Базовая настройка сервера Debian после установки

serveradmin.ru/debian-nastroyka-servera

12 августа 2019 г.

После установки нового сервера приходится выполнять один и тот же набор стандартных настроек. Сегодня мы займемся базовой настройкой сервера под управлением операционной системы **Debian**. Я приведу практические советы по небольшому увеличению безопасности и удобству администрирования, основанные на моем личном опыте.

Данная статья является частью единого цикла статьей про сервер Debian.

Введение

Любая работа с сервером после установки чаще всего начинается со стандартных обязательных действий, без которых либо не получится продвинуться дальше, либо будет неудобно работать. Например, вам в любом случае необходимо выполнить сетевые настройки, желательно обновить систему и установить часовой пояс. Рекомендуется сразу настроить автообновление времени, подрихтовать параметры sshd, установить midnight commander и выполнить другие настройки.

Об этом я хочу рассказать в статье. Я буду делиться своим реальным опытом работы. Это не значит, что нужно делать так, как я. Я могу в чем-то ошибаться, что-то делать не так удобно, как можно было бы сделать. Это просто советы, которые кому-то помогут узнать что-то новое, а кто-то возможно поделится со мной чем-то новым для меня, либо укажет на мои ошибки. Мне бы хотелось, чтобы это было так. Своими материалами я не только делюсь с вами знаниями, но и сам узнаю что-то новое в том числе и из комментариев и писем на почту.

Указываем сетевые параметры

Итак, у нас в наличии только что установленная система. Узнать или проверить ее версию можно командами:

```
# uname -a
Linux debian10 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19) x86_64 GNU/Linux

# lsb_release -a
No LSB modules are available.
Distributor ID: Debian
Description:    Debian GNU/Linux 10 (buster)
Release:        10
Codename:       buster
```

Очень подробно про настройку сети в Debian я написал в отдельной статье. Рекомендую с ней ознакомиться. Здесь же кратко выполним основное. Для настройки сети, необходимо отредактировать файл */etc/network/interfaces.* Сделаем это:

```
# nano /etc/network/interfaces
```

Для получения IP адреса по dhcp достаточно будет следующего содержания:

```
allow-hotplug eth0
iface eth0 inet dhcp
```

Если у вас статический адрес, то его настроить можно следующими параметрами в файле:

```
allow-hotplug eth0
iface eth0 inet static
address 192.168.1.24
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 192.168.1.1
```

Сохраняем файл. Теперь нужно выполнить перезапуск сети. В Debian это делается командой:

```
# systemctl restart networking.service
```

В системном логе /var/log/syslog при этом будут записи:

```
debian10 systemd[1]: Stopping Raise network interfaces...
debian10 systemd[1]: networking.service: Succeeded.
debian10 systemd[1]: Stopped Raise network interfaces.
debian10 systemd[1]: Starting Raise network interfaces...
debian10 systemd[1]: Started Raise network interfaces.
```

Будьте аккуратны при настройке и перезапуске сети, если подключаетесь к серверу удаленно. Обязательно должен быть доступ к консоли на случай, если где-то ошибетесь и потеряете доступ к серверу.

К сетевым настройкам я отношу установку пакета **net-tools**, в состав которого входят старые и привычные утилиты для работы с сетью - ifconfig, netstat, route и другие. В современных дистрибутивах их заменили одной командой ip, но лично мне вывод некоторых старых команд, конкретно, netstat, нравится больше, поэтому я иногда ими тоже пользуюсь.

