Веб-сервисы FreeBSD

Делаем веб-сервер на базе FreeBSD, nginx, php-fpm. Веб-сайт на примере WordPress.

[Введение](#_c713xnuw3xem)

[nginx+php-fpm](#_819ud2jqeqjp)

[nginx](#_qkd3nznw3o1l)

[Безопасность nginx](#_ma6f10wgpbrf)

[Балансировка http- и tcp-трафика](#_slrrf1ys7fhx)

[Fpm](#_cygyo3nlvvuc)

[Настройка пула](#_xfuilsr8cy02)

[Установим nginx из портов](#_2mywftwqmfy5)

[Устанавливаем WordPress](#_6gu1m1cwcsmy)

[Утилита siege](#_7a92q424xxyd)

[Домашнее задание](#_3tfrjxxltv85)

[Дополнительные материалы](#_y937sk8fclye)

[Используемая литература](#_uvp6qax5r1ok)

# 

# Введение

Одна из задач, с которыми чаще всего сталкивается системный администратор, – настройка веб-сервера. Многие слышали о LAMP (Linux+Apache+Mysql+PHP) или LEMP (Linux+Nginx+Mysql+PHP). Мы создадим веб-сервис, установив на FreeBSD Nginx, php-fpm, perconadb и WordPress.

# nginx+php-fpm

nginx+php-fpm позволяет создать быстрый сервер, который может не только отдавать статическое содержимое, но и выполнять php-скрипты

## Nginx

Nginx – веб-сервер и обратный (реверс) прокси, разработанный Игорем Сысоевым, появился еще в Rambler. Он имеет другие принципы работы, нежели Apache2, который на каждое соединение порождает (форкает) дочерний процесс, занимающий (при настройках по умолчанию) 8 Мб памяти. Apache2 до сих пор подвержен атаке Slow Request, когда клиент подключается к серверу, забирает контент, при этом «зарезая» скорость, и повторяет это многократно. Apache2 – ресурсоемкий сервер, поэтому злоумышленник легко может исчерпать память сервера и забить канал связи.

Nginx использует разные методы обработки соединения, зависящие от операционной системы и ее ядра. Это poll (до версии ядра 2.4) и epoll (с версии 2.4) для Linux, Kqueue для FreeBSD, select для Sun Solaris. То есть используются функции ядра, которые обрабатывают события, не порождая дочерние потоки. Аналогично работает балансировщик Haproxy.

При старте nginx поднимает пул соединений и обрабатывает соединения на основе этого пула.

Имеется один рабочий мастер-процесс, который открывает 80-й порт и распределяет соединения по дочерним процессам-воркерам. Мастер-процесс осуществляет основной обмен данными между клиентом и сервером, а также отвечает за кеширование.

Основной вид настройки для виртуального хоста:

|  |
| --- |
| server {  listen 80;  server\_name example.ru www.example.ru;  …  } |

* Секция server:
  + задает порт или IP-адрес и порт (в данном случае порт 80 на всех адресах);
  + server\_name – доменное имя (или несколько имен), при обращении по которому (и при передаче такого параметра host в HTTP-заголовке) сервер должен выдать клиенту главную страницу сайта.

Что необходимо сконфигурировать:

* worker\_processes – количество процессов-воркеров. Как правило, равно количеству ядер в системе. Иногда встречаются иные варианты, например, по количеству жестких дисков, но в 90% случаев оно выставляется по количеству ядер. Также поддерживается значение auto.
* timer\_resolution – уменьшает разрешение таймеров в рабочих процессах.
* worker\_rlimit\_nofile – изменяет ограничение на максимальное число открытых файлов (RLIMIT\_NOFILE) для рабочих процессов. Используется для увеличения ограничения без перезапуска главного процесса. По умолчанию имеет значение 700.

Далее:

|  |
| --- |
| events {  use kqueue;  worker\_connections 1024;  } |

* Какой метод обработки соединений выберем? Если вы не выбрали его, nginx сам определит ОС и используемый для нее метод. Так, для FreeBSD это будет kqueue.
* worker\_connections – сколько соединений может обработать один воркер. Число worker\_connections, умноженное на число воркеров worker\_processes, – число соединений, которые сервер может обработать одновременно. Например, если worker\_connections = 1024 и worker\_processes = 2 (двухъядерный процессор), то сервер сможет обрабатывать до 2048 соединений.

Рассмотрим более подробно vhost:

|  |
| --- |
| server {  server\_name localhost;  listen 127.0.0.1:80;  root /var/www/localhost;  index index.php;  location / {  index index.html index.php;  }  location ~ \.php$  {  include fastcgi\_params;  fastcgi\_pass unix:/var/run/php5-fpm.sock;  fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  }  location ~\*\.(gif|jpg|png)$ {  root /var/www/localhost;  expires 30d;  } |

Секция server содержит:

* server\_name – доменное имя сервера;
* listen – IP-адрес и прослушиваемый порт, т. е. в примере мы слушаем 80-й порт на локалхосте.
* root – указывает директорию, где хранится корень сайта. На эту директорию должны быть права, чтобы пользователь nginx имел доступ.
* Далее index – перечисление того, что считается индексом (html или php и т. д.)

Далее идет описание локейшенов, здесь они задаются тремя секциями.

* В первом локейшене указано, что корневой индекс может быть index.html или index.php.
* Второй локейшен описывает те пути, которые оканчиваются на php. Используем конфигурацию fast cgi, для подключения используем UNIX-сокет и указываем, какие параметры мы передаем php.
* Последний локейшен – кеширование. Мы указываем, что картинки будем кешировать в течение 30 дней. Первый запрос пойдет напрямую, заберет файл с диска, далее будет забирать из кеша.

### Безопасность nginx

Nginx – довольно безопасный сервер, поддерживает многие возможности, знакомые нам по Apache. Также он поддерживает базовую аутентификацию (basic\_auth), ограничение по IP-адресам, geoip, ограничение по странам, городам, с помощью регулярных выражений можно разрешать/запрещать post/get запросы, в нём имеется mod\_sequrity, позволяющий создавать Web Aplication Firewall.

Nginx не подвержен атаке Slow Request. В nginx сравнительно редко находят уязвимости, что видно из лога разработки.

Nginx поддерживает http/2 и spdy.

### Балансировка http- и tcp-трафика

Nginx отлично работает в качестве балансировщика. В нем реализовано несколько алгоритмов балансировки. По умолчанию – Round Robin, передающий запросы серверам по очереди, есть балансировка на основе ip hash, есть модули, которые делают Sticky Session, чтобы запросы попадали на один и тот же сервер. В документации описано, как написать свой мод.

Кроме http, nginx умеет делать балансировку tcp-трафика, то есть его можно поставить как фронтенд перед любыми другими серверами (например, почтовым), но на бэкенде. Сложность такого подхода в том, что в общем случае бэкенд увидит IP-адрес балансировщика, а не отправителя. Для этого, например, у harproxy есть режим transparent, а в nginx проблема также решается подключением соответствующих модулей.

Таким образом, nginx может работать как шлюз, повышая безопасность и отказоустойчивость системы.

Архитектура и модули nginx таковы, что позволяют создать довольно необычные и полезные вещи – например, широковещательный сервер вместо той же Wowza (она написана на Java, тяжелая, платная).

Nginx хорошо работает с кешированием, в нем реализовано динамическое кеширование. Он умеет собирать страницу из блоков, используя технологию SSI (Server Side Includes). Похожей функциональностью обладает Varnish, который тоже хорошо работает как кеширующий сервер и также собирает страницы из блоков, но использует другую технологию – ESI (Edge Side Includes). В России Varnish недостаточно популярен.

## Fpm

Fpm – реализация fast-cgi для php. Изначально это был разработанный badoo патч для php, теперь это отдельный сервер. У php есть ряд проблем, которые решает php-fpm.

Fpm реализует продвинутое управление процессами с корректной (graceful) процедурой остановки и запуска, имеет возможность запуска воркеров с различным uid/gid/chroot-окружением, а также запуска на различных портах с использованием разных php.ini (замещение safe\_mode).

Можно запускать Fpm на разных портах, можно на разных unix-сокетах. Можно логировать стандартные потоки вывода (stdout) и ошибок (stderr); Аварийный перезапуск будет полезен в случае внезапного разрушения opcode-кеша, либо если умер child.

В Fpm имеется поддержка ускоренной загрузки (accelerated upload).

Более подробно об этом можно прочесть на сайте разработчика: <http://php-fpm.org>.

В настоящее время php-fpm – уже не отдельный патч, а неотъемлемая часть php. Использование php-fpm – это переход от обычной модели к фронтенд-бэкенд-архитектуре, что более безопасно, надежно и масштабируемо. Все это очень хорошо работает с микросервисами, автоматизацией и развертыванием.

### Настройка пула

php-fpm и nginx работают схоже, php-fpm так же не плодит процессы, в нем такой же механизм обработки.

Рассмотрим пример настройки пула:

|  |
| --- |
| [pool]  listen = /var/run/php5-fpm.sock  listen.backlog = -1  user = www-data  group = www-data  pm = dynamic  pm.max\_children = 20  pm.start\_servers = 8  pm.min\_spare\_servers = 8  pm.max\_spare\_servers =12  pm.max\_requests = 2048  request\_slowlog\_timeout = 10s  slowlog = /var/log/php-slow.log  catch\_workers\_output = yes  env[HOSTNAME] = $HOSTNAME  env[PATH] = /usr/local/bin:/usrbin:/bin  env[TMP] = tmp  env[TMPDIR] = tmp  env[TEMP] = tmp  env[$OSTYPE] = $OSTYPE  env [MACHTYPE] = $MACHTYPE |

В квадратных скобках название пула, которое будет отображаться в процессах.

* listen – расположение UNIX-сокета;
* listen.backlog = -1 – длина бэклога, для BSD должна быть бесконечной, для Linux – максимальным 32-битным числом;
* user = www-data – пользователь;
* group = www-data – группа, обычно от чьего имени запускается веб-сервис;
* pm = dynamic – настройка, типичная для php-fpm: какой тип пула используется – динамический, статический или on demand (встречается редко). В статическом пуле одновременно стартует фиксированное количество дочерних процессов, в динамическом – минимальное, которое может увеличиваться до максимального в зависимости от нагрузки;
* pm.max\_children = 20 – максимальное число дочерних процессов;
* pm.start\_servers = 8 – стартует 8 процессов серверов;
* pm.min\_spare\_servers = 8 – минимальное число неактивных процессов сервера;
* pm.max\_spare\_servers =12 – максимальное число неактивных процессов сервера;
* pm.max\_requests = 2048 – максимальное количество запросов;
* request\_slowlog\_timeout = 10s – slow log: если скрипт отрабатывает более чем 10 секунд, данные об этом будут записываться в slowlog;
* slowlog = /var/log/php-slow.log – расположение slowlog;
* catch\_workers\_output = yes – отлавливаем воркеры на случай, если есть ошибки.

Далее переменные окружения, которые будут доступны в PHP:

|  |
| --- |
| env[HOSTNAME] = $HOSTNAME  env[PATH] = /usr/local/bin:/usrbin:/bin  env[TMP] = tmp  env[TMPDIR] = tmp  env[TEMP] = tmp  env[$OSTYPE] = $OSTYPE  env [MACHTYPE] = $MACHTYPE |

* env[HOSTNAME] – имя хоста;
* env[PATH] – путь к исполняемым бинарным файлам;
* env[TMP], env[TMPDIR], env[TEMP] – временная директория;
* env[$OSTYPE] – тип операционной системы;
* env [MACHTYPE] – разрядность.

