# Заметки для выполнения тестового задания на DevOps-практикум

## Запуск приложения локально:

1. Операционная система Ubuntu 18.04
2. Обновляем индексы пакетов в системе под суперпользователем:

sudo apt update

1. Обновляем пакеты по до последних версий

sudo apt upgrade

1. Установим git для клонирования репозитория

sudo apt install git

1. Создадим папку для данной задачи

mkdir test

1. перейдём в неё

cd test

1. Склонируем репозиторий в папку test

git clone <https://github.com/anfederico/Flaskex>

1. В файле runtime.txt содержится наименовании версии python (3.6.0), которая будет использоваться. По умолчанию на данной ОС стоит версия 3.6.9. Поэтому мы установим дополнительную версию python 3.6.0, с использованием команды make altinstall

cd /opt

sudo wget https://www.python.org/ftp/python/3.6.0/Python-3.6.0.tgz

sudo tar xzf Python-3.6.0.tgz

cd Python-3.6.0

sudo ./configure —enable-optimizations

sudo make altinstal

1. Переключимся на дополнительную версию python 3.6.0

update-alternatives --config python3

1. Установим pip для нашей версии python и обновим её

apt install python3-pip

python3.6 -m pip install --upgrade pip

1. В файле requirements.txt не указано ни одной версии компонентов. Это вызвало сомнения, поэтому поискав на просторах интернета нашёл версии компонентов для реализации данного приложения

<https://www.natanetwork.com/portal/knowledgebase/59/Framework-Flask-App-Flaskex-di-cPanel.html>

1. Указываем версии в requirements.txt и запускаем сборку приложения из папки Flaskex

