**Практика. Как защитить свой сервер**

OpenSSH-сервер. Безопасность HTTPS: сертификаты, Let's Encrypt. Продвинутая настройка встроенного фаервола iptables. Как настроить безопасную работу и проверить, не завелся ли на сервере вредитель.

**Оглавление**

[Введение](#_vqnwkm1ttzvn)

[Подготовка тестового сервера](#_xov2grg5wgeq)

[Подготовим SSH-доступ](#_ls2tnw6r3yqy)

[Защищаем пользователей](#_x6767vwsjz41)

[Создаем нового](#_ae6tgxejuji)

[Блокируем ненужных](#_ecuihd7cztmv)

[Защищаем SSH](#_8hxkdc15uqqn)

[Создание и настройка пары ключей в Windows](#_jsxem48eapv)

[Отключаем возможность логиниться руту, и доступ по паролям](#_fuv5ztchmdky)

[Защищаем HTTP](#_yvg72cdvrj1d)

[TLS и SSL-сертификаты](#_5aahx6cy6gh6)

[Проверка mod\_ssl и доступ по HTTPS](#_9dshmaato3vq)

[Настройка сайта с самоподписанным сертификатом](#_qxuxy7a6gesu)

[Создание сертификата с помощью Let’s Encrypt](#_7m921mieg8vd)

[Тонкая настройка iptables](#_gmbs1vmztujs)

[Доступ по ssh](#_ufgtcy4fjde1)

[Доступ по http и https](#_60vuid9uv4aw)

[Другие полезные утилиты](#_a4adfafpqjyx)

[Практическое задание](#_3tfrjxxltv85)

# 

# Введение

Настраивать безопасный удаленный доступ будем на нашем подготовленном виртуальном сервере в облаке GCP.

Но если у вас нет возможности запустить облако GCP, можно приобрести виртуальный сервер VDS, например здесь: <https://firstbyte.ru/?from=624>.

# Подготовка тестового сервера

Если вы будете работать с виртуальным сервером в облаке GCP или уже приобрели себе VDS, то смело пропускайте этот раздел.

В нем мы будем создавать имитацию сервера VDS (виртуального выделенного сервера) с помощью описанных ниже команд. Здесь и далее подразумевается, что вы работаете под Ubuntu Linux (настройки в разных ОС могут отличаться).

ВАЖНО! Не делайте так на настоящей машине. По большей части это не защита, а наоборот — создание уязвимостей, которые в следующих разделах мы будем закрывать.

Чтобы сымитировать работу с удаленным сервером, нужно разблокировать суперпользователя и назначить ему пароль. По умолчанию в Ubuntu пользователь **root** заблокирован и не имеет пароля. Чтобы можно было залогиниться пользователем, нужно назначить пароль и разблокировать его.

Сначала назначим пароль:

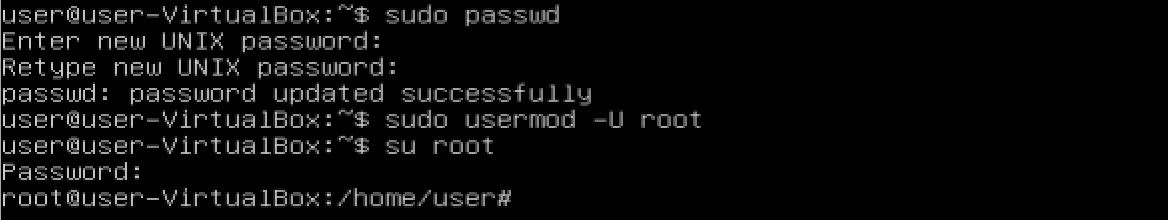
|  |
| --- |
| $ sudo passwd |

Разблокируем пользователя:

|  |
| --- |
| $ sudo usermod -L root |

Проверим, что пароль работает:

|  |
| --- |
| $ su root |



Теперь нужно установить **ssh server**:

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

Проверяем, что **openssh-server** слушает порты:

|  |
| --- |
| # netstat -ntpl |

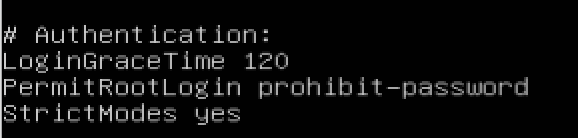


Также для полноты эксперимента придется разрешить удаленный доступ пользователя по паролю.

Поправим файл:

|  |
| --- |
| # nano /etc/ssh/sshd\_config |

Пользователь **root** не может логиниться с помощью пароля:

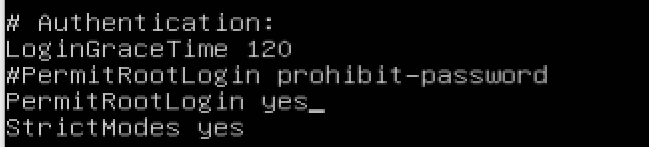


Закомментим строку:

|  |
| --- |
| #PermitRootLogin prohibit-password |

И добавим:

|  |
| --- |
| PermitRootLogin yes |



Сохраняем:

* в nano:
  + Ctrl-O
  + Enter
  + Ctrl-X
* в vi
  + :wq!

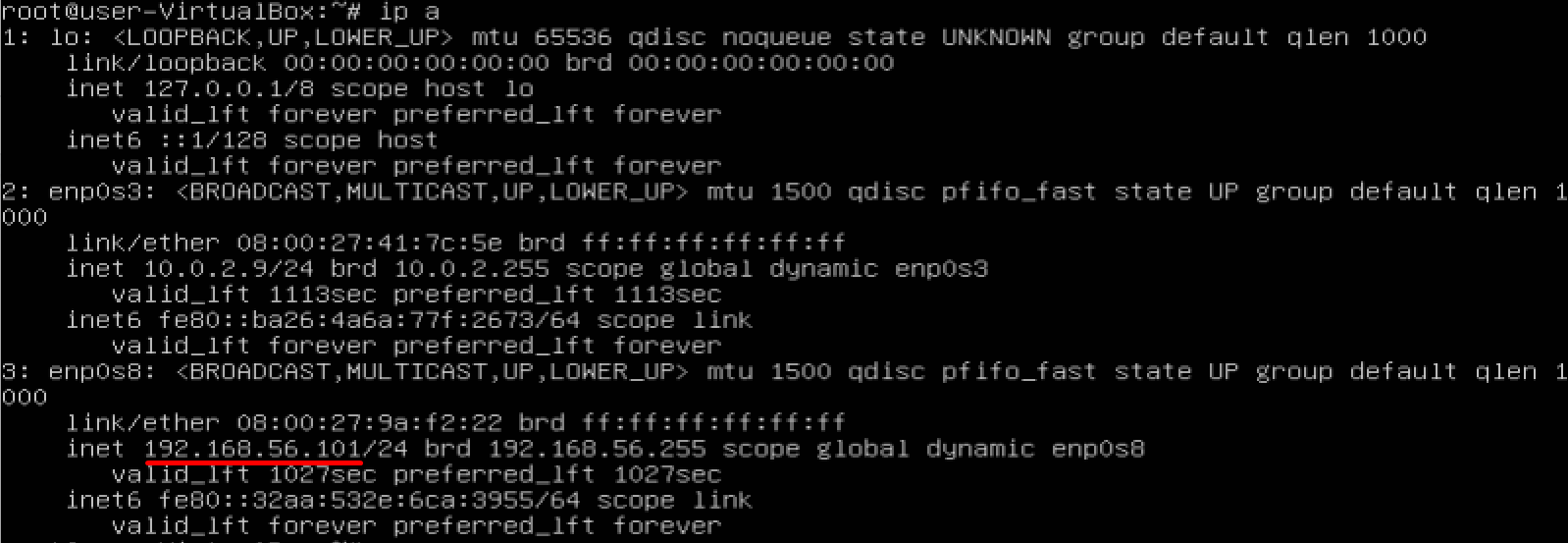
Рестартуем сервер командой **service**:

|  |
| --- |
| # service sshd restart |

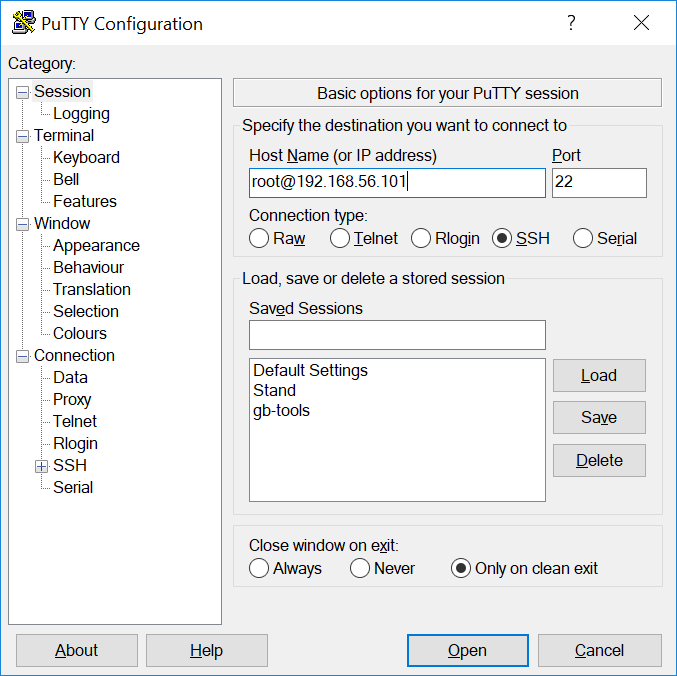
Либо, если вы уверены, что используется система инициализации **SystemD** (Ubuntu 16.0 и выше), то:

|  |
| --- |
| # systemctl restart sshd |

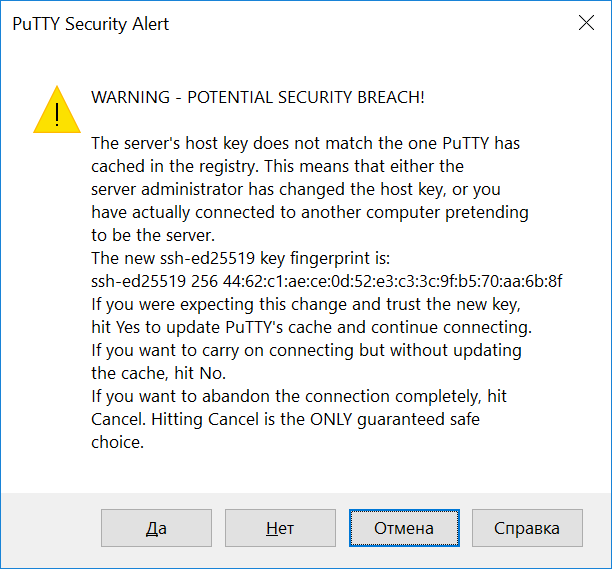
Посмотрим наш IP-адрес:



Можно подключиться с помощью **ssh** (macOS или Linux) либо **putty** (Windows):

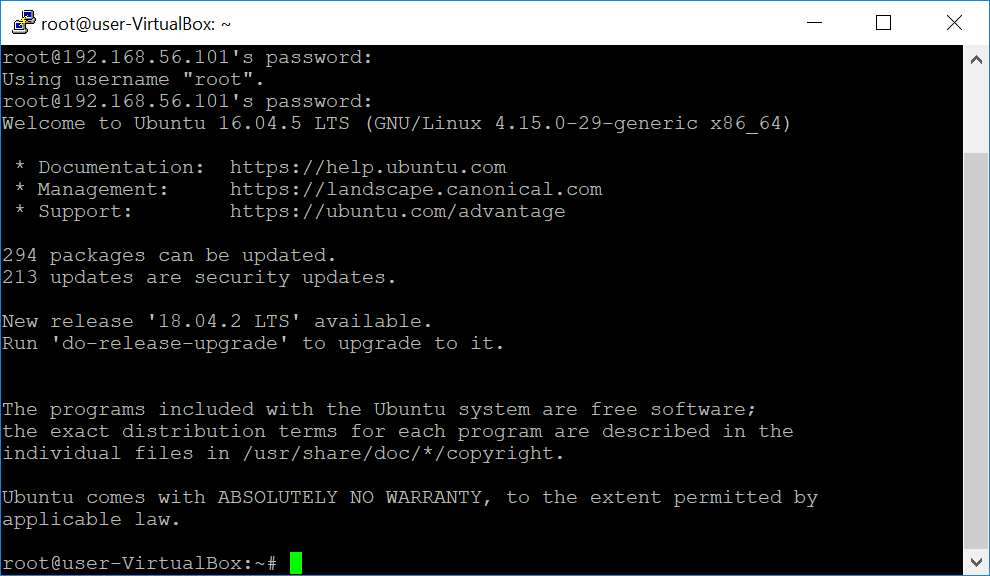


Предупреждение, что ключ сервера нам не известен:



Жмем «Да».

Вводим пароль. Зашли:



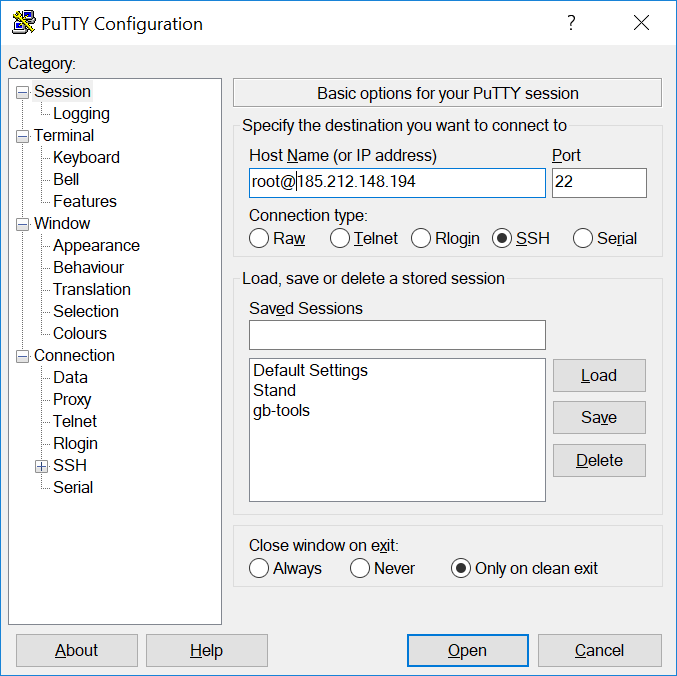
Это та конфигурация, с которой нам пришлось бы работать, если бы мы купили VDS.

## Подготовим SSH-доступ

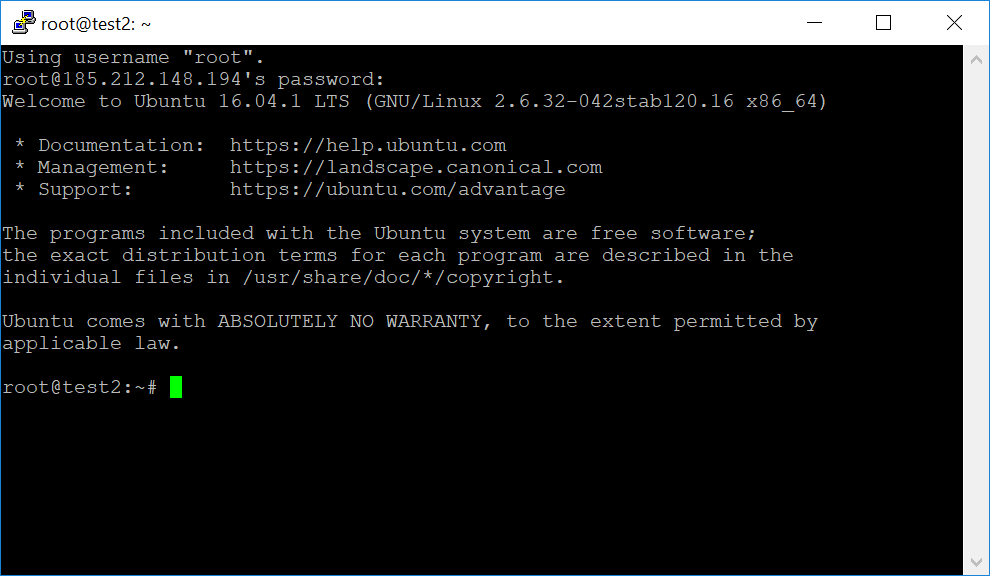
Когда мы заказываем VDS у провайдера, письмо с доступами приходит на почту:



Подключаемся по **ssh**:



Такой способ подключения небезопасен. Но пока у нас есть только он.



Подключились.

Теперь начинаем работать.

# 

# Защищаем пользователей

## Создаем нового пользователя

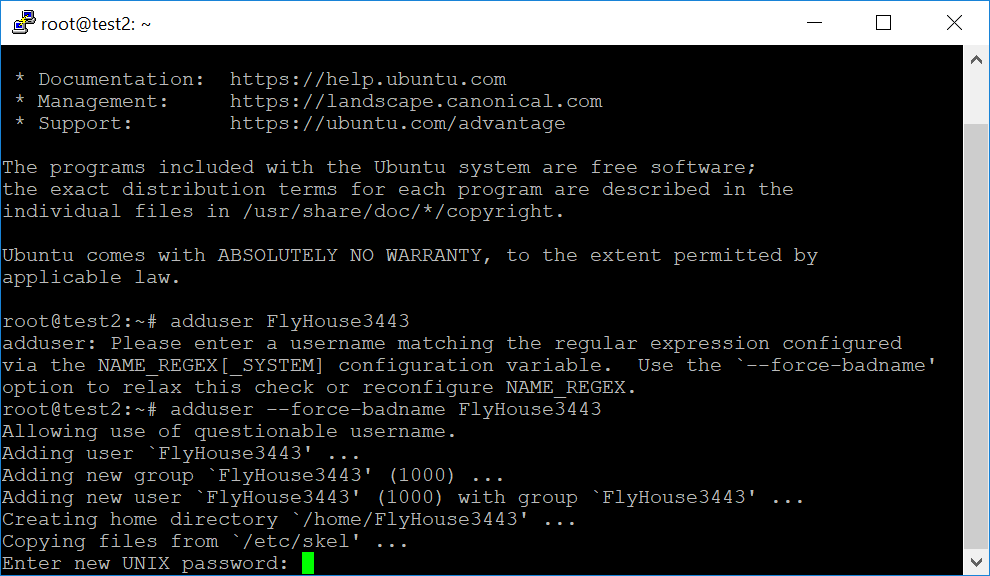
Теперь мы создадим пользователя, под которым будем работать. Этот пользователь (как и другие) не должен иметь словарного имени — то есть такие пользователи, как **user**, **deploy**, **admin** и подобные не подойдут.

