У меня имеется свежеустановленный сервер CentOS 7 на VDS с виртуализацией KVM. Я расскажу о том, как сделать базовую настройку сервера для использования его в любом качестве на ваше усмотрение. Это может быть web сервер, vpn сервер, сервер мониторинга. Я расскажу о начальных настройках системы CentOS, которые повышают безопасность и удобство работы с сервером. Отмечу, что в 7-й версии системы произошли некоторые изменения по сравнению с предыдущими версиями.

Содержание:

Введение

Для настройки практически любого сервера требуется выполнить ряд стандартных шагов, которые мало чем отличаются в различных ситуациях. Какой бы функционал вы не готовили, вам придется настроить правильное время и включить его автообновление. Без установки сетевых настроек я вообще не представляю работу современного сервера. В голову не приходит ни один пример. Один и тот же набор настроек практически на автомате выполняется после установки. Своими наработками по этой теме я хочу поделиться с вами — то, что я в первую очередь настраиваю на новоиспеченном сервере centos.

> Еще раз обращаю внимание, что данные настройки я делаю на виртуальном сервере. Если у вас железный сервер, то рекомендуется выполнить еще некоторые настройки, о которых я здесь не упоминаю. К ним относится, к примеру, настройка и проверка отказоустойчивости при выходе одного из дисков. Отключение регулярной проверки массива mdadm и др.

Начальная настройка CentOS 7

```
Итак, у нас имеется:
# uname -a
Linux zeroxzed.ru 3.10.0-123.20.1.el7.x86_64 #1 SMP Thu Jan 29 18:05:33 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
```

Первым делом обновим базовую систему:

```
# yum update
```

Для удобства администрирования, я всегда устанавливаю Midnight Commander, или просто mc:

```
# yum 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
```

Дальше нам пригодятся сетевые утилиты. В минимальной настройке вы будете удивлены, когда наберете команду:

```
# ifconfig
```

И увидите ответ:

```
-bash: ifconfig: command not found
```

По крайней мере я, когда впервые это увидел, прилично удивился. Подумал, что ошибся в написании команды, перепроверил все несколько раз, но без результата. Оказалось, что надо отдельно установить пакет для выполнения ifconfig и прочих сетевых утилит.

Вместо ifconfig в CentOS 7 теперь утилита **ip**. Я не понимаю, зачем пилить отдельные программы для управления сетевыми настройками, если ifconfig и так отлично справляется с задачей. К тому же мне всегда нравилось, что в различных дистрибутивах линукс все примерно одинаковое. С помощью ifconfig можно настроить сеть не только в linux, но и в freebsd. Это удобно. А когда в каждом дистрибутиве свой инструмент это неудобно. Так что предлагаю установить привычный ifconfig.

Сделаем это:

```
# yum install net-tools
```

Теперь, чтобы у нас работали команды nslookup или, к примеру, host необходимо установить пакет bind-utils. Если этого не сделать, то на команду:

```
# nslookup
```

Будет вывод:

```
-bash: nslookup: command not found
```

Так что устанавливаем bind-utils:

```
# yum install bind-utils
```

Отключаем SELinux. Его использование и настройка отдельный разговор. Сейчас я не буду этим заниматься. Так что отключаем:

```
# mcedit /etc/sysconfig/selinux
```

меняем значение
SELINUX=disabled
Чтобы изменения вступили в силу, перезагружаемся:

```
# reboot
```

Можно без перезагрузки применить отключение SElinux:

```
# setenforce 0
```

Указываем сетевые параметры

Теперь произведем настройку сети в CentOS. Для этого открываем файл /etc/sysconfig/network-scripts/ifcfg-eth0

```
# mcedit /etc/sysconfig/network-scripts/ifcfg-eth0
```

В поле IPADDR вводим свой адрес, в NETMASK маску сети, в GATEWAY шлюз, DNS1 адрес днс сервера. Сохраняем файл и перезапускаем сеть для применения настроек:

```
# /etc/init.d/network restart
```

Настраиваем firewall

Очень подробно вопрос настройки iptables в CentOS 7 я рассмотрел отдельно. Сейчас мы быстро и просто настроим firewall. В CentOS 7 в качестве фаервола выступает iptables. По-умолчанию он запущен. Чтобы посмотреть текущие правила, нужно ввести команду:

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

Сразу хочу предупредить, что не имея доступа к консоли сервера, настраивать firewall плохая идея. Даже если вы очень хорошо понимаете что делаете и проделывали подобное много раз, все равно есть шанс остаться без доступа к серверу. Так что первым делом перед настройкой iptables проверяем доступ к консоли через KVM или физически.

