Удаленное администрирование

Установка и настройка ssh-сервера. Использование ssh-клиента. Проброс портов. Генерация ключей. Socks5-прокси. Удаленная работа с графическими приложениями

[SSH](#_oq7f8phvod44)

[SOCKS](#_7375w7smet58)

[SSH-туннелирование](#_on6sgcooxnwd)

[Практика проброса порта к mysql](#_xirh8zmil07e)

Удаленное копирование файлов

Практическое [задание](#_3tfrjxxltv85)

[Дополнительные материалы](#_jeeoh8oz8lfe)

[Используемая литература](#_uvp6qax5r1ok)

# 

# SSH

Как правило, удаленное администрирование выполняется с применением **ssh** или **VNC**. Протокол **ssh** широко используется, стандартно работает на порте 22 и включает такие сервисы, как **scp** и **sftp**.

В работе **ssh** использует протокол Диффи-Хеллмана для генерации общего секретного ключа. Для аутентификации применяется пароль или алгоритм **RSA** c публичным и приватным ключом (рекомендуется).

На **VDS** ssh-сервер обычно установлен, а в виртуальной машине устанавливается с помощью этой команды.

|  |
| --- |
| # apt update && apt install openssh-server |

Проверить статус.

|  |
| --- |
| # systemctl status sshd.service |

Настройки хранятся в **/etc/sshd\_config** и по умолчанию выглядят так.

|  |
| --- |
| # Package generated configuration file # See the sshd\_config(5) manpage for details  # What ports, IPs and protocols we listen for Port 22 # Use these options to restrict which interfaces/protocols sshd will bind to #ListenAddress :: #ListenAddress 0.0.0.0 Protocol 2 # HostKeys for protocol version 2 HostKey /etc/ssh/ssh\_host\_rsa\_key HostKey /etc/ssh/ssh\_host\_dsa\_key HostKey /etc/ssh/ssh\_host\_ecdsa\_key HostKey /etc/ssh/ssh\_host\_ed25519\_key #Privilege Separation is turned on for security UsePrivilegeSeparation yes  # Lifetime and size of ephemeral version 1 server key KeyRegenerationInterval 3600 ServerKeyBits 1024  # Logging SyslogFacility AUTH LogLevel INFO  # Authentication: LoginGraceTime 120 PermitRootLogin yes #prohibit-password StrictModes yes  RSAAuthentication yes PubkeyAuthentication yes #AuthorizedKeysFile %h/.ssh/authorized\_keys  # Don't read the user's ~/.rhosts and ~/.shosts files IgnoreRhosts yes # For this to work you will also need host keys in /etc/ssh\_known\_hosts RhostsRSAAuthentication no # similar for protocol version 2 HostbasedAuthentication no # Uncomment if you don't trust ~/.ssh/known\_hosts for RhostsRSAAuthentication #IgnoreUserKnownHosts yes  # To enable empty passwords, change to yes (NOT RECOMMENDED) PermitEmptyPasswords no  # Change to yes to enable challenge-response passwords (beware issues with # some PAM modules and threads) ChallengeResponseAuthentication no  # Change to no to disable tunnelled clear text passwords #PasswordAuthentication yes  # Kerberos options #KerberosAuthentication no #KerberosGetAFSToken no #KerberosOrLocalPasswd yes #KerberosTicketCleanup yes  # GSSAPI options #GSSAPIAuthentication no #GSSAPICleanupCredentials yes  X11Forwarding yes X11DisplayOffset 10 PrintMotd no PrintLastLog yes TCPKeepAlive yes #UseLogin no  #MaxStartups 10:30:60 #Banner /etc/issue.net  # Allow client to pass locale environment variables AcceptEnv LANG LC\_\*  Subsystem sftp /usr/lib/openssh/sftp-server  # Set this to 'yes' to enable PAM authentication, account processing, # and session processing. If this is enabled, PAM authentication will # be allowed through the ChallengeResponseAuthentication and # PasswordAuthentication. Depending on your PAM configuration, # PAM authentication via ChallengeResponseAuthentication may bypass # the setting of "PermitRootLogin without-password". # If you just want the PAM account and session checks to run without # PAM authentication, then enable this but set PasswordAuthentication # and ChallengeResponseAuthentication to 'no'. UsePAM yes |

* Port 22 — имеет смысл заменить на нестандартный.
* **ListenAdress** — можно указать только один из доступных адресов для прослушивания.
* **PermitRootLogin** — имеет смысл заменить на **no** (убедитесь, что есть пользователи в группе **sudo**).
* **PasswordAuthentication** — по умолчанию **yes**, но лучше поставить в **no** после генерации ключей.
* **X11Forwarding** — возможность запуска X11-приложений. Если такой необходимости нет, лучше поставить **no**.

Разберем генерацию ключей. Вы находитесь на клиентской машине.

