Разработка Веб-Фронтенда

На нашем будущем сервере будет постоянно запущен веб-сервер Nginx. Он предоставит пользователю возможность управлять системой через веб-приложение в окне браузера по временному порту 7000. Если пользователь захочет использовать собственный веб-сервер, в приложении ему будет доступен веб-сервер Apache2 с поддержкой PHP. Службы OpenSSH и Samba, автозапуск которых настраивается для создания веб-приложения, также будут по умолчанию отключены с возможностью их постоянного включения через панель управления.

Настройка Nginx на обработку PHP

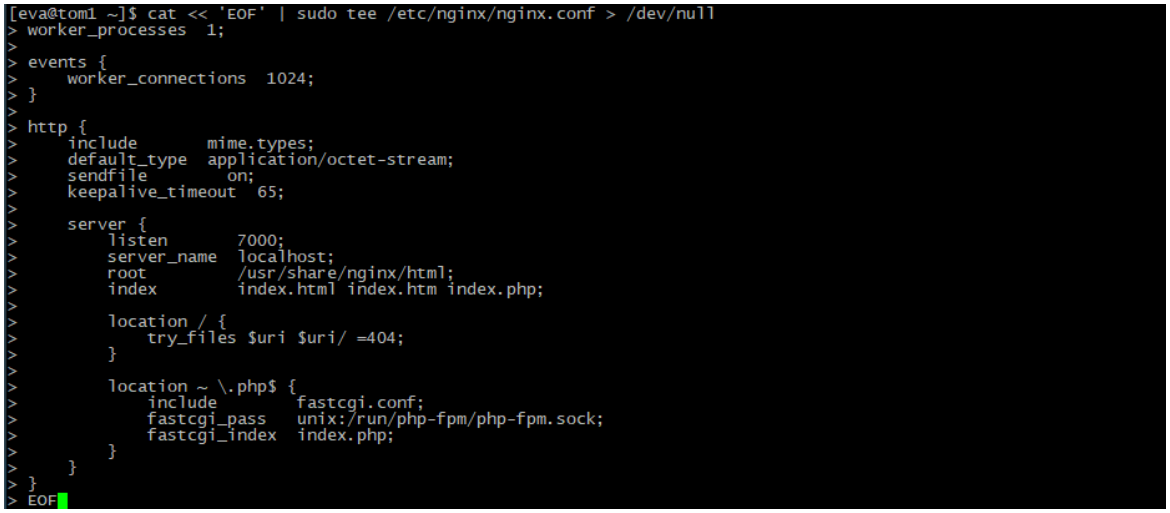
Файл nginx.conf

Мы перепишем конфигурационный файл nginx.conf, добавив правильный блок location ~ \.php$, работающий через UNIX-сокет.

#bash

```
cat << 'EOF' | sudo tee /etc/nginx/nginx.conf > /dev/null
worker_processes  1;

events {
    worker_connections  1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;
    sendfile        on;
    keepalive_timeout  65;

    server {
        listen       7000;
        server_name  localhost;
        root         /usr/share/nginx/html;
        index        index.html index.htm index.php;

        location / {
            try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
            include        fastcgi.conf;
            fastcgi_pass   unix:/run/php-fpm/php-fpm.sock;
            fastcgi_index  index.php;
        }
    }
```

```
}
EOF
```

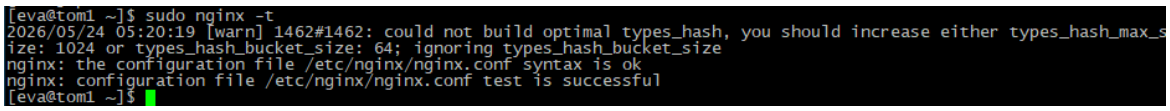


Тестирование синтаксиса nginx.conf

Нам нужно протестировать синтаксис Nginx, чтобы убедиться, что все скобки закрыты и сокет прописан без ошибок.

#bash

```
sudo nginx -t
```



(Тест пройден успешно (syntax is ok, test is successful). Предупреждение о types_hash — это стандартный информационный ворнинг, на работу он не влияет.)

