**ssh <имя\_пользователя>@<публичный IP>– после открытия командной строки вводим данную команду, имя подключенного к виртуальной машине пользователя можно узнать, открыв ВМ на Яндекс Cloud и зайдя в раздел метаданные.**

**ssh** [**rezvanov@**51.250.45.124](mailto:rezvanov@51.250.45.124%20)

ssh -i <C:/Users/Павел/.ssh/id\_ed25519.pub> <pavelrezvanov >@<158.160.156.246>

**ssh -i <путь\_к\_ключу/id\_ed25519.pub> <** **pavelrezvanov >@<**158.160.156.246**>**

**sudo su – вводим команду, чтобы получить полные права после входа на сервер**

**cd ~ - перейти в корневую директорию**

**cd - - перейти на одну папку выше**

**ls –a – показать все папки, в том числе скрытые**

**ctrl + D - выйти из сервера**

**#cloud-config**

**runcmd: []**

**Установка Docker на сервер:**

https://docs.docker.com/engine/install/ubuntu/

docker version – проверить версию докера

исправить зависимости:

Вот как я исправил эту проблему: Обратитесь к [документации Docker Compose](https://docs.docker.com/compose/install/)

1. sudo curl -L https://github.com/docker/compose/releases/download/1.21.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
2. sudo chmod +x /usr/local/bin/docker-compose

После того, как вы выполните команду curl , она поместит docker-compose в

/usr/local/bin

которой нет в PATH. Чтобы исправить это, создайте символическую ссылку:

1. sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

И теперь, если вы сделаете: docker-compose --version

sudo systemctl start docker запустить докер

Добавим пользователя "ubuntu" в группу "docker" с помощью команды:

sudo usermod -aG docker имя\_пользователя

Далее можно добавить пользователя в группу "docker", чтобы использовать Docker Engine без "sudo" или необходимости переключаться на пользователя "root".

Перезапустить виртуальную машину

А только потом устанавливать зависимости, но команда:

Надо sudo вначале:

sudo docker-compose -f docker-compose.build.yml build

sudo rm -r data

sudo docker-compose up –d

cat ./metaboscan.sql | docker exec -i lab\_work\_dbn\_1 psql -U postgres -d postgres

**docker-compose up (сначала)**

**ctrl + C**

**docker-compose up -d**

Активация перезапуска приложения при ошибках или при перезагрузке сервера – ВМ:

cd /var/share

Следующая команда гарантирует автоперезапуск всех запущенных контейнеров.

$ docker update --restart always $(docker ps -q)

Использование пакетных менеджеров для этих целей не рекомендуется (<https://docs.docker.com/config/containers/start-containers-automatically/>) Вводить данную команду каждый раз, когда разворачивается новый контейнер.

**Установка СУБД**

sudo apt install postgresql

pg\_config --version – узнать версию

Установка Гита:

sudo apt install -y git

git --version

Клонируем репо:

git clone адрес

git clone [git@github.com:FimaLab/LIMS-for-metabolomics.git](mailto:git@github.com:FimaLab/LIMS-for-metabolomics.git)

1. Проверьте статус службы с помощью команды: `sudo systemctl status postgresql.service`.
2. Если сервис не запустился автоматически, можно сделать это вручную с помощью команды: `sudo systemctl start postgresql.service`.

