МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики  
Кафедра суперкомпьютеров и общей информатики

**Отчет по лабораторной работе №3**

Дисциплина: «Development Operations»

Тема: **«Docker»**

Выполнила: Кривоконева Е.А.

Группа: 6133-010402D

Самара 2021

**ЗАДАНИЕ**

1

1.1. Install Docker.

1.2. Create Dockerfile > Image > Container for Node.js web-app from example (<https://docs.docker.com/get-started/nodejs/build-images/>) (<https://www.youtube.com/watch?v=BC19Gl2u4wc>)

1.3. Push Docker-Image to Docker Hub.

2

2.1. (simple-script-1) Create and run Docker container with python-application

2.2. (mount-directory-2) Create and run Docker container with directory mounting 2.3. (ports-3) Create and run Docker container with port forwarding

2.4. (compose-4) Create and run Docker container with docker-compose

2.5. (proxy-system-5) Create and run Docker containers with load distribution

2.6. (docker-swarm-6) Create and run Docker-Swarm configuration

**ХОД РАБОТЫ**

Перед установкой Docker Engine на компьютер произведем настройку Docker repository используя следующие команды (<https://docs.docker.com/engine/install/debian/#installation-methods>):

$ sudo apt-get update

$ sudo apt-get install \

ca-certificates \

curl \

gnupg \

lsb-release

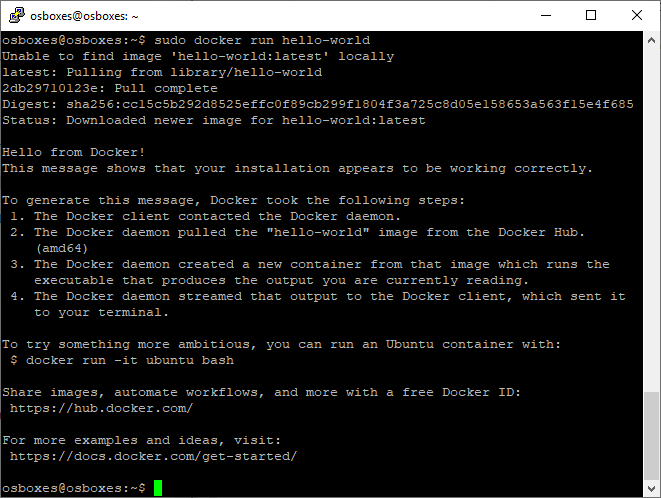
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

Чтобы убедиться в том, что Docker Engine установлен правильно, запустим образ hello-world:

$ sudo docker run hello-world



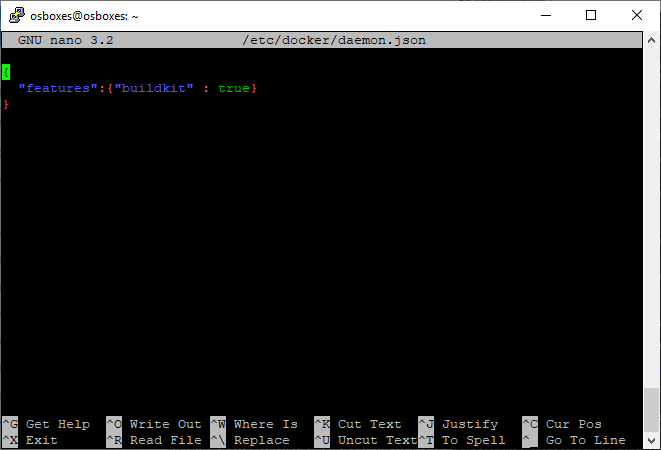
Далее выполняем действия по созданию образа с сайта <https://docs.docker.com/language/nodejs/build-images/>.

Чтобы включить docker BuildKit по умолчанию, создадим файл daemon.json, а затем добавим в него следующий код:

{

"features":{"buildkit" : true}

}



Перезапустим Docker командой:

$ sudo /etc/init.d/docker restart

и установим Node.js по инструкции с <https://github.com/nodesource/distributions/blob/master/README.md#manual-installation>:

$ KEYRING=/usr/share/keyrings/nodesource.gpg

$ curl -fsSL https://deb.nodesource.com/gpgkey/nodesource.gpg.key | gpg --dearmor | sudo tee "$KEYRING" >/dev/null

$ gpg --no-default-keyring --keyring "$KEYRING" --list-keys

$ VERSION=node\_17.x

$ KEYRING=/usr/share/keyrings/nodesource.gpg

$ DISTRO="$(lsb\_release -s -c)"

$ echo "deb [signed-by=$KEYRING] https://deb.nodesource.com/$VERSION $DISTRO main" | sudo tee /etc/apt/sources.list.d/nodesource.list

$ echo "deb-src [signed-by=$KEYRING] https://deb.nodesource.com/$VERSION $DISTRO main" | sudo tee -a /etc/apt/sources.list.d/nodesource.list

Обновим списки пакетов и установим Node.js:

$ sudo apt-get update

$ sudo apt-get install nodejs

Теперь разработаем простое REST API приложение, используя Node.js. Для этого создадим каталог с именем node-docker с помощью команды

$ mkdir node-docker

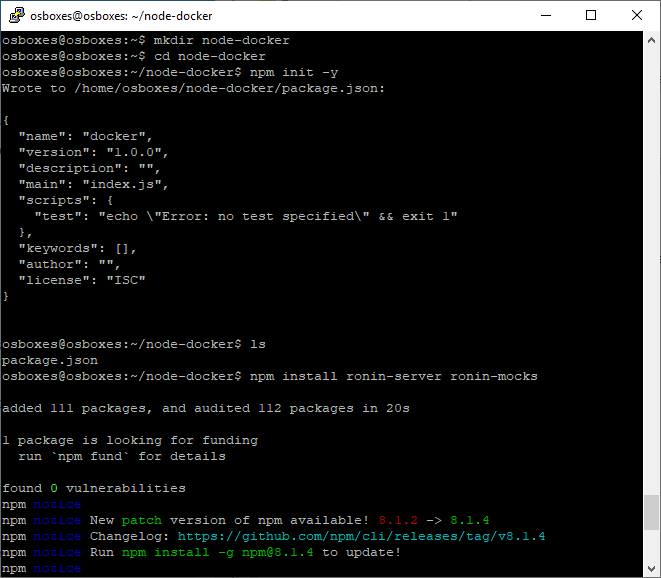
и выполним следующие действия:

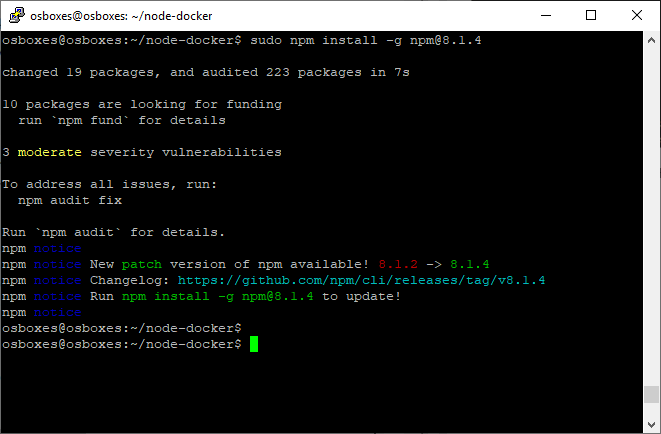
$ cd node-docker

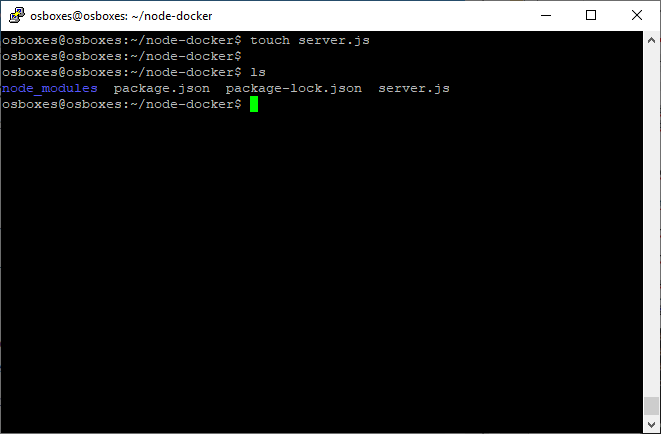
$ npm init -y

$ npm install ronin-server ronin-mocks

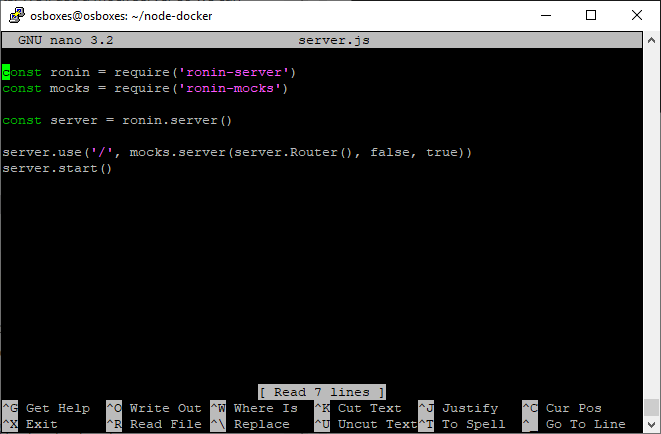
$ touch server.js







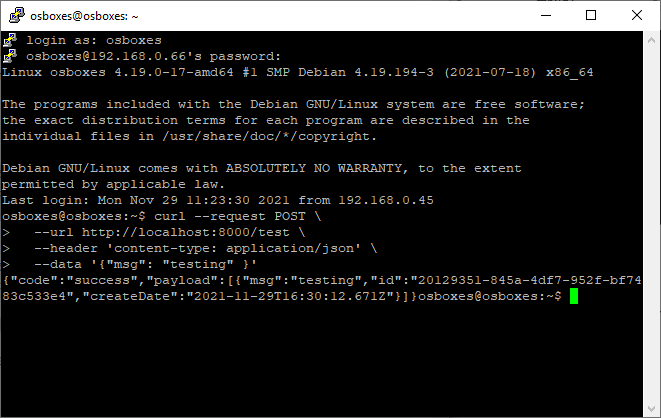
В файл server.js и добавим следующий код:

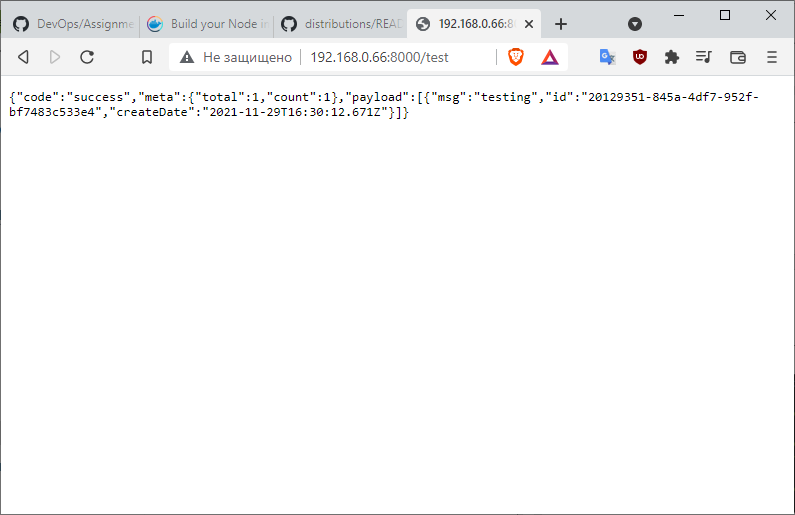


Запустим сервер командой:

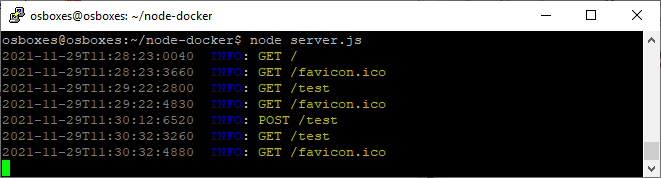
$ node server.js

Откроем новый терминал и отправим JSON POST запрос в API, а затем, произведя GET запрос, проверим, что данные были сохранены.





Если вернуться к терминалу, на котором работает наш сервер, можно увидеть следующие записи в журналах:



Сформируем файл dockerfile со следующим содержанием:

# syntax=docker/dockerfile:1

FROM node:12.18.1

ENV NODE\_ENV=production

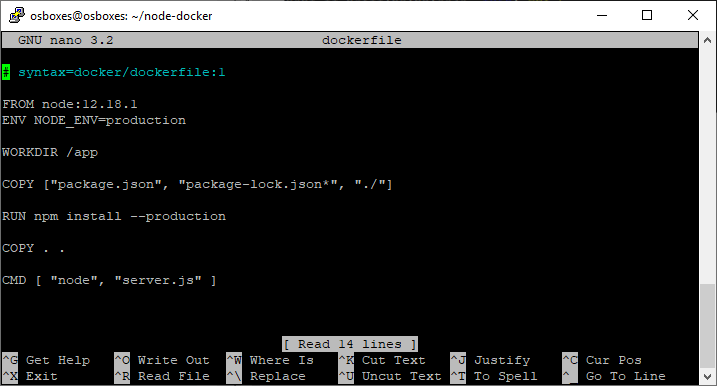
WORKDIR /app

COPY ["package.json", "package-lock.json\*", "./"]

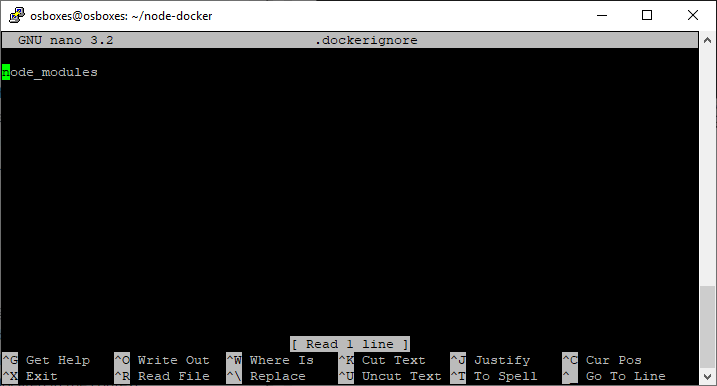
RUN npm install --production

COPY . .

CMD [ "node", "server.js" ]

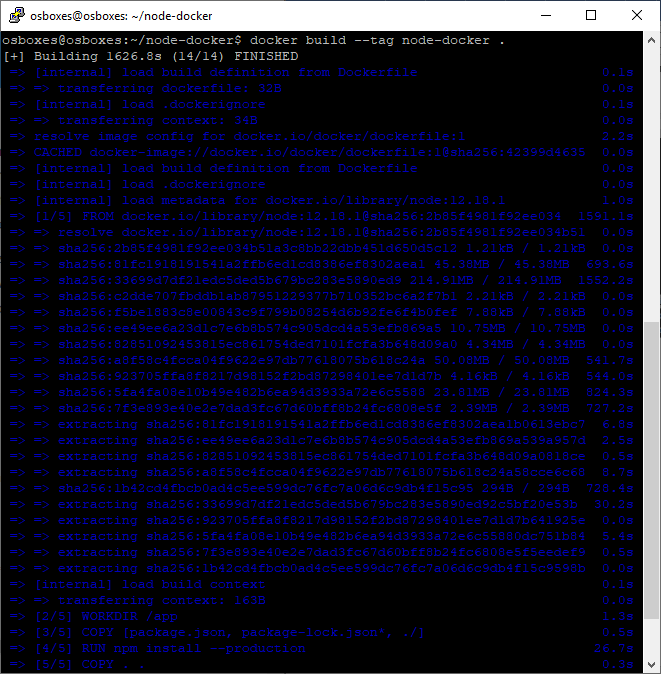


Чтобы повысить производительность, исключим файлы и каталоги, добавив файл .dockerignore:

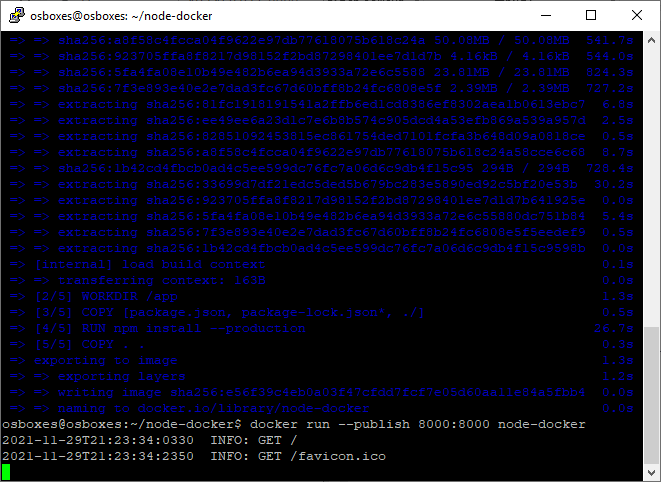


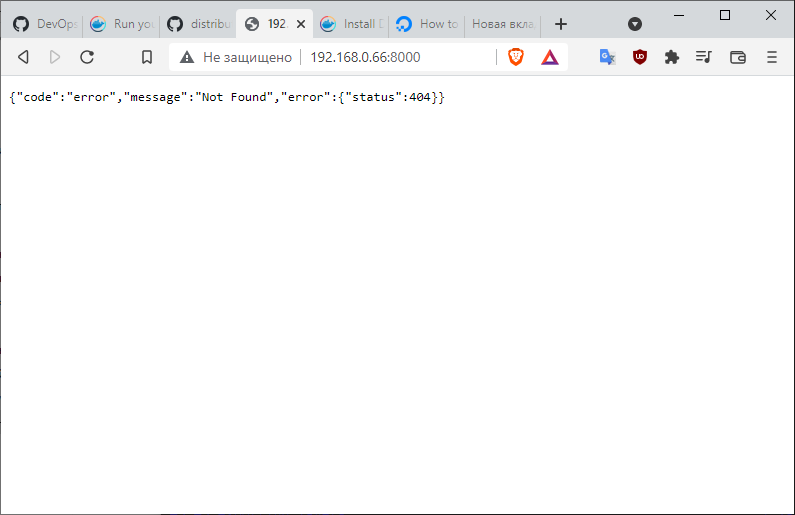
Создадим образ Docker, используя команду

$ docker build --tag node-docker .

