

Введение в UNIX-системы

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](#)

[Подготовка тестового сервера](#)

[Подготовим SSH-доступ](#)

[Защищаем пользователей](#)

[Создаем нового](#)

[Блокируем ненужных](#)

[Защищаем SSH](#)

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

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

[Защищаем HTTP](#)

[TLS и SSL-сертификаты](#)

[Проверка mod_ssl и доступ по HTTPS](#)

[Настройка сайта с самоподписанным сертификатом](#)

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

[Тонкая настройка iptables](#)

[Доступ по ssh](#)

[Доступ по http и https](#)

[Другие полезные утилиты](#)

[Практическое задание](#)

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 пропусайте 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 симитировать 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
```

```
user@user-VirtualBox:~$ sudo passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
user@user-VirtualBox:~$ sudo usermod -U root
user@user-VirtualBox:~$ su root
Password:
root@user-VirtualBox:/home/user#
```

Теперь нужно установить **ssh server**:

```
# apt update
# apt install openssh-server
```

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

```
# netstat -ntpl
```

```
root@user-VirtualBox:~# netstat -ntlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.1.1:53            0.0.0.0:*                LISTEN      842/dnsmasq
tcp        0      0 0.0.0.0:22              0.0.0.0:*                LISTEN      2752/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*                LISTEN      668/cupsd
tcp6       0      0 :::22                   :::*                    LISTEN      2752/sshd
tcp6       0      0 :::1:631                :::*                    LISTEN      668/cupsd
```

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:

```
# Authentication:
LoginGraceTime 120
PermitRootLogin prohibit-password
StrictModes yes
```

Закомментируем строку:

```
#PermitRootLogin prohibit-password
```

И добавим:

```
PermitRootLogin yes
```

```
# Authentication:
LoginGraceTime 120
#PermitRootLogin prohibit-password
PermitRootLogin yes_
StrictModes yes
```

Сохраняем:

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

Рестартуем сервер командой **service**:

```
# service sshd restart
```

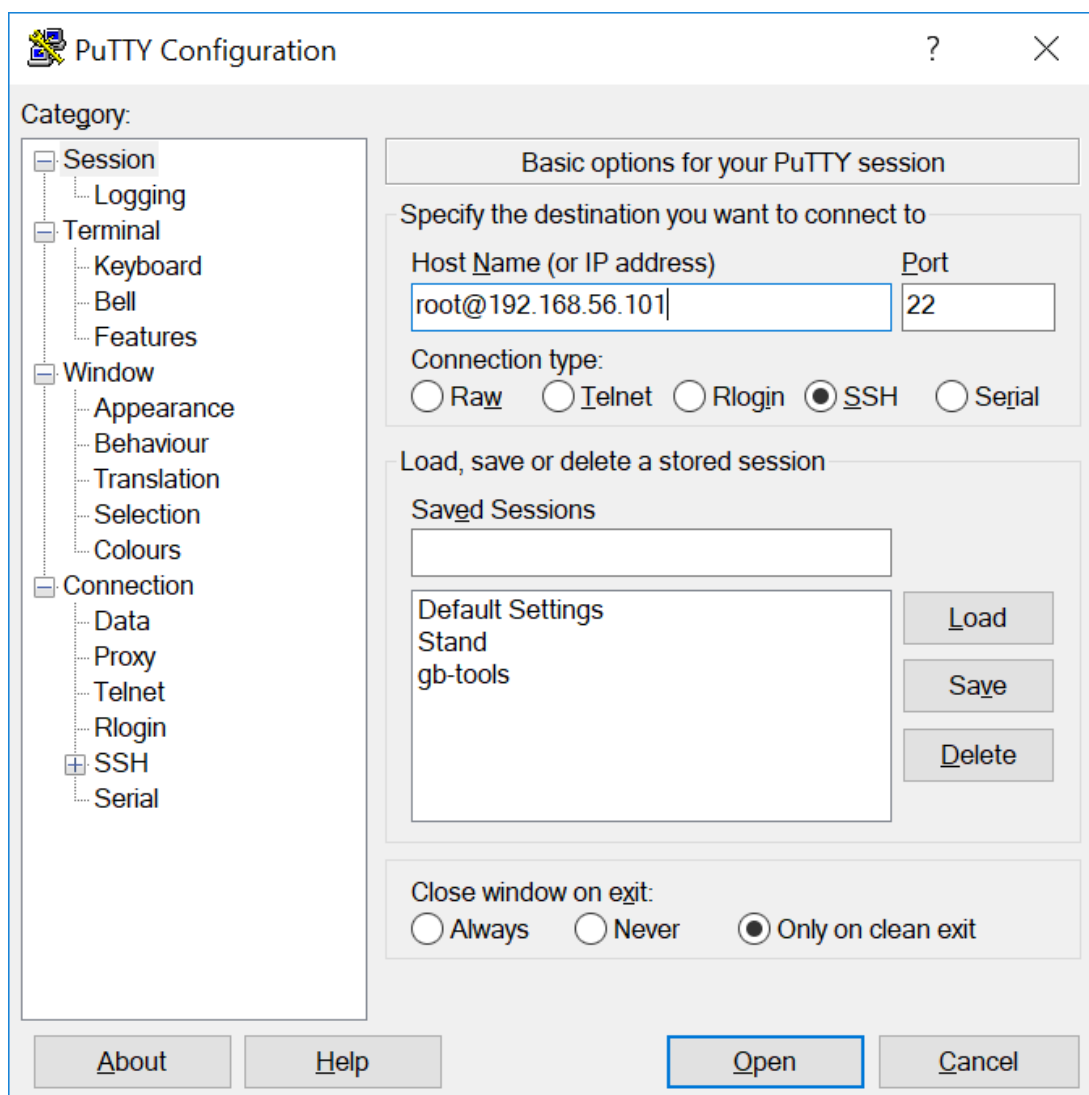
Либо, если вы уверены, что используется система инициализации **SystemD** (Ubuntu 16.0 и выше), то:

```
# systemctl restart sshd
```

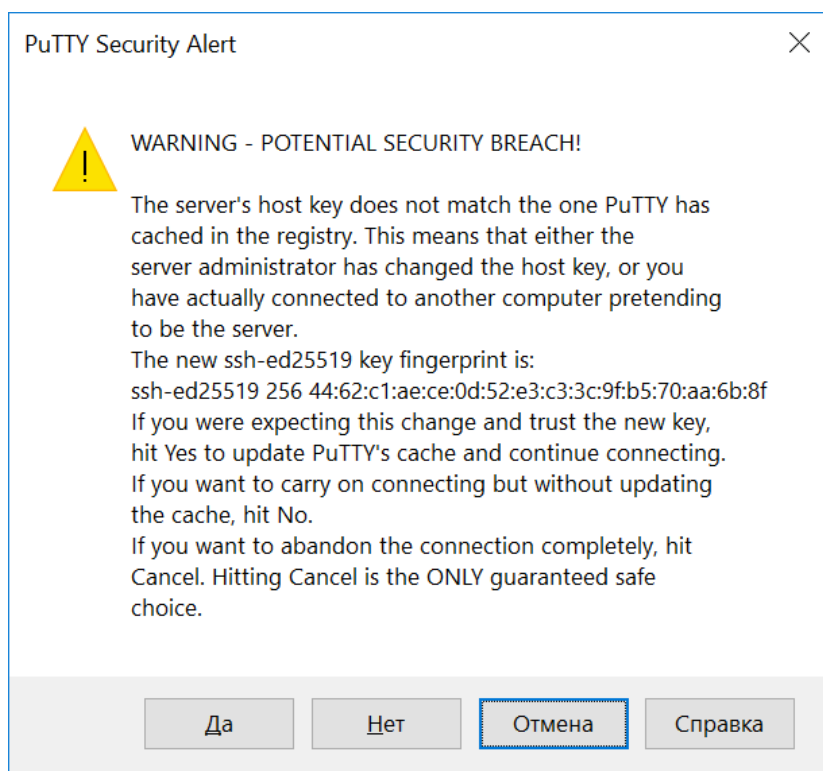
Посмотрим наш IP-адрес:

```
root@user-VirtualBox:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:41:7c:5e brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.9/24 brd 10.0.2.255 scope global dynamic enp0s3
        valid_lft 1113sec preferred_lft 1113sec
    inet6 fe80::ba26:4a6a:77f:2673/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:9a:f2:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic enp0s8
        valid_lft 1027sec preferred_lft 1027sec
    inet6 fe80::32aa:532e:6ca:3955/64 scope link
        valid_lft forever preferred_lft forever
```

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



Предупреждение, что ключ сервера нам не известен:



Жмем «Да».

Вводим пароль. Зашли:

```
root@user-VirtualBox: ~  
root@192.168.56.101's password:  
Using username "root".  
root@192.168.56.101's password:  
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-29-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
  
294 packages can be updated.  
213 updates are security updates.  
  
New release '18.04.2 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
root@user-VirtualBox:~#
```

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:

Активация Виртуального Сервера



FirstByte.ru billing@firstbyte.ru
Вам: Test Test ^

сегодня в 21:11

Активация Виртуального сервера

Здравствуйте, Test Test!

Настоящим письмом уведомляем, что на ваше имя был зарегистрирован Виртуальный Сервер. Предлагаем распечатать данное сообщение для удобства использования в дальнейшем.

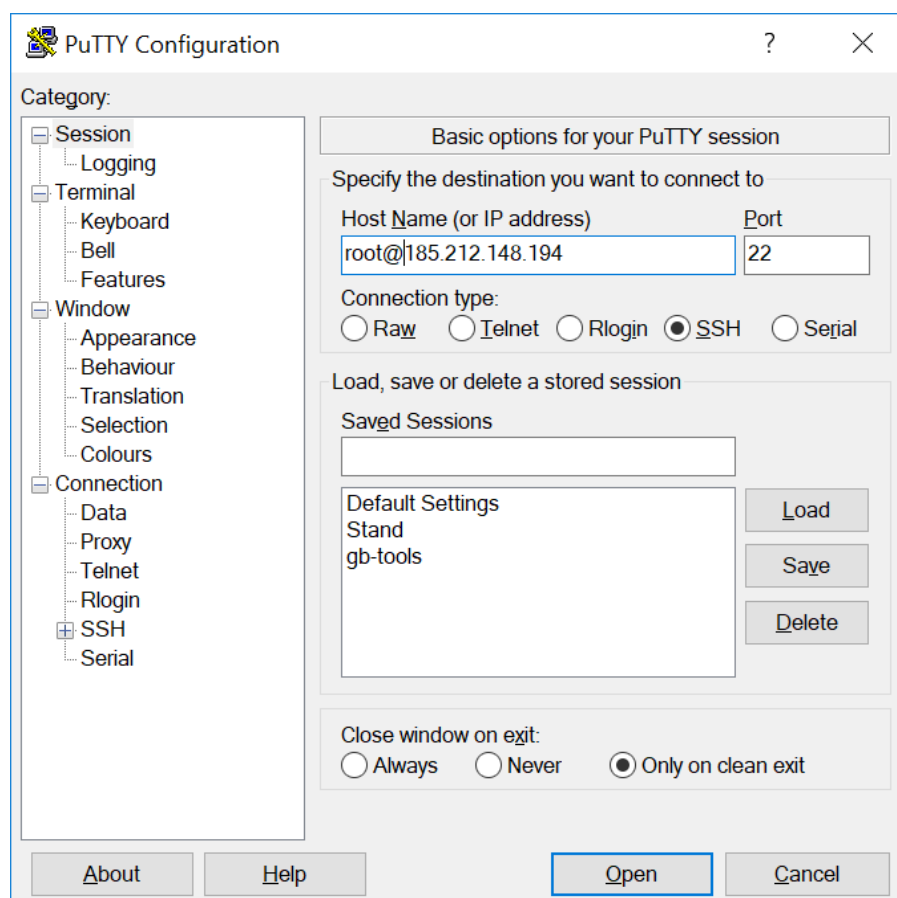
Информация о сервере

- Тарифный план: MSK-OVZ-SSD-1
- Дата открытия: 2019-02-21
- Доменное имя: test2.example.com
- IP-адрес сервера: 185.212.148.194
- Пользователь: root
- Пароль: ~~root~~

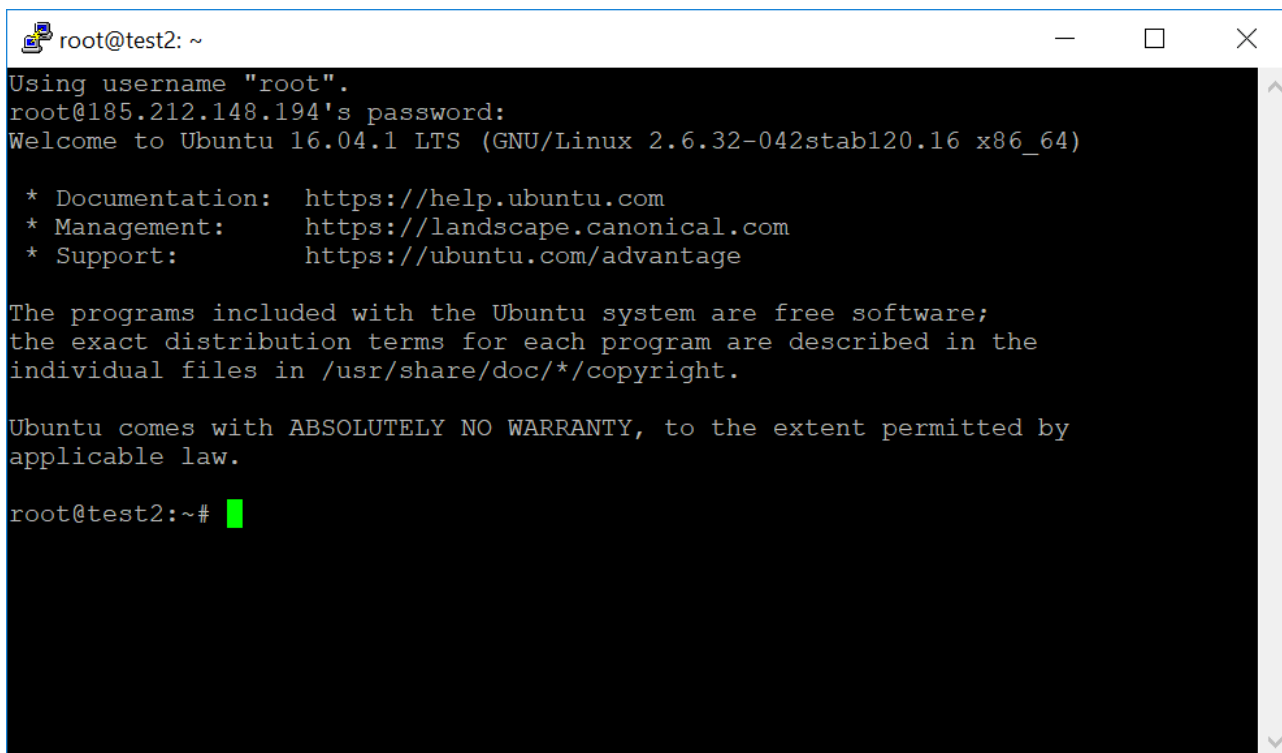
VMmanager OVZ - внешняя панель управления сервером

Во внешней панели управления сервером вы можете перезапустить сервер, переустановить операционную систему, посмотреть статистику по нагрузке сервера и трафику

Подключаемся по ssh:



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

A terminal window titled 'root@test2: ~' with standard window controls. The terminal text shows a login process for 'root' on IP '185.212.148.194'. It displays the Ubuntu version '16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)' and provides links for documentation, management, and support. It also includes a disclaimer about warranty. The prompt ends at 'root@test2:~#' with a green cursor.

```
root@test2: ~
Using username "root".
root@185.212.148.194's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@test2:~#
```

Подключились.

Теперь начинаем работать.

Защищаем пользователей

Создаем нового пользователя

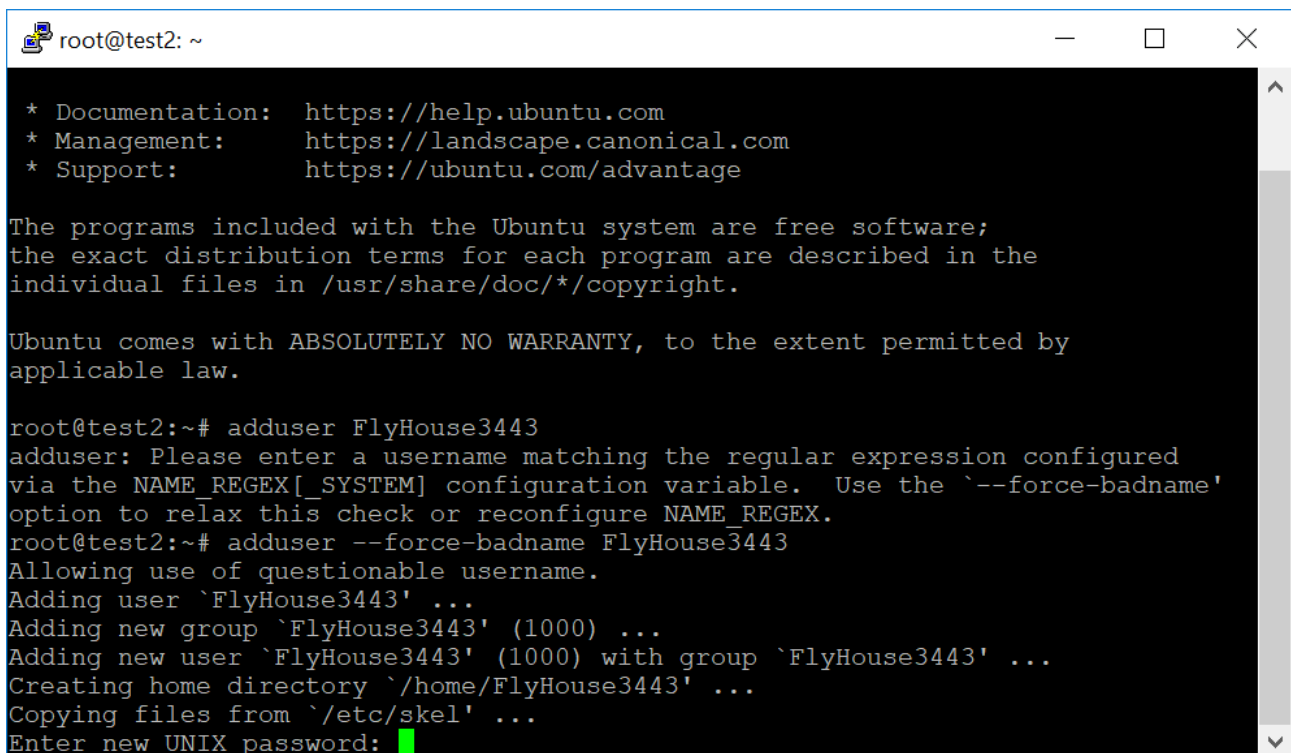
Теперь мы создадим пользователя, под которым будем работать. Этот пользователь (как и другие) не должен иметь словарного имени — то есть такие пользователи, как **user**, **deploy**, **admin** и подобные не подойдут.

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.



