```
[root@centos8 yum.repos.d]# dnf install mariadb mariadb-server
Last metadata expiration check: 0:24:53 ago on Tue 08 Oct 2019 06:07:26 PM MSK.
Dependencies resolved.
================================================================================
 Package                  Arch       Version                        Repository
                                                                             Size
================================================================================
Installing:
 mariadb                  x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream
                                                                            6.2 M
 mariadb-server           x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream
                                                                             16 M
Installing dependencies:
 mariadb-common           x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream
                                                                             62 k
 mariadb-connector-c      x86_64     3.0.7-1.el8                    AppStream
                                                                            148 k
 mariadb-connector-c-config noarch   3.0.7-1.el8                    AppStream
                                                                             13 k
 mariadb-errmsg           x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream
                                                                            232 k
 perl-DBD-MySQL           x86_64     4.046-2.module_el8.0.0+72+668237d8 AppStream
                                                                            156 k
 perl-DBI                 x86_64     1.641-2.module_el8.0.0+66+feleca09 AppStream
                                                                            740 k
 perl-Digest              noarch     1.17-395.el8                   AppStream
                                                                             27 k
 perl-Digest-MD5          x86_64     2.55-396.el8                   AppStream
                                                                             37 k
 perl-Net-SSLeay          x86_64     1.85-6.el8                     AppStream
                                                                            358 k
 perl-URI                 noarch     1.73-3.el8                     AppStream
                                                                            116 k
 perl-libnet              noarch     3.11-3.el8                     AppStream
                                                                            121 k
 perl-Data-Dumper         x86_64     2.167-399.el8                  BaseOS
                                                                             58 k
 perl-Encode              x86_64     4:2.97-3.el8                   BaseOS
                                                                            1.5 M
 perl-Getopt-Long         noarch     1:2.50-4.el8                   BaseOS
                                                                             63 k
 perl-HTTP-Tiny           noarch     0.074-1.el8                    BaseOS
                                                                             58 k
 perl-MIME-Base64         x86_64     3.15-396.el8                   BaseOS
                                                                             31 k
 perl-Math-BigInt         noarch     1:1.9998.11-5.el8              BaseOS
                                                                            195 k
 perl-Math-Complex        noarch     1.59-416.el8                   BaseOS
                                                                            108 k
 perl-Pod-Escapes         noarch     1:1.07-395.el8                 BaseOS
                                                                             20 k
 perl-Pod-Perldoc         noarch     3.28-396.el8                   BaseOS
                                                                             86 k
 perl-Pod-Simple          noarch     1:3.35-395.el8                 BaseOS
                                                                            213 k
 perl-Pod-Usage           noarch     4:1.69-395.el8                 BaseOS
                                                                             34 k
 perl-Storable            x86_64     1:3.11-3.el8                   BaseOS
                                                                             98 k
 perl-Term-ANSIColor      noarch     4.06-396.el8                   BaseOS
                                                                             46 k
 perl-Term-Cap            noarch     1.17-395.el8                   BaseOS
                                                                             23 k
 perl-Text-ParseWords     noarch     3.30-395.el8                   BaseOS
                                                                             18 k
 perl-Time-Local          noarch     1:1.280-1.el8                  BaseOS
                                                                             34 k
 perl-podlators           noarch     4.11-1.el8                     BaseOS
                                                                            118 k
 psmisc                   x86_64     23.1-3.el8                     BaseOS
                                                                            151 k
```


В связи с выходом нового релиза популярной операционной системы пришло время актуализировать некоторые статьи. Сегодня я настрою

производительный веб сервер CentOS 8 на свежих версиях nginx, php-fpm, где сам php версии 7.2. Сейчас пока нет необходимости использовать сторонние репозитории для php, так как в стандартных присутствует свежая и актуальная версия.

Содержание:

Цели статьи

1. Описать общий принцип настройки web сервера на базе nginx.
2. Показать, как установить актуальные версии nginx, php-fpm и mariadb.
3. Привести примеры своих конфигов для перечисленных сервисов.
4. Настроить бесплатные ssl сертификаты let’s encrypt для сайтов.
5. Показать, как настроить selinux для web сервера nginx.
6. Записать и показать видео настройки веб сервера.

Данная статья является частью единого цикла статьей про сервер Centos.

Введение

В этой статье я расскажу, как настроить производительный web сервер на базе популярного стека технологий — nginx и php-fpm. В связи с выходом нового релиза Centos 8, многие статьи на эту тему стали не актуальны, так как версии софта в базовых репозиториях обновились и тот же php нет смысла ставить из стороннего репозитория.

Работать будем на сервере под управлением CentOS 8. Если у вас его еще нет, то читайте мои статьи на тему установки и базовой настройки centos. Не забудьте уделить внимание теме настройки iptables. В данной статье я ее не буду касаться, хотя тема важная для web сервера.

В самом конце я покажу, как настроить SELinux применительно к данному веб серверу. Приведу список правил, которые нужно будет применить конкретно к моей статье. Правила не универсальные. В зависимости от настройки web сервера, они будут отличаться. Если вам не хочется тратить на это свое время, или вам он не нужен, то просто отключите selinux. Если же нужен, то пока только добавляйте конфиги, но ничего не запускайте, пока не дойдете до самого конца. С включенным и не настроенным SELinux все равно ничего не заработает.

В своей тестовой среде я буду использовать следующие сущности.

z.serveradmin.ru	имя тестового виртуального хоста и сайта
/web/sites	директория для размещения виртуальных хостов
75.37.225.138	внешний ip адрес сервера
pma.serveradmin.ru	имя виртуального хоста для phpmyadmin

Подопытным сервером будет выступать виртуальная машина от ihor, характеристики следующие:

Процессор	4 ядра
Память	4 Gb
Диск	80 Gb SSD

Это кастомная настройка параметров. Они не оптимальны по цене, но мне были нужны именно такие.

Установка nginx на CentOS 8

Для установки самой свежей стабильной версии nginx на centos подключим официальный репозиторий. Я использую именно родной репозиторий по следующей причине. Мне больше нравится начальное расположение и формат конфигурационных файлов. Устанавливая софт из официальных репозиториев ты помимо того, что имеешь самую свежую и актуальную версию nginx, так и расположение и формат конфигурационных файлов будет одинаковый во всех системах.

Банальный пример. Если ставить nginx из официального репозитория centos 8, настройка дефолтного виртуального хоста будет прямо в основном конфиге nginx.conf. Мне лично это не удобно, так как в этом файле я храню набор настроек без привязки к виртуальным хостам. Если установить nginx из официального репозитория, конфигурация дефолтного хоста будет в отдельном конфиге default.conf.

Подключаем репозиторий nginx в centos 8. Я обычно использую mainline версию. Она имеет все нововведения на борту и достаточно стабильна. По крайней мере у меня никогда проблем с ней не было. Если вы хотите стабильную версию, то используйте репозиторий stable. Рисуем конфиг репозитория */etc/yum.repos.d/nginx.repo.*

```
[nginx-mainline]
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
```

```
module_hotfixes=true
```

Устанавливаем nginx на сервер.

```
# dnf install nginx
```

```
[root@centos8 yum.repos.d]# dnf install nginx
Last metadata expiration check: 0:00:20 ago on Mon 07 Oct 2019 03:43:34 PM MSK.
Dependencies resolved.
================================================================================
 Package                 Arch                 Version                 Repository                 Size
================================================================================
Installing:
 nginx                   x86_64               1:1.17.4-1.el8.ngx      nginx-mainline             798 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 798 k
Installed size: 2.9 M
Is this ok [y/N]:
```


Запускаем nginx и добавляем в автозагрузку.

```
# systemctl start nginx
# systemctl enable nginx
```

Проверяем, запустился ли web сервер. Для этого идем по ссылке *http://75.37.225.138/*. Вы должны увидеть стандартную страницу заглушку.

