

AWS E LARAVEL

Configurando um servidor EC2 Amazon com Laravel

Eucurso.com.br

Este é um ebook produzido para o portal EuCurso.com.br.

Você pode utilizá-lo livremente para qualquer fim, inclusive comercial. Porém não está autorizada a distribuição do ebook em qualquer portal. Divulgações podem ser feitas com links para o portal eucurso.com.br.

Todos os direitos reservados para Alfamídia

Vamos Configurar o Framework Laravel na Nuvem

Neste ebook nós vamos guia-lo no passo a passo para configurar o ambiente Laravel na nuvem Amazon.

Nós vamos assumir que temos uma instância Amazon Linux 2 AMI recém-criada. Nosso e-book **AWS - Instanciando um Servidor na Nuvem Amazon** mostra o passo a passo para criar esta instância e pode ser baixado gratuitamente do portal eucurso.com.br.

Este ebook é principalmente prático, mas começaremos com alguns conceitos básicos que são essenciais para entendermos o que será construído nas páginas seguintes:

AWS: AWS é Amazon Web Services, e é a plataforma de nuvem da Amazon.

Amazon EC2: O EC2 (Elastic Cloud Computing) é um dos muitos serviços oferecidos no AWS, e será o que utilizaremos neste ebook. É através do EC2 que executaremos um servidor virtual no qual nossa aplicação Laravel será executada.

Laravel: O Laravel é um framework PHP. De forma resumida, através do Laravel conseguimos criar aplicações PHP de forma mais rápida.

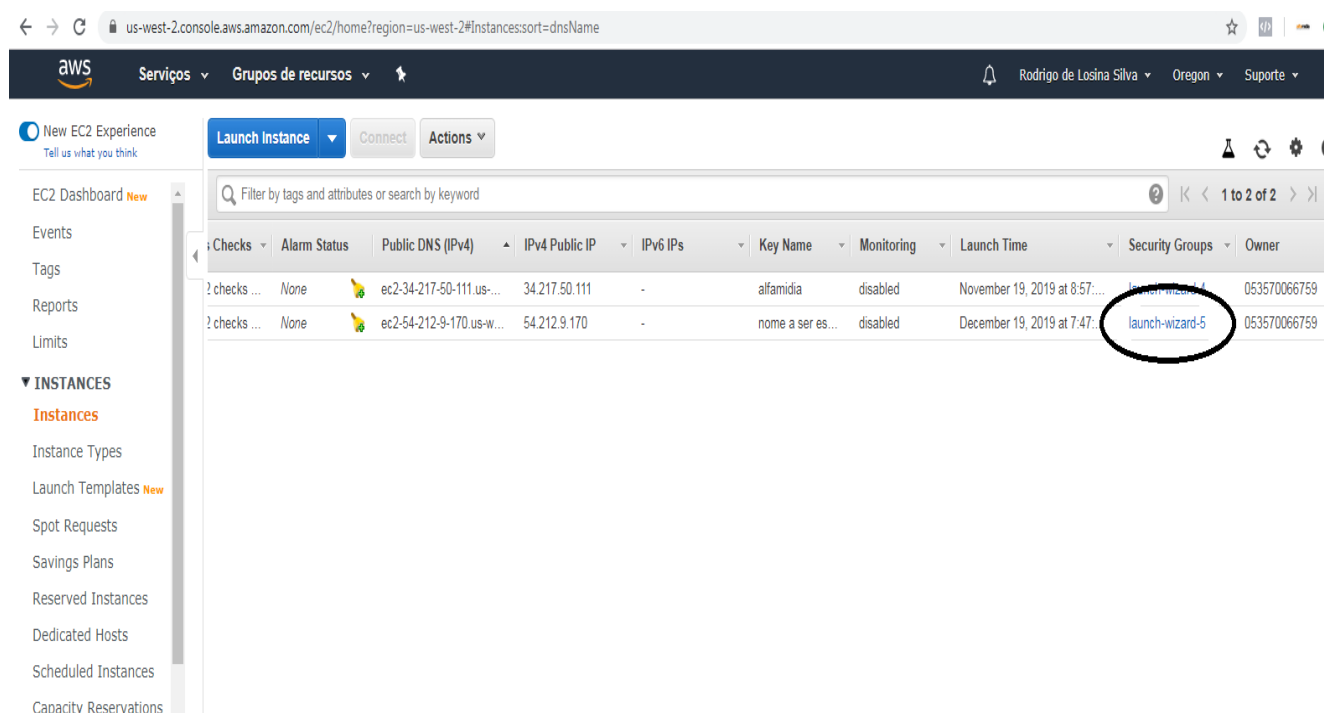
MariaDB: MariaDB é um banco de dados baseado no MySQL.

Liberando Portas no Servidor AWS

Quando você digita um endereço de um site, o que ocorre é que um servidor web na internet irá tratar sua requisição e respondê-la, enviando uma página HTML.

Pois bem, esta requisição chega até o servidor através de uma ‘porta’, que é definida por um número e precisa ser liberada no servidor. Então, nosso próximo passo será liberar as portas vinculadas ao HTTP e HTTPS em nossa instância.

No EC2 Dashboard, temos um link para o “Security Groups”, conforme mostra na figura a seguir (você provavelmente terá que deslocar a barra de scroll para a direita, pois é uma das últimas colunas).



Clique neste link para editar as informações de segurança. Na tela a seguir selecione a aba *Inbound*, conforme mostra a figura, e clique em *Edit*.

us-west-2.console.aws.amazon.com/ec2/home?region=us-west-2#SecurityGroups:search=sg-0bdabd0175b84c435;sort=groupid

aws Serviços Grupos de recursos

New EC2 Experience Tell us what you think

Create Security Group Actions

search: sg-0bdabd0175b84c435 Add filter

Name	Group ID	Group Name	VPC ID	Owner	Description
	sg-0bdabd0175b84c435	launch-wizard-5	vpc-2116b244	053570066759	launch-wizard-5 created 2019-12-19T19:34:02.185-03:00

Security Group: sg-0bdabd0175b84c435

Description Inbound Outbound Tags

Edit

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	Anywhere	

Em seguida, na janela que será exibida, selecione Add Rule, com Type = HTTP e Source=Anywhere, e clique SAVE. Como na figura a seguir. Faça o mesmo para Type = HTTPS.

Edit inbound rules

Type	Protocol	Port Range	Source	Description	
HTTP	TCP	80	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom	:::0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom	0.0.0.0/0	e.g. SSH for Admin Desktop
SSH	TCP	22	Custom	:::0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere	0.0.0.0, :::0	e.g. SSH for Admin Desktop

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Cancel Save

Instalação Básica do LAMP

A própria Amazon já possui um tutorial que mostra a instalação de boa parte dos serviços que iremos precisar para configurar nosso ambiente de desenvolvimento Laravel, e vamos aqui seguir reproduzindo os passos indicados no tutorial.

Estes serviços são popularmente conhecidos como LAMP (ou WAMP em Windows), e incluem principalmente um servidor Web Apache e um servidor de banco de dados, no caso o MariaDB.

Faça a conexão com sua instância do servidor Linux, como mostrado no ebook **AWS - Instanciando um Servidor na Nuvem Amazon**. Após, copie os comandos a seguir para o shell do Linux.

```
sudo yum update -y
sudo amazon-linux-extras install -y lamp-mariadb10.2-php7.2
php7.2
sudo yum install -y httpd mariadb-server
sudo systemctl start httpd
sudo systemctl enable httpd
sudo chown -R ec2-user /var/www
sudo chmod 2775 /var/www && find /var/www -type d -exec sudo
chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0664 {} \;
cd /var/www/html
echo "<?php echo 'teste'; ?>" > teste.php
```

Dica: você pode copiar as linhas diretamente do ebook, e clicar com o botão direito na janela do Putty, que ele automaticamente copiará os comandos para o Shell. Copie todas elas, ou linha a linha para acompanhar a ação de cada uma.

Veja, a seguir, algumas informações sobre os comandos executados:

sudo: o comando sudo indica que o comando seguinte será executado como super usuário. Assim, “sudo yum update -y” nada mais é que o comando “yum update -y” executado com permissão de superusuário.

yum: o yum é uma ferramenta para gerenciar e instalar pacotes.

