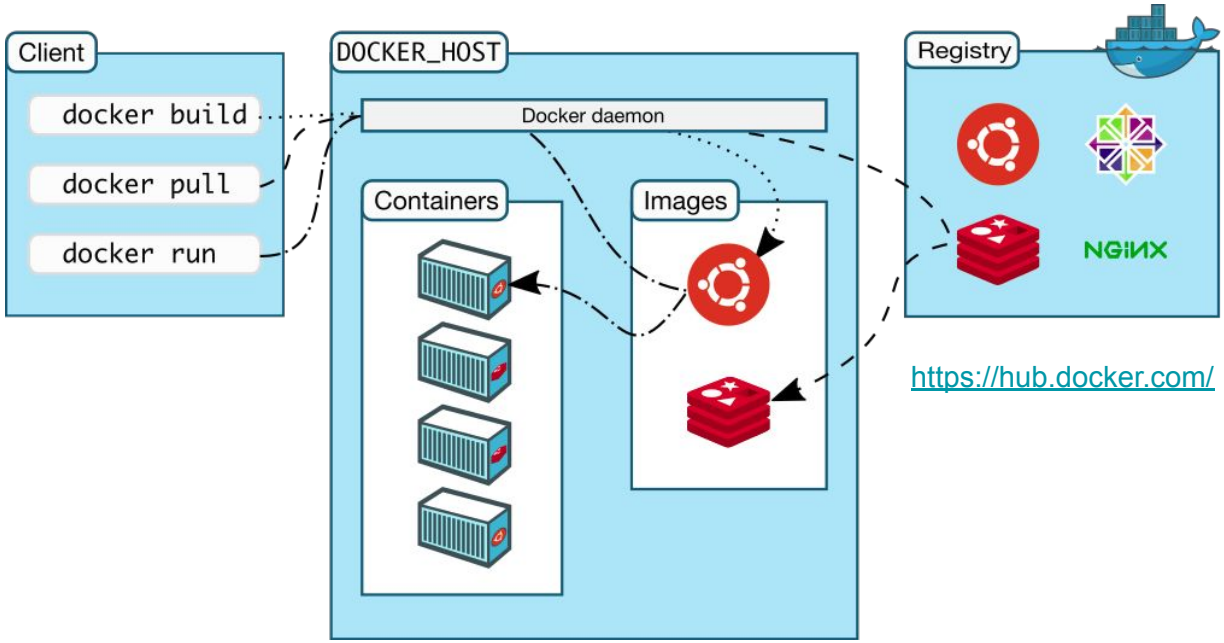
Docker-контейнер строится на основе образа.

Суть преобразования образа в контейнер состоит в добавлении верхнего слоя, для которого разрешена запись. Результаты работы приложения (файлы) пишутся именно в этом RW-слое.

Такой подход позволяет использовать данные одного образа для запуска множества контейнеров.



Docker Engine

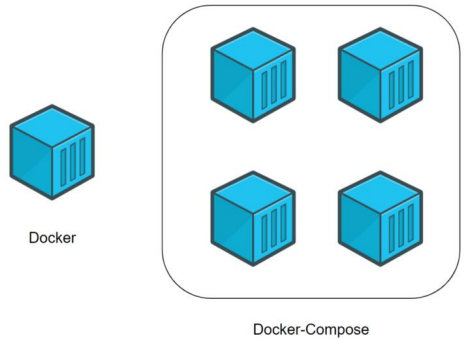


https://hub.docker.com/

Docker compose

Docker Compose - это инструментальное средство, входящее в состав Docker и предназначенное для развертывания проектов, состоящих из нескольких контейнеров.

```
$ sudo pip install docker-compose
```



Пример docker-compose.yml для Yii2-приложения

```
version: '2'

services:

  yii2:
    image: yiisoftware/yii2-php:7.3-fpm
    volumes:
      - ./app:/app

  nginx:
    build:
      context: Dockerfiles/nginx
    image: yii2-demo-nginx
    ports:
      - '3000:80'
    volumes:
      - ./app:/app

  postgresql:
    build:
      context: Dockerfiles/postgresql
    image: yii2-demo-postgresql
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=password
      - POSTGRES_DB=postgres
    volumes:
      - ./pg_data:/var/lib/postgresql/data
    ports:
      - '5432:5432'
```

Основные команды Docker Compose

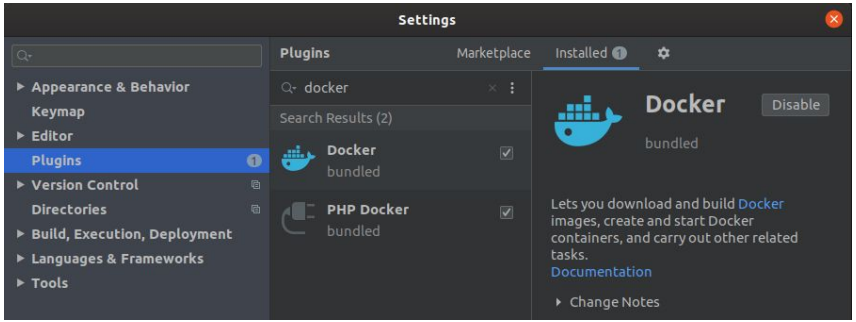
Запуск всех контейнеров	`$ docker-compose up -d`
Просмотр состояния контейнеров	`$ docker-compose ps`
Запуск конкретного контейнера	`$ docker-compose up -d <serviceName>`
Просмотр логов контейнера	`$ docker-compose logs <serviceName>`
Запуск команды внутри контейнера	`$ docker-compose exec <serviceName> <command>`
Остановка и удаление всех контейнеров	`$ docker-compose down`