Ключи генерируются этой командой.

|  |
| --- |
| ssh-keygen -t rsa |

Потребуется ввести пароль: не к **ssh**, а к ключу. Если планируется использовать этот ключ в скриптах, пароль указывать не надо. Если забудете пароль, ключ станет бесполезным.

В директории **.ssh** будут созданы файлы **id\_rsa** и **id\_rsa.pub**. **id\_rsa** остается на вашей машине — это приватный ключ. **id\_rsa.pub** — публичный ключ, его можно загрузить на сервер. Машины, на которые вы загрузили публичный ключ, позволят подключиться. Загрузите публичный ключ **id\_rsa.pub** в домашнюю директорию на удаленной Linux-машине, которую будете администрировать.

С помощью **scp** скопируйте ключ.

|  |
| --- |
| cd ~/.ssh scp id\_rsa.pub user@1.2.3.4:/home/user -p 49111 |

Вместо **1.2.3.4** и **user** подставьте актуальные IP-адрес и имя пользователя. Нужно будет ввести пароль (ведь ключи еще не настроены). Вместо **49111** подставьте нужный порт (если это 22, то можно не указывать).

Затем подключитесь к удаленной машине.

|  |
| --- |
| $ ssh user@1.2.3.4 -p 49111 |

Скопируйте ключи.

|  |
| --- |
| $ cat id\_rsa.pub >> ~/.ssh/authorized\_keys $ rm id\_rsa.pub |

В файле **/etc/ssh/sshd\_config** установите следующие значения.

|  |
| --- |
| RSAAuthentication yes PubkeyAuthentication yes  PasswordAuthentication no |

Перезапустим **sshd**.

|  |
| --- |
| #systemctl restart sshd.service |

Теперь можно подключаться без использования пароля (с ним уже не получится). Будьте внимательны при настройке: лучше отключить аутентификацию по паролю, когда убедились, что она выполняется по ключу.

При работе в Windows c **putty** можно сгенерировать ключ с помощью **puttygen**. Загрузить публичный ключ **.pub** на сервер можно с помощью любого sftp-клиента. В настройках аутентификации **putty ssh** укажите путь к приватному ключу (обычно с расширением **.ppk**).

Обратите внимание, что форматы ключей отличаются: в **authorized\_keys** нужно вставить ключ в формате **OpenSSH** (начинается с **ssh-rsa AAAA…**).

Подробнее: <https://lode.in/administration/generatsiya-ssh-klyuchej-s-putty-i-nastrojka-avtorizatsii-po-zakry-tomu-klyuchu-ssh.html>.

### SOCKS

Протокол **SOCKS v5** одобрен организацией **IETF** (Internet Engineering Task Force) в качестве стандарта Internet и включен в **RFC 1928** (<http://ypn.ru/366/>). Согласно источнику <http://ypn.ru/366/> протокол **SOCKS** относится к сеансовому уровню.

Аутентификация в **SOCKS5** основана на сертификатах **X.509** (на базе **RSA**) или паролях.

Современные браузеры могут использовать SOCKS5-прокси. В качестве адреса отправителя будет виден адрес SOCKS5-сервера, но при этом DNS-запросы (в отличие от классического HTTP-прокси, где DNS-запросы выполняет прокси-сервер) будут выполняться с вашей машины.

В качестве SOCKS5-прокси можно использовать openssh-сервер на Linux-машине и **putty** на клиентской (или **ssh** для Linux и MAC OS X-машин).

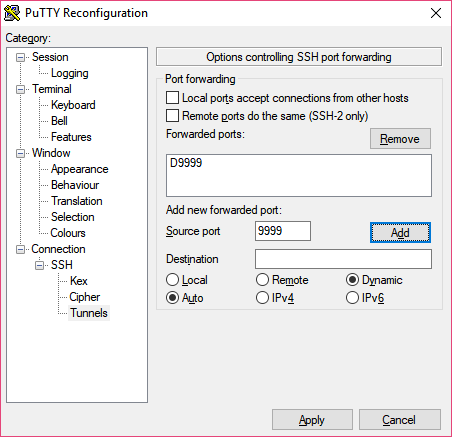
Разберем, что нужно для настройки SOCKS5-прокси. На сервере (в идеале — VDS-сервере с «белым» IP-адресом и доступом в Интернет) должен быть установлен OpenSSH-сервер (по идее и разрешенные записи в **iptables**), позволяющий выполнять подключения по протоколу **SSH** (secure shell) **SFTP** и предоставляющий множество возможностей туннелирования.

Если OpenSSH-сервер еще не установлен (для Ubuntu).

|  |
| --- |
| $ sudo apt install openssh-server |

Теперь подключаемся к серверу по **ssh** с помощью **putty** (если не меняли порт на 22). Затем заходим в настройки и выбираем **Connection — SSH — Tunnels**. Ставим тип порта **Dynamic**. В **Source port** вписываем номер необходимого TCP-порта (например, 9999). Нажимаем **Add** (иначе не сохранится). Применяем сохраненные данные в **Apply**.