Если ее не видите, скорее всего у вас включен и не настроен firewalld. В начале статьи я приводил ссылку на статью с настройкой сервера и настройкой iptables. Если вы с ними познакомились, значит либо уже открыли нужные порты, либо сейчас это сделаете. Для тех, кто не хочет со всем этим разбираться, я просто покажу, как открыть порты 80 и 443 на firewalld, который используется в centos 8 по-умолчанию.

```
# firewall-cmd — permanent — zone=public — add-service=http
# firewall-cmd — permanent — zone=public — add-service=https
# firewall-cmd — reload
```

Проверить, открылись ли порты можно командой.

```
# firewall-cmd — list-all
```

```
[root@centos8 yum.repos.d]# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: ens18
  sources:
  services: cockpit dhcpv6-client http https ssh
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:

[root@centos8 yum.repos.d]#
```


Теперь выполним небольшую начальную настройку. Очень подробно вопрос настройки nginx я разбирал отдельно, рекомендую познакомиться со статьей. Там описано все то, что используется далее в конфигах. Здесь же я их просто привожу, без комментариев и объяснений. Начнем с основного конфига */etc/nginx/nginx.conf*.

```
user  nginx;
worker_processes auto;
worker_cpu_affinity auto;
worker_rlimit_nofile 30000;
pid /var/run/nginx.pid;
pcre_jit on;

events {
    worker_connections 8192;
    multi_accept on;
}

http {

    # Basic ######################
    sendfile                    on;
    tcp_nopush                  on;
    tcp_nodelay                 on;
    reset_timedout_connection   on;
    keepalive_timeout           120;
    keepalive_requests          1000;
    types_hash_max_size         2048;
    server_tokens               off;
    send_timeout                30;
    client_body_timeout         30;
    client_header_timeout       30;
    server_names_hash_max_size  4096;

    # Limits #####################
    client_max_body_size        10m;
    client_body_buffer_size     128k;
    client_body_temp_path       /var/cache/nginx/client_temp;
```

```
    proxy_connect_timeout       60;
    proxy_send_timeout          60;
    proxy_read_timeout          60;
    proxy_buffer_size           4k;
    proxy_buffers               8 16k;
    proxy_busy_buffers_size     64k;
    proxy_temp_file_write_size  64k;
    proxy_temp_path             /var/cache/nginx/proxy_temp;

    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    # Logs ########################

    log_format  main    '$remote_addr - $host [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"'
        'rt=$request_time ut=$upstream_response_time '
        'cs=$upstream_cache_status';
    log_format full     '$remote_addr - $host [$time_local] "$request" '
                'request_length=$request_length '
                'status=$status bytes_sent=$bytes_sent '
                'body_bytes_sent=$body_bytes_sent '
                'referer=$http_referer '
                'user_agent="$http_user_agent" '
                'upstream_status=$upstream_status '
                'request_time=$request_time '
                'upstream_response_time=$upstream_response_time '
                'upstream_connect_time=$upstream_connect_time '
                'upstream_header_time=$upstream_header_time';

    access_log  /var/log/nginx/access.log  main;
    error_log  /var/log/nginx/error.log;
```

```
    # Gzip ########################

    gzip on;
    gzip_static on;
    gzip_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss
text/javascript application/javascript image/x-icon image/svg+xml application/x-font-ttf;
    gzip_comp_level 9;
    gzip_proxied any;
    gzip_min_length 1000;
    gzip_disable "msie6";
    gzip_vary on;

    etag off;

    # Cache ########################

    #proxy_cache_valid 1m;
    #proxy_cache_key $scheme$proxy_host$request_uri$cookie_US;
    #proxy_cache_path /web/sites/nginx_cache levels=1:2 keys_zone=main:1000m;

    # Zone limits ################

    limit_conn_zone $binary_remote_addr zone=perip:10m;
    limit_req_zone $binary_remote_addr zone=lim_5r:10m rate=5r/s; # lim for dynamic page
    limit_req_zone $binary_remote_addr zone=lim_1r:10m rate=1r/s; # lim for search page
    limit_req_zone $binary_remote_addr zone=lim_10r:10m rate=10r/s;

    # SSL ##########################

    ssl_session_cache shared:SSL:50m;
    ssl_session_timeout 1d;
    ssl_session_tickets on;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
```

```
    ssl_ciphers 'TLS13-CHACHA20-POLY1305-SHA256:TLS13-AES-128-GCM-SHA256:TLS13-AES-256-GCM-SHA384:ECDHE:ECDHE-ECDSA-
CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-
SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-
SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-
RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-
SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
    ssl_prefer_server_ciphers on;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;
    ssl_stapling on;
    ssl_stapling_verify on;
    add_header Strict-Transport-Security max-age=15768000;
    resolver 8.8.8.8;

    include /etc/nginx/conf.d/*.conf;

    # For monitoring ############
    server {
    listen 127.0.0.1:80;
    server_name status.localhost;
    keepalive_timeout   0;
    allow       127.0.0.1;
    deny        all;
    access_log  off;

        location /server-status {
    stub_status on;
        }

        location /status {
    access_log off;
    allow 127.0.0.1;
    deny all;
```

```
    include fastcgi_params;
    fastcgi_pass   unix:/run/php-fpm/www.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        }
    }
}
```

Сохраните конфиг и проверьте его корректность командой:

```
# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Если ошибок нет, то можно применить.

```
# nginx -s reload
```

Теперь создадим структуру каталогов для сайтов. Я обычно все сайты размещаю отдельно, например в */web/sites*. Создадим там два сайта — основной и панель phpmyadmin для него. Привожу просто для примера, вешать на отдельный виртуальный хост phpmyadmin не обязательно, хотя в некоторых случаях это бывает удобно.

```
# mkdir -p /web/sites/z.serveradmin.ru/{www,log}
# mkdir -p /web/sites/pma.serveradmin.ru/{www,log}
# chown -R nginx. /web/sites/
```

Создадим конфиги nginx для этих виртуальных хостов. Я сразу буду делать их с учетом https, который мы настроим позже. Так что после создания не надо перезапускать веб сервер и проверять работу — будут ошибки. Виртуальный хост сайта показан на примере **wordpress**. Конфигурация собрана на основе

рекомендаций из официальной документации конкретно для веб сервера nginx.

Конфиг для основного сайта — /etc/nginx/conf.d/z.serveradmin.ru.conf

```
server {
    listen 443 ssl http2;
    server_name z.serveradmin.ru;
    root /web/sites/z.serveradmin.ru/www/;
    index index.php index.html index.htm;
    access_log /web/sites/z.serveradmin.ru/log/access.log main;
    error_log /web/sites/z.serveradmin.ru/log/error.log;

    ssl_certificate             /etc/letsencrypt/live/z.serveradmin.ru/fullchain.pem;
    ssl_certificate_key         /etc/letsencrypt/live/z.serveradmin.ru/privkey.pem;

    location / {
    try_files $uri $uri/ /index.php?$args;
    }

    location ~* ^.+.(js|css|png|jpg|jpeg|gif|ico|woff)$ {
    access_log off;
    expires max;
    }

    location ~ \.php$ {
    try_files  $uri =404;
    fastcgi_pass   unix:/run/php-fpm/www.sock;
    fastcgi_index index.php;
    fastcgi_param DOCUMENT_ROOT /web/sites/z.serveradmin.ru/www/;
    fastcgi_param SCRIPT_FILENAME /web/sites/z.serveradmin.ru/www$fastcgi_script_name;
    fastcgi_param PATH_TRANSLATED /web/sites/z.serveradmin.ru/www$fastcgi_script_name;
    include fastcgi_params;
```

