

Создание образов Docker с помощью Dockerfile

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье описывается внутренняя структура образов и контейнеров в Docker, а также метод создания собственных образов. Процесс создания образов осуществляется с помощью файла инструкции Dockerfile и инструмента монтирования образа Docker. Результатом работы станет описание процесса создания инструкций для разработки нового образа на примерах приложений Python и Nodejs.

Ключевые слова: Docker, образ, контейнер, контейнеризация.

Creating Docker images using Dockerfile

Vikhlyaev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article describes the internal structure of images and containers in Docker, as well as a method for creating your own images. The process of creating images is carried out using the Dockerfile instruction file and the Docker image mount tool. The result of the work will be a description of the process of creating instructions for developing a new image using examples of Python and Nodejs applications.

Keywords: Docker, image, container, containerization.

1 Введение

1.1 Актуальность

Создание собственных образов с использованием Dockerfile является важным аспектом современной разработки программного обеспечения, который значительно упрощает управление зависимостями и средой выполнения приложений. Особенность использования Dockerfile заключается в его декларативной природе: разработчик описывает желаемую конфигурацию среды и порядок ее построения. Это повышает воспроизводимость сборок, снижая вероятность ошибок, вызванных различиями в окружении. При этом Dockerfile предоставляет гибкость в управлении слоями файловой системы, что позволяет оптимизировать размер итогового образа и сократить время сборки за счет использования кэширования. Важным элементом разработки Dockerfile является использование базовых образов, которые служат отправной точкой для

создания кастомизированных конфигураций. Это упрощает процесс разработки, так как разработчику достаточно указать только те изменения, которые требуются для работы приложения.

1.2 Обзор исследований

М.В.Слива, Л.В.Левкин реализовали метод контейнеризации приложений с помощью платформы Docker [1]. А.П.Голушко, В.Г.Жуков исследовали безопасность контейнеров Docker на этапе их создания и запуска [2]. Д.Я.Смирнов, А.Д. Ишкова, О.Г. Яскевич провели научную работу по отладке, тестированию и развёртыванию приложений при помощи Docker [3]. К.В.Феофантов, А.В.Власов, Г.И.Афанасьев описал создание docker-образа postgresql [4]. В.В.Евстратов сравнил технологии контейнеризации и виртуализации [5].

1.3 Цель исследования

Цель исследования – описать процесс создания образа с помощью инструкций Dockerfile.

2 Материалы и методы

Для работы используется программа на стороне клиента Docker Desktop и самостоятельно созданный текстовый файл Dockerfile.

3 Результаты и обсуждения

Чтобы создать контейнер, нужен образ. Образ можно создать с нуля или использовать готовый. Помимо этого, есть возможность создать собственный образ из готового образа. В Docker образ представляет собой набор слоёв. Слои являются набором файлов, наложенных на предыдущий слой. Сам Docker использует ядро Linux и его файловую систему. У каждого образа есть первый базовый слой, как правильно операционная система один из дистрибутивов Linux. Так как Docker уже имеет ядро Linux, дистрибутив накладывается на него. Именно поэтому, несмотря на то, что каждый контейнер имеет собственную операционную систему, он занимает мало места в диске. Изоляция контейнеров друг от друга обусловлена механизмами namespaces и control groups доступными в Linux.

Следующий слой также набор файлов. Это могут быть интерпретаторы языков программирования Python, Nodejs или база данных. То есть набор файлов, установленных в определённых директориях одного из предыдущих слоёв. В данном примере, файлами входящие в пакет Nodejs установлены на операционной системе. Таким образом, при скачивании образа Nodejs доступен дистрибутив Linux с установленным интерпретатором Nodejs. Слои в готовом образе нельзя изменять, но можно поверх добавить собственный набор слоёв. Например, директорию с программой на языке JavaScript (рис.1).

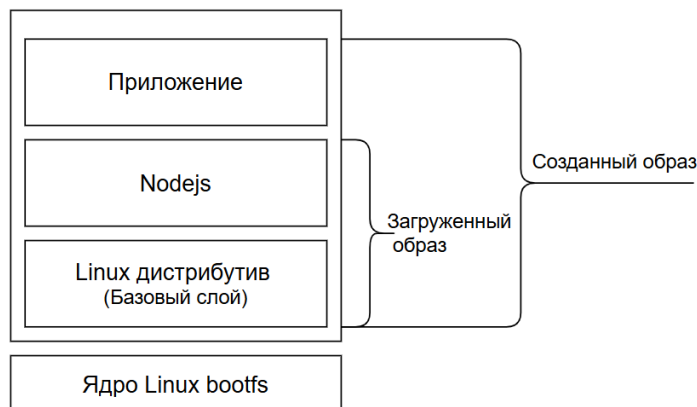


Рис. 1. Последовательность слоёв в Docker контейнере

Для создания собственного образа, на основе существующего используется файл `Dockerfile`. В нём верхнем регистре написаны команды, создающие новый слой. При создании нового слоя, как правило, создаётся группа слоёв. Каждый такой слой кэшируется и при повторном создании образа кэшированные слои подгружаются, за счёт чего процесс тратит меньше времени и ресурсов. Если какой-нибудь из слоёв изменяется процесс переборки, начинается с самого нижнего изменённого слоя.

Чтобы создать образ приложения `Nodejs` сначала требуется загрузить готовый образ, используя команду «FROM». Здесь пишется название образа и тег. Тег означает версию, помимо цифр может быть словом, например, можно использовать тег `latest`, тогда загрузится самая последняя версия образа. Тег `latest` является тегом по умолчанию, если не указать версию образа загрузится самая новая версия.

