**Практическая работа №8**

**Установка и настройка Web-сервера nginx на базе Ubuntu-сервер**

**Цель:**

1. **Настроить сеть**

В терминале пишем команду «**nano /etc/netplan/01-netcfg.yaml**» 🡪 в открывшемся редакторе задаём следующие значения:

**IP:**192.168.223.107

**Маска:**255.255.252.0

**Шлюз:**192.168.220.1

**DNS-сервер:**192.168.220.2

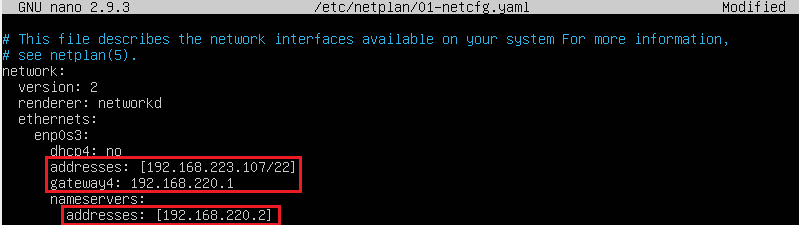
Пояснение:

**addresses** – задаёт IP-адрес.

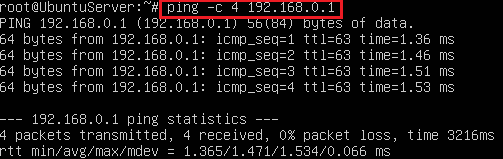
**gateway4** – задаёт шлюз (на данный момент указывается через **routes**)

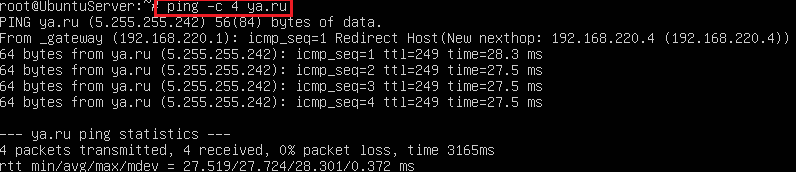
**addresses** – (второй по счёту) задаёт DNS сервер





Проверим подключение с помощью «**ping**» используем ключ «**-c**» для того, чтобы он не шёл бесконечно.





1. **Установить web-сервер nginx**

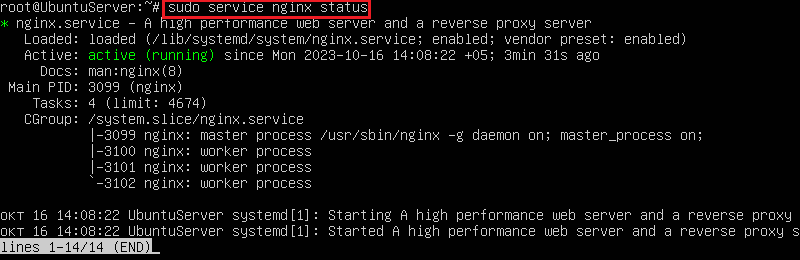
Сначала обновим список пакетов в репозитории, для этого воспользуемся командой «**sudo apt update**» 🡪 дальше установим Nginx, пишем команду «**sudo apt install nginx**» и ждём окончания установки 🡪 теперь, после установки, добавим его в автозагрузку «**sudo systemctl enable nginx**» 🡪 проверим его статус «**sudo service nginx status**»







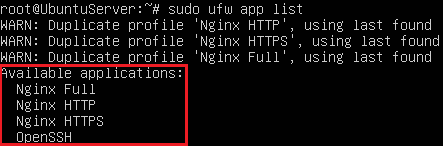




**Настроим брандмауэр**

Первым делом проверим список доступных приложений с помощью команды «**sudo ufw app list**» 🡪 теперь нужно запустит брандмауэр и разрешить передачу по портам «**Nginx Full**» и «**OpenSSH**», для этого выполни следующие команды поочерёдно «**sudo ufw enable**», «**sudo ufw allow 'Nginx Full'**», «**sudo ufw allow 'OpenSSH'**» 🡪 теперь проверим наши изменения и пропишем команду «**sudo ufw status**»

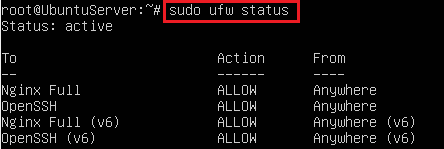


!!!!!!