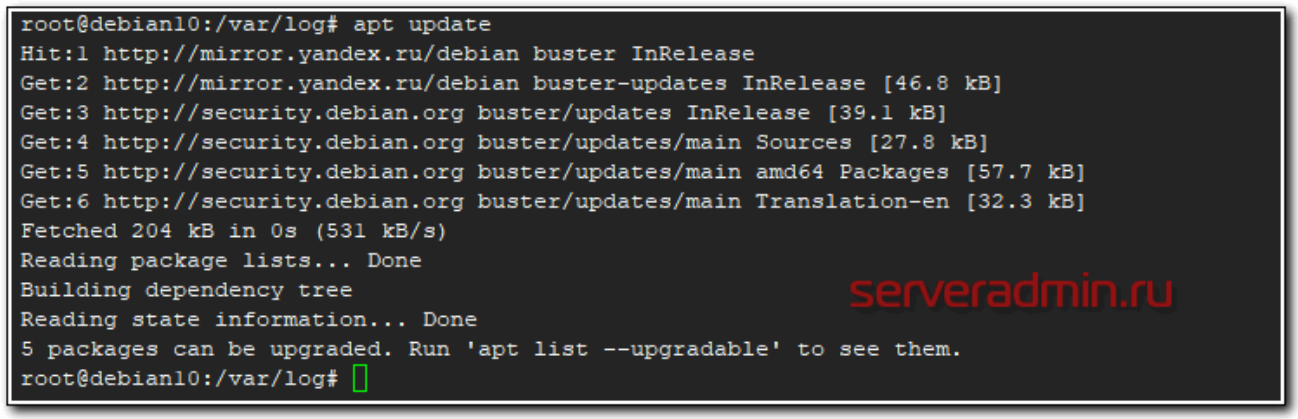
```
# apt install net-tools
```

На этом настройка сети закончена.

Обновление системы, отличие apt upgrade от dist-upgrade и full-upgrade

Сеть настроили, теперь можно обновить систему и пакеты. В Debian это делается достаточно просто. Воспользуемся несколькими командами. Сначала обновим локальный индекс пакетов до последних изменений в репозиториях:

```
# apt update
```

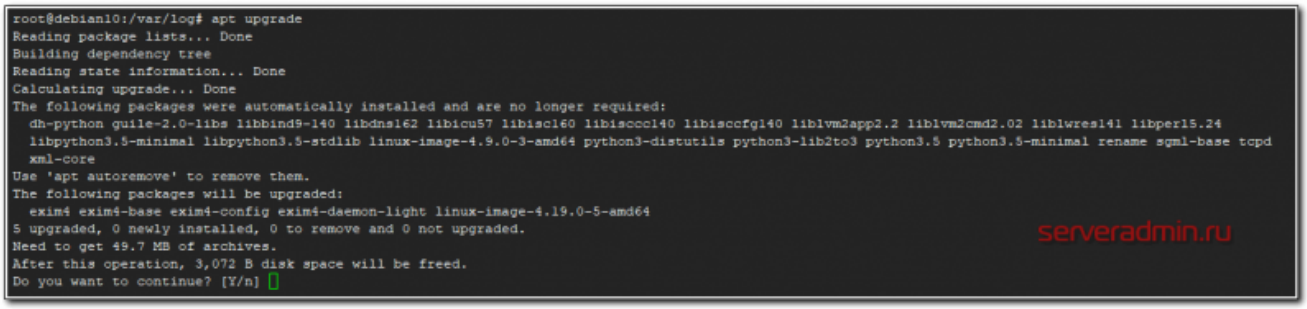


Посмотреть список пакетов, готовых к обновлению, можно с помощью команды:

```
# apt list --upgradable
```

Теперь выполним простое обновление всех пакетов системы:

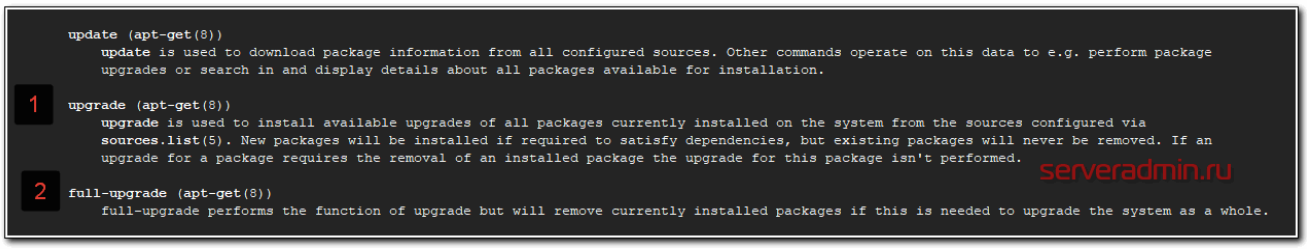
```
# apt upgrade
```



Ключ **upgrade** выполняет только обновление одной версии пакета на другую, более свежую. Он не будет устанавливать или удалять пакеты, даже если это необходимо для обновления других. Это наиболее безопасный и надежный вариант обновления, но он может обновить не все. Например, с ее помощью не обновить ядро до более свежей версии.