```
    fastcgi_param QUERY_STRING $query_string;
    fastcgi_param REQUEST_METHOD $request_method;
    fastcgi_param CONTENT_TYPE $content_type;
    fastcgi_param CONTENT_LENGTH $content_length;
    fastcgi_param HTTPS on;
    fastcgi_intercept_errors on;
    fastcgi_ignore_client_abort off;
    fastcgi_connect_timeout 60;
    fastcgi_send_timeout 180;
    fastcgi_read_timeout 180;
    fastcgi_buffer_size 128k;
    fastcgi_buffers 4 256k;
    fastcgi_busy_buffers_size 256k;
    fastcgi_temp_file_write_size 256k;
    }

    location = /favicon.ico {
    log_not_found off;
    access_log off;
    }

    location = /robots.txt {
    allow all;
    log_not_found off;
    access_log off;
    }

    location ~ /\.ht {
    deny all;
    }
}

server {
```

```
     listen  443 ssl http2;
     server_name  www.z.serveradmin.ru;
     return 301 https://z.serveradmin.ru$request_uri;
}

server {
    listen 80;
    server_name z.serveradmin.ru;
    root /web/sites/z.serveradmin.ru/www/;
    index index.php index.html index.htm;
    access_log /web/sites/z.serveradmin.ru/log/access.log main;
    error_log /web/sites/z.serveradmin.ru/log/error.log;

    location / {
    return 301 https://z.serveradmin.ru$request_uri;
    }
}

server {
     listen  80;
     server_name  www.z.serveradmin.ru;
     return 301 http://z.serveradmin.ru$request_uri;
}
```

В данной конфигурации настроены все необходимые редиректы, при этом отключен редирект файла robots.txt. Он отдельно отдается по http и https. Для phpmyadmin рисуем конфиг попроще. Я сразу его закрываю отдельным паролем с помощью basic auth.

```
# mcedit /etc/nginx/conf.d/pma.serveradmin.ru.conf
```

```
server {
    listen 443 ssl http2;
    server_name pma.serveradmin.ru;
    root /web/sites/pma.serveradmin.ru/www/;
    index index.php index.html index.htm;
    access_log /web/sites/pma.serveradmin.ru/log/access.log main;
    error_log /web/sites/pma.serveradmin.ru/log/error.log;

    auth_basic "Restricted Content";
    auth_basic_user_file /etc/nginx/.htpasswd;

    ssl_certificate             /etc/letsencrypt/live/pma.serveradmin.ru/fullchain.pem;
    ssl_certificate_key         /etc/letsencrypt/live/pma.serveradmin.ru/privkey.pem;

    location ~ \.php$ {
    fastcgi_pass   unix:/run/php-fpm/www.sock;
    fastcgi_index index.php;
    fastcgi_param DOCUMENT_ROOT /web/sites/pma.serveradmin.ru/www/;
    fastcgi_param SCRIPT_FILENAME /web/sites/pma.serveradmin.ru/www$fastcgi_script_name;
    fastcgi_param PATH_TRANSLATED /web/sites/pma.serveradmin.ru/www$fastcgi_script_name;
    include fastcgi_params;
    fastcgi_param QUERY_STRING $query_string;
    fastcgi_param REQUEST_METHOD $request_method;
    fastcgi_param CONTENT_TYPE $content_type;
    fastcgi_param CONTENT_LENGTH $content_length;
    fastcgi_intercept_errors on;
    fastcgi_ignore_client_abort off;
    fastcgi_connect_timeout 60;
    fastcgi_send_timeout 180;
    fastcgi_read_timeout 180;
    fastcgi_buffer_size 128k;
    fastcgi_buffers 4 256k;
```

```
    fastcgi_busy_buffers_size 256k;
    fastcgi_temp_file_write_size 256k;
    }
}

server {
    listen 80;
    server_name pma.serveradmin.ru;
    root /web/sites/pma.serveradmin.ru/www/;
    index index.php index.html index.htm;
    access_log /web/sites/pma.serveradmin.ru/log/access.log main;
    error_log /web/sites/pma.serveradmin.ru/log/error.log;

    location / {
    return 301 https://pma.serveradmin.ru$request_uri;
    }
}
```

Создаем файл с логином и паролем для доступа.

```
# sh -c "echo -n 'pma:' >> /etc/nginx/.htpasswd"
# sh -c "openssl passwd -apr1 >> /etc/nginx/.htpasswd"
```

В данном случае **pma** — имя пользователя, а пароль будет задан в консоли при выполнении второй команды.

Сохраняем конфиги хостов и движемся дальше. Пока nginx перезапускать не надо. Некоторые сущности, которые упоминются в конфиге, еще не настроены. Одну из них сразу добавим, чтобы не забыть. Нам надо сформировать файл dhparam.pem. Процесс длится долго, до получаса на слабых виртуалках. Этот файл нужен для повышения безопасности и получения максимального рейтинга ssl. Насколько этот параметр критичен в реальности, не берусь судить. Если тороплюсь, то настраиваю без него.

```
# openssl dhparam -out /etc/ssl/certs/dhparam.pem 4096
```

Установка php-fpm

Установка php-fpm в Centos 8 сильно упростилась по сравнению с предыдущей версией, потому что в базовом репозитории хранится актуальная версия php 7.2, которой можно пользоваться. Пока нет необходимости подключать сторонние репозитории, так как версия 7.2 вполне свежа и актуальна. Если у вас нет необходимости использовать что-то новее, то можно остановиться на этой версии.

Устанавливаем php и php-fpm в CentOS 8, а так же некоторые популярные модули.

```
# dnf install php-fpm php-cli php-mysqlnd php-json php-gd php-ldap php-odbc php-pdo php-opcache php-pear php-xml php-xmlrpc php-mbstring php-snmp php-soap php-zip
```