Придумаем свое имя пользователя, например **FlyHouse3443**.

Система проверяет валидность имени, т. е. у нее есть свое представление о его формате. Но с помощью параметра **--force-badname** эту проверку возможно игнорировать. Создадим такого пользователя:

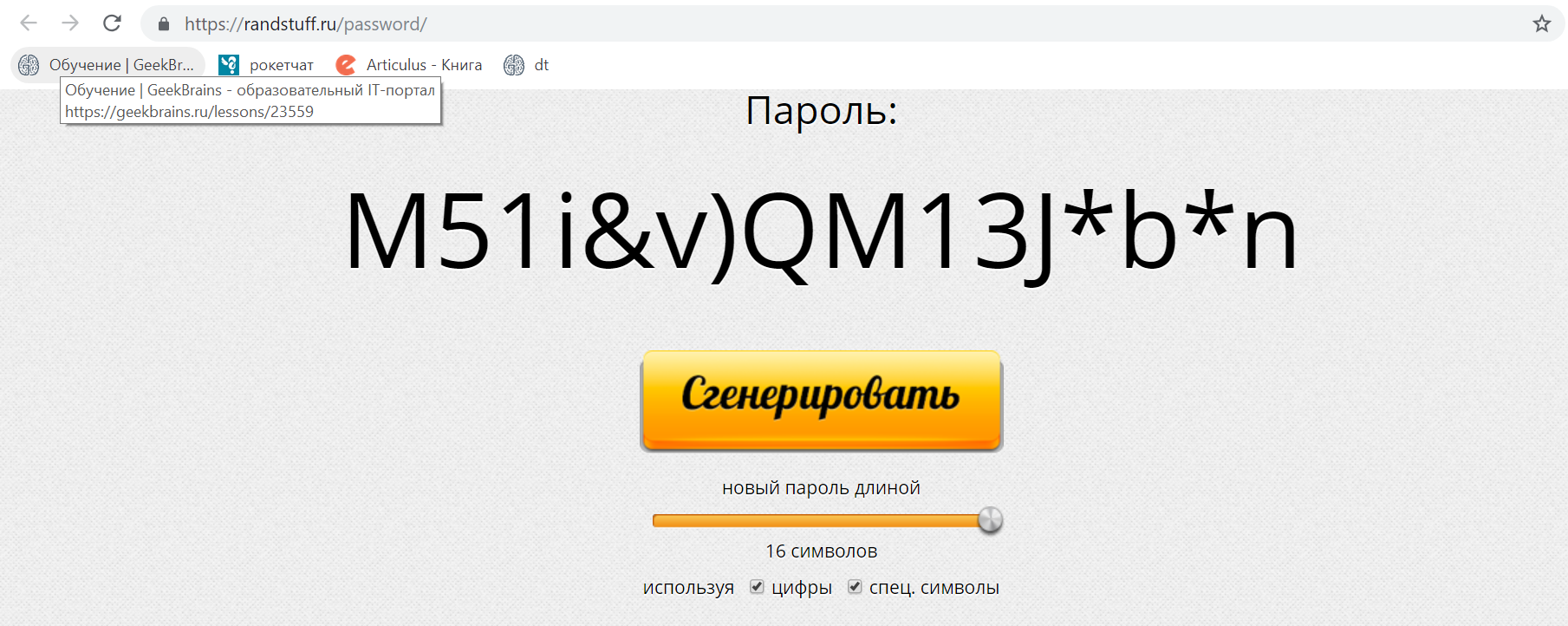
|  |
| --- |
| #adduser --force-badname FlyHouse3443 |

Когда вы будете его создавать, пожалуйста, не используйте это имя. Придумайте другое, но оригинальное.



Утилита попробует ввести пароль. Находим в поисковике генератор пароля.

Убедитесь, что сайт работает по https, а не по http.



Длина — 16 символов, цифры и спецсимволы использовать:

|  |
| --- |
| M51i&v)QM13J\*b\*n |

Возьмите часть пароля.

Сгенерируйте еще раз.

|  |
| --- |
| 4T&9T#8886np\*&sK |

Возьмите часть пароля, склейте в один.

|  |
| --- |
| M51i&v)Q86np\*&sK |

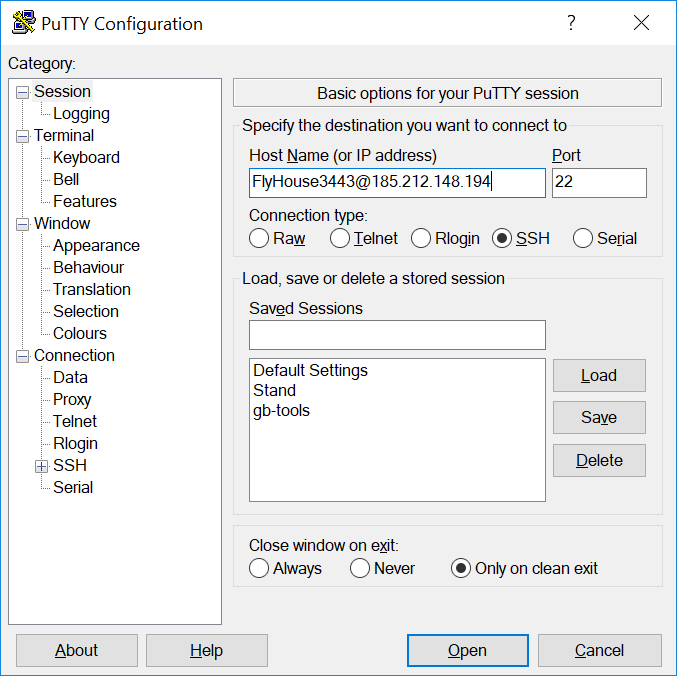
Замените несколько символов.

|  |
| --- |
| Z53i&v(Q!6Np\*&sT |

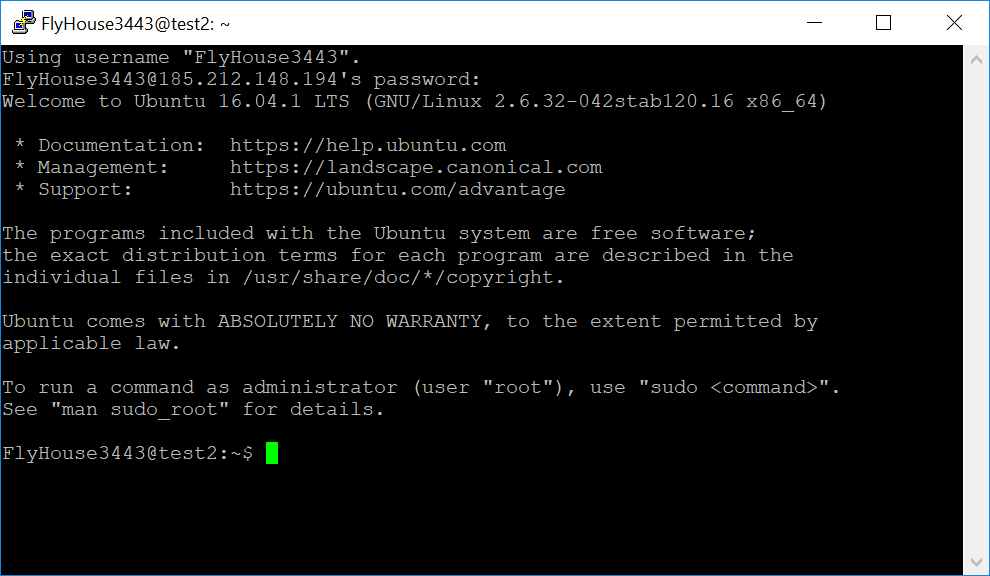
Добавим пользователя в группу **sudo**:

|  |
| --- |
| #usermod -a -G sudo FlyHouse3443 |

Теперь подключаемся созданным пользователем:

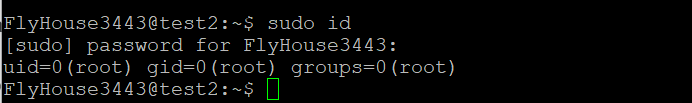


Вводим созданный пароль:



Проверяем, что мы можем работать с помощью **sudo**:

|  |
| --- |
| $ sudo id |



## Блокируем ненужных пользователей

Теперь можно заблокировать суперпользователя:

|  |
| --- |
| $ sudo usermod -L root |

А также любого другого стандартного пользователя, например **ubuntu**:

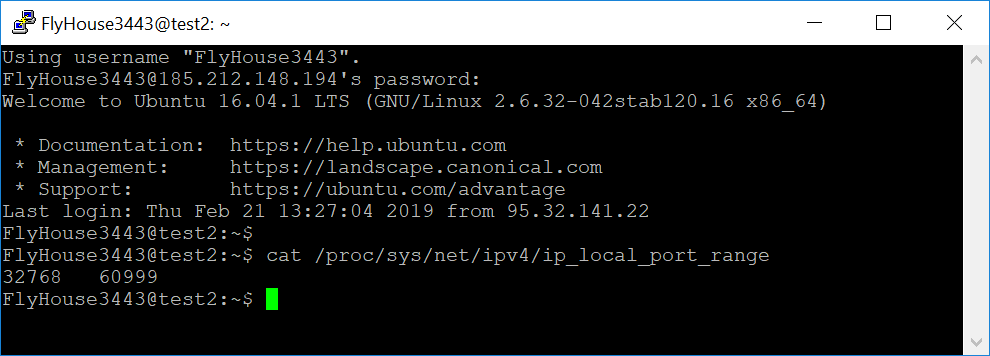
|  |
| --- |
| $ sudo usermod -L ubuntu |

# Защищаем SSH

Для дополнительной защиты сервера нам также следует поменять 22/TCP-порт службы SSH на нестандартный. Есть мнение, что так делать не стоит, чтобы не нарушать стандарты TCP/IP. Но в интернете идет война за белые IP-адреса, и целые ботнеты работают на захваченных машинах. На каждый белый IP-адрес будут идти атаки. Так что если не выполните описанные здесь действия, не удивляйтесь, когда при подключении по ssh по руту увидите что-то вроде 20 000 неудачных попыток залогиниться при входе в систему.

Посмотрим список случайных портов:

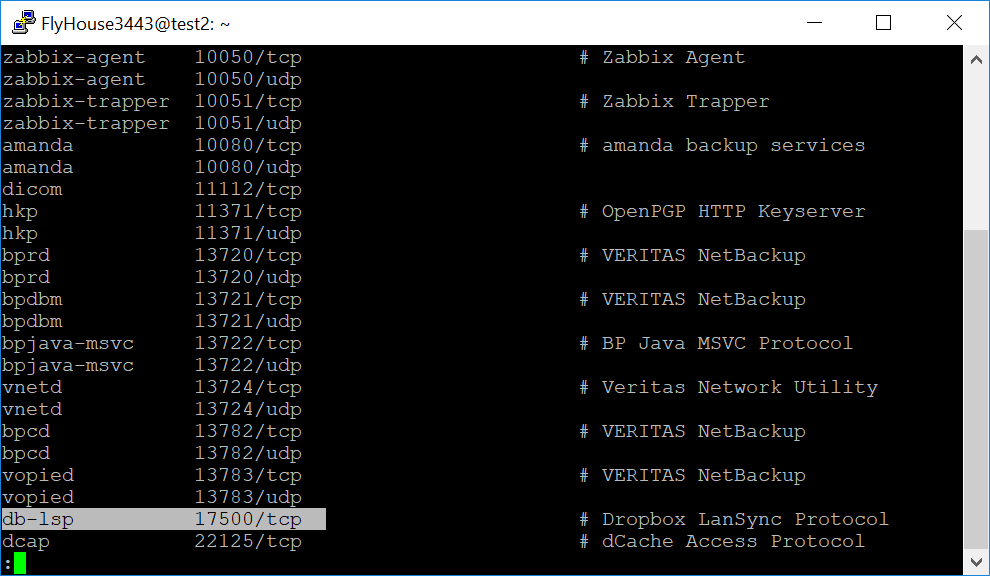
|  |
| --- |
| $ cat /proc/sys/net/ipv4/ip\_local\_port\_range |



Можем выбрать любой из случайных портов или

порт неиспользуемого сервера.

|  |
| --- |
| $ less /etc/services |



Например, 17500 или незадействованный 17501 или 17511.

Выходим из **less**, нажав **q**.

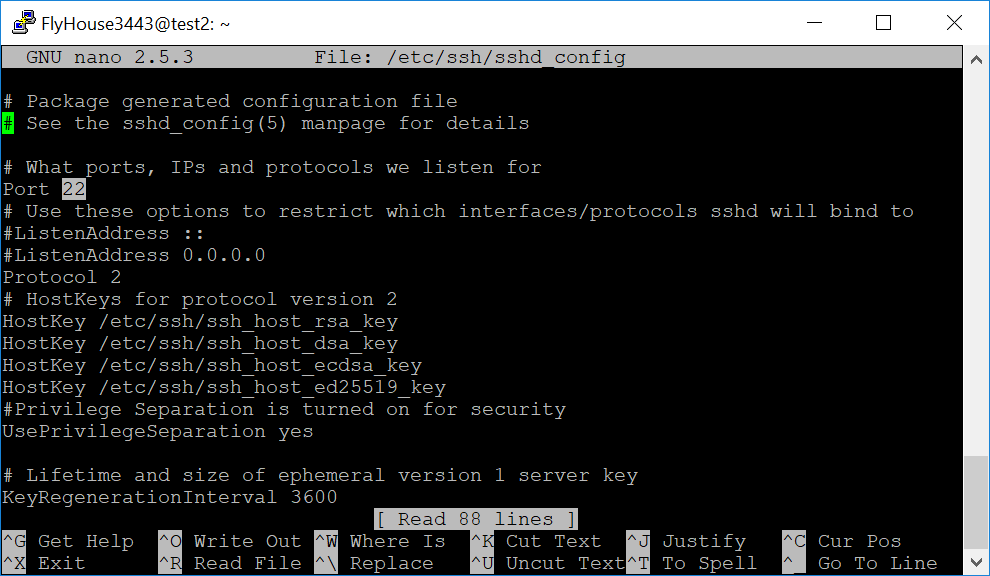
И решаем, что выберем динамический диапазон. Пусть это будет 39192 (а вы используйте другой номер).

Для редактирования используем **vi** или **nano**. Последний надо установить.

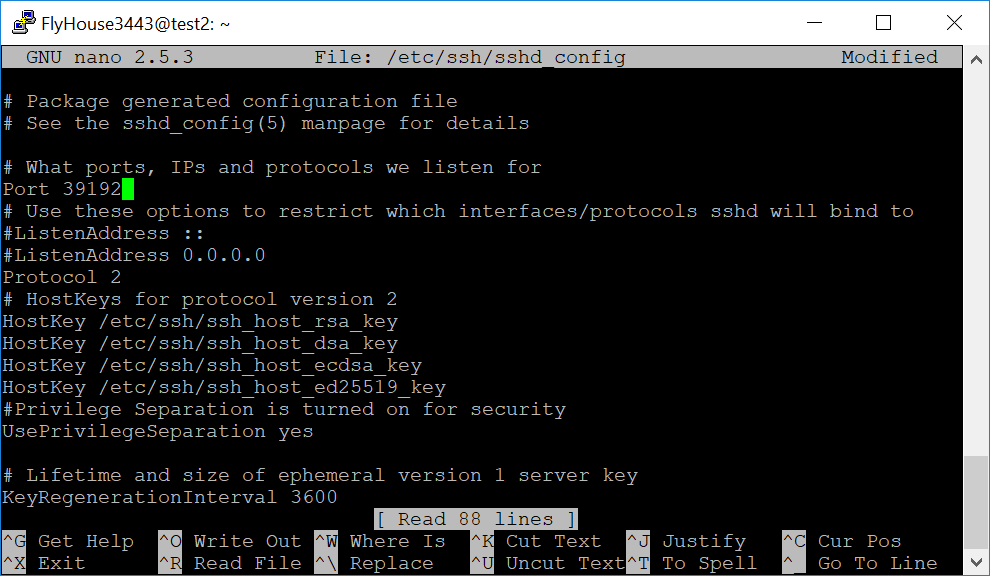
|  |
| --- |
| $ sudo apt update  $ sudo apt install nano |

Правим **/etc/ssh/sshd\_config**:

|  |
| --- |
| $ sudo nano /etc/ssh/sshd\_config |



Меняем 22 на выбранный 39192:



Ctrl-O, сохраняем: Enter. Ctrl-X — выходим.

Если же предпочитаете vi, то **:wq!**

Нужно перезапустить sshd-сервер.

