OTUS

Онлайн образование

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Тема вебинара

Docker - основы контейнеризации

Шиков Станислав

Начальник отдела автоматизации ООО “Кодер”

Эл. почта stenlav@mail.ru

Правила вебинара

Активно
участвуем

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

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. Размер мирового рынка коммерческого контейнерного программного обеспечения будет расти в 2018-2023 гг. в среднем на 30% ежегодно, превысив к концу периода $1,6 млрд — такой прогноз дается в отчете Technology Multi-Tenant Server Software Market Tracker аналитиков IHS Markit.
3. Знание в области контейнеризации поможет в работе, как программиста так и системного администратора. Также этот навык востребован в большинстве современных IT вакансий

Let’s Go

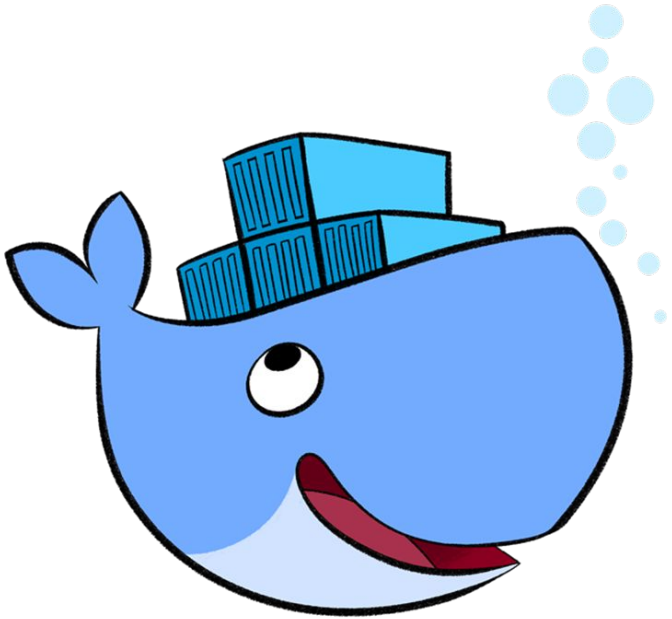
Контейнеризация vs Виртуализация

Виртуальные машины

Контейнеры

Что такое Docker?

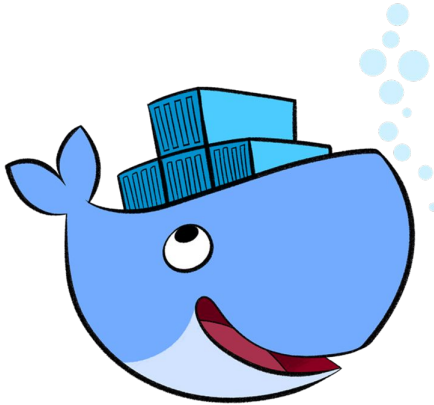
Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер.



Ключевые особенности Docker

Docker позволил использовать совершенно иной подход к построению архитектуры приложений

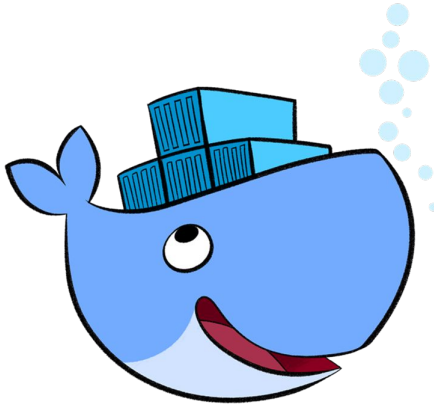
- Один процесс – один контейнер (одна функция – одна ответственность)
- Контейнер self-contained (все что необходимо для его работы есть в самом контейнере, в т.ч. все зависимости)
- Образы не занимают много места на диске (напр., alpine)



Из чего состоит Docker?

