INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO DE JANEIRO

CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

PROJETO INTEGRADOR II

CONTEINERIZAÇÃO EM REDES DE COMPUTADORES

Yasmin da Mata Sales

Arraial do Cabo

2022

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO DE JANEIRO

CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMPUTADORES

PROJETO INTEGRADOR II

CONTEINERIZAÇÃO EM REDES DE COMPUTADORES

Yasmin da Mata Sales

Projeto Integrador II apresentado ao Curso Superior de Tecnologia em Redes de Computadores do Instituto Federal de Educação Ciência e Tecnologia do Rio de Janeiro, Campus Arraial do Cabo.

Arraial do Cabo

2022

Ficha catalográfica elaborada por
Monica de Oliveira Tinoco
CRB7 4850

S163c

Sales, Yasmin da Mata.

Conteinerização em redes de computadores / Yasmin da Mata Sales. – Arraial do Cabo, RJ, 2022.

40 f.; il.; 21 cm.

Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Redes de Computadores) – Instituto Federal de Educação, Ciência e Tecnologia do Rio de Janeiro, 2022.

Orientador: Prof. M.Sc. Anderson Alves de Albuquerque, M.Sc. Armando Martins de Souza.

1. Redes de computadores. 2. Docker. 3. Ansible. I. Albuquerque, Souza, Armando Martins de. II. Título.

IFRJ/CAC/CoBib CDU 004.7

RELATÓRIO DO PROJETO INTEGRADOR II

CONTEINERIZAÇÃO EM REDES DE COMPUTADORES

Yasmin da Mata Sales

Projeto Integrador II avaliado em 14 de dezembro de 2022 pela banca examinadora:

Prof. Me. Anderson Alves de Albuquerque
Orientador

Prof. Me. Armando Martins de Souza
Orientador

Prof. Me. Marcelo Simas Mattos
Membro interno

Prof. Me. Rafael de Oliveira Costa
Membro interno

Arraial do Cabo,
2022

AGRADECIMENTOS

Aos meus pais e irmão, que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto eu me dedicava à realização deste trabalho.

Aos professores, pelas correções, ensinamentos, conselhos, ajuda e pela paciência que me permitiram apresentar um melhor desempenho no meu processo de formação profissional ao longo do curso.

RESUMO

O projeto visa a implementação de contêineres na infraestrutura de aplicações de serviços de uma empresa, os quais são responsáveis por empacotar suas aplicações e dependências com a finalidade de serem executadas de forma isolada e eficiente tendo como resultado serviços mais leves, rápidos, facilmente escaláveis e economizando *hardware*, *software* e rede de uma máquina.

O uso de contêineres terá como finalidade a criação e instalação de um ambiente responsável por automatizar a configuração básica inicial de sistemas operacionais *Linux.* Inclusive, em alguns *Unix,* como, por exemplo: o *Solaris* (ORACLE, 2022) em uma empresa. Para auxiliar nessa criação serão utilizados os *softwares: Docker* que é responsável pela criação e execução do contêiner e a ferramenta *Ansible* que tem como finalidade a automação e configuração de servidores.

Palavras-chave: Contêiner. *Docker*. Automatizar as tarefas. Infraestrutura por códigos. *Ansible*.

ABSTRACT

The project aims at implementing containers in a company's service application infrastructure, which are responsible for packaging their applications and dependencies in order to run in an isolated and efficient manner, resulting in lighter, faster, easily scalable and efficient services. saving a machine's hardware, software and network.

The use of containers will have the purpose of creating and installing an environment responsible for automating the initial basic configuration of Linux operating systems. Including, in some Unix, such as, for example: Solaris (ORACLE, 2022) in a company. To assist in this creation, the following software will be used: Docker, which is responsible for creating and running the container, and the Ansible tool, which aims to automate and configure servers.

Keywords: Container. Docker. Automate tasks. Infrastructure by codes. Ansible.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Processing Unit - Unidade de processamento central
IP	Internet Protocol - Protocolo de Internet
SSH	Secure Shell - Shell seguro
TI	Tecnologia da Informação
VM	Virtual Machine - Máquina Virtual
WinRM	Windows Remote Management - Gerenciamento remoto do Windows
WSL	Windows Subsystem for Linux - Subsistema Windows para Linux
YAML	Ain’t Markup Language - Não é linguagem de marcação

SUMÁRIO

1 INTRODUÇÃO

Atualmente, empresas de vários segmentos e tamanhos são altamente dependentes da tecnologia e sua infraestrutura. Devido a isso, temos alguns fatores que devemos levar em consideração ao implantar e manter essa infraestrutura tecnológica, como exemplo, podemos citar: i. tempo gasto no preparo do equipamento; ii. padronização entre os equipamentos; iii. treinamento da equipe responsável, entre outros. Esses são apenas alguns pontos a serem levados em consideração quando falamos nesse tipo de infraestrutura.

O contêiner é uma aplicação usada por grandes empresas como a Google (CLOUD, 2022) e a Netflix (NETFLIX, 2022) e que tem mudado a forma de se trabalhar com virtualização, pois as máquinas virtuais têm algumas desvantagens em relação ao contêiner (RED HAT, jan. 2020), por exemplo, o tamanho da máquina virtual é em gigabyte, enquanto no contêiner sua representação é feita em megabyte. Ela executa várias funções simultâneas, pois tem um sistema operacional instalado na VM (*Virtual Machine*), ou seja, necessita de mais recursos da sua máquina, por outro lado, um contêiner usa o mesmo *kernel* da sua máquina hospedeira, em outras palavras, é uma máquina que hospeda as máquinas convidadas (virtuais), o que o torna mais leve.

Diante disso, o foco desse projeto é criar ambientes para utilização de servidores *Linux*. Assim, será possível utilizar contêiner para atender a área corporativa em instituições organizacionais e acadêmicas, bem como, demandas em disciplinas envolvendo sistemas operacionais e programação.

2 PROBLEMA

Atualmente, as empresas estão oferecendo cada vez mais serviços através da Internet, ao entendermos que a cada dia temos uma maior demanda de clientes a serem atendidos, passamos a ter uma exigência de disponibilizar mais recursos computacionais para atender esta demanda. Muitos serviços oferecidos possuem uma infraestrutura complexa, o que dificulta bastante a administração frente a escalabilidade necessária para atender a demanda crescente.

Nesse ambiente, torna-se trabalhoso a administração dos serviços e servidores manualmente, necessitando muitas vezes utilizar uma ferramenta que proporciona uma elasticidade, isto significa, adaptar às mudanças conforme com o projeto correspondente (HERBST, KOUNEV, REUSSNER, 2013), e padronização para administrar os sistemas

computacionais. Outro ponto importante é a indisponibilidade do servidor em casos de manutenção do sistema.

A elasticidade permite o aumento ou diminuição da infraestrutura de acordo com a demanda que pode ocorrer durante um determinado período, dias, meses, semanas e após isso deixar de ser necessário. Como exemplos, podemos citar: i. o último dia da entrega do Imposto de Renda (IRPF); ii. as últimas horas da *black friday*.

