Tutorial: implantação de Servidor Web NGINX em container Docker no VirtualBox com Linux

Daiane Maria dos Santos Ribeiro[1], Nereu Vítor Pereira Lima[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano – IFSertãoPE Campus Salgueiro

BR 232, km 508, sentido Recife – 56.000-000 – Salgueiro – PE – Brasil

{daiane.ribeiro@ifsertao-pe.edu.br, nereu.vitor@aluno.ifsertao-pe.edu.br

1. Introdução

Um servidor web é um sistema de software e hardware responsável por receber, interpretar e responder a requisições feitas por clientes via protocolo de transferência de hipertexto (HTTP, Hypertext Transfer Protocol) ou protocolo de transferência de hipertexto seguro (HTTPS, Hypertext Transfer Protocol Secure), geralmente entregando páginas web e serviços digitais (W3C, 2023). Existem inúmeros servidores web disponíveis para uso, mas o NGINX é um dos servidores mais adequado para ser instalado em um computador de uso pessoal, pois ele apresenta algumas vantagens em relação a outros servidores web, como por exemplo, excelente aproveitamento de CPU e RAM, ótimo para lidar com múltiplas conexões simultâneas, muito seguro e eficiente, ideal para ambientes de produção com WordPress, Laravel, APIs REST etc, alto desempenho mesmo em máquinas mais modestas, entre outras.

Para ilustrar como um servidor web funciona, nós iremos apresentar nesse trabalho um tutorial para criação e utilização de um container com o NGNIX, que será utilizado a partir do Docker na máquina VirtualBox com o sistema operacional Linux. Ademais, apresentaremos, também, como disponibilizar a atividade desenvolvida no GitHub.

O link de acesso do projeto desenvolvido nesse trabalho é: https://github.com/daianesr35/servidor-nginx-docker.

Antes de apresentar o tutorial, faremos nas subseções a seguir, uma breve explicação sobre as tecnologias usadas nessa atividade.

1.1. Container com NGNIX

Um container é uma unidade executável de software que empacota o código da aplicação juntamente com as suas bibliotecas e dependências para o seu funcionamento. Dessa forma, ele permite que o código seja executado em qualquer ambiente de computação, seja ele Desktop, TI tradicional ou em infraestrutura de nuvem (SUSNJARA, SMALLEY, 2025).

Os contêineres funcionam por meio da virtualização do Sistema Operacional (SO), no qual os recursos do kernel do SO podem ser utilizados para isolar processos e controlar

a quantidade de memória, CPU e disco que esses processos podem utilizar. Oposto ao que as máquinas virtuais (MVs) realizam, a virtualização do sistema operacional (normalmente o Linux) realizada pelos contêineres, garante que somente a aplicação, suas bibliotecas, arquivos de configuração e dependências estejam presentes no interior do mesmo. Por conta da ausência do sistema operacional hospedeiro, tecnologia utilizada pelas máquinas virtuais, os contêineres tendem a serem mais leves, e, portanto, mais rápidas em comparação às MVs (SUSNJARA, SMALLEY, 2025). Ademais, para tornar o processo de conteinerização mais rápido e fácil é importante usar o Docker.

1.2. Docker

O Docker é uma plataforma de código aberto que permite aos desenvolvedores construir, implementar, executar, realizar atualizações e o gerenciar contêineres (SUSNJARA, SMALLEY, 2025).

Utilizar o Docker é muito importante, pois além dele tornar o processo de conteinerização mais rápido e fácil, também desempenha um papel crucial no desenvolvimento moderno de software, mais especificamente na área de microsserviços (SUSNJARA, SMALLEY, 2025).

A utilização do Docker para este trabalho foi realizada para facilitar e tornar ágil a criação, implementação e a gestão de um container com NGNIX e as suas dependências, para que o mesmo atue como um servidor web, independente do ambiente que esteja sendo utilizado. Além disso, decidimos usar o GitHub para disponibilizar o container com NGNIX criado nesse trabalho.

1.3. GitHub

O Github é uma plataforma baseada em nuvem onde é possível, armazenar, compartilhar e trabalhar com outras pessoas para escrever códigos. O uso do GitHub se dá por meio de repositórios remotos que podem ser públicos ou privados e podem receber solicitações de mudanças no projeto através de pull requests, onde o proprietário decide se aceitará ou não. Recebe, também, dúvidas sobre o projeto que podem ser esclarecidas através de issues criadas pelos os usuários (GITHUB, 2025).

Dessa maneira, escolhemos o GitHub para disponibilizar o container com NGNIX criado nesse trabalho. Além do professor da disciplina ter a possibilidade de acompanhar o desenvolvimento da atividade.

2. Passo a passo da criação e uso do container com NGNIX no VirtualBox

Nesta seção iremos apresentar as etapas que realizamos desde a inicialização da Máquina Virtual (MV), do uso do servidor NGNIX dentro do container no VirtualBox, até a publicação do container no GitHub.

2.1. Etapa 1 - Instalando o Docker na MV com Linux

1. O VirtualBox e Ubuntu Server já estava instalado no notebook. Então, nós inicializamos a MV o logamos no Ubuntu.

2. Em seguida, abrimos o terminal e executamos os comandos:

- `sudo apt update` – Esse comando atualizou a lista de pacotes disponíveis nos repositórios do Ubuntu. Isso é importante, porque garante que o sistema saiba quais são as versões mais recentes dos programas antes de instalar qualquer coisa, como mostra a figura 1.

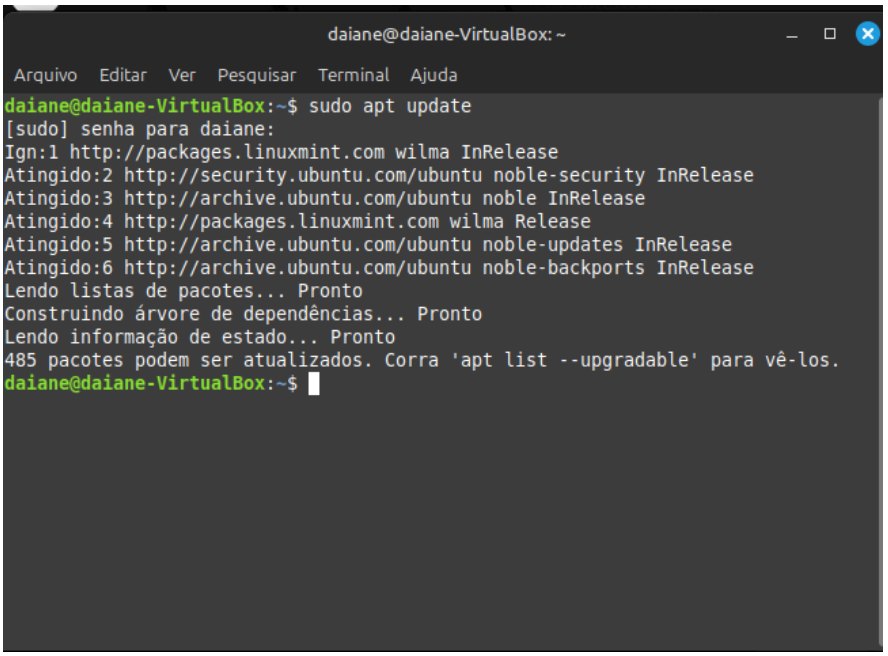

Figura 1. Atualização de listas de pacotes no Ubuntu.