```
[root@centos8 ~]# dnf install php-fpm php-cli php-mysqlnd php-gd php-ldap php-odbc php-pdo php-opcache php-pear php-xml php-xmlrpc php-mbstring php-snmp ph
p-soap php-zip
Last metadata expiration check: 0:00:35 ago on Mon 07 Oct 2019 04:48:54 PM MSK.
Dependencies resolved.
================================================================================================================================================
 Package                                Arch                Version                                                Repository                Size
================================================================================================================================================
Installing:
 php-cli                                x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                3.1 M
 php-fpm                                x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                1.6 M
 php-gd                                 x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                 83 k
 php-ldap                               x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                 79 k
 php-mbstring                           x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                580 k
 php-mysqlnd                            x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                190 k
 php-odbc                               x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                 88 k
 php-opcache                            x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                230 k
 php-pdo                                x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                122 k
 php-pear                               noarch              1:1.10.5-8.module_el8.0.0+56+d1ca79aa                  AppStream                358 k
 php-pecl-zip                           x86_64              1.15.3-1.module_el8.0.0+56+d1ca79aa                    AppStream                 51 k
 php-snmp                               x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                 74 k
 php-soap                               x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                177 k
 php-xml                                x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                188 k
 php-xmlrpc                             x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                 89 k
Installing dependencies:
 gd                                     x86_64              2.2.5-6.el8                                            AppStream                144 k
 httpd-filesystem                       noarch              2.4.37-11.module_el8.0.0+172+85fc1f40                  AppStream                 34 k
 jbigkit-libs                           x86_64              2.1-14.el8                                             AppStream                 55 k
 libXpm                                 x86_64              3.5.12-7.el8                                           AppStream                 58 k
 libjpeg-turbo                          x86_64              1.5.3-7.el8                                            AppStream                155 k
 libtiff                                x86_64              4.0.9-13.el8                                           AppStream                188 k
 libwebp                                x86_64              1.0.0-1.el8                                            AppStream                273 k
 libzip                                 x86_64              1.5.1-1.module_el8.0.0+56+d1ca79aa                     AppStream                 63 k
 mariadb-connector-c                    x86_64              3.0.7-1.el8                                            AppStream                148 k
 mariadb-connector-c-config             noarch              3.0.7-1.el8                                            AppStream                 13 k
 net-snmp                               x86_64              1:5.8-7.el8_0.1                                        AppStream                352 k
 net-snmp-agent-libs                    x86_64              1:5.8-7.el8_0.1                                        AppStream                748 k
 nginx-filesystem                       noarch              1:1.14.1-8.module_el8.0.0+5+258f653c                   AppStream                 24 k
 php-common                             x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                653 k
 php-process                            x86_64              7.2.11-1.module_el8.0.0+56+d1ca79aa                    AppStream                 84 k
 unixODBC                               x86_64              2.3.7-1.el8                                            AppStream                458 k
 lm_sensors-libs                        x86_64              3.4.0-17.20180522git70f7e08.el8                        BaseOS                    58 k
 net-snmp-libs                          x86_64              1:5.8-7.el8_0.1                                        BaseOS                   821 k
 perl-Data-Dumper                       x86_64              2.167-399.el8                                          BaseOS                    58 k
Enabling module streams:
 httpd                                                      2.4
 nginx                                                      1.14
 php                                                        7.2

Transaction Summary
================================================================================================================================================
Install  34 Packages

Total download size: 11 M
Installed size: 38 M
Is this ok [y/N]:
```

Проверим конфигурацию php-fpm, которая располагается в файле */etc/php-fpm.d/www.conf*. Изменим пользователя, под которым будет работать php-fpm с apache на nginx.

```
user = nginx
group = nginx
```

Перезапускаем php-fpm и проверяем, запущен ли сокет.

```
# systemctl restart php-fpm
# ll /run/php-fpm/www.sock
srw-rw----+ 1 root root 0 Oct  7 16:59 /run/php-fpm/www.sock
```

Все в порядке, php-fpm запущен и готов к работе. Сделаем еще одну важную настройку. Назначим nginx владельцем директории для хранения сессий php.

```
# chown -R nginx. /var/lib/php/session
```

Более безопасно было бы для каждого сайта делать отдельную директорию для сессий и определять ее в настройках php. Но это уже частный случай. В общем случае, можно оставить так.

Возможно, вам захочется поставить более свежу версию php. Как это сделать, читайте в отдельной статье — обновление php-7.2 до 7.4 в CentOS 8.

Для того, чтобы проверить работу нашего веб сервера, нужно установить ssl сертификаты. Без них nginx с текущим конфигом не запустится. Исправляем это.

Настройка бесплатного ssl сертификата Lets Encrypt

Устанавливаем пакет **certbot** для получения бесплатного ssl сертификата от let’s encrypt. В репозиториях его пока нет, поэтому поставим вручную с сайта

разрабочиков.

```
# cd ~
# wget https://dl.eff.org/certbot-auto
# mv certbot-auto /usr/local/bin/certbot-auto
# chown root /usr/local/bin/certbot-auto
# chmod 0755 /usr/local/bin/certbot-auto
```

При первом запуске certbot-auto предложит установить зависимости, которые ему нужны для работы.

```
# certbot-auto
```

```
[root@centos8 ~]# certbot-auto
Bootstrapping dependencies for RedHat-based OSes that will use Python3... (you can skip this with --no-bootstrap)
dnf is /usr/bin/dnf
dnf is hashed (/usr/bin/dnf)
Last metadata expiration check: 0:16:46 ago on Mon 07 Oct 2019 04:48:54 PM MSK.
Package openssl-1:1.1.1-8.el8.x86_64 is already installed.
Package ca-certificates-2018.2.24-6.el8.noarch is already installed.
Dependencies resolved.
==================================================================================================================================
 Package                        Arch            Version                                        Repository                  Size
==================================================================================================================================
Installing:
 gcc                            x86_64          8.2.1-3.5.el8                                  AppStream                   23 M
 python3-virtualenv             noarch          15.1.0-18.module_el8.0.0+33+0a10c0e1           AppStream                  1.7 M
 python36                       x86_64          3.6.8-2.module_el8.0.0+33+0a10c0e1             AppStream                   19 k
 python36-devel                 x86_64          3.6.8-2.module_el8.0.0+33+0a10c0e1             AppStream                   16 k
 redhat-rpm-config              noarch          116-1.el8.0.1                                  AppStream                   82 k
 augeas-libs                    x86_64          1.10.1-8.el8                                   BaseOS                     392 k
 libffi-devel                   x86_64          3.1-18.el8                                     BaseOS                      28 k
 openssl-devel                  x86_64          1:1.1.1-8.el8                                  BaseOS                     2.3 M
Installing dependencies:
 annobin                        x86_64          8.64-1.el8                                     AppStream                  187 k
 cpp                            x86_64          8.2.1-3.5.el8                                  AppStream                   10 M
 dwz                            x86_64          0.12-9.el8                                     AppStream                  109 k
 efi-srpm-macros                noarch          3-2.el8                                        AppStream                   22 k
 ghc-srpm-macros                noarch          1.4.2-7.el8                                    AppStream                  9.3 k
 go-srpm-macros                 noarch          2-16.el8                                       AppStream                   14 k
 isl                            x86_64          0.16.1-6.el8                                   AppStream                  841 k
 libmpc                         x86_64          1.0.2-9.el8                                    AppStream                   59 k
 ocaml-srpm-macros              noarch          5-4.el8                                        AppStream                  9.4 k
 openblas-srpm-macros           noarch          2-2.el8                                        AppStream                  7.9 k
 perl-srpm-macros               noarch          1-25.el8                                       AppStream                   11 k
 platform-python-devel          x86_64          3.6.8-2.el8_0.0.1                              AppStream                  242 k
 python-rpm-macros              noarch          3-37.el8                                       AppStream                   14 k
```


После установки зависимостей, certbot ругнется на то, что не смог настроить нам nginx, но и хорошо, мы уже сами все настроили. Наша следующая задача — получить бесплатные сертификаты. Для этого временно остановим nginx, если он вдруг оказался запущен и подтвердим владение доменами с помощью temporary webserver, который certbot поднимет сам на время верификации доменов.

```
# systemctl stop nginx
# certbot-auto certonly
```

```
How would you like to authenticate with the ACME CA?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: Nginx Web Server plugin (nginx) [Misconfigured]
2: Spin up a temporary webserver (standalone)
3: Place files in webroot directory (webroot)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate number [1-3] then [enter] (press 'c' to cancel): 2
Plugins selected: Authenticator standalone, Installer None
Please enter in your domain name(s) (comma and/or space separated)  (Enter 'c'
to cancel): z.serveradmin.ru
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for z.serveradmin.ru
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/z.serveradmin.ru/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/z.serveradmin.ru/privkey.pem
   Your cert will expire on 2020-01-05. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot-auto
   again. To non-interactively renew *all* of your certificates, run
   "certbot-auto renew"
 - If you like Certbot, please consider supporting our work by:
```

```
   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

То же самое делаем для второго домена — pma.serveradmin.ru.

