

Docker Desktop em Ambiente Windows 64 bits

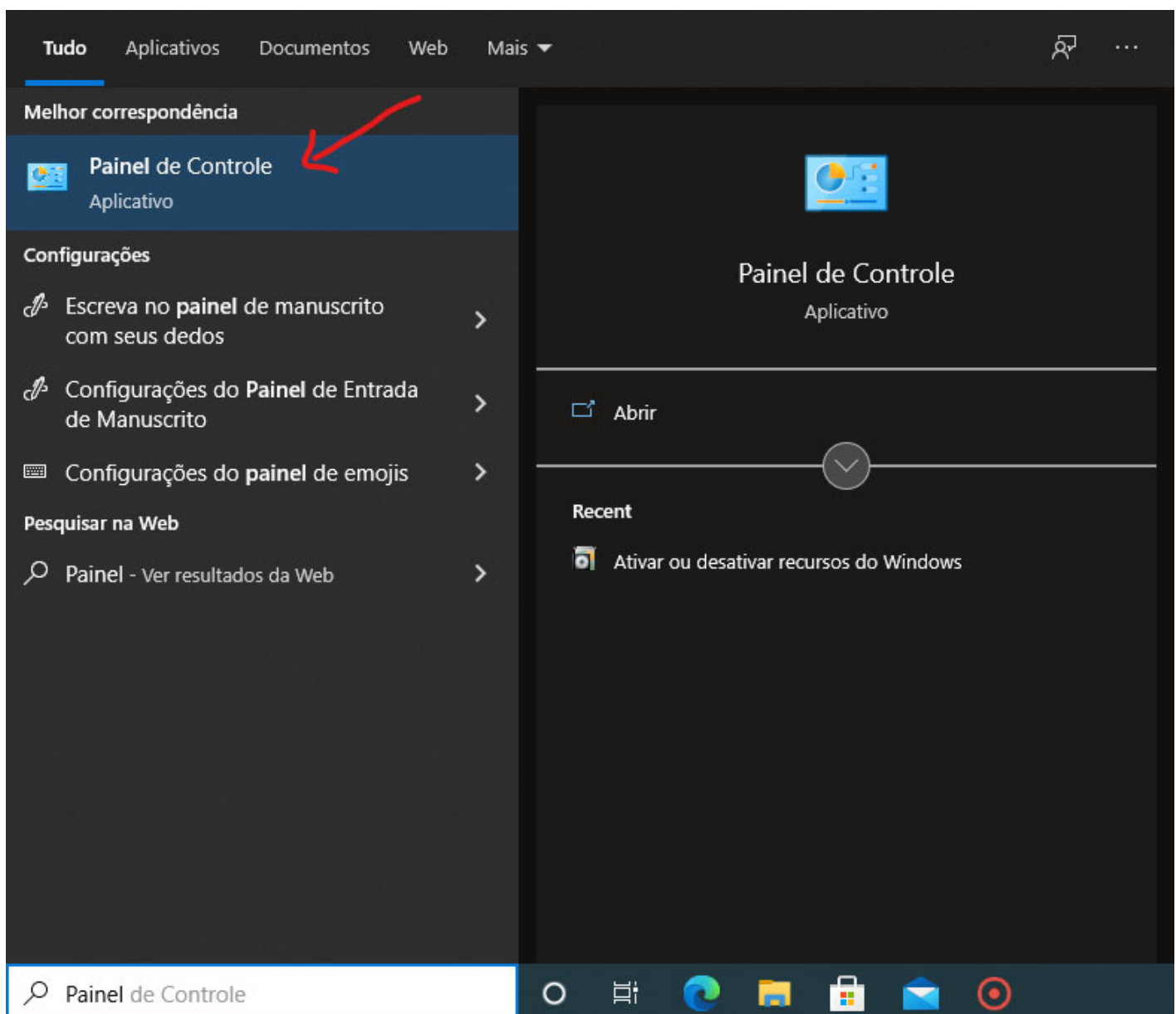
O guia abaixo tem por objetivo ensinar a instalar o Docker em ambiente Windows. Com o docker devidamente instalado e configurado vamos posteriormente portar o back-end para este ambiente. Este guia é APENAS para ambiente Windows (64 Bits).

NOTA: Caso deseje ter o docker no ambiente Linux, recomendo que reveja os guias que foram produzidos na **COM310 - Infraestrutura para Sistemas de Software - 3o Bimestre**. Lá eu detalho todo o processo de instalação no Linux Ubuntu 20.04. Para fins de funcionalidade, a aplicação que estamos desenvolvendo neste curso vai funcionar tanto com docker no ambiente Linux quanto no Windows.

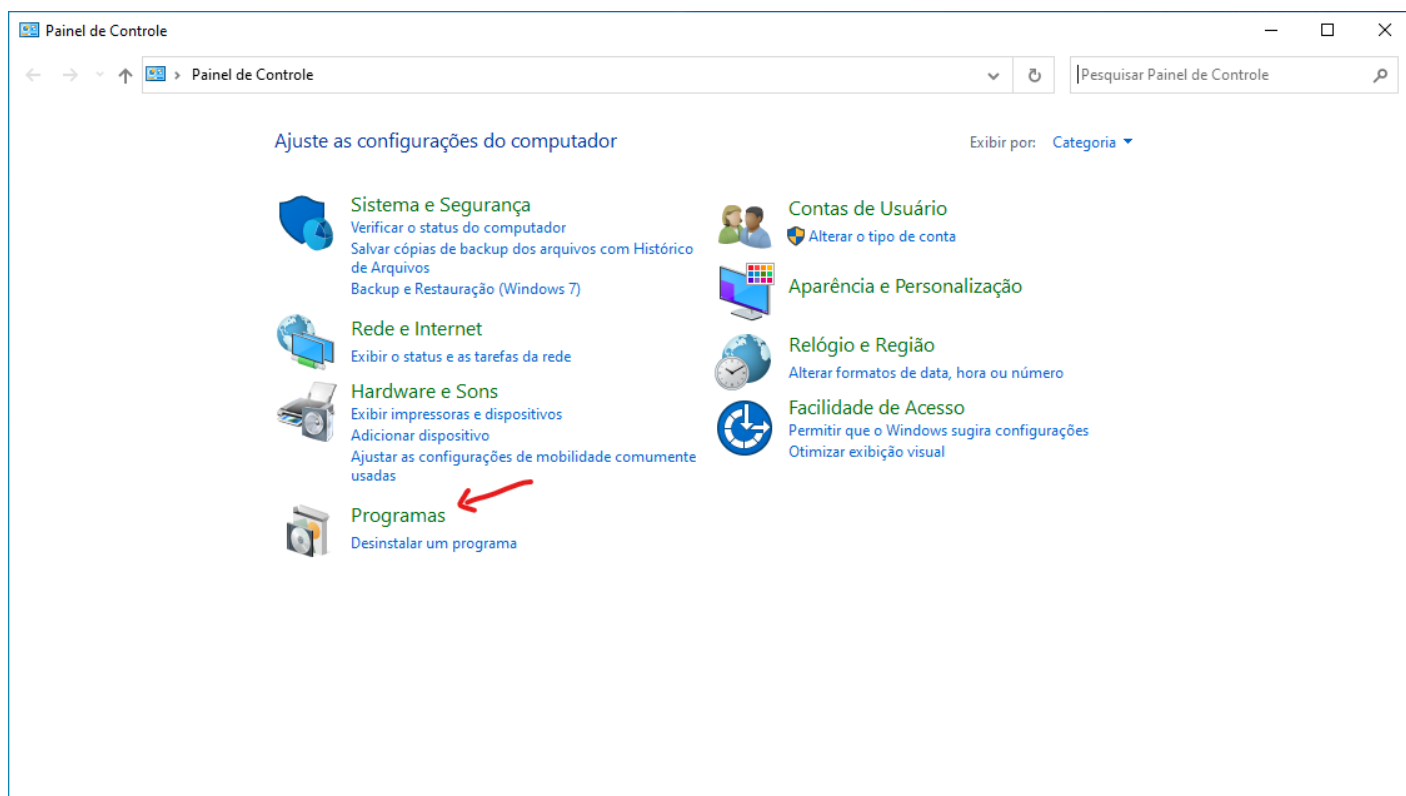
Pré-Requisitos

O primeiro passo é ter em mente que o seu hardware (desktop ou notebook) precisa ter suporte à virtualização. E que este suporte esteja devidamente habilitado no BIOS do seu dispositivo. Por isso, antes de prosseguir, **HABILITE** o suporte à virtualização no seu dispositivo, caso ele seja um hardware que tenha este suporte. Após isso, vamos prosseguir com o restante do guia.

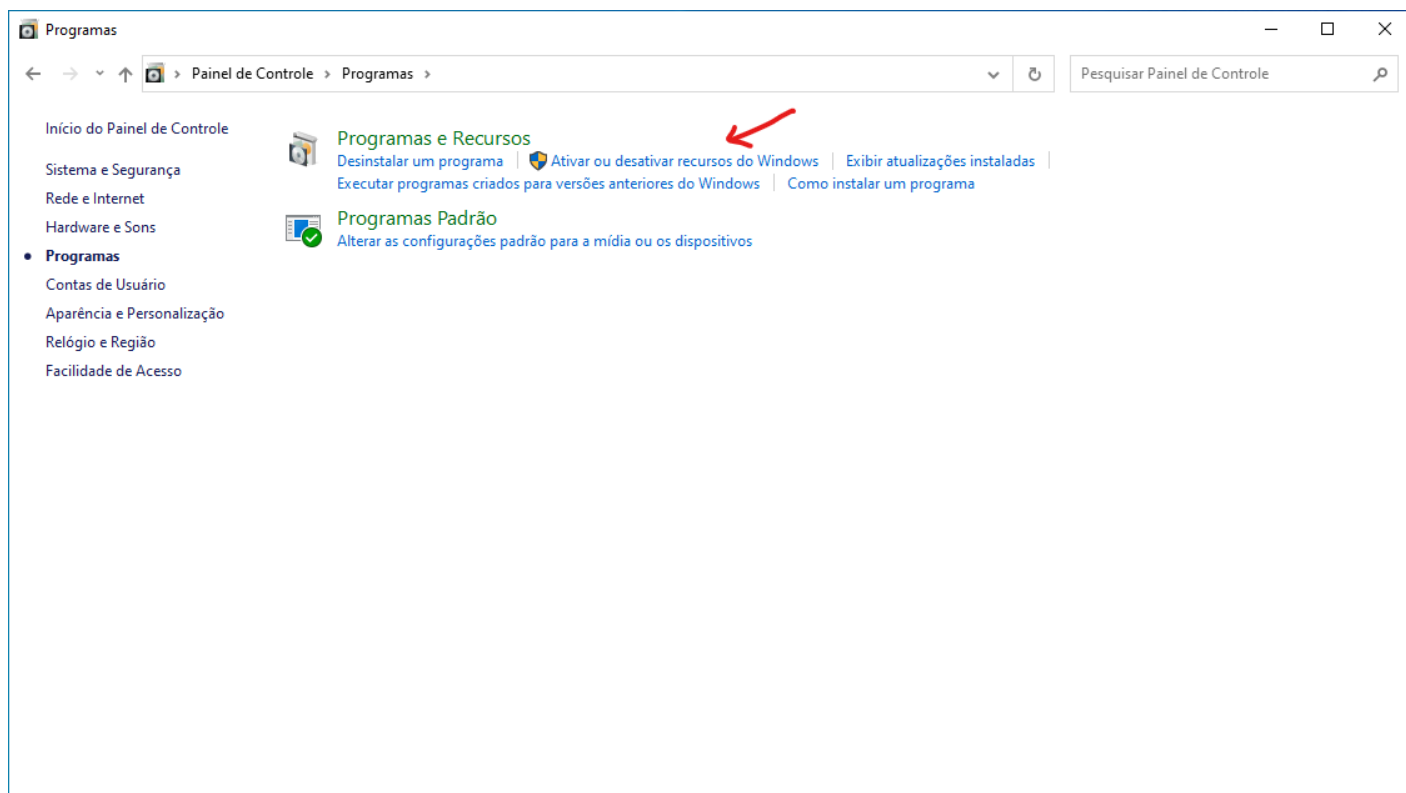
Após a verificação sugerida anteriormente, o próximo passo é ter certeza de que o Hyper-V está habilitado em seu computador. Prossiga conforme indicado na figura a seguir.



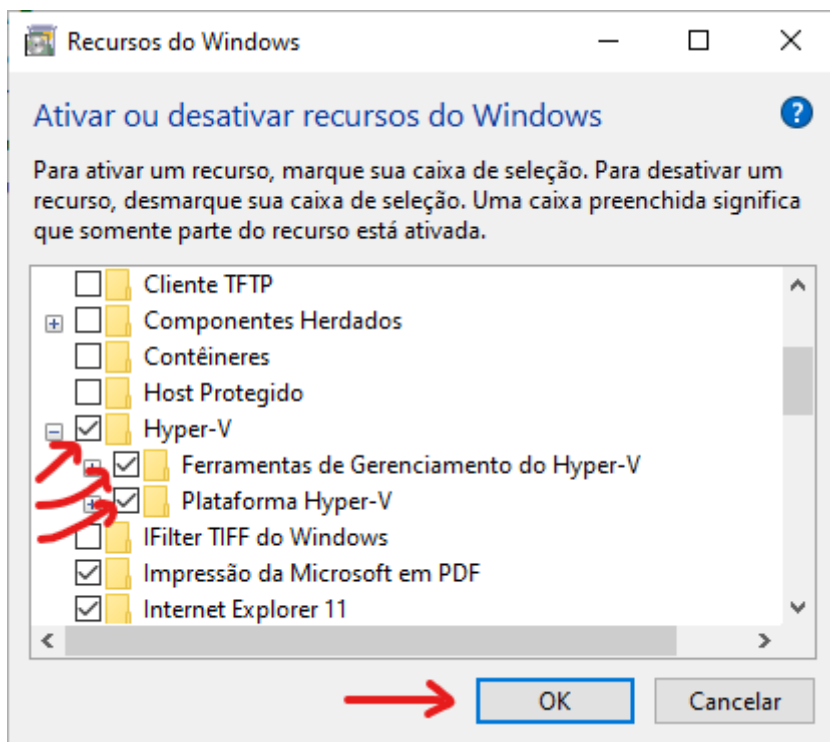
Clique em **Programas** conforme indicado abaixo.



Prossiga, clicando em **Ativar/Desativar** Recursos do Windows.

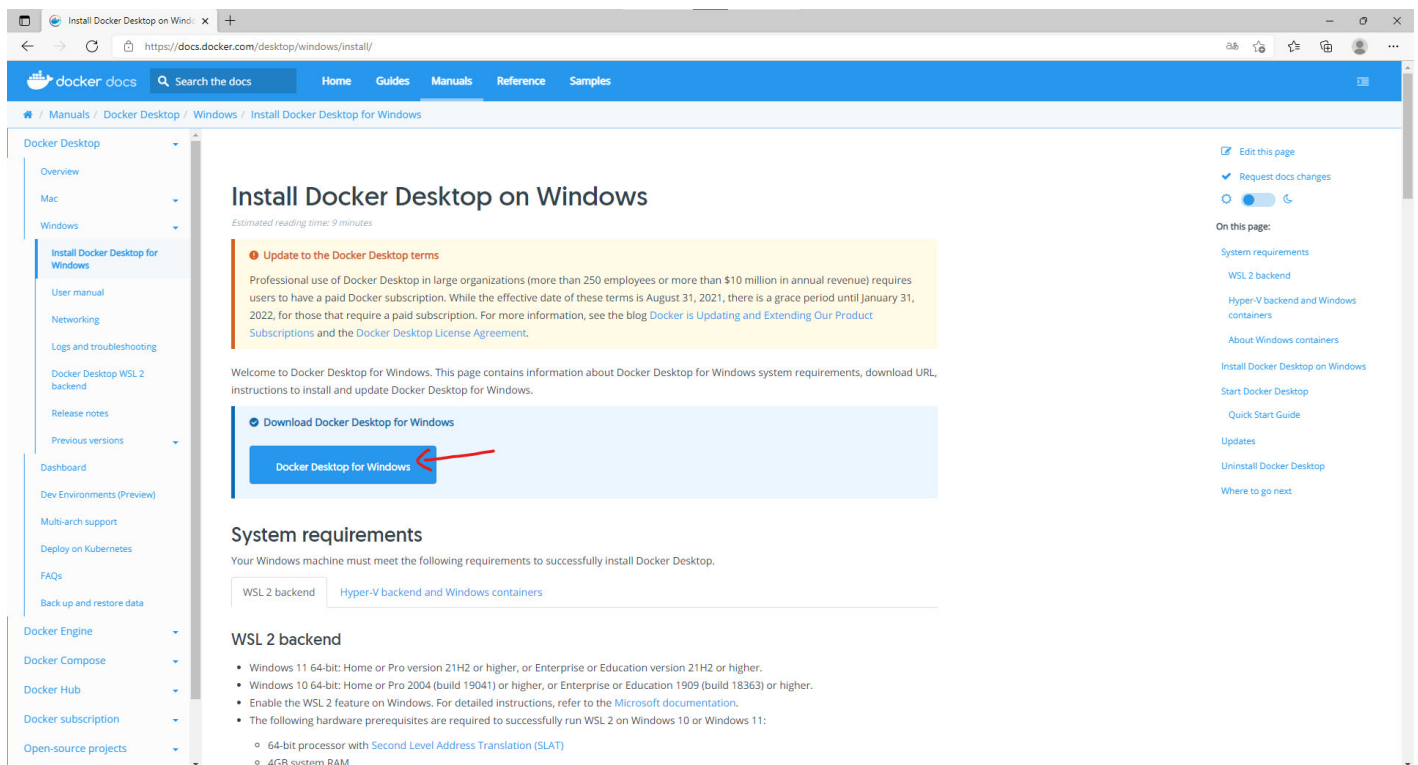


Na imagem que segue, procure pelo *Hyper-V* e **HABILITE-O**. Em seguida, clique em **OK**.



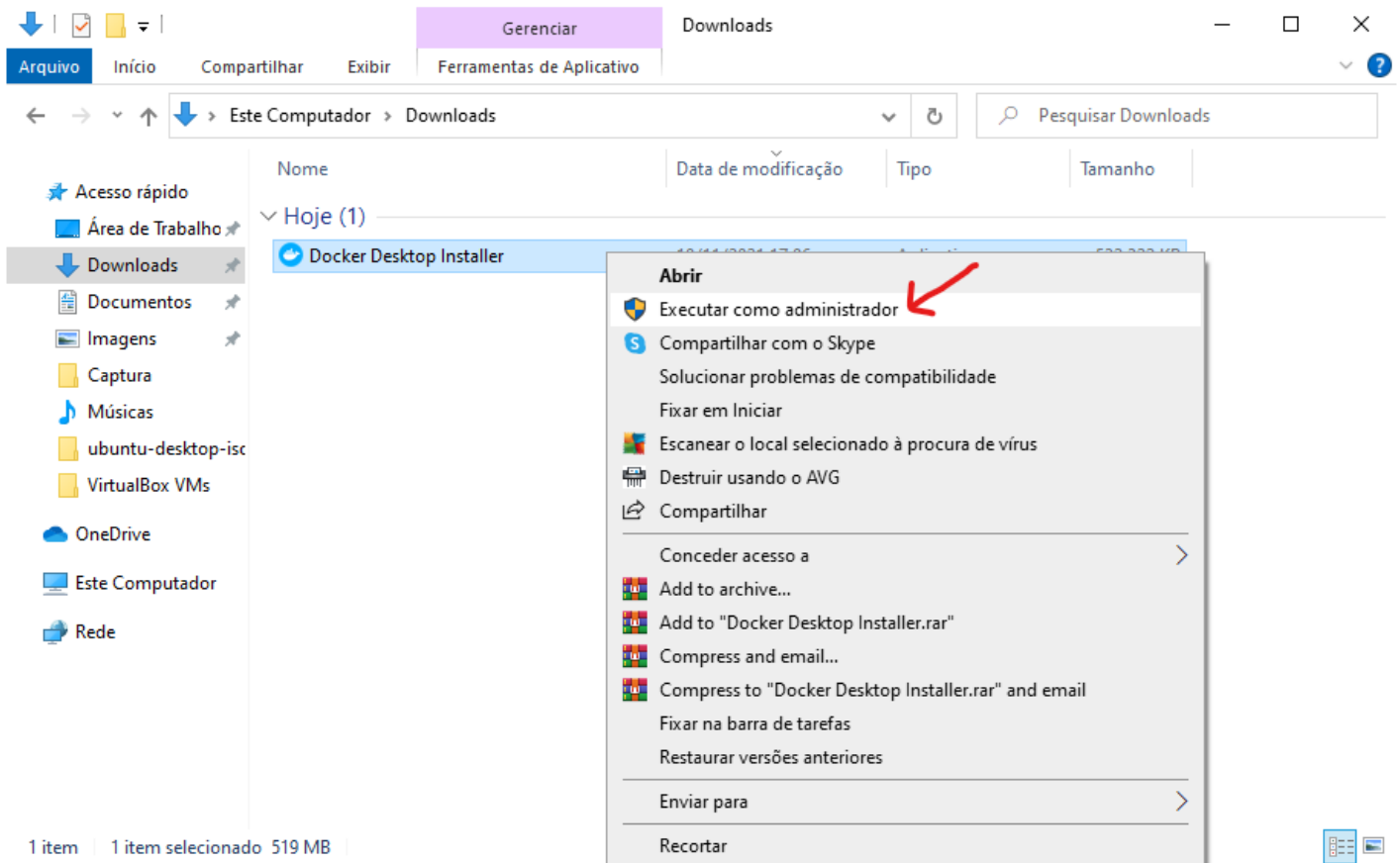
Download do Docker

Após clicar em OK, será pedido para **reiniciar o computador**. Pros siga com a *Reinicialização*. Após a reinicialização, faça uma busca no Google com a seguinte string: **install docker for windows**. Clique no primeiro link, que aparece na busca: <https://docs.docker.com/desktop/windows/install/>. Pros siga com o download do docker conforme a figura a seguir.