- `sudo apt install -y docker.io` – Esse commando instalou o pacote `docker.io`, que é o Docker disponível nos repositórios do Ubuntu. Assim, o Docker foi instalado na MV, como ilustra a figura 2, permitindo a criação e gerenciamento de containers.

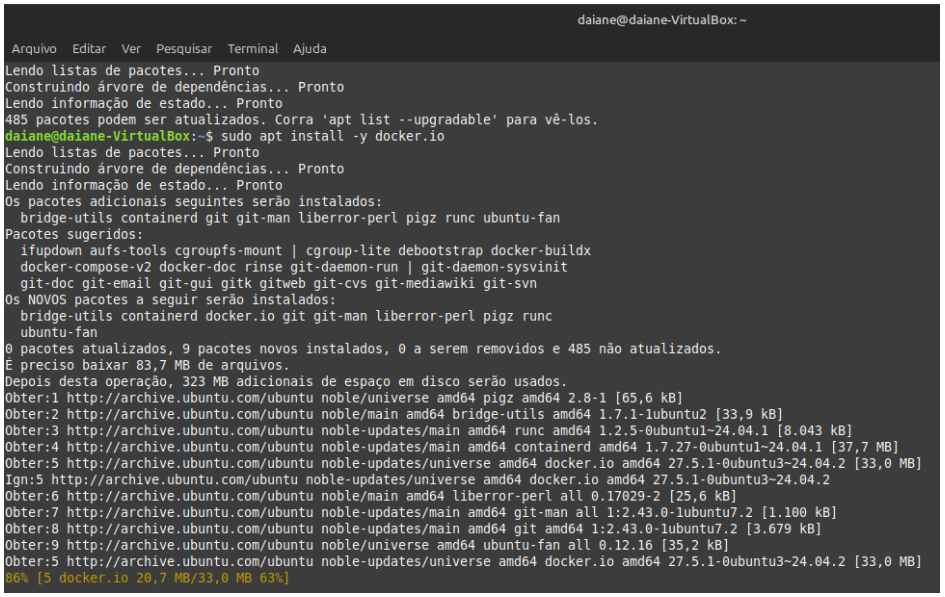

Figura 2. Instalação do docker.oi.

- `sudo systemctl start docker` – Esse comando iniciou o serviço do Docker imediatamente.
- `sudo systemctl enable docker` – Esse commando configurou o Docker para iniciar automaticamente toda vez que o Ubuntu for ligado.
- `sudo usermod -aG docker $USER` – Esse comando adicionou o usuário atual ao grupo de permissões chamado docker. Esses últimos três commandos podem ser visualizados na figura 3.

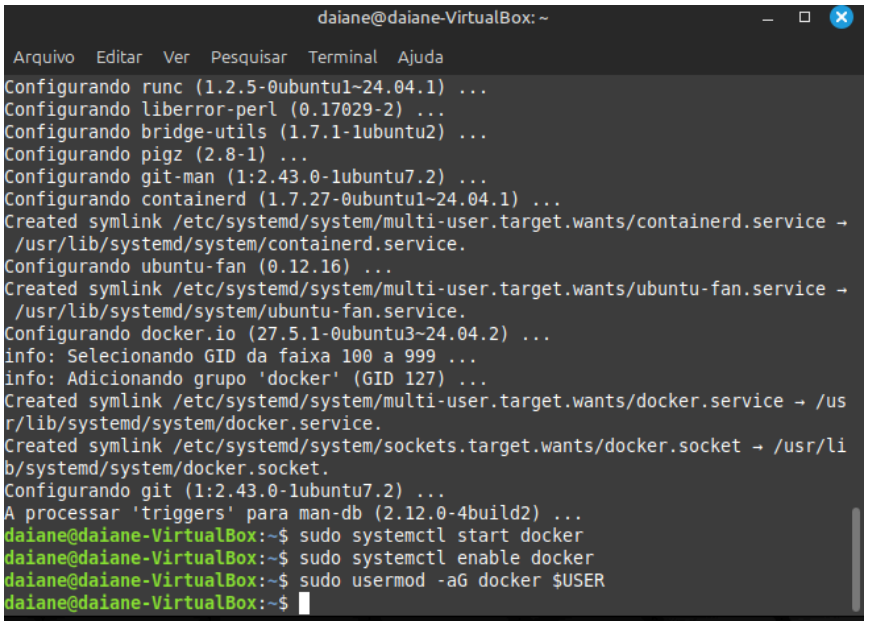

Figura 3. Iniciando, configurando e adicionando usuário no docker.

- `sudo reboot` – Esse comando foi aplicado para executar as permissões corretamente. Para isso, ele reinicia o sistema.

Depois que o Sistema foi reiniciado, o passo a passo descrito nessa etapa 1 foi corretamente implementado. Então, foi possível criar um container com NGNIX para ser usado a partir do docker dentro do VirtualBox. O passo a passo para a criação do container com NGINX está descrito na etapa 2.

2.2. Etapa 2 - Criando um container com o NGNIX

1. No terminal do VirtualBox, executamos os comandos:

- `docker run -d -p 80:80 --name meu-nginx nginx` – Esse commando criou e iniciou um container do Docker com o servidor web NGINX rodando na porta 80, como mostra a figura 4. O nome personalizado do container criado é: `meu-nginx`.
- `docker ps` – Esse comando mostrou todos os containers do Docker que estavam em execução (ver figura 4).

