



# Administrator Linux.Professional

## Docker: основы работы с контейнеризацией



Проверить, идет ли запись

# Меня хорошо видно && слышно?



Ставим "+", если все хорошо  
"-", если есть проблемы



# Преподаватель

## Лавлинский Николай

**Технический директор «Метод Лаб»**

Более 15 лет в веб-разработке

Преподавал в ВУЗе более 10 лет  
Более 4 лет в онлайн-образовании

Специализация: оптимизация производительности, ускорение сайтов  
и веб-приложений

[https://t.me/methodlab\\_tg](https://t.me/methodlab_tg)

<https://www.methodlab.ru/>

<https://rutube.ru/channel/24617406/>

<https://www.youtube.com/c/NickLavlinsky>

[https://www.youtube.com/@site\\_support](https://www.youtube.com/@site_support)

<https://vkvideo.ru/@methodlab>



# Правила вебинара



Активно  
участвуем



Off-topic обсуждаем  
в ТГ-группе



Задаем вопрос  
в чат или голосом



Вопросы вижу в чате,  
могу ответить не сразу

## Условные обозначения



Индивидуально



Время, необходимое  
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или  
задайте вопрос

# Маршрут вебинара

Знакомство

Что такое контейнеризация

Возможности и особенности Docker

Практика

Рефлексия



# Цели вебинара

После занятия вы сможете

1. Понимать разницу между контейнеризацией и виртуализацией
2. Понимать основные принципы работы Docker
3. Применить на практике основы технологий контейнеризации
4. Познакомиться с технологией Docker-compose



# Смысл

## Зачем вам это уметь

1. В 2019 г. эксперты Gartner говорили, что 30% организаций по всему миру используют контейнерные приложения, а в 2022 г. эта цифра достигает 70%.
2. Контейнеризация — технология, на пике популярности в IT
3. Знание в области контейнеризации поможет в работе, как программиста так и системного администратора. Также этот навык востребован в большинстве современных IT-вакансий

# Контейнеризация

# Контейнеризация vs Виртуализация



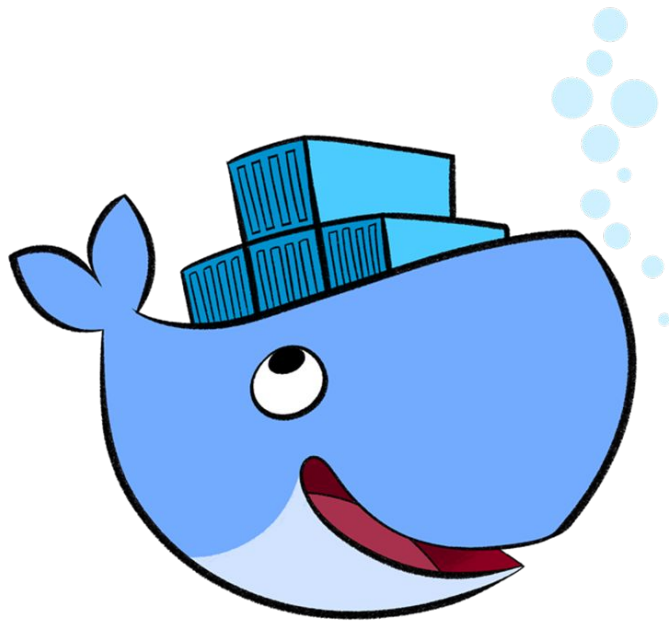
Виртуальные машины



Контейнеры

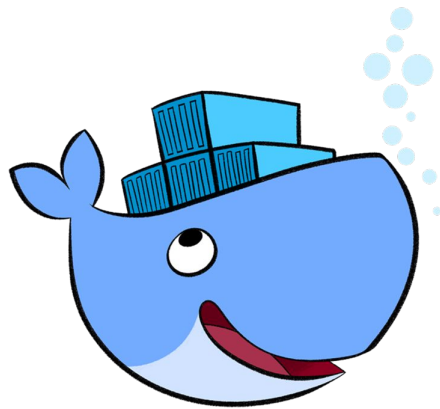
# Что такое Docker?