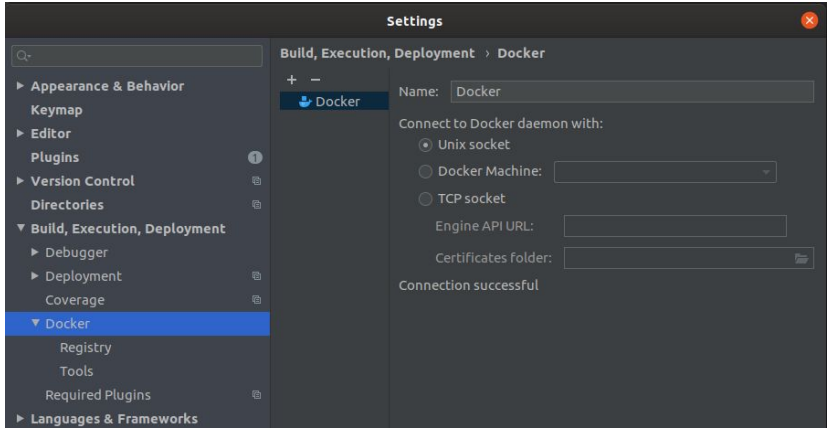
* Все команды выполняются в каталоге, где расположен файл docker-compose.yml

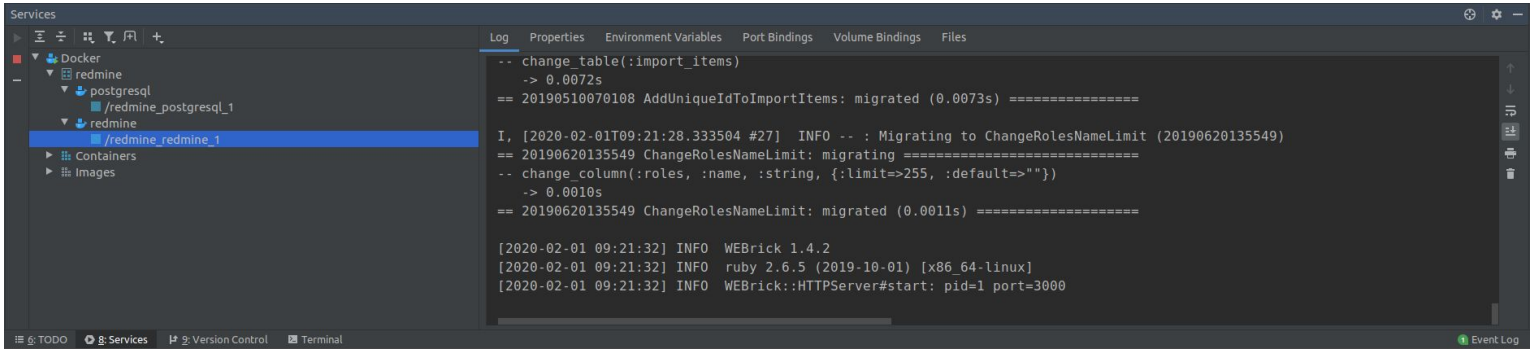
Команды очистки

Общее потребление места	`$ docker system df`
Удаление всех остановленных контейнеров	`$ docker container prune`
Удаление всех образов, не связанных с контейнерами	`$ docker image prune`
Удаление всех томов, не связанных с контейнерами	`$ docker volume prune`
Удаление всего вышеперечисленного	`$ docker system prune`

Плагин “Docker” для PhpStorm







Дополнительная информация и источники

- Официальная документация: https://docs.docker.com/
- Статья “Docker. Начало”: https://habr.com/ru/post/353238/
- Статья “Изучаем Docker, часть 3: файлы Dockerfile”: https://habr.com/ru/company/ruvds/blog/439980/
- Статья “Руководство по Docker Compose для начинающих”: https://habr.com/ru/company/ruvds/blog/450312/
- Статья “Docker Tips: Очистите свою машину от хлама”: https://habr.com/ru/post/486200/
- Видео “Learn Docker in 12 Minutes”: https://www.youtube.com/watch?v=YFl2mCHdv24
- Видео “Docker Compose in 12 Minutes”: https://www.youtube.com/watch?v=Qw9zlE3t8Ko
- **Примеры из презентации:** https://git.cloud-team.ru/lections/docker