В 7-й версии CentOS для управления iptables разработан новый инструмент под названием firewalld и все управление производится через него. Я не понял зачем это сделали, и не могу сказать, удобнее с ним стало или нет. По мне, так удобнее использовать одни и те же наработки по iptables. Мигрируя от сервера к серверу и от дистрибутива к дистрибутиву, я просто редактирую скрипт настроек фаервола.

Но CentOS зачем-то придумали firewalld, в Ubuntu стоит ufw, но суть одна и та же — это утилиты для конфигурирования iptables, который один и тот же во всех дистрибутивах. Я привык управлять iptables через самописный скрипт, который переношу от сервера к серверу и редактирую под конкретные потребности. Этим скриптом я и поделюсь. Так что для начала остановим и отключим firewalld:

```
# systemctl stop firewalld
# systemctl disable firewalld
rm '/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service'
rm '/etc/systemd/system/basic.target.wants/firewalld.service'
```

Установим утилиты для iptables:

```
# yum -y install iptables-services
```

Включим автозапуск iptables:

```
# systemctl enable iptables
```

Теперь создадим файл /etc/iptables_rules.sh следующего содержания:

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

# Интерфейс который смотрит в интернет
export WAN=eth0
export WAN_IP=149.154.71.205

# 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 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

# Рзрешаем 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

# Открываем порт для ssh
$IPT -A INPUT -i $WAN -p tcp --dport 22 -j ACCEPT
# Открываем порт для DNS
#$IPT -A INPUT -i $WAN -p udp --dport 53 -j ACCEPT
# Открываем порт для NTP
#$IPT -A INPUT -i $WAN -p udp --dport 123 -j ACCEPT

# Логирование
```

```
# Все что не разрешено, но ломится отправим в цепочку undef

$IPT -N undef_in
$IPT -N undef_out
$IPT -N undef_fw
$IPT -A INPUT -j undef_in
$IPT -A OUTPUT -j undef_out
$IPT -A FORWARD -j undef_fw

# Логируем все из undef

$IPT -A undef_in -j LOG --log-level info --log-prefix "-- IN -- DROP "
$IPT -A undef_in -j DROP
$IPT -A undef_out -j LOG --log-level info --log-prefix "-- OUT -- DROP "
$IPT -A undef_out -j DROP
$IPT -A undef_fw -j LOG --log-level info --log-prefix "-- FW -- DROP "
$IPT -A undef_fw -j DROP

# Записываем правила
/sbin/iptables-save  > /etc/sysconfig/iptables
```

В принципе, добавить нечего, в файле даны все комментарии. В таком виде, логи всего заблокированного будут писаться в файл /var/log/messages и записей там будет очень много. Так что в обычной работе эти строки нужно закомментировать, и использовать только во время отладки. Более подробное описание правил и примеры настроек firewall в случае, когда ваш сервер является шлюзом локальной сети, приведено по ссылке в начале раздела.

Делаем файл с правилами исполняемым и запускаем:

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

Проверяем, применились ли правила:

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

При каждом запуске файла с правилами iptables, все изменения записываются в файл /etc/sysconfig/iptables и применяются при загрузке системы.

Настройка SSH в CentOS 7

Дальше внесем некоторые изменения в работу ssh для увеличения безопасности. По-умолчанию, сервис работает на 22 порту и если все оставить как есть, то мы получим огромное количество попыток авторизоваться. Боты сканят непрерывно интернет и подбирают пароли к ssh. Чтобы обезопасить себя от сканов простых ботов, изменим порт, на котором работает ssh. Можно выбрать любой пятизначный номер, это не принципиально. От автоматического сканирования это защитит. Повесим демон ssh на 25333 порт. Для этого редактируем файл /etc/ssh/sshd_config

```
# mcedit /etc/ssh/sshd_config
```

Раскомментируем строку Port 22 и заменим значение 22 на 25333.
Так же я обычно разрешаю подключаться по ssh пользователю root. Мне так удобнее. Проблем с этим у меня никогда не возникало. Если вы считаете, что это не безопасно, не трогайте эту настройку. Чтобы разрешить пользователю root подключаться по ssh, раскомментируйте строку PermitRootLogin yes.

Сохраняем файл. Теперь обязательно изменяем настройки iptables, добавляем в разрешенные подключения вместо 22 порта 25333. Если этого не сделать, то после перезапуска sshd мы потеряем удаленный доступ к серверу. Итак, открываем /etc/iptables_rules.sh и меняем в строке

```
$IPT -A INPUT -i $WAN -p tcp --dport 22 -j ACCEPT
```

22 на 25333 и исполняем файл. Наше текущее соединение не оборвется, так как оно уже установлено, но заново подключиться по ssh к 22 порту уже н получится.

Перезапускаем sshd:

```
# systemctl restart sshd
```

Проверяем какой порт слушает sshd:

```
# netstat -tulpn | grep sshd
tcp        0      0 0.0.0.0:25333           0.0.0.0:*               LISTEN      1799/sshd
tcp6       0      0 :::25333                :::*                    LISTEN      1799/sshd
```

Если вывод такой же как у меня, то все в порядке, теперь к ssh можно подключаться по 25333 порту.

Добавим еще одну небольшую настройку. Иногда, когда возникают проблемы с dns сервером, логин по ssh подвисает на 30-60 секунд. Вы просто ждете после ввода логина, когда появится возможность ввести пароль. Чтобы избежать этого замедления, укажем ssh не использовать dns в своей работе. Для этого в конфиге раскомментируем строку с параметром UseDNS и отключим его. По-умолчанию он включен.

```
UseDNS no
```

Для применения изменений нужно перезапустить ssh службу, как мы уже делали ранее.

Настраиваем время

Узнать, какое время на сервере можно с помощью команды date:

```
# date
```

Чтобы сменить часовой пояс, необходимо выбрать подходящий файл часовой зоны в /usr/share/zoneinfo. В случае, если у вас часовой пояс Москвы, выполните следующее:

```
# mv /etc/localtime /etc/localtime.bak
# ln -s /usr/share/zoneinfo/Europe/Moscow /etc/localtime
```

Либо можете воспользоваться специальной утилитой, которая входит в комплект CentOS 7. Делает она ровно то же самое:

```
# timedatectl set-timezone Europe/Moscow
```

В CentOS 7 есть утилита для синхронизации времени **chrony**. В стандартной установке она должна быть установлена в системе, в минимальной ее нет. Если у вас она не стоит, то устанавливайте вручную:

```
# yum install chrony
```

Запускаем chrony и добавляем в автозагрузку:

```
# systemctl start chronyd
# systemctl enable chronyd
```

Проверяем, нормально ли запустился:

```
# systemctl status chronyd
● chronyd.service - NTP client/server
 Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)
 Active: active (running) since Fri 2016-08-05 00:33:09 MSK; 52min left
 Main PID: 667 (chronyd)
 CGroup: /system.slice/chronyd.service
 └─667 /usr/sbin/chronyd

Aug 05 00:33:09 centos.local systemd[1]: Starting NTP client/server...
Aug 05 00:33:09 centos.local chronyd[667]: chronyd version 2.1.1 starting (+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +DEBUG
+ASYNCDNS +IPV6 +SECHASH)
Aug 05 00:33:09 centos.local chronyd[667]: Generated key 1
Aug 05 00:33:09 centos.local systemd[1]: Started NTP client/server.
Aug 05 00:33:26 centos.local chronyd[667]: Selected source 85.21.78.91
Aug 05 00:33:26 centos.local chronyd[667]: System clock wrong by -3595.761368 seconds, adjustment started
Aug 04 23:33:30 centos.local chronyd[667]: System clock was stepped by -3595.761368 seconds
```

Все в порядке, сервис работает. После запуска он автоматически синхронизирует время.

Для синхронизации времени вы можете воспользоваться более привычно программой, которая присутствует практически во всех unix дистрибутивах — **ntp**. Устанавливаем утилиту синхронизации времени ntp в CentOS:

```
# yum install ntp
```

Разово синхронизируем время:

```
# /usr/sbin/ntpdate pool.ntp.org
```

Если ntpdate не работает, посмотрите материал, может это ваш случай. Запустим демон синхронизации и запишем его запуск в автозагрузку:

```
# systemctl start ntpd
# systemctl enable ntpd
ln -s '/usr/lib/systemd/system/ntpd.service' '/etc/systemd/system/multi-user.target.wants/ntpd.service'
```

Теперь наши часы будут автоматически синхронизироваться с сервером времени.

Более подробно об этой теме написано отдельно в моем материале — установка, настройка и синхронизация времени в CentOS.

> Не используйте одновременно оба демона синхронизации времени — chrony и ntp. Выберите какой-нибудь один. Лично я не вижу в них разницы, сам чаще всего ставлю привычный ntp.

Добавление репозиториев

Для инсталляции различного софта необходимо подключить репозитории в CentOS. Наиболее популярные это EPEL и rpmforge, поэтому добавим их. Сначала ставим EPEL. С ним все просто, он добавляется из стандартного репозитория:

```
# yum install epel-release
```

Устанавливаем rpmforge:

```
# rpm --import http://apt.sw.be/RPM-GPG-KEY.dag.txt
# yum install http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.3-1.el7.rf.x86_64.rpm
```

В настоящее время приведенная выше ссылка не работает по неизвестным причинам, я надеюсь, что это временные проблемы с сайтом. Пока можно использовать альтернативную:

```
# yum install
http://repository.it4i.cz/mirrors/repoforge/redhat/el7/en/x86_64/rpmforge/RPMS/rpmforge-release-0.5.3-1.el7.rf.x86_64.rpm
```

> По последним данным, репозиторий rpmforge закрыт и больше не поддерживается https://github.com/repoforge/rpms/issues/375
> https://wiki.centos.org/AdditionalResources/Repositories/RPMForge

Настройка хранения истории в bash_history

Полезным будет внести некоторые изменения в стандартный механизм сохранения истории команд. Он часто выручает, когда надо вспомнить одну из ранее введенных команд. Стандартные настройки имеют некоторые ограничения, которые неудобны. Вот их список:

1. По-умолчанию, сохраняются только последние 1000 команд. Если их будет больше, то более старые будут удаляться и заменяться новыми.
2. Не указаны даты выполнения команд, только их список в порядке выполнения.
3. Файл со списком команд обновляется после завершения сессии. При параллельных сессиях часть команд может быть утеряна.
4. Сохраняются абсолютно все команды, хотя в хранении некоторых нет никакого смысла.

Список последних выполненных команд хранится в домашней директории пользователя в файле *.bash_history* (в начале точка). Его можно открыть любым редактором и посмотреть. Для более удобного вывода списка, можно в консоли ввести команду:

```
# history
```

и увидеть пронумерованный список. Быстро найти конкретную команду, можно с помощью фильтрации только нужных строк, например вот так:

```
# history | grep yum
```

Так мы увидим все варианты запуска команды yum, которые хранятся в истории. Исправим перечисленные недостатки стандартных настроек хранения истории команд в CentOS 7. Для этого нужно отредактировать файл *.bashrc*, который находится в том же каталоге, что и файл с историей. Добавляем в него следующие строки:

```
export HISTSIZE=10000
export HISTTIMEFORMAT="%h %d %H:%M:%S "
PROMPT_COMMAND='history -a'
export HISTIGNORE="ls:ll:history:w:htop"
```

Первый параметр увеличивает размер файла до 10000 строк. Можно сделать и больше, хотя обычно хватает такого размера. Второй параметр указывает, что необходимо сохранять дату и время выполнения команды. Третья строка вынуждает сразу же после выполнения команды сохранять ее в историю. В последней строке мы создаем список исключений для тех команд, запись которых в историю не требуется. Я привел пример самого простого списка. Можете дополнить его на свое усмотрение.

Для применения изменений необходимо разлогиниться и подключиться заново или выполнить команду:

```
# source ~/.bashrc
```

Автоматическое обновление системы

Для поддержания безопасности сервера на должном уровне необходимо как минимум своевременно его обновлять — как само ядро с системными утилитами, так и остальные пакеты. Можно делать это вручную, но для более эффективной работы лучше автоматизировать рутинные действия. Не обязательно устанавливать обновления автоматически, но как минимум проверять их появление. Я обычно придерживаюсь такой стратегии.

Для автоматической проверки обновлений нам поможет утилита **yum-cron**. Ставится она традиционно через yum из стандартного репозитория.

```
# yum install yum-cron
```

После установки создается автоматическое задание на выполнение утилиты в */etc/cron.daily* и */etc/cron.hourly*. По-умолчанию, утилита скачивает найденные обновления, но не применяет их. Вместо этого, администратору на локальный почтовый ящик root отправляется уведомление об обновлениях. Дальше вы уже в ручном режиме заходите и решаете, устанавливать обновления или нет в удобное для вас время. Мне такой режим работы видится наиболее удобным, поэтому я не меняю эти настройки.

Конфигурационные файлы yum-cron находятся по адресу */etc/yum/yum-cron.conf* и *yum-cron-hourly.conf*. Они неплохо прокомментированы, так что в подробных разъяснениях не нуждаются. Обращаю внимание на раздел **[email]**, где можно указать параметры отправки сообщений. По-умолчанию стоит отправка почты через локальный хост. Можно тут изменить параметры и отправлять сообщения через сторонний почтовый сервер. Но вместо этого лично я предпочитаю глобально для всего сервера настроить пересылку локальной почты root на внешний почтовый ящик через авторизацию на другом smtp сервере.

Отключаем флуд сообщений в /var/log/messages

В дефолтной установке системы CentOS 7, весь ваш системный лог */var/log/messages* через некоторое время работы сервера будет забит следующими записями.