**Docker** — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер.



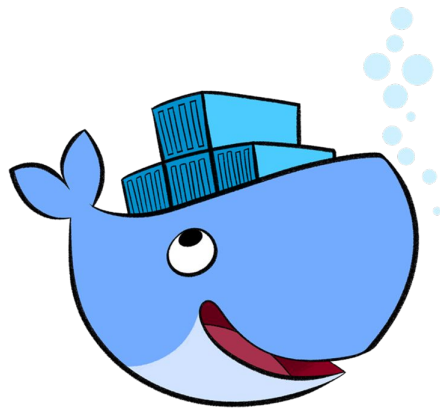
# Особенности Docker

- Один процесс – один контейнер (одна функция – одна ответственность)
- Контейнер self-contained (все что необходимо для его работы есть в самом контейнере, в т.ч. все зависимости)
- Образы не занимают много места на диске



# Из чего состоит Docker?

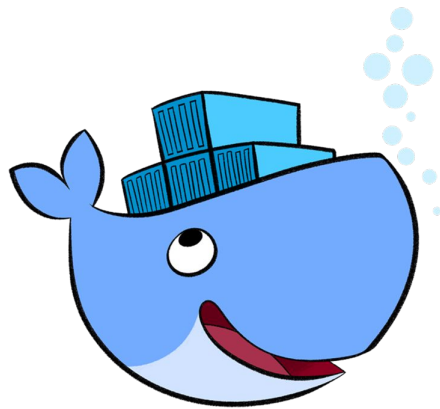
- Серверная часть в виде демона (docker engine)
- API-интерфейс для взаимодействия с docker
- Command line interface (CLI)



# Docker daemon

**Docker daemon** – инструмент для запуска и контроля над контейнерами

- Работает на хостовой машине
- Скачивает образы и запускает контейнеры
- Связывает контейнеры по сети
- Собирает логи
- Создает образы из Dockerfile

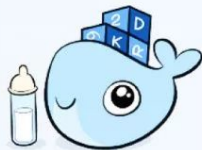


# Основные понятия Docker

- **Docker-файл (Dockerfile)** — скрипт сборки образа
- **Docker-образ (Docker image)** — дистрибутив контейнера
- **Docker-контейнер (Docker container)** — развёрнутый экземпляр образа
- **Том (Volume)** — постоянное хранилище данных для Docker
- **Реестр (Docker registry)** — централизованное хранилище образов



