

# Tutorial como instalar Docker no Raspberry Pi

## Objetivos

- Entender o que é o software Docker e sua proposta
- Prover os passos necessários para instalar o Docker no Raspberry Pi

## Setup

Para que seja possível seguir este tutorial, é necessário que tenha os itens disponíveis:

- Raspberry Pi
- Fonte de alimentação dimensionada com corrente mínima para seu Raspberry Pi
- Unidade de armazenamento: cartão SD, pen drive, disco rígido USB ou SSD USB.
- Conexão com internet (wi-fi ou cabeada)

## O que é o software Docker?

O **Docker** é também uma plataforma que facilita a criação, implantação e execução de aplicativos em *containers*. Um *container* é um encapsulamento leve e independente que pode conter todo o código e as bibliotecas necessárias para executar um aplicativo, incluindo o próprio aplicativo. Desta forma, o Docker permite empacotar um aplicativo juntamente com suas dependências em um *container*, garantindo que ele funcione consistentemente em outros ambientes, seja em um ambiente de desenvolvimento, teste ou produção. É interessante destacar que um *container* pode não funcionar em todos os ambientes, é necessário verificar a compatibilidade do software encapsulado com arquitetura x86 e arm, porém há formas de emular a compatibilidade.

Aqui estão algumas características e conceitos-chave relacionados ao Docker:

1. **Containers:** Os *containers* são unidades isoladas que incluem o aplicativo, suas bibliotecas e dependências, tornando-os portáteis e independentes do ambiente em que são executados. Eles são mais leves que **máquinas virtuais** e compartilham o kernel do sistema operacional com o host, o que os torna mais eficientes.
2. **Imagens:** As imagens do Docker são modelos somente leitura que contêm instruções para criar um *container*. Você pode pensar nelas como um plano de construção para um *container*. Imagens podem ser compartilhadas e versionadas, o que facilita a colaboração e a implantação consistente de aplicativos.
3. **Dockerfile:** Um Dockerfile é um arquivo de configuração que define como uma imagem do Docker deve ser construída. Ele especifica quais bibliotecas e dependências devem ser instaladas no *container* e como o aplicativo deve ser configurado.
4. **Orquestração:** O Docker pode ser usado em conjunto com ferramentas de orquestração, como o Kubernetes, para gerenciar a implantação e o dimensionamento de aplicativos em *containers* em ambientes de produção. Essas ferramentas permitem que você automatize tarefas como balanceamento de carga, escalabilidade automática e recuperação de falhas.
5. **Docker Hub:** É um repositório público de imagens do Docker, onde você pode encontrar imagens prontas para uso em uma ampla variedade de aplicativos e serviços.
6. **Isolamento:** Os contêineres Docker fornecem um alto grau de isolamento entre aplicativos, o que significa que eles não interferem uns nos outros e não afetam o sistema host.
7. **Portabilidade:** Como os *containers* encapsulam todas as dependências necessárias, eles são altamente portáteis. Você pode criar um *container* em um ambiente de desenvolvimento e executá-lo em qualquer lugar que tenha o Docker instalado.

Em resumo, o Docker simplifica o processo de desenvolvimento, implantação e gerenciamento de aplicativos, tornando-os mais eficientes, escaláveis e consistentes em diferentes ambientes. É amplamente utilizado na indústria de desenvolvimento de software para criar e distribuir aplicativos de forma confiável e rápida.

## Passo a passo de como instalar o Docker no Raspberry Pi

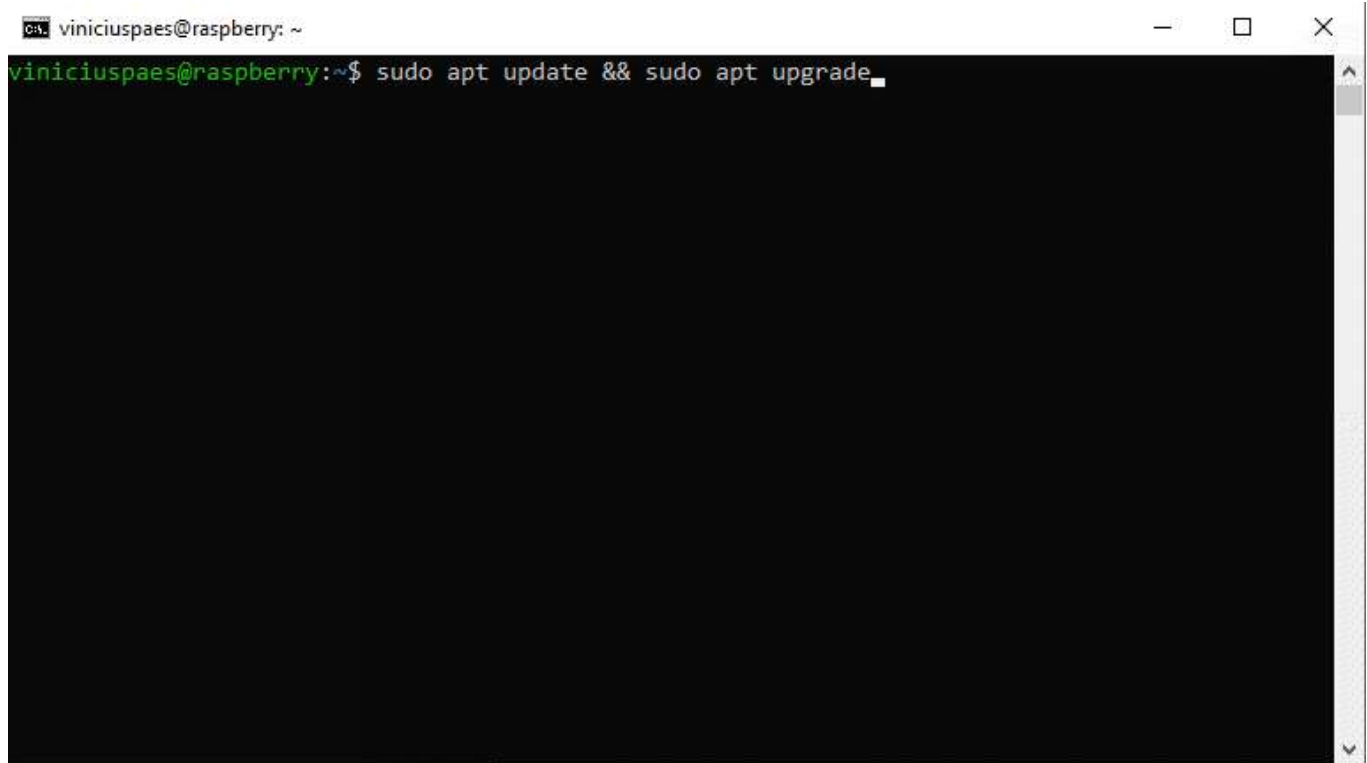
### Passo 1 – Conexão no raspberry

Caso esteja utilizando seu Raspberry de forma “headless”, basta conectar ao mesmo por SSH. É necessário saber o endereço IP do seu Raspberry, neste exemplo vamos adotar o usuário de login como “viniciuspaes” e o IP sendo **192.168.1.100**:  
`ssh viniciuspaes@192.168.1.100`