Как рассчитать стартовый сервер:

|  |
| --- |
| dynamic  pm.max\_children  pm.start\_servers  Рассчитывается по формуле min\_spare\_severs+(max\_spare\_servers - min\_spare\_servers)/2  pm.min\_spare\_servers  pm.max\_spare\_servers  pm.max\_request  requests\_slowlog\_timeout  slowlog  catch\_workers\_output |

То есть мы берем минимальное число неактивных процессов сервера и прибавляем к нему разделенную на два разницу между максимальным и минимальным числом неактивных серверов.

## Установим nginx из портов

|  |
| --- |
| cd /usr/ports  make search name="nginx" |

Нам доступны

* nginx – версия по умолчанию;
* full – полная версия, как в Ubuntu;
* lite – облегченная версия без всего лишнего;
* naxsi – защищенная версия с возможностью создать Web Application Firewall;
* devel – версия для разработчиков.

Установим nginx:

|  |
| --- |
| cd www/nginx  make config |

Доступны модули (от разработчиков):

* DSO – поддержка динамических модулей. Чтобы модуль можно было подключить, его необходимо собрать под нужную операционную систему и нужную версию nginx, положить в директорию /usr/libexec/nginx, и nginx после перезапуска подключит модуль;
* DEBUG – собрать с поддержкой отладки;
* DEBUGLOG – включить debug log (--with-debug);
* AIO – поддержка асинхронного файлового ввода-вывода;
* IPV6 – поддержка IPv6;
* GOOGLE\_PERFTOOLS – perf от Google;
* HTTP\_AUTH\_REQUEST – модуль авторизации;
* HTTP\_DAV – webdav – расширение HTTP для работы с документами/файлами. Используется среди прочего в Яндекс.Диск и Google Drive.
* HTTP\_FLV – FLV для видео по запросу;
* HTTP\_GEOIP – работает с базой GeoIP, позволяя определять по IP-адресу страну/город;
* HTTP\_GZIP\_STATIC – если сайт мало изменяется, позволяет отдавать сайт из архива;
* HTTP\_MP4 – поддержка mp4 для отдачи видео;
* HTTP\_PERL – модуль perl позволяет писать конфигурационные файлы на Perl;
* HTTP\_REALIP – realip – если nginx стоит за балансировщиком, без этого модуля будет виден IP-адрес не отправителя, а балансировщика. С модулем realip будет виден настоящий адрес отправителя;
* HTTP\_REWRITE – htto\_rewrite\_module, функция, аналогичная одноименной в Apache2.
* HTTP\_SECURE\_LINK – защита от прямых ссылок: веб-приложение для каждого запроса генерирует ссылку вида filename.png?хэш, nginx проверяет хэш, и если сгенерировано верно, отдает файл клиенту. С другого IP-адреса эта ссылка уже работать не будет. Так реализуется защита от скачивания контента;
* HTTP\_SSL – шифрование SSL, HTTPS;
* HTTP\_STATUS – аналог status у Apache;
* MAIL, MAIL\_IMAP, MAIL\_POP3, MAIL\_SMTP – nginx может проксировать почту;
* HTTPV2 – поддержка http/2;
* NJS – модуль http\_javascript;
* STREAM – модуль для проксирования TCP;
* STREAM\_SSL – поддержка SSL для TCP;
* WWW – примеры для html.

Сторонние модули:

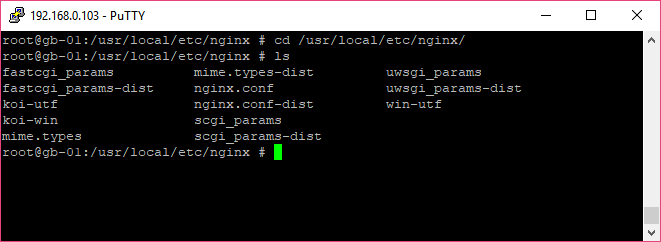
* HEADERS\_MORE – свои http-заголовки;
* HTTP\_AUTH\_DIGEST, \_KRBS, \_LDAP, \_PAM – поддержка дайджест-, Kerberos-, LDAP- и PAM-аутентификации;
* HTTP\_DAV\_EXT – более расширенный вариант того, что идет из коробки, поддерживаются дополнительные команды для WebDAV;
* HTTP\_JSON\_STATUS – расширенный json\_status;
* HTTP\_REDIS – может напрямую забирать данные от Redis;
* HTTP\_TARANTOOL, HTTP\_UPLOAD, HTTP\_UPLOAD\_PORGRESS – Tarantool Upstream module, upload и upload\_porgress: с их помощью на одном nginx можно сделать быструю загрузку без применения скриптов;
* HTTP\_UPSTREAM\_CHECK – апстрим – группа серверов? куда перенаправляется запрос от клиента (балансировка). Nginx не умеет активно проверять бэкенды, то есть по умолчанию применяется пассивная проверка состояния. К примеру, ошибка 404 от бэкенда считается нормальным результатом, хотя на самом деле она может быть признаком проблемы. Модуль подключает активные проверки бэкендов;
* HTTP\_UPSTREAM\_STICKY – модуль для sticky sessions – заменяет ip-hash-балансировку: клиент будет попадать на один и тот же бэкенд;
* DYNAMIC\_UPSTREAM – если указаны доменные имена, они резолвятся при первом запуске. Если адрес сменился, все равно будет использоваться старый IP. Этот модуль в такой ситуации перерезолвит доменные имена через некоторое время.
* LUA – модуль Lua, позволяет писать конфиги на языке Lua.
* MEMC – модуль memcached, позволяет забирать данные напрямую.
* MODSECURITY – порт mod\_security от Apache;
* NAXSI – naxsi;
* PASSENGER – пассендежер для Ruby;
* RTMP – модуль для создания сервера потокового видео (rtmp или hls/hsd).

|  |
| --- |
| make install clean |

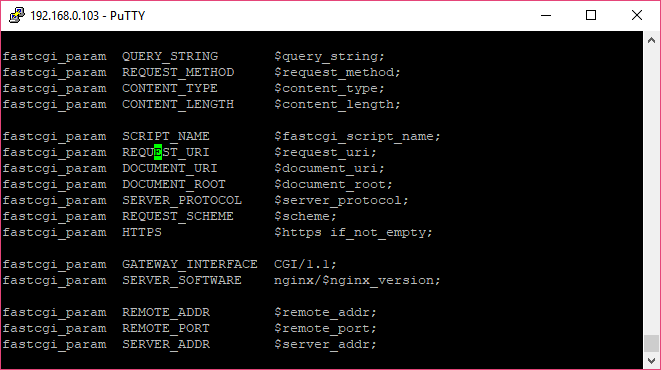
Проверяем:

|  |
| --- |
| nginx -V |

Далее настроим конфиги:



|  |
| --- |
| vi fastcgi\_params |



Менять здесь ничего не нужно: в файле фактически описываются переменные fast-cgi. Раньше нужно было добавлять SCRIPT\_NAME, сейчас не требуется.

Далее:

|  |
| --- |
| vi nginx.conf |

|  |
| --- |
| user www;  worker\_processes 1;  # This default error log path is compiled-in to make sure configuration parsing  # errors are logged somewhere, especially during unattended boot when stderr  # isn't normally logged anywhere. This path will be touched on every nginx  # start regardless of error log location configured here. See  # https://trac.nginx.org/nginx/ticket/147 for more info.  #  #error\_log /var/log/nginx/error.log;  #  #pid logs/nginx.pid;  events {  worker\_connections 1024;  }  nginx.conf: 115 lines, 2796 characters.  root@gb-01:/usr/local/etc/nginx # cat nginx.co  cat: nginx.co: No such file or directory  root@gb-01:/usr/local/etc/nginx # cat nginx.conf  user www;  worker\_processes 1;  # This default error log path is compiled-in to make sure configuration parsing  # errors are logged somewhere, especially during unattended boot when stderr  # isn't normally logged anywhere. This path will be touched on every nginx  # start regardless of error log location configured here. See  # https://trac.nginx.org/nginx/ticket/147 for more info.  #  #error\_log /var/log/nginx/error.log;  #  #pid logs/nginx.pid;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  '$status $body\_bytes\_sent "$http\_referer" '  '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  access\_log /var/log/nginx/access.log main;  sendfile on;  tcp\_nopush on;  keepalive\_timeout 65;  gzip on;  server {  listen 80;  server\_name localhost;  #charset koi8-r;  access\_log /var/log/nginx/localhost.access.log main;  location / {  root /usr/local/www/wp;  index index.php;  }  #error\_page 404 /404.html;  # redirect server error pages to the static page /50x.html  #  error\_page 500 502 503 504 /50x.html;  location = /50x.html {  root /usr/local/www/nginx-dist;  }  # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000  #  location ~ \.php$ {  root /usr/local/www/wp;  fastcgi\_pass 127.0.0.1:9000;  fastcgi\_index index.php;  fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  include fastcgi\_params;  }  # deny access to .htaccess files, if Apache's document root  # concurs with nginx's one  #  location ~ /\.ht {  deny all;  }  }  } |

* Пользователь: www;
* так как ядро одно, worker\_process один;
* для events у нас будет использоваться kqueque (не указан, потому что мы используем FreeBSD), число соединений оставим 1024. Тип по умолчанию оставим octet-stream;
* настроим формат лога по умолчанию, название формата лога – main:
  + $remote\_addr – удаленный адрес;
  + $remote\_user – удаленный пользователь, если есть;
  + $time\_local – локальное время;
  + $request – какой запрос;
  + $status – какой статус мы вернули (200, 404 и т. д.);
  + $body\_bytes\_sent – сколько байт мы отправили;
  + $http\_referer – откуда пришел запрос;
  + $http\_user\_agent – UserAgent;
  + $http\_x\_forwarded\_for – цепочка прокси-серверов и IP-адрес отправителя;
* acess\_log запишем в /var/log/nginx/access.log, формат будем использовать заданный нами main;
* sendfile – позволяет на уровне ядра избежать двойного копирования, что позволит отдавать как можно быстрее;
* tcp\_nopush – накапливать данные в буфере, пока не будет сформирован минимальный размер MSS для tcp-сегмента;
* keepalive – если новый запрос поступает в течение 65 секунд после предыдущего, будем использовать уже открытое tcp-соединение;
* включим gzip.

Далее следуют настройки виртуальных хостов:

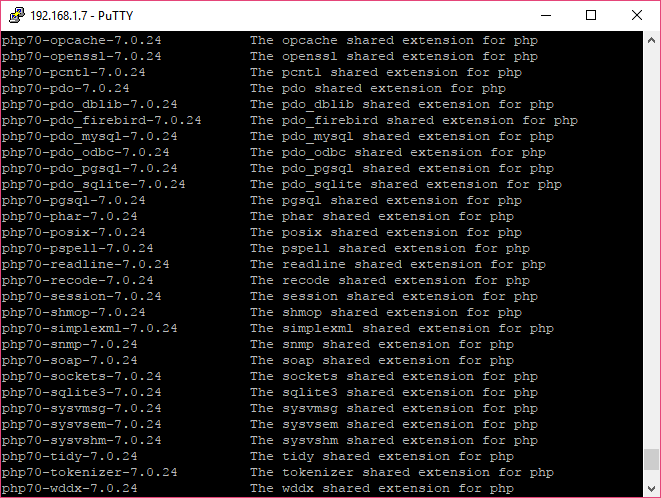
* listen – будем слушать 80-й порт на всех IP-адресах;
* charset – можно не указывать;
* root – укажем, где будет лежать наш сайт на WordPress;
* access\_log раскомментируем и укажем лог для данного виртуального хоста: он тоже будет main (можно использовать разные форматы, можно было бы указать другой);
* индекс сразу php;
* error по умолчанию;
* proxy\_pass для Apache использовать не будем, удаляем:

|  |
| --- |
| # proxy the PHP scripts to Apache listening on 127.0.0.1:80  #  #location ~ \.php$ {  # proxy\_pass http://127.0.0.1;  #} |

* fcgi используем, будем использовать php-fpm, раскомментируем;
* запретим выдачу файлов .htaccess, которые могут оказаться «наследством» от Apache2.

