# ft_server

**COMANDOS PROJETO**

**Para construir imagem:**

```
docker build -t ft_server .
```

**Para rodar:**

```
docker run -it -p80:80 -p443:443 ft_server
```

**Para autoindex:**

```
sh ./srcs/s_autoindex.sh
```

| | | |
|---|---|---|
| **Login MariaDB:** | root | “” |
| **Loging Wordpress:** | wordpres | 1234 |

---

**Roteiro**
Criar imagem para gerar container
    Aprender comandos para listar imagens, listar containers, apagar imagens, apagar containers e rodar containers.
Na imagem, instalar;
   1) Via package manager (apt):
      a.OS;
         Fazer update e upgrade;
      b.Utilitarios
         Wget para dwnlds
         Openssl;
         Vim;
         Man pages;
      c.NGINX;
      d.MariaDB;
      e.PHP:
         Instalar libraries/modulos oriundas dos demais softwares.
   2) Via download com wget:
      a.phpMyAdmin
      b.Wordpress

---

**Docker:**

Docker container ls: mostra containers
Docker container ls —all: mostra containers inclusive rodando

Docker container run [container]: ativa container (se nao tiver imagem, faz download do site

Docker image: lista imagens

Docker container exec -it [container] /bin/bash: executa comando no container (no caso -it eh interativo com tty e executa bash)

Docker container cp [container_orig:caminho] [container_dest:caminho]: copia arquivos de origem para destino. Para maquina local basta omitir container

Docker stop [container]: para container

Docker rm -f [container]: deleta container

Docker system df: lista espaço em disco ocupado por imagens e containers

Any Dockerfile you write should be optimized so that the instructions are ordered by how frequently they change—with instructions that are unlikely to change at the start of the Dockerfile, and instructions most likely to change at the end.

Apagando varios containers:

```
docker ps -a | grep "pattern" | awk '{print $1}' | xargs docker rm
docker container ls -a | grep "Exited (127)" | cut -d' ' -f1 | xargs docker rm
```

Apagando todos containers parados:

```
docker rm $(docker ps -a -q)
```

Apagando imagens:

```
docker images -a | grep "pattern" | awk '{print $3}' | xargs docker rmi
```

Docker run roda a partir de uma imagem
Docker start roda a partir de um container existente

**Plugando em um container que esta rodando para interagir com ele:**

```
docker container exec -i -t trivia /bin/sh (-it para ser interativo com tty ,
depois nome do container, depois comando a executar)
```

**Iniciando um container existente em modo interativo:**

```
docker container start -ia [container]
```

Iniciando um container a partir de um imagem em modo interativo:

```
Docker container run -it [container] /bin/bash
```

```
$ docker container run --rm -it --entrypoint /bin/sh pinger (executa um
container a partir da imagem “pinger’ porem entrando pelo shell, dando override
no Dockerfile)
```

Docker volume create [VOLUME]: cria um “disco virtual” persistente no host que pode ser mapeado para dentro dos containers com flag -v [VOLUME]:[DIR CONTAINER]
Se varios containers forem acessar o volume ao mesmo tempo eh preciso tomar cuidado com acessos de read e write (para nao ter 2 writes ao mesmo tempo por ex)
Docker volume rm [VOLUME]: deleta volume do host
Docker volume ls: lista volumes que existem no host

Volumes podem ser criados no Dockerfile

---

**NGINX:**

## LIVROS:

**Rahul Soni (auth.) - Nginx_ From Beginner to Pro -Apress (2016).pdf**

Capitulos 1,2,3,6,7 (6 e 7 bons para LEMP / ft_server)

**Valery Kholodkov - Nginx Essentials-Packt Publishing (2015).pdf**

Capitulos 1 e 2

**Martin Fjordvald, Clement Nedelcu - Nginx HTTP Server - Fourth Edition_ Harness the power of Nginx to make the most of your infrastructure and serve pages faster than ever before-Packt Publishing - eb**

Capitulo 10 (instalacao do Wordpress)

