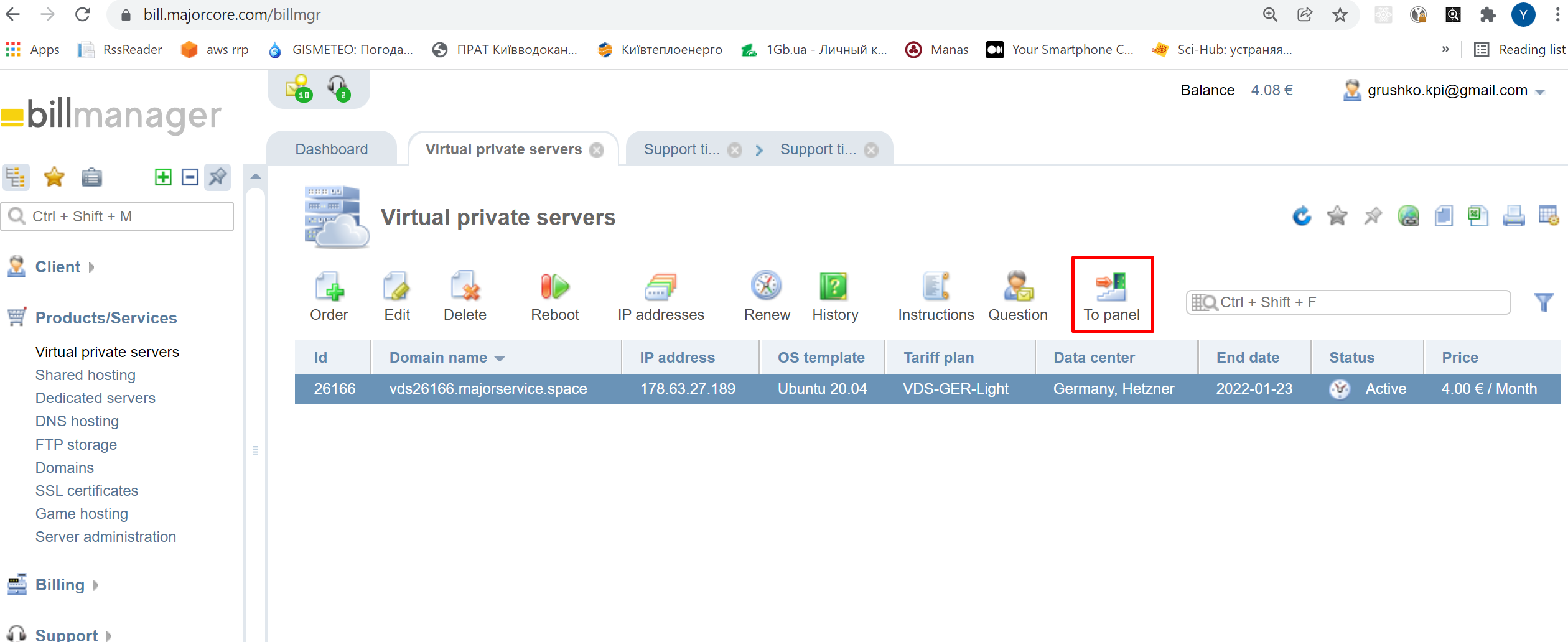
**Креди**

пароль від кабінету virtual machine (треба заходити з особистого кабінету): yjhlbxtcrbq *(нордический)*



**Креды к ssh:**

Ip:178.63.27.189

user: root

password: yjhlbxtcrbq *(нордический)*

**Креды к Гитхабу:**

Username: YaroslavGrushko

Password: yjhlbxtcrbq1 (нордический1)

**Информация о mail-cервере**

* Email: mail.helixtip.top

## DNSmanager - панель управления серверами имён

* **Ссылка:**<https://ns1.majorcore.com:1500/dnsmgr>
* **Пользователь:**user1016
* **Пароль:**IOhs6xf2Jwue

**working configuration:**

Ubuntu 20.0.4

node:16.13.1.

