|  |  |  |  |
| --- | --- | --- | --- |
| **Студент:** | Лялюк | **Допуск:** |  |
| **Отчет по лабораторной работе №2**  «Установка python и запуск веб-сервера flask» | | | |
| Дата выполнения |  | Дата защиты |  |
| Оценка |  | Подпись |  |

Цель работы:

|  |
| --- |
| Получить навыки по установке и сборке из исходного кода пакетов в Linux, запуску веб-приложений flask |

Задачи работы:

|  |
| --- |
| 1. Установить и настроить последнюю версию Python 2. Установить модули и пакеты, flask, gunicorn, nginx 3. Развернуть приложение с авторизацией и БД 4. Создать виртуальное окружение 5. Сконфигурировать проксирование 6. Сделать запуск приложения автоматическим |

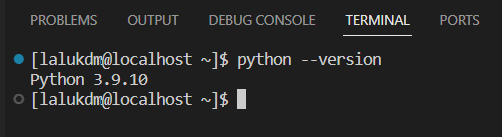
Краткий конспект теоретической части:

|  |
| --- |
| Python: высокоуровневый интерпретируемый язык программирования, ориентированный на производительность разработчика  Виртуальное окружение (venv): Изолированное пространство для отдельного проекта, позволяющее не засорять компьютер и не допускать конфликтов версий  Установка и сборка пакетов в Linux: может осуществляться с помощью пакетного менеджера pip (package installer for python)  Установка модулей Python: осуществляется как и установка пакетов: pip install <Название модуля>  Настройка фаерволла в Linux: фаерволл – межсетевой экран, это технологический барьер, защищающий сеть от нежелательного доступа посмотреть настройки можно: firewall-cmd –list-all после изменения постоянных файерволл необходимо перезагружать --reload  Запуск, остановка и диагностика сервисов в Linux: осуществление контроля над системными службами происходит через команду systemctl <действие> <имя сервиса>. Существуют следующие действия:  Запуск сервиса - start Перезапуск сервиса - restart Перезагрузка сервиса для обновления файлов - reload Проверка сервиса - status Остановка сервиса - stop |

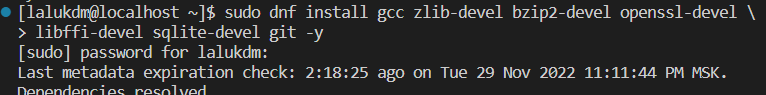
Результаты выполнения всех заданий следует наглядно подтверждать скриншотами. Все операции следует выполнять от учетной записи студента без прав администратора.

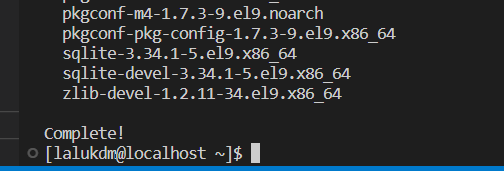
Задание 1. Загрузить исходный код последней стабильной версии Python. Продемонстрировать этапы сборки и установки. Показать версию установленного Python.

Текущая версия питона:

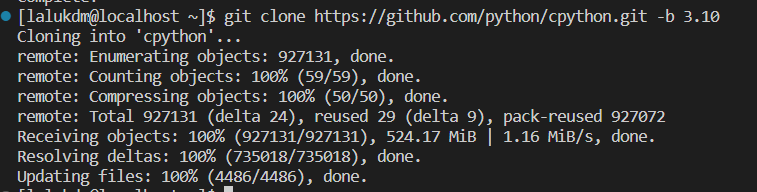


Установим вспомогательное ПО:





Клонируем репозиторий с питоном переключившись на стабильную ветку с помощью ключа –b:

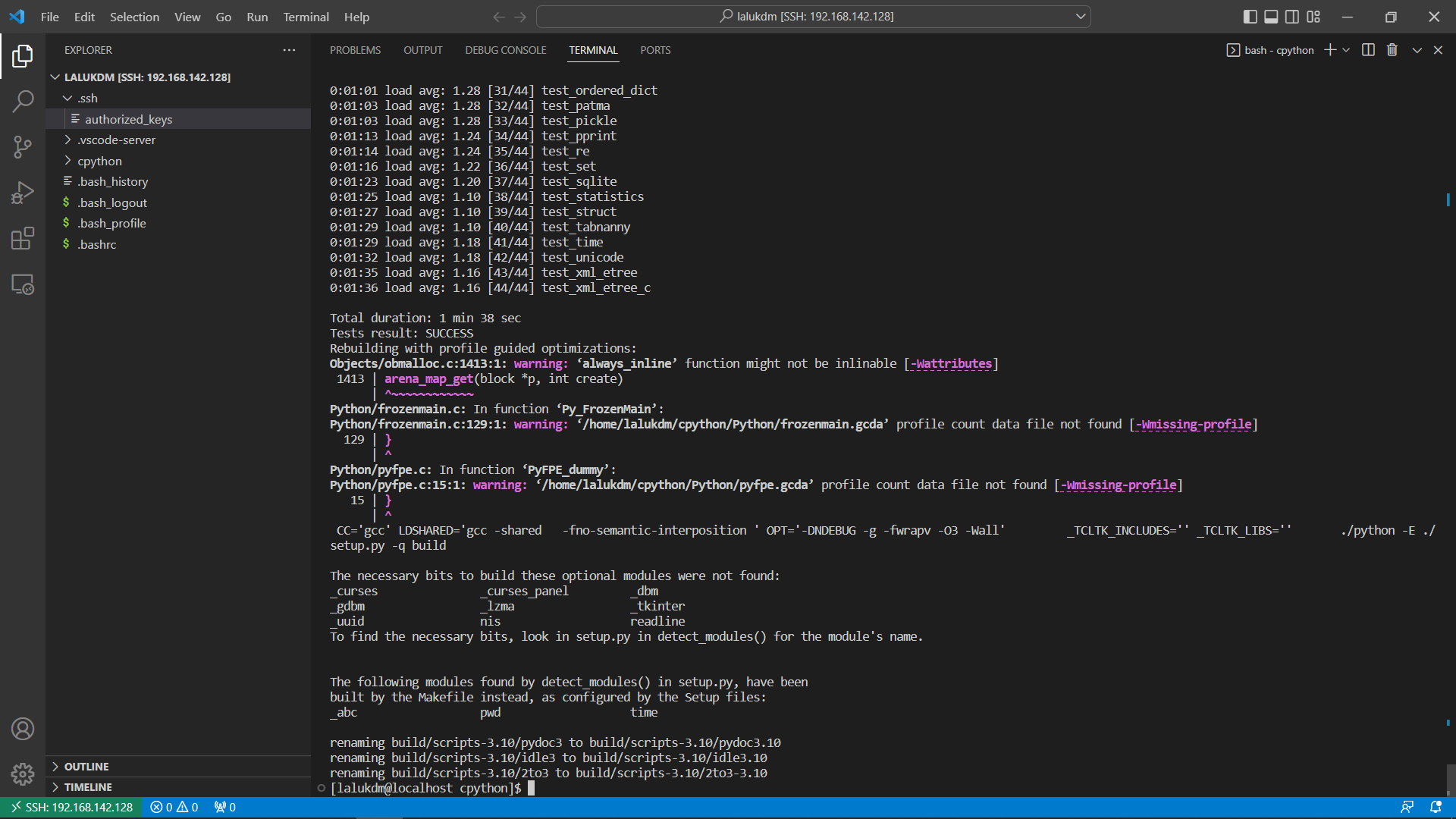


Из папки с проектом настраиваем конфигурацию:



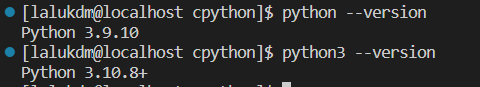
Собираем питон:





Получили сборку, доступную в текущей директории:



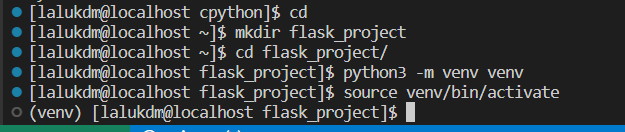


Также сделаем его доступным на системном уровне:

sudo make -s -j2 install

Задание 2. Установить необходимые модули и развернуть flask-приложение с функционалом авторизации. Для хранения пользователей использовать СУБД SQLite3.