Docker, как технологию контейнеризации с открытым исходным кодом, представляет собой клиент-серверное приложение. Условно его можно разделить на три составляющие:

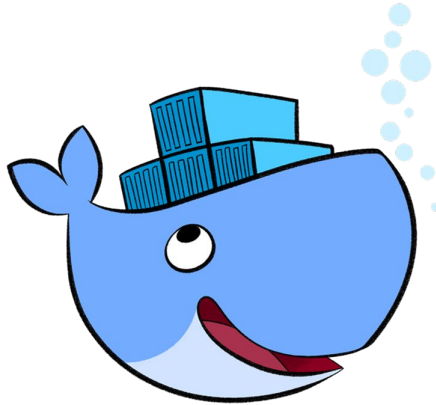
- Серверную часть в виде демона (dockerd)
- API-интерфейса для взаимодействия с docker
- Command line interface (CLI)



Docker daemon

Docker daemon – инструмент для запуска и контроля над контейнерами

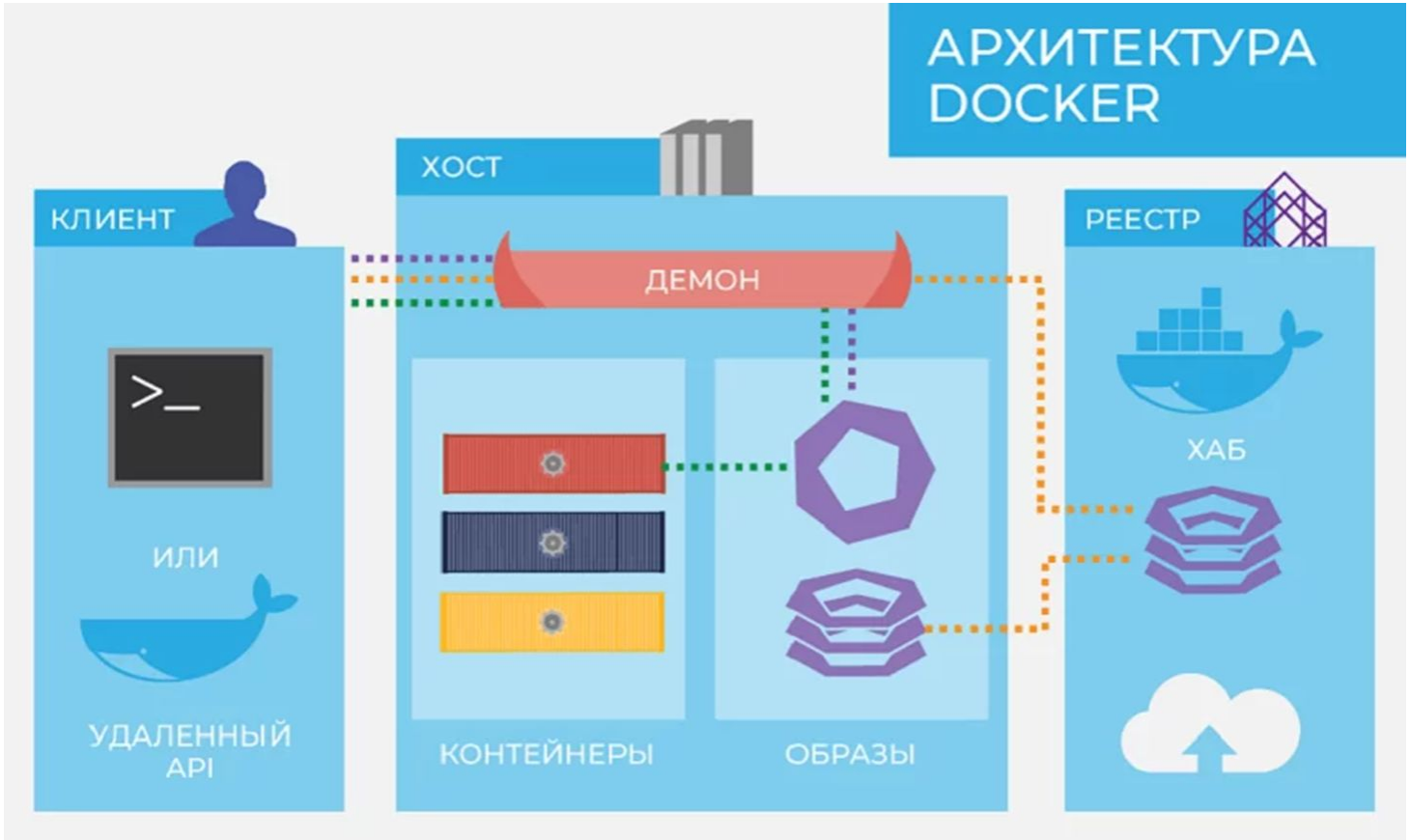
- работает на хостовой машине
- скачивает образы и запускает контейнеры
- связывает контейнеры по сети
- собирает логи
- создает образы из Dockerfile