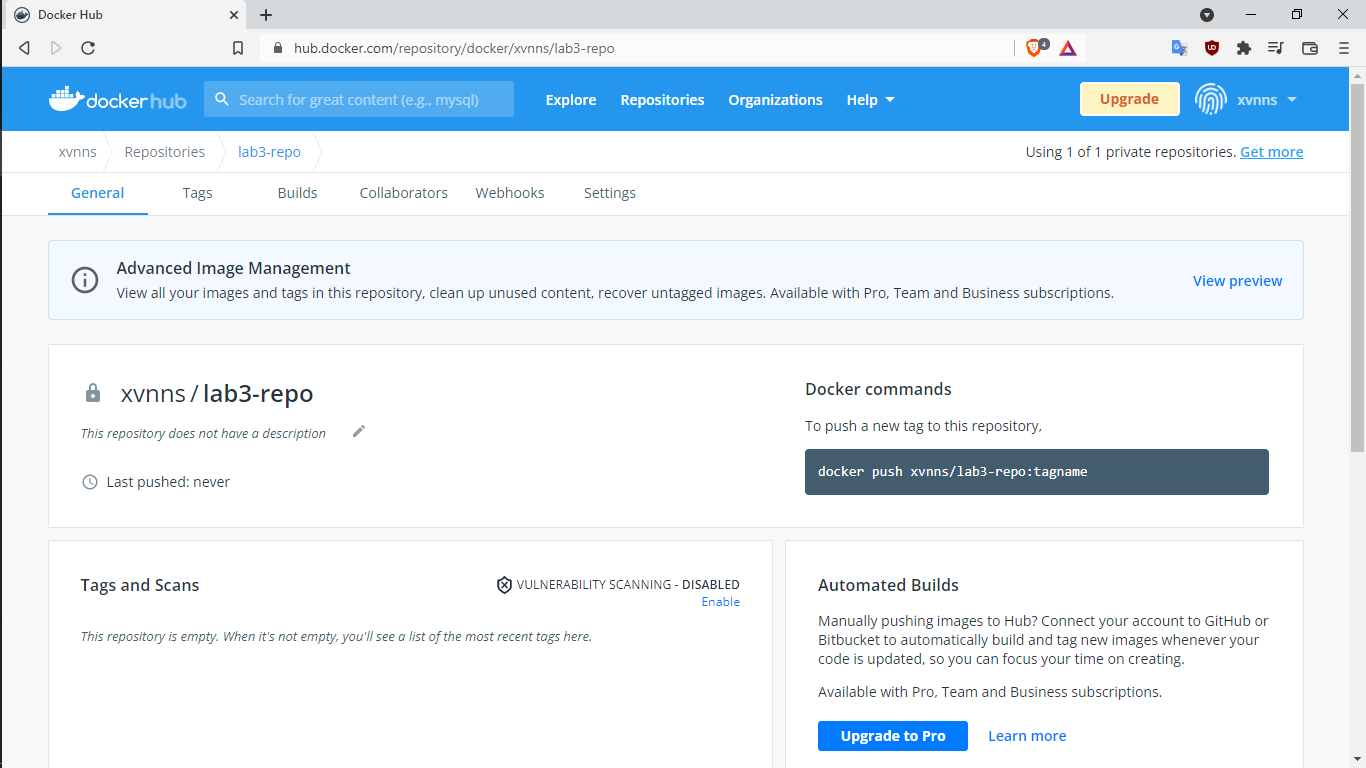


Запустим Docker контейнер на порту 8000:





Для того, чтобы выполнить загрузку образа в Docker Hub, создадим новый репозиторий:

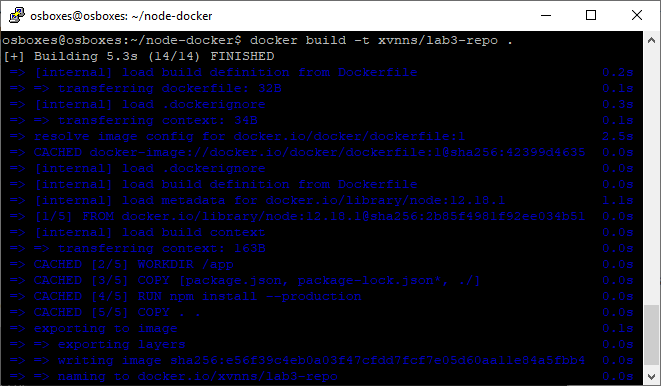


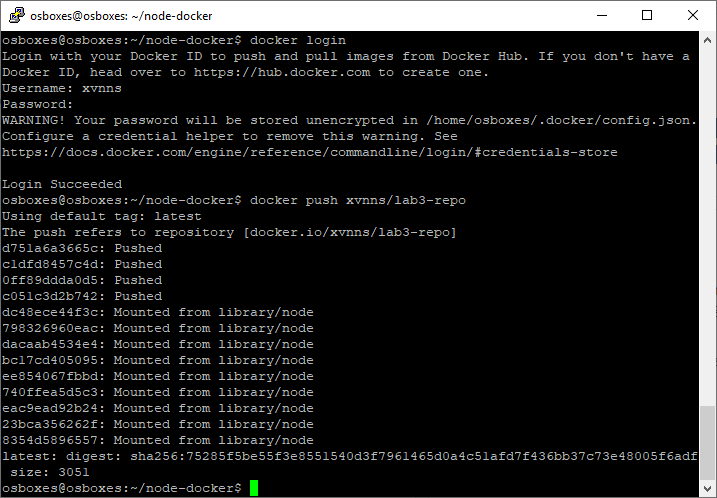
Выполним следующие команды в консоли:

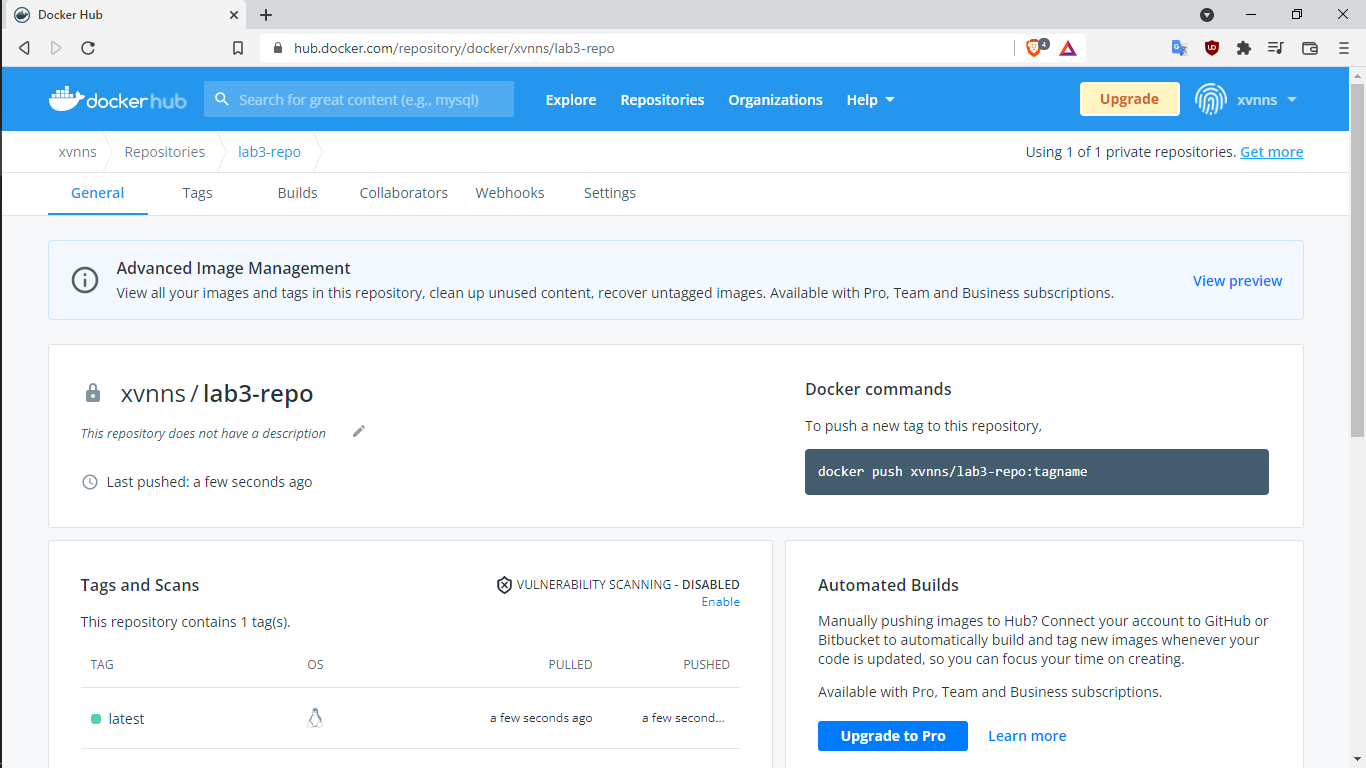
$ docker build -t xvnns/lab3-repo .

