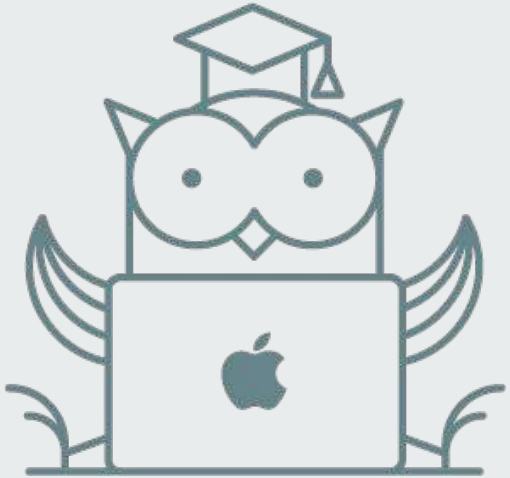
Курс «Администратор Linux».

SSH Tips and Tricks

Занятие № 1

Дмитрий Молчанов
Григорий Ожегов

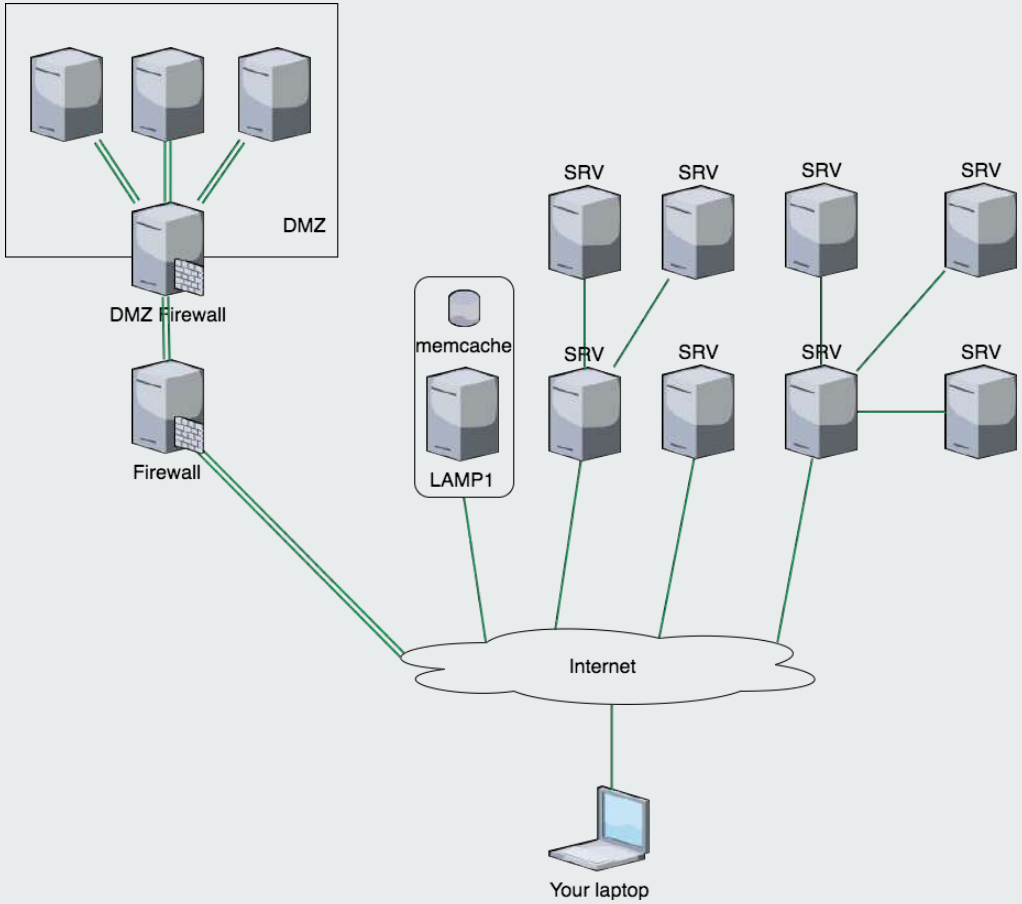


Как мы используем ssh (обычно)

- Доступ к серверу
- Scp с/на сервер или sftp
- Ключи для аутентификации

Для чего мы используем ssh

- Доступ к большому количеству серверов
- Доступ к серверам через jumpbox
- Доступ к серверу в DMZ через 2 firewall’а
- Доступ к сервису (memcache/redis/whatever) слушающему только loopback на сервере.