Проверка содержимого файла nginx.conf

#bash

```
cat -n /etc/nginx/nginx.conf
```

```
[eva@tom1 ~]$ cat -n /etc/nginx/nginx.conf
     1  worker_processes  1;
     2
     3  events {
     4      worker_connections  1024;
     5  }
     6
     7  http {
     8      include       mime.types;
     9      default_type  application/octet-stream;
    10      sendfile        on;
    11      keepalive_timeout  65;
    12
    13      server {
    14          listen       7000;
    15          server_name  localhost;
    16          root         /usr/share/nginx/html;
    17          index        index.html index.htm index.php;
    18
    19          location / {
    20              try_files $uri $uri/ =404;
    21          }
    22
    23          location ~ \.php$ {
    24              include        fastcgi.conf;
    25              fastcgi_pass   unix:/run/php-fpm/php-fpm.sock;
    26              fastcgi_index  index.php;
    27          }
    28      }
    29  }
[eva@tom1 ~]$
```

Перезапуск веб-сервера

#bash

```
sudo systemctl restart nginx
```

```
[eva@tom1 ~]$ sudo systemctl restart nginx
[eva@tom1 ~]$
```

Проверка статуса службы

#bash

```
sudo systemctl status nginx
```

```
[eva@tom1 ~]$ sudo systemctl status nginx
● nginx.service - nginx web server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Sun 2026-05-24 05:31:24 UTC; 2min 13s ago
 Invocation: c70deed23faa4bcca7e4807d5183a936
    Process: 1520 ExecStart=/usr/bin/nginx (code=exited, status=0/SUCCESS)
   Main PID: 1521 (nginx)
      Tasks: 2 (limit: 11644)
     Memory: 4.8M (peak: 5.3M)
        CPU: 16ms
     CGroup: /system.slice/nginx.service
             ├─1521 "nginx: master process /usr/bin/nginx"
             └─1522 "nginx: worker process"

May 24 05:31:24 tom1 systemd[1]: Stopping nginx web server...
May 24 05:31:24 tom1 systemd[1]: nginx.service: Deactivated successfully.
May 24 05:31:24 tom1 systemd[1]: Stopped nginx web server.
May 24 05:31:24 tom1 systemd[1]: Starting nginx web server...
May 24 05:31:24 tom1 nginx[1520]: 2026/05/24 05:31:24 [warn] 1520#1520: could not build optimal types_hash, you shou
d increase either types_hash_max_size: 1024 or types_hash_bucket_size: 64; ignoring types_hash_bucket_size
May 24 05:31:24 tom1 systemd[1]: Started nginx web server.
[eva@tom1 ~]$
```

Проверим директорию

Чтобы не запутаться в структуре бэкенда и фронтенда, давайте сначала проверим, что сейчас вообще находится внутри корневой папки Nginx на tom_1.

#bash

```
ls -la /usr/share/nginx/html/
```

Проверим запуск и работу веб-сервера на порту 7000 в браузере на странице http://192.168.1.72:7000/

Проверяем автозапуск службы php-fpm

Убедимся, что служба PHP-FPM активирована в автозапуске, чтобы при старте флешки в ОЗУ она запустилась сама вместе с Nginx.

#bash

```
systemctl is-enabled php-fpm
```

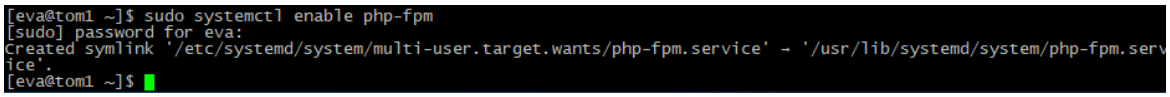
(Примечание: статус disabled - выключена)

Включаем службу php-fpm в автозапуск

Включим службу PHP-FPM в автозапуске, чтобы при старте флешки в ОЗУ она запустилась сама вместе с Nginx.

#bash

```
sudo systemctl enable php-fpm
```



(Примечание: созданы системные симлинки)

Проверяем статус автозапуска службы php-fpm

Убедимся, что служба PHP-FPM активирована в автозапуске, чтобы при старте флешки в ОЗУ она запустилась сама вместе с Nginx. Убедимся, что система теперь рапортует правильный статус.

#bash

```
systemctl is-enabled php-fpm
```

(Примечание: статус enabled - включена)

Снятие системной изоляции с PHP

В Arch Linux служба `php-fpm` по умолчанию заперта (ProtectSystem=full). Из-за этого PHP видит системные файлы пользователей как Read-Only. Снимем это ограничение.

