МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

А.К. Талашко

Учебно-методические указания к выполнению лабораторных работ по дисциплинам «Распределенные автоматизированные информационные системы», «Распределенные информационно-аналитические системы» для студентов специальностей и направлений

10.05.02 – «Информационная безопасность телекоммуникационных систем»,
10.05.03 – «Информационная безопасность автоматизированных систем»,
10.05.04 – «Информационно-аналитические системы безопасности»

Томск 2023

Оглавление

Лабораторная работа №1 – Стек LEMP

Введение

Набор LEMP — это комплекс программного обеспечения, используемый для обслуживания динамических веб-страниц и веб-приложений. Аббревиатура LEMP обозначает операционную систему Linux с веб-сервером Nginx. Данные серверной части хранятся в базе данных MySQL, а динамическая обработка выполняется PHP.

Установка веб-сервера Nginx

Для показа веб-страниц посетителям сайта мы будем использовать современный и эффективный веб-сервер Nginx.

Все программное обеспечение, используемое в этой процедуре, берется из заданных по умолчанию хранилищ пакетов Ubuntu. Это означает, что мы можем использовать для установки набор управления пакетами apt.

Поскольку в этом сеансе мы будем использовать apt впервые, нужно обновить указатель пакетов вашего сервера. Затем нужно выполнить установку сервера:

```
sudo apt update

sudo apt install nginx
```

В Ubuntu 18.04 настроен запуск Nginx сразу же после установки.

Если вы используете брандмауэр ufw, то вам нужно будет разрешить подключение к Nginx. Nginx регистрируется в ufw после установки, и поэтому процедура довольно простая.

Рекомендуется применять самый ограничивающий профиль, который будет разрешать желаемый трафик. Поскольку в этом модуле вы не настроили SSL для своего сервера, вам нужно будет только разрешить трафик на порту 80.

Для этого введите следующую команду:

```
sudo ufw allow 'Nginx HTTP'
```

Для проверки изменения используйте команду:

```
sudo ufw status
```

Результат выполнения этой команды покажет, что трафик HTTP разрешен:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

После добавления нового правила брандмауэра вы можете проверить, запущен ли сервер, указав в браузере доменное имя вашего сервера или публичный IP-адрес.

Если у вас нет доменного имени, указывающего на ваш сервер, и вы не знаете публичный IP-адрес вашего сервера, вы можете найти его, введя в терминал следующую команду:

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

Команда выведет несколько IP-адресов. Вы можете попробовать каждый из них в своем браузере.

Также вы можете проверить доступность IP-адреса из других мест в интернете:

```
curl -4 icanhazip.com
```

Введите полученный адрес в браузер, и вы попадете на страницу Nginx по умолчанию:

```
http://server_domain_or_IP
```

Установка MySQL для управления данными сайта

Теперь, когда у вас есть веб-сервер, вам нужно установить MySQL (система управления базами данных) для хранения и управления данными вашего сайта.

Для установки MySQL введите следующее:

```
sudo apt install mysql-server
```

Программное обеспечение СУБД MySQL установлено, но его настройка еще не завершена.

Для защиты установки в комплект MySQL входит скрипт, запрашивающий подтверждение небезопасных изменений параметров по умолчанию. Запустите скрипт, введя следующую команду:

```
sudo mysql_secure_installation
```

Скрипт предложит настроить VALIDATE PASSWORD PLUGIN.

Предупреждение. Эту функцию следует активировать при наличии разумных оснований. Если она активирована, MySQL будет отклонять пароли, не соответствующие определенным критериям, и выводить сообщение об ошибке. Это может вызвать проблемы, если вы используете слабый пароль в сочетании с программным обеспечением для автоматической настройки учетных данных пользователя MySQL, например, с пакетами Ubuntu для phpMyAdmin. Оставить проверку отключенной достаточно безопасно, но для входа в базу данных всегда нужно использовать надежные уникальные пароли.

Выберите Y для активации или любой другой вариант, чтобы продолжить без активации этой функции.

```
VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No:
```

Если вы активировали функцию проверки, скрипт также попросит вас задать уровень проверки пароля. Помните, что если вы укажете самый высокий уровень **2**, система будет выводить сообщения об ошибке при попытке установки пароля, который не будет содержать цифры, буквы в верхнем и нижнем регистре и специальные символы, или будет содержать распространенные словарные слова.

```
There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

Затем вам нужно будет отправить и подтвердить пароль пользователя root:

```
Please set the password for root here.
New password:
Re-enter new password:
```

Для всех остальных вопросов нужно выбирать Y и нажимать ENTER в каждом диалоговом окне. В результате будут удалены некоторые анонимные пользователи и тестовая база данных, отключена возможность удаленного входа для пользователя root, и будут загружены новые правила, чтобы СУБД MySQL немедленно использовала внесенные изменения.

В системах Ubuntu с СУБД MySQL 5.7 (и более поздними версиями) для пользователя **root** СУБД MySQL по умолчанию для аутентификации задан плагин auth_socket, а не пароль. Во многих случаях это обеспечивает более высокую безопасность и удобство, однако это также может осложнить ситуацию, если вам нужно предоставить доступ к пользователю внешней программе (например, phpMyAdmin).

Если же вы предпочитаете использовать пароль при подключении к MySQL в качестве пользователя **root**, метод аутентификации нужно изменить с auth_socket на mysql_native_password. Для этого откройте командную строку MySQL через терминал:

```
sudo mysql
```

Затем проверьте, какой метод аутентификации используют ваши аккаунты пользователей MySQL с помощью следующей команды:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

```
Output
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             |                                           | auth_socket           | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
4 rows in set (0.00 sec)
```

В этом примере вы можете видеть, что **root** пользователь действительно использует метод аутентификации с помощью плагина auth_socket. Чтобы

настроить для учетной записи **root** аутентификацию с помощью пароля, выполните следующую команду ALTER USER. Обязательно измените значение password на надежный пароль по вашему выбору:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'password';
```

Затем выполните команду FLUSH PRIVILEGES, которая просит сервер перезагрузить предоставленные таблицы и ввести в действие изменения:

```
FLUSH PRIVILEGES;
```

Проверьте методы аутентификации, применяемые для каждого из ваших пользователей, чтобы подтвердить, что **root**-пользователь больше не использует для аутентификации плагин auth_socket:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

```
 Output
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             | *3636DACC8616D997782ADD0839F92C1571D6D78F | mysql_native_password | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
4 rows in set (0.00 sec)
```

В результатах этого примера можно увидеть, что теперь пользователь `root` СУБД MySQL проходит аутентификацию с помощью пароля. Убедившись в этом на своем сервере, вы можете выйти из оболочки MySQL:

```
exit
```

Примечание. После настройки аутентификации с помощью пароля для пользователя **root** СУБД MySQL вы не сможете получать доступ к MySQL с помощью команды sudo mysql, которая использовалась до этого. Вместо этого нужно запустить следующую команду:

```
mysql -u root -p
```

После ввода заданного пароля вы увидите командную строку MySQL.

Теперь СУБД настроена, и вы можете переходить к установке PHP.

Установка PHP и настройка Nginx для использования процессора PHP

Теперь у вас есть Nginx для обслуживания ваших страниц и MySQL для хранения и управления данными, однако у вас до сих пор не установлено ПО,

которое может генерировать динамический контент. Для этого требуется установить PHP.

Поскольку Nginx не поддерживает нативную обработку PHP, как некоторые другие веб-серверы, вам потребуется установить php-fpm, т.е. «менеджер процессов fastCGI». Мы укажем Nginx передавать запросы PHP в это программное обеспечение для обработки.

Примечание. В зависимости от поставщика облачных услуг вам может потребоваться установить хранилище Ubuntu universe, которое включает бесплатное программное обеспечение и программное обеспечение с открытым исходным кодом, поддерживаемое сообществом Ubuntu, прежде чем устанавливать пакет php-fpm. Для этого можно ввести следующую команду:

```
sudo add-apt-repository universe
```

Установите модуль php-fpm с дополнительным вспомогательным пакетом php-mysql, который позволит PHP взаимодействовать с серверной частью вашей базы данных. При установке будут загружены необходимые файлы ядра PHP. Введите следующее:

```
sudo apt install php-fpm php-mysql
```

Теперь у вас установлены все требуемые компоненты набора LEMP, однако вам нужно внести еще несколько изменений конфигурации, чтобы Nginx использовал процессор PHP для динамического контента.

Это изменение конфигурации выполняется уровне блока сервера (блоки сервера похожи на виртуальные хосты в Apache). Откройте новый файл конфигурации блока сервера в каталоге /etc/nginx/sites-available/. В этом примере новый файл конфигурации блока сервера имеет имя example.com, хотя вы можете использовать любое желаемое имя:

```
sudo nano /etc/nginx/sites-available/example.com
```

Если вам потребуется восстановить конфигурацию по умолчанию, вы можете отредактировать новый файл конфигурации блока сервера, а не изменять используемый по умолчанию файл.

Добавьте в новый файл конфигурации блока следующее содержимое, которое взято с некоторыми модификациями из файла конфигурации блока сервера по умолчанию:

```
/etc/nginx/sites-available/example.com
server {
        listen 80;
        root /var/www/html;
        index index.php index.html index.htm index.nginx-debian.html;
        server_name example.com;

        location / {
                try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
        }

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

Ниже описано действие этих директив и блоков расположения:

- `listen` — определяет, что будет прослушивать порт Nginx. В данном случае он будет прослушивать порт 80, используемый по умолчанию для протокола HTTP.
- `root` — определяет корневой каталог документа, где хранятся файлы, обслуживаемые сайтом.
- `index` — задает для Nginx приоритет обслуживания файлов с именем index.php (при наличии) при запросе файла индекса.
- `server_name` — определяет, какой серверный блок должен использоваться для заданного запроса вашего сервера. **Эта директива должна указывать на доменное имя или публичный IP-адрес вашего сервера.**
- `location / ` — первый блок расположения включает директиву try_files, которая проверяет наличие файлов, соответствующих запросу URI. Если Nginx не сможет найти соответствующий файл, будет возвращена ошибка 404.
- `location ~ \.php$` — этот блок расположения отвечает за фактическую обработку PHP посредством указания Nginx на файл конфигурации fastcgi-php.conf и файл php7.2-fpm.sock file, который объявляет, какой сокет ассоциирован с php-fpm.
- `location ~ /\.ht` — последний блок расположения отвечает за файлы .htaccess, которые Nginx не обрабатывает. При добавлении директивы deny all из файлов .htaccess в корневой каталог документа они не будут выводиться посетителям.

После добавления этого содержания следует сохранить и закрыть файл. Для активации нового серверного блока создайте символическую ссылку от

нового файла конфигурации серверного блока (в каталоге /etc/nginx/sites-available/) на каталог /etc/nginx/sites-enabled/:

```
sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/
```

Затем уберите ссылку на файл конфигурации по умолчанию из каталога /sites-enabled/:

```
sudo unlink /etc/nginx/sites-enabled/default
```

Примечание. Если вам потребуется восстановить конфигурацию по умолчанию, вы можете сделать это посредством воссоздания символической ссылки:

```
sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/
```

Протестируйте новый файл конфигурации на ошибки синтаксиса:

```
sudo nginx -t
```

При появлении сообщений о каких-либо ошибках, вернитесь и повторно проверьте ваш файл, прежде чем продолжать.

Когда вы будете готовы, перезагрузите Nginx для внесения необходимых изменений:

```
sudo systemctl reload nginx
```

Это завершает установку и настройку набора LEMP. Однако будет разумно убедиться, что все компоненты могут связываться друг с другом.

Создание файла PHP для тестовой конфигурации

Теперь набор LEMP должен быть полностью настроен. Вы можете протестировать его, чтобы убедиться, что Nginx может правильно передавать файлы .php на процессор PHP.

Используйте текстовый редактор, чтобы создать тестовый файл PHP с именем info.php в корневом каталоге документа:

```
sudo nano /var/www/html/info.php
```

Введите в новый файл следующие строки: Это корректный код PHP, который будет возвращать информацию о вашем сервере:

```
/var/www/html/info.php
<?php
phpinfo();
```

После завершения редактирования сохраните и закройте файл.

Теперь вы можете открыть эту страницу в браузере, указав в адресной строке доменное имя вашего сервера или публичный IP-адрес и добавив /info.php:

```
http://your_server_domain_or_IP/info.php
```

Вы увидите веб-страницу с информацией о вашем сервере, сгенерированную PHP:

PHP Version 7.2.5-0ubuntu0.18.04.1

System	Linux lemp-1804-2 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Build Date	May 9 2018 17:21:02
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/fpm
Loaded Configuration File	/etc/php/7.2/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/fpm/conf.d
Additional .ini files parsed	/etc/php/7.2/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.2/fpm/conf.d/10-opcache.ini, /etc/php/7.2/fpm/conf.d/10-pdo.ini, /etc/php/7.2/fpm/conf.d/20-calendar.ini, /etc/php/7.2/fpm/conf.d/20-ctype.ini, /etc/php/7.2/fpm/conf.d/20-exif.ini, /etc/php/7.2/fpm/conf.d/20-fileinfo.ini, /etc/php/7.2/fpm/conf.d/20-ftp.ini, /etc/php/7.2/fpm/conf.d/20-gettext.ini, /etc/php/7.2/fpm/conf.d/20-iconv.ini, /etc/php/7.2/fpm/conf.d/20-json.ini, /etc/php/7.2/fpm/conf.d/20-mysqli.ini, /etc/php/7.2/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7.2/fpm/conf.d/20-phar.ini, /etc/php/7.2/fpm/conf.d/20-posix.ini, /etc/php/7.2/fpm/conf.d/20-readline.ini, /etc/php/7.2/fpm/conf.d/20-shmop.ini, /etc/php/7.2/fpm/conf.d/20-sockets.ini, /etc/php/7.2/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.2/fpm/conf.d/20-sysvsem.ini, /etc/php/7.2/fpm/conf.d/20-sysvshm.ini, /etc/php/7.2/fpm/conf.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
with Zend OPcache v7.2.5-0ubuntu0.18.04.1, Copyright (c) 1999-2018, by Zend Technologies

Если ваша страница выглядит таким образом, вам удалось успешно реализовать обработку PHP с помощью Nginx.

После подтверждения того, что Nginx корректно отображает страницу, рекомендуется удалить созданный вами файл, поскольку он может дать неавторизованным пользователям определенную информацию о вашей конфигурации, которая может быть использована при попытке взлома. При необходимости вы всегда сможете восстановить этот файл.

Удалите файл:

```
sudo rm /var/www/html/info.php
```

Теперь на вашем сервере Ubuntu 18.04 имеется полностью настроенный и работающий набор LEMP.

Лабораторная работа №2 – Стек LAMP

Введение

Стек LAMP - это набор программного обеспечения с открытым исходным кодом, которой обычно устанавливается на сервер для отображения динамических веб-сайтов и веб-приложений. Эта аббревиатура обозначает операционную систему Linux с установленным веб-сервером Apache. Данные сайта хранятся в базе данных MySQL, динамический контент обрабатывается с помощью PHP.

Установка Apache и настройка файрвола

Веб-сервер Apache в настоящее время является одним из самых популярных веб-серверов в мире. Он хорошо документирован и используется значительную часть времени с момента создания сети Интернет, что делает его прекрасным выбором для хостинга веб-сайта.

Установим Apache используя менеджер пакетов Ubuntu `apt`:

```
sudo apt update
```

```
sudo apt install apache2
```

Поскольку мы используем команду `sudo`, эти команды будут выполняться с привилегиями root. В процессе установки операционная система запросит ваш пароль пользователя.

После ввода пароля `apt` сообщит, какие пакеты будут установлены и сколько места они займут на диске. Нажмите `Y` и `Enter` для продолжения установки.

Настройка файрвола для разрешения веб-трафика

Теперь убедимся, что ваш файрвол пропускает HTTP и HTTPS трафик. Мы будем исходить из предположения, что вы уже выполнили инструкции по первичной настройке сервера и включили файрвол UFW. Для начала убедимся, что UFW имеет профиль для Apache следующей командой:

```
sudo ufw app list
```

```
Вывод
```

```
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

Проверим настройку профиля `Apache Full`, она должна разрешать трафик для портов `80` и `443`:

```
sudo ufw app info "Apache Full"
```

```
Вывод
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent
Apache web server.

Ports:
  80,443/tcp
```

Разрешим входящий HTTP и HTTPS трафик для этого профиля:

```
sudo ufw allow in "Apache Full"
```

Проверить результат установки можно набрав в вашем веб-браузере публичный IP адрес вашего сервера (если вы еще не знаете, как найти публичный IP адрес вашего сервера, смотрите следующий раздел этой статьи):

```
http://IP_адрес_вашего_сервера
```

Вы увидите страницу Apache, отображаемую по умолчанию для информации и целей тестирования. Она должна выглядеть похожим образом:

Если вы видите эту страницу, ваш веб-сервер корректно установлен и доступен через файрвол.

Как найти публичный IP адрес вашего сервера

Если вы не знаете публичный IP адрес вашего сервера, его можно определить несколькими способами. Обычно, это адрес, который вы используете для соединения с вашим сервером по SSH.

Определить этот адрес можно с помощью командной строки. Сначала используйте инструмент `iproute2` для получения вашего адреса набрав следующую команду:

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

Результатом выполнения этой команды будут две или три строки, содержащие корректный адрес. Ваш компьютер, возможно, сможет использовать только один из них, поэтому попробуйте каждый вариант.

В качестве альтернативы можно узнать, как `curl` видит ваш сервер. Это можно сделать следующим образом:

```
sudo apt install curl
```

```
curl http://icanhazip.com
```

В независимости от метода, который вы использовали для получения своего IP адреса, вы можете использовать этот IP адрес для доступа к серверу через адресную строку веб-браузера.

Установка MySQL

Теперь, когда наш веб-сервер установлен и запущен, пора установить MySQL. MySQL это система управления базами данных. Она организует и обеспечит доступ к базам данных, в которых ваш сайт может хранить информацию.

Мы можем вновь использовать `apt` для загрузки и установки программного обеспечения:

```
sudo apt install mysql-server
```

Обратите внимание: В данном случае вам нет необходимости предварительно выполнять команду `sudo apt update`, т.к. мы выполняли ее недавно при установке Apache, и индекс пакетов на вашем компьютере уже должен быть обновлен.

Вам будет показан список пакетов, которые будут установлены, а также сколько места на диске они займут. Нажмите `Y` для продолжения установки.

После завершения установки нам потребуется выполнить некоторые дополнительные команды, чтобы наше окружение MySQL было настроено безопасным образом. Введите следующую команду:

```
sudo mysql_secure_installation
```

В результате выполнения этой команды вам будет предложено настроить плагин валидации паролей (`VALIDATE PASSWORD PLUGIN`).

Внимание: Решение включать плагин валидации паролей или нет носит субъективный характер. При включении все пароли, которые не удовлетворяют определённым критериям безопасности, будут отвергаться MySQL с сообщением об ошибке. Это может вызывать проблемы, если вы

используете “слабые” пароли совместно с программным обеспечением, которое конфигурирует профили пользователей MySQL, например, пакеты Ubuntu для phpMyAdmin. Вы можете оставить валидацию паролей отключенной, но в этом случае вам следует всегда использовать “сильные” уникальные пароли для пользователей базы данных.

Введите `Y` для включения плагина или что-нибудь другое для продолжения без его включения:

```
VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No:
```

Если вы включили валидацию паролей, вам будет предложено установить уровень надёжности паролей при валидации. Имейте в виду, что при выборе значения `2` (самый строгий уровень валидации), вы будете получать ошибки при попытке задать пароль без цифр, букв в верхнем и нижнем регистре, а также без специальных символов, а также при попытке использовать пароль, основанный на распространённых словах, которые уязвимы для подбора паролей по словарю.

```
There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

Вне зависимости от того, включили вы плагин валидации паролей или нет, далее вам будет предложено задать пароль для пользователя **root** для MySQL. Это административный аккаунт пользователя в MySQL, который имеет повышенные привилегии. Вы можете рассматривать его, как аналог пользователя root для самого сервера (с той лишь разницей, что это аккаунт для MySQL). Задайте сильный уникальный пароль, не оставляйте пароль пустым.

Если вы включили валидацию паролей, вам будет показан уровень надёжности заданного вами ранее пароля root пользователя, а также вам будет предложено изменить этот пароль. Если вы не хотите менять пароль, введите `N` или “no”:

```
Using existing password for root.

Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : n
```

На все последующие вопросы просто вводите `Y` и нажимайте клавишу `ENTER` для выбора настроек по умолчанию. При этом удалятся некоторые тестовые пользователи и базы данных, будет отключена возможность удаленного доступа с учетной записью root-пользователя, и все изменения будут немедленно применены в MySQL.

Обратите внимание на то, что на серверах Ubuntu, использующих MySQL 5.7 (и более поздние версии), root пользователь в MySQL настроен таким образом, что его аутентификация по умолчанию происходит с помощью плагина `auth_socket`, а не с помощью пароля. Это во многих случаях повышает безопасность, но, в то же время, может усложнить настройку доступа к root пользователю для некоторых программ (например, phpMyAdmin).

Если вы хотите настроить root пользователя на использование пароля, вам необходимо изменить метод аутентификации с `auth_socket` на `mysql_native_password`. Для того, чтобы это сделать, войдите в оболочку MySQL в терминале:

```
sudo mysql
```

Далее просмотрите метод аутентификации для каждого из ваших пользователей MySQL с помощью следующей команды:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

```
Вывод
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             |                                           | auth_socket           | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
4 rows in set (0.00 sec)
```

В этом примере ваш пользователь root использует аутентификацию с помощью плагина `auth_socket`. Для изменения этой настройки на использование пароля используйте следующую команду `ALTER USER`. Не забудьте изменить `password` на ваш сильный пароль:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'password';
```

Далее выполните команду `FLUSH PRIVILEGES`, которая применит внесённые изменения:

```
FLUSH PRIVILEGES;
```

Проверьте методы авторизации для пользователей ещё раз для того, чтобы убедиться, что пользователь root более не использует плагин `auth_socket` для авторизации:

```
SELECT user,authentication_string,plugin,host FROM mysql.user;
```

```
Вывод
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             | *3636DACC8616D997782ADD0839F92C1571D6D78F | mysql_native_password | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
4 rows in set (0.00 sec)
```

Как можно видеть на представленном выводе теперь **root** пользователь MySQL аутентифицируется с использованием пароля. После того, как мы в этом убедились, можно выйти из оболочки MySQL:

```
exit
```

Теперь ваша система управления базами данных установлена и мы можем двигаться дальше и установить PHP, последний компонент LAMP.

Установка PHP

PHP это компонент, который будет обрабатывать код для отображения динамического контента. Он может запускать скрипты, подключаться к нашим базам данных MySQL для получения информации и передавать обработанный контент в наш веб-сервер для отображения.

Мы можем вновь воспользоваться менеджером пакетов `apt` для установки компонентов. Мы также добавим некоторые вспомогательные пакеты, чтобы код на PHP мог работать с нашим сервером Apache, а также обращаться к базе данных MySQL:

```
sudo apt install php libapache2-mod-php php-mysql
```

Данная команда должна установить PHP без каких либо проблем. Вскоре мы это проверим.

В большинстве случаев, мы захотим изменить способ, который Apache использует для поиска файлов при запросе директории. На данный момент, если пользователь запрашивает директорию с сервера, Apache в первую очередь будет искать файл с названием `index.html`. Мы хотим, чтобы наш веб-сервер отдавал предпочтение PHP файлам, так что мы настроим Apache таким образом, чтобы сначала он искал файлы `index.php`.

Для этого введите следующую команду, чтобы открыть файл `dir.conf` в текстовом редакторе с привилегиями пользователя root:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Содержимое файла будет выглядеть следующим образом:

```
/etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>
```

Мы хотим переместить индексный файл PHP `index.php`, выделенный выше, на первое место после спецификации `DirectoryIndex` следующим образом:

```
/etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm
</IfModule>
```

После внесения правок сохраните и закройте файл сочетанием клавиш `CTRL-X`. Вам придется подтвердить операцию вводом `Y` и затем нажатием клавиши `ENTER` для подтверждения места сохранения файла.