## Understanding the Default Configuration

The default configuration of Nginx looks similar to the following:

```
user nginx;
worker_processes 1;
error_log /var/log/nginx/error.log warn; pid /var/run/nginx.pid;
events {
    worker_connections  1024;
}
http {
include /etc/nginx/mime.types;
default_type application/octet-stream;
log_format main '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
'"$http_user_agent" "$http_x_forwarded_for"'; access_log
/var/log/nginx/access.log main;
    sendfile        on;
    #tcp_nopush     on;
    keepalive_timeout  65;
    #gzip  on;
include /etc/nginx/conf.d/*.conf; }
```

```
server {
    listen       80;
    server_name  localhost;
#charset koi8-r;
#access_log /var/log/nginx/log/host.access.log main;
location / {
2. root /etc/nginx/html; index index.html index.htm;
}
#error_page 404 /404.html;
# redirect server error pages to the static page /50x.html #
error_page 500 502 503 504 /50x.html;
location = /50x.html {
        root   /usr/share/nginx/html;
    }
# proxy the PHP scripts to Apache listening on 127.0.0.1:80 #
#location ~ \.php$ {
# proxy_pass http://127.0.0.1;
#}
# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000 #
#location ~ \.php$ {
* #  root
*     #     fastcgi_pass
*     #     fastcgi_index
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}
# deny access to .htaccess files, if Apache's document root # concurs with
nginx's one
#
#location ~ /\.ht {
# deny all;
#} }
```

Configuracoes de servers em um arquivo separado;

http://app1.com or http://www.app1.com http://app2.com or http://www.app2.com:

```
server {
        listen 80;
server_name app1.com www.app1.com; location / {
                root /etc/nginx/html/app1;
} }
server {
        listen 80;
server_name app2.com www.app2.com; location / {
                root /etc/nginx/html/app2;
} }
```

Autoindex:

```
server {
        listen 80;
server_name 127.0.0.1 localhost; root /etc/nginx/html;
        location /common/ {
                root /etc/nginx/html;
                index NON_EXISTENT_FILE;
                autoindex on;
} }
```

Teste das configurações do nginx:

```
nginx -t
```

Ligar / desligar / restartar nginx:

```
service nginx [start / stop / restart]
```

---

SSL:

• Logon to WFE1

**ssh -p 3026 user1@127.0.0.1**

• Create a directory to hold all SSL-related files sudo mkdir /etc/nginx/ssl

• Use the following command to create a private key and a certificate:

**sudo openssl req -nodes -days 3650 -x509 -newkey rsa:2048 -keyout /etc/nginx/ssl/private.key -out /etc/nginx/ssl/cert.crt**

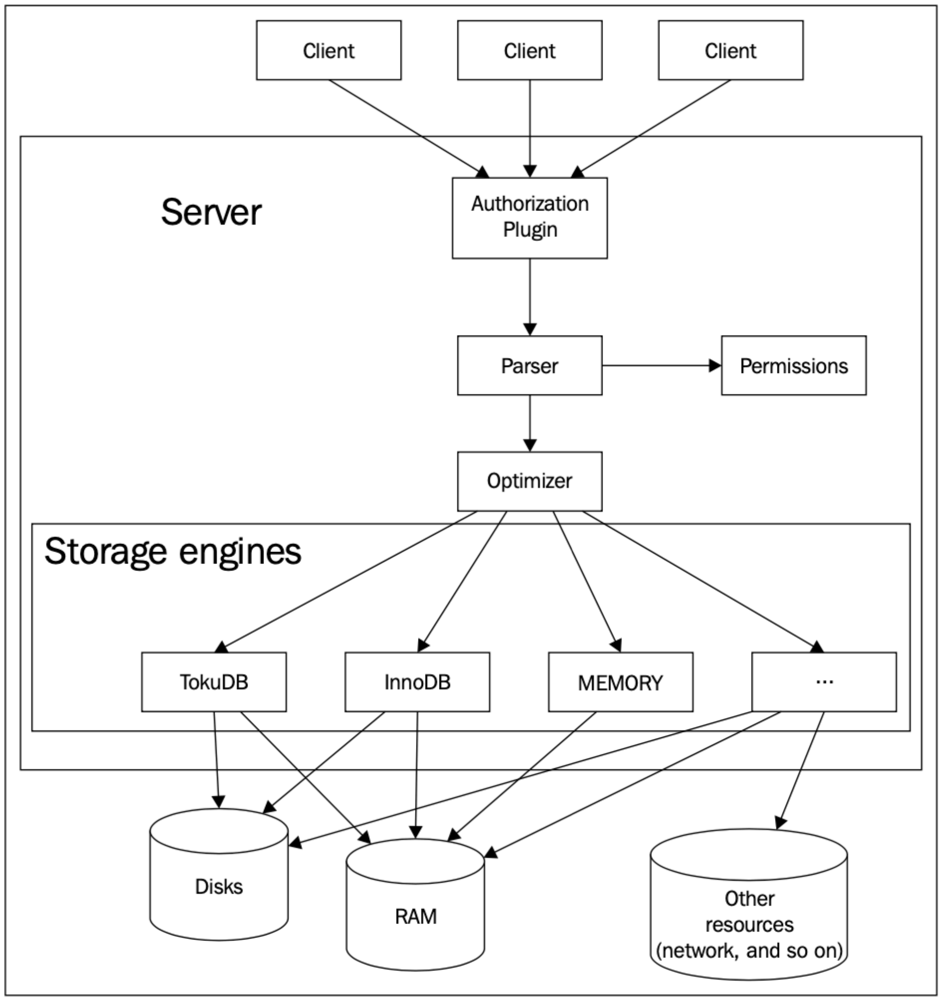
```
Generating a 2048 bit RSA private key .........................................................................................
.............................................................+++

.+++
writing new private key to '/etc/nginx/ssl/private.key'
-----
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but
you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:Rahul Soni Organizational Unit Name (eg, section)
[]:
Common Name (eg, your name or your server's hostname) []:localhost

Email Address []:
```

```
server {
    listen 80;
    server_name  localhost;
    listen 443 ssl;
     ssl_certificate /etc/nginx/ssl/cert.crt;
     ssl_certificate_key /etc/nginx/ssl/private.key;
    location / {
        root /usr/share/nginx/html; index index.html index.htm;
        index index.html index.htm;
    }
}
```