Рестартуем сервер командой **service**:

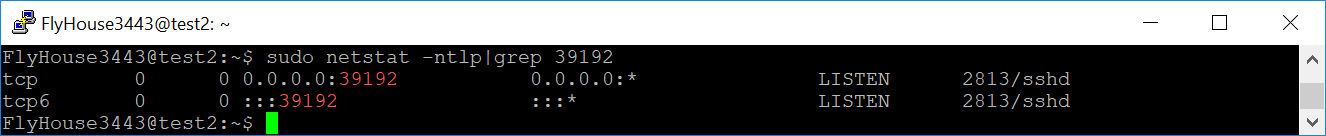
|  |
| --- |
| # service sshd restart |

Если вы уверены, что используется система инициализации SystemD (Ubuntu 16.0 и выше), то:

|  |
| --- |
| # systemctl restart sshd |

Проверяем:

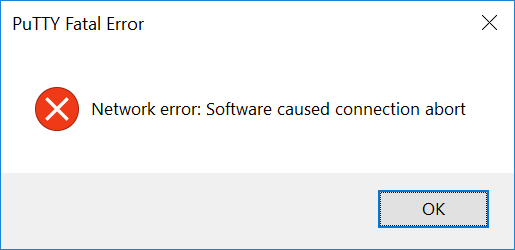
|  |
| --- |
| $ sudo netstat -ntlp|grep 39192 |



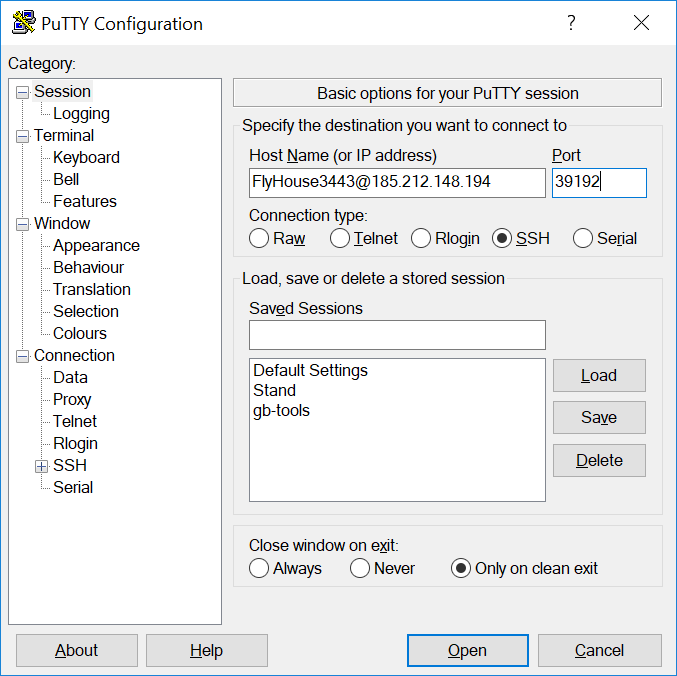
Обратите внимание, что все текущие сессии до закрытия так и останутся на 22/TCP.

Перезагружаем сервер и пробуем подключиться по 22-му порту и по новому (в нашем случае — 39192):

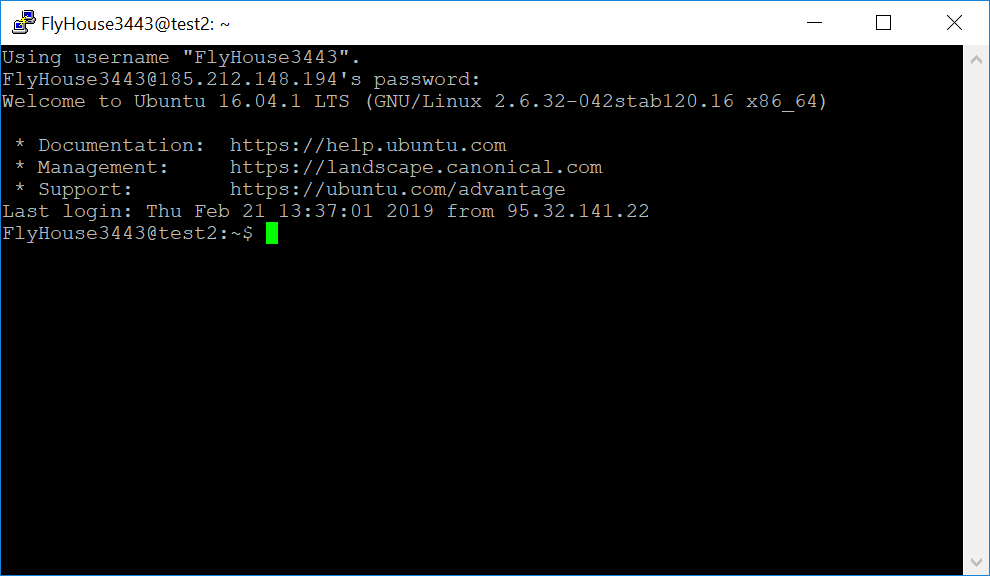
|  |
| --- |
| $ sudo systemctl reboot |



Подключаемся по новому порту:



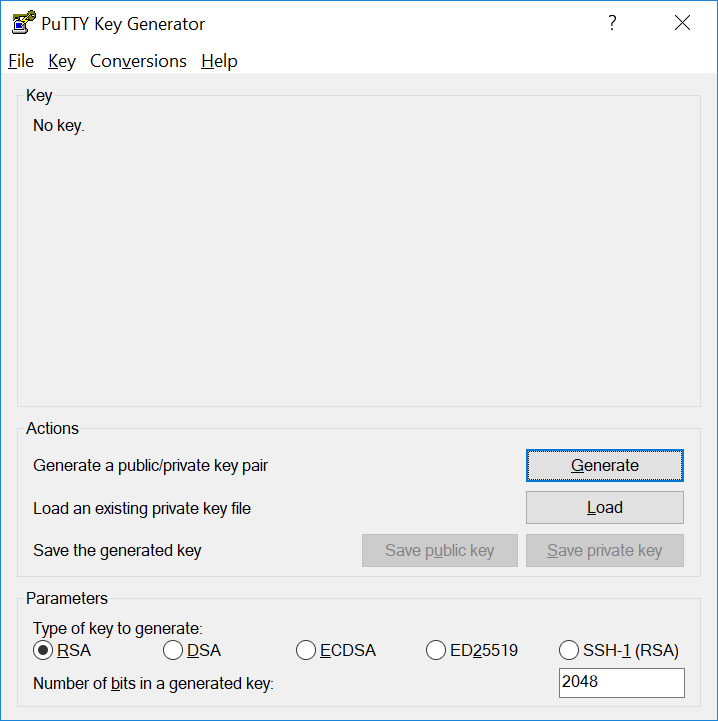
Мы подключились по паролю:



Теперь нужно сгенерировать ключ.

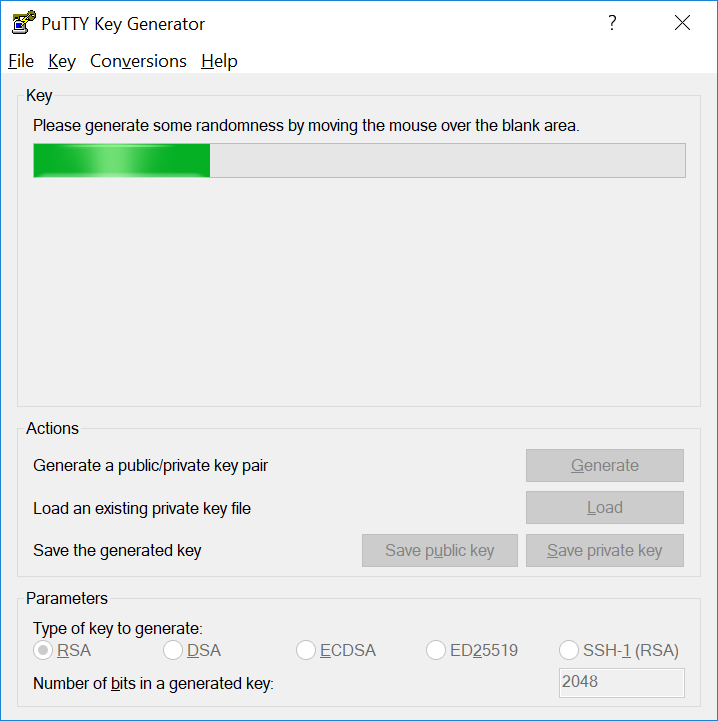
## Создание и настройка пары ключей в Windows

Понадобится **puttygen**:

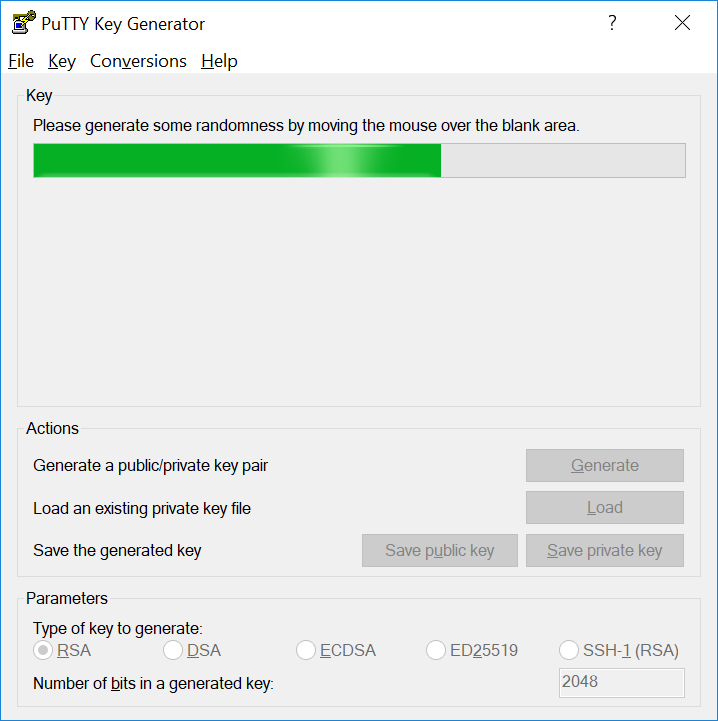


Жмем кнопку **Generate**.

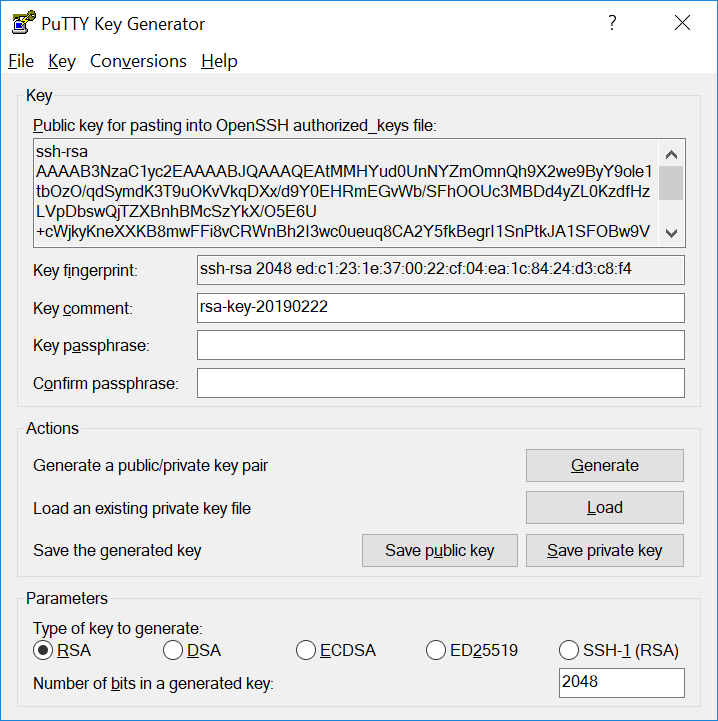
Для генерации хорошего ключа нужно, чтобы он основывался на случайных числах. Если числа не случайные, то по закономерностям можно подобрать ключ. Чтобы получить более-менее близкие к настоящим случайные числа, программа просит подвигать курсором мыши.



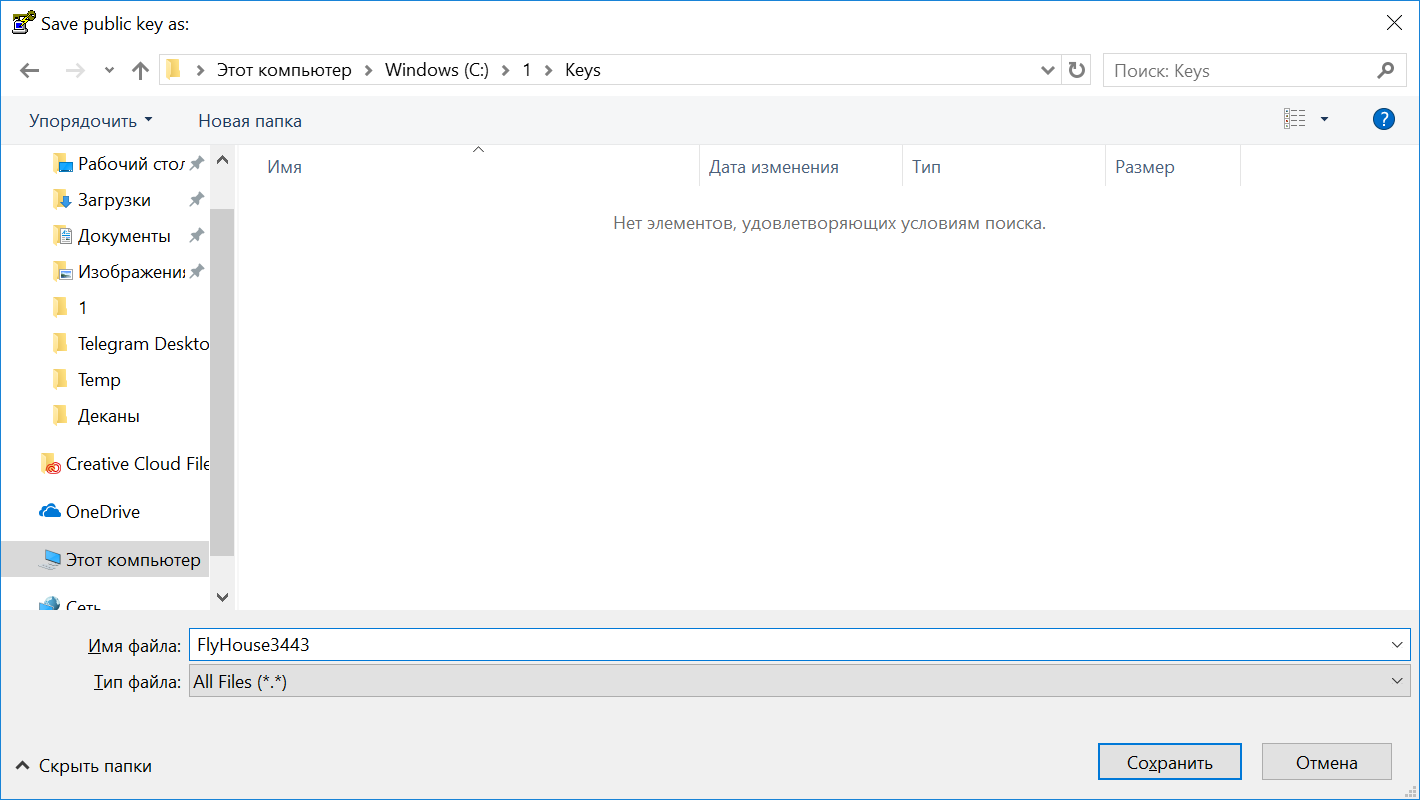
Двигаем, ключ создается:



Ключ сгенерирован:

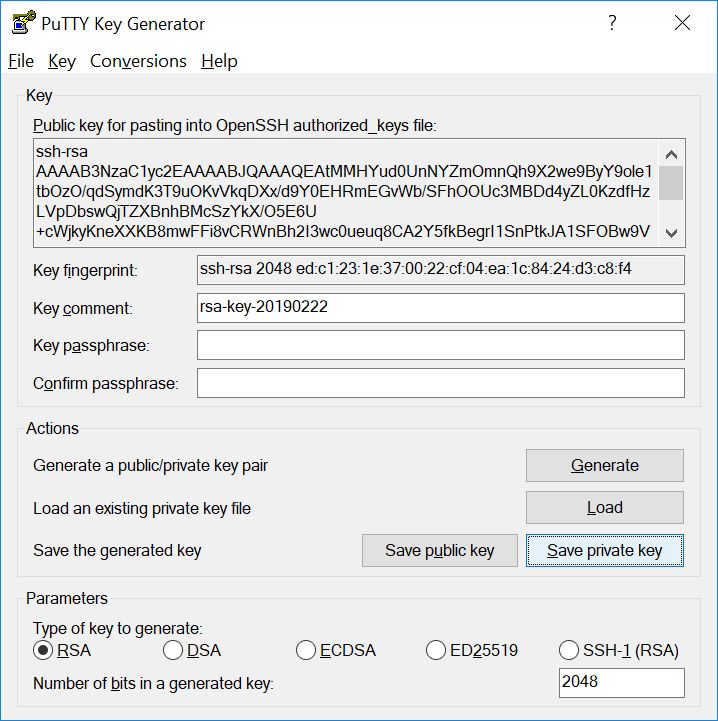


Сохраняем публичный ключ:

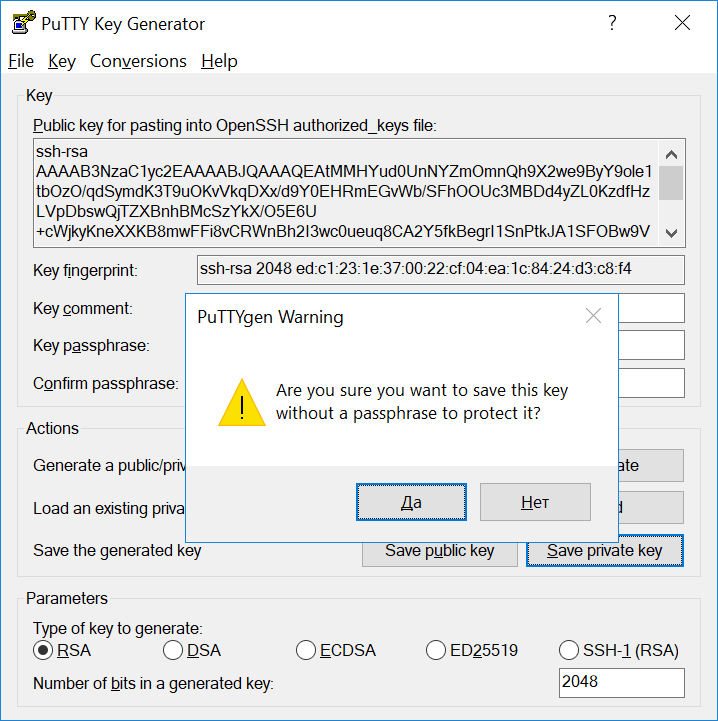


Обратите внимание, что **puttygen** сохраняет публичный (открытый) ключ без расширения. Это тот же ключ, который вы можете скопировать из поля ввода. Но чтобы работать с публичным ключом, надо сохранить приватный (закрытый) ключ. Это тот ключ, который не покидает того места, где был сгенерирован.