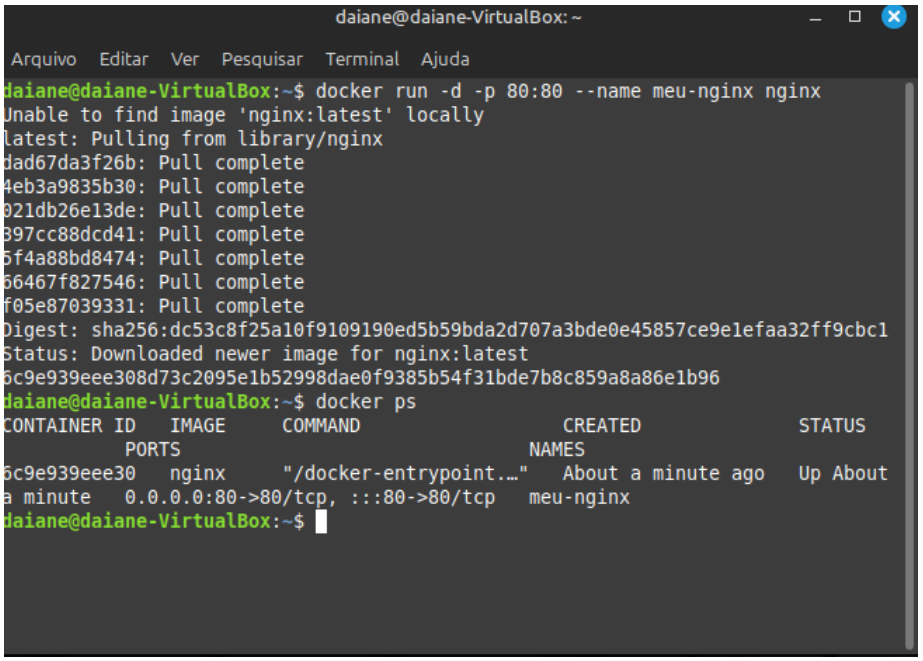

Figura 4. Criação do container com o Nginx e verificação dos containers em execução, respectivamente.

Para acessar o IP da MV através do navegador de qualquer dispositivo, nós mudamos a configuração de Rede na MV, conforme descrito na etapa 3.

2.3. Etapa 3 – Alterando a configuração de rede no VirtualBox

1. Desligamos a MV no VirtualBox.

2. No VirtualBox, selecionamos a Maquina Virtual Linux2025 e clicamos em Configurações.

3. Em seguida, selecionamos Rede e Adaptador1.

4. Na opção “Contectado a”, escolhemos `Placa em modo bridge`.

5. Em “Nome”, selecionamos a placa de rede Wi-Fi do notebook.

6. Clicamos em Ok e iniciamos a MV novamente.

Todos esses passos podem ser visualizados na figura 5.

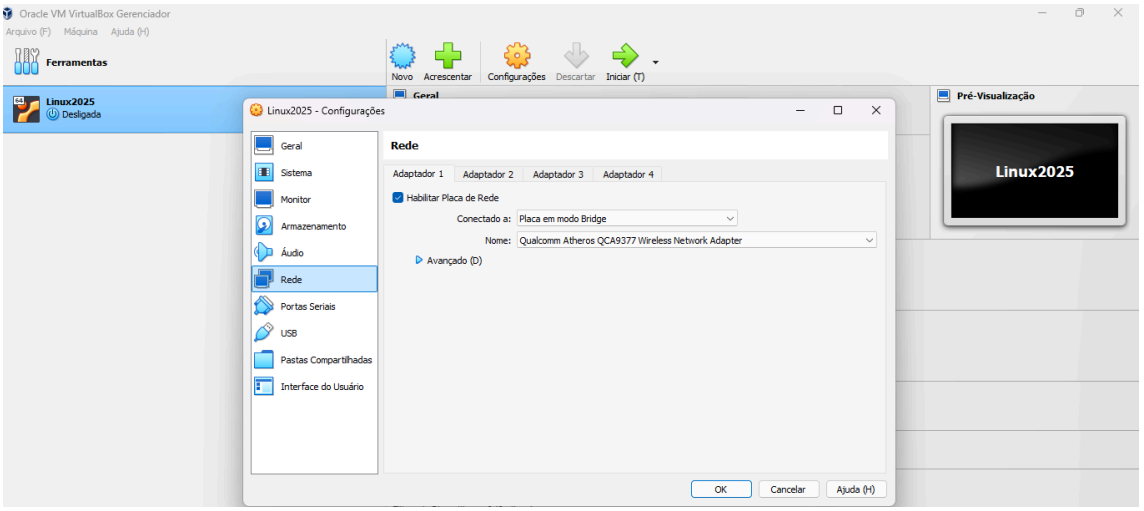

Figura 5. Configuração de rede no VirtualBox.

2.4. Etapa 4 – Descobrindo o IP da MV e acessando o NGNIX pelo navegador de qualquer dispositivo.

1. Com a MV inicializada, executamos no terminal o comando: `ip a` - Esse comando nos apresentou o IP da MV. O IP da MV está destacado na figura 6.

```
daiane@daiane-VirtualBox:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:12:90:58 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.4/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
       valid_lft 7114sec preferred_lft 7114sec
    inet6 2804:29b8:5071:2896:bee1:47c9:9d93:1ef2/64 scope global temporary dynamic
       valid_lft 259198sec preferred_lft 86000sec
    inet6 2804:29b8:5071:2896:29da:5769:c03d:bf90/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 259198sec preferred_lft 172798sec
    inet6 fe80::ed63:ee2:9783:e3ff/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:60:1c:26:69 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:60ff:fe1c:2669/64 scope link
       valid_lft forever preferred_lft forever
5: veth917dfd9@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 76:33:67:e6:87:76 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::7433:67ff:fee6:8776/64 scope link
       valid_lft forever preferred_lft forever
```

Figura 6. Localizando o IP da MV.

A **enp0s3** é a interface de rede NAT padrão do VirtualBox. O inet, que está abaixo da interface **enp0s3**, mostra o endereço IPv4 da MV: 192.168.1.4.

2. Em seguida, digitamos http:// 192.168.1.4, no navegador do notebook e visualizamos a imagem apresentada na figura 7.

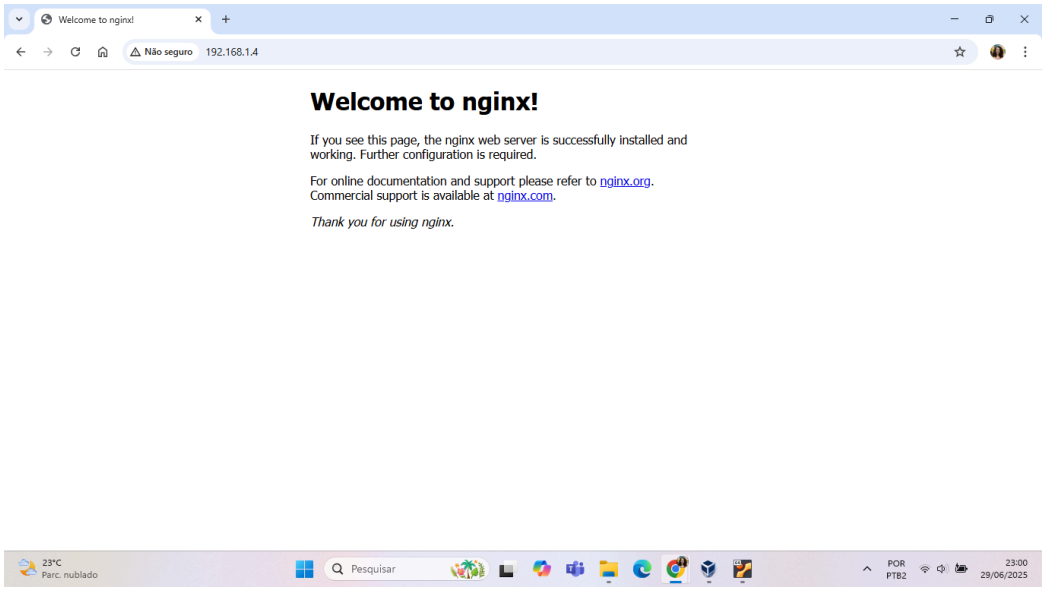

Figura 7. Página padrão no NGNIX.

