Разворот приложений на Linux

1. **Обновление системы и установка необходимых для работы пакеты**

   Обновите систему
   $ dnf update –y

   Установите компоненты для работы.
   $ dnf install ffmpeg nginx docker-ce docker-ce-cli dotnet-sdk-3.1 postgresql13-server postgresql13-contrib postgresql13 libgdiplus wget gcc openssl-devel tk-devel readline-devel ncurses-devel ncurses-c++-libs gdbm-devel sqlite libffi libsq3-devel bzip2-devel tk-devel uuid-devel libffi libffi-devel uuid uuid-devel libuuid-devel lzma-devel libnsl2-devel git

   Сконфигурируете SELinux или отключите его
   $ Для этого в файле /etc/selinux/config укажите SELINUX=disabled вместо SELINUX=enforcing.

   Перезагрузите систему
   $ Reboot

2. **Сборка приложения**
   Склонируйте репозиторий с основным приложением в удобное для вас место, любым удобным способом

   Перейдите в корневую папку с репозиторием и выполните восстановление пакетов
   $ dotnet restore --runtime linux-x64 RmsSolution.sln --configfile ./.nuget/nuget.config"

   Выполните сборку проекта из папки проекта
   $ dotnet publish ./BarsUp.WebHost/BarsUp.WebHost.csproj --configuration Release --output /opt/lifeface/ --runtime linux-x64

3. **Сборка Telegram бота**
   Склонируйте репозиторий с основным приложением в удобное для вас место, любым удобным способом

   Перейдите в корневую папку с репозиторием и выполните восстановление пакетов
   $ dotnet restore --runtime linux-x64 IMonitorYourPsychoStateBot.sln

   Выполните сборку проекта из папки проекта
   $ dotnet publish IMonitorYourPsychoStateBot.sln --configuration Release --output /opt/tg_bot/-- runtime linux-x64

4. **Разворот Python сервисов обработки изображений**
   Установите Python 3.8.19
   $ git clone https://github.com/yyuu/pyenv.git /usr/local/pyenv
   $ export PYENV_ROOT=/usr/local/pyenv

$ /usr/local/pyenv/bin/pyenv install 3.8.19

Склонируйте репозиторий с основным приложением в /opt/RestServerPython/

Создайте виртуальное окружение для RestServerPython сервиса

$ /root/.pyenv/versions/3.8.19/bin/virtualenv --prompt="(RSP)" --python=/root/.pyenv/versions/3.8.19/bin/python /opt/RestServerPython/venv

Создайте сервис в systemd и наполните его описанием

$ touch /etc/systemd/system/RestServerPython.service

RestServerPython.service

Отредактируйте config.py

```
APP_CONFIG = {
'UPLOAD_FOLDER': "Путь до папки, где будут лежать изображения",
'fer_model': "путь до файла conv_model_0123456.h5",
'bars_model': "путь до файла bars_fer6.hdf5",
}
```

Перейдите в директорию с приложением и установите зависимости

```
$ source venv/bin/activate
$ pip install -r requirements.txt
$ deactivate
```

Склонируйте репозиторий с основным приложением в /opt/ModelService/

Создайте виртуальное окружение для ModelService сервиса

$ /root/.pyenv/versions/3.8.19/bin/virtualenv --prompt="(MS)" --python=/root/.pyenv/versions/3.8.19/bin/python /opt/ModelService/venv

Создайте сервис в systemd и наполните его описанием

$ touch /etc/systemd/system/ModelService.service

ModelService.service

Отредактируйте config.py

```
APP_CONFIG = {
'model': "путь до файла conv_model_0123456.h511"
}
```

Перейдите в директорию с приложением и установите зависимости

```
$ source venv/bin/activate
$ pip install -r requirements.txt
$ deactivate
```

5. **Разворот Docker сервиса обработки изображений**

Распакуйте архив с приложением в папку
$ unzip aumodel.zip -d /tmp/aumodel/

Выполните сборку Docker образа
$ docker build -t aumodel .