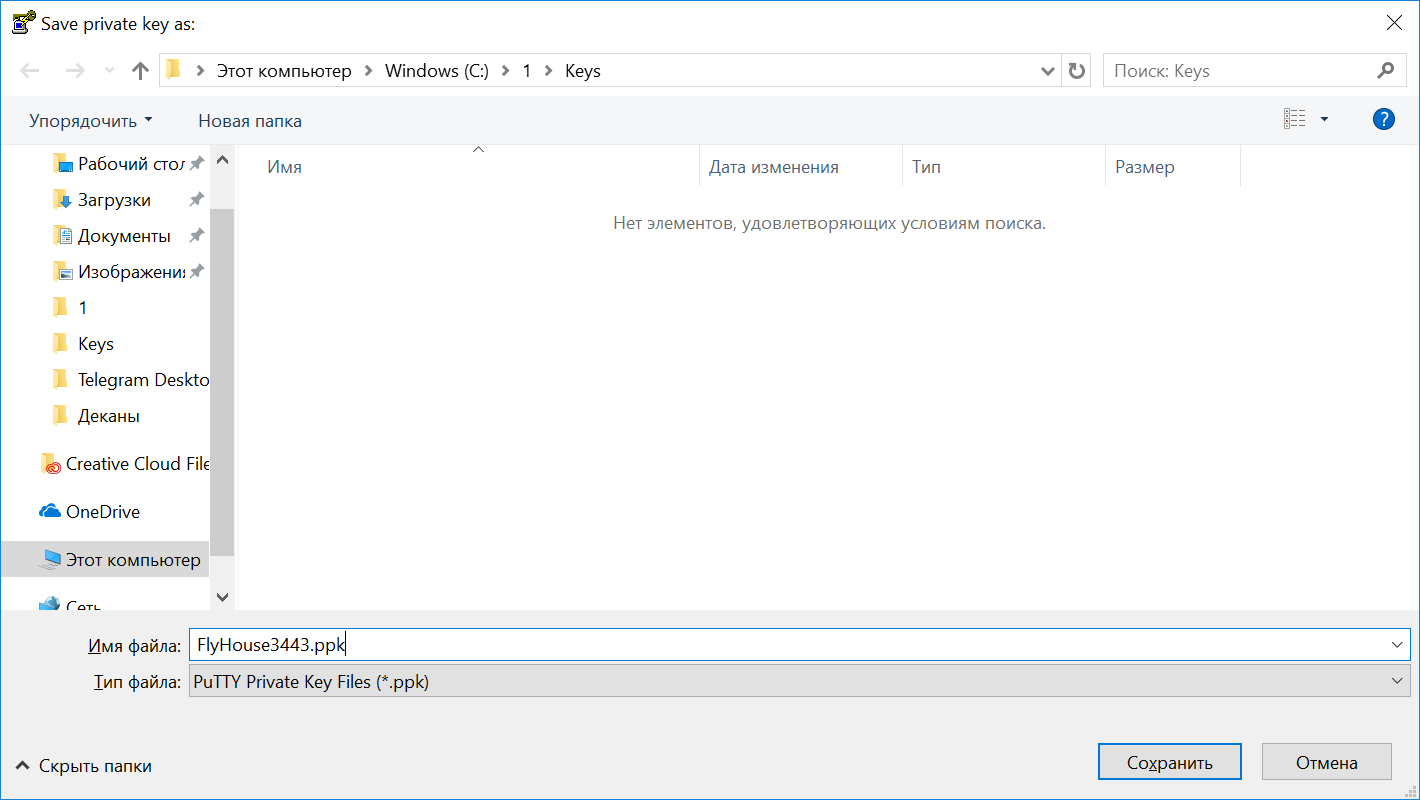
Жмем **Save private key**:



После этого **puttygen** предупреждает, что мы не установили парольную фразу. Обратите внимание, что парольная фраза нужна для доступа не на удаленный сервер, а для к ключу. Если ключ будет использоваться сервером, то без ввода парольной фразы стартовать не будет. Мы не будем задавать пароль для ключа.

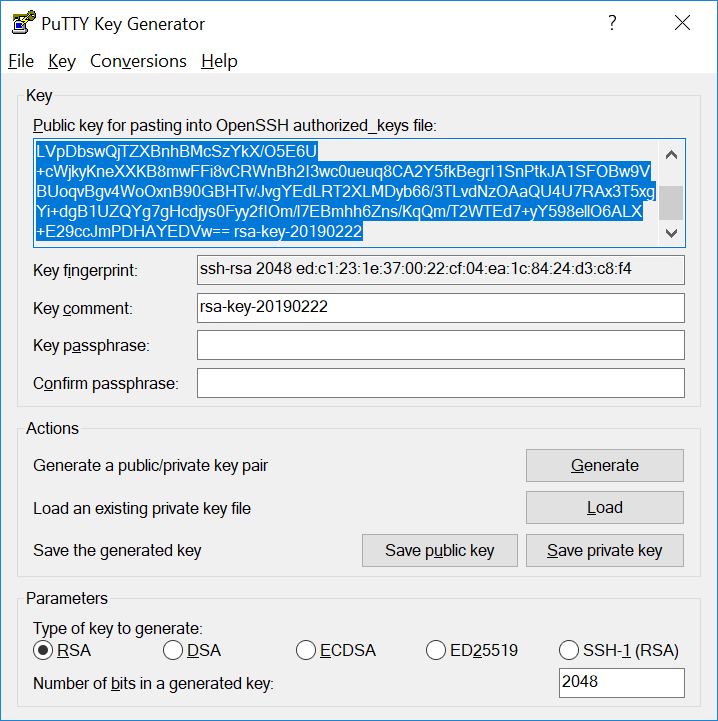


Жмем «Да».



Сохраняем приватный (закрытый) ключ с расширением **ppk**. Не перепутайте ключи!

Далее копируем публичный ключ из окна программы:



Заходим на удаленную машину (пока еще по паролю).

Если находимся не в домашней директории, переходим в нее:

|  |
| --- |
| $ cd |

Создаем папку **.ssh** в домашней директории:

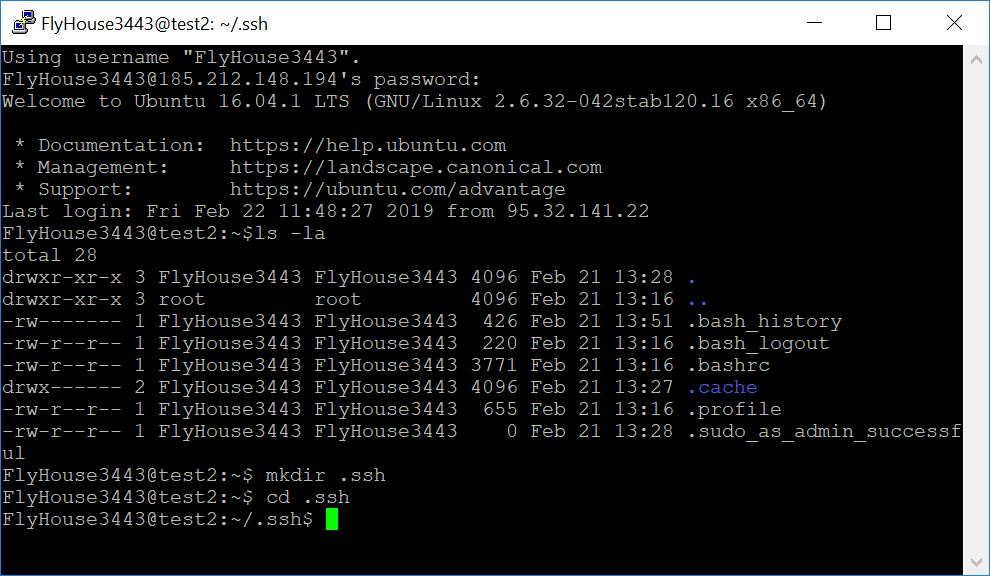
|  |
| --- |
| $ mkdir .ssh |

Права на эту директорию должны быть 700, выставим их:

|  |
| --- |
| $ chmod 700 .ssh |

Переходим в нашу новую директорию .**ssh**:

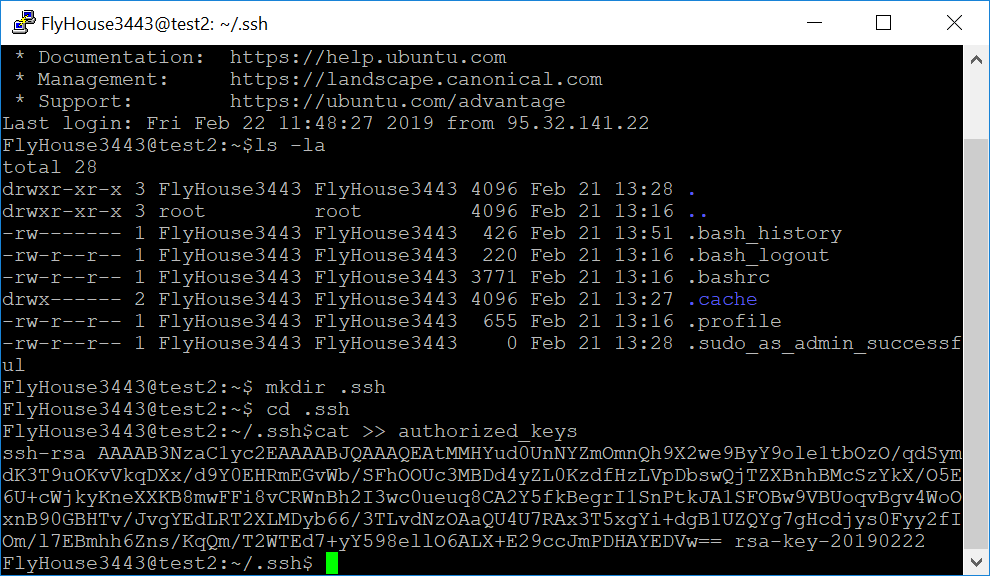
|  |
| --- |
| $ cd .ssh |



Далее мы будем вставлять наш публичный ключ с помощью команды:

|  |
| --- |
| $ cat > authorized\_keys |

Обязательно убедитесь, что в буфере обмена именно публичный ключ. Затем нажимаем правой кнопкой мыши — вставляется ключ:

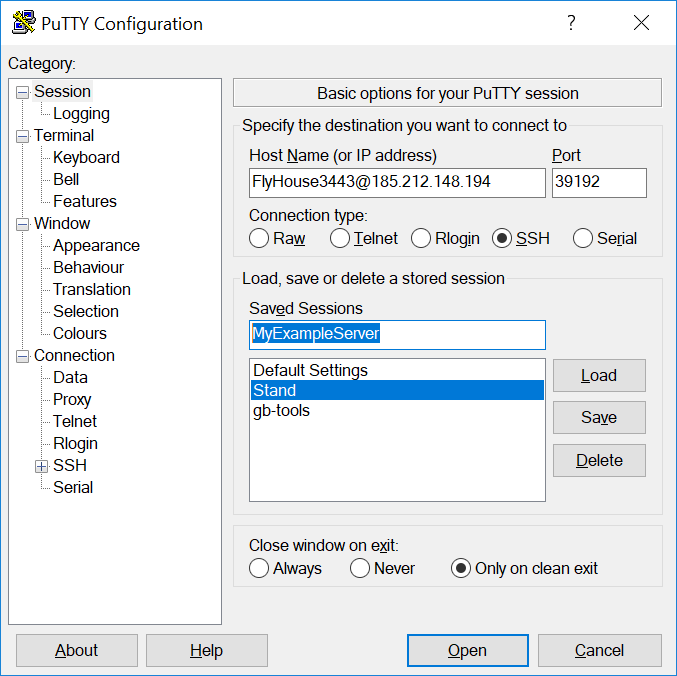


Затем жмем Enter и завершаем вводом Ctrl-D.

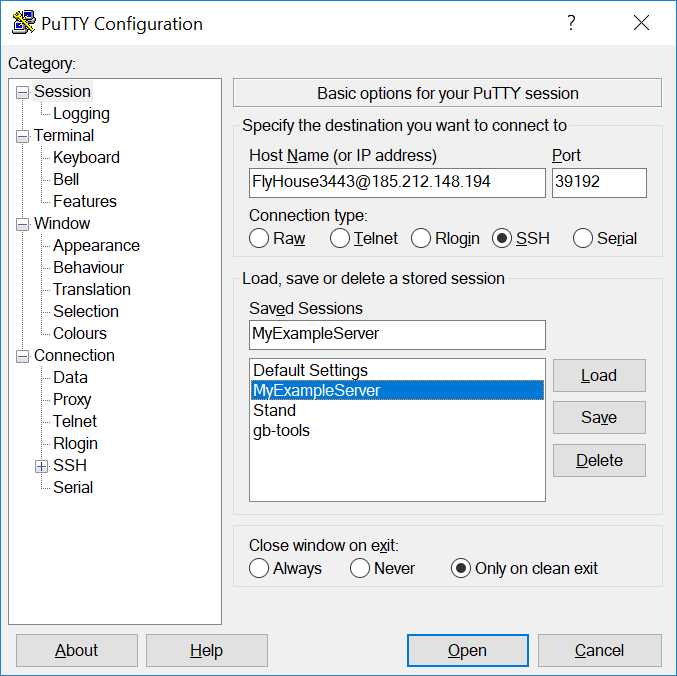
И последнее: права на этот файл должны быть 600, то есть чтение и запись для владельца. Назначим их:

|  |
| --- |
| $ chmod 600 authorized\_keys |

Теперь настраиваем PuTTY: вводим имя пользователя, IP-адрес, наш нестандартный порт, а в Saved Session задаем название для сервера:

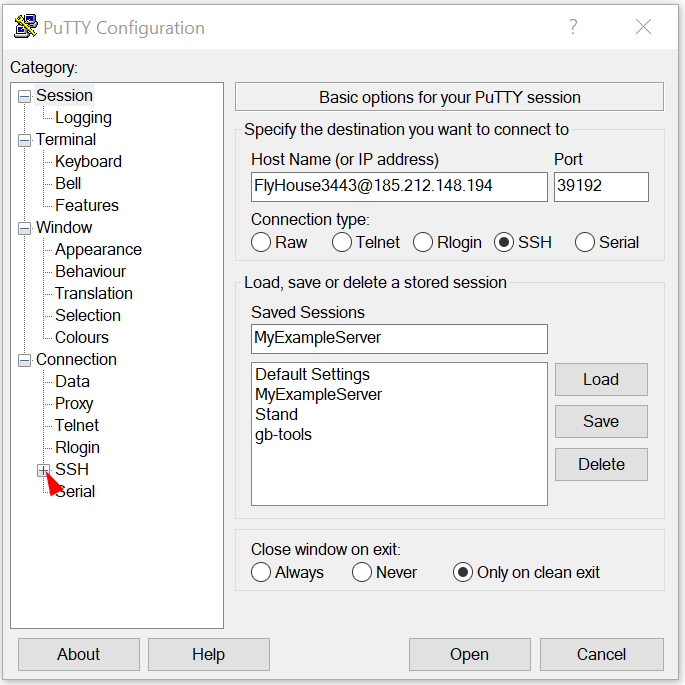


Нажимаем **Save**.

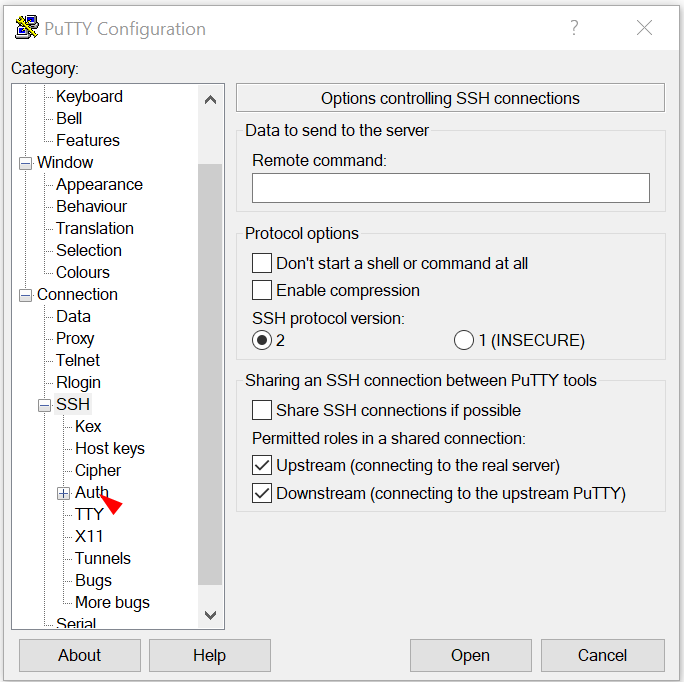


Теперь выполним настройки для сервера в **Saved Sessions**.