A figura 7 mostra a página padrão do NGNIX, indicando que o container com o NGNIX foi instalado com sucesso na MV.

Em seguida, para personalizar o código da página index.html, tivemos que acessar o container do NGNIX. Essa personalização está descrita na etapa 5.

2.5. Etapa 5 – Personalizando a página index.html e criando novas pastas no NGNIX.

1. No terminal da MV, executamos os comandos:

- `docker exec -it meu-nginx bash` – Esse comando abriu um terminal dentro do container chamado `meu-nginx`.
- `cd /usr/share/nginx/html` – Dentro do container, esse comando localizou a página onde estava o index.html.
- `ls` – Esse comando listou os arquivos da pasta.
- `nano index.html` – Com esse comando nós personalisamos o arquivo index.html com o editor nano.

A execução de todos os comando, listados para essa etapa 5, estão ilustrados nas figuras 8 e 9.

Para salvar o arquivo index.html pressionamos `Ctrl + O` no teclado e em seguida pressionamos a tecla `Enter`. Para sair do referido arquivo pressionamos `Ctrl + X`.

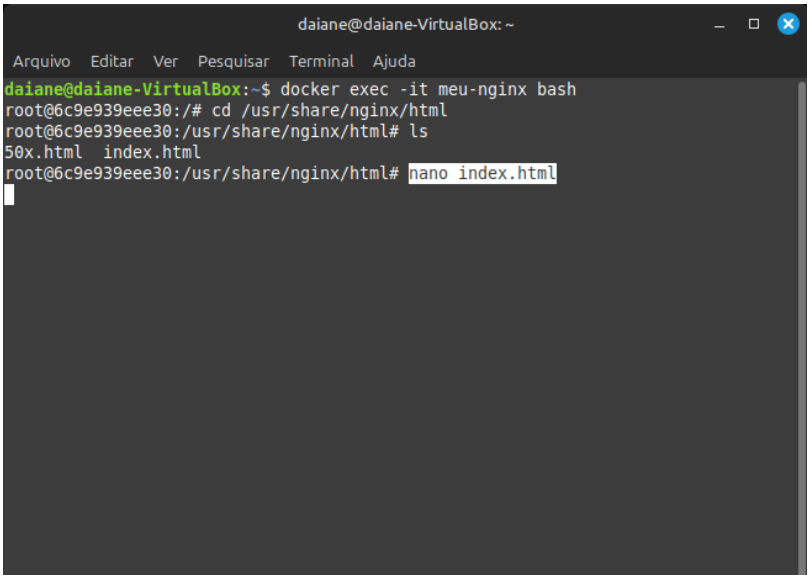

Figura 8. Acessando o arquivo index.html.

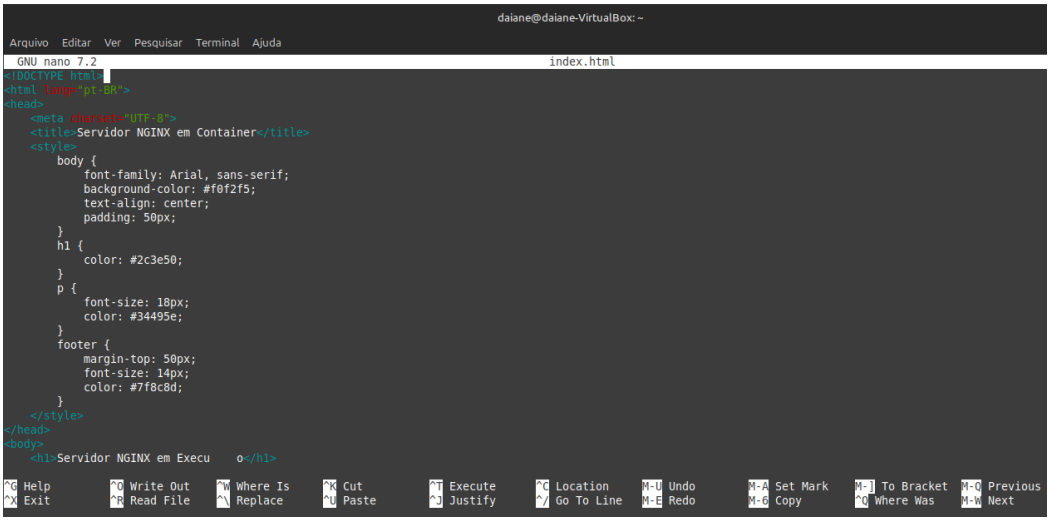

Figura 9. Personalizando no nano o arquivo index.html.

2. Para mostrar que é possível acessar o NGNIX através do navegador de qualquer dispositivo, depois da atualização do arquivo index.html, nós acessamos o NGNIX pelo navegador de um celular, a página atualizada pode ser verificada pela figura 10.

Figura 10. Arquivo index.html personalizado.

3. Em seguida, criamos as pastas daiane e nereu dentro do container, no mesmo diretório onde está o index.html principal, que é servido pelo NGNIX. Em cada uma das pastas (daiane, nereu) foram criados arquivos index.html, que podem ser acessados através do index.html principal. Para criarmos essas duas pastas, com os arquivos index.html, executamos no terminal da MV os comandos:

- `docker exec -it meu-nginx bash` - Esse comando abriu um terminal dentro do container chamado `meu-nginx`.
- `cd /usr/share/nginx/html` – Esse comando acessou o diretório HTML do Nginx.
- `mkdir daiane nereu` – Esse comando criou as pastas daiane e nereu.
- `nano daiane/index.html` – Por meio desse comando criamos e personalisamos a pasta e o arquivo `daiane/index.html`. Em seguida, salvamos o arquivo com `Ctrl + O` e `Enter` e saímos do arquivo com `Ctrl + X`.
- `nano nereu/index.html` – Com esse comando criamos e personalisamos a pasta e o arquivo `nereu/index.html`. Em seguida, salvamos o arquivo com `Ctrl + O` e `Enter` e saímos do arquivo com `Ctrl + X`.