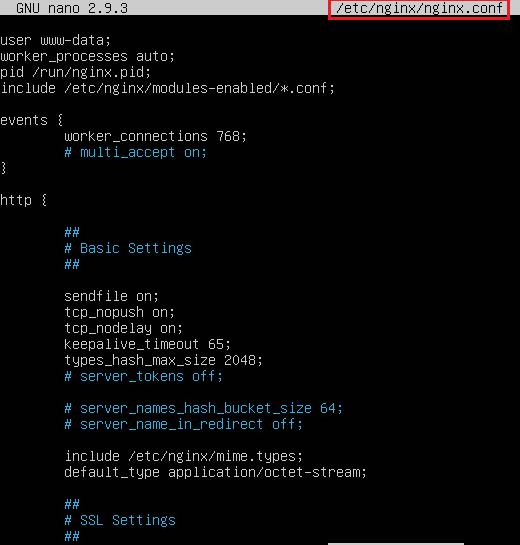




**Настройка Nginx**

Изучим первым делом главный файл конфигурации, для этого введём команду «**sudo nano /etc/nginx/nginx.conf**»





Небольшое объяснение файла:

* user — это пользователь, от которого осуществляются все рабочие процессы.
* worker\_processes — число рабочих процессов сервера. Оно должно быть не больше, чем количество ядер процессора. Параметр auto установит число автоматически.
* pid — файл с номером главного процесса.
* include — отвечает за подключение иных файлов конфигурации, удовлетворяющих заданной маске.
* events — контекст, состоящий из директив, влияющих на работу сетевого соединения.
  + worker\_connections — максимальное число одновременно работающих соединений одного рабочего процесса.
  + multi\_accept — флаг, который может быть как включен (on), так и выключен (off). Если он включен, то рабочий процесс будет принимать все новые соединения, иначе только одно.
  + use — указывает метод обработки соединений. По умолчанию сервер выбирает наиболее подходящий и эффективный.

http — контекст, состоящий из директив, отвечающих за работу HTTP-сервера.

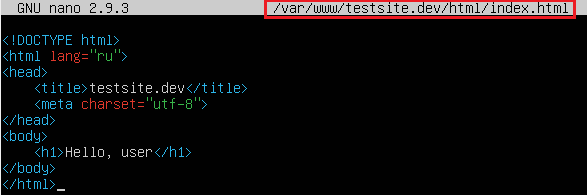
* sendfile — включает (on) или отключает (off) метод отправки данных sendfile().
* tcp\_nopush, tcp\_nodelay — параметры, влияющие на производительность. Первый заставляет сервер отправлять заголовки HTTP-ответов одним пакетом, а второй позволяет не буферизировать данные и отправлять их короткими очередями.
* keepalive\_timeout — параметр, отвечающий за время ожидания keep-alive соединения до его разрыва со стороны сервера.
* keepalive\_requests — максимальное число запросов по одному keep-alive соединению.
* error\_log — лог ошибок веб-сервера. Для сбора ошибок в определенной секции (http, server и т.д.) необходимо разместить директиву внутри нее.
* gzip — сжатие контента.

**Настройка виртуальных хостов**

Первым делом создадим папку для сайта с помощью следующей команды «**sudo mkdir -p /var/www/testsite.dev/html**» 🡪 далее добавим индексный файл «**sudo nano /var/www/testsite.dev/html/index.html**» и заполним его следующими строчками, которые указаны на скрине. Сохраняем его с помощью сочетания клавиш «**Ctrl+O**» 🡪 после создадим конфигурационный файл сайта в папке **sites-available**, пишем команду «**sudo nano /etc/nginx/sites-available/testsite.dev.conf**», в открывшемся файле пишем следующие команды, указанные на скрине 🡪 теперь создадим ссылку в директории **sites-enabled** на конфигурацию сайта **testsite.dev**, чтобы добавить его из доступных во включенные, делается это следующей командой «**sudo ln -s /etc/nginx/sites-available/testsite.dev.conf /etc/nginx/sites-enabled/**» 🡪 проведём тестирование конфигурации, пишем команду «**sudo nginx -t**» 🡪 отключим сайт по умолчанию, удалив запись о дефолтном виртуальном хосте с помощью команды «**sudo rm /etc/nginx/sites-enabled/default**» 🡪 перезагрузим веб-сервер командой «**sudo systemctl restart nginx**»

















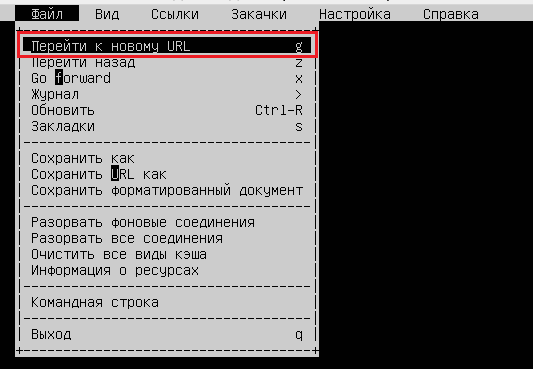


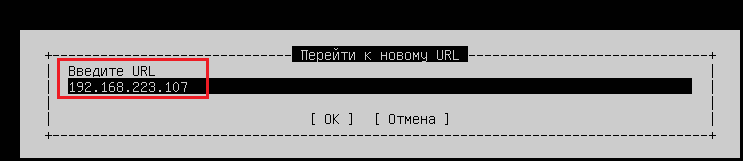
1. **Установим консольный браузер и проверим работу web-сервера**

