**Lesson 5 SSH generate**

SSH-ключ **Secure Shell (безопасная оболочка)**— это специальный код, который позволяет удалённому компьютеру понять кто вы есть и какими правами на этом компьютере обладаете.

## **Как работает SSH-ключ**

SSH-ключ разделён на две части. Одна часть называется приватной и должна всегда храниться только на вашем компьютере. Вторая часть ключа называется публичной и эту часть нужно копировать на другие компьютеры. При подключении к удалённому компьютеру он сравнивает публичную часть, которую вы ему дали с приватной частью, которая хранится у вас. Если части ключа совпадают, то вы получаете доступ к удалённому компьютеру (на самом деле всё гораздо сложнее, но принцип работы именно такой). На вашем компьютере может быть создано сколько угодно SSH-ключей. То есть вы можете использовать один SSH-ключ для доступа к сотне компьютеров или для каждого удалённого компьютера создавать отдельный ключ.

## **Защита SSH ключа**

Вы никогда и ни при каких обстоятельствах не должны никому передавать приватную часть своего ключа. Чтобы эту приватную часть нельзя было украсть можно её зашифровать, тогда даже если она попадёт в чужие руки её не смогут использовать. Если вы зашифруете приватную часть SSH-ключа, то каждый раз при его использовании вам нужно будет вводить ключ шифрования, называемый passphrase. Есть общее правило безопасности: если ключ используется не скриптом, а человеком, то он должен быть зашифрован.

## **Как создать SSH-ключ под Windows**

Windows не имеет встроенной поддержки SSH протокола. Поэтому, для создания SSH-ключа нужна специальная программа. Подойдёт Git Bash, которая [устанавливается вместе с Git под Windows](https://blog.vistro.ru/ide/git/how-to-install-git-on-windows/).

Инструкция по созданию SSH-ключа:

1. запустите Git Bash (используется только в ОС Windows, на Linux & MacOS есть свои терминалы);
2. введите команду ssh-keygen;
3. программа предложит выбрать место, куда сохранить ключ. По умолчанию создаётся каталог .ssh в папке текущего пользователя Windows. Например, у меня это C:\Users\Denis\.ssh. Этот вариант подходит, поэтому нажимайте Enter;
4. программа запросит passphrase для шифрования ключа. Если просто нажать Enter, то приватная часть ключа не будет зашифрована. Помните, что если SSH-ключ используется не скриптом, то он должен быть зашифрован. Относитесь к passphrase как к паролю, его нужно запомнить;
5. после нажатия Enter, программа попросит ввести passphrase повторно. Если вы не шифруете ключ, то ничего не вводите. Следующее нажатие Enter завершит создание ключа.

Теперь в каталоге .ssh созданы два файла:

* id\_rsa — приватный ключ (никому не передавать);
* id\_rsa.pub — публичный ключ (нужно копировать на другие компьютеры).

Git — система управления версиями с распределенной архитектурой. В отличие от некогда популярных систем вроде CVS и Subversion (SVN), где полная история версий проекта доступна лишь в одном месте, в Git каждая рабочая копия кода сама по себе является репозиторием. Это позволяет всем разработчикам хранить историю изменений в полном объеме.

There are many color codes, but you will often see only the 7 colors listed below:

* **White (No color code):** Regular File or Normal File
* **Blue:** Directory
* **Bright Green:** Executable File
* **Bright Red:** Archive file or Compressed File
* **Magenta:** Image File
* **Cyan:** Audio File
* **Sky Blue:** Symbolic Link File

Подсказки по уроку:

**Статья про ГИТ:**

<https://habr.com/ru/post/541258/>

**генерируем ключ SSH:**

ssh-keygen

**Посмотреть доступ к файлам с ключами:**

ls -la .ssh

**Имя сервера:**

ssh -i ~/.ssh/id\_rsa [ec2-user@linux.telran-edu.de](mailto:ec2-user@linux.telran-edu.de)

**Добавление ключа редактором nano:**

nano .ssh/authorized\_keys

**показать ключ:**

cat .ssh/id\_rsa.pub

**Выйти из nano:**

ctrl+x

**Процессы.**

Что такое процессы в терминале?

Любая программа, запущенная в ОС будет процессом. Иногда эти процессы виснут, ведут себя странно и их нужно отслеживать.

Как же посмотреть список процессов?

Я сейчас зайду на наш учебный сервер

и введу команду **top**

**top** (table of processes) — консольная команда, которая выводит список работающих в системе процессов и информацию о них. По умолчанию она в реальном времени сортирует их по нагрузке на процессор.