Dica: digitando “*man yum*” você poderá ver o manual de uso do yum, incluindo os parâmetros disponíveis no comando.

Os três primeiros comandos fazem a atualização e instalação dos pacotes do Apache, MariaDB e PHP.

sudo systemctl start httpd: este é o comando que inicializa o Apache. A partir deste momento seu servidor já está respondendo a requisições Web.

sudo systemctl enable httpd: com este comando você configura para o Apache ser iniciado sempre que o sistema for carregado.

sudo chown -R ec2-user /var/www

sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;
find /var/www -type f -exec sudo chmod 0664 {} \;

Estes dois comandos tornam o ec2-user (o seu usuário) como o dono de tudo que está abaixo de /var/www, que é a pasta onde serão colocados os arquivos acessíveis pela internet.

cd /var/www/html: altera sua pasta atual para a pasta html.

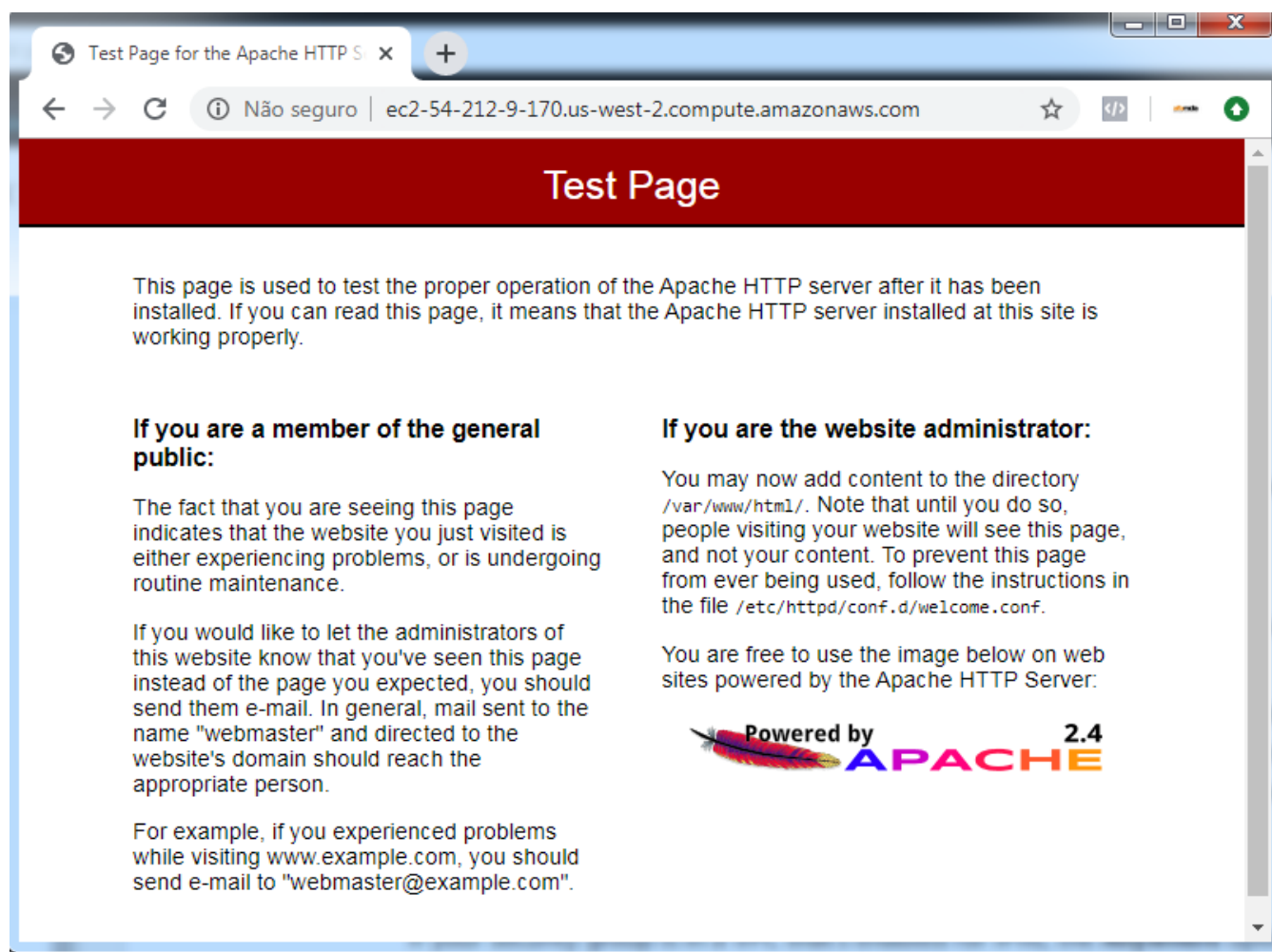
echo "<?php echo 'teste'; ?>" > teste.php: o comando “echo” simplesmente exibe na tela o que você digitar. Nós escrevemos um comando mínimo em PHP, e utilizamos o comando “>” para redirecionar a saída, da tela para o arquivo teste.php.

Como resultado destas ações, temos os servidores do Apache, do PHP e do MariaDB instalados. Mudamos o dono da pasta /var/www/html para nosso usuário, e ainda criamos um arquivo *teste.php* com um pequeno código PHP que exibe “teste”. Acompanhe a seguir os testes desta etapa de nosso processo:

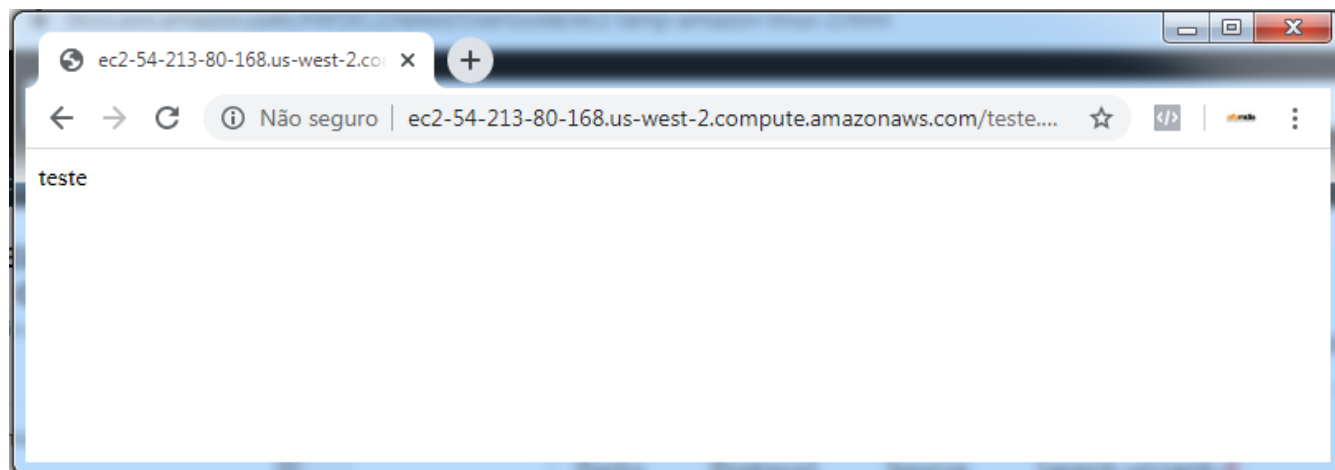
Testando o Apache

Nós acabamos de instalar o servidor Apache e liberar as portas de acesso ao mesmo. Basta agora acessarmos o DNS público para conferirmos que o Apache está funcionando.