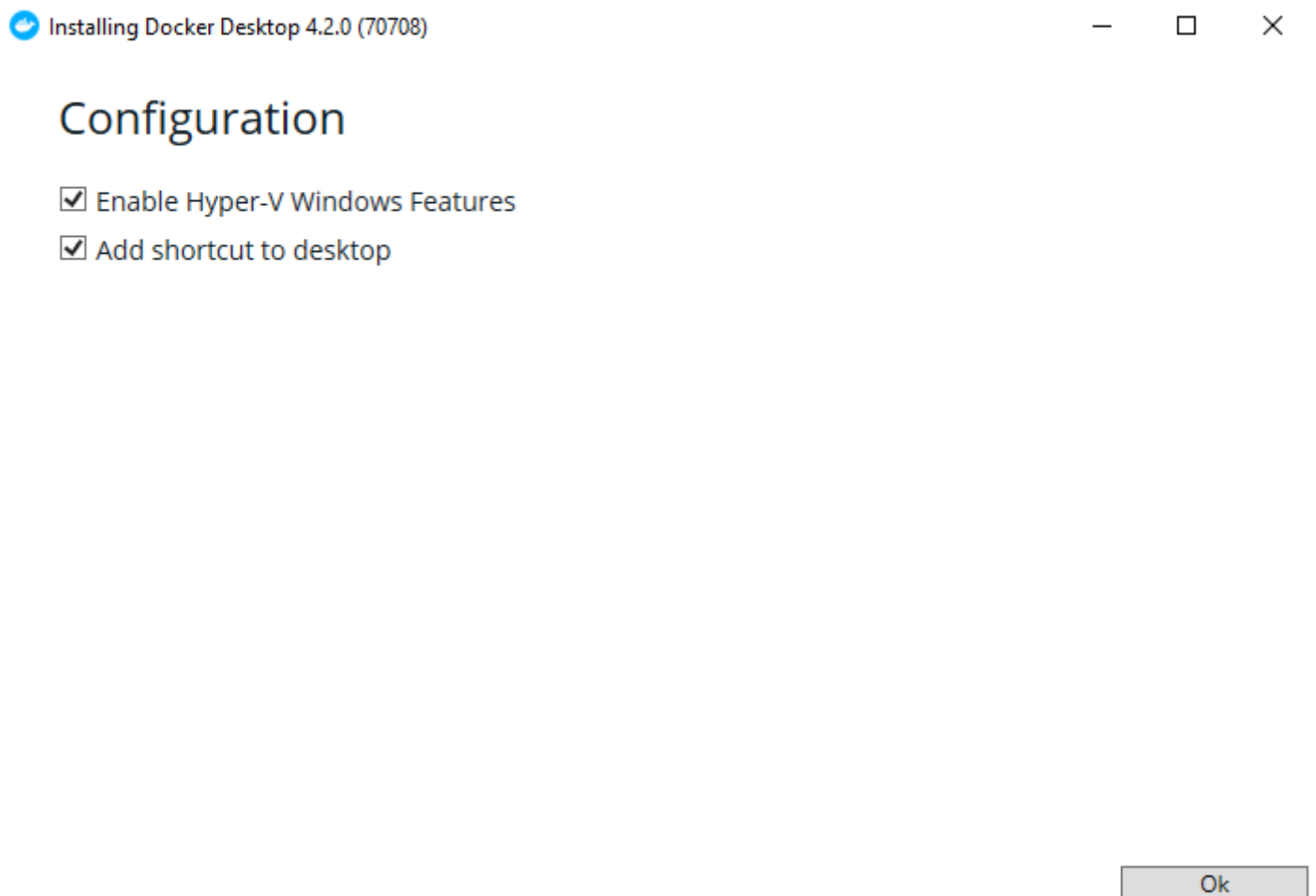


Instalação do Docker

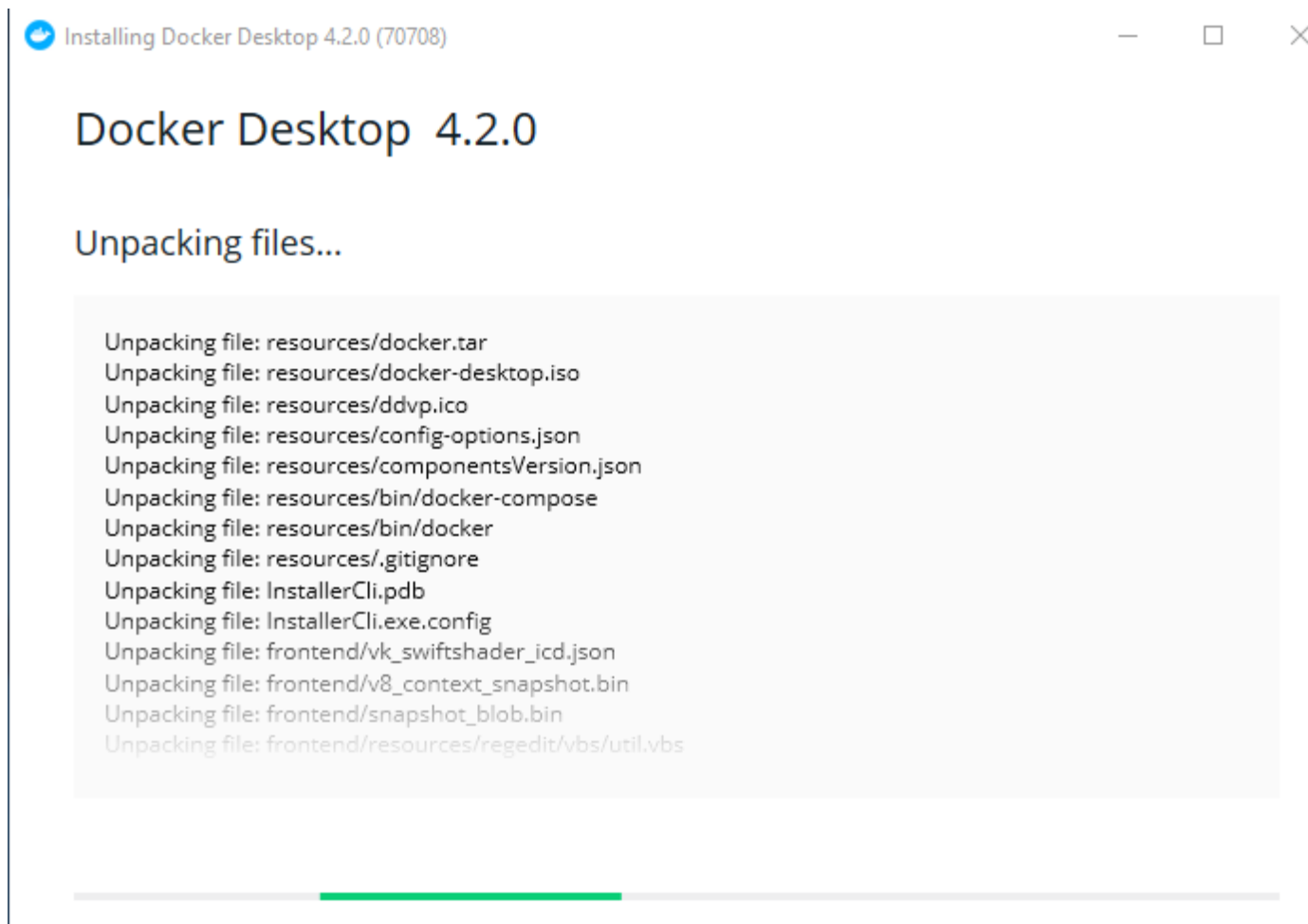
Após o download, você deve executar **como administrador** o **.exe** baixado, como mostrado abaixo. Quando perguntado na próxima tela, clique em **SIM**.



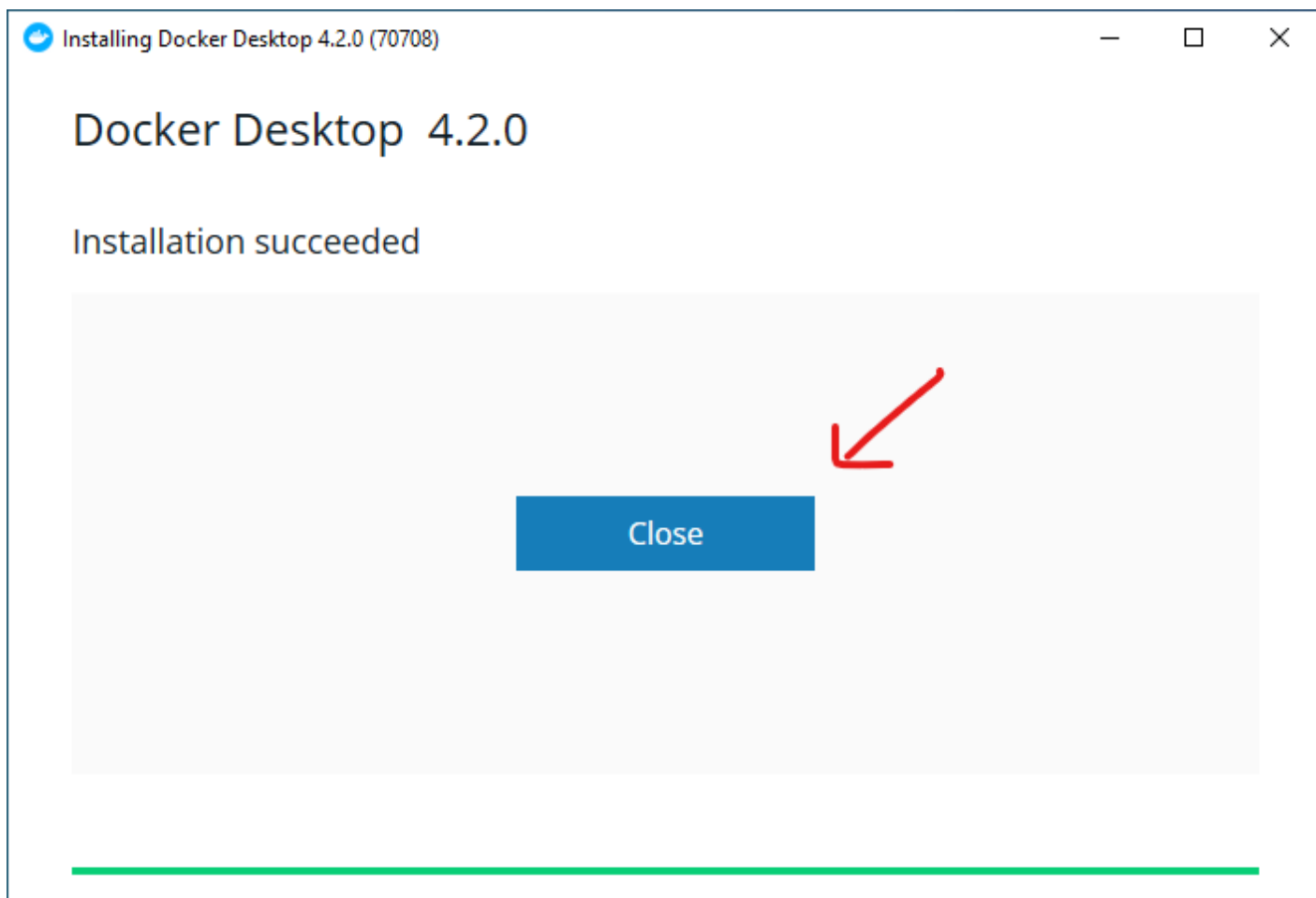
O processo de instalação deve prosseguir sem nenhum problema e o resultado será a tela apresentada na figura abaixo. O próximo passo é observar a imagem a seguir. Matenha os checkbox preenchidos, conforme já indicado na instalação e dê OK.



Após clicar em OK o processo de instalação inicia conforme a figura abaixo. Todo o processo pode demorar um pouco. Aguarde um instante até que ele finalize.



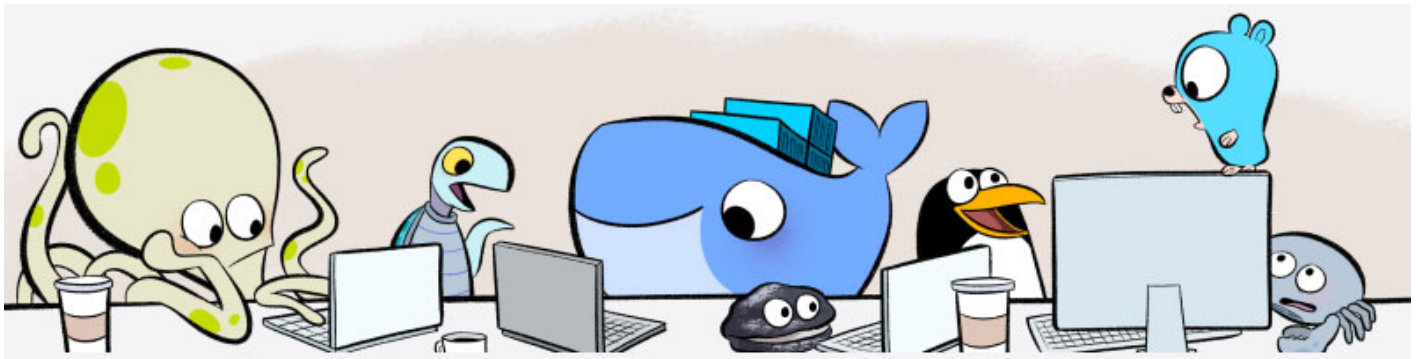
Após o processo finalizar, você vai ver uma tela como mostrada seguir. **Reinicie o computador** como é solicitado.



Após clicar em **Close** para fechar a tela de instalação, procure pelo ícone do Docker em sua Área de Trabalho e dê um **duplo click** neste ícone para iniciar o docker.



Marque o checkbox conforme indicado na seta e clique em **OK**, aceitando os termos de serviços do docker.



Our Service Agreement has Changed

- annual revenue), personal use, education, and non-commercial open source projects.
- It requires a paid subscription for professional use in larger enterprises.
 - The effective date of these terms is August 31, 2021. There is a **grace period** until January 31, 2022 for those that will require a paid subscription to use Docker Desktop.
 - The existing Docker Free subscription has been renamed **Docker Personal** and we have introduced a Docker Business subscription.
 - The Docker Pro, Team, and Business subscriptions include commercial use of Docker Desktop.

We're introducing a new product subscription, **Docker Business**, for large enterprises that require features like registry restrictions, SSO, secure software supply chain management, and more.

It's also important to note that the licensing and distribution terms for Docker and Moby **open source** projects, such as Docker Engine, are not changing.

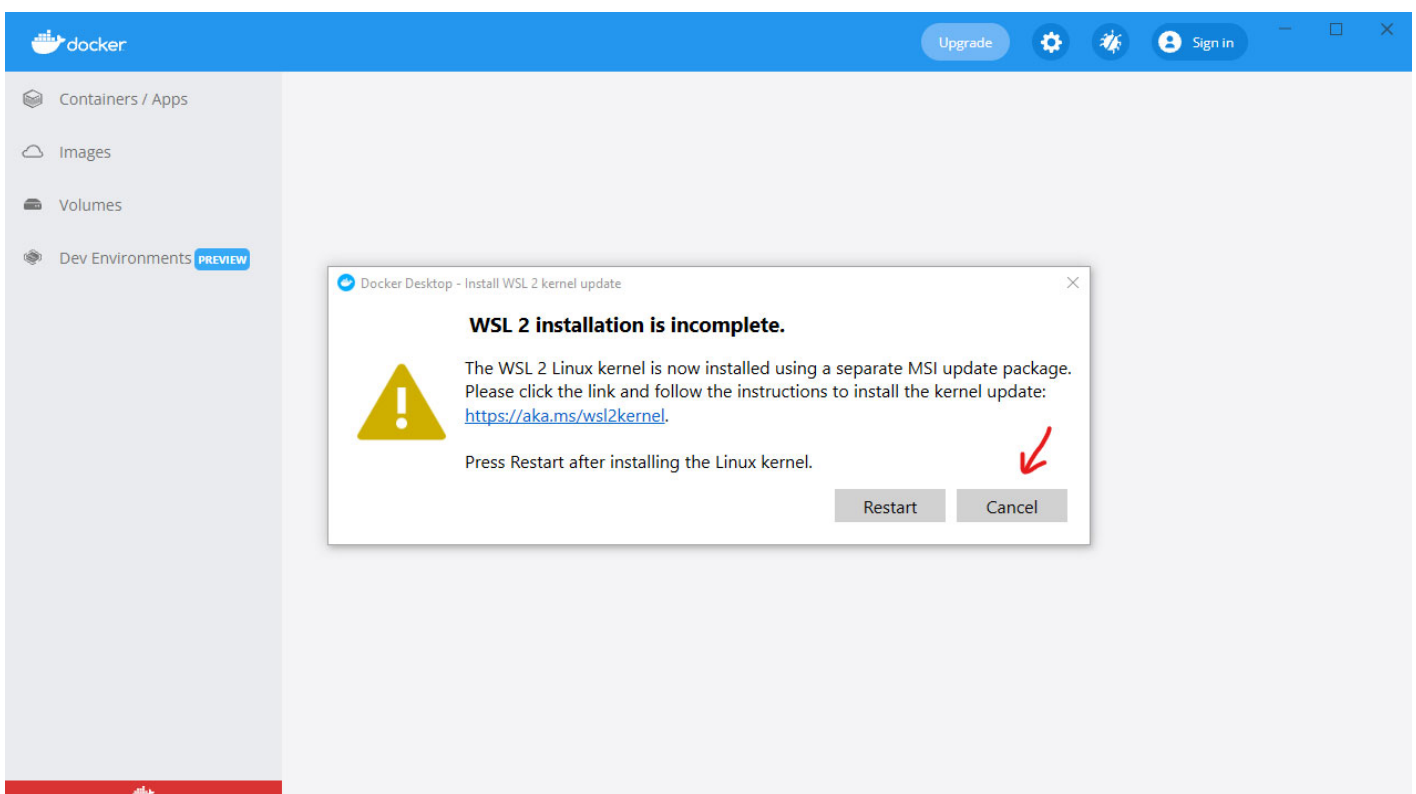
I accept the terms ☒

[View Full Terms](#) 