cd LIMS-for-metabolomics

устанавливаем зависимости и дальнейшая инициализация по файлу редми

Ключи для развертнвания

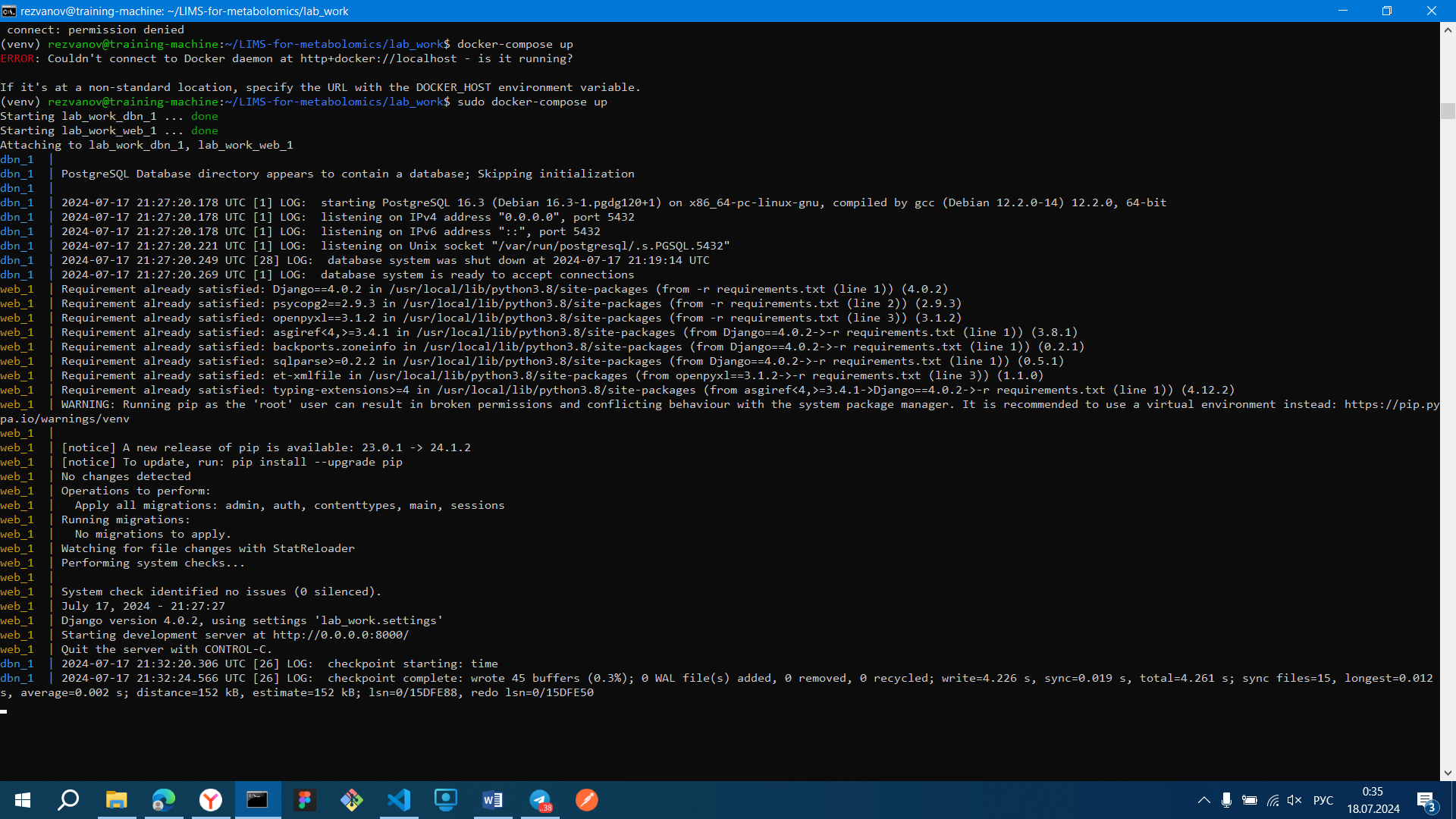
Создаем ключ на сервере

Открываем папку и файл с публичным ключом на сервере

cat id\_ed25519.pub

Добавляем его на гит в ключи для развёртывания конкретного репозитория

docker compose



**Для запуска контейнеров через Docker-compose используются следующие команды:**

1. **docker-compose build** — собрать проект.
2. **docker-compose up -d** — запустить проект.
3. **docker-compose down** — остановить проект.
4. **docker-compose logs -f [service name]** — посмотреть логи сервиса.
5. **docker-compose ps** — вывести список контейнеров в конкретной директории
6. **docker-compose exec [service name] [command]** — выполнить команду в контейнере.
7. **docker-compose images** — список образов.