A execução de todos os comandos descritos nesse passo está ilustrada nas figuras 11, 12 e 13.

Depois, atualizamos o index.html principal com os links do index.html da pasta daiane e nereu.

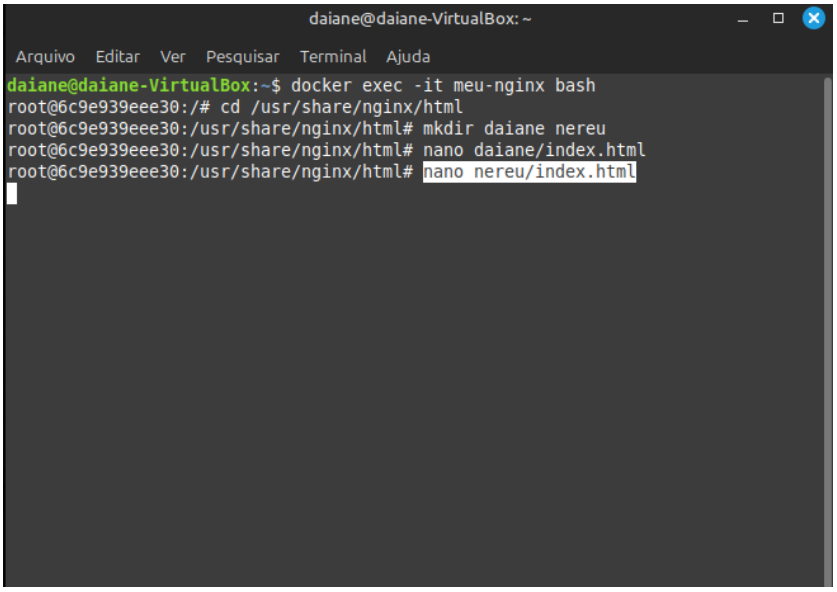

Figura 11. Criando a pasta daiane e nereu e o arquivo index em cada pasta.

Figura 12. Personalizando o index.html na pasta daiane.

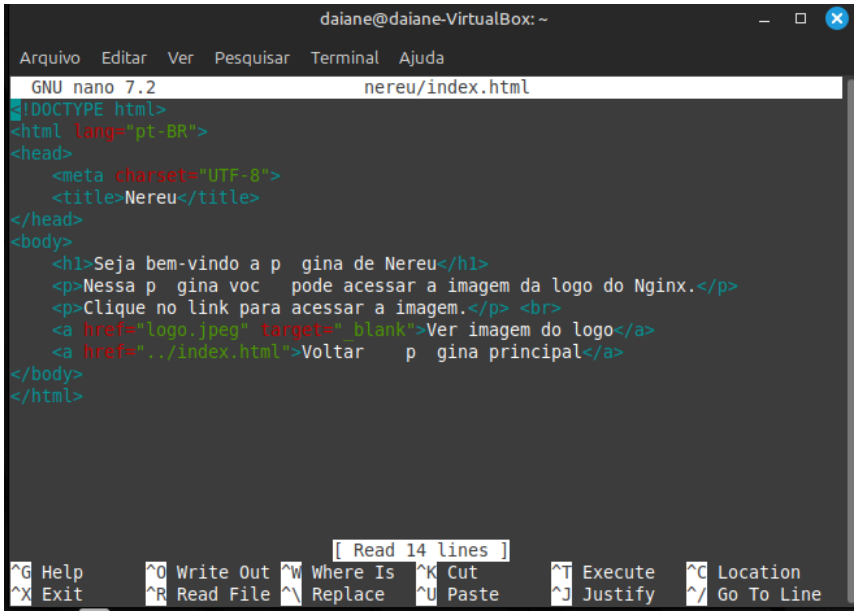

Figura 13. Personalizando o index.html na pasta daiane.

Pelas figuras 12 e 13 é possível perceber que nós copiamos arquivos externos para dentro do container com o NGNIX, pois no arquivo index.html da pasta daiane, podemos acessar esse tutorial em pdf e no arquivo index.html da pasta nereu, é possível acessar a imagem da logo do NGNIX. Iremos explicar como fizemos isso no passo 4.

4. O arquivo tutorial.pdf e logo.Jpeg foram salvos na MV, na pasta download. Então, executamos no terminal o comando:

- `docker cp /home/daiane/Downloads/tutorial.pdf meu-nginx:/usr/share/nginx/html/daiane/tutorial.pdf` – Esse comando foi usado fora do container para mover o arquivo tutorial.pdf para a pasta `/usr/share/nginx/html/daiane/` dentro do container `meu-nginx`.
- docker cp /home/daiane/Downloads/logo.jpeg meu-nginx:/usr/share/nginx/html/nereu/logo.jpeg – Esse comando foi usado fora do container para mover o arquivo logo.jpeg para a pasta `/usr/share/nginx/html/nereu/` dentro do container `meu-nginx`.

A figura 14 mostra a implementação dos comandos acima, funcionando corretamente.

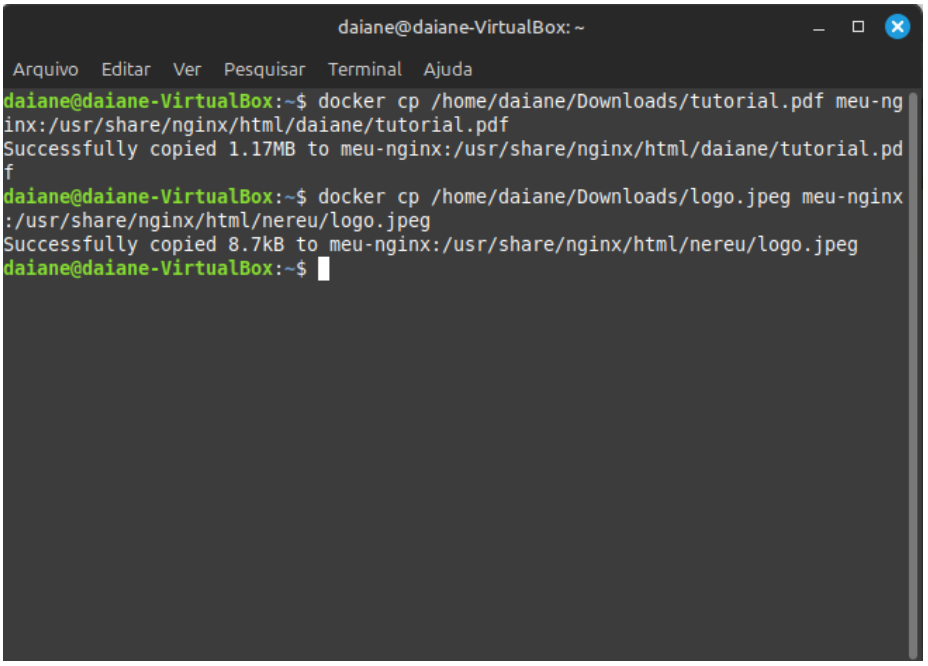

Figura 14. Copiando o arquivo tutorial.pdf e logo.jpeg para dentro do container.