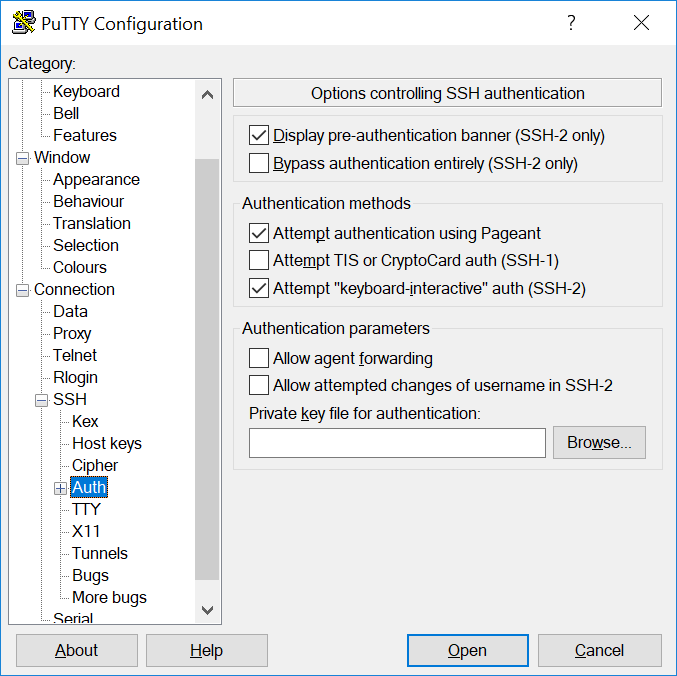
Надо настроить ключи. Кликаем на SSH:



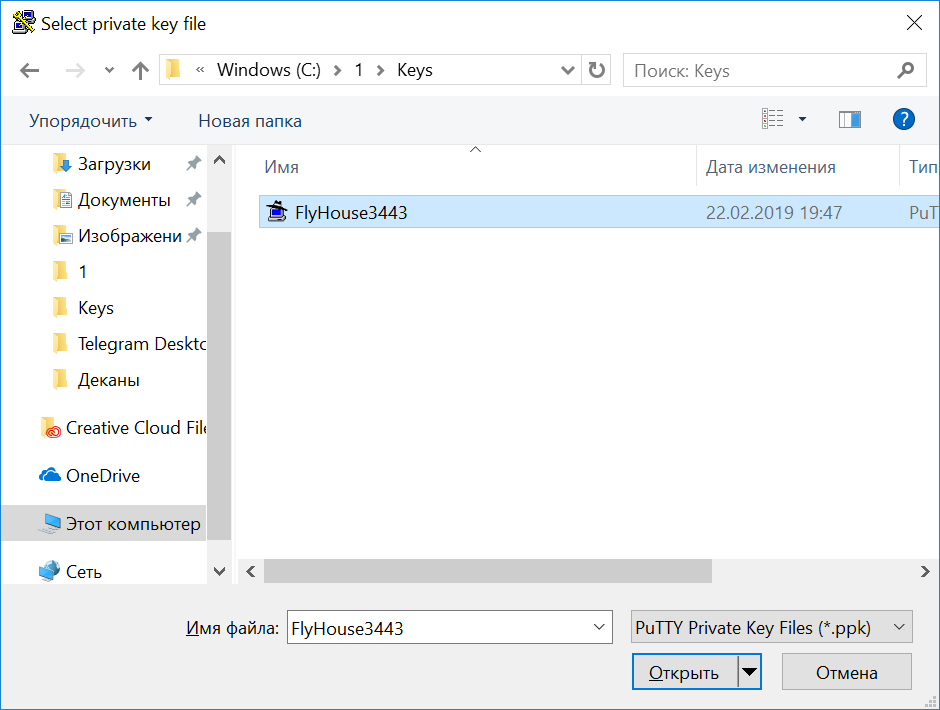
Кликаем на **Auth**:



Убеждаемся, что раздел **Auth** выбран:

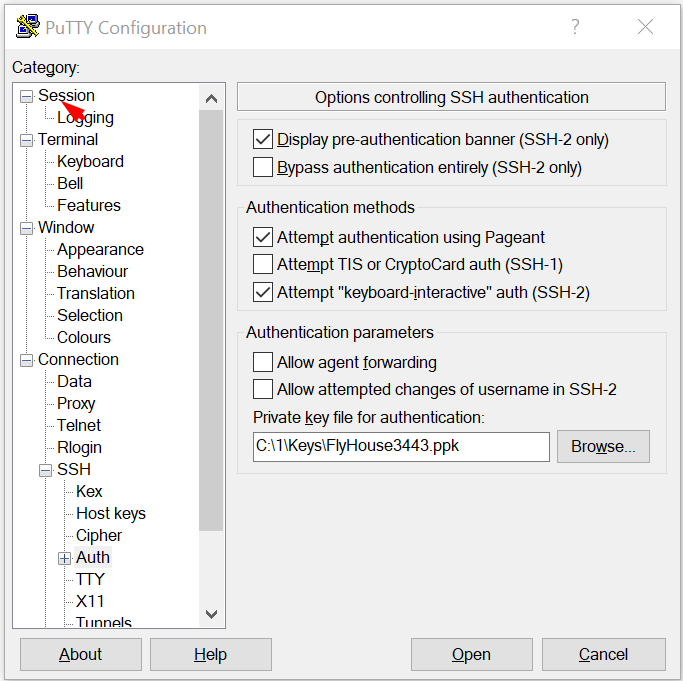


Жмем **Browse** и выбираем сохраненный приватный ключ:

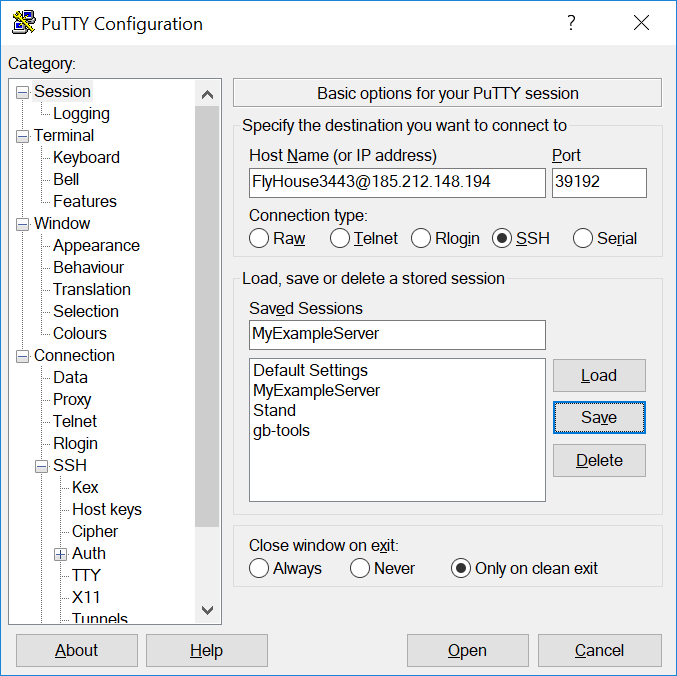


Это **ppk** (Windows по умолчанию скрывает разрешение).

Жмем «Открыть», а затем — **Session**:



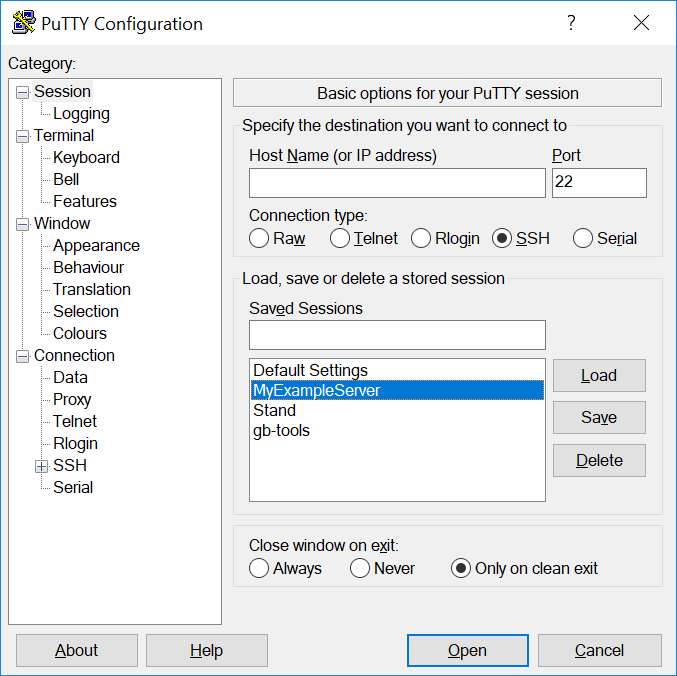
Теперь жмем **Save**:



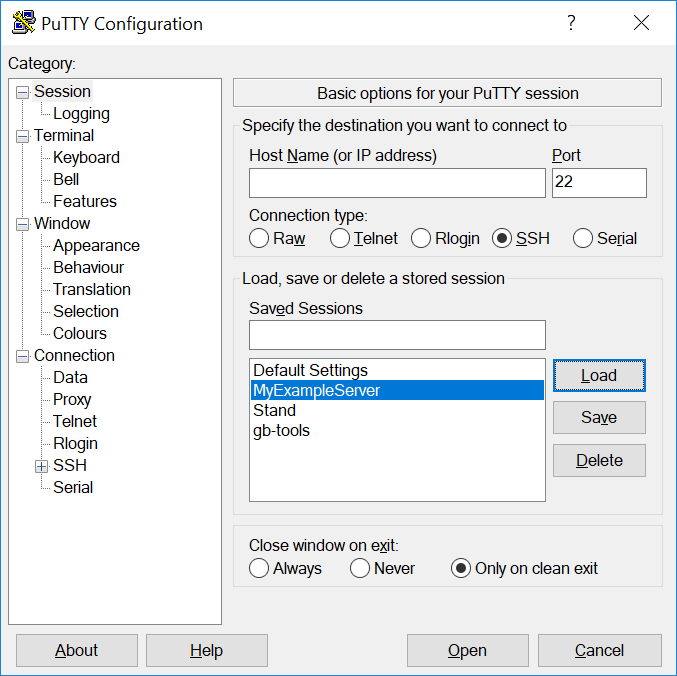
Теперь вы можете закрыть и заново открыть **Putty**:

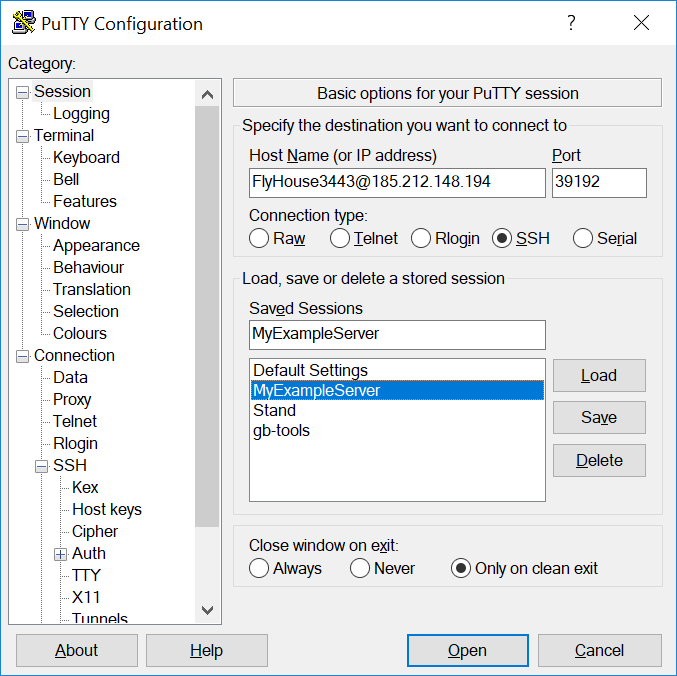


Выбираем наш сервер:

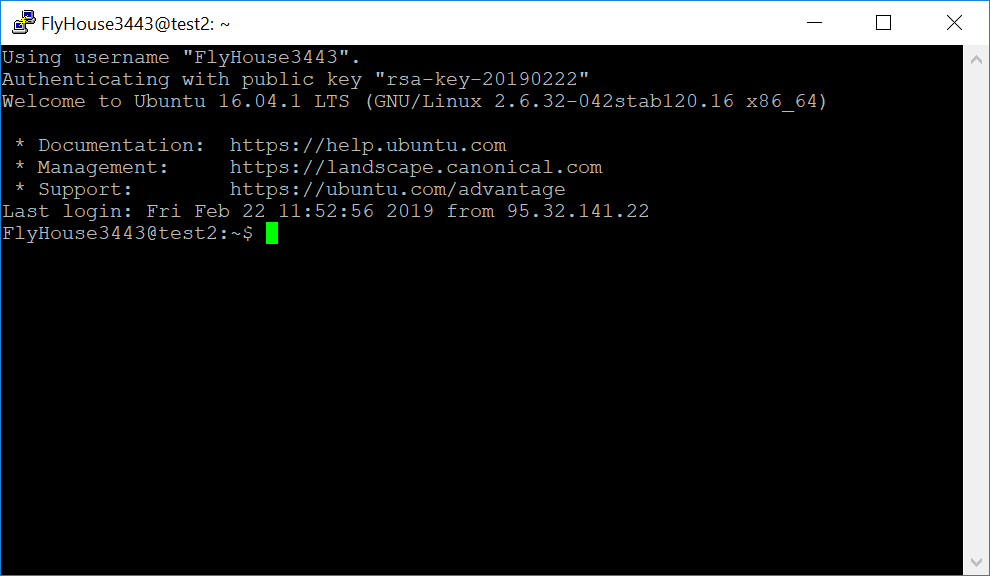


И жмем **Load**:





Жмем **Open**:

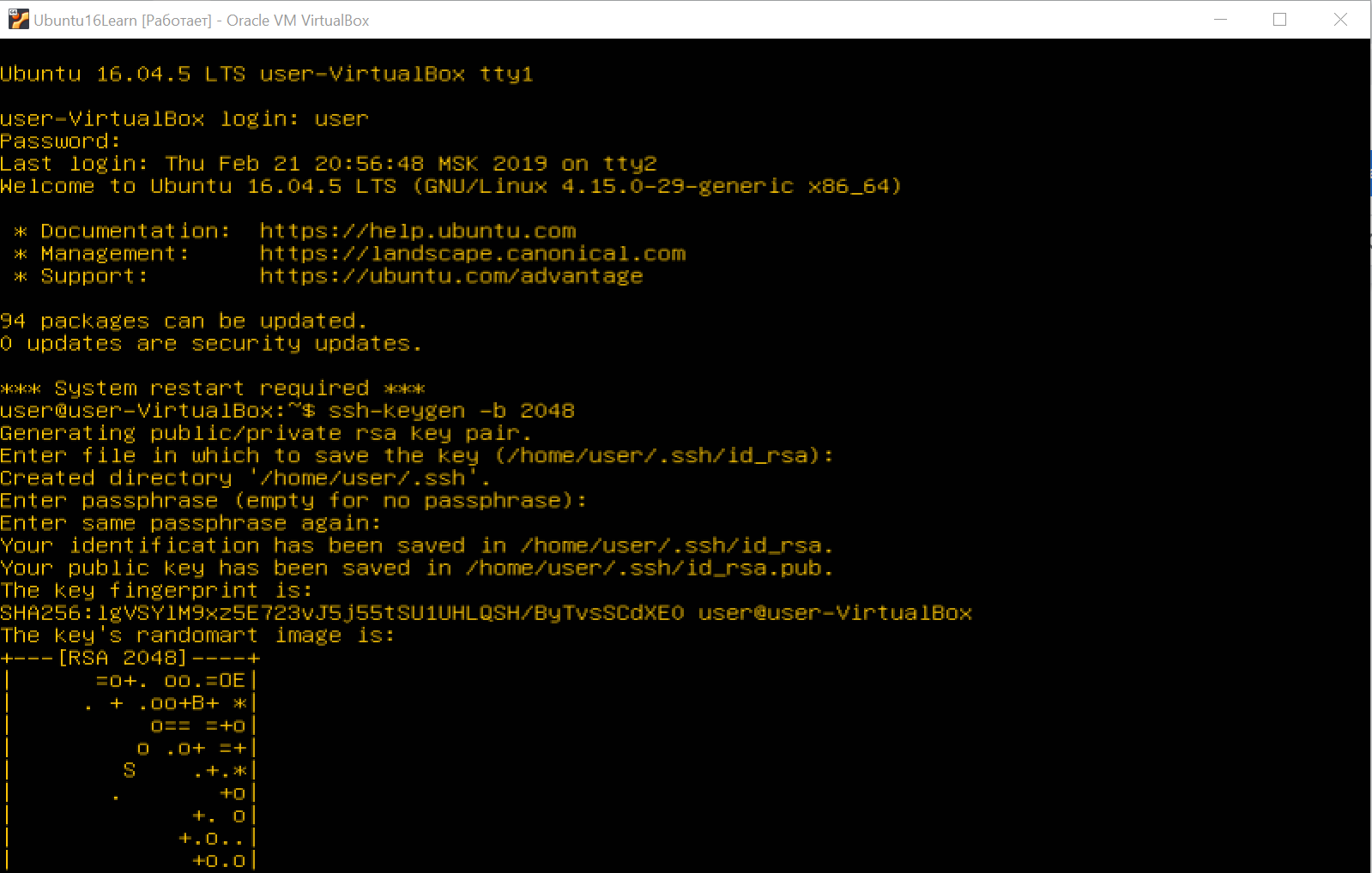


Ура, наш сервер подключился по ключу!