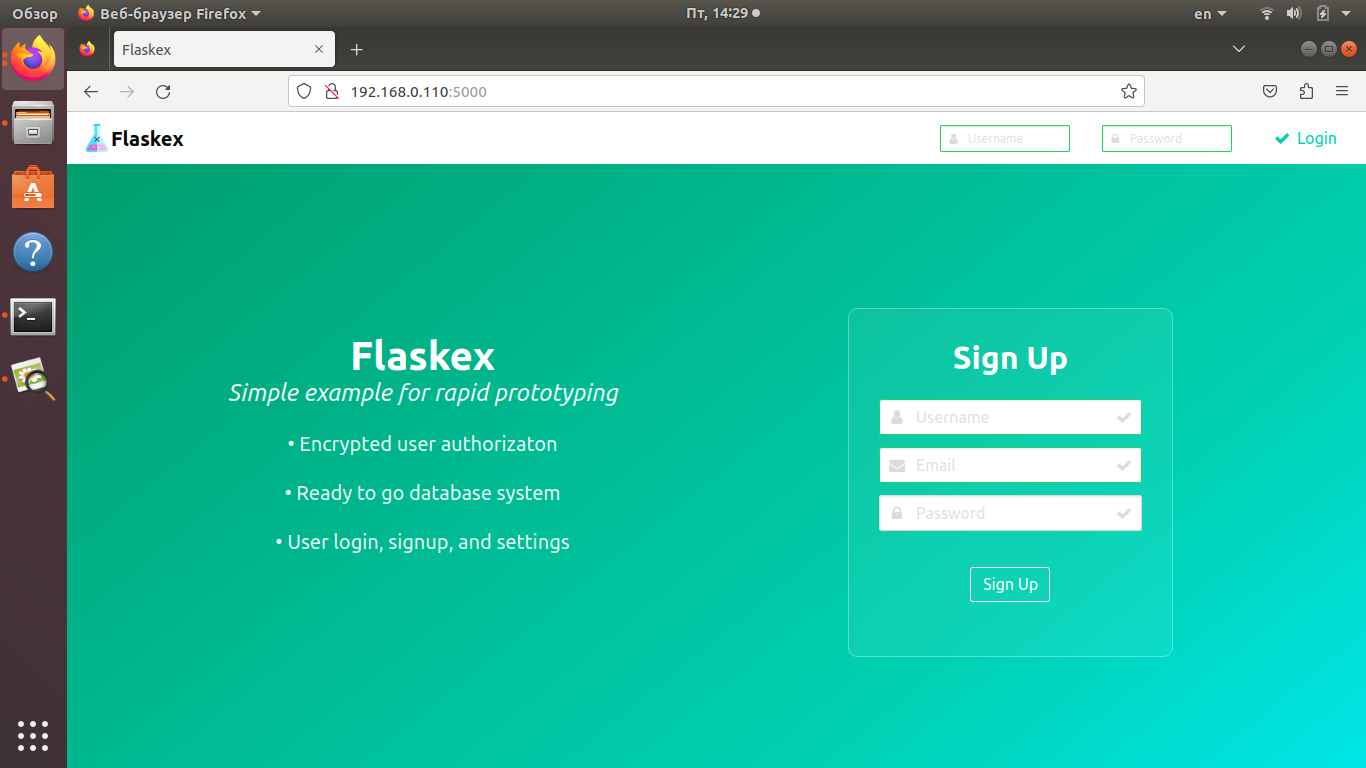
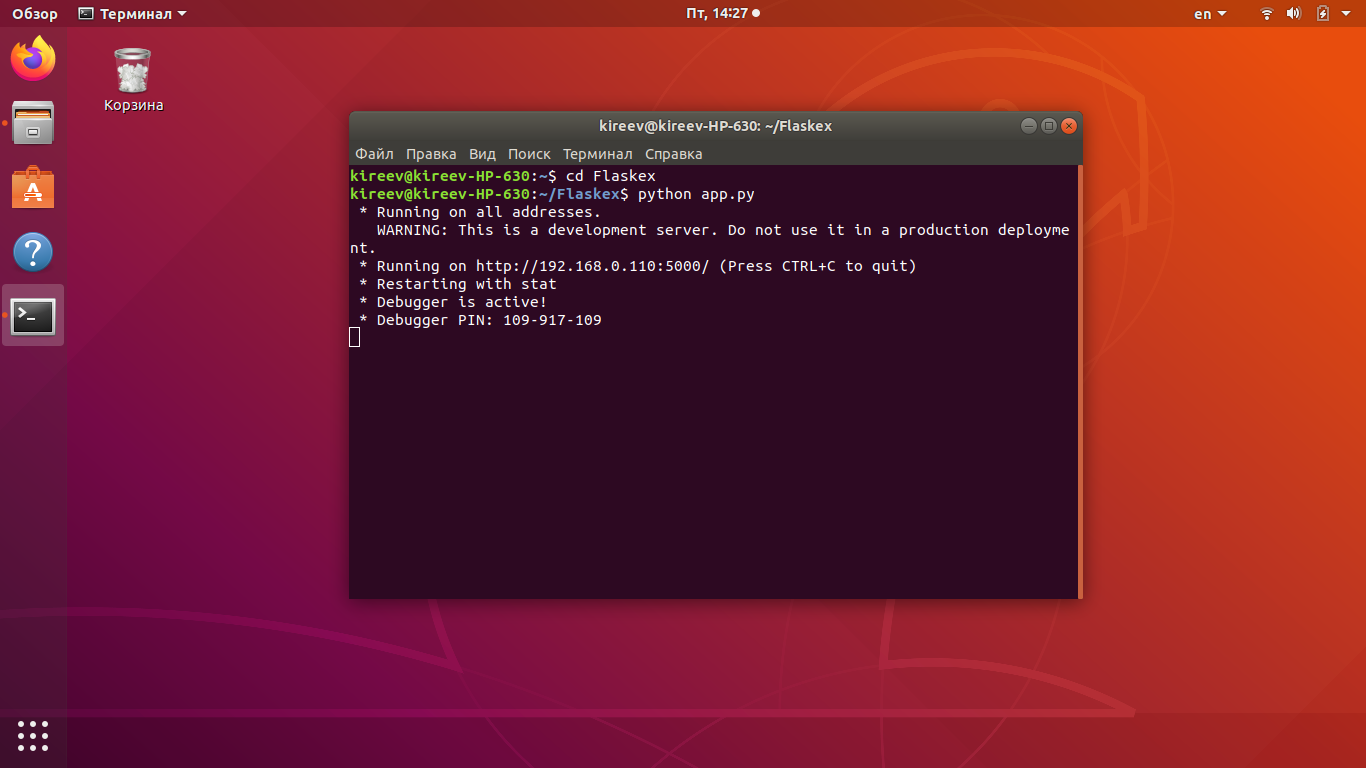
cd Flaskex

pip install -r requirements.txt

1. После сборки запускаем приложение

python app.py

Приложение работает. Проверим по адресу http;//192.168.0.110:5000/



## Упаковка приложения в docker

1. Устанавливаем docker (инструкция по ссылке)