---

## MariaDB:

The following schema represents the architecture of MariaDB:



**Comando apt:**

update (apt-get(8))

update is used to download package information from all configured sources. Other commands operate on this data to e.g. perform package
upgrades or search in and display details about all packages available for installation.

upgrade (apt-get(8))

upgrade is used to install available upgrades of all packages currently installed on the system from the sources configured via
sources.list(5). New packages will be installed if required to satisfy dependencies, but existing packages will never be removed. If an

upgrade for a package requires the removal of an installed package the upgrade for this package isn't performed.

-y: responde yes para todas perguntas
-q: quiet, suprime indicadores de progresso

**dpkg —list:** lista todos pacotes instalados

---

**PHP:**
Php eh uma linguagem para confecção de paginas dinâmicas web. Pode ser usado em servidores web (precisa do php-fpm), linha de comando interativa ou para fazer aplicativos. Dependendo do uso necessario instalar pacotes alem dos pacotes defaults, ver listas abaixo:

Lista todas libraries do php:

```
apt-cache search php | more
```

Instalar php e algumas de suas libraries:

*Php-gettext: phpmyadmin, ja instala default com php
* php-mbstring: a PHP extension used to manage non-ASCII strings and convert strings to different encodings (phpmyadmin)
* php-zip: a PHP module that supports uploading .zip files to phpMyAdmin
* php-gd: another PHP module, this one enables support for the GD Graphics Library

**Libraries necessárias para o Wordpress:**

```
PHP Extensions #PHP Extensions


WordPress core makes use of PHP extensions. If the preferred extension is
missing WordPress will either have to do more work to do the task the module
helps with or, in the worst case, will remove functionality. Therefore the PHP
extensions listed below are recommended.
* curl – Performs remote request operations.
* dom – Used to validate Text Widget content and to automatically configuring
IIS7+.
* exif – Works with metadata stored in images.
* fileinfo – Used to detect mimetype of file uploads.
* hash – Used for hashing, including passwords and update packages.
* json – Used for communications with other servers.
* mbstring – Used to properly handle UTF8 text.
* mysqli – Connects to MySQL for database interactions.
* libsodium – Validates Signatures and provides securely random bytes.
* openssl – Permits SSL-based connections to other hosts.
* pcre – Increases performance of pattern matching in code searches.
* imagick – Provides better image quality for media uploads. See
WP_Image_Editor is incoming! for details. Smarter image resizing (for smaller
images) and PDF thumbnail support, when Ghost Script is also available.
* xml – Used for XML parsing, such as from a third-party site.
* zip – Used for decompressing Plugins, Themes, and WordPress update packages.
For the sake of completeness, below is a list of the remaining PHP modules
WordPress may use in certain situations or if other modules are unavailable.
These are fallbacks or optional and not necessarily needed in an optimal
environment, but installing them won't hurt.
* filter – Used for securely filtering user input.
* gd – If Imagick isn't installed, the GD Graphics Library is used as a
functionally limited fallback for image manipulation.
* iconv – Used to convert between character sets.
* mcrypt – Generates random bytes when libsodium and /dev/urandom aren't
available.
* simplexml – Used for XML parsing.
* xmlreader – Used for XML parsing.
* zlib – Gzip compression and decompression.
These extensions are used for file changes, such as updates and plugin/theme
installation, when files aren't writeable on the server.
* ssh2
* ftp
* sockets (For when the ftp extension is unavailable)
```

