**Docker**

Оглавление

[DOCKERFILE 1](#__RefHeading___Toc1169_1028856298)

[Различия между ENTRYPOINT и CMD 2](#__RefHeading___Toc1171_1028856298)

[4. Комбинирование ENTRYPOINT и CMD 2](#__RefHeading___Toc1173_1028856298)

[MOUNT FILES AND DISKS 4](#__RefHeading___Toc1175_1028856298)

[VOLUMES 4](#__RefHeading___Toc1177_1028856298)

[volume in Dockerfile 4](#__RefHeading___Toc1179_1028856298)

[DOCKER COMPOSE 5](#__RefHeading___Toc1181_1028856298)

[Команды 5](#__RefHeading___Toc1183_1028856298)

[Примеры: 7](#__RefHeading___Toc1185_1028856298)

# DOCKERFILE

# - комментарий

ARG CODE\_VERSION=alpine – аргумент сборки только на момент сборки

FROM node:$CODE\_VERSION -из чего образ сделан

Label version=”1.0” -указывает версию

MAINTAINER Vladimir -кто автор образа

USER root -под кем будет запускаться

WORKDIR /opt/app – создется папка где будет запускаться, можно использовать несколько

ARG MY\_ARG – можно передать в него значение во время сборки и использовать

ADD \*.json ./ - копирует все json\

ADD –-chown=root files\* /somedir

ADD my\_arc.tar.gz /somedir/ -разархивирует архив

ADD [http://url.tar.gz](http://url.tar.gz/) /somedir/

COPY . . - копирует из текущей директории в workdir рекомедуется вместо АДД, если нужно скачать и разархивировать лучше использовать curt и tar

COPY \*.json ./ -только копирует

SHELL [“bin/sh”, ”-c”] – позволяет переопределить дефолтный басш

RUN echo Hello – запуск команды

RUN apt-get update && apt-get install git -y объединять слои

ENV Foo=1 – определили переменную, остается в собранном образе

ENV Foo1=$Foo

VOLUME [“/data”] – создает volume на хост машине

ENTRYPOINT [‘top’,’-c’] – команды запуска

ENTRYPOINT ["/home/noroot/entrypoint.sh"]

CMD [‘node’, “./cdsmkcmd/1.js”]

CMD [‘python’, “main.py ”]

### Различия между ENTRYPOINT и CMD

* Назначение: ENTRYPOINT используется для определения основной цели контейнера, в то время как CMD обеспечивает поведение по умолчанию, которое может быть переопределено.
* Переопределение: При запуске контейнера вы можете легко переопределить инструкцию CMD, предоставив дополнительные аргументы командной строки, в то время как инструкцию ENTRYPOINT нельзя переопределить, если она явно не указана с помощью флага -entrypoint.
* Взаимодействие: Если в Dockerfile указаны и ENTRYPOINT, и CMD, инструкция CMD предоставляет аргументы по умолчанию для инструкции ENTRYPOINT.

### 4. Комбинирование ENTRYPOINT и CMD

Используя ENTRYPOINT и CMD вместе в Dockerfile, вы можете создавать гибкие образы контейнеров, которые имеют четко определенную цель и настраиваемое поведение.

При совместном использовании инструкция ENTRYPOINT определяет команду по умолчанию, которая будет выполняться, а инструкция CMD предоставляет аргументы по умолчанию для этой команды.

Пример Dockerfile:

FROM ubuntu:latest

ENTRYPOINT ["/bin/ping"]

CMD ["localhost"]

В этом примере инструкция ENTRYPOINT указывает, что контейнер должен выполнить команду ping.

Инструкция CMD предоставляет аргумент по умолчанию “localhost”.

Когда вы запускаете контейнер без дополнительных аргументов, он выполнит команду ping localhost.

Однако если при запуске контейнера в качестве аргумента указать другое имя хоста или IP-адрес, он отменит инструкцию CMD и выполнит ping указанной цели.