В самой консоли делать ничего не надо, достаточно, чтобы прошла аутентификация (по паролю, а лучше по сертификату). После этого можно использовать SOCKS5-прокси.



Если использовать не **putty**, а **ssh**, то же самое можно сделать при подключении к ssh-серверу с помощью такой команды.

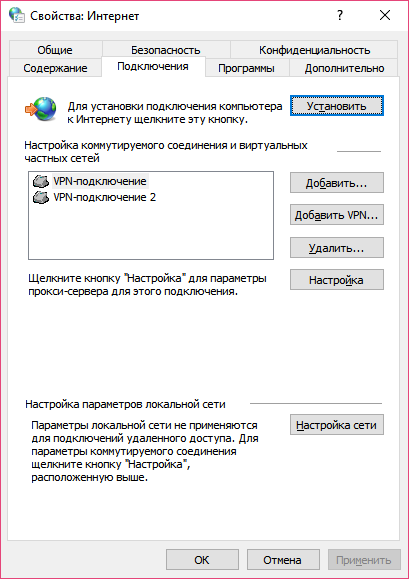
|  |
| --- |
| ssh -f -C2qTnN -D <порт> <удаленный\_пользователь>@<удаленный\_сервер> |

* **-f** — запросит переход **ssh** в фоновый режим только перед выполнением команды.
* **-C** — включит сжатие всех данных (включая **stdin**, **stdout**, **stderr** и данные для перенаправленных Х11- и TCP/IP-соединений).
* **-2** — принудит **ssh** использовать только протокол версии 2.
* **-q** — включит тихий режим, который подавляет все предупреждения и диагностические сообщения. Будут отображены только фатальные ошибки.
* **-T** — отменит переназначение терминала.
* **-n** — перенаправит стандартный ввод из **/dev/null** (предотвратит чтение из стандартного ввода).
* **-N** — укажет, что удаленную команду не надо выполнять.
* **-D [локальный IP : ]** — порт.

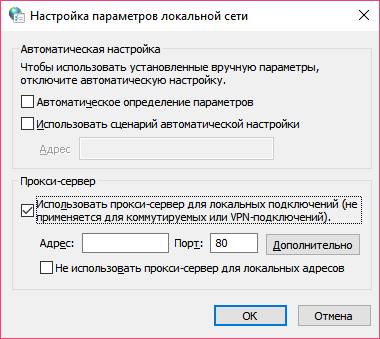
Пример.

|  |
| --- |
| ssh -f -C2qTnN -D 9999 user@blablablator.ru |

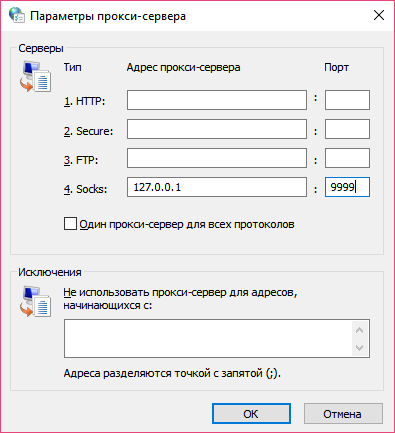
Заходим в настройки браузера, выбираем «Настройки прокси-сервера». В окне настроек браузера выбираем «Настройки сети».



Включаем галочку «Прокси-сервер» и жмем «Дополнительно».



Задаем в качестве SOCKS-прокси 127.0.0.1 и указанный нами порт.



Нажимаем «ОК» и заходим в браузере на <https://2ip.ru> или [https://ip.badrequest.me](https://ip.badrequest.me/). Проверяем, что теперь работаем через **VDS**.

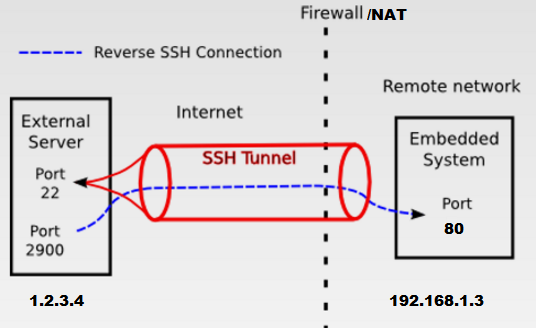
### SSH-туннелирование

SSH (secure shell) — протокол, разработанный на смену устаревшим **telnet** и **rlogin**. Он фактически вытеснил и **ftp**, и протоколы удаленного копирования **UUCP** (**unix-to-unix cp**), и **RCP** (использующего в качестве транспорта **rsh** (**remote shell**), часть пакета **rlogin**). Протокол **SFTP** не является разновидностью **FTP**, хотя и выполняет аналогичные функции. **SFTP** использует в качестве транспорта **SSH**, как и протокол **SCP**, применяющий **SSH** для удаленного копирования файлов. В качестве клиента может выступать **scp**, **WinSCP**.