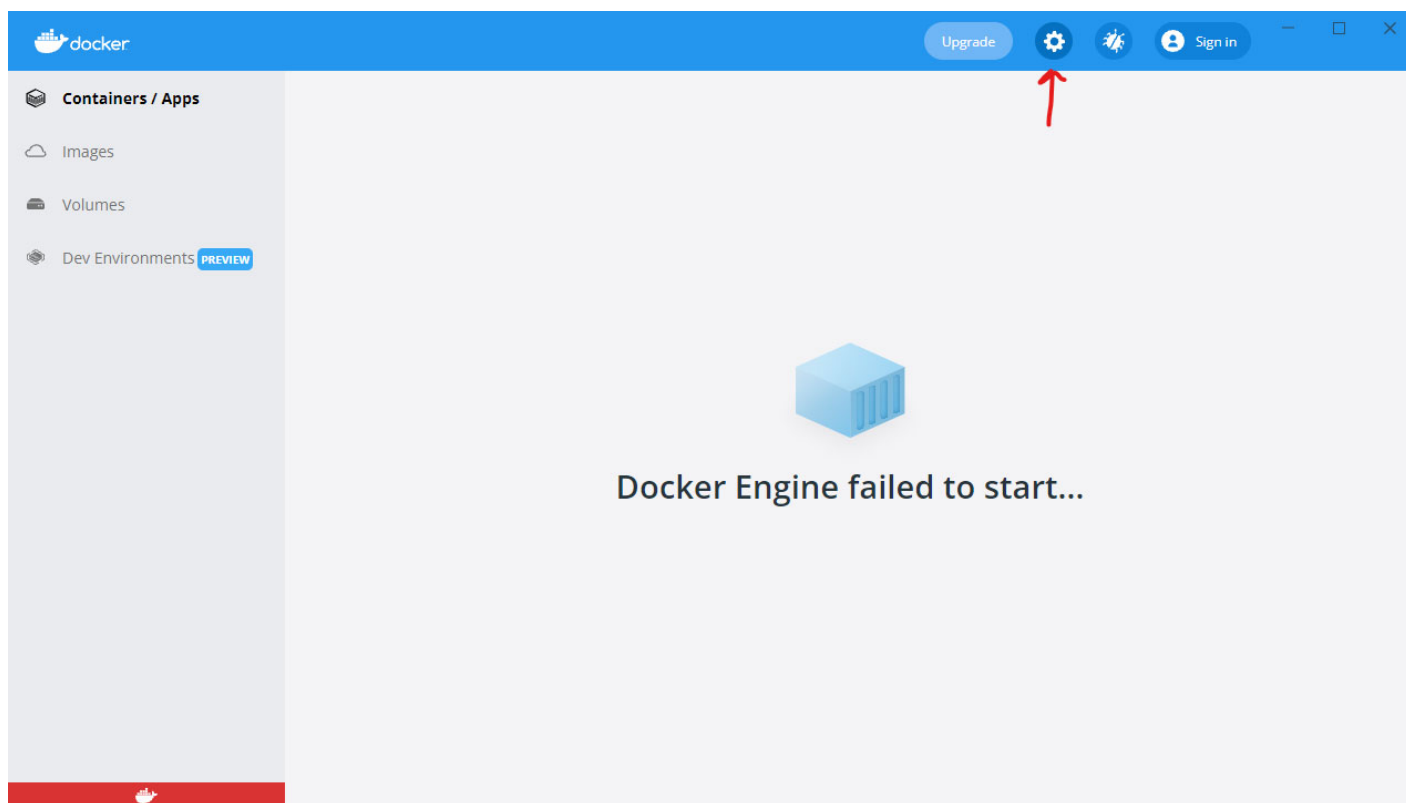
[Decline and Close Application](#)

[Accept](#)

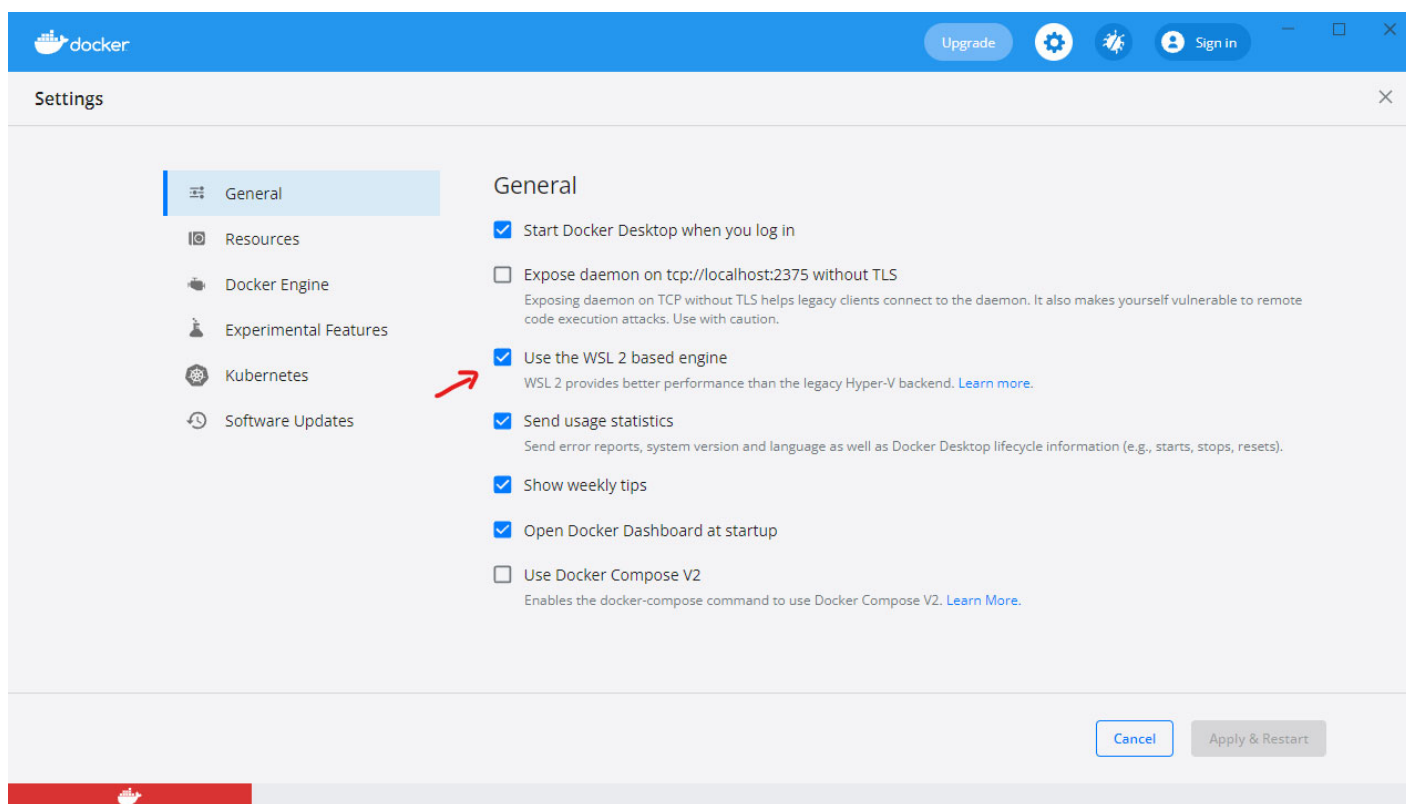
A próxima tela vai indicar que falta o WSL Linux Kernel. Você pode clicar em **Cancel**, pois neste caso ao invés de termos um ambiente com o kernel linux dentro do Windows, vamos utilizar o ambiente já ajustado e habilitado com o Hyper-V.



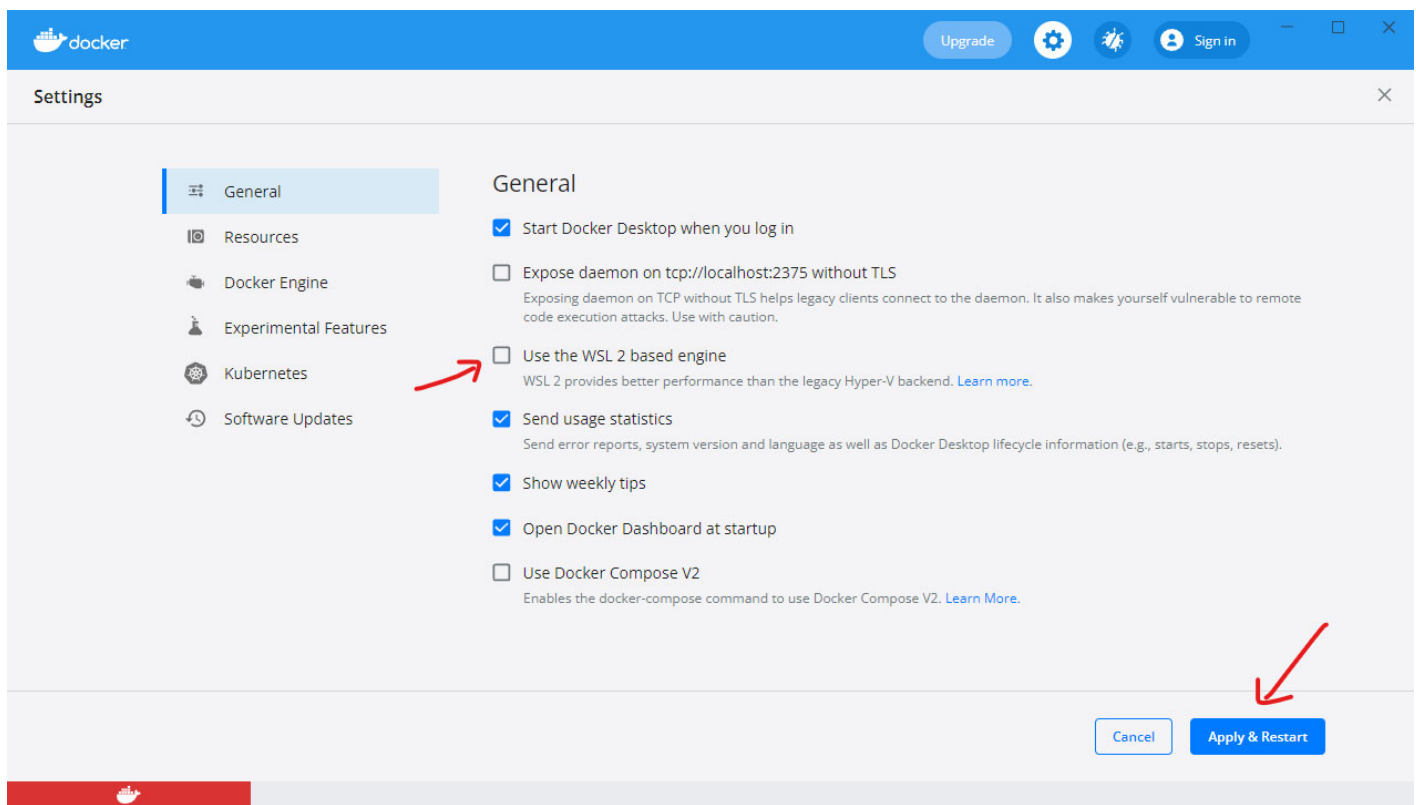
O próximo passo é acessarmos as configurações do ambiente docker **para desabilitarmos o WSL** e mantermos apenas a integração com o Hyper-v. Prossiga conforme indicado na tela abaixo.



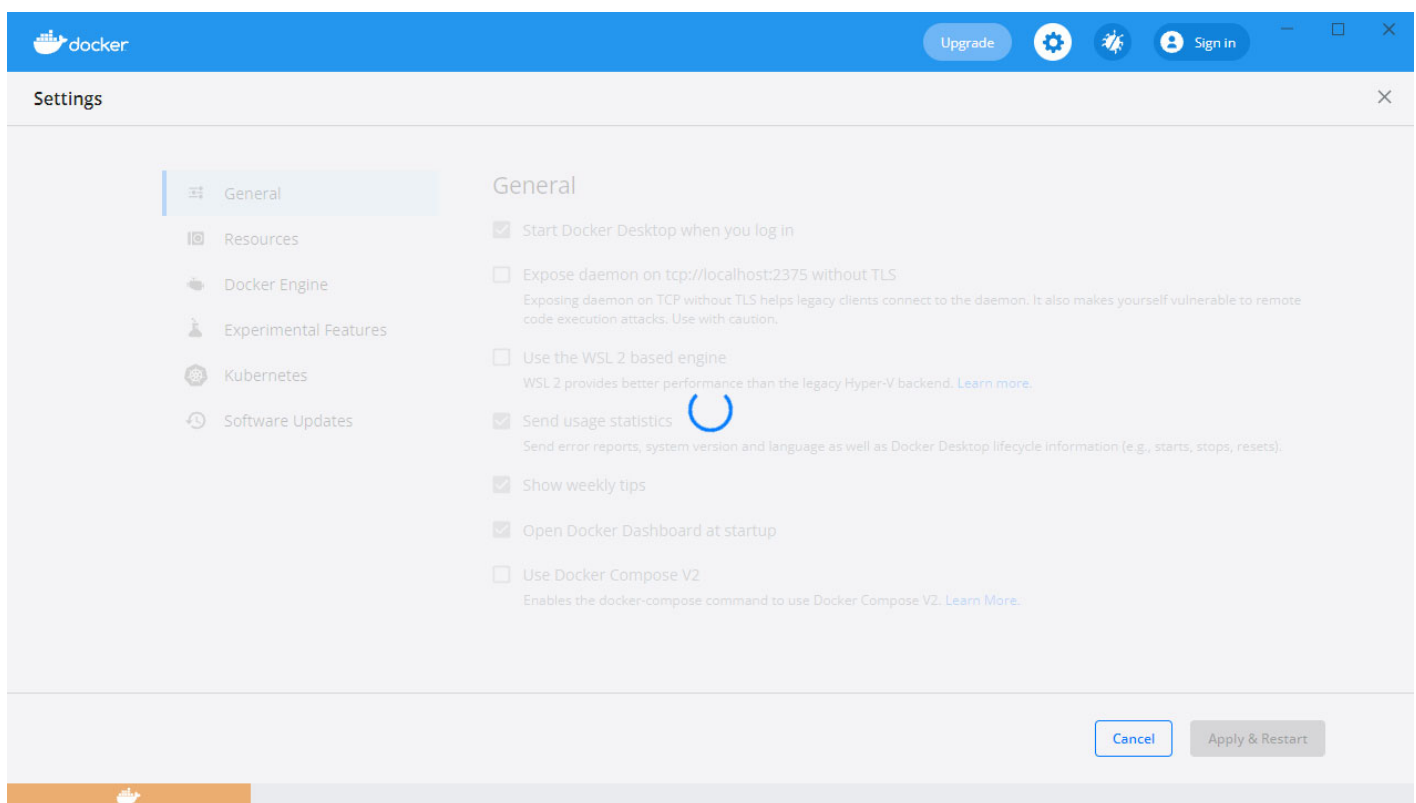
Verifique que a engine WSL 2 está habilitada. Ela **não pode ficar habilitada** como a figura a seguir.



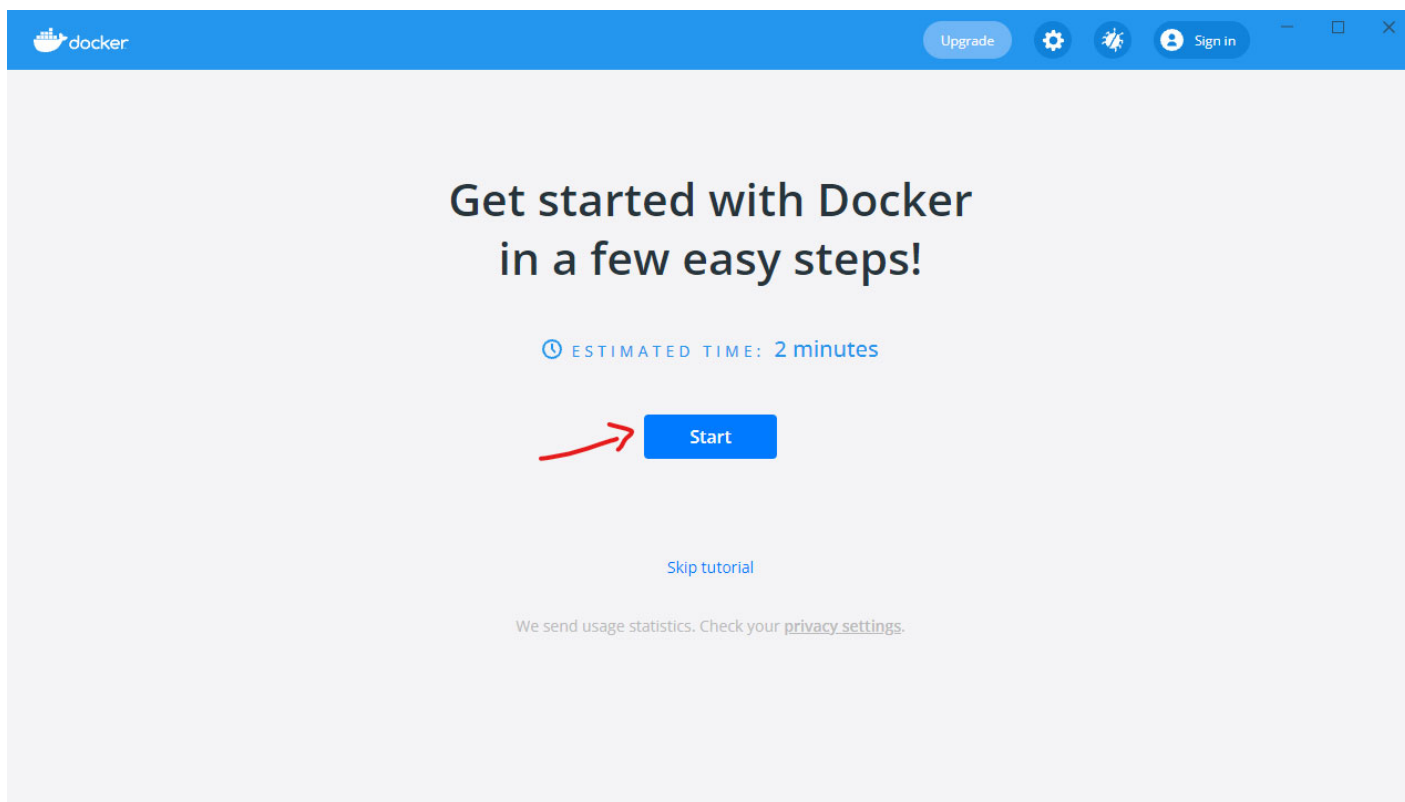
E **deve ser desabilitada** conforme a figura abaixo. Depois de desmarcar para desabilitar, clique em **Accept & Restart**.