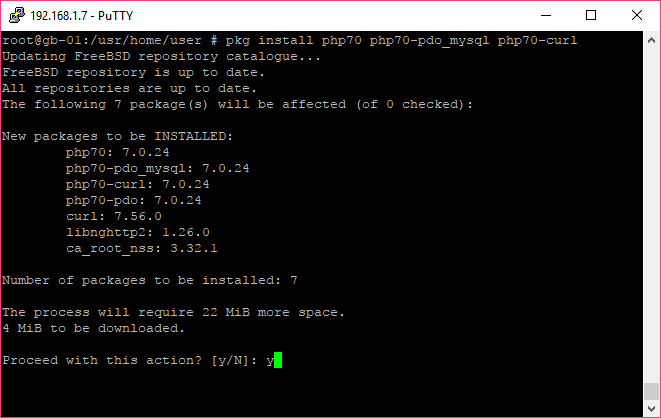
php поставим не из портов (это долго), а через pkg.

|  |
| --- |
| pkg search php73 |

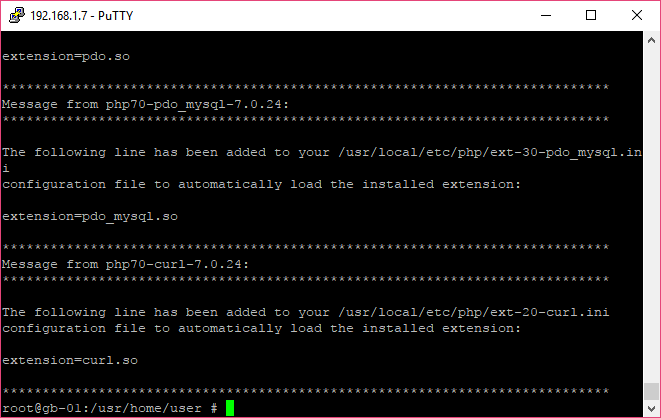


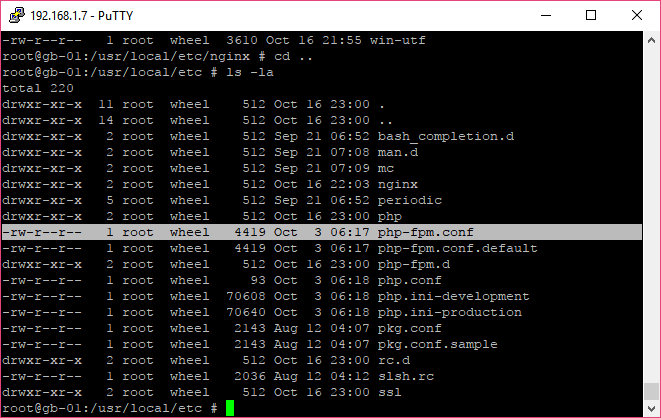
Нам понадобится сам php, pdo\_mysql для работы с mysql и curl для WordPress.

|  |
| --- |
| pkg install php73 php73-pdo\_mysql php73-mysqli php73-curl php73-pecl-mcrypt php73-mbstring php73-zlib php73-gd php73-json php73-hash |



Готово.

   
  
Появился php-fpm.conf.

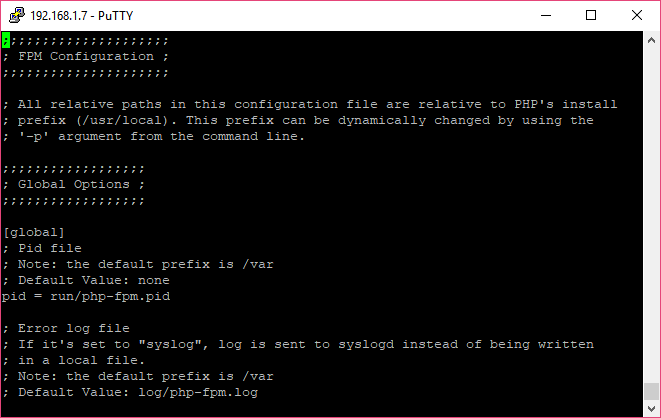


Настроим:

|  |
| --- |
| vi php-fpm.conf |

pid по умолчанию будет в /var.

error\_log по умолчанию пишет в syslog, не будем трогать.

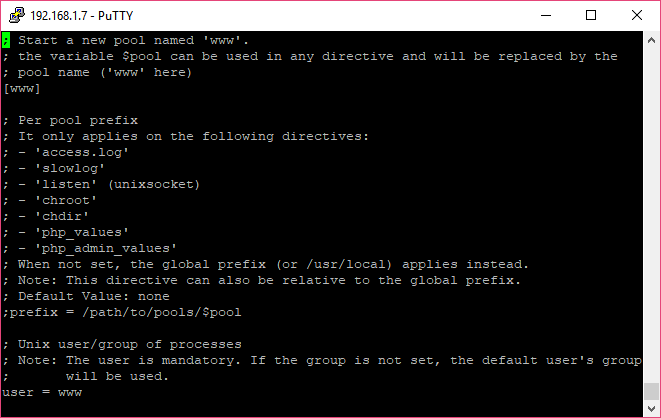


events.mechanism надо заменить с epol для linux на kqueue.

|  |
| --- |
| ; Specify the event mechanism FPM will use. The following is available:  ; - select (any POSIX os)  ; - poll (any POSIX os)  ; - epoll (linux >= 2.5.44)  ; - kqueue (FreeBSD >= 4.1, OpenBSD >= 2.9, NetBSD >= 2.0)  ; - /dev/poll (Solaris >= 7)  ; - port (Solaris >= 10)  ; Default Value: not set (auto detection)  ;events.mechanism = epoll  events.mechanism = kqueue |

Далее настроим пул:

|  |
| --- |
| cd php-fpm.d  vi www.conf |



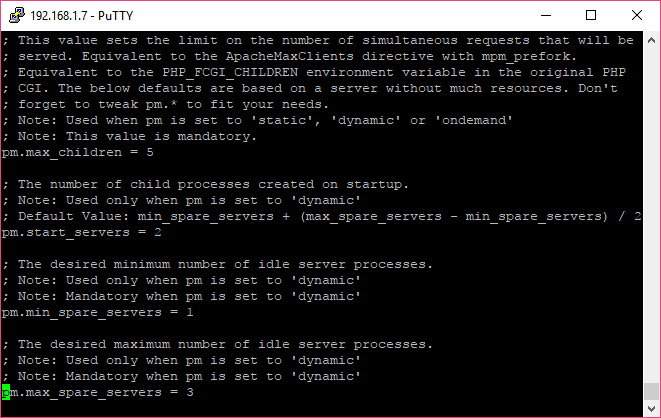
* Имя пула – www.
* Имя пользователя – www.
* Группа www.
* listen=127.0.0.1:9000 – оставим.
* listen.backlog=-1 – оставим.
* Для listener устанавливаем:

|  |
| --- |
| listen.owner = www  listen.group = www  listen.mode = 0660 |

В комментариях описаны статический, динамический и on demand пулы.

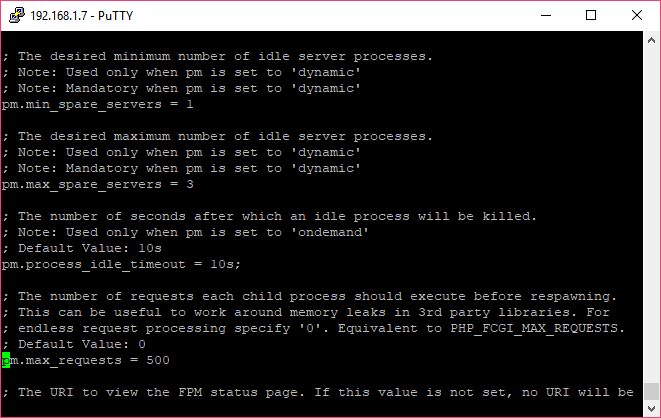
По умолчанию:

* pm = dynamic;
* для базовой конфигурации (среднестатистического сервера) будет достаточно конфигурации по умолчанию.



Раскомментируем:

* таймаут 10 секунд;
* max\_request 500 на 1 worker.

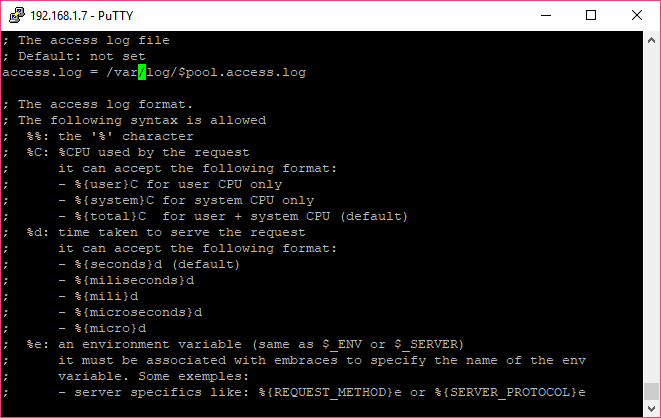


У php-fpm также есть статус, который может понадобиться:

|  |
| --- |
| ;pm.status\_path = /status |

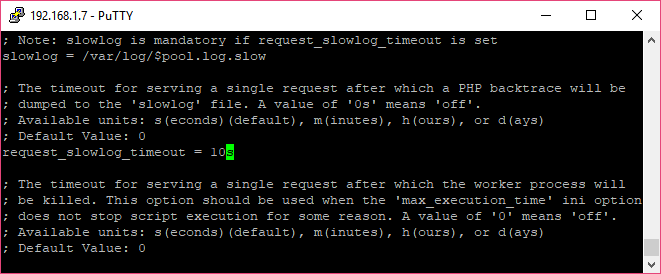
но мы пока его раскомментировать не будем.

Раскомментируем и поправим путь для лога:



Далее следует подробное описание формата лога.

За ним следует slow.log, его тоже будем писать в /var/log. Таймаут можно задать в секундах, минутах, часах, днях. Значение «0» – отключить. Мы поставим 10 секунд.



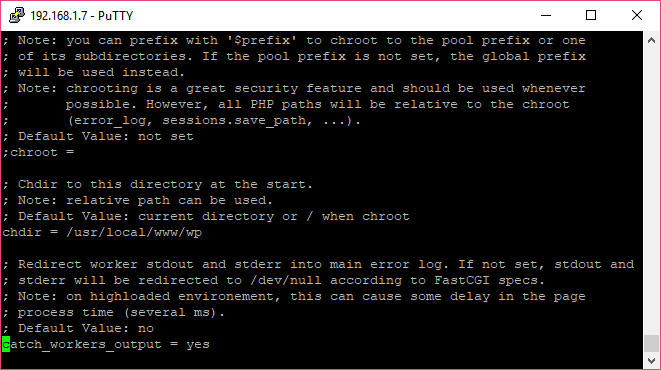
Раскомментируем лимит на дескрипторы открытых файлов.

|  |
| --- |
| ; Set open file descriptor rlimit.  ; Default Value: system defined value  rlimit\_files = 1024 |

rlimit\_core менять не будем.

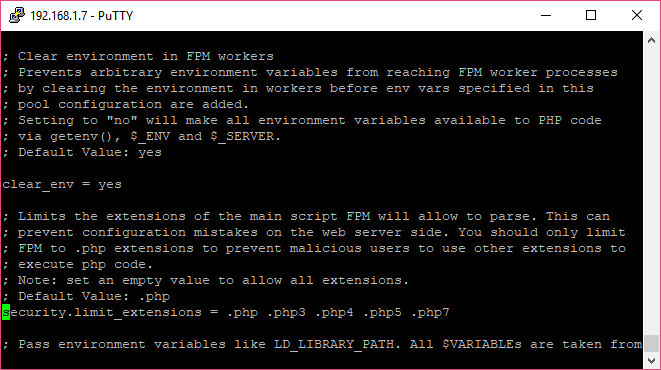
Далее:

* chroot трогать не будем;
* chdir поменяем;
* catch\_worker\_output раскомментируем.