После этого нам необходимо перезапустить Apache для применения внесенных изменений. Вы можете сделать это при помощи команды:

```
sudo systemctl restart apache2
```

Мы также можем проверить статус сервиса `apache2` используя `systemctl`:

```
sudo systemctl status apache2
```

Пример Вывода

```
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Tue 2018-04-23 14:28:43 EDT; 45s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 13581 ExecStop=/etc/init.d/apache2 stop (code=exited, status=0/SUCCESS)
  Process: 13605 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)
    Tasks: 6 (limit: 512)
   CGroup: /system.slice/apache2.service
           ├─13623 /usr/sbin/apache2 -k start
           ├─13626 /usr/sbin/apache2 -k start
           ├─13627 /usr/sbin/apache2 -k start
           ├─13628 /usr/sbin/apache2 -k start
           ├─13629 /usr/sbin/apache2 -k start
           └─13630 /usr/sbin/apache2 -k start
```

Чтобы расширить функциональность PHP, мы можем установить некоторые дополнительные модули. Для просмотра доступных опций для модулей и библиотек PHP вы можете передать результат `apt search` в `less`, которая позволит вам проматывать вывод:

```
apt search php- | less
```

Используйте клавиши стрелок вверх и вниз для проматывания списка, для выхода нажмите `Q`.

В результате вам будут показаны все опциональные компоненты, которые можно установить, сопровождаемые кратким описанием для каждого:

```
bandwidthd-pgsql/bionic 2.0.1+cvs20090917-10ubuntu1 amd64
  Tracks usage of TCP/IP and builds html files with graphs

bluefish/bionic 2.2.10-1 amd64
  advanced Gtk+ text editor for web and software development

cacti/bionic 1.1.38+ds1-1 all
  web interface for graphing of monitoring systems

ganglia-webfrontend/bionic 3.6.1-3 all
  cluster monitoring toolkit - web front-end

golang-github-unknwon-cae-dev/bionic 0.0~git20160715.0.c6aac99-4 all
  PHP-like Compression and Archive Extensions in Go

haserl/bionic 0.9.35-2 amd64
  CGI scripting program for embedded environments

kdevelop-php-docs/bionic 5.2.1-1ubuntu2 all
  transitional package for kdevelop-php

kdevelop-php-docs-l10n/bionic 5.2.1-1ubuntu2 all
  transitional package for kdevelop-php-l10n
...
:
```

Чтобы получить больше информации по каждому модулю, вы можете поискать в Интернете или посмотреть полное описание пакета при помощи команды:

```
apt show package_name
```

Ответ будет содержать много текста, среди которого есть поле `Description`. Именно оно и будет содержать расширенное описание функциональности, предоставляемой модулем.

Например, чтобы узнать назначение модуля `php-cli`, мы можем выполнить команду:

```
apt show php-cli
```

Помимо большого количества прочей информации, вы увидите следующее:

```
Вывод
...
Description: command-line interpreter for the PHP scripting language (default)
 This package provides the /usr/bin/php command interpreter, useful for
 testing PHP scripts from a shell or performing general shell scripting tasks.
 .
 PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used
 open source general-purpose scripting language that is especially suited
 for web development and can be embedded into HTML.
 .
 This package is a dependency package, which depends on Ubuntu's default
 PHP version (currently 7.2).
...
```

Если после изучение вы решили, что хотите установить пакет, вы можете сделать это используя команду `apt install` как мы делали ранее при установке другого программного обеспечения.

Если мы решили, что хотим установить `php-cli`, мы можем ввести команду:

```
sudo apt install php-cli
```

Для установки сразу нескольких модулей, вы можете перечислить их через пробелы следом за командой `apt install` следующим образом:

```
sudo apt install package1 package2 ...
```

Теперь ваш стек LAMP установлен и сконфигурирован. Однако перед внесением любых других изменений и перед установкой приложений нам ещё стоит протестировать настройку PHP на случай возможных проблем.

Тестирование работы PHP на вашем веб-сервере

Чтобы проверить, что наша система сконфигурирована должным образом, мы можем создать простой PHP скрипт. Назовём этот скрипт `info.php`. Чтобы Apache нашел и обработал файл корректно, он должен быть сохранен в специальной директории, которая называется “web root”.

В Ubuntu 18.04 данная директория находится по адресу `/var/www/html/`.

Мы можем создать там файл введя следующую команду:

```
sudo nano /var/www/html/info.php
```

Откроется пустой файл. Введём в файл следующий текст, который является корректным PHP кодом:

info.php
```
<?php
phpinfo();
?>
```

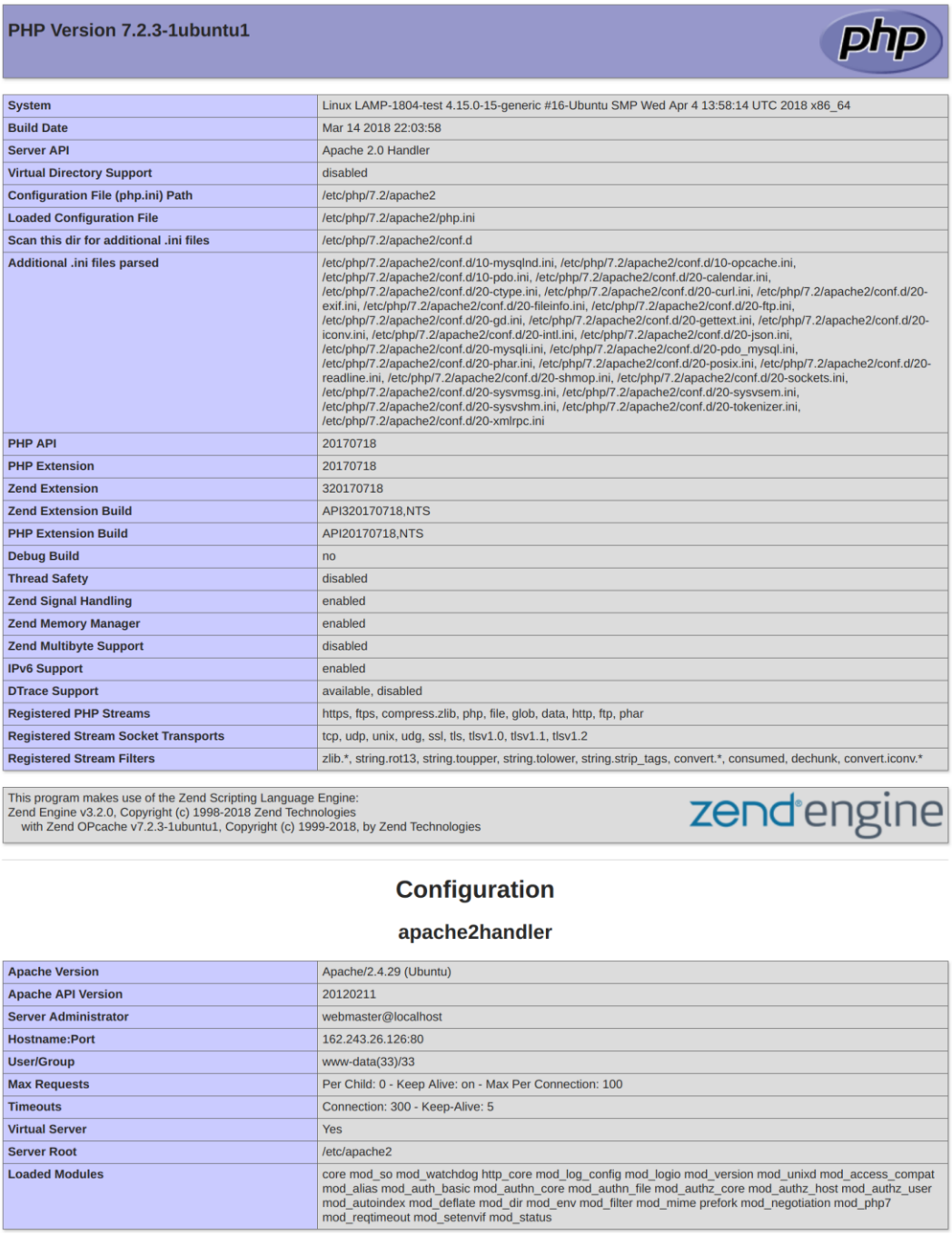
После внесения изменений сохраните и закройте файл.

Теперь мы можем проверить, может ли веб-сервер корректно отображать контент, сгенерированный PHP скриптом. Для проверки нам просто нужно открыть данную страницу в веб-браузере. Вам снова потребуется публичный IP-адрес сервера.

Откроем этот адрес:

```
http://IP_адрес_вашего_сервера/info.php
```

Страница, на которую вы попадете, должна выглядеть похожим образом:



PHP Version 7.2.3-1ubuntu1

System	Linux LAMP-1804-test 4.15.0-15-generic #16-Ubuntu SMP Wed Apr 4 13:58:14 UTC 2018 x86_64
Build Date	Mar 14 2018 22:03:58
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-curl.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-intl.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini, /etc/php/7.2/apache2/conf.d/20-xmlrpc.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
with Zend OPcache v7.2.3-1ubuntu1, Copyright (c) 1999-2018, by Zend Technologies

Configuration

apache2handler

Apache Version	Apache/2.4.29 (Ubuntu)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	162.243.26.126:80
User/Group	www-data(33)/33
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog http_core mod_log_config mod_logio mod_version mod_unixd mod_access_compat mod_alias mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime prefork mod_negotiation mod_php7 mod_reqtimeout mod_setenvif mod_status

Данная страница содержит информацию о вашем сервере с точки зрения PHP. Она полезна для отладки и чтобы удостовериться в корректности применения настроек.

Если все прошло успешно, значит ваш PHP работает корректно.

Вы, возможно, захотите удалить этот файл после теста, т.к. он может дать информацию о вашем сервере неавторизованным пользователям. Для удаления файла введите команду:

```
sudo rm /var/www/html/info.php
```

Лабораторная №3 – Фреймворк Laravel

Введение

Фреймворк Laravel пользуется популярностью за счет ряда факторов:

- он объектно-ориентированный;
- соблюдается модель MVC;
- доступна поддержка нескольких баз данных.

Кроме того, в нем доступна масса инструментов для развертывания приложений и упрощения веб-разработки. Рассмотрим, как выглядит установка Laravel на Ubuntu.

Подготовка к установке

Чтобы запустить на веб-сервере Laravel, в Ubuntu необходимо удовлетворить ряд зависимостей. Все необходимые для работы компоненты имеются в наличии на виртуальной машине Laravel Homestead. Поэтому разработчики рекомендуют использовать именно ее для запуска на локальной машине.

Тем не менее, развертывать и тестировать приложения в основном придется на веб-серверах с Linux, а не на Homestead. Чтобы установить Laravel на таком веб-сервере, на нем должны быть установлена версия PHP >= 7.2.0 и следующие расширения PHP:

- BCMath PHP Extension;
- Ctype PHP Extension;
- JSON PHP Extension;
- Mbstring PHP Extension;
- OpenSSL PHP Extension;
- PDO PHP Extension;
- Tokenizer PHP Extension;
- XML PHP Extension.