STOPSIGNAL 9 сигнал при останвке контейнера

EXPOSE 80/TCP – Указывает на порт но не пробрасывает его

ENV *usual* "noroot"  
RUN useradd -m -d /home/${*usual*} ${*usual*}  
RUN chown -R ${*usual*} /home/${*usual*}  
RUN chmod -R 755 /home/${*usual*}  
RUN chsh -s /bin/bash ${*usual*}  
USER ${*usual*}  
WORKDIR /home/${*usual*}  
RUN echo ${*USER*} && whoami && pwd  
ADD ./11/. .  
RUN ls -la  
USER root  
WORKDIR root

задать нового пользователя



# MOUNT FILES AND DISKS

## VOLUMES

docker volume create NAME Создание volume

docker volume inspect NAME вывод информации о volumes

docker volume ls список volumes

docker volume prune удаление неиспользуемых volumes

docker volume rm NAME удаление одного volumes

docker run -d --name volume-3 -p 3003:3000 -v vol-1:/opt/app/data demo4:latest мы монтируем содерждимое хостового volume vol-1 в папку data образа :/opt/app/data и в дальнейшем пишем туда. При монтирование содержимое папки в образе замещается файлами из volume

полный синтаксис:

docker run -it --mount type=volume,src=v1,dst=/opt/app demo:latest

монтирование локальных папок

services:  
 frontend:  
 build: ./frontend  
 restart: always  
 ports:  
 - '3000:3000'  
 volumes:  
 - /app/node\_moduls - папка которую не хотим замещать и будем использовать  
 - ./frontend:/app - папка с файлами приложения с локального диска

mysql:  
 image: mysql  
 restart: always  
 environment:  
 MYSQL\_ROOT\_PASSWORD: password  
 MYSQL\_DATABASE: time\_db  
 volumes:  
 - mysql\_<data:/var/lib/mysql> - монитирование через волуме  
 adminer:  
 image: adminer  
 restart: always  
 ports:  
 - '8888:8080'  
volumes:  
 mysql\_data: - монитирование через волуме

### volume in Dockerfile

FROM node:14-alpine

RUN mkdir postman1

WORKDIR /postman1

VOLUME [“/post”] - в докерфайле объявляется каталог содержимое которого будет биндится в volume на хост машине. При создании image не создается, создается только при создании контейнера:

docker run -it --name post3 -v demo-postman:latest – создаст на хост машине каталог вида: d1f0b6b90d786575b21b7594367d5a726222538b80b4

Чтобы создать удобный и читаемый Volume нужно create volume Имя

Далее дополнительно указать как в примере:

docker run -it --name post3 -v Имя:/postman1 demo-postman:latest

docker run --rm --mount type=volume,src=my\_python,target=/bind python python /bind/python.py

# DOCKER COMPOSE

## Команды

docker-compose --version

docker-compose up – поднимает все файлы в docker-compose

docker-compose stop – остановить контейнеры из списка docker-compose

docker-compose start – запустить контейнеры из списка docker-compose

docker-compose down – остановить контейнеры из списка docker-compose и удалить все неиспользуемое содержимое

docker-compose – f имя файла up -d

docker-compose – f имя файла down

docker compose up -d --build

docker-compose logs – логи по контайнерам

простой файл

version: '3'  
  
services:  
 my-app:  
 build: ./app  
  
 mongo:  
 image: mongo

docker-compose images – список всех images используемых в docker-compose

docker-compose top – список запущенных процессов

docker-compose --profile start\_collection up -d - в docker-compose прописывается профиль и позволят запускать конкретные блоки