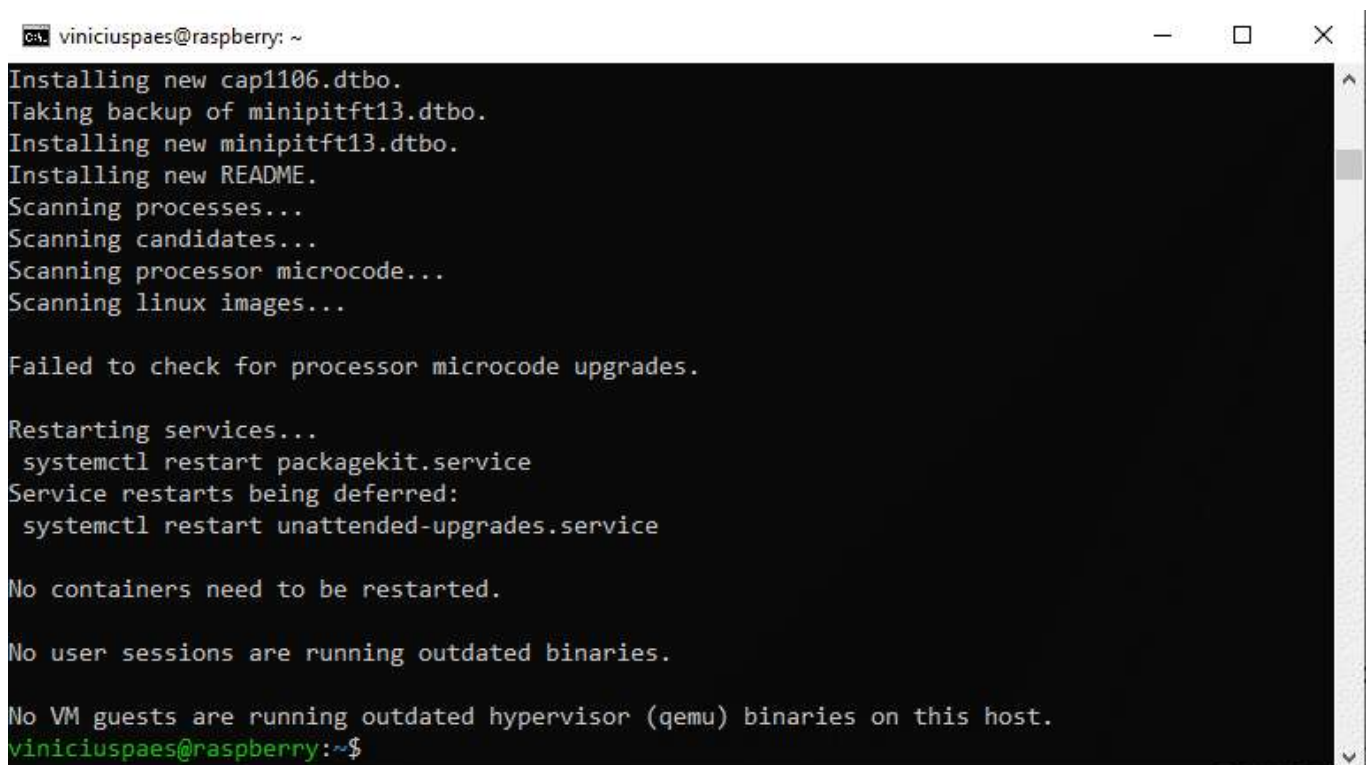
## Passo 2 – Atualização de software do Raspberry

Utilize o comando abaixo para realizar update da lista de pacotes que podem ser atualizados e em seguida realizar o upgrade destes:

```
sudo apt update && sudo apt upgrade
```



```
viniciuspaes@raspberrypi: ~$ sudo apt update && sudo apt upgrade
```



```
viniciuspaes@raspberrypi: ~$ sudo apt update && sudo apt upgrade
Installing new cap1106.dtbo.
Taking backup of minipitft13.dtbo.
Installing new minipitft13.dtbo.
Installing new README.
Scanning processes...
Scanning candidates...
Scanning processor microcode...
Scanning linux images...

Failed to check for processor microcode upgrades.

Restarting services...
systemctl restart packagekit.service
Service restarts being deferred:
systemctl restart unattended-upgrades.service

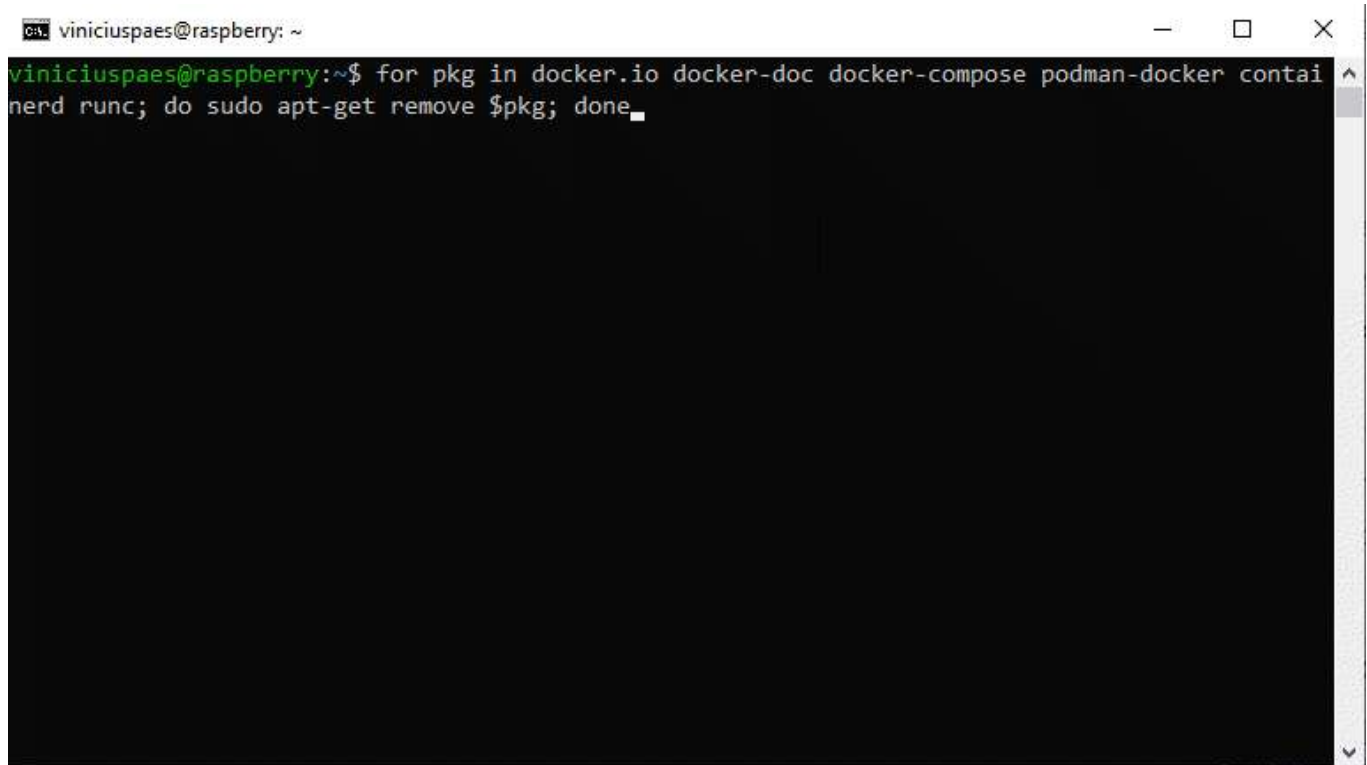
No containers need to be restarted.

No user sessions are running outdated binaries.

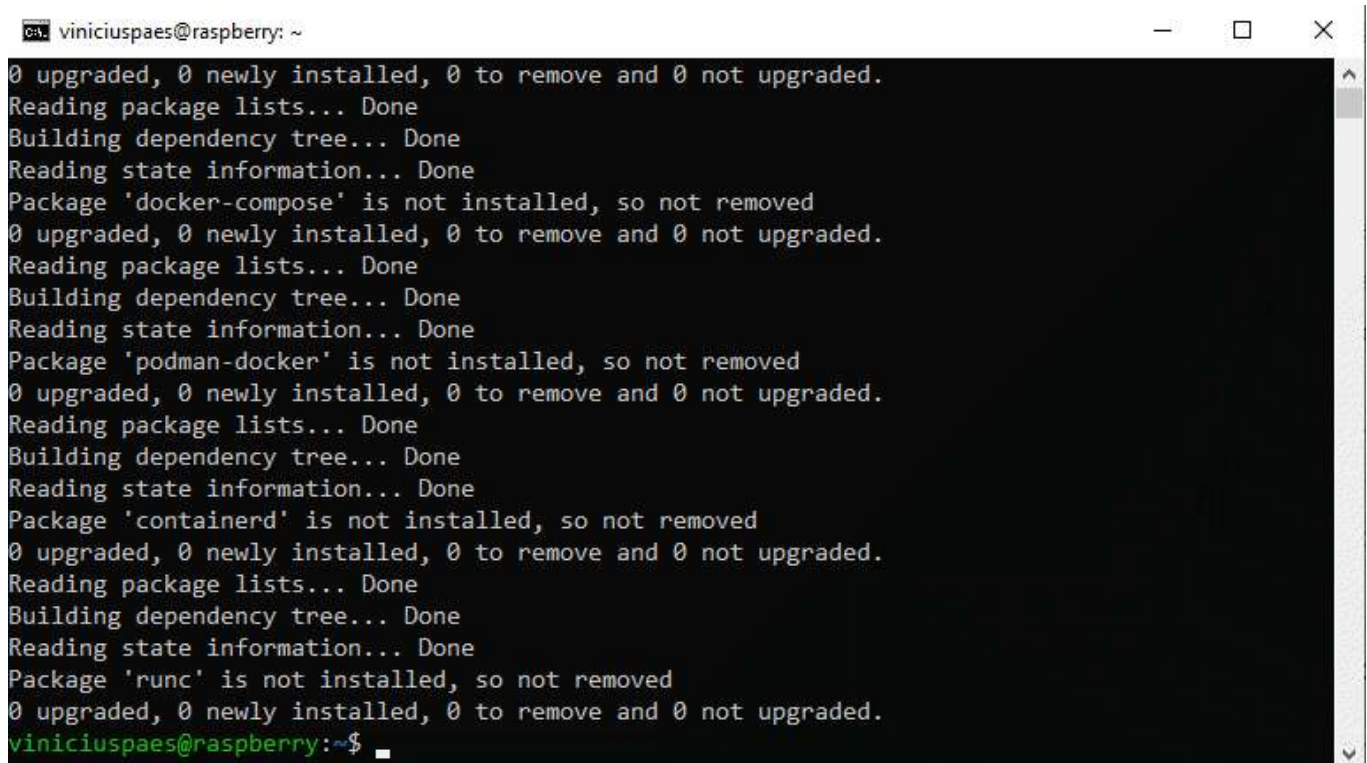
No VM guests are running outdated hypervisor (qemu) binaries on this host.
viniciuspaes@raspberrypi: ~$
```