После того, как получили сертификаты, можно проверить конфиг nginx и запустить его. Ошибок быть не должно. Если они есть, то разбирайтесь с ними, возможно где-то ошиблись.

```
# nginx -t
# systemctl start nginx
```

Настройка nginx на этом завершена. Он должен корректно запуститься и работать в рабочем режиме.

Теперь сделаем так, чтобы сертификаты автоматически обновлялись перед истечением срока действия. Для этого необходимо изменить конфигурации доменов. Они располагаются в директории */etc/letsencrypt/renewal*. Так как мы генерировали сертификаты с помощью временного веб сервера, наш текущий конфиг *z.serveradmin.ru.conf* выглядит вот так:

```
# renew_before_expiry = 30 days
version = 0.39.0
archive_dir = /etc/letsencrypt/archive/z.serveradmin.ru
cert = /etc/letsencrypt/live/z.serveradmin.ru/cert.pem
privkey = /etc/letsencrypt/live/z.serveradmin.ru/privkey.pem
chain = /etc/letsencrypt/live/z.serveradmin.ru/chain.pem
fullchain = /etc/letsencrypt/live/z.serveradmin.ru/fullchain.pem

# Options used in the renewal process
[renewalparams]
account = a9e5d37f80dafd9f0a2f9a2b019cbc3e
authenticator = standalone
server = https://acme-v02.api.letsencrypt.org/directory
```

Приводим его к следующему виду:

```
# renew_before_expiry = 30 days
version = 0.39.0
archive_dir = /etc/letsencrypt/archive/z.serveradmin.ru
cert = /etc/letsencrypt/live/z.serveradmin.ru/cert.pem
privkey = /etc/letsencrypt/live/z.serveradmin.ru/privkey.pem
chain = /etc/letsencrypt/live/z.serveradmin.ru/chain.pem
fullchain = /etc/letsencrypt/live/z.serveradmin.ru/fullchain.pem

# Options used in the renewal process
[renewalparams]
account = a9e5d37f80dafd9f0a2f9a2b019cbc3e
authenticator = webroot
server = https://acme-v02.api.letsencrypt.org/directory
post_hook = nginx -s reload
[[webroot_map]]
z.serveradmin.ru = /web/sites/z.serveradmin.ru/www
```

По аналогии делаете с остальными виртуальными хостами, для которых используете бесплатные сертификаты let’s encrypt. Осталось дело за малым — настроить автоматический выпуск новых ssl сертификатов, взамен просроченным. Для этого добавляем в */etc/crontab* следующую строку:

```
# Cert Renewal
30 2 * * * root /usr/local/bin/certbot-auto renew — post-hook "nginx -s reload" >> /var/log/le-renew.log
```

Все, с сертификатами закончили. Двигаемся дальше в настройке web сервера.

Установка mariadb на CentOS 8

Дошла очередь до установки сервера баз данных mysql для web сервера на CentOS 8 — **MariaDB**. Я буду устанавливать последнюю стабильную версию на момент написания статьи — 10.4 из официального репозитория mariadb.

Для того, чтобы подключить репозиторий MariaDB, можно воспользоваться специальной страницей на официальном сайте, где можно задать параметры системы и получить конфиг репозитория.

В моем случае конфиг получился следующий.

```
# cat /etc/yum.repos.d/mariadb.repo
```

```
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.4/centos8-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

Устанавливаем последнюю версию mariadb на centos.

```
# dnf install boost-program-options
# dnf install MariaDB-server MariaDB-client – disablerepo=AppStream
```

```
[root@centos8 yum.repos.d]# dnf install mariadb mariadb-server
Last metadata expiration check: 0:24:53 ago on Tue 08 Oct 2019 06:07:26 PM MSK.
Dependencies resolved.
================================================================================
 Package                  Arch       Version                        Repository  Size
================================================================================
Installing:
 mariadb                  x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream  6.2 M
 mariadb-server           x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream   16 M
Installing dependencies:
 mariadb-common           x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream   62 k
 mariadb-connector-c      x86_64     3.0.7-1.el8                    AppStream  148 k
 mariadb-connector-c-config noarch   3.0.7-1.el8                    AppStream   13 k
 mariadb-errmsg           x86_64     3:10.3.11-2.module_el8.0.0+35+6f2527ed AppStream  232 k
 perl-DBD-MySQL           x86_64     4.046-2.module_el8.0.0+72+668237d8 AppStream  156 k
 perl-DBI                 x86_64     1.641-2.module_el8.0.0+66+feleca09 AppStream  740 k
 perl-Digest              noarch     1.17-395.el8                   AppStream   27 k
 perl-Digest-MD5          x86_64     2.55-396.el8                   AppStream   37 k
 perl-Net-SSLeay          x86_64     1.85-6.el8                     AppStream  358 k
 perl-URI                 noarch     1.73-3.el8                     AppStream  116 k
 perl-libnet              noarch     3.11-3.el8                     AppStream  121 k
 perl-Data-Dumper         x86_64     2.167-399.el8                  BaseOS      58 k
 perl-Encode              x86_64     4:2.97-3.el8                   BaseOS     1.5 M
 perl-Getopt-Long         noarch     1:2.50-4.el8                   BaseOS      63 k
 perl-HTTP-Tiny           noarch     0.074-1.el8                    BaseOS      58 k
 perl-MIME-Base64         x86_64     3.15-396.el8                   BaseOS      31 k
 perl-Math-BigInt         noarch     1:1.9998.11-5.el8              BaseOS     195 k
 perl-Math-Complex        noarch     1.59-416.el8                   BaseOS     108 k
 perl-Pod-Escapes         noarch     1:1.07-395.el8                 BaseOS      20 k
 perl-Pod-Perldoc         noarch     3.28-396.el8                   BaseOS      86 k
 perl-Pod-Simple          noarch     1:3.35-395.el8                 BaseOS     213 k
 perl-Pod-Usage           noarch     4:1.69-395.el8                 BaseOS      34 k
 perl-Storable            x86_64     1:3.11-3.el8                   BaseOS      98 k
 perl-Term-ANSIColor      noarch     4.06-396.el8                   BaseOS      46 k
 perl-Term-Cap            noarch     1.17-395.el8                   BaseOS      23 k
 perl-Text-ParseWords     noarch     3.30-395.el8                   BaseOS      18 k
 perl-Time-Local          noarch     1:1.280-1.el8                  BaseOS      34 k
 perl-podlators           noarch     4.11-1.el8                     BaseOS     118 k
 psmisc                   x86_64     23.1-3.el8                     BaseOS     151 k
```


Запускаем mariadb и добавляем в автозагрузку.

```
# systemctl start mariadb
# systemctl enable mariadb
```

Запускаем скрипт начальной конфигурации mysql и задаем пароль для root. Все остальное можно оставить по-умолчанию.

```
# /usr/bin/mysql_secure_installation
```

Сервер баз данных mysql для нашего web сервера готов. Продолжаем настройку. Установим панель управления mysql — phpmyadmin.

Установка phpmyadmin

Для того, чтобы установить phpmyadmin на наш web сервер, достаточно просто распаковать в директроию с виртуальным хостом исходники панели. Все остальные настройки у нас уже готовы, в том числе виртуальный хост с tls сертификатом.

Идем на сайт https://www.phpmyadmin.net и копируем ссылку на последнюю версию панели. Затем загружаем ее через консоль.