```
Oct 16 14:01:01 xs-files systemd: Created slice user-0.slice.
Oct 16 14:01:01 xs-files systemd: Starting user-0.slice.
Oct 16 14:01:01 xs-files systemd: Started Session 14440 of user root.
Oct 16 14:01:01 xs-files systemd: Starting Session 14440 of user root.
Oct 16 14:01:01 xs-files systemd: Removed slice user-0.slice.
Oct 16 14:01:01 xs-files systemd: Stopping user-0.slice.
Oct 16 15:01:01 xs-files systemd: Created slice user-0.slice.
Oct 16 15:01:01 xs-files systemd: Starting user-0.slice.
Oct 16 15:01:01 xs-files systemd: Started Session 14441 of user root.
Oct 16 15:01:01 xs-files systemd: Starting Session 14441 of user root.
Oct 16 15:01:01 xs-files systemd: Started Session 14442 of user root.
Oct 16 15:01:01 xs-files systemd: Starting Session 14442 of user root.
Oct 16 15:01:01 xs-files systemd: Removed slice user-0.slice.
Oct 16 15:01:01 xs-files systemd: Stopping user-0.slice.
```

```
Oct 16 16:01:01 xs-files systemd: Created slice user-0.slice.
Oct 16 16:01:01 xs-files systemd: Starting user-0.slice.
Oct 16 16:01:01 xs-files systemd: Started Session 14443 of user root.
Oct 16 16:01:01 xs-files systemd: Starting Session 14443 of user root.
Oct 16 16:01:01 xs-files systemd: Removed slice user-0.slice.
```

Никакой практической пользы они не несут, поэтому отключим их. Для этого создадим отдельное правило для rsyslog, где перечислим все шаблоны сообщений, которые будем вырезать. Разместим это правило в отдельном файле */etc/rsyslog.d/ignore-systemd-session-slice.conf*.

```
# cd /etc/rsyslog.d && mcedit ignore-systemd-session-slice.conf
```

```
if $programname == "systemd" and ($msg contains "Starting Session" or $msg contains "Started Session" or $msg contains "Created slice" or $msg contains "Starting user-" or $msg contains "Starting User Slice of" or $msg contains "Removed session" or $msg contains "Removed slice User Slice of" or $msg contains "Stopping User Slice of") then stop
```

Сохраняем файл и перезапускаем rsyslog для применения настроек.