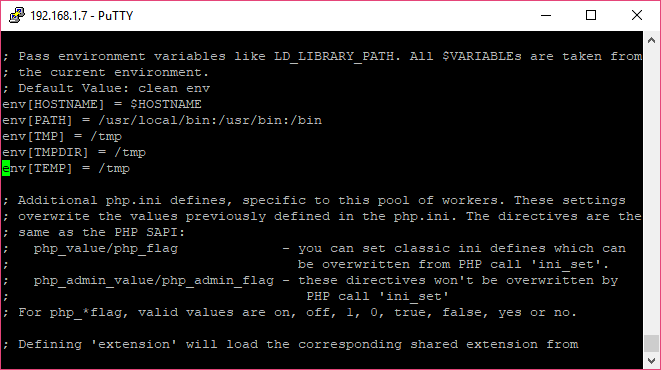


clear environment будем очищать, раскомментируем и меняем на yes.

sequrity limit тоже раскомментируем.

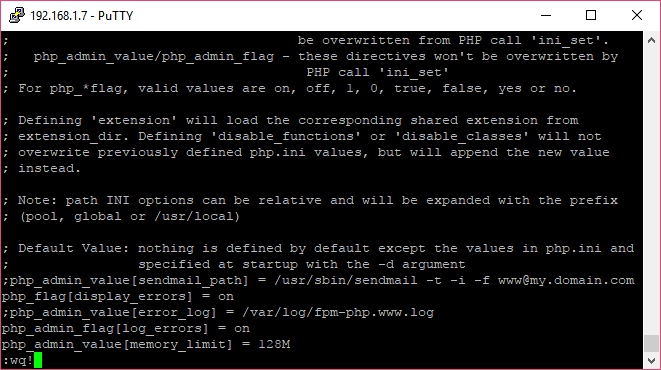


Далее – переменные окружения.



Переменные из php.ini тоже можно переопределить, например:

* раскомментируем и поставим on: php\_flag[display\_errors] = off;
* раскомментируем: php\_admin\_flag[log\_errors] = on;
* раскомментируем и заменим 32 на 128: php\_admin\_value[memory\_limit] = 128M.



Сохраняем:

|  |
| --- |
| ; Start a new pool named 'www'.  ; the variable $pool can be used in any directive and will be replaced by the  ; pool name ('www' here)  [www]  ; Per pool prefix  ; It only applies on the following directives:  ; - 'access.log'  ; - 'slowlog'  ; - 'listen' (unixsocket)  ; - 'chroot'  ; - 'chdir'  ; - 'php\_values'  ; - 'php\_admin\_values'  ; When not set, the global prefix (or /usr/local) applies instead.  ; Note: This directive can also be relative to the global prefix.  ; Default Value: none  ;prefix = /path/to/pools/$pool  ; Unix user/group of processes  ; Note: The user is mandatory. If the group is not set, the default user's group  ; will be used.  user = www  group = www  ; The address on which to accept FastCGI requests.  ; Valid syntaxes are:  ; 'ip.add.re.ss:port' - to listen on a TCP socket to a specific IPv4 ;address on  ; a specific port;  ; '[ip:6:addr:ess]:port' - to listen on a TCP socket to a specific IPv6 ;address on  ; a specific port;  ; 'port' - to listen on a TCP socket to all addresses  ; (IPv6 and IPv4-mapped) on a specific port;  ; '/path/to/unix/socket' - to listen on a unix socket.  ; Note: This value is mandatory.  listen = 127.0.0.1:9000  ; Set listen(2) backlog.  ; Default Value: 511 (-1 on FreeBSD and OpenBSD)  listen.backlog = -1  ; Set permissions for unix socket, if one is used. In Linux, read/write  ; permissions must be set in order to allow connections from a web server. Many  ; BSD-derived systems allow connections regardless of permissions.  ; Default Values: user and group are set as the running user  ; mode is set to 0660  listen.owner = www  listen.group = www  listen.mode = 0660  ; When POSIX Access Control Lists are supported you can set them using  ; these options, value is a comma separated list of user/group names.  ; When set, listen.owner and listen.group are ignored  ;listen.acl\_users =  ;listen.acl\_groups =  ; List of addresses (IPv4/IPv6) of FastCGI clients which are allowed to connect.  ; Equivalent to the FCGI\_WEB\_SERVER\_ADDRS environment variable in the original  ; PHP FCGI (5.2.2+). Makes sense only with a tcp listening socket. Each address  ; must be separated by a comma. If this value is left blank, connections will be  ; accepted from any ip address.  ; Default Value: any  ;listen.allowed\_clients = 127.0.0.1  ; Specify the nice(2) priority to apply to the pool processes (only if set)  ; The value can vary from -19 (highest priority) to 20 (lower priority)  ; Note: - It will only work if the FPM master process is launched as root  ; - The pool processes will inherit the master process priority  ; unless it specified otherwise  ; Default Value: no set  ; process.priority = -19  ; Choose how the process manager will control the number of child processes.  ; Possible Values:  ; static - a fixed number (pm.max\_children) of child processes;  ; dynamic - the number of child processes are set dynamically based on the  ; following directives. With this process management, there will be  ; always at least 1 children.  ; pm.max\_children - the maximum number of children that can  ; be alive at the same time.  ; pm.start\_servers - the number of children created on startup.  ; pm.min\_spare\_servers - the minimum number of children in 'idle'  ; state (waiting to process). If the number  ; of 'idle' processes is less than this  ; number then some children will be created.  ; pm.max\_spare\_servers - the maximum number of children in 'idle'  ; state (waiting to process). If the number  ; of 'idle' processes is greater than this  ; number then some children will be killed.  ; ondemand - no children are created at startup. Children will be forked when  ; new requests will connect. The following parameter are used:  ; pm.max\_children - the maximum number of children that  ; can be alive at the same time.  ; pm.process\_idle\_timeout - The number of seconds after which  ; an idle process will be killed.  ; Note: This value is mandatory.  pm = dynamic  ; The number of child processes to be created when pm is set to 'static' and the  ; maximum number of child processes when pm is set to 'dynamic' or 'ondemand'.  ; This value sets the limit on the number of simultaneous requests that will be  ; served. Equivalent to the ApacheMaxClients directive with mpm\_prefork.  ; Equivalent to the PHP\_FCGI\_CHILDREN environment variable in the original PHP  ; CGI. The below defaults are based on a server without much resources. Don't  ; forget to tweak pm.\* to fit your needs.  ; Note: Used when pm is set to 'static', 'dynamic' or 'ondemand'  ; Note: This value is mandatory.  pm.max\_children = 5  ; The number of child processes created on startup.  ; Note: Used only when pm is set to 'dynamic'  ; Default Value: min\_spare\_servers + (max\_spare\_servers - min\_spare\_servers) / 2  pm.start\_servers = 2  ; The desired minimum number of idle server processes.  ; Note: Used only when pm is set to 'dynamic'  ; Note: Mandatory when pm is set to 'dynamic'  pm.min\_spare\_servers = 1  ; The desired maximum number of idle server processes.  ; Note: Used only when pm is set to 'dynamic'  ; Note: Mandatory when pm is set to 'dynamic'  pm.max\_spare\_servers = 3  ; The number of seconds after which an idle process will be killed.  ; Note: Used only when pm is set to 'ondemand'  ; Default Value: 10s  pm.process\_idle\_timeout = 10s;  ; The number of requests each child process should execute before respawning.  ; This can be useful to work around memory leaks in 3rd party libraries. For  ; endless request processing specify '0'. Equivalent to PHP\_FCGI\_MAX\_REQUESTS.  ; Default Value: 0  pm.max\_requests = 500  ; The URI to view the FPM status page. If this value is not set, no URI will be  ; recognized as a status page. It shows the following informations:  ; pool - the name of the pool;  ; process manager - static, dynamic or ondemand;  ; start time - the date and time FPM has started;  ; start since - number of seconds since FPM has started;  ; accepted conn - the number of request accepted by the pool;  ; listen queue - the number of request in the queue of pending  ; connections (see backlog in listen(2));  ; max listen queue - the maximum number of requests in the queue  ; of pending connections since FPM has started;  ; listen queue len - the size of the socket queue of pending connections;  ; idle processes - the number of idle processes;  ; active processes - the number of active processes;  ; total processes - the number of idle + active processes;  ; max active processes - the maximum number of active processes since FPM  ; has started;  ; max children reached - number of times, the process limit has been reached,  ; when pm tries to start more children (works only for  ; pm 'dynamic' and 'ondemand');  ; Value are updated in real time.  ; Example output:  ; pool: www  ; process manager: static  ; start time: 01/Jul/2011:17:53:49 +0200  ; start since: 62636  ; accepted conn: 190460  ; listen queue: 0  ; max listen queue: 1  ; listen queue len: 42  ; idle processes: 4  ; active processes: 11  ; total processes: 15  ; max active processes: 12  ; max children reached: 0  ;  ; By default the status page output is formatted as text/plain. Passing either  ; 'html', 'xml' or 'json' in the query string will return the corresponding  ; output syntax. Example:  ; http://www.foo.bar/status  ; http://www.foo.bar/status?json  ; http://www.foo.bar/status?html  ; http://www.foo.bar/status?xml  ;  ; By default the status page only outputs short status. Passing 'full' in the  ; query string will also return status for each pool process.  ; Example:  ; http://www.foo.bar/status?full  ; http://www.foo.bar/status?json&full  ; http://www.foo.bar/status?html&full  ; http://www.foo.bar/status?xml&full  ; The Full status returns for each process:  ; pid - the PID of the process;  ; state - the state of the process (Idle, Running, ...);  ; start time - the date and time the process has started;  ; start since - the number of seconds since the process has started;  ; requests - the number of requests the process has served;  ; request duration - the duration in µs of the requests;  ; request method - the request method (GET, POST, ...);  ; request URI - the request URI with the query string;  ; content length - the content length of the request (only with POST);  ; user - the user (PHP\_AUTH\_USER) (or '-' if not set);  ; script - the main script called (or '-' if not set);  ; last request cpu - the %cpu the last request consumed  ; it's always 0 if the process is not in Idle state  ; because CPU calculation is done when the request  ; processing has terminated;  ; last request memory - the max amount of memory the last request consumed  ; it's always 0 if the process is not in Idle state  ; because memory calculation is done when the request  ; processing has terminated;  ; If the process is in Idle state, then informations are related to the  ; last request the process has served. Otherwise informations are related to  ; the current request being served.  ; Example output:  ; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ; pid: 31330  ; state: Running  ; start time: 01/Jul/2011:17:53:49 +0200  ; start since: 63087  ; requests: 12808  ; request duration: 1250261  ; request method: GET  ; request URI: /test\_mem.php?N=10000  ; content length: 0  ; user: -  ; script: /home/fat/web/docs/php/test\_mem.php  ; last request cpu: 0.00  ; last request memory: 0  ;  ; Note: There is a real-time FPM status monitoring sample web page available  ; It's available in: /usr/local/share/php/fpm/status.html  ;  ; Note: The value must start with a leading slash (/). The value can be  ; anything, but it may not be a good idea to use the .php extension or it  ; may conflict with a real PHP file.  ; Default Value: not set  ;pm.status\_path = /status  ; The ping URI to call the monitoring page of FPM. If this value is not set, no  ; URI will be recognized as a ping page. This could be used to test from outside  ; that FPM is alive and responding, or to  ; - create a graph of FPM availability (rrd or such);  ; - remove a server from a group if it is not responding (load balancing);  ; - trigger alerts for the operating team (24/7).  ; Note: The value must start with a leading slash (/). The value can be  ; anything, but it may not be a good idea to use the .php extension or it  ; may conflict with a real PHP file.  ; Default Value: not set  ;ping.path = /ping  ; This directive may be used to customize the response of a ping request. The  ; response is formatted as text/plain with a 200 response code.  ; Default Value: pong  ;ping.response = pong  ; The access log file  ; Default: not set  access.log = /var/log/$pool.access.log  ; The access log format.  ; The following syntax is allowed  ; %%: the '%' character  ; %C: %CPU used by the request  ; it can accept the following format:  ; - %{user}C for user CPU only  ; - %{system}C for system CPU only  ; - %{total}C for user + system CPU (default)  ; %d: time taken to serve the request  ; it can accept the following format:  ; - %{seconds}d (default)  ; - %{miliseconds}d  ; - %{mili}d  ; - %{microseconds}d  ; - %{micro}d  ; %e: an environment variable (same as $\_ENV or $\_SERVER)  ; it must be associated with embraces to specify the name of the env  ; variable. Some exemples:  ; - server specifics like: %{REQUEST\_METHOD}e or %{SERVER\_PROTOCOL}e  ; - HTTP headers like: %{HTTP\_HOST}e or %{HTTP\_USER\_AGENT}e  ; %f: script filename  ; %l: content-length of the request (for POST request only)  ; %m: request method  ; %M: peak of memory allocated by PHP  ; it can accept the following format:  ; - %{bytes}M (default)  ; - %{kilobytes}M  ; - %{kilo}M  ; - %{megabytes}M  ; - %{mega}M  ; %n: pool name  ; %o: output header  ; it must be associated with embraces to specify the name of the header:  ; - %{Content-Type}o  ; - %{X-Powered-By}o  ; - %{Transfert-Encoding}o  ; - ....  ; %p: PID of the child that serviced the request  ; %P: PID of the parent of the child that serviced the request  ; %q: the query string  ; %Q: the '?' character if query string exists  ; %r: the request URI (without the query string, see %q and %Q)  ; %R: remote IP address  ; %s: status (response code)  ; %t: server time the request was received  ; it can accept a strftime(3) format:  ; %d/%b/%Y:%H:%M:%S %z (default)  ; The strftime(3) format must be encapsuled in a %{<strftime\_format>}t tag  ; e.g. for a ISO8601 formatted timestring, use: %{%Y-%m-%dT%H:%M:%S%z}t  ; %T: time the log has been written (the request has finished)  ; it can accept a strftime(3) format:  ; %d/%b/%Y:%H:%M:%S %z (default)  ; The strftime(3) format must be encapsuled in a %{<strftime\_format>}t tag  ; e.g. for a ISO8601 formatted timestring, use: %{%Y-%m-%dT%H:%M:%S%z}t  ; %u: remote user  ;  ; Default: "%R - %u %t \"%m %r\" %s"  ;access.format = "%R - %u %t \"%m %r%Q%q\" %s %f %{mili}d %{kilo}M %C%%"  ; The log file for slow requests  ; Default Value: not set  ; Note: slowlog is mandatory if request\_slowlog\_timeout is set  slowlog = /var/log/$pool.log.slow  ; The timeout for serving a single request after which a PHP backtrace will be  ; dumped to the 'slowlog' file. A value of '0s' means 'off'.  ; Available units: s(econds)(default), m(inutes), h(ours), or d(ays)  ; Default Value: 0  request\_slowlog\_timeout = 10s  ; The timeout for serving a single request after which the worker process will  ; be killed. This option should be used when the 'max\_execution\_time' ini option  ; does not stop script execution for some reason. A value of '0' means 'off'.  ; Available units: s(econds)(default), m(inutes), h(ours), or d(ays)  ; Default Value: 0  ;request\_terminate\_timeout = 0  ; Set open file descriptor rlimit.  ; Default Value: system defined value  rlimit\_files = 1024  ; Set max core size rlimit.  ; Possible Values: 'unlimited' or an integer greater or equal to 0  ; Default Value: system defined value  ;rlimit\_core = 0  ; Chroot to this directory at the start. This value must be defined as an  ; absolute path. When this value is not set, chroot is not used.  ; Note: you can prefix with '$prefix' to chroot to the pool prefix or one  ; of its subdirectories. If the pool prefix is not set, the global prefix  ; will be used instead.  ; Note: chrooting is a great security feature and should be used whenever  ; possible. However, all PHP paths will be relative to the chroot  ; (error\_log, sessions.save\_path, ...).  ; Default Value: not set  ;chroot =  ; Chdir to this directory at the start.  ; Note: relative path can be used.  ; Default Value: current directory or / when chroot  chdir = /usr/local/www/wp  ; Redirect worker stdout and stderr into main error log. If not set, stdout and  ; stderr will be redirected to /dev/null according to FastCGI specs.  ; Note: on highloaded environement, this can cause some delay in the page  ; process time (several ms).  ; Default Value: no  catch\_workers\_output = yes  ; Clear environment in FPM workers  ; Prevents arbitrary environment variables from reaching FPM worker processes  ; by clearing the environment in workers before env vars specified in this  ; pool configuration are added.  ; Setting to "no" will make all environment variables available to PHP code  ; via getenv(), $\_ENV and $\_SERVER.  ; Default Value: yes  clear\_env = yes  ; Limits the extensions of the main script FPM will allow to parse. This can  ; prevent configuration mistakes on the web server side. You should only limit  ; FPM to .php extensions to prevent malicious users to use other extensions to  ; execute php code.  ; Note: set an empty value to allow all extensions.  ; Default Value: .php  security.limit\_extensions = .php .php3 .php4 .php5 .php7  ; Pass environment variables like LD\_LIBRARY\_PATH. All $VARIABLEs are taken from  ; the current environment.  ; Default Value: clean env  env[HOSTNAME] = $HOSTNAME  env[PATH] = /usr/local/bin:/usr/bin:/bin  env[TMP] = /tmp  env[TMPDIR] = /tmp  env[TEMP] = /tmp  ; Additional php.ini defines, specific to this pool of workers. These settings  ; overwrite the values previously defined in the php.ini. The directives are the  ; same as the PHP SAPI:  ; php\_value/php\_flag - you can set classic ini defines which can  ; be overwritten from PHP call 'ini\_set'.  ; php\_admin\_value/php\_admin\_flag - these directives won't be overwritten by  ; PHP call 'ini\_set'  ; For php\_\*flag, valid values are on, off, 1, 0, true, false, yes or no.  ; Defining 'extension' will load the corresponding shared extension from  ; extension\_dir. Defining 'disable\_functions' or 'disable\_classes' will not  ; overwrite previously defined php.ini values, but will append the new value  ; instead.  ; Note: path INI options can be relative and will be expanded with the prefix  ; (pool, global or /usr/local)  ; Default Value: nothing is defined by default except the values in php.ini and  ; specified at startup with the -d argument  ;php\_admin\_value[sendmail\_path] = /usr/sbin/sendmail -t -i -f www@my.domain.com  php\_flag[display\_errors] = on  ;php\_admin\_value[error\_log] = /var/log/fpm-php.www.log  php\_admin\_flag[log\_errors] = on  php\_admin\_value[memory\_limit] = 128M |