DMZ
DMZ Firewall
Firewall
memcache
LAMP1
SRV
SRV
SRV
SRV
SRV
SRV
SRV
SRV
Internet
Your laptop

Проблемы с которыми мы сталкиваемся

- Вводить пароль сотню раз на дню - лень. Помнить много разных паролей - тем более.
- Аутентификация на ключах позволяет нам не вводить пароль каждый раз при установке сессии.
- Когда надо ходить через какой-то сервер - часто ключ копируется на промежуточный сервер.
- Ключ может “нечаянно” утечь
  - Украли
  - Закоммитил в гитхаб
  - Еще что-то
- Если ключ утек, то беда - считай все двери открыты.

Какие возможности у ssh есть еще

- ssh-agent (для аутентификации по ключам)
  - ForwardAgent
- PortForward
  - LocalPortForward
  - RemotePortForward
- ProxyCommand
- STDIN/OUT Forwarding
- ControlMaster
- .ssh/config
- From-stanza в authorized_keys

Как работает аутентификация на ключах

- Ssh-ключ состоит из 2х частей.
  - Публичная - *keyname*.pub файл. Данные которые записываются в authorized_keys
  - Приватная - *keyname*. Ответная часть ключа, которая подходит только к одной pub-части. Должна храниться в одном месте, желательно зашифрованной (защищенной паролем)
- Устанавливая соединение мы передаем следующую информацию:
  - user, host, port

    `ssh user@host [-p port]` **или** `ssh -l user -p port host`

    Если не указан user - используется локальное имя пользователя

    Если не указан port - используется 22

Как работает аутентификация на ключах

- Соединившись и передав имя пользователя клиент “передает сообщение” подписанное приватным ключом.
- После соединения сервер знает под каким пользователем мы пытаемся соединиться и смотрит в файл с разрешенными для пользователя ключами ($HOME/.ssh/authorized_keys). Если он находит там ключ, который соответствует тому, которым подписано сообщение - процесс аутентификации считается пройденным.
- Пароля вводить для входа не требуется, потребуется ввести пароль только для расшифровки ключа.