Откройте переопределение настроек службы:

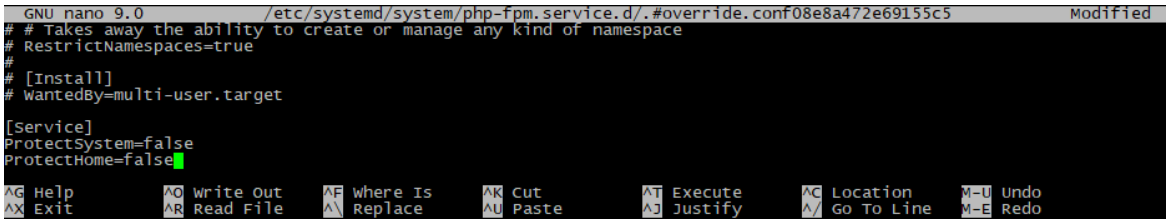
#bash

```
sudo systemctl edit php-fpm
```

Вставте свои строки строго между первой и второй строками комментариев (обычно там есть явная подсказка ### Lines below this comment will be discarded или аналогичная).:

ini

```
[Service]
ProtectSystem=false
ProtectHome=false
```



(Ctrl+O для сохранения, затем Enter и Ctrl+X для выхода)

(Вывод означает правильно отредактированную конфигурацию, и systemd успешно создал конфигурационный файл (drop-in файл) с вашими настройками по пути /etc/systemd/system/php-fpm.service.d/override.conf)

Перезапустите службу:

```
sudo systemctl restart php-fpm
```

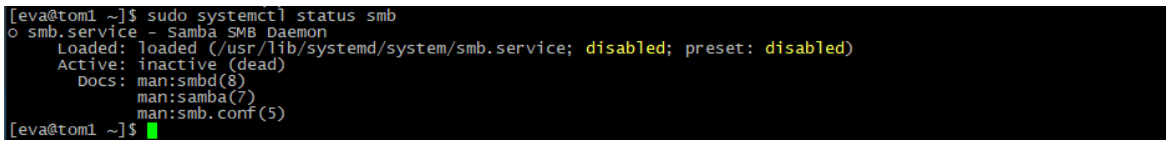
Настройка Samba-сервера

Проверяем состояние Samba-сервера на tom_1

Нам нужно узнать, запущен ли демон Samba (smb), чтобы понять, сможем ли мы сразу подключить сетевой диск в Windows.
Выполните в терминале команду:

#bash

```
sudo systemctl status smb
```



(Этот вывод означает, что служба установлена в системе, но её автозапуск отключен и в данный момент служба не запущена (остановлена))

Создание конфигурационного файла smb.conf

Создадим конфигурационный файл в редакторе nano.
Выполните в терминале команду:

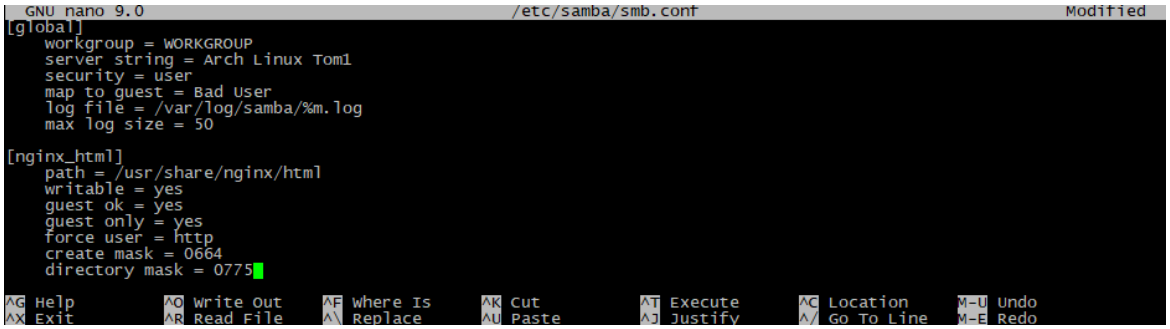
#bash

```
sudo nano /etc/samba/smb.conf
```

Файл пустой и готов к заполнению. Вставьте в него следующий минимальный рабочий конфиг, чтобы открыть доступ к директории Nginx (/usr/share/nginx/html) с автоматическим наследованием безопасных прав доступа (775 для папок и 664 для файлов):

ini

```
[global]
    workgroup = WORKGROUP
    server string = Arch Linux Tom1
    security = user
    map to guest = Bad User
    log file = /var/log/samba/%m.log
    max log size = 50

[nginx_html]
    path = /usr/share/nginx/html
    writable = yes
    guest ok = yes
    guest only = yes
    force user = http
    create mask = 0664
    directory mask = 0775
```



Файл изменен. Нажмите последовательно:CTRL + O, затем клавишу Enter (для записи файла).CTRL + X (для выхода из редактора nano)

Проверка синтаксиса

выполните встроенную команду Samba для проверки синтаксиса файла конфигурации:

#bash

```
testparm -s
```

Тест синтаксиса пройден успешно (Loaded services file OK). Ошибок в файле smb.conf нет.
Сетевая папка nginx_html определена верно.
Следующий шаг — запуск и добавление службы Samba в автозагрузку.