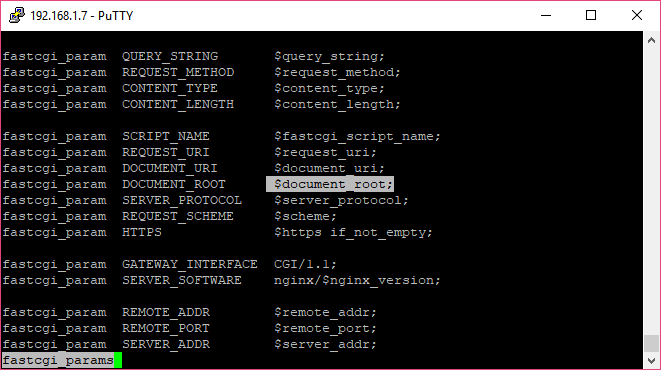
Создадим директорию и установим для нее права:

|  |
| --- |
| mkdir /usr/local/www/wp&&chown www /usr/local/www/wp |

Далее нужно в настройках виртуального хоста поменять переменные для php-fpm.

Имена переменных можно посмотреть в fastcgi\_params:

|  |
| --- |
| less fastcgi\_params |



Далее:

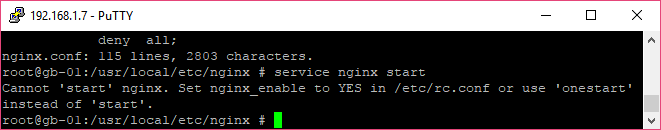
|  |
| --- |
| vi nginx.conf |

Меняем fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name на fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;:

|  |
| --- |
| location ~ \.php$ {  root /usr/local/www/wp;  fastcgi\_pass 127.0.0.1:9000;  fastcgi\_index index.php;  fastcgi\_param SCRIPT\_FILENAME  include fastcgi\_params;  } |

Запускаем сервис:

|  |
| --- |
| service nginx start |

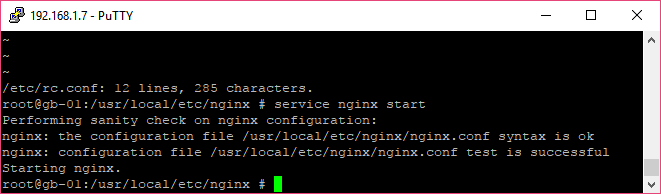


Он просит добавить в /etc/rc.

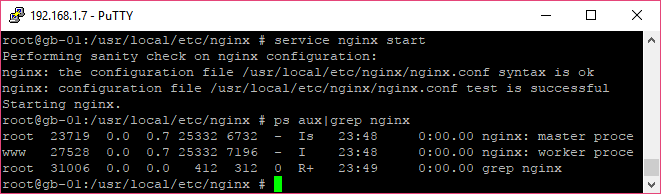
|  |
| --- |
| vi /etc/rc.conf |
|  |

Добавляем (и для php\_fpm сразу же):

|  |
| --- |
| clear\_tmp\_enable="YES"  syslogd\_flags="-ss"  sendmail\_enable="NONE"  hostname="gb-01"  keymap="ru.koi8-r.win.kbd"  ifconfig\_em0="DHCP"  sshd\_enable="YES"  ntpd\_enable="YES"  # Set dumpdev to "AUTO" to enable crash dumps, "NO" to disable  dumpdev="AUTO"  nginx\_enable="YES"  php\_fpm\_enable="YES" |



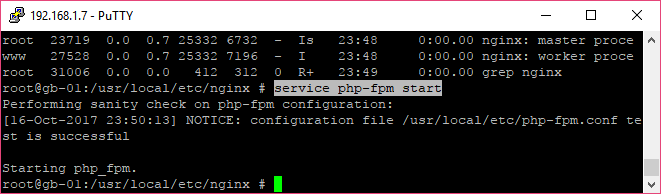
Проверяем:



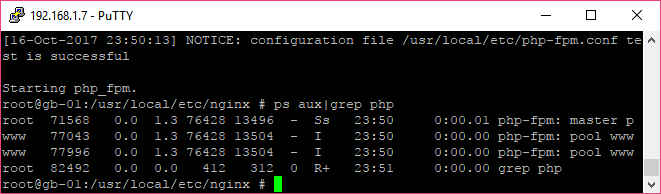
Один мастер-процесс, один воркер.