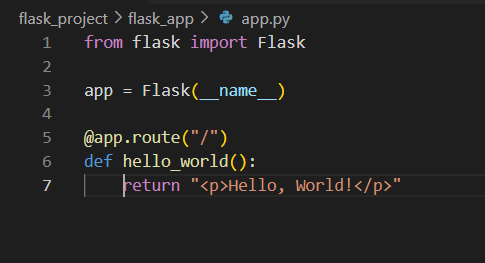
Создадим виртуальное окружение:

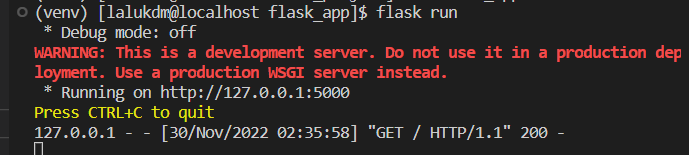


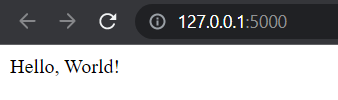
Загружаем flask в виртуальном окружении с помощью пакетного менеджера pip:



Создадим шаблонное приложение и запустим его:







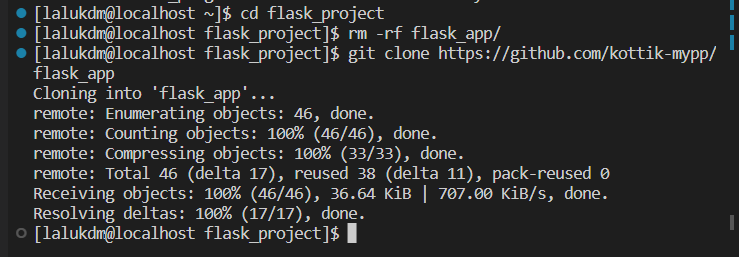
Открыв порт 5000 в настройках фаерволла командой:

sudo firewall-cmd –add-port=5000/tcp –permanent

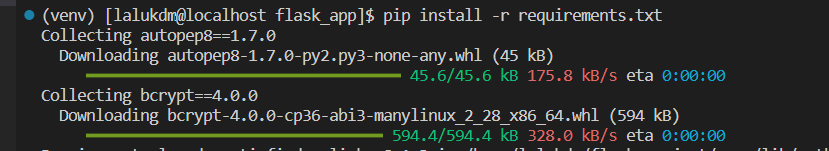
Можем лаского попросить фласк запуститься с использование ip-адреса ВМ:

flask run –host=0.0.0.0

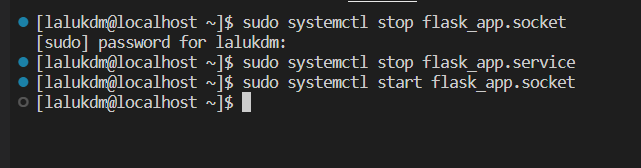
Развернем готовый проект с авторизацией:



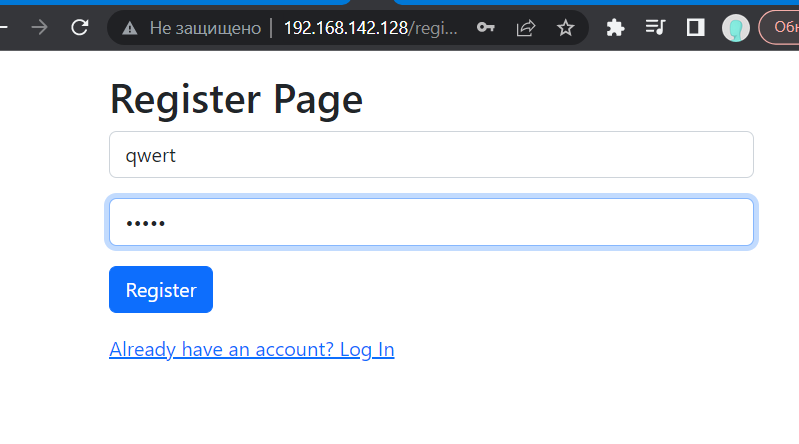
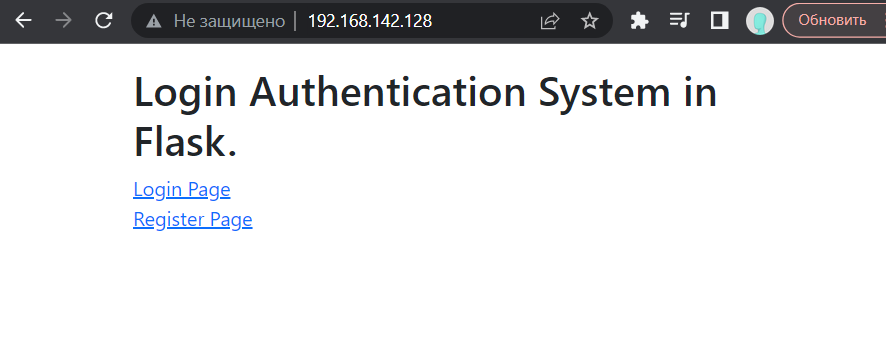
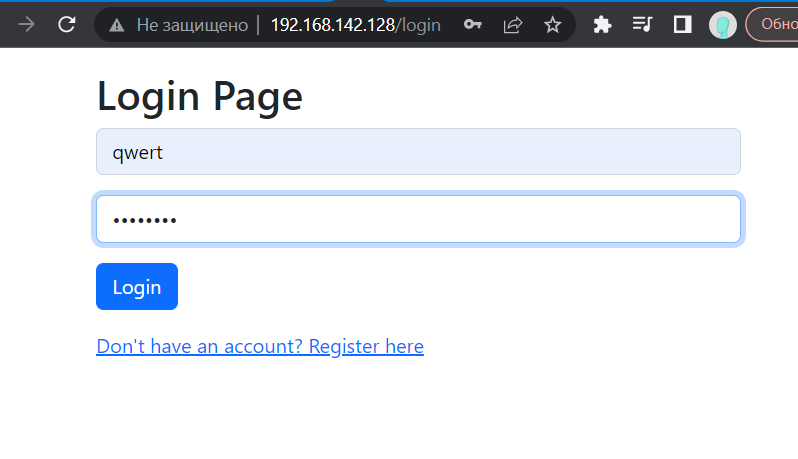
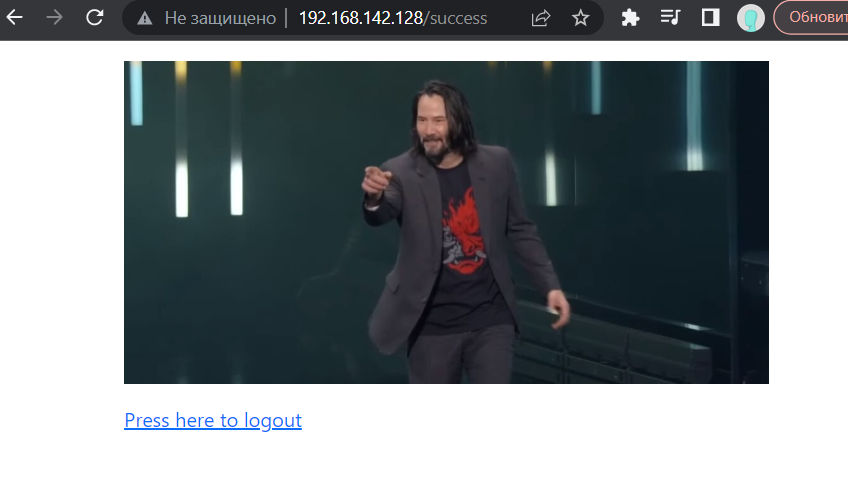
Установим требуемые модули и пакеты:



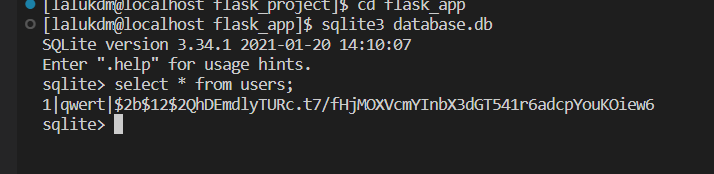
Перезапустим службу и сокет приложения:



Увидим приложение:

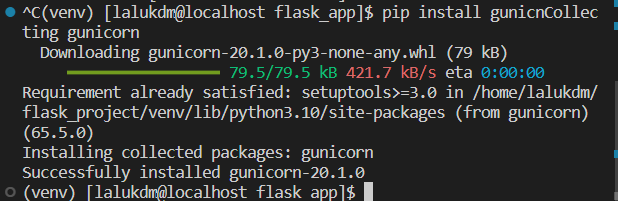
  

Также можем видеть содержимое таблицы БД (зарегистрированных пользователей):



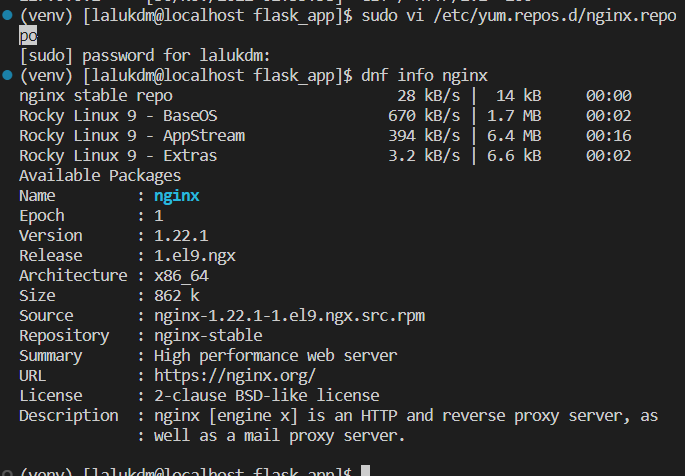
Задание 3. Установить необходимые модули для запуска flask-приложения при помощи gunicorn.

Установим gunicorn в виртуальное окружение:

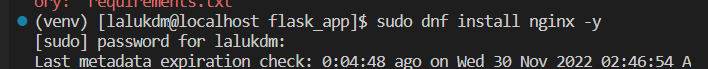


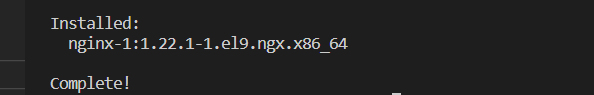
Задание 4. Установить nginx из репозитория разработчиков. Сконфигурировать его так, чтобы он проксировал http-запросы с порта 80 запущенному приложению.

Создадим новый файл репозитория через sudo пользуясь текстовым редактором vi:

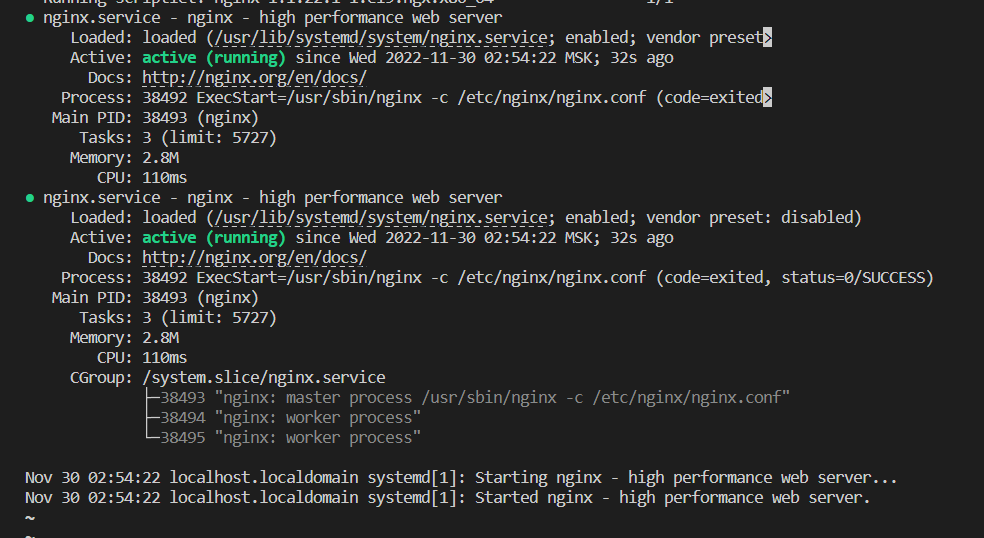


Устанавливаем nginx:

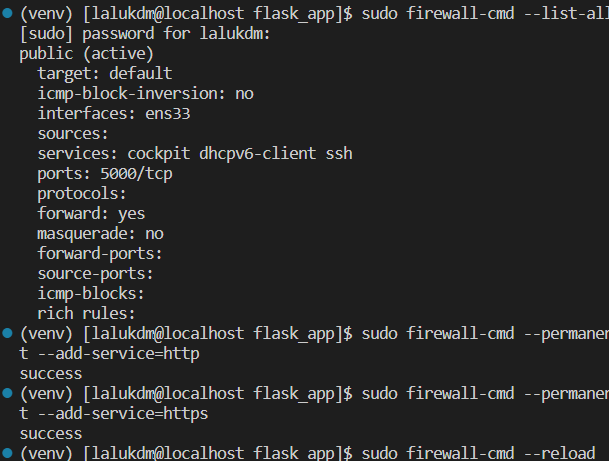




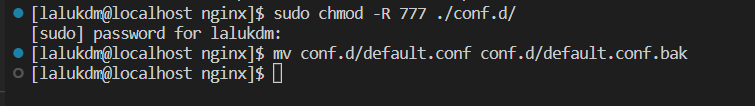
Удостоверимся в работе веб-сервера:



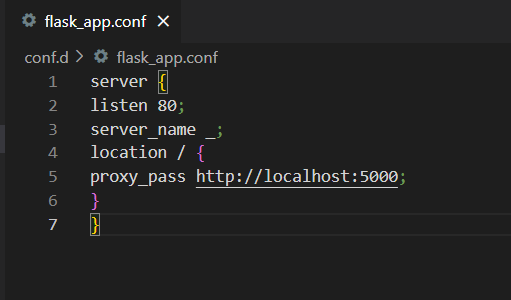
Добавим разрешающие правила для протоколов http и https:



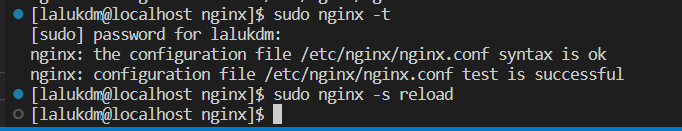
Получим права на изменение файлов каталога /etc/nginx/:



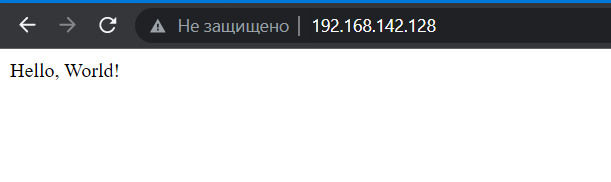
Создадим файл конфигурации проксирующий все запросы с 80-го на 5000-ый порт хоста:



И проверим что все правильно:

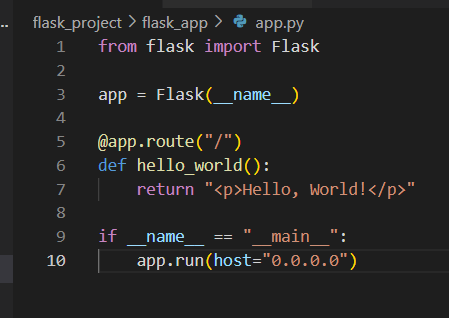


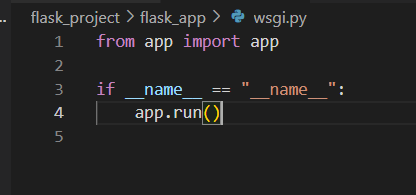
Запустив приложение увидим:



Задание 5. Настроить системные сервис и сокет для автоматического запуска gunicorn и nginx.