```
root@test2: ~
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

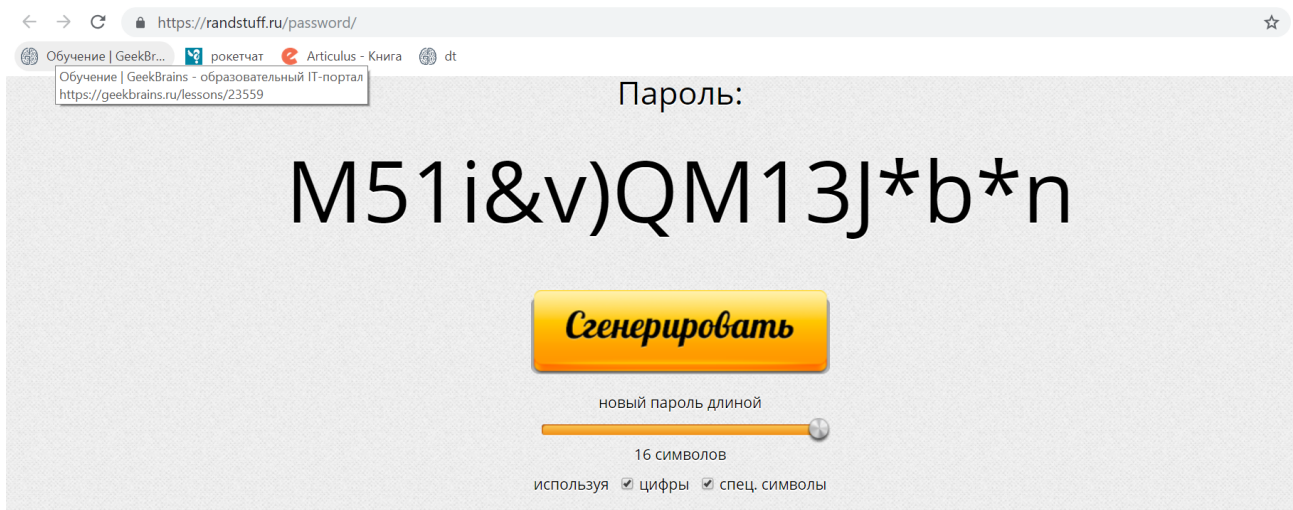
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@test2:~# adduser FlyHouse3443
adduser: Please enter a username matching the regular expression configured
via the NAME_REGEX[_SYSTEM] configuration variable.  Use the '--force-badname'
option to relax this check or reconfigure NAME_REGEX.
root@test2:~# adduser --force-badname FlyHouse3443
Allowing use of questionable username.
Adding user `FlyHouse3443' ...
Adding new group `FlyHouse3443' (1000) ...
Adding new user `FlyHouse3443' (1000) with group `FlyHouse3443' ...
Creating home directory `/home/FlyHouse3443' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: 
```

Утилита попытается 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
```

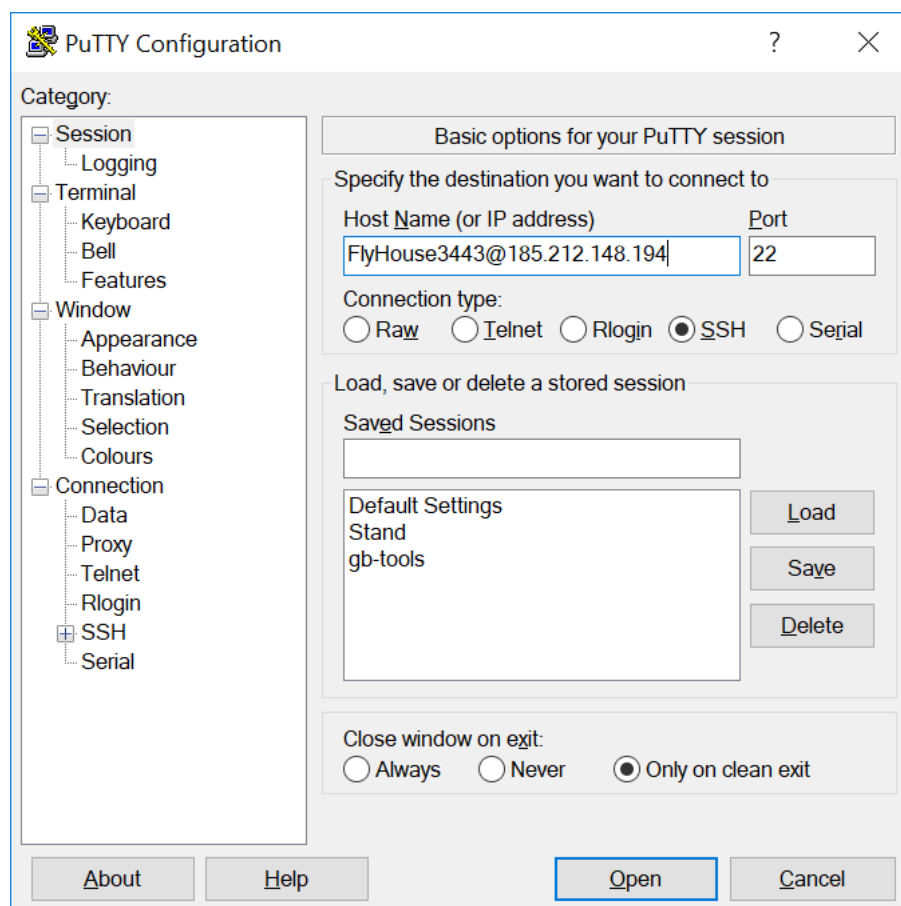
Замените несколько символов.

```
Z53i&v(Q!6Np*&sT
```

Добавим пользователя в группу **sudo**:

```
#usermod -a -G sudo FlyHouse3443
```

Теперь подключаемся созданным пользователем:



Вводим созданный пароль:

```
FlyHouse3443@test2: ~
Using username "FlyHouse3443".
FlyHouse3443@185.212.148.194's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

FlyHouse3443@test2:~$
```

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

```
$ sudo id
```

```
FlyHouse3443@test2:~$ sudo id
[sudo] password for FlyHouse3443:
uid=0(root) gid=0(root) groups=0(root)
FlyHouse3443@test2:~$
```

Блокируем ненужных пользователей

Теперь можно заблокировать суперпользователя:

```
$ sudo usermod -L root
```

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

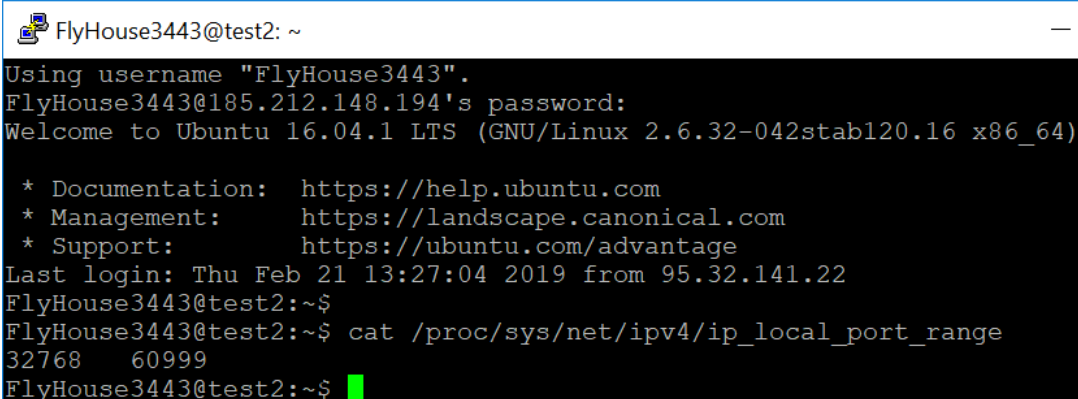
```
$ sudo usermod -L ubuntu
```

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
```



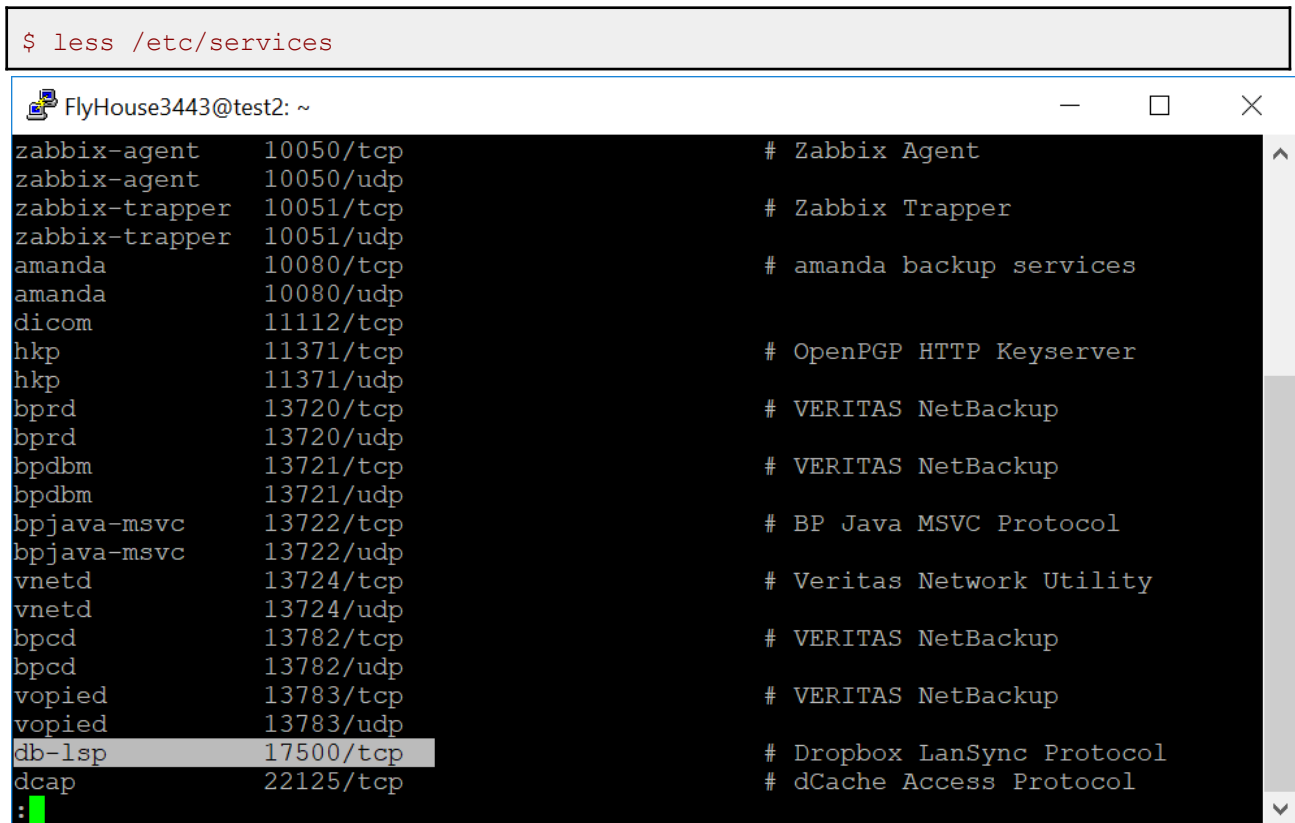
```
FlyHouse3443@test2: ~
Using username "FlyHouse3443".
FlyHouse3443@185.212.148.194's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Thu Feb 21 13:27:04 2019 from 95.32.141.22
FlyHouse3443@test2:~$
FlyHouse3443@test2:~$ cat /proc/sys/net/ipv4/ip_local_port_range
32768 60999
FlyHouse3443@test2:~$
```

Можем выбрать любой из случайных портов или

порт неиспользуемого сервера.

```
$ less /etc/services
```



Service	Port/Protocol	Comment
zabbix-agent	10050/tcp	# Zabbix Agent
zabbix-agent	10050/udp	
zabbix-trapper	10051/tcp	# Zabbix Trapper
zabbix-trapper	10051/udp	
amanda	10080/tcp	# amanda backup services
amanda	10080/udp	
dicom	11112/tcp	
hkp	11371/tcp	# OpenPGP HTTP Keyserver
hkp	11371/udp	
bprd	13720/tcp	# VERITAS NetBackup
bprd	13720/udp	
bpdbm	13721/tcp	# VERITAS NetBackup
bpdbm	13721/udp	
bpjava-msvc	13722/tcp	# BP Java MSVC Protocol
bpjava-msvc	13722/udp	
vnetd	13724/tcp	# Veritas Network Utility
vnetd	13724/udp	
bpcd	13782/tcp	# VERITAS NetBackup
bpcd	13782/udp	
vopied	13783/tcp	# VERITAS NetBackup
vopied	13783/udp	
db-lsp	17500/tcp	# Dropbox LanSync Protocol
dcap	22125/tcp	# dCache Access Protocol

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
```

```
FlyHouse3443@test2: ~
GNU nano 2.5.3 File: /etc/ssh/sshd config
# 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

[ Read 88 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Меняем 22 на выбранный 39192:

```
FlyHouse3443@test2: ~
GNU nano 2.5.3 File: /etc/ssh/sshd config Modified
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 39192
# 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

[ Read 88 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

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

Если же предпочитаете vi, то :wq!

Нужно перезапустить sshd-сервер.