**SSH**, **SFTP** и **SCP** используют для работы порт **22/TCP**.

**SSH** применяет протокол Диффи-Хеллмана для получения общего симметричного ключа. Для аутентификации может использоваться как пароль (аналогично работе программы **login** перед началом сессии **shell** или **bash**), так и RSA-сертификат. Это удобно не только для удаленного подключения, но и для работы с Git, который в качестве транспорта использует **https** или **ssh**. В случае **ssh** Git будет работать, не спрашивая пароля. При этом на клиентской стороне используется приватный ключ, а на серверной (на удаленной машине или git-хостинге) — публичный.

**SSH** позволяет туннелировать прикладные протоколы, пробрасывая TCP-порты, обходить NAT и файрволы, пробрасывать порты в обратную сторону и многое другое.



На картинке изображено подключение к машине, находящейся за NAT. Предположим, ее адрес — 192.168.1.3, соответственно, нет возможности принимать входящие соединения из внешней сети. Или у машины есть «белый» IP-адрес, но административно файрволом закрыта возможность передавать данные на порт 80 локальной машины (и другие ее порты, кроме исходящего трафика). В этом случае клиент подключается к внешнему серверу с «белым» IP-адресом (например, 1.2.3.4) по **ssh** и пробрасывает свой локальный порт 80 на порт сервера 1.2.3.4:2900.

Подключаясь на сервере к 1.2.3.4:2900, внешний клиент получает доступ к 192.168.1.3:80, который без **ssh** недоступен.

Запуск на клиенте.

|  |
| --- |
| ssh -R 127.1:80:1.2.3.4:2900 user@1.2.3.4 |

Обратите внимание, 127.1 — корректная запись для адреса 127.0.0.1.

Также с помощью **ssh** можно получить доступ к приложению, слушающему только 127.0.0.1 адрес.

В Linux.

|  |
| --- |
| ssh -L 8600:127.1:3306 user@1.2.3.4 |

После этого, локально подключаясь к 127.0.0.1:8600 mysql-клиентом, вы будете работать с mysql на 1.2.3.4, слушающем 127.0.0.1:3306.

Более подробно изучить возможности **ssh**, научиться генерировать и использовать RSA-сертификаты для него можно по этому [материалу](https://habrahabr.ru/post/122445/). Узнать, как использовать ssh для VPN, можно здесь: <https://habrahabr.ru/post/319158/>. В разделе «Практика» этого урока есть пример подключения к **mysql** с использованием **putty**.

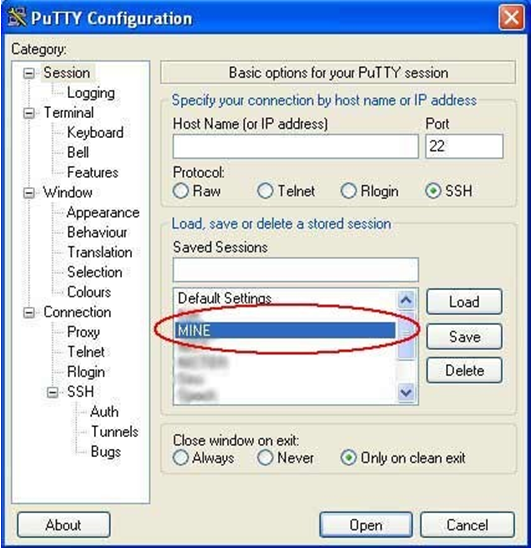
## Практика проброса порта к mysql

Чтобы работать с удаленной **mysql** или любой другой БД, когда сервер слушает только **localhost**, можно применить **ssh**.

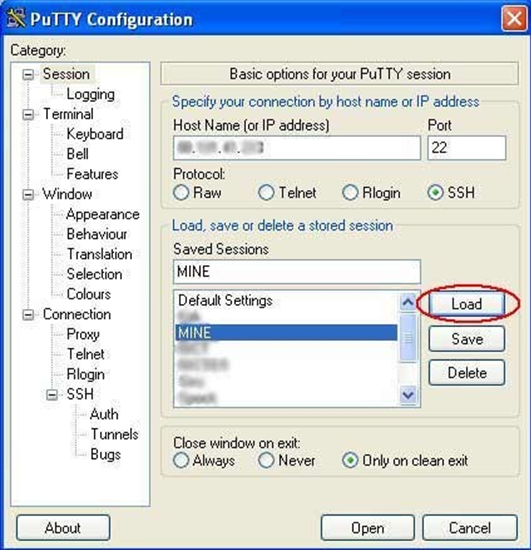
В данном примере используется mysql-сервер на Linux и один из графических MySQL-клиентов Workbench. Чтобы скачать MySQL Workbench, надо завести учетную запись на сервере Oracle. Будем считать, что используем версию Workbench под Windows. После запуска поступит предложение загрузить и установить дополнительное ПО с сайта Microsoft. После инсталляции появится программа MySQL Workbench. Рассмотрим, как настроить подключение MySQL Workbench через SSH-туннель под Windows.