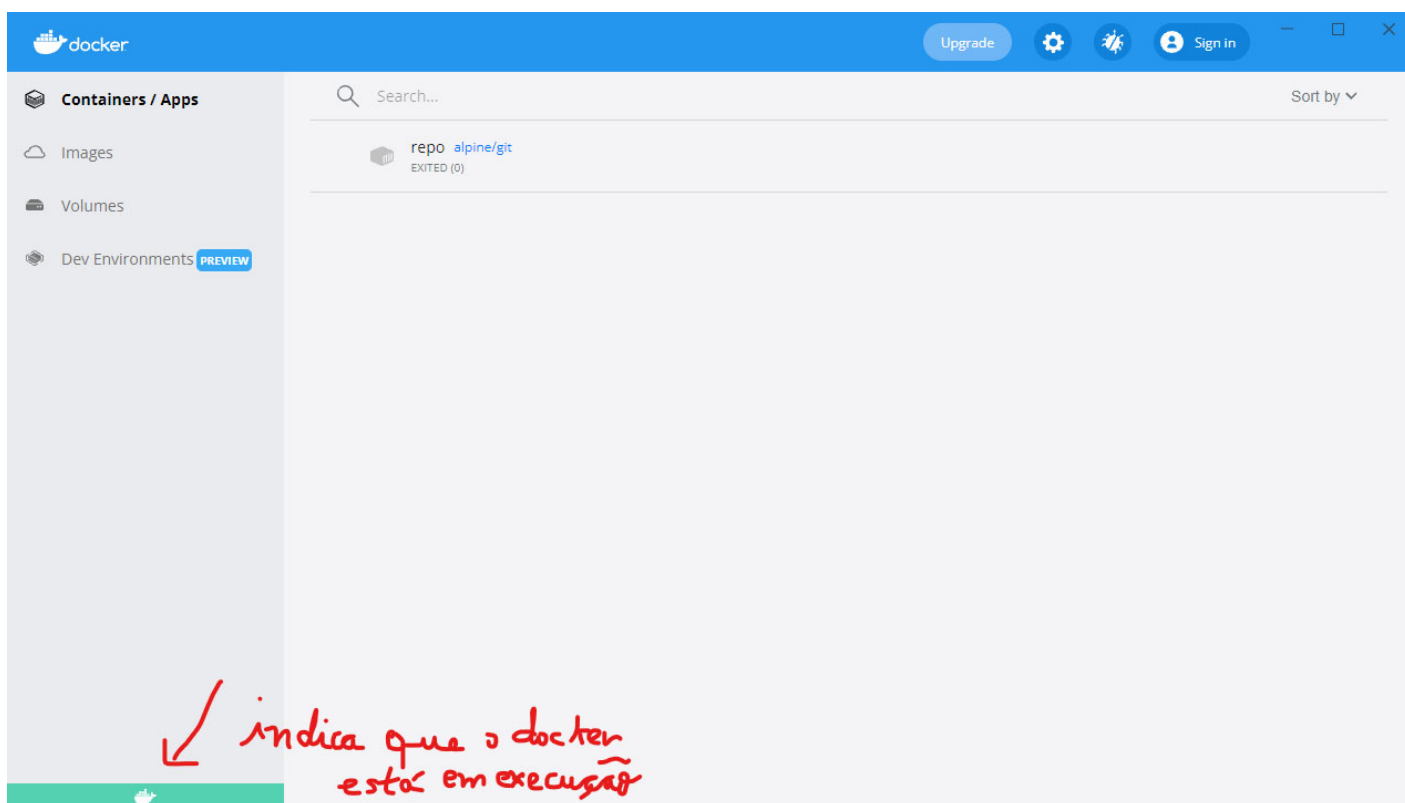
Até que o processo de desabilitação finalize, você terá uma figura como mostrada abaixo:



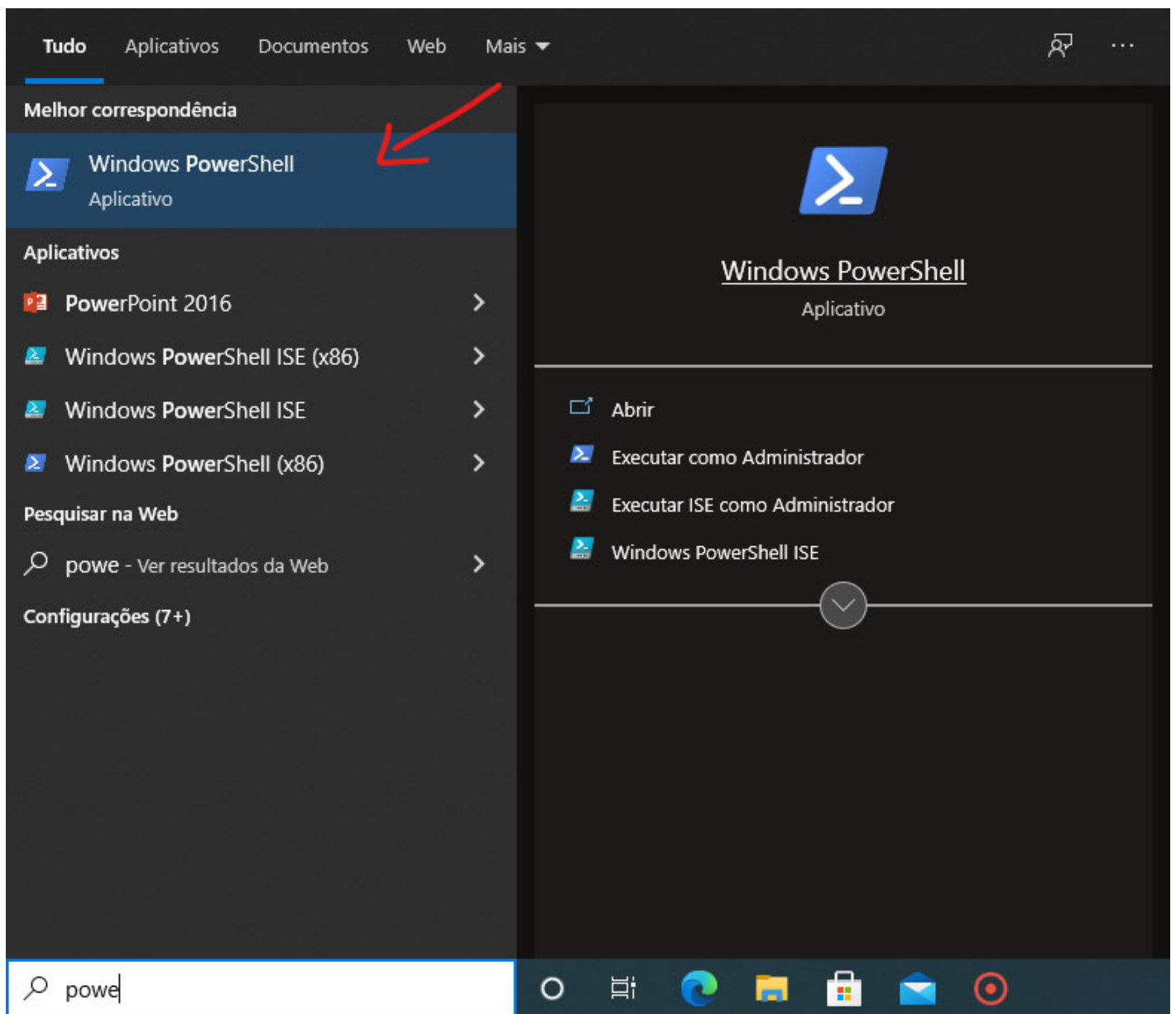
Você pode clicar em Start para fazer um pequeno tutorial em que é mostrado como fazer download de uma imagem, interagir com a mesma, etc. Mas também pode clicar em Skip Tutorial.



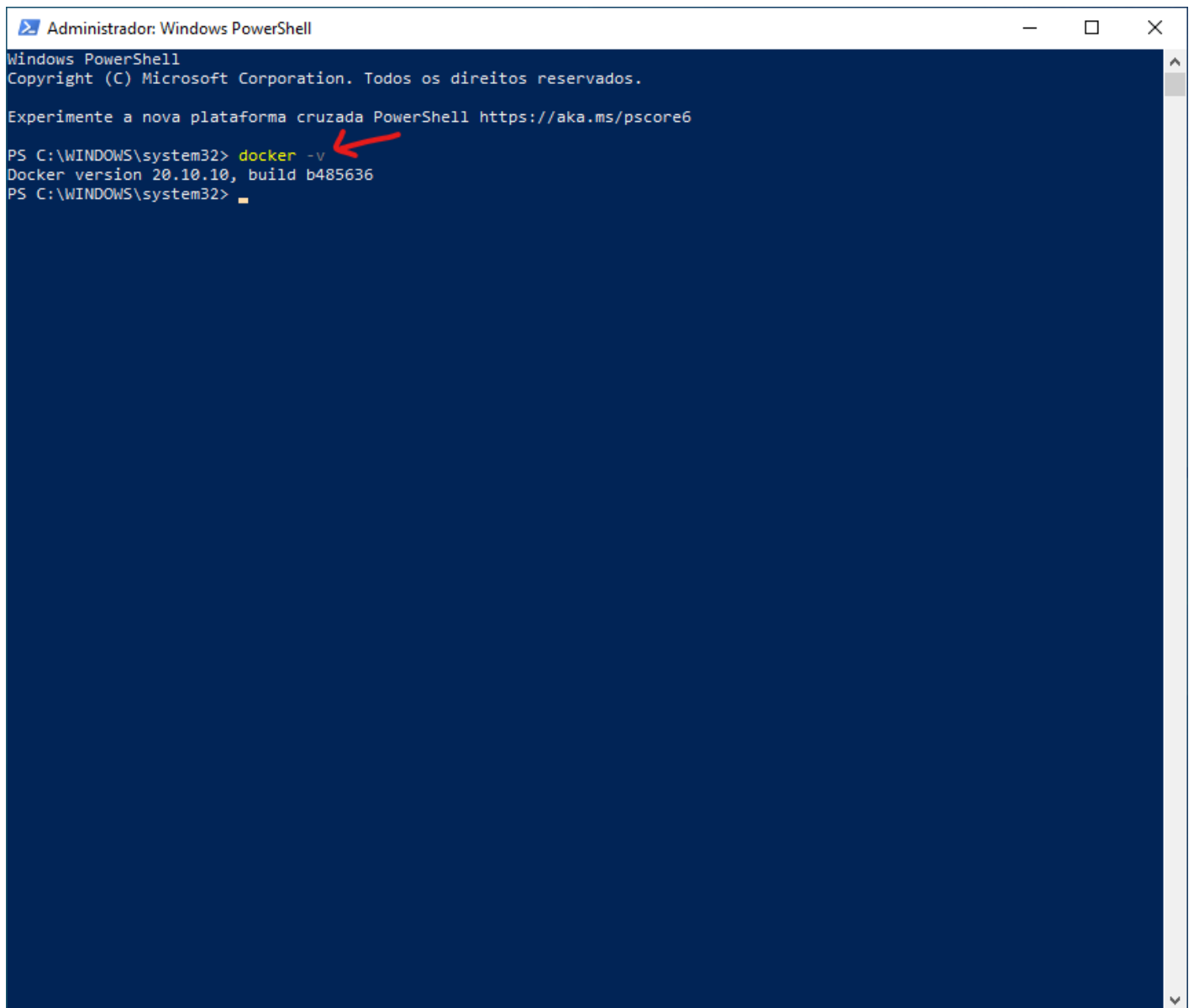
Se clicar em Skip Tutorial, terá a tela descrita abaixo. Desta forma podemos verificar que temos o docker em execução conforme indicado no ícone em verde logo na parte inferior esquerda como destacado na seta.



Vamos também fazer um teste pelo PowerShell do Windows e verificar que o docker responde a alguns simples comandos. Abra o PowerShell conforme indicado a seguir.



Execute primeiro o comando para conhecer a versão instalada do docker.



```
Administrador: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> docker -v
Docker version 20.10.10, build b485636
PS C:\WINDOWS\system32>
```

A screenshot of a Windows PowerShell terminal window titled "Administrador: Windows PowerShell". The window has a dark blue background and a light gray title bar with standard Windows window controls. The text displayed in the terminal is as follows: "Windows PowerShell", "Copyright (C) Microsoft Corporation. Todos os direitos reservados.", "Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6", "PS C:\WINDOWS\system32> docker -v", "Docker version 20.10.10, build b485636", and "PS C:\WINDOWS\system32>". A red arrow points to the word "docker" in the command line.

Em seguida, digite docker e terá diversas opções no menu. Isso indica que temos todo o ambiente preparado para que possamos transferir o ambiente do back-end para uma imagem docker. Com isso na Semana 7 poderemos implantar esta imagem na nuvem da Google e também da Azure.

```
Administrador: Windows PowerShell
PS C:\WINDOWS\system32> docker
Usage: docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default
                        "C:\\Users\\julio\\.docker")
  -c, --context string  Name of the context to use to connect to the
                        daemon (overrides DOCKER_HOST env var and
                        default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level
                        ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "C:\\Users\\julio\\.docker\\ca.pem")
  --tlscert string      Path to TLS certificate file (default
                        "C:\\Users\\julio\\.docker\\cert.pem")
  --tlskey string       Path to TLS key file (default
                        "C:\\Users\\julio\\.docker\\key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  builder      Manage builds
  buildx*      Build with BuildKit (Docker Inc., v0.6.3)
  compose*     Docker Compose (Docker Inc., v2.1.1)
  config       Manage Docker configs
  container    Manage containers
  context       Manage contexts
  image        Manage images
  manifest     Manage Docker image manifests and manifest lists
  network      Manage networks
  node         Manage Swarm nodes
  plugin       Manage plugins
  scan*        Docker Scan (Docker Inc., 0.9.0)
  secret       Manage Docker secrets
  service      Manage services
  stack        Manage Docker stacks
  swarm        Manage Swarm
  system       Manage Docker
  trust        Manage trust on Docker images
  volume       Manage volumes

Commands:
  attach      Attach local standard input, output, and error streams to a running container
```

Validação da Instalação do Docker

Depois de instalado, o Docker Desktop é executado automaticamente como um serviço. Ele é mostrado no System Tray quando você faz login no Windows após a reinicialização. Mas como realmente sabemos que está funcionando? Para verificar se o Docker Desktop está funcionando já abrimos anteriormente um console de linha de comando e executamos o comando `docker`. Como certificamos, a instalação ocorreu de forma adequada, pois vimos uma referência de comando do Docker.

Por fim, para certificarmos que de fato podemos criar imagens e lidar com elas, vamos executar uma imagem de contêiner de exemplo chamada `hello-world`, executando o comando `docker run hello-world`. Abra um console (pode ser o PowerShell ou o CMD) e execute o comando a seguir:

```
docker run hello-world
```

E teremos a saída abaixo:

```
PS C:\> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:c3b4ada4687bbaa170745b3e4dd8ac3f194ca95b2d0518b417fb47e5879d9b5f
Status: Downloaded newer image for hello-world:latest

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

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

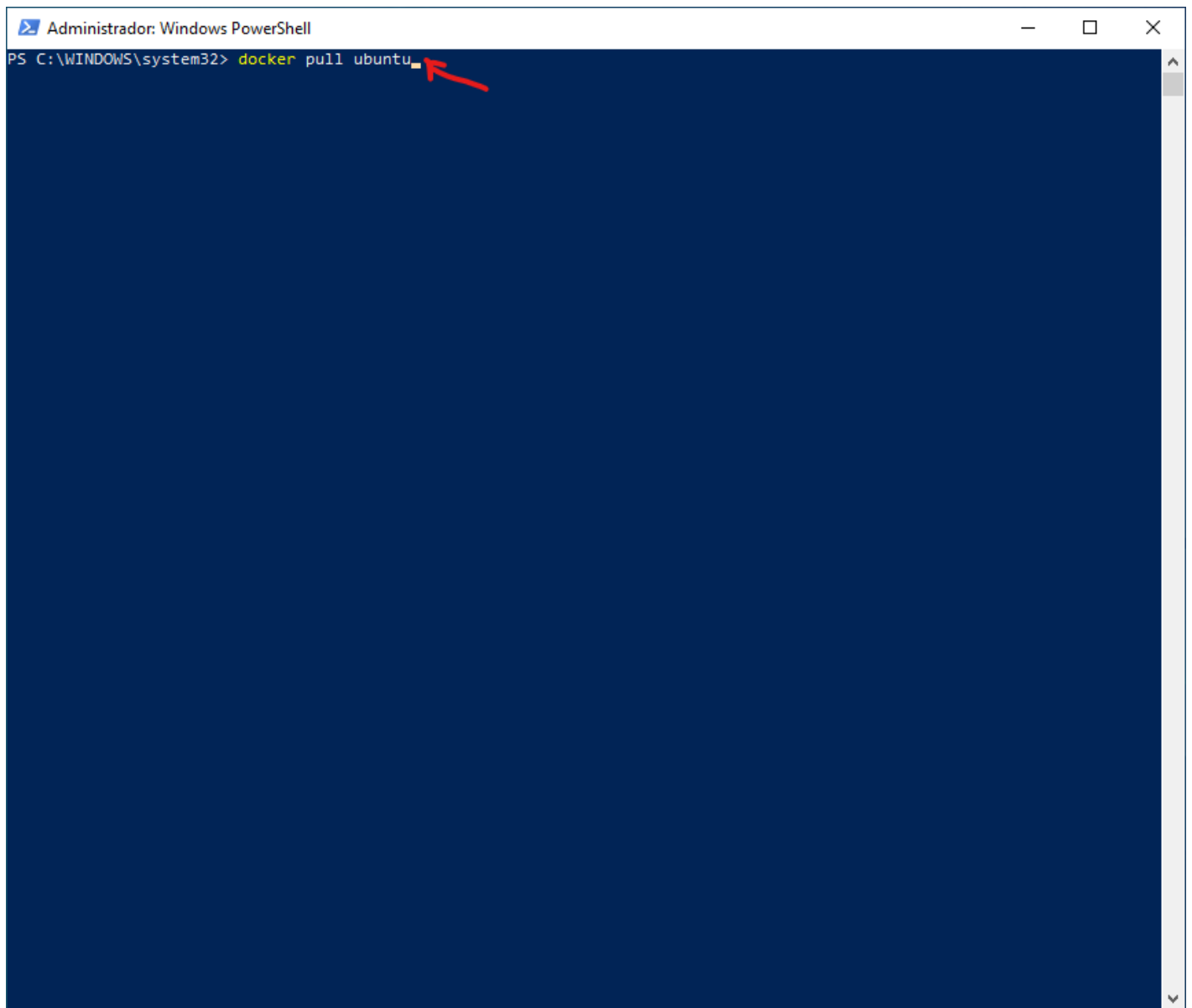
Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Preparação de Imagem Linux com o Laravel

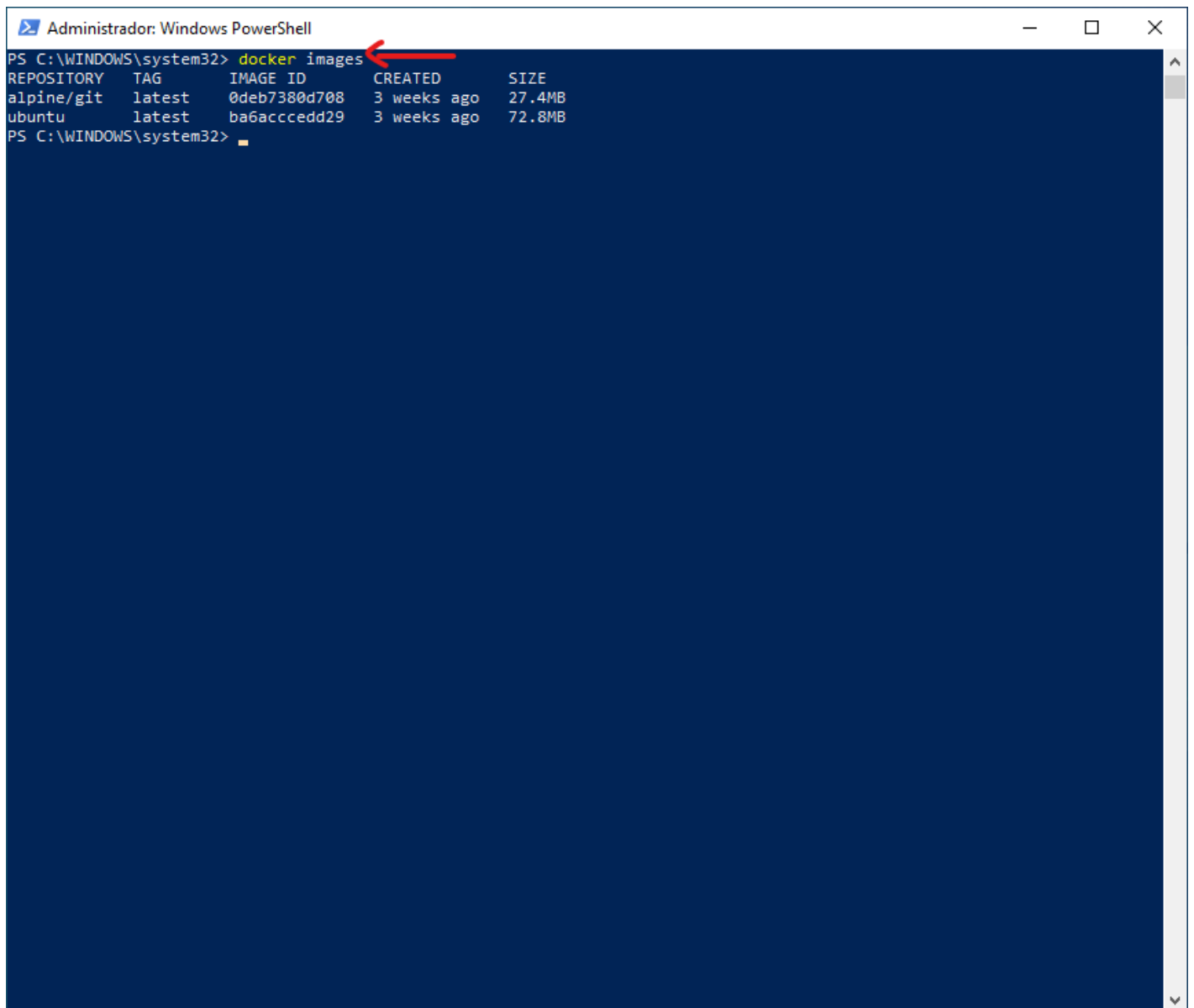
O objetivo desta etapa do tutorial é ter o todo o código do back-end com o ambiente Laravel devidamente implantado em uma imagem Linux Ubuntu para que na Semana 7 possamos fazer o deploy desta imagem na Google e também na Azure. O objetivo é que tenhamos o back-end remoto em uma nuvem computacional e o front-end funcional localmente (no seu desktop ou notebook) como foi desenvolvido na Semana 3. Alguns ajustes no front-end serão feitos para que isso possa ser possível e vamos mostrar isso mais adiante.

Considerando que você seguiu todos os passos anteriormente e tem o docker funcionando, vamos baixar uma imagem Ubuntu do repositório do docker hub. Abrir um PowerShell e prossiga com descrito na tela a seguir.



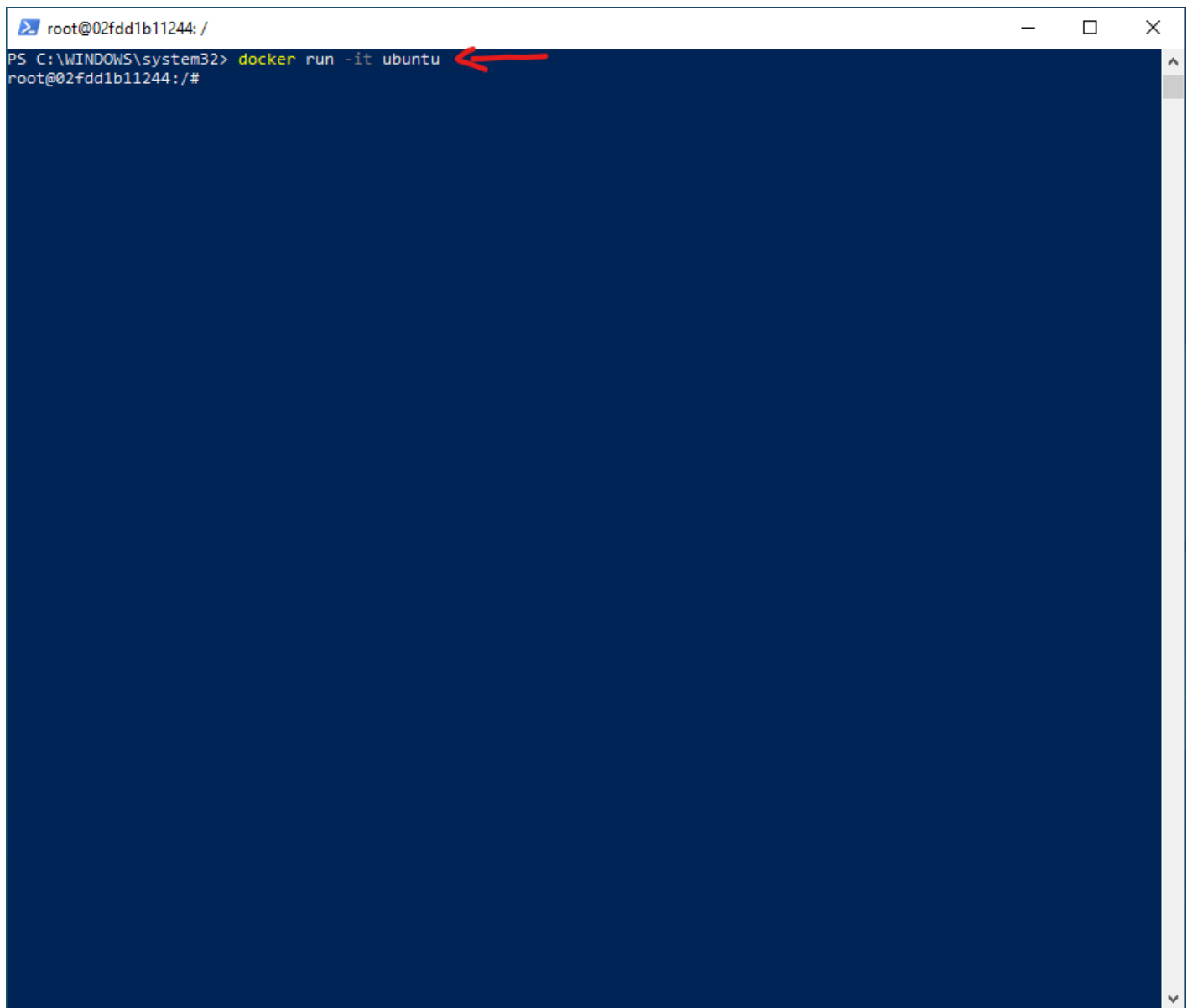
```
Administrador: Windows PowerShell
PS C:\WINDOWS\system32> docker pull ubuntu
```

Com o comando `docker images` podemos verificar a imagem que acabmos de baixar. Observe que o tamanho da imagem Linux é bem pequeno. Mas esse tamanho vai crescer bastante pois teremos que instalar e configurar vários pacotes.

A screenshot of a Windows PowerShell terminal window titled "Administrador: Windows PowerShell". The terminal shows the command "docker images" being executed, which lists two Docker images: "alpine/git" and "ubuntu". A red arrow points to the "docker images" command. The terminal output is as follows:

```
PS C:\WINDOWS\system32> docker images
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
alpine/git    latest    0deb7380d708  3 weeks ago  27.4MB
ubuntu        latest    ba6acccedd29  3 weeks ago  72.8MB
PS C:\WINDOWS\system32>
```

Agora o que precisamos fazer é iniciar o container com a imagem baixada. Depois da execução do comando apresentado na tela abaixo você vai ganhar posse do terminal e nele fará tudo o que precisa para ter o back-end com laravel todo funcional.

A terminal window with a dark blue background. The title bar at the top shows 'root@02fdd1b11244: /' and standard window controls. The terminal text shows a Windows command prompt 'PS C:\WINDOWS\system32>' followed by the command 'docker run -it ubuntu' in green text. A red arrow points to the end of this command. Below it, the prompt 'root@02fdd1b11244:/#' is visible, indicating the user is now inside the Ubuntu container.

```
root@02fdd1b11244: /
PS C:\WINDOWS\system32> docker run -it ubuntu
root@02fdd1b11244:/#
```

É importante que você não saia do terminal até que tenhamos instalado os pacotes. Assim, **NÃO DIGITE** *exit* no terminal, pois você perderá todas as alterações feitas na imagem, até que façamos um commit ao final. Prossiga com a atualização com o comando `apt-get update` como indicado na sequência.

```
PS C:\WINDOWS\system32> docker run -it ubuntu
root@02fdd1b11244:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [1229 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [11.3 MB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [682 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/universe amd64 Packages [807 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/multiverse amd64 Packages [30.1 kB]
43% [6 Packages 3634 kB/11.3 MB 32%] 268 kB/s 47s
```

O próximo passo é instalar algumas ferramentas que nos darão suporte para saber o IP da imagem, etc.

root@02fdd1b11244: /

root@02fdd1b11244:/# apt-get install nano net-tools systemctl

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following additional packages will be installed:

file libexpat1 libmagic-mgc libmagic1 libmpdec2 libpython3-stdlib libpython3.8-minimal libpython3.8-stdlib
libreadline8 libsqlite3-0 libssl1.1 mime-support python3 python3-minimal python3.8 python3.8-minimal readline-common
xz-utils

Suggested packages:

hunspell python3-doc python3-tk python3-venv python3.8-doc binutils binfmt-support readline-doc tini
| dumb-init

The following NEW packages will be installed:

file libexpat1 libmagic-mgc libmagic1 libmpdec2 libpython3-stdlib libpython3.8-minimal libpython3.8-stdlib
libreadline8 libsqlite3-0 libssl1.1 mime-support nano net-tools python3 python3-minimal python3.8 python3.8-minimal
readline-common systemctl xz-utils

0 upgraded, 21 newly installed, 0 to remove and 0 not upgraded.

Need to get 7936 kB of archives.

After this operation, 34.8 MB of additional disk space will be used.

Do you want to continue? [Y/n] Y

```
root@02fdd1b11244: /
Selecting previously unselected package systemctl.
Preparing to unpack .../7-systemctl_1.4.3424-2_all.deb ...
Unpacking systemctl (1.4.3424-2) ...
Setting up net-tools (1.60+git20180626.aebd88e-1ubuntu1) ...
Setting up mime-support (3.64ubuntu1) ...
Setting up libmagic-mgc (1:5.38-4) ...
Setting up libsqlite3-0:amd64 (3.31.1-4ubuntu0.2) ...
Setting up libmagic1:amd64 (1:5.38-4) ...
Setting up file (1:5.38-4) ...
Setting up xz-utils (5.2.4-1ubuntu1) ...
update-alternatives: using /usr/bin/xz to provide /usr/bin/lzma (lzma) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/lzma.1.gz because associated file /usr/share/man/man1/xz.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/unlzma.1.gz because associated file /usr/share/man/man1/unxz.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzcat.1.gz because associated file /usr/share/man/man1/xzcat.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzmore.1.gz because associated file /usr/share/man/man1/xzmore.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzless.1.gz because associated file /usr/share/man/man1/xzless.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzdiff.1.gz because associated file /usr/share/man/man1/xzdiff.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzcmp.1.gz because associated file /usr/share/man/man1/xzcmp.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzgrep.1.gz because associated file /usr/share/man/man1/xzgrep.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzegrep.1.gz because associated file /usr/share/man/man1/xzegrep.1.gz (of link group lzma) doesn't exist
update-alternatives: warning: skip creation of /usr/share/man/man1/lzfgrep.1.gz because associated file /usr/share/man/man1/xzfgrep.1.gz (of link group lzma) doesn't exist
Setting up nano (4.8-1ubuntu1) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/editor.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group editor) doesn't exist
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/pico.1.gz because associated file /usr/share/man/man1/nano.1.gz (of link group pico) doesn't exist
Setting up libmpdec2:amd64 (2.4.2-3) ...
Setting up readline-common (8.0-4) ...
Setting up libreadline8:amd64 (8.0-4) ...
Setting up libpython3.8-stdlib:amd64 (3.8.10-0ubuntu1~20.04.1) ...
Setting up python3.8 (3.8.10-0ubuntu1~20.04.1) ...
Setting up libpython3-stdlib:amd64 (3.8.2-0ubuntu2) ...
Setting up python3 (3.8.2-0ubuntu2) ...
running python rtupdate hooks for python3.8...
running python post-rtupdate hooks for python3.8...
Setting up systemctl (1.4.3424-2) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
root@02fdd1b11244: /#
```

A partir deste ponto o que farei é basicamente repetir os passos para instalar e configurar o Apache, PHP, MySQL e Composer como aprendemos no Guia da Semana 2. Então o que você deve fazer é seguir os passos do Guia da Semana 2 para Linux e considerar apenas os itens:

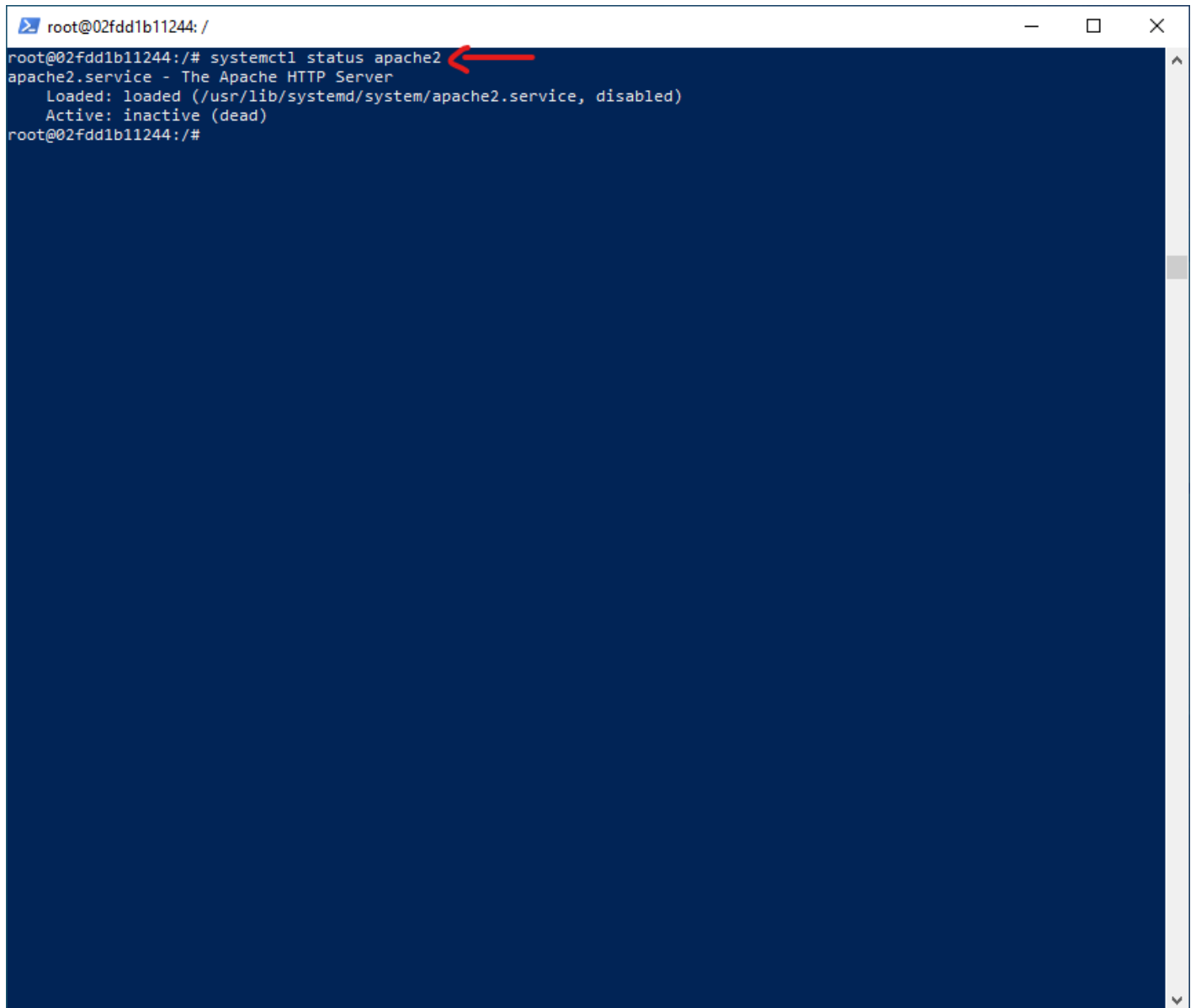
- 1)
- 2)
- 3)
- 4)
- 5)
- 7)
- 8)

Para facilitar, eu vou mostrar os passos tela a tela como destacado a seguir.

Vamos instalar o Apache!

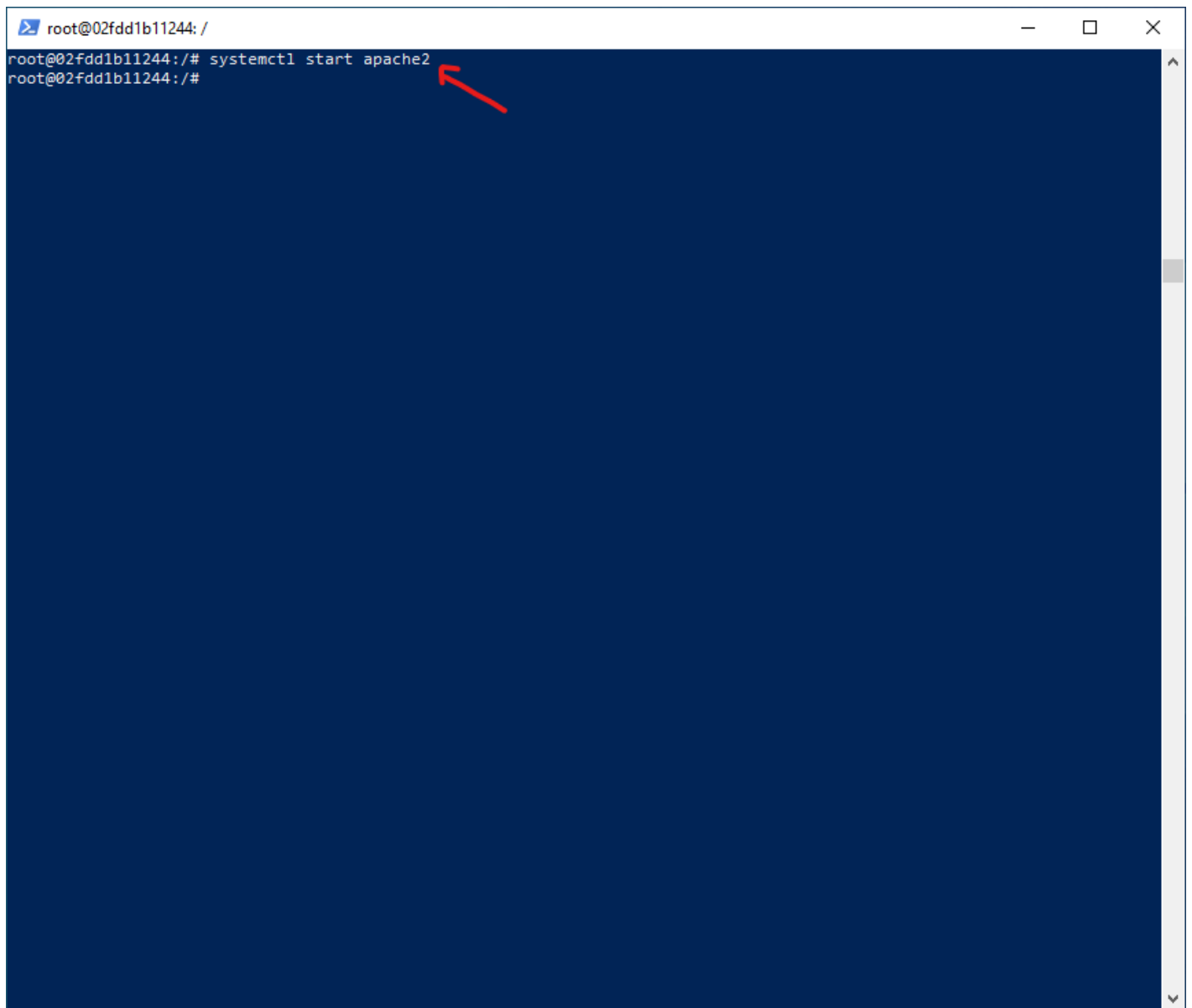
```
root@02fdd1b11244: /
root@02fdd1b11244:/# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils ca-certificates krb5-locales libapr1 libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libasn1-8-heimdal libbrotli1 libcurl4 libgdbm-compat4 libgdbm6 libgssapi-krb5-2 libgssapi3-heimdal
  libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal libicu66 libjansson4 libk5crypto3
  libkeyutils1 libkrb5-26-heimdal libkrb5-3 libkrb5support0 libldap-2.4-2 libldap-common liblua5.2-0 libnghttp2-14
  libperl5.30 libpsl5 libroken18-heimdal librtmp1 libsasl2-2 libsasl2-modules libsasl2-modules-db libssh-4
  libwind0-heimdal libxml2 netbase openssl perl perl-modules-5.30 publicsuffix ssl-cert tzdata
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser ufw gdbm-110n krb5-doc krb5-user
  libsasl2-modules-gssapi-mit | libsasl2-modules-gssapi-heimdal libsasl2-modules-ldap libsasl2-modules-otp
  libsasl2-modules-sql perl-doc libterm-readline-gnu-perl | libterm-readline-perl-perl make libb-debug-perl
  liblocale-codes-perl openssl-blacklist
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils ca-certificates krb5-locales libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libasn1-8-heimdal libbrotli1 libcurl4 libgdbm-compat4 libgdbm6
  libgssapi-krb5-2 libgssapi3-heimdal libhcrypto4-heimdal libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal
  libicu66 libjansson4 libk5crypto3 libkeyutils1 libkrb5-26-heimdal libkrb5-3 libkrb5support0 libldap-2.4-2
  libldap-common liblua5.2-0 libnghttp2-14 libperl5.30 libpsl5 libroken18-heimdal librtmp1 libsasl2-2 libsasl2-modules
  libsasl2-modules-db libssh-4 libwind0-heimdal libxml2 netbase openssl perl perl-modules-5.30 publicsuffix ssl-cert
  tzdata
0 upgraded, 49 newly installed, 0 to remove and 0 not upgraded.
Need to get 21.7 MB of archives.
After this operation, 105 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Execute o comando abaixo para verificar que o Apache não está em execução.