python: 3.8.10

flask: 1.1.4

pip install flask==1.1.4

flask-sqlAlchemy: 2.5.1 (latest 29.01.2022)

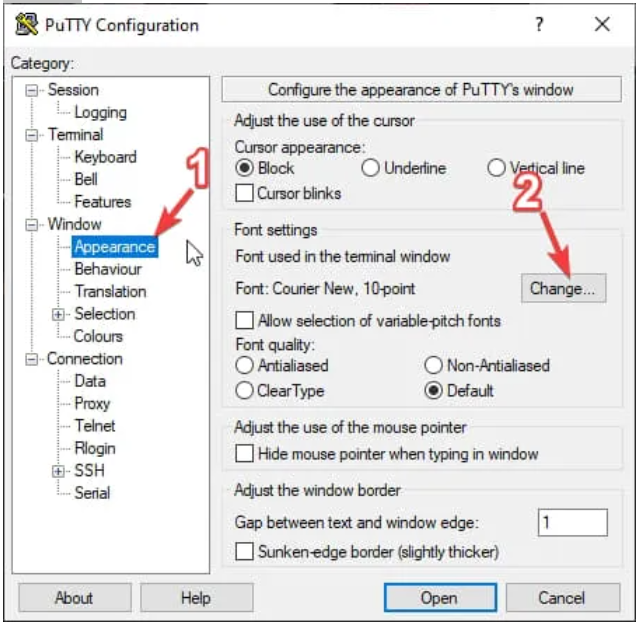
SQLAlchemy-1.4.31

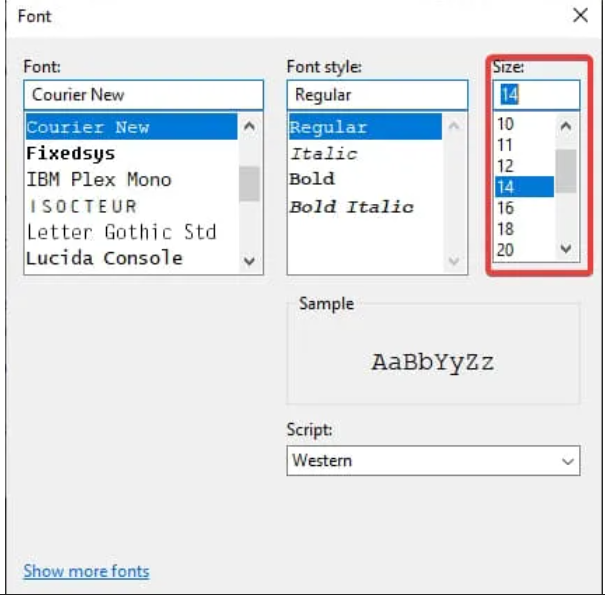
greenlet-1.1.2

pyjwt: 1.7.1

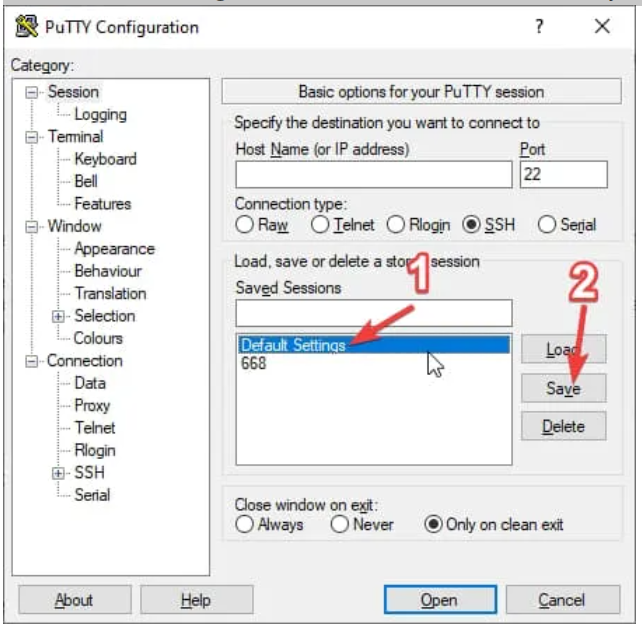
**Міняємо дефолтний розмір тексту в putty:**

Open PuTTY. In **PuTTY Configuration** window, select **Window** > **Appearance** > **Font settings** > **Change**





Now switch back to **Session** section in the left hand side. In **Load, save or delete a stored session**, select **Default Settings** (click once), then click **Save** to make your settings permanent.