**Libraries necessárias para o phpMyAdmin:**

```
Requirements


Web server


Since phpMyAdmin's interface is based entirely in your browser, you'll need a
web server (such as Apache, nginx, IIS) to install phpMyAdmin's files into.
PHP


* You need PHP 7.1.3 or newer, with session support, the Standard PHP Library
(SPL) extension, hash, ctype, and JSON support.
* The mbstring extension (see mbstring) is strongly recommended for
performance reasons.
* To support uploading of ZIP files, you need the PHP zip extension.
* You need GD2 support in PHP to display inline thumbnails of JPEGs
("image/jpeg: inline") with their original aspect ratio.
* When using the cookie authentication (the default), the openssl extension is
```

```
strongly suggested.
* To support upload progress bars, see 2.9 Seeing an upload progress bar.
* To support XML and Open Document Spreadsheet importing, you need the libxml
extension.
* To support reCAPTCHA on the login page, you need the openssl extension.
* To support displaying phpMyAdmin’s latest version, you need to enable
allow_url_open in your php.ini or to have the curl extension.
```

**Libraries necessárias para o NGINX:**

```
    Php-fpm: fast cgi  , para Nginx
```

**Libraries necessárias para o MariaDB:**

```
Overview of the MySQL PHP drivers ¶


Introduction


Depending on the version of PHP, there are either two or three PHP APIs for
accessing the MySQL database. PHP 5 users can choose between the deprecated
mysql extension, mysqli, or PDO_MySQL. PHP 7 removes the mysql extension,
leaving only the latter two options.
```

