

Контейнеризация приложений

Построение стенда

Схема виртуального лабораторного стенда



Рисунок 1. Схема стенда

1. Создать виртуальные машины для работы

Название виртуальной машины	Источник	Тип инстанса	Сети для внешнего подключения
web-server	Образ-Ubuntu-server20.04	medium	external-net

Так же нужно проверить развернутую инфраструктуру на соответствие схеме на рисунке 1.

1. Установка Docker

1. Обновляем информацию о пакетах в репозиториях и обновляем установленные пакеты:

```
sudo apt update
sudo apt full-upgrade -y
```

2. Добавляем репозитории официальные репозитории Docker:

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
echo \  
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/ubuntu \  
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update
```

3. Установим пакеты Docker

```
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y
```

4. Проверка работы Docker

```
sudo docker run hello-world
```

```
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
c1ec31eb5944: Pull complete  
Digest: sha256:4bd78111b6914a99dbc560e6a20eab57ff6655aea4a80c50b0c5491968cbc2  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/
```

5. Добавляем пользователя в группу `docker`

```
sudo groupadd docker  
sudo usermod -aG docker $USER
```

После того как текущий пользователь был добавлен в группу Docker, мы можем проверить функциональность Docker без необходимости использования привилегий суперпользователя.

```
docker run hello-world
```

2. Написание Dockerfile

1. Клонировем исходный проект с использованием системы контроля версий Git.

```
sudo apt install git -y
git clone https://gitlab.resds.ru/itt/sample-project.git
cd sample-project
```

в данном проекте собраны все компоненты из прошлой работы

2. Пишем Dockerfile для frontend части проекта Для выполнения этой задачи предлагаем создать файл с названием Dockerfile в директории notes-app.

```
FROM node:18-slim
WORKDIR /usr/local/app
COPY package*.json ./
RUN npm i && npm cache clean --force
COPY . .
RUN npm run build -- --prod
CMD ["npm","serve","-s","build"]
```

3. Запускаем сборку контейнера и проверяем его работоспособность

```
docker build -t notes-app:latest .
docker run -p 3000:3000 notes-app:latest
```

И проверяем работоспособность приложения открыв в браузере страницу

4. Прodelываем аналогичные действия для notes-backend

```
FROM node:18-slim
WORKDIR /usr/local/app
COPY package*.json ./
RUN npm i && npm cache clean --force
COPY . .
CMD ["npm","start"]
```

Собираем контейнер и проверяем

```
docker build -t notes-back:latest .
docker run -p 5000:5000 notes-back:latest
```

Для проверки можно использовать запрос к API

```
curl --location --request POST 'http://127.0.0.1:5000/api/notes' \  
--header 'Content-Type: application/json' \  
--data-raw '{  
  "index": "1",  
  "note": "test message"  
}'
```

Как можно заметить, приложения, запущенные внутри виртуальной машины, становятся доступными извне.

5. Создаем общий файл Docker-compose для запуска приложения.

```
services:  
  notes-app:  
    build: ./notes-app/  
    restart: always  
  notes-back:  
    build: ./notes-backend/  
    restart: always  
  nginx:  
    image: nginx:1.25.3-alpine-slim  
  volumes:  
    - ./nginx/default.conf:/etc/nginx/conf.d/default.conf  
  ports:  
    - "80:80"  
  restart: always  
  depends_on:  
    - notes-app  
    - notes-back
```

С учетом данного способа запуска контейнеров следует обратить внимание на то, что только порт 80 становится общедоступным.

Версия #6

Тарабанов Илья Федорович создал 6 февраля 2024 14:35:51

Тарабанов Илья Федорович обновил 16 мая 2024 19:15:31