Немного скучных слов

- Docker-файл (Docker-file)
- Docker-образ (Docker-image)
- Docker-контейнер (Docker-container)
- Том (Volume)

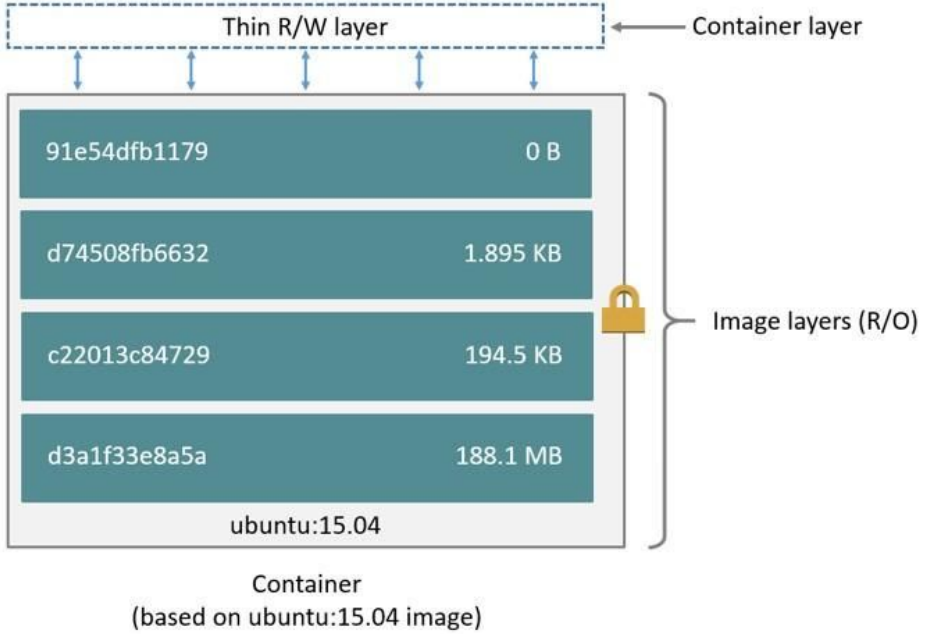
Как работает Docker



Структура контейнера

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Dockerfile



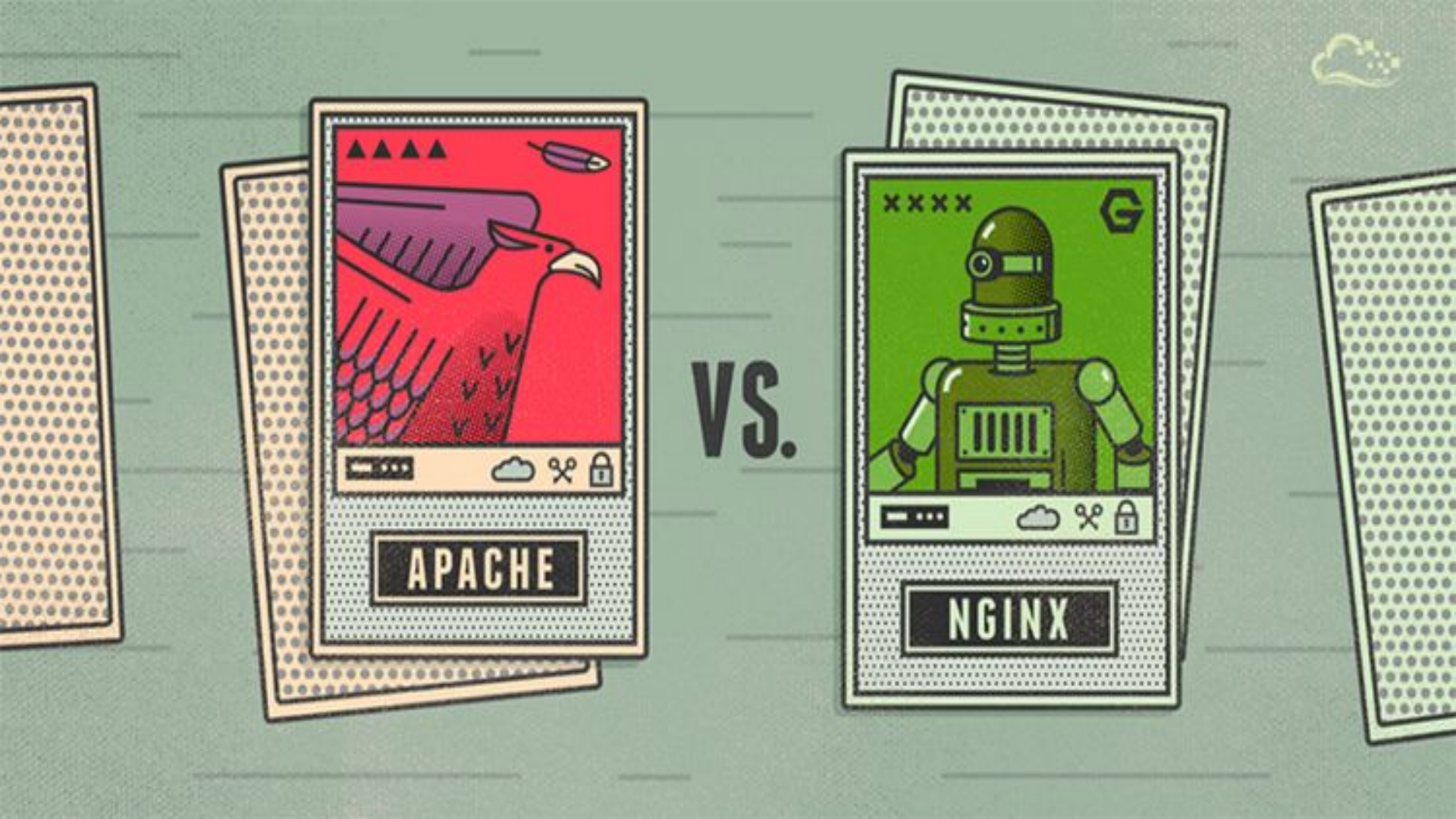
Инструкции Dockerfile

1. `FROM`— задаёт базовый (родительский) образ.
2. `LABEL` — описывает метаданные. Например — сведения о том, кто создал и поддерживает образ.
3. `ENV` — устанавливает постоянные переменные среды.
4. `RUN` — выполняет команду и создаёт слой образа. Используется для установки в контейнер пакетов.
5. `COPY` — копирует в контейнер файлы и папки.
6. `ADD` — копирует файлы и папки в контейнер, может распаковывать локальные .tar-файлы.
7. `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`.
8. `WORKDIR` — задаёт рабочую директорию для следующей инструкции.
9. `ARG` — задаёт переменные для передачи Docker во время сборки образа.
10. `ENTRYPOINT` — предоставляет команду с аргументами для вызова во время выполнения контейнера. Аргументы не переопределяются.
11. `EXPOSE` — указывает на необходимость открыть порт.
12. `VOLUME` — создаёт точку монтирования для работы с постоянным хранилищем.

