Практика №3 Клиент-серверная архитектура ПО

Цель работы: развертывание собственного VDS в инфраструктуре НГТУ. Деплой веб-приложения.

Задачи:

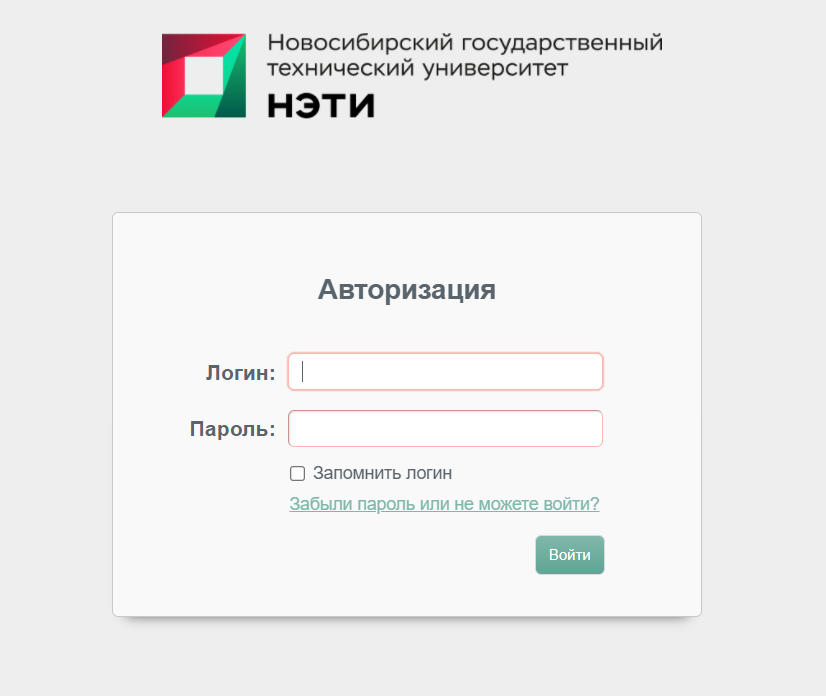
1. Авторизация на сайте cloud.nstu.ru;
2. Знакомство с устройством хостинга;
3. Создание виртуального сервера:
   * 1. Выбор ОС;
     2. Установка времени работы;
     3. Выбор кол-ва виртуальных ядер, оперативной памяти;
     4. Задание квоты дискового пространства.
4. Установка необходимых компонентов рабочего окружения и развертывание веб – сервиса:
   * 1. Собственный VPN сервер;
     2. Веб – сайт;
     3. Приложение.
5. Настройка внешнего доступа.

Варианты предметной области:

1. Образовательная среда
2. Интернет – магазин одежды
3. Онлайн заказ доставки
4. Заказ билетов в кино
5. Заказ документов с «МоихУслуг»
6. Вызов службы доверия
7. Военкомат
8. Заказ оборудования
9. Запись к врачу
10. Социальная сеть
11. Оплата онлайн услуг
12. Вызов мастера
13. Donatealerts
14. Заказ автозапчастей
15. Покупка мебели
16. Заказ электротостеров
17. Заказ гекконов
18. Пожарная служба
19. Чат внутри компании
20. Школьное образование
21. Переводчик
22. Конвертор эмоций

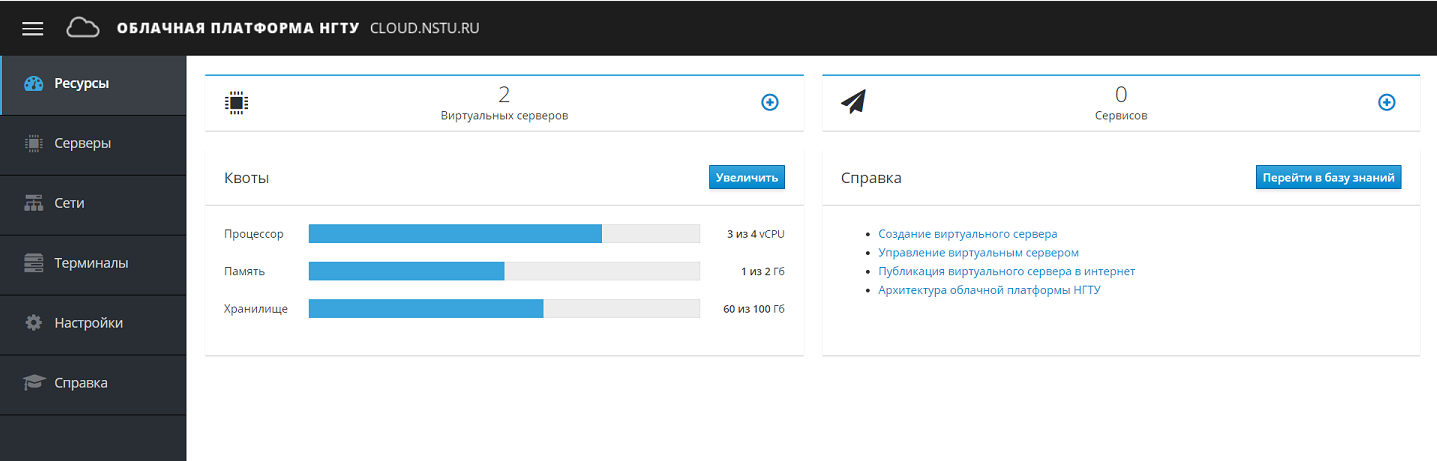
Реализация:

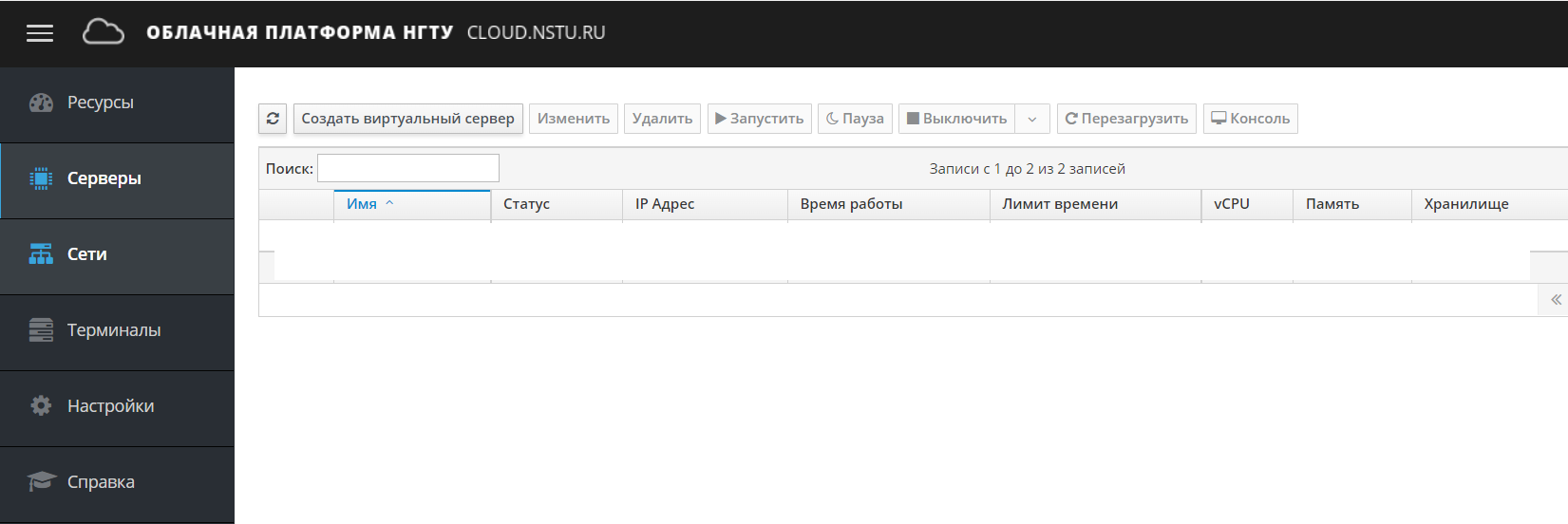
1. Авторизуемся на сайте cloud.nstu.ru, вводим данные от учетной записи ЛК НГТУ