**Додаємо юзера admin в Ubuntu:**

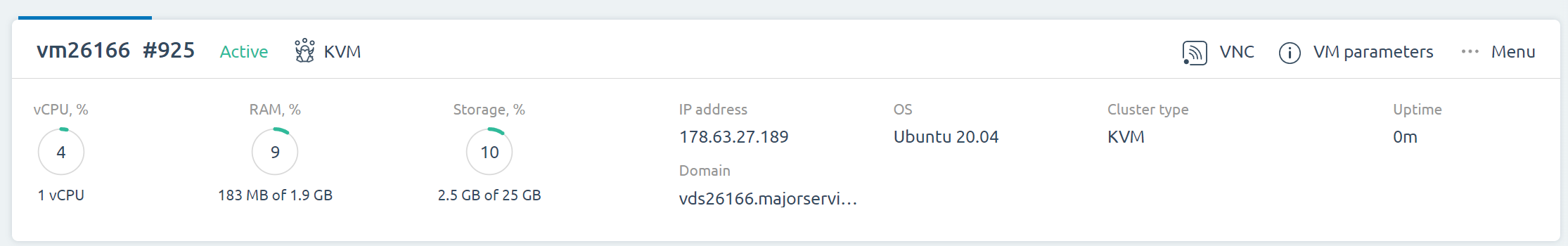
To add a new user in Ubuntu run

sudo adduser admin

Enter password and other needed info to create a user account on Ubuntu server.

user: admin   
password: yjhlbxtcrbq *(нордический)*

Загрузка пустої Ubuntu:



Login as admin:

su admin

Install [ufw](https://wiki.ubuntu.com/UncomplicatedFirewall), the Uncomplicated Firewall, and configure it to only allow external traffic on port 22 (ssh), 80 (http), 443 (https), 5432 (postgres) and 5000(flask). Any other ports will not be allowed.

$ sudo apt-get install -y ufw

$ sudo ufw allow ssh

$ sudo ufw allow http

$ sudo ufw allow 443/tcp

$ sudo ufw allow 5432/tcp

$ sudo ufw allow 5000/tcp

$ sudo ufw --force enable

$ sudo ufw status

Проверка открытых портов:

sudo ss -tulpn | grep LISTEN

список всех запущенных юзером процессов

ps ux

список всех процессов

top

**Створюємо postgresql БД**

Оновлюєм packages

sudo apt-get update && sudo apt-get upgrade

Install PostgreSQL by running the command below

sudo apt-get install postgresql

Перегружаем виртуальную машину и проверяем версию

psql -V

Настройка внешнего подключения в файле:

/etc/postgresql/12/main/postgresql.conf

За допомогою WinScp (заходимо під root) редагуємо цей файл. Дозволяємо підключатися з будь яких хостів. Змінюємо ось це:

**listen\_addresses = 'localhost'** на ось це:

**listen\_addresses = '\*'**

Bounding to 0.0.0.0 interface is not enough. Remote connections also need a proper authentication rule being set in **pg\_hba.conf**. By adding the following entry at the end, you will allow connections from any IP.