---

**NGINX:**
**Iniciando:**

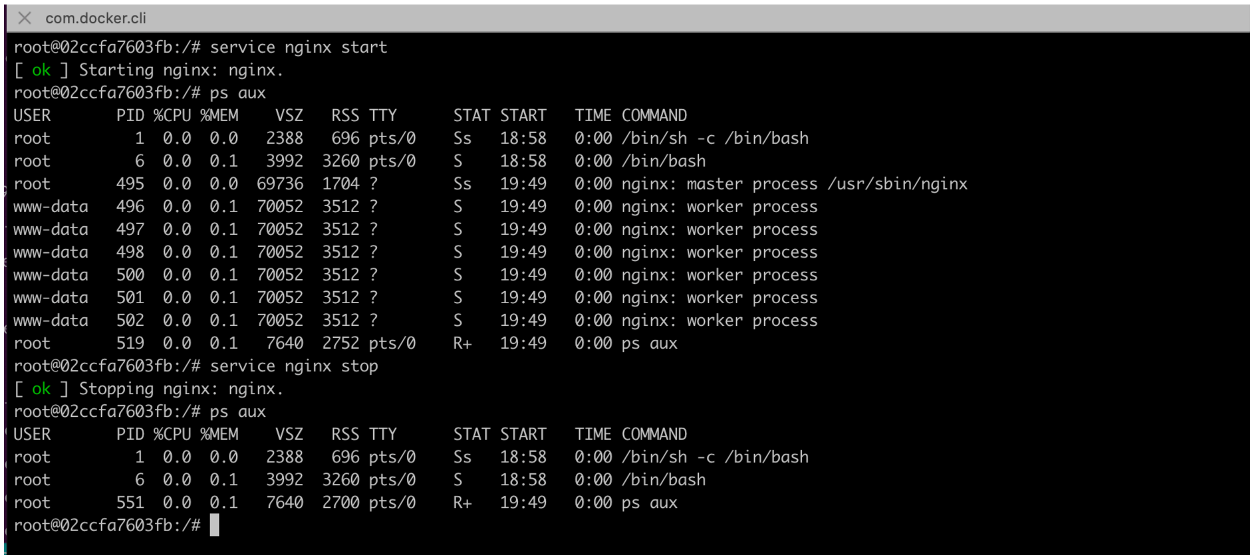
```
service nginx start
```

**Parando:**

```
service nginx stop
```

**Verificando se proceso esta rodando:**

```
ps aux
```



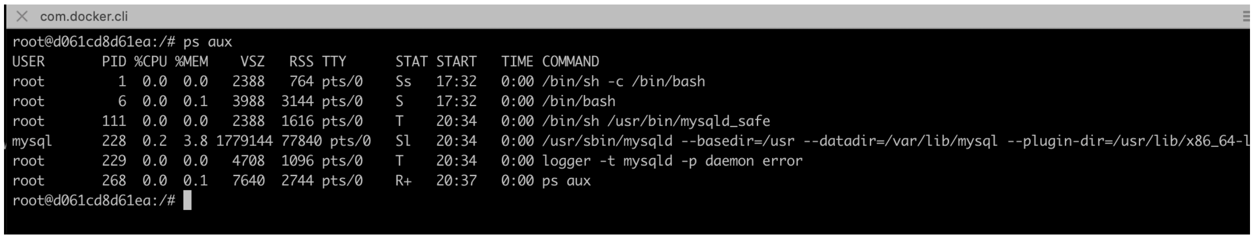
```
root@02ccfa7603fb:/# service nginx start
[ ok ] Starting nginx: nginx.
root@02ccfa7603fb:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2388   696 pts/0    Ss   18:58   0:00 /bin/sh -c /bin/bash
root         6  0.0  0.1   3992  3260 pts/0    S    18:58   0:00 /bin/bash
root       495  0.0  0.0  69736  1704 ?        Ss   19:49   0:00 nginx: master process /usr/sbin/nginx
www-data   496  0.0  0.1  70052  3512 ?        S    19:49   0:00 nginx: worker process
www-data   497  0.0  0.1  70052  3512 ?        S    19:49   0:00 nginx: worker process
www-data   498  0.0  0.1  70052  3512 ?        S    19:49   0:00 nginx: worker process
www-data   500  0.0  0.1  70052  3512 ?        S    19:49   0:00 nginx: worker process
www-data   501  0.0  0.1  70052  3512 ?        S    19:49   0:00 nginx: worker process
www-data   502  0.0  0.1  70052  3512 ?        S    19:49   0:00 nginx: worker process
root       519  0.0  0.1   7640  2752 pts/0    R+   19:49   0:00 ps aux
root@02ccfa7603fb:/# service nginx stop
[ ok ] Stopping nginx: nginx.
root@02ccfa7603fb:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2388   696 pts/0    Ss   18:58   0:00 /bin/sh -c /bin/bash
root         6  0.0  0.1   3992  3260 pts/0    S    18:58   0:00 /bin/bash
root       551  0.0  0.1   7640  2700 pts/0    R+   19:49   0:00 ps aux
root@02ccfa7603fb:/#
```

---

**Mariadb;**

Mysqladmin: utilitario para mudar password
DESCRIPTION
       mysqladmin is a client for performing administrative operations. You can use it to check the server's configuration and current status, to
       create and drop databases, and more.

Mysql_dsafe: inicializa deamon
Mysqld: tb starta deamon, mas congela terminal
Mysqld&: starta no background, se der ctrl-Z volta para o prompt
Kill -9 [pid]: derruba o deamon (tem que ver o pid antes com PS AUX)



```
root@d061cd8d61ea:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2388   764 pts/0    Ss   17:32   0:00 /bin/sh -c /bin/bash
root         6  0.0  0.1   3988  3144 pts/0    S    17:32   0:00 /bin/bash
root       111  0.0  0.0   2388  1616 pts/0    T    20:34   0:00 /bin/sh /usr/bin/mysqld_safe
mysql      228  0.2  3.8 1779144 77840 pts/0   Sl   20:34   0:00 /usr/sbin/mysqld --basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/x86_64-l
root       229  0.0  0.0   4708  1096 pts/0    T    20:34   0:00 logger -t mysqld -p daemon error
root       268  0.0  0.1   7640  2744 pts/0    R+   20:37   0:00 ps aux
root@d061cd8d61ea:/#
```

Cria usuario:

```
mysql -u root -p -e "GRANT USAGE ON *.* TO 'usuario'@'localhost' IDENTIFIED BY 'senha’;”
```