#bash

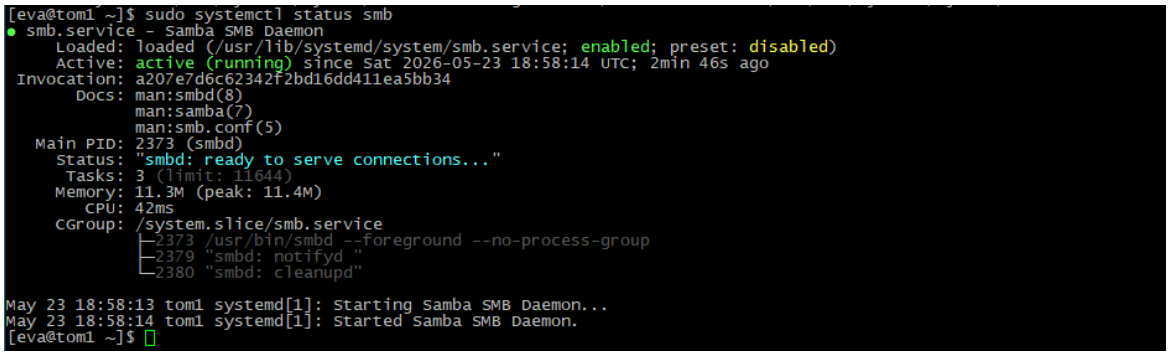
```
sudo systemctl enable --now smb
```

(Симлинк успешно создан, служба добавлена в автозапуск.)
Следующий шаг — обязательная проверка статуса запущенного демона Samba.

Проверка статуса

#bash

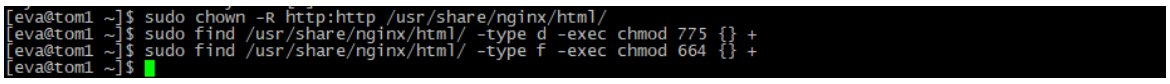
```
sudo systemctl status smb
```



(Служба smb работает в режиме active (running) и полностью готова принимать сетевые подключения.)
Службы настроены. Передаем владение корневым каталогом сайта встроенному веб-пользователю http. Задаем права 775 для всех папок (чтобы Samba и Nginx могли создавать файлы) и 664 для файлов (только чтение и запись, без флагов исполнения):.
Выполните в терминале команду:

#bash

```
sudo chown -R http:http /usr/share/nginx/html/
sudo find /usr/share/nginx/html/ -type d -exec chmod 775 {} +
sudo find /usr/share/nginx/html/ -type f -exec chmod 664 {} +
```



(Права 777/664 назначены успешно.)
Следующий обязательный шаг по нашему плану — проверка того, как система применила эти права к содержимому каталога.

#bash

```
ls -la /usr/share/nginx/html
```

(Проверка прав прошла успешно. Строки drwxrwxr-x для текущей папки (.) и -rw-rw-r– для файлов index.html и 50x.html подтверждают, что доступ полностью открыт на чтение, запись для пользователей системы.)

Подключение сетевой папки в Windows

Получим точный текущий IP-адрес сервера tom_1

Выполните в терминале команду:

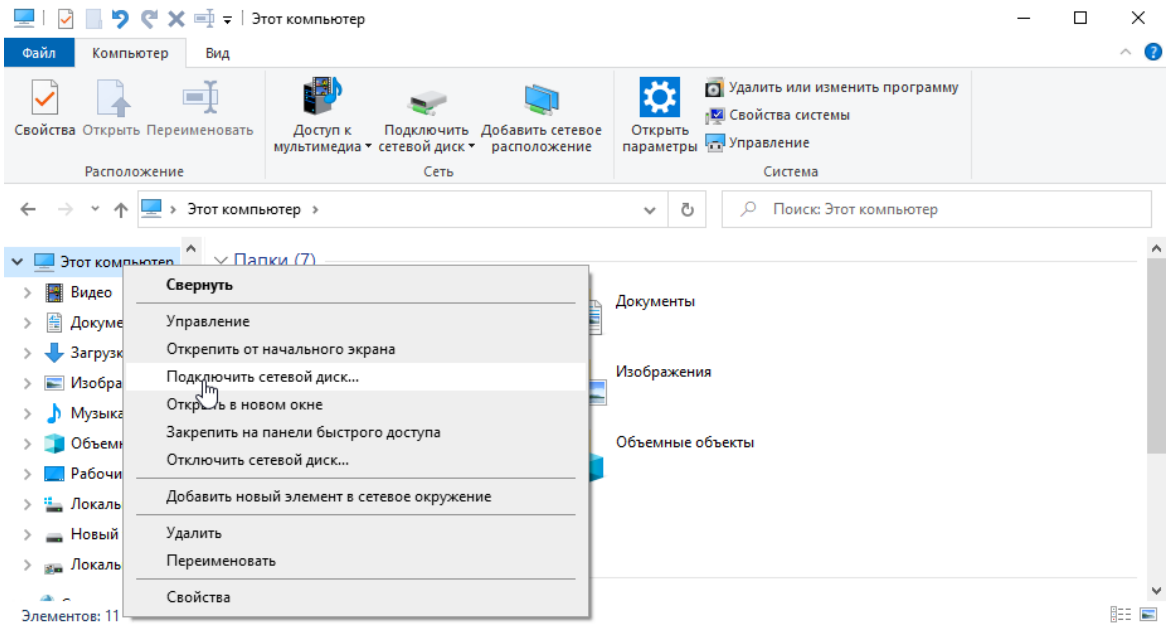
#bash

```
ip -br address show scope global | awk '{print $3}' | cut -d/ -f1
```