Assim, foi possível acessar o arquivo tutorial.pdf pela página index.html da pasta daiane e acessar o arquivo logo.jpeg pela página index.html da pasta nereu, como mostram as figuras 15 e 16, respectivamente.

Figura 15. Página index.html da pasta daiane.

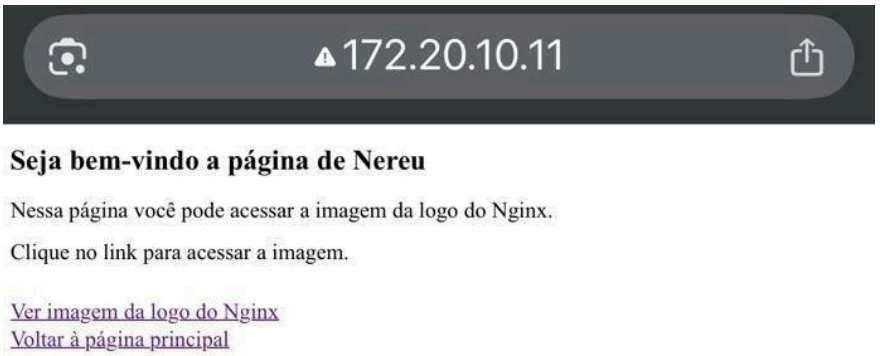

Figura 16. Página index.html da pasta nereu.

Além disso, o projeto desenvolvido foi colocado no GitHub, que é uma plataforma gratuita onde é possível publicar arquivos. A descrição de como subir o projeto nessa nuvem está apresentada na etapa 6.

2.6. Etapa 6 – Subir o projeto no GitHub.

1. Acessamos o site do GitHub: https://github.com/ e logamos no Sistema.

2. Criamos um novo repositório (ver figura 17):

- Nome: servidor-nginx-docker
- Deixamos como público
- Clicamos em Criar repositório

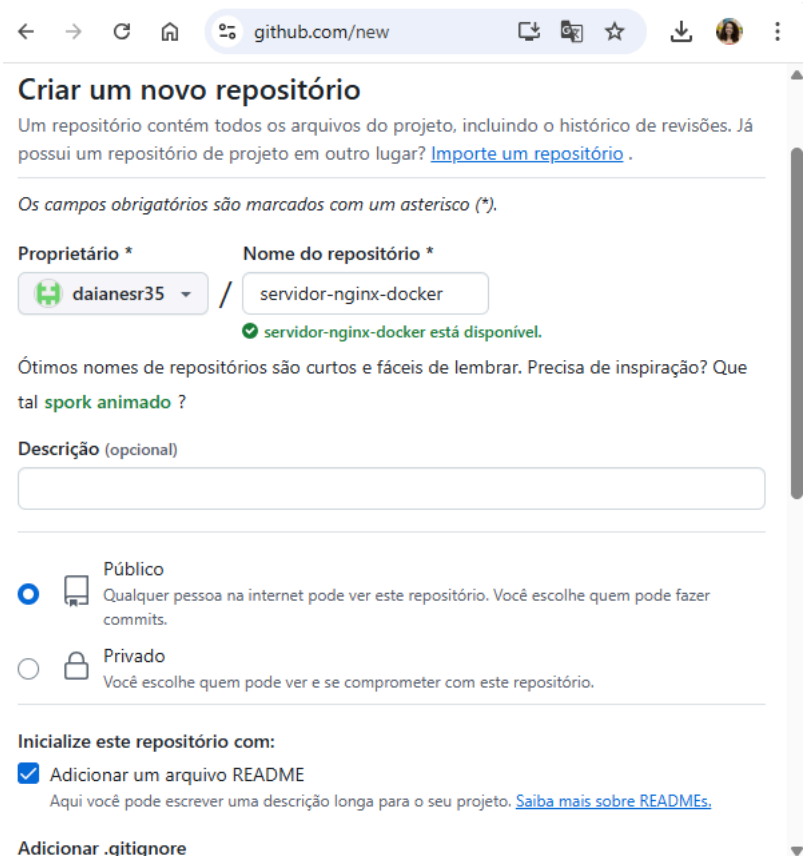

Figura 17. Criando um novo repositório no GitHub.

3. Na MV, instalamos o GitHub. Para isso, executamos os seguintes comandos no terminal da MV:

- `sudo apt update` – Esse comando atualizou a lista de pacotes disponíveis nos repositórios do Ubuntu, como mostra a figura 18.

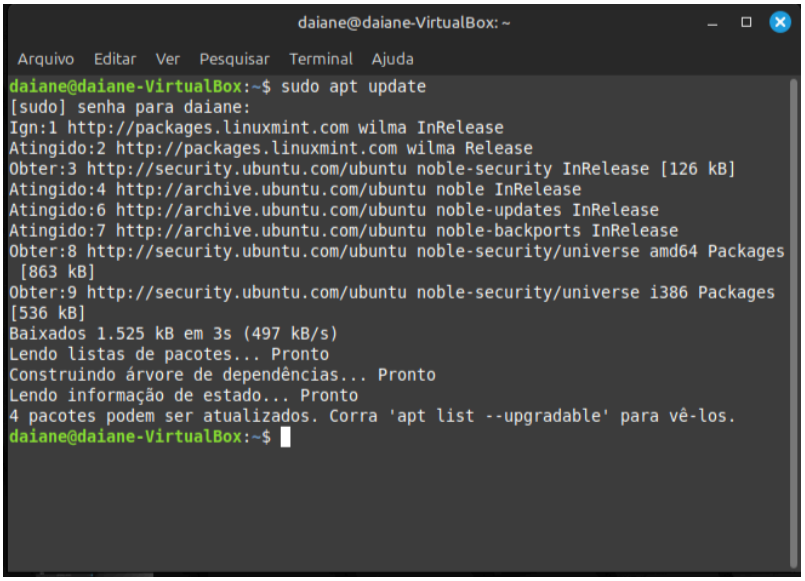

Figura 18. Atualização de lista de pacotes da MV.

- `sudo apt install git` – Esse comando instalou o Git, que é o sistema de controle de versão usado para trabalhar com repositórios no GitHub, como ilustra a figura 19.