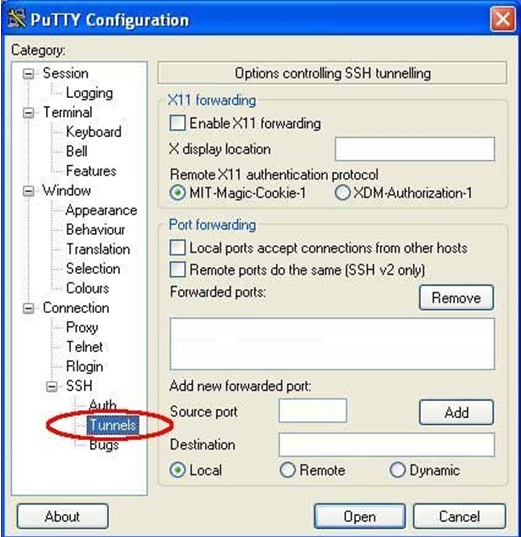
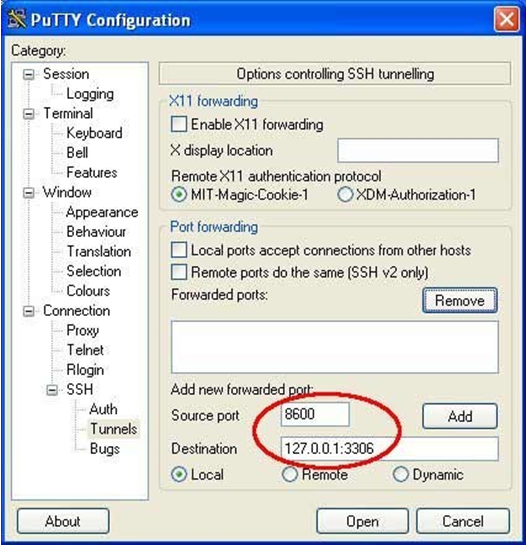
Локальный SSH-туннель, который мы будем использовать, позволяет пробросить локальный TCP-порт на порт удаленного сервера, доступный с сервера, к которому мы подключаемся по SSH. Будем соединять локальный порт (8600) с портом MySQL (3306) на том же сервере, к которому подключаемся по SSH. После настройки туннеля мы настраиваем подключение MySQL Workbench на соединение с локальным портом (8600), который будет присоединен через ssh-туннель к MySQL-порту удаленного сервера. Сам сервер будет считать, что подключение к нему выполняется локально, как если бы клиент располагался на нем самом. Одновременно решаем две задачи: защищаем соединение с сервером с помощью шифрования ssh и запрещаем удаленное подключение к MySQL по сети с помощью опции настроек сервера **bind-address=127.0.0.1**. В нашем примере будем использовать **Putty** —популярный ssh-клиент под Windows (<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>).

У нас уже есть ранее настроенное соединение с сервером MySQL, его имя — MINE. Выбираем его из списка соединений.

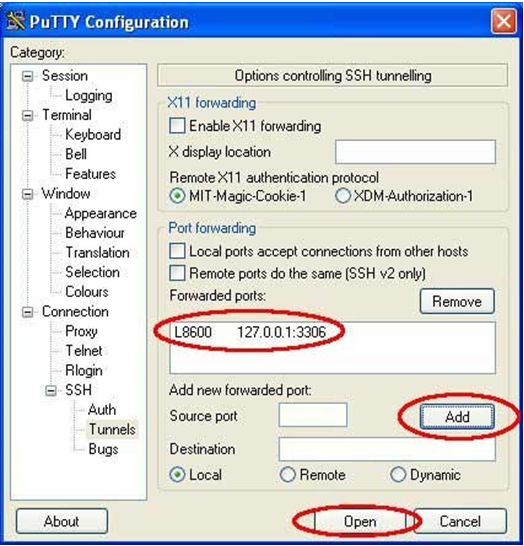


Затем кликаем **Load**.

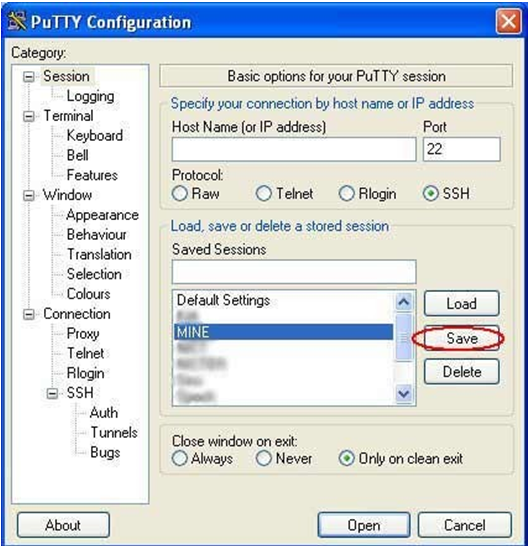


Переходим в раздел **Connections — SSH — Tunnels**. Заполняем информацию по соединению: 8600 — локальный порт; 127.0.0.1:3306 — хост и порт, с которым соединяется локальный порт на другой стороне туннеля.

