

Семинар 4

Решение

FROM ubuntu:20.04

Используйте инструкцию **RUN**, чтобы выполнить команды внутри контейнера во время сборки образа.

Например, для обновления пакетов в Ubuntu и установки Python 3, вы можете написать:

```
RUN apt-get update && apt-get install -y python3
```

Используйте инструкцию **COPY** или **ADD**, чтобы скопировать файлы или директории из вашей локальной системы внутрь контейнера. Например, чтобы скопировать файл "app.py" из текущей директории внутрь контейнера в директорию "/app/", вы можете написать:

```
COPY app.py /app/
```

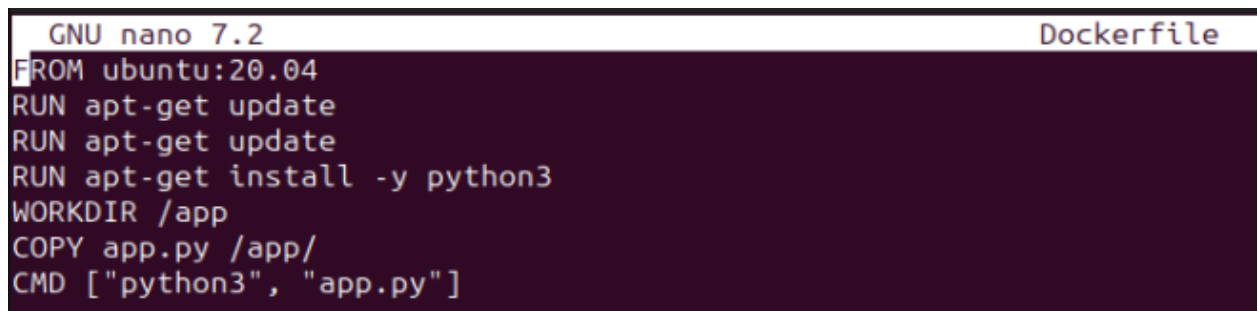
Установите рабочую директорию с помощью инструкции **WORKDIR**. Это указывает контейнеру, в какой директории выполнять команды по умолчанию. Например:

```
WORKDIR /app
```

Определите команду, которая будет выполняться при запуске контейнера, с помощью инструкции **CMD**. Эта команда будет выполнена, если при запуске контейнера не указана другая команда.

Например, чтобы запустить приложение Python 3, вы можете написать:

```
CMD ["python3", "app.py"]
```

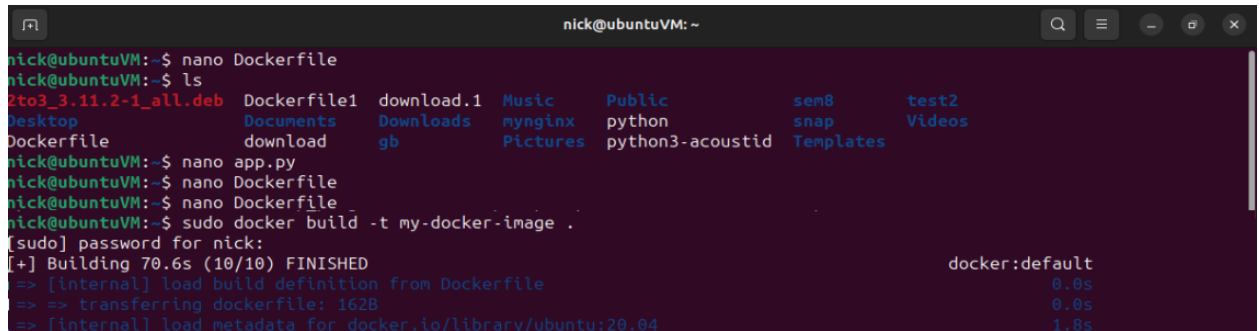
A screenshot of a terminal window with a dark background. The title bar at the top shows "GNU nano 7.2" on the left and "Dockerfile" on the right. The terminal content displays the following Dockerfile instructions: FROM ubuntu:20.04, RUN apt-get update, RUN apt-get update, RUN apt-get install -y python3, WORKDIR /app, COPY app.py /app/, and CMD ["python3", "app.py"].

```
GNU nano 7.2 Dockerfile
FROM ubuntu:20.04
RUN apt-get update
RUN apt-get update
RUN apt-get install -y python3
WORKDIR /app
COPY app.py /app/
CMD ["python3", "app.py"]
```

При необходимости, вы можете также использовать инструкцию **ENTRYPOINT** для определения точки входа, которая будет выполняться при запуске контейнера. Эта инструкция обычно используется, чтобы определить исполняемую программу, которая будет запущена, и может комбинироваться с **CMD**.