Пример Dockerfile

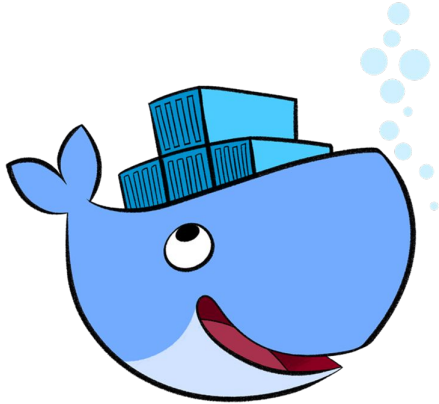
```
FROM alpine:3.12
ARG tf_ver=0.12.28
ARG tflint_ver=0.16.2
RUN apk update && apk upgrade
RUN wget https://releases.hashicorp.com/terraform/${tf_ver}/terraform_${tf_ver}_linux_amd64.zip \
    && unzip terraform_${tf_ver}_linux_amd64.zip && rm terraform_${tf_ver}_linux_amd64.zip \
    && mv terraform /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 apk add --no-cache curl
CMD ["/bin/sh"]
```

Немного практики...


APACHE
VS.
NGINX

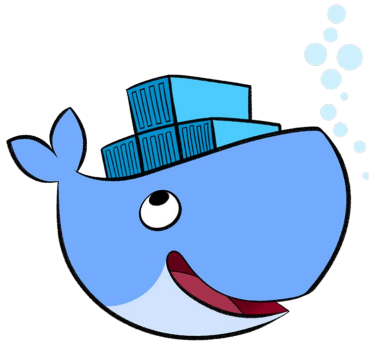
Основные команды

- **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 “dockername/hash” –** показывает логи контейнера. Ключ -f отображает обновления логов в режиме реального времени
- **docker start/stop “dockername/hash” –** запуск существующего контейнера
- **docker rm –** удаление контейнера
- **docker rmi –** удаление образа



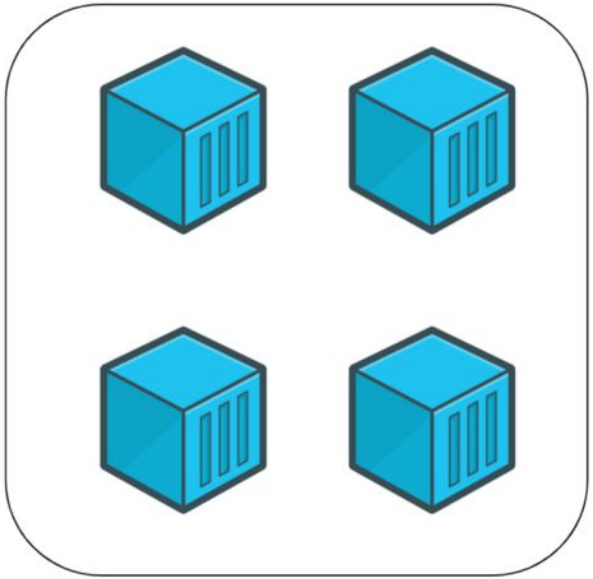
Основные команды

- **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 “dockername/hash” –** показывает детальную информацию о запущенном контейнере

