Сегодня я познакомлю вас со своим видением начальной конфигурации универсального сервера на популярной ОС. Я расскажу о том, как сделать базовую настройку сервера centos сразу после установки для использования его в любом качестве на ваше усмотрение. Приведенные практические советы повышают безопасность и удобство работы с сервером. Статья будет актуальна для двух последних релизов Centos — 7 и 8.

Содержание:

Цели статьи

1. Перечислить начальные настройки centos, которые я выполняю на свежеустановленном сервере.
2. Показать примеры конфигураций, которые я использую в типовой настройке.
3. Дать советы по настройке centos на основе своего опыта работы с системой.
4. Привести список типовых программ и утилит, которые помогают администрировать сервер.

Данная статья является частью единого цикла статьей про сервер Centos.

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 после установки.

После выхода нового релиза Centos 8, стало тяжело в единой статье описывать начальную настройку обоих серверов, но мне не захотелось разделять статью, так как на нее идет много входящих ссылок из разных мест. Удобнее поддерживать общий материал по обоим релизам, чем я и займусь. Заодно наглядно будут видны отличия двух версий, которые пару лет после выхода centos 8 будут актуальны обе и использовать придется и ту, и другую версию, в зависимости от ситуации.

В Centos 7 используется пакетный менеджер **yum**, а в Centos 8 — **dnf**. При этом оставили символьную ссылку с yum на dnf, так что можно писать как первое название, так и второе. Для единообразия я везде буду использовать yum, а вас предупреждаю, просто чтобы вы понимали, почему я делаю именно так. Реально в CentOS 8 используется dnf, это другой, более современный пакетный менеджер, которые позволяет работать с разными версиями одного и того же софта. Для этого используются отдельные репозитории, которые появились для centos 8.

Начальная настройка CentOS

Лично я любую настройку системы, будь то centos или другая, после установки начинаю с того, что полностью обновляю систему. Если установочный образ был свежий, либо установка велась по сети, то скорее всего обновлений никаких не будет. Чаще всего они есть, так как установочные образы не всегда

регулярно обновляются.

Обновляем систему

```
# yum update
```

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

```
# yum install mc
```

```
[root@centos8 ~]# yum install mc
Last metadata expiration check: 0:28:00 ago on Sat 05 Oct 2019 08:13:36 PM MSK.
Dependencies resolved.
================================================================================
 Package                 Arch          Version                 Repository    Size
================================================================================
Installing:
 mc                      x86_64        1:4.8.19-9.el8          AppStream    1.9 M
Installing dependencies:
 perl-Carp               noarch        1.42-396.el8            BaseOS        30 k
 perl-Errno              x86_64        1.28-416.el8            BaseOS        76 k
 perl-Exporter           noarch        5.72-396.el8            BaseOS        34 k
 perl-File-Path          noarch        2.15-2.el8              BaseOS        38 k
 perl-File-Temp          noarch        0.230.600-1.el8         BaseOS        63 k
 perl-IO                 x86_64        1.38-416.el8            BaseOS       141 k
 perl-PathTools          x86_64        3.74-1.el8              BaseOS        90 k
 perl-Scalar-List-Utils  x86_64        3:1.49-2.el8            BaseOS        68 k
 perl-Socket             x86_64        4:2.027-2.el8           BaseOS        59 k
 perl-Text-Tabs+Wrap     noarch        2013.0523-395.el8       BaseOS        24 k
 perl-Unicode-Normalize  x86_64        1.25-396.el8            BaseOS        82 k
 perl-constant           noarch        1.33-396.el8            BaseOS        25 k
 perl-interpreter        x86_64        4:5.26.3-416.el8        BaseOS       6.3 M
 perl-libs               x86_64        4:5.26.3-416.el8        BaseOS       1.6 M
 perl-macros             x86_64        4:5.26.3-416.el8        BaseOS        72 k
 perl-parent             noarch        1:0.237-1.el8           BaseOS        20 k
 perl-threads            x86_64        1:2.21-2.el8            BaseOS        61 k
 perl-threads-shared     x86_64        1.58-2.el8              BaseOS        48 k

Transaction Summary
================================================================================
Install  19 Packages

Total download size: 11 M
Installed size: 28 M
Is this ok [y/N]:
```