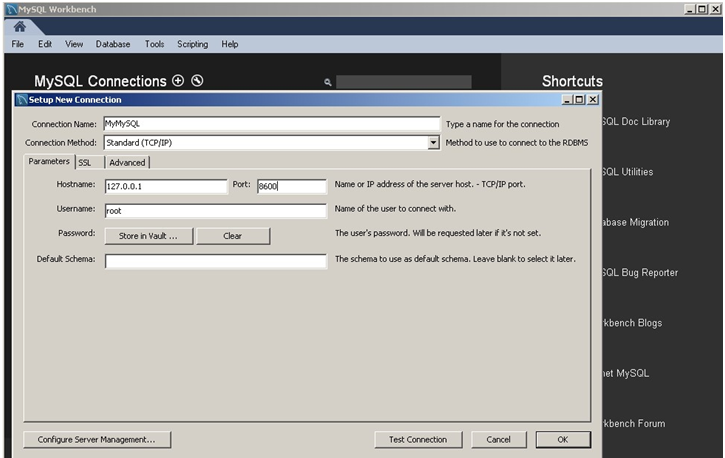
Нажимаем **Add**.



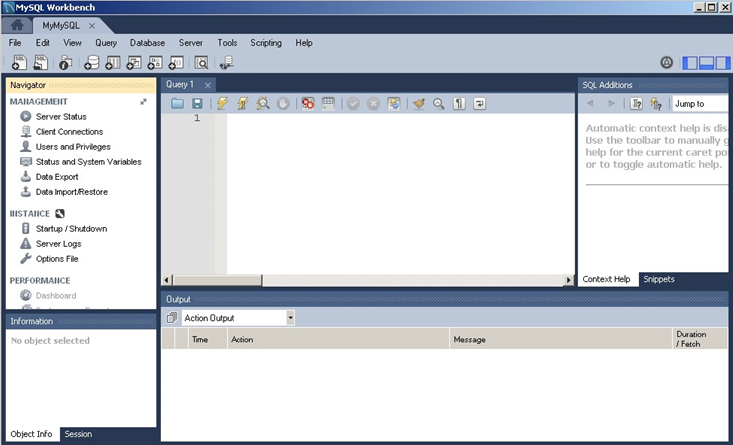
Возвращаемся в раздел со списком соединений (**Session**) и сохраняем настройки, кликнув **Save**.



Запускаем MySQL Workbench и создаем соединение со следующими параметрами: (**Hostname:127.0.0.1, Port: 8600**).



Открываем соединение: оно работает через ssh-туннель.



Если на клиентской системе, где вы запускаете MySQL Workbench, установлен Linux, создать ssh-туннель еще проще: это делается с помощью опции **-L** стандартного ssh-клиента.

|  |
| --- |
| ssh -L 8600:localhost:3306 your-mysql-server |

**Удаленное копирование файлов**

## scp

scp (Secure CoPy) — утилита для удаленного копирования файлов по сети. По поведению похожа на cp, для транспорта использует ssh. Если на сервере установлен ssh (например, openssh-server), то на него можно закачивать файлы с помощью scp, и точно также можно получать данные с помощью этой утилиты.

scp — разумная альтернатива FTP, позволяющая копировать файлы по защищенному каналу.

Копирование файла на удаленный сервер. Структура аналогична утилите cp:

scp <что\_копируем> <куда\_копируем>

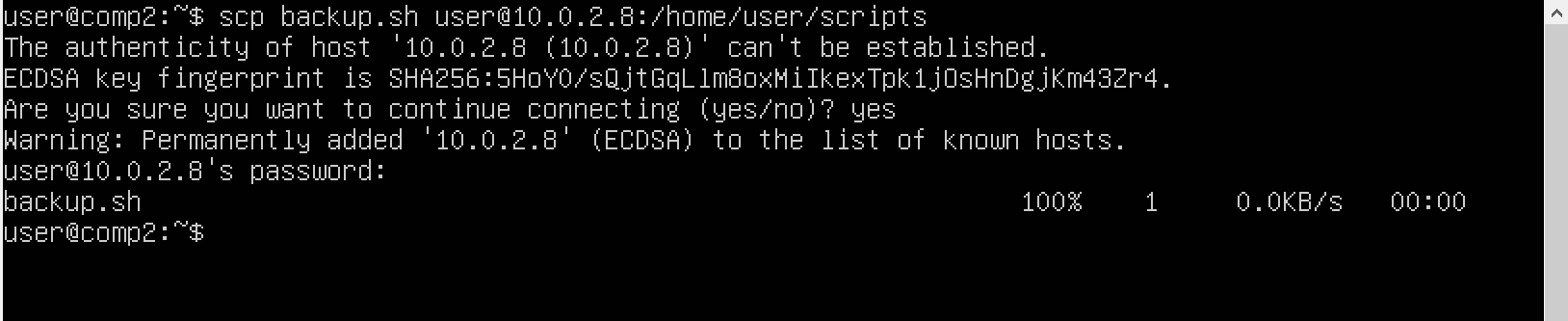
Можно обращаться по IP-адресу или по доменному имени