В Ubuntu эти зависимости можно установить при помощи команды в терминале:

```
sudo apt install php php-mysql php-mbstring php-tokenizer php-xml
php-json php-common
```

```
[redacted]-vPro:~$ sudo apt install php php-mysql php-mbstring php-tokenizer php-xml php-json php-common
[sudo] пароль для [redacted]:
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Заметьте, вместо «php-tokenizer» выбирается «php7.0-common»
Следующие пакеты устанавливались автоматически и больше не требуются:
  dconf-cli gir1.2-ibus-1.0 libibus-1.0-5 python-glade2 python-gobject-2 python-gtk2
Для их удаления используйте «sudo apt autoremove».
Будут установлены следующие дополнительные пакеты:
  php7.0 php7.0-cli php7.0-fpm php7.0-json php7.0-mbstring php7.0-mysql php7.0-opcache php7.0-readline php7.0-xml
Предлагаемые пакеты:
  php-pear
Следующие НОВЫЕ пакеты будут установлены:
  php php-common php-json php-mbstring php-mysql php-xml php7.0 php7.0-cli php7.0-common php7.0-fpm php7.0-json
  php7.0-mbstring php7.0-mysql php7.0-opcache php7.0-readline php7.0-xml
Обновлено 0 пакетов, установлено 16 новых пакетов, для удаления отмечено 0 пакетов, и 53 пакетов не обновлено.
Необходимо скачать 4.247 kB архивов.
После данной операции объём занятого дискового пространства возрастёт на 16,6 MB.
Хотите продолжить? [Д/н] Д
```

Об успешной установке будут свидетельствовать следующие строки в терминале:

```
Creating config file /etc/php/7.0/mods-available/mysqli.ini with new version

Creating config file /etc/php/7.0/mods-available/pdo_mysql.ini with new version
Настраивается пакет php-mysql (1:7.0+35ubuntu6.1) …
Настраивается пакет php7.0-xml (7.0.33-0ubuntu0.16.04.9) …

Creating config file /etc/php/7.0/mods-available/dom.ini with new version

Creating config file /etc/php/7.0/mods-available/simplexml.ini with new version

Creating config file /etc/php/7.0/mods-available/wddx.ini with new version

Creating config file /etc/php/7.0/mods-available/xml.ini with new version

Creating config file /etc/php/7.0/mods-available/xmlreader.ini with new version

Creating config file /etc/php/7.0/mods-available/xmlwriter.ini with new version

Creating config file /etc/php/7.0/mods-available/xsl.ini with new version
Настраивается пакет php-xml (1:7.0+35ubuntu6.1) …
Обрабатываются триггеры для systemd (229-4ubuntu21.23) …
Обрабатываются триггеры для ureadahead (0.100.0-19.1) …
Обрабатываются триггеры для php7.0-fpm (7.0.33-0ubuntu0.16.04.9) …
[redacted]-vPro:~$
```

Установка через Composer

Чтобы работать с Laravel потребуется предварительно установить Composer, поскольку фреймворк управляет с его помощью своими зависимостями:

```
sudo apt install composer
```

```
[redacted]-vPro:~$ sudo apt install composer
[sudo] пароль для [redacted]:
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
  dconf-cli gir1.2-ibus-1.0 libibus-1.0-5 python-glade2 python-gobject-2 python-gtk2
Для их удаления используйте «sudo apt autoremove».
Будут установлены следующие дополнительные пакеты:
  javascript-common jsonlint libjs-excanvas mercurial mercurial-common php-cli-prompt php-composer-semver
  php-composer-spdx-licenses php-json-schema php-symfony-console php-symfony-filesystem php-symfony-finder
  php-symfony-process
Предлагаемые пакеты:
  php-zip apache2 | lighttpd | httpd qct kdiff3 | kdiff3-qt | kompare | meld | tkcvs | mgdiff wish python-mysqldb
  python-pygments python-openssl php-symfony-event-dispatcher php-psr-log
Следующие НОВЫЕ пакеты будут установлены:
  composer javascript-common jsonlint libjs-excanvas mercurial mercurial-common php-cli-prompt
  php-composer-semver php-composer-spdx-licenses php-json-schema php-symfony-console php-symfony-filesystem
  php-symfony-finder php-symfony-process
Обновлено 0 пакетов, установлено 14 новых пакетов, для удаления отмечено 0 пакетов, и 53 пакетов не обновлено.
Необходимо скачать 2.421 kB архивов.
После данной операции объём занятого дискового пространства возрастёт на 13,2 MB.
Хотите продолжить? [Д/н] Д
```

После установки появится сообщение об автоматической настройке следующих пакетов:

```
Распаковывается mercurial-common (3.7.3-1ubuntu1.2) …
Выбор ранее не выбранного пакета mercurial.
Подготовка к распаковке …/mercurial_3.7.3-1ubuntu1.2_amd64.deb …
Распаковывается mercurial (3.7.3-1ubuntu1.2) …
Обрабатываются триггеры для man-db (2.7.5-1) …
Настраивается пакет php-symfony-console (2.7.10-0ubuntu2) …
Настраивается пакет php-symfony-filesystem (2.7.10-0ubuntu2) …
Настраивается пакет php-symfony-finder (2.7.10-0ubuntu2) …
Настраивается пакет php-symfony-process (2.7.10-0ubuntu2) …
Настраивается пакет php-json-schema (1.6.1-1build1) …
Настраивается пакет php-composer-spdx-licenses (1.1.2-1build1) …
Настраивается пакет php-composer-semver (1.2.0-1build1) …
Настраивается пакет jsonlint (1.4.0-1build1) …
Настраивается пакет php-cli-prompt (1.0.1+dfsg-1build1) …
Настраивается пакет composer (1.0.0~beta2-1) …
Настраивается пакет javascript-common (11) …
Настраивается пакет libjs-excanvas (0.r3-4) …
Настраивается пакет mercurial-common (3.7.3-1ubuntu1.2) …
Настраивается пакет mercurial (3.7.3-1ubuntu1.2) …
```

Теперь можно переходить непосредственно к установке фреймворка. Сначала необходимо загрузить его установщик через Composer:

```
composer global require laravel/installer
```

```
[redacted]-vPro:~$ composer global require laravel/installer
Changed current directory to /home/enemy/.config/composer
Using version ^2.3 for laravel/installer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Installing symfony/process (v3.4.37)
    Downloading: 100%

  - Installing symfony/polyfill-ctype (v1.13.1)
    Downloading: 100%

  - Installing symfony/filesystem (v3.4.37)
    Downloading: 100%

  - Installing symfony/polyfill-mbstring (v1.13.1)
    Downloading: 100%

  - Installing psr/log (1.1.2)
    Downloading: 100%

  - Installing symfony/debug (v3.4.37)
    Downloading: 100%
```

В некоторых случаях выполнение этой команды может занять определенное время.

```
    Downloading: 100%

  - Installing psr/http-message (1.0.1)
    Downloading: 100%

  - Installing guzzlehttp/psr7 (1.6.1)
    Downloading: 100%

  - Installing guzzlehttp/promises (v1.3.1)
    Downloading: 100%

  - Installing guzzlehttp/guzzle (6.5.2)
    Downloading: 100%

  - Installing laravel/installer (v2.3.0)
    Downloading: 100%

symfony/console suggests installing symfony/event-dispatcher ()
symfony/console suggests installing symfony/lock ()
guzzlehttp/psr7 suggests installing zendframework/zend-httphandlerrunner (Emit PSR-7 responses)
guzzlehttp/guzzle suggests installing ext-intl (Required for Internationalized Domain Name (IDN) support)
Writing lock file
Generating autoload files
```

Далее, обязательно нужно поместить общесистемный каталог bin от Composer в свой каталог $PATH. Это позволит Ubuntu распознать исполняемый файл Laravel.

Этот каталог расположен в разных местах, в зависимости от дистрибутива. В Ubuntu он располагается по следующему пути:

```
$HOME/.config/composer/vendor/bin
```

Пользователи других дистрибутивов могут поискать здесь:

```
$HOME/.composer/vendor/bin
```

Использование Laravel Installer

Запустить установщик Laravel в Composer можно следующей командой в терминале:

```
composer global about
```

После ее выполнения отобразится список, где фреймворк будет расположен в первых рядах.

Далее достаточно ввести команду laravel new, после чего свежий фреймворк Laravel будет установлен в заранее указанном каталоге. К примеру, данная команда позволит создать папку под названием blog, где будут находиться Laravel со всеми установленными зависимостями:

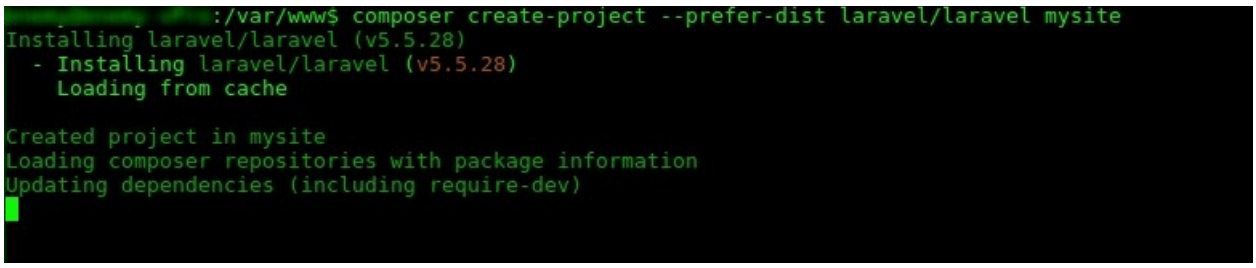
```
laravel new blog
```

Запуск фреймворка через Composer Create-Project

В Composer поддерживается возможность установки Laravel через команду composer create-project:

```
composer create-project --prefer-dist laravel/laravel blog
```

Эту команду необходимо выполнять, предварительно перейдя в папку, где будет храниться будущий сайт. Например, на локальном сервере Apache по умолчанию это каталог: cd /var/www/.



Работа на локальном сервере

После того, как Laravel установлен, следует убедиться в его работоспособности с помощью скрипта командной строки artisan (Artisan CLI). Для этого надо активировать встроенный в PHP локальный сервер с помощью команды:

```
php artisan serve
```

```
          -vPro:/var/www/mysite$ php artisan serve
Laravel development server started: <http://127.0.0.1:8000>
```

После ее ввода станет доступен локальный сервер по адресу http://localhost:8000. Перейдя по нему, можно увидеть заставку-приветствие «You have arrived» («Вы прибыли»).

Чтобы увидеть полный список команд для artisan, введите:

```
php artisan list
```

Последующая настройка

Публичная директория

Когда Laravel будет успешно установлен, сразу же потребуется настроить перенаправление файлов приложения в общедоступную папку /public. Файл index.php в этом каталоге используется как фронт-контроллер для всех HTTP-запросов, поступающих в создаваемое приложение.

Файлы настроек

Конфигурационные файлы — это PHP-скрипты, хранящиеся в application/config и вложенных подпапках.

Сами файлы представляет собой набор пар ключ массива => значение, где «ключ» — имя отдельной опции.

Права на доступ

В большинстве случаев, при настройке Laravel будет нужно отредактировать права доступа к директории /var/www, где по умолчанию хранятся веб-приложения. В противном случае фреймворк не сможет нормально функционировать.

Как получить права доступа

1. Добавить себя в группу владельцев веб-сервера www-data:

```
sudo usermod -a -G www-data $USER
```

Если пользователь вошел с root правами, этот пункт опускается.

2. Изменить права владельца на Laravel-проект командой:

```
sudo chgrp -R www-data /var/www/html/project
```

3. Даются права на запись в папку-хранилище storage:

```
sudo chmod -R 775 /var/www/html/project/storage
```

Для пользователя root (помимо первого пункта) из алгоритма исключается команда sudo.

Ключ приложения

Еще одна необходимая вещь, которую обязательно стоит сделать, когда Laravel установлен на хостинге – создать ключ приложения. При отсутствии установленного ключа приложения сеансы пользователей и другая конфиденциальная информация не будут защищены.

Он представляет собой случайно сгенерированную строку в 32 символа. Если для установки использовался Composer или установщик Laravel, ключ автоматически создается указанной командой:

```
php artisan key:generate
```

После создания ключ следует вписать как параметр в файл окружения .env. Если такой файл первоначально существовал в Laravel с расширением .env.example, то следует обязательно изменить расширение на .env.

Дополнительная настройка

Laravel сразу же после установки почти не нуждается в более подробном конфигурировании. Однако при желании можно ознакомиться с файлом конфигурации config/app.php и его документацией.

В нем указано несколько параметров, включая часовой пояс и локализацию, которые можно изменить с учетом особенностей разрабатываемого приложения.

Настройка веб-сервера

Настройка общедоступной директории

После установки и настройки Laravel нужно обязательно указать встроенную во фреймворк папку public в качестве корневого каталога веб-сервера. К этой общедоступной папке будут идти все входящие обращения из Сети. Обрабатывать входящие HTTP-запросы будет расположенный в public файл фронт-контроллер index.php.