Рестартуем сервер командой **service**:

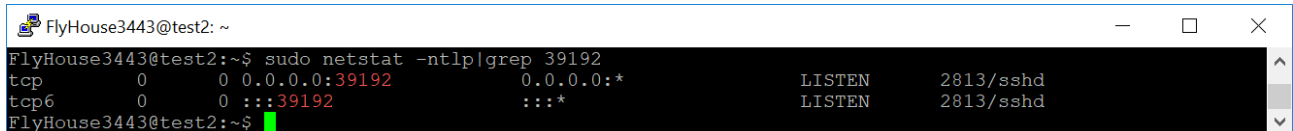
```
# service sshd restart
```

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

```
# systemctl restart sshd
```

Проверяем:

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

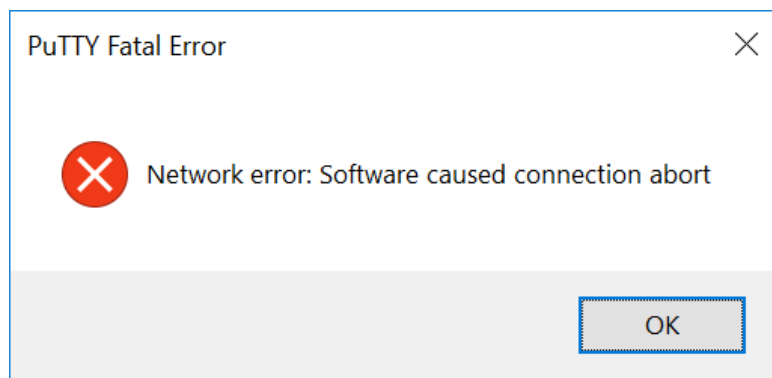


```
FlyHouse3443@test2: ~  
FlyHouse3443@test2:~$ sudo netstat -ntlp|grep 39192  
tcp        0      0 0.0.0.0:39192      0.0.0.0:*        LISTEN     2813/sshd  
tcp6       0      0 :::39192          :::*             LISTEN     2813/sshd  
FlyHouse3443@test2:~$
```

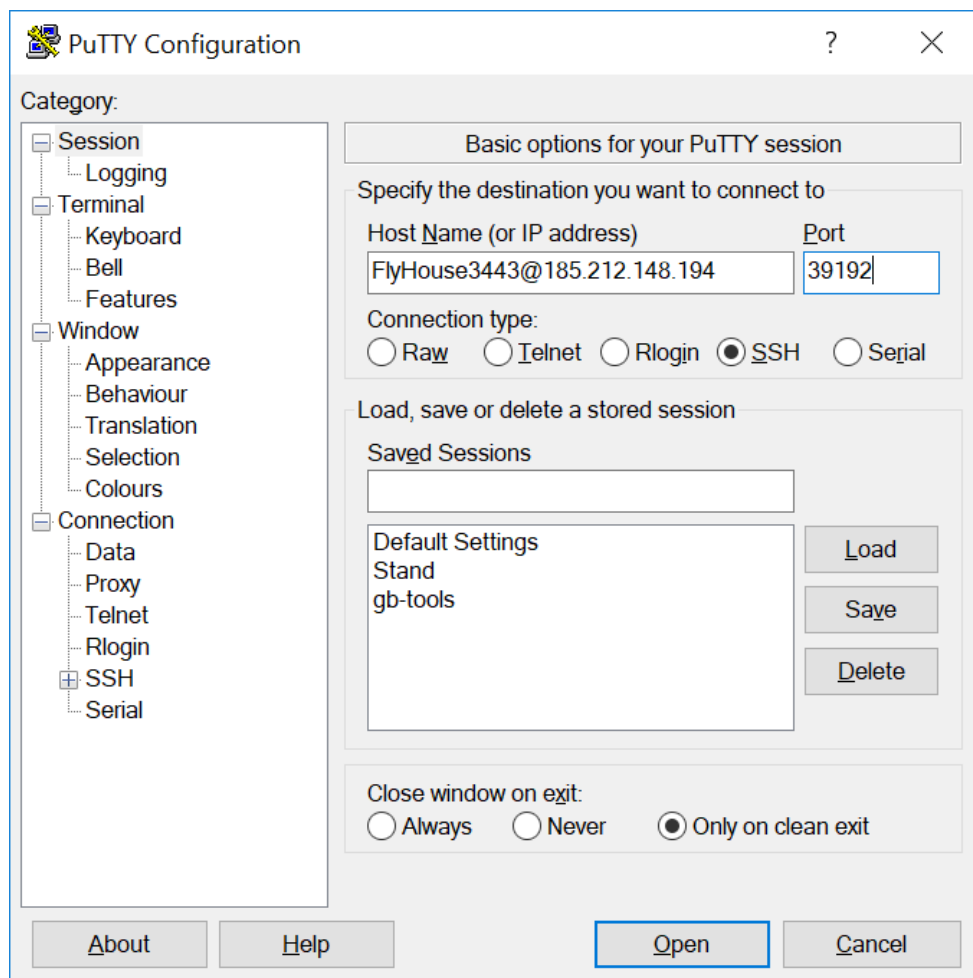
Обратите внимание, что все текущие сессии до закрытия так и останутся на 22/TCP.

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

```
$ sudo systemctl reboot
```



Подключаемся по новому порту:



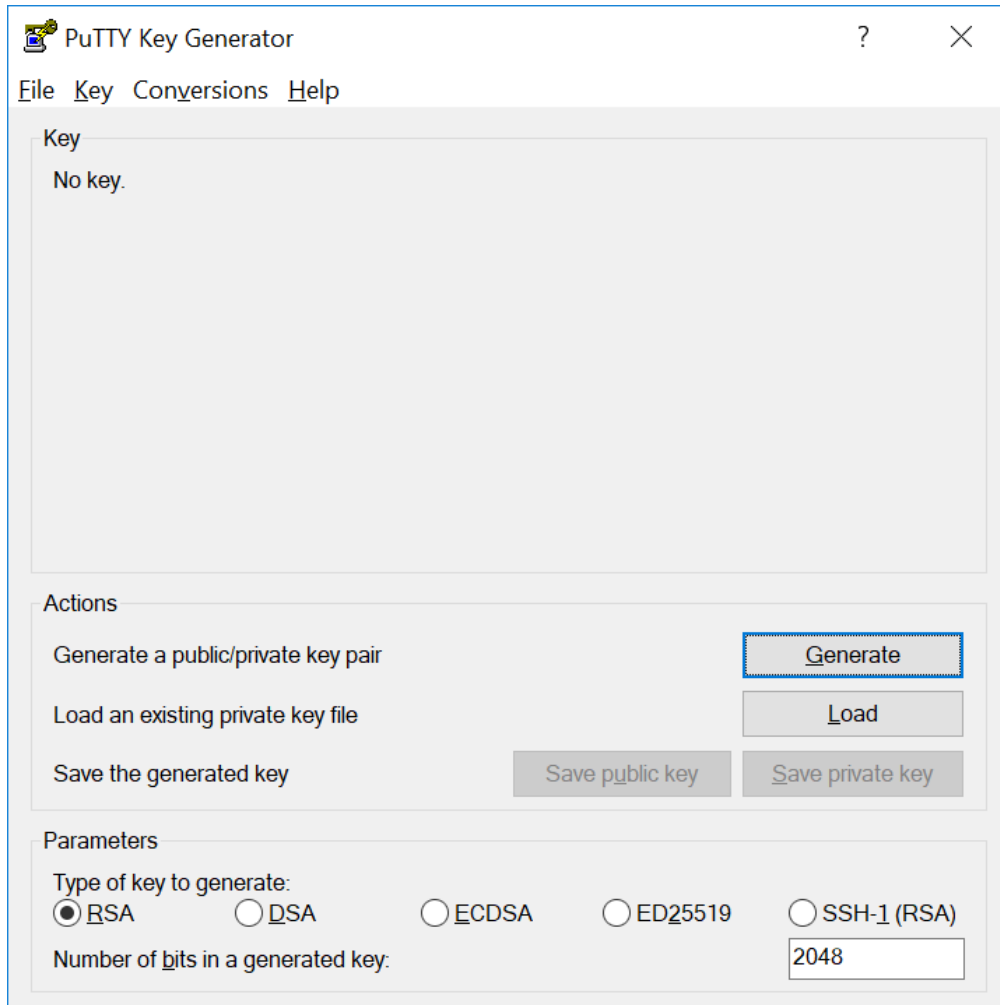
Мы подключились по паролю:

```
FlyHouse3443@test2: ~  
Using username "FlyHouse3443".  
FlyHouse3443@185.212.148.194's password:  
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
Last login: Thu Feb 21 13:37:01 2019 from 95.32.141.22  
FlyHouse3443@test2:~$
```

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.

PutTY Key Generator

File Key Conversions Help

Key

Please generate some randomness by moving the mouse over the blank area.

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

Save the generated key Save public key Save private key

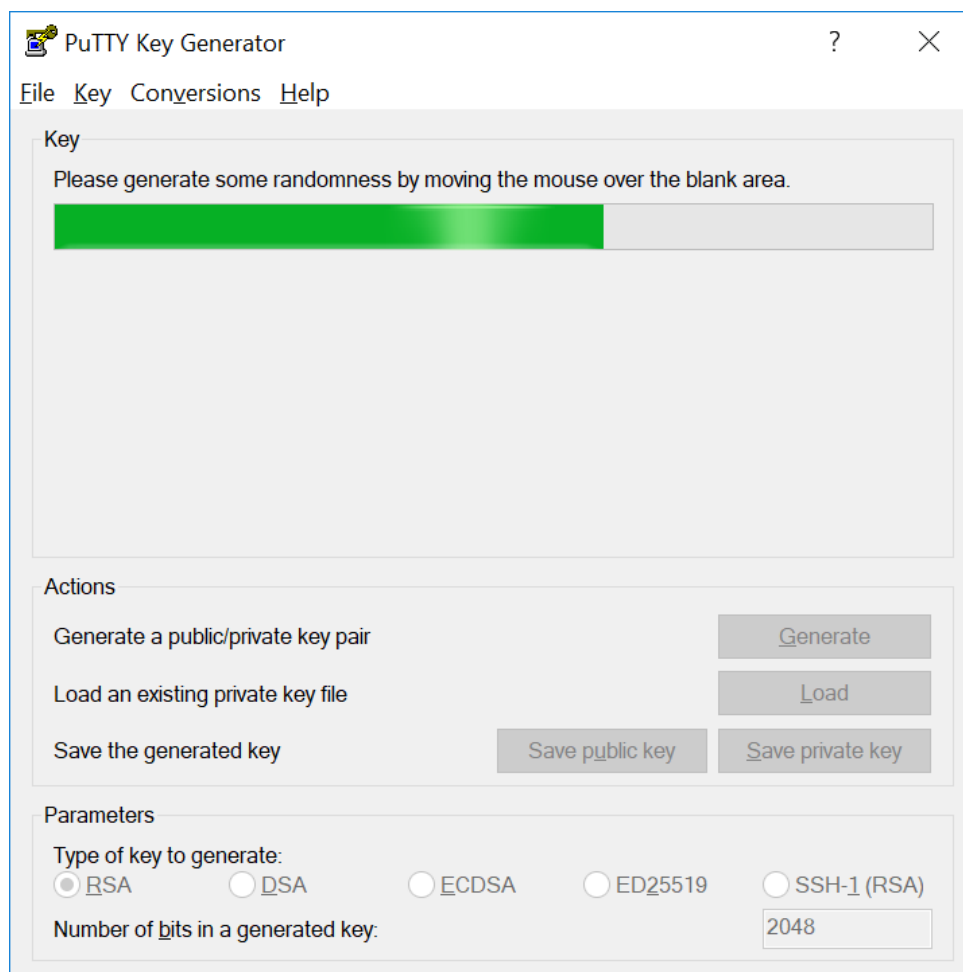
Parameters

Type of key to generate:


☒ RSA ☐ DSA ☐ ECDSA ☐ ED25519 ☐ SSH-1 (RSA)

Number of bits in a generated key:

Двигаем, ключ создается:



Ключ сгенерирован:

 PuTTY Key Generator ? ×

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAtMMHYud0UnNYZmOmnQh9X2we9ByY9ole1
tbOzO/qdSymdK3T9uOKvVqDXx/d9Y0EHRmEGvWb/SFhOOUc3MBDd4yZL0KzdfHz
LVpDbswQjTZXBnhBMcSzYkX/O5E6U
+cWjkyKneXXKB8mwFFi8vCRWnBh2l3wc0ueuq8CA2Y5fkBegr11SnPtkJA1SFOBw9V
```

Key fingerprint: ssh-rsa 2048 ed:c1:23:1e:37:00:22:cf:04:ea:1c:84:24:d3:c8:f4

Key comment: rsa-key-20190222

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair Generate

Load an existing private key file Load

Save the generated key Save public key Save private key

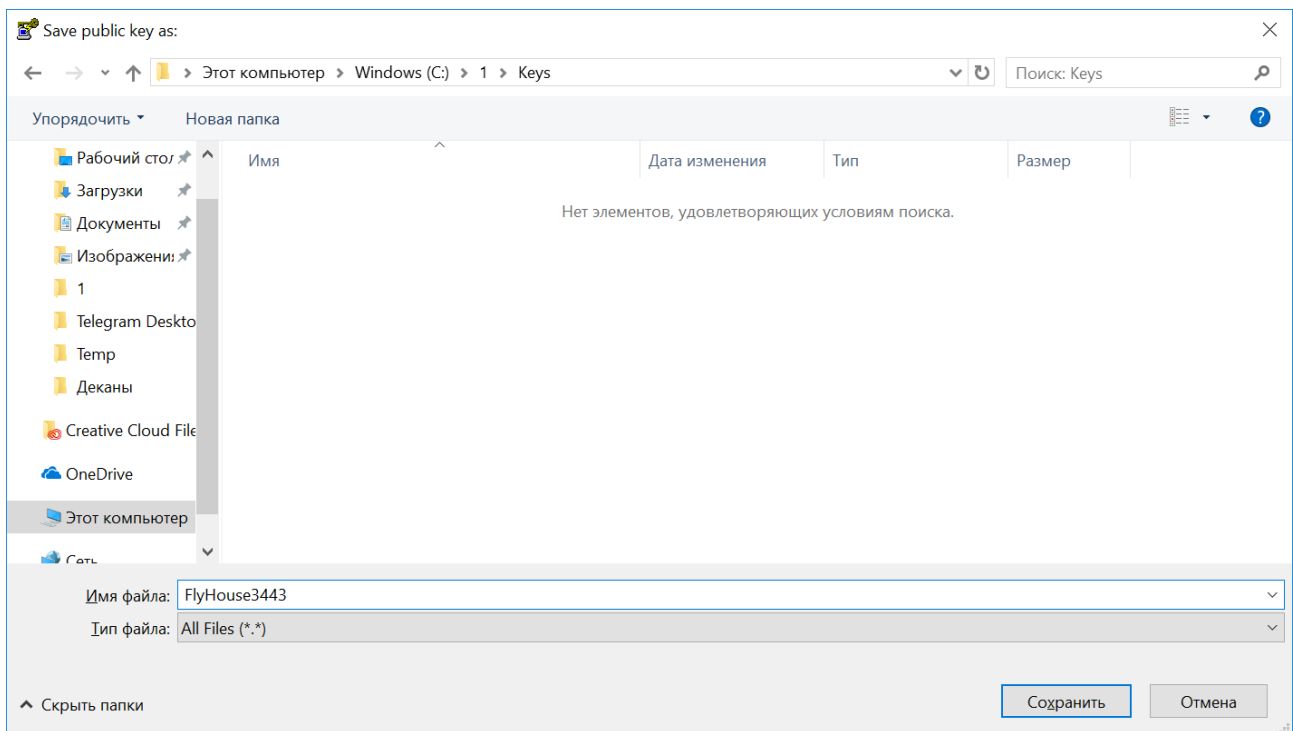
Parameters

Type of key to generate:

☒ RSA ☐ DSA ☐ ECDSA ☐ ED25519 ☐ SSH-1 (RSA)

Number of bits in a generated key: 2048

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**:

PuTTY Key Generator

File Key Conversions Help

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAiMMHYud0UnNYZmOmnQh9X2we9ByY9ole1
tbOzO/qdSymdK3T9uOKvVkqDXx/d9Y0EHRmEGvWb/SFhOOuc3MBDd4yZL0KzdfHz
LVpDbSwQjTZXBnhBMcSzYkX/O5E6U
+cWjkyKneXXKB8mwFFi8vCRWnBh2I3wc0ueuq8CA2Y5fkBegr1SnPtkJA1SFOBw9V
```

Key fingerprint: ssh-rsa 2048 ed:c1:23:1e:37:00:22:cf:04:ea:1c:84:24:d3:c8:f4

Key comment: rsa-key-20190222

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair **Generate**

Load an existing private key file **Load**

Save the generated key **Save public key** **Save private key**

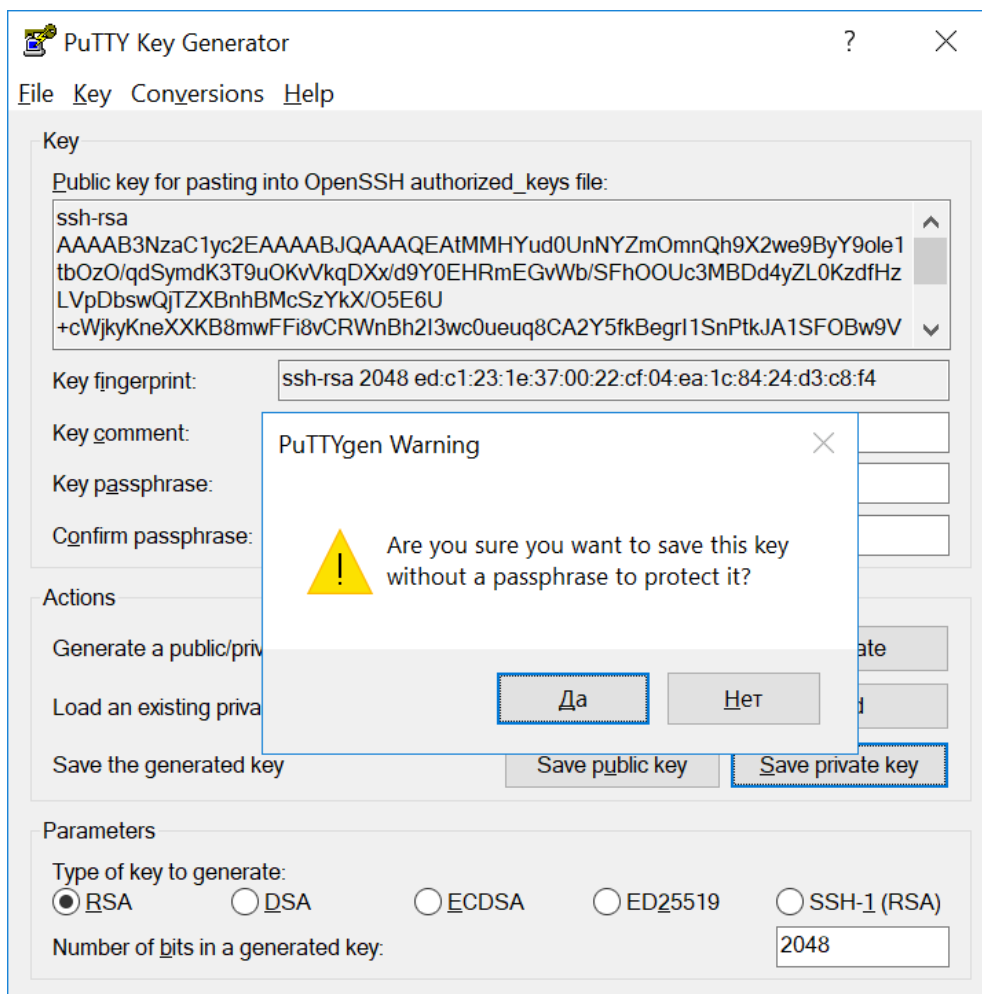
Parameters

Type of key to generate:

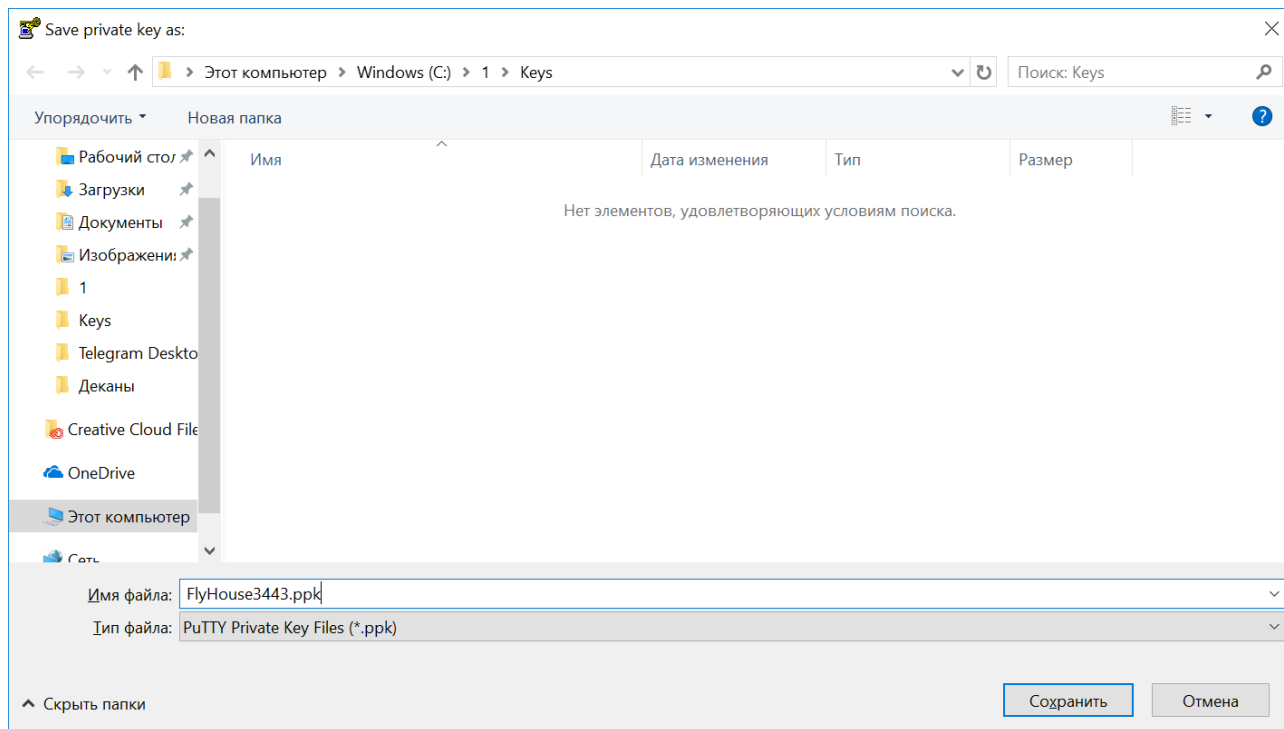
☒ RSA ☐ DSA ☐ ECDSA ☐ ED25519 ☐ SSH-1 (RSA)

Number of bits in a generated key: 2048

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.

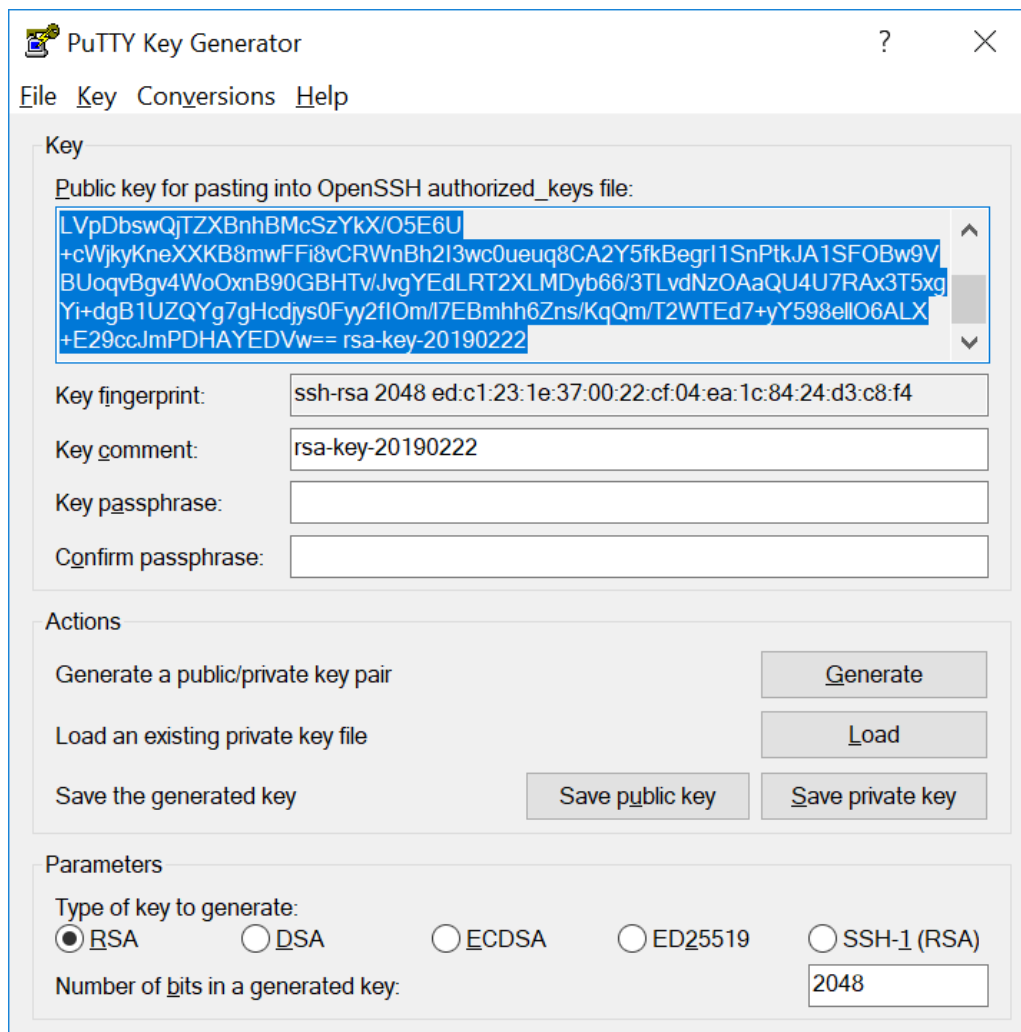


Жмем «Да».



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
```

```
FlyHouse3443@test2: ~/ssh
Using username "FlyHouse3443".
FlyHouse3443@185.212.148.194's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 11:48:27 2019 from 95.32.141.22
FlyHouse3443@test2:~$ls -la
total 28
drwxr-xr-x 3 FlyHouse3443 FlyHouse3443 4096 Feb 21 13:28 .
drwxr-xr-x 3 root        root        4096 Feb 21 13:16 ..
-rw----- 1 FlyHouse3443 FlyHouse3443  426 Feb 21 13:51 .bash_history
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443  220 Feb 21 13:16 .bash_logout
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443 3771 Feb 21 13:16 .bashrc
drwx----- 2 FlyHouse3443 FlyHouse3443 4096 Feb 21 13:27 .cache
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443   655 Feb 21 13:16 .profile
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443    0 Feb 21 13:28 .sudo_as_admin_successful
FlyHouse3443@test2:~$ mkdir .ssh
FlyHouse3443@test2:~$ cd .ssh
FlyHouse3443@test2:~/.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:

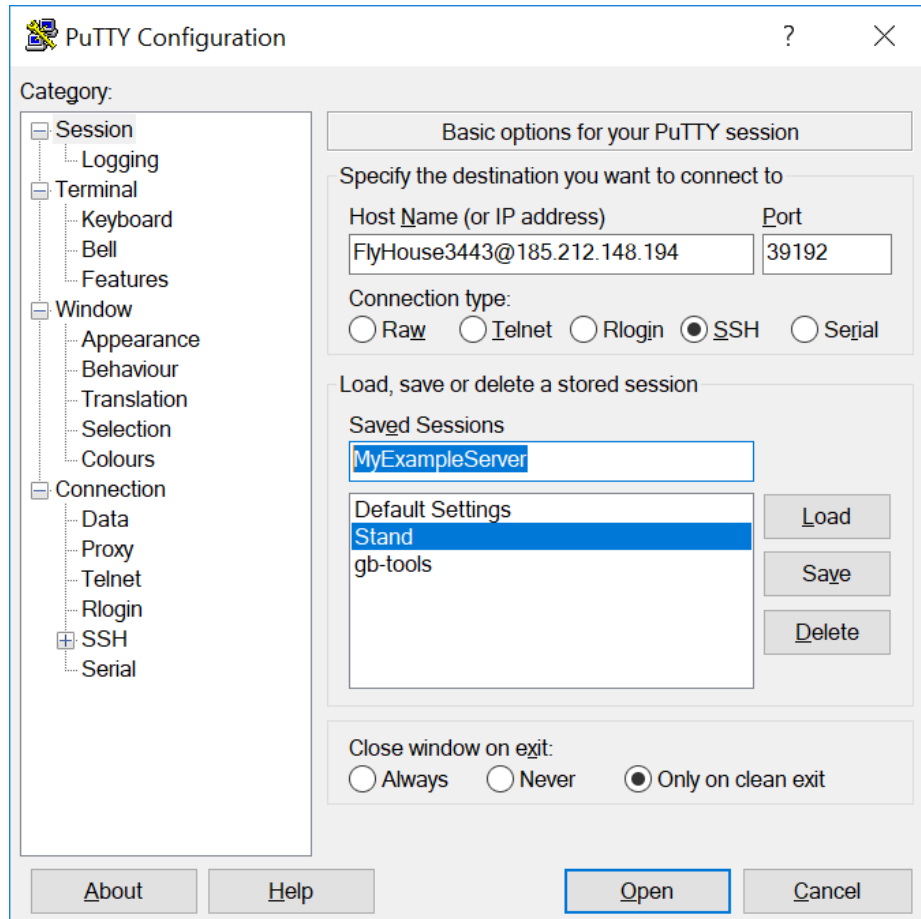
```
FlyHouse3443@test2: ~/ssh
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 11:48:27 2019 from 95.32.141.22
FlyHouse3443@test2:~$ls -la
total 28
drwxr-xr-x 3 FlyHouse3443 FlyHouse3443 4096 Feb 21 13:28 .
drwxr-xr-x 3 root        root        4096 Feb 21 13:16 ..
-rw----- 1 FlyHouse3443 FlyHouse3443  426 Feb 21 13:51 .bash_history
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443  220 Feb 21 13:16 .bash_logout
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443 3771 Feb 21 13:16 .bashrc
drwx----- 2 FlyHouse3443 FlyHouse3443 4096 Feb 21 13:27 .cache
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443   655 Feb 21 13:16 .profile
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443    0 Feb 21 13:28 .sudo_as_admin_successful
FlyHouse3443@test2:~$ mkdir .ssh
FlyHouse3443@test2:~$ cd .ssh
FlyHouse3443@test2:~/.ssh$cat >> authorized_keys
ssh-rsa AAAAB3NzaClyc2EAAAABJQAAAQEAtMMHYud0UnNYZmOmnQh9X2we9ByY9oleltbOzO/qdSym
dK3T9uOKvVlkqDXx/d9Y0EHRmEGvWb/SFh0OUc3MBDD4yZL0KzdfHzLVpDbswQjTZXBnhBMCSzYkX/O5E
6U+cWjkyKneXXKB8mwFFi8vCRWnBh2I3wc0ueuq8CA2Y5fkBegrI1SnPtkJA1SFOBw9VBuOqvBgV4WoO
xnB90GBHTv/JvgYEDLRT2XLMdyb66/3TLvdNz0AaQU4U7RAx3T5xgYi+dgB1UZQYg7gHcdjys0Fyy2fI
Om/l7EBmhh6Zns/KqQm/T2WTEd7+yY598ell06ALX+E29ccJmPDHAYEDVw== rsa-key-20190222
FlyHouse3443@test2:~/.ssh$
```

Затем жмем Enter и завершаем вводом Ctrl-D.

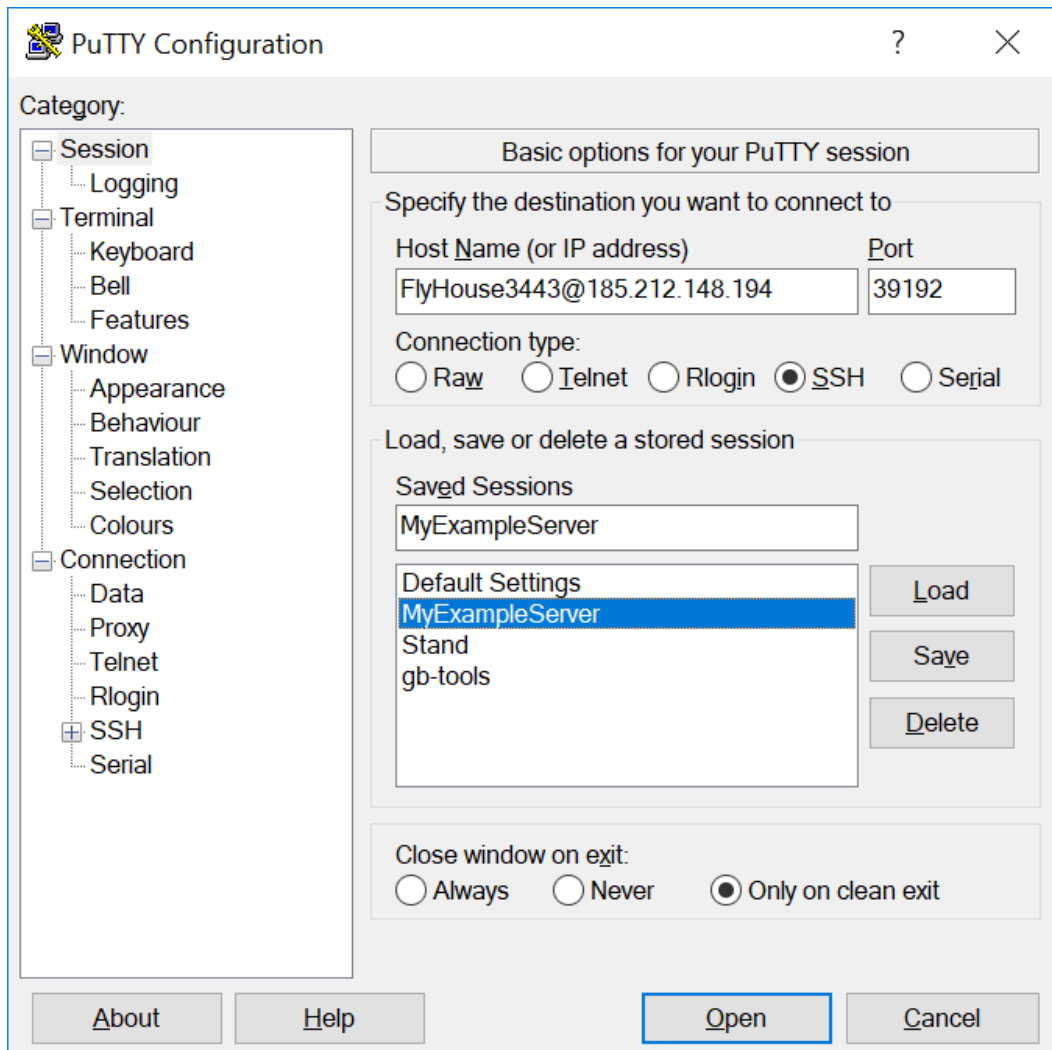
И последнее: права на этот файл должны быть 600, то есть чтение и запись для владельца. Назначим их:

```
$ chmod 600 authorized_keys
```

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

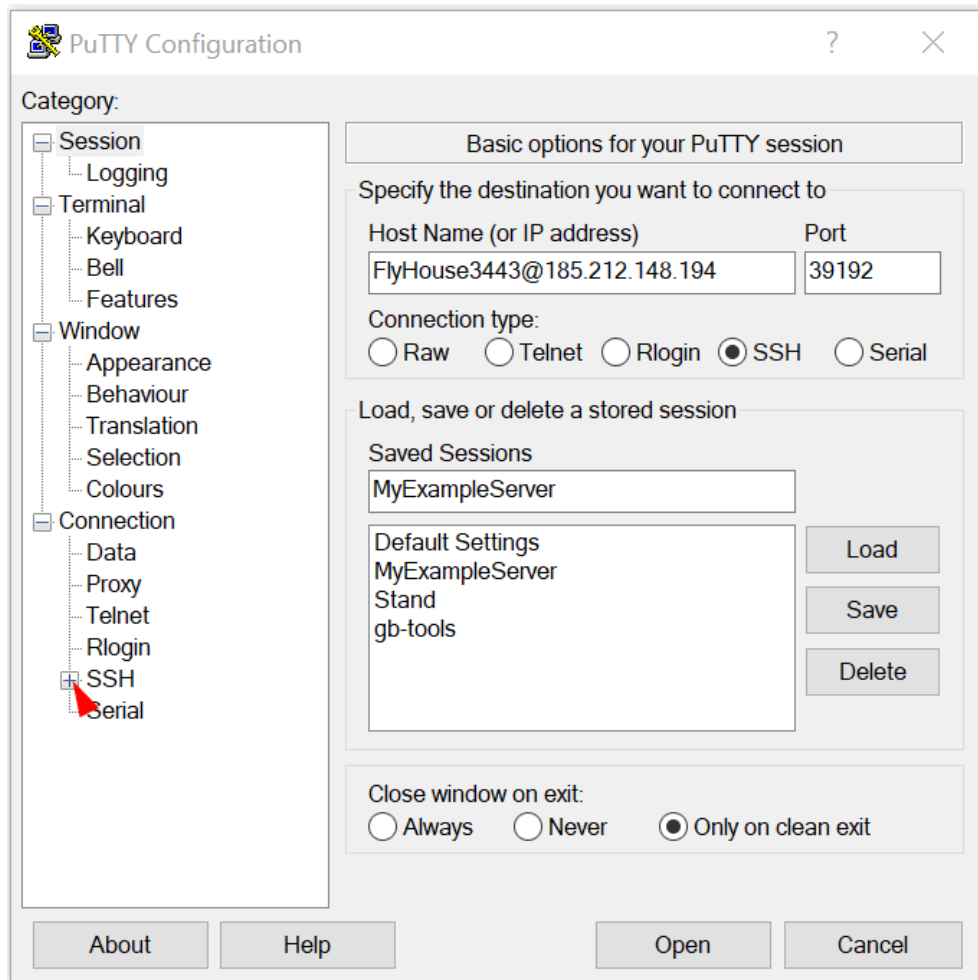


Нажимаем **Save**.