1. Знакомство с устройством хостинга

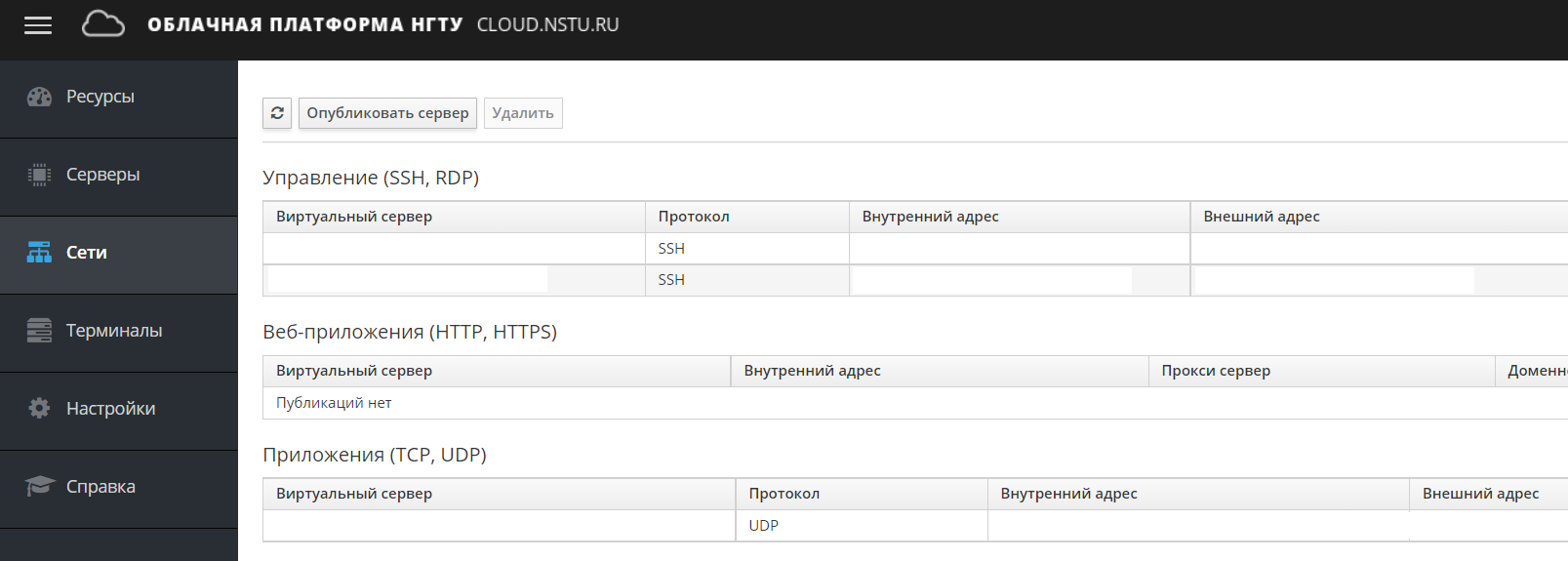
Открывается главная страница – «Ресурсы», на которой отображена потребляемая часть из доступной квоты ресурсов. Основными вкладками для работы являются: «Серверы», «Сети»





На странице «Серверы» отображена информация о состоянии серверов, производится управление и создание.

С вкладки «Сети» производится публикация серверов в общий доступ (SSH, RDP, HTTP\HTTPS, TCP, UDP)

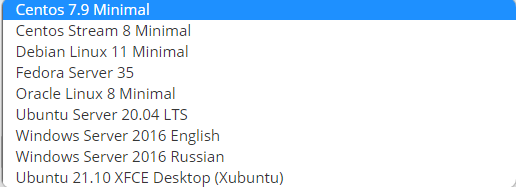


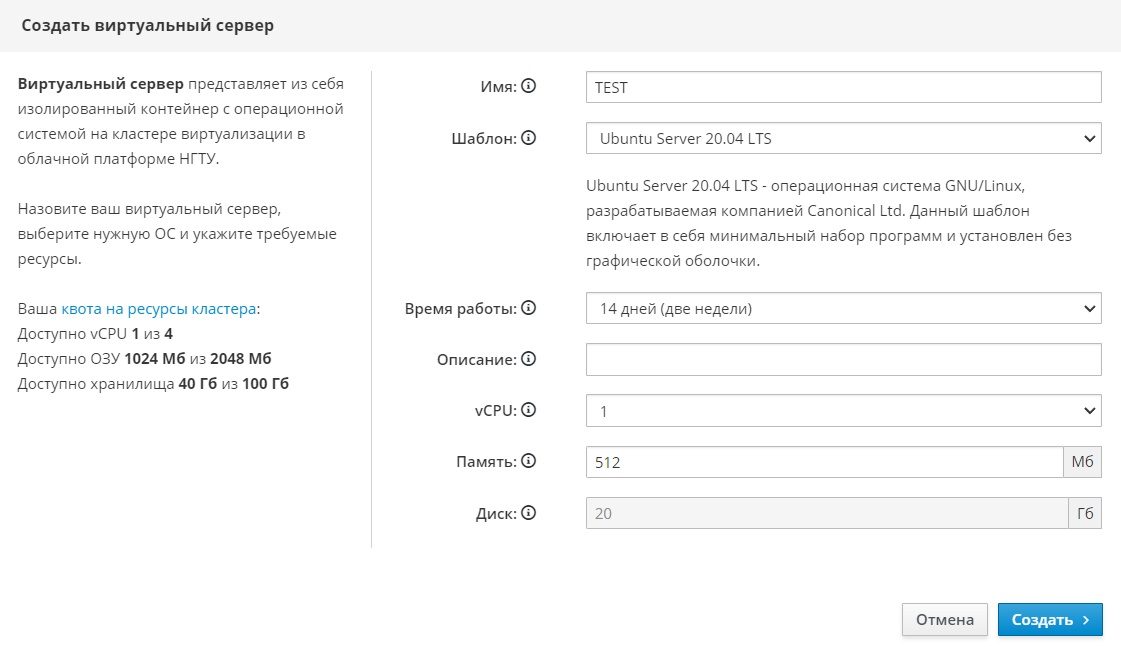
На странице «Терминалы» можно получить доступ к ПО, предоставляемое НГТУ посредством RDP подключение.

1. Создание виртуального сервера.

Во вкладке «Серверы» нажимаем на кнопку – «Создать виртуальный сервер»

Доступно достаточное количество образов ОС, предпочтительнее использовать Ubuntu. Либо GUI (Ubuntu 21.10 XFCE Desktop), либо серверную (Ubuntu Server 20.04 LTS). На ваше усмотрение. У Ubuntu наблюдается меньше проблем с пакетами, при установке в отличие от других GNU/Linux, но не возбраняется использование и других ОС, при этом дальнейшие инструкции могут различаться. Большая часть ОС – серверные версии, потребляют меньшее кол-во ресурсов, но требуют бОльших навыков работы с терминалом. Так, серверным версиям ОС, для ненагруженной, обычной работы требуется ~256 – 512 Мб оперативной памяти и минимум 1 vCPU. Версиям ОС с графической оболочкой требуется от 512 Мб оперативной памяти и минимум 1vCPU. Дисковое пространство, задано по умолчанию, 20 Гб должно хватить.

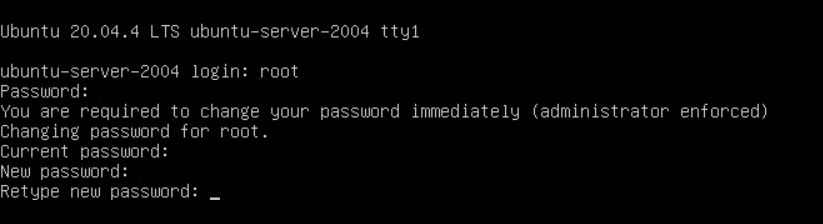


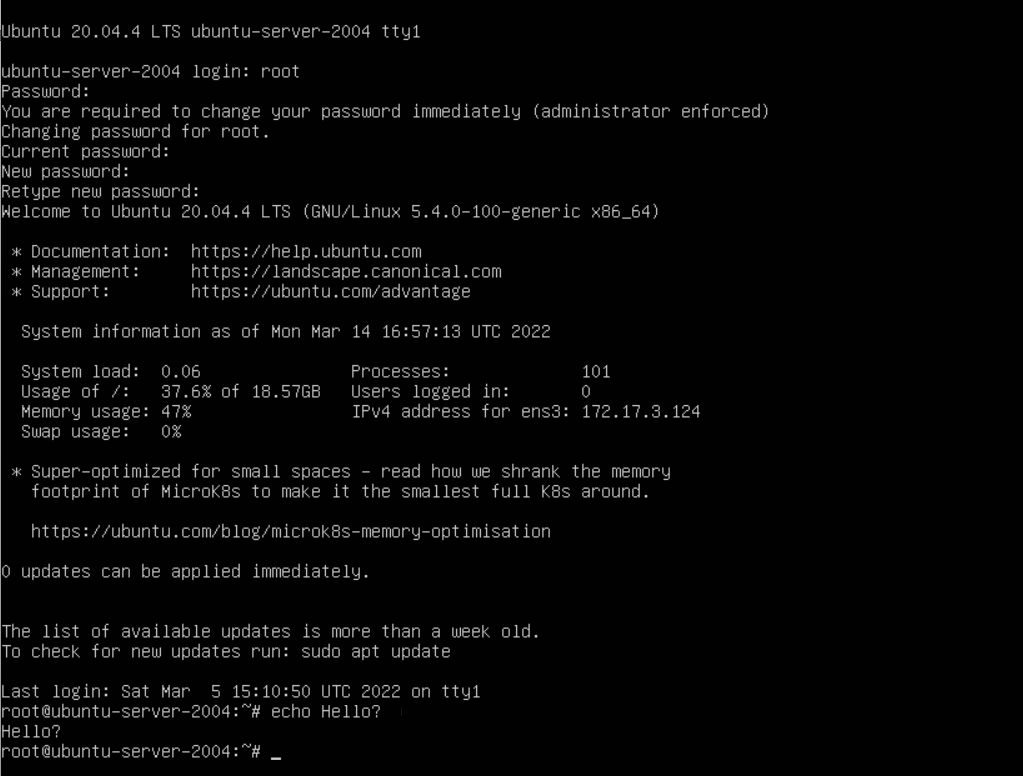


После завершения процессов создания виртуального сервера и его инициализации, нажимаем на кнопку «Запустить».