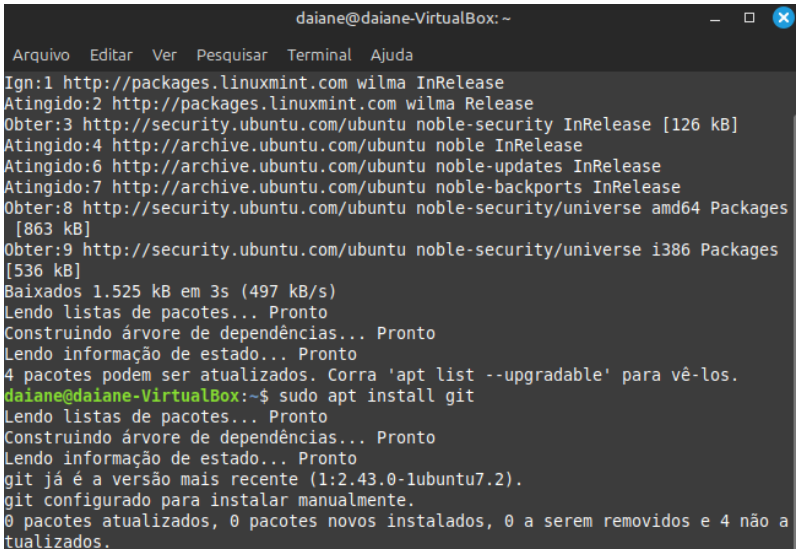

Figura 19. Instalação do Git na MV.

4. Configuramos o Git, por meio dos comandos:

- `git config --global user.name "daiane"` – Esse comando definiu o nome de usuário que será associado a todos os commits feitos com o Git na MV usada.
- `git config --global user.email "daiane.ribeiro@ifsertao-pe.edu.br"` – Esse comando definiu o e-mail do usuário que será associado aos commits.

5. Copiamos o index.html do container com NGNIX para a pasta pessoal na MV. No terminal da MV, executamos:

- `docker cp meu-nginx:/usr/share/nginx/html/index.html ~/index.html` – Esse comando copiou o arquivo index.html do container para a pasta pessoal da MV.

6. Criamos uma nova pasta para o repositório dentro da MV, através do comando:

- `mkdir ~/projeto-nginx` – Esse comando criou uma nova pasta chamada projeto-nginx dentro da pasta pessoal na MV.
- `cd ~/projeto-nginx` – Esse comando permitiu entrar na pasta projeto-nginx para começar a trabalhar dentro dela (criar arquivos, iniciar um repositório Git, etc).

7. Movemos o index.html para dentro da pasta criada no passo anterior:

- `mv ~/index.html .` - Esse comando moveu o arquivo index.html da pasta pessoal para o diretório projeto-nginx.

8. Iniciamos o GitHub na pasta projeto-nginx, por meio do comando abaixo:

- `git init`

A figura 20 ilustra a execução dos passos 4, 6, 7 e 8 dessa etapa.

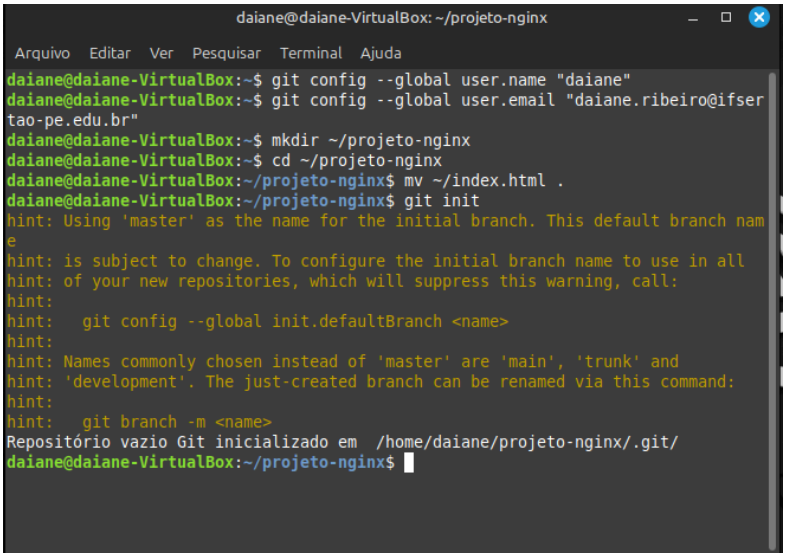

Figura 20. Configuração do Git na MV.

9. Conectamos o repositório local ao repositório remoto no GitHub, usando o comando abaixo:

- `git remote add origin https://github.com/daianesr35/servidor-nginx-docker.git`

10. Criamos o commit e enviamos para o GitHub:

- `git add index.html` - Esse comando adicionou o arquivo index.html à área de preparação do Git, indicando que ele está pronto para ser incluído no próximo commit.
- `git commit -m "Versão oficial do index.html usado no container NGINX"` - Esse comando registrou oficialmente a versão do index.html no repositório local, criando um ponto no histórico do projeto com uma mensagem explicativa.
- `git branch -M main` – Esse comando definiu o nome do branch principal como main, renomeando-o se for necessário.
- `git push -u origin main` – Esse comando enviou o commit feito localmente para o repositório remoto no GitHub, no branch main.

A execução dos comandos dos passos 9 e 10 está ilustrada na figura 21.

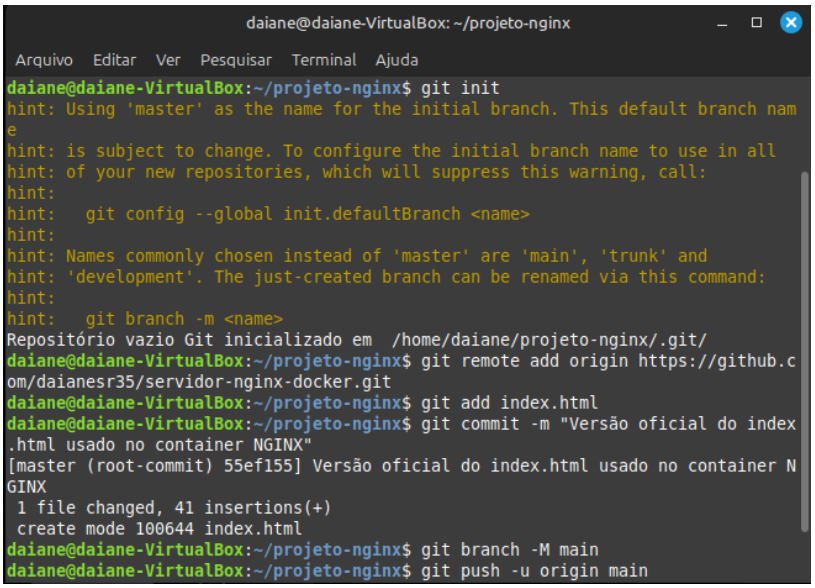

Figura 21. Conexão do repositório local com o repositório remoto.