|  |
| --- |
| scp backup.sh user@server.com:/home/user/scripts |

или

|  |
| --- |
| scp backup.sh user@10.0.2.8:/home/user/scripts |

Понадобится ввести пароль



Происходит аутентификация по ssh, и обратите внимание, что у пользователя должны быть права на ту директорию, куда нужно закачать файл.

Если ssh сервер находится на нестандартном порту, то следует указать это с помощью ключа -P (ключ должен быть указан до указания путей)

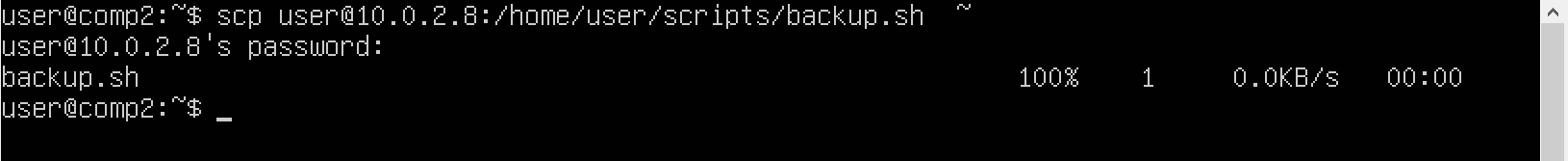
|  |
| --- |
| scp -P 12345 backup.sh user@server.com:/home/user/scripts |

или

|  |
| --- |
| scp -P 12345 backup.sh user@10.0.2.8:/home/user/scripts |

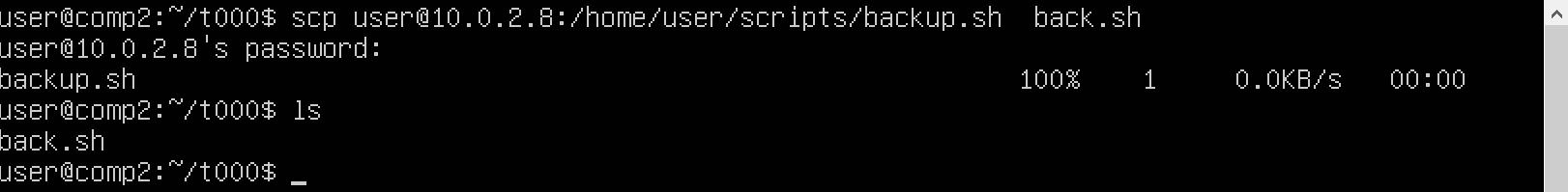
Скопировать с удаленной машины, указав куда копировать

|  |
| --- |
| scp [user@server.com](mailto:user@server.com):/home/user/scripts/backup.sh ~ |



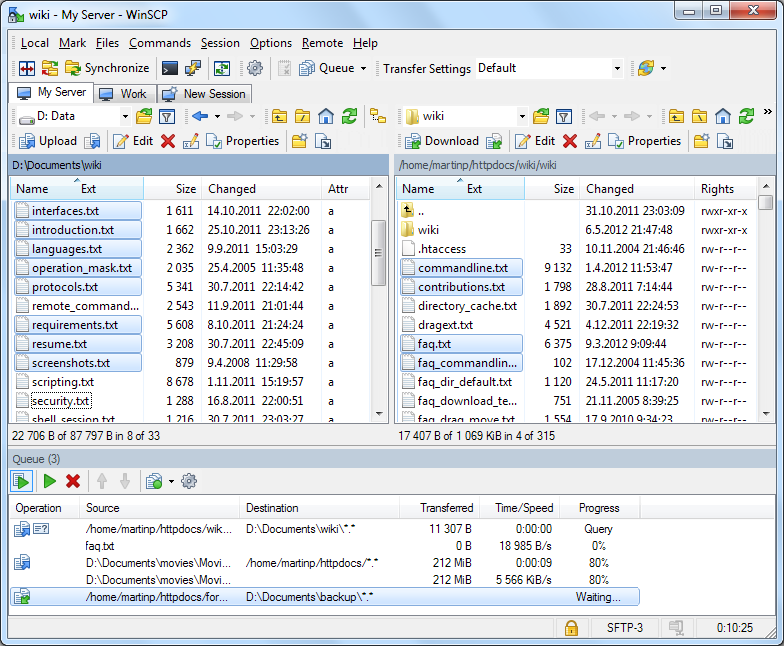
или можно указать новое имя

|  |
| --- |
| scp user@10.0.2.8:/home/user/scripts back.sh |



scp входит в состав GNU/Linux и других UNIX-подобных операционных систем.