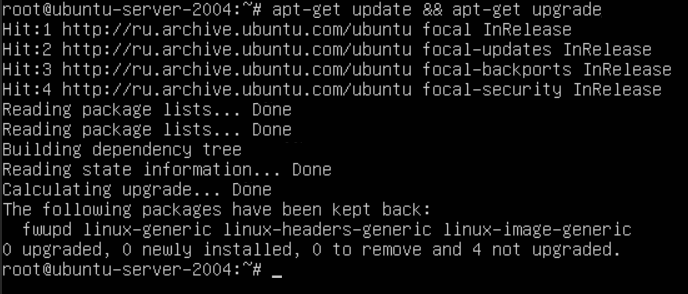
После успешного запуска виртуального сервера, нажимаем на кнопку «Консоль»

Откроется рабочее окружение сервера. Авторизуемся, система просит изменить пароль. Корневой пользователь для GNU/Linux систем – **root**. По умолчанию используется пароль для ВСЕХ систем: **Pa$$w0rd!**





Обновим хранилища пакетного менеджера apt: **apt-get update && apt-get upgrade**



Далее необходимо настроить SSH соединение и использовать его для подключения к серверу со своей машины напрямую (минуя сайт хостинга и т.п.), это более удобно и имеет некоторые преимущества. Ссылка на гайд по публикации SSH размещена в приложении. Для подключения по SSH рекомендуется использовать программу Termius или PuTTY, также можно воспользоваться консольной утилитой SSH. Подключение осуществляется командой:

**ssh -p ПОРТ** [**пользователь@ssh.cloud.nstu.ru**](mailto:пользователь@ssh.cloud.nstu.ru)

***ssh -p 6666*** [***root@ssh.cloud.nstu.ru***](mailto:root@ssh.cloud.nstu.ru)

1. Установка необходимых компонентов рабочего окружения и развертывание веб – сервиса

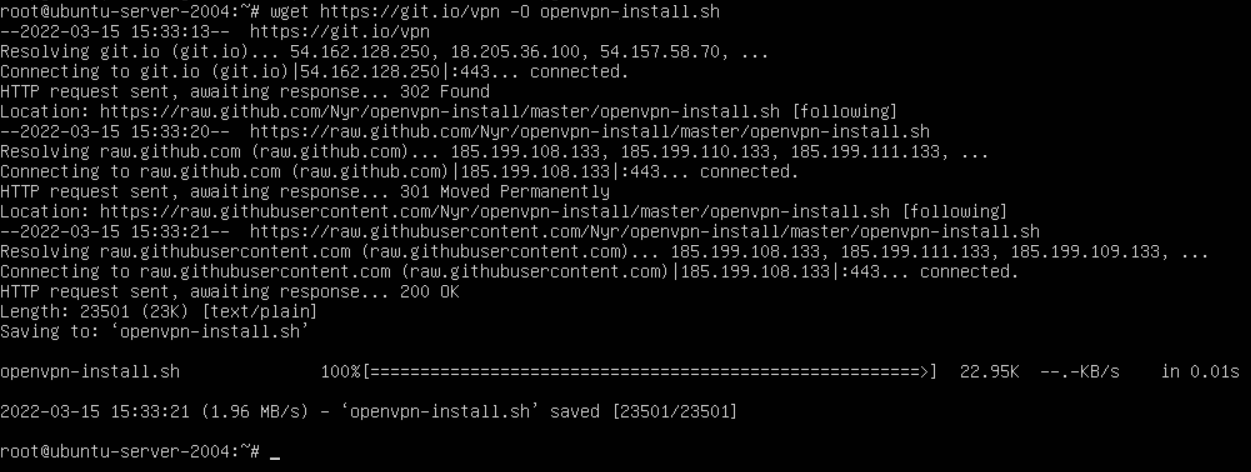
Необходимо в рамках работы необходимо реализовать пункты а - с:

Пункты b-c необходимо выполнить по предметной области, указанной в варианте.

* 1. Собственный VPN сервер

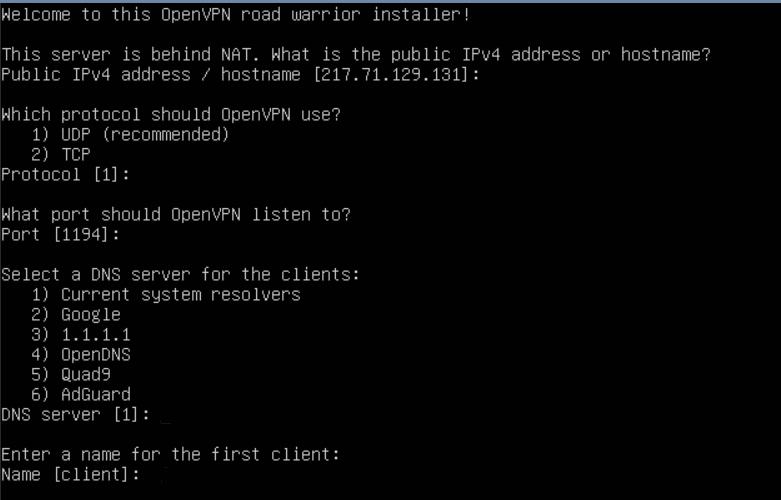
VPN актуальное средство в текущих реалиях, в ЛР показано развертывание VPN, используя реализацию технологии виртуальной частной сети OpenVPN с открытым исходным кодом. В дальнейшем рекомендуется не хранить и не использовать реализованную технологию на хостинге НГТУ. Реализация выполняется в учебных целях.

Сперва необходимо загрузить скрипт, который упрощает процедуру настройки и создания клиентов OpenVPN: **wget** [**https://git.io/vpn -O openvpn-install.sh**](https://git.io/vpn%20-O%20openvpn-install.sh)

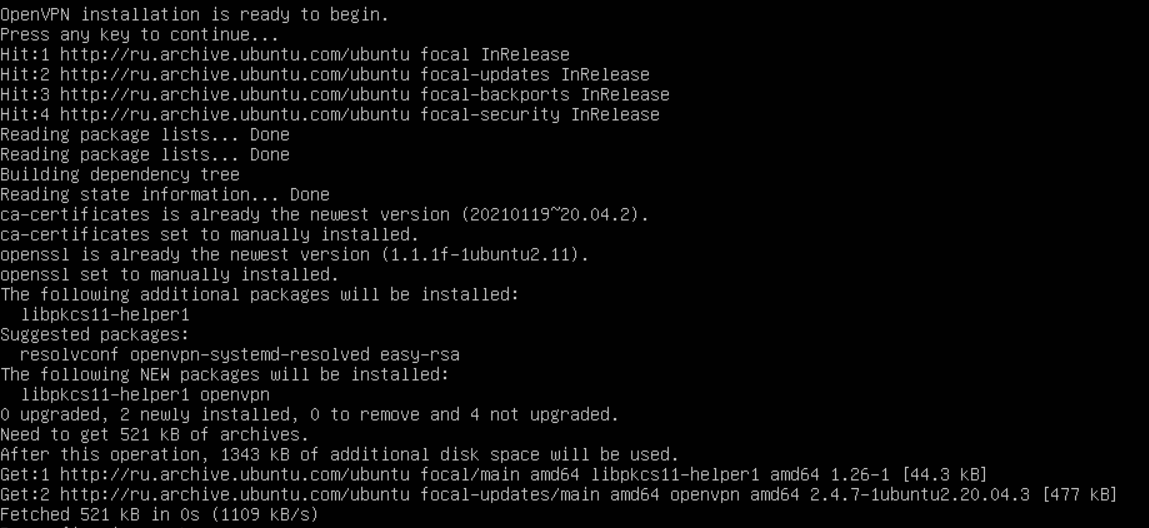
****Данный скрипт очень сильно упрощает жизнь, проделывая достаточную работу по развертыванию сервиса. (Кому интересно, предлагаю посмотреть какая примерно работа проделывается «под капотом» скрипта – <https://www.dmosk.ru/instruktions.php?object=openvpn-ubuntu-install>)

Затем, запускаем созданный скрипт: **sudo bash ./openvpn-install.sh**

Спокойно можем нажимать ENTER, нам подходят все значения по умолчанию. Экспериментирование не возбраняется 😊



Далее скрипт установить все зависимости, создаст сертификаты и еще много чего крутого.



Сохраняем клиентский конфигурационный файл (client.ovpn), который в дальнейшем будем использовать для авторизации клиента. На этом работа на сервере закончена.