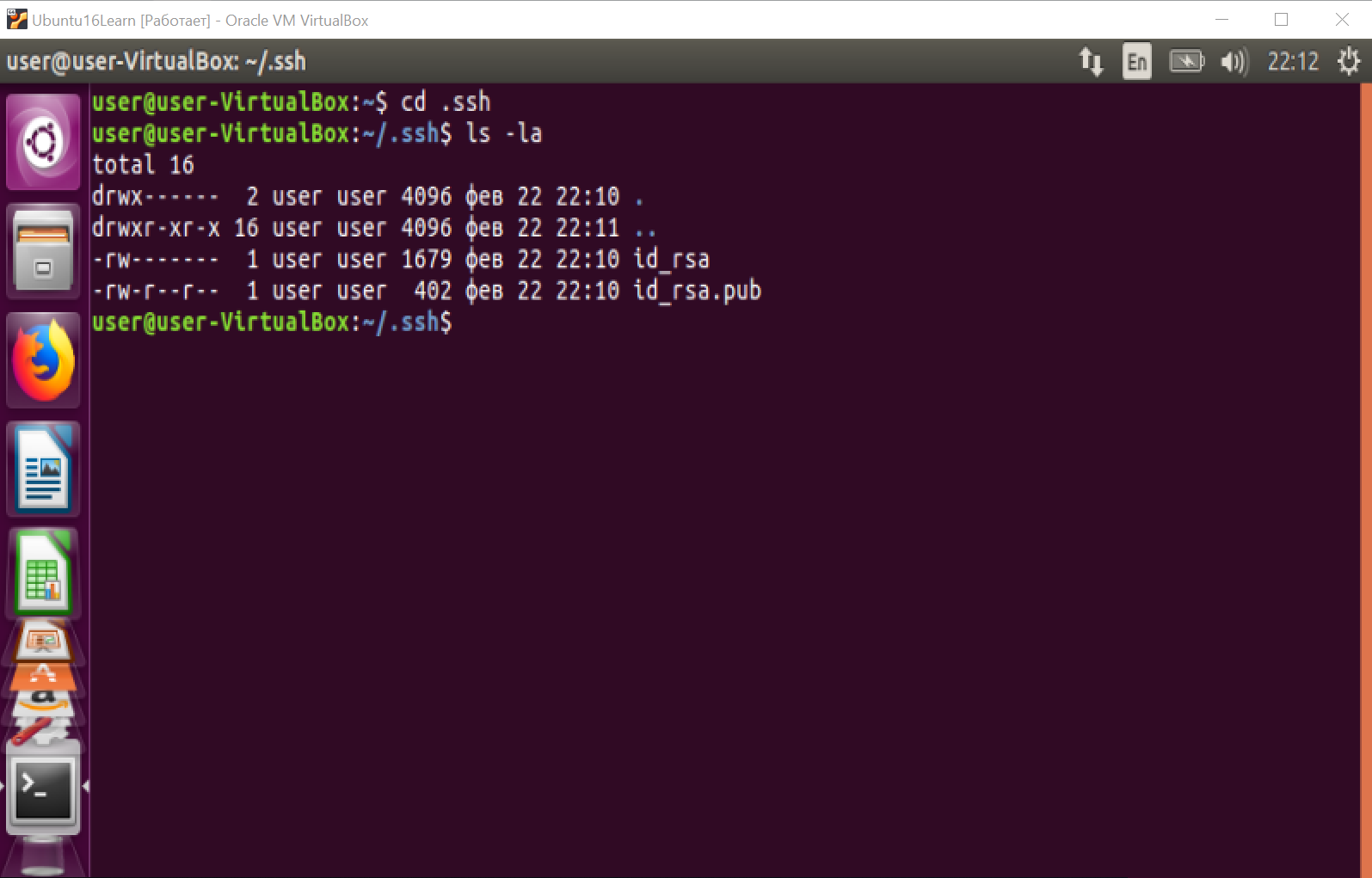
Теперь рассмотрим, как сгенерировать ключ на Linux или Mac (делается идентично).

В Linux:

Перейдите в терминал Ctrl-Alt-F1 (или Ctrl-Alt-F2, если вы в режиме X11):



Либо откройте терминал с помощью Ctrl-Alt-T:

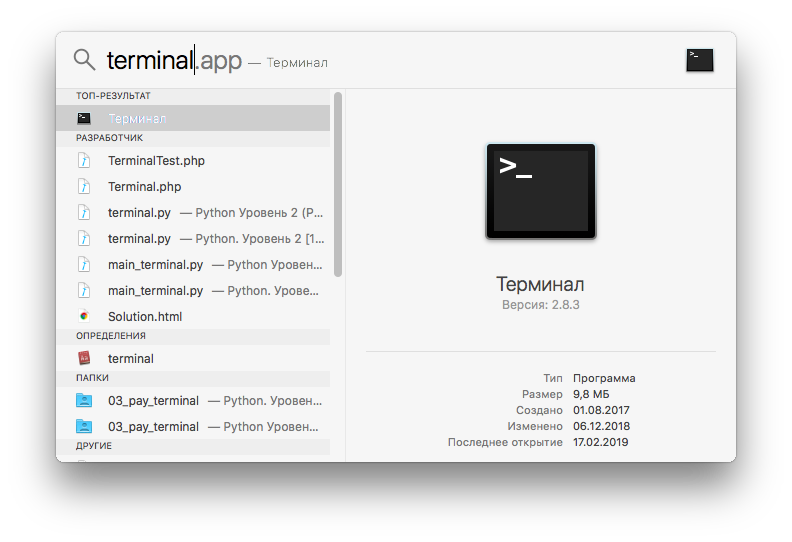


**В Mac:**

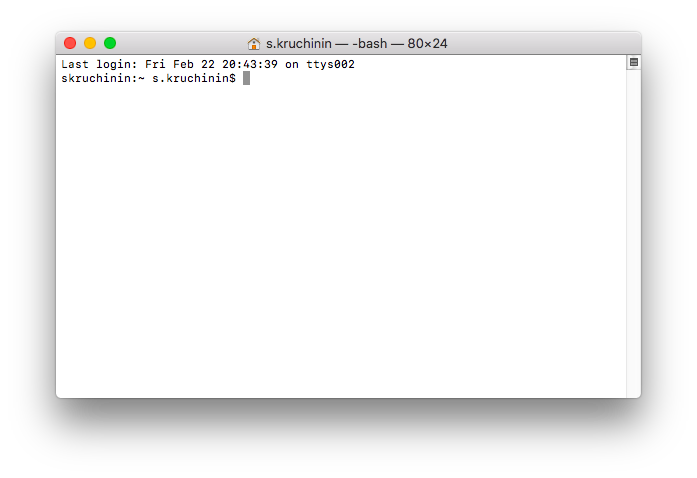
1. CMD + пробел:



1. Набираем terminal



1. Кликаем на «Терминал» и он запускается:

Далее работаем аналогично изложенным выше шагам.

Если нет директории **.ssh**, создадим ее:

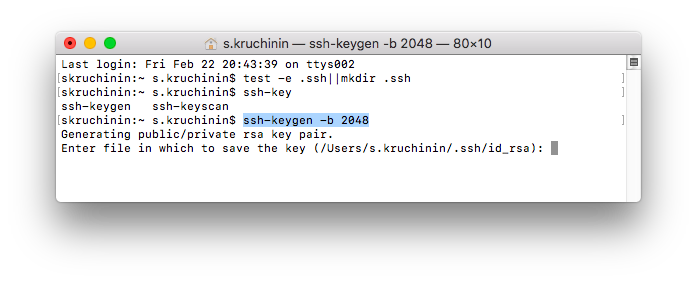
|  |
| --- |
| $ test -e .ssh||mkdir .ssh |

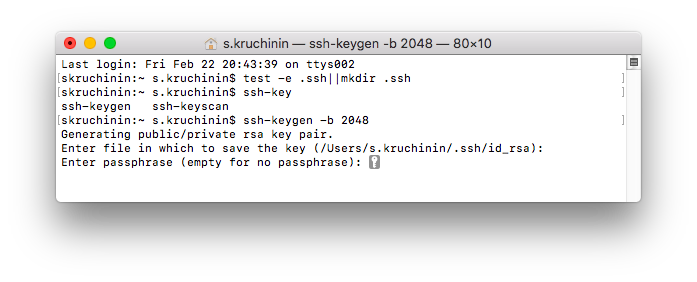
Переходим в **.ssh**:

|  |
| --- |
| $ cd .ssh |

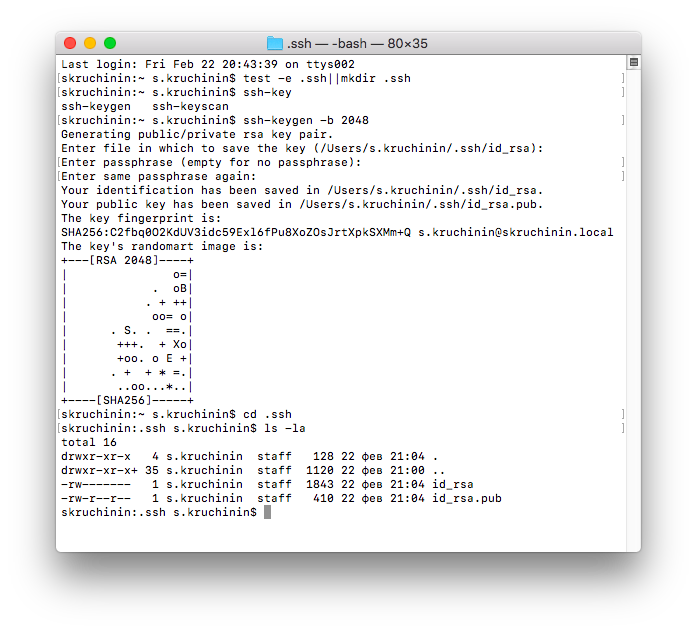
Запускаем **ssh-keygen**:

|  |
| --- |
| $ ssh-keygen -b 2048 |

Соглашаемся:

Обратите внимание, что парольная фраза нужна для доступа не на удаленный сервер, а к ключу. Если ключ будет использоваться сервером, то без ввода парольной фразы стартовать не будет. Мы не будем задавать пароль для ключа.

Пара ключей сгенерирована:

Проверяем c помощью:

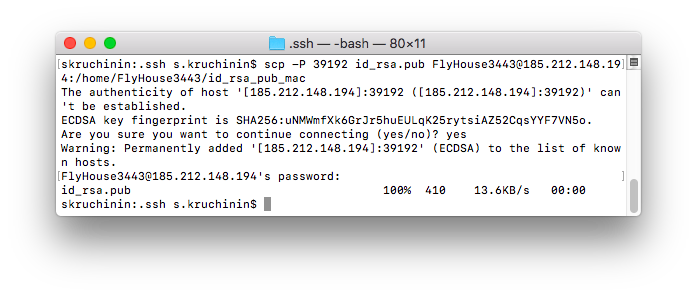
|  |
| --- |
| $ ls -la |

Обратите внимание, что:

* **id\_rsa** — приватный ключ (никогда не распространяйте его, он не должен покинуть вашу машину);
* **id\_rsa.pub** — публичный ключ, именно его вы будете загружать на сервер.

Для загрузки публичного ключа на сервер воспользуемся **scp**:

|  |
| --- |
| $ scp -P 39192 id\_rsa.pub FlyHouse3443@185.212.148.194:/home/FlyHouse3443/id\_rsa\_pub\_mac |

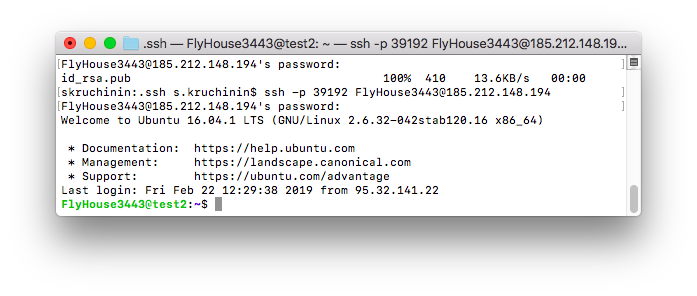
Добавляем хост в список известных, вводим пароль.

Теперь нам понадобится подключиться еще раз по паролю по **ssh**:

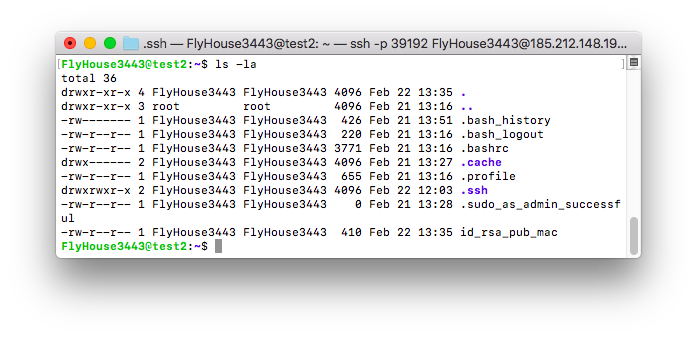
|  |
| --- |
| $ ssh -p 39192 FlyHouse3443@185.212.148.194 |

Обратите внимание, что ключ задания пароля у **scp** и **ssh** отличается регистром.

Подключились:



Файл на месте:



Добавляем полученный файл:

|  |
| --- |
| $ cat id\_rsa\_pub\_mac >>.ssh/authorized\_keys |

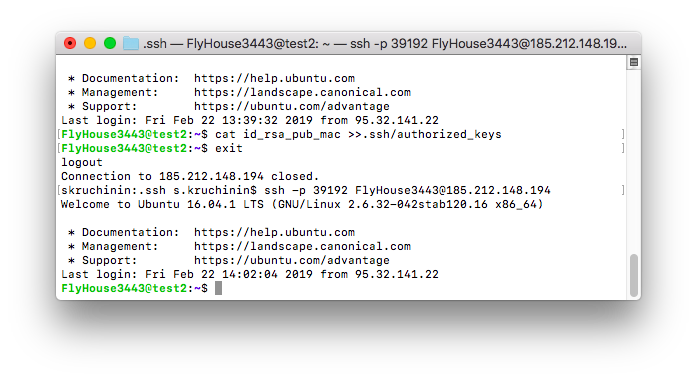
Выходим:

|  |
| --- |
| $ exit |

И подключаемся снова:

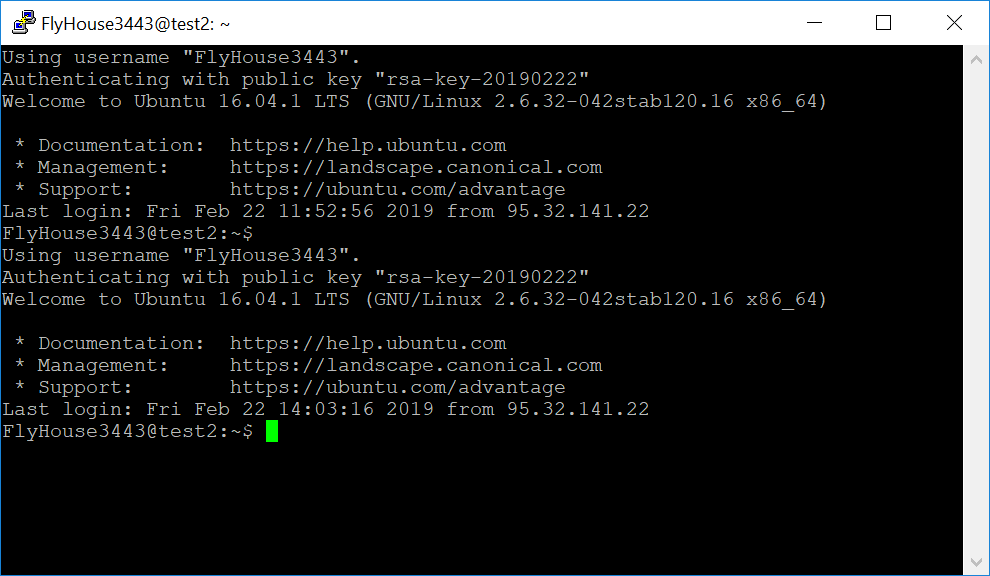
|  |
| --- |
| $ ssh -p 39192 FlyHouse3443@185.212.148.194 |

Работает — мы зашли без ввода пароля:



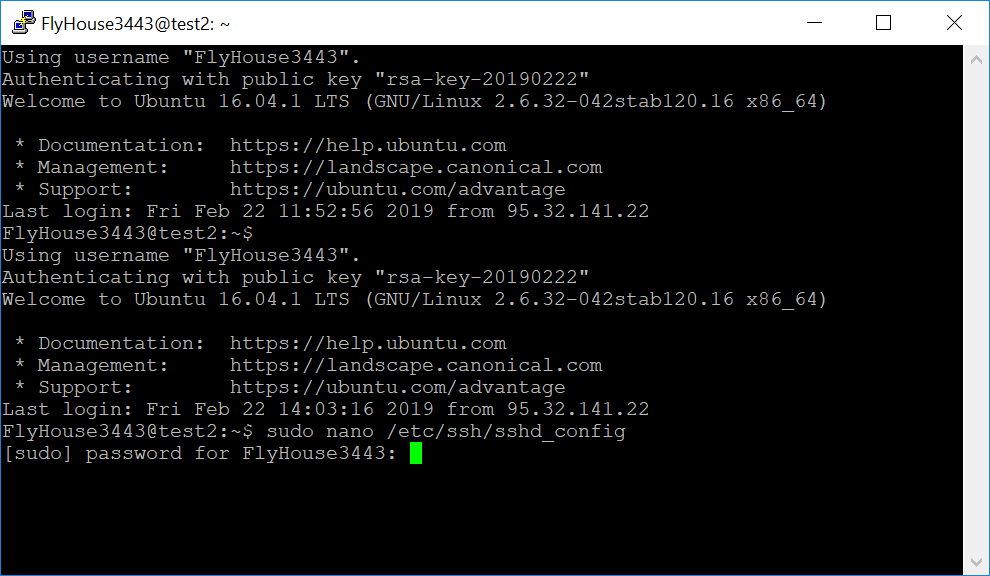
## Отключаем возможность логиниться руту и доступ по паролям