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 зависимости от набора начальных пакетов, которые вы выбираете при установке системы, у вас будет тот или иной набор сетевых утилит. Вот список тех, к которым привык лично я — ifconfig, netstat, nslookup и некоторые другие. Если она вам нужны, так же как и мне, то предлагаю их установить отдельно, если они еще не стоят. Если вам они особо не нужны и вы ими не пользуетесь, то можете пропустить их установку. Проверим, что у нас имеется в системе на текущий момент

```
# ifconfig
```

Если увидите ответ:

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

Значит утилита не установлена. Вместо ifconfig в CentOS теперь утилита **ip**. Это относится не только к центос. Такая картина почти во всех популярных современных дистрибутивах Linux. Я с давних времен привык к ifconfig, хотя последнее время практически не пользуюсь. 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 не удобно. Хотя сейчас это уже не очень актуально, так как с Freebsd больше не работаю, а утилита ip есть во всех дистрибутивах linux. Тем не менее, если вам нужен ifconfig, то можете установить пакет **net-tools**, в который она входит:

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

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, можно перезагрузиться:

```
# reboot
```

А если хотите без перезагрузки применить отключение SELinux, то выполните команду:

```
# setenforce 0
```

Постоянно получаю очень много критики на тему отключения SELinux. Я знаю, как он работает, умею его настраивать. Это реально не очень сложно и освоить не трудно. Это мой осознанный выбор, хотя иногда я его настраиваю. Мой формат работы с системой таков, что SELinux мне чаще всего не нужен, поэтому я не трачу на него время и в базовой настройке centos отключаю. Безопасность системы — комплексная работа, особенно в современном мире web разработки, где правят бал микросервисы и контейнеры. SELinux нишевый инструмент, которые нужен не всегда и не везде. Поэтому в данном статье ему не место. Кому нужно, будет отдельно включать SELinux и настраивать.

Указываем сетевые параметры

Продолжаем базовую настройку centos после установки. Теперь произведем настройку сети, если по какой-то причине не сделали это во время установки, либо если вам надо их изменить. В общем случае, сеть в Centos настраивается с помощью **NetworkManager** и его консольной утилиты **nmtui**. Она идет в

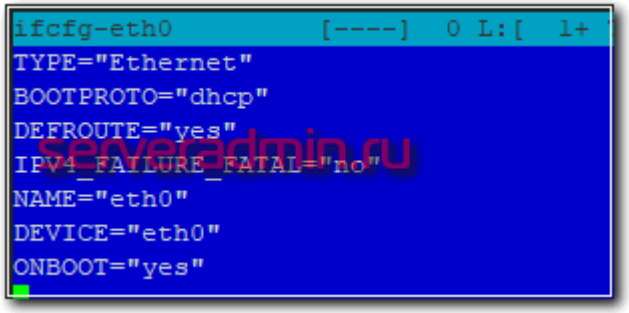
базовой устновке системы. Там простой и понятный графический интерфейс, так что рассказывать нечего. Я больше привык настраивать сеть через конфигурационные файлы network-scripts. В centos 7-й версии они есть из коробки, в 8-й версии их убрали. Чтобы воспользоваться ими для настройки сети, надо отдельно установить пакет **network-scripts**.

```
# yum install network-scripts
```

Теперь можно выполнить настройку сети. Для этого открываем файл */etc/sysconfig/network-scripts/ifcfg-eth0*

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

Если вы получаете сетевые настройки по dhcp, то минимальный набор настроек в конфигурационном файле будет такой.



```
TYPE="Ethernet"
BOOTPROTO="dhcp"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
NAME="eth0"
DEVICE="eth0"
ONBOOT="yes"
```

Для настройки статического ip адреса настройки будут следующие.

```
TYPE="Ethernet"
BOOTPROTO="none"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
NAME="eth0"
DEVICE="eth0"
ONBOOT="yes"
IPADDR=192.168.167.117
DNS1=192.168.167.113
PREFIX=28
GATEWAY=192.168.167.113
```

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

```
# systemctl restart network
```

Настраиваем firewall

Очень подробно вопрос настройки iptables в CentOS я рассмотрел отдельно. Сейчас мы быстро и просто настроим firewall. В CentOS 7 в качестве базового

фаервола выступает **iptables**. По-умолчанию он запущен. Чтобы посмотреть текущие правила, нужно ввести команду:

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

В Centos 8 вместо iptables используется **nftables**. Сразу скажу, что я с ним еще не разбирался и не знаю, буду ли. Реально, меня полностью устраивают iptables. Я ни разу не сталкивался с тем, что что-то не получалось или было невозможно настроить с их помощью. У меня написана куча конфигов к ним, настроены роли ansible с шаблонами правил. Я не понимаю, зачем мне все это переводить в nftables и какую выгоду я с этого получу. Поэтому пока я продолжаю использовать iptables.