```
# cd ~
# wget https://files.phpmyadmin.net/phpMyAdmin/4.9.1/phpMyAdmin-4.9.1-all-languages.zip
```

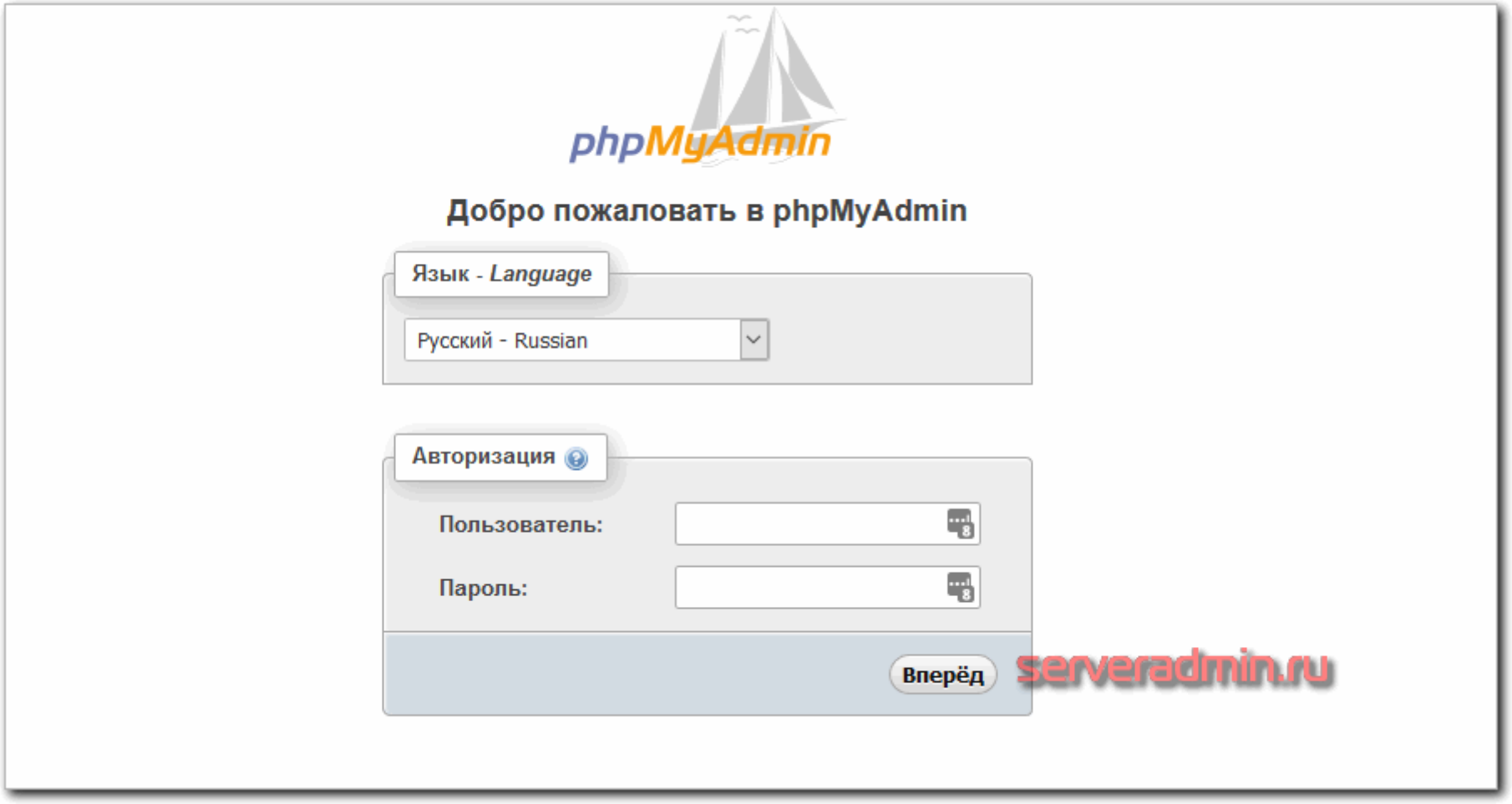
Архив упакован в zip. Если у вас нет на сервере пакета unzip, установите его.

```
# dnf install unzip
```

Распаковываем исходники в директорию виртуального хоста.

```
# unzip phpMyAdmin-4.9.1-all-languages.zip
# cp -R phpMyAdmin-4.9.1-all-languages/* /web/sites/pma.serveradmin.ru/www
# chown -R nginx. /web/sites/pma.serveradmin.ru/www
```

Можно заходить и проверять работу phpmyadmin, пройдя по адресу pma.serveradmin.ru. Ее установка закончена.



Более подробно вопрос установки и настройки phpmyadmin я рассматривал отдельно. Можете зайти в панель и создать базу mysql для тестового сайта, например, wordpress. Затем через консоль загрузить исходники cms и распаковать их.

```
# cd ~
# wget https://ru.wordpress.org/latest-ru_RU.tar.gz
# tar xzvf latest-ru_RU.tar.gz
# cp -R wordpress/* /web/sites/z.serveradmin.ru/www
# chown -R nginx. /web/sites/z.serveradmin.ru/www
```

После этого открывайте в браузере страницу *z.serveradmin.ru* и увидите приветсвие установщика wordpress.


Добро пожаловать. Прежде чем мы начнём, потребуется информация о базе данных. Вот что вам необходимо знать до начала процедуры установки.
1. Имя БД
2. Имя пользователя базы данных
3. Пароль к базе данных
4. Хост БД
5. Префикс таблиц (если вы хотите запустить более чем один WordPress на одной базе)
Мы используем эту информацию, чтобы создать файл wp-config.php. Если по какой-то причине автоматическое создание файла не удалось, не волнуйтесь. Всё это предназначено лишь для заполнения файла настроек. Вы можете просто открыть wp-config-sample.php в текстовом редакторе, внести вашу информацию и сохранить его под именем wp-config.php. Нужна помощь? Пожалуйста.
Скорее всего, эти данные были предоставлены вашим хостинг-провайдером. Если у вас нет этой информации, свяжитесь с их службой поддержки. А если есть...
Вперёд!
serveradmin.ru

На этом основная настройка web сервера завершена. Он уже полностью работоспособен. Если вам достаочно этого функционала, то можете сразу переходить к настройке ротации логов.

Доступ к сайту по sftp

Если вы администратор и единственный пользователь, то больше можно ничего не делать. Вы и так сможете загрузить на сервер все что нужно тем или иным способом. Если же вы будете передавать управление сайтами другим людям, им нужен доступ к директории с исходниками сайта. Раньше для этих целей повсеместно использовали ftp. Если вы хотите так сделать, у меня есть статья по настройке ftp сервера vsftpd.

Я же предлагаю использовать **sftp** по нескольким причинам:

1. Он безопаснее.
2. Его быстрее настроить.
3. Не надо отдельно настраивать firewall.

Статью по настройке sftp доступа я уже тоже писал, все подробности там. Здесь без комментариев выполним необходимые действия.

Создаем пользователя для подключения к сайту и добавляем его в группу sftp. Я обычно использую имя пользователя пересекающееся с названием сайта. Так удобнее управлять.