На клиенте будь то телефон или компьютер, устанавливаем ПО OpenVPN GUI

ПК версия – <https://openvpn.net/community-downloads/>

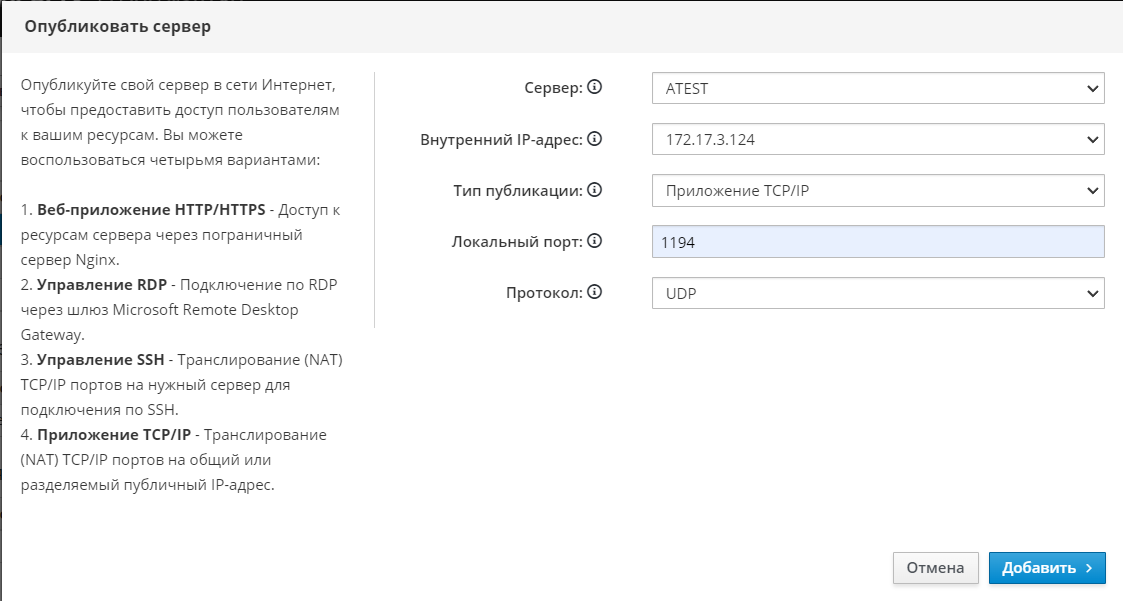
**Использоваться будет именно *OpenVPN GUI***

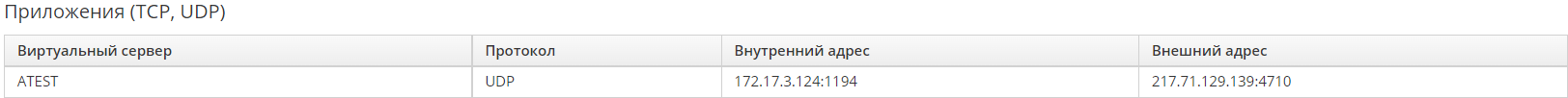
Далее нам необходимо скачать созданный конфигурационный файл на клиента, сделать это можно с помощью консольной утилиты scp, предварительно опубликовав сервер по SSH:

**scp -P ПОРТ** [**пользователь\_на\_сервере@ssh.cloud.nstu.ru:/ПУТЬ/К/ФАЙЛУ/НА/СЕРВЕРЕ**](mailto:пользователь_на_сервере@ssh.cloud.nstu.ru:/ПУТЬ/К/ФАЙЛУ/НА/СЕРВЕРЕ) **ПУТЬ\НА\ЛОКАЛЬНОЙ\МАШИНЕ**

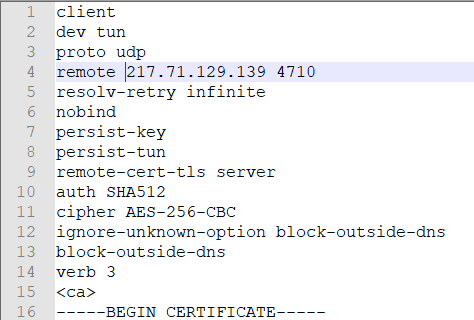
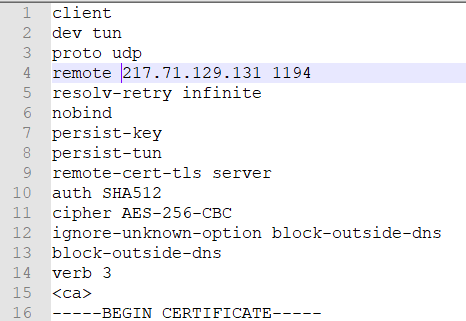
**scp -P 6666** [**root@ssh.cloud.nstu.ru:/root/client.ovpn D:\**](mailto:root@ssh.cloud.nstu.ru:/root/client.ovpn%20D:\)

После загрузки, также необходимо на сайте хостинга в разделе «Сети» опубликовать наш сервер по протоколу UDP (Тип публикации: Приложение TCP/IP; Протокол UDP; Порт 1194) (Ссылка на гайд в приложении)

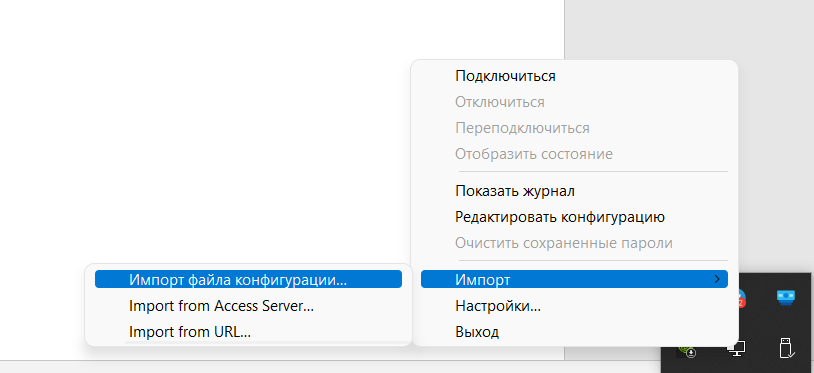




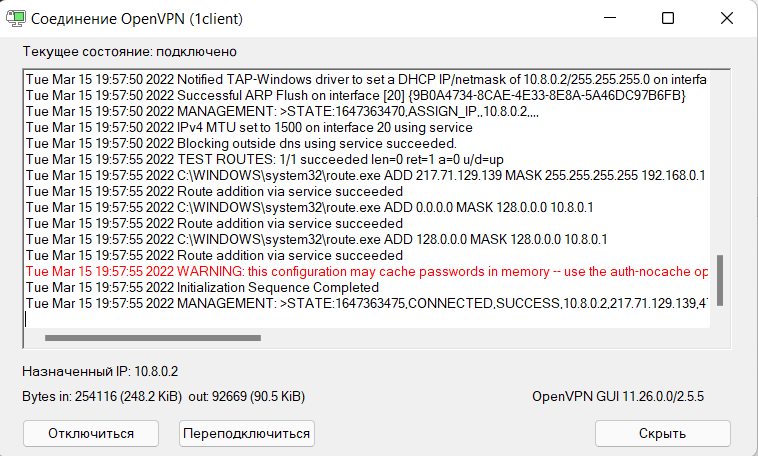
Затем необходимо открыть скачанный клиентский конфигурационный файл (client.ovpn) и заменить в нем IP PORT, на тот, который указан на сайте хостинга в разделе «Сети» -> «Внешний адрес», после публикации сервера.



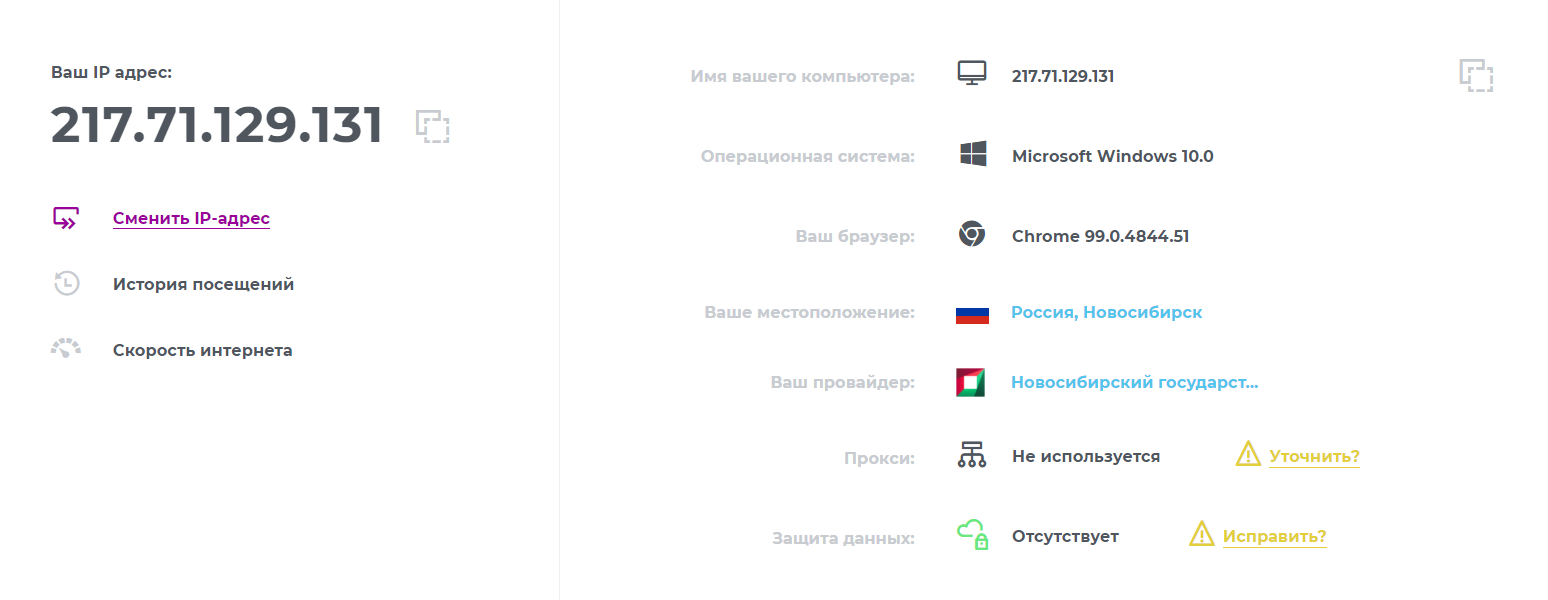
Сохраняем, закрываем, затем запускаем OpenVPN GUI, после запуска в оверлее появится значок приложения, нажимаем по нему правой кнопкой мыши – «Импорт файла конфигурации»