### Passo 3 – Remover softwares que podem gerar conflitos

Caso tenha instalado o docker anteriormente pelo gerenciador de pacotes apt, é possível que cause conflitos de instalação. O recomendado é instalar diretamente pelo repositório oficial do Docker, para ter acesso as versões mais atuais. Com o comando abaixo, vamos procurar por diversos pacotes relacionados ao docker e solicitar a remoção, caso estejam instalados pelo apt: `for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done`



```
viniciuspaes@raspberrypi: ~  
viniciuspaes@raspberrypi:~$ for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```



```
viniciuspaes@raspberrypi: ~  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Package 'docker-compose' is not installed, so not removed  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Package 'podman-docker' is not installed, so not removed  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Package 'containerd' is not installed, so not removed  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Package 'runc' is not installed, so not removed  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
viniciuspaes@raspberrypi:~$
```

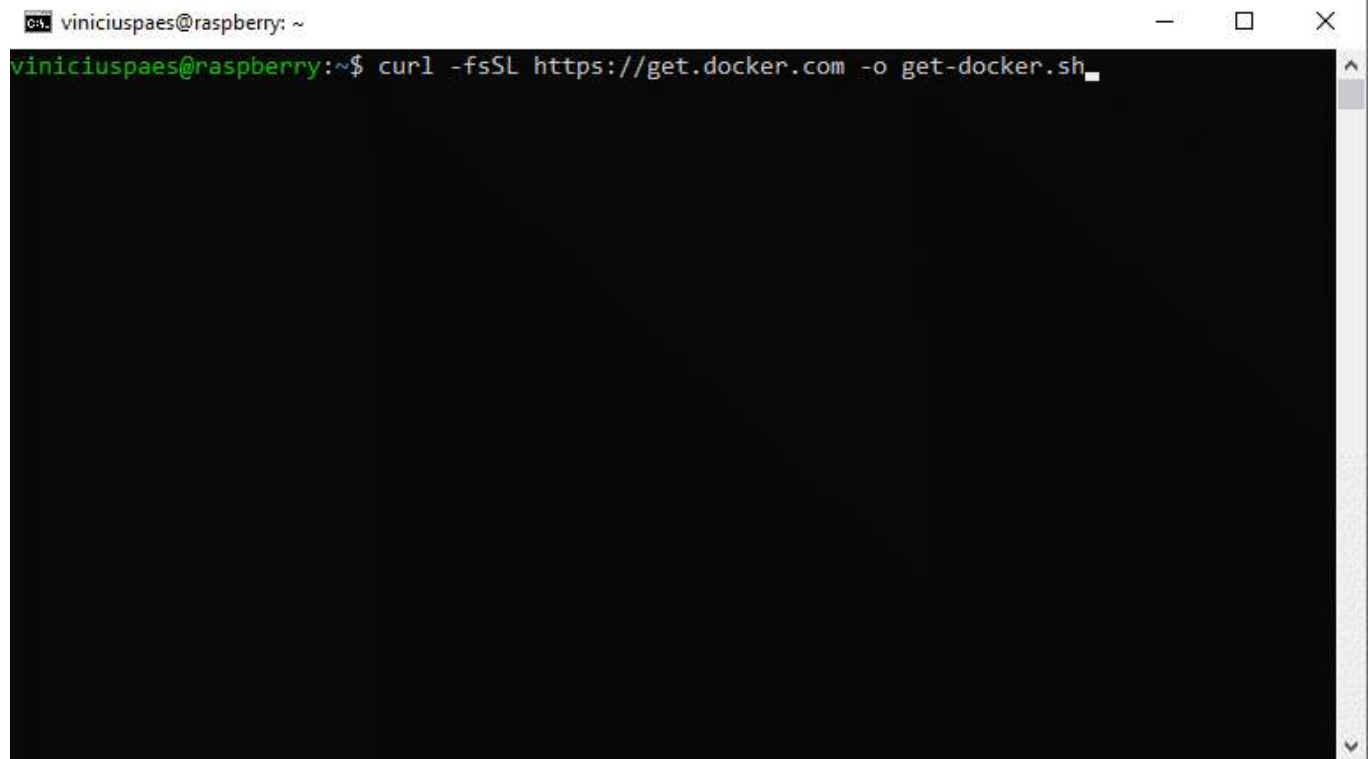
### Passo 4 – Download do script oficial de instalação do Docker e execução

Neste passo, vamos utilizar o Curl para realizar o download do script de instalação do Docker:

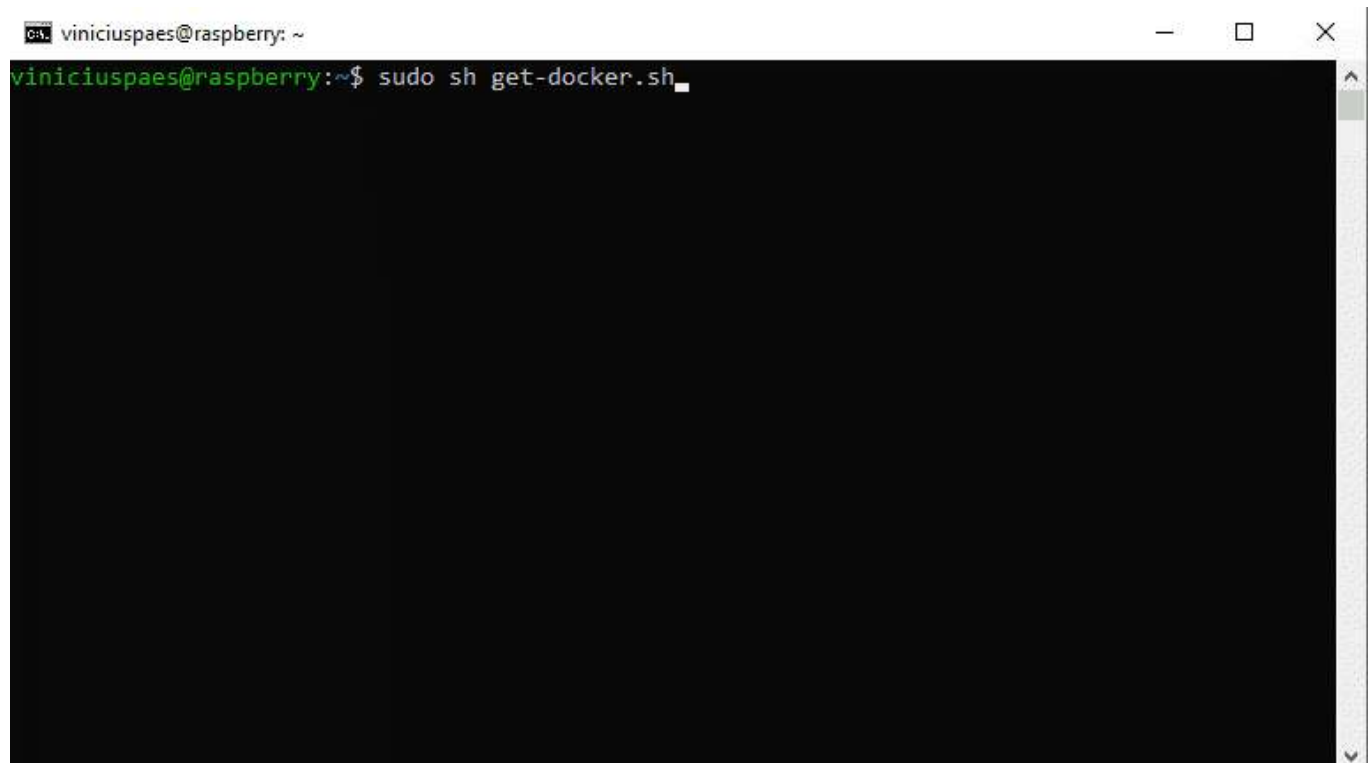
```
curl -fsSL https://get.docker.com -o get-docker.sh
```

Agora precisamos executar o script com o comando:

```
sudo sh get-docker.sh
```



```
viniciuspaes@raspberrypi: ~  
viniciuspaes@raspberrypi:~$ curl -fsSL https://get.docker.com -o get-docker.sh
```



```
viniciuspaes@raspberrypi: ~  
viniciuspaes@raspberrypi:~$ sudo sh get-docker.sh
```

```
viniciuspas@raspberry: ~  
GitCommit: de40ad0  
=====
```

To run Docker as a non-privileged user, consider setting up the Docker daemon in rootless mode for your user:

```
dockerd-rootless-setuptool.sh install
```

Visit <https://docs.docker.com/go/rootless/> to learn about rootless mode.

To run the Docker daemon as a fully privileged service, but granting non-root users access, refer to <https://docs.docker.com/go/daemon-access/>

WARNING: Access to the remote API on a privileged Docker daemon is equivalent to root access on the host. Refer to the 'Docker daemon attack surface' documentation for details: <https://docs.docker.com/go/attack-surface/>

```
=====
```

viniciuspas@raspberry:~\$

### Passo 5 – Adicionar usuário não privilegiado para ter permissão de execução do Docker.

Neste exemplo, vou utilizar o usuário **viniciuspas**, adicionando o mesmo no grupo do Docker, desta forma este usuário poderá executar o docker e gerenciar os containers.

```
sudo usermod -aG docker viniciuspas
```

```
viniciuspas@raspberry: ~  
viniciuspas@raspberry:~$ sudo usermod -aG docker viniciuspas
```

### Passo 6 – Reiniciar Conexão ao Raspberry

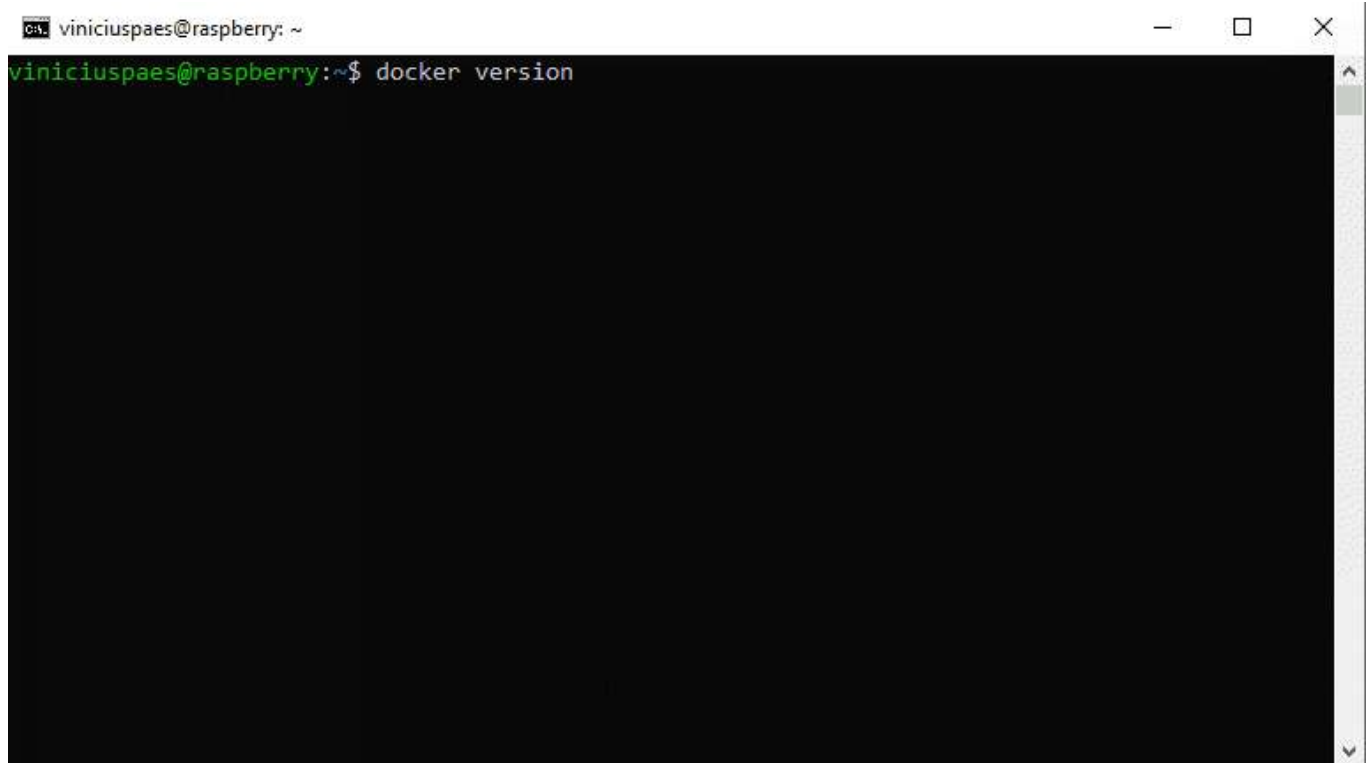
Uma etapa necessária é realizar o logout e posterior login no seu raspberry, para que as modificações de permissão do grupo do seu usuário tenham efeito. Caso tenha formatado o seu raspberry do zero para seguir este tutorial, provavelmente o sistema

operacional também deve pedir para que realize a reinicialização. Então é recomendado que utilize o comando para reiniciar o raspberry pi:  
`sudo reboot now`

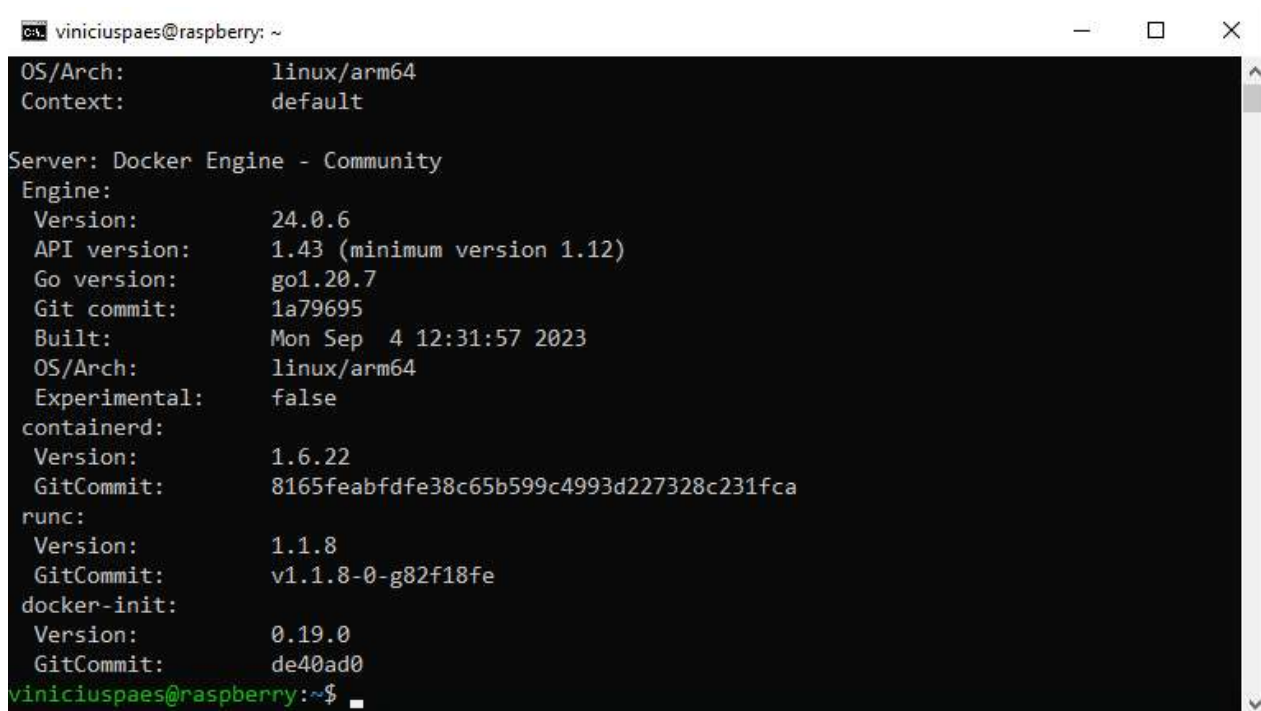
## Passo 7 – Nova conexão e verificar versão do Docker instalado

Realize novamente a conexão SSH ao seu raspberry após a reinicialização, caso esteja com um setup “headless”. Agora podemos ver a versão do Docker instalado com o comando:

`docker version`

A terminal window titled 'viniciuspaes@raspberrypi: ~' with standard window controls. The command 'docker version' is entered at the prompt. The output is not visible in this screenshot.