Затем через команду «WORKDIR» можно создать каталог, в котором будет, размещается проект. Также эта директория станет целевой, а до этого курсор был в корне.

Команда «COPY» копирует указанные файлы. В данном случае конфигурационные файлы `package.json` и `package-lock.json`. Они содержат информацию о сторонних модулях, присевающих в программе.

Команда «RUN» запускает скрипт в терминале. Команда «`npm install`» интерпретируется менеджером пакетов NPM, который уже установлен вместе с `Nodejs` в готовый образ. Таким образом, загрузятся все зависимости проекта. Далее копируются все файлы проекта. Файлы `package.json` и `package-lock.json` уже есть в целевой директории, а значит, скопированы не будут или скопируются с заменой.

Проект загружен, значит можно запускать через команду «CMD». Команда принимает список аргументов. Иными словами, сколько слов требуется для запуска программы разделённых пробелом, столько элементов в массиве должно быть.

Стоит отметить, что можно использовать команду «COPY» только один раз и загрузить все файлы сразу, а после установить зависимости. Но, как упоминалось, Docker кэширует все слои и при изменении одного слоя слои стоящие до него просто подгружаются из кэша. Если нужно будет добавить или изменить функцию даже в одном файле потребуется заново пересоздавать образ. При попытке пересоздать слой все слои до «RUN npm install» включительно будут взяты из кэша и только следующие будут пересобраны, конечно, если не потребуются добавление новых зависимостей. Иными словами, это действие для ускорения сборки (рис.2).

```
FROM node:20.16
WORKDIR /app
COPY package*.json .
RUN npm install
COPY . .
CMD ["npm", "start"]
```

Рис. 2. Содержание Dockerfile для создания образа приложения Nodejs

На примере создания образа для приложения Python, устанавливается образ с теком alpine, что значит минимальная по размеру версия. Далее создаётся рабочая директория. Если нужны дополнительные зависимости, то они устанавливаются с помощью менеджер Python пакетов pip. Например, модуль pymongo для доступа к базе данных MongoDB. В данном случае так же как для Nodejs можно было иметь файл со списком зависимостей. Затем копирование всех файлов в рабочую директорию и запуск (рис.3).

```
FROM python:alpine
WORKDIR /app
RUN pip install pymongo
COPY . .
CMD ["python", "main.py"]
```

Рис. 3. Содержание Dockerfile для создания образа приложения Python

Создание образа запускается после запуска команды «Docker build» с указанием пути к Dockerfile, если по каким-то причинам имя файла отличается нужно применить флаг «-f» с указанием имени. Также стоит использовать флаг «-t» для указания имени образа с тегом (name_image:1.0.0). Тег опционален, если он опускается, то Docker

автоматически добавит тег latest. При этом при пере сборке образ с одинаковым именем и тегом пере запишется, а не создастся новый рядом.

Все образы недоступны для записи, образ создаётся на основе контейнера. После запуска команды «`Docker build`» из базового образа, которым является последний образ с группой слоёв, создается контейнер. К нему добавляется слой для записи, новые файлы. В конце вызывается команда `docker commit`, так создаётся новый образ. Когда приложению в рабочем контейнере нужно создать рядом файл, он создаётся именно в верхнем слое для записи, директории операционной системы.

4 Выводы

В результате исследования был подробно описан процесс создания контейнеров в Docker. Рассмотрено структурное содержание файла Dockerfile. Приведены примеры инструкции для создания образов программ Python и Nodejs. Представлено внутреннее содержание образов и контейнеров.

Библиографический список

1. Слива М.В., Левкин Л.В. Контейнеризация приложений с помощью платформы Docker // В сборнике: Современное программирование. Материалы IV Международной научно-практической конференции. Под общей редакцией Т.Б. Казиахмедова. Нижневартонск, 2022. С. 52-57.
2. Голушко А.П., Жуков В.Г. Информационная безопасность контейнеров Docker на этапе их создания и запуска // В сборнике: Решетневские чтения. Материалы XXVI Международной научно-практической конференции, посвященной памяти генерального конструктора ракетно-космических систем академика М.Ф. Решетнева. В 2-х частях. Под общей редакцией Ю.Ю. Логинова. Красноярск, 2022. С. 389-392.
3. Смирнов Д.Я., Ишкова А.Д., Яскевич О.Г. Отладка, тестирование и развёртывание приложений при помощи Docker // В сборнике: Интеллектуальные информационные системы. Труды Международной научно-практической конференции. Воронеж, 2023. С. 152-154.
4. Феофантов К.В., Власов А.В., Афанасьев Г.И. Создание docker-образа postgresql // Современные научные исследования и инновации. 2017. № 2. С. 86-89.
5. Евстратов В.В. Контейнеризация как современный способ виртуализации // Молодой ученый. 2020. № 49 (339). С. 7-9.
6. Солонько М.К., Краснов К.А. Docker-контейнеры // Телекоммуникации и информационные технологии. 2020. Т. 7. № 1. С. 65-69.
7. Егоров Б.Ю., Таций Е.В., Анохина И.Ю. Докеризация python приложения // В сборнике: Современные информационные технологии в образовании и научных исследованиях (СИТОНИ-2023). Сборник материалов VIII Всероссийской научно-технической конференции. Донецк, 2023. С. 98-102.