Импортируем client.ovpn, нажимаем подключиться – вуаля!



Наш виртуальный сетевой интерфейс имеет IP адрес в локальной сети – 10.8.0.2, таким образом мы оказались внутри локальной сети внешней инфраструктуры. Различные компании часто предоставляют возможность доступа во внутреннюю сеть из разных точек мира свои сотрудникам, посредством VPN. Внешний адрес также изменился



* 1. Веб – сайт

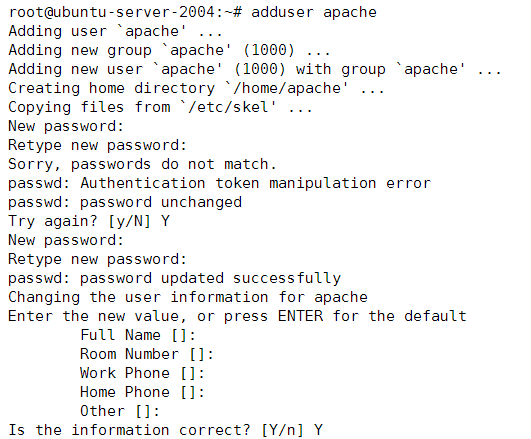
Для реализации собственного веб-сайта необходимо установить веб – сервер, программа которая по запросу будет «отдавать» пользователю нужные страницы.

HTTP-сервер Apache — самый широко используемый веб – сервер в мире. Он имеет множество мощных функций, включая динамически загружаемые модули, надежную поддержку различных форматов медиа и интеграцию с другим популярным программным обеспечением.

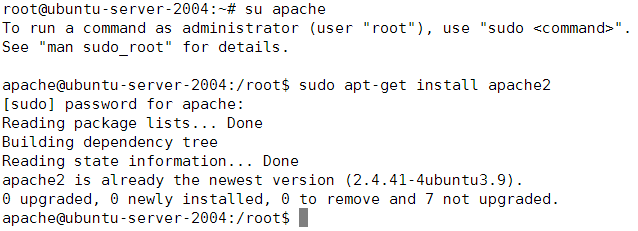
Перед началом работы, необходимо создать пользователя без прав root, но с доступом к команде sudo. (sudo — программа, дающая возможность системным администраторам позволять пользователям выполнять команды как root)

Создаем пользователя с помощью команды: **adduser имя**

Добавляем пользователя в группу sudo для того, чтобы он мог выполнять команды с повышенными привилегиями: **usermod -aG sudo имя**



Сменим пользователя (**su имя**) и установим пакет apache2: **sudo apt-get apache2**



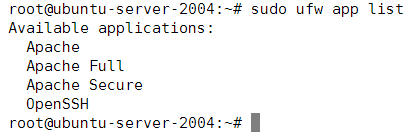
Настроим брандмауэр, для доступа к веб портам по умолчанию (HTTP 80, HTTPS 443)

Если вы выполнили предварительные указания, у вас должен быть установлен брандмауэр UFW, настроенный для ограничения доступа к вашему серверу.

Во время установки, Apache регистрируется в UFW, предоставляя несколько профилей приложений, которые можно использовать для включения или отключения доступа к Apache через брандмауэр.

Выведем список профилей приложений ufw, введя следующую команду: **sudo ufw app list**

Увидим данный список приложений

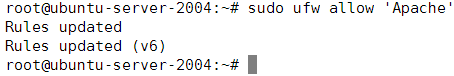


Как показал вывод, есть три профиля, доступных для Apache:

* Apache: этот профиль открывает только порт 80 (нормальный веб-трафик без шифрования)
* Apache Full: этот профиль открывает порт 80 (нормальный веб-трафик без шифрования) и порт 443 (трафик с шифрованием TLS/SSL)
* Apache Secure: этот профиль открывает только порт 443 (трафик с шифрованием TLS/SSL)

Применим самый ограничивающий профиль, который будет разрешать заданный трафик. Поскольку в этом модуле мы еще не настроили SSL для нашего сервера, нам нужно будет только разрешить трафик на порту 80 – протокол HTTP: **sudo ufw allow 'Apache'**

Также необходимо открыть 22 порт, по которому осуществляет SSH соединение, иначе мы не сможем подключаться вновь в будущем**: sudo ufw allow 22**