Доступ к большому количеству серверов

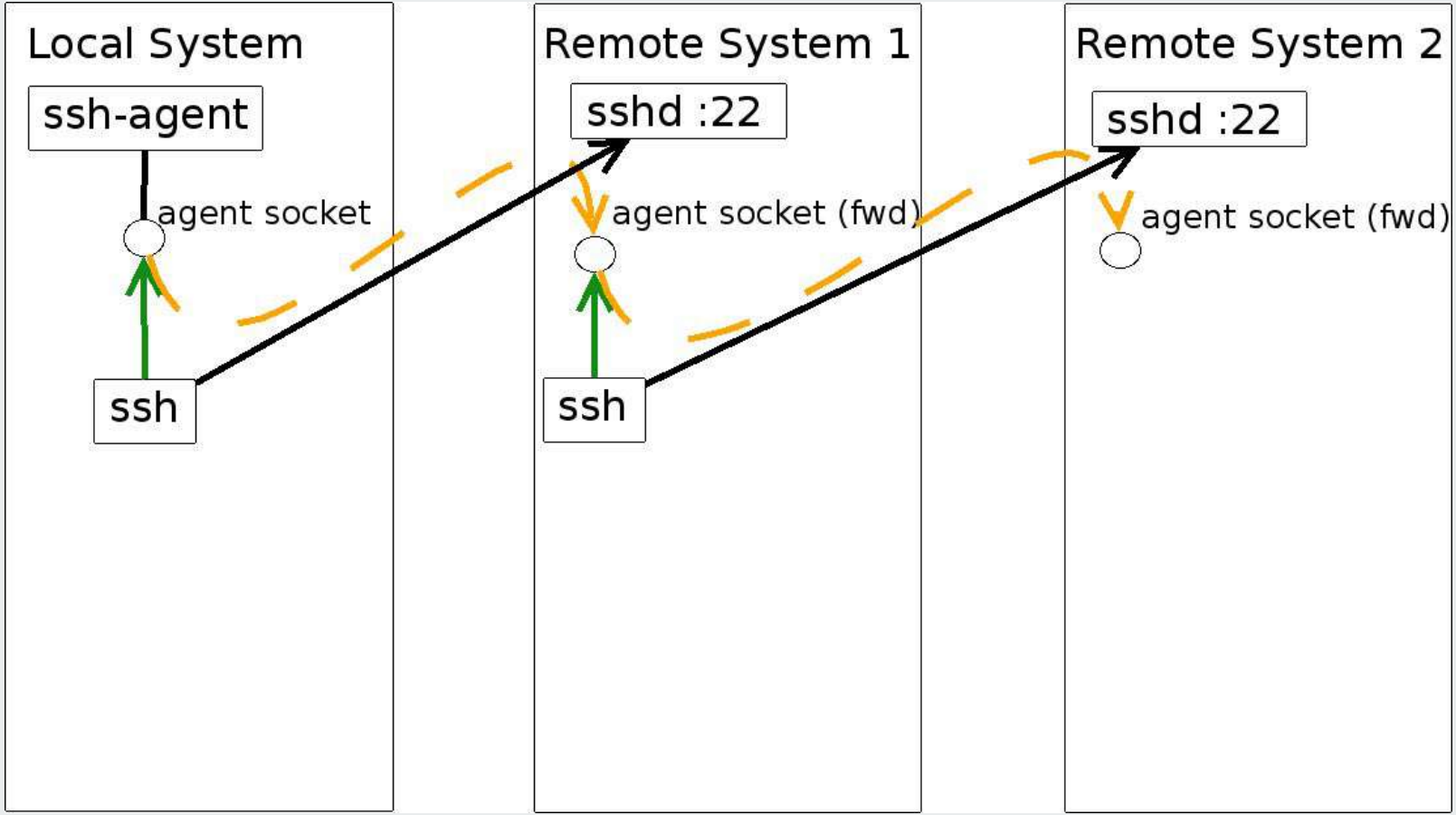
Нам помогут ssh-agent, ForwardAgent и пароль(шифрование) ключа.

Ssh-agent - программа, которая зачитывает на старте или по команде ключи, расшифровывает их держит их в памяти и открывает сокет для ssh-клиента, через который тот может использовать расшифрованые ключи для аутентификации.

У клиента ssh есть опция, которая позволяет “брать с собой” этот сокет в удаленную систему, что позволяет пользоваться им там.

Это дает возможность использовать один ключ везде и не копировать его с сервера на сервер.

Ssh-agent forwarding схема



ForwardAgent использование

Для того, чтобы ипользовать эту возможность надо:

- запустить ssh-agent

  ```
  eval `ssh-agent -s`
  ```
- Указать опцию -A при установке соединения, либо добавить запись в .ssh/config (Об этом позже).

```
ssh -A -p 20022 -l root 127.0.0.1
```