Переподключаемся — разумеется, без ввода пароля:



Правим **/etc/ssh/sshd\_config**:

|  |
| --- |
| $ sudo nano /etc/ssh/sshd\_config |

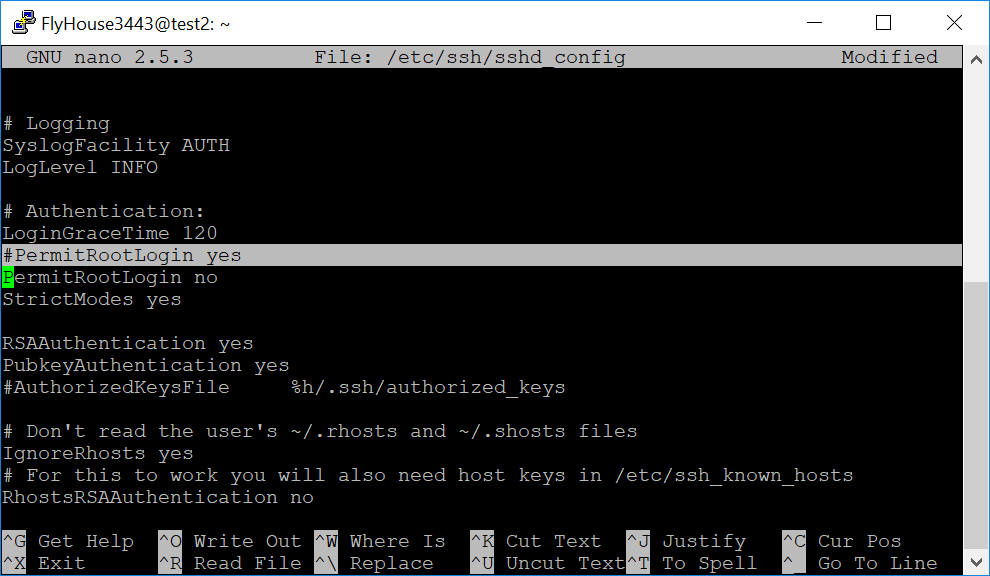


Пароль все равно нам нужен для рутовых операций. Вводим его и меняем:

|  |
| --- |
| PermitRootLogin yes |

на

|  |
| --- |
| PermitRootLogin no |

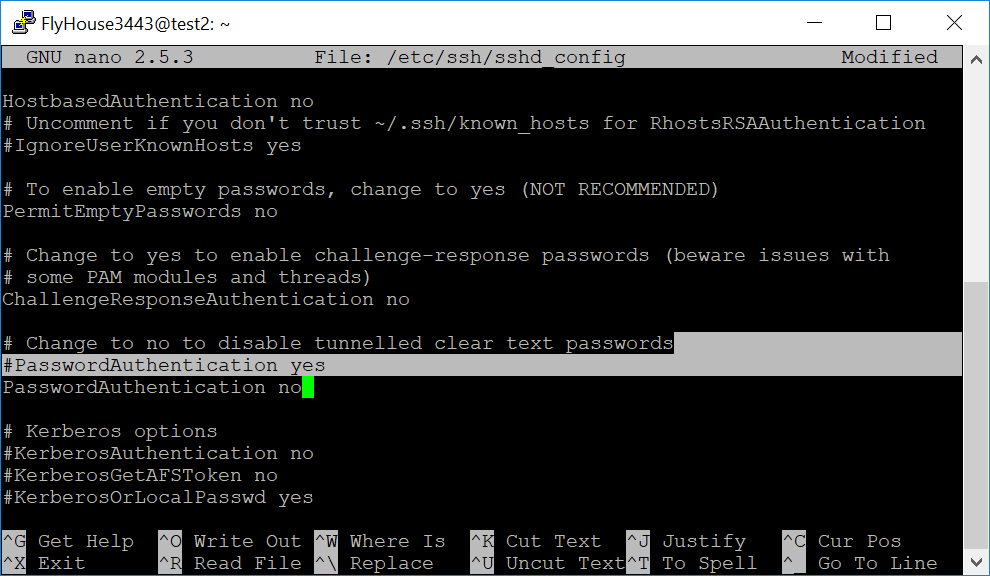


Далее меняем:

|  |
| --- |
| PasswordAuthentication yes |

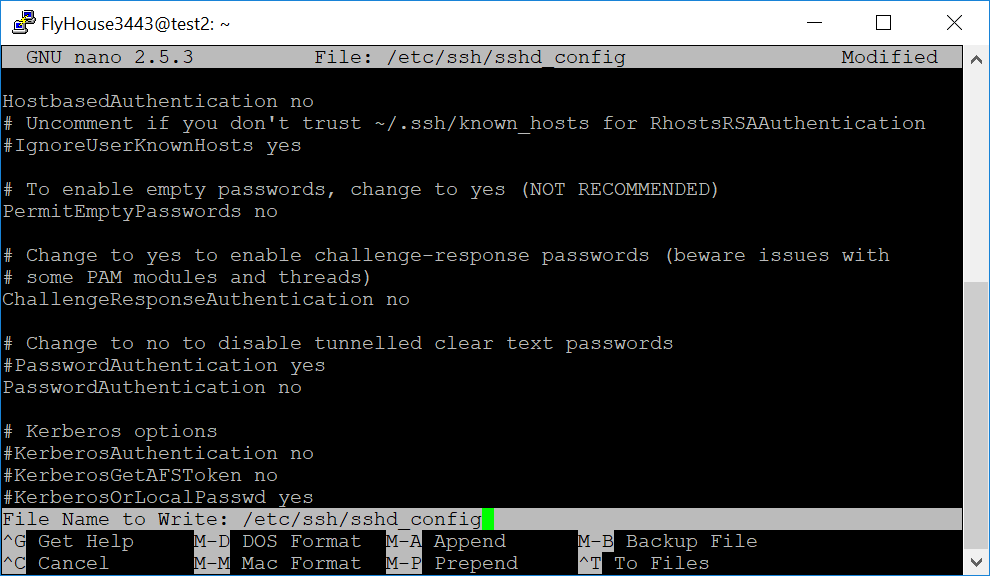
на

|  |
| --- |
| PasswordAuthentication no |



Сохраняем в **nano**:

* Ctrl-O
* Enter
* Ctrl-X



Созраняем в **vi**:

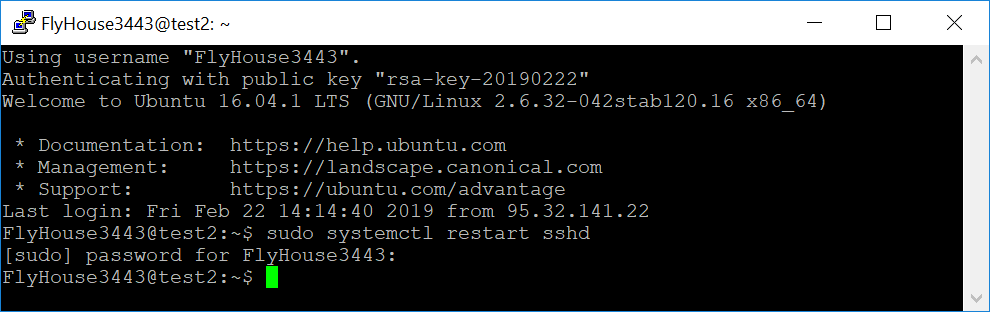
**:wq!**

Далее рестартуем **ssh**. Рестартуем сервер командой **service**:

|  |
| --- |
| $ sudo service sshd restart |

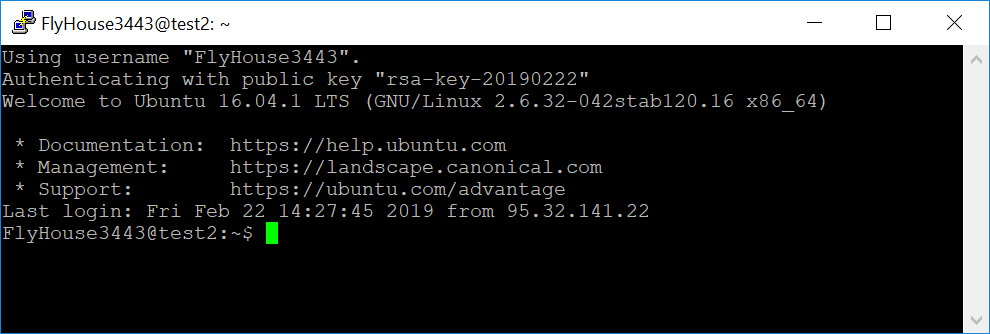
Если вы уверены, что используется система инициализации SystemD (Ubuntu 16.0 и выше), то:

|  |
| --- |
| $ sudo systemctl restart sshd |



Вводим пароль.

Теперь пробуем подключиться по ключу:



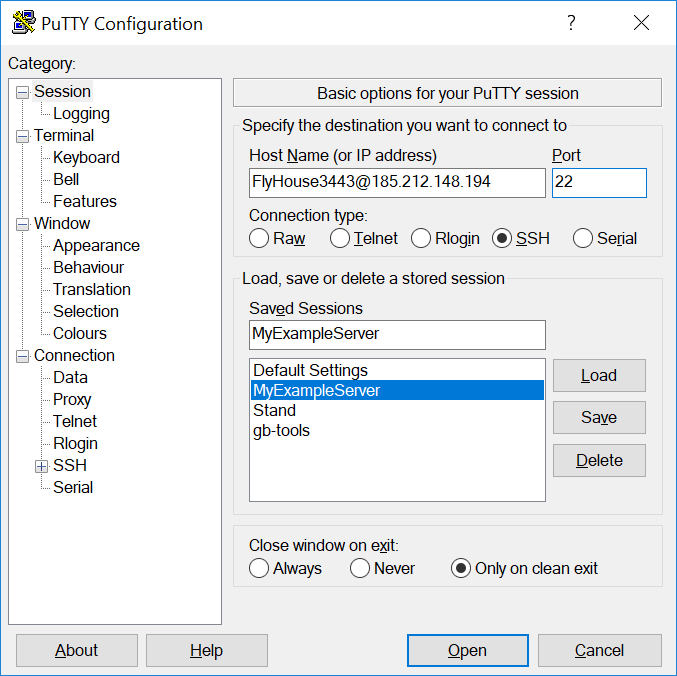
Если подключились — значит, все хорошо.

Если нет, то надо переустановить удаленный VDS-сервер или зайти по VNC — там получится залогиниться рутом. Но надеемся, что с этим вам пока не придется столкнуться.

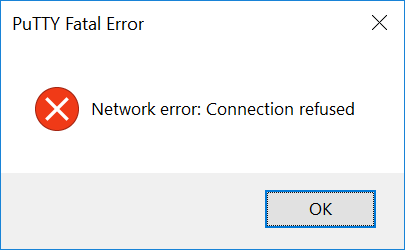
Теперь контрольный выстрел.

По **22/TCP**:

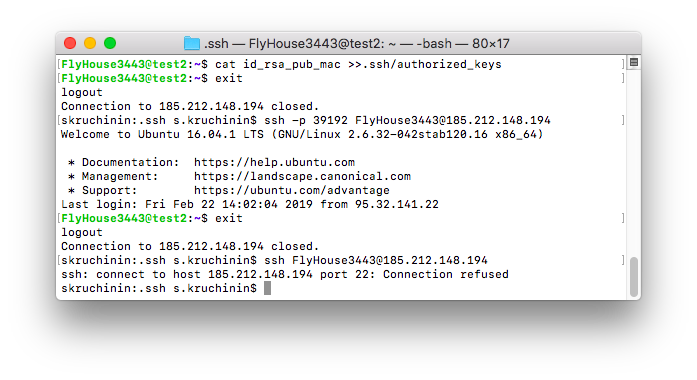
1. Жмем **Load** и меняем порт на 22:



Результат:



Аналогично на Mac и в Linux:



Для полного спокойствия чуть позже мы также заблокируем ненужные порты и воспользуемся для этого уже знакомым нам встроенным в Linux фаерволом **iptables**.

# Защищаем HTTP

HTTP — небезопасный протокол. Пароли, отправленные через POST или сообщенные вам, могут попасть в руки любому, кто имеет доступ к одной из промежуточных машин — провайдеру, соседям по хостингу и так далее.

Поэтому все чаще используется HTTPS. HTTP оправдан только в том случае, если сайт только отдает незашифрованное содержимое (например, простейшая домашняя страничка или информационный ресурс). Если на сайте присутствует форма авторизации, используйте HTTPS. Более того, наличие HTTPS является основным требованием Google и Яндекс для ранжирования, так что этот протокол оправдан даже для сайтов, не получающих данных от пользователя.

Технически HTTPS — это протокол HTTP, использующий сессию TLS. А HTTP непосредственно использует сессию TCP. Для работы по HTTPS принято использовать порт 443.

## TLS и SSL-сертификаты

TLS — новый протокол безопасности, основанный на SSL (TLSv1 успел устареть, это надо учитывать). Его идея состоит в использовании алгоритма RSA, сложность взлома которого определяется алгебраической сложностью задачи нахождения простых сомножителей натурального числа.

RSA — алгоритм асимметричного шифрования. Вместо общего симметричного ключа используется пара асимметричных ключей: публичный и приватный. То, что зашифровано одним, может быть расшифровано парным ключом. При этом публичный ключ распространяется корреспондентам, а приватный остается только у выпустившего его. Публикация приватного ключа — ошибка, требующая перевыпуска ключа и сертификатов.

Ассиметричные алгоритмы могут одновременно решить только одну задачу из двух:

* **шифрование трафика** — если шифруется публичным ключом, расшифруется приватным. Зашифровать может любой, а прочитать — только владелец публичного ключа. Трафик недоступен для прослушивания, но подтвердить подлинность отправителя невозможно;
* **аутентификация** — проверка подлинности абонента. Отправитель шифрует приватным ключом, а получатели расшифровывают публичным. Если они смогли расшифровать, значит послать мог только отправитель, чей публичный ключ у них есть. Но расшифровать такой трафик может любой.

Так как на практике требуется решение обеих задач, применяется третья сторона — удостоверяющий центр. Публичные ключи удостоверяющих центров импортируются в браузеры их производителем. Удостоверяющая способность таких центров (CA) не вызывает сомнений.

Владелец сайта выпускает приватный и публичный ключи и подписывает публичный ключ у удостоверяющего центра. Для этого центру сертификации передается запрос на подпись по стандарту **X.509**, а в ответ он отправляет сертификат. Так браузер может проверить, является ли сайт тем, за кого себя выдает.

Сертификат — публичный ключ и дополнительная информация, хеш от которой зашифрован приватным ключом удостоверяющего центра. Подпись ключа — услуга, предоставляемая удостоверяющими центрами. Можно подписать сайт собственно сгенерированным удостоверяющим центром, но браузеры будут выводить сообщение, что валидность ресурса проверить невозможно. Но для тестирования вариант самоподписанного сертификата подойдет.

Зачастую на практике, говоря **SSL**, подразумевают **TLS**. Тем не менее **SSL v3** и **TLS v1** уже признаны небезопасными, и это надо иметь в виду при настройке сервисов, использующих шифрование.

## Проверка mod\_ssl и доступ по HTTPS

Активирует **default-ssl** сайт:

|  |
| --- |
| # sudo a2ensite default-ssl |

Перечитайте конфигурацию сервера и проверьте — ничего не вышло, не активирован модуль **ssl**.

Активируйте:

|  |
| --- |
| # sudo a2enmod ssl |

Перезагрузите сервер, проверьте **test.tst** — теперь все работает. Если посмотрим, какой сертификат выдан, увидим:

|  |
| --- |
| Имя сертификата: **Ubuntu**.  Выдан **Ubuntu.** |

Можно попробовать подготовить свой сертификат для проверки работы.

Если установлен **openssl**, сгенерировать самоподписанный ключ можно с помощью скрипта **/usr/lib/ssl/apache2­/mod\_ssl/gentestcrt.sh**.

Мы рассмотрим вариант генерации самоподписанного сертификата по шагам.

## Настройка сайта с самоподписанным сертификатом

Разберем ручной способ создания самоподписанных сертификатов. На практике используется подпись сертификатов настоящим удостоверяющим центром. Генерация самоподписанных сертификатов может быть полезна для понимания механизма работы, а также для тестирования.

Сначала создадим собственный центр сертификации (CA). Для этого добавим частный ключ для CA:

|  |
| --- |
| # cd /etc/apache2/  # mkdir ssl  # chmod 700 ssl  # cd ssl  #openssl genrsa -des3 -out my-ca.key |

Секретная фраза — пароль к центру сертификации. Создаем сертификат.

|  |
| --- |
| # openssl req -new -x509 -days 3650 -sha256 -key my-ca.key -out my-ca.crt |

Придется ввести пароль, который уже запомнили, и ответить на вопросы. Создаем ключ для **Apache2**:

|  |
| --- |
| # openssl genrsa -des3 -out my-apache.key 1024 |

Потребуется ввести секретную фразу. Ее мы удалим из сертификата сервера **Apache**, чтобы она не запрашивалась при каждом старте:

|  |
| --- |
| # openssl rsa -in my-apache.key -out new.my-apache.key  # mv new.my-apache.key my-apache.key |

Создаем запрос на подпись сертификата **Certificate Signature Request (csr)**, чтобы подписать в центре сертификации:

|  |
| --- |
| # openssl req -new -key my-apache.key -out my-apache.csr |

Раньше необходимо было поместить в поле **Common name FQDN** доменное имя — например, **test.tst**. Сейчас используются только значения **DNS.1**, **DNS.2** (и так далее) в секции **[ alt\_names ],** так как в поле **Common name** можно было указать только одно доменное имя. Более того, сейчас Google Chrome даже не пытается анализировать поле **Common name**.