Создадим следующие приложения:

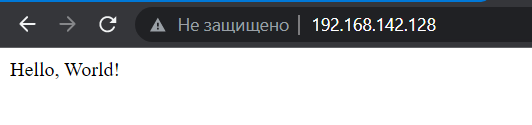




Запустим gunicorn, вызвав имя модуля(wsgi) и имя вызываемого объекта в этом модуле и указав порт для привязки, чтобы проверить его работу командой:

gunicorn --bind 0.0.0.0:5000 wsgi:app

И сможем видеть:



Для автоматического запуска создадим системный сервис:

sudo vi /etc/systemd/system/flask\_app.service

Следущего содержания:

[Unit]

Description=Gunicorn instance to serve flask\_app

Requires=flask\_app.socket

After=network.target

[Service]

Type=notify

User=lalukdm

Group=nginx

RuntimeDirectory=gunicorn

WorkingDirectory=/home/lalukdm/flask\_project/flask\_app

Environment="PATH=/home/lalukdm/flask\_project/venv/bin"

ExecStart=/home/lalukdm/flask\_project/venv/bin/gunicorn \

--access-logfile - \

--workers 3 \

--bind unix:/run/flask\_app.sock \

wsgi:app

[Install]

WantedBy=multi-user.target

А также файл системного сокета, через который общаются nginx и gunicorn:

sudo vi /etc/systemd/system/flask\_app.socket

Содержание:

[Unit]

Description=gunicorn socket

[Socket]

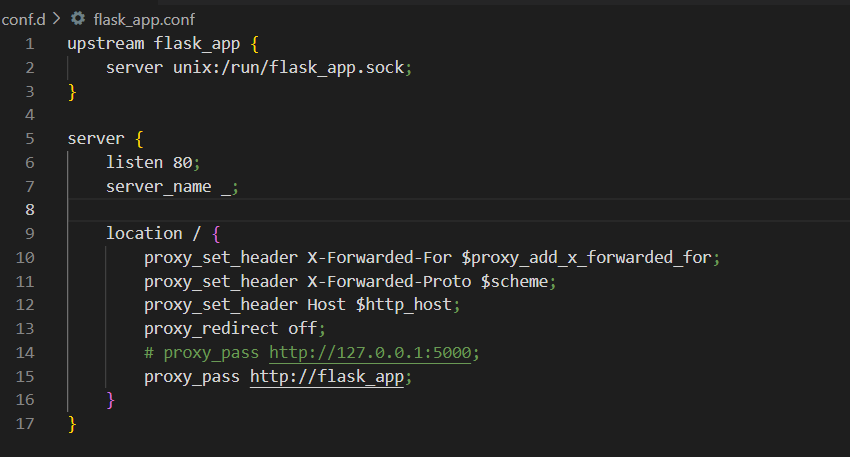
ListenStream=/run/flask\_app.sock

SocketUser=nginx

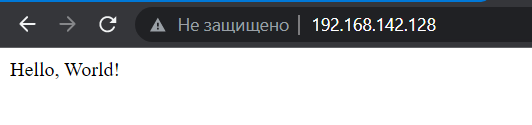
[Install]

WantedBy=sockets.target

А конфиг nginx сменим на:



Теперь приложение запускается автоматически:



Контрольные вопросы:

1. Что такое пакетный менеджер?

2. Почему возможен вызов программы по ее названию без указания полного пути к исполняемому файлу?

3. Какие порты называются открытыми и прослушиваемыми?

4. Что такое демон и сервис в контексте Linux?

5. Чем различаются файловый и сетевой сокеты?