```
viniciuspaes@raspberrypi:~$ docker version
```

A terminal window titled 'viniciuspaes@raspberrypi: ~' with standard window controls. The command 'docker version' has been executed, and the output is displayed. The output shows details for the Docker Engine, containerd, and runc.

```
OS/Arch:      linux/arm64
Context:      default

Server: Docker Engine - Community
Engine:
  Version:    24.0.6
  API version: 1.43 (minimum version 1.12)
  Go version: go1.20.7
  Git commit: 1a79695
  Built:      Mon Sep  4 12:31:57 2023
  OS/Arch:    linux/arm64
  Experimental: false
containerd:
  Version:    1.6.22
  GitCommit:  8165feabfdfe38c65b599c4993d227328c231fca
runc:
  Version:    1.1.8
  GitCommit:  v1.1.8-0-g82f18fe
docker-init:
  Version:    0.19.0
  GitCommit:  de40ad0
viniciuspaes@raspberrypi:~$
```

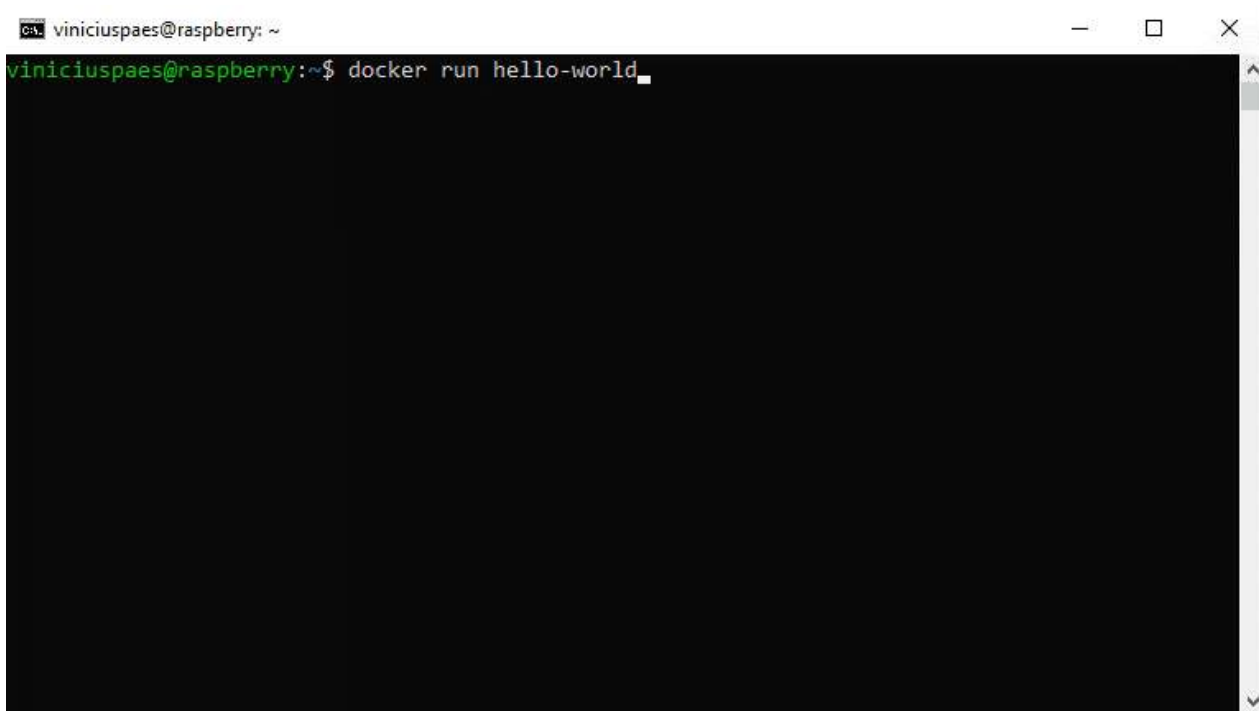
## Passo 8 – Testar instalação do docker com execução da imagem de hello-world



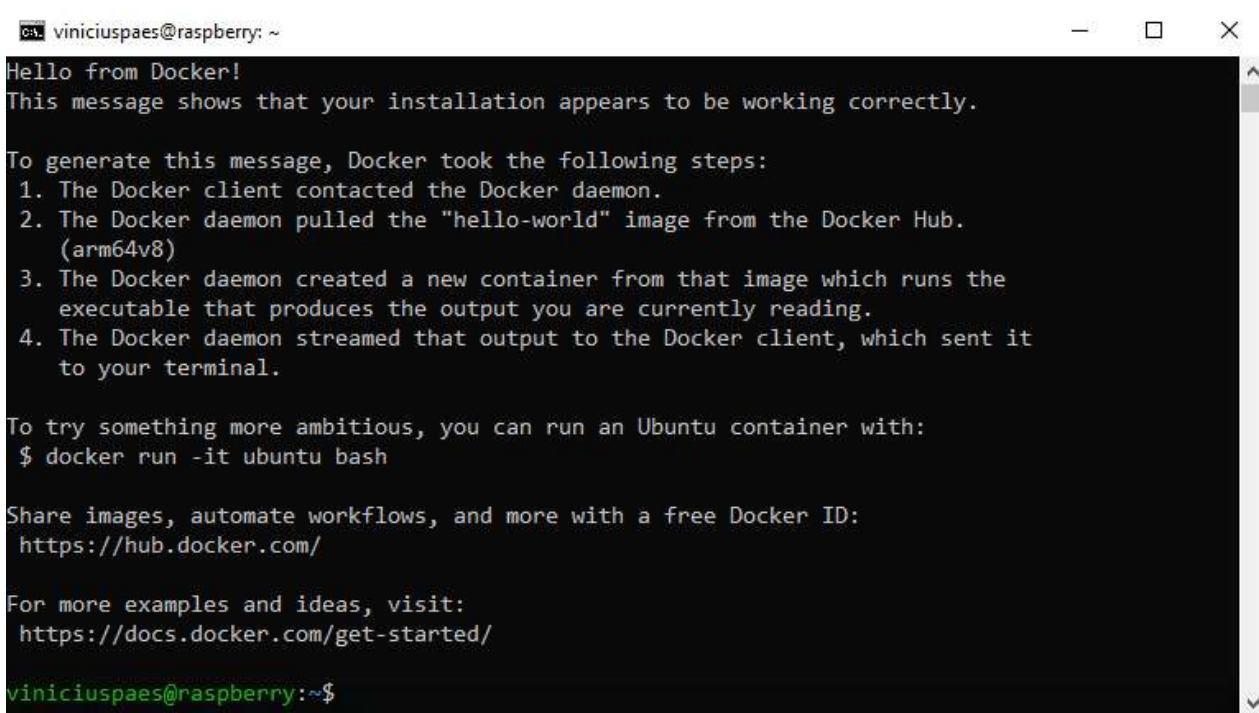
Para finalizar a instalação do Docker é interessante testar a mesma executando a imagem de hello-world. Vamos então utilizar o comando:

```
docker run hello-world
```

Tudo certo! Caso tenha sucesso na execução da imagem de hello-world o docker foi instalado e configurado corretamente. Caso tenha algum erro, interessante verificar se seu usuário tem privilégio de execução do Docker, igual configuramos no passo 5. Se ainda tiver dúvidas, você pode testar a execução do Docker com o comando “sudo”, para ter certeza que não é um problema de privilégio de execução.

A terminal window titled 'viniciuspaes@raspberrypi: ~' with standard window controls. The command 'docker run hello-world' has been entered at the prompt, and the cursor is at the end of the line.

```
viniciuspaes@raspberrypi:~$ docker run hello-world
```

A terminal window titled 'viniciuspaes@raspberrypi: ~' with standard window controls. It shows the output of the 'docker run hello-world' command. The output includes a greeting, a confirmation message, a list of steps Docker took, instructions for running an Ubuntu container, and links to Docker documentation and the Docker Hub.

```
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.
   (arm64v8)
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/

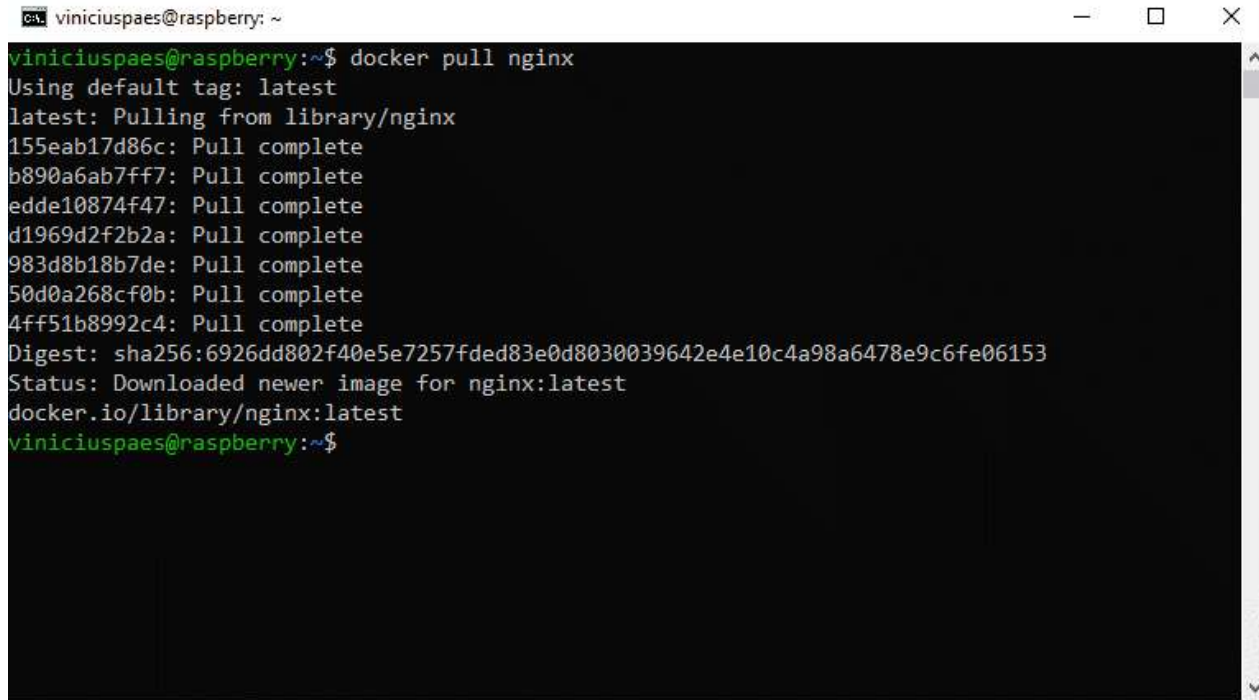
viniciuspaes@raspberrypi:~$
```

## Lista de comandos úteis para utilização do Docker

### Fazer o download de uma imagem de um container

Você pode buscar por imagens prontas diretamente no site do docker hub. No exemplo do comando abaixo, vamos realizar somente o download da imagem oficial do nginx:

```
docker pull nginx
```

A terminal window titled 'viniciuspases@raspberrypi: ~' showing the command 'docker pull nginx' and its output. The output indicates that the latest version of the nginx image is being pulled from the Docker library. It lists several layers being pulled, each marked as 'Pull complete'. The digest is shown as 'sha256:6926dd802f40e5e7257fded83e0d8030039642e4e10c4a98a6478e9c6fe06153'. The status is 'Downloaded newer image for nginx:latest'. The terminal ends with the prompt 'viniciuspases@raspberrypi:~\$'.

## Criar um container a partir de uma imagem

O comando docker run permite iniciar um container a partir de uma imagem. No exemplo do comando abaixo, vamos criar um container que contém a instalação do servidor de páginas nginx:

```
docker run nginx
```

Ao executar o comando, caso o docker não encontre a imagem localmente, ele irá buscar a mesma diretamente no docker hub e irá criar o container. Mas caso já tenha feito o download da imagem com o comando docker pull, o container é criado mais rapidamente, pois não precisa realizar o download da mesma.

O problema do comando acima, é que o terminal fica bloqueado na execução. Como opção, podemos passar por referência o prefixo -d, para que o container fique em funcionamento em segundo plano.