Dockerfile

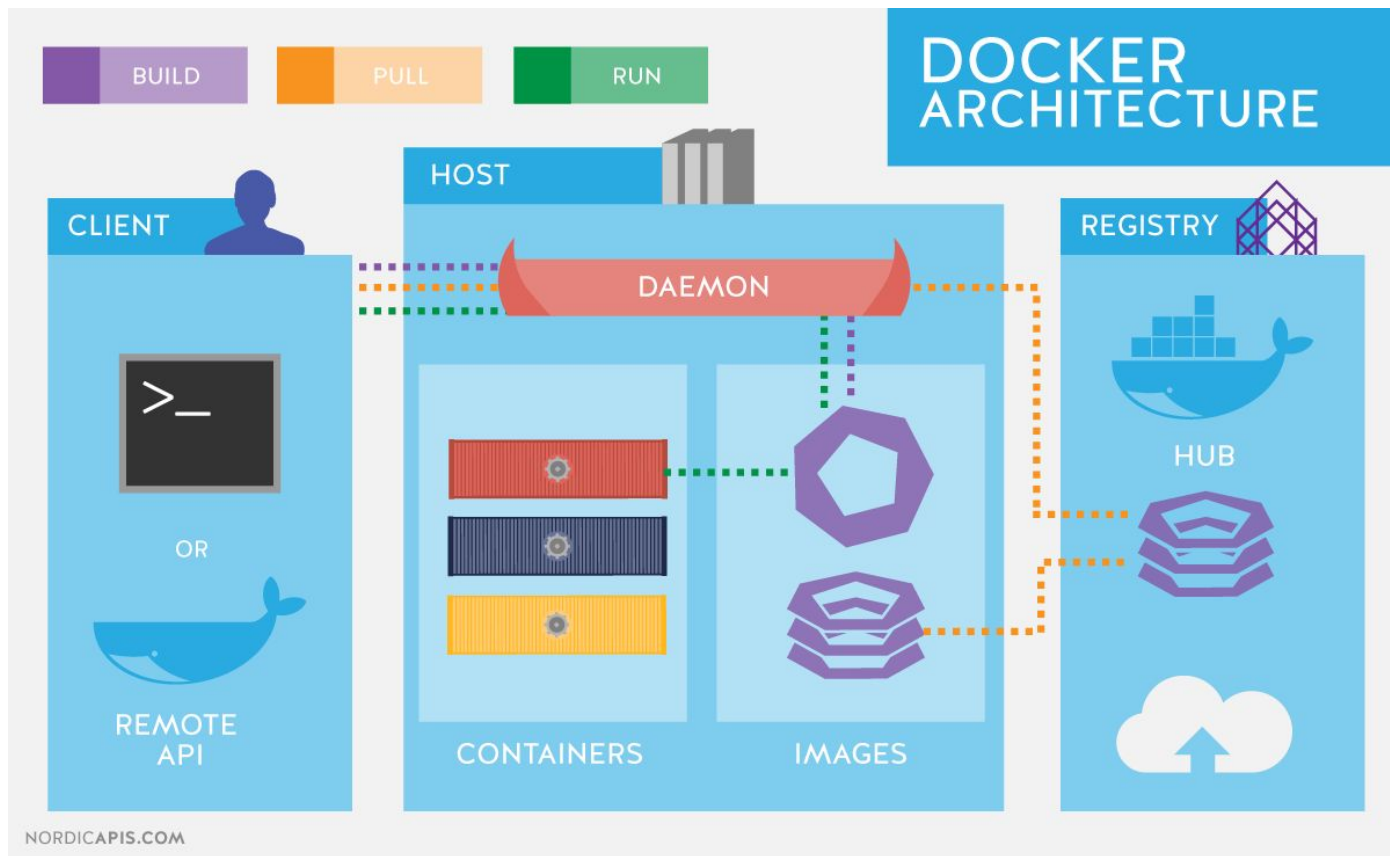


Docker Image

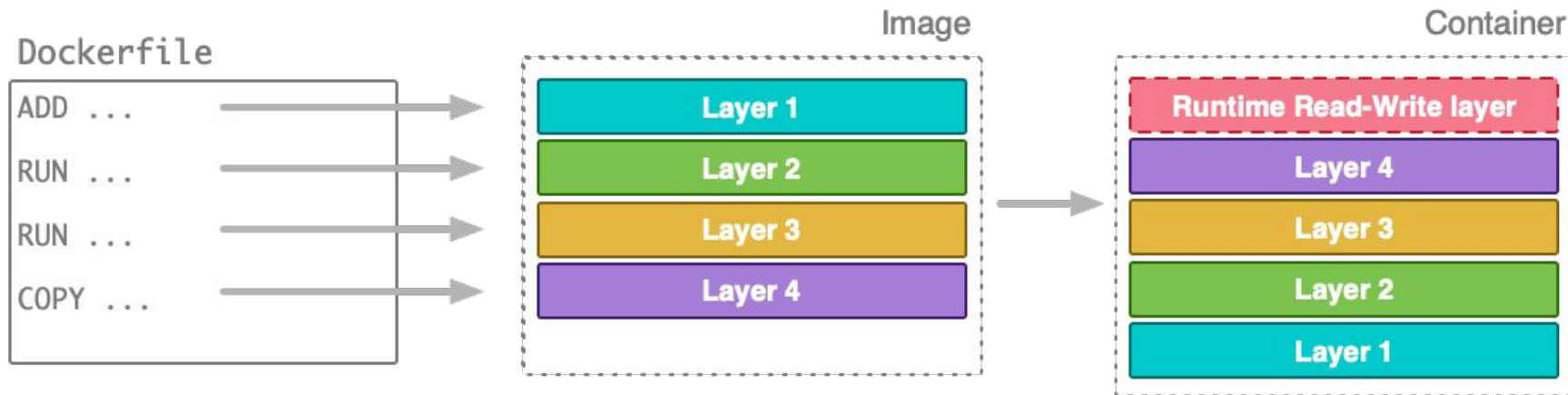


Docker Container

# Архитектура Docker



# Слои образа docker (OverlayFS)



# Вопросы?



Ставим “+”,  
если вопросы есть



Ставим “-”,  
если вопросов нет

# Запуск контейнера в Docker

# Запуск контейнера с Nginx

Detached  
(демон)

Проброс портов  
хост:контейнер

Имя образа  
образ:тэг

```
docker run -d --name nginx1 -p 8090:80 -v /var/www/html:/usr/share/nginx/html nginx
```

Создать и  
запустить  
контейнер

Подключить том  
(volume)  
хост:контейнер

# Запуск и создание: docker run

- **-d** (**--detach**) — запуск в фоне
- **-i** (**--interactive**) — подключение STDIN к контейнеру
- **--name** — имя контейнера
- **-t** (**--tty**) — подключение псевдотерминала к контейнеру
- **-a** (**--attach**) — подключение STDIN, STDOUT, STDERR к контейнеру
- **--rm** — удалить контейнер после остановки
- **-v** (**--mount**) — подключить тома (volume)
- **--network** — указание сети
- **--restart** — политика запуска (**no**, **on-failure[:max-retries]**, **always**, **unless-stopped**)

<https://docs.docker.com/engine/reference/run/>

<https://docs.docker.com/config/containers/start-containers-automatically/>



# Вопросы?



Ставим “+”,  
если вопросы есть



Ставим “-”,  
если вопросов нет



# Сборка контейнера

# Инструкции Dockerfile

- **FROM** — задаёт базовый (родительский) образ.
- **LABEL** — описывает метаданные. Например — сведения о том, кто создал и поддерживает образ.
- **WORKDIR** — задаёт рабочую директорию для следующей инструкции.
- **ENV** — устанавливает постоянные переменные среды.
- **ARG** — задаёт переменные для передачи Docker во время сборки образа.
- **RUN** — выполняет команду и создаёт слой образа. Используется для установки в контейнер пакетов.
- **COPY** — копирует в контейнер файлы и папки.