Запустите контейнер с приложением
$ docker run --rm --name aumodel -p 5557:5555 aumodel

6. **Разворот Telegram бота**

Создайте сервис в systemd и наполните его описанием
$ touch /etc/systemd/system/tg_bot.service


tg_bot.service

Сделайте файл запуска приложения исполняемым
$ chmod +x /opt/bot/IMonitorYourPsychoStateBot

Отредактируйте файл конфигурации приложения
$ mcedit appsettings.json
// Токен от телеграм бота
"BotToken":

// Порт и адрес основного приложения
"FaceValidationServiceUrl": "http:// Порт и адрес основного приложения **/mobileapi/photo/validate",**
"FaceProcessServiceUrl": "http:// Порт и адрес основного приложения **/mobileapi/photo/process",**
"VideoProcessServiceUrl": "http:// Порт и адрес основного приложения **/VideoProcessing/Process",**
"AuthLoginUrl": "http:// Порт и адрес основного приложения **/mobileapi/Auth/BotLogin",**
"AuthRegisterUrl": "http:// Порт и адрес основного приложения **/mobileapi/Auth/BotRegistration",**
"AuthGetServerDateUrl": "http:// Порт и адрес основного приложения **/mobileapi/Auth/GetCurrentDate",**
"FeedbackCreateUrl": "http:// Порт и адрес основного приложения **/mobileapi/Feedback/SaveFeedback",**
"TrialsUpdateUrl": "http:// Порт и адрес основного приложения **/mobileapi/Trials/Update",**
"GetReportDataUrl": Порт и адрес основного приложения **/mobileapi/PsychoEmotionalState/GetBotPsychoStateAsync",**
"UpdateUserProfileUrl": "http:// Порт и адрес основного приложения **/mobileapi/Profile/UpdateProfileAsync",**
"GetTrialsPhotoUrl": "http:// Порт и адрес основного приложения **/mobileapi/Photo/BotGetTrialsImages"**

// База данных для подключения приложения tg_bot
"ConnectionStrings": {

7. **Разворот основного приложения**

Создайте сервис в systemd и наполните его описанием

$ touch /etc/systemd/system/lifeface.service


lifeface.service

Сделайте файл запуска приложения исполняемым
$ chmod +x /opt/lifeface/BarsUp.WebHost

Отредактируйте файл конфигурации приложения
$ mcedit appsettings.json
// Адрес и порт публикации приложение посредством Kestrel
"urls":

Отредактируйте файл конфигурации приложения
mcedit barsup.config.default.json
// База данных для подключения приложения lifeface
"Database": {
"ConnectionString":

// Адреса подключения к сервисам обработки данных
"AppSettings": {
"ServiceUrls": {
"EmotionAnalyzeServiceUrl": "Адрес и порт RestServerPython.service **/ProcessFace",**
"StressService": " Адрес и порт ModelService. **/ProcessEmotions",**
"AnxietyService": " Адрес и порт ModelService **/ProcessEmotions",**
"DepressService": " Адрес и порт ModelService.service **/ProcessEmotions",**
"FaceValidationServiceUrl": " Адрес и порт RestServerPython.service **/ValidateFace",**
"ActionUnitService": " Адрес и порт докер контейнера **/process"**

// Путь до бинарника ffmpeg (узнать можно командой **which mmpeg**)
"FFmpegSettings": {
"Path":

8. **Настройте Nginx**
Создайте три conf файла в директории /etc/nginx/conf.d для основного приложения и python сервисов.
$ server {
listen 80;
location / {
proxy_pass http://127.0.0.1:5005;

```
        client_max_body_size 200M;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_read_timeout 600;
    }
}

$ server {
    listen 81;
    location / {
        proxy_pass http://127.0.0.1:5555;
        client_max_body_size 200M;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_read_timeout 600;
    }
}

$ server {
    listen 82;
    location / {
        proxy_pass http://127.0.0.1:5556;
        client_max_body_size 200M;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_read_timeout 600;
    }
}
```

9. **Конфигурация БД Postgres**

Смените локаль на русскую

```
$ localectl set-locale LANG=ru_RU.UTF-8
```

Инициализируйте и запустите базу данных

```
$ /usr/pgsql-13/bin/postgresql-13-setup initdb
$ systemctl enable postgresql-13.service --now
```

Создайте пользователя в бд, базы данных и выполните восстановление базы из дампа

```
$ CREATE USER bars WITH PASSWORD '*****';
$ CREATE DATABASE lifeface WITH OWNER bars;
$ CREATE DATABASE bot WITH OWNER bars;
```

10. **Перезапустите демонов и запустите службу с приложением**

```
$ systemctl daemon-reload
$ systemctl --now enable RestServerPython.service
$ systemctl --now enable ModelService.service
```

```
$ systemctl --now enable lifeface.service
$ systemctl --now enable tg_bot.service
```