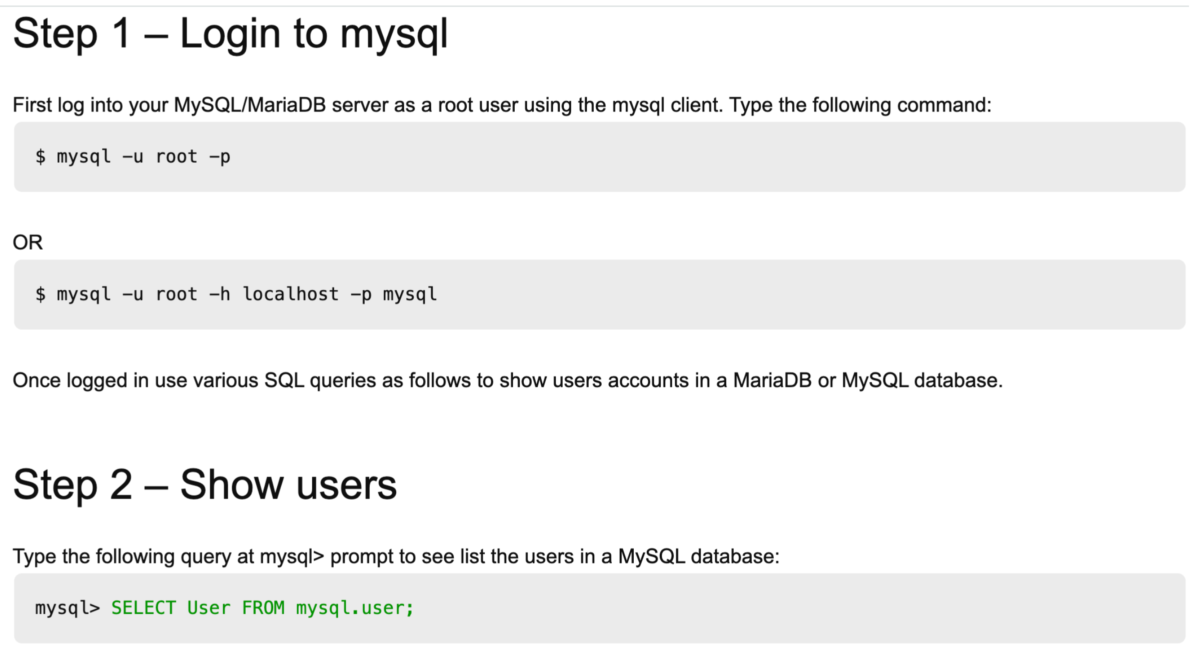
Concede acesso total:

```
mysql -u root -p -e "GRANT ALL ON *.* TO 'usuario'@'localhost';"
```

logando:

```
mysql -u [usuario] -p
    (Entrar senha qdo pedir)
```

Listando usuarios:



## Step 1 – Login to mysql

First log into your MySQL/MariaDB server as a root user using the mysql client. Type the following command:

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

OR

```
$ mysql -u root -h localhost -p mysql
```

Once logged in use various SQL queries as follows to show users accounts in a MariaDB or MySQL database.

## Step 2 – Show users

Type the following query at mysql> prompt to see list the users in a MySQL database:

```
mysql> SELECT User FROM mysql.user;
```

**Dump do database:**
Da para salvar o banco de dados em um arquivo

- Através do comando “Export” dentro do phpmyadmin
- Através da linha de comando

```
mysqldump wordpress > [arquivo].sql
```

Restaurando um database:
Atraves da linha de comando:

```
mysql wordpress < [arquivo].sql
```

***Obs para o Wordpress: ao subir o container pela 1a vez o db do wordpress estara criado (pelo Dockerfile) porem vazio. Ao entrar no wordpress sera feito um processo de configuração. Se derrubar o container e subir de novo devera passar pelo mesmo processo. Para evitar isso, pode salvar o banco de dados do wordpress fora do container e coloca-lo dentro de novo via Dockerfile (ou atraves de um volume***

***persistente do Docker)***

---

DOCKER EXPOSING COMMAND
[networking - Docker EXPOSE vs command line -p option (boot2docker) - Stack Overflow]()
[What is the difference between "expose" and "publish" in Docker? - Stack Overflow]()