A questão que vai acompanhar esse projeto é: demonstrar uma forma de tornar a infraestrutura de uma empresa mais escalável, no menor tempo possível e com baixo custo.

3 HIPÓTESE

O problema que se pretende resolver ou ajudar a resolver com este trabalho é implementar serviços no menor tempo possível para atender a uma demanda maior desses serviços, podendo ser aplicada em empresas que necessitem atender a demanda variável dos clientes, o que implica no aumento ou na redução de seus serviços.

Diante do contexto descrito anteriormente, foi elaborado um projeto para tentar solucionar de maneira viável e acessível como mostra os próximos tópicos.

4 OBJETIVOS

A partir da infraestrutura de aplicações de serviços de uma empresa, pretende-se implantar os contêineres, que são responsáveis por empacotar aplicações e suas dependências com a finalidade de serem executadas de forma isolada e eficiente (ABDALLA, 2018), tendo como resultado serviços mais leves, rápidos, facilmente escaláveis e economizando hardware, software e rede de uma máquina.

4.1 OBJETIVO GERAL

Criar e instalar um ambiente para automatizar a configuração básica inicial de sistemas operacionais *Linux* em uma empresa através do uso de contêineres, como por exemplo, criar usuário, configurar *hostname*, configurar endereço IP (*Internet Protocol*), pré-instalar pacotes, entre outros.

A partir do contêiner será possível viabilizar de forma orquestrada e padronizada a escalabilidade das aplicações e sistemas operacionais em ambientes complexos mitigando os gastos de recursos computacionais (CPU, memória, etc). Também é possível garantir a

portabilidade, ou seja, carregar o mesmo sistema ou aplicações entre diferentes máquinas e de diferentes sistemas operacionais sem que haja perdas ou mudanças de serviços.

4.2 Objetivos Específicos

Nas subseções a seguir será detalhado o objetivo deste projeto.

4.2.1 Docker

O *Docker* é um *software livre* que é responsável por entregar serviços empacotados de virtualização em nível de sistema operacional e segundo Abdalla:

> "É uma tecnologia de virtualização de processos, em que aplicações são encapsuladas em contêineres e iniciadas como um processo isolado no sistema operacional." (ABDALLA, 2018)

Além disso, o *Docker* será empregado para encapsular as aplicações e suas dependências em uma mesma imagem (APÊNDICE A), gerando um único produto para cada função (Figura 1).

O destaque neste trabalho é explicar o que são os conteúdos citados anteriormente, mostrar suas: i. características; ii. função; iii. importância; iv. ferramentas; v. aplicação; vi. instalação do *Docker*; vii. configuração do *Docker*; viii. comandos básicos e avançados para que esse *software* possa funcionar dentro do proposto nesse projeto; ix. criação e construção de imagens e contêineres.

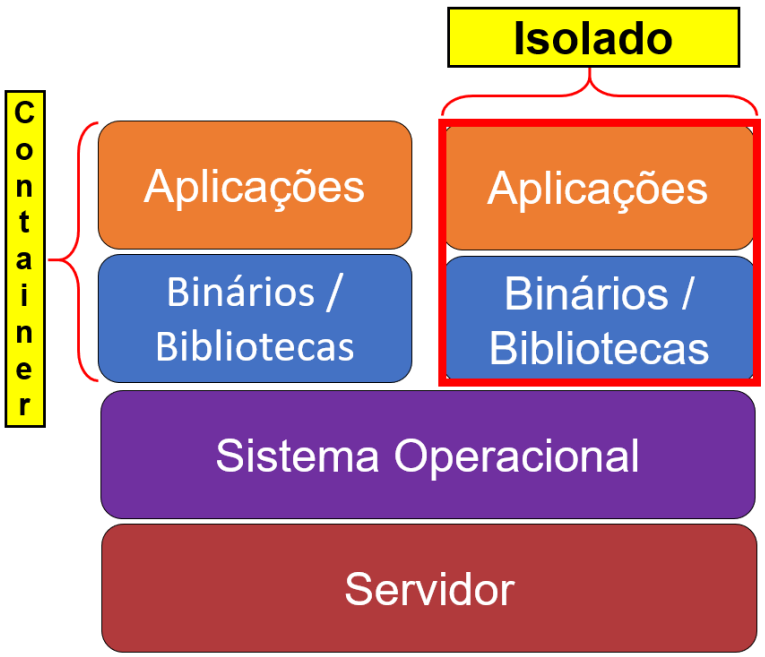

Figura 1 - Arquitetura do contêiner (adaptado de RESEARCHGATE, 2022)

A figura 1 apresenta a arquitetura de um contêiner, o qual o servidor tem o seu sistema operacional e os contêineres se utilizam desse mesmo sistema operacional da máquina hospedeira. Cada contêiner é isolado em pacotes com as aplicações e suas dependências, que podem ser binários e bibliotecas.

4.2.2 Ansible

O *Ansible* (RED HAT, nov. 2020) é uma ferramenta de código aberto que tem a finalidade de automatizar, gerenciar e configurar servidores. Será implementado no ambiente para tornar automático toda a configuração básica de uma empresa.

O objetivo desse projeto além de explicar o conteúdo citado acima, é destacar: i. características; ii. funções; iii. aplicações; iv. instalação do *Ansible*; v. passos para atuar no desenvolvimento desta aplicação.

5 Justificativa

Diante do exposto na contextualização, o mercado necessita que a administração de sistemas atenda aos requisitos mencionados no tópico problema, permitindo atender melhor a quantidade crescente de usuários cada vez mais conectados. Ademais, com a Internet das Coisas (IoT) a demanda por soluções ágeis e elásticas tendem a aumentar.

Ao observar essa crescente demanda de profissionais em administração de servidores, especialmente em redes de computadores, notou-se a viabilidade de elaborar um projeto para explorar com ênfase a virtualização em nível de sistema operacional (Figura 1), com a proposta de automatizar processos de instalação e configuração de sistemas operacionais, que são longos e requerem muito tempo.

Assim, é possível atender a parques informáticos com grande quantidade de servidores *Linux*. Dessa forma, seria possível tornar o ambiente de trabalho mais escalável, melhor gerenciável, mais confiável, evitando erros humanos, padronizando e gerenciando os servidores (TECMUNDO, 2020).

O projeto em questão terá o foco no mercado de trabalho e poderá ser aplicado em empresas de portes variados, como por exemplo, a Linux Solutions, especializada em suporte *Linux*.

6 IMPLEMENTAR UM AMBIENTE COM *DOCKER* E *ANSIBLE*

6.1 DOCKER

Nas subseções a seguir são detalhados os componentes do *Docker* e se tem explicações sobre instalação e implementação dele.