```
docker run -d nginx
```

O problema do comando acima é que não mapeamos a porta de execução do container. Como o nginx pode funcionar como um servidor de páginas, para que seja possível testar o funcionamento, é interessante que a porta http seja mapeada para o host.

Podemos então mapear a porta do host para o container com o prefixo -p [portaHost]:[portaContainer]. Neste exemplo, vamos mapear a porta 8080 do host para ser redirecionada para a porta 80 do Container:

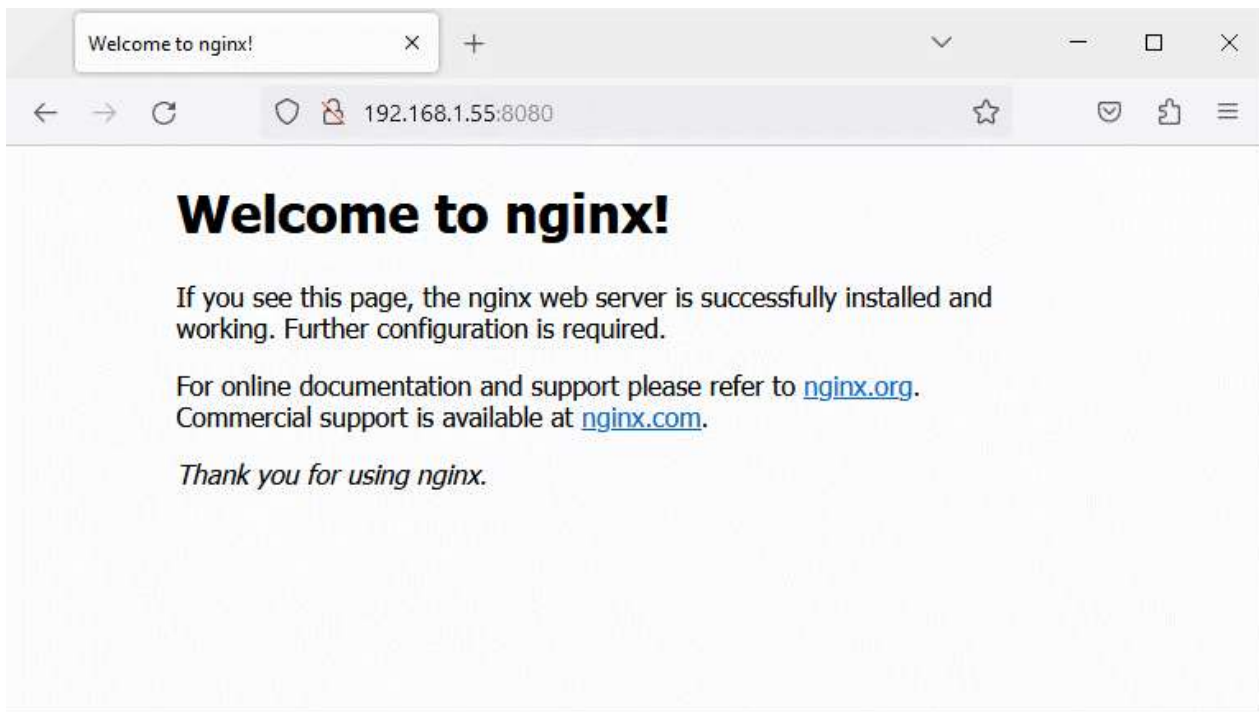
```
docker run -d -p 8080:80 nginx
```



```
viniciuspaes@raspberrypi: ~  
viniciuspaes@raspberrypi:~$ docker run nginx  
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration  
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/  
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh  
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf  
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf  
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh  
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh  
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh  
/docker-entrypoint.sh: Configuration complete; ready for start up  
2023/09/08 19:35:24 [notice] 1#1: using the "epoll" event method  
2023/09/08 19:35:24 [notice] 1#1: nginx/1.25.2  
2023/09/08 19:35:24 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)  
2023/09/08 19:35:24 [notice] 1#1: OS: Linux 5.15.0-1036-raspi  
2023/09/08 19:35:24 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576  
2023/09/08 19:35:24 [notice] 1#1: start worker processes  
2023/09/08 19:35:24 [notice] 1#1: start worker process 30  
2023/09/08 19:35:24 [notice] 1#1: start worker process 31  
2023/09/08 19:35:24 [notice] 1#1: start worker process 32  
2023/09/08 19:35:24 [notice] 1#1: start worker process 33
```

```
viniciuspaes@raspberrypi: ~  
viniciuspaes@raspberrypi:~$ docker run -d nginx  
c46cd9b051a92e0ac0a9bad47abfbd61ee6b30a391ee7d83b0cdb04e8af98fa7  
viniciuspaes@raspberrypi:~$
```

```
viniciuspaes@raspberrypi: ~$ docker run -d -p 8080:80 nginx
```



## Verificar os containers em funcionamento

Para verificar quais container estão ativos, podemos utilizar o comando:  
`docker ps`

Neste exemplo, verificamos 2 containers sendo executados. Ambos os containers utilizam a imagem oficial do nginx:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
58a23436fc0e	nginx	"/docker-entrypoint...."	4 minutes ago	Up 4 minutes	0.0.0.0:8080->80/tcp, :::8080->80/tcp	festive_cori
c46cd9b051a9	nginx	"/docker-entrypoint...."	7 minutes ago	Up 7 minutes	80/tcp	pensive_carson

```
viniciuspases@raspberrypi: ~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
58a23436fc0e	nginx	"/docker-entrypoint..."	4 minutes ago	Up 4 minutes	0.0.0.0:8080
c46cd9b051a9	nginx	"/docker-entrypoint..."	7 minutes ago	Up 7 minutes	80/tcp

```
viniciuspases@raspberrypi:~$
```

## Parar e remover um container

Vamos remover o container da imagem do nginx que esquecemos de mapear a porta externa do host. Olhando a saída do Docker ps, temos o ID do container que queremos remover. Neste caso o ID é:

c46cd9b051a9

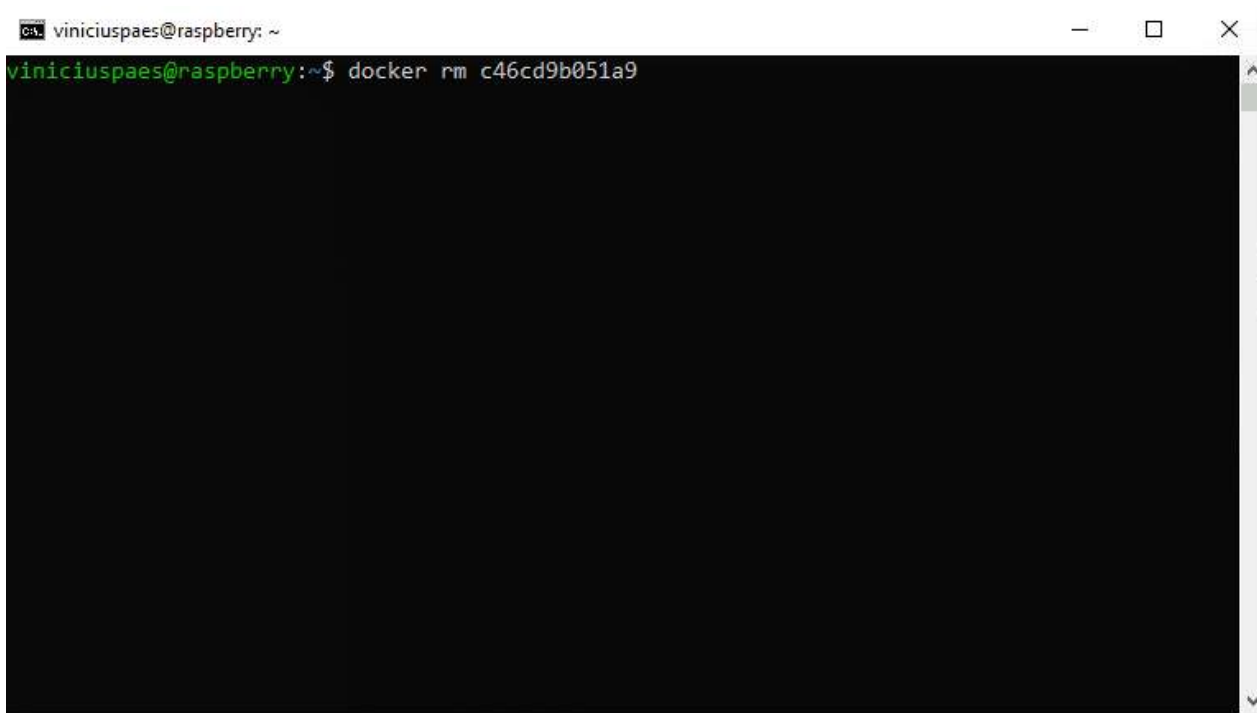
Não é possível remover um container que está em funcionamento, então primeiro precisamos parar o container com o comando:

```
docker stop c46cd9b051a9
```