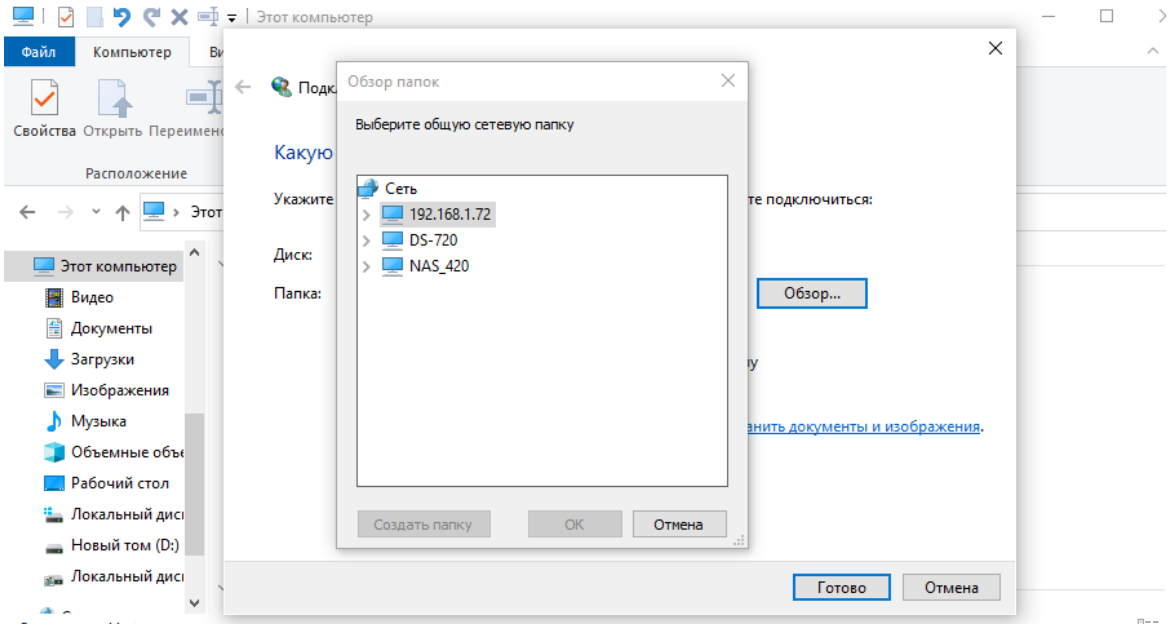
```
[eva@tom1 ~]$ ip -br address show scope global | awk '{print $3}' | cut -d/ -f1
192.168.1.72
[eva@tom1 ~]$
```

Теперь папка готова к подключению в качестве сетевого диска в среде Windows, чтобы вы могли открыть её через Notepad++.