Cтарутем php-fpm:

|  |
| --- |
| service php-fpm start |



Проверяем:



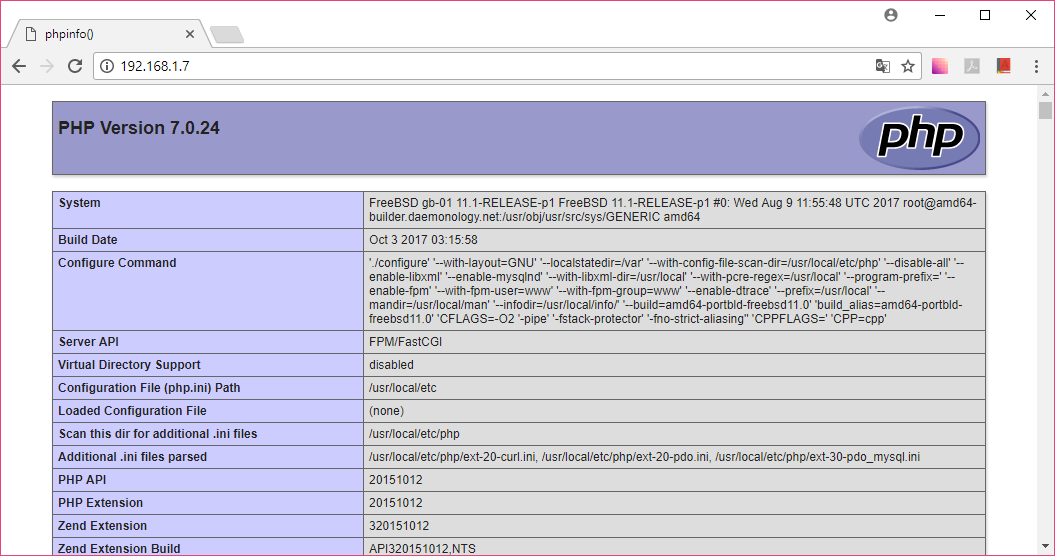
Один мастер-процесс, два пула.

Чтобы проверить, работает ли php-fpm, создадим index.php.

|  |
| --- |
| vi /usr/local/www/wp/index.php |

|  |
| --- |
| <?php  phpinfo();  ?> |

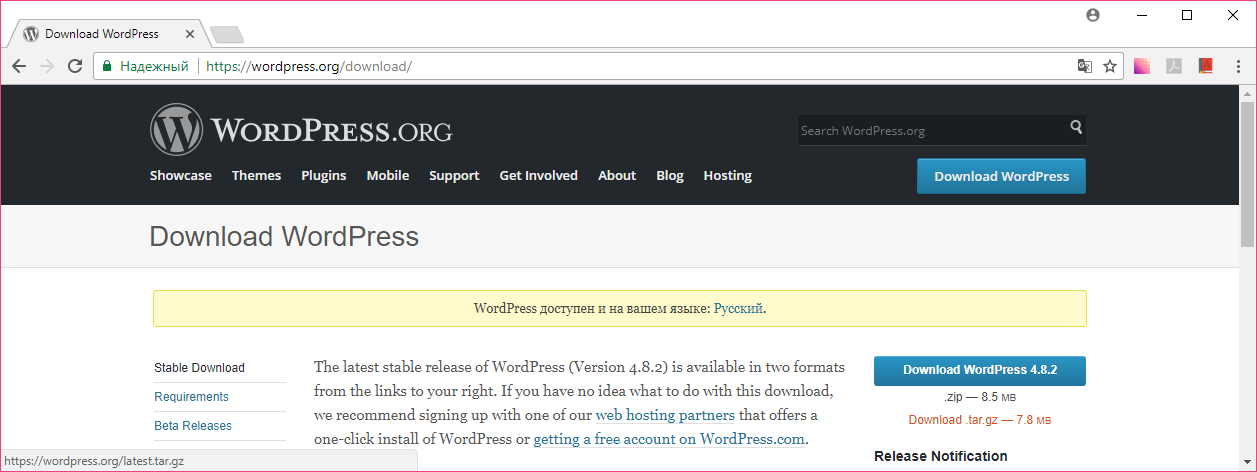
Проверяем. Если все было сделано верно:



## Устанавливаем WordPress

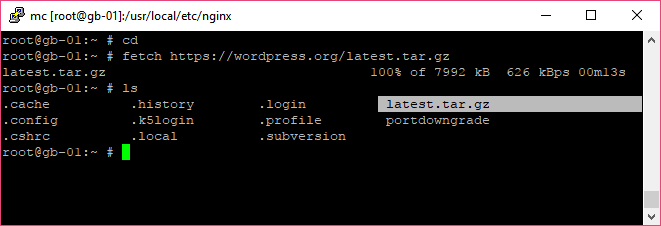
Утилиты wget в FreeBSD нет, но есть похожая утилита fetch.

Заходим на <http://wordpresss.org>, копируем ссылку Download.tar.gz.



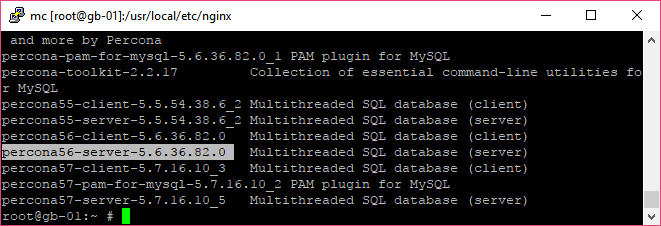
Используем fetch:

|  |
| --- |
| cd  fetch URL-архива |

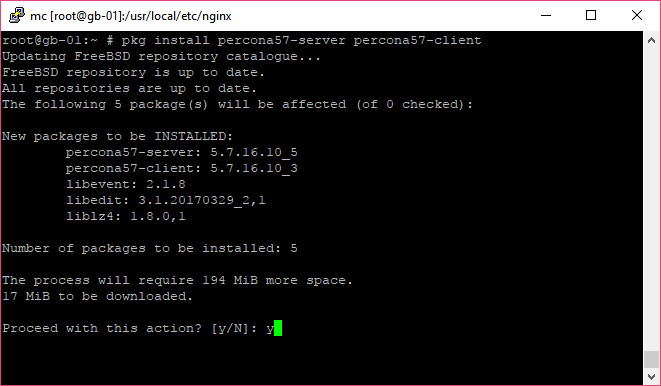


Далее нам нужна СУБД, но мы будем использовать не mysql, а ее форк percona.

|  |
| --- |
| pkg search percona |

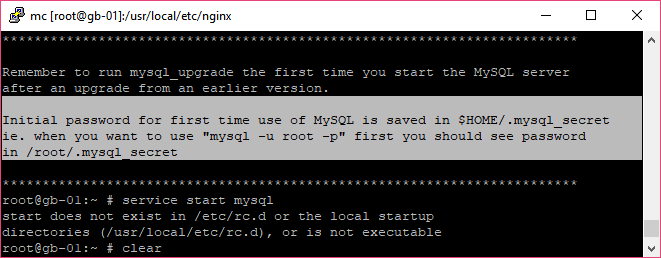


Установим версию 5.7.



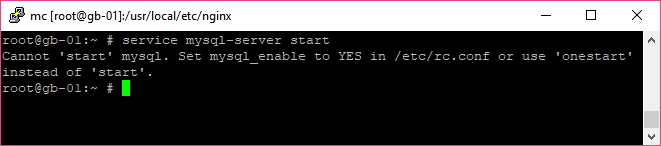
Кроме perconadb, также имеется форк mysql – mariadb – от Monty (Michael Widenius), создателя mysql.

Установилась.



Обратите внимание на сообщение.

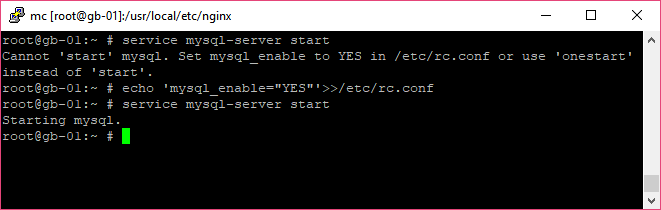
Пытаемся стартовать.

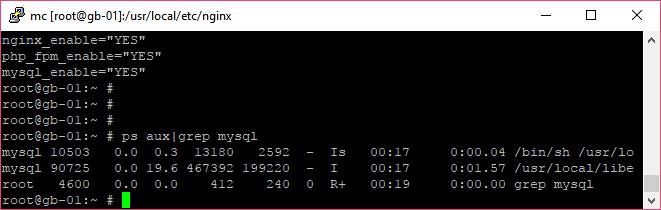


Точно так же нужно добавить строку:

|  |
| --- |
| mysql\_enable="YES" |

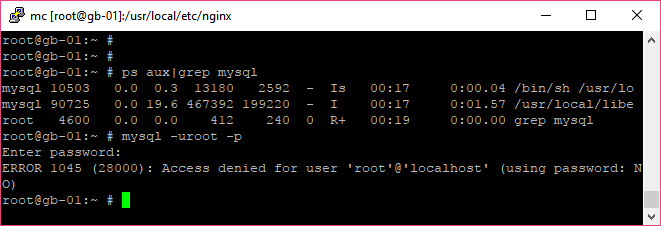
в файл /etc/rc.conf



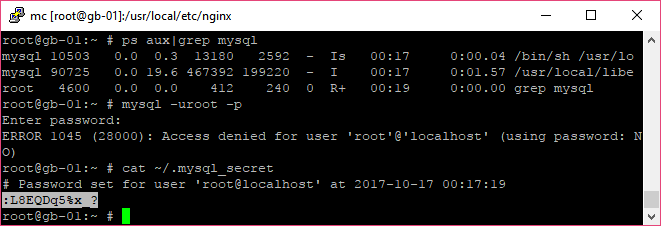


Пытаемся зайти:

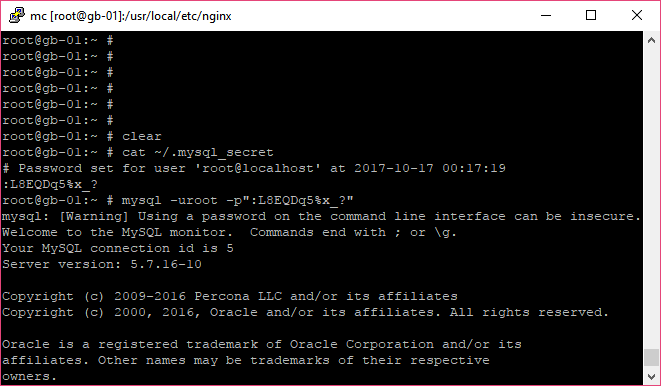
|  |
| --- |
| mysql -uroot -p |



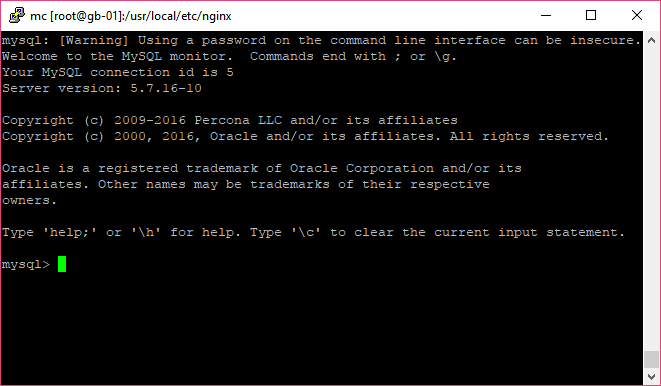
вспоминаем о /root/:



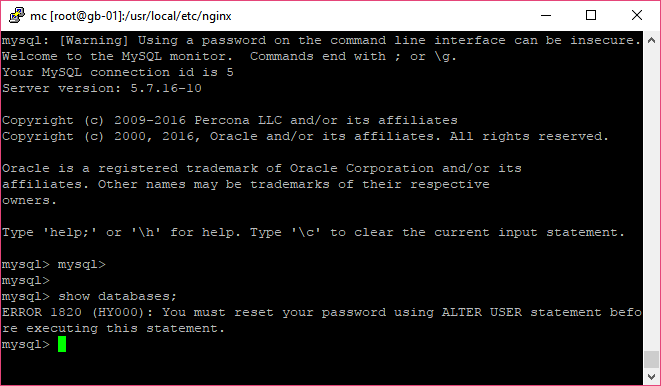
Еще раз пытаемся зайти:



Зашли.