```
# useradd -s /sbin/nologin -d /web/sites/z.serveradmin.ru z.serveradmin.ru
# passwd z.serveradmin.ru
# groupadd sftp
# usermod -a -G sftp z.serveradmin.ru
```

Открываем конфиг ssh по пути */etc/ssh/sshd_config* и комментируем там одну строку, добавляя далее несколько новых.

```
#Subsystem sftp /usr/libexec/openssh/sftp-server
Subsystem sftp internal-sftp -u 022
Match Group sftp
```

```
ChrootDirectory %h
ForceCommand internal-sftp -u 022
```

Перезапускаем службу sshd.

```
# systemctl restart sshd
```

Теперь нужно сделать владельцем каталогов */web/sites* и */web/sites/z.serveradmin.ru* пользователя root и убрать у остальных права на запись в эти каталоги. Без этого параметр ChrootDirectory работать не будет и при подключении получите ошибку.

```
# chown root /web/sites && chown root /web/sites/z.serveradmin.ru
# chmod 0755 /web/sites && chmod 0755 /web/sites/z.serveradmin.ru
```

Этого уже достаточно, чтобы вы могли подключиться к сайту, к примеру, с помощью программы winscp. Если что-то пойдет не так и будут какие-то ошибки, то смотреть подробности нужно в логе */var/log/secure*. С помощью такого подключения вы можете загрузить к себе исходники сайта, но не править их. У вас доступ только на чтение. В таком подходе возникает много нюансов с правами к файлам и директориям. Дальше я расскажу, как их аккуратно и грамотно разрулить, чтобы у нас не было проблем с дальнейшей работой сайтов от разных пользователей и были все необходимые права.

Работа с сайтами разных пользователей на одном веб сервере

Самый простой способ решить проблему с правами доступа, это сделать владельцем папки с сайтом пользователя, который подключается по sftp. Тогда он сможет нормально работать с файлами, загружать и удалять их. Если доступ в качестве группы установить для nginx, то в целом все будет работать. Для каких-то сайтов такой вариант может оказаться подходящим. То есть сделать надо вот так:

```
# chown -R z.serveradmin.ru:nginx /web/sites/z.serveradmin.ru/
# chmod -R 0775 /web/sites/z.serveradmin.ru/
```

И не забываем обратно вернуть владельца на наш chroot каталог, иначе не подключимся по sftp.

```
# chown root /web/sites/z.serveradmin.ru/
# chmod 0755 /web/sites/z.serveradmin.ru/
```

Но при такой схеме будут проблемы с движками сайтов, которые автоматом что-то к себе загружают, потому что php-fpm работает от пользователя nginx, а у нас права nginx только на группу стоят, где нет прав на запись. К примеру, wordpress не сможет автоматически загружать плагины, будет просить доступ к ftp. Это можно исправить просто выставив права 0775 на все каталоги, но это путь костылей. В общем, могут возникнуть некоторые неудобства. Сейчас мы их исправим.

А теперь сделаем все красиво. Назначаем владельцем содержимого нашего сайта только отдельного пользователя.

```
# chown -R z.serveradmin.ru:z.serveradmin.ru /web/sites/z.serveradmin.ru/
```

Еще раз обращаю внимание на важный нюанс. Chroot доступ для sftp не будет работать, если владельцем директории, куда чрутимся, будет не root. Будете получать ошибку.

```
fatal: bad ownership or modes for chroot directory "/web/sites/z.serveradmin.ru" [postauth]
```

Возвращаем обратно рута владельцем корня chroot.

```
# chown root /web/sites/z.serveradmin.ru/
# chmod 0755 /web/sites/z.serveradmin.ru/
```

Обращаю внимание, что сначала мы рекурсивно назначаем права на все содержимое директорий, а потом возвращаем владельца root только на корень домашнего каталога пользователя z.serveradmin.ru.

Добавляем пользователя nginx в группу z.serveradmin.ru, чтобы web сервер имел доступ на чтение всех файлов виртуального хоста.

```
# usermod -aG z.serveradmin.ru nginx
```

Создаем отдельный pool для php-fpm, который будет обслуживать сайт z.serveradmin.ru и будет запускаться от этого пользователя. Для этого копируем существующий конфиг */etc/php-fpm.d/www.conf* и изменяем в нем несколько строк.

```
# cd /etc/php-fpm.d && cp www.conf z.serveradmin.ru.conf
```

```
[z.serveradmin.ru]
user = z.serveradmin.ru
group = z.serveradmin.ru
listen = /run/php-fpm/z.serveradmin.ru.sock
listen.owner = z.serveradmin.ru
listen.group = z.serveradmin.ru
listen.mode = 0660
;listen.acl_users = apache,nginx # эту строку комментируем
```

Мы поменяли название пула, запустили его от отдельного пользователя и назначили ему отдельный сокет. Теперь идем в настройки этого виртуального хоста в nginx — */etc/nginx/conf.d/z.serveradmin.ru.conf* и везде меняем старое значение сокета

```
fastcgi_pass unix:/run/php-fpm/www.sock;
```

на новое

```
fastcgi_pass unix:/run/php-fpm/z.serveradmin.ru.sock;
```

Перезапускаем nginx и php-fpm и проверяем работу сайта от отдельного пользователя.

```
# systemctl restart nginx
# systemctl restart php-fpm
```

Я рекомендую подключиться по sftp, закинуть еще раз исходники wordpress уже по sftp, установить его и добавить новую тему, чтобы проверить, что все корректно работает. По аналогии проделанные выше действия повторяются для всех остальных сайтов.

Настройка SELinux для web сервера

Раздел для тех, кто хочет настроить SELinux на своем web сервере. Сначала ставим пакет **policycoreutils-python-utils** если он еще не установлен. Он нам нужен для утилиты **semanage**.

```
# dnf install policycoreutils-python-utils
```

Дальше нам нужно подготовить набор правил для selinux при нашей текущей конфигурации web сервера. Общий смысл их следующий:

1. У нас нестандартная директория для сайтов — /web/sites, по-умолчанию запросы будут блокироваться.
2. Надо разрешить nginx взаимодействовать с сокетом.
3. Позволить nginx управлять параметром rlimit_nofile.
4. И еще некоторое количество запретов, которые сможете сами проверить.

Суть описываемой дальше настройки selinux сводится к тому, чтобы следить за ограничениями, которые он накладывает на работу сервиса и добавлять исключения в правила, если вы с ними согласны. Методом нескольких итераций готовится набор правил selinux.

Я для этого пользуюсь следующими командами. Просмотр сработавших блокировок selinux для nginx.

```
# grep nginx /var/log/audit/audit.log | audit2why
```

Можете внимательно посмотреть суть блокировок. На первый взгляд выглядит не понятно, но в целом все гуглится. Можно разобраться, если есть желание. Подробно этот процесс описан в статье на сайте nginx.com.

Сформировать список правил для selinux.

```
# grep nginx /var/log/audit/audit.log | audit2allow -m nginx > ~/nginx.te
```

Этот файл можно потом вручную дополнять или изменять. После того, как все правила будут собраны в один файл, готовлю модуль для selinux.

```
# checkmodule -M -m -o nginx.mod nginx.te
# semodule_package -m nginx.mod -o nginx.pp
```

В конце загружаю этот модуль в selinx.

```
# semodule -i nginx.pp
```

Можно не собирать правила самому, а сразу получить готовый модуль для загрузки и установить его.

```
# grep nginx /var/log/audit/audit.log | audit2allow -M nginx
# semodule -i nginx.pp
```

Потом повторите все то же самое для php-fpm и можно проверять работу web сервера. Если в процессе проверки окажется, что что-то еще не работает, опять смотрите audit.log и добавляйте новые правила, пересобирайте и загружайте модуль. Так, через несколько итераций, получится рабочий набор правил для selinux.

Убедиться, что модуль загружен, можно командой.