Настройка веб-адресов

Apache

В установленном Laravel по умолчанию присутствует файл public/.htaccess. Он используется для того, чтобы URL-адреса отображались без указания фронт-контроллера index.php. Если Laravel установлен на веб-сервере Apache, для этого следует активировать модуль mod_rewrite. Тогда веб-сервер будет учитывать опции файла .htaccess.

Возможно, включенный в Laravel файл .htaccess не сможет заработать на установленном сервере. Тогда решить проблему может созданный аналогичный файл со следующим кодом:

```
Options +FollowSymLinks -Indexes
RewriteEngine On
RewriteCond %{HTTP:Authorization} .
RewriteRule .* - [E=HTTP_AUTHORIZATION:%{HTTP:Authorization}]
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^ index.php [L]
```

Nginx

Пользователям Nginx потребуется добавить еще одну директиву в конфигурации сайта. Она направит все запросы на фронт-контроллер index.php:

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

Основы маршрутизации

Простейшие маршруты Laravel принимают URI и замыкание, обеспечивая нетрудоемкий и выразительный метод определения маршрутов и поведения без сложных конфигурационных файлов маршрутизации:

```
use Illuminate\Support\Facades\Route;

Route::get('/greeting', function () {

    return 'Hello World';

});
```

Файлы маршрутов по умолчанию

Все маршруты Laravel должны быть определены в файлах маршрутов, находящихся в вашем каталоге routes. Эти файлы автоматически загружаются вашим поставщиком App\Providers\RouteServiceProvider приложения. Файл routes/web.php определяет маршруты для вашего веб-интерфейса. Этим маршрутам назначается группа посредников web, которая обеспечивает такие функции, как состояние сессии и защита от CSRF. Маршруты в routes/api.php не сохраняют состояния и им назначается группа посредников api.

Для большинства приложений вы начнете с определения маршрутов в файле routes/web.php. К маршрутам, определенным в routes/web.php, можно получить доступ, введя URL-адрес определенного маршрута в вашем браузере. Например, вы можете получить доступ к следующему маршруту, перейдя по адресу http://example.com/user в своем браузере:

```
use App\Http\Controllers\UserController;

Route::get('/user', [UserController::class, 'index']);
```

Маршруты, определенные в файле routes/api.php, вложены в группу маршрутов в RouteServiceProvider. Внутри этой группы автоматически применяется префикс URI /api, поэтому вам не нужно вручную добавлять его к каждому маршруту в файле маршрутов. Вы можете изменить префикс и другие параметры группы маршрутов, изменив свой класс RouteServiceProvider.

Доступные методы маршрутизатора

Маршрутизатор позволяет регистрировать маршруты, отвечающие на любой HTTP-метод:

```
Route::get($uri, $callback);

Route::post($uri, $callback);

Route::put($uri, $callback);

Route::patch($uri, $callback);

Route::delete($uri, $callback);

Route::options($uri, $callback);
```

Иногда требуется зарегистрировать маршрут, отвечающий на несколько HTTP-методов. Вы можете сделать это с помощью метода match. Или вы даже можете зарегистрировать маршрут, отвечающий на все HTTP-методы, используя метод any:

```
Route::match(['get', 'post'], '/', function () {

    //

});

Route::any('/', function () {

    //

});
```

При определении нескольких маршрутов, которые используют один и тот же URI, маршруты, использующие методы get, post, put, patch, delete и options, должны быть определены перед маршрутами, использующими методы any, match и redirect. Это гарантирует, что входящий запрос соответствует правильному маршруту.

Внедрение зависимости

Вы можете объявить любые зависимости, необходимые для вашего маршрута, в сигнатуре замыкания вашего маршрута. Объявленные зависимости будут автоматически извлечены и внедрены в замыкание с помощью контейнера служб Laravel. Например, вы можете объявить класс Illuminate\Http\Request, чтобы текущий HTTP-запрос автоматически был внедрен в замыкание вашего маршрута:

```
use Illuminate\Http\Request;

Route::get('/users', function (Request $request) {
    // ...
});
```

Защита от CSRF

Помните, что любые HTML-формы, указывающие на маршруты POST, PUT, PATCH или DELETE, которые определены в файле маршрутов web, должны включать поле токена CSRF. В противном случае запрос будет отклонен. Вы можете прочитать больше о защите от CSRF в документации CSRF:

```
<form method="POST" action="/profile">
    @csrf
    ...
</form>
```

Маршруты перенаправлений

Если вы определяете маршрут, который перенаправляет на другой URI, то вы можете использовать метод Route::redirect. Этот метод имеет лаконичную запись, так что вам не нужно определять полный маршрут или контроллер для выполнения простого перенаправления:

```
Route::redirect('/here', '/there');
```

По умолчанию Route::redirect возвращает код состояния 302. Вы можете переопределить код состояния, используя необязательный третий параметр:

```
Route::redirect('/here', '/there', 301);
```

Или вы можете использовать метод Route::permanentRedirect для возврата кода состояния 301:

```
Route::permanentRedirect('/here', '/there');
```

При использовании параметров маршрута в маршрутах перенаправления, следующие параметры зарезервированы Laravel и не могут быть использованы: destination и status.

Маршруты представлений

Если ваш маршрут должен возвращать только HTML-шаблон, то вы можете использовать метод Route::view. Как и метод redirect, этот метод имеет лаконичную запись, так что вам не нужно полностью определять маршрут или контроллер. Метод view принимает URI в качестве первого аргумента и имя шаблона в качестве второго аргумента. Кроме того, вы можете указать массив данных для передачи в шаблон в качестве необязательного третьего аргумента:

```
Route::view('/welcome', 'welcome');

Route::view('/welcome', 'welcome', ['name' => 'Taylor']);
```

При использовании параметров маршрута в маршрутах представлений, следующие параметры зарезервированы Laravel и не могут быть использованы: view, data, status и headers.

Написание контроллеров

Основные понятия о контроллерах

Давайте посмотрим на отвлеченный пример контроллера. Обратите внимание, что он расширяет базовый класс контроллера App\Http\Controllers\Controller, включенный в Laravel:

```
<?php
namespace App\Http\Controllers;
use App\Http\Controllers\Controller;
use App\Models\User;
class UserController extends Controller
{
    /**
     * Показать профиль конкретного пользователя.
     *
     * @param  int  $id
     * @return \Illuminate\View\View
     */
    public function show($id)
    {
        return view('user.profile', [
            'user' => User::findOrFail($id)
        ]);
    }
}
```

Вы можете определить маршрут к этому методу контроллера следующим образом:

```
use App\Http\Controllers\UserController;
Route::get('/user/{id}', [UserController::class, 'show']);
```

Когда входящий запрос совпадает с указанным URI маршрута, будет вызван метод show класса App\Http\Controllers\UserController, и параметры маршрута будут переданы методу.

Контроллеры не требуют расширения базового класса. Однако у вас не будет доступа к удобным функциям, таким как методы middleware иauthorize.

Контроллеры одиночного действия

Если действие контроллера является особенно сложным, вам может показаться удобным посвятить целый класс контроллера этому единственному действию. Для этого вы можете определить один метод __invoke в контроллере:

```
<?php
namespace App\Http\Controllers;
use App\Http\Controllers\Controller;
use App\Models\User;
class ProvisionServer extends Controller
{
    /**
     * Подготовить новый веб-сервер.
     *
     * @return \Illuminate\Http\Response
     */
    public function __invoke()
    {
        // ...
    }
}
```

При регистрации маршрутов для контроллеров одиночного действия вам не нужно указывать метод контроллера. Вместо этого вы можете просто передать маршрутизатору имя контроллера:

```
use App\Http\Controllers\ProvisionServer;
```

```
Route::post('/server', ProvisionServer::class);
```

Вы можете сгенерировать вызываемый контроллер, используя параметр --invokable команды make:controller Artisan:

```
php artisan make:controller ProvisionServer --invokable
```

Заготовки контроллера можно настроить с помощью публикации заготовок.

Посредник контроллера

Посредник может быть назначен маршрутам контроллера в ваших файлах маршрутизации:

```
Route::get('profile', [UserController::class, 'show'])->middleware('auth');
```

Или вам может быть удобно указать посредника в конструкторе вашего контроллера. Используя метод middleware в конструкторе вашего контроллера, вы можете назначить посредника действиям контроллера:

```
class UserController extends Controller
{
    /**
     * Создать новый экземпляр контроллера.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
        $this->middleware('log')->only('index');
        $this->middleware('subscribed')->except('store');
    }
}
```

Контроллеры также позволяют регистрировать посредника с помощью замыкания. Это обеспечивает удобный способ определения встроенного посредника для одного контроллера без определения целого класса посредника:

```
$this->middleware(function ($request, $next) {
    return $next($request);
});
```

Представления

Конечно, нецелесообразно возвращать целые строки HTML-документов непосредственно из ваших маршрутов и контроллеров. К счастью, шаблоны предоставляют удобный способ разместить весь наш HTML в отдельных файлах. Шаблоны отделяют логику контроллера / приложения от логики представления и хранятся в каталоге resources/views. Простой шаблон может выглядеть примерно так:

```
<!-- Шаблон сохранен в `resources/views/greeting.blade.php` -->
<html>
    <body>
        <h1>Привет, {{ $name }}</h1>
    </body>
</html>
```

Поскольку этот шаблон сохранен в resources/views/greeting.blade.php, мы можем вернуть его, используя глобальный помощник view, например:

```
Route::get('/', function () {
    return view('greeting', ['name' => 'James']);
});
```

Создание и отрисовка шаблонов

Вы можете создать шаблон, поместив файл с расширением .blade.php в каталог resources/views вашего приложения. Расширение .blade.php сообщает фреймворку, что файл содержит шаблон Blade. Шаблоны Blade содержат HTML, а также директивы Blade, которые позволяют легко выводить значения, создавать операторы «если», выполнять итерацию данных и многое другое.

После того как вы создали шаблон, вы можете вернуть его из маршрута или контроллера вашего приложения, используя глобальный помощник view:

```
Route::get('/', function () {
    return view('greeting', ['name' => 'James']);
});
```

Шаблон также могут быть возвращены с помощью фасада View:

```
use Illuminate\Support\Facades\View;
return View::make('greeting', ['name' => 'James']);
```

Как видно, первый аргумент, переданный помощнику view, соответствует имени файла шаблона в каталоге resources/views. Второй аргумент – это массив данных, которые должны быть доступны в шаблоне. В этом случае мы передаем переменную name, которая будет выведена в шаблоне с использованием синтаксиса Blade.

Вложенные каталоги шаблонов

Шаблоны также могут быть вложены в подкаталоги каталога resources/views. «Точечная нотация» используется для указания вложенности шаблона. Например, если ваш шаблон хранится в resources/views/admin/profile.blade.php, то вы можете вернуть его из маршрута / контроллера вашего приложения следующим образом:

```
return view('admin.profile', $data);
```

Имена каталогов шаблонов не должны содержать символа ..

Использование первого доступного шаблона

Используя метод first фасада View, вы можете отобразить первый шаблон, который существует в переданном массиве шаблонов. Это может быть полезно, если ваше приложение или пакет позволяют настраивать или перезаписывать шаблоны:

```
use Illuminate\Support\Facades\View;
return View::first(['custom.admin', 'admin'], $data);
```

Определение наличия шаблона

Если вам нужно определить, существует ли шаблон, вы можете использовать фасад View. Метод exists вернет true, если он существует:

```
use Illuminate\Support\Facades\View;

if (View::exists('emails.customer')) {
    //
}
```

Передача данных шаблону

Как вы видели в предыдущих примерах, вы можете передать массив данных шаблонам, чтобы сделать эти данные доступными для них:

```
return view('greetings', ['name' => 'Victoria']);
```

При передаче информации таким образом данные должны быть массивом с парами ключ / значение. После предоставления данных в шаблон вы можете получить доступ к каждому значению, используя ключи данных, схожее с <?php echo $name; ?>.