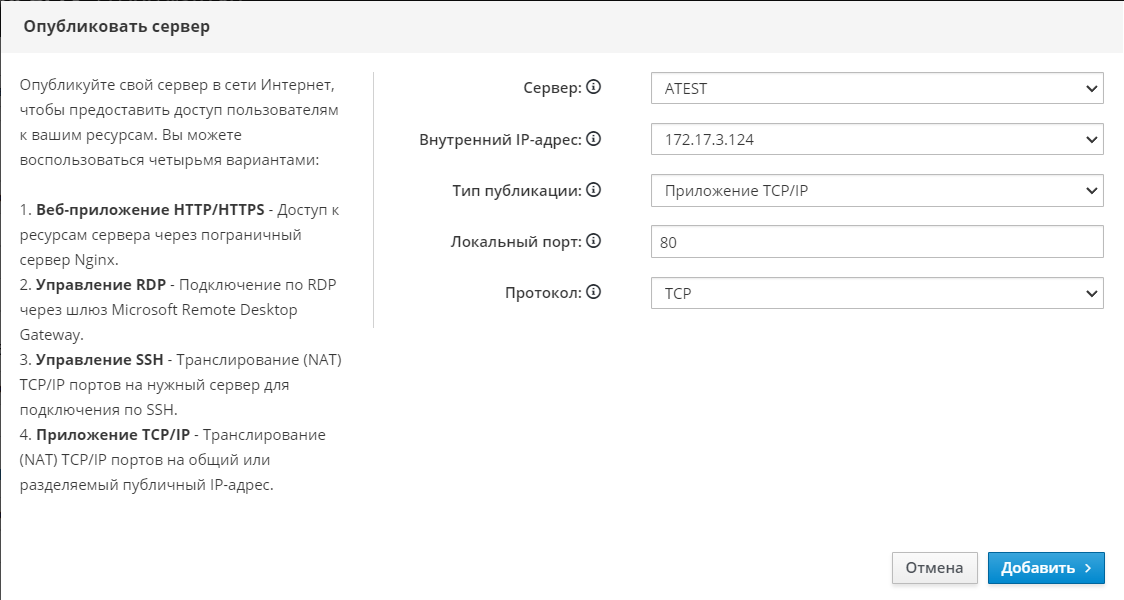
****

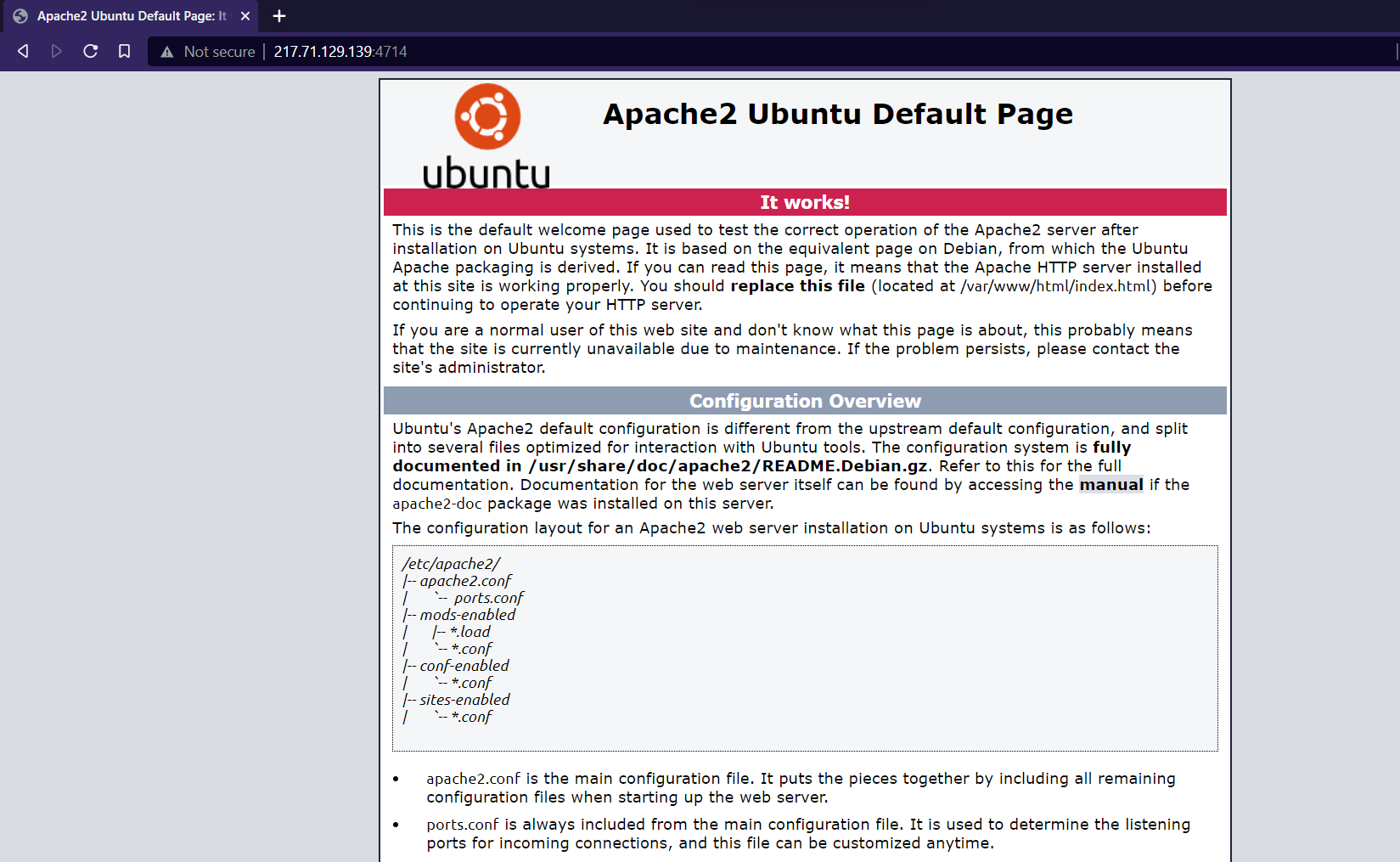
Если по умолчанию у вас не включен брандмауэр, его можно включить и проверить статус разрешенных приложений командой: **sudo ufw enable && ufw status**

Проверить статус веб – сервера можно командой: **sudo systemctl status apache2**

Если веб – сервер выключен, включить его можно командой: **sudo systemctl start apache2**

Остановить: **sudo systemctl stop apache2**

Теперь нужно опубликовать наш сервер на 80 порту, чтобы сайт был доступен из вне. Переходим на сайт хостинга НГТУ, раздел «Сайт» -> «Опубликовать сервер» (Ссылка на гайд в приложении)После публикации переходим с любого устройства по внешнему IP адресу. Видим приветственное окно Apache. Веб – сервер работает, осталось лишь добавить контент.



Перед добавлением контента, нужно изучить несколько основных и важных файлов и директорий Apache:

**Контент**

* */var/www/html*: веб-контент, в состав которого по умолчанию входит только показанная ранее страница Apache по умолчанию, выводится из каталога */var/www/html*. Это можно изменить путем изменения файлов конфигурации Apache.

**Конфигурация сервера**

* */etc/apache2*: каталог конфигурации Apache. Здесь хранятся все файлы конфигурации Apache.
* */etc/apache2/apache2conf*: главный файл конфигурации Apache. Его можно изменить для внесения изменений в глобальную конфигурацию Apache. Этот файл отвечает за загрузку многих других файлов в каталоге конфигурации.
* */etc/apache2/ports.conf*: этот файл задает порты, которые будет прослушивать Apache. По умолчанию Apache прослушивает порта 80, а если активирован модуль с функциями SSL, он также прослушивает порт 443.
* */etc/apache2/sites-available/*: каталог, где можно хранить виртуальные хосты для каждого сайта (когда на одной машине содержится несколько доменов, здесь же редактируется корневой каталог сайта). Apache не будет использовать файлы конфигурации из этого каталога, если они не будут связаны с каталогом sites-enabled. Обычно все изменения конфигурации серверных блоков выполняются в этом каталоге, а затем активируются посредством ссылки на другой каталог с помощью команды a2ensite.
* */etc/apache2/sites-enabled/*: каталог, где хранятся активные виртуальные хосты для каждого сайта. Обычно они создаются посредством создания ссылок на файлы конфигурации из каталога sites-available с помощью команды a2ensite. Apache считывает файлы конфигурации и ссылки из этого каталога при запуске или перезагрузке, когда компилируется полная конфигурация.
* */etc/apache2/conf-available/*, */etc/apache2/conf-enabled*/: эти каталоги имеют те же отношения, что и каталоги sites-available и sites-enabled, но используются для хранения фрагментов конфигурации, которые не принадлежат виртуальному хосту. Файлы из каталога conf-available можно активировать с помощью команды a2enconf и отключить с помощью команды a2disconf.
* */etc/apache2/mods-available/*, */etc/apache2/mods-enabled/*: эти каталоги содержат доступны и активированные модули соответственно. Файлы с расширением .load содержат фрагменты для загрузки определенных модулей, а файлы с расширением .conf содержат конфигурации этих модулей. Модули можно активировать и отключать с помощью команд a2enmod и a2dismod.

**Журналы сервера**

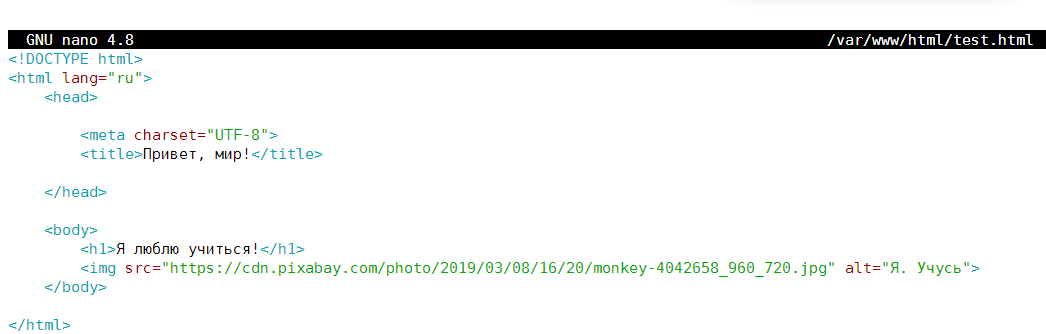
* */var/log/apache2/access.log*: по умолчанию каждый запрос веб-сервера регистрируется в этом файле журналда, если Apache не настроен по-другому.
* */var/log/apache2/error.log*: по умолчанию все ошибки регистрируются в этом файле. Директива LogLevel в конфигурации Apache указывает, насколько детальные записи регистрируются в журналах ошибок.

Основной контент для хоста по умолчанию хранится в рабочей сервера директории */var/www/html*, конфигурация этого хоста содержится в директории */etc/apache2/sites-available/000-default.conf*. В данном пособии будем редактировать контент сайта по умолчанию, дабы не путать информацией по созданию отдельных хостов, но опять-таки, экспериментирование не возбраняется 😊