Ключ **dist-upgrade** или **full-upgrade** (это одно и то же) в дополнение к upgrade обрабатывает все изменения зависимостей для новых пакетов и во время работы может удалять ненужные и ставить необходимые пакеты для обновления. Вот

выдержка из документации по поводу этих двух ключей.



Так что после обычного обновления, делаем еще full-upgrade.

```
# apt full-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  dh-python guile-2.0-libs libbind9-140 libdns162 libicu57 libisc160 libisccc140
libisccfg140 liblvm2app2.2 liblvm2cmd2.02 liblwres141 libperl5.24
  libpython3.5-minimal libpython3.5-stdlib linux-image-4.9.0-3-amd64 python3-
distutils python3-lib2to3 python3.5 python3.5-minimal rename sgml-base tcpd
  xml-core
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Мне предлагается удалить старые пакеты, которые больше уже не нужны. Это зависимости от старых версий софта, который уже обновился и получил новые пакеты из зависимостей, а эти ему больше не нужны. Очистим их командой:

```
# apt autoremove
```

Рекомендую делать это регулярно после обновлений, чтобы старые пакеты не занимали лишнее место на диске.

На этом обновление системы закончено. Если вы хотите обновить версию релиза, например Debian 9 обновить до Debian 10 Buster, то читайте отдельный материал.

Настройка ssh

Теперь внесем некоторые изменения в настройки сервера **ssh**. Я рекомендую его запускать на нестандартном порту для исключения лишних общений с ботами, которые регулярно сканируют интернет и подбирают пароли пользователей по словарям.

Существует расхожее мнение, что менять порт ssh это наивность, а не защита. Надо просто настроить сертификаты, fail2ban или еще каким-то образом защитить ssh порт, к примеру, с помощью ограничений iptables, и т.д. Тем не менее, я все же рекомендую порт сменить на нестандартный. Даже если у вас все защищено от подбора паролей,

так как вы используете сертификаты, лишние запросы к ssh порту тратят ресурсы сервера, хоть и не очень большие. Идет установка соединения, обмен рукопожатиями и т.д. Зачем вам это нужно?

По-умолчанию в Debian, впрочем как и в любом другом дистрибутиве Linux, ssh сервер работает на 22 порту. Изменим этот порт, к примеру, на 23331. Так же я еще изменяю конфигурацию для разрешения подключения по ssh пользователя root с использованием пароля. В Debian из коробки пользователь root по ssh паролем авторизовываться не может. Изменим и это. Открываем файл настроек:

```
# nano /etc/ssh/sshd_config
```

И изменяем там следующие строки. Приводим их к виду:

```
Port 23331
PermitRootLogin yes
```

Сохраняем изменения и перезапускаем сервер ssh следующей командой:

```
# service sshd restart
```

Проверяем изменения:

```
# netstat -tulnp | grep ssh

tcp 0 0 0.0.0.0:23331 0.0.0.0:* LISTEN 925/sshd
tcp6 0 0 :::23331 :::* LISTEN 925/sshd
```

Все в порядке, сервер слушает 23331 порт. Теперь новое подключение будет осуществлено только по порту 23331. При этом, после перезапуска ssh, старое подключение не будет разорвано.

Я знаю, что многие возражают против подключения рутом к серверу. Якобы это небезопасно и т.д. и т.п. Мне эти доводы кажутся не убедительными. Не понимаю, в чем может быть проблема, если у меня нормальный сложный пароль на root, который не получится подобрать или сбрутить. Ни разу за всю мою работу системным администратором у меня не возникло проблем с этим моментом. А вот работать так значительно удобнее, особенно, когда необходимо оперативно куда-то подключиться по форс мажорным обстоятельствам.

Отдельно тему подключения к серверу под root я рассмотрел в статье про sudo. Кому интересно, переходите в нее и делитесь своим мнением на этот счет.

Установка утилит mc, htop, iftop

Следующим шагом я настраиваю некоторые полезные утилиты, которыми регулярно пользуюсь в повседневной работе. Первая из них это всем известный двухпанельный файловый менеджер Midnight Commander. Установим **mc** на наш сервер:

```
# apt install mc
```

И сразу же для него включаю подсветку синтаксиса всех файлов, которые не обозначены явно в файле */usr/share/mc/syntax/Syntax* синтаксисом для sh и bash скриптов. Этот универсальный синтаксис нормально подходит для конфигурационных файлов, с которыми чаще всего приходится работать на сервере. Перезаписываем файл *unknown.syntax*. Именно этот шаблон будет применяться к .conf и .cf файлам, так как к ним явно не привязано никакого синтаксиса.

```
# cp /usr/share/mc/syntax/sh.syntax /usr/share/mc/syntax/unknown.syntax
```

Я сразу же ставлю редактором по-умолчанию **mcedit**. Для этого просто выбираю его из меню при первом редактировании какого-нибудь файла. Если у вас такое меню не появляется, можете вызвать его сами и выбрать необходимый редактор по-умолчанию:

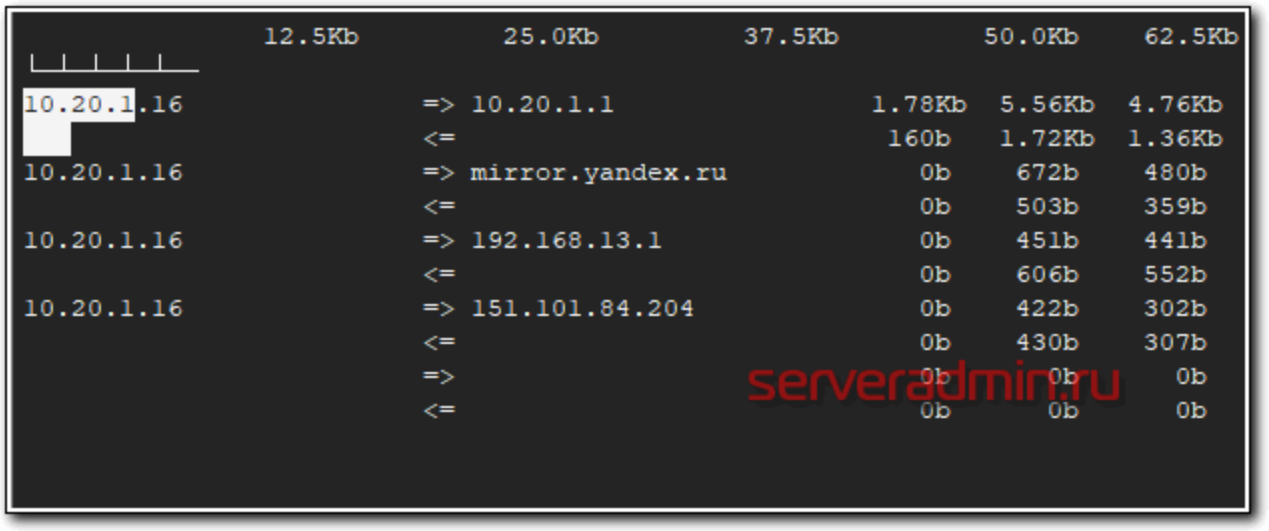
```
# select-editor

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano <---- easiest
 2. /usr/bin/mcedit
 3. /usr/bin/vim.tiny

Choose 1-3 [1]: 2
```

Так же я рекомендую очень удобный диспетчер задач - **htop**. Мне он помог, к примеру, решить проблему Взлома сервера CentOS. Ставим его на сервер:

```
# apt install htop
```

Полезной утилитой, позволяющей смотреть сетевую загрузку в режиме реального времени, является **iftop**. Очень рекомендую. Более простого и удобного инструмента мне не попадалось, хотя я много перепробовал подобных вещей. Устанавливаем iftop на сервер:

```
# apt install iftop
```



Настройка и обновление времени в Debian

Теперь проверим установленный часовой пояс, время и включим автоматическую синхронизацию времени с удаленного сервера. Очень подробно этот вопрос я рассмотрел в отдельной статье - настройка времени в Debian.

Узнать дату, время, часовой пояс можно командой **date**:

```
# date

Mon 12 Aug 2019 02:29:03 PM MSK
```

Если все указано верно, то менять ничего не нужно. Если же у вас неправильное время или указан часовой пояс не соответствующий вашему, то настроить это можно следующим образом. Сначала обновим часовые пояса:

```
# apt install tzdata
```

Теперь выберем правильный часовой пояс с помощью команды:

```
# dpkg-reconfigure tzdata
```

Выбирая соответствующие пункты визарда, указываете свой часовой пояс.

Дальше синхронизируем время с сервером времени в интернете. Для разовой или ручной синхронизации понадобится отдельная утилита**.** Установим **ntpdate** на сервер:

```
# apt install ntpdate
```

И синхронизируем время:

```
# ntpdate-debian

12 Aug 14:30:21 ntpdate[8688]: adjust time server 89.109.251.21 offset 0.004529 sec
```

Если получаете ошибку:

```
12 Aug 14:30:21 ntpdate[8688]: the NTP socket is in use, exiting
```

Значит у вас уже работает служба ntp. Ее нужно остановить и обновить время вручную. Хотя если она работает, то у вас и так должно быть все в порядке.

Для того, чтобы время автоматически синхронизировалось без вашего участия с определенной периодичностью, используется инструмент **ntp**. Установим его:

```
# apt install ntp
```

После установки он сам запустится и будет автоматически синхронизировать часы сервера. Проверим, запустился ли сервис ntpd:

```
# netstat -tulnp | grep ntp

udp        0      0 10.20.1.16:123          0.0.0.0:*
8855/ntpd
udp        0      0 127.0.0.1:123           0.0.0.0:*
8855/ntpd
udp        0      0 0.0.0.0:123             0.0.0.0:*
8855/ntpd
udp6       0      0 fe80::cce1:23ff:fe4:123 :::*
8855/ntpd
udp6       0      0 ::1:123                 :::*
8855/ntpd
udp6       0      0 :::123                  :::*
8855/ntpd
```

Настройка firewall (iptables) в Debian

В качестве firewall в Debian по-умолчанию используется **iptables**, его и будем настраивать. Изначально фаервол полностью открыт и пропускает весь трафик. Проверить список правил iptables можно следующей командой:

```
# iptables -L -v -n

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

Обращаю пристальное внимание на то, что настраивать firewall без прямого доступа к консоли сервера не следует. Особенно, если вы не очень разбираетесь в этом и

копируете команды с сайта. Шанс ошибиться очень высок. Вы просто потеряете удаленный доступ к серверу.
Создадим файл с правилами iptables:

```
# mcedit /etc/iptables.sh
```

Очень подробно вопрос настройки iptables я рассмотрел отдельно, рекомендую ознакомиться. Хотя в примере другая ОС linux, принципиальной разницы нет, настройки iptables абсолютно одинаковые, так как правила одни и те же.