В качестве альтернативы передаче полного массива данных вспомогательной функции view вы можете использовать метод with для добавления некоторых данных в шаблон. Метод with возвращает экземпляр объекта представления, так что вы можете продолжить связывание методов перед возвратом шаблона:

```
return view('greeting')
            ->with('name', 'Victoria')
            ->with('occupation', 'Astronaut');
```

Общедоступные данные для всех шаблонов

Иногда требуется сделать данные общедоступными для всех шаблонов, отображаемыми вашим приложением. Вы можете сделать это, используя метод share фасада View. Как правило, вызов метода share осуществляется в методе boot поставщика служб. Вы можете добавить их в класс App\Providers\AppServiceProvider или создать отдельного поставщика для их размещения:

```
<?php
namespace App\Providers;
use Illuminate\Support\Facades\View;
class AppServiceProvider extends ServiceProvider
{
    /**
     * Регистрация любых служб приложения.
     *
     * @return void
     */
    public function register()
    {
        //
    }
    /**
     * Загрузка любых служб приложения.
     *
     * @return void
     */
    public function boot()
    {
        View::share('key', 'value');
    }
}
```

Индивидуальное задание

Создайте не менее пяти различных цепочек «Маршрут > Контроллер > Представление».

Лабораторная работа №4 – Docker

Введение

Docker — это приложение, упрощающее процесс управления процессами приложения в *контейнерах*. Контейнеры позволяют запускать приложения в процессах с изолированными ресурсами. Они похожи на виртуальные машины, но более портативные, более эффективно расходуют ресурсы и в большей степени зависят от операционной системы хоста.

Предварительные требования

Для выполнения этого руководства вам потребуется следующее:

- Один сервер Ubuntu, включающий пользователя non-root user с привилегиями sudo и брандмауэр.
- Учетная запись на Docker Hub, для создания собственных образов и загрузки их на Docker Hub.

Установка Docker

Пакет установки Docker, доступный в официальном репозитории Ubuntu, может содержать не самую последнюю версию. Чтобы точно использовать самую актуальную версию, мы будем устанавливать Docker из официального репозитория Docker. Для этого мы добавим новый источник пакета, ключ GPG от Docker, чтобы гарантировать загрузку рабочих файлов, а затем установим пакет.

Первым делом обновите существующий список пакетов:

```
sudo apt update
```

Затем установите несколько необходимых пакетов, которые позволяют `apt` использовать пакеты через HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl
software-properties-common
```

Добавьте ключ GPG для официального репозитория Docker в вашу систему:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
```

Добавьте репозиторий Docker в источники APT:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
```

Потом обновите базу данных пакетов и добавьте в нее пакеты Docker из недавно добавленного репозитория:

```
sudo apt update
```

Убедитесь, что установка будет выполняться из репозитория Docker, а не из репозитория Ubuntu по умолчанию:

```
apt-cache policy docker-ce
```

Вы должны получить следующий вывод, хотя номер версии Docker может отличаться:

```
Output of apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:19.03.9~3-0~ubuntu-focal
  Version table:
     5:19.03.9~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
```

Обратите внимание, что `docker-ce` не установлен, но является кандидатом на установку из репозитория Docker для Ubuntu 20.04 (версия `focal`).

Установите Docker:

```
sudo apt install docker-ce
```

Docker должен быть установлен, демон-процесс запущен, а для процесса активирован запуск при загрузке. Проверьте, что он запущен:

```
sudo systemctl status docker
```

Вывод должен выглядеть примерно следующим образом, указывая, что служба активна и запущена:

```
Output
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2020-05-19 17:00:41 UTC; 17s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 24321 (dockerd)
      Tasks: 8
     Memory: 46.4M
     CGroup: /system.slice/docker.service
             └─24321 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

После установки Docker у вас будет доступ не только к службе Docker (демон-процесс), но и к утилите командной строки `docker` или клиенту Docker. Мы узнаем, как использовать команду `docker` позже в этом обучающем руководстве.

Настройка команды Docker без sudo

По умолчанию команда `docker` может быть запущена только пользователем **root** или пользователем из группы **docker**, которая автоматически создается при установке Docker. Если вы попытаетесь запустить команду `docker` без префикса `sudo` или с помощью пользователя, который не находится в группе **docker**, то получите следующий вывод:

```
Output
docker: Cannot connect to the Docker daemon. Is the docker daemon running on this host?.
See 'docker run --help'.
```

Если вы не хотите каждый раз вводить `sudo` при запуске команды `docker`, добавьте свое имя пользователя в группу `docker`:

```
sudo usermod -aG docker ${USER}
```

Чтобы применить добавление нового члена группы, выйдите и войдите на сервер или введите следующее:

```
su - ${USER}
```

Вы должны будете ввести пароль вашего пользователя, чтобы продолжить.

Проверьте, что ваш пользователь добавлен в группу **docker**, введя следующее:

```
id -nG

Output
sammy sudo docker
```

Если вам нужно добавить пользователя в группу `docker`, для которой вы не выполнили вход, объявите имя пользователя явно, используя следующую команду:

```
sudo usermod -aG docker username
```

В дальнейшем подразумевается, что вы запускаете команду `docker` от имени пользователя в группе **docker**. В обратном случае вам необходимо добавлять к командам префикс `sudo`.

Использование команды Docker

Использование `docker` подразумевает передачу ему цепочки опций и команд, за которыми следуют аргументы. Синтаксис имеет следующую форму:

```
docker [option] [command] [arguments]
```

Чтобы просмотреть все доступные субкоманды, введите:

```
docker
```

Для 19-й версии Docker полный список субкоманд выглядит следующим образом:

```
Output
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes
```

Чтобы просмотреть параметры, доступные для конкретной команды, введите:

```
docker docker-subcommand --help
```

Чтобы просмотреть общесистемную информацию о Docker, введите следующее:

```
docker info
```

Давайте изучим некоторые из этих команд. Сейчас мы начнем работать с образами.

Работа с образами Docker

Контейнеры Docker получают из образов Docker. По умолчанию Docker загружает эти образы из Docker Hub, реестр Docker, контролируемые Docker, т.е. компанией, реализующей проект Docker. Любой может размещать свои образы Docker на Docker Hub, поэтому большинство приложений и дистрибутивов Linux, которые вам потребуется, хранят там свои образы.

Чтобы проверить, можно ли получить доступ к образам из Docker Hub и загрузить их, введите следующую команду:

```
docker run hello-world
```

Данный вывод говорит о том, что Docker работает корректно:

```
Output
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:6a65f928fb91fcfbc963f7aa6d57c8eeb426ad9a20c7ee045538ef34847f44f1
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

...
```

Docker первоначально не смог найти локальный образ `hello-world`, поэтому он загрузил образ из Docker Hub, который является репозиторием по умолчанию. После того как образ был загружен, Docker создал контейнер из образа, а приложение внутри контейнера было исполнено, отобразив сообщение.

Вы можете выполнять поиск доступных на Docker Hub с помощью команды `docker` с субкомандой `search`. Например, чтобы найти образ Ubuntu, введите:

```
docker search ubuntu
```

Скрипт пробежится по Docker Hub и вернет список всех образов с именами, совпадающими со строкой запроса. В данном случае вывод будет выглядеть примерно следующим образом:

```
Output
NAME                                                     DESCRIPTION                                     STARS
OFFICIAL            AUTOMATED
ubuntu                                                   Ubuntu is a Debian-based Linux operating sys…   10908
[OK]
dorowu/ubuntu-desktop-lxde-vnc                           Docker image to provide HTML5 VNC interface …   428
[OK]
rastasheep/ubuntu-sshd                                   Dockerized SSH service, built on top of offi…   244
[OK]
consol/ubuntu-xfce-vnc                                   Ubuntu container with "headless" VNC session…   218
[OK]
ubuntu-upstart                                           Upstart is an event-based replacement for th…   108
[OK]
ansible/ubuntu14.04-ansible                              Ubuntu 14.04 LTS with
...
```

В столбце **OFFICIAL OK** указывает на образ, созданный и поддерживаемый компанией, реализующей проект. После того как вы определили образ, который хотели бы использовать, вы можете загрузить его на свой компьютер с помощью субкоманды `pull`.

Запустите следующую команду, чтобы загрузить официальный образ `ubuntu` на свой компьютер:

```
docker pull ubuntu
```

Вывод должен выглядеть следующим образом:

```
Output
Using default tag: latest
latest: Pulling from library/ubuntu
d51af753c3d3: Pull complete
fc878cd0a91c: Pull complete
6154df8ff988: Pull complete
fee5db0ff82f: Pull complete
Digest:
sha256:747d2dbbaaee995098c9792d99bd333c6783ce56150d1b11e333bbceed5c54d7
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

После того как образ будет загружен, вы сможете запустить контейнер с помощью загруженного образа с помощью субкоманды `run`. Как вы уже видели на примере `hello-world`, если образ не был загружен, когда `docker` выполняется с субкомандой `run`, клиент Docker сначала загружает образ, а затем запускает контейнер с этим образом.

Чтобы просмотреть образы, которые были загружены на ваш компьютер, введите:

```
docker images
```

Вывод команды должен выглядеть примерно следующим образом:

```
Output
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest              1d622ef86b13        3 weeks ago         73.9MB
hello-world         latest              bf756fb1ae65        4 months ago        13.3kB
```

Как вы увидите далее в этом обучающем руководстве, образы, которые вы используете для запуска контейнеров, можно изменить и использовать для создания новых образов, которые затем могут быть загружены (*помещены*) на Docker Hub или другие реестры Docker.

Запуск контейнеров Docker

Контейнер `hello-world`, который вы запустили на предыдущем шаге, служит примером контейнера, который запускается и завершает работу после отправки тестового сообщения. Контейнеры могут быть гораздо более полезными, чем в примере выше, а также могут быть интерактивными. В конечном счете они очень похожи на виртуальные машины, но более бережно расходуют ресурсы.

В качестве примера мы запустим контейнер с самым последним образом образ Ubuntu. Сочетание переключателей **-i** и **-t** предоставляет вам доступ к интерактивной командной оболочке внутри контейнера:

```
docker run -it ubuntu
```

Необходимо изменить приглашение к вводу команды, чтобы отразить тот факт, что вы работаете внутри контейнера, и должны иметь следующую форму:

```
Output
root@d9b100f2f636:/#
```

Обратите внимание на идентификатор контейнер в запросе команды. В данном примере это `d9b100f2f636`. Вам потребуется этот идентификатор для определения контейнера, когда вы захотите его удалить.

Теперь вы можете запустить любую команду внутри контейнера. Например, сейчас мы обновим базу данных пакетов внутри контейнера. Вам

не потребуется начинать любую команду с `sudo`, потому что вы работаете внутри контейнера как **root**-пользователь:

```
apt update
```

После этого вы можете установите любое приложение внутри контейнера. Давайте установим Node.js:

```
apt install nodejs
```

Эта команда устанавливает Node.js внутри контейнера из официального репозитория Ubuntu. После завершения установки проверьте, что Node.js был установлен успешно:

```
node -v
```

Вы увидите номер версии, отображаемый в терминале:

```
Output
v10.19.0
```

Любые изменения, которые вы вносите внутри контейнера, применяются только к контейнеру.

Чтобы выйти из контейнера, введите `exit`.

Управление контейнерами Docker

После использования Docker в течение определенного времени, у вас будет много активных (запущенных) и неактивных контейнеров на компьютере. Чтобы просмотреть **активные**, используйте следующую команду:

```
docker ps
```

Вывод будет выглядеть примерно следующим образом:

```
Output
CONTAINER ID        IMAGE               COMMAND             CREATED
```

В этом обучающем руководстве вы запустили два контейнера: один из образа `hello-world` и другой из образа `ubuntu`. Оба контейнера больше не запущены, но все еще существуют в вашей системе.

Чтобы просмотреть все контейнеры — активные и неактивные, воспользуйтесь командой `docker ps` с переключателем `-a`:

```
docker ps -a
```

Вывод будет выглядеть следующим образом:

```
1c08a7a0d0e4        ubuntu              "/bin/bash"         2 minutes ago       Exited (0) 8
seconds ago                       quizzical_mcnulty
a707221a5f6c        hello-world         "/hello"            6 minutes ago       Exited (0) 6
minutes ago                       youthful_curie
```

Чтобы просмотреть последний созданный вами контейнер, передайте переключатель `-l`:

```
docker ps -l
```

```
CONTAINER ID  IMAGE       COMMAND       CREATED       STATUS              PORTS           NAMES

1c08a7a0d0e4  ubuntu      "/bin/bash"   2 minutes ago Exited (0) 40 seconds ago quizzical_mcnulty
```

Чтобы запустить остановленный контейнер, воспользуйтесь `docker start` с идентификатором контейнера или именем контейнера. Давайте запустим контейнер на базе Ubuntu с идентификатором `1c08a7a0d0e4`:

```
docker start 1c08a7a0d0e4
```

Контейнер будет запущен, а вы сможете использовать `docker ps`, чтобы просматривать его статус:

```
OutputONTAINER ID  IMAGE       COMMAND      CREATED         STATUS         PORTS           NAMES
1c08a7a0d0e4       ubuntu      "/bin/bash"  3 minutes ago      Up     5 seconds quizzical_mcnulty
```

Чтобы остановить запущенный контейнер, используйте `docker stop` с идентификатором или именем контейнера. На этот раз мы будем использовать имя, которое Docker присвоил контейнеру, т.е. `quizzical_mcnulty`:

```
docker stop quizzical_mcnulty
```

После того как вы решили, что вам больше не потребуется контейнер, удалите его с помощью команды `docker rm`, снова добавив идентификатор контейнера или его имя. Используйте команду `docker ps -a`, чтобы найти идентификатор или имя контейнера, связанного с образом `hello-world`, и удалить его.