# Инструкции Dockerfile 2

- **ADD** — копирует файлы и папки в контейнер, может распаковывать локальные .tar-файлы
- **EXPOSE** — указывает на необходимость открыть порт
- **VOLUME** — создаёт точку монтирования для работы с постоянным хранилищем
- **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
- **ENTRYPOINT** — предоставляет команду с аргументами для вызова во время выполнения контейнера. Аргументы не переопределяются

# ENTRYPOINT и CMD

**Инструкция CMD** предоставляет Docker команду, которую нужно выполнить при запуске контейнера. Результаты выполнения этой команды не добавляются в образ во время его сборки. Например, это может быть bash скрипт, или запуск исполняемого файла.

Вот ещё кое-что, что нужно знать об инструкции CMD:

1. В одном файле Dockerfile может присутствовать лишь одна инструкция CMD. Если в файле есть несколько таких инструкций, система проигнорирует все кроме последней.
2. Инструкция CMD может иметь ехес-форму. Если в эту инструкцию не входит упоминание исполняемого файла, тогда в файле должна присутствовать инструкция ENTRYPOINT. В таком случае обе эти инструкции должны быть представлены в формате JSON.
3. Аргументы командной строки, передаваемые docker run, переопределяют аргументы, предоставленные инструкции CMD в Dockerfile.

# ENTRYPOINT и CMD

**Инструкция ENTRYPOINT** позволяет задавать команду с аргументами, которая должна выполняться при запуске контейнера. Она похожа на команду CMD, но параметры, задаваемые в ENTRYPOINT, не перезаписываются в том случае, если контейнер запускают с параметрами командной строки. Вместо этого аргументы командной строки, передаваемые в конструкции вида `docker run my_image_name`, добавляются к аргументам, задаваемым инструкцией ENTRYPOINT. Например, после выполнения команды вида `docker run my_image bash` аргумент `bash` добавится в конец списка аргументов, заданных с помощью ENTRYPOINT.