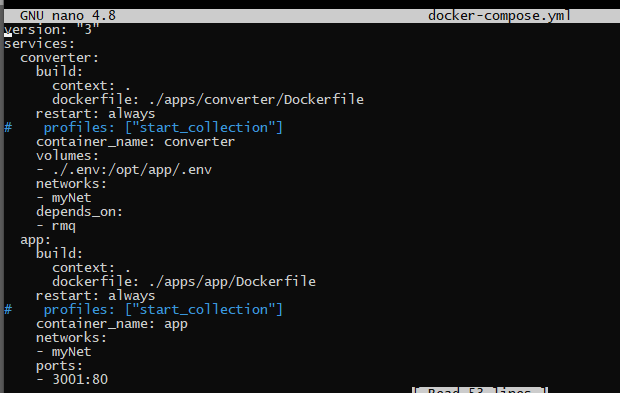
docker-compose run api – позволяет запустить один сервис

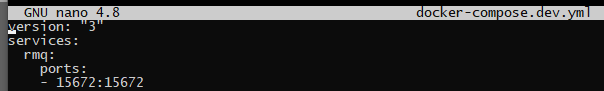
docker-compose --env-file .env.compose up -d - собирает переменные в docker-compose.yml из .env.compose Необходимо в docker-compose изменить переменную на container\_name: "${NAME}"

docker-compose --env-file .env.compose config - показывает как будет выглядеть итоговый файл с переменными из .env.compose

COMPOSE\_PROJECT\_NAME=11111 docker-compose --env-file .env.compose up -d - заменяет дефолтное имя проекты берущееся из названия папки на 11111

docker-compose -f docker-compose.yml -f docker-compose.dev.yml up -d - через 2-ой файл можно дополнить конфигурации сервиса, например портами



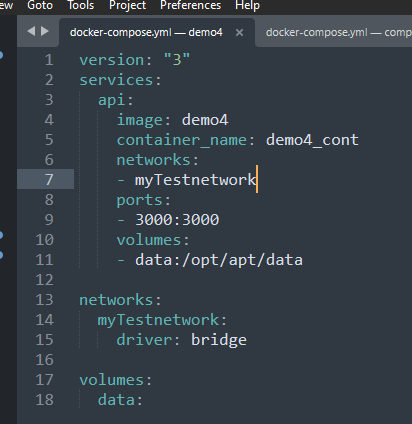


docker-compose -f docker-compose.yml -f docker-compose.dev.yml config – соберет и покажет такую конфигурацию на экране

docker-compose up – d 2-ой вариант наследования указывать в коде extends и дополнить еще один файл

|  |  |
| --- | --- |
|  |  |

## Примеры:



version: "3"

services:

api:

build: # собирает из докер файла

context: .

dockerfile: ./apps/api/Dockerfile

restart: always # перестартовывает сервис в случае отключения

container\_name: api

volumes:

- ./.env:/opt/app/.env

networks:

- myNet

depends\_on: # вначале запускается образ указанный в зависимости

- rmq

rmq:

image: rabbitmq:3-management

restart: always

networks:

- myNet

environment: # передает внутрь докера переменные

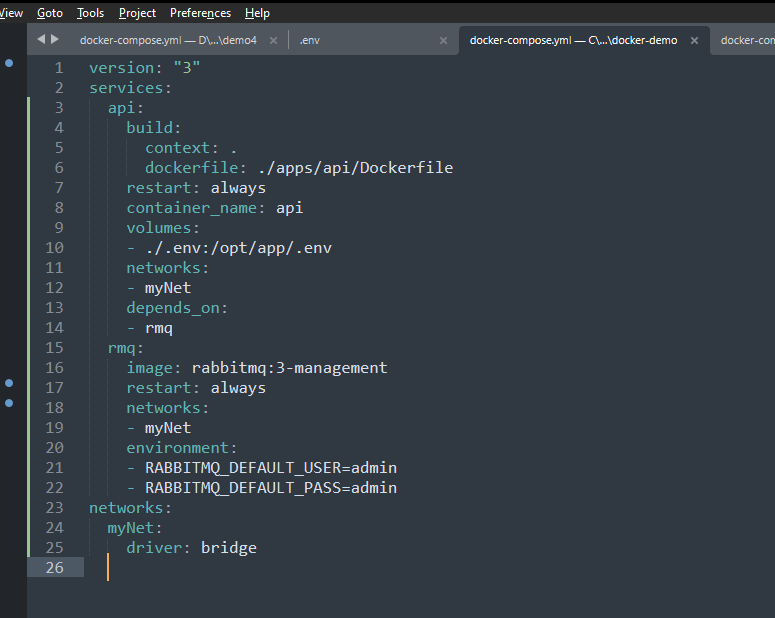
- RABBITMQ\_DEFAULT\_USER=admin

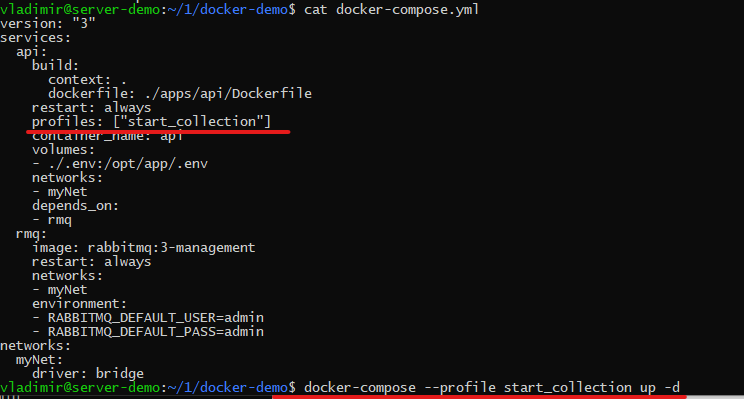
- RABBITMQ\_DEFAULT\_PASS=admin

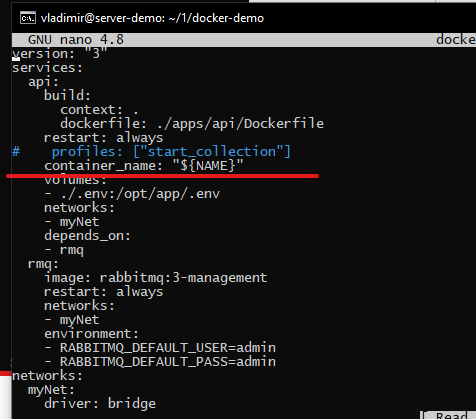
networks:

myNet:

driver: bridge







version: "3"

services:

api:

build:

context: .

dockerfile: ./apps/api/Dockerfile

restart: always

# profiles: ["start\_collection"]

container\_name: "${NAME}"

volumes:

- ./.env:/opt/app/.env

networks:

- myNet

depends\_on:

- rmq

rmq:

image: rabbitmq:3-management

restart: always

networks:

- myNet

environment:

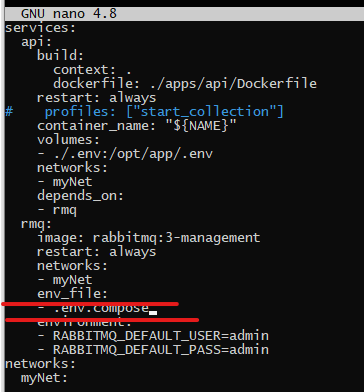
- RABBITMQ\_DEFAULT\_USER=admin

- RABBITMQ\_DEFAULT\_PASS=admin

networks:

myNet:

driver: bridge



GNU nano 4.8 docker-compose.yml Modified services:

api:

build:

context: .

dockerfile: ./apps/api/Dockerfile

restart: always

# profiles: ["start\_collection"]

container\_name: "${NAME}"

volumes:

- ./.env:/opt/app/.env

networks:

- myNet

depends\_on:

- rmq

rmq:

image: rabbitmq:3-management

restart: always

networks:

- myNet

env\_file:

- .env.compose

networks:

myNet:

mysql:  
 image: mysql  
 restart: always  
 environment:  
 MYSQL\_HOST: 'mysql'  
 MYSQL\_PORT: '3306'  
 MYSQL\_USER: 'root'  
 MYSQL\_PASSWOR: 'password'  
 MYSQL\_DB: 'time\_db'