A terminal window with a dark blue background and white text. The window title bar shows a terminal icon, the text 'root@02fdd1b11244: /', and standard window controls (minimize, maximize, close). The terminal content shows the command 'systemctl status apache2' being executed. The output indicates that the 'apache2.service' is loaded but inactive (dead). A red arrow points to the command line.

```
root@02fdd1b11244: /  
root@02fdd1b11244:/# systemctl status apache2  
apache2.service - The Apache HTTP Server  
   Loaded: loaded (/usr/lib/systemd/system/apache2.service, disabled)  
   Active: inactive (dead)  
root@02fdd1b11244:/#
```

Prossiga com o comando a seguir para colocar o Apache em execução.



```
root@02fdd1b11244: /  
root@02fdd1b11244: /# systemctl start apache2  
root@02fdd1b11244: /#
```

Verifique novamente o status com o comando abaixo e verá que o Apache está rodando corretamente.

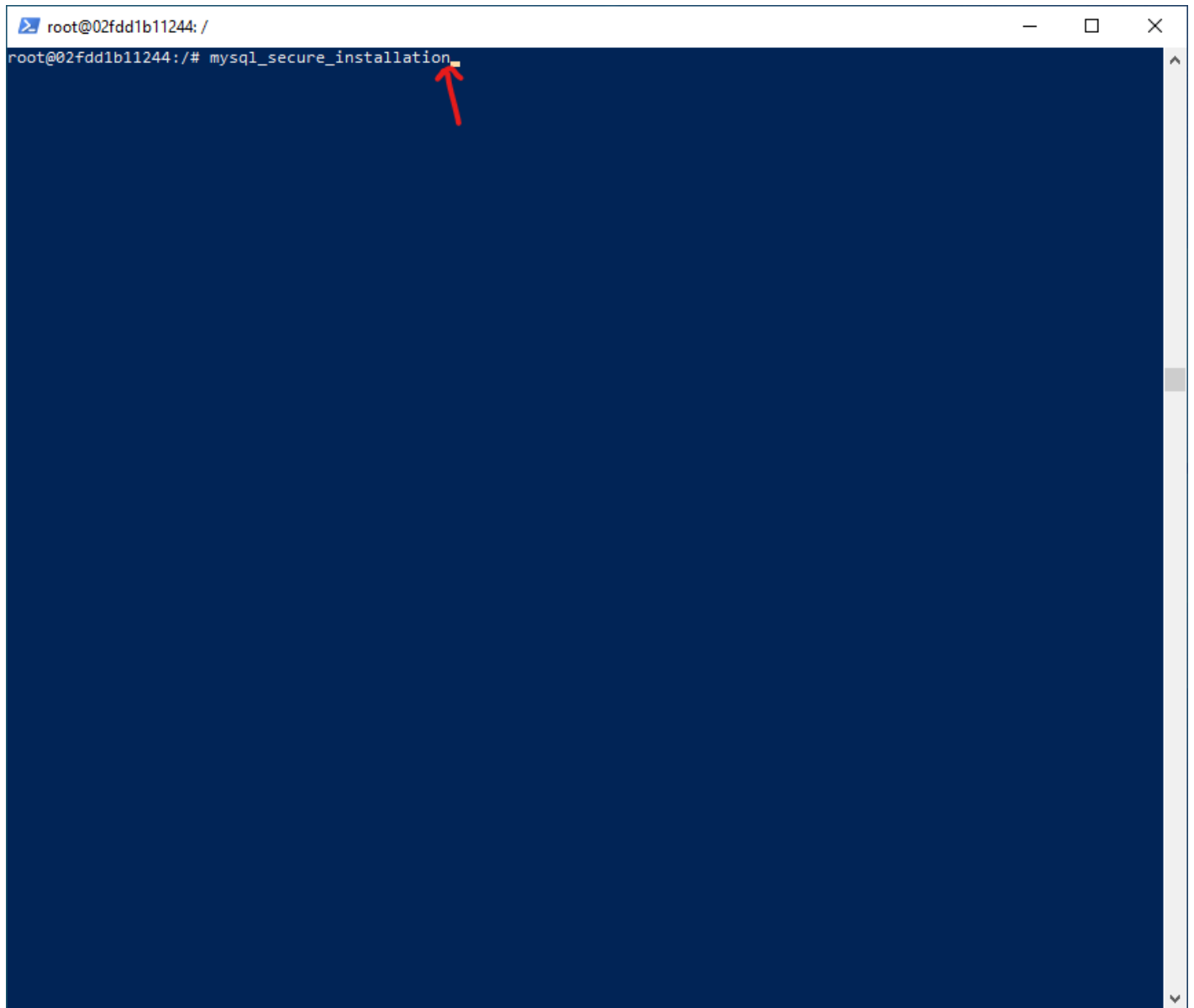
```
root@02fdd1b11244: /  
root@02fdd1b11244:/# systemctl status apache2  
apache2.service - The Apache HTTP Server  
  Loaded: loaded (/usr/lib/systemd/system/apache2.service, disabled)  
  Active: active (running)  
root@02fdd1b11244:/#
```

Vamos agora instalar o MySQL Server!

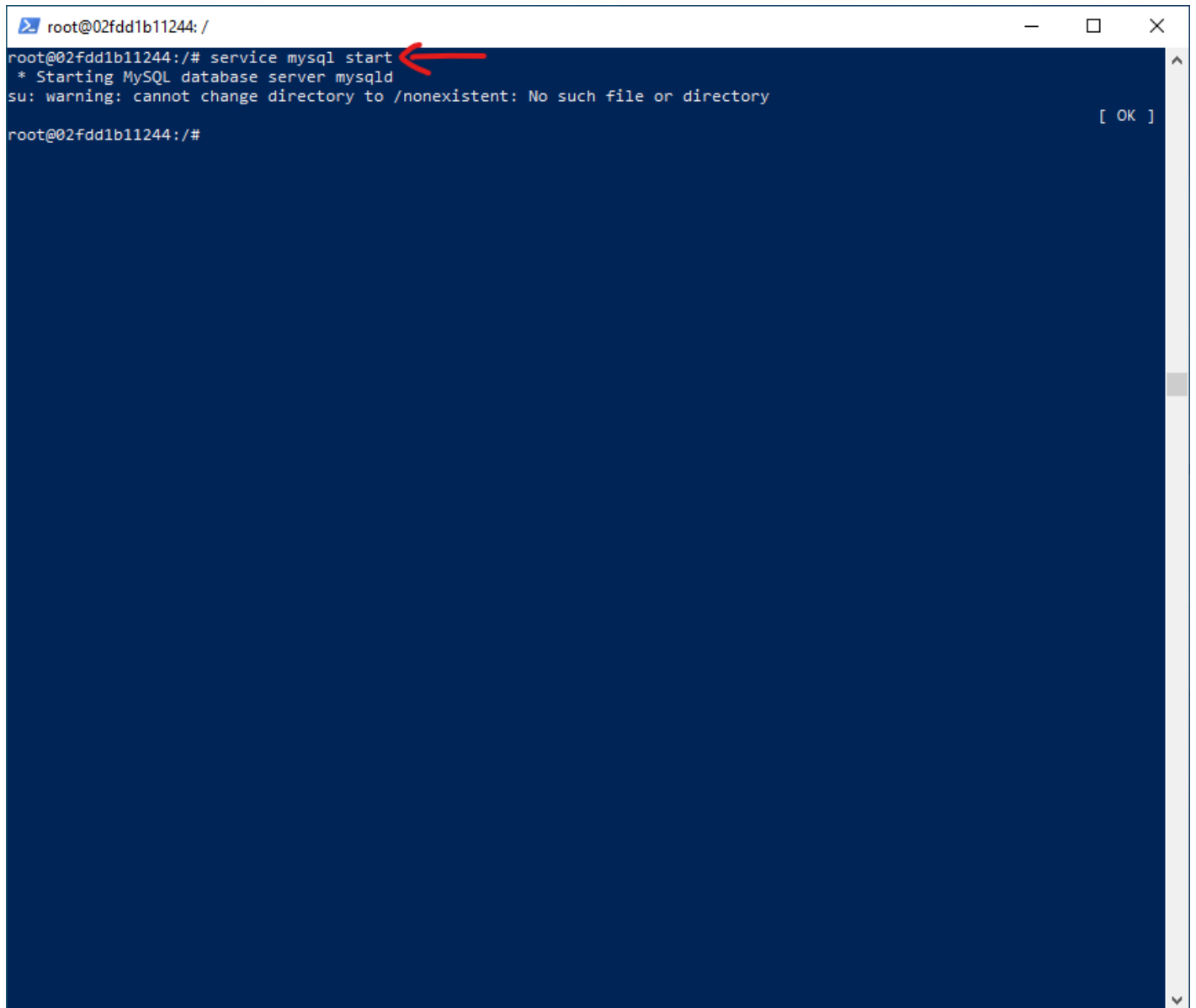
```
root@02fdd1b11244: /  
root@02fdd1b11244:/# apt-get install mysql-server  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  libaio1 libbsd0 libcgi-fast-perl libcgi-pm-perl libedit2 libencode-locale-perl libevent-core-2.1-7  
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl  
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmecab2 libnuma1 libtimedate-perl liburi-perl  
  mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server-8.0  
  mysql-server-core-8.0 psmisc  
Suggested packages:  
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyca  
The following NEW packages will be installed:  
  libaio1 libbsd0 libcgi-fast-perl libcgi-pm-perl libedit2 libencode-locale-perl libevent-core-2.1-7  
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl  
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libmecab2 libnuma1 libtimedate-perl liburi-perl  
  mecab-ipadic mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server  
  mysql-server-8.0 mysql-server-core-8.0 psmisc  
0 upgraded, 30 newly installed, 0 to remove and 0 not upgraded.  
Need to get 32.1 MB of archives.  
After this operation, 264 MB of additional disk space will be used.  
Do you want to continue? [Y/n]
```

Prossiga com as modificações na senha padrão do root do MySQL. Escolha um padrão de tamanho de senha adequada para você e quando solicitado a senha, utilize (por exemplo) esta senha: **Gio2017!** . *Esta é apenas uma dica de senha!!!*

```
root@02fdd1b11244: /  
root@02fdd1b11244: /# mysql_secure_installation_
```

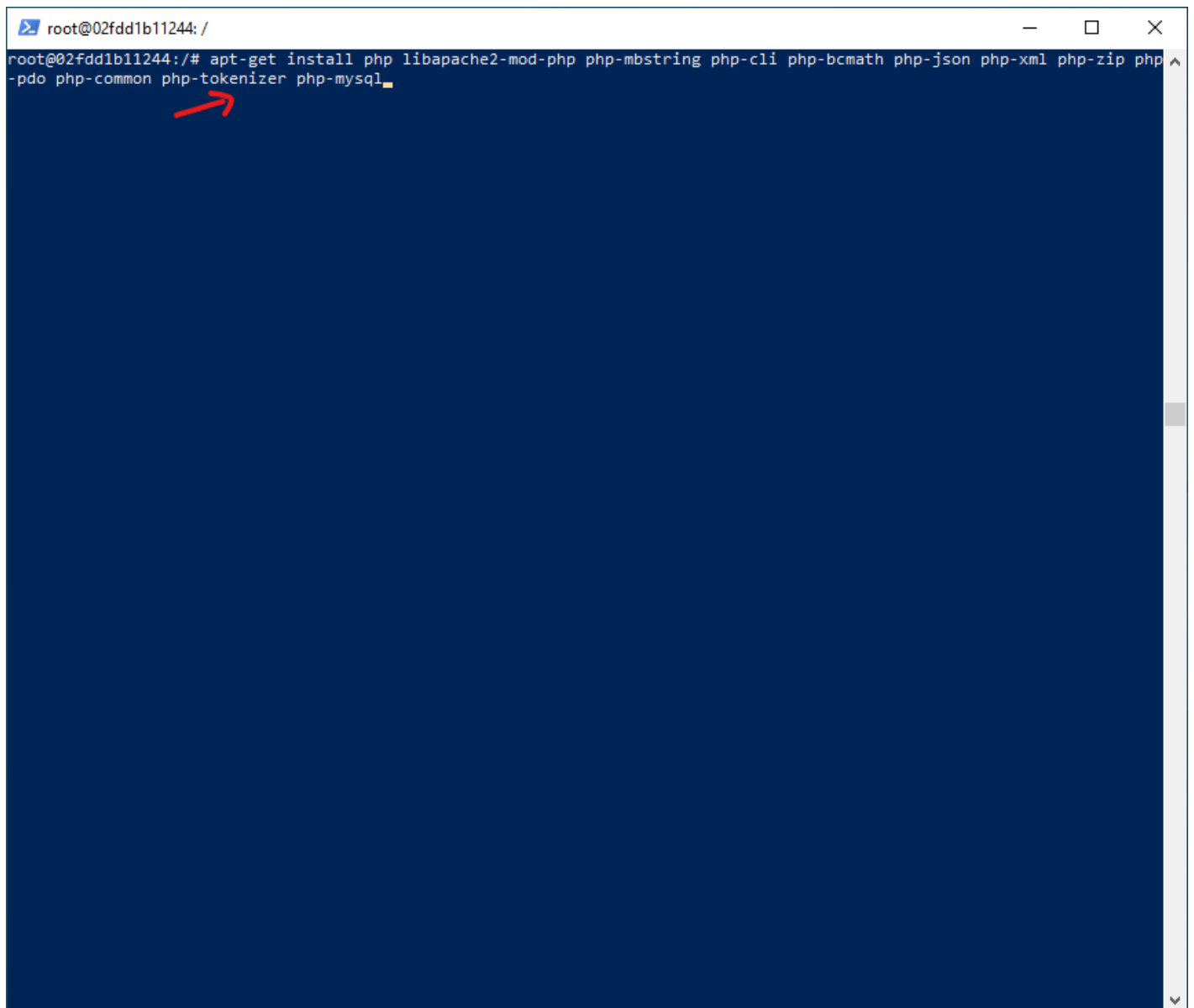
A terminal window with a dark blue background. The title bar shows 'root@02fdd1b11244: /'. The command prompt is 'root@02fdd1b11244: /#'. The command entered is 'mysql_secure_installation_'. A red arrow points to the underscore at the end of the command. The terminal has a scrollbar on the right side.

Inicie o mysql com o comando abaixo:



```
root@02fdd1b11244: /  
root@02fdd1b11244:/# service mysql start  
* Starting MySQL database server mysqld  
su: warning: cannot change directory to /nonexistent: No such file or directory  
[ OK ]  
root@02fdd1b11244:/#
```

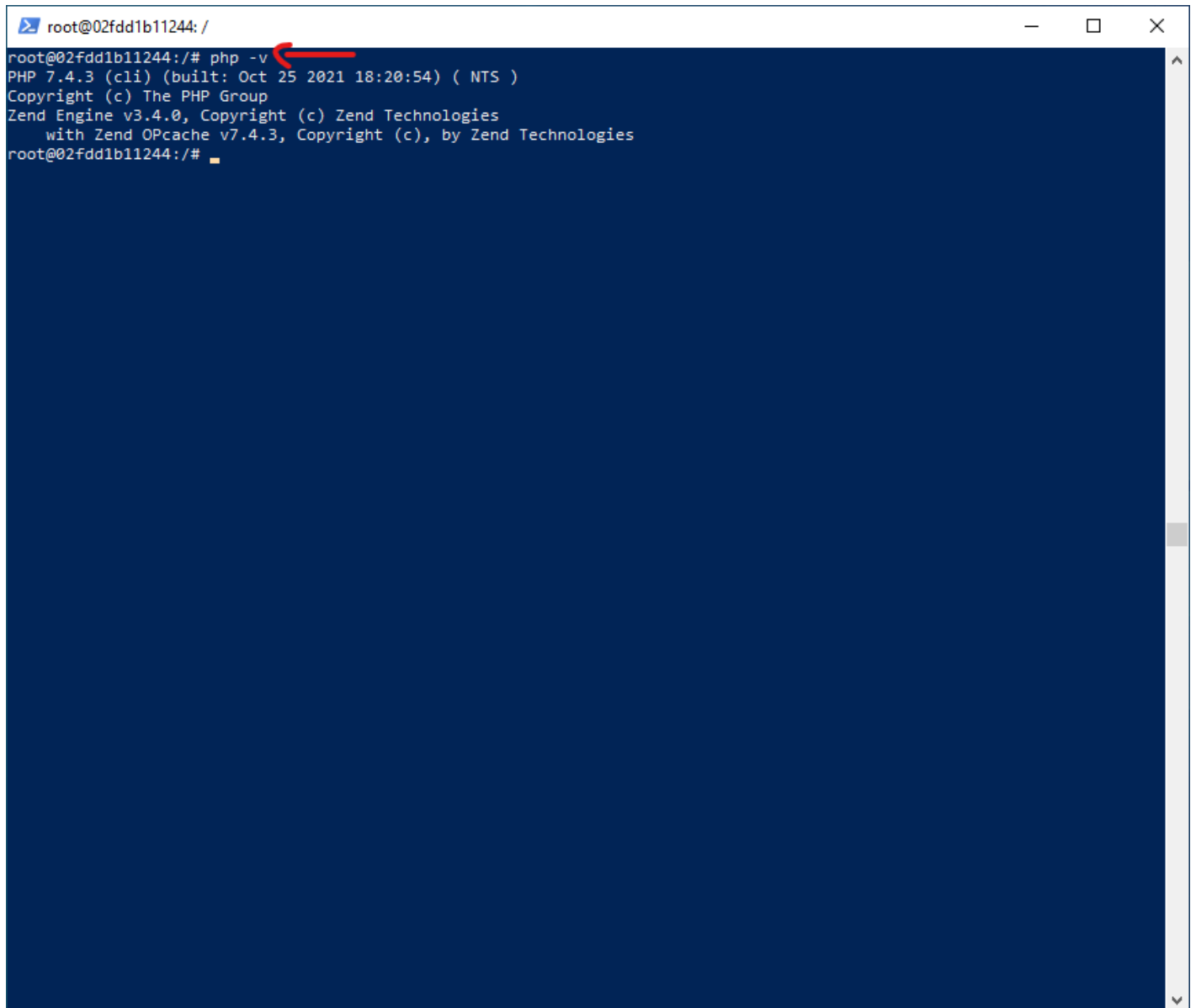
A próxima etapa é instalar o PHP. Vamos em frente com o comando a seguir.



A terminal window with a dark blue background. The title bar shows 'root@02fdd1b11244: /'. The command entered is 'apt-get install php libapache2-mod-php php-mbstring php-cli php-bcmath php-json php-xml php-zip php-pdo php-common php-tokenizer php-mysql'. A red arrow points to the end of the command line.

```
root@02fdd1b11244: /  
root@02fdd1b11244:/# apt-get install php libapache2-mod-php php-mbstring php-cli php-bcmath php-json php-xml php-zip php-  
-pdo php-common php-tokenizer php-mysql
```


Na sequência podemos verificar com o comando abaixo a versão recentemente instalada.

A terminal window with a dark blue background and white text. The window title bar shows 'root@02fdd1b11244: /' and standard window controls. The terminal content shows the command 'php -v' being executed, followed by the output: 'PHP 7.4.3 (cli) (built: Oct 25 2021 18:20:54) (NTS)', 'Copyright (c) The PHP Group', 'Zend Engine v3.4.0, Copyright (c) Zend Technologies', and 'with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies'. A red arrow points to the 'v' in the command. The prompt 'root@02fdd1b11244: /#' is visible at the end of the output.