Когда заходит новый пользователь:

docker stop $(docker ps -a -q)

docker rm $(docker ps -a -q)

Создание общей папки:

<https://losst.pro/obshhie-papki-linux>

cd /var/share

Установка Nginx — это HTTP-сервер.

sudo apt update *# обновляем список пакетов (программ), доступных для установки*

sudo apt install -y nginx *# устанавливаем nginx*

После установки нужно настроить файрвол — прослойку между сервером и внешним миром, которая решает, какие запросы пускать дальше, а какие отклонять.

В операционной системе Ubuntu по умолчанию установлен файрвол — ufw. Сконфигурируем его, разрешив некоторые виды трафика. Для этого используйте команды:

sudo ufw allow 'Nginx Full'

sudo ufw allow OpenSSH

sudo ufw allow 'Nginx Full' открывает на текущей машине два порта: 80 и 443, на них приходят http и https запросы к серверу. OpenSSH открывает порт 22 — соединение по ssh, чтобы мы могли подключаться к серверу через консоль и при работающем файрволе.

После этого нужно включить файрвол:

После этого нужно включить файрвол:

Скопировать кодBASH

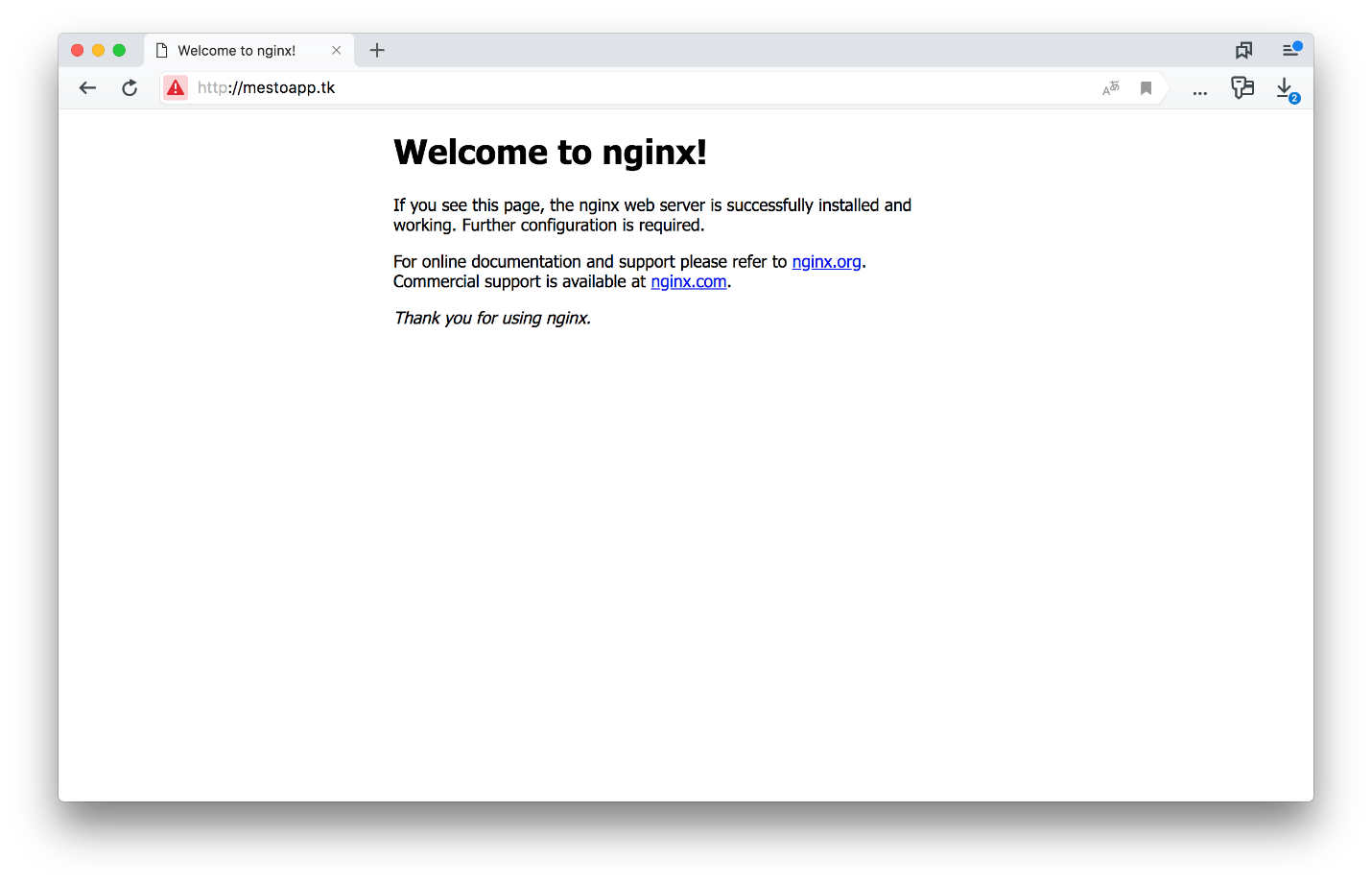
sudo ufw enable

Осталось запустить nginx и добавить его в автозапуск. Это можно сделать за один шаг при помощи следущей команды :

Скопировать кодBASH

sudo systemctl enable --now nginx

Откройте в браузере адрес вашего сервера(публичный IP). Вы должны увидеть:



Это значит, что nginx запустился и работает на 80 порту. Но сейчас он просто отдаёт стандартную html-страницу, а мы хотим, чтобы запросы перенаправлялись. Для этого нужно изменить конфигурационный файл nginx.