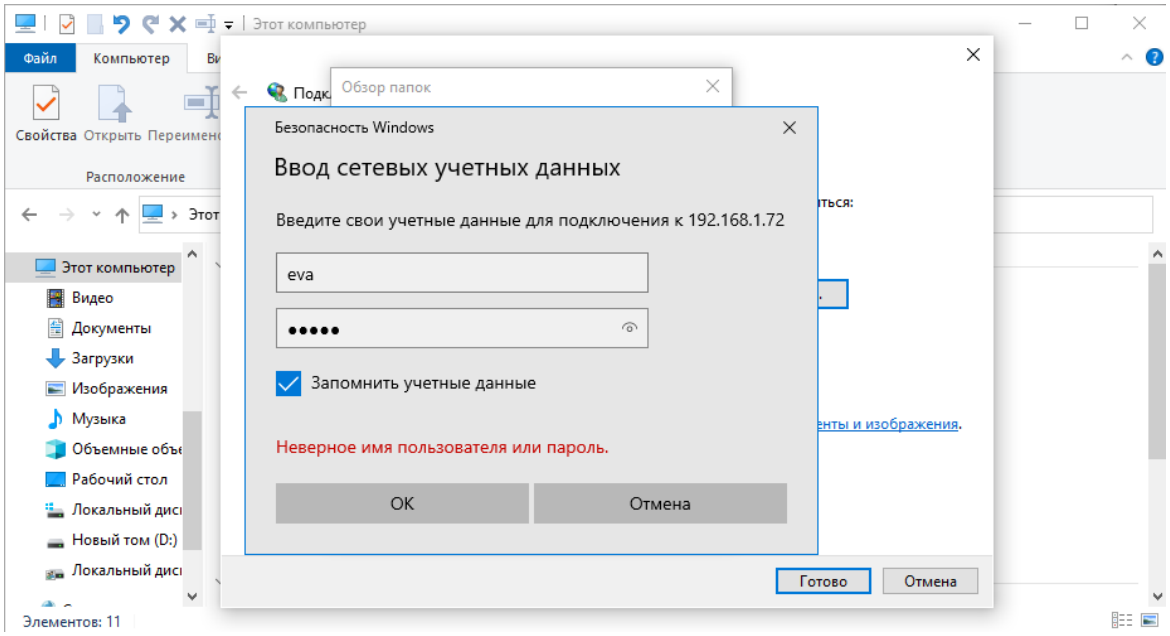
1. Откройте Проводник (Этот компьютер) на вашей Windows-машине.
2. В верхнем меню нажмите кнопку «Подключить сетевой диск» (или нажмите правой кнопкой мыши по «Этот компьютер» → «Подключить сетевой диск»).



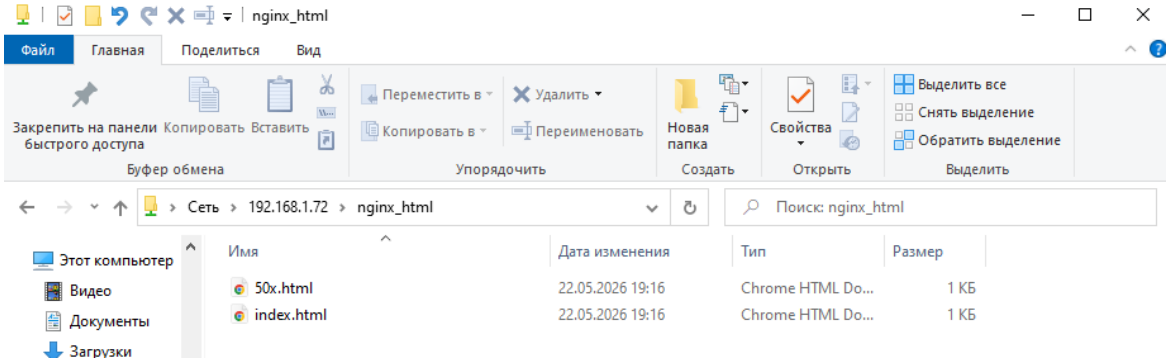
1. В поле «Папка» введите сетевой путь, используя IP-адрес вашего сервера tom_1 (из логов PuTTY: 192.168.1.72 или через кнопку обзор)



Введите сетевые учетные данные



Зайдите через проводник



(В корне /usr/share/nginx/html/ только файлы 50x.html и index.html.)
Разворачиваем структуру каталогов для нашего веб-интерфейса. Создадим стандартные папки для стилей, скриптов и серверной логики.

Системный пользователь http

Пользователь http в Arch Linux — это встроенный системный пользователь, от имени которого по умолчанию работают веб-серверы (например, Apache или Nginx) и сопутствующие им службы.
Он создается автоматически при установке этих программ для изоляции процессов и обеспечения безопасности.

Назначение

- Безопасность: Веб-серверы не должны работать под правами суперпользователя (root). Если злоумышленник найдет уязвимость в вашем сайте, он получит доступ только к файлам с правами пользователя http, что убережет остальную систему от взлома.
- Права на файлы: Этот пользователь владеет файлами и папками, к которым сервер имеет доступ (обычно они расположены в директории /srv/http/).
- Группа http: Для удобства существует одноименная группа http, в которую могут входить другие пользователи, чтобы иметь возможность редактировать файлы сайта без изменения прав доступа к ним через sudo

Применение

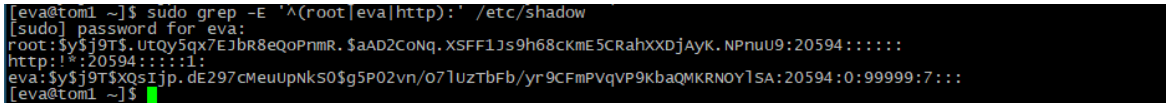
- Размещение сайтов: При настройке Nginx, Apache или PHP-FPM часто требуется указывать, что процесс должен запускаться от имени http.
- Настройка разрешений (Permissions): Если сайт выдает ошибку доступа (например, 403 Forbidden), обычно это означает, что у пользователя http нет прав на чтение нужных файлов или папок.
- Безопасность каталогов: Если скриптам на сайте нужно загружать файлы на сервер, папке загрузки необходимо выдать права (владельца) для пользователя http

Вывод строк пользователей системы

Выводим строки трех нами известных пользователей из базы данных.