Поэтому понадобится сделать еще ряд дополнительных действий (до выполнения шагов, изложенных выше), либо использовать скрипт генерации сертификата.

Отредактируем файл **/etc/ssl/openssl.cnf**:

|  |
| --- |
| nano /etc/ssl/openssl.cnf |

Раскомментируем строку:

|  |
| --- |
| req\_extensions = v3\_req |

Дальше нужно найти секцию **[ v3\_req ]** (пробелы имеют значение). Найдите в разделе такие строки:

|  |
| --- |
| basicConstraints = CA:FALSE keyUsage = nonRepudiation, digitalSignature, keyEncipherment |

Сразу под ними допишем следующее:

|  |
| --- |
| subjectAltName = @alt\_names [ alt\_names ] DNS.1 = www.server.com DNS.2 = server.com DNS.3 = admin.server.com |

Секция **alt\_names** будет содержать альтернативные имена доменов, которые требуется включить в сертификат. Замените их на свои. После этой операции нужно вернуться на три шага назад, сгенерировать **my-apache.key** и **csr**.

Теперь подпишем сертификат **Apache** в центре сертификации:

|  |
| --- |
| # openssl x509 -req -in my-apache.csr -out my-apache.crt -sha256 -CA my-ca.crt -CAkey my-ca.key -CAcreateserial -days 36  # chmod 0400 \*.key |

В качестве альтернативы всем вышеописанным действиям можно воспользоваться следующим скриптом. Поменяйте значения в секциях **[ dn ]**, **[ alt\_names ]** и добавьте при необходимости **DNS.2** (и т. д.):

|  |
| --- |
| #!/bin/bash  echo "Creating key and request..."  openssl req -new -nodes -keyout my-apache.key -out my-apache.csr -config <(  cat <<-EOF  [req]  default\_bits = 4096  prompt = no  default\_md = sha256  req\_extensions = req\_ext  x509\_extensions = usr\_cert  distinguished\_name = dn  [ dn ]  C=RU  ST=Moscow Region  L=Moscow  O=Geekbrains, OOO  OU=IT department  emailAddress=admin@geekbrains.ru  CN = \*.geekbrains.ru  [ usr\_cert ]  basicConstraints=CA:FALSE  nsCertType = client, server, email  keyUsage = nonRepudiation, digitalSignature, keyEncipherment  extendedKeyUsage = serverAuth, clientAuth, codeSigning, emailProtection  nsComment = "OpenSSL Generated Certificate"  subjectKeyIdentifier=hash  authorityKeyIdentifier=keyid,issuer  [ req\_ext ]  subjectAltName = @alt\_names  basicConstraints = CA:FALSE  keyUsage = nonRepudiation, digitalSignature, keyEncipherment  extendedKeyUsage = serverAuth, clientAuth, codeSigning, emailProtection  [ alt\_names ]  DNS.1 = \*.geekbrains.ru  EOF  )  echo "Signing..."  openssl x509 -req -in my-apache.csr -CA my-ca.crt -CAkey my-ca.key -CAcreateserial -out my-apache.crt -days 5000 -extensions req\_ext -extfile <(  cat <<-EOF  [req]  default\_bits = 4096  prompt = no  default\_md = sha256  req\_extensions = req\_ext  x509\_extensions = usr\_cert  [ usr\_cert ]  basicConstraints=CA:FALSE  nsCertType = client, server, email  keyUsage = nonRepudiation, digitalSignature, keyEncipherment  extendedKeyUsage = serverAuth, clientAuth, codeSigning, emailProtection  nsComment = "OpenSSL Generated Certificate"  subjectKeyIdentifier=hash  authorityKeyIdentifier=keyid,issuer  [ req\_ext ]  subjectAltName = @alt\_names  basicConstraints = CA:FALSE  keyUsage = nonRepudiation, digitalSignature, keyEncipherment  extendedKeyUsage = serverAuth, clientAuth, codeSigning, emailProtection  [ alt\_names ]  DNS.1 = \*.geekbrains.ru  EOF  )  echo "Done!" |

Далее редактируем **default-ssl.conf**:

|  |
| --- |
| SSLCertificateFile /etc/apache2/ssl/my-apache.crt  SSLCertificateKeyFile /etc/apache2/ssl/my-apache.key  SSLCACertificateFile /etc/apache2/ssl/my-ca.crt |

Перезагружаем сервер, проверяем [**https://test.tst**](https://test.tst). Смотрим информацию о сертификате и видим, что отображается созданный и подписанный нами образец.

## Создание сертификата с помощью Let’s Encrypt

Сначала установим **Certbot**:

|  |
| --- |
| $ cd /usr/local/sbin  $ sudo wget <https://dl.eff.org/certbot-auto>  $ sudo chmod a+x /usr/local/sbin/certbot-auto |

Получим сертификат:

|  |
| --- |
| $ sudo certbot-auto certonly --webroot --agree-tos --email myemail@domen.ru -w /var/www/ -d domen.ru -d [www.domen.ru](http://www.domen.ru)  $ sudo certbot-auto --apache |

* **--webroot** — специальный ключ, повышающий надежность работы **Certbot** под **Nginx**;
* **--agree-tos** — автоматическое согласие с Условиями предоставления услуг (Terms of Services);
* **--email myemail@domen.ru** — ваш email. Будьте внимательны: его нельзя изменить. Он потребуется для восстановления доступа к домену и для его продления;
* **-w /var/www** — указываем корневую директорию сайта;
* **-d domen.ru** — через ключ **-d** указываем, для каких доменов запрашиваем сертификат. Начинать надо c домена второго уровня **domen.ru** и через такой же ключ указывать поддомены: например, **-d www.domen.ru -d opt.domen.ru**.

Команда **sudo certbot-auto --apache** внесет необходимые изменения в конфигурационные файлы Apache.

Скрипт **Certbot** начнет работу, предложит установить дополнительные пакеты. Соглашайтесь и ждите завершения.

При успешном завершении работы **Certbot** поздравляет с генерацией сертификата и выдает следующее сообщение:

|  |
| --- |
| IMPORTANT NOTES:  - If you lose your account credentials, you can recover through  e-mails sent to sammy@digitalocean.com  - Congratulations! Your certificate and chain have been saved at  /etc/letsencrypt/live/domen.com/fullchain.pem. Your  cert will expire on 2017-03-12. To obtain a new version of the  certificate in the future, simply run Let's Encrypt again.  - Your account credentials have been saved in your Let's Encrypt  configuration directory at /etc/letsencrypt. You should make a  secure backup of this folder now. This configuration directory will  also contain certificates and private keys obtained by Let's  Encrypt so making regular backups of this folder is ideal.  - If like Let's Encrypt, please consider supporting our work by:    Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate  Donating to EFF: https://eff.org/donate-le |

Когда установка SSL-сертификата **Apache Ubuntu** будет завершена, вы найдете созданные файлы сертификатов в папке **/etc/letsencrypt/live/domen.ru/**. В этой папке будет четыре файла:

* **cert.pem** — ваш сертификат домена;
* **chain.pem** — сертификат цепочки Let's Encrypt;
* **fullchain.pem** — **cert.pem** и **chain.pem** вместе;
* **privkey.pem** — секретный ключ вашего сертификата.

Теперь вы можете зайти на сайт по **https**. Чтобы проверить, как работает **SSL** и правильно ли выполнена установка ssl-сертификата на сайт, можно открыть в браузере эту ссылку:

|  |
| --- |
| https://www.ssllabs.com/ssltest/analyze.html?d=domen.ru&latest |

**SSL** настроен. Стоит иметь в виду, что сертификаты, полученные от **Let’s Encrypt**, действительны в течение 90 дней, поэтому рекомендуется продлевать их каждые 60 дней.

Клиент **letsecnrypt** имеет команду **renew**, которая позволяет проверить установленные сертификаты и обновить их, если до истечения срока осталось меньше 30 дней.

Чтобы запустить процесс обновления для всех настроенных доменов, выполните:

|  |
| --- |
| $ sudo certbot-auto renew |

Если сертификат был выдан недавно, команда проверит его дату истечения и выдаст сообщение, что продление пока не требуется. Если вы создали сертификат для нескольких доменов, в выводе будет показан только основной. Но обновление будет актуально для всех.

Самый простой способ автоматизировать этот процесс — добавить вызов утилиты в планировщик **cron**. Для этого выполним команду:

|  |
| --- |
| $ crontab -e |

И добавим строку:

|  |
| --- |
| 30 2 \* \* 1 /usr/local/sbin/certbot-auto renew |

Команда обновления будет выполняться каждый понедельник в 2:30. Информация про результат выполнения будет сохраняться в файл **/var/log/le-renewal.log**.

# Тонкая настройка iptables

Для полного спокойствия и защиты нашего сервера от хакеров мы заблокируем ненужные порты. для этого воспользуемся встроенным в Linux фаерволом iptables.

## Доступ по ssh

Обратите внимание: имеет значение порядок правил в цепочке. Сравните:

|  |
| --- |
| # iptables -A INPUT -p tcp -j DROP  # iptables -A INPUT -p tcp --dport=22 -j ACCEPT |

Этот вариант полностью отрежет компьютер от внешнего мира. Несмотря на то что второе правило разрешает порт 22 (ssh), до него дело не дойдет, так как при срабатывании правила (а оно сработает на любой пакет с TCP-сегментом), будет переход на DROP и завершение движения в цепочке.

А такой вариант сначала разрешит порт 22 и дальше проверять не будет. Те, кто не 22, пойдут дальше и будут отброшены:

|  |
| --- |
| # iptables -A INPUT -p tcp --dport=22 -j ACCEPT  # iptables -A INPUT -p tcp -j DROP |

Рассмотрим примеры. Установим для таблиц по умолчанию политики DROP.

***Примечание:*** *не делайте так, работая по ssh. Сначала по ssh настройте правила, дающие доступ к машине, а уже потом применяйте политики. На локальной машине можно сразу.*

|  |
| --- |
| # iptables -P INPUT DROP  # iptables -P OUTPUT DROP  # iptables -P FORWARD DROP |

Таким способом мы запретим и межпроцессный обмен, и обмен через локальную петлю. Не будет действовать локальный DNS-сервер (в Ubuntu он работает по адресу 127.0.1.1), PHP не сможет обратиться к MySQL и так далее. Поэтому:

|  |
| --- |
| # iptables -A INPUT -i lo -j ACCEPT  # iptables -A OUTPUT -o lo -j ACCEPT |

Запрет ICMP — это неправильно. ICMP — часть механизма IP и служит для нормальной работы стека TCP/IP. Машины, у которых полностью запрещен ICMP, создают проблемы при конфигурации сетей, являясь своего рода «черными дырами».

|  |
| --- |
| # iptables -A INPUT -p icmp -j ACCEPT  # iptables -A OUTPUT -p icmp -j ACCEPT |

Далее следует разрешить локальные соединения с динамических портов (мы уже знаем, где их посмотреть):

|  |
| --- |
| # cat /proc/sys/net/ipv4/ip\_local\_port\_range  32768 61000  # iptables -A OUTPUT -p TCP --sport 32768:61000 -j ACCEPT  # iptables -A OUTPUT -p UDP --sport 32768:61000 -j ACCEPT |

Разрешить только те пакеты, которые мы запросили:

|  |
| --- |
| # iptables -A INPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT  # iptables -A INPUT -p UDP -m state --state ESTABLISHED,RELATED -j ACCEPT |

Но если работаем как сервер, следует разрешить и нужные порты:

|  |
| --- |
| # iptables -A INPUT -i eth0 -p TCP --dport 22 -j ACCEPT  # iptables -A OUTPUT -o eth0 -p TCP --sport 22 -j ACCEPT |

Разрешили **ssh**. Если работали через него, теперь можно применять политики по умолчанию.

## Доступ по http и https

Аналогично откройте доступ на стандартные порты нашего веб-сервера 80 и 443. Проверьте порты настроенной машины с помощью **nmap** самостоятельно (просканируйте порты с другой машины).

Примеры проброса порта в контейнер:

|  |
| --- |
| # iptables -t nat -A PREROUTING --dst 1.2.3.4 -p tcp --dport 80 -j DNAT --to-destination 10.0.0.2 |

Пример проброса с одного порта на другой:

|  |
| --- |
| # iptables -t nat -D PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080 |

Более подробно смотрите в официальной документации.

|  |
| --- |
| $ man iptables |

# Другие полезные утилиты

Посмотреть скрытые файлы (их имена начинаются с точки):

|  |
| --- |
| $ ls -la |

Посмотреть историю команд:

|  |
| --- |
| $ ls ~/.bash\_history |

Посмотреть список прослушиваемых tcp-сокетов:

|  |
| --- |
| $ netstat -ntl |

Просмотреть список всех tcp-сокетов, в том числе исходящих:

|  |
| --- |
| $ netstat -ntla |

Просмотреть список всех tcp-сокетов, в том числе исходящих. При этом указать, какой процесс использует сокет (могут потребоваться права **sudo**, если процесс запускали не вы):

|  |
| --- |
| $ netstat -ntlap |

Посмотреть перечень процессов:

|  |
| --- |
| $ ps ax |

Обратите внимание, откуда были запущены процессы.

Посмотреть пользователей:

|  |
| --- |
| $ cat /etc/passwd  $ ls /home |

Посмотреть неудачные попытки аутентификации:

|  |
| --- |
| $ sudo less /var/log/auth.log |

Посмотреть **crontab**:

|  |
| --- |
| $ sudo less /etc/crontab |

Посмотреть локальный **crontab**:

|  |
| --- |
| $ crontab -l |

Редактировать локальный **crontab**:

|  |
| --- |
| $ crontab -e |

Посмотреть ssh-ключи:

|  |
| --- |
| $ less .ssh/authorized\_keys |

# Практическое задание

1. Настроить сетевой фильтр, чтобы из внешней сети можно было обратиться только к сервисам http и ssh (80 и 443).
2. Запросы, идущие на порт 8080, перенаправлять на порт 80.
3. Настроить доступ по ssh только для вашего IP-адреса (или из всей сети вашего провайдера).
4. \* Создать нового пользователя, сгенерировать для него новые сертификаты. Настроить доступ на сервер вновь созданного пользователя с использованием сертификатов. Подключиться с помощью **putty** или **ssh** без ввода пароля (используя только сертификат).

*Примечание: сертификат может быть подготовлен как в Ubuntu, так и с помощью puttygen в windows.*

1. \* Ваши коллеги, студенты, настраивали VDS-сервер для использования на командном проекте. Через некоторое время сервер был заблокирован. Студенты связались с хостером, он предоставил abuse-письмо (настоящий IP-адрес машины студентов был заменен на 203.0.113.198)

|  |
| --- |
| Dear Sir/Madam,  We have detected abuse from the IP address ( 203.0.113.198 ), which according to a whois lookup is on your network. We would appreciate if you would investigate and take action as appropriate. Any feedback is welcome but not mandatory.  Log lines are given below, but please ask if you require any further information.  (If you are not the correct person to contact about this please accept our apologies - your e-mail address was extracted from the whois record by an automated process. This mail was generated by Fail2Ban.)  IP of the attacker: 203.0.113.198  You can contact us by using: abuse-reply@keyweb.de  Addresses to send to:  audit@ntx.ru  ==================== Excerpt from log for 203.0.113.198 ====================  Note: Local timezone is +0100 (CET)  Nov 27 11:17:33 shared06 sshd[12365]: Invalid user user from 203.0.113.198  Nov 27 11:17:33 shared06 sshd[12365]: pam\_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=203.0.113.198  Nov 27 11:17:34 shared06 sshd[12365]: Failed password for invalid user user from 203.0.113.198 port 44638 ssh2  Nov 27 11:17:34 shared06 sshd[12365]: Received disconnect from 203.0.113.198 port 44638:11: Bye Bye [preauth]  Nov 27 11:17:34 shared06 sshd[12365]: Disconnected from 203.0.113.198 port 44638 [preauth]  ==================== Excerpt from log for 203.0.113.198 ====================  Note: Local timezone is +0100 (CET)  Nov 27 11:20:21 shared04 sshd[13311]: Invalid user user from 203.0.113.198  Nov 27 11:20:21 shared04 sshd[13311]: pam\_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=203.0.113.198  Nov 27 11:20:23 shared04 sshd[13311]: Failed password for invalid user user from 203.0.113.198 port 55092 ssh2  Nov 27 11:20:23 shared04 sshd[13311]: Received disconnect from 203.0.113.198 port 55092:11: Bye Bye [preauth]  Nov 27 11:20:23 shared04 sshd[13311]: Disconnected from 203.0.113.198 port 55092 [preauth] |

Под честное слово хостер на некоторое время запустил машину, чтобы студенты обнаружили вредоносное ПО и вылечили машину.

В качестве тренажера вам будет предложен образ машины: один чистый, второй зараженный. Образ зараженной машины — тренажер, он содержит деактивированное ПО, лишенное настоящей активности, но в него добавлены компоненты, имитирующие ее. IP-адрес с белого (в письме он обозначен как 203.0.113.198) заменен на адрес в 10.0.0.0/8 или 192.168.0.0/24 сети.

Вам необходимо создать виртуальную машину и прикрепить виртуальный жесткий диск с, возможно, зараженной ОС.

В качестве практического задания вы сдаете файл doc / pdf / google doc, в котором описываете, какими командами и что вы проверяли, какие активности были обнаружены и что вы предприняли и рекомендуете сделать, чтобы через некоторое время машину не заблокировали снова.

Ссылка на образ жесткого диска (использовалась версия Virtual Box 6.0): <https://drive.google.com/file/d/1_8XkZVnGoFQArM5t4wYSkpNkMPMhJxHo/view?usp=sharing>

Примечание. *Задания 4 и 5 даны для тех, кому упражнений 1–3 показалось недостаточно.*