В nginx есть две директории для хранения конфигураций: sites-available/ и sites-enabled/. В первой хранятся конфигурации всех хостов, а во второй — ссылки только на активные. Перед редактированием конфигурации нужно создать символическую ссылку между sites-available и sites-enabled:

sudo ln -s /etc/nginx/sites-available/<имя\_конфига> /etc/nginx/sites-enabled/<имя\_конфига>

Для конфигурации по умолчанию (default) символическая ссылка уже создана, так что этот этап можно пропустить.

Теперь переходим к редактированию файла конфигурации. В командной строке введите:

sudo nano /etc/nginx/sites-available/default

# sudo значит запустить команду как суперпользователь

# nano — это текстовый редактор

# /etc/nginx/sites-available/default — путь к конфигурационному файлу nginx

Удалите содержимое этого файла и вставьте туда такую конфигурацию:

server {

listen 80;

root /var/www/html;

location / {

proxy\_pass http://localhost:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

После того как всё сделаете, нажмите Ctrl+X. Nano спросит у вас, хотите ли вы сохранить изменения, нажимайте y и Enter.

После обновления конфигурации nginx нужно перезапустить. Но если в конфигурации окажутся ошибки, то в лучшем случае работа приложения будет остановлена. Nginx позволяет изменять и тестировать конфигурацию «на лету». Для этого до перезапуска nginx введите команду:

sudo nginx -t

Она позволит проверить конфигурацию на наличие ошибок в синтаксисе команд. Если всё прошло успешно, вы получите такой ответ:

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok

nginx: configuration file /etc/nginx/nginx.conf test is successful

Уже после этого можно перезагрузить конфигурацию nginx:

sudo systemctl reload nginx

Теперь попробуйте послать из Postman запрос:

Пришел ответ - мы настроили проксирование запросов через nginx!