```
# systemctl restart rsyslog
```

Необходимо понимать, что в данном случае мы отключаем флуд в лог файл только на локальном сервере. Если вы храните логи на удаленном syslog сервере, то данное правило нужно будет настраивать именно на нем.

Установка iftop, atop, htop, lsof на CentOS 7

И напоследок добавим несколько полезных утилит, которые могут пригодиться в процессе эксплуатации сервера.

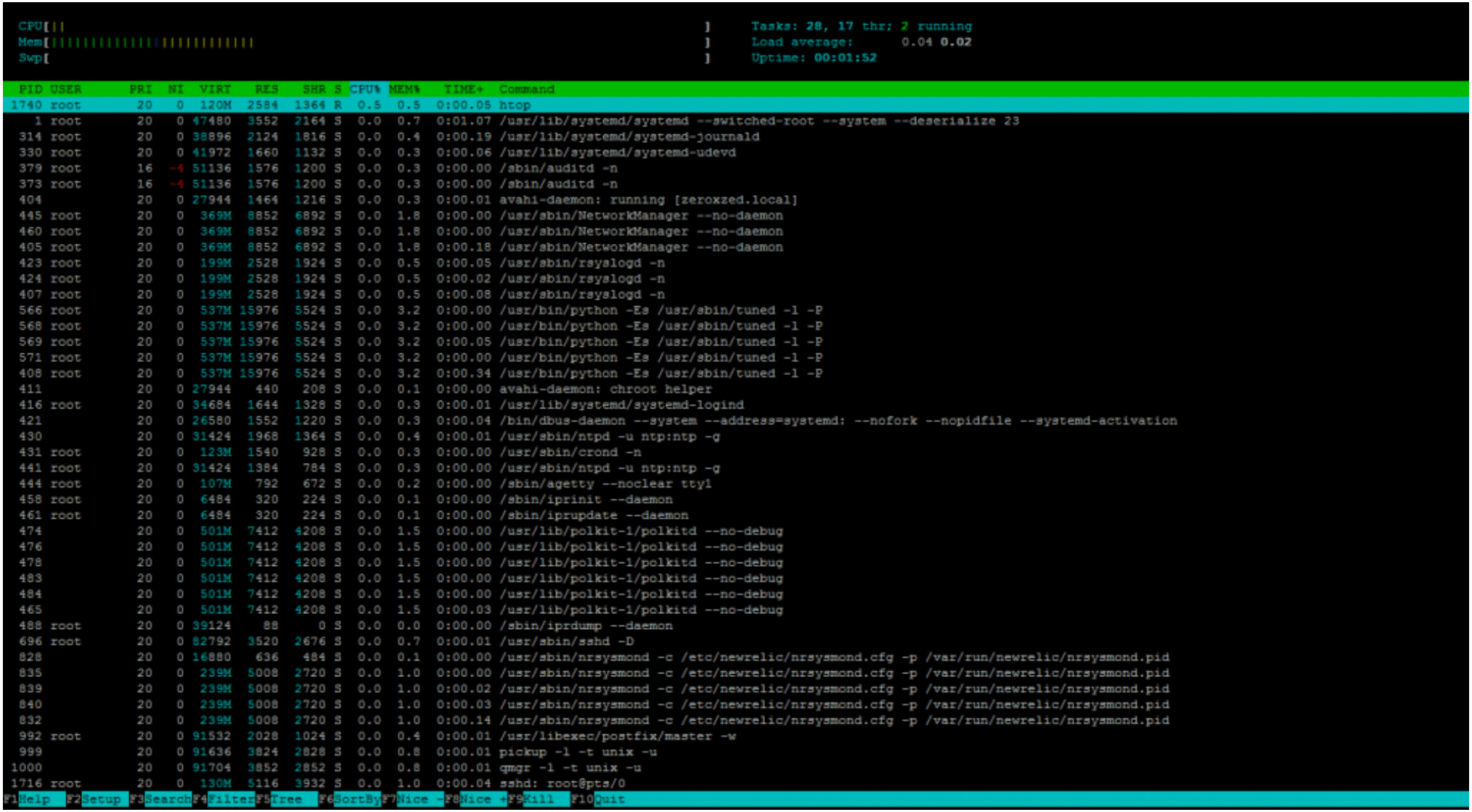
iftop показывает в режиме реального времени загрузку сетевого интерфейса, может запускаться с различными ключами, не буду останавливаться на этом подробно, в интернете есть информация на эту тему. Ставим:

```
# yum install iftop
```

И два интересных диспетчера задач, я чаще всего пользуюсь htop, но иногда пригодится и atop. Ставим оба, сами посмотрите, разберетесь, что вам больше нравится, подходит:

```
# yum -y install htop
# yum -y install atop
```

Вот как выглядит htop:


CPU[||
Mem[||||||||||||||||||||||||||||
Swp[
Tasks: 28, 17 thr; 2 running
Load average: 0.04 0.02
Uptime: 00:01:52
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1740 root 20 0 120M 2584 1364 R 0.5 0.5 0:00.05 htop
1 root 20 0 47480 3552 2164 S 0.0 0.7 0:01.07 /usr/lib/systemd/systemd --switched-root --system --deserialize 23
314 root 20 0 38896 2124 1816 S 0.0 0.4 0:00.19 /usr/lib/systemd/systemd-journald
330 root 20 0 41972 1660 1132 S 0.0 0.3 0:00.06 /usr/lib/systemd/systemd-udevd
379 root 16 -4 51136 1576 1200 S 0.0 0.3 0:00.00 /sbin/auditd -n
373 root 16 -4 51136 1576 1200 S 0.0 0.3 0:00.00 /sbin/auditd -n
404 20 0 27944 1464 1216 S 0.0 0.3 0:00.01 avahi-daemon: running [zeroxzed.local]
445 root 20 0 369M 8852 6892 S 0.0 1.8 0:00.00 /usr/sbin/NetworkManager --no-daemon
460 root 20 0 369M 8852 6892 S 0.0 1.8 0:00.00 /usr/sbin/NetworkManager --no-daemon
405 root 20 0 369M 8852 6892 S 0.0 1.8 0:00.18 /usr/sbin/NetworkManager --no-daemon
423 root 20 0 199M 2528 1924 S 0.0 0.5 0:00.05 /usr/sbin/rsyslogd -n
424 root 20 0 199M 2528 1924 S 0.0 0.5 0:00.02 /usr/sbin/rsyslogd -n
407 root 20 0 199M 2528 1924 S 0.0 0.5 0:00.08 /usr/sbin/rsyslogd -n
566 root 20 0 537M 15976 5524 S 0.0 3.2 0:00.00 /usr/bin/python -Es /usr/sbin/tuned -l -P
568 root 20 0 537M 15976 5524 S 0.0 3.2 0:00.00 /usr/bin/python -Es /usr/sbin/tuned -l -P
569 root 20 0 537M 15976 5524 S 0.0 3.2 0:00.05 /usr/bin/python -Es /usr/sbin/tuned -l -P
571 root 20 0 537M 15976 5524 S 0.0 3.2 0:00.00 /usr/bin/python -Es /usr/sbin/tuned -l -P
408 root 20 0 537M 15976 5524 S 0.0 3.2 0:00.34 /usr/bin/python -Es /usr/sbin/tuned -l -P
411 20 0 27944 440 208 S 0.0 0.1 0:00.00 avahi-daemon: chroot helper
416 root 20 0 34684 1644 1328 S 0.0 0.3 0:00.01 /usr/lib/systemd/systemd-logind
421 20 0 26580 1552 1220 S 0.0 0.3 0:00.04 /bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
430 20 0 31424 1968 1364 S 0.0 0.4 0:00.01 /usr/sbin/ntpd -u ntp:ntp -g
431 root 20 0 123M 1540 928 S 0.0 0.3 0:00.00 /usr/sbin/crond -n
441 root 20 0 31424 1384 784 S 0.0 0.3 0:00.00 /usr/sbin/ntpd -u ntp:ntp -g
444 root 20 0 107M 792 672 S 0.0 0.2 0:00.00 /sbin/agetty --noclear tty1
458 root 20 0 6484 320 224 S 0.0 0.1 0:00.00 /sbin/iprinit --daemon
461 root 20 0 6484 320 224 S 0.0 0.1 0:00.00 /sbin/iprupdate --daemon
474 20 0 501M 7412 4208 S 0.0 1.5 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
476 20 0 501M 7412 4208 S 0.0 1.5 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
478 20 0 501M 7412 4208 S 0.0 1.5 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
483 20 0 501M 7412 4208 S 0.0 1.5 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
484 20 0 501M 7412 4208 S 0.0 1.5 0:00.00 /usr/lib/polkit-1/polkitd --no-debug
465 20 0 501M 7412 4208 S 0.0 1.5 0:00.03 /usr/lib/polkit-1/polkitd --no-debug
488 root 20 0 39124 88 0 S 0.0 0.0 0:00.00 /sbin/iprdump --daemon
696 root 20 0 82792 3520 2676 S 0.0 0.7 0:00.01 /usr/sbin/sshd -D
828 20 0 16880 636 484 S 0.0 0.1 0:00.00 /usr/sbin/nrsysmond -c /etc/newrelic/nrsysmond.cfg -p /var/run/newrelic/nrsysmond.pid
835 20 0 239M 5008 2720 S 0.0 1.0 0:00.00 /usr/sbin/nrsysmond -c /etc/newrelic/nrsysmond.cfg -p /var/run/newrelic/nrsysmond.pid
839 20 0 239M 5008 2720 S 0.0 1.0 0:00.02 /usr/sbin/nrsysmond -c /etc/newrelic/nrsysmond.cfg -p /var/run/newrelic/nrsysmond.pid
840 20 0 239M 5008 2720 S 0.0 1.0 0:00.03 /usr/sbin/nrsysmond -c /etc/newrelic/nrsysmond.cfg -p /var/run/newrelic/nrsysmond.pid
832 20 0 239M 5008 2720 S 0.0 1.0 0:00.14 /usr/sbin/nrsysmond -c /etc/newrelic/nrsysmond.cfg -p /var/run/newrelic/nrsysmond.pid
992 root 20 0 91532 2028 1024 S 0.0 0.4 0:00.01 /usr/libexec/postfix/master -w
999 20 0 91636 3824 2828 S 0.0 0.8 0:00.01 pickup -l -t unix -u
1000 20 0 91704 3852 2852 S 0.0 0.8 0:00.01 qmgr -l -t unix -u
1716 root 20 0 130M 5116 3932 S 0.0 1.0 0:00.04 sshd: root@pts/0
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice - F8Nice + F9Kill F10Quit

Для вывода информации о том, какие файлы используются теми или иными процессами, советую поставить утилиту **lsof**. Она скорее всего рано или поздно пригодится, когда будете диагностировать работу сервера.

```
# yum install lsof
```

Рекомендую еще установить несколько нужных и полезных программ, которые часто необходимы, но отсутствуют в минимальной установке — **wget**, **bzip2**, **traceroute**, **gdisk**.