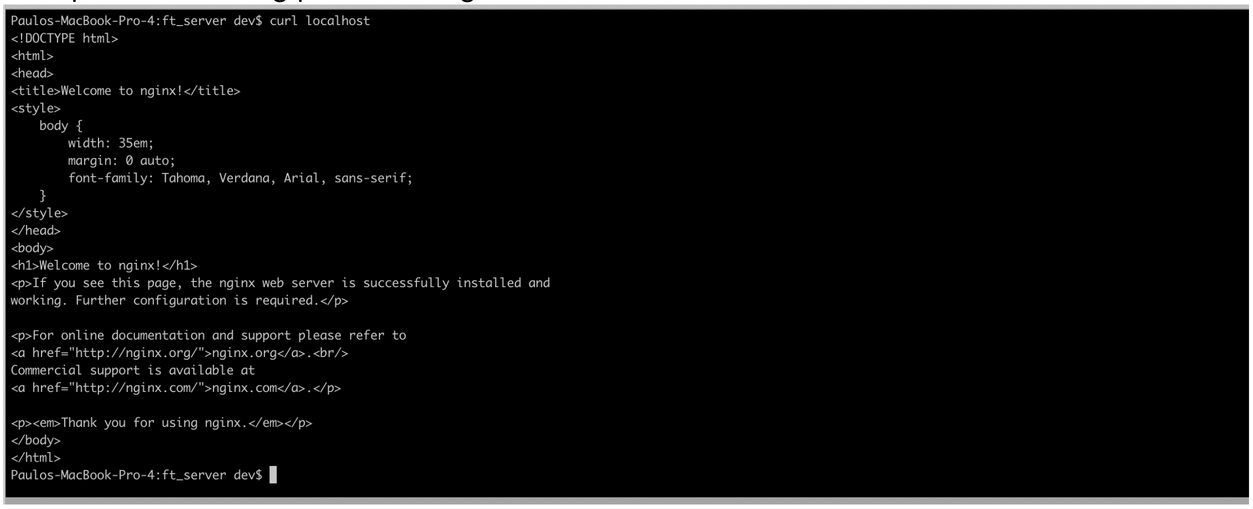
---

TESTANDO O NGINX:
de fora do container rodar:

```
curl localhost
```

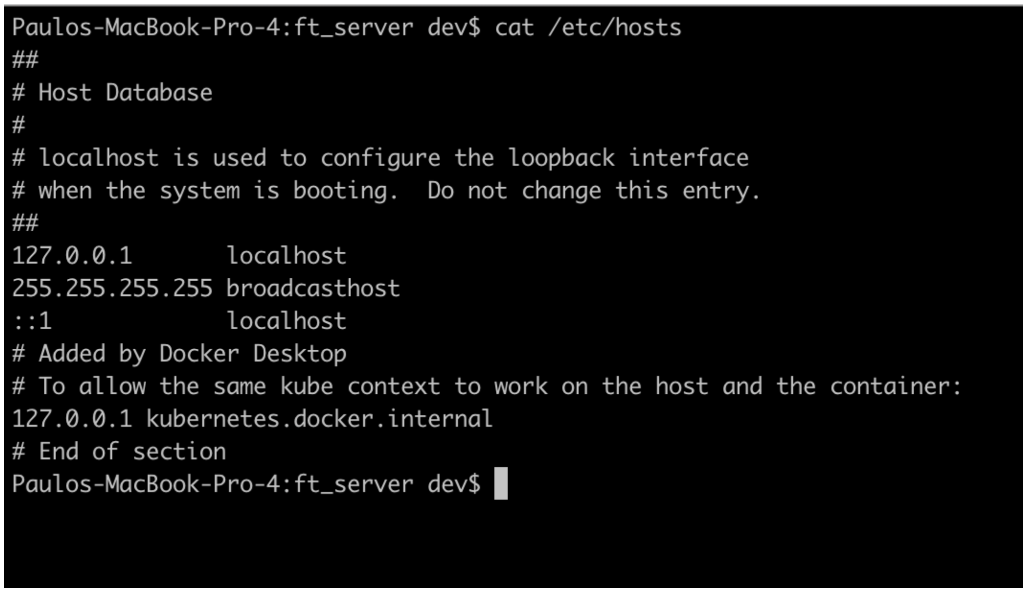
```
curl 127.0.0.1
```

tem que retornar msg padrão do nginx



```
Paulos-MacBook-Pro-4:ft_server dev$ curl localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
Paulos-MacBook-Pro-4:ft_server dev$ 
```