Расположение лендинга на сервере:

В командной строке Windows

scp -r ./landing/\* [rezvanov@51.250.45.124:/var/share](mailto:rezvanov@51.250.45.124:/var/share)

на сервере:

sudo chmod 0777 /var/www делаем католог общедоступным для редактирования и т.д

sudo cp index\_web.html /var/www

sudo nano /etc/nginx/sites-available/default

server {

listen 80;

server\_name \_;

root /var/www;

index index\_web.html;

}

sudo nginx –t

sudo systemctl reload nginx

Докер + стримлит

<https://docs.streamlit.io/deploy/tutorials/docker>

<https://maelfabien.github.io/project/Streamlit/#dockerfile>

https://www.rockyourcode.com/run-streamlit-with-docker-and-docker-compose/

scp -r ./dossier/\* rezvanov@51.250.45.124:/var/share/dossier

scp -r ./.streamlit/\* [rezvanov@51.250.45.124:/var/share/preclinical\_container/.streamlit](mailto:rezvanov@51.250.45.124:/var/share/preclinical_container/.streamlit)

**docker-compose build** — собрать проект.

**docker-compose up -d** — запустить проект

После изменений:

docker ps – узнать запущенные контейнеры во всей системе

удалить папки project и data в директории контейнера

sudo rm -r data

sudo rm -r project

docker rm --force preclinical\_container\_streamlit\_1 – принудительное удаление контейнера и заново сборка и запуск

docker update --restart always $(docker ps -q) – активировать перезапуск контейнеров

scp -r ./landing/\* rezvanov@158.160.173.155:/var/www

nano – редактировать текстовый файл

docker images – показать контейнеры

mv file newfile – переименовать файл

Порты для джанги:

В самих файлах джанги не надо порты менять, только в docker-compose.yml

ports: - "8503:8000"

Команды для приложения BioEque Manager

sudo docker-compose -p dossier build --no-cache

docker-compose -p dossier **up -d**

docker rm --force dossier\_streamlit\_1

при пересборке sudo docker-compose -p dossier build --no-cache

Надо внимательно с портами для сервера и корпоративной версии

На сервере:

8501 доклиника

8505 вторая икнока Переформатирование данных с прибора

8502 первичная обработка

Метабоскан дэш 8050

Джанго 8000

Биоэкви менеджер 8506

**React**

**ssh** [**rezvanov@**84.201.170.212](mailto:rezvanov@84.201.170.212)

docker-compose -p digital-lab-ecosystem build --no-cache

docker build --target build-stage -t my-react-app . для подробных логов при сборке

docker-compose -p digital-lab-ecosystem **up -d**

docker ps

scp -r ./public/\* rezvanov@84.201.170.212:/var/www/digital-lab-ecosystem/public

Новая конфигурация для nginx

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

# Сервер для HTTP на порту 8500

server {

listen 8500;

server\_name \_;

location / {

proxy\_pass http://localhost:8500; # IP и порт контейнера

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

location /css/ {

root C:/lab\_projects/var/www;

}

}

# Сервер для HTTP на порту 8500

server {

listen 8550;

server\_name \_;

location / {

proxy\_pass http://localhost:8550; # IP и порт контейнера

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

location /css/ {

root C:/lab\_projects/var/www;

}

}

} - **это видна**

**Сервер**

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

# Сервер для HTTP на порту 8500

server {

listen 8500;

server\_name \_;

location / {

proxy\_pass http://localhost:8500; # IP и порт контейнера

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

location /css/ {

root C:/lab\_projects/var/www;

}

}

}

}

sudo nano /etc/nginx/sites-available/default

docker ps

docker rm --force preclinical\_container\_streamlit\_1 – принудительное удаление контейнера и заново сборка и запуск

docker-compose -p digital-lab-ecosystem up –d – cамая рабочая команда

Валидация по доклинике:

**ls -l** — показывает время последнего изменения (модификации).

sudo lsof -i :8500 – узнать какой процесс занимает конкретный порт

sudo kill -9 920

Т.к приложение реакта и досье запускается через докер, то автозапуск nginx и mongo не нужены, их автозапуск будет мешать сооветсвующих приложениям