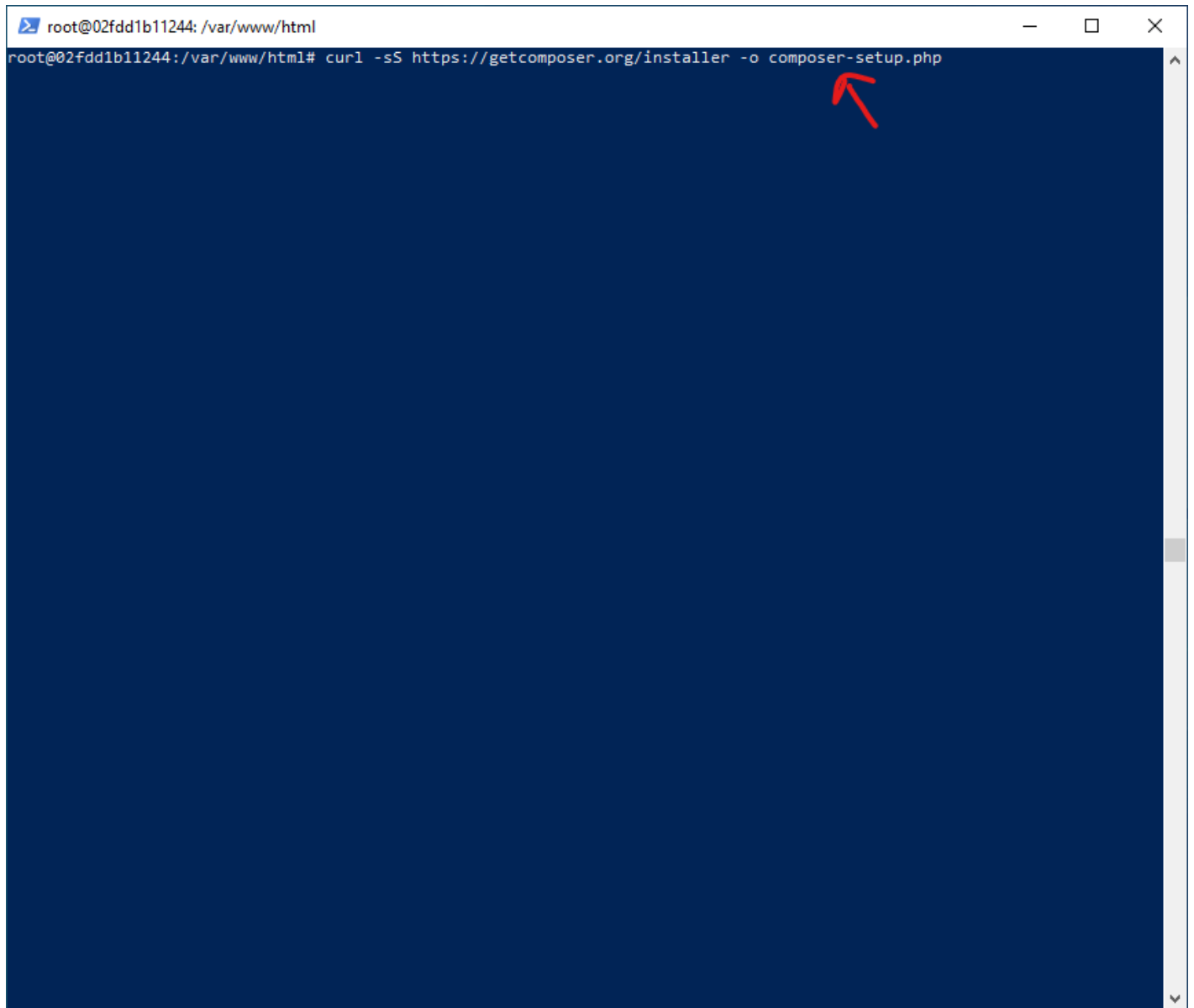
```
root@02fdd1b11244: /  
root@02fdd1b11244: /# php -v  
PHP 7.4.3 (cli) (built: Oct 25 2021 18:20:54) ( NTS )  
Copyright (c) The PHP Group  
Zend Engine v3.4.0, Copyright (c) Zend Technologies  
with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies  
root@02fdd1b11244: /#
```

A seguir instale o curl!


```
root@02fdd1b11244: /var/www/html
root@02fdd1b11244:/var/www/html# apt install curl
```



Agora chegou o momento de baixar o composer!!



A terminal window with a dark blue background. The title bar shows 'root@02fdd1b11244: /var/www/html'. The command prompt shows 'root@02fdd1b11244:/var/www/html# curl -sS https://getcomposer.org/installer -o composer-setup.php'. A red arrow points to the end of the command.

```
root@02fdd1b11244: /var/www/html
root@02fdd1b11244:/var/www/html# curl -sS https://getcomposer.org/installer -o composer-setup.php
```

Prossiga com a instalação do composer!

```
root@02fdd1b11244: /var/www/html
root@02fdd1b11244:/var/www/html# php composer-setup.php --install-dir=/usr/local/bin --filename=composer
All settings correct for using Composer
Downloading...

Composer (version 2.1.12) successfully installed to: /usr/local/bin/composer
Use it: php /usr/local/bin/composer
root@02fdd1b11244:/var/www/html#
```

Por último, precisamos configurar o servidor web Apache para hospedar o site Laravel. Para que isso funcione, precisamos criar um arquivo host virtual. Execute o comando abaixo:

```
pico /etc/apache2/sites-available/laravel.conf
```

E adicione as linhas abaixo:

```
<VirtualHost *:80>
ServerName localhost
ServerAdmin admin@example.com
DocumentRoot /var/www/html/laravel-crud-app/public
<Directory /var/www/html/laravel-crud-app>
AllowOverride All
</Directory>
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Salve e feche o arquivo. Para salvar o arquivo tecle: **CRTL+X** e depois **Y** e em seguida **ENTER**.

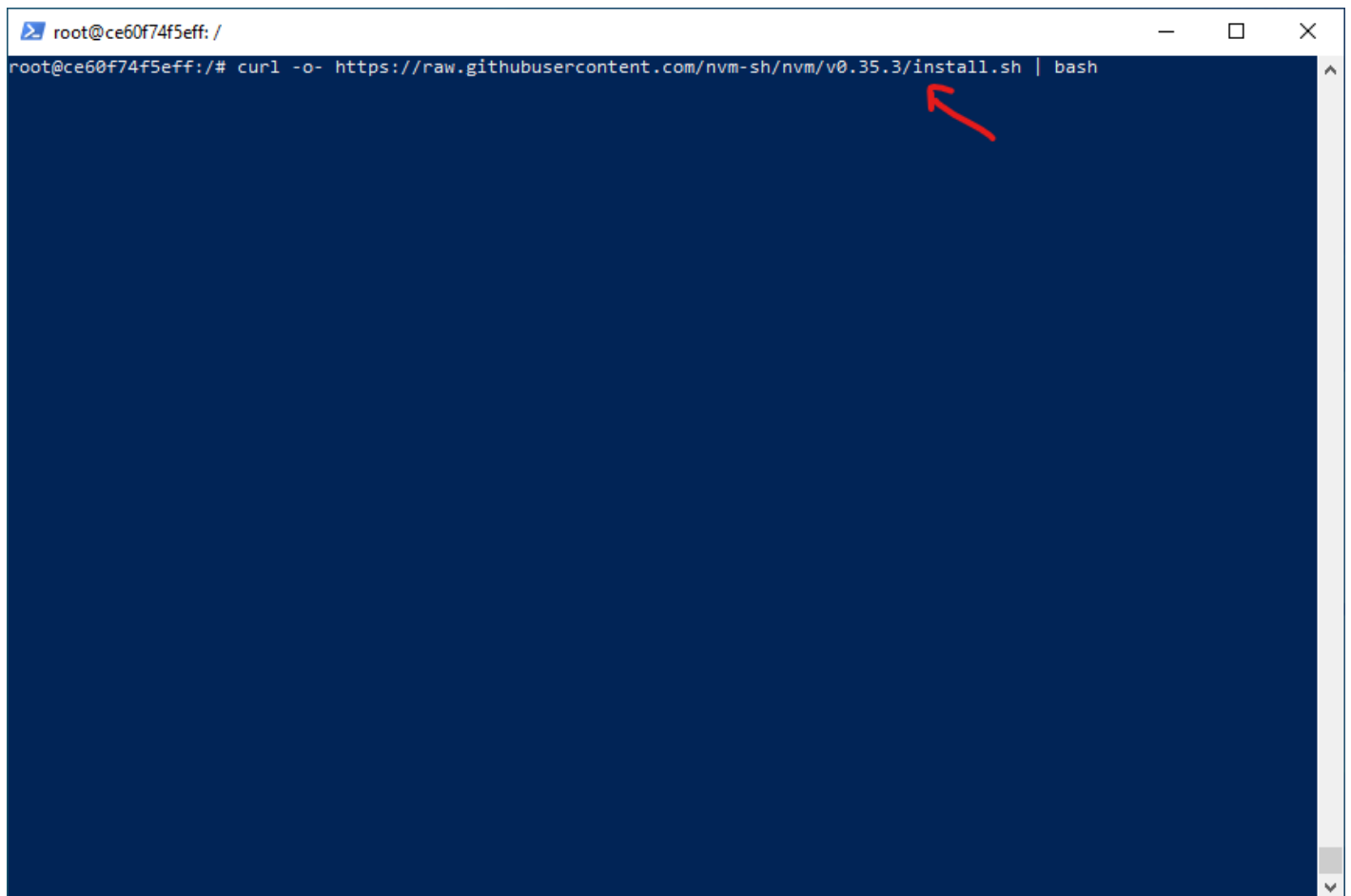
Em seguida, habilite o site do Laravel e o módulo de reescrita do Apache usando estes dois comandos.

```
a2ensite laravel.conf
```

```
a2enmod rewrite
```

Caso esteja seguindo os passos que indiquei lá do Guia da Semana 2, você não precisa executar o restante dos comandos depois do a2enmod.

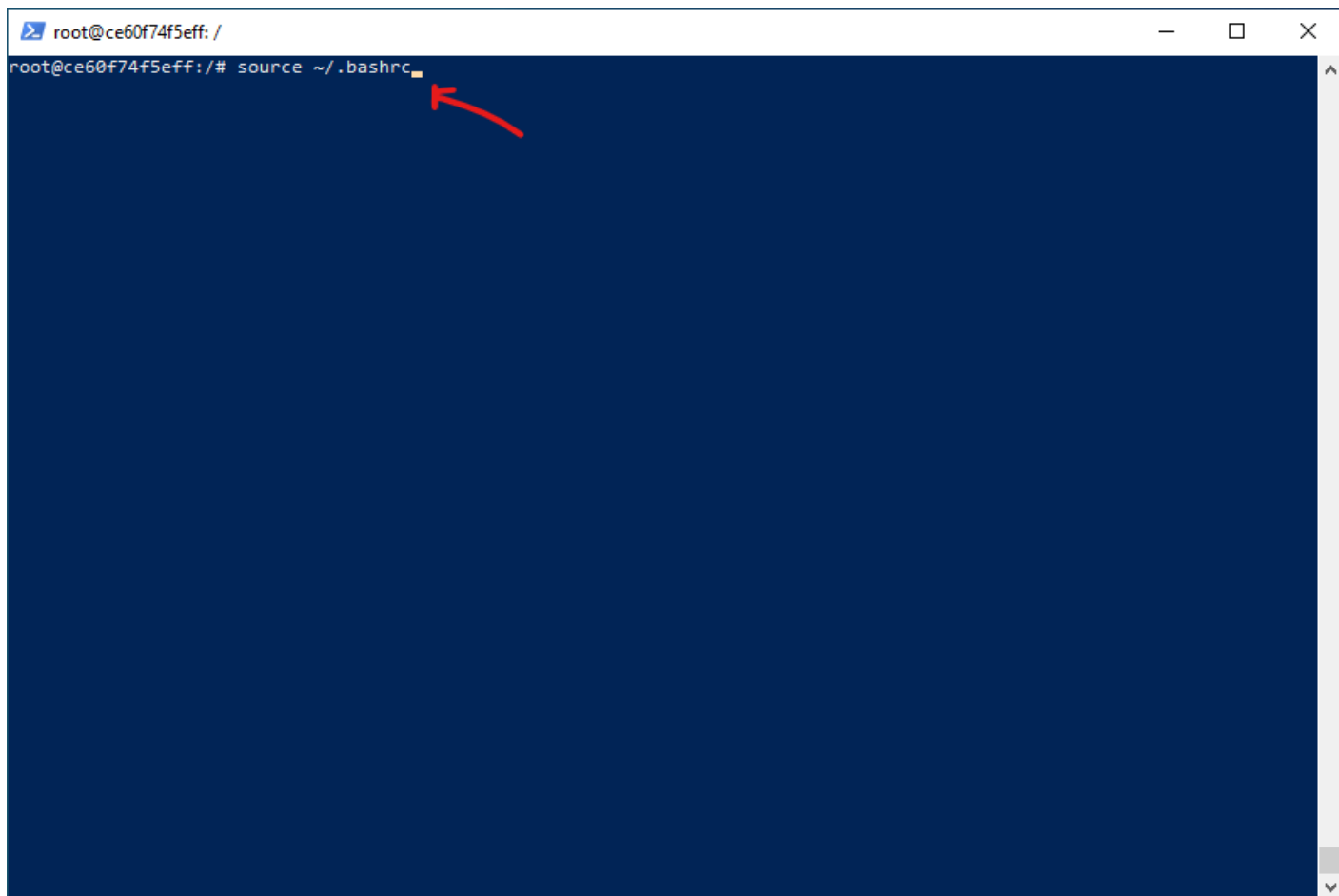
O próximo passo é instalarmos o Node.js. O Node.js é um ambiente de execução do JavaScript criado para a programação do lado do servidor e que permite aos desenvolvedores criarem funcionalidades de back-ends utilizando o JavaScript. Para baixar o script de instalação do Node.js no terminal do Linux da nossa imagem docker, execute o comando:

A terminal window with a dark blue background and white text. The title bar shows 'root@ce60f74f5eff: /'. The command prompt shows 'root@ce60f74f5eff:/# curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash'. A red arrow points to the URL in the command. The terminal is mostly empty below the command line.

```
root@ce60f74f5eff: /  
root@ce60f74f5eff:/# curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
```

Execute o comando a seguir para que o Node.js esteja disponível no path global do Linux.

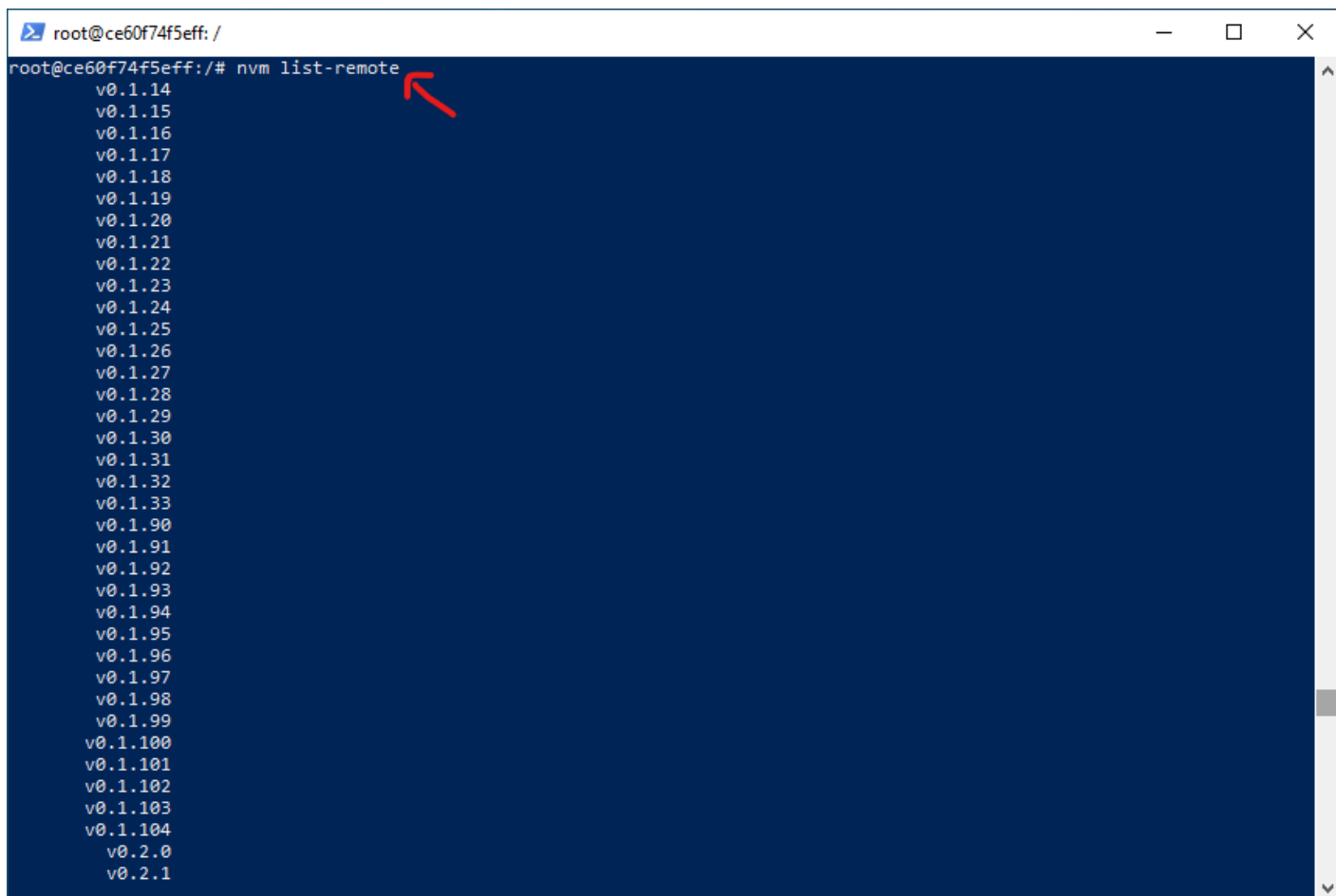
```
root@ce60f74f5eff: /
root@ce60f74f5eff: /# source ~/.bashrc
```



Próximo para ver lista de versões disponíveis.

```
root@ce60f74f5eff: /
root@ce60f74f5eff: /# nvm list-remote
```

- v0.1.14
- v0.1.15
- v0.1.16
- v0.1.17
- v0.1.18
- v0.1.19
- v0.1.20
- v0.1.21
- v0.1.22
- v0.1.23
- v0.1.24
- v0.1.25
- v0.1.26
- v0.1.27
- v0.1.28
- v0.1.29
- v0.1.30
- v0.1.31
- v0.1.32
- v0.1.33
- v0.1.90
- v0.1.91
- v0.1.92
- v0.1.93
- v0.1.94
- v0.1.95
- v0.1.96
- v0.1.97
- v0.1.98
- v0.1.99
- v0.1.100
- v0.1.101
- v0.1.102
- v0.1.103
- v0.1.104
- v0.2.0
- v0.2.1



E instale a versão indicada abaixo:

```
root@ce60f74f5eff: /
root@ce60f74f5eff: /# nvm install v14.18.0
```

Certifique-se depois da versão que acabamos de instalar.

```
root@ce60f74f5eff: /
root@ce60f74f5eff: /# nvm list
->      v14.18.0
      system
default -> v14.18.0
node -> stable (-> v14.18.0) (default)
stable -> 14.18 (-> v14.18.0) (default)
iojs -> N/A (default)
unstable -> N/A (default)
lts/* -> lts/gallium (-> N/A)
lts/argon -> v4.9.1 (-> N/A)
lts/boron -> v6.17.1 (-> N/A)
lts/carbon -> v8.17.0 (-> N/A)
lts/dubnium -> v10.24.1 (-> N/A)
lts/erbium -> v12.22.7 (-> N/A)
lts/fermium -> v14.18.1 (-> N/A)
lts/gallium -> v16.13.0 (-> N/A)
root@ce60f74f5eff: /#
```

```
root@ce60f74f5eff: /
root@ce60f74f5eff: /# node -v
v14.18.0
root@ce60f74f5eff: /#
```

O próximo passo é instalar o npm, que é o gerenciador de pacotes do Node.js.

```
root@ce60f74f5eff: /
root@ce60f74f5eff: /# apt-get install npm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  npm
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 583 kB of archives.
After this operation, 3468 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/universe amd64 npm all 6.14.4+ds-1ubuntu2 [583 kB]
80% [1 npm 581 kB/583 kB 100%]
```


Como muitos pacotes foram instalados precisamos salvar todas essas alterações efetuadas na imagem do Ubuntu. ****Matenha esse PowerShell aberto. ****

Abra outro PowerShell ou o CMD do Windows e execute o seguinte comando:

```
docker ps
```

Você verá na saída deste comando que o container tem um ID. Copie este ID e prossiga com o comando a seguir no novo shell que acabou de abrir.