<https://www.digitalocean.com/community/tutorials/docker-ubuntu-18-04-1-ru>

sudo apt update

sudo apt install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"

sudo apt update

apt-cache policy docker-ce

sudo apt install docker-ce

sudo systemctl status docker

1. Добавим пользователя в группу docker, чтобы работать без sudo

sudo usermod -aG docker kireev

1. Создаем dockerfile:

nano Dockerfile

1. FROM python:3.6.0 //указываем образ и версию python
2. COPY . /test  //скопируем в контейнер папку
3. WORKDIR /test //задаем рабочую директорию
4. RUN pip3 install --upgrade pip // обновляем pip3 до последней версии
5. RUN pip install -r requirements.txt //запускаем сборку приложения
6. RUN pip install Flask --upgrade // обновляем Flask до последней версии
7. RUN pip install Jinja2==3.0.0rc2 // Устанавливаем определенную версию шаблонизатора python для Flask ( для того, чтобы исключить ошибку Flask : 'Token' object has no attribute 'test' | render\_template error)
8. EXPOSE 5000 // указываем порт
9. CMD ["python", "app.py"] //запускаем приложение

Сохраняем

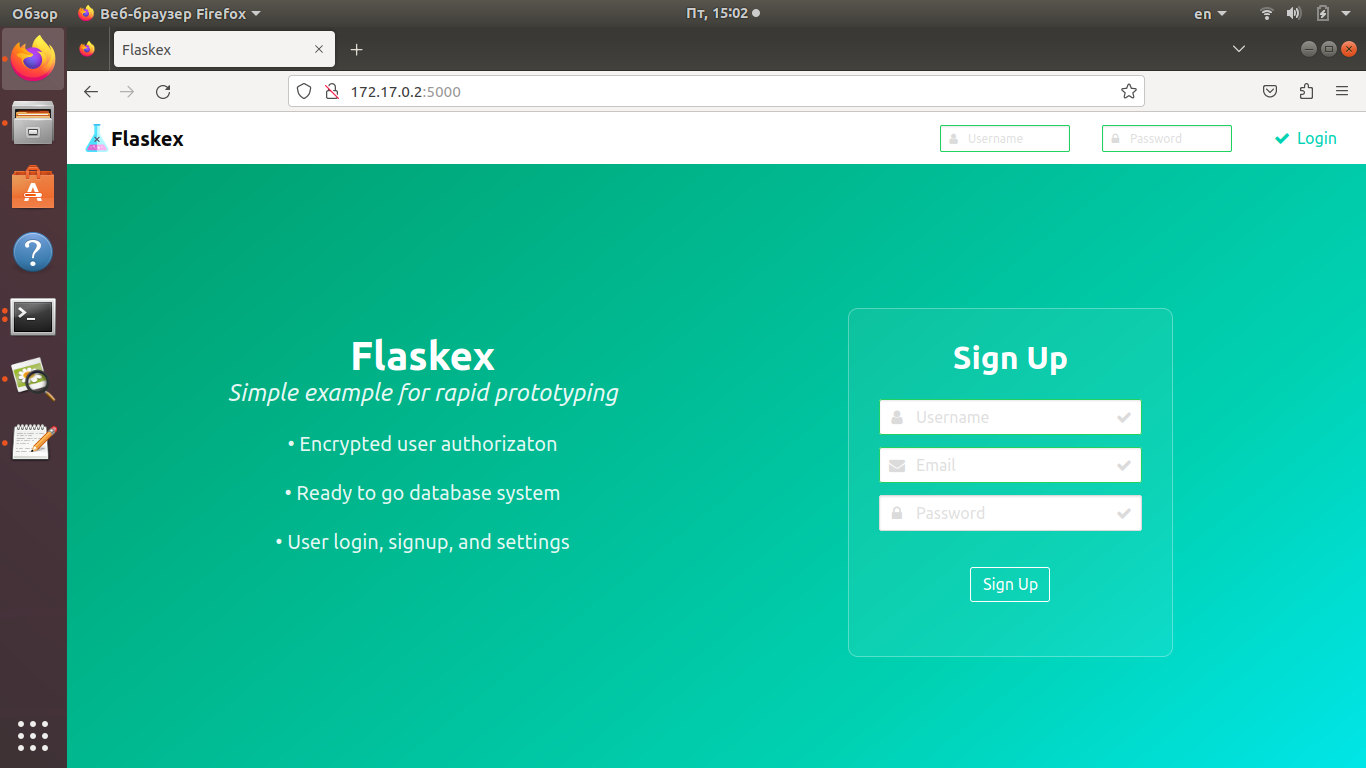
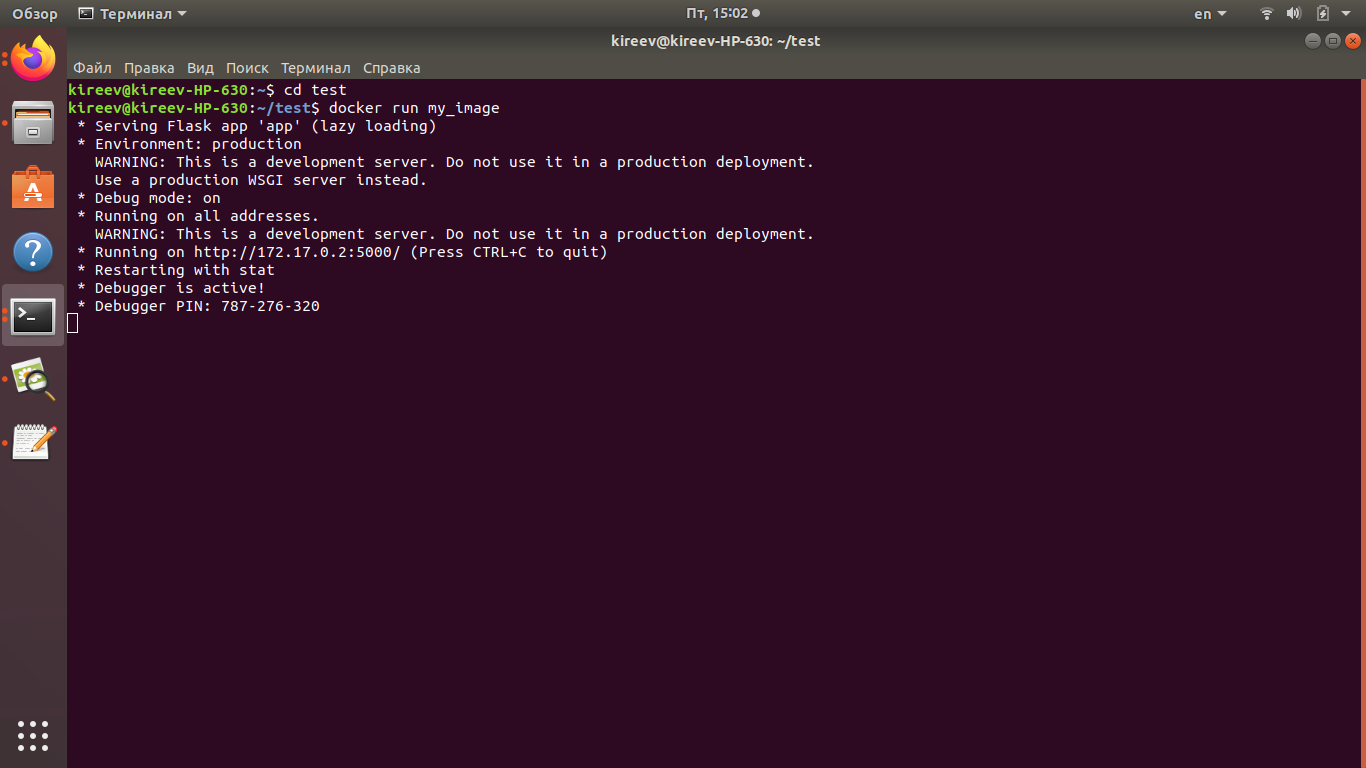
1. Создаем образ image с именем my\_image

docker build –t my\_image .

1. Запустим контейнер

docker run my\_image

1. Приложение работает. Проверим по адресу http;//192.168.0.110:5000/



## Запуск упакованного приложения через docker-compose.

1. Устанавливаем docker-compose (инструкция по ссылке)

<https://www.digitalocean.com/community/tutorials/how-to-install-docker-compose-on-ubuntu-18-04-ru>

sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

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

docker-compose –version

*Примечание: необходимо проверить версии docker и docker-compose на соответствие версий (таблицу соответствия можно найти по ссылкам ниже)*

* <https://sreeninet.wordpress.com/2017/03/28/comparing-docker-compose-versions/>
* <https://docs.docker.com/compose/compose-file/compose-file-v3/>

1. Создадим файл docker-compose.yml в нашей папке
2. version: "3.3" //указываем версию docker-compose
3. services:
4. web:
5. build: .
6. ports:
7. - "8000:5000"

Служба web использует образ, созданный из Dockerfile текущего каталога. Затем он привязывает контейнер и хост-компьютер к открытому порту 8000.

1. Запустим наше приложение

docker compose up

1. Приложение работает. Проверим по адресу <http://172.20.0.2:5000/>

