Как упаковать приложение в Docker

Что эксперт готовит к уроку

Опорный конспект с подсказками для демонстрации. Пример простого приложения для упаковки в контейнер (желательно, написанного на Python) и развертывания. Текст и решение домашнего задания по созданию docker-файлов.

Какие вопросы нужно раскрыть в уроке

▼ Как работать с docker-файлами

- зачем он нужен докер-файл;
- как выглядят докер-файлы:
  - что в них находится,
  - что означают поля конфигурации и как с ними работают;
- как копировать docker-файлы;
- как менять поля внутри docker-файлов (например, поменять порт);
- как написать простой docker-файл (1 контейнер);

▼ Как упаковать приложение в docker-контейнер

- как установить Docker на компьютер;
- как выглядит интерфейс Docker;
- как создать, идентифицировать, остановить или удалить контейнер

```
docker run -d -p 80:80 docker/[image name]
docker ps
docker stop <container-id>
docker rm <container-id>
```

- как создать образ контейнера для приложения:

    - как создать docker-файл и настроить поля (объясните, что значат основные поля, какие зависимости необходимо скачать),
    - как создать образ с помощью `docker build`,

```
docker build -t [image name]
```

    - в чем ограничения `docker build`;
- как сопоставить порты и запустить приложение:
    - объясните, что несколько контейнеров не могут использовать один и тот же порт;

```
docker run -d -p 3000:3000 [image name]
```

    - объясните, зачем нужно останавливать контейнеры и покажите, как это сделать;
- как проверить, что приложение работает верно;
- как обновить код приложения внутри контейнера.

▼ Как упаковать простое приложение в docker-контейнер с помощью Docker Compose

- что такое Docker Compose и `docker-compose.yml`;
- как выглядит процесс упаковки приложения с помощью Docker Compose (концептуально);
- как используют `docker-compose.yml` файлы;
- как выглядит `docker-compose.yml` файл и как его читать;
- как заполнять основные поля в `docker-compose.yml` файле.

> 💡 Укажите поля, на которые нужно обращать внимание Junior-сотрудникам (разработчикам, автотестировщикам, системным архитекторам) и которые, возможно, придется самостоятельно менять.

Домашнее задание

1. Создайте docker-файл и настройте поля с учетом представленных факторов.

   > 💡 1. Подготовьте приложение (желательно, на Python) и описание окружения, чтобы студенты должны создать docker-файл.
   > 2. Подготовьте эталонное решение, в котором вы верно заполнили docker-файл.
   > 3. На видео объясните, как вы заполнили файл. Проведите небольше демо: запустите контейнер в виртуальное окружение, чтобы показать, что все верно настроено.

2. Заполните поля в предоставленном docker-compose файле.

   > 💡 1. Подготовьте `docker-compose.yml` файл и список настроек, которые студенты должны указать. Подготовьте контейнер с приложением для этого задания.
   > 2. Подготовьте эталонное решение, в котором вы верно заполнили `docker-compose.yml` файл.
   > 3. На видео объясните, как вы заполнили файл. Проведите небольше демо: запустите контейнер в тестовое окружение, чтобы показать, что все верно настроено.

Во время съемки важно

- объяснять темы с помощью схем, на примерах с кодом (где это уместно);

- объяснять логику того, что вы показываете (проговаривать алгоритм действий);
- давать упражнения, чтобы учащиеся попробовали сами что-то сделать, а потом показывать «эталонное решение» и объяснять нюансы;
- учить верной терминологии;
- объяснять технические термины простым языком;
- **дублировать технические термины на русском и английском языках.**

Источник, который мы использовали

https://learn.microsoft.com/ru-ru/visualstudio/docker/tutorials/docker-tutorial