No EC Dashboard, já vimos no ebook anterior como obter o DNS público, pois utilizamos o mesmo para fazer o acesso através do Putty. Digite o mesmo DNS em seu browser, que a página de teste do Apache deverá ser exibida, como na figura a seguir:



Como dissemos, além de instalar o Apache, também criamos um arquivo “teste.php” para testar a instalação do PHP. Acesse o endereço <http://DOMINIO/teste.php> e confira se o resultado é equivalente a imagem a seguir:



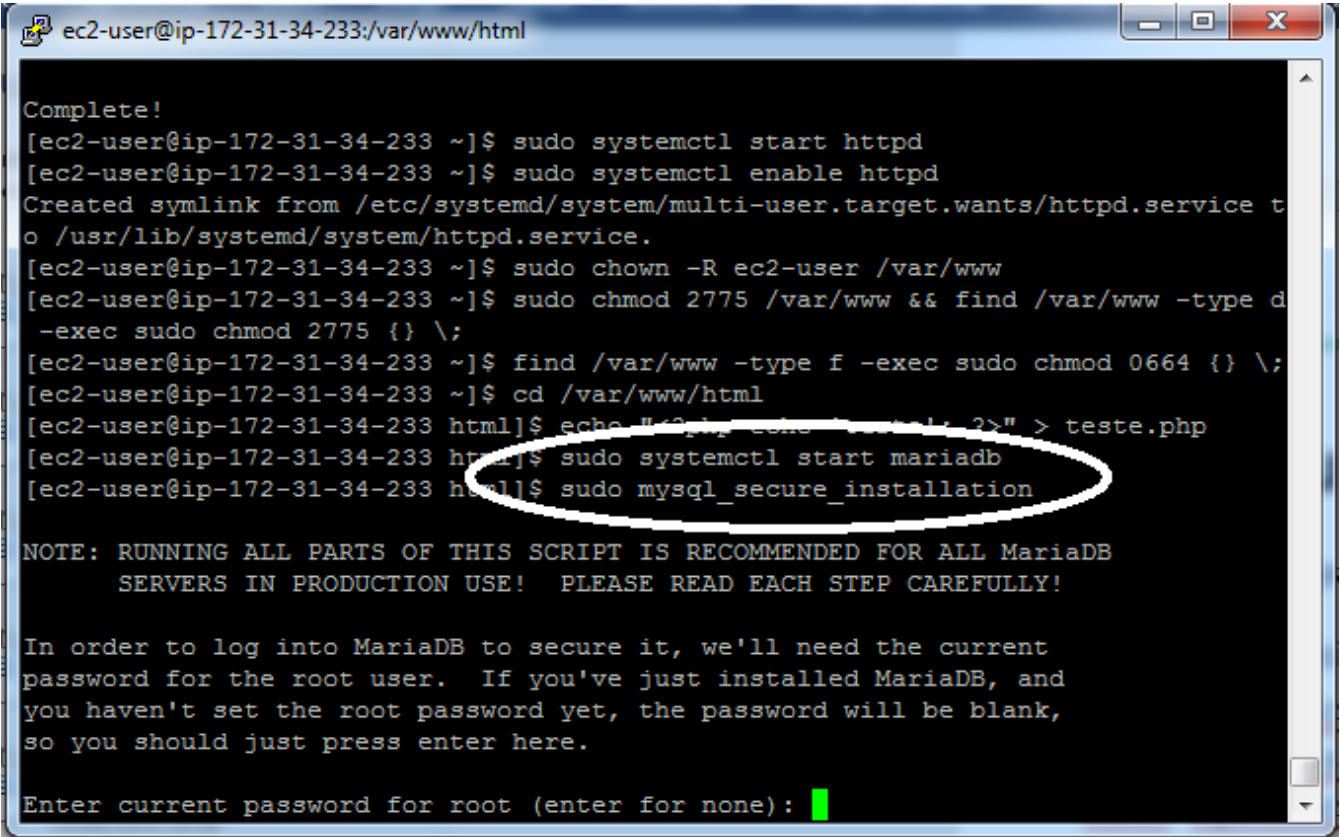
Configurando o MariaDB

Nosso próximo passo é finalizar a configuração do servidor de bancos de dados.

Utilize a sequência de comandos a seguir:

```
sudo systemctl start mariadb
sudo mysql_secure_installation
```

Observe a imagem a seguir. Apenas tecle ENTER, e selecione “y” quando perguntado se deseja criar uma nova senha. Digite uma nova senha para o servidor de banco de dados. Lembre-se que você precisará desta senha para fazer o acesso ao MariaDB.



```
ec2-user@ip-172-31-34-233:/var/www/html

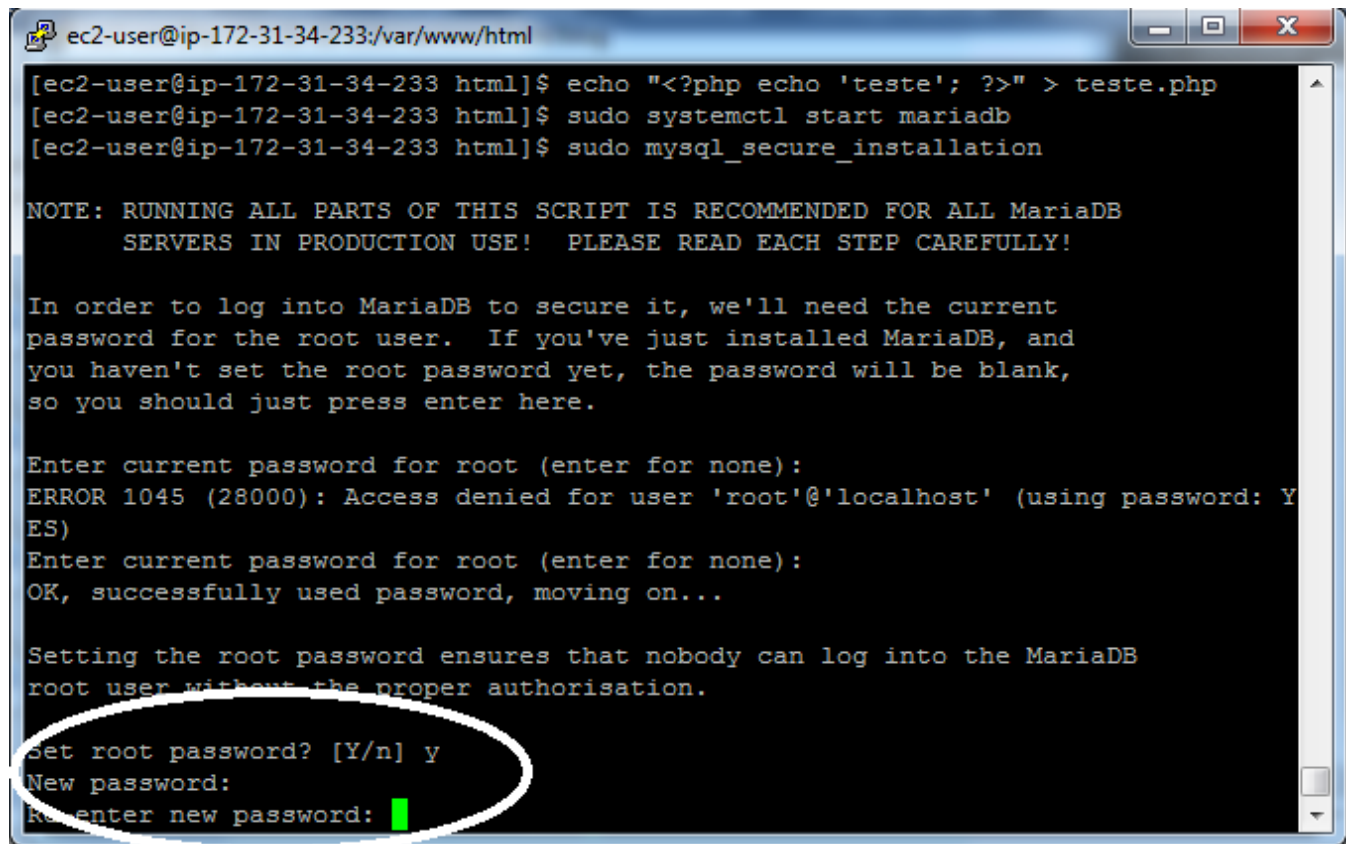
Complete!
[ec2-user@ip-172-31-34-233 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-34-233 ~]$ sudo systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-34-233 ~]$ sudo chown -R ec2-user /var/www
[ec2-user@ip-172-31-34-233 ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod 2775 {} \;
[ec2-user@ip-172-31-34-233 ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
[ec2-user@ip-172-31-34-233 ~]$ cd /var/www/html
[ec2-user@ip-172-31-34-233 html]$ echo "php echo 'teste!'; ?&gt;" &gt; teste.php
[ec2-user@ip-172-31-34-233 html]$ sudo systemctl start mariadb
[ec2-user@ip-172-31-34-233 html]$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):</pre
```

Após digitar ENTER, digite y e escreva duas vezes a nova senha:

A terminal window titled 'ec2-user@ip-172-31-34-233:/var/www/html' with standard window controls. The terminal shows the execution of several commands: creating a test PHP file, starting the MariaDB service, and running the security script. The script prompts for the root password, which is entered as 'y'. It then prompts for a new password, which is entered as 'y'. The prompt 'Re-enter new password:' is followed by a green cursor. The text 'Set root password? [Y/n] y' and the subsequent 'New password:' and 'Re-enter new password:' lines are circled in white.

```
ec2-user@ip-172-31-34-233:/var/www/html
[ec2-user@ip-172-31-34-233 html]$ echo "<?php echo 'teste'; ?>" > teste.php
[ec2-user@ip-172-31-34-233 html]$ sudo systemctl start mariadb
[ec2-user@ip-172-31-34-233 html]$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

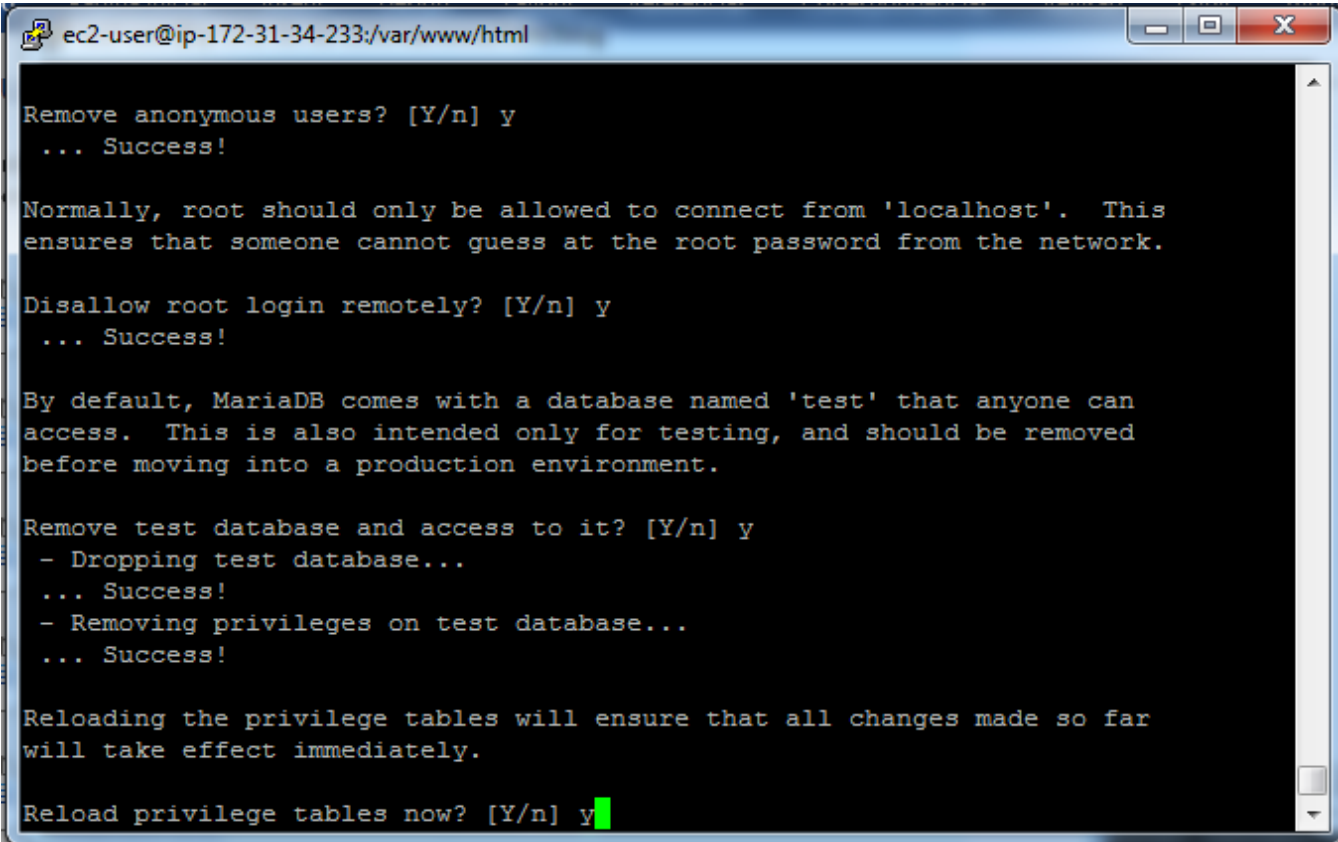
In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password: y
```

Selecione “y” para as perguntas seguintes, que irão remover o usuário anônimo e outras configurações que podem representar riscos de segurança em ambientes de produção.

A terminal window titled 'ec2-user@ip-172-31-34-233:/var/www/html' with standard window controls. The terminal output shows the following steps: 1. 'Remove anonymous users? [Y/n] y' followed by '... Success!'. 2. A message: 'Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.' 3. 'Disallow root login remotely? [Y/n] y' followed by '... Success!'. 4. A message: 'By default, MariaDB comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.' 5. 'Remove test database and access to it? [Y/n] y' followed by '- Dropping test database...', '... Success!', '- Removing privileges on test database...', and '... Success!'. 6. A message: 'Reloading the privilege tables will ensure that all changes made so far will take effect immediately.' 7. 'Reload privilege tables now? [Y/n] y' with a green cursor at the end.

```
ec2-user@ip-172-31-34-233:/var/www/html
Remove anonymous users? [Y/n] y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
```

Utilize o comando a seguir para incluir também o MariaDB na inicialização do sistema:

```
sudo systemctl enable mariadb.
```

Instalando o phpMyAdmin

O phpMyAdmin é um software web desenvolvido em PHP que vai nos permitir acessar interativamente nosso banco de dados MariaDB. Para este ebook ele será útil como forma de criarmos e testarmos o acesso ao banco de dados através do Laravel.