$ docker login

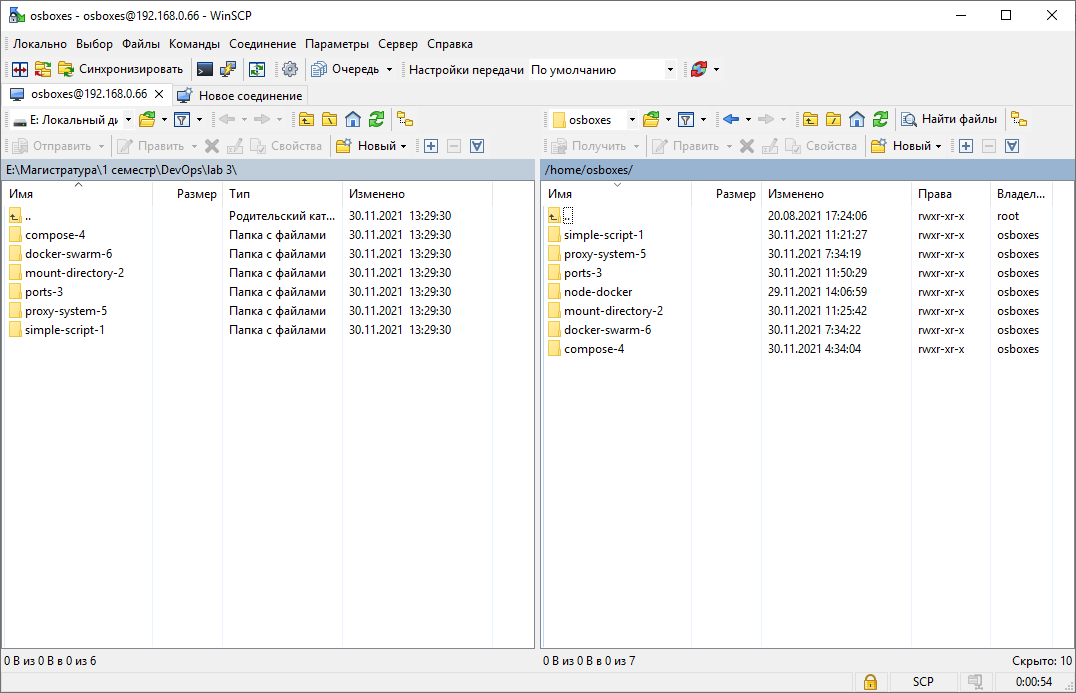
$ docker push xvnns/lab3-repo







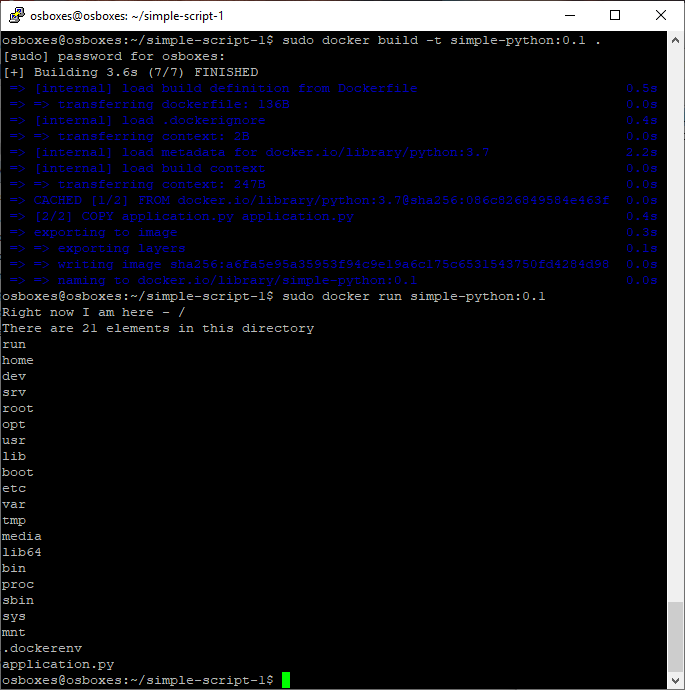
Для перехода ко второму этапу выполнения лабораторной работы, необходимо скачать файлы из https://github.com/j-avdeev/DevOps/tree/master/Assignment3%20Docker и переместить их в /home/osboxes/:



**simple-scrript-1**

$ sudo docker build -t simple-python:0.1 .

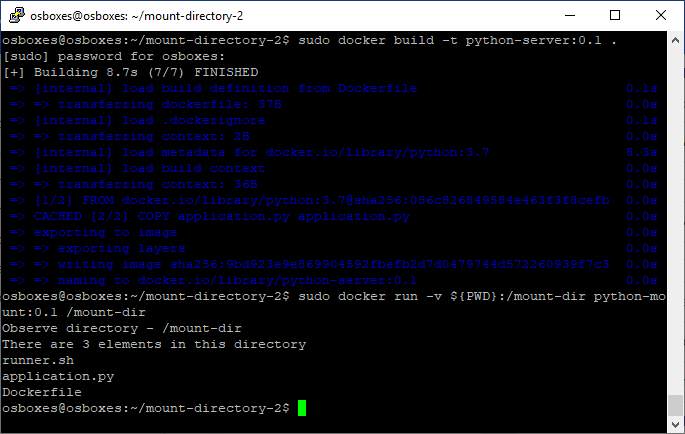
$ sudo docker run simple-python:0.1



**mount-directory-2**

$ sudo docker build -t python-mount:0.1 .

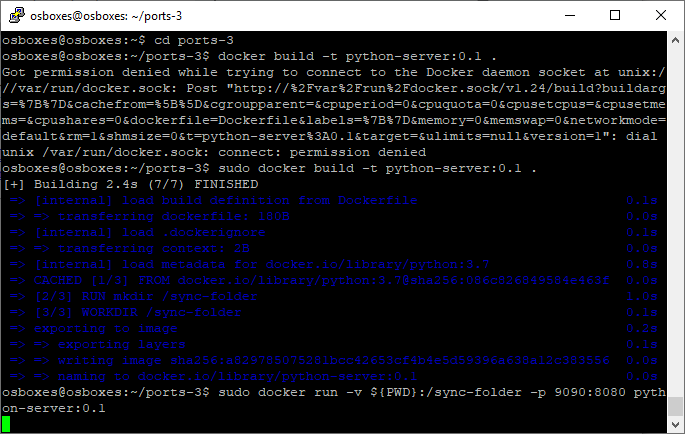
$ sudo docker run -v ${PWD}:/mount-dir python-mount:0.1 /mount-dir

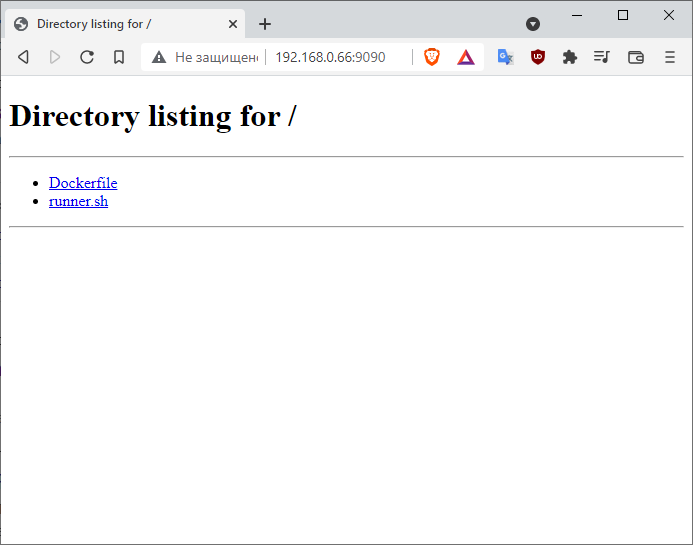


**ports-3**

$ docker build -t python-server:0.1 .

$ docker run -v ${PWD}:/sync-folder -p 9090:8080 python-server:0.1



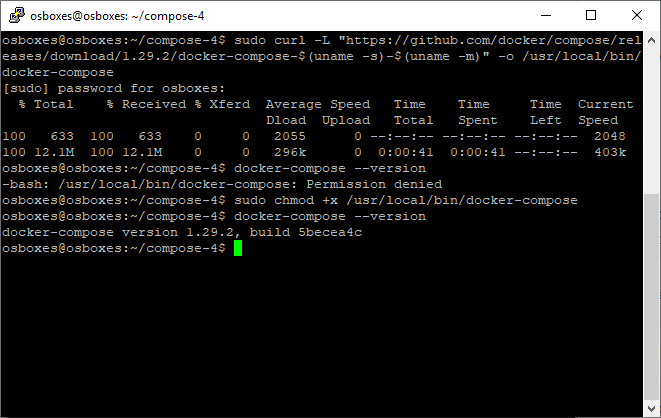


**compose-4**

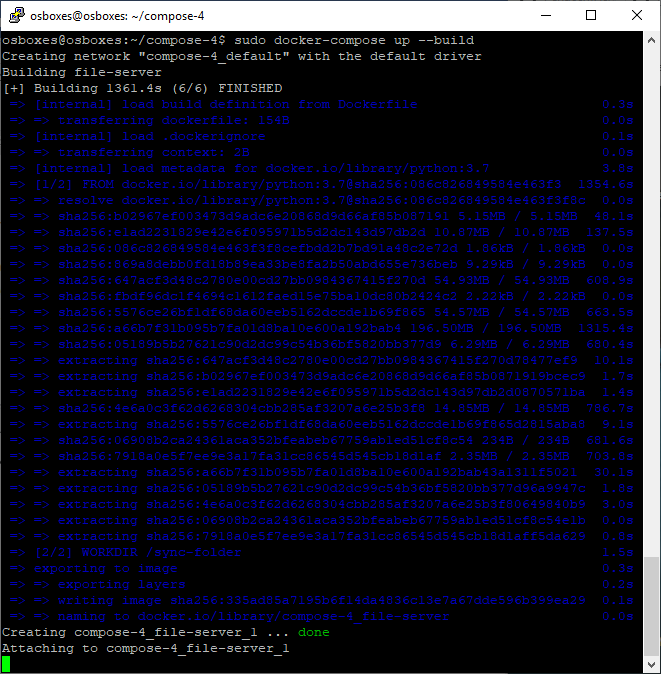
Устанавливаем текущую версию Docker Compose:

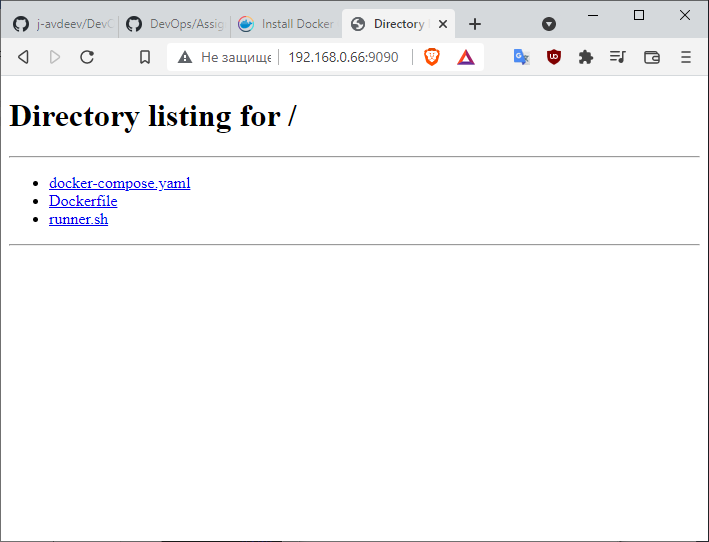
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

$ sudo chmod +x /usr/local/bin/docker-compose



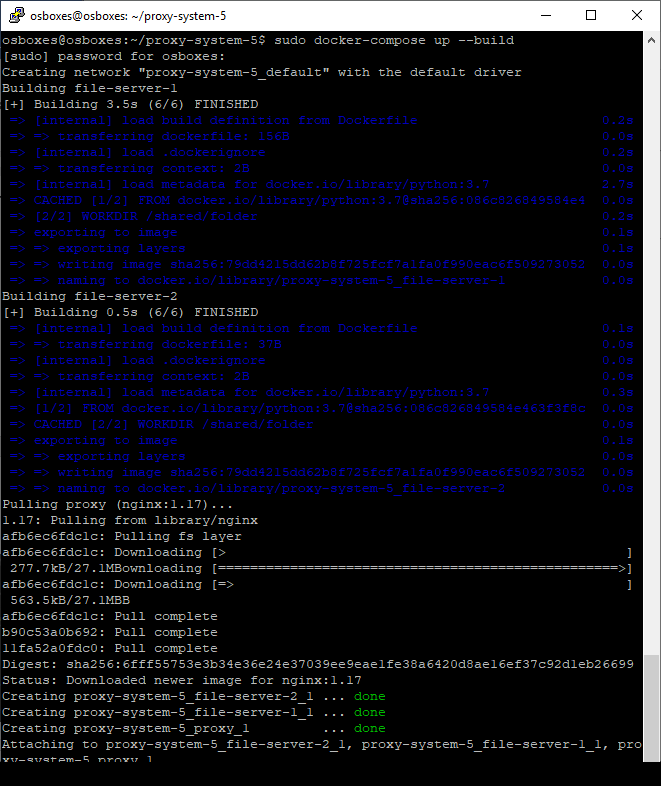
$ sudo docker-compose up –build

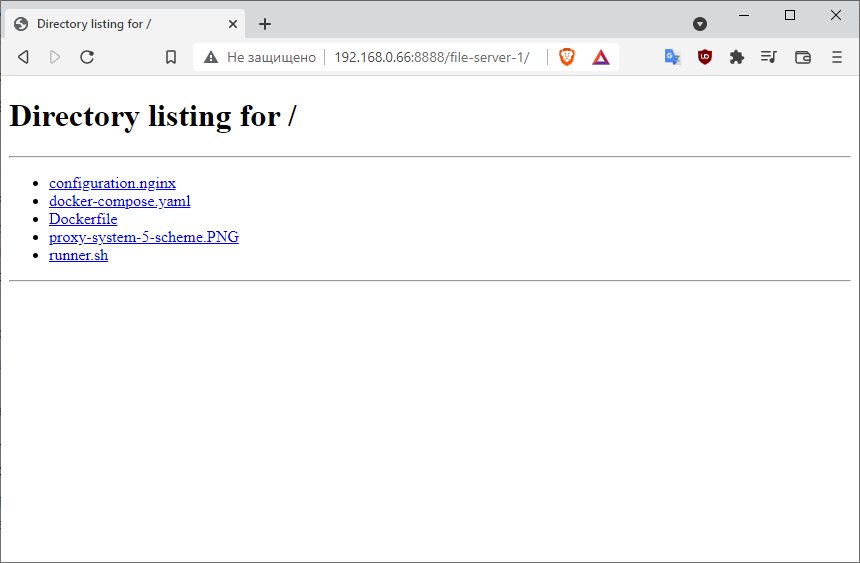


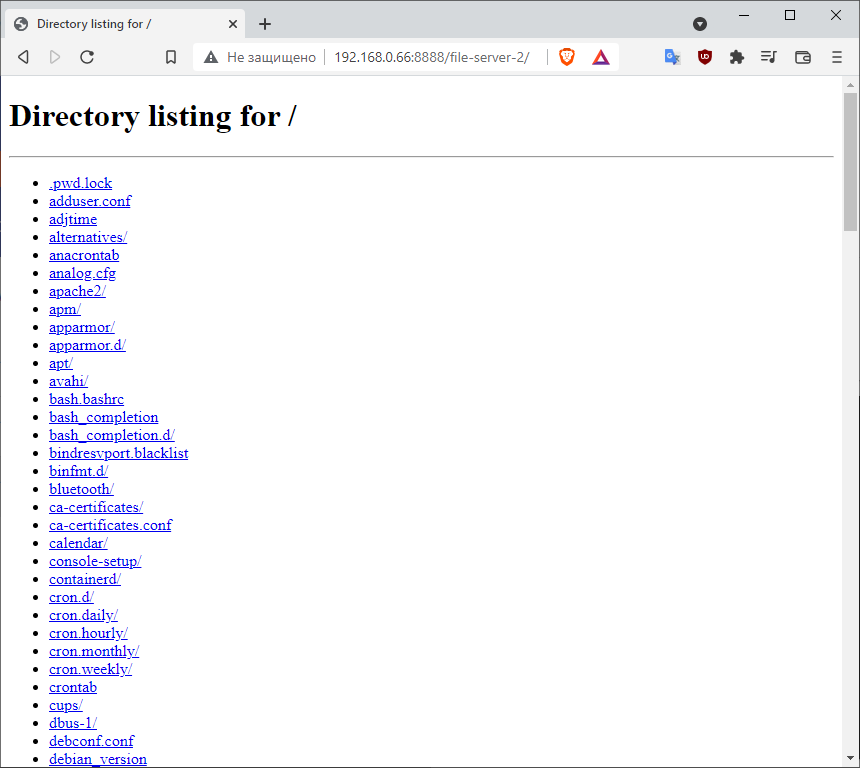


**proxy-system-5**

$ sudo docker-compose up --build

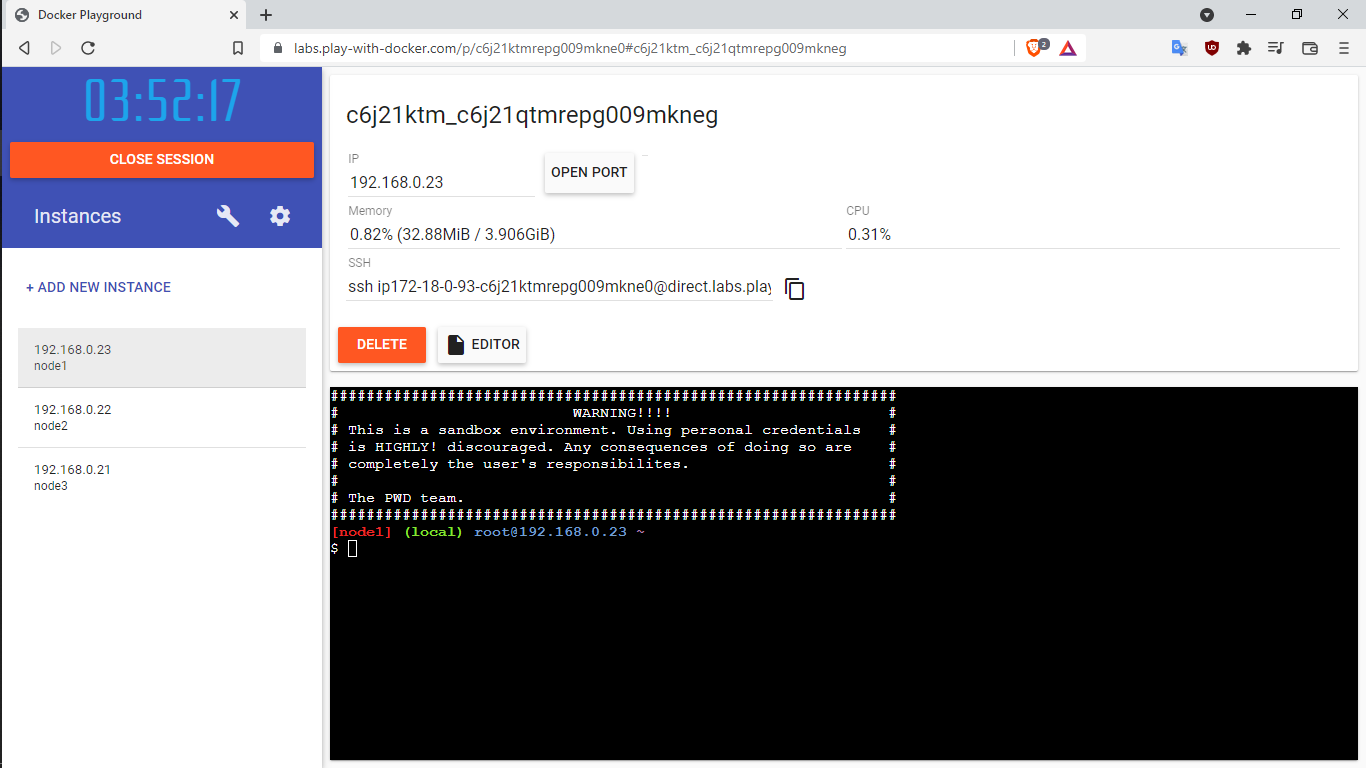






**Docker Swarm**

Переходим на <https://labs.play-with-docker.com/> и создаем там три виртуальные машины:



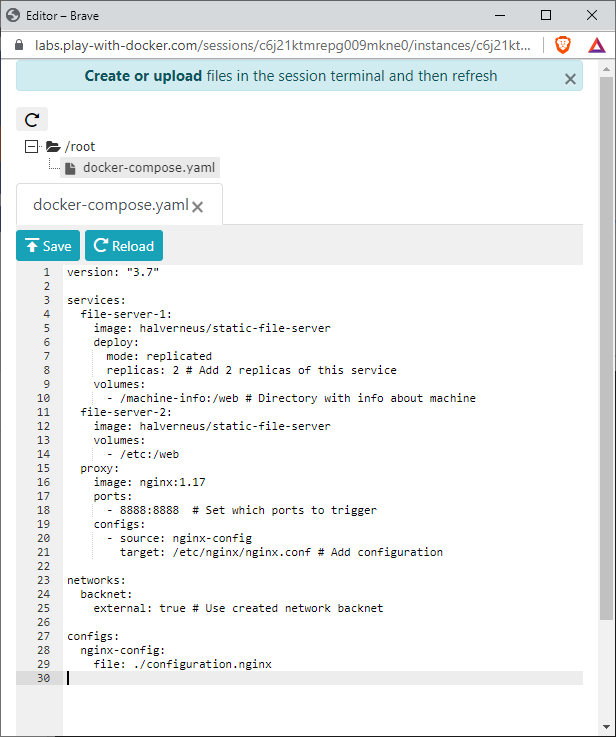
С помощью команд в node1

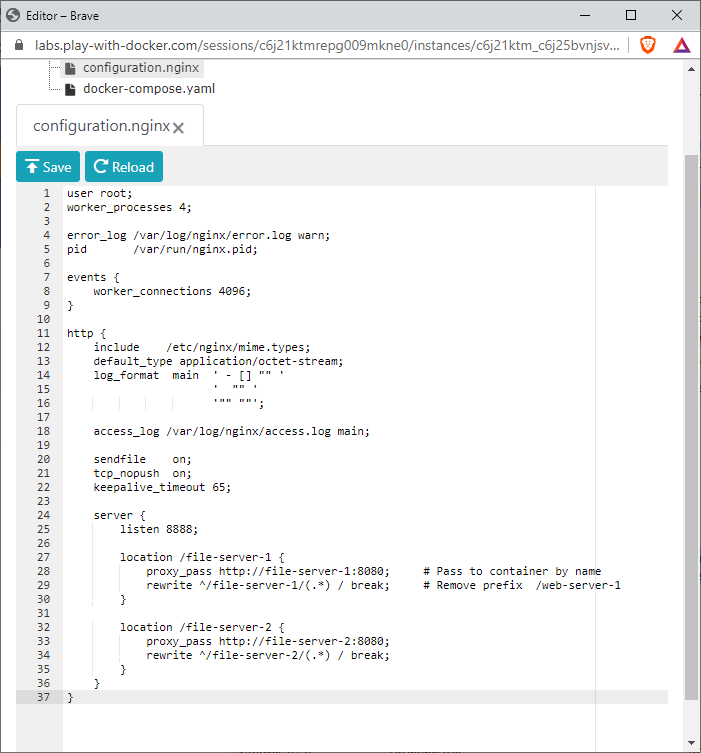
$ touch docker-compose.yaml

$ touch configuration.nginx

$ touch run-vis.sh

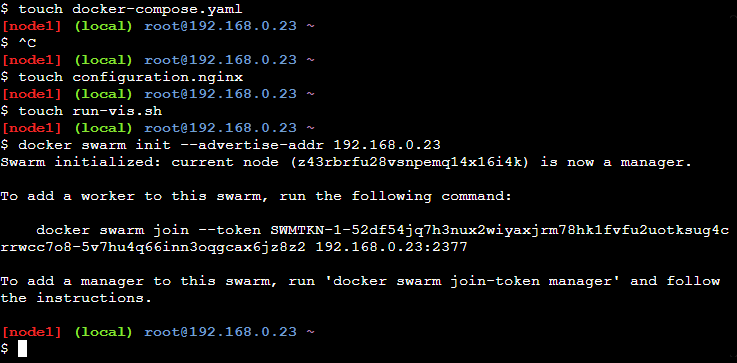
создаем следующие файлы:





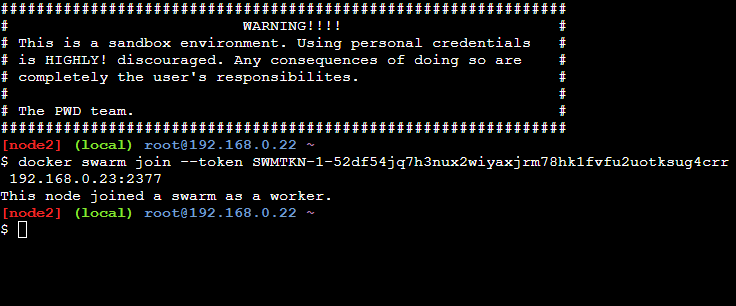
Далее в node1:

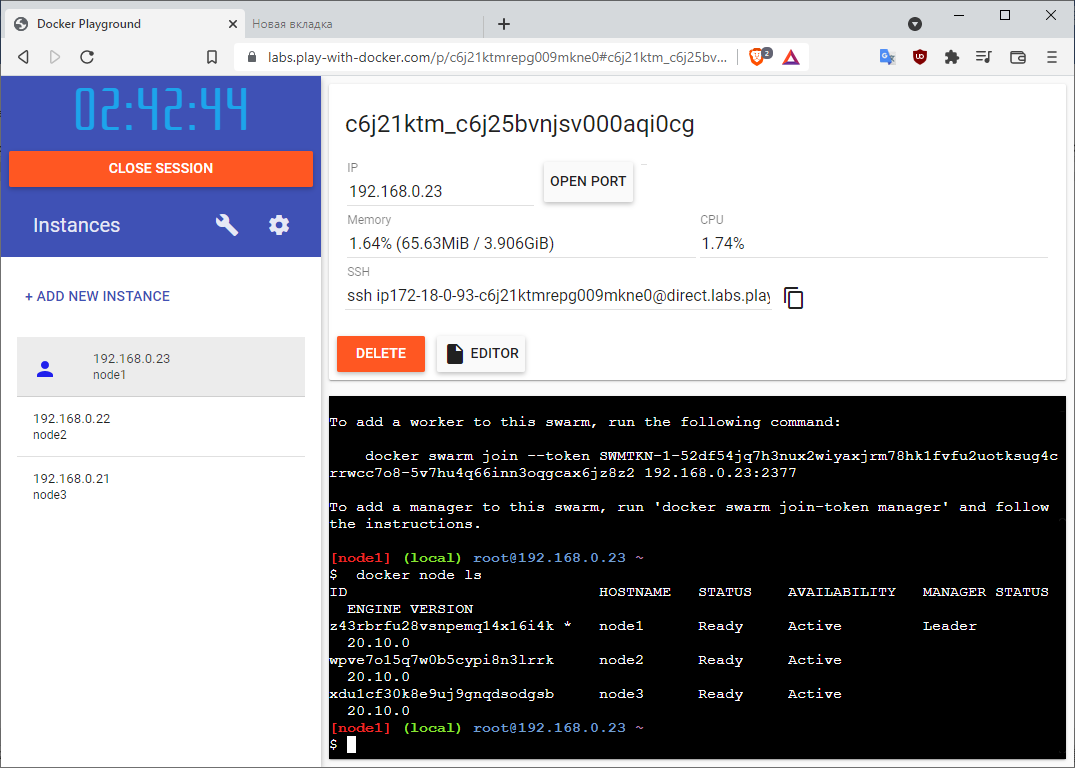
$ docker swarm init --advertise-addr 192.168.0.23