6.1.1 CONTÊINER

O contêiner é um método utilizado na implementação e na execução de aplicativos distribuídos (DOCKER, 2022) sem que haja a necessidade de configuração manual de uma máquina virtual para cada um desses aplicativos. Assim, em um parque informático, com muitos equipamentos, o contêiner pode automatizar a tarefa de criação, instalação e configuração do sistema operacional e aplicações. O objetivo final é isolar e facilitar a portabilidade dessas aplicações.

6.1.2 SOFTWARE

O *Docker* cria pacotes de software em unidades padronizadas chamadas de contêineres que têm tudo o que uma aplicação precisa para ser executado, inclusive bibliotecas, ferramentas de sistema, código e *runtime,* em outras palavras, é a execução do contêiner. Ao usar o *Docker* (APÊNDICE B), é possível implantar e escalar rapidamente aplicações em qualquer sistema operacional e ter a certeza de que o seu código será executado.

Uma vantagem ao utilizar o contêiner é o uso de microsserviços, o qual consiste em dividir seus serviços e funcionalidades em diversas imagens, com isso a arquitetura de uma aplicação se torna muito mais versátil e funcional, por exemplo, na hora de uma manutenção ou atualização em parte do sistema, nem toda a aplicação ficará indisponível.

6.1.3 INSTALAÇÃO DO *DOCKER*

A instalação do Docker pode ocorrer com apenas uma linha de comando dependendo do sistema operacional ou em alguns sistemas é necessário instalar também repositórios, o qual é um lugar de armazenamento de arquivos que podem ser recuperados e instalados em um computador.

6.1.3.1 No Debian

No sistema operacional Debian, para executar a instalação do *Docker* precisará de alguns pré requisitos, o comando abaixo foi usado para instalar esses pacotes que são pré requisitos que deixam o apt usar pacotes pelo HTTPS:

```
apt-get install apt-transport-https ca-certificates curl gnupg2
software-properties-common
```

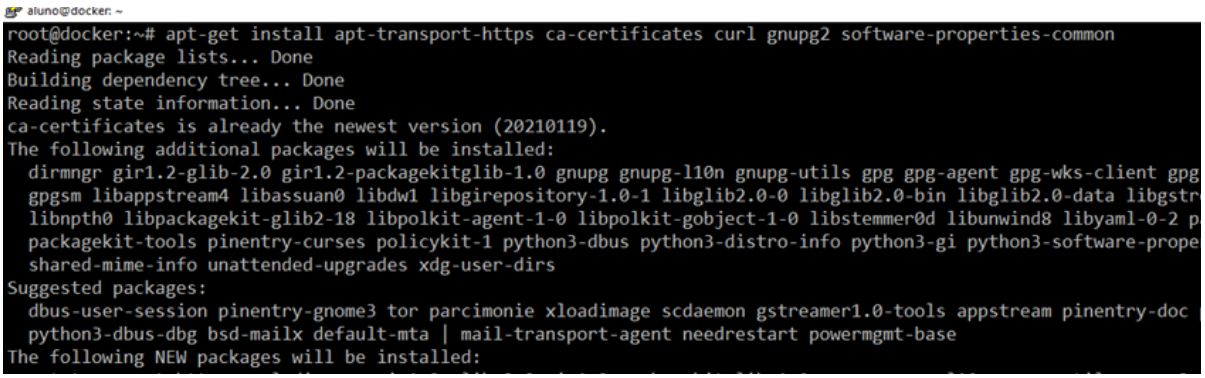

Figura 2 - Instalar pré requisitos

Para assegurar que o pacote de instalação do *Docker* disponível no repositório oficial do *Debian* seja o mais recente é importante adicionar a chave GPG, a qual usa o método de chave pública e privada com a finalidade de garantir a transferência de informações entre pares utilizando criptografia assimétrica:

```
curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key
add -
```

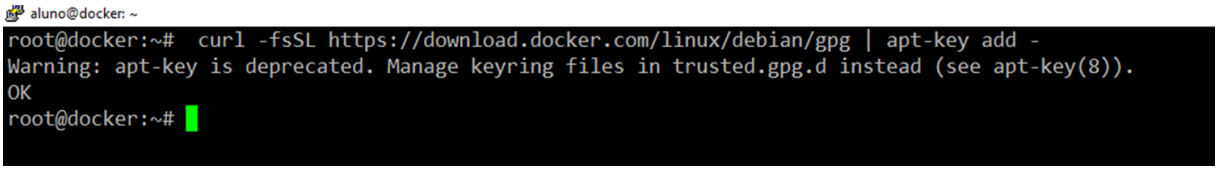

Figura 3 - Adicionar chave GPG

A mensagem gerada pelo sistema na figura 3: *"uso de apt-key está obsoleto, exceto para o uso de apt-key del em scripts de mantenedor para remover chaves existentes do chaveiro principal"*, é um aviso que não impede a atualização do sistema.

Adicionar o repositório do *Docker* às fontes do APT:

```
add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/debian $(lsb_release -cs) stable"
```

Figura 4 - Adicionar repositório *Docker*

Atualizar a lista de pacotes existentes:

```
apt-get update && apt-get upgrade
```