Міняємо ось це:

# "local" is for Unix domain socket connections only

local all all peer

# IPv4 local connections:

host all all 127.0.0.1/32 md5

# IPv6 local connections:

host all all ::1/128 md5

на ось це:

# "local" is for Unix domain socket connections only

local all all peer

# IPv4 local connections:

host all all 0.0.0.0/0 md5

# IPv6 local connections:

host all all ::/0 md5

Перезапускаємо postgresql сервер:

sudo systemctl restart postgresql

Заходимо в psql shell під ім’ям postgres:

sudo -u postgres psql

Додаємо до БД юзера admin з паролем нордический (Eng)

CREATE USER admin WITH password 'yjhlbxtcrbq';

Виходимо з psql:

\q

Встановлюємо нову локаль (ru)

Виводимо всі локалі

locale -a

Генеруємо нову локаль:

sudo locale-gen ru\_RU.UTF-8

Конфігуруємо локаль:

sudo dpkg-reconfigure locales

нажимаємо ок

перезапускаємо psql:

sudo systemctl restart postgresql

Створюємо нову Бд

Заходимо в psql під ім’ям postgres:

sudo -u postgres psql

CREATE DATABASE admin WITH owner=admin ENCODING = 'UTF-8' lc\_collate = 'ru\_RU.utf8' lc\_ctype = 'ru\_RU.utf8' template template0;

строка підключення у Фласку:

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgres://admin:yjhlbxtcrbq@178.63.27.189:5432/admin'

(ім’я бази admin)

To make an effect on the changes made, we need to restart the PostgreSQL service. We can do that by running the command below

Виходимо з psql:

\q

Заходимо в ssh, і виходимо в корінь: cd ../../

Потім набираємо команду:

sudo /etc/init.d/postgresql restart

Now that our PostgreSQL database is ready. we can log into PostgreSQL and create a username and a database.

sudo -u postgres -i

psql

We have to create a password for the user with the command below

ALTER USER admin WITH PASSWORD 'yjhlbxtcrbq';

To create a database with the user created, run

CREATE DATABASE admin WITH OWNER admin;

Виходимо з psql:

\q

Заходимо під admin-ом:

psql -U admin –W

Заходимо в postgres:

$ psql -U postgres

Додаємо права Superuser-a до ролі admin:

ALTER USER admin WITH SUPERUSER;

Проверить відкриті порти:

$ sudo ss -tulwn | grep LISTEN

Перевіряємо статус БД:

service postgresql status

перезапутити сервер postgres:

sudo systemctl restart postgresql.service

**Готуємо back-end для публікації на сервер**

Створюємо і активуємо віртуальне оточення (в windows):

Звходимо в директорію проєкту back-end:

cd back-end

python -m venv venv

cd venv/scripts

.\activate

В venv (локально) встановлюємо flask і python-dotenv:

pip install flask==1.1.4

pip install python-dotenv

Створюємо файл **.flaskenv** з конфігурацією оточення (що знаходиться поряд з app.py в проекті) в якому запишемо:

FLASK\_APP=app.py

FLASK\_ENV=development

Check flask version:

flask –version

встановлюємо flask-sqlAlchemy

pip install -U Flask-SQLAlchemy==2.5.1

В venv (локально) встановлюємо інші бібліотеки:

pip install flask\_sqlalchemy sqlalchemy pandas datetime pyjwt uuid

pip install --upgrade pyjwt==1.7.1

запишемо в файл requirments.txt у Windows всі необхідні бібліотеки:

pip freeze > requirements.txt

**Деплоїм файли проекту на сервер**

Встановлюємо nginx і git:

sudo apt-get -y install nginx git

Знайти домашню диреткорію користувача:

echo $HOME

Переходимо у домашню директорію користувача:

cd /home/admin

Скачуємо репозиторій створеного раніше проєкту на гітхаб:

git clone https://github.com/YaroslavGrushko/happy-life.git

скачуємо з конкретної гілки (Deploy-2):

git clone --single-branch --branch Deploy-2 [https://github.com/YaroslavGrushko/happy-life.git](https://github.com/YaroslavGrushko/happy-life.git%20)

оновлюємо з конкретної гілки (після клонування з конкретної гілки воно запам’ятовує звідки ми клонували і наступний раз, щоб оновити з тої ж гілки слід просто набрати git pull):

git pull

Коли є merge-конфлікти:

git add \*

git stash

git pull

**Створюємо back-end на сервері**

Переходимо до ssh серверу helixtiptop

cd back-end

подивитись файли/папки поточної директорії:

ls -a

перейти в папку проєкту:

cd happy-life

видалити директорію (якщо треба):

sudo rm -r -f happy-life

встановлюємо з корня додаткові бібліотеки python:

sudo apt-get update –y

sudo apt-get -y install python3.8-venv python3.8-dev python3.8-psycopg2 libpq-dev

sudo apt-get install -y python3.8-psycopg2

Проблема:

E: Unable to locate package psycopg2

Рішення:

sudo apt-get update –y

sudo apt-get install -y python3-psycopg2

переходимо у happy-life/back-end і створюємо віртуальне оточення

в linux:

python3 -m venv venv

source venv/bin/activate

Видає помилку, що не встановлений python3.8-venv:

Рішення:

sudo apt install python3.8-venv

переходимо у папку з віртуальним оточенням

cd back-end/venv/bin (ubuntu)

запускаємо віртуальне оточення:

linux:

source activate

деактивация venv (linux)

deactivate

переходимо в папку back-end і встановлюємо всі пакети з requirmnetns.txt

pip install -r requirements.txt

Запусттити наш flask – додаток можна з директорії back-end

flask run

**Зараз будемо налаштовувати бекенд:**

Переходимо з корню в папку happy-life/back-end/venv/bin:

cd happy-life/back-end/venv/bin

запускаємо віртуальне оточення:

source activate

повертаємось в back-end:

cd ../../

встановлюємо, psycopg2, а також пакети яких не вистачає в requirements.txt (в наслідок того,що цей файл створювався локално з python 3.10.2 у windows, а в Ubuntu на сервері встановлений python 3.8 де немає цих пакетів) у venv:

pip install psycopg2-binary flask\_sqlalchemy pyjwt pandas

Встановлюємо gunicorn:

pip install gunicorn

переходимо у директорію happy-life/back-end і запускаємо gunicorn:

gunicorn -b 127.0.0.1:5000 app:app

перейдімо в /etc/systemd/system

створюємо helixtiptop.service  – файл за допомогою WinScp:

відкриваємо файл і вставляємо туди наступний вміст:

**[Unit]**

**Description=Flask api for helixtip-top project**

**After=network.target**

**[Service]**

**User=admin**

**WorkingDirectory=/home/admin/happy-life/back-end**

**ExecStart=/home/admin/happy-life/back-end/venv/bin/gunicorn -b 127.0.0.1:5000 app:app --timeout 300 --graceful-timeout 300**

**Restart=always**

**[Install]**

**WantedBy=multi-user.target**

Запуск gunicorn в ручному режимі:

зупиняємо сервіс

sudo systemctl stop helixtiptop

Переходимо: **cd happy-life/back-end**

admin@helixtip:~/happy-life/back-end/venv/bin/gunicorn -b 127.0.0.1:5000 app:app --timeout 300

**зупинити gunicorn**

ctrl+c

Перезапускаємо систему:

sudo systemctl daemon-reload

Запускаємо наш flask-бекенд, як сервіс

sudo systemctl start helixtiptop

Зупинити flask-бекенд – сервіс:

sudo systemctl stop helixtiptop

Щоб перевірити що сервіс стартував введіть команду:

sudo systemctl status helixtiptop

переходимо в папку back-end і встановлюємо всі пакети з requirmnetns.txt

pip install -r requirements.txt

**Налаштовуємо nginx**

nginx знаходиться в :

cd /etc/nginx

видаляємо конфігурацю nginx по –замовченню:

sudo rm /etc/nginx/sites-enabled/default

sudo rm /etc/nginx/sites-enabled/helixtip.top

щоб скопіювати файл:

cp helixtip.top helixtip\_copy.top

Дивлячись на те, що наш проєкт лежить в home/admin/smartphone-test директорії, створюємо конфігураційний файл helixtip.top в директорії **/etc/nginx/sites-available** з таким вмістом