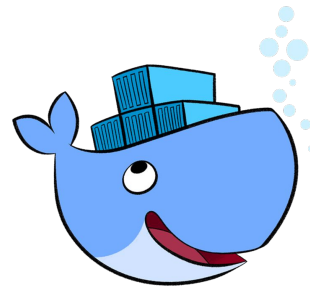
В документации к Docker есть несколько рекомендаций, касающихся того, какую инструкцию, CMD или ENTRYPOINT, стоит выбрать в качестве инструмента для выполнения команд при запуске контейнера:

1. Если при каждом запуске контейнера нужно выполнять одну и ту же команду — используйте ENTRYPOINT.
2. Если контейнер будет использоваться в роли приложения — используйте ENTRYPOINT.
3. Если вы знаете, что при запуске контейнера вам понадобится передавать ему аргументы, которые могут перезаписывать аргументы, указанные в Dockerfile, используйте CMD.

Документация Docker рекомендует использовать ехес-форму ENTRYPOINT: `ENTRYPOINT ["executable", "param1", "param2"]`.

# ENTRYPOINT vs CMD

<https://github.com/nginxinc/docker-nginx/blob/master/Dockerfile-debian.template>



# Пример Dockerfile

```
FROM alpine:3.12
ARG tf_ver=0.12.28
ARG tflint_ver=0.16.2
RUN apk update && apk upgrade
RUN wget https://releases.hashicorp.com/terraform/${tf_ver}/terraform_${tf_ver}_linux_amd64.zip \
    && unzip terraform_${tf_ver}_linux_amd64.zip && rm terraform_${tf_ver}_linux_amd64.zip \
    && mv terraform /usr/local/bin/
RUN wget https://github.com/terraform-linters/tflint/releases/download/v${tflint_ver}/tflint_linux_amd64.zip \
    && unzip tflint_linux_amd64.zip && rm tflint_linux_amd64.zip \
    && mv tflint /usr/local/bin/
RUN echo ${hello} ${tf_ver} ${tflint_ver}
CMD ["/bin/sh"]
```

# Вопросы?



Ставим “+”,  
если вопросы есть

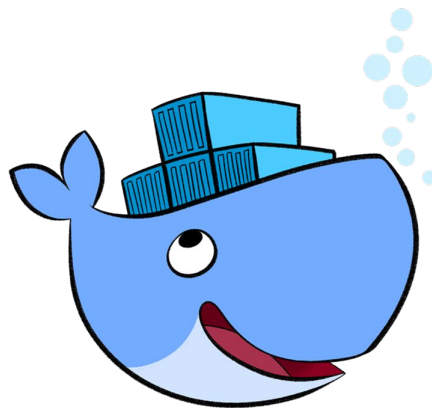


Ставим “-”,  
если вопросов нет

# Практика работы с Docker

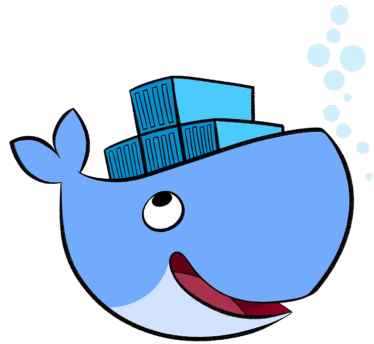
# Основные команды

- **docker run** – создание и первый запуск контейнера из образа
- **docker ps** – показывает список запущенных контейнеров. Ключ `-a` показывает список всех контейнеров на хосте
- **docker logs** – показывает логи контейнера. Ключ `-f` отображает обновления логов в режиме реального времени
- **docker start/stop** – запуск существующего контейнера
- **docker rm** – удаление контейнера
- **docker rmi** – удаление образа