11. Criamos o arquivo README.md com uma breve explicação do projeto, como mostra a figura 22. Na MV executamos:

- `nano README.md`

Depois pressionamos `Ctrl + O` e em seguida `Enter` para salvar o arquivo criado e pressionamos `Ctrl + X` para sair do arquivo.

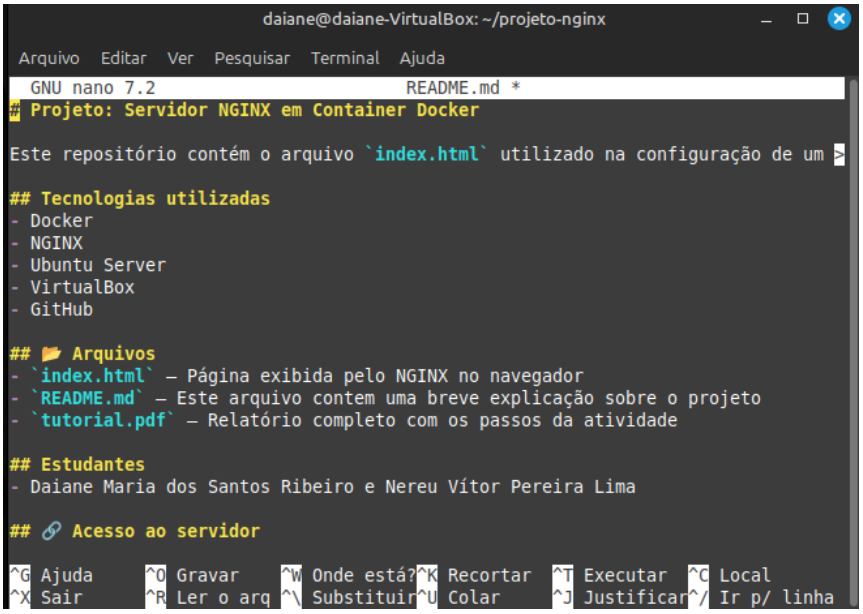

Figura 22. Editando o arquivo README.md.

12. Copiamos o arquivo tutorial.pdf para a pasta do projeto. Como o arquivo tutorial.pdf estava na pasta pessoal, usamos o comando abaixo para copiar o arquivo para a pasta projeto-nginx:

- `cp ~/tutorial.pdf ~/projeto-nginx/`

13. Adicionamos e enviamos os arquivos README.md e tutorial.pdf para o GitHub, através dos comando abaixo:

- `git add README.md tutorial.pdf`
- `git commit -m "Adicionando README.md e tutorial.pdf ao repositório"`
- `git push origin main`

14. Verificamos no GitHub e notamos a presença dos arquivos index.html, README.md e tutorial.pdf, conforme mostra a figura 23.

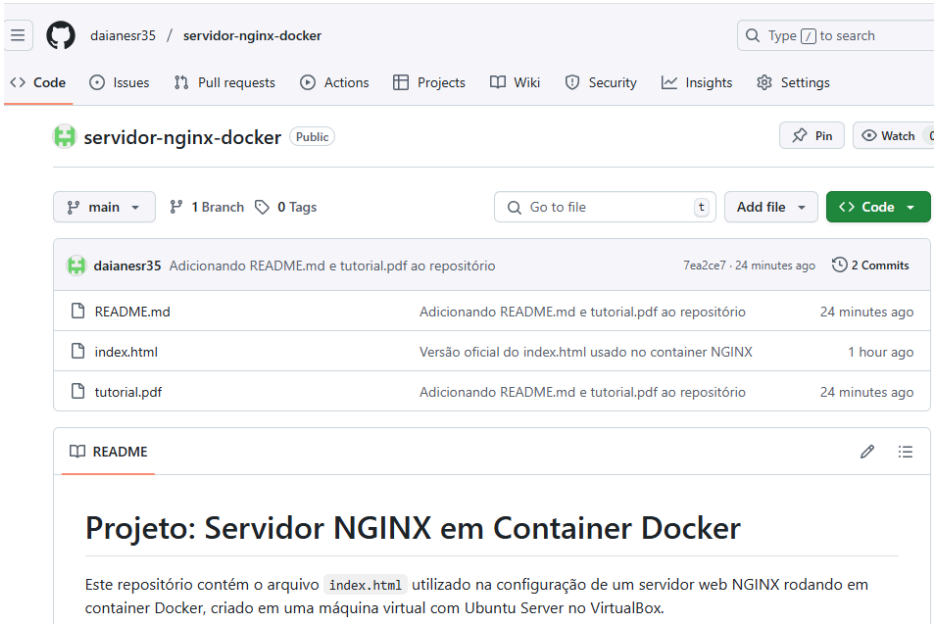

Figura 23. Arquivos sincronizados no GitHub.

Considerações Finais

A realização desta atividade possibilitou a compreensão prática e detalhada do processo de implantação de um servidor web com NGNIX utilizando contêineres Docker em uma máquina virtual no VirtualBox, com sistema operacional Linux. Durante o desenvolvimento do tutorial, foi possível observar as vantagens da conteinerização, como leveza, portabilidade, rapidez na execução e facilidade de replicação do ambiente.

A criação do container com NGNIX permitiu não apenas simular o funcionamento de um servidor web real, como também possibilitou a personalização do conteúdo servido, com a modificação do arquivo index.html e a adição de novos diretórios e arquivos acessíveis pelo navegador, inclusive a partir de dispositivos externos à máquina virtual.

Outro ponto relevante foi a disponibilização do projeto no GitHub, o que garante a rastreabilidade das versões e facilita a colaboração, compartilhamento e reuso do ambiente por outros usuários. A integração entre Docker e GitHub reforça o uso de boas práticas no desenvolvimento de sistemas em ambientes modernos, como a computação em nuvem.

Portanto, a atividade proporcionou uma vivência prática importante sobre administração e segurança de redes, contribuindo para a formação técnica e crítica quanto à construção, publicação e manutenção de serviços em ambientes virtuais e distribuídos.

Referências

W3C – WORLD WIDE WEB CONSORTIUM. (2023) “HTTP – Hypertext Transfer Protocol Overview”, https://www.w3.org/Protocols/, May.

SUSNJARA, S.; SMALLEY, I. (2025). “O que são contêineres?”, https://www.ibm.com/br-pt/think/topics/containers, July

SUSNJARA, S.; SMALLEY, I. (2025). “O que é Docker?”, https://www.ibm.com/br-pt/think/topics/docker. July.

GITHUB (2025). “Sobre o GitHub e o Git”, https://docs.github.com/pt/get-started/start-your-journey/about-github-and-git, July.