**# Default server configuration**

**#**

**server {**

**listen 80;**

**listen [::]:80;**

**# SSL configuration**

**#**

**# listen 443 ssl default\_server;**

**# listen [::]:443 ssl default\_server;**

**#**

**# Note: You should disable gzip for SSL traffic.**

**# See: https://bugs.debian.org/773332**

**#**

**# Read up on ssl\_ciphers to ensure a secure configuration.**

**# See: https://bugs.debian.org/765782**

**#**

**# Self signed certs generated by the ssl-cert package**

**# Don't use them in a production server!**

**#**

**# include snippets/snakeoil.conf;**

**root /home/admin/happy-life/build;**

**# Add index.php to the list if you are using PHP**

**index index.html;**

**server\_name helixtip.top;**

**location / {**

**# First attempt to serve request as file, then**

**# as directory, then fall back to displaying a 404.**

**try\_files $uri $uri/ =404;**

**}**

**#backend location (localhost:5000)**

**location /api/flask {**

**include proxy\_params;**

**proxy\_pass http://localhost:5000;**

**}**

**# pass PHP scripts to FastCGI server**

**#**

**#location ~ \.php$ {**

**# include snippets/fastcgi-php.conf;**

**#**

**# # With php-fpm (or other unix sockets):**

**# fastcgi\_pass unix:/var/run/php/php7.4-fpm.sock;**

**# # With php-cgi (or other tcp sockets):**

**# fastcgi\_pass 127.0.0.1:9000;**

**#}**

**# deny access to .htaccess files, if Apache's document root**

**# concurs with nginx's one**

**#**

**#location ~ /\.ht {**

**# deny all;**

**#}**

**}**

**# Virtual Host configuration for example.com**

**#**

**# You can move that to a different file under sites-available/ and symlink that**

**# to sites-enabled/ to enable it.**

**#**

**#server {**

**# listen 80;**

**# listen [::]:80;**

**#**

**# server\_name example.com;**

**#**

**# root /var/www/example.com;**

**# index index.html;**

**#**

**# location / {**

**# try\_files $uri $uri/ =404;**

**# }**

**#}**

, де root – це директорія з якої читається статичний контент сайту,

index – це пускова точка статичного контенту сайту

Перезапускаємо nginx, щоб застосувати зміни:

sudo systemctl reload nginx

переходимо в корінь серверу

cd ../../../../

Підключаємо сімлінк до даного файлу, що ми створили (пов’язуємо диреткорію sites\_available (там де ми створили файл) та пусту директорію sites-enabled):

sudo ln -s /etc/nginx/sites-available/helixtip.top /etc/nginx/sites-enabled/helixtip.top

**Підлючення безкоштовного ssl від Let’s Encrypt:**

Getting a certificate from Let's Encrypt is fairly easy, since the whole process is automated. Assuming you are using an Ubuntu based server, you have to begin by installing their open source certbot tool on your server:

$ sudo apt-get install software-properties-common

$ sudo add-apt-repository -r ppa:certbot/certbot

$ sudo apt update

$ sudo apt-get update

$ sudo apt-get install certbot

Переходимо в home/admin і Створюємо директорію (можна з WinScp):

mkdir certbot\_verification

щоб продивитися всі папки в поточній директорії:

ls -a

## Встановлюємо SSL A+ Grade

За допомогою WinScp створюємо директорію sslDhParam в директорії etc (яка знаходиться в корні).

Using the openssl tool, you can run the following command:

sudo openssl dhparam -out **/etc/sslDhParam/dhparam.pem** 2048

Запускаємо Cerbot, щоб отримати сертифікат:

$ sudo certbot certonly --webroot -w /home/admin/certbot\_verification -d helixtip.top

Через 90 днів (31.12.2020) сертифікат треба поновити наступною командою:

sudo certbot renew

заходимо в папку /etc/nginx/sites-available і відкриваємо файл helixtip.top (це конфігурація nginx для нашого сайту):

Змінюємо його вміст на такий:

**# Default server configuration**

**#**

**server {**

**listen 80;**

**server\_name helixtip.top;**

**return 301 https://helixtip.top$request\_uri;**

**}**

**server {**

**# SSL configuration**

**#**

**# listen 443 ssl default\_server;**

**# listen [::]:443 ssl default\_server;**

**listen 443 ssl;**

**ssl on;**

**ssl\_certificate /etc/letsencrypt/live/helixtip.top/fullchain.pem;**

**ssl\_certificate\_key /etc/letsencrypt/live/helixtip.top/privkey.pem;**

**ssl\_dhparam /etc/ssldhparam/dhparam.pem;**

**# include snippets/snakeoil.conf;**

**server\_name helixtip.top;**

**access\_log /var/log/nginx/nginx.vhost.access.log;**

**error\_log /var/log/nginx/nginx.vhost.error.log;**

**#root /home/admin/happy-life/build;**

**# Add index.php to the list if you are using PHP**

**#index index.html;**

**# for letsencrypt (cerbot) varification**

**location ~ /.well-known {**

**root /home/admin/certbot\_verification;**

**}**

**location / {**

**#create-react-app:**

**#root /home/admin/happy-life/build;**

**#index index.html;**

**#try\_files $uri /index.html;**

**#next.js app:**

**root /home/admin/happy-life;**

**#reverse proxy for next js server**

**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;**

**}**

**#backend location (localhost:5000)**

**location /api/flask {**

**include proxy\_params;**

**proxy\_pass http://localhost:5000;**

**proxy\_read\_timeout 300;**

**proxy\_connect\_timeout 300;**

**proxy\_send\_timeout 300;**

**}**

**}**

Перезапускаєємо nginx, щоб застосувати зміни:

sudo systemctl reload nginx

Перевірка правильності конфігурації nginx:

sudo nginx -t

Перезапускаємо nginx

sudo systemctl reload nginx

**Встановлюємо node.js:**

Встановлюємо curl:

sudo apt install curl

Встановлюємо node 16 (lts) разом з npm 8.1.2 з nodeSource репозиторію:

curl -fsSL https://deb.nodesource.com/setup\_16.x | sudo -E bash -

sudo apt-get install -y nodejs

перевіряємо версію:

node -v

При потребі видаляємо node:

sudo apt-get remove nodejs

sudo apt-get remove npm

Then go to /etc/apt/sources.list.d and remove any node list if you have. Then do a

sudo apt-get update

Check for any .npm or .node folder in your home folder and delete those.

If you type

which node

you can see the location of the node. Try

 which npm

Локально:

Закоментувати вміст файлу: **pages/api/[...args].js**

З папки проєкту: **cd happy-life**

**git pull**

заходимо в директорію happy-life та встановлюємо всі бібліотеки з package.json:

cd home/admin/happy-life

npm i

робимо build next-проєкту на сервері

npm run build

встановлюємо pm2 задля того, щоб запустити next як сервіс:

sudo npm i --global pm2

Запускаємо сервіс next.js (pm2) **із папки проєкту**:

pm2 start npm --name "next" -- start

Зупиняємо сервіс next.js (pm2):

з корня **cd home/admin/happy-life**

pm2 stop next

**Якщо треба оновити проєкт:**

Якщо треба оновити картинки локально, то слід скачати їх з сервера (за допомогою winSCP)

Локально:

Закоментувати вміст файлу: **pages/api/[...args].js**

З папки проєкту: **cd happy-life**

**git pull**

**sudo systemctl reload nginx**

**sudo systemctl daemon-reload**

**sudo systemctl stop helixtiptop**

**sudo systemctl start helixtiptop**

**pm2 stop next**

**npm i (якщо є нові бібліотеки)**

**npm run build**

**pm2 start npm --name "next" -- start**

**Щоб видалити процес: pm2 delete 0**

**pm2 start 0**

**bash ./front-r.sh**

**Коли треба оновити фронтенд на хості:**

З корню проекта **admin@helixtip:~$**

bash ./front-p.sh

**Запустити бекенд руками задля дебагінгу:**

Переходимо в happy-life/back-end/venv/bin/

Активуємо віртуальне оточення:

linux:

source activate

переходимо в папку happy-life/back-end і запускаємо gunicorn:

gunicorn -b 127.0.0.1:5000 app:app

**pip install openpyxl**