Начиная с 7-й версии CentOS для настройки firewall разработан новый инструмент под названием firewalld и все управление производится через него. Он же используется и в Centos 8, только управляет при этом nftables, но сами правила для firewalld одинаковые в обоих версиях. Я не понял зачем в принципе придумали firewalld 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 либо шаблон ansible с правилами.

На тему отключения firewalld я тоже время от времени получаю критику, типа надо изучать новое и не цепляться за старое. Так вот, firewalld я изучил и пользуюсь относительно регулярно. Он долгое время использовался в Bitrixenv, с которым я работаю давно и плотно. Кстати, в последних версиях этого окружения bitrix отказался от firewalld. Удивился, когда это заметил. В связи с этим, firewalld мне приходится знать и использовать. И мне он нравится меньше, чем нативные правила в iptables и мои скрипты для управления ими.

Свое мнение не навязываю и убеждать никого не хочу. Делюсь в статье про настройку centos тем, что использую сам. А вы выбирайте тот инструмент, что вам больше подходит. 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.

> 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 сервера.

```
# 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 install iptables-services
```

Включим автозапуск iptables:

```
# systemctl enable iptables
```

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

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

# Интерфейс который смотрит в интернет
export WAN=eth0
export WAN_IP=147.15.218.72

# 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 http
$IPT -A INPUT -i $WAN -p tcp --dport 80 -j ACCEPT
# Открываем порт для https
$IPT -A INPUT -i $WAN -p tcp --dport 443 -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 смотрите в отдельной статье, ссылку на которую я уже приводил в самом начале раздела.

Делаем файл с правилами исполняемым и запускаем:

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

Продолжаем настраивать centos. Внесем некоторые изменения в работу ssh для небольшого увеличения безопасности. Хотя речь стоит вести больше не о безопасности, а об удобстве и эффективности. По-умолчанию, сервис ssh 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. Это не доставляет в реальности каких-то серьезных хлопот и тем не менее, подобные запросы забивают лог secure и трятят некоторые ресурсы сервера как миниум на установку соединения и рукопожатия (handshake).

Чтобы немного закрыть 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.

```
Port 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.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 не получится.

Перезапускаем sshd:

```
# systemctl restart sshd
```

Кстати, если вы не отключили SELinux, то просто так не сможете сменить порт ssh. Получите ошибку при перезапуске.

```
SELinux is preventing /usr/sbin/sshd from name_bind access on the tcp_socket port 25333.
```

Наглядный пример того, как работает защита SELinux. Если у вас кто-то заберется на сервер через какую-то уязвимость и захочет открыть отдельный ssh серввер на каком-то нестандартном порту, у него ничего не получится.

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. Настройку службы sshd в centos закончили, двигаемся дальше.

Настраиваем время

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

```
# date
```

Чтобы сменить часовой пояс, можете воспользоваться специальной утилитой, которая входит в состав systemd.

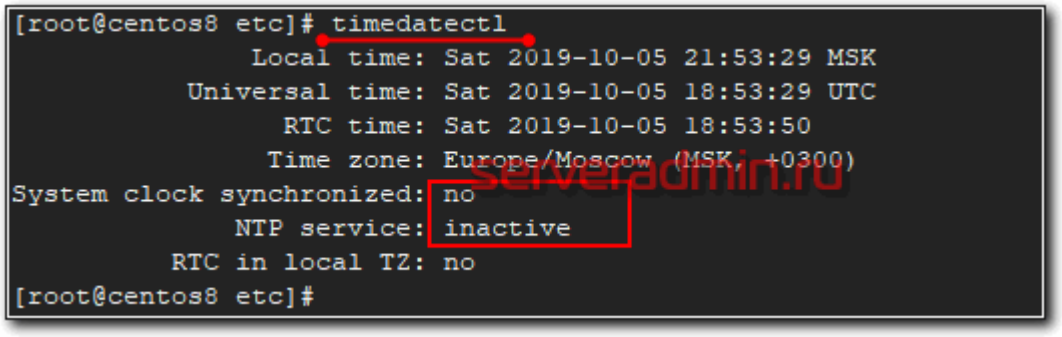
```
# timedatectl set-timezone Europe/Moscow
```

По факту, эта утилита меняет символьную ссылку. Рассказываю, чтобы вы понимали, как реально меняется часовой пояс.

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

Если вы вручную замените символьную ссылку на другую, точно так же поменяете часовой пояс системы.

Теперь проверим статус службы по обновлению времени через интернет. Для этого можно использовать указанную выше команду **timedatectl** без параметров.



В CentOS 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, как и у меня, что видно по скриншоту, то устанровим и настроим вручную:

```
# yum install chrony
```

Запускаем chrony и добавляем в автозагрузку:

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

Проверяем, нормально ли запустился:

```
# systemctl status chronyd
```

```
[root@centos8 etc]# systemctl start chronyd
[root@centos8 etc]# systemctl enable chronyd
Created symlink /etc/systemd/system/multi-user.target.wants/chronyd.service → /usr/lib/systemd/system/chronyd.service.
[root@centos8 etc]# systemctl status chronyd
● chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2019-10-05 21:55:39 MSK; 34s ago
     Docs: man:chronyd(8)
           man:chrony.conf(5)
 Main PID: 9399 (chronyd)
    Tasks: 1 (limit: 4599)
   Memory: 1.1M
   CGroup: /system.slice/chronyd.service
           └─9399 /usr/sbin/chronyd

Oct 05 21:55:39 centos8 systemd[1]: Starting NTP client/server...
Oct 05 21:55:39 centos8 chronyd[9399]: chronyd version 3.3 starting (+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SIGND +ASYNCDNS +SECHASH +IPV6 +DEBU>
Oct 05 21:55:39 centos8 chronyd[9399]: Using right/UTC timezone to obtain leap second data
Oct 05 21:55:39 centos8 systemd[1]: Started NTP client/server.
Oct 05 21:55:45 centos8 chronyd[9399]: Selected source 37.235.209.151
Oct 05 21:55:45 centos8 chronyd[9399]: System clock TAI offset set to 37 seconds
Oct 05 21:55:45 centos8 chronyd[9399]: System clock wrong by 26.804655 seconds, adjustment started
Oct 05 21:56:11 centos8 chronyd[9399]: System clock was stepped by 26.804655 seconds
Oct 05 21:56:12 centos8 chronyd[9399]: Selected source 128.0.142.251
[root@centos8 etc]# timedatectl
               Local time: Sat 2019-10-05 21:56:26 MSK
           Universal time: Sat 2019-10-05 18:56:26 UTC
                 RTC time: Sat 2019-10-05 18:56:26
                Time zone: Europe/Moscow (MSK, +0300)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
[root@centos8 etc]#
```


Все в порядке, сервис настроен и работает. После запуска он автоматически синхронизирует время. Теперь наши часы будут автоматически синхронизироваться с сервером времени в интернете.

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

Добавление репозиториев

При настройке centos частенько нужен софт, которого нет в стандартной репе. Для инсталляции дополнительных пакетов необходимо подключить репозитории в CentOS. Наиболее популярный это EPEL. Раньше был rpmforge, но уже как несколько лет закрыт. Про него все позабыли. Подключаем репозиторий EPEL. С ним все просто, он добавляется из стандартной репы:

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

Так же для CentOS 7 крайне полезен репозиторий REMI, который позволяет установить больее свежие версии php, в отличие от тех, что есть в стандартном репозитории. Напомню, что это версия php 5.4, которая уже никуда не годится и снята с поддержки.

```
# rpm -Uhv http://rpms.remirepo.net/enterprise/remi-release-7.rpm
```

Для Centos 8 remi пока не актуален, но думаю, что это временно. В принципе, мне этих двух репозиториев в centos обычно хватает в общем случае. Другие подключаются уже под конкретные нужды для установки различного софта.

Настройка хранения истории в bash_history

Двигаемся дальше по настройке системы centos на сервере. 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. 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
```

По настройке хранения истории команд все. В файле .bashrc можно много чего настроить интересного. Я одно время увлекался и экспериментировал, но потом все забросил, так как не имеет смысла. Работая с серверами заказчиков я чаще всего вижу дефолтный bash, поэтому лучше привыкать и работать именно в нем. А отдельные настройки и украшательства это удел личных компьютеров и серверов. Не рабочих. Так что больше я ничего не настраивать по стандарту в centos сервере в этом плане.