Для Windows есть графический клиент WinSCP



## rsync

rsync — утилита не просто для копирования, но для синхронизации. rsync оптимизирована для передачи данных (использует сжатие), кроме того rsync передает не целиком файлы, а только разницу. rsync может применяться как локально, так и удаленно, а также может работать через ssh.

rsync отлично подходит для решения задач бэкапирования.

Установка:

|  |
| --- |
| # apt install rsync |

Формат команды

|  |
| --- |
| rsync опции источник приемник |

Опции rsync можно узнать с помощью команды

|  |
| --- |
| rsync --help |

их очень много.

Наиболее интересные:

* -v, --version — Выводить подробную информацию о процессе копирования. **Полезно использовать, когда rsync выполняется вручную.**
* --stat - показать статистику передачи, **полезная опция при ручном выполнении rsync.**
* -a, --archive — **Режим архивирования (аналог -rltgoD, то есть, копировать симлинки, сохранять время модификации, сохранять владельца группы, сохранять владельца файла, сохранять файлы устройств и специальные файлы).** Часто используемая возможность.
* -q, --quiet — Тихий режим, вывод сообщений только об ошибках
* -c, --checksum — Пропуск на основе контрольной суммы, а не на оценке даты и времени
* -D — сохранять файлы устройств и специальные файл, то же самое что --devices --specials
* --devices — сохранять файлы устройств (только для рута)
* --specials — сохранять специальные файлы
* -r, --recursive — Копировать рекурсивно заходя в директории
* -R, --relative — использовать относительные пути
* -b, --backup — создавать резервные копии (сохранять иерархию и использовать суффикс для обозначения, что это бэкапы)
* -u, --update – если перезаписываемый файл имеет более новую дату, то не перезаписывать
* -l, --links - копировать символьные ссылки как симлинки
* -L, --copy-links — преобразовывать симлинки в файлы, основываясь на содержимом файлов, куда ссылаются симлинки
* -H, --hard-links — сохранять при копировании хардлинки
* -p, --perms — сохранять при копировании права для файлов
* -g, --group - сохранять группу файла
* -t, --times - сохранять при копировании исходное время редактирования файла
* -z, --compress — сжимать файлы при передачи
* --delete - удалять файлы места назначения, которых нет в источнике
* --exclude=PATTERN - исключить файлы, соответствующие шаблону PATTERN
* --progress - выводить прогресс передачи файла
* -e — указать используемый транспорт, например ssh. Достаточно полезная опция, так как на практике rsync и используется по ssh.

Пример использования (используется ssh с доступом по нестандартному порту 12345/TCP)

|  |
| --- |
| sudo rsync -ae "ssh -p 12345" /var/www develop@10.0.2.8:/home/develop/backup |

# Практическое задание

1. Настроить ssh-аутентификацию по ключу.
2. nc

# Дополнительные материалы

1. <https://habrahabr.ru/post/319158/>
2. <https://ru.wikipedia.org/wiki/SSH>
3. <http://greendail.ru/node/474>
4. <http://help.ubuntu.ru/wiki/bind9>
5. <https://www.youtube.com/watch?v=2MLE7TIljlY>
6. <http://rus-linux.net/MyLDP/sec/generiruem-kluchi-ssh.html>
7. <https://lode.in/administration/generatsiya-ssh-klyuchej-s-putty-i-nastrojka-avtorizatsii-po-zakry-tomu-klyuchu-ssh.html>
8. <https://www.8host.com/blog/ustanovka-i-nastrojka-vnc-v-ubuntu-16-04/>

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы.

1. <https://ru.wikipedia.org/wiki/SELinux>
2. <http://www.clinuxworld.com/security/155-selinux>
3. <http://docscom.ru/blog/nix/84.html>
4. <https://linuxsoid.club/setup-use-acl-linux>
5. <http://rus-linux.net/MyLDP/sec/generiruem-kluchi-ssh.html>
6. Установка и настройка **ssh**: <http://a-rm.ru/materials/unix-servera/ustanovka-i-nastroyka-ssh-ubuntu>
7. Подробно об **ssh**: <https://habrahabr.ru/post/122445/>
8. <https://ru.wikipedia.org/wiki/WinSCP>
9. <https://ru.wikipedia.org/wiki/WinSCP#/media/File:WinSCP-5.5.2.png>