Como nos exemplos anteriores, apenas copie o código a seguir para o shell, que o phpMyAdmin será instalado.

```
sudo yum install php-mbstring -y

sudo systemctl restart httpd

sudo systemctl restart php-fpm

cd /var/www/html

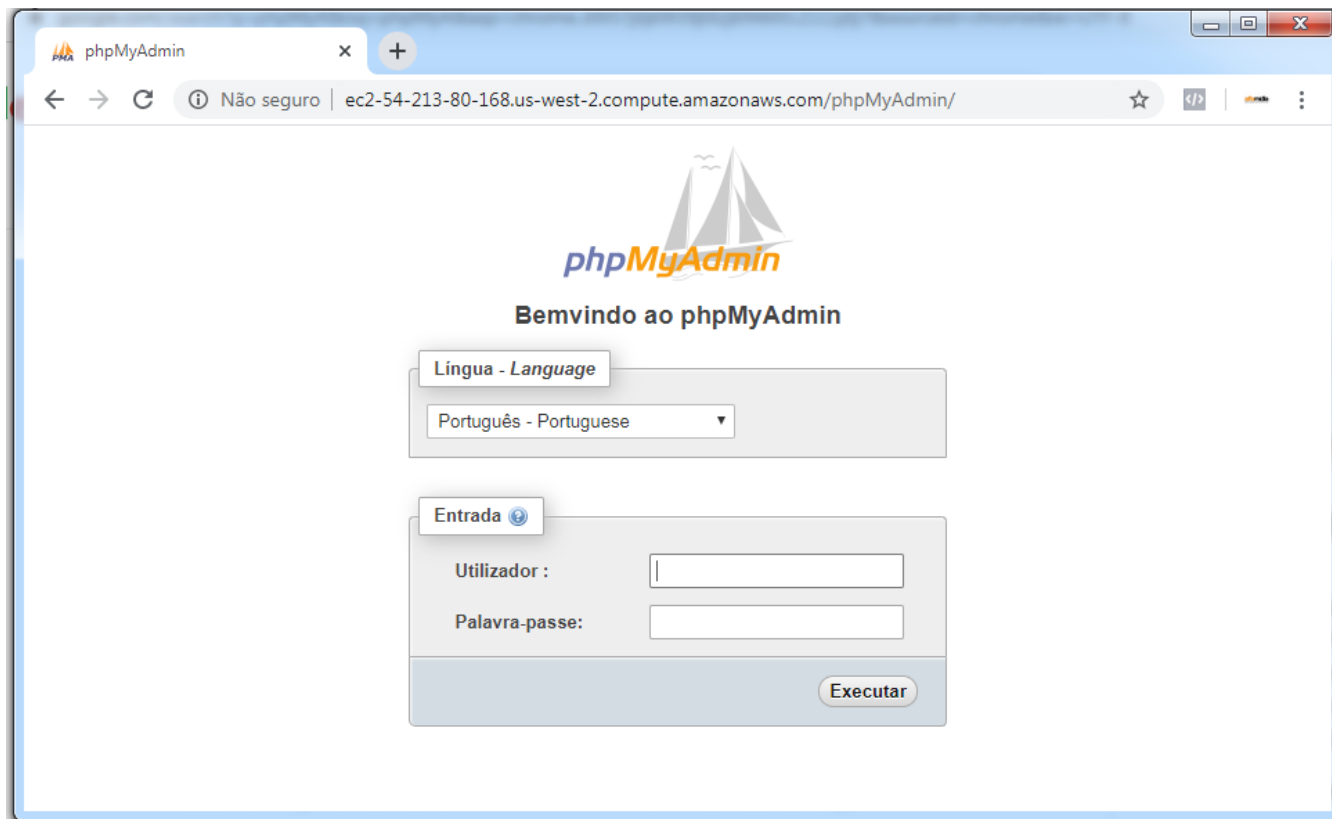
wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-
all-languages.tar.gz

mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all-
languages.tar.gz -C phpMyAdmin --strip-components 1

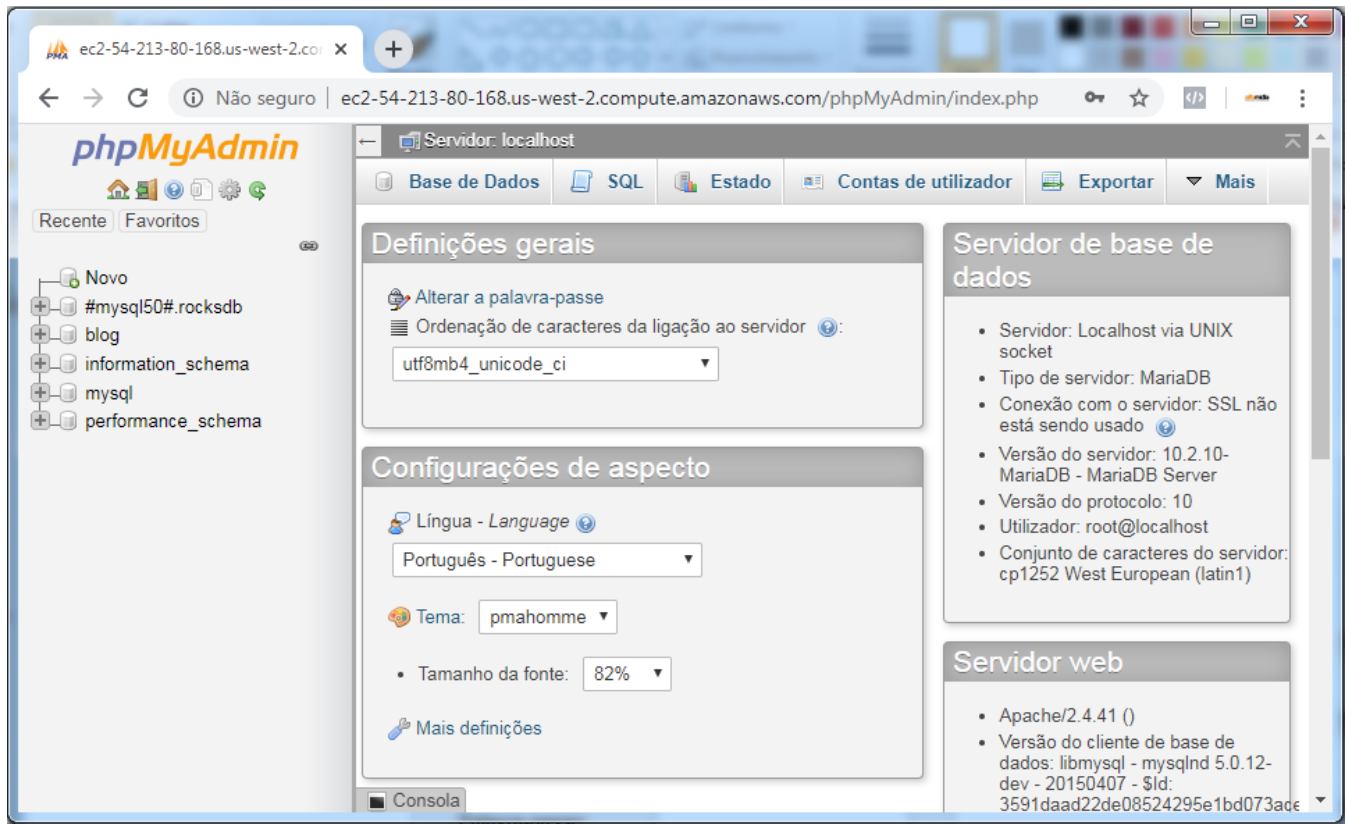
rm phpMyAdmin-latest-all-languages.tar.gz
```

O que fizemos com esta sequência de códigos foi instalar uma biblioteca do php necessário para o phpMyAdmin (o mbstring) e baixar e copiar para dentro da pasta html uma pasta denominada phpMyAdmin com todo o código da ferramenta.

Acessando <http://dominio/phpMyAdmin>, deve ser exibida uma imagem equivalente a esta:



Entre com “root” como utilizador, e com a senha que você criou anteriormente para o MariaDB, que uma tela como esta será exibida:



Instalando Ferramentas para o Laravel

Para instalar o Laravel, teremos que instalar algumas ferramentas. Execute os comandos a seguir:

```
sudo yum install git -y
```

Com este comando estamos instalando o GIT. Trata-se de um sistema de controle de versões, e será muito importante para agilizar a instalação de aplicações Laravel e manter o versionamento à medida que você desenvolver aplicações próprias.

```
sudo yum install php-xml -y
```

O *php-xml* é uma biblioteca php que é necessária para instalarmos o Laravel.

```
curl -sS https://getcomposer.org/installer | php
```

```
sudo mv composer.phar /usr/bin/composer
```

```
chmod +x /usr/bin/composer
```

Com estes 3 comandos fizemos o download do Composer da internet e configuramos para ele poder ser executado a partir de qualquer pasta do *shell*.

```
sudo /bin/dd if=/dev/zero of=/var/swap.1 bs=1M count=1024
```

```
sudo /sbin/mkswap /var/swap.1
```

```
sudo /sbin/swapon /var/swap.1
```

Estes comandos irão disponibilizar uma área de memória adicional para seu Linux, o que será necessário para instalar o Laravel.