Добавляем набор простых правил для базовой настройки. Все необходимое вы потом сможете сами открыть или закрыть по аналогии с существующими правилами:

```
#!/bin/bash
#
# Объявление переменных
export IPT="iptables"

# Активный сетевой интерфейс
export WAN=ens18
export WAN_IP=10.20.1.16

# Очистка всех цепочек iptables
$IPT -F
$IPT -F -t nat
$IPT -F -t mangle
$IPT -X
$IPT -t nat -X
$IPT -t mangle -X

# Установим политики по умолчанию для трафика, не соответствующего ни одному из
правил
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

# разрешаем локальный траффик для loopback
$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT

# разрешаем пинги
$IPT -A INPUT -p icmp --icmp-type echo-reply -j ACCEPT
$IPT -A INPUT -p icmp --icmp-type destination-unreachable -j ACCEPT
$IPT -A INPUT -p icmp --icmp-type time-exceeded -j ACCEPT
$IPT -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Разрешаем исходящие соединения самого сервера
$IPT -A OUTPUT -o $WAN -j ACCEPT

# Состояние ESTABLISHED говорит о том, что это не первый пакет в соединении.
# Пропускать все уже инициированные соединения, а также дочерние от них
$IPT -A INPUT -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
# Пропускать новые, а так же уже инициированные и их дочерние соединения
$IPT -A OUTPUT -p all -m state --state ESTABLISHED,RELATED -j ACCEPT
# Разрешить форвардинг для уже инициированных и их дочерних соединений
$IPT -A FORWARD -p all -m state --state ESTABLISHED,RELATED -j ACCEPT

# Включаем фрагментацию пакетов. Необходимо из-за разных значений MTU
$IPT -I FORWARD -p tcp --tcp-flags SYN,RST SYN -j TCPMSS --clamp-mss-to-pmtu

# Отбрасывать все пакеты, которые не могут быть идентифицированы
# и поэтому не могут иметь определенного статуса.
$IPT -A INPUT -m state --state INVALID -j DROP
$IPT -A FORWARD -m state --state INVALID -j DROP
```

```
# Приводит к связыванию системных ресурсов, так что реальный
# обмен данными становится не возможным, обрубаем
$IPT -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
$IPT -A OUTPUT -p tcp ! --syn -m state --state NEW -j DROP

# Открываем порт для ssh (!!!не забудьте указать свой порт, который вы изменили
ранее!!!)
$IPT -A INPUT -i $WAN -p tcp --dport 22 -j ACCEPT
# Открываем порт для web сервера
$IPT -A INPUT -i $WAN -p tcp --dport 80 -j ACCEPT
$IPT -A INPUT -i $WAN -p tcp --dport 443 -j ACCEPT

# Записываем правила в файл
/sbin/iptables-save > /etc/iptables_rules
```

Даем файлу права на запуск:

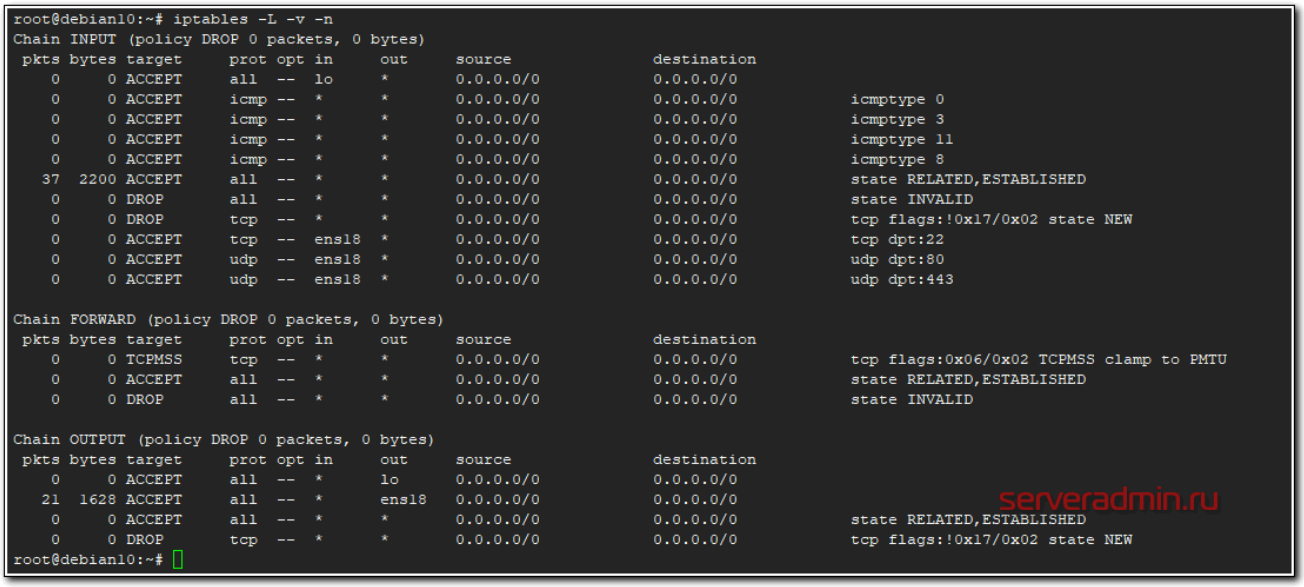
```
# chmod 0740 /etc/iptables.sh
```

Запускаем скрипт:

```
sh /etc/iptables.sh
```

Проверяем правила:

```
# iptables -L -v -n
```



Проверяем, что правила записались в файл */etc/iptables_rules*. Если их там нет, то записываем их вручную.

```
# /sbin/iptables-save > /etc/iptables_rules
```