или

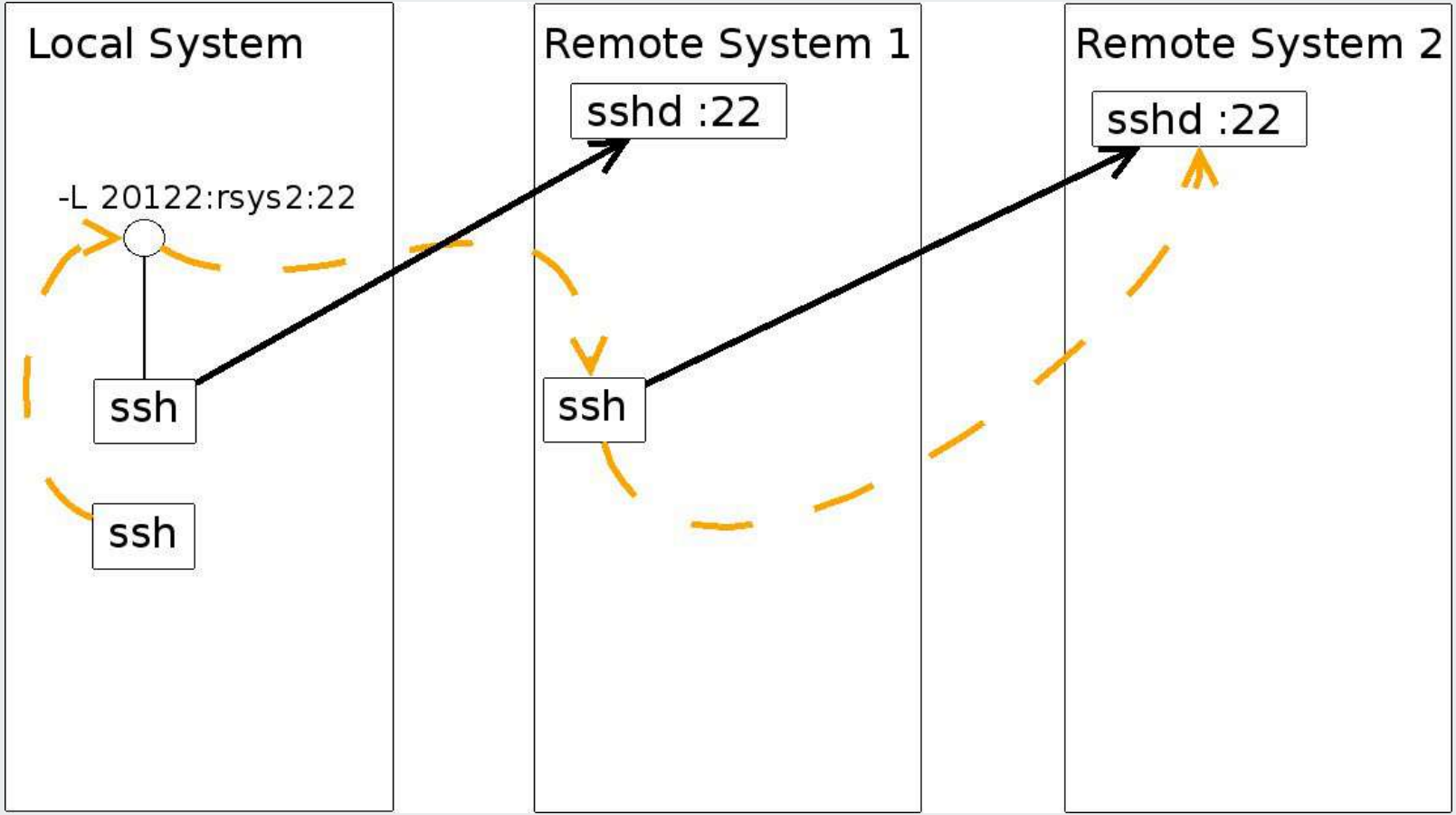
```
ssh -oForwardAgent=yes -p 20022 -l root 127.0.0.1
```

Эта команда говорит, что надо соединиться с хостом *127.0.0.1* на порт *20022* под пользователем *root* и использовать ForwardAgent

Local Port Forward

Проброс локального порта - функция ssh-клиента, когда он открывает локальный слушающий сокет, и соединения к этому сокету открываются от имени ssh-сервера на второй стороне ssh-соединения. Это позволяет получить доступ к ресурсам к которым у локальной стороны соединения доступа нет, но есть у удаленной стороны. Например локальный (слушающий только на 127.0.0.1) memcached или mysql или сервер в удаленной сети, который недоступен снаружи.