#bash

```
sudo grep -E '^(root|eva|http):' /etc/shadow
```



root:yj9T$...:20594:::::

- Второе поле — длинный хэш y.... Это зашифрованный пароль суперпользователя.Число 20594 — дата последнего изменения пароля (в днях от 1970 года).В конце строки — пустые поля. Это значит, что для root нет никаких ограничений.

eva:yj9T$...:20594:0:99999:7:::

- Тоже видим хэш личного пароля.
- Параметры 0:99999:7 означают стандартные правила пользователя: пароль можно менять сразу (0), он действует 99999 дней, а за 7 дней до истечения система начнет предупреждать. Восьмое поле пустое — аккаунт не блокируется.

http:!*:20594::::::1:

- Второе поле содержит !*. Это маркер того, что пароль заблокирован (вход по паролю невозможен, учетка техническая).Внимание в самый конец строки: ::::::1:
- На предпоследней позиции (8-е поле) стоит цифра 1. В системе Linux это означает, что учетная запись принудительно заблокирована подсистемой безопасности PAM через 1 день после начала эпохи Unix (то есть 2 января 1970 года).

Изменяем параметры пользователя http

Уберем эту единицу из конца строки, чтобы сделать учетную запись бессрочной.

#bash

```
sudo chage -E -1 http
```

Проверка изменений в файле /etc/shadow

#bash

```
sudo grep '^http:' /etc/shadow
```

(Строка завершается чистыми двоеточиями (::::::), что означает: блокировка PAM полностью снята, аккаунт http стал бессрочным)

Настройка беспарольного доступа в sudoers

Чтобы PHP-скрипты могли вызывать утилиты управления аккаунтами без ввода пароля и без наличия текстового экрана (TTY), настроим правила безопасности. Команды будут пробрасываться во внешнюю систему через утилиту `systemd-run` для обхода ограничений безопасности PHP.

Откройте конфигурационный файл строго через visudo:

#bash

```
sudo EDITOR=nano visudo
```

В самый конец файла добавьте следующие строки:

text

```
http ALL=(ALL) NOPASSWD: /usr/bin/useradd, /usr/bin/userdel,
/usr/bin/usermod, /usr/bin/chpasswd, /usr/bin/groupadd,
/usr/bin/groupdel
http ALL=(ALL) NOPASSWD: /usr/bin/sh, /usr/bin/systemd-run,
/usr/bin/sed
Defaults:http !requiretty
```

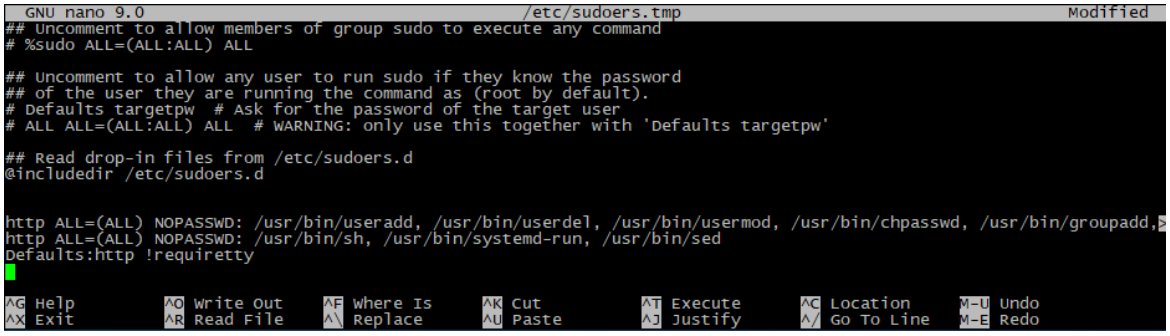


Схема веб-панели управления в основной системе (tom_1)

Папка веб-сервера nginx_html находится по пути /usr/share/nginx/html/ и имеет следующую структуру файлов бэкенда (PHP) и фронтенда (JS/CSS):

```
/usr/share/nginx/html/                    # Корневая директория веб-сервера Nginx
├── index.html                            # Главный интерфейс панели (вкладки,
таблицы, модальные окна)
```