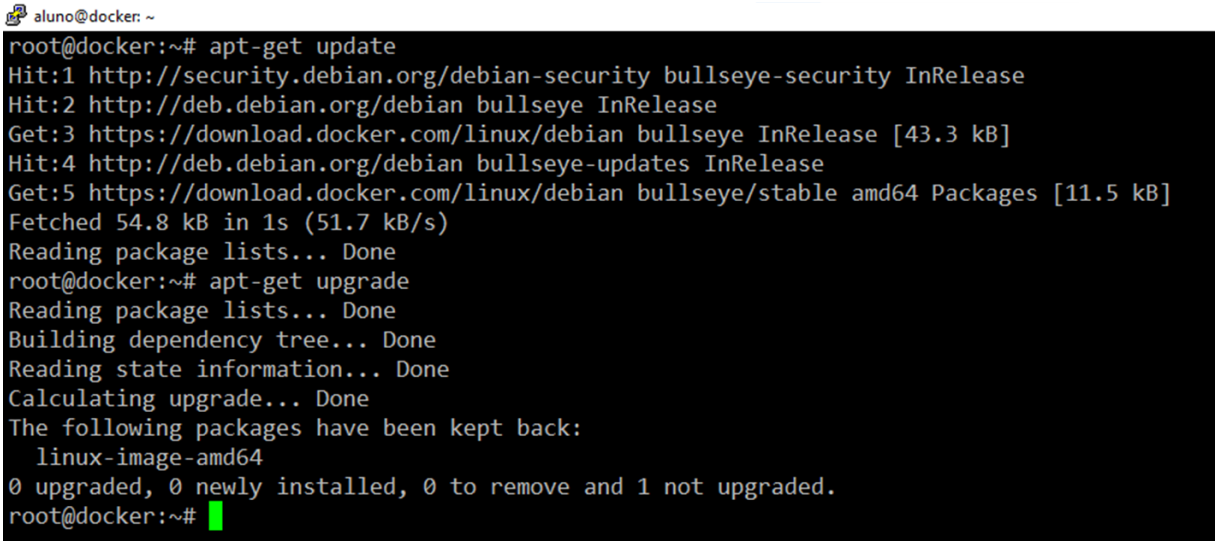

Figura 5 - Atualizar os pacotes

Instale o *Docker* no Debian:

```
apt-get install docker-ce
```

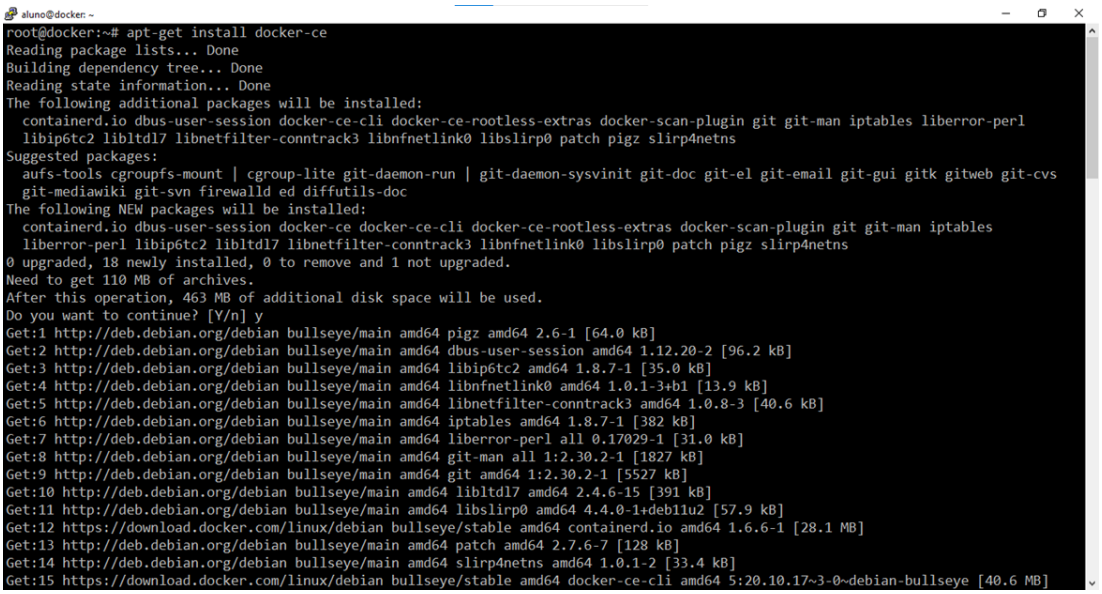

Figura 6 - Instalar o *Docker* no Debian

6.1.3.2 No Fedora

Instalação do *Docker* no sistema operacional Fedora se dá com apenas um comando, após essa instalação é importante reiniciar o serviço:

```
yum install docker
```

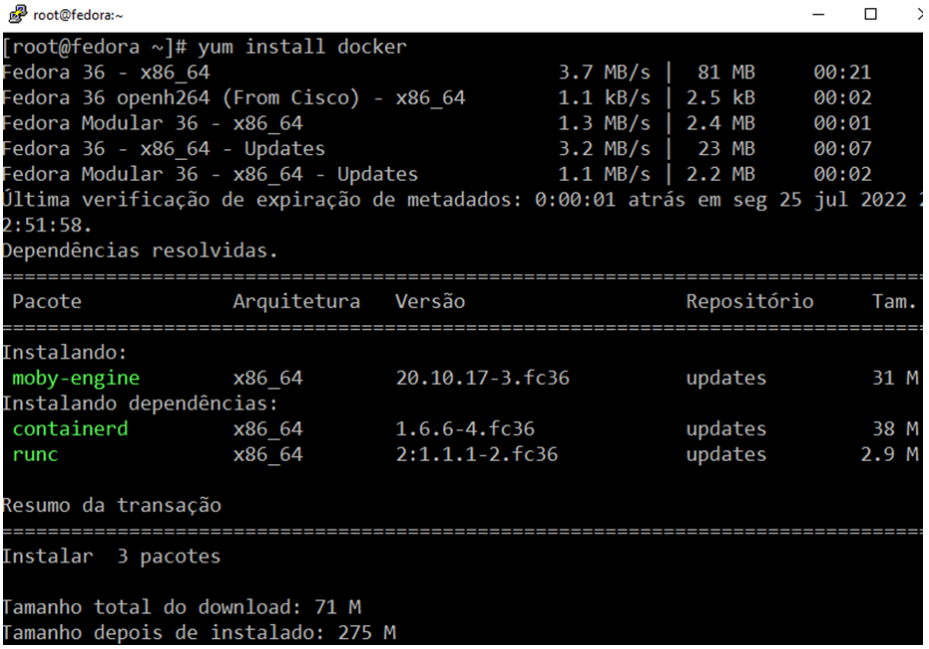

Figura 7 - Instalar o *Docker* no Fedora

6.1.3.3 No *Windows*

O *Docker* foi implementado no *windows*, mas para executar essa ação foi importante habilitar o *WSL (Windows Subsystem for Linux)*, o qual é um recurso opcional disponível e que permite executar binários e *scripts* em *Linux* diretamente no *windows*, traduzindo as instruções enviadas para o sistema com uma instrução válida para o *kernel* do *windows*. Com ele é possível ter um ambiente idêntico a de uma distribuição *Linux* sem precisar usar uma máquina virtual ou algo do tipo com essa finalidade.

Para executar a habilitação do WSL, no *prompt* de comando do *windows*, como administrador, insira o comando abaixo:

```
dism.exe /online /enable-feature
/featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

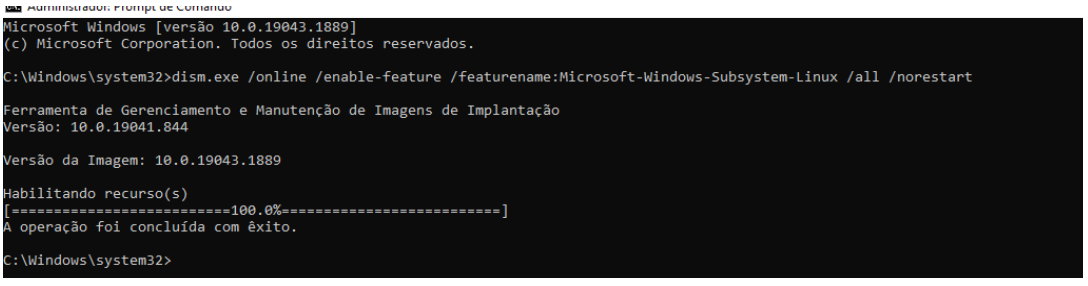

Figura 8 - Habilitar o WSL

O próximo passo é habilitar o recurso de uma máquina virtual, ainda como administrador no *prompt* de comandos, insira:

```
dism.exe /online /enable-feature
/featurename:VirtualMachinePlatform /all /norestart
```

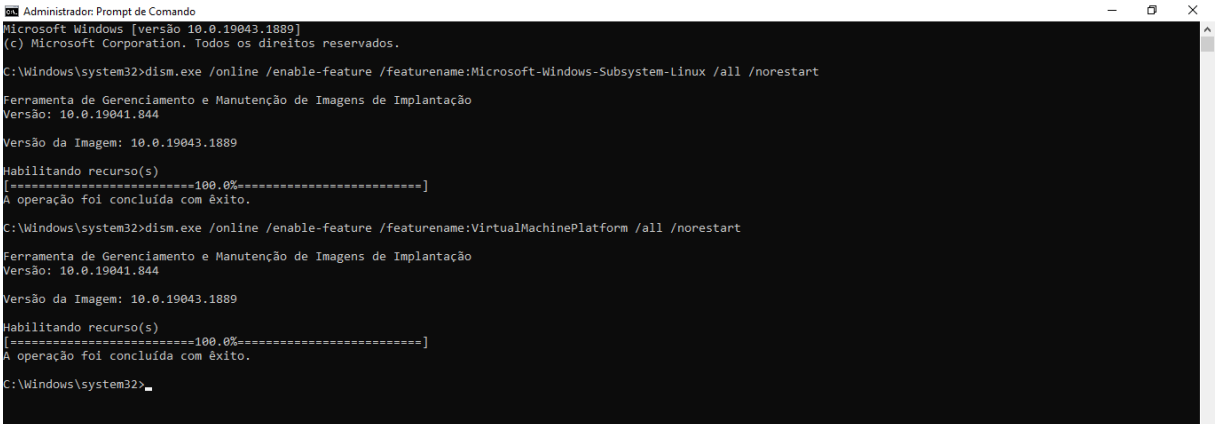

Figura 9 - Habilitar o recurso de uma máquina virtual

Para que o computador possa reconhecer essas modificações é importante reiniciar ele e logo após baixar e instalar o WSL. Para finalizar essa instalação foi usado o comando abaixo, ele serve para especificar que é para usar o WSL na versão 2, pois somente nessa versão que inclui o próprio *kernel* do *Linux* com compatibilidade total com a chamada do sistema:

```
wsl --set-default-version 2
```

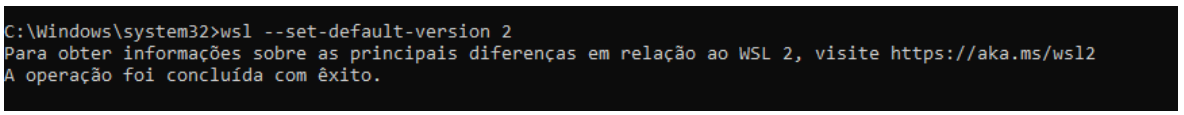

Figura 10 - Usar o WSL na versão 2

E por último, instale o *Docker* (DOCKER, 2022) diretamente do site oficial, seguindo o passo a passo da instalação descrito na página da internet, ao final da instalação terá uma imagem como essa:

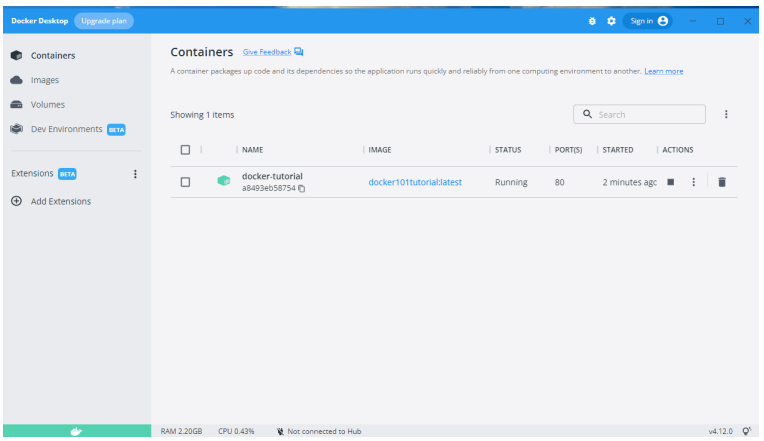

Figura 11 - *Docker* instalado no *Windows*

6.1.4 COMANDOS

Durante o estudo do *Docker* também foram realizados alguns testes aplicando o que foi aprendido na prática, com o propósito de validar o projeto, são eles: procurar, visualizar, baixar, instalar e executar algumas imagens disponíveis no repositório oficial. Na execução do *Docker* foram utilizados comandos básicos de manutenção, como por exemplo, instalação de pacotes, remoção da imagem criada.

Na figura abaixo está um exemplo de como procurar as imagens disponíveis no repositório do *Docker,* neste caso, está sendo procurada a imagem do sistema operacional Ubuntu:

Figura 12 - Procurar as imagens disponíveis

Após a escolha da imagem a qual quer adicionar a sua máquina, poderá fazer a instalação usando o parâmetro *pull,* que serve para instalar a imagem que está sendo declarada, nesse caso é o sistema operacional *Ubuntu*, como mostra a figura a seguir:

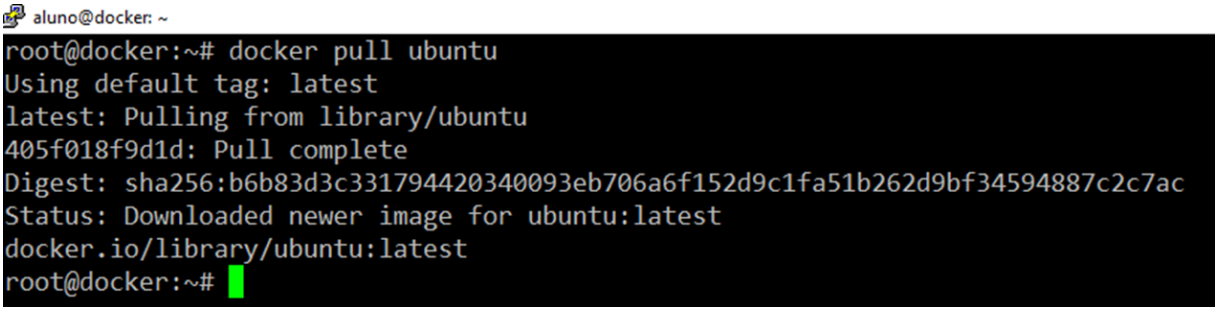

Figura 13 - Instalar a imagem Ubuntu

A figura abaixo mostra como executar a imagem, ou seja, criar um contêiner (APÊNDICE A):

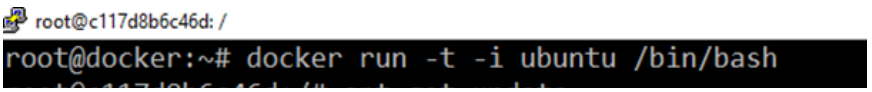

