?????????? ?? SSH

1. ????????? ? ??????? ?????????

????????? SSH-??????? (?? ???????)

Для Debian/Ubuntu серверная часть SSH устанавливается пакетом `openssh-server`.

```
sudo apt update && sudo apt install -y openssh-server
```

????????? ? ???????????? (`/etc/ssh/sshd_config`)

Это главный конфигурационный файл SSH-сервера. После внесения изменений **необходимо перезапустить службу**: `sudo systemctl restart sshd`. Ниже приведен пример безопасной конфигурации.

```
#Порт, на котором слушает SSH. 22 - стандарт. Для безопасности рекомендуется сменить.
Port 2222
#Запретить вход для пользователя root. Всегда используйте обычного пользователя с sudo.
PermitRootLogin no


#Максимальное количество попыток ввода пароля перед разрывом соединения.
MaxAuthTries 3


#Отключить аутентификацию по паролю (самый важный шаг для безопасности!).
#После этого вход будет возможен ТОЛЬКО по SSH-ключу.
PasswordAuthentication no


#Включить аутентификацию по ключам.
PubkeyAuthentication yes


#Путь к файлу с разрешенными публичными ключами.
AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
```

```
#Запретить вход пользователям с пустыми паролями.
PermitEmptyPasswords no


#Использовать PAM (Pluggable Authentication Modules). Важно для многих систем.
UsePAM yes
```

Ситуация из жизни: Вы настраиваете новый сервер в интернете. Первым делом после установки ОС вы заходите по временному паролю, настраиваете `sshd_config` как в примере, добавляете свой SSH-ключ (см. раздел 2) и перезапускаете SSH. Только после этого сервер можно считать минимально защищенным.

2. ?????? ? SSH-??????? (PublicKey Authentication)

??? ?: ????????? ?????? (?? ????? ??????????)

Эта команда создает пару ключей: приватный (`id_rsa`), который нужно хранить в секрете, и публичный (`id_rsa.pub`), который вы будете копировать на серверы.

```
ssh-keygen -t rsa -b 4096
```

Вам будет предложено ввести пароль для ключа. Это дополнительный слой безопасности: даже если кто-то украдет ваш приватный ключ, он не сможет им воспользоваться без пароля.

??? ?: ??????????? ?????????? ????? ?? ??????

Самый простой и безопасный способ — использовать утилиту `ssh-copy-id`. Она сама создаст нужные файлы и установит правильные права доступа на сервере.

```
ssh-copy-id username@your_server_ip
```

Ситуация из жизни: Вы получили доступ к новому рабочему серверу. Чтобы не вводить пароль каждый раз и повысить безопасность, вы одной этой командой прописываете свой ключ на сервере. После этого можно отключать парольную аутентификацию.

3. ??????? ???????????? ? ?????????????

1. ??????? ???????????

Стандартный способ подключения с использованием вашего ключа по умолчанию.

```
ssh username@your_server_ip
```

2. ??????????? ?? ????????????? ????

Используется, если вы изменили порт в `sshd_config` для повышения безопасности.

```
ssh -p 2222 username@your_server_ip
```

3. ??????????? ? ????????? ??????????? ?????

Полезно, если у вас несколько ключей для разных проектов или серверов.

```
ssh -i ~/.ssh/my_other_key username@your_server_ip
```

4. ??????????? ????? ?? ?????? (SCP)

Простой способ загрузить файл на удаленный сервер.

```
scp /path/to/local/file.txt username@your_server_ip:/path/to/remote/directory/
```

5. ??????????? ????? ? ??????? (SCP)

Ключ `-r` (рекурсивно) позволяет скопировать целый каталог.

```
scp -r username@your_server_ip:/path/to/remote/directory /path/to/local/destination
```

6. ????????????? ????????? (rsync)

Rsync — более мощный инструмент, чем scp. Он копирует только измененные файлы, что идеально для бэкапов и развертывания сайтов.

```
rsync -avz --progress /path/to/local/project/ username@your_server_ip:/var/www/project
```

7. ????????? ??????? ????? (?????? ? ????????? ??)

Позволяет безопасно подключиться к базе данных, работающей на удаленном сервере, как будто она запущена у вас локально.

```
ssh -L 5433:localhost:5432 username@your_server_ip
```

Ситуация из жизни: На сервере работает PostgreSQL на порту 5432, но доступ к нему извне закрыт брандмауэром. Эта команда "пробрасывает" удаленный порт 5432 на ваш локальный порт 5433. Теперь вы можете подключиться к `localhost:5433` вашим любимым GUI-клиентом для баз данных.

8. ????????? ??????? ????? (???????? ????????? ??????)

Позволяет сделать ваш локальный веб-сервер доступным через публичный IP вашего удаленного сервера.

```
ssh -R 8080:localhost:3000 username@your_server_ip
```

Ситуация из жизни: Вы разработали веб-приложение на своем ноутбуке (на порту 3000) и хотите показать его коллеге. Эта команда делает так, что при обращении к `your_server_ip:8080` трафик будет перенаправляться на ваш ноутбук.

9. ????????????? ????? ???????????? (~/.ssh/config)

Чтобы не вводить каждый раз длинные команды, можно создать файл конфигурации. Создайте файл `~/.ssh/config` и добавьте в него блок:

```
Host my-prod-server
    HostName your_server_ip
    User your_username
    Port 2222
    IdentityFile ~/.ssh/prod_key
```

После этого для подключения достаточно выполнить короткую команду:

```
ssh my-prod-server
```

10. ???????? SSH-??????? (?????? ? ?????? ?? NAT)

Это продвинутый метод, позволяющий получить доступ к компьютеру, который находится за роутером или брандмауэром (например, ваш домашний ПК).

На домашнем ПК (за NAT) выполните:

```
ssh -R 9090:localhost:22 user_on_public_server@public_server_ip
```

Эта команда говорит: "Подключись к публичному серверу и сделай так, чтобы трафик, приходящий на его порт 9090, перенаправлялся на мой локальный 22 порт".

Теперь, находясь на публичном сервере, вы можете подключиться к своему домашнему ПК:

```
ssh -p 9090 user_on_home_pc@localhost
```

Revision #4
Created 5 October 2025 12:12:35 by Admin
Updated 5 October 2025 12:16:05 by Admin