Instalando o Laravel

Execute os comandos a seguir:

```
cd /var/www/html
```

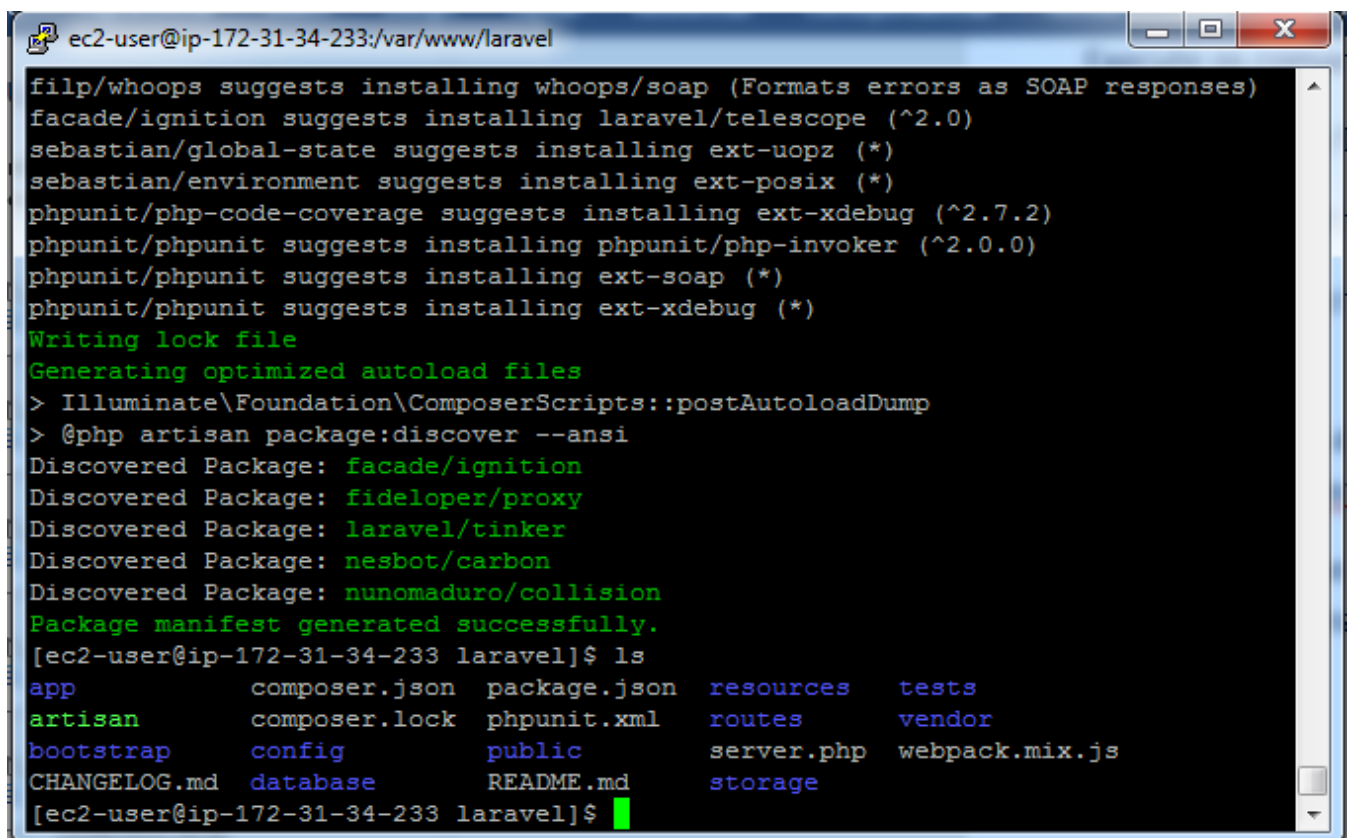
```
git clone https://github.com/laravel/laravel.git
```

```
cd laravel
```

```
composer install
```

Nós navegamos até a pasta inicial de nosso servidor web, utilizamos o *GIT* para carregar a biblioteca Laravel, e executamos o *composer* para instalar o Laravel.

Executando na sequência o comando *ls*, que mostra o conteúdo da pasta, enxergamos os arquivos da instalação do laravel:

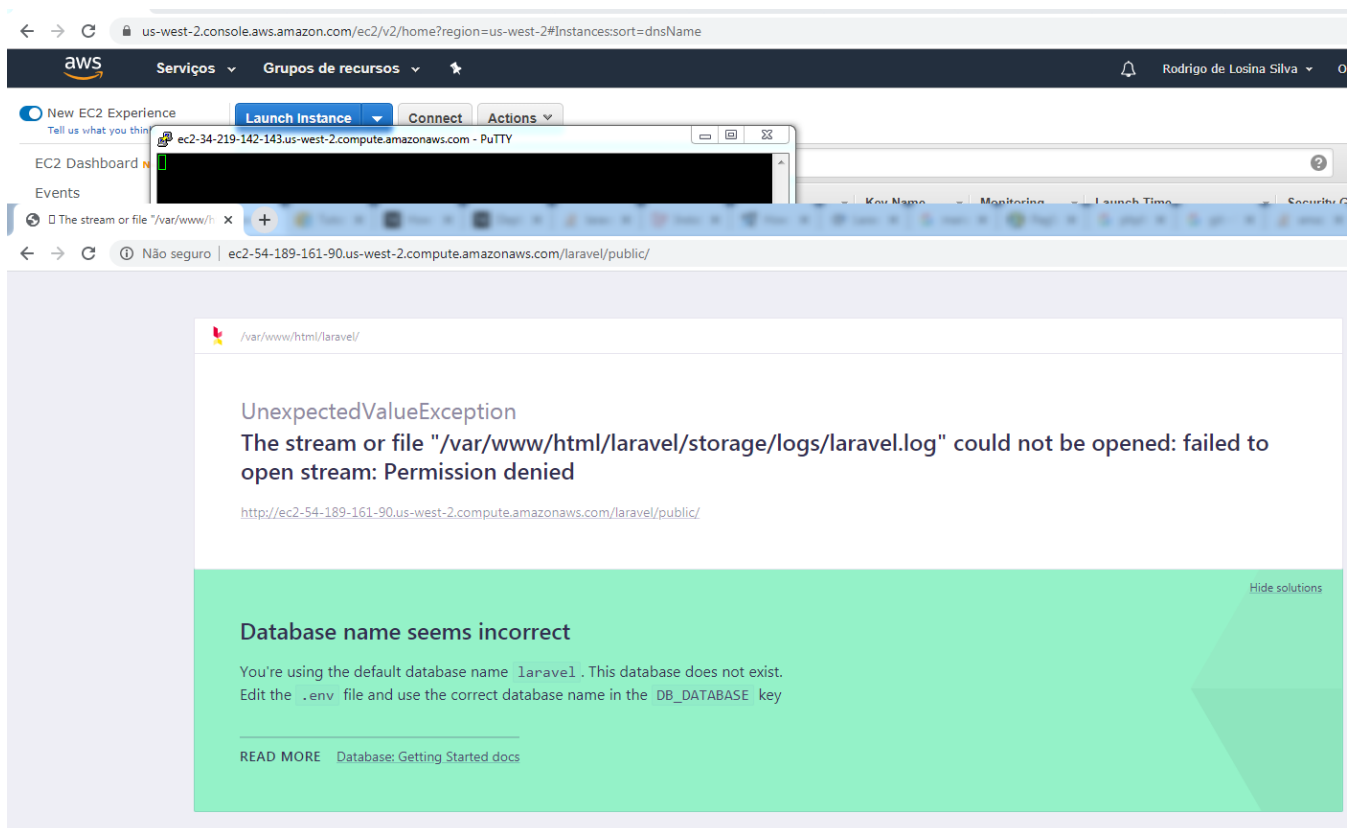


```
ec2-user@ip-172-31-34-233:/var/www/laravel
filp/whoops suggests installing whoops/soap (Formats errors as SOAP responses)
facade/ignition suggests installing laravel/telescope (^2.0)
sebastian/global-state suggests installing ext-uopz (*)
sebastian/environment suggests installing ext-posix (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.7.2)
phpunit/phpunit suggests installing phpunit/php-invoker (^2.0.0)
phpunit/phpunit suggests installing ext-soap (*)
phpunit/phpunit suggests installing ext-xdebug (*)
Writing lock file
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: facade/ignition
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
[ec2-user@ip-172-31-34-233 laravel]$ ls
app                composer.json      package.json       resources          tests
artisan            composer.lock      phpunit.xml        routes             vendor
bootstrap          config             public             server.php         webpack.mix.js
CHANGELOG.md       database           README.md          storage
```

Nosso passo seguinte é criar o arquivo de configuração do laravel, que é chamado de `.env`. Inicialmente vamos apenas mover um arquivo pré definido:

```
mv .env.example .env
```