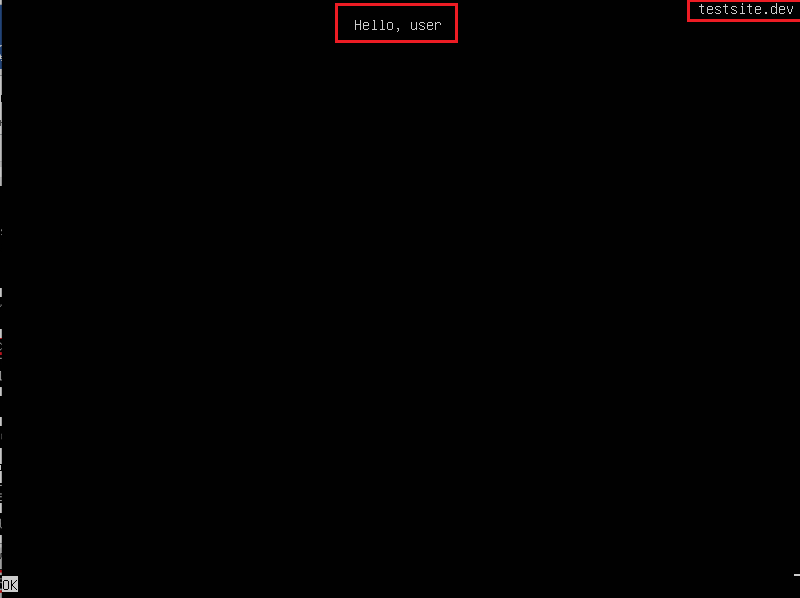
Установим консольный браузер «**links**» для этого напишем команду «**sudo apt install links**» 🡪 запустим браузер с помощью команды «**links**» 🡪 нажимаем «**F10**» и выбираем пункт «**Перейти к новому URL**» 🡪 в открывшейся строчке пишем наш **IP** и нажимаем «**Enter**»



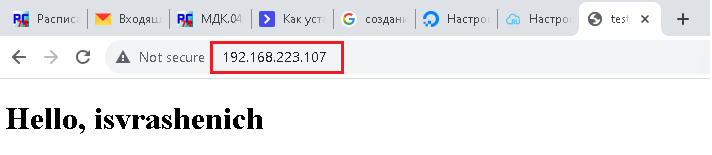




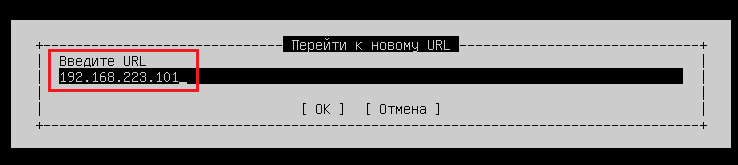




**Проверим на Windows в браузере:**

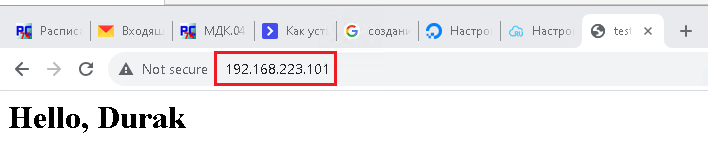


1. **Проверим другой комп (в данном случае соседа), введём IP: 192.168.223.101**



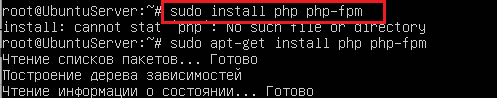


**Проверим на Windows в браузере:**



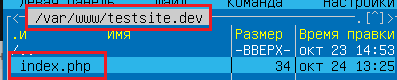
1. **Создадим второй виртуальный хост. Установим MOODLE**

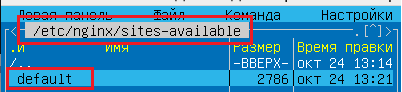






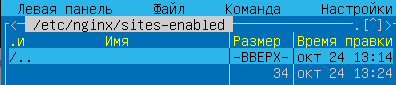






Комментируем все строчки и в самом низу прописываем следующий код



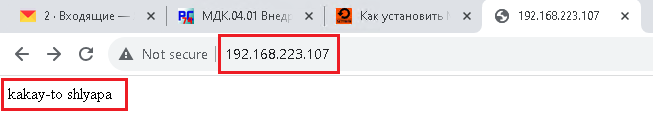


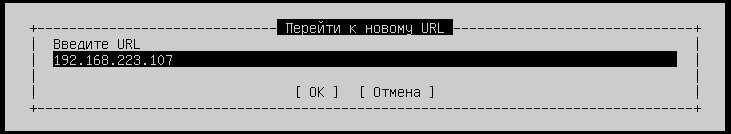














1. **Установка MOODLE**