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 настроить автоматическое выполнение. Не обязательно именно устанавливать обновления автоматически, но как минимум проверять их появление. Я обычно придерживаюсь такой стратегии.

Yum-cron

Для автоматической проверки обновлений в Centos 7 нам поможет утилита **yum-cron**. Ставится она традиционно через yum из стандартного репозитория.

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

После установки yum-cron 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.

Настраивать yum-cron можно через, конфигурационные файлы, которые 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.

Dnf-automatic

Как я уже говорил ранее, в Centos 8 используется другой пакетный менеджер — dnf. Настройка обновления пакетов там выполняется через утилиту **dnf-automatic**. Поставим ее и настроим.

```
# yum install dnf-automatic
```

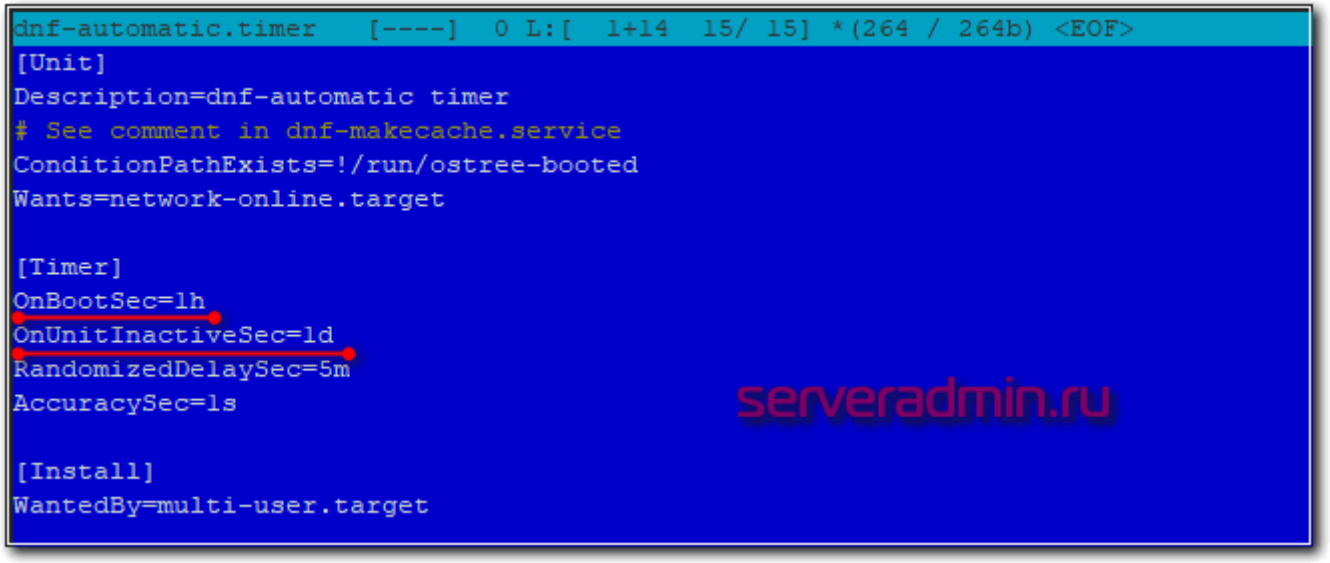
Управлением запуском по расписанию занимается уже не cron, а systemd своим встроенным планировщиком. Посмотреть таймеры автоматического запуска можно командой:

```
# systemctl list-timers *dnf-*
```

Если там нет ни одного задания, то добавить таймер можно вручную:

```
# systemctl enable --now dnf-automatic.timer
```

Дефолтный таймер настроен на запуск dnf-automatic через час после загрузки сервера и ежедневное повторение. Конфиг таймера живет тут — */etc/systemd/system/multi-user.target.wants/dnf-automatic.timer*.



Конфиг для dnf-automatic живет в */etc/dnf/automatic.conf*. По-умолчанию он только скачивает обновления, но не применят их. Конфиг хорошо прокомментирован, так что можете его настроить так, как пожелаете. Отдельных пояснений не требуется. Настраивайте обновление пакетов системы на свое усмотрение. Как я уже сказал, автоматически только качаю их. Установку всегда держу под контролем с ручным управлением.