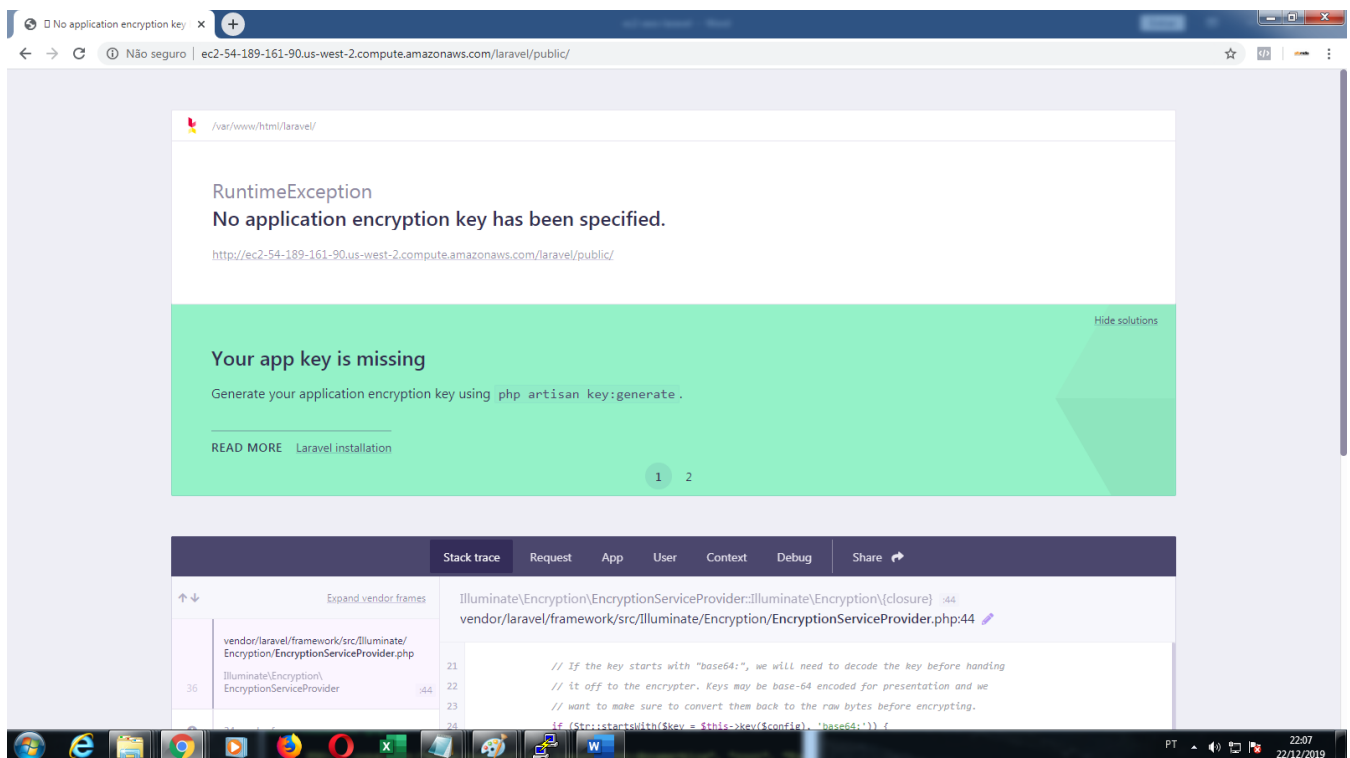
Neste momento nossa aplicação laravel estará instalada, porém apresentando alguns erros ainda. Teste ela acessando o endereço <http://dominio/laravel/public> como na imagem a seguir:



O primeiro problema que devemos resolver é alterar a permissão da pasta *storage* dentro de laravel, pois a aplicação precisa ter permissão para escrever e criar arquivos nesta pasta. Execute o comando a seguir:

```
chmod -R 777 storage
```

Executando novamente a página web, podemos observar uma nova mensagem de erro:



O Laravel exige a criação de uma chave criptografada para manter a segurança no envio de informações de formulário. Para gerar esta chave, execute o comando a seguir:

```
php artisan key:generate
```

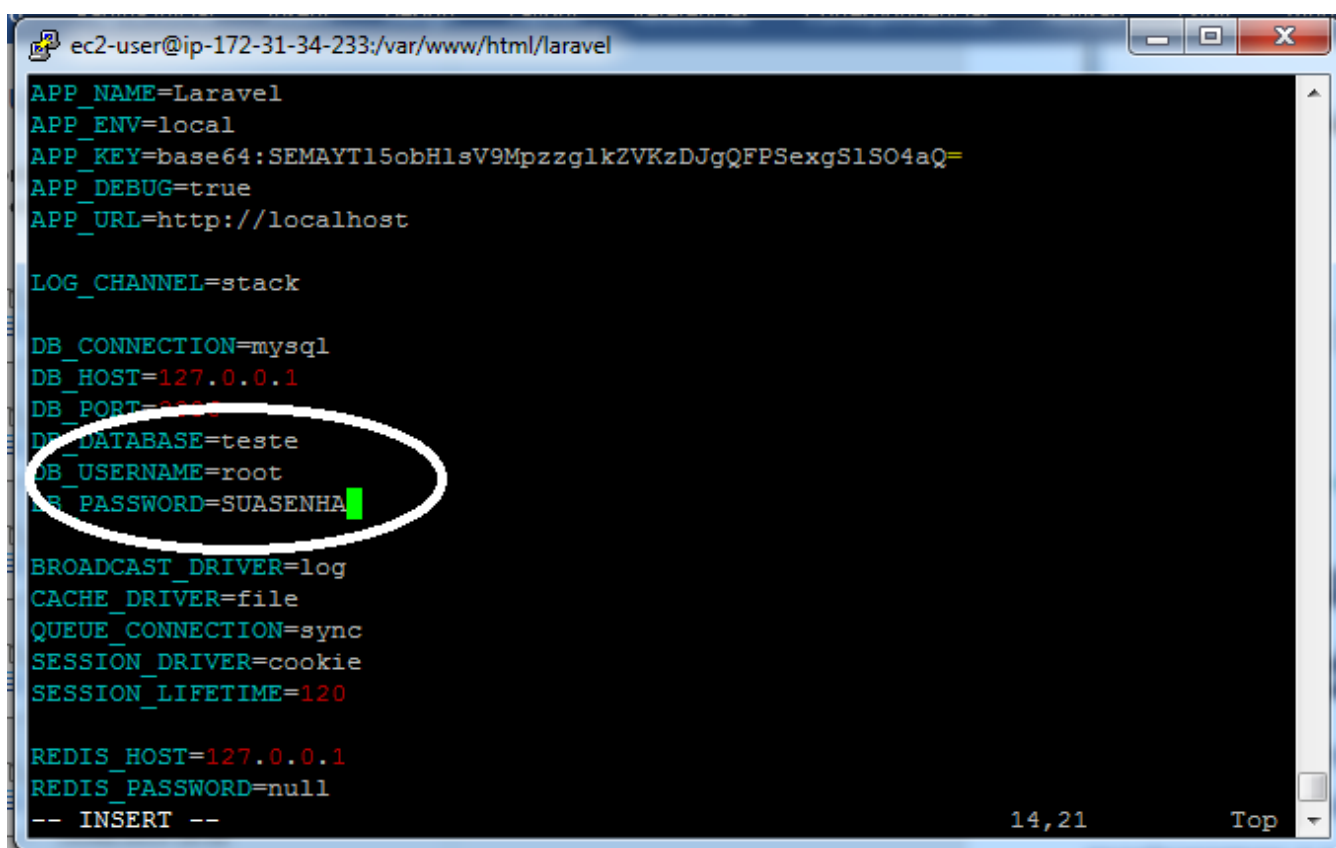
Executando novamente a aplicação Web, temos nosso aplicativo laravel rodando:



Configurando o Acesso ao Banco de Dados

Nós estamos, nesta etapa, com todo o ambiente configurado e uma aplicação laravel executando, porém ainda sem acesso a bancos de dados. Em um futuro ebook mostraremos a construção de um CRUD em laravel, mas nosso objetivo aqui é configurar o acesso correto ao banco de dados.