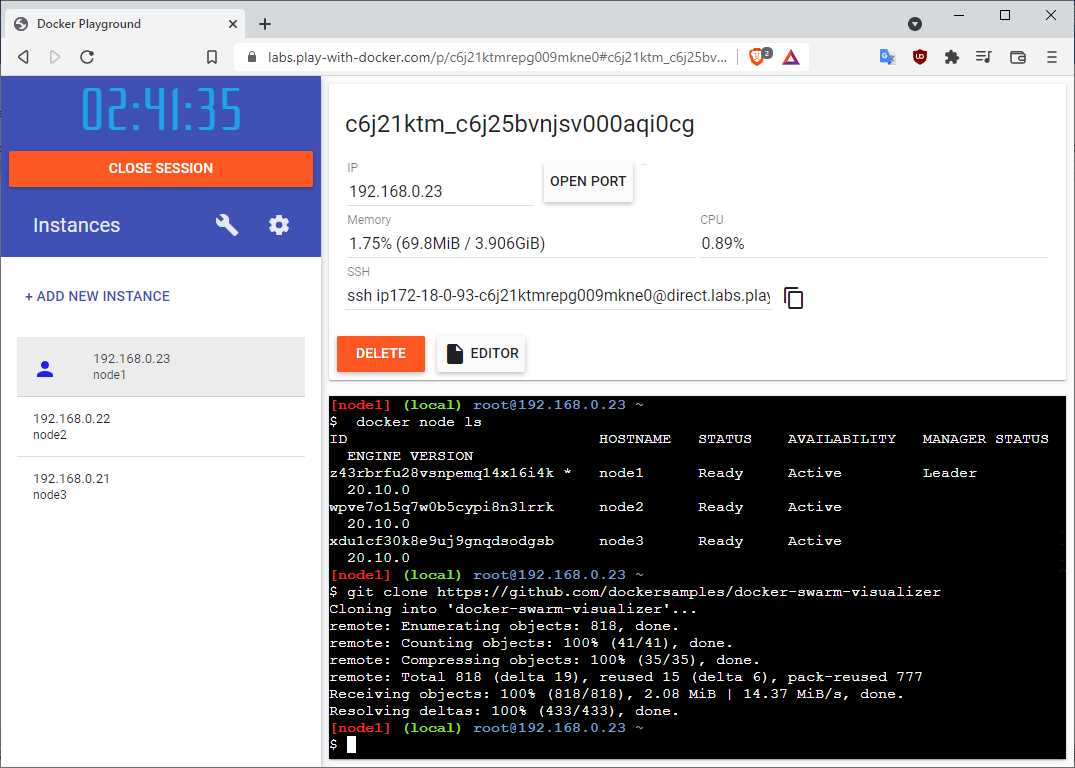
node2 и node3:

$ docker swarm join --token SWMTKN-1-52df54jq7h3nux2wiyaxjrm78hk1fvfu2uotksug4crr 192.168.0.23:2377



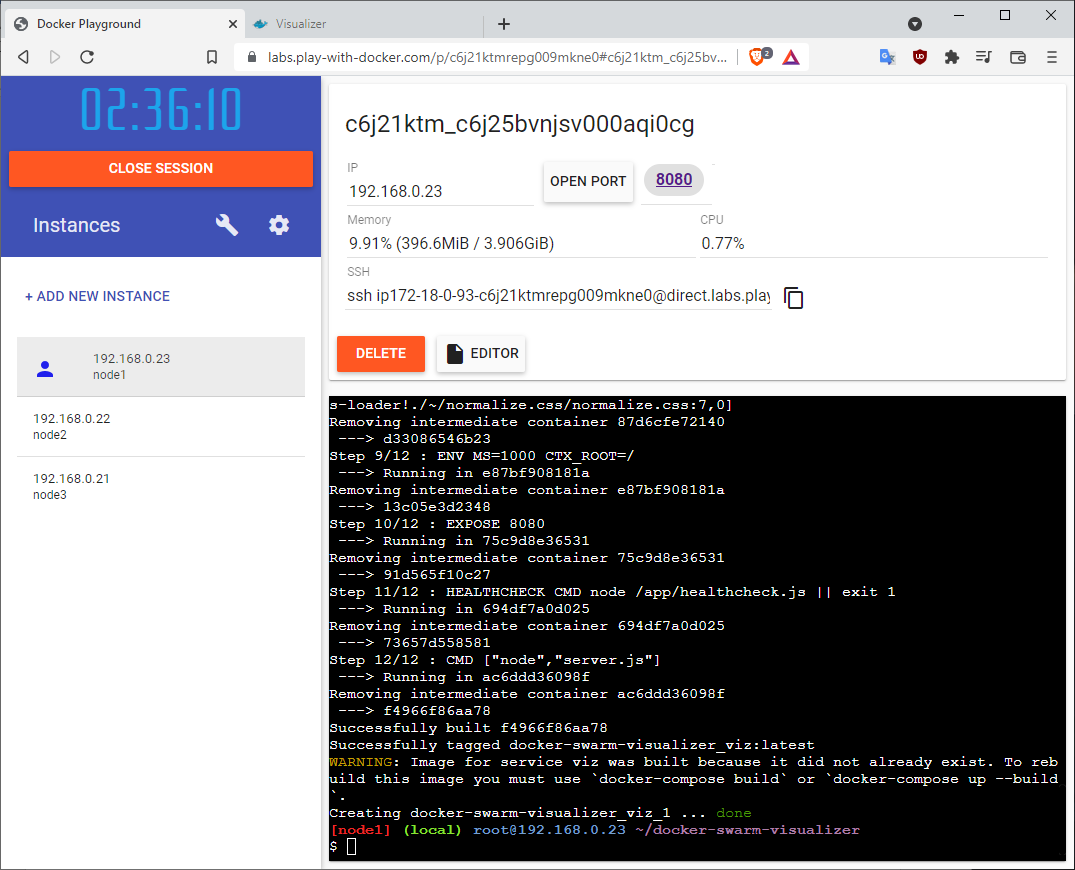


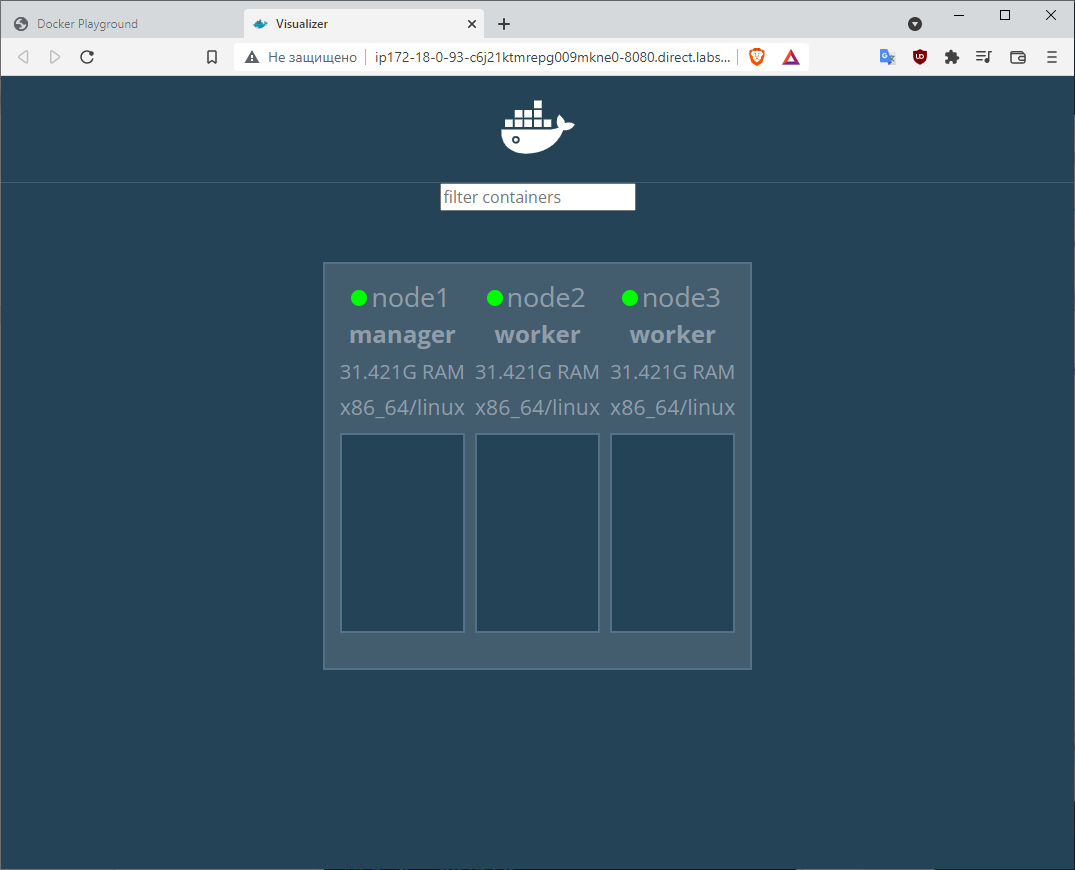
$ git clone https://github.com/dockersamples/docker-swarm-visualizer