Но поработать не получится:

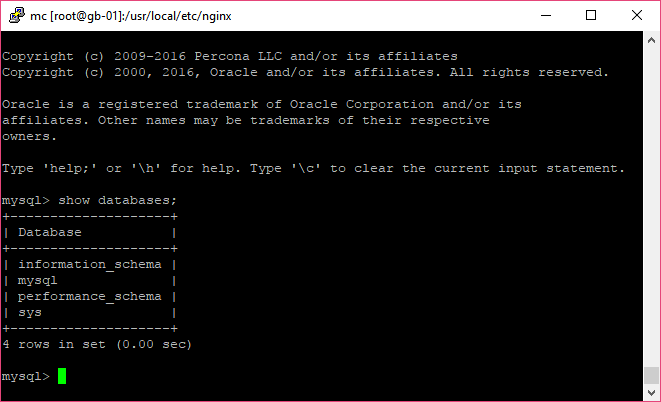


Сначала нужно изменить пароль. Можно сделать так, как предложено, а можно выйти (quit) и изменить его утилитой mysql\_secure\_installation.

Меняем пароль рута, удаляем анонимных пользователей и т. д.

mysql> SET PASSWORD = PASSWORD('your\_new\_password');

Заходим с указанным паролем:



Создадим базу данных для WordPress :

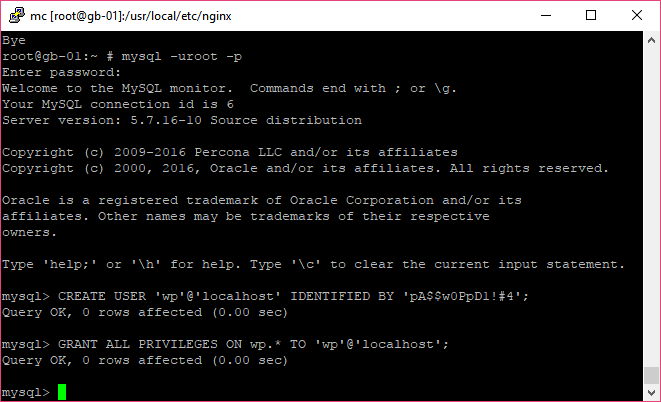
|  |
| --- |
| create database wp; |

Проверяем:

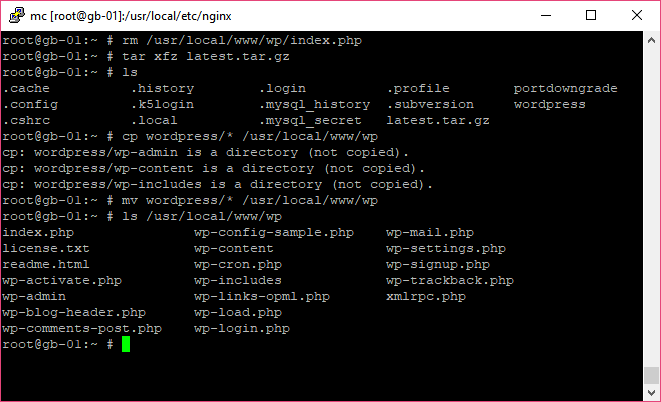
|  |
| --- |
| show databases; |

Создадим пользователя и дадим ему права (пароль используйте свой):

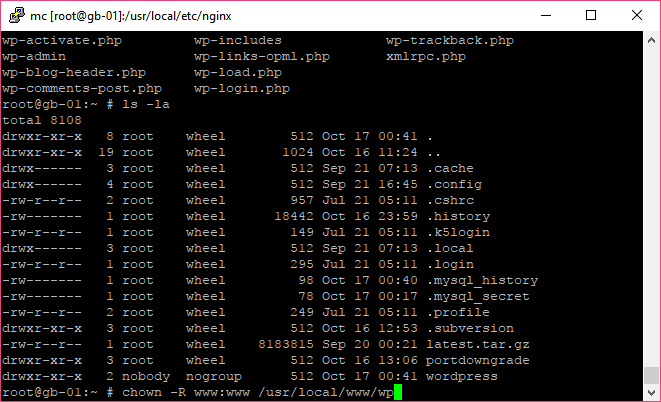
|  |
| --- |
| CREATE USER 'wp'@'localhost' IDENTIFIED BY 'pA$$w0PpD1!#4';  GRANT ALL PRIVILEGES ON wp.\* TO 'wp'@'localhost'; |



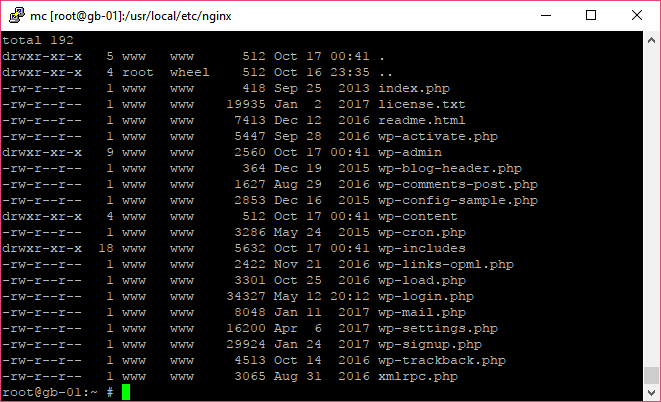
Удалим тестовый index.php, Распакуем Wordpress и переместим в директорию для wp:



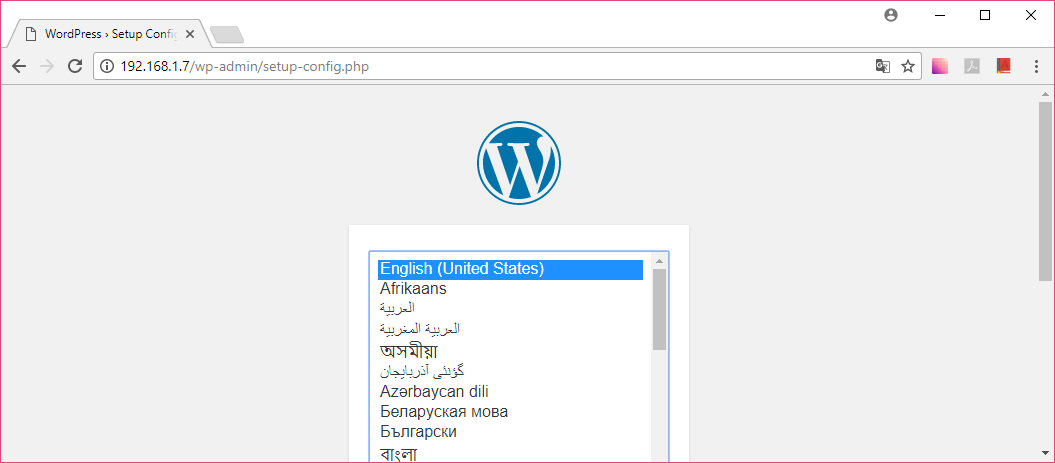
Проверим права и поменяем:



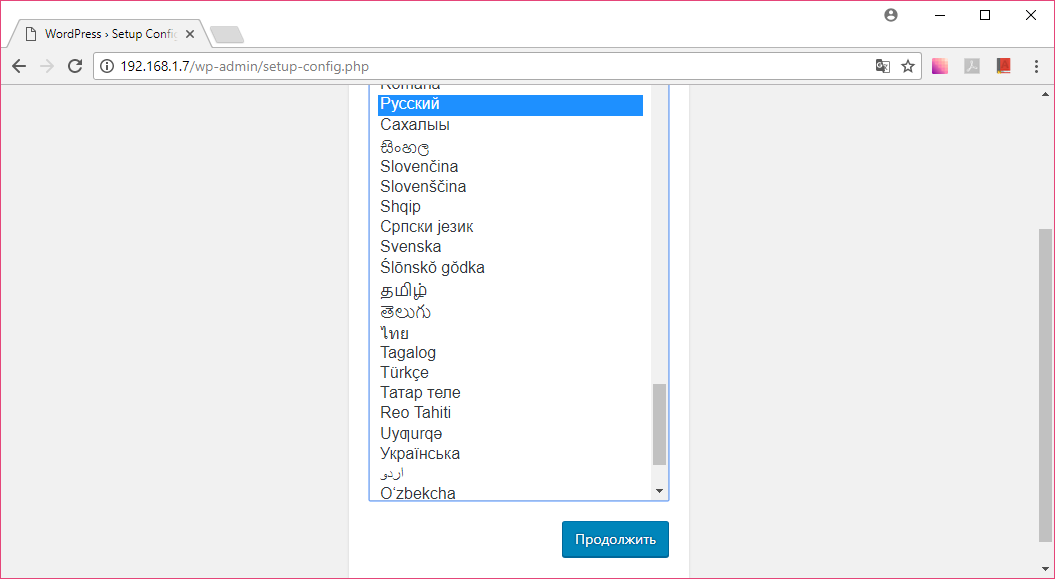
Теперь все верно:



Заходим в браузер:



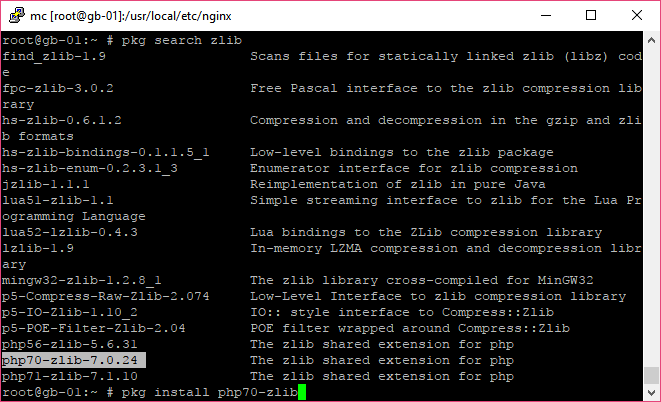
Выберем русский язык:



Ошибка:



Доустановим zlib:

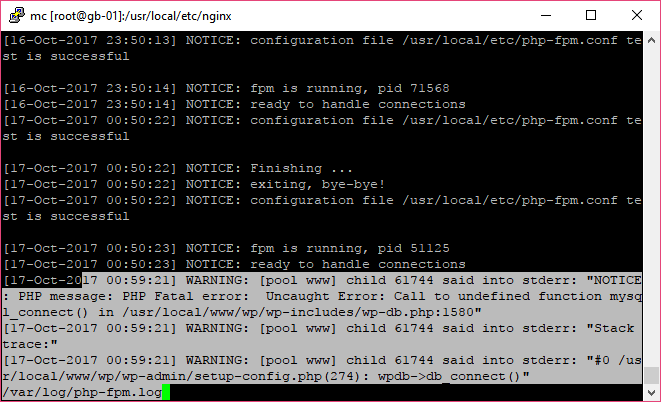


Перезапускаем php-fpm и обновляем страницу.

Вводим значения.

