**№1.** **Docker, docker swarm, docker compose, haproxy.**

**Цель работы:** получить навыки работы с docker, docker swarm, socker compose, haproxy.

1. Создаём новое приложение Node.js – в файле index.js прописываем следующее (рисунок 1):

* **nano index.js**

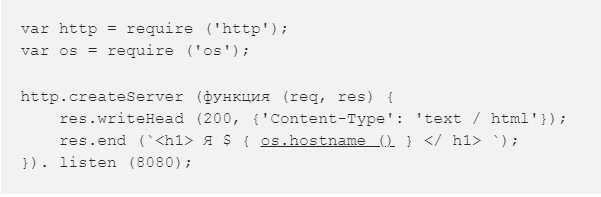


Рисунок 1. Текст index.js

1. Docker - изуем приложение, для этого создаём файл Dockerfile со следующим кодом (рисунок 2):



Рисунок 2. Текст Dockerfile

1. Создаем docker-образ нашего приложения, вводим в консоли следующую команду (рисунок 3):

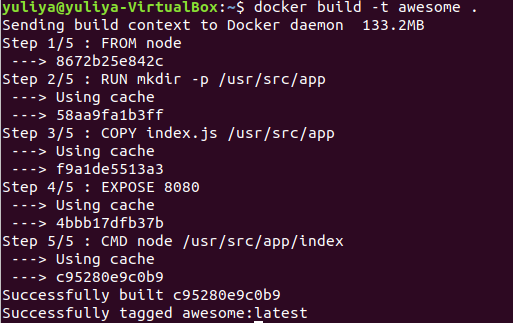
* **docker build -t awesome . **

Рисунок 3. Вызов команды docker build –t awesome .

1. Для нашего HTTP-сервера мы будем использовать HAProxy , это означает, что нам нужно создать контейнер с HAProxy, который будет прослушивать порт 80 и загружать баланс для разных контейнеров Node.js на порт 8080. Чтобы создать наши контейнеры (Node.js приложения и HAProxy), мы будем использовать Docker Compose, напишем наш docker-compose.yml файл (рисунок 4):

* **nano docker-compose.yml**

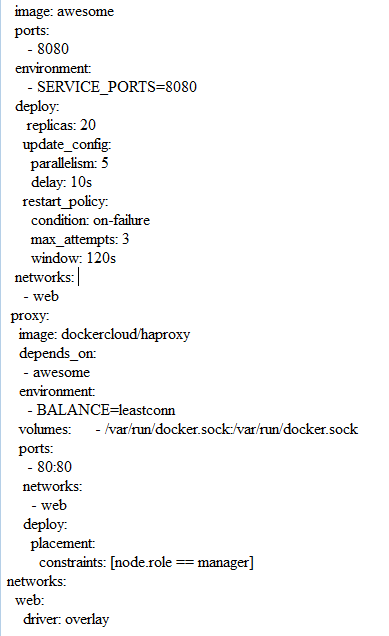


Рисунок 4. Текст docker-compose.yml

Чтоб включить docker swarm пишем в консоли:

* **docker swarm init**

1. Cоздаём stack с названием prod из файла **docker-compose.yml:**

* **docker stack deploy --compose-file=docker-compose.yml prod**

1. Следующей командой увидим запущенные сервисы (рисунок 5)

* **docker service ls**

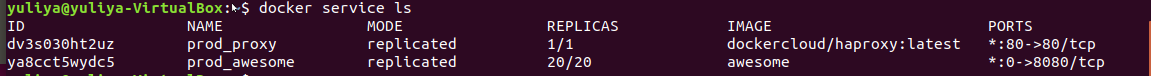


Рисунок 5. Запущенные сервисы - docker service

Когда мы откроем браузер и впишем <http://localhost> , то у нас выйдет такой результат – контейнер со своим идентификатором (рисунок 6).

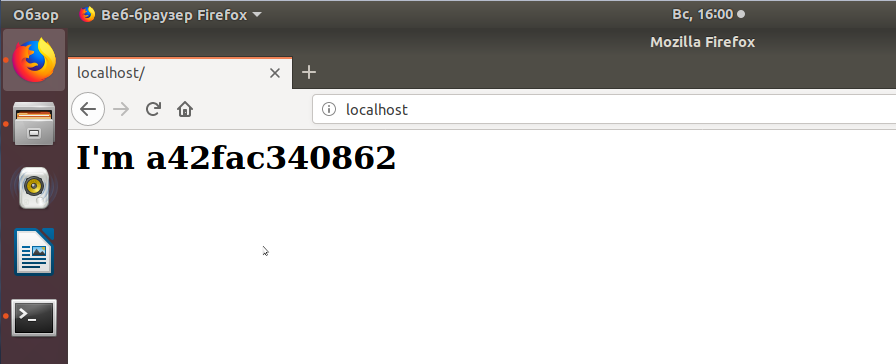


Рисунок 6. Страница в браузере localhost с идентификатором контейнера

Чтобы наверняка убедиться в существовании контейнера с идентификатором a42fac340862, в консоли сделаем проверку (рисунок 7) и, да, такой контейнер существует.

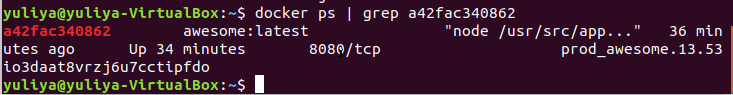


Рисунок 7. Проверка существования контейнера по идентификатору