```
# yum install wget bzip2 traceroute gdisk
```

На этом у меня все. Базовая настройка CentOS 7 закончена, можно приступать к установке и настройке основного функционала.

Настройка системной почты

В завершение настройки сервера CentOS 7 сделаем так, что бы почта, адресованная локальному root, отправлялась через внешний почтовый сервер на выбранный почтовый ящик. Если этого не сделать, то она будет локально складываться в файл */var/spool/mail/root*. А там может быть важная и полезная информация. Настроим ее отправку в ящик системного администратора.

Подробно об этом я рассказал в отдельной статье — Отправка почты через консоль с авторизацией в linux. Здесь кратко только команды и быстрая настройка. Ставим необходимые пакеты:

```
# yum install mailx cyrus-sasl cyrus-sasl-lib cyrus-sasl-plain
```

Рисуем примерно такой конфиг для postfix.

```
cat /etc/postfix/main.cf
```

```
## DEFAULT CONFIG BEGIN ######################
queue_directory = /var/spool/postfix
command_directory = /usr/sbin
daemon_directory = /usr/libexec/postfix
```

```
data_directory = /var/lib/postfix
mail_owner = postfix
inet_interfaces = localhost
inet_protocols = all
unknown_local_recipient_reject_code = 550
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases

debug_peer_level = 2
debugger_command =
         PATH=/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin
         ddd $daemon_directory/$process_name $process_id & sleep 5

sendmail_path = /usr/sbin/sendmail.postfix
newaliases_path = /usr/bin/newaliases.postfix
mailq_path = /usr/bin/mailq.postfix
setgid_group = postdrop
html_directory = no
manpage_directory = /usr/share/man
sample_directory = /usr/share/doc/postfix-2.10.1/samples
readme_directory = /usr/share/doc/postfix-2.10.1/README_FILES
## DEFAULT CONFIG END ######################

# Имя сервера, которое выводит команда hostname
myhostname = centos7-test.xs.local
# Здесь по логике нужно оставлять только домен, но в данном случае лучше оставить полное имя сервера, чтобы в поле
отправитель
# фигурировало полное имя сервера, так удобнее разбирать служебные сообщения
mydomain = centos7-test.xs.local
mydestination = $myhostname
myorigin = $mydomain
# Адрес сервера, через который будем отправлять почту
relayhost = mailsrv.mymail.ru:25
```

```
smtp_use_tls = yes
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_security_level = may
```

Создаем файл с информацией об имени пользователя и пароле для авторизации.

```
# mcedit /etc/postfix/sasl_passwd
```

```
mailsrv.mymail.ru:25 admin@mymail.ru:password
```

Создаем db файл.

```
# postmap /etc/postfix/sasl_passwd
```

Теперь можно перезапустить postfix и проверить работу.

```
# systemctl restart postfix
```

К стандартному алиасу для root в */etc/aliases*, добавьте внешний адрес, куда будет дублироваться почта, адресованная root. Для этого редактируем указанный файл, изменяя последнюю строку.

Было:

```
#root: marc
```

Стало

```
root: root,admin@mymail.ru
```

Обновляем базу сертификатов:

```
# newaliases
```

Отправим письмо через консоль локальному руту:

```
# df -h | mail -s "Disk usage" root
```

Письмо должно уйти на внешний ящик. На этом настройка локальной почты закончена. Теперь все письма, адресованные локальному root, например, отчеты от cron, будут дублироваться на внешний почтовый ящик, причем с отправкой через нормальный почтовый сервер. Так что письма будут нормально доставляться, не попадая в спам (хотя не обязательно, есть еще эвристические фильтры).

Заключение

> Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

Мы выполнили некоторые начальные шаги по настройке сервера CentOS 7, которые я обычно делаю при подготовке сервера. Я не претендую на абсолютную истину, возможно что-то упускаю или делаю не совсем верно. Буду рад разумным и осмысленным комментариям и замечаниям с предложениями.

Полезно после базовой настройки сразу же подключить сервер к системе мониторинга. Либо настроить ее, если у вас еще нет. У меня есть подробный цикл статей по настройке мониторинга:

1. Пример настройки сервера мониторинга zabbix, либо только подключение centos к мониторингу путем установки на него агента.
2. В отдельной рубрике zabbix есть много примеров для мониторинга различных полезных метрик.

Из наиболее популярных и масштабных статей по настройке различного функционала на базе сервера centos хочу отметить следующие:

- Почтовый сервер postfix.
- Файловый сервер samba с интеграцией в AD.
- Настройка шлюза centos и прокси сервера squid.
- Пошаговая инструкция по настройке asterisk для среднего офиса.
- Пример соединения двух офисов с помощью openvpn.
- Настройка веб сервера на базе apache+php или nginx+php-fpm.

И никогда не забывайте про бэкап и его проверку — бэкап или перенос linux сервера.

Видео по настройке CentOS 7