Obs: o local host tem que estar configurado na máquina padrão no /etc/hosts:



```
Paulos-MacBook-Pro-4:ft_server dev$ cat /etc/hosts
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
# Added by Docker Desktop
# To allow the same kube context to work on the host and the container:
127.0.0.1 kubernetes.docker.internal
# End of section
Paulos-MacBook-Pro-4:ft_server dev$ 
```

**Nos Chrome e Safari nao funcionava o "localhost" (nem em modo privado). No Firefox funciona.**

---

**PHPMYADMIN:**

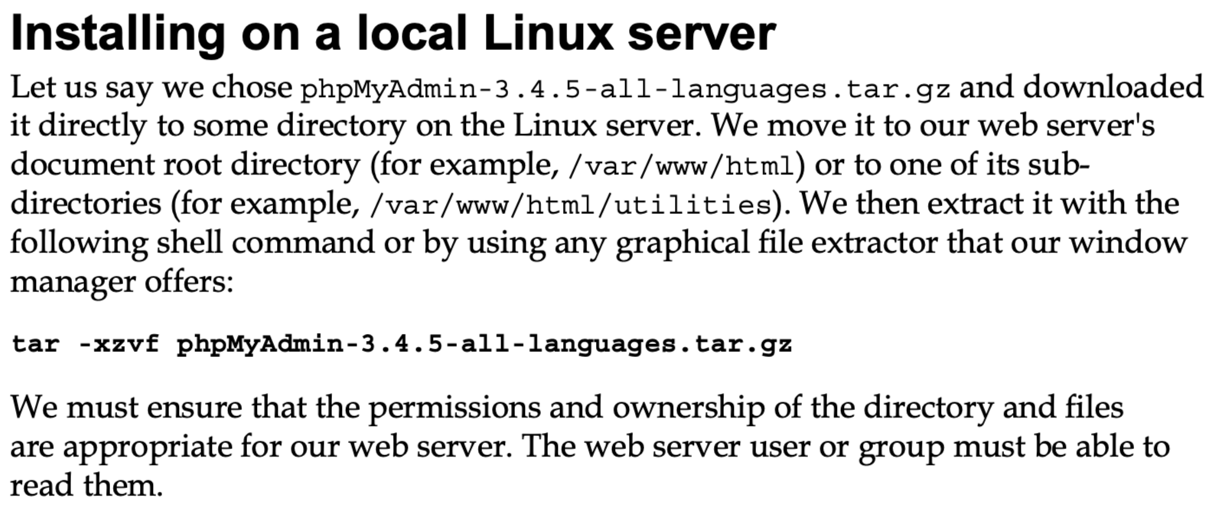
## What is phpMyAdmin?

phpMyAdmin (official home page at `http://www.phpmyadmin.net`) is a web application written in PHP; it contains (like most web applications) XHTML, CSS, and JavaScript client code. This application provides a complete web interface for administering MySQL databases, and is widely recognized as the leading application in this field.

The goal of phpMyAdmin is to offer a complete web-based management of MySQL servers and data, and to keep up with MySQL and web standards evolution. While the product is always evolving, it supports all standard operations along with extra features.

The development team constantly fine-tunes the product based on the reported bugs and requested features, releasing new versions regularly.

phpMyAdmin offers features that cover basic MySQL database and table operations. It also has an internal system that maintains metadata to support advanced features. Finally, system administrators can manage users and privileges from phpMyAdmin. It is important to note that phpMyAdmin's choice of available operations depends on the rights the user has on a specific MySQL server.



## Installing on a local Linux server

Let us say we chose `phpMyAdmin-3.4.5-all-languages.tar.gz` and downloaded it directly to some directory on the Linux server. We move it to our web server's document root directory (for example, `/var/www/html`) or to one of its sub-directories (for example, `/var/www/html/utilities`). We then extract it with the following shell command or by using any graphical file extractor that our window manager offers:

```
tar -xzvf phpMyAdmin-3.4.5-all-languages.tar.gz
```

We must ensure that the permissions and ownership of the directory and files are appropriate for our web server. The web server user or group must be able to read them.

Link para gerar blowfish:
https://phpsolved.com/phpmyadmin-blowfish-secret-generator/?g=
[insert_php]echo%20$code;[/insert_php]

---

**Wordpress:**
Apos instalado, logar via browser (localhost), fazer o processo de setup interativo e depois copiar o arquivo de configuração gerado (wp-config.php) para o /srcs.
Ele sera a base para jogar dentro dos containers depois.

---

Meu projeto;

- user MariaDB: root senha: “"
- user WP: wordpress senha 1234