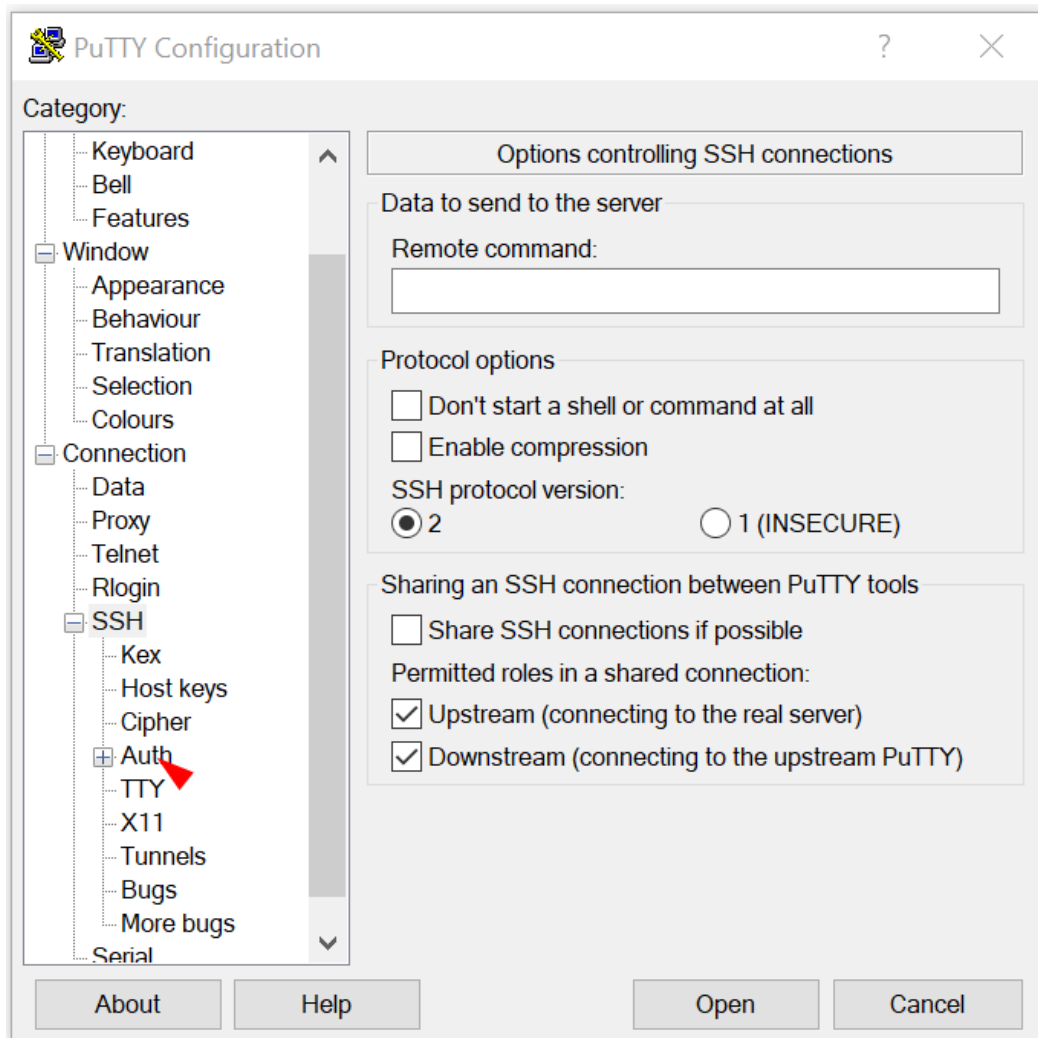


Теперь выполним настройки для сервера в **Saved Sessions**.

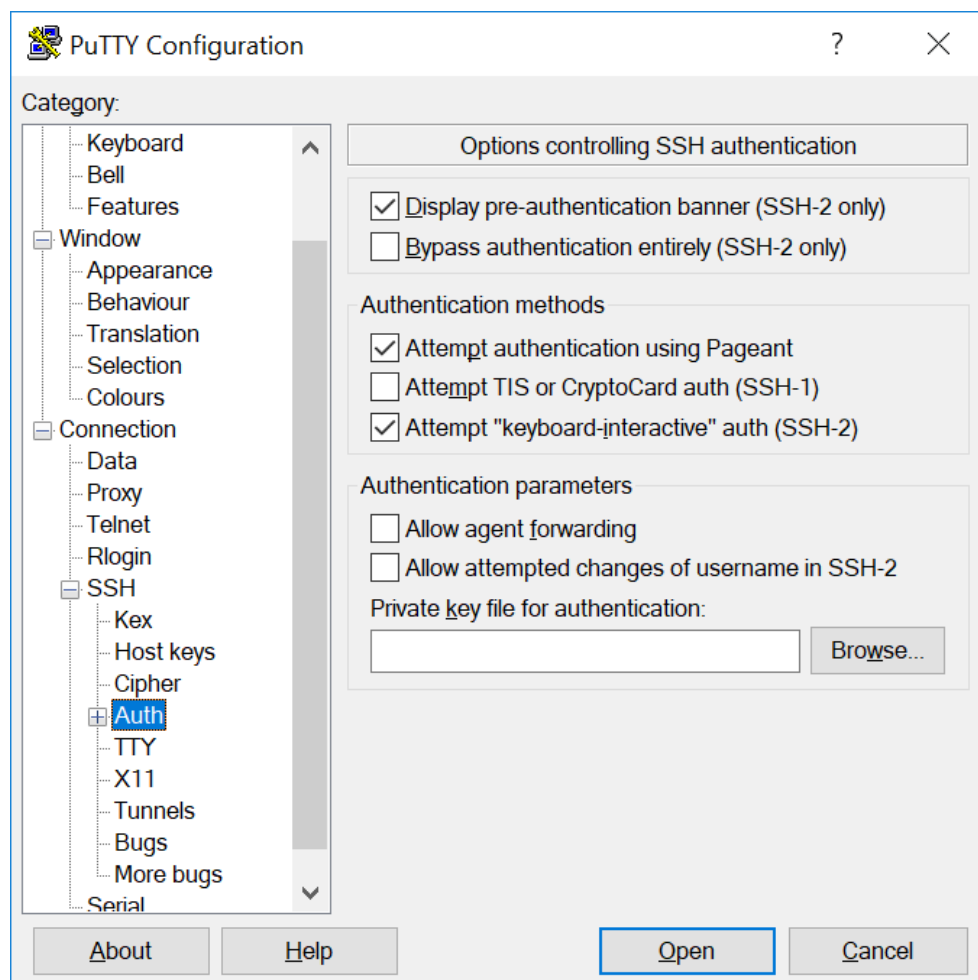
Надо настроить ключи. Кликаем на SSH:



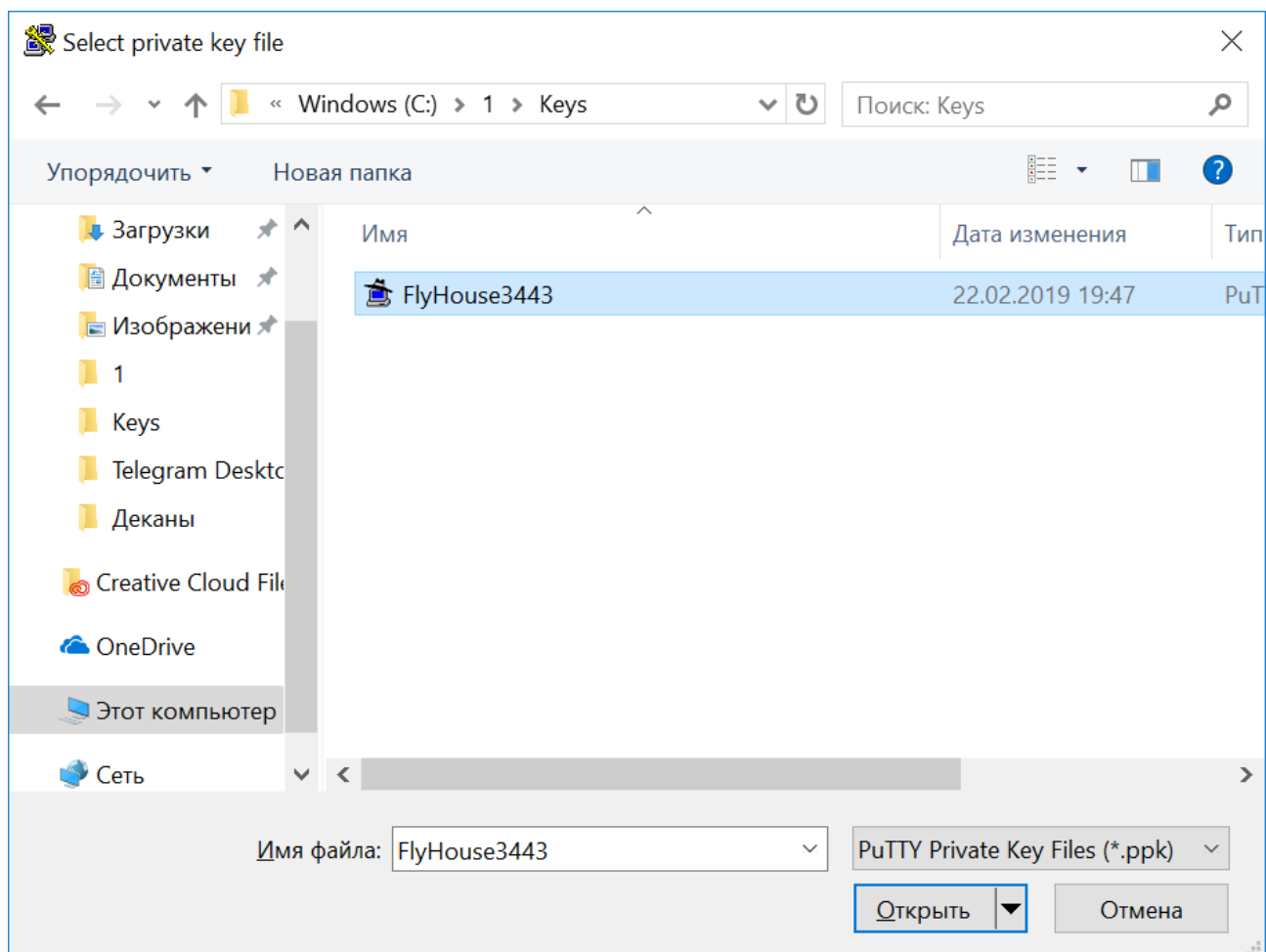
Кликаем на **Auth**:



Убеждаемся, что раздел **Auth** выбран:

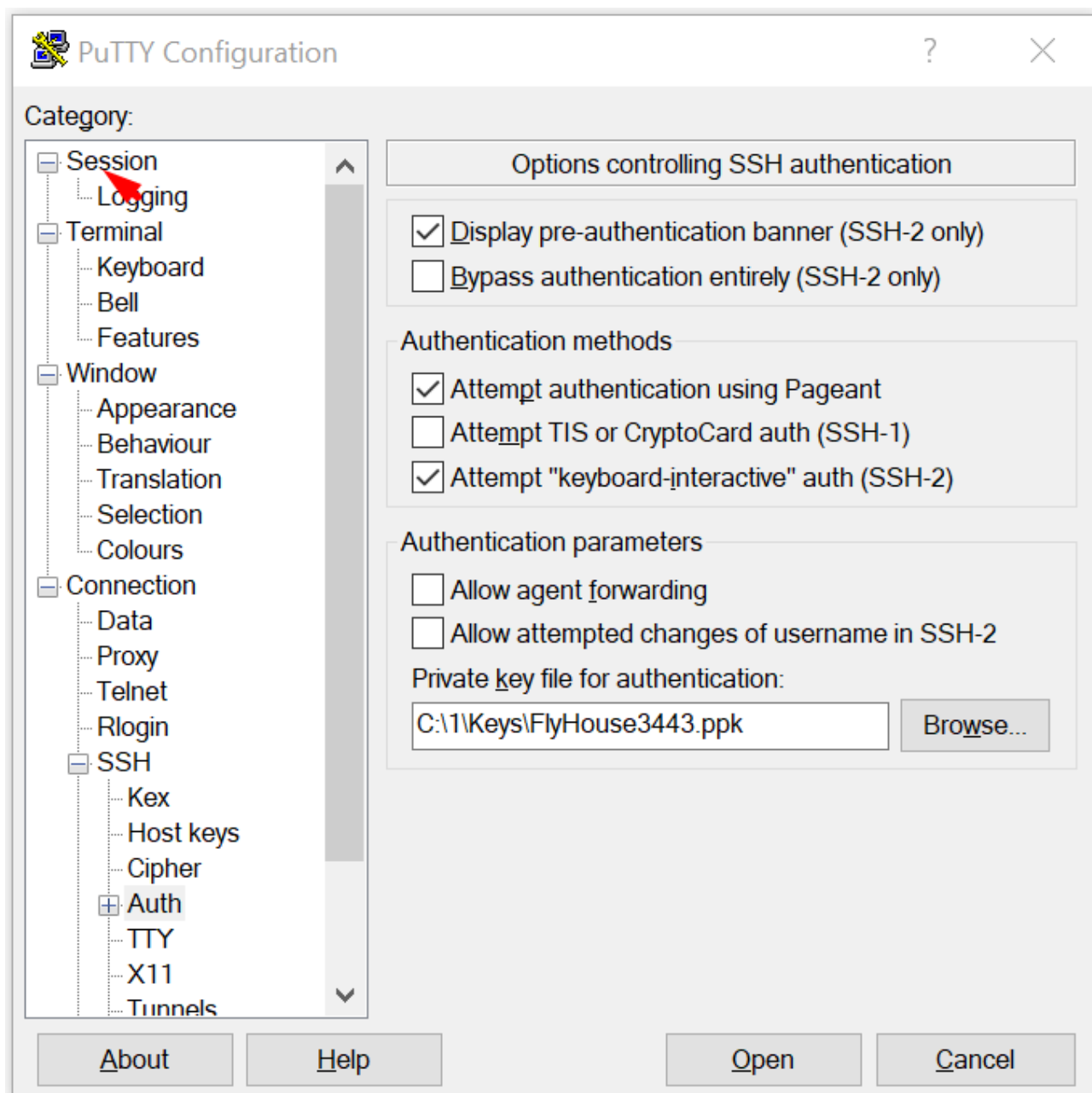


Жмем **Browse** и выбираем сохраненный приватный ключ:

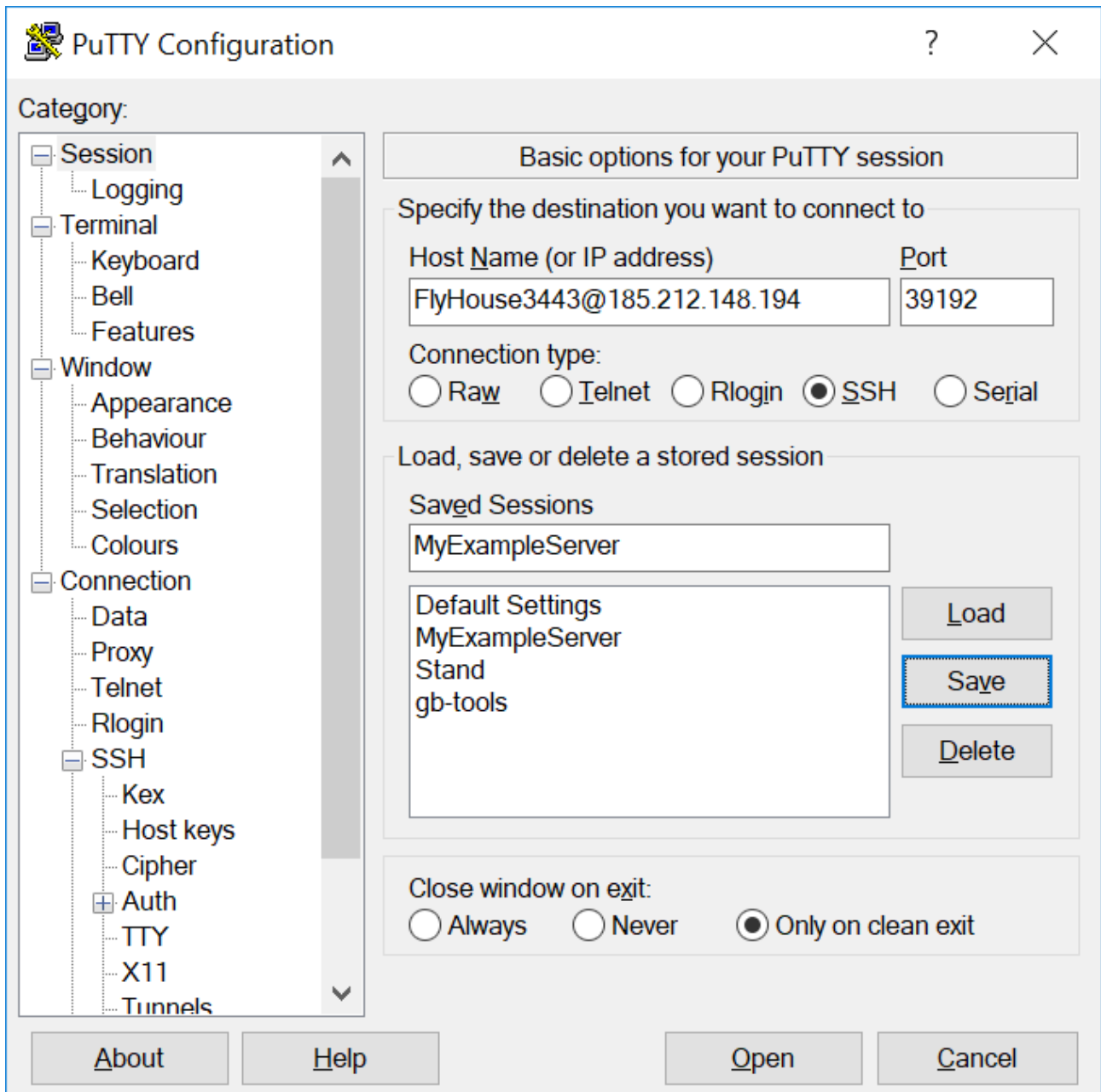


Это **ppk** (Windows по умолчанию скрывает разрешение).

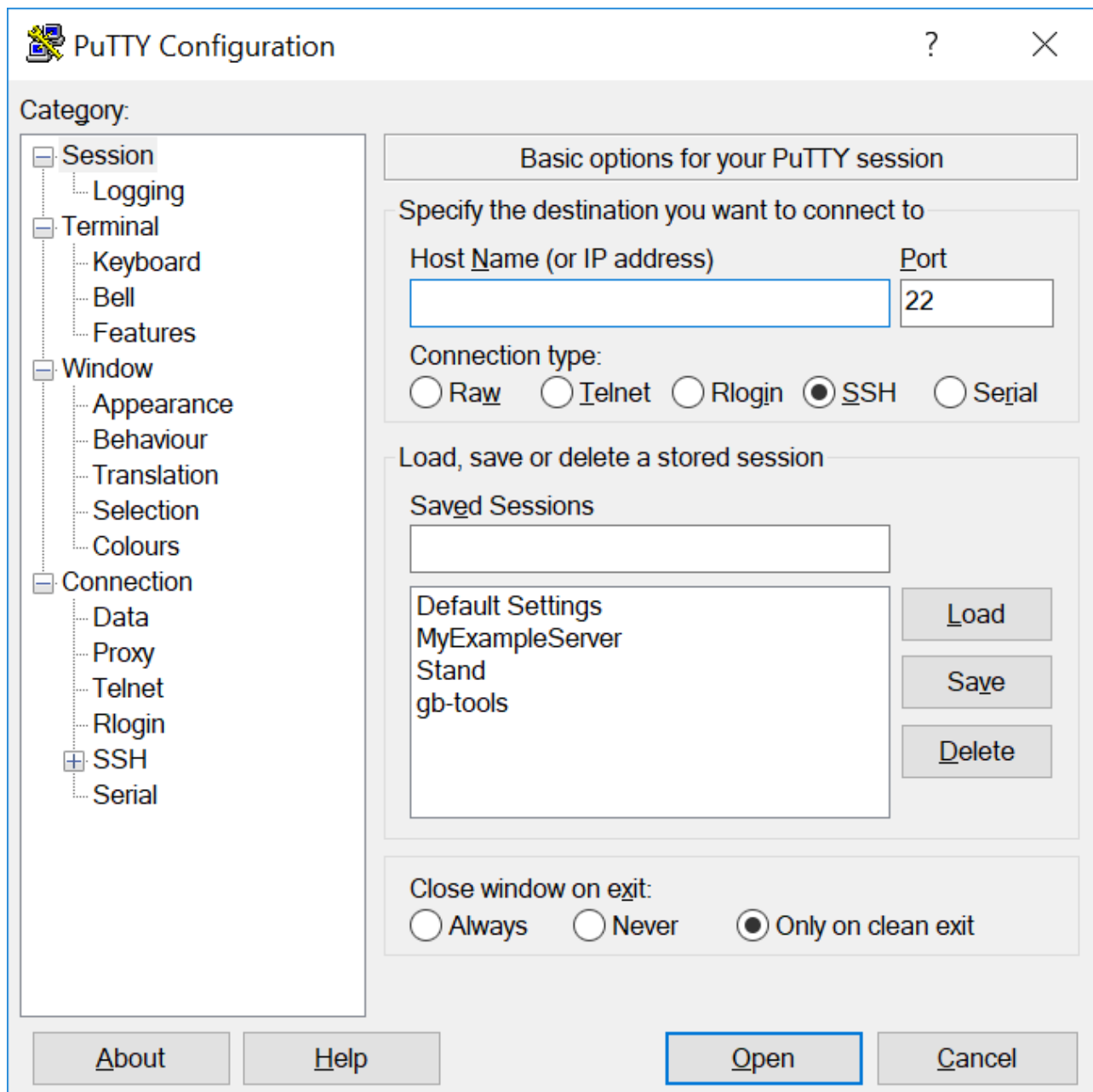
Жмем «Открыть», а затем — **Session**:



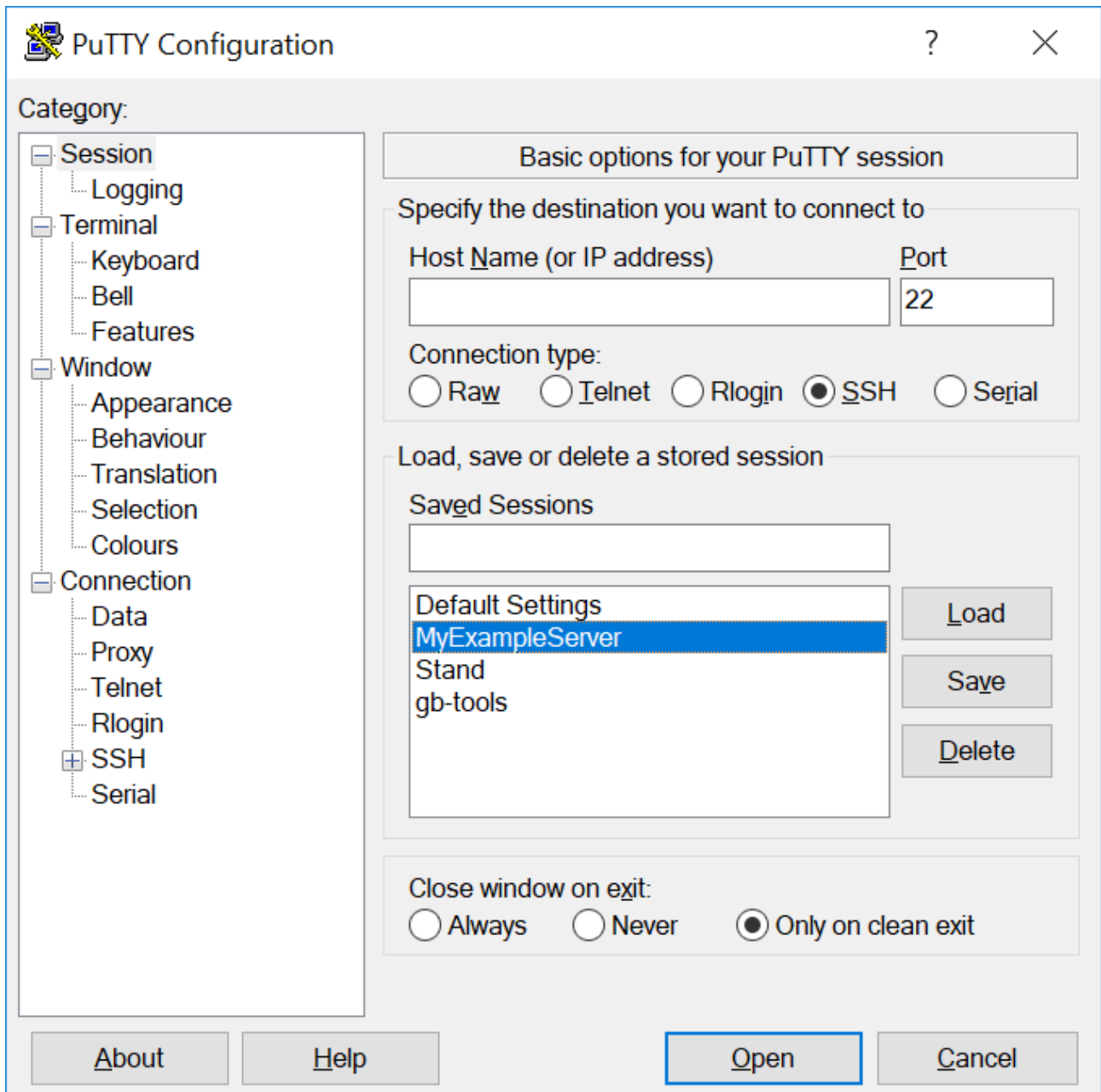
Теперь жмем **Save**:



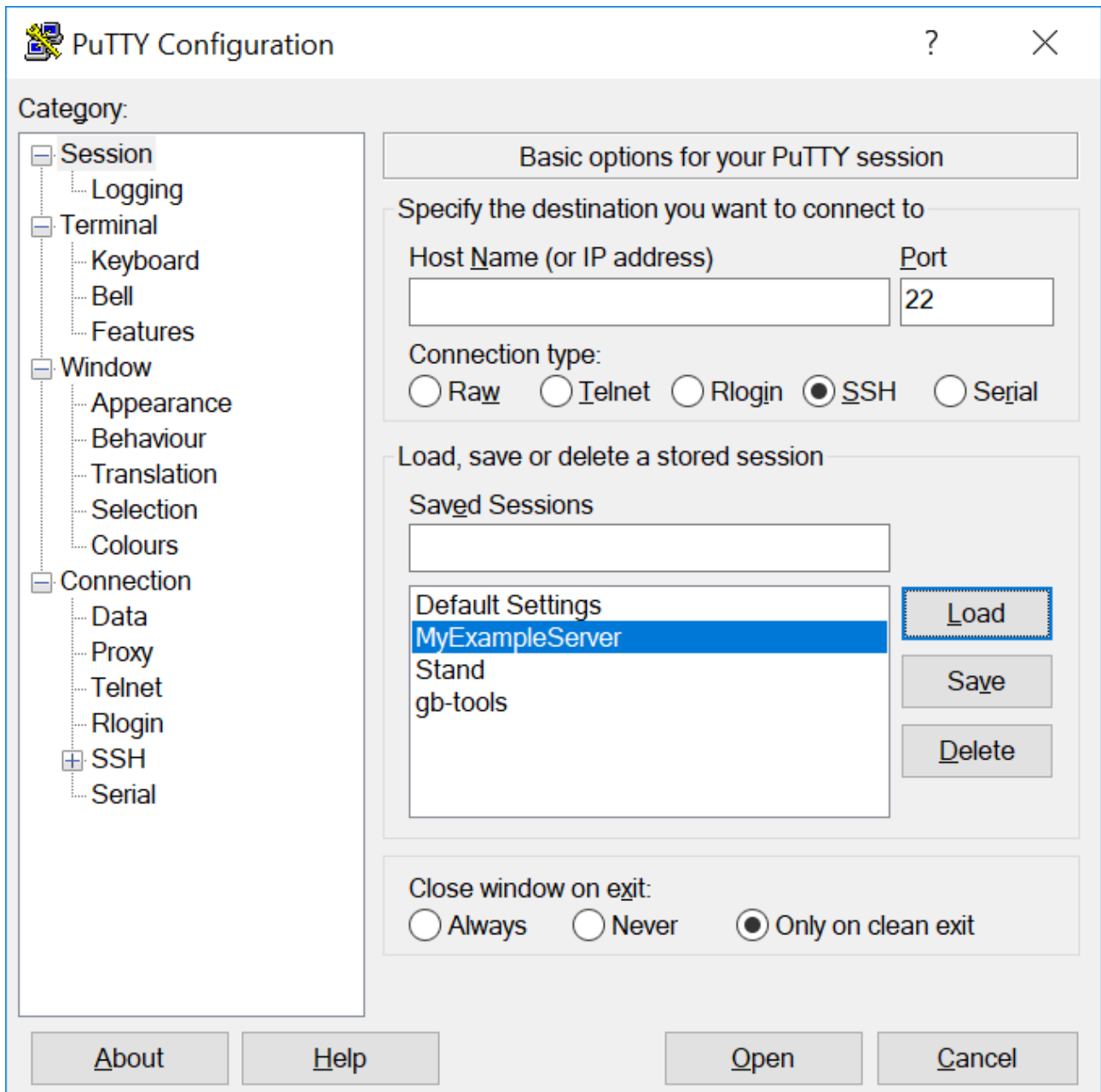
Теперь вы можете закрыть и заново открыть **PuTTY**:

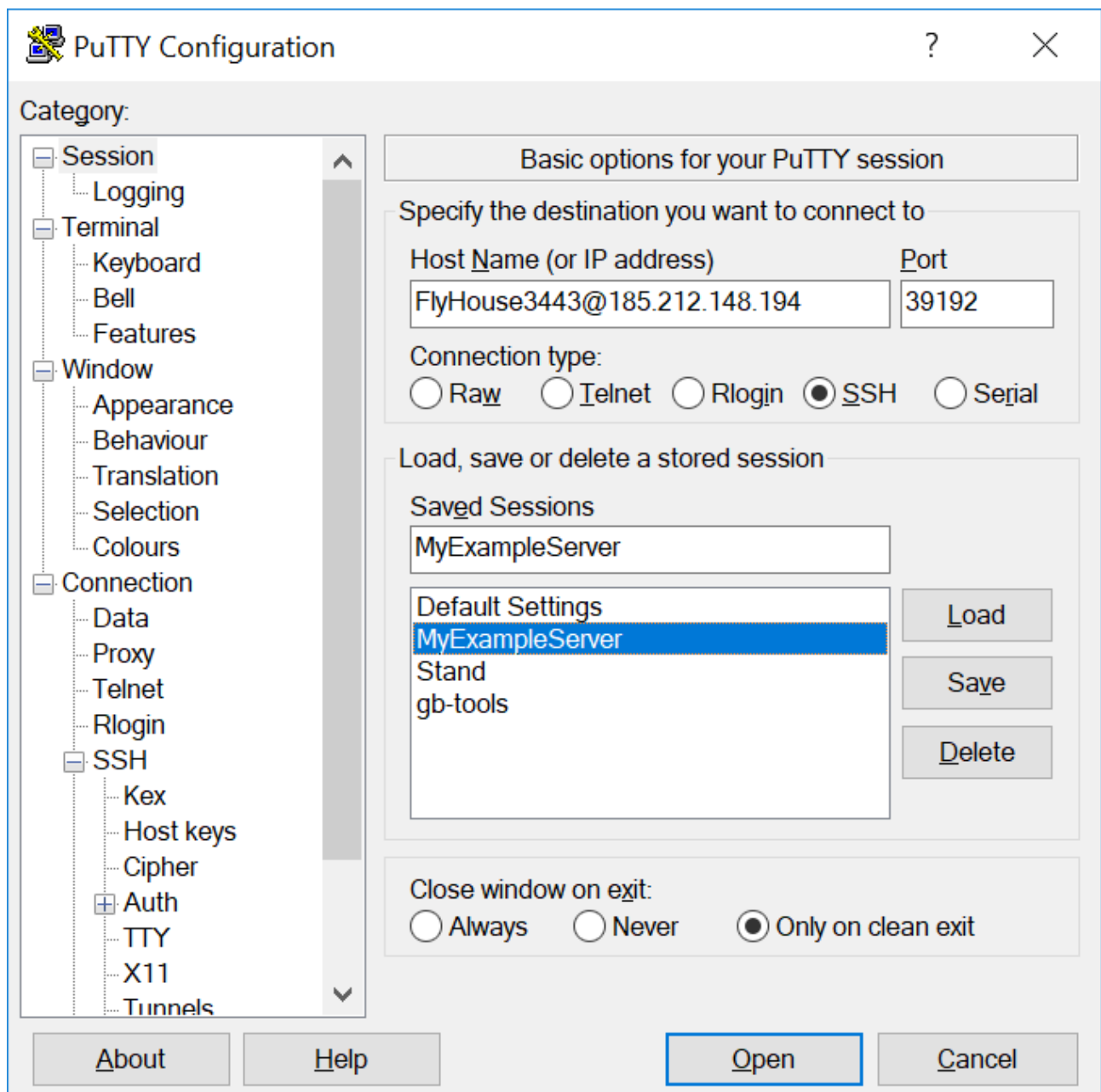


Выбираем наш сервер:

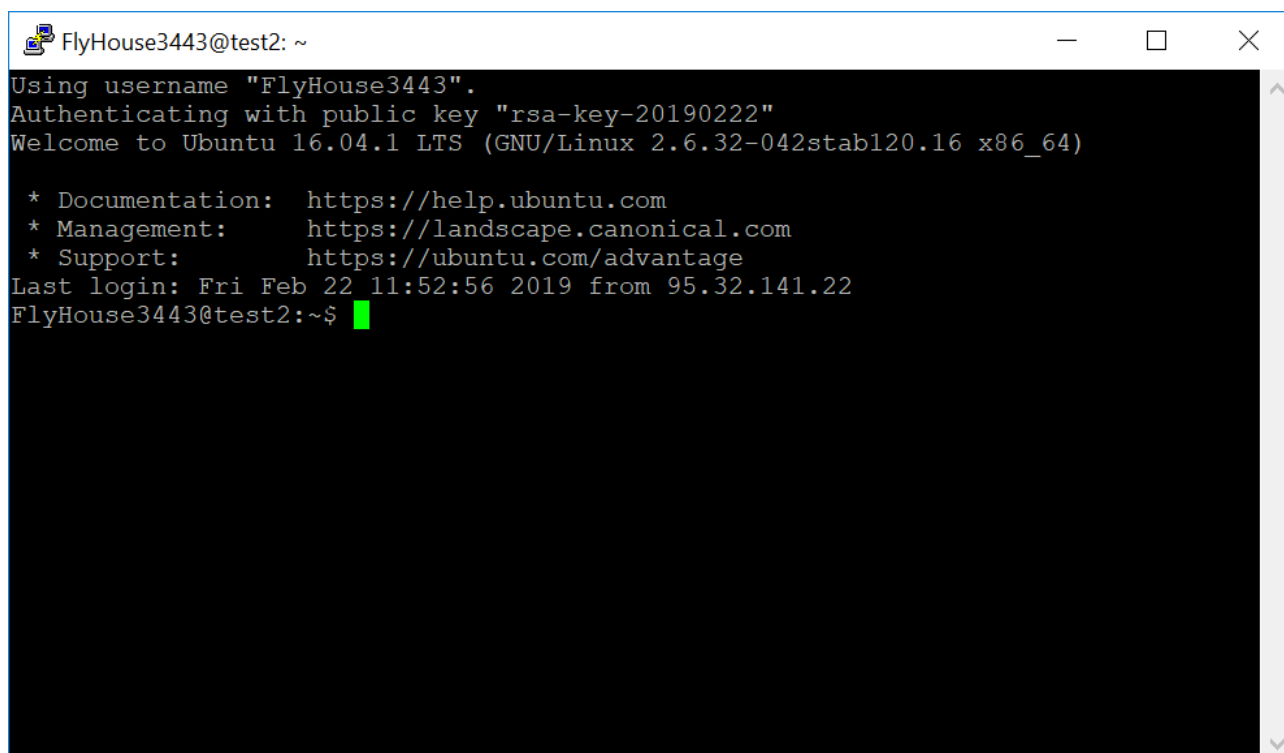


И жмем Load:





Жмем **Open**:

A terminal window titled 'FlyHouse3443@test2: ~' with standard window controls. The terminal output shows an SSH login process: 'Using username "FlyHouse3443".', 'Authenticating with public key "rsa-key-20190222"', and 'Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)'. It then lists links for documentation, management, and support, followed by the last login time: 'Last login: Fri Feb 22 11:52:56 2019 from 95.32.141.22'. The prompt 'FlyHouse3443@test2:~\$' is followed by a green cursor.

```
FlyHouse3443@test2: ~
Using username "FlyHouse3443".
Authenticating with public key "rsa-key-20190222"
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 11:52:56 2019 from 95.32.141.22
FlyHouse3443@test2:~$
```

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):

```
Ubuntu 16.04.5 LTS user-VirtualBox tty1
user-VirtualBox login: user
Password:
Last login: Thu Feb 21 20:56:48 MSK 2019 on tty2
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-29-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

94 packages can be updated.
0 updates are security updates.

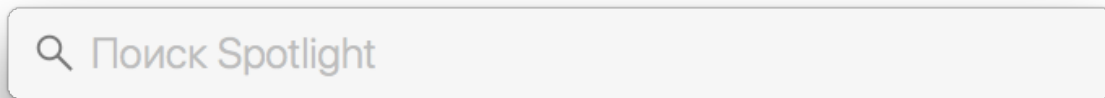
*** System restart required ***
user@user-VirtualBox:~$ ssh-keygen -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:lgVSYlM9xzSE723vJ5j55tSU1UHLQSH/ByTvsSCdXE0 user@user-VirtualBox
The key's randomart image is:
+---[RSA 2048]---+
|      =o+. oo.=OE|
|    . + .oo+B+ *|
|      o== =+o|
|    o .o+ =+|
|    S  .+. *|
|      .  +o|
|      +. o|
|     +.o..|
|    +o.o|
```

Либо откройте терминал с помощью Ctrl-Alt-T:

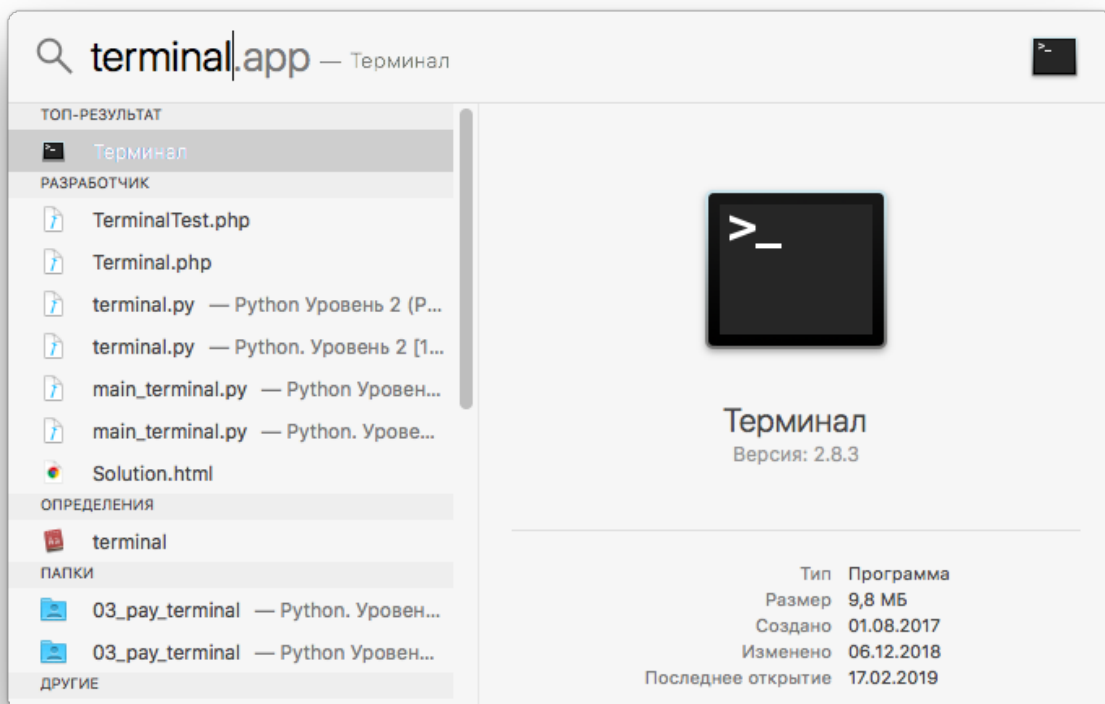
```
user@user-VirtualBox: ~/.ssh
user@user-VirtualBox:~$ cd .ssh
user@user-VirtualBox:~/.ssh$ ls -la
total 16
drwx----- 2 user user 4096 фев 22 22:10 .
drwxr-xr-x 16 user user 4096 фев 22 22:11 ..
-rw----- 1 user user 1679 фев 22 22:10 id_rsa
-rw-r--r-- 1 user user 402 фев 22 22:10 id_rsa.pub
user@user-VirtualBox:~/.ssh$
```


В Mac:

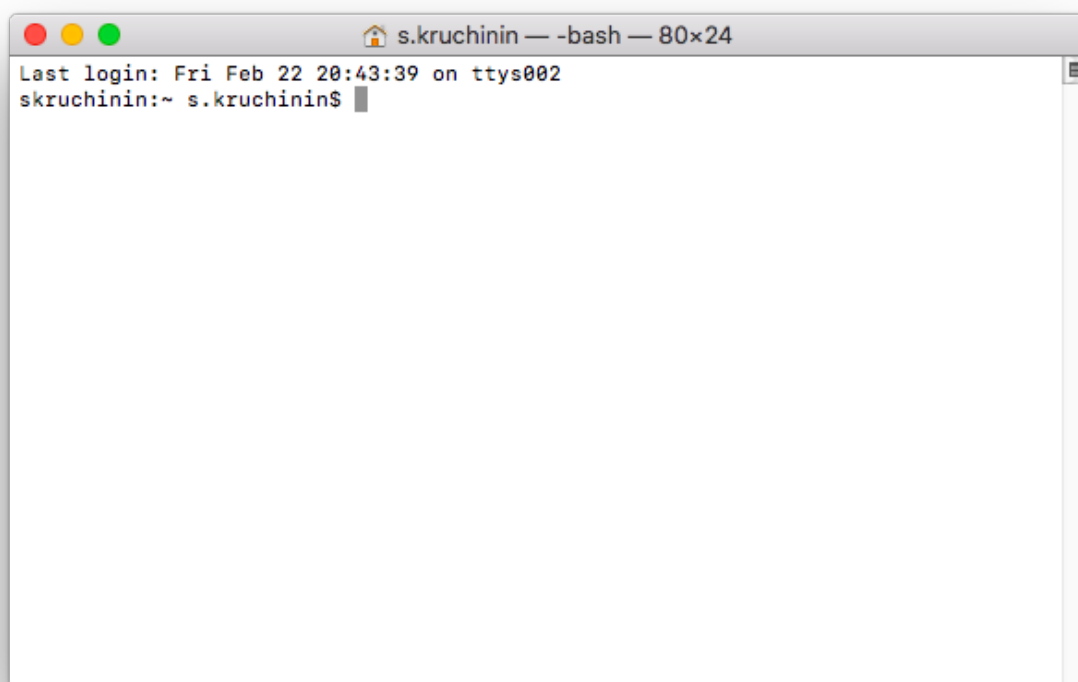
1. CMD + пробел:



2. Набираем terminal



3. Кликаем на «Терминал» и он запускается:



Далее работаем аналогично изложенным выше шагам.

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

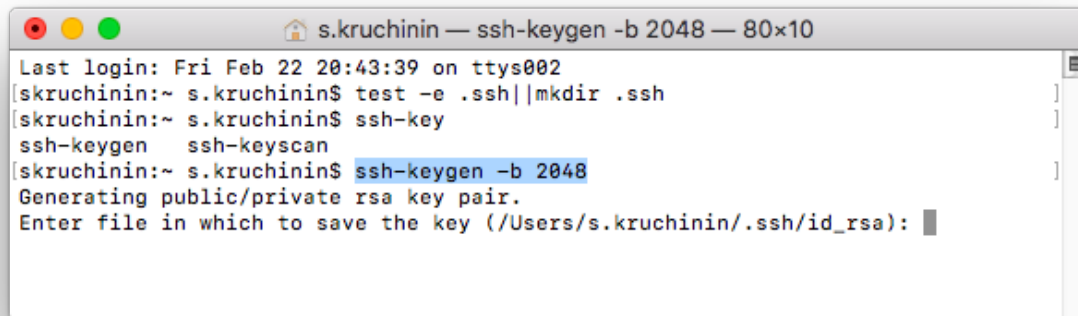
```
$ test -e .ssh||mkdir .ssh
```

Переходим в **.ssh**:

```
$ cd .ssh
```

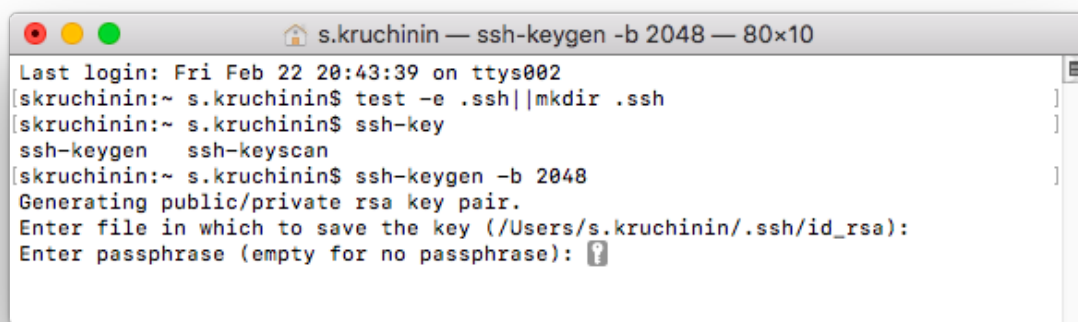
Запускаем **ssh-keygen**:

```
$ ssh-keygen -b 2048
```



```
s.kruchinin — ssh-keygen -b 2048 — 80x10
Last login: Fri Feb 22 20:43:39 on ttys002
[skruchinin:~ s.kruchinin$ test -e .ssh||mkdir .ssh
[skruchinin:~ s.kruchinin$ ssh-key
ssh-keygen  ssh-keyscan
[skruchinin:~ s.kruchinin$ ssh-keygen -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/s.kruchinin/.ssh/id_rsa):
```

Соглашаемся:



```
s.kruchinin — ssh-keygen -b 2048 — 80x10
Last login: Fri Feb 22 20:43:39 on ttys002
[skruchinin:~ s.kruchinin$ test -e .ssh||mkdir .ssh
[skruchinin:~ s.kruchinin$ ssh-key
ssh-keygen  ssh-keyscan
[skruchinin:~ s.kruchinin$ ssh-keygen -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/s.kruchinin/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
```

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:

```
.ssh — -bash — 80x35
Last login: Fri Feb 22 20:43:39 on ttys002
[skruchinin:~ s.kruchinin$ test -e .ssh||mkdir .ssh
[skruchinin:~ s.kruchinin$ ssh-keygen
ssh-keygen ssh-keyscan
[skruchinin:~ s.kruchinin$ ssh-keygen -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/s.kruchinin/.ssh/id_rsa):
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /Users/s.kruchinin/.ssh/id_rsa.
Your public key has been saved in /Users/s.kruchinin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:C2fbq002KdUV3idc59Exl6fPu8XoZ0sJrtXpkSXmm+Q s.kruchinin@skruchinin.local
The key's randomart image is:
+----[RSA 2048]-----+
|          o=|
|         . oB|
|        . + ++|
|       oo= o|
|      . S. . ==.|
|     +++ . + Xo|
|    +oo. o E +|
|   . + + * =.|
|  ..oo...*..|
+-----[SHA256]-----+
[skruchinin:~ s.kruchinin$ cd .ssh
[skruchinin:.ssh s.kruchinin$ ls -la
total 16
drwxr-xr-x  4 s.kruchinin  staff   128 22 фев 21:04 .
drwxr-xr-x+ 35 s.kruchinin  staff  1120 22 фев 21:00 ..
-rw-----  1 s.kruchinin  staff  1843 22 фев 21:04 id_rsa
-rw-r--r--  1 s.kruchinin  staff   410 22 фев 21:04 id_rsa.pub
skruchinin:.ssh s.kruchinin$
```

Проверяем с 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
```

```
.ssh — -bash — 80x11
[skruchinin:.ssh s.kruchinin$ scp -P 39192 id_rsa.pub FlyHouse3443@185.212.148.194:/home/FlyHouse3443/id_rsa_pub_mac
The authenticity of host '[185.212.148.194]:39192 ([185.212.148.194]:39192)' can't be established.
ECDSA key fingerprint is SHA256:uNMWmfXk6GrJr5huEULqK25rytsiAZ52CqsYYF7VN5o.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[185.212.148.194]:39192' (ECDSA) to the list of known hosts.
[FlyHouse3443@185.212.148.194's password:
id_rsa.pub 100% 410 13.6KB/s 00:00
skruchinin:.ssh s.kruchinin$
```

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
```

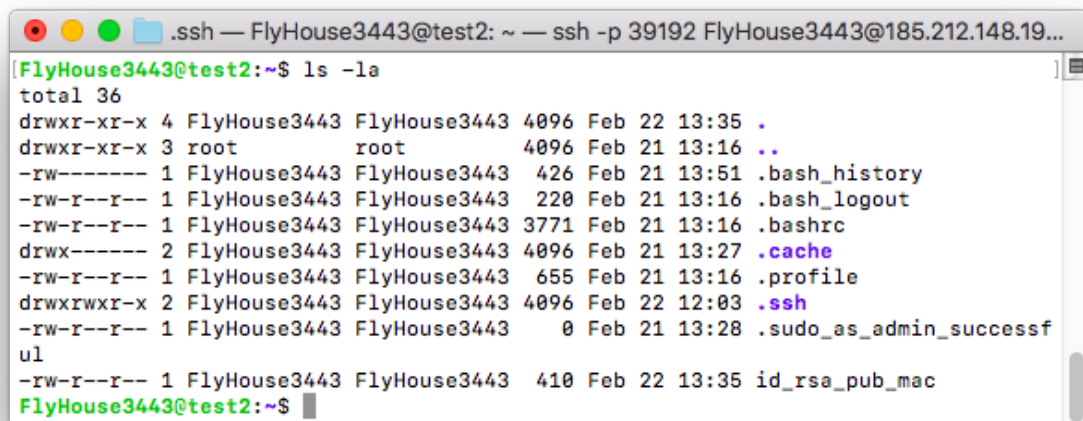
Обратите внимание, что ключ задания пароля у **scp** и **ssh** отличается регистром.

Подключились:

```
.ssh — FlyHouse3443@test2: ~ — ssh -p 39192 FlyHouse3443@185.212.148.19...
[FlyHouse3443@185.212.148.194's password:
id_rsa.pub 100% 410 13.6KB/s 00:00
[skruchinin:.ssh s.kruchinin$ ssh -p 39192 FlyHouse3443@185.212.148.194
[FlyHouse3443@185.212.148.194's password:
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 12:29:38 2019 from 95.32.141.22
FlyHouse3443@test2:~$
```

Файл на месте:



```
.ssh — FlyHouse3443@test2: ~ — ssh -p 39192 FlyHouse3443@185.212.148.19...
[FlyHouse3443@test2:~$ ls -la
total 36
drwxr-xr-x 4 FlyHouse3443 FlyHouse3443 4096 Feb 22 13:35 .
drwxr-xr-x 3 root          root          4096 Feb 21 13:16 ..
-rw----- 1 FlyHouse3443 FlyHouse3443  426 Feb 21 13:51 .bash_history
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443  220 Feb 21 13:16 .bash_logout
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443 3771 Feb 21 13:16 .bashrc
drwx----- 2 FlyHouse3443 FlyHouse3443 4096 Feb 21 13:27 .cache
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443  655 Feb 21 13:16 .profile
drwxrwxr-x 2 FlyHouse3443 FlyHouse3443 4096 Feb 22 12:03 .ssh
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443    0 Feb 21 13:28 .sudo_as_admin_successful
-rw-r--r-- 1 FlyHouse3443 FlyHouse3443  410 Feb 22 13:35 id_rsa_pub_mac
FlyHouse3443@test2:~$
```

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:

```
.ssh — FlyHouse3443@test2: ~ — ssh -p 39192 FlyHouse3443@185.212.148.19...

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 13:39:32 2019 from 95.32.141.22
[FlyHouse3443@test2:~$ cat id_rsa_pub_mac >>.ssh/authorized_keys
[FlyHouse3443@test2:~$ exit
logout
Connection to 185.212.148.194 closed.
[skruchinin:.ssh s.kruchinin$ ssh -p 39192 FlyHouse3443@185.212.148.194
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 14:02:04 2019 from 95.32.141.22
FlyHouse3443@test2:~$
```

Отключаем возможность логиниться руту и доступ по паролям

Переподключаемся — разумеется, без ввода пароля:

```
FlyHouse3443@test2: ~
Using username "FlyHouse3443".
Authenticating with public key "rsa-key-20190222"
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 11:52:56 2019 from 95.32.141.22
FlyHouse3443@test2:~$
Using username "FlyHouse3443".
Authenticating with public key "rsa-key-20190222"
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 14:03:16 2019 from 95.32.141.22
FlyHouse3443@test2:~$
```

Правим `/etc/ssh/sshd_config`:

```
$ sudo nano /etc/ssh/sshd_config
```

```
FlyHouse3443@test2: ~
Using username "FlyHouse3443".
Authenticating with public key "rsa-key-20190222"
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 11:52:56 2019 from 95.32.141.22
FlyHouse3443@test2:~$
Using username "FlyHouse3443".
Authenticating with public key "rsa-key-20190222"
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 14:03:16 2019 from 95.32.141.22
FlyHouse3443@test2:~$ sudo nano /etc/ssh/sshd_config
[sudo] password for FlyHouse3443: █
```

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

```
PermitRootLogin yes
```

на

```
PermitRootLogin no
```



```
FlyHouse3443@test2: ~
GNU nano 2.5.3      File: /etc/ssh/sshd_config      Modified
# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
#PermitRootLogin yes
PermitRootLogin no
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

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Далее меняем:

```
PasswordAuthentication yes
```

на

```
PasswordAuthentication no
```

```
FlyHouse3443@test2: ~
GNU nano 2.5.3      File: /etc/ssh/sshd_config      Modified
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
PasswordAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

Сохраняем в **nano**:

- Ctrl-O
- Enter
- Ctrl-X

```
FlyHouse3443@test2: ~
GNU nano 2.5.3      File: /etc/ssh/sshd_config      Modified
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
PasswordAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
File Name to Write: /etc/ssh/sshd_config
^G Get Help  M-D DOS Format  M-A Append      M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend     ^T To Files
```

Созраняем в **vi**:

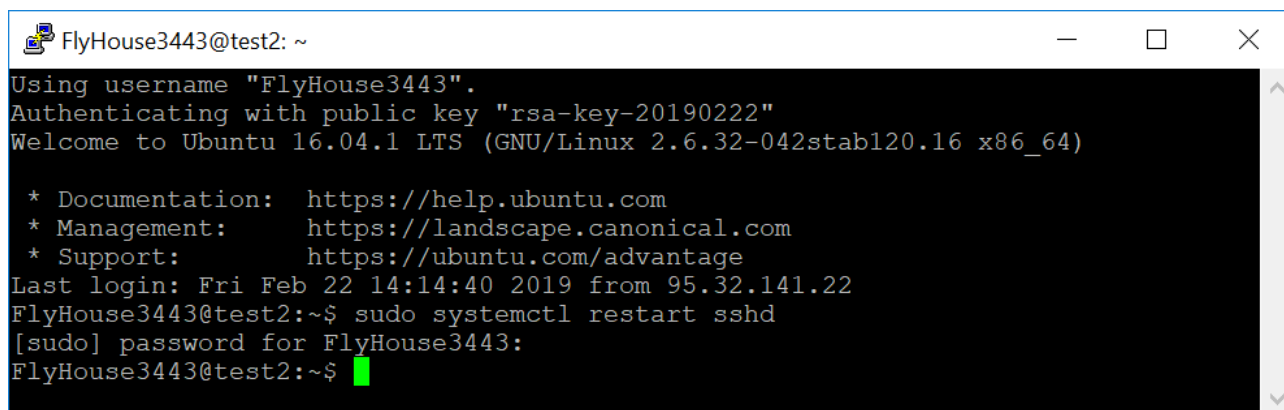
:wq!

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
```

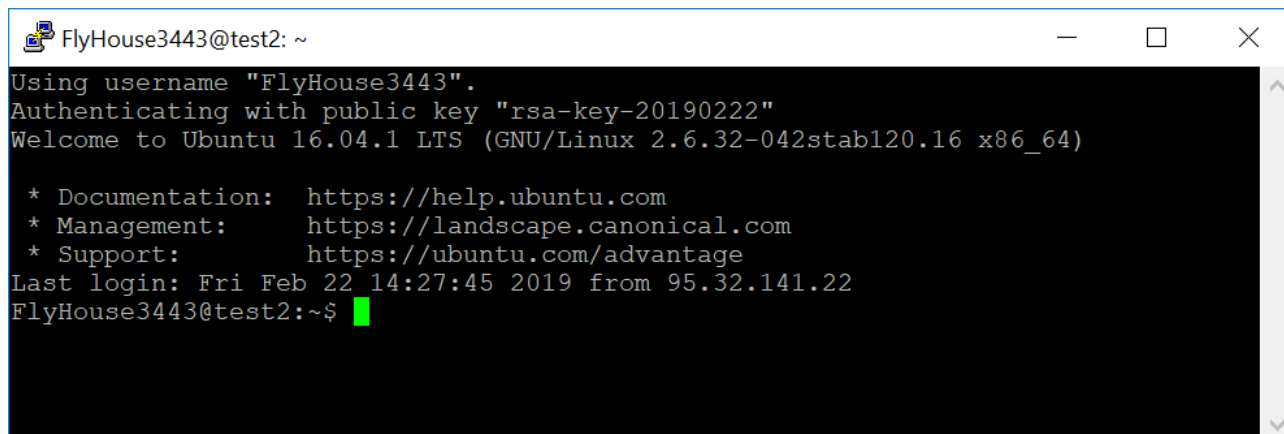


```
FlyHouse3443@test2: ~
Using username "FlyHouse3443".
Authenticating with public key "rsa-key-20190222"
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 14:14:40 2019 from 95.32.141.22
FlyHouse3443@test2:~$ sudo systemctl restart sshd
[sudo] password for FlyHouse3443:
FlyHouse3443@test2:~$
```

Вводим пароль.

Теперь пробуем подключиться по ключу:



```
FlyHouse3443@test2: ~
Using username "FlyHouse3443".
Authenticating with public key "rsa-key-20190222"
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 14:27:45 2019 from 95.32.141.22
FlyHouse3443@test2:~$
```

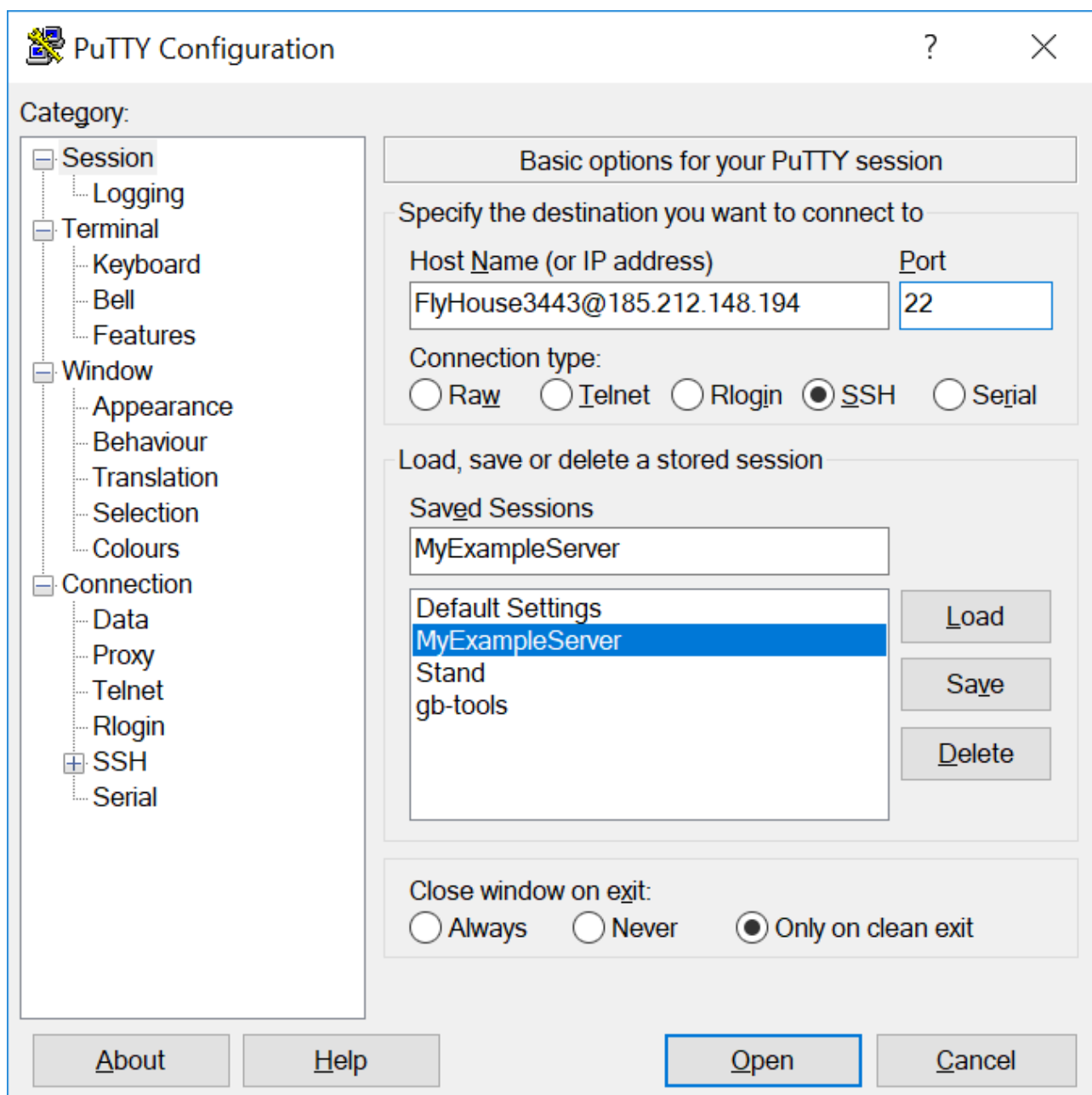
Если подключились — значит, все хорошо.

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.

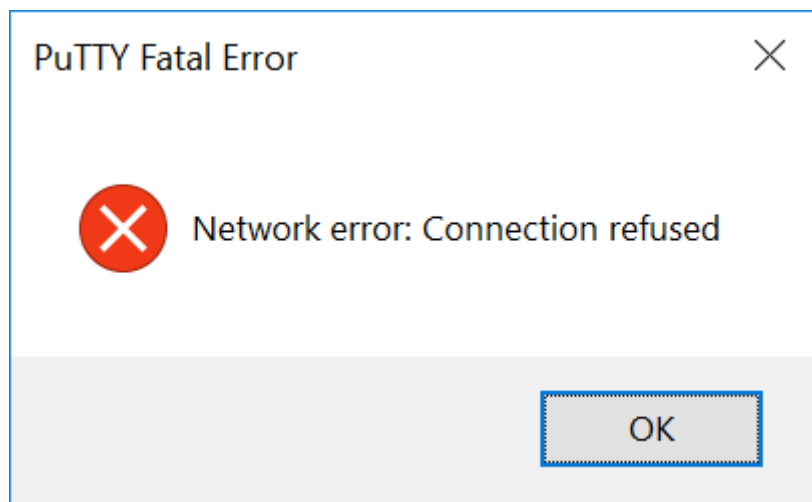
Теперь контрольный выстрел.

По **22/TCP**:

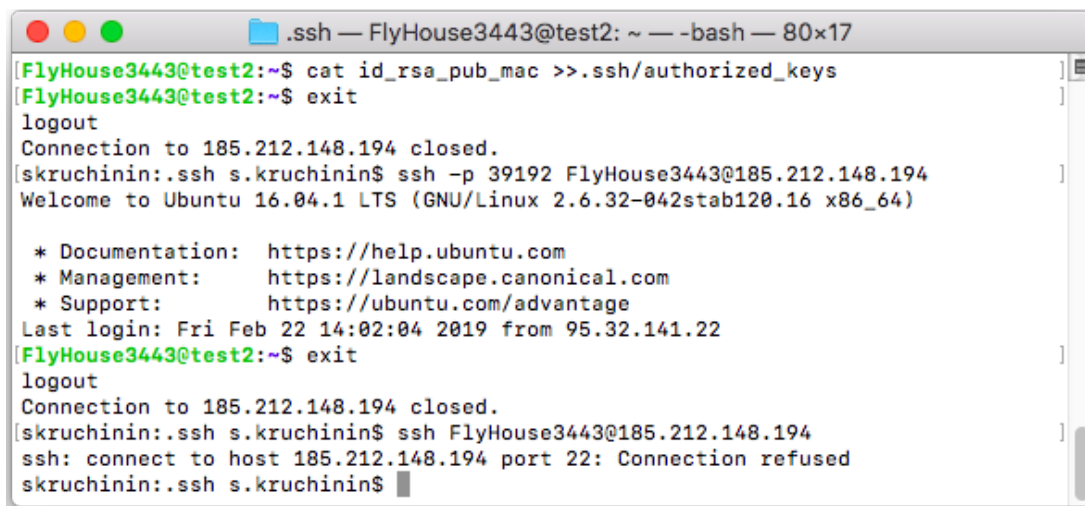
1. Жмем **Load** и меняем порт на 22:



Результат:



Аналогично на Mac и в Linux:



```
.ssh — FlyHouse3443@test2: ~ — -bash — 80x17
FlyHouse3443@test2:~$ cat id_rsa_pub_mac >>.ssh/authorized_keys
FlyHouse3443@test2:~$ exit
logout
Connection to 185.212.148.194 closed.
[skruchinin:~.ssh s.kruchinin$ ssh -p 39192 FlyHouse3443@185.212.148.194
Welcome to Ubuntu 16.04.1 LTS (GNU/Linux 2.6.32-042stab120.16 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri Feb 22 14:02:04 2019 from 95.32.141.22
FlyHouse3443@test2:~$ exit
logout
Connection to 185.212.148.194 closed.
[skruchinin:~.ssh s.kruchinin$ ssh FlyHouse3443@185.212.148.194
ssh: connect to host 185.212.148.194 port 22: Connection refused
skruchinin:~.ssh s.kruchinin$
```

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.

Асимметричные 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>. 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
$ 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 с 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.

5. * 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.