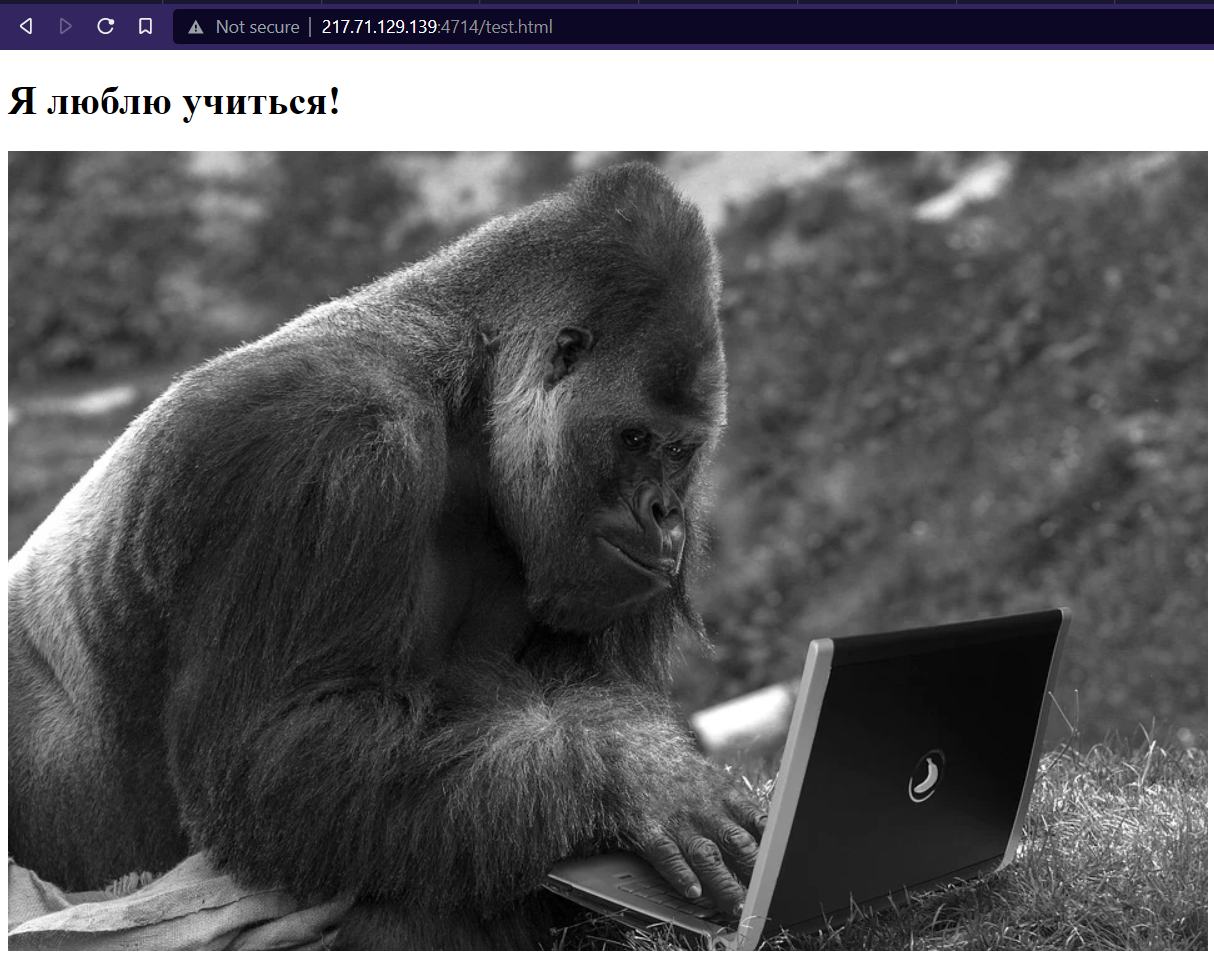
В директории с контентом *(/var/www/html*), можно создавать, изменять, удалять необходимые для функционирования вашего сайта файлы (HTML, CSS, JS и т.п.):

* Создайте страницу сайта*;*





* Переходим.



Далее вам необходимо реализовать свою версию веб – сайта, основанную на предметной области.

Полезные команды при работе с Apache:

* Включить сайт: **sudo a2ensite НАЗВАНИЕ.conf**

Станет актуальным конфиг в каталоге */etc/apache2/sites-available/* и переместиться в каталог актуальных */etc/apache2/sites-enabled/*

* Выключить сайт: **sudo a2dissite НАЗВАНИЕ.conf**
* Проверить ошибки конфигурации: **sudo apache2ctl configtest**
* Перезапустить Apache: **sudo systemctl restart apache2**
* Команды для загрузки файла на сервер (Сервер должен быть опубликован по SSH):

**scp -P ПОРТ C:\ПУТЬ ФАЙЛА\НА\ЛОКАЛЬНОЙ\МАШИНЕ** [**пользователь\_на\_сервере@ssh.cloud.nstu.ru:/ПУТЬ/НА/СЕРВЕРЕ**](mailto:пользователь_на_сервере@ssh.cloud.nstu.ru:/ПУТЬ/НА/СЕРВЕРЕ)

**scp -P 6666 D:\client.conf** [**root@ssh.cloud.nstu.ru:/root/**](mailto:root@ssh.cloud.nstu.ru:/root/%20)

С сервера:

**scp -P ПОРТ** [**пользователь@ssh.cloud.nstu.ru:/ПУТЬ/К/ФАЙЛУ/НА/СЕРВЕРЕ**](mailto:пользователь@ssh.cloud.nstu.ru:/ПУТЬ/К/ФАЙЛУ/НА/СЕРВЕРЕ) **ПУТЬ\НА\ЛОКАЛЬНОЙ\МАШИНЕ**

**scp -P 6666** [**root@ssh.cloud.nstu.ru:/root/client.conf**](mailto:root@ssh.cloud.nstu.ru:/root/client.conf%20)  **D:\**

Можно использовать параметр **-r**, чтобы выгрузить целую директорию

* Если вы хотите создать другой виртуальный хост, тогда необходимо создать директорию по пути */var/www/,* в которой будет храниться весь контент, создать конфигурационный файл по пути */etc/apache2/sites-avaible/* в который поместить примерно следующее:

<VirtualHost \*:80>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/your\_domain

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Перезапустить Apache

* 1. Приложение

Данное приложение представляет из себя некий код, взаимодействие с которым осуществляется посредством обращения к серверу. Идейно приложение представляет из себя эхо – скрипт, который отсылает обратно, пришедшие на сервер данные. Реализовано на Python, используя сокеты (https://digital2.ru/zametki-python-18-setevoe-programmirovanie/). Чтобы обеспечить работоспособность данного приложения необходимо:

* Установить все зависимости (ЯП, внешние зависимости)
* «Открыть» в брандмауэре необходимый порт
* Опубликовать сервер
* Запустить, убедиться в работоспособности
* По желанию можно реализовать демона, который позволяет работать некоторому приложению в фоновом режиме (<https://losst.ru/chto-takoe-demony-v-linux>)

Итак, мы разработаем и серверную, и клиентскую модель. Шаги можно рассматривать так:

* Программа сервера сокетов запускается сначала и ждет любого запроса.
* Клиентская программа инициирует диалог.
* Серверная программа реагирует на запросы клиента соответственно.
* Клиентская программа будет завершена, если пользователь введет сообщение «bye». Серверная программа также завершится, когда завершится клиентская программа, это необязательно, и мы можем поддерживать выполнение серверной программы на неопределенный срок или завершить работу с помощью какой-либо конкретной команды в клиентском запросе.

Серверная реализация выглядит так:

Язык python3 предустановлен на Ubuntu, но при его отсутствии, устанавливается как обычный пакет (**sudo apt install python3.9**). Компилятор gcc устанавливается командой: **sudo apt install gcc**

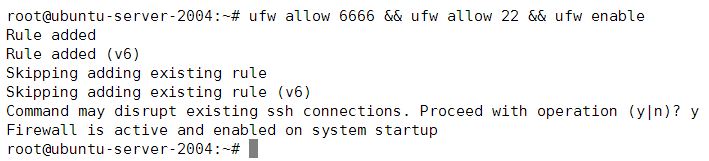
Данный пример показывает вам, как можно использовать сокеты для передачи данных с клиента на сервер. Реализованное является примером интеграции сокетов в ваше приложение.

Для начала необходимо установить все зависимости. Т.к. Python предустановлен и пакет socket входит в набор стандартных пакетов Python 3.8 – данный шаг опускается. Чаще всего все зависимости (ЯП, компиляторы, окружение) можно установить с помощью **sudo apt-get install**

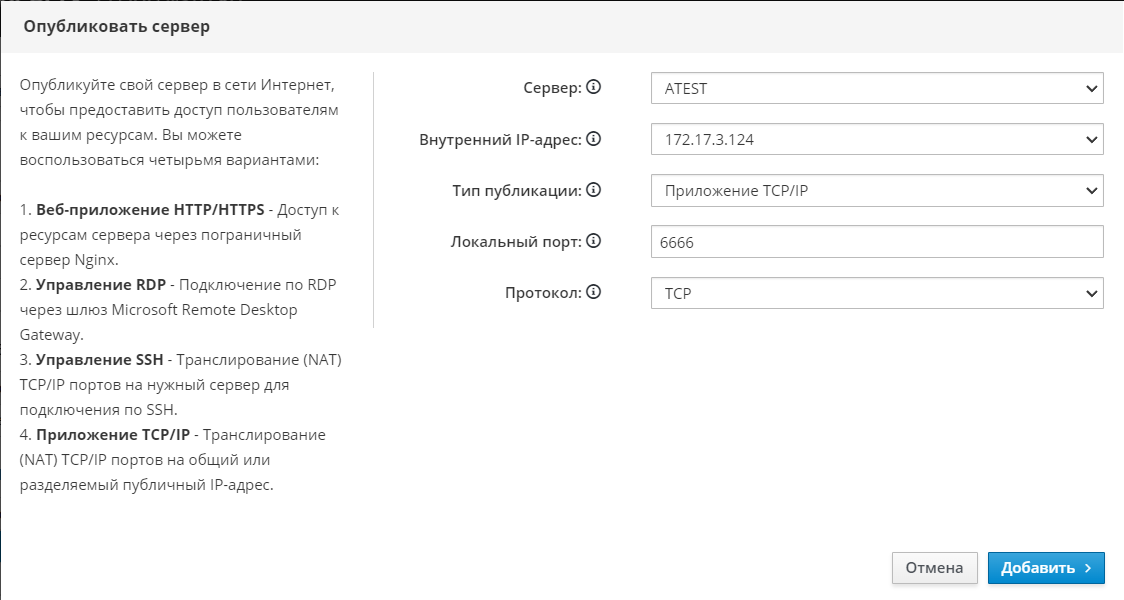
Затем необходимо разрешить необходимый порт в брандмауэре, который будет «слушать» сокет, сделать это можно командой: **sudo** **ufw allow ПОРТ**