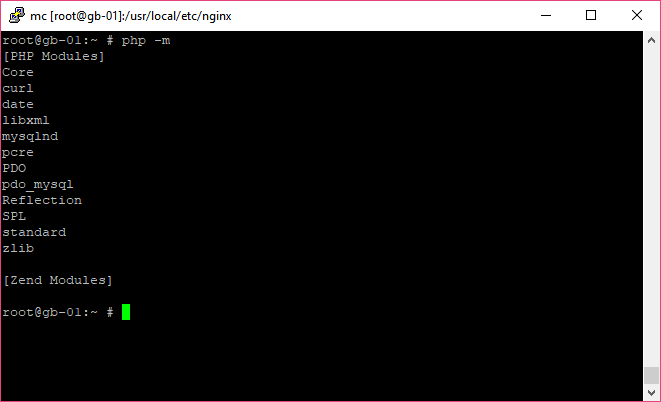
Если возникает ошибка 500, смотрим логи.

|  |
| --- |
| less /var/log/php-fpm.log |

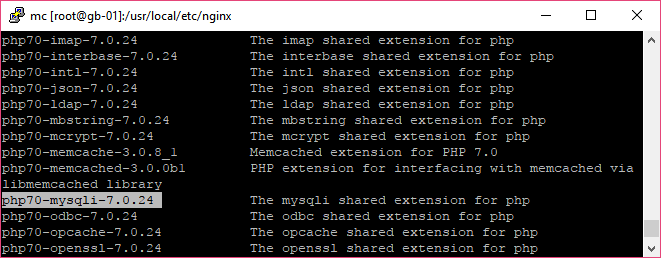


Модули:

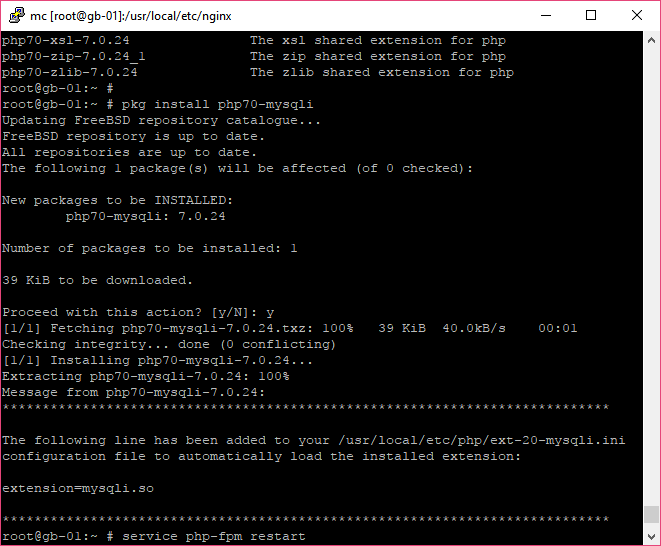
|  |
| --- |
| php -m |



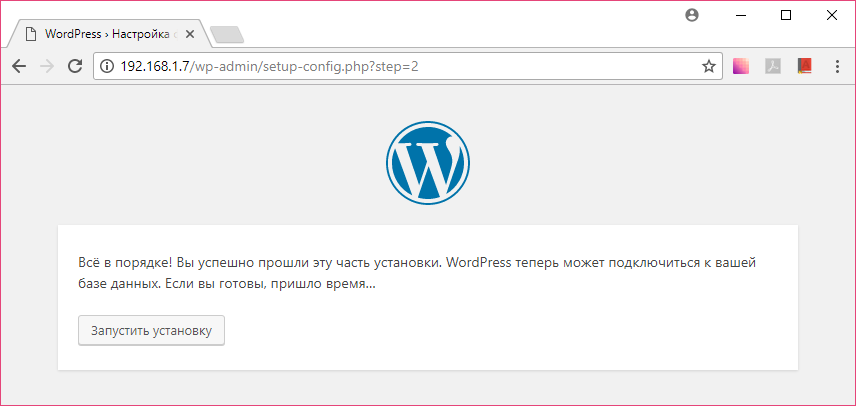
|  |
| --- |
| pkg search php70- |

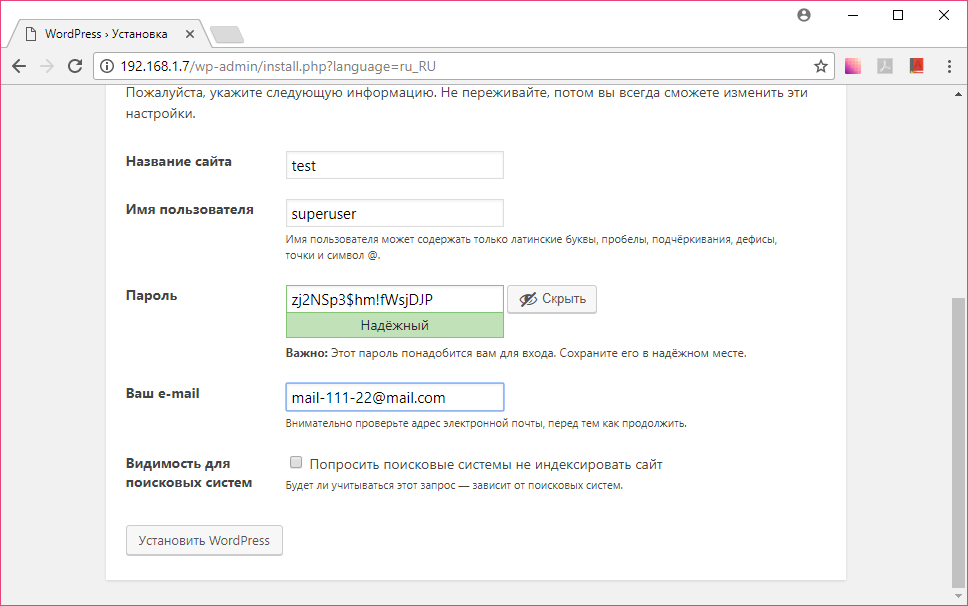


Доустановим mysql и перезапустим php-fpm.



Обновляем:



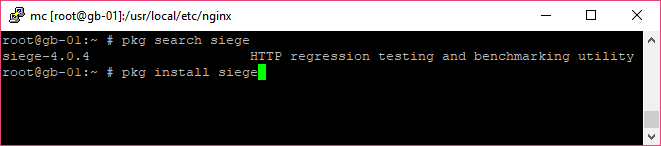


Теперь протестируем, сколько мы можем вынести соединений

## -сУтилита siege

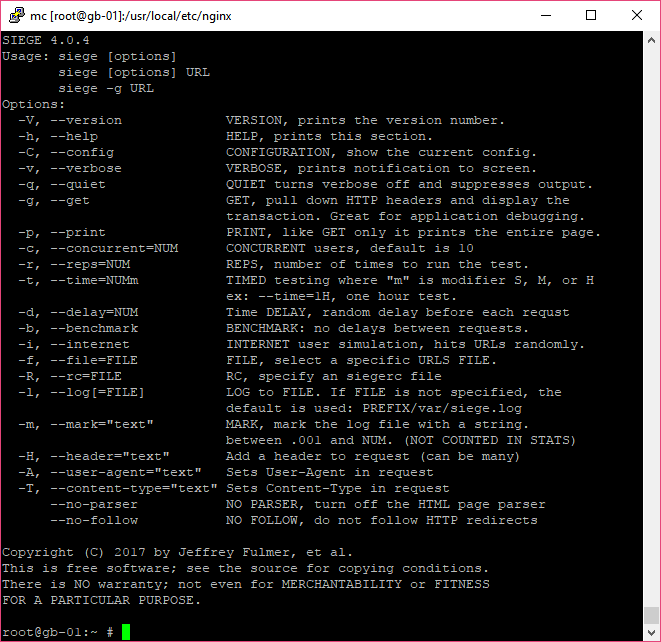
Тестировать будем утилитой siege.

|  |
| --- |
| pkg search siege  pkg install siege |



Использование:

|  |
| --- |
| siege |

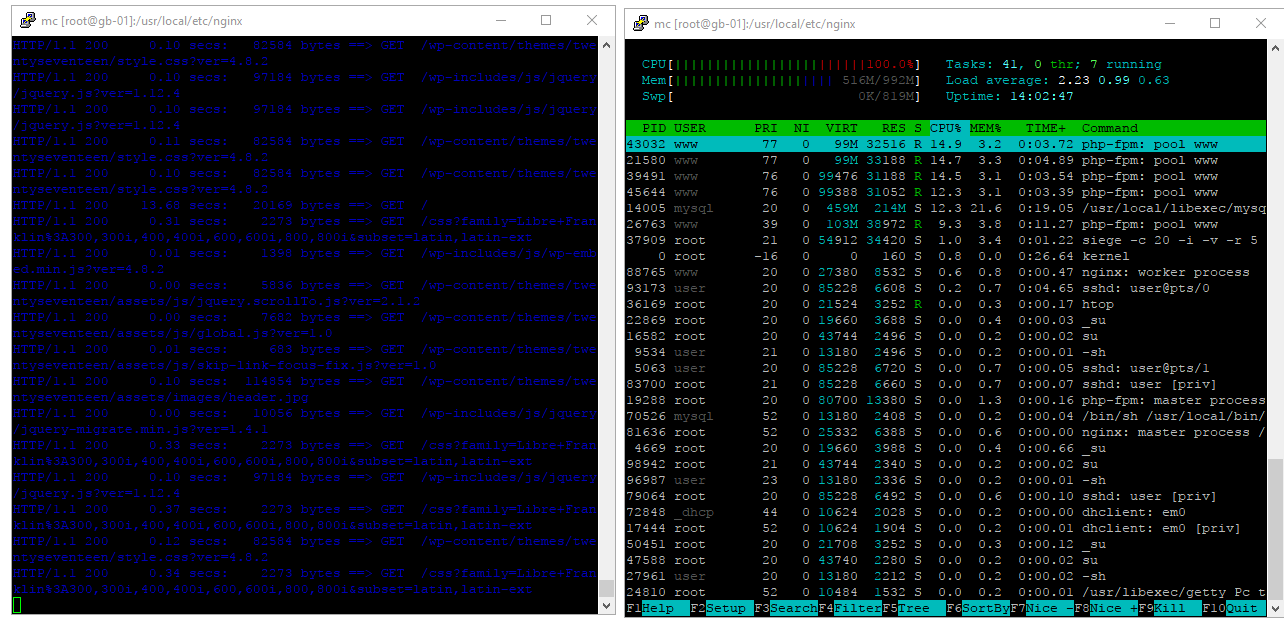


Запустим:

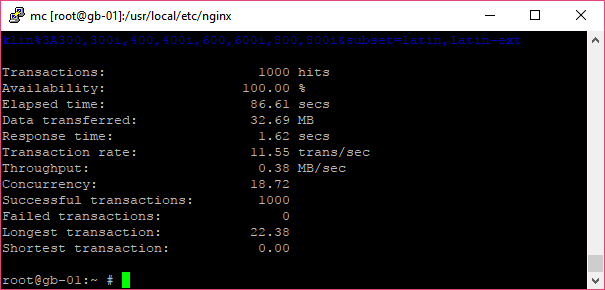
|  |
| --- |
| siege -c 20 -i -v -r 5 http://192.168.1.7 |

* -с – 20 конкурентных пользователей (по умолчанию 10);
* -i – симуляция Интернета (с хитами);
* -v – verbose (вывод детальной информации о каждом обращении к серверу);
* -r 5 – каждый пользователь сделает по пять заходов.

В отдельном окне запустим htop:



Статистика:



Попробуйте увеличить число пользователей, проверьте, как откликается сайт, зайдя через браузер.

# Домашнее задание

1. Установить nginx.
2. Установить php-fpm, wordpress, percona.
3. Установить модули.
4. Выполнить базовую настройку wp.
5. Установить siege и протестировать.

# Дополнительные материалы

1. <http://nginx.org/ru/docs/ngx_core_module.html>
2. <http://php-fpm.org>;
3. <http://wordpresss.org>
4. <https://dev.mysql.com/doc/refman/5.6/en/alter-user.html>

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. <http://nginx.org/ru/docs/ngx_core_module.html>
2. <http://php.net/manual/ru/install.fpm.configuration.php>
3. <https://habrahabr.ru/post/68480/>