Local Port Forward



Local Port Forward

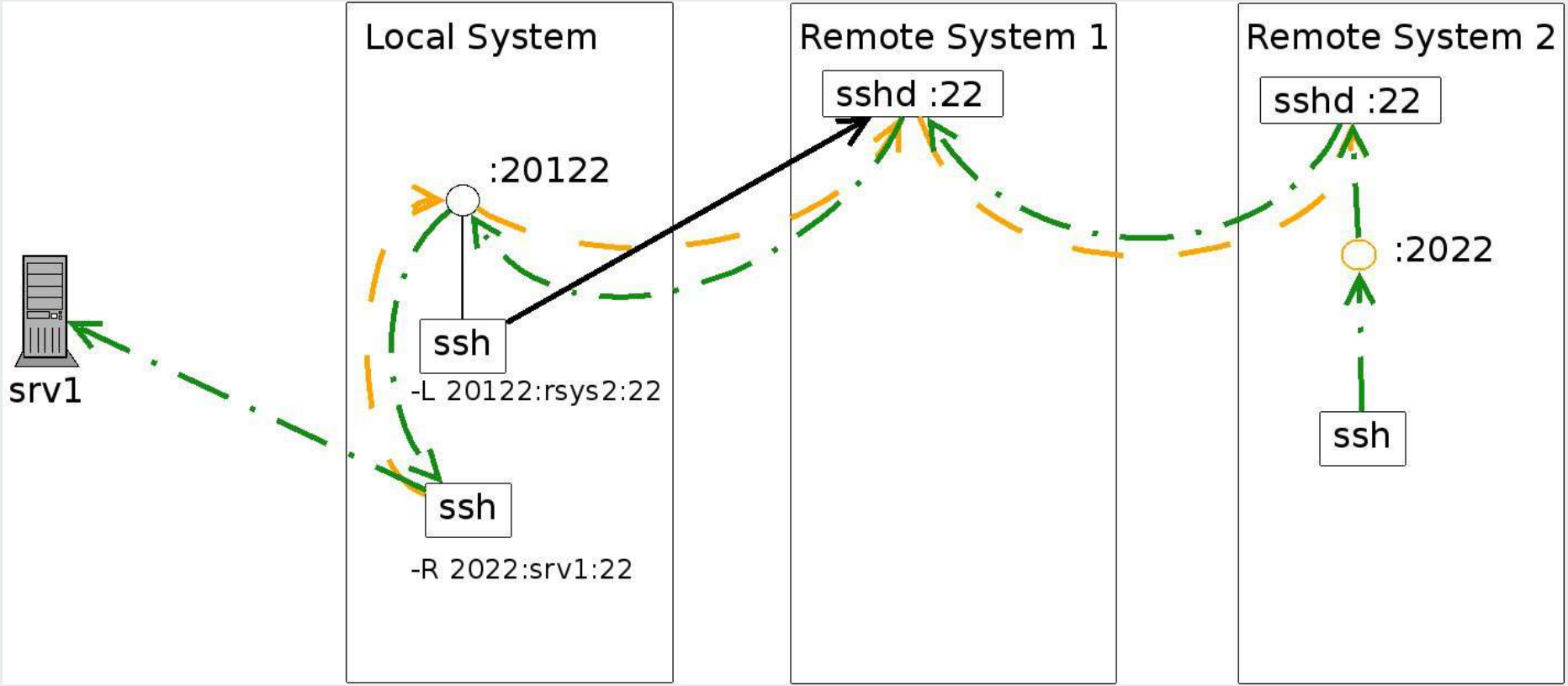
На предыдущем слайде схематически изображено:

1. 3 системы - Local, Remote 1, Remote 2
2. на Local запущен ssh -L 20122:remote_system_2:22 remote_system_1.
   То есть установлено соединение с remote_system_1 с пробросом локального порта 20122 в адрес remote_system_2:22.
3. на Local запущен ssh обращающийся к локальному сокету на порту 20122 и соединяющийся через Remote System 1 с Remote System 2. Соединение с Remote System 2 устанавливается от имени sshd обслуживающего соединение установленное в предыдущем пункте.

Remote Port Forward

Проброс удаленных портов - возможность ssh-клиента дать команду удаленному ssh-серверу, обслуживающему соединение, открыть слушающий сокет и все соединения приходящие на этот сокет будут открываться от имени этого клиента.

Remote Port Forward



Remote Port Forward

На схеме, в дополнение к схеме с Local Port Forward мы видим:

1. srv1 - сервер внешний по отношению к Local System
2. ssh -p 20122 -R 2022:srv1:22 localhost:
   ssh-клиент соединяющийся с портом 20122 (проброшен на схеме LocalPortForward) говорит удаленному sshd (Remote System 2) открыть порт 2022 и все соединения, которые будут туда устанавливаться, будут устанавливаться от имени ssh клиента (с -R) с srv1:22
3. ssh-клиент на Remote System 2 устанавливает соединение с 127.0.0.1:2022 и соединяется с srv1:22 от имени ssh-клиета из п.2

Proxy Command

Иногда, для промежуточных соединений нам не достаточно использовать ssh, а надо, к примеру, использовать nc (netcat), чтобы установить соединение через http-сервер с использованием метода connect.

```
ProxyCommand /usr/bin/nc -X connect -x proxy_host:proxy_port %h %p
```

Такой ProxyCommand проинструктирует ssh сначала запустить nc, чтобы тот соединился с proxy_host:proxy_port и выполнил метод conncet на %h %p (реальный хост и порт к которым ssh надо соединиться). В этом случае ssh-клиент будет работать с потоками ввода/вывода nc, а не с сетевым сокетом.

STDIN/STDOUT forward

Опция -W host:port. Может использоваться совместно с ProxyCommand, вместо nc, когда для того, чтобы достучаться до удаленного хоста надо установить соединение по ssh.

В этом случае устанвливается ssh-соединение, от имени удаленного sshd .обслуживающего это соединение устанавливается tcp-соединение с host:port и in/out этого соединения передаются вызывающему ssh-клиенту.

STDIN/STDOUT forward

```
$ ssh -W ip.molchanov.pp.ru:80 vps
GET / HTTP/1.1
Host: ip.molchanov.pp.ru

HTTP/1.1 200 OK
Server: nginx/1.8.0
Date: Sun, 29 Oct 2017 20:24:31 GMT
Content-Type: application/octet-stream
Content-Length: 26
Connection: keep-alive

Your address is 5.9.159.69
^C
$ telnet ip.molchanov.pp.ru 80
Trying 5.9.159.69...
Connected to molchanov.pp.ru.
Escape character is '^]'.
GET / HTTP/1.1
Host: ip.molchanov.pp.ru

HTTP/1.1 200 OK
Server: nginx/1.8.0
Date: Sun, 29 Oct 2017 20:25:14 GMT
Content-Type: application/octet-stream
Content-Length: 30
Connection: keep-alive

Your address is 46.188.121.183
^CConnection closed by foreign host.
```

Control Master

Одной из самых тяжелых стадий установки ssh-соединения является изначальный обмен ключами и последующий обмен ключами для симметричного шифрования сессии. Когда у вас возникает необходимость поддерживать много сессий с одним и тем же сервером, особенно удаленным, то опция мультиплексирования ssh-соединения может очень сильно облегчить жизнь, потому что в данном случае в рамках одного ssh-соединения будет устанавливаться несколько сессии и, вместо полноценного соединения, будет добавляться еще одна сессия, минуя все тяжелые фазы.

Control Master

Для использования Control Master надо указывать несколько опций ssh-клиенту:

для первого соединения:

-M -S ~/.ssh/ssh_mux_%h_%p_%r (ControlMaster yes, ControlPath...)

для последующих:

-S ~/.ssh/ssh_mux_%h_%p_%r (ControlPath …)

Либо настроить в .ssh/config

.ssh/config

Мы узнали о куче разных опций, но каждый раз городить большие командные строки - тяжело, особенно если их надо переиспользовать.

У ssh-клиента есть свой конфиг - файл .ssh/config, который может быть использован для всякого рода настроек. Например:

```
Host *
    ForwardAgent yes
    ControlMaster auto
    ControlPersist 5
    ControlPath ~/.ssh/ssh_mux_%h_%p_%r
```

Включает для всех хостов ForwardAgent, устанавливает ControlMaster в auto (если сокет есть, то использовать его, а если нет - создать), ControlPersist в 5 (не закрывать сокет 5 секунд после завершения), ControlPath

.ssh/config

или

```
host s2
    user root
    hostname 127.0.0.1
    port 20122
    LocalForward 127.0.0.1:20123 192.168.0.2:22
```

Указывает, что для хоста s2 (может не резолвиться и не существовать вовсе) ssh должен использовать для соединения 127.0.0.1:20122 (hostname:port), использовать имя root и пробрасывать локальный порт 20123 в 192.168.0.2:22 с той машины, к которой мы соединяемся.

подробнее обо всем этом можно почитать в man ssh_config

Спасибо за внимание.

Дмитрий Молчанов
Григорий Ожегов