```
docker commit -m "Laravel Crud - Version-1.0" -a "seu nome" ID-DO-SEU-CONTAINER seunome/com320-app
```

Este processo pode demorar um pouco até finalizar. Após finalizado, execute o comando:

```
docker images
```

E terá uma saída parecida com esta:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
julio/com320-app-v1.0	latest	930b815d6fb1	2 hours ago	2.05GB
ubuntu	latest	1318b700e415	3 months ago	72.8MB
hello-world	latest	d1165f221234	8 months ago	13.3kB

Pronto, feito isso, agora **você pode fechar o primeiro PowerShell** que utilizamos para instalar todos os pacotes.

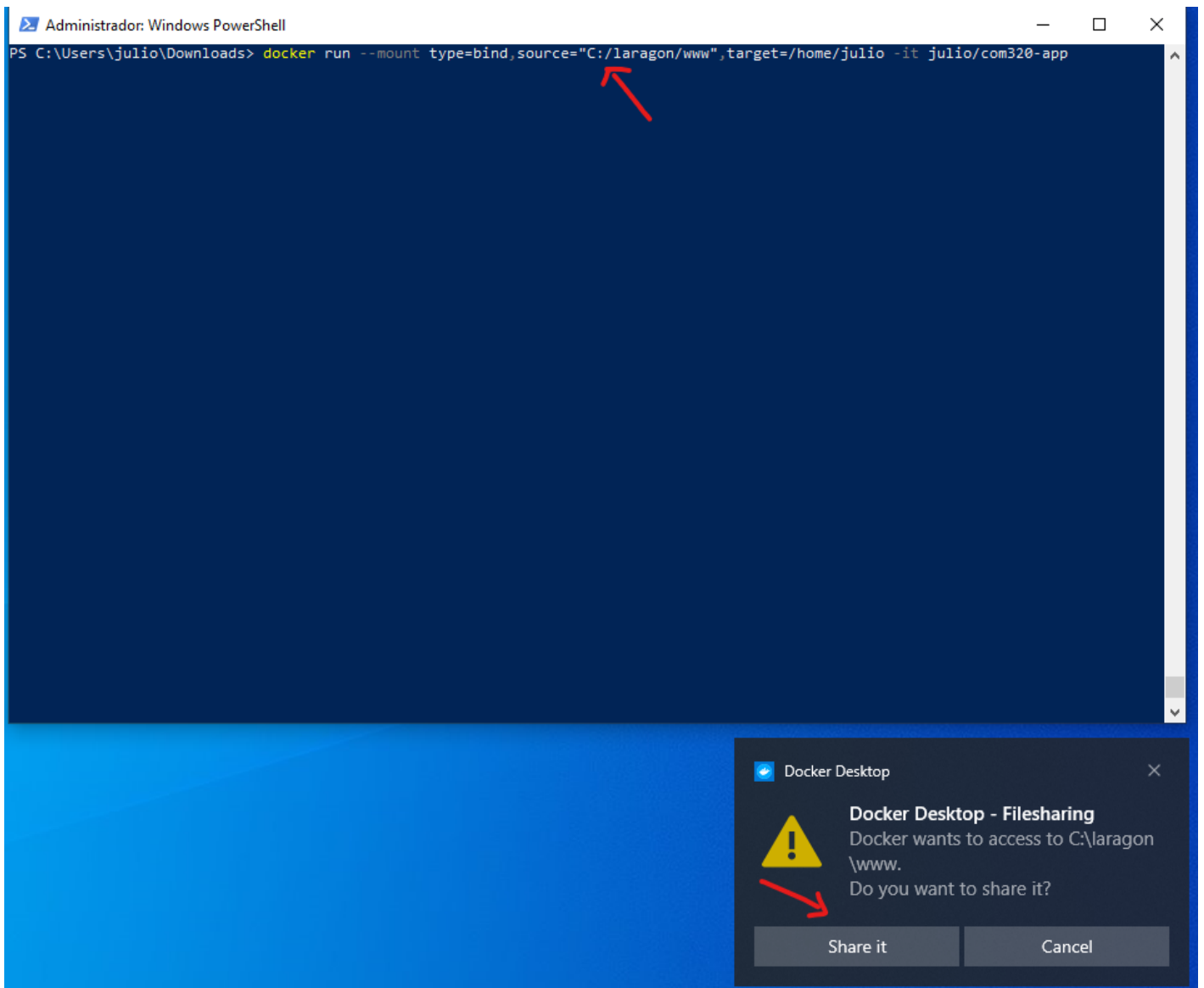
Execução da Imagem Atualizada, Abertura de Portas e Cópia do Projeto laravel-crud-app da Semana 5

O que temos até o momento é uma imagem Linux toda customizada para poder executar um back-end com Laravel e ferramentas correlatas. O que precisamos fazer é seguir alguns passos e o primeiro deles é copiar os arquivos do host (no nosso caso o Windows) para a imagem Linux que atualizamos.

No novo PowerShell, você pode agora executar a nova imagem, já atualizada. Como estamos no Windows, eu considero que o projeto laravel-crud-app está na pasta: **C:\laragon\www**.

```
docker run --mount type=bind,source="C:/laragon/www/",target=/home/julio -it julio/com320-app-v1
```

O resultado da execução do comando anterior pode ser visto a seguir

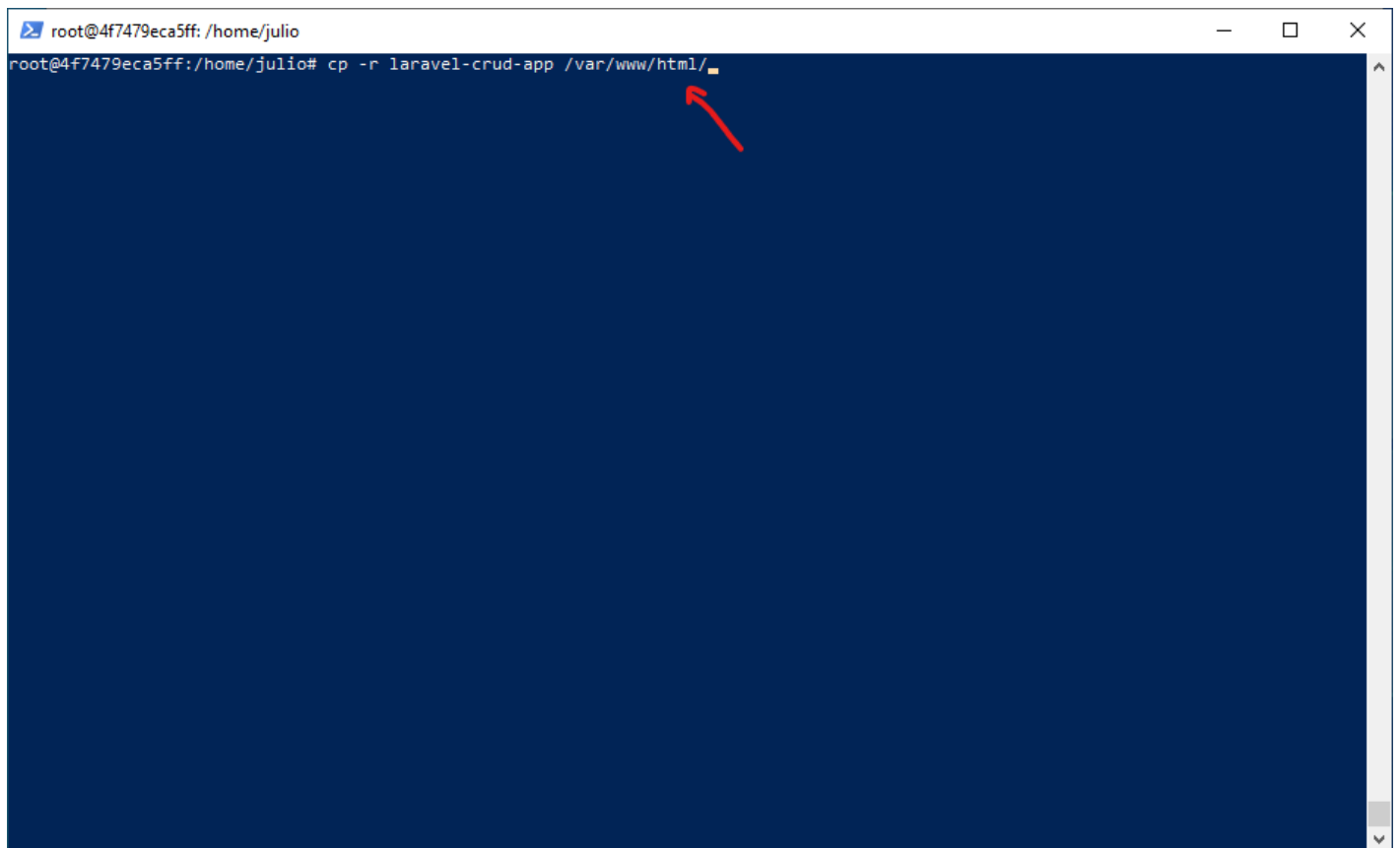


Depois de feito procedimento de montagem remoto, você pode acessar a pasta no docker que está sendo mapeada do host (Windows). No meu caso eu pedi para montar em: `/home/julio`. O que temos que fazer é fazer a cópia do laravel-crud-app para a pasta do apache em: `/var/www/html`. Primeiro eu entro em `/home/julio`:

```
cd /home/julio
```

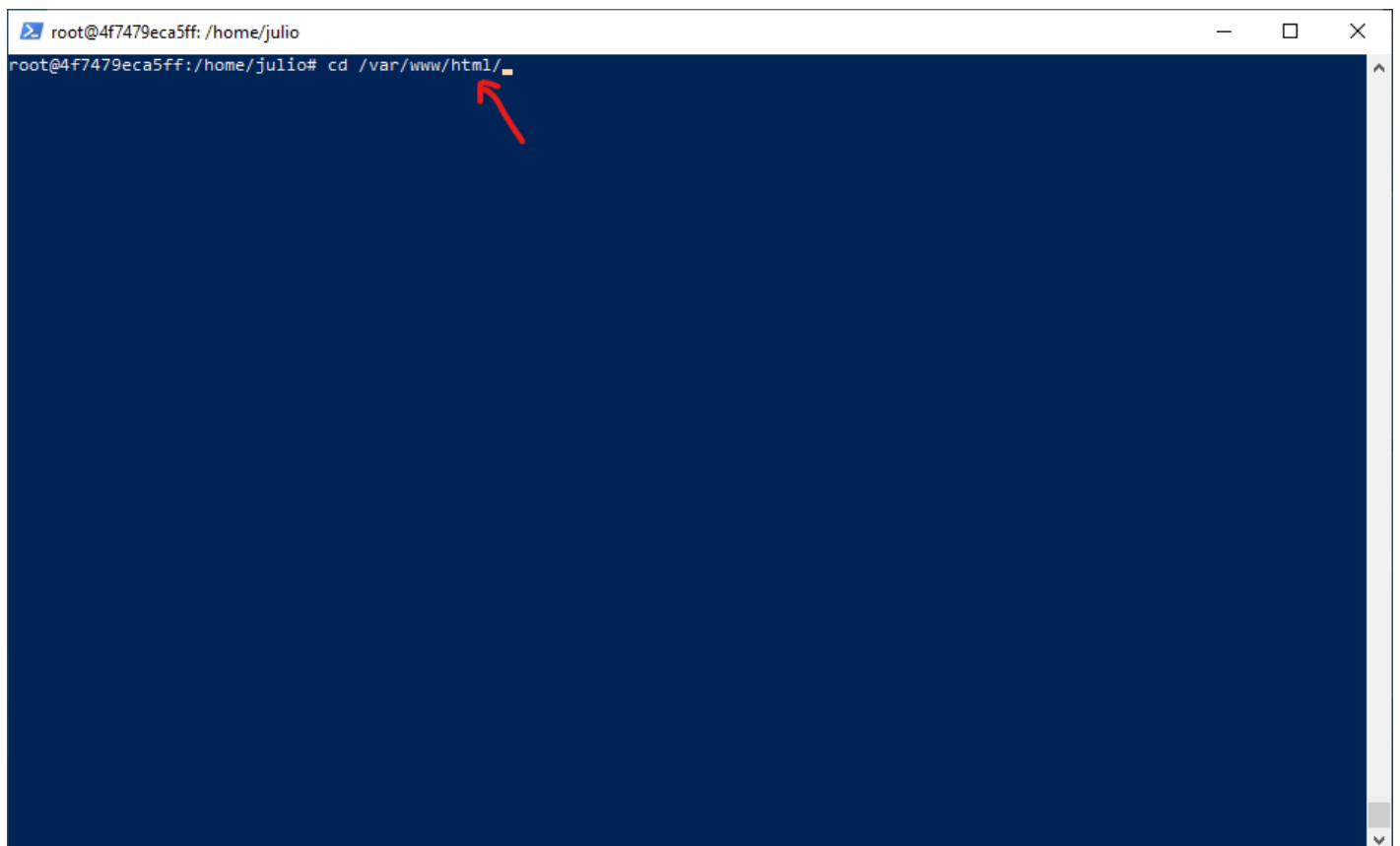
Depois executo o comando de cópia como a seguir:

```
root@4f7479eca5ff: /home/julio
root@4f7479eca5ff:/home/julio# cp -r laravel-crud-app /var/www/html/
```



Na sequência vamos acessar a pasta /var/www/html e ter certeza que a cópia dos arquivos foi feita corretamente.

```
root@4f7479eca5ff: /home/julio
root@4f7479eca5ff:/home/julio# cd /var/www/html/
```



O resultado é mostrado a seguir.

```
root@4f7479eca5ff: /var/www/html
root@4f7479eca5ff:/var/www/html# ls -la
total 92
drwxr-xr-x  1 root root  4096 Nov 11 18:37 .
drwxr-xr-x  1 root root  4096 Nov 11 07:31 ..
-rw-r--r--  1 root root 57721 Nov 11 08:09 composer-setup.php
-rw-r--r--  1 root root 10918 Nov 11 07:32 index.html
-rw-r--r--  1 root root   18 Nov 11 07:51 info.php
drwxr-xr-x 12 root root  4096 Nov 11 18:37 laravel-crud-app
root@4f7479eca5ff:/var/www/html#
```

Matenha o PowerShell aberto sem digitar exit!!!

Atualização do Composer e validação da migração da aplicação laravel

Quando copiamos um projeto Laravel de um host para outro, ou seja da nossa máquina local para um servidor em produção, temos que ajustar algumas configurações para que a aplicação rode adequadamente.

Ainda dentro da imagem Linux, acesse a pasta **/var/www/html/laravel-crud-app**.

```
cd /var/www/html/laravel-crud-app
```

Na sequência execute o comando:

```
compose update
```

Este processo pode demorar um pouco!!

Considerando que esteja dentro da pasta **laravel-crud-app**, execute também o comando a seguir para ajustar problemas de permissão de arquivos do laravel.

```
chmod -R 777 bootstrap/cache storage
```

Ainda na pasta **laravel-crud-app**, você deve acessar o arquivo **.env** para verificarmos se as configurações da base de dados que fizemos na Semana 5 estão OK e são as mesmas depois da cópia para a imagem docker.

```
pico .env
```

Verifique basicamente as linhas abaixo e constate que usuário, senha, porta estão OK.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_db
DB_USERNAME=root
DB_PASSWORD=Gio2017!)
```

Agora vamos prosseguir na linha de comando para limpar o ambiente que foi copiado, com a execução dos comandos a seguir:

```
php artisan key:generate
php artisan cache:clear
php artisan migrate
```

Como fizemos novos ajustes na imagem depois do último commit, é o momento para novamente salvar todas as alterações com um novo commit. **Matenha esse PowerShell aberto.** Abra outro PowerShell ou o CMD do Windows e execute o seguinte comando:

```
docker ps
```

Você verá na saída deste comando que o container tem um novo ID. Copie este novo ID e prossiga com o comando a seguir no novo shell que acabou de abrir.

```
docker commit -m "Laravel Crud - Version-2.0" -a "seu nome" NOVO-ID-DO-SEU-CONTAINER seunome/com320-app-v2.0
```

Este processo pode demorar um pouco até finalizar. Após finalizado, execute o comando:

```
docker images
```

E terá uma saída parecida com esta:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
julio/com320-app-v2.0	latest	930b815d6fb1	3 hours ago	2.15GB
julio/com320-app-v1.0	latest	930b815d6fb1	3 hours ago	2.05GB
ubuntu	latest	1318b700e415	2 months ago	72.8MB
hello-world	latest	d1165f221234	8 months ago	13.3kB

Pronto, feito isso, agora você pode fechar o PowerShell anterior que utilizamos para finalizar os últimos ajustes.

Vamos na sequência executar a nova imagem com o comando a seguir:

```
docker run -it -p 8080:80 julio/com320-app-v2.0
```

Significados:

- it - mantém o prompt de comando ativo para o usuário
- p - 8080:80 - especifica que o host vai abrir a porta 8080 para que haja comunicação interna com o container
- julio/com320-app-v2.0 - é o nome da imagem

A saída gerada será um prompt parecido como o que já conhecemos:

```
root@09b523c9ed7c:/#
```

Vamos então neste prompt iniciar o apache e o mysql:

```
service apache2 start
service mysql start
```

Matenha o PowerShell aberto para que possamos testar o front-end que criamos na Semana 3 com o back-end que está na imagem docker. Vamos executar esses passos na próxima seção.

Teste do Front-End com o Back-End

Ao final do [Guia da Semana 3] (<https://gitlab.com/univesp-com320/guias-praticos-semana3/-/tree/main/guia-windows-angular-crud-app#rodando-a-aplica%C3%A7%C3%A3o>) aprendemos como executar o front-end. Na ocasião você deveria se perguntar porque não conseguimos cadastrar os dados via back-end. Isso ocorre porque dividimos a aplicação em duas partes e finalizamos o back-end na Semana 5. Mas faltava toda a ligação que efetuamos neste Guia da Semana 6. Vamos aos testes!!!

Passo 1

Antes de inicializar o front-end precisamos fazer uma pequena atualização em um arquivo que indica o caminho do back-end. Acesse a seguinte pasta (Windows):

```
No Windows
C:/laragon/www/angular-crud-app/src/environments
```

Abra o arquivo **environment.ts** e altere a linha:

```
url_api: 'http://localhost/api/'
```

para

```
url_api: 'http://localhost:8080/api/'
```

Passo 2

Abra um novo PowerShell, acesse a pasta da aplicação **angular-crud-app** e execute o comando:

```
npm start
```

A saída gerada deve ser parecida como mostrada a seguir

```
> angular-crud-app@0.0.0 start /var/www/html/angular-crud-app
> ng serve
```

```
✓ Browser application bundle generation complete.
```

```
Initial Chunk Files | Names | Size
vendor.js | vendor | 2.49 MB
polyfills.js | polyfills | 128.51 kB
main.js | main | 21.86 kB
runtime.js | runtime | 12.56 kB
styles.css | styles | 1.90 kB

| Initial Total | 2.65 MB

Lazy Chunk Files | Names | Size
default-node_modules_angular_forms___ivy_ngcc___fesm2015_forms_js.js | - | 314.56 kB
src_app_components_universidades_universidades_module_ts.js | - | 27.59 kB
src_app_components_pais_pais_module_ts.js | - | 18.35 kB

Build at: 2021-11-11T22:50:04.679Z - Hash: 8cd130606fd9265726be - Time: 13081ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/

✓ Compiled successfully.
```

Passo 3

Veja que na penúltima linha ele indica que para acessarmos o front-end temos que abrir o browser no link: <http://localhost:4200/>. Faça isso e interaja com a aplicação, realizando o cadastro de países, universidades, realizando remoções e atualizações.

Conclusão

Espero que todos consigam chegar ao final deste guia e vejam o ambiente em funcionamento. Vejam que não é uma tarefa trivial construir aplicações Web. Mesmo que tenhamos a disposição inúmeros frameworks que facilitam o processo pois encapsulam diversas camadas, ainda assim no começo tudo pode parecer muito obscuro. Isso ocorre pois embora seja mais rápido construir a aplicação, é preciso conhecer de linguagens como Java Script, entender um pouco do funcionamento dos frameworks de modo a explorá-los e aprender a médio prazo. Que este material seja o primeiro a despertar o seu interesse em aprender ainda mais.