```
# semodule -l | grep nginx
```

В целом, по selinux все. Мы просто разрешили все, что веб сервер просил. По идее, надо вдумчиво во всех правилах разбираться и разрешать только то, что считаешь нужным. Я честно скажу, что selinux знаю не очень хорошо. Дальше загрузки готовых модулей и автоматического создания модулей с помощью audit2allow я не двигался. Руками модули никогда не писал. Если есть какой-то более осмысленный и правильный способ настройки selinux на кастомной конфигурации веб сервера, буду рад полезной информации.

Хорошая практическая статья по ручной настройке selinux для web сервера — https://habr.com/ru/post/322904/. Там же есть ссылки на другие статьи автора на тему selinux. Написано содержательно и наглядно, рекомендую для тех, кто будет знакомиться с технологией.

Ротация логов веб сервера nginx

Последний штрих в настройке web сервера — ротация логов виртуальных хостов. Если этого не сделать, то через какое-то, обычно продолжительное, время возникает проблема в связи с огромным размером лог файла.

У нас уже будет файл конфигурации **logrotate** для nginx, который был создан во время установки — */etc/logrotate.d/nginx*. Приведем его к следующему виду:

```
/var/log/nginx/*log
/web/sites/pma.serveradmin.ru/log/*log {

    create 0644 nginx nginx
    size=10M
    rotate 10
    missingok
```

```
    notifempty
    compress
    sharedscripts
    postrotate
        /bin/kill -USR1 `cat /run/nginx.pid 2>/dev/null` 2>/dev/null || true
    endscript
}

/web/sites/z.serveradmin.ru/log/*log {

    create 0644 z.serveradmin.ru z.serveradmin.ru
    size=10M
    rotate 10
    missingok
    notifempty
    compress
    sharedscripts
    postrotate
        /bin/kill -USR1 `cat /run/nginx.pid 2>/dev/null` 2>/dev/null || true
    endscript
}
```

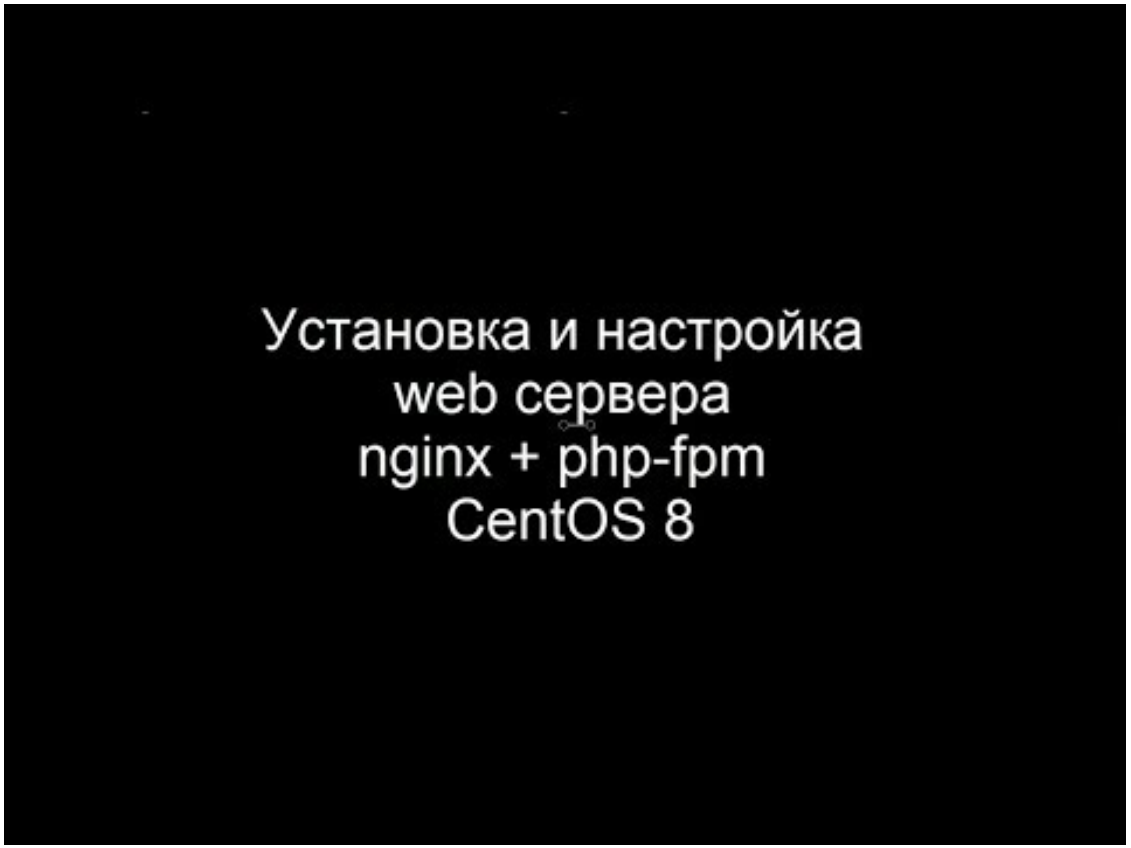
Я предлагаю ротировать файлы логов по достижению ими размера в 10Мб, сжимать после ротации и хранить 10 архивов с логом. Для виртуальных хостов, работающих от отдельного пользователя, новые логи создаются сразу с соответствующими правами, чтобы у пользователя был доступ к ним. Для всех остальных хостов можно использовать самое первое правило, просто добавляя туда новые пути для логов.

Обращаю внимание на важный нюанс при ротации логов по размеру. Скорее всего в общем случае она будет работать не так, как вы ожидаете. Подробности читайте по ссылке. Я привел пример простой конфигурации. Все параметры вы можете поменять по своему усмотрению. Примеров конфигурации logrotate в интернете много.

На этом все. Я рассмотрел все основные моменты, которые необходимы для установки и настройки производительного web сервера на основе nginx и php-fpm последних версий. При этом рассказал о некоторых вещах, которые повышают удобство и гибкость эксплуатации сервера.

Видео

В завершении полное видео настройки web сервера на основе приведенной статьи. Если у кого-то что-то не получается, посмотрите, как это сделал я.



Watch this video on YouTube

Заключение

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

Тема настройки веб сервера обширна. Рассмотреть все варианты в одной статье невозможно, так как функционал будет разниться, в зависимости от назначения сервера. Тем не менее приведу еще несколько ссылок на материалы, которые имеют отношение к настройке web сервера:

- Полный бэкап сервера или отдельных сайтов.
- Мониторинг веб сервера и веб сайта с помощью zabbix.
- Защита админки wordpress с помощью fail2ban.
- Если у вас будут проблемы с ботами, то пригодится статья по блокировке доступа к сайту по странам или защита сайта от ddos.

Если еще что-то полезное вспомню, добавлю ссылки. Пока вроде все. Статья получилось большой и насыщенной. Было не просто все собрать воедино, проверить, связать между собой и оформить в последовательное повествование. Мог где-то ошибиться. Жду комментариев и отзывов. Написал все по своему опыту, как я обычно настраиваю веб сервера. Предполагаю что-то можно сделать более удобно и правильно. Буду рад научиться.

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

Помогла статья? Подписывайся на telegram канал автора

Анонсы всех статей, плюс много другой полезной и интересной информации, которая не попадает на сайт.