После того как вы создали Dockerfile, вы можете собрать Docker-образ с помощью команды `docker build`. Например:

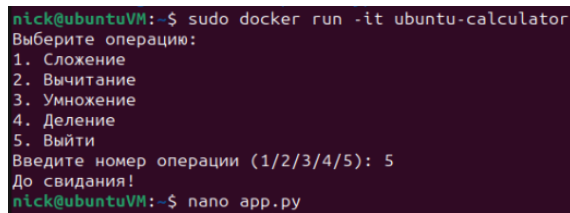
```
docker build -t my-docker-image .
```

A terminal window titled 'nick@ubuntuVM: ~' showing the process of building a Docker image. The user runs 'nano Dockerfile', then 'ls' which shows a directory listing including '2to3_3.11.2-1_all.deb', 'Dockerfile1', 'download.1', 'Music', 'Public', 'sen8', and 'test2'. The user then runs 'nano app.py', 'nano Dockerfile', and finally 'sudo docker build -t my-docker-image .'. The build process shows progress: '[+] Building 70.6s (10/10) FINISHED', followed by '[=> [internal] load build definition from Dockerfile', '[=> => transferring dockerfile: 162B', and '[=> [internal] load metadata for docker.io/library/ubuntu:20.04'. The output shows the image is built successfully.

```
nick@ubuntuVM:~$ nano Dockerfile
nick@ubuntuVM:~$ ls
2to3_3.11.2-1_all.deb  Dockerfile1  download.1  Music      Public      sen8      test2
Desktop               Documents    Downloads   mynginx    python      snap      Videos
Dockerfile            download     gb          Pictures   python3-acoustid  Templates
nick@ubuntuVM:~$ nano app.py
nick@ubuntuVM:~$ nano Dockerfile
nick@ubuntuVM:~$ sudo docker build -t my-docker-image .
[sudo] password for nick:
[+] Building 70.6s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                  0.0s
=> => transferring dockerfile: 162B                                0.0s
=> [internal] load metadata for docker.io/library/ubuntu:20.04      1.0s
```

После сборки образа, вы можете создать контейнер на его основе с помощью команды `docker run`. Например:

```
docker run -it my-docker-image
```

A terminal window showing the execution of a Docker container. The user runs 'sudo docker run -it ubuntu-calculator'. The container starts and displays a menu in Russian: 'Выберите операцию: 1. Сложение 2. Вычитание 3. Умножение 4. Деление 5. Выйти'. The user enters '5' to exit. The prompt then returns to the host machine.

```
nick@ubuntuVM:~$ sudo docker run -it ubuntu-calculator
Выберите операцию:
1. Сложение
2. Вычитание
3. Умножение
4. Деление
5. Выйти
Введите номер операции (1/2/3/4/5): 5
До свидания!
nick@ubuntuVM:~$ nano app.py
```

Это запустит контейнер, и он выполнит команду, указанную в инструкции CMD или ENTRYPOINT.

```
nick@ubuntuVM: ~  
GNU nano 7.2 app.py  
def add(x, y):  
    return x + y  
  
def subtract(x, y):  
    return x - y  
  
def multiply(x, y):  
    return x * y  
  
def divide(x, y):  
    if y == 0:  
        return "Ошибка: деление на ноль"  
    return x / y  
  
def menu():  
    print("Выберите операцию:")  
    print("1. Сложение")  
    print("2. Вычитание")  
    print("3. Умножение")  
    print("4. Деление")  
    print("5. Выйти")  
  
while True:  
    menu()  
    choice = input("Введите номер операции (1/2/3/4/5): ")  
  
    if choice in ('1', '2', '3', '4'):  
        num1 = float(input("Введите первое число: "))  
        num2 = float(input("Введите второе число: "))
```

```
def add(x, y):
```

```
    return x + y
```

```
def subtract(x, y):
```

```
    return x - y
```

```
def multiply(x, y):
```

```
    return x * y
```

```
def divide(x, y):
```

```
    if y == 0:
```

```
        return "Ошибка: деление на ноль"
```

```
    return x / y
```

```
def menu():
```

```
    print("Выберите операцию:")
```

```
    print("1. Сложение")
```

```
    print("2. Вычитание")
```

```
    print("3. Умножение")
```

```
    print("4. Деление")
```

```
    print("5. Выйти")
```

```
while True:
```

```
    menu()
```

```
    choice = input("Введите номер операции (1/2/3/4/5): ")
```

```
    if choice in ('1', '2', '3', '4'):
```

```
num1 = float(input("Введите первое число: "))

num2 = float(input("Введите второе число: "))


if choice == '1':

    print("Результат:", add(num1, num2))

elif choice == '2':

    print("Результат:", subtract(num1, num2))

elif choice == '3':

    print("Результат:", multiply(num1, num2))

elif choice == '4':

    print("Результат:", divide(num1, num2))

elif choice == '5':

    print("До свидания!")

    break

else:

    print("Неверный ввод. Попробуйте снова.")
```