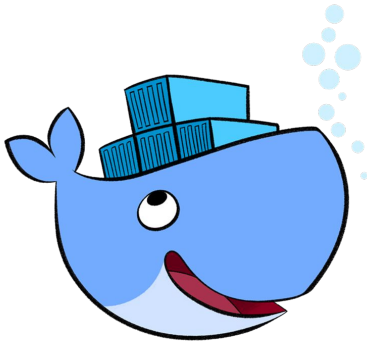


Docker-compose

Docker



Docker-Compose



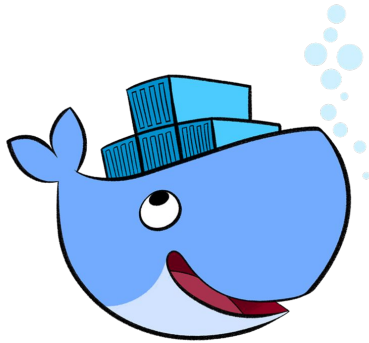
Docker-compose

```
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
```



Список материалов для изучения

1. **Docker install** - https://docs.docker.com/get-docker/

2. **Portainer** - https://docs.portainer.io/start/install-ce/server

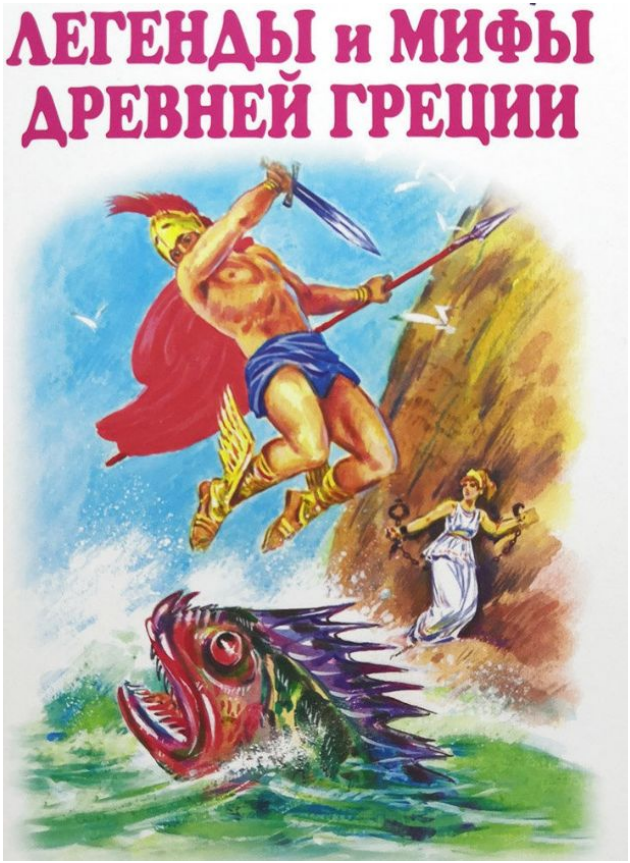
Рефлексия

Мифология

Миф 1
Я вообще не могу посмотреть внутрь контейнера!

Миф 2
Docker под macOS и Windows толком не работает

Миф 3
Docker — это “enterprise” и “devops”



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?

Домашнее задание

1. Написать Dockerfile на базе apache/nginx который будет содержать две статичные web-страницы на разных портах. Например, 80 и 3000.
2. Пробросить эти порты на хост машину. Обе страницы должны быть доступны по адресам localhost:80 и localhost:3000
3. Добавить 2 вольюма. Один для логов приложения, другой для web-страниц.

Доп.*

1. Написать Docker-compose для приложения Redmine, с использованием опции build.
2. Добавить в базовый образ redmine любую кастомную тему оформления.
3. Убедиться что после сборки новая тема доступна в настройках.

https://www.redmine.org/projects/redmine/wiki/themes

https://github.com/farend/redmine_theme_farend_bleuclair#demo

Спасибо за внимание!

Приходите на следующие вебинары

Шиков Станислав Александрович

Руководитель отдела автоматизации ООО “Кодер”

Эл. почта stenlav@mail.ru