# Основные команды

- **docker build** – собрать образ из Dockerfile
- **docker pull/push** – аналогично системе git, получить или отправить образ в хранилище
- **docker cp** – скопировать файл/директорию в контейнер или наоборот
- **docker exec** – запуск команды в запущенном контейнере
- **docker inspect** – показывает детальную информацию о запущенном контейнере

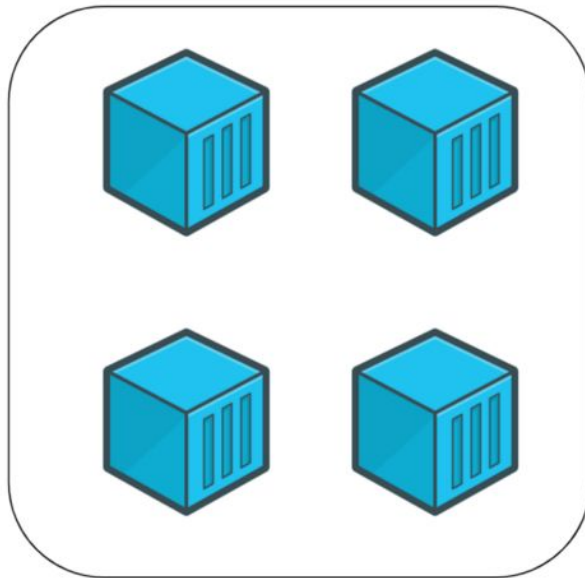


# Docker compose

# Docker compose



Docker

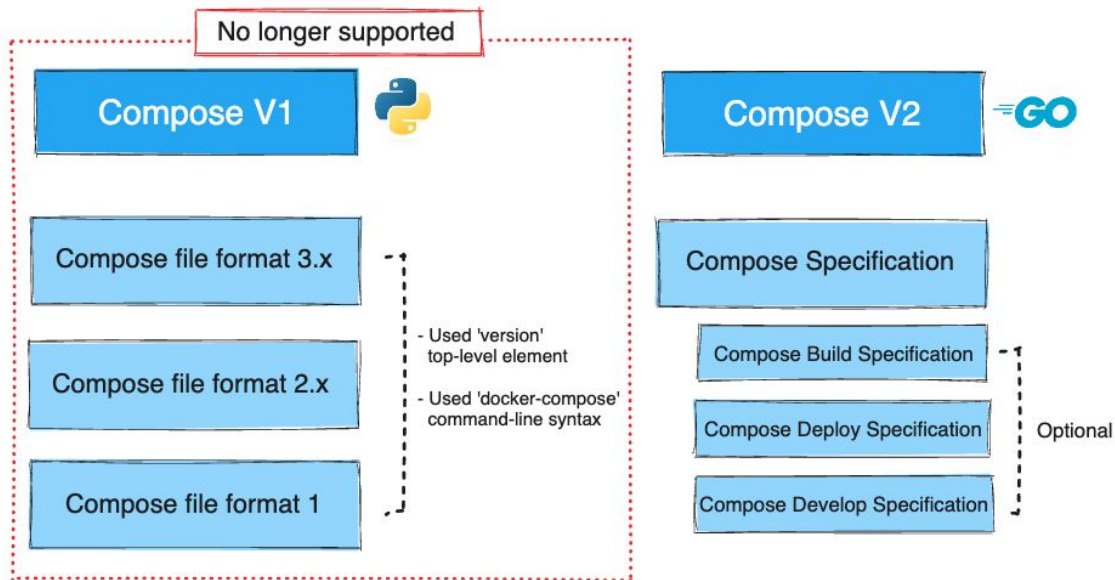


Docker-Compose

<https://docs.docker.com/compose/intro/history/>

[https://doc.owncloud.com/server/next/admin\\_manual/installation/docker/](https://doc.owncloud.com/server/next/admin_manual/installation/docker/)

# Docker compose (v1 и v2)



<https://docs.docker.com/compose/intro/history/>

[https://doc.owncloud.com/server/next/admin\\_manual/installation/docker/](https://doc.owncloud.com/server/next/admin_manual/installation/docker/)