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент:** |  | **Допуск:** |  |
| **Отчет по лабораторной работе №3**  «Выпуск TLS-сертификатов и работа с git-репозиториями» | | | |
| Дата выполнения |  | Дата защиты |  |
| Оценка |  | Подпись |  |

Цель работы:

|  |
| --- |
| Получить навыки по выпуску сертификатов, работе с git-репозиториями |

Задачи работы:

|  |
| --- |
|  |

Краткий конспект теоретической части:

|  |
| --- |
| HTTPS и TLS:  Инфраструктура PKI:  Алгоритм Диффи-Хеллмана:  Репозиторий:  Git:  GitHub: |

Результаты выполнения всех заданий следует наглядно подтверждать скриншотами. Все операции следует выполнять от учетной записи студента без прав администратора.

Задание 1. Создать базовую инфраструктуру удостоверяющего центра. В CN указать фамилию и имя студента. Сертификат созданного УЦ загрузить на хост и добавить в доверенные корневые.

Задание 2. Выпустить сертификат для веб-сервера. Настроить nginx на прослушивание порта 443. Настроить перенаправление с 80 порта на 443. Продемонстрировать его работоспособность.

Задание 3. Создать репозиторий с проектом с авторизацией. Настроить .gitignore так, чтобы файл базы данных не версионировался.

Задание 4. Внести изменение в файлы в репозитории, создать новый коммит, откатить проект к предыдущей версии.

Задание 5. Синхронизировать локальный репозиторий с удаленным на GitLab. Приложить ссылку на репозиторий.

Контрольные вопросы:

1. Что такое PKI?

2. В чем суть принципа forward secrecy?

3. Что такое репозиторий?

4. Каковы базовые команды работы с git-репозиторием?

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент:** |  | **Допуск:** |  |
| **Отчет по лабораторной работе №4**  «Инфраструктура в контейнерах» | | | |
| Дата выполнения |  | Дата защиты |  |
| Оценка |  | Подпись |  |

Цель работы:

|  |
| --- |
| Получить навыки по развертыванию приложений в контейнерах, |

Задачи работы:

|  |
| --- |
|  |

Краткий конспект теоретической части:

|  |
| --- |
| Контейнерная инфраструктура:  Docker:  Открытие портов в контейнерах:  Монтирование томов в контейнерах:  Docker Compose:  Взаимодействие сервисов в docker compose: |

Результаты выполнения всех заданий следует наглядно подтверждать скриншотами. Все операции следует выполнять от учетной записи студента без прав администратора.

Задание 1. Развернуть вторую виртуальную машину. На ней установить docker, docker compose. Запустить контейнер с nginx с кастомной веб-страничкой с текстом «Hello from <student\_name>»

Задание 2. Написать Dockerfile для сборки образа c системными пакетами (gcc g++ musl-dev), необходимыми для установки модулей проекта с авторизацией. Собрать кастомный образ по этому Dockerfile.

Задание 3. Написать файл docker-compose.yaml, в котором описать сервисы nginx и python. В контейнер сервиса python необходимо смонтировать директорию с приложением. Показать работу приложения по 80 порту, логи docker compose с отображением запросов.

Задание 4. Добавить в docker-compose.yaml сервис postgresql. Смонтировать контейнеру сервиса postgresql инициализационный скрипт, в котором создается пользователь, база и таблички. Продемонстрировать работоспособность и логи работы всех трех сервисов.

Задание 5. Настроить перенаправление запросов с первой виртуальной машины на вторую в контейнер с приложением. Показать работы приложения по https, логи с запросами на второй виртуальной машине.

Задание 6. Написать скрипт для автоматического развертывания всей инфраструктуры в контейнерах. Продемонстрировать работу скрипта. Приложить ссылки на репозитории с кодом инфраструктуры и кодом приложения.

Задание 7. Составить структурную схему развернутой на двух виртуальных машинах инфраструктуры.

Контрольные вопросы:

1. Что такое контейнер?

2. Что такое образ контейнера?

3. Как можно смонтировать файлы в контейнер?

4. Каковы преимущества применения контейнерной инфраструктуры?