Para isso teremos que editar o arquivo .env. Se você está familiarizado com um editor como o vi, isso é relativamente simples. Apenas digite “vi .env”, e altere os dados conforme mostrado a seguir:



```
ec2-user@ip-172-31-34-233:/var/www/html/laravel
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:SEMayTl5obHlsV9Mpzzg1kZVKzDJgQFPSexgSlSO4aQ=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

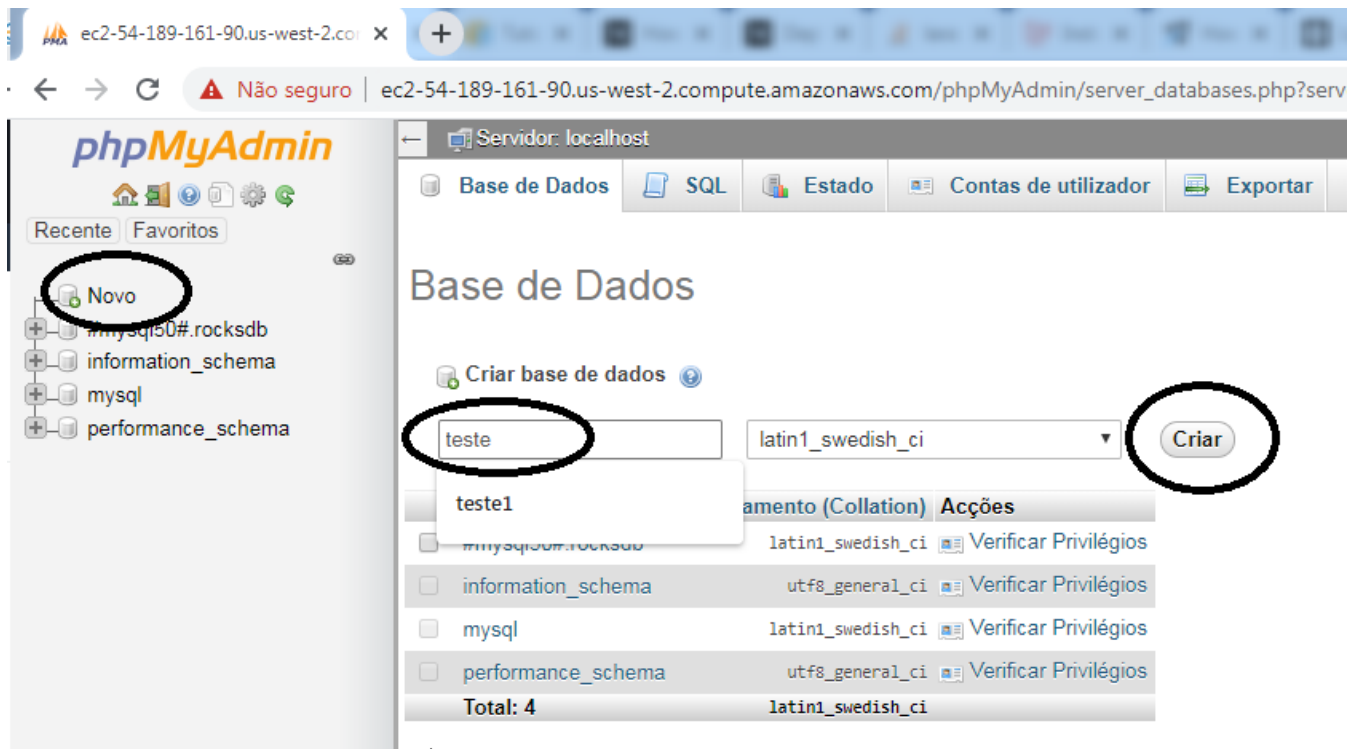
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=teste
DB_USERNAME=root
DB_PASSWORD=SUASENHA

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=cookie
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
-- INSERT --
```

Dica: o VI é um editor que altera entre um modo de comandos e um modo de edição. Inicialmente digite “i”, para poder editar. Faça as alterações, digite ESC, e na sequência “:x” para gravar e sair do editor.

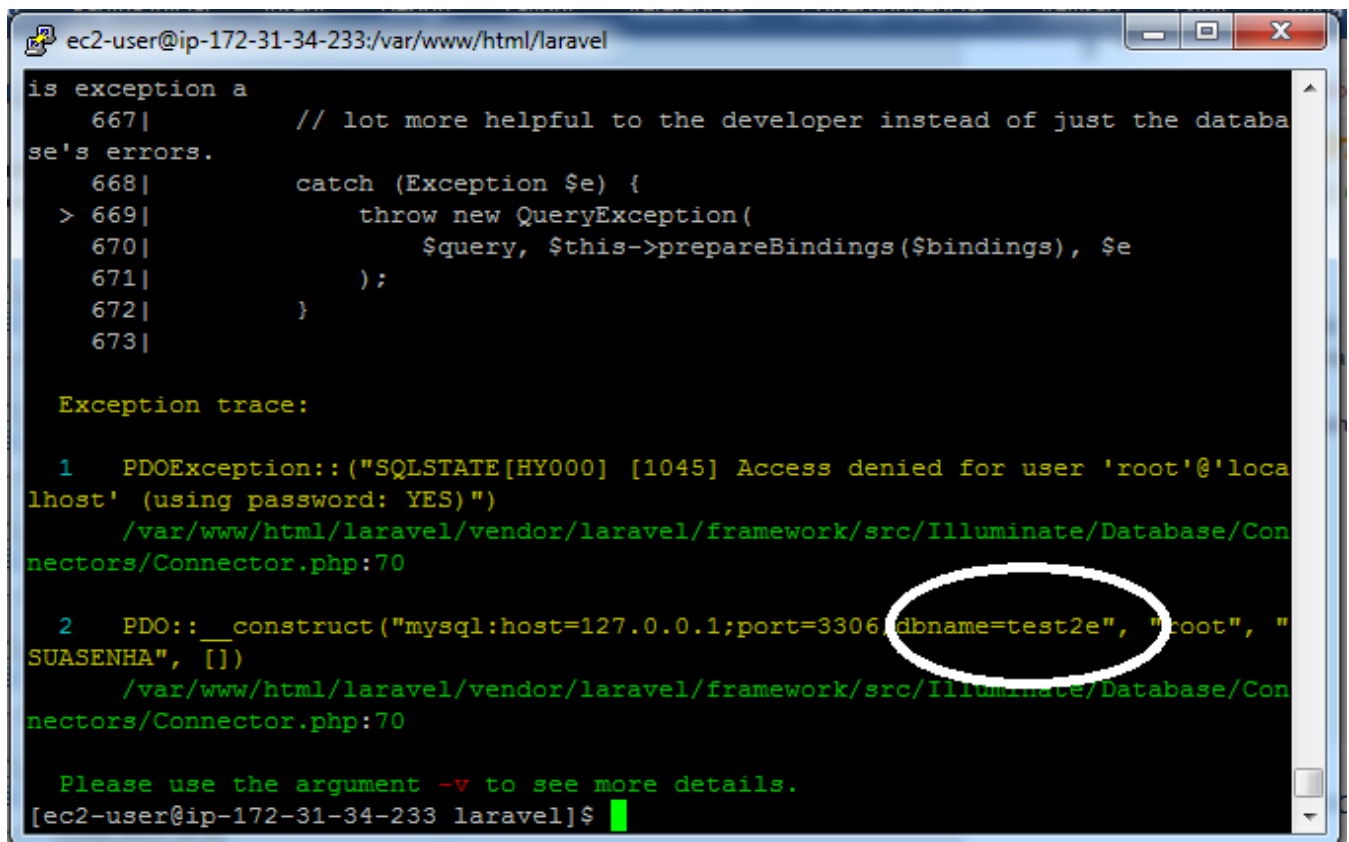
Para testar se o acesso ao banco de dados está funcionando corretamente, cria o banco de dados teste através do phpMyAdmin, conforme figura a seguir:



A seguir execute os comandos:

```
php artisan make:migration tabela_produtos --create=produtos
php artisan migrate
```

Na figura a seguir, alteramos o arquivo .env, trocando o banco de dados de “teste2” para “test2e” para mostrar o tipo de mensagem de erro que será mostrada se o acesso do laravel ao banco de dados não estiver funcionando.



```
ec2-user@ip-172-31-34-233:/var/www/html/laravel
is exception a
667| // lot more helpful to the developer instead of just the databa
se's errors.
668| catch (Exception $e) {
> 669|     throw new QueryException(
670|         $query, $this->prepareBindings($bindings), $e
671|     );
672| }
673|

Exception trace:

1  PDOException::("SQLSTATE[HY000] [1045] Access denied for user 'root'@'loca
localhost' (using password: YES)")
/var/www/html/laravel/vendor/laravel/framework/src/Illuminate/Database/Con
nectors/Connector.php:70

2  PDO::__construct("mysql:host=127.0.0.1;port=3306;dbname=test2e", "root", "
SUASENHA", [])
/var/www/html/laravel/vendor/laravel/framework/src/Illuminate/Database/Con
nectors/Connector.php:70

Please use the argument -v to see more details.
[ec2-user@ip-172-31-34-233 laravel]$
```

Não sendo exibida nenhuma mensagem de erro, seu ambiente está configurado e você poderá começar o desenvolvimento de sua primeira aplicação laravel.