```
docker rm youthful_curie
```

Вы можете запустить новый контейнер и присвоить ему имя с помощью переключателя `--name`. Вы также можете использовать переключатель `--rm`, чтобы создать контейнер, который удаляется после остановки. Изучите команду `docker run help`, чтобы получить больше информации об этих и прочих опциях.

Контейнеры можно превратить в образы, которые вы можете использовать для создания новых контейнеров. Давайте посмотрим, как это работает.

Внесение изменений в контейнер для образа Docker

После запуска образа Docker вы можете создавать, изменять и удалять файлы так же, как и с помощью виртуальной машины. Эти изменения будут применяться только к данному контейнеру. Вы можете запускать и останавливать его, но после того как вы уничтожите его с помощью команды `docker rm`, изменения будут утрачены навсегда.

Данный раздел показывает, как сохранить состояние контейнера в виде нового образа Docker.

После установки Node.js внутри контейнера Ubuntu у вас появился контейнер, запускающий образ, но этот контейнер отличается от образа, который вы использовали для его создания. Но позже вам может снова потребоваться этот контейнер Node.js в качестве основы для новых образов.

Затем внесите изменения в новый экземпляр образа Docker с помощью следующей команды.

```
docker commit -m "What you did to the image" -a "Author Name" container_id
repository/new_image_name
```

Переключатель **-m** используется в качестве сообщения о внесении изменений, которое помогает вам и остальным узнать, какие изменения вы внесли, в то время как **-a** используется для указания автора. `container_id` — это тот самый идентификатор, который вы отмечали ранее в этом руководстве, когда запускали интерактивную сессию Docker. Если вы не создавали дополнительные репозитории на Docker Hub, `repository`, как правило, является вашим именем пользователя на Docker Hub.

Например, для пользователя **sammy** с идентификатором контейнера `d9b100f2f2f6` команда будет выглядеть следующим образом:

```
docker commit -m "added Node.js" -a "sammy" d9b100f2f636 sammy/ubuntu-
nodejs
```

Когда вы *вносите* образ, новый образ сохраняется локально на компьютере. Позже в этом обучающем руководстве вы узнаете, как добавить образ в реестр Docker, например, на Docker Hub, чтобы другие могли получить к нему доступ.

Список образов Docker теперь будет содержать новый образ, а также старый образ, из которого он будет получен:

```
docker images
```

Вывод будет выглядеть следующим образом:

```
Output
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
sammy/ubuntu-nodejs   latest              7c1f35226ca6        7 seconds ago       179MB
...
```

В данном примере `ubuntu-nodejs` является новым образом, который был получен из образа `ubuntu` на Docker Hub. Разница в размере отражает внесенные изменения. В данном примере изменение состояло в том, что NodeJS был установлен. В следующий раз, когда вам потребуется запустить контейнер, использующий Ubuntu с предустановленным NodeJS, вы сможете использовать новый образ.

Вы также можете создавать образы из `Dockerfile`, что позволяет автоматизировать установку программного обеспечения в новом образе. Однако это не относится к предмету данного обучающего руководства.

Загрузка образов Docker в репозиторий Docker

Следующим логическим шагом после создания нового образа из существующего является предоставление доступа к этому образу нескольким вашим друзьям или всему миру на Docker Hub или в другом реестре Docker, к которому вы имели доступ. Чтобы добавить образ на Docker Hub или любой другой реестр Docker, у вас должна быть там учетная запись.

Чтобы загрузить свой образ, выполните вход в Docker Hub.

```
docker login -u docker-registry-username
```

Вам будет предложено использовать для аутентификации пароль Docker Hub. Если вы указали правильный пароль, аутентификация должна быть выполнена успешно.

Примечание. Если ваше имя пользователя в реестре Docker отличается от локального имени пользователя, которое вы использовали при создании образа, вам потребуется пометить ваш образ именем пользователя в реестре. Для примера, приведенного на последнем шаге, вам необходимо ввести следующую команду:

```
docker tag sammy/ubuntu-nodejs docker-registry-username/ubuntu-nodejs
```

Затем вы можете загрузить свой образ с помощью следующей команды:

```
docker push docker-registry-username/docker-image-name
```

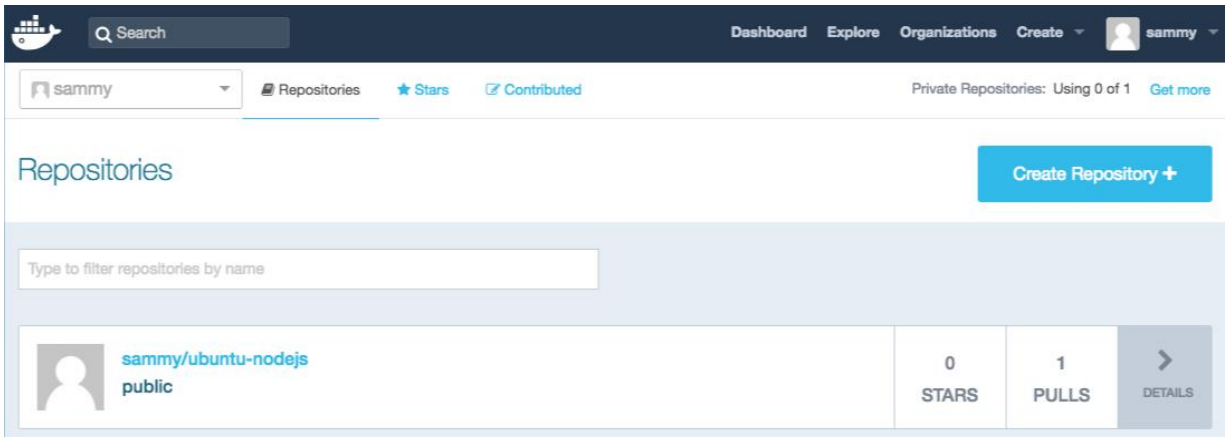
Чтобы загрузить образ `ubuntu-nodejs` в репозиторий **sammy**, необходимо использовать следующую команду:

```
docker push sammy/ubuntu-nodejs
```

Данный процесс может занять некоторое время, необходимое на загрузку образов, но после завершения вывод будет выглядеть следующим образом:

```
Output
The push refers to a repository [docker.io/sammy/ubuntu-nodejs]
e3fbbfb44187: Pushed
5f70bf18a086: Pushed
a3b5c80a4eba: Pushed
7f18b442972b: Pushed
3ce512daaf78: Pushed
7aae4540b42d: Pushed

...
```

После добавления образа в реестр он должен отображаться в панели вашей учетной записи, как на изображении ниже.



Если при попытке добавления возникает подобная ошибка, вы, скорее всего, не выполнили вход:

```
Output
The push refers to a repository [docker.io/sammy/ubuntu-nodejs]
e3fbbfb44187: Preparing
5f70bf18a086: Preparing
a3b5c80a4eba: Preparing
7f18b442972b: Preparing
3ce512daaf78: Preparing
7aae4540b42d: Waiting
unauthorized: authentication required
```

Выполните вход с помощью команды `docker login` и повторите попытку загрузки. Проверьте, появился ли образ на вашей странице репозитория Docker Hub.

Теперь вы можете использовать `docker pull sammy/ubuntu-nodejs`, чтобы загрузить образ на новый компьютер и использовать его для запуска нового контейнера.

Лабораторная работа №5 – Очереди, RabbitMQ

Введение

Очередь — структура данных с дисциплиной доступа к элементам «первый пришёл — первый вышел». Добавление элемента возможно лишь в конец очереди, выборка — только из начала очереди, при этом выбранный элемент из очереди удаляется.

Десять причин, почему очереди сообщений являются жизненно важным компонентом для любой архитектуры или приложения:

1. Слабое связывание — очереди сообщений создают неявные интерфейсы обмена данными, которые позволяют процессам быть независимыми друг от друга т.е вы просто определяете формат сообщений отправляемых от одного процесса другому.
2. Избыточность — Очереди позволяют избежать случаев неэкономного использования ресурсов процесса(например памяти) в результате хранения необработанной (лишней) информации.
3. Масштабируемость — очереди сообщений позволяют распределить процессы обработки информации. Таким образом, они позволяют легко наращивать скорость, с которой сообщения добавляются в очередь и обрабатываются.
4. Эластичность и возможность выдерживать пиковые нагрузки — очереди сообщений могут выполнять роль своего рода буфера для накопления данных в случае пиковой нагрузки, смягчая тем самым нагрузку на систему обработки информации и не допуская ее отказа.
5. Отказоустойчивость — очереди сообщений позволяют отделить процессы друг от друга, так что если процесс, который обрабатывает сообщения из очереди падает, то сообщения могут быть добавлены в очередь на обработку позднее, когда система восстановится.
6. Гарантированная доставка — использование очереди сообщений гарантирует, что сообщение будет доставлено и обработано в любом случае (пока есть хотя бы один обработчик).
7. Гарантированный порядок доставки — большая часть систем очередей сообщений способны обеспечить гарантии того, что данные будут обрабатываться в определённом порядке (чаще всего в том порядке в котором они поступили).
8. Буферизация — очереди сообщений позволяет отправлять и получать сообщения при этом работая с максимальной эффективностью, предлагая буферный слой — процесс записи в очередь может происходить настолько быстро, насколько быстро это в состоянии выполнить очередь сообщений, а не обработчик сообщения.

9. Понимание потоков данных — очереди сообщений позволяют выявлять узкие места в потоках данных приложения, легко можно определить какая из очередей забивается, какая простаивает и определить что необходимо делать — добавлять новых обработчиков сообщений или оптимизировать текущую архитектуру.
10. Асинхронная связь — очереди сообщений предоставляют возможность асинхронной обработки данных, которая позволяет поместить сообщение в очередь без обработки, позволяя системе обработать сообщение позднее, когда появится возможность.

Установка RabbitMQ с management-плагином

Теория по message queue

О́чередь — структура данных с дисциплиной доступа к элементам «первый пришёл — первый вышел». Добавление элемента возможно лишь в конец очереди, выборка — только из начала очереди, при этом выбранный элемент из очереди удаляется.

Десять причин, почему очереди сообщений являются жизненно важным компонентом для любой архитектуры или приложения:

- Слабое связывание — очереди сообщений создают неявные интерфейсы обмена данными, которые позволяют процессам быть независимыми друг от друга т.е вы просто определяете формат сообщений отправляемых от одного процесса другому.
- Избыточность — Очереди позволяют избежать случаев неэкономного использования ресурсов процесса(например памяти) в результате хранения необработанной (лишней) информации.
- Масштабируемость — очереди сообщений позволяют распределить процессы обработки информации. Таким образом, они позволяют легко наращивать скорость, с которой сообщения добавляются в очередь и обрабатываются.
- Эластичность и возможность выдерживать пиковые нагрузки — очереди сообщений могут выполнять роль своего рода буфера для накопления данных в случае пиковой нагрузки, смягчая тем самым нагрузку на систему обработки информации и не допуская ее отказа.
- Отказоустойчивость — очереди сообщений позволяют отделить процессы друг от друга, так что если процесс, который обрабатывает

сообщения из очереди падает, то сообщения могут быть добавлены в очередь на обработку позднее, когда система восстановится.

- Гарантированная доставка — использование очереди сообщений гарантирует, что сообщение будет доставлено и обработано в любом случае (пока есть хотя бы один обработчик).
- Гарантированный порядок доставки — большая часть систем очередей сообщений способны обеспечить гарантии того, что данные будут обрабатываться в определённом порядке (чаще всего в том порядке в котором они поступили).
- Буферизация — очереди сообщений позволяет отправлять и получать сообщения при этом работая с максимальной эффективностью, предлагая буферный слой — процесс записи в очередь может происходить настолько быстро, насколько быстро это в состоянии выполнить очередь сообщений, а не обработчик сообщения.
- Понимание потоков данных — очереди сообщений позволяют выявлять узкие места в потоках данных приложения, легко можно определить какая из очередей забивается, какая простаивает и определить что необходимо делать — добавлять новых обработчиков сообщений или оптимизировать текущую архитектуру.
- Асинхронная связь — очереди сообщений предоставляют возможность асинхронной обработки данных, которая позволяет поместить сообщение в очередь без обработки, позволяя системе обработать сообщение позднее, когда появится возможность.

Установка RabbitMQ с management-плагином

Воспользуемся полученными в предыдущей лабораторной работе знаниями и запустим RabbitMQ в контейнере

Напишем docker-compose.yml

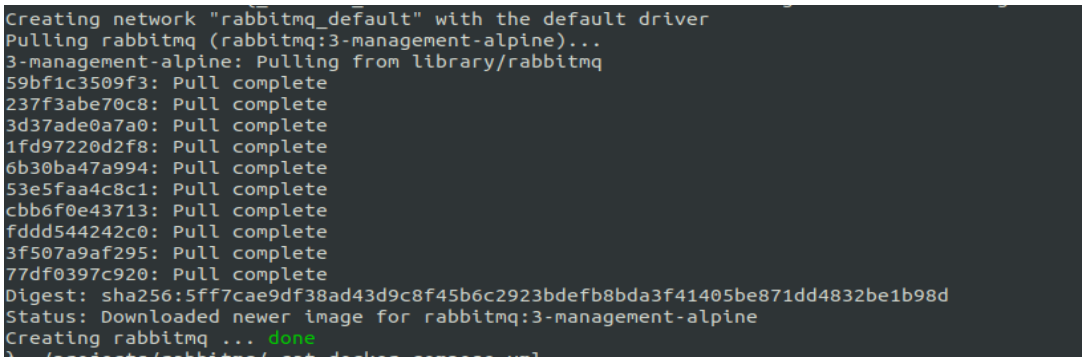
```
services:
    rabbitmq:
        image: rabbitmq:3-management-alpine
        container_name: rabbitmq
        environment:
            RABBITMQ_DEFAULT_USER: user
            RABBITMQ_DEFAULT_PASS: password
```