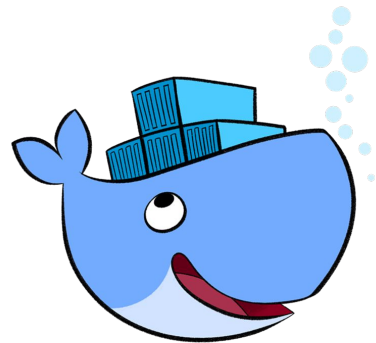
# Docker compose — redmine

```
version: '3.1'
```

```
services:
```

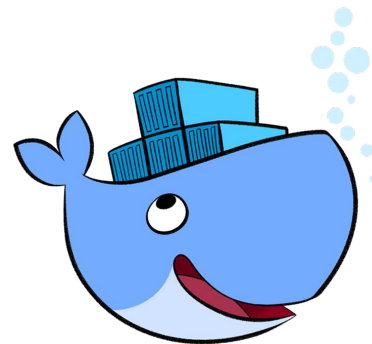
```
  redmine:  
    image: redmine  
    restart: always  
    ports:  
      - 8080:3000  
    environment:  
      REDMINE_DB_MYSQL: db  
      REDMINE_DB_PASSWORD: example  
      REDMINE_SECRET_KEY_BASE: supersecretkey
```

```
  db:  
    image: mysql:5.7  
    restart: always  
    environment:  
      MYSQL_ROOT_PASSWORD: example  
      MYSQL_DATABASE: redmine
```



# Docker compose — Wordpress

```
wordpress:
  depends_on:
    - db
  image: wordpress:5.1.1-fpm-alpine
  container_name: wordpress
  restart: unless-stopped
  env_file: .env
  environment:
    - WORDPRESS_DB_HOST=db:3306
    - WORDPRESS_DB_USER=$MYSQL_USER
    - WORDPRESS_DB_PASSWORD=$MYSQL_PASSWORD
    - WORDPRESS_DB_NAME=wordpress
  volumes:
    - wordpress:/var/www/html
  networks:
    - app-network
```



<https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-with-docker-compose>  
<https://timeweb.cloud/tutorials/docker/kak-ustanovit-wordpress-s-pomoshchyu-docker>



# Список материалов для изучения

1. **Docker install** - <https://docs.docker.com/get-docker/>
2. **Portainer** - <https://docs.portainer.io/start/install-ce/server>

# Вопросы?



Ставим “+”,  
если вопросы есть



Ставим “-”,  
если вопросов нет

# Практика

# Домашнее задание

1. Установите Docker на машину.
2. Установите Docker Compose — как плагин.
3. Создайте свой кастомный образ nginx на базе alpine. После запуска nginx должен отдавать кастомную страницу (достаточно изменить дефолтную страницу nginx).
4. Определите разницу между контейнером и образом. Вывод опишите в домашнем задании.
5. Ответьте на вопрос: Можно ли в контейнере собрать ядро?



Сроки выполнения: указаны в личном кабинете



# Рефлексия

# Цели вебинара

## Проверка достижения целей

1. Понимать разницу между контейнеризацией и виртуализацией
2. Понимать основные принципы работы Docker
3. Применить на практике основы технологий контейнеризации
4. Познакомиться с технологией Docker compose



# Вопросы для проверки

По пройденному материалу всего вебинара

1. Отличия виртуализации от контейнеризации
2. Что такое Docker и с чем его едят?
3. Что такое Dockerfile? Как запустить контейнер?
4. Что такое Docker compose?



# Рефлексия



Насколько тема была для вас сложной?



Как будете применять на практике то, что узнали на вебинаре?

**Заполните, пожалуйста,  
опрос о занятии  
по ссылке в чате**

Спасибо за внимание!

# Приходите на следующие вебинары



**Лавлинский Николай**

Технический директор «Метод Лаб»

[https://t.me/methodlab\\_tg](https://t.me/methodlab_tg)

<https://www.methodlab.ru/>

<https://rutube.ru/channel/24617406/>

<https://www.youtube.com/c/NickLavlinsky>

[https://www.youtube.com/@site\\_support](https://www.youtube.com/@site_support)

<https://vkvideo.ru/@methodlab>