Em seguida podemos prosseguir com a remoção do container:

```
docker rm c46cd9b051a9
```

```
viniciuspases@raspberrypi: ~$ docker stop c46cd9b051a9
```

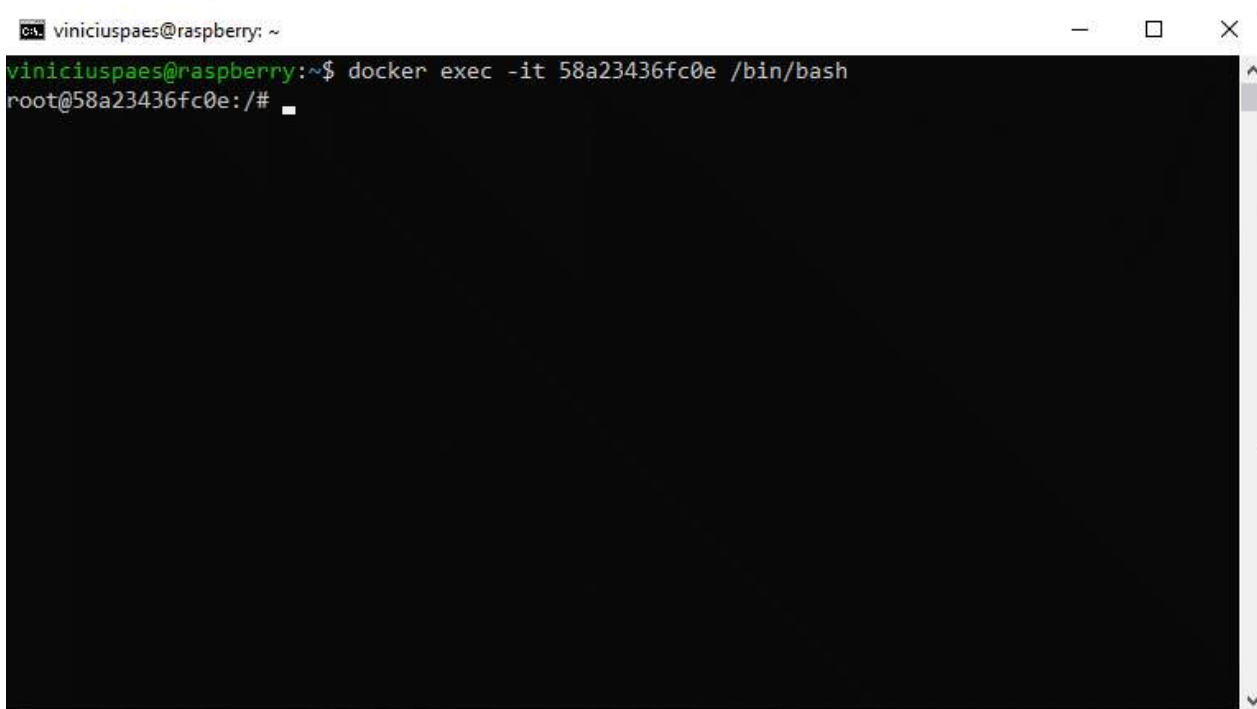


```
viniciuspaes@raspberrypi: ~$ docker rm c46cd9b051a9
```

## Entrar dentro de um container e executar comandos

Caso precise entrar dentro do container que possui ID “58a23436fc0e” para executar algum comando:  
`docker exec -it 58a23436fc0e /bin/bash`

No comando acima, executamos o bash dentro do container. Se o comando acima não funcionar, é possível que a imagem que esteja utilizando não ter o bash. Para sair do container, lembre-se de digitar “exit”.

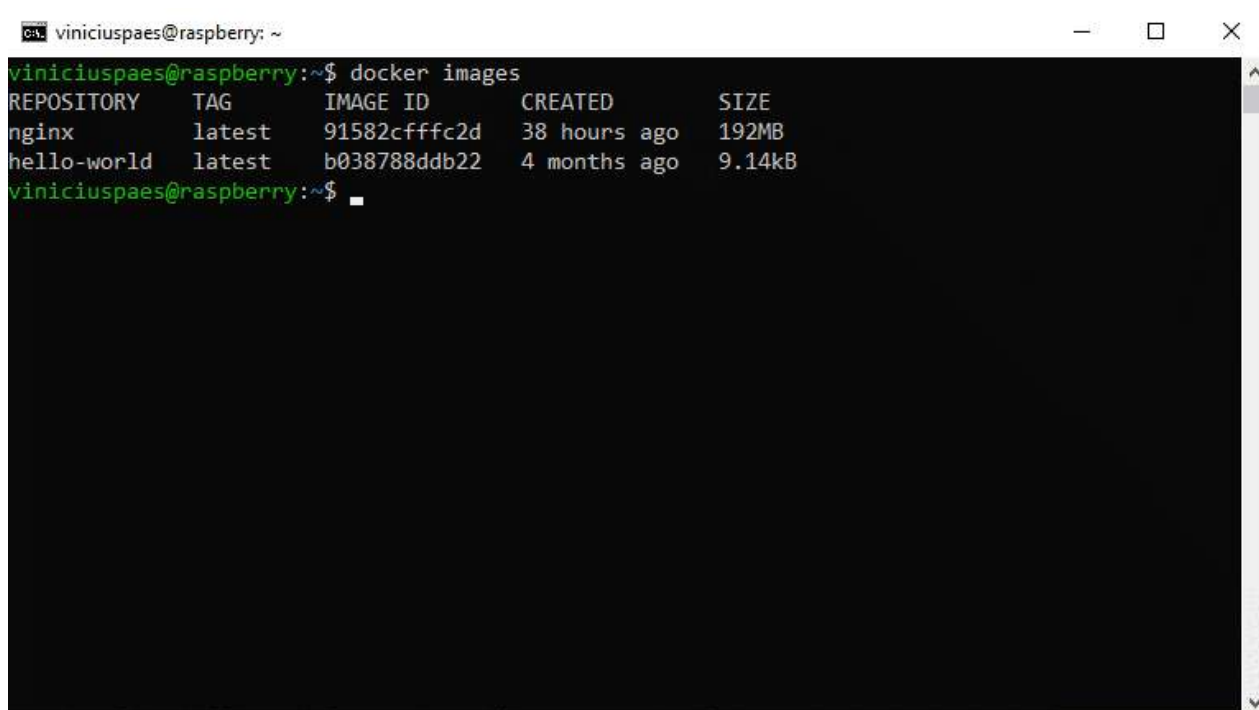


```
viniciuspaes@raspberrypi: ~$ docker exec -it 58a23436fc0e /bin/bash
root@58a23436fc0e:/#
```

## Listar as imagens de containers que já fez download

Caso queira verificar quais imagens de containers já fez download e estão disponíveis em seu computador, utilize o comando:  
`docker images`

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	91582cffffc2d	38 hours ago	192MB
hello-world	latest	b038788ddb22	4 months ago	9.14kB

A terminal window titled 'viniciuspaes@raspberrypi: ~' with standard window controls. It shows the command 'docker images' being executed. The output is a table with columns: REPOSITORY, TAG, IMAGE ID, CREATED, and SIZE. Two images are listed: 'nginx:latest' with ID '91582cffffc2d' created '38 hours ago' and size '192MB', and 'hello-world:latest' with ID 'b038788ddb22' created '4 months ago' and size '9.14kB'. The prompt 'viniciuspaes@raspberrypi:~\$' is shown at the bottom.

```
viniciuspaes@raspberrypi:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
nginx         latest    91582cffffc2d  38 hours ago   192MB
hello-world    latest    b038788ddb22   4 months ago   9.14kB
viniciuspaes@raspberrypi:~$
```

## Considerações Finais

A instalação do Docker no Raspberry Pi é bem simples e pode ser realizada com poucos passos. É um excelente setup para começar a aprender a utilizar o docker e gerenciar aplicativos ou projetos pessoais na sua rede local. Os principais comandos de utilização do Docker também foram listados

Caso tenha dúvidas sobre a instalação ou execução do Docker, basta escrever aí nos comentários!