```
    ports:
        - 5672:5672
        - 15672:15672
```

Поднимем контейнер с помощью команды

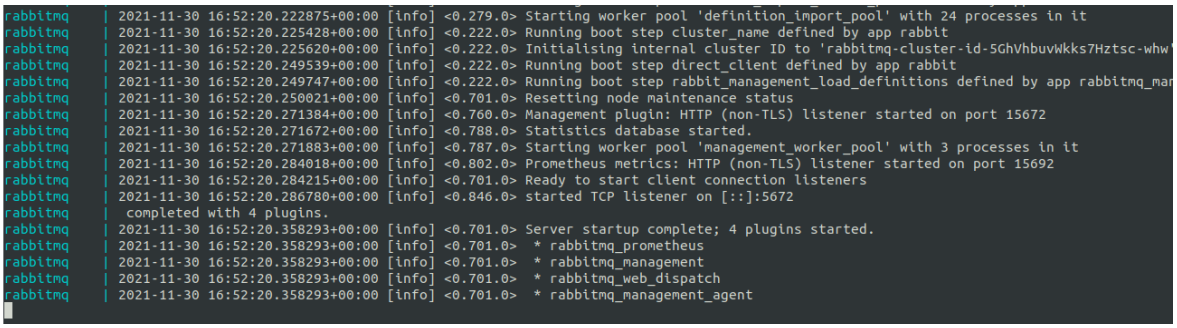
```
docker-compose up -d
```

Сначала произойдет сборка

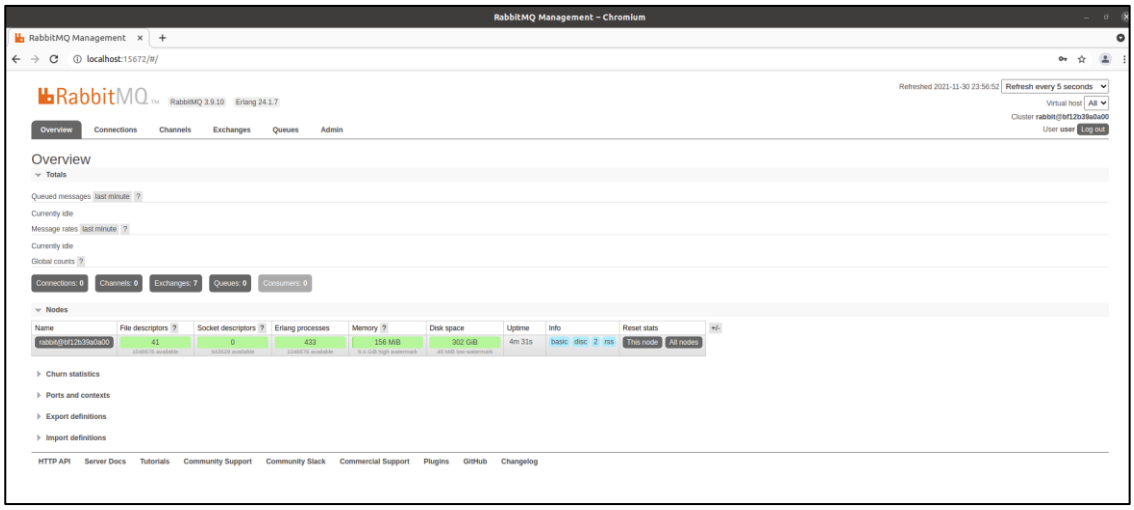


Далее контейнер запустился, посмотрим логи с помощью команды

```
docker-compose logs
```



Зайдем с логином и паролем, проверим что Management plugin работает

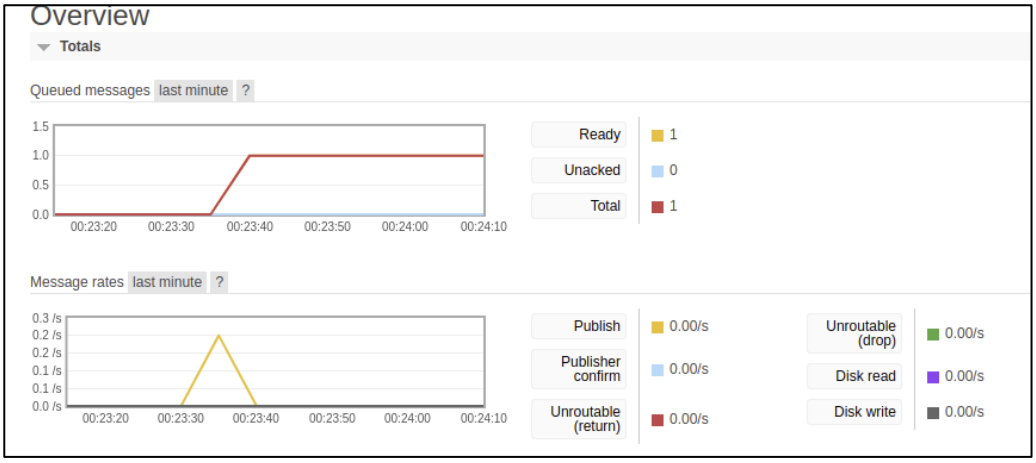


Написание producer

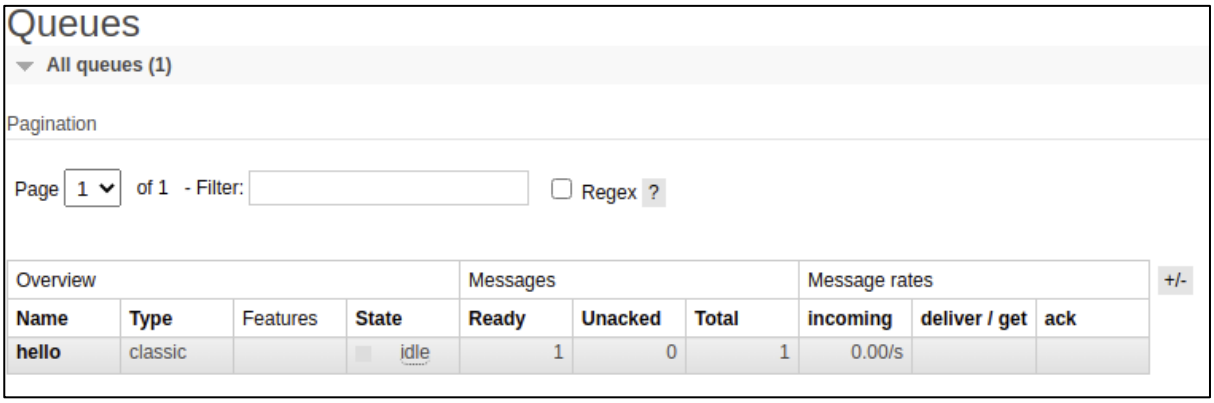
Напишем код для постановки сообщения в очередь.

```python
#!/usr/bin/env python
import pika


if __name__ == '__main__':
    connection = pika.BlockingConnection(
        pika.ConnectionParameters(
            'localhost',
            credentials=pika.credentials.PlainCredentials(
                'user',
                'password'
            ),
        )
    )
    channel = connection.channel()

    channel.queue_declare(queue='hello')

    channel.basic_publish(
        exchange='',
        routing_key='hello',
        body=b'Hello World!'
    )
    print(" [x] Sent 'Hello World!'")

    connection.close()
```

Выполним записанный код, после чего увидим, что RabbitMQ принял одно сообщение.



В разделе Queues можно увидеть, что создалась одна очередь



Написание consumer

Напишем код для приема сообщения из очереди.

```python
#!/usr/bin/env python
import pika


def handler(ch, method, properties, message):
    print(f" [x] Received {message}")


if __name__ == '__main__':
    connection = pika.BlockingConnection(
        pika.ConnectionParameters(
            'localhost',
            credentials=pika.credentials.PlainCredentials(
                'user',
                'password'
            ),
        )
    )
    channel = connection.channel()

    channel.queue_declare(queue='hello')

    channel.basic_consume(
        queue='hello',
        auto_ack=True,
        on_message_callback=handler
    )

    print(' [*] Waiting for messages.')
    channel.start_consuming()
```

Запустим consumer, который сразу же обработает сообщение

```
/usr/bin/python3.8 /home/hagassaan/projects/rabbitmq/consumer.py
 [*] Waiting for messages.
 [x] Received b'Hello World!'
```

Обратим внимание на графики в UI RabbitMQ

Сообщение ушло.

Индивидуальное задание:

Доработать producer и consumer так, чтобы producer ставил через RabbitMQ задачу, которую возьмет consumer, выполнит её и вернет результат обратно.

В качестве задачи можно взять любую математическую операцию над числом (например, возведение числа в степень)