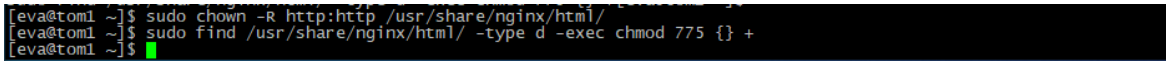
```
├── css/
│   └── style.css                    # Стили оформления интерфейса панели управления
├── js/
│   └── app.js                       # Клиентская логика (асинхронные Fetch-запросы к API, фильтры)
└── api/
    ├── users.php                    # Серверный обработчик для системных пользователей (/etc/passwd)
    └── groups.php                   # Серверный обработчик для системных групп (/etc/group)
```

Создание директорий

Временно изменим права для работы в консоли

#bash

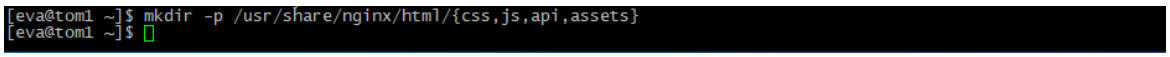
```
sudo chown -R http:http /usr/share/nginx/html/
sudo find /usr/share/nginx/html/ -type d -exec chmod 775 {} +
```



Выполните в терминале PuTTY одну команду:

#bash

```
mkdir -p /usr/share/nginx/html/{css,js,api,assets}
```

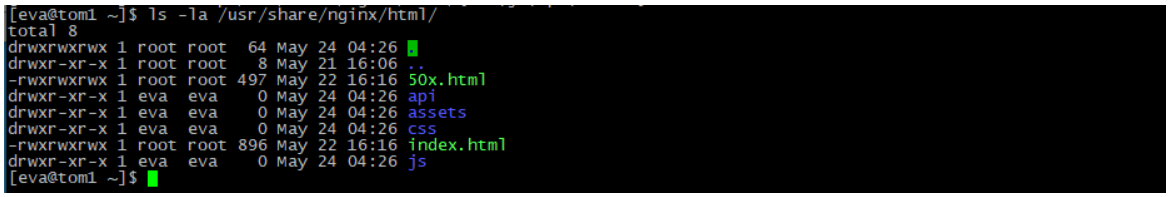


Папки созданы. Теперь обязательный шаг контроля: проверяем, какие права доступа и владельцы назначены для новых директорий, чтобы Windows-пользователь Samba и веб-сервер Nginx могли с ними работать.

Контроль папок

#bash

```
ls -la /usr/share/nginx/html/
```



(Папки создались под пользователем eva, но у них стоят ограниченные права drwxr-xr-x. Из-за этого Windows через Samba не сможет создавать или изменять файлы внутри этих подкаталогов.)

Права доступа к файлам

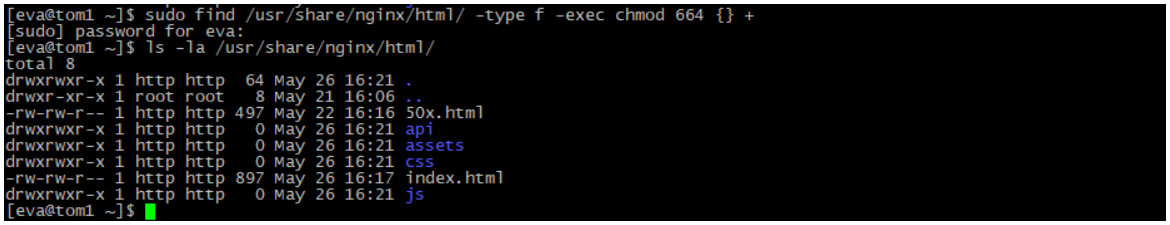
#bash

```
sudo find /usr/share/nginx/html/ -type f -exec chmod 664 {} +
```

Проверка назначения прав пользователя

#bash

```
ls -la /usr/share/nginx/html/
```



(Права drwxrwxr-x (775) успешно применились ко всем новым директориям (api, assets, css, js) - подсвечены синим, и -rw-rw-r– к файлам они подсвечены белым. Теперь пользователи eva и системный пользователь http, и Samba имеют полный доступ.)

!!!!!!!!!!! Разработка Веб-Фронтенда !!!!!!!!!!!

!!!!!! Убрать перед сборкой iso !!!!!!!! Проверить версии linux Драйверы сетевой карты реального сервера Изменить временно на 192.168.1.150 Стереть sfid

запись исо Изменить постоянно на 192.168.1.72 Вернуть sfid

!!!!!!!!!!! переходим к iso !!!!!!!!!!!