Отключаем флуд сообщений в /var/log/messages

Продолжая настройку centos, исправим одно небольшое неудобство. В дефолтной установке системы 7-й версии, 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.
```

В Centos 8 я их не заметил, поэтому там ничего делать не надо. Никакой практической пользы сообщения 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

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

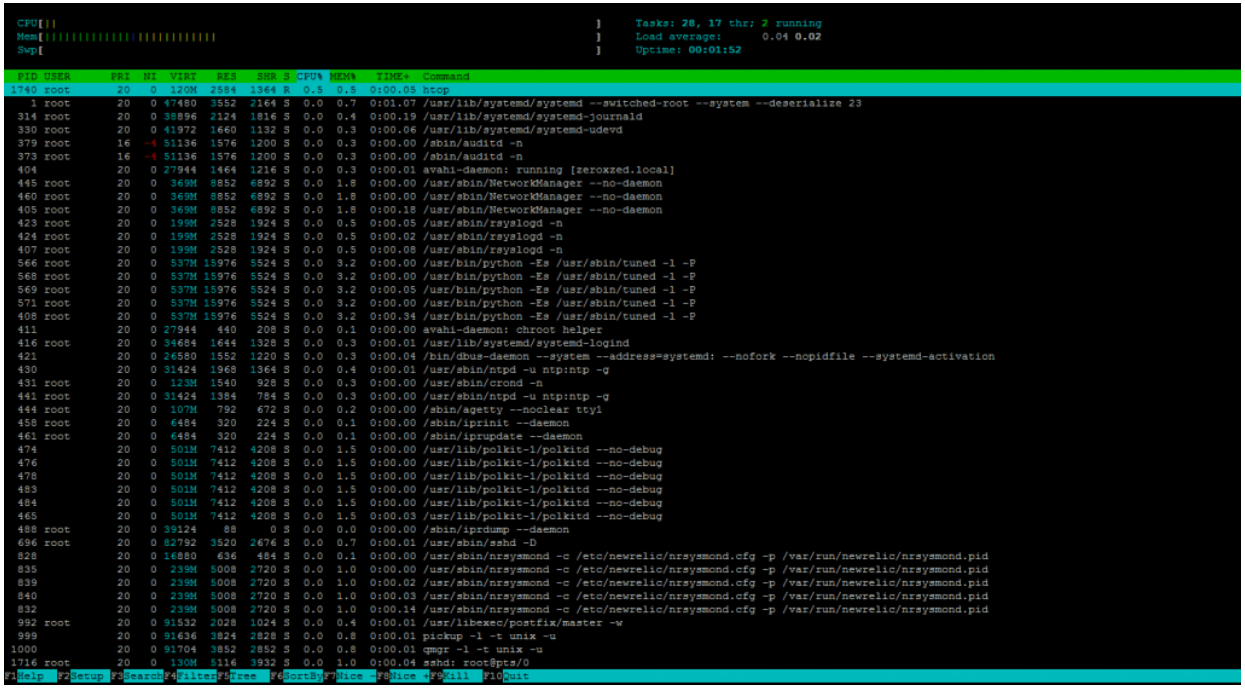
iftop показывает в режиме реального времени загрузку сетевого интерфейса, может запускаться с различными ключами, не буду останавливаться на этом подробно, в интернете есть информация на эту тему. Ставим:

```
# 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 install htop
# yum install atop
```

Вот как выглядит htop:



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 закончена, можно приступать к установке и настройке основного функционала.

Настройка системной почты

В завершение настройки сервера CentOS 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 postfix
```

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 = centos-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 = centos-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. Если вы будете использовать ящик от Яндекса, то скорее всего получите ошибку в логе почтового сервера и письмо

не будет отправлено.

```
relay=smtp.yandex.ru[77.88.21.158]:25, delay=0.25, delays=0/0/0.24/0.01, dsn=5.7.1, status=bounced (host
smtp.yandex.ru[77.88.21.158] said: 553 5.7.1 Sender address rejected: not owned by auth user. (in reply to MAIL FROM
command))
```

Ошибка эта означает то, что у вас в качестве отправителя почты указан не тот же ящик, что вы используете для авторизации. Как это исправить, я рассказываю в отдельной статье — как изменить адрес отправителя в postfix. С другими почтовыми системами, где нет подобной проверки, все должно быть нормально и так.

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 полноценный 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, которые я обычно делаю при подготовке сервера сразу после установки. 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.

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

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

Помогла статья? Есть возможность отблагодарить автора