так же мы можем посмотреть количество процессов через команду

**ps -ef**

Вот значение основных колонок в выводе утилиты:

* **UID** - имя пользователя, от имени которого работает процесс;
* **PID** - идентификатор пользователя;
* **PPID** - идентификатор родительского процесса пользователя;
* **C** - расходование ресурсов процессора, в процентах;
* **SZ** - размер процесса;
* **RSS** - реальный размер процесса в памяти;
* **PSR** - ядро процессора, на котором выполняется процесс;
* **STIME** - время, когда процесс был запущен;
* **TTY** - если процесс привязан к терминалу, то здесь будет выведен его номер;
* **TIME** - общее время выполнения процесса (user + system);
* **CMD** - команда, которой был запущен процесс, если программа не может прочитать аргументы процесса, он будет выведен в квадратных скобках;

Чтобы посмотреть список процессов в виде дерева, и понимать какой процесс имеет какие дочерние процессы, выполните команду:

**ps -efH**

Для просмотра списка процессов с потоками используйте опцию -L:

**ps -efL**

Здесь появятся ещё две дополнительных колонки:

* **LWP - Это сокращение от LightWeight Process. Идентификатор потока;**
* **NLWP - количество потоков у этого процесса.**

### **УТИЛИТА TOP**

Программа позволяет интерактивно просматривать список запущенных процессов Linux. Чтобы вывести список процессов Linux выполните команду:

**top**

Колонки, которые выводит программа очень похожи на ps:

* **PID -** идентификатор процесса;
* **USER -** имя пользователя, от имени которого выполняется процесс;
* **PR -** приоритет планировщика, установленный для процесса;
* **NI -** рекомендуемый приоритет процесса. Это значение можно менять, может не совпадать с реальным приоритетом планировщика;
* **VIRT -** всё, что находится в памяти, используется или зарезервировано для использования;
* **RES -** всё, что находится в оперативной памяти и относится к процессу. Расшифровывается как Resident Memory Size, указывается в килобайтах;
* **SHR -** часть памяти из RES, которую занимают ресурсы, доступные для использования другим процессам. Расшифровывается - Shared Memory Size.
* **S -** состояние процесса: D - ожидает завершения операции, R - запущен, S - спит, T - остановлен, t - остановлен отладчиком, Z - зомби;
* **%CPU -** процент использования ресурсов процессора;
* **%MEM -** процент использования ресурсов оперативной памяти на основе колонки RES;
* **TIME -** общее процессорное время, которое процесс использовал с момента запуска;
* **COMMAND** - команда, с помощью которой был запущен процесс.
* **Проце́сс-зо́мби, зо́мби** ([англ.](https://ru.wikipedia.org/wiki/%25D0%2590%25D0%25BD%25D0%25B3%25D0%25BB%25D0%25B8%25D0%25B9%25D1%2581%25D0%25BA%25D0%25B8%25D0%25B9_%25D1%258F%25D0%25B7%25D1%258B%25D0%25BA) zombie process, [англ.](https://ru.wikipedia.org/wiki/%25D0%2590%25D0%25BD%25D0%25B3%25D0%25BB%25D0%25B8%25D0%25B9%25D1%2581%25D0%25BA%25D0%25B8%25D0%25B9_%25D1%258F%25D0%25B7%25D1%258B%25D0%25BA) defunct process) — дочерний процесс в [Unix](https://ru.wikipedia.org/wiki/Unix)-системе, завершивший своё выполнение, но ещё присутствующий в списке процессов [операционной системы](https://ru.wikipedia.org/wiki/%25D0%259E%25D0%25BF%25D0%25B5%25D1%2580%25D0%25B0%25D1%2586%25D0%25B8%25D0%25BE%25D0%25BD%25D0%25BD%25D0%25B0%25D1%258F_%25D1%2581%25D0%25B8%25D1%2581%25D1%2582%25D0%25B5%25D0%25BC%25D0%25B0), чтобы дать родительскому процессу считать код завершения.

**Для того чтобы сделать вывод программы цветным, нажмите Z**

Чтобы вывести дерево процессов Linux нажмите сочетание клавиш **Shift+V:**

Для отображения потоков нажмите **Shift + H:**

Выйдем из TOP нажмем **ctrl+z**

и введем:

**ps -ef | wc -l**

Так мы посмотрим сколько процессов запущено в цифрах.

И мы можем отфильтроваться по слову ssh:

**ps -ef | grep ssh**

Тут мы видим наш процесс, который был запущен нашей системой:

**root 4744 1 0 Feb01 ? 00:00:03 /usr/sbin/sshd -D**

Это значит, что система ожидает, когда к ней подключатся другие пользователи по ssh.

И мы видим идентификатор. После слова root.

Далее первый процесс запускает второй:

**root 4744 1 0 Feb01 ? 00:00:03 /usr/sbin/sshd -D**

**root 19727 4744 0 12:08 ? 00:00:00 sshd: ec2-user [priv]**

И тут мы видим, что идентификаторы процесса совпадают.

Итак. Теперь о наших процессах. Если мы сами что-то запускаем на сервере и мы не является администраторами, то мы можем завершать процессы командой **kill**

Всем сделать аккаунт на:

**github.com**

Пароль держите при себе, желательно на рабочем столе.