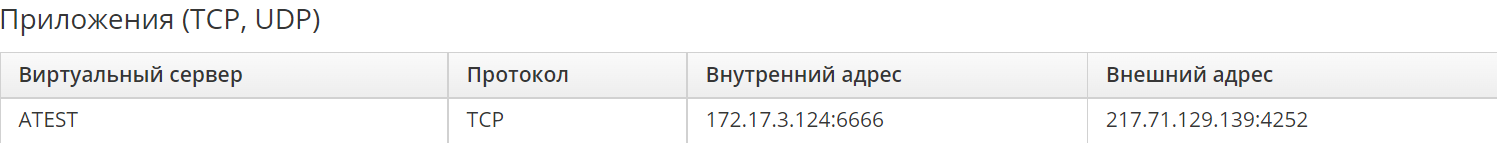
Также необходимо открыть 22 порт, по которому осуществляет SSH соединение, иначе мы не сможем подключаться вновь в будущем**: sudo ufw allow 22**

Если брандмауэр не включен, включаем его командой: **sudo ufw enable**



Необходимо на сайте хостинга в разделе «Сети» опубликовать наш сервер по протоколу TCP (Тип публикации: Приложение TCP/IP; Протокол TCP; Порт ЛЮБОЙ ВЫШЕ 1024) (Ссылка на гайд в приложении)





Для удобной выгрузки/загрузки исходного кода можно использовать утилиту scp:

Команды для загрузки файла на сервер (Сервер должен быть опубликован по SSH):

На сервер:

**scp -P ПОРТ C:\ПУТЬ ФАЙЛА\НА\ЛОКАЛЬНОЙ\МАШИНЕ** [**пользователь\_на\_сервере@ssh.cloud.nstu.ru:/ПУТЬ/НА/СЕРВЕРЕ**](mailto:пользователь_на_сервере@ssh.cloud.nstu.ru:/ПУТЬ/НА/СЕРВЕРЕ)

**scp -P 6666 D:\client.conf** [**root@ssh.cloud.nstu.ru:/root/**](mailto:root@ssh.cloud.nstu.ru:/root/%20)

С сервера:

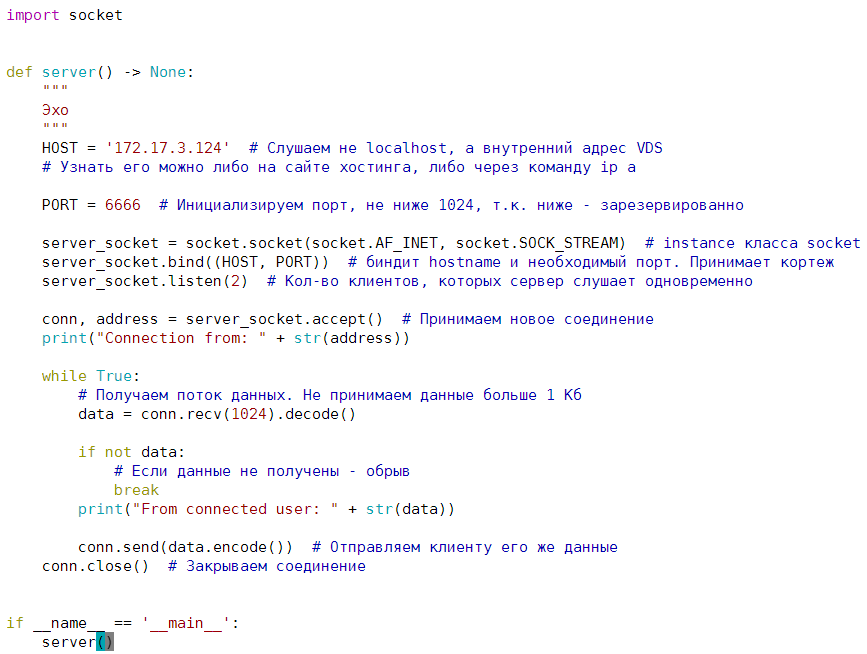
**scp -P ПОРТ** [**пользователь@ssh.cloud.nstu.ru:/ПУТЬ/К/ФАЙЛУ/НА/СЕРВЕРЕ**](mailto:пользователь@ssh.cloud.nstu.ru:/ПУТЬ/К/ФАЙЛУ/НА/СЕРВЕРЕ) **ПУТЬ\НА\ЛОКАЛЬНОЙ\МАШИНЕ**

**scp -P 6666** [**root@ssh.cloud.nstu.ru:/root/client.conf D:\**](mailto:root@ssh.cloud.nstu.ru:/root/client.conf%20%20D:\)

Можно использовать параметр **-r**, чтобы выгрузить целую директорию

Исходный код серверной и клиентской части:

Реализация серверной части server.py:



Реализация клиентской части client.py:



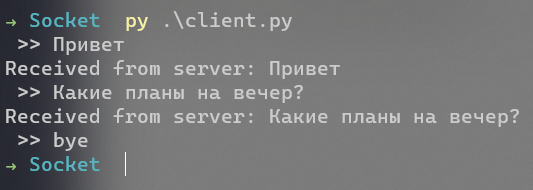
Из – за того, что наш сервер виртуальный (То есть выделены некоторые вычислительные ресурсы из кластера), он находится на определенном порту гипервизора (Программа или аппаратная схема, обеспечивающая или позволяющая одновременное, параллельное выполнение нескольких операционных систем на одном и том же хост-компьютере.), поэтому при подключении к сокету с клиента мы обращаемся на внешний адрес и порт гипервизора (PORT, HOST), который в свою очередь перенаправляет запрос на нужный порт нашего сервера. Именно поэтому мы указываем внешний адрес в client.py, также хочется обратить внимание, что в server.py поле HOST содержит внутренний адрес сервера, а не localhost, иначе сокет будет слушать не то сетевое устройство.

Запуск сервера, командой: **sudo python3 server.py**

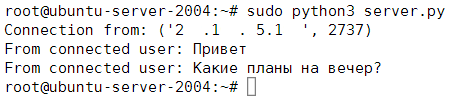
Запуск клиента на другой машине командой: **py client.py**

Продемонстрировано взаимодействие, отправляем данные с клиента, сервер посылает нам эхо ответ

Клиент:



Сервер:



На данном принципе строится «общение» между клиентом и сервером, клиентское приложение отправляет данные на «обработку», сервер, проделав работу отдаёт ответ. Серверное приложение можно реализовать в виде демона, который следить за процессом, перезапускает в случае неполадок, работает в фоновом режиме. Также активный процесс не будет занимать терминал.

Листинг в приложении

ПРИЛОЖЕНИЕ

Тот самый легендарный гайд по публикации сервера – <https://cloud.nstu.ru/wiki/публикация_виртуального_сервераы>

Листинг client.py:

import socket

def client() -> None:

"""

Эхо

"""

HOST = '217.71.129.139' # Внешний IP PORT сервера, в нашем случае гипервизора

# который перенаправляет запрос на нужный порт нашего сервера

PORT = 4252

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # Создаем instance класса

client\_socket.connect((HOST, PORT)) # Подключаемся к серверу

message = input(" >> ") # Вводим данные

while message.lower().strip() != 'bye':

client\_socket.send(message.encode()) # Отправялем данные

data = client\_socket.recv(1024).decode() # Получаем ответ

print('Received from server: ' + data)

message = input(" >> ") # Повтор ввода

client\_socket.close() # закрываем соединение

if \_\_name\_\_ == '\_\_main\_\_':

client()

Листинг server.py:

import socket

def server() -> None:

"""

Эхо

"""

HOST = '172.17.3.124' # Слушаем не localhost, а внутренний адрес VDS

# Узнать его можно либо на сайте хостинга, либо через команду ip a

PORT = 6666 # Инициализируем порт, не ниже 1024, т.к. ниже - зарезервированно

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) # instance класса socket

server\_socket.bind((HOST, PORT)) # биндит hostname и необходимый порт. Принимает кортеж

server\_socket.listen(2) # Кол-во клиентов, которых сервер одновременно слушает

conn, address = server\_socket.accept() # Принимаем новое соединение

print("Connection from: " + str(address))

while True:

# Получаем поток данных. Не принимаем данные больше 1 Кб

data = conn.recv(1024).decode()

if not data:

# Если данные не получены - обрыв

break

print("From connected user: " + str(data))

conn.send(data.encode()) # Отправляем клиенту его же данные

conn.close() # Закрываем соединение

if \_\_name\_\_ == '\_\_main\_\_':

server()