Figura 14 - Executar a imagem Ubuntu

A figura abaixo mostra como visualizar as imagens disponíveis em sua máquina:

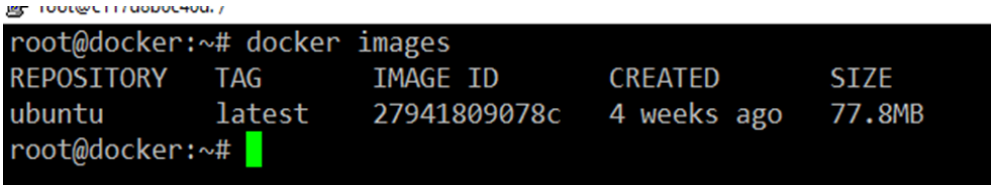

Figura 15 - Visualizar as imagens disponíveis

A figura abaixo serve para ver o código de identificação do contêiner executado e/ou em execução, como esse código é único para cada contêiner é possível remover ou alterar o nome dele na sua máquina, como está sendo mostrado na figura a seguir, onde primeira identifica o código do contêiner que quer usar, depois salva esse mesmo contêiner alterando o seu nome e por último remove o contêiner através do nome criado:

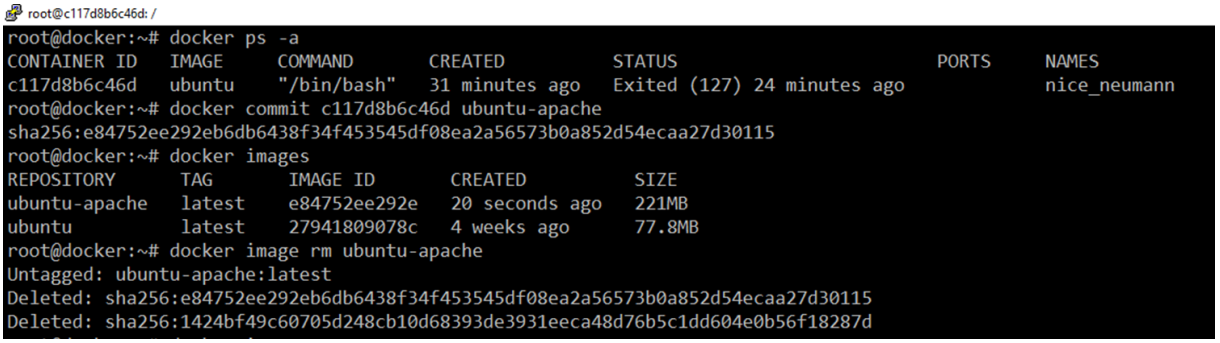

Figura 16 - Visualizar o ID e remover as imagens

Está é uma outra opção de executar a imagem, nesse caso, está sendo executado, segundo comando usado na figura abaixo, através do código de identificação do contêiner, mostrado no primeiro comando:

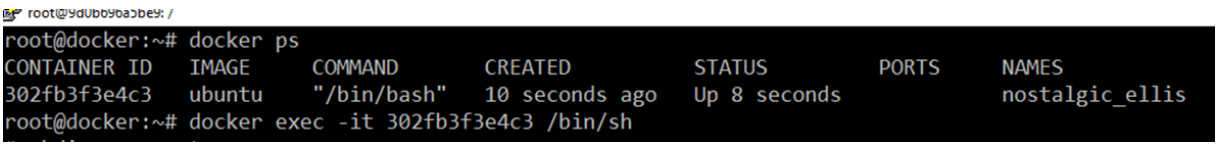

Figura 17 - Executar a imagem

Outra aplicação que foi desenvolvida nesse período é como salvar a imagem modificada, por exemplo, incluir um arquivo texto na imagem ou algo mais complexo, ao sair da execução pode usar o comando *commit* (`docker commit <ID> <nome_da_imagem :latest`) ele é responsável por salvar a imagem com modificações.

Para tornar possível a execução de qualquer configuração no contêiner sem que seja preciso estar na máquina em que ele está instalado pode usar o comando `docker exec` (SIMIC, 2019) e ele só permite executar comandos em um contêiner já em execução, dessa forma se estiver parado ou pausado irá falhar.

6.1.5 *OPENSSH* E *NGINX*

Foram realizadas as aplicações no *SSH* que é um protocolo específico de conexão entre cliente e servidor de internet, usando criptografia (SAGAR, 2021) e o *NGINX* que é um servidor web.

A figura a seguir é um exemplo de um arquivo texto, *dockerfile* (APÊNDICE A), de criação de um contêiner, onde está sendo instalado o sistema operacional *Ubuntu* na versão 16.04 e o *SSH* e também está sendo criado um usuário e senha para conexão do *SSH*:

```
 GNU nano 6.0                          dockerfile
FROM ubuntu:16.04

RUN apt-get update && apt-get install openssh-server -y
RUN apt-get install lsof -y
RUN mkdir /var/run/sshd

ARG USER=yasmin
ARG PASS="@123"
RUN useradd  -m $USER && echo "$USER:$PASS" | chpasswd
RUN echo "root:$PASS" | chpasswd


EXPOSE 22
CMD ["/usr/sbin/sshd","-D"]
```

Figura 18 - Arquivo texto SSH

A figura a seguir é um exemplo de criação de uma imagem no *Docker,* usando o parâmetro *build* que serve para construir a imagem (APÊNDICE A):

```
root@docker:~# docker build -t sshd_teste .
Sending build context to Docker daemon  12.29kB
Step 1/10 : FROM ubuntu:16.04
16.04: Pulling from library/ubuntu
58690f9b18fc: Pull complete
b51569e7c507: Pull complete
da8ef40b9eca: Pull complete
fb15d46c38dc: Pull complete
Digest: sha256:20858ebbc96215d6c3c574f781133ebffdc7c18d98af4f294cc4c04871a6fe61
Status: Downloaded newer image for ubuntu:16.04
```

Figura 19 - Criar imagem do *Docker*

A figura abaixo é um exemplo de visualização das imagens disponíveis na máquina que está sendo utilizada, essa figura tem a finalidade de mostrar que a imagem anterior foi construída e está disponível na máquina em execução:

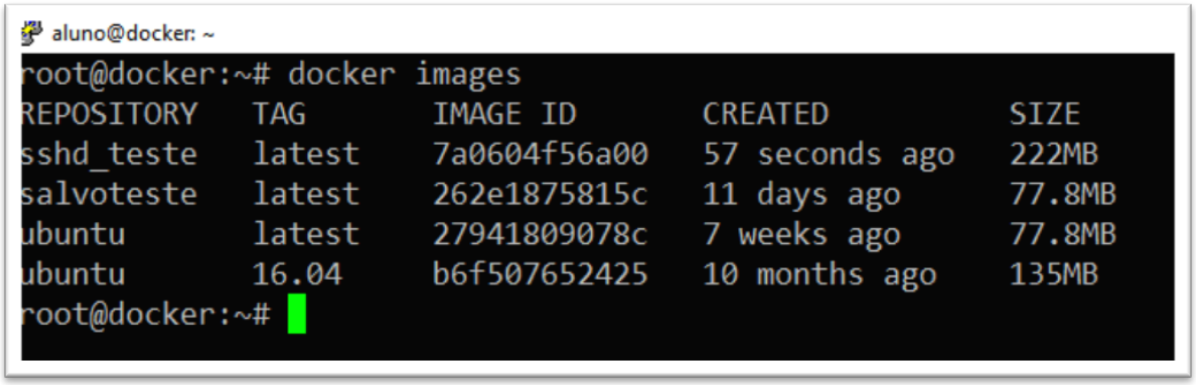

Figura 20 - Ver as novas imagens

O exemplo a seguir é para executar uma imagem, isto é, criar o contêiner (APÊNDICE A) nomeando como será utilizado no futuro:

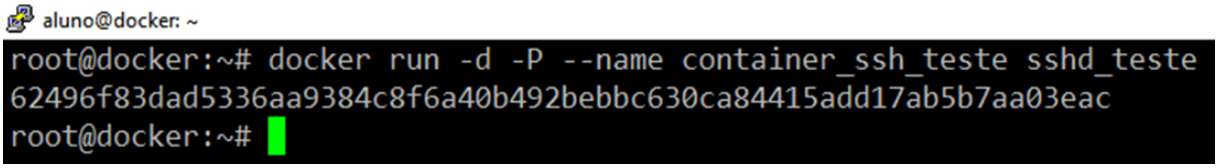

Figura 21 - Executar o contêiner

Este é um exemplo de comando, onde mostra as portas, usando o parâmetro *port*, que estão disponíveis para serem acessados por usuários externos no contêiner *container_ssh_teste*, nesse caso, a porta disponível é para acesso através do protocolo *SSH*:

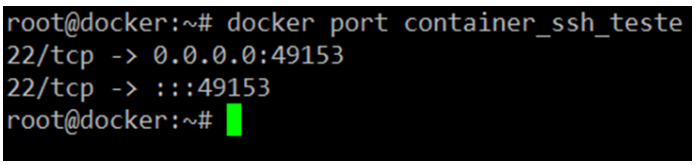

Figura 22 - Visualizar as portas disponíveis

A figura a seguir serve para inspecionar informações sobre qual é o endereço do *IP* do contêiner que o *SSH* está instalado:

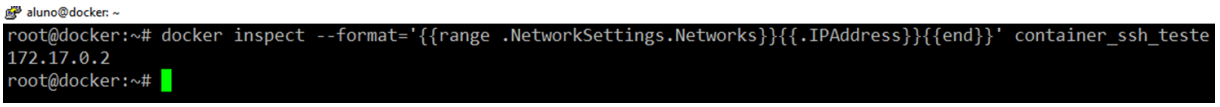

Figura 23 - Visualizar o endereço IP do contêiner

A figura abaixo mostra a conexão *SSH* sendo realizada de uma máquina *Linux* para o contêiner, nesta máquina foi somente instalado um *Linux* para servir como estação de trabalho para usuário:

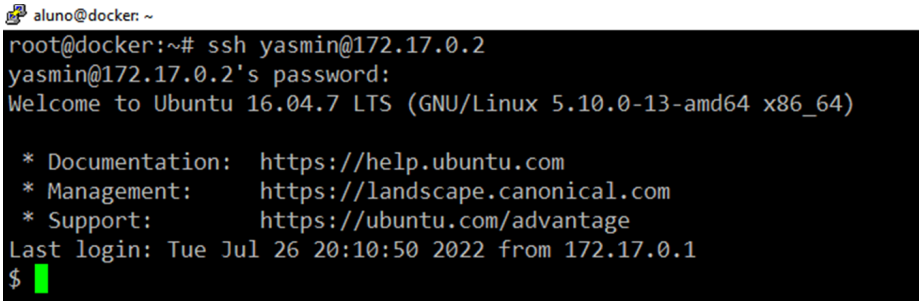

Figura 24 - Conexão SSH sendo realizada

Na figura a seguir está sendo criado um arquivo texto, onde está será instalado o servidor web *NGINX,* observe a primeira linha com o parâmetro *FROM nginx*, logo após também é instalado o *SSH* para realizar conexão diretamente com o contêiner:

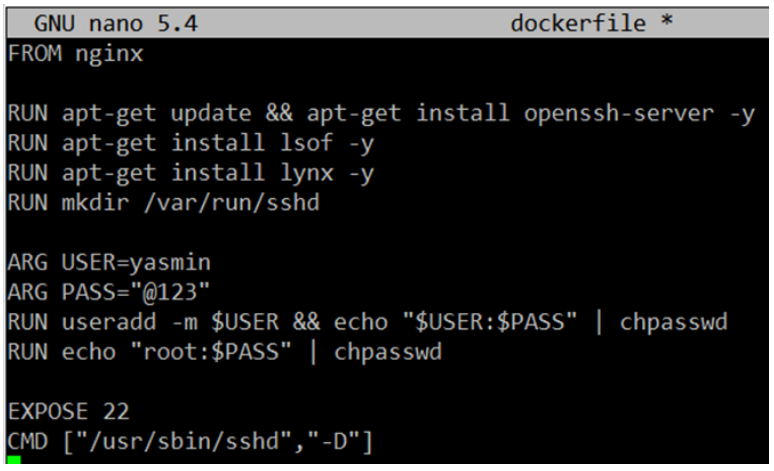

Figura 25 - Arquivo texto NGINX

Após a criação do arquivo texto do servidor web será realizado a construção da sua imagem (APÊNDICE A) na máquina hospedeira com o nome nginx_teste, como mostra a figura a seguir:

```
root@docker:~# docker build -t nginx_teste .
Sending build context to Docker daemon  14.85kB
Step 1/11 : FROM nginx
latest: Pulling from library/nginx
461246efe0a7: Pull complete
060bfa6be22e: Pull complete
b34d5ba6fa9e: Pull complete
8128ac56c745: Pull complete
44d36245a8c9: Pull complete
ebcc2cc821e6: Pull complete
Digest: sha256:1761fb5661e4d77e107427d8012ad3a5955007d997e0f4a3d41acc9ff20467c7
Status: Downloaded newer image for nginx:latest
 ---> 670dcc86b69d
Step 2/11 : RUN apt-get update && apt-get install openssh-server -y
 ---> Running in 3ec2ab12e16e
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 k
]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
```

Figura 26 - Criar imagem NGINX

Para conferir se a imagem está disponível pode ser utilizado o comando da figura abaixo:

```
aluno@docker: ~
root@docker:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED              SIZE
nginx_teste   latest    88a2a1c2626c   About a minute ago   248MB
sshd_teste    latest    7a0604f56a00   4 hours ago          222MB
nginx         latest    670dcc86b69d   7 days ago           142MB
salvoteste    latest    262e1875815c   11 days ago          77.8MB
ubuntu        latest    27941809078c   7 weeks ago          77.8MB
ubuntu        16.04     b6f507652425   10 months ago        135MB
root@docker:~#
```

Figura 27 - Ver a nova imagem NGINX

A figura abaixo mostra como criar o contêiner, ou seja, executar a imagem (APÊNDICE A) do *NGINX*:

```
root@docker:~# docker run -d -P --name container_nginx_teste nginx_teste
9ad6ffa8c0afde5e101ae91be80bbb1a3a73e787d5199e1f8a6529f612789e1f
root@docker:~#
```

Figura 28 - Executar o contêiner NGINX

Na figura a seguir está sendo mostrada as portas disponíveis do contêiner que está o servidor web:

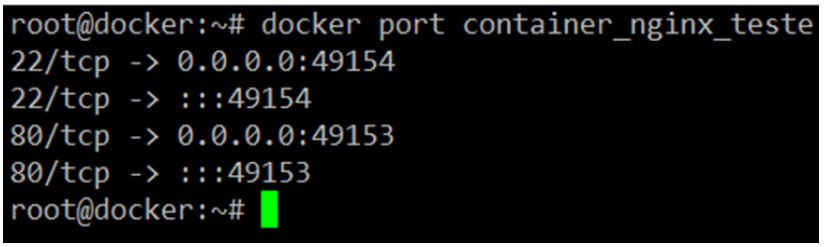

Figura 29 - Visualizar as portas disponíveis NGINX

Para visualizar a informação do IP do disponível do contêiner que está o servidor web:

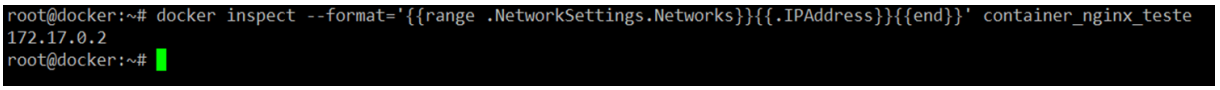

Figura 30 - Visualizar o endereço IP disponível NGINX

A figura abaixo está mostrando a conexão através do protocolo SSH no contêiner que está instalado o servidor web:

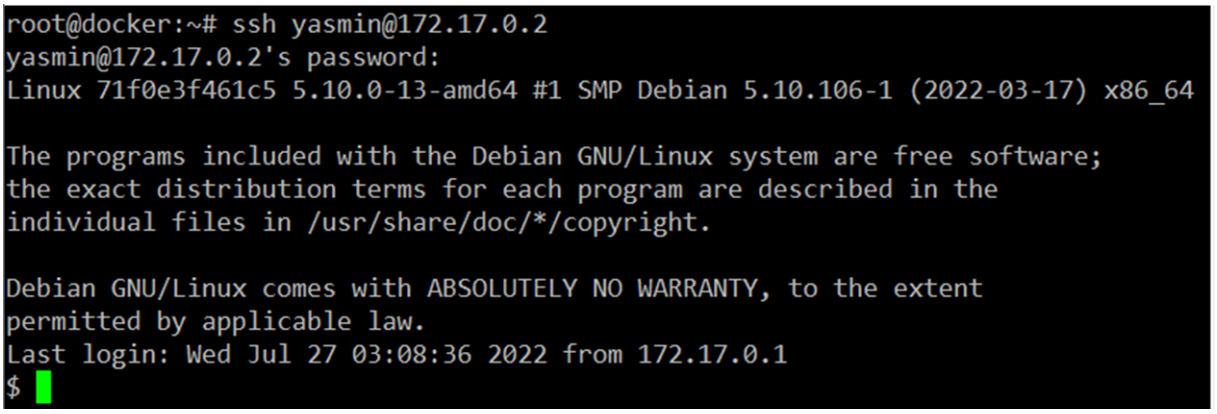

Figura 31 - Conexão SSH sendo realizada NGINX

6.2 ANSIBLE

O *Ansible* (ANSIBLE, 2022) é uma ferramenta que permite automatizar a infraestrutura em TI, pois todas as etapas da automação podem ser realizadas pelo próprio *Ansible*, como por exemplo, Web, Banco de Dados, Rede, Nuvem, *Cluster*, Monitoramento, *Windows* e outros. Essa ferramenta já possui módulos prontos que facilitam a utilização do serviço de automação, inclusive existem módulos para contêiner, como também para outros tipos de virtualização (VMware, AWS, Azure, entre outros), as quais interagem facilmente com outras tarefas do processo.

O arquivo que gera na execução do *Ansible* utiliza um formato YAML (*Ain't Markup Language* - não é linguagem de marcação), é uma linguagem declarativa de dados, tem como característica a fácil compreensão do código e também usa a linguagem *Python* para estender as suas funcionalidades (ANSIBLE, 2022).

Essa ferramenta também permite que as configurações de servidores sejam mais confiáveis, evitando erros e/ou falhas humanas, e padronizando as máquinas a serem configuradas, pois todas as configurações estarão no mesmo lugar, podendo servir inclusive de documentação para a organização.

Para que o *Ansible* possa funcionar só precisa de uma máquina de controle que é um servidor com o *Ansible* instalado e tenha acesso aos servidores que serão administrados pelo protocolo SSH ou WinRM, a qual é uma forma de acesso remoto para quem for utilizar o sistema operacional *Windows*.

O *Ansible* possui quatro conceitos novos e que devem ser compreendidos com a finalidade de colocar em prática toda a aprendizagem, são eles: 1- inventário: onde estão armazenados os endereços das máquinas que serão acessadas e configuradas; 2- módulos ou *tasks*: são tarefas que serão executados uma de cada vez, seguindo a ordem proposta pelo administrador; 3- *play*: é um conjunto de módulos que serão executados sequencialmente; 4- *playbook*: é um arquivo YAML que contêm um ou mais *plays*, ou também pode ser definido como a descrição das tarefas.

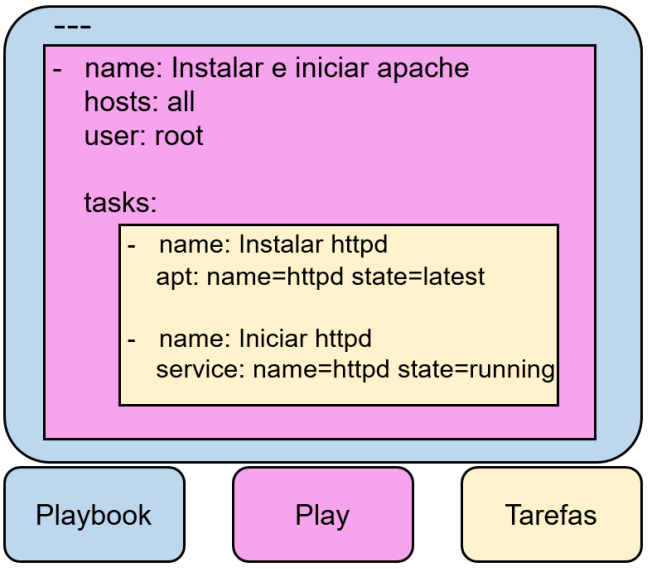

Figura 32 - Estrutura do