$cd docker-swarm-visualizer

$docker-compose up -d

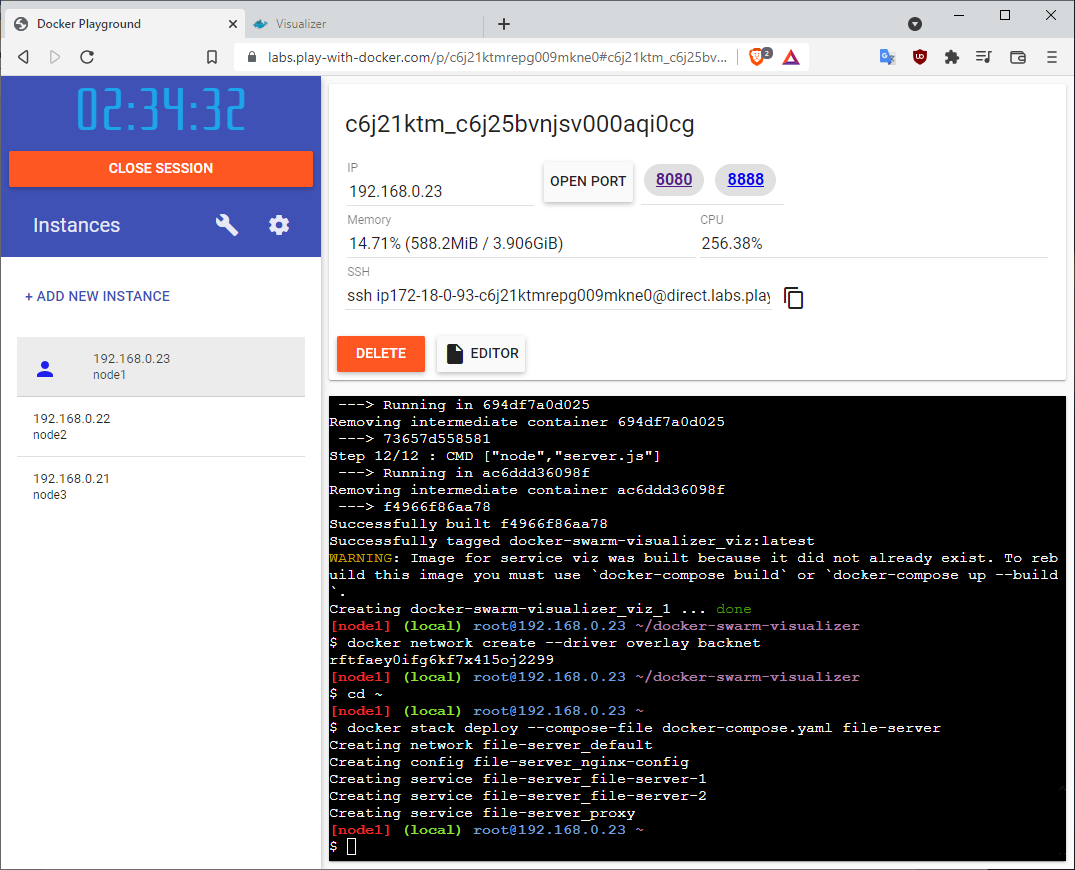


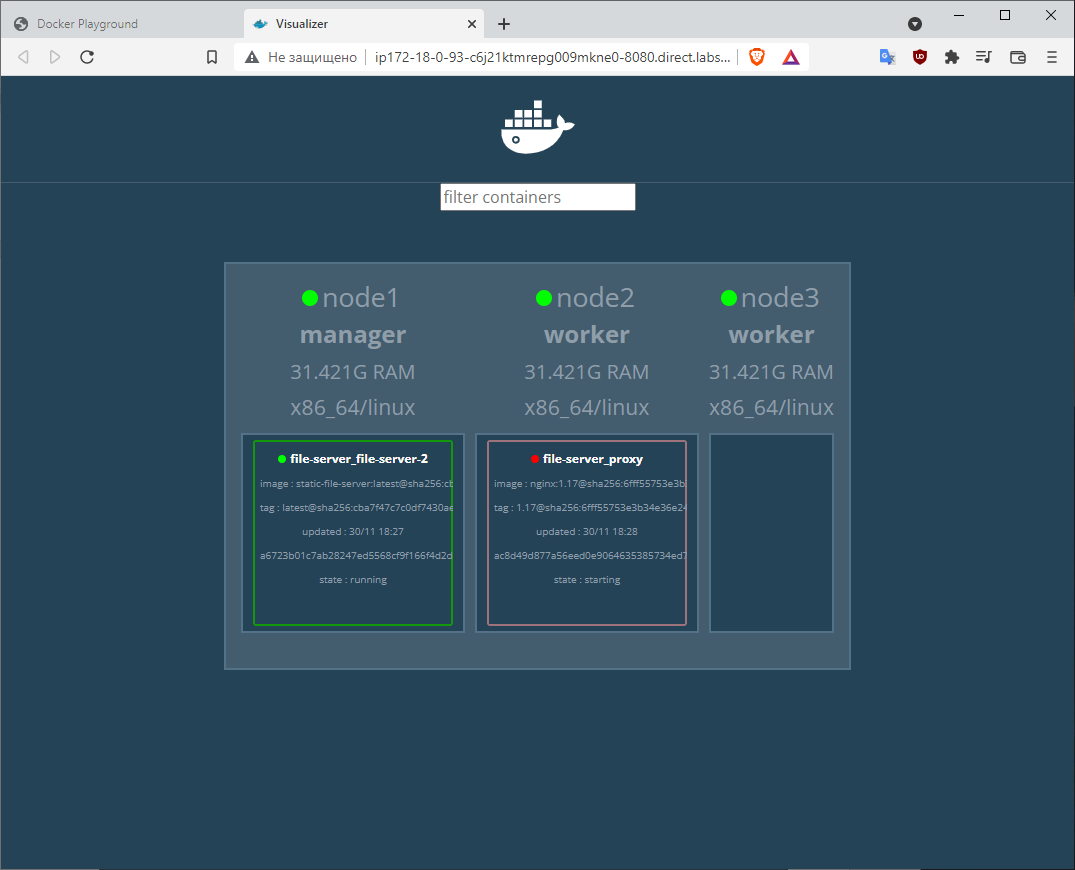


$ docker network create --driver overlay backnet

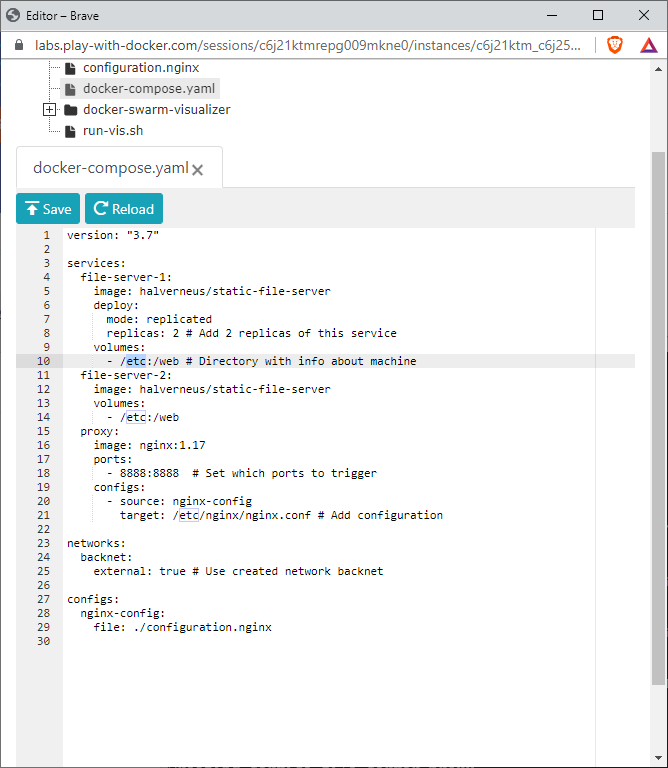
$ cd ~

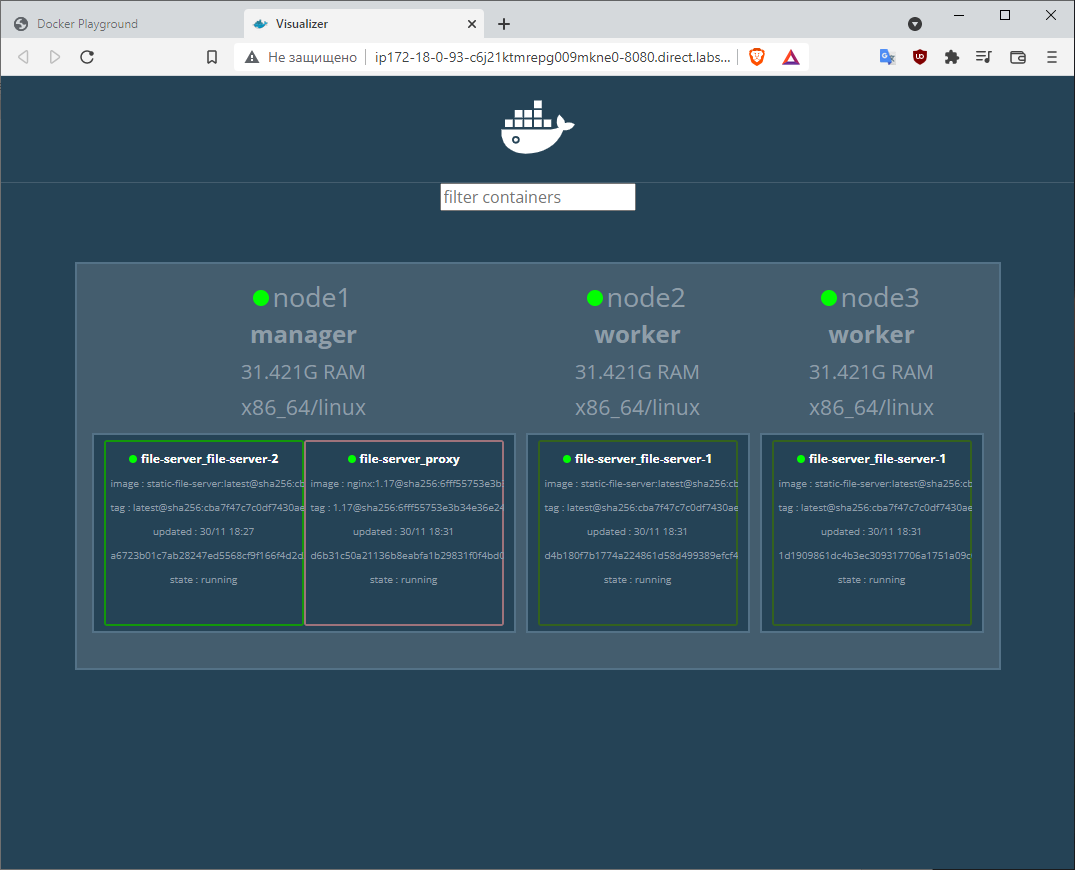
$ docker stack deploy --compose-file docker-compose.yaml file-server





Исправим ошибку в файле docker-compose.yaml – изменим /machine-info на /etc:





**ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы были изучены основы работы с Docker-образами, контейнерами, а также с технологиями Docker Compose и Docker Swarm.