Правила применились и произошла их запись в файл */etc/iptables_rules*. Теперь нужно сделать так, чтобы они применялись при загрузке сервера. Для этого делаем следующее. Открываем файл */etc/network/interfaces* и добавляем в него строку pre-up iptables-restore < /etc/iptables_rules Должно получиться вот так:

```
# cat /etc/network/interfaces

allow-hotplug eth0
iface eth0 inet dhcp
pre-up iptables-restore < /etc/iptables_rules
```

Для проверки перезагрузите сервер и посмотрите правила iptables. Должен загрузиться настроенный набор правил из файла */etc/iptables_rules*.

Настройка логов cron

По-умолчанию, в Debian нет отдельного лог файла для событий cron, они все сыпятся в общий лог */var/log/syslog*. Лично мне это не очень нравится, я предпочитаю выводить эти события в отдельный файл. Об этом я написал отдельно - вывести логи cron в отдельный файл. Рекомендую пройти по ссылке и настроить, если вам это необходимо. Там очень кратко и только по делу, не буду сюда копировать эту информацию.

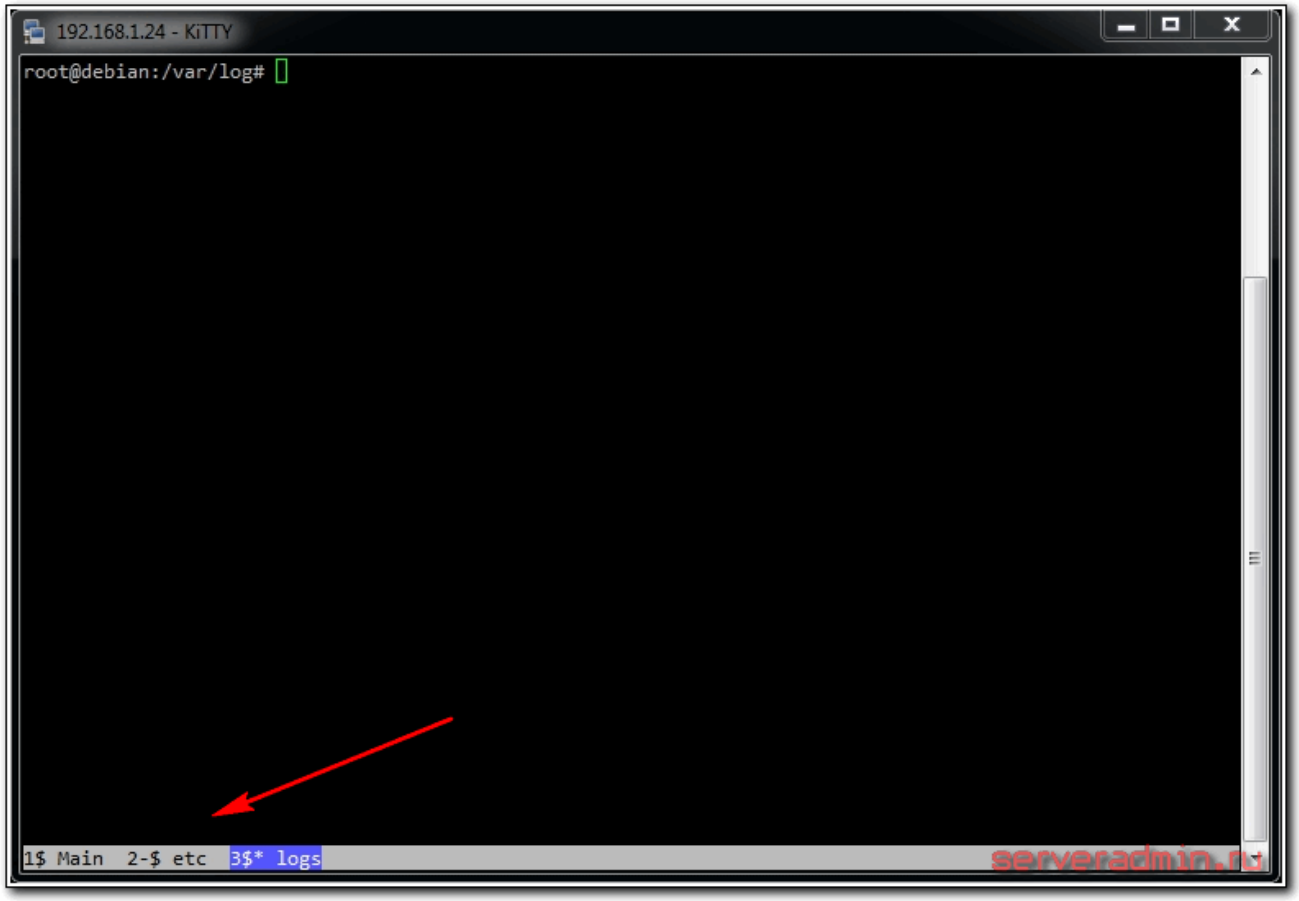
Установка и настройка screen

Я привык в своей работе пользоваться консольной утилитой screen. Изначально она задумывалась как инструмент, который позволяет запустить что-то удаленно в консоли, отключиться от сервера и при этом все, что выполняется в консоли продолжит свою работу. Вы сможете спокойно вернуться в ту же сессию и продолжить работу.

Первое время я именно так и использовал эту утилиту. Редко ее запускал, если не забывал, когда выполнялся какой-то длительный процесс, который жалко было прервать из-за случайного обрыва связи или необходимости отключить ноутбук от сети и куда-то переместиться.

Позже я решил подробнее ознакомиться с этим инструментом и обнаружил, что там есть несколько удобных моментов, которые можно использовать в ежедневной работе. Вот как использую утилиту screen я. При подключении к серверу у меня запускается screen с тремя окнами 1, 2, 3. Первое окно автоматически переходит в каталог /, второе в /etc, третье в /var/log. Я осмысленно назвал эти окна: Main, etc, logs соответственно. Внизу находится строка состояния, в которой отображен список всех открытых окон и подсвечено активное окно.

С помощью горячих клавиш я очень быстро переключаюсь между окнами в случае необходимости. Вот как выглядит мое рабочее окно ssh подключения:



Переключаюсь между окнами с помощью стандартных горячих клавиш screen: ctrl+a 1, ctrl+a 2, ctrl+a 3. Я специально изменил нумерацию, чтобы она начиналась не с 0 по-дефолту, а с 1. Так удобнее на клавиатуре переключать окна. Кнопка 0 находится слишком далеко от 1 и 2.

Чтобы настроить такую же работу screen, как у меня, достаточно выполнить несколько простых действий. Сначала устанавливаем screen:

```
# apt install screen
```

Создаем в каталоге */root* конфигурационный файл *.screenrc* следующего содержания:

```
# mcedit /root/.screenrc
```

```
#Выводим строку состояния
hardstatus alwayslastline "%-Lw%{= BW}%50>%n%f* %t%{-}%+Lw%<"

# Добавляем некоторые настройки
startup_message off
defscrollback 1000
defutf8 on
shell -$SHELL

# Создаем несколько окон
chdir
screen -t Main 1
chdir /etc
screen -t etc 2
chdir /var/log
screen -t logs 3

# Активное первое окно после запуска
select 1
```

Для знакомства с настройками, горячими клавишами и вариантами применения утилиты screen можно по адресу http://itman.in/ssh-screen/ Мне помог этот материал. Написано кратко, по делу и доходчиво.

Заключение

Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!
Теперь можно перезагрузить сервер и проверить, все ли в порядке. У меня все в порядке, проверил :) На этом базовая настройка сервера debian окончена. Можно приступать к конфигурации различных сервисов, под которые он настраивался. Об этом я рассказываю в отдельных статьях.

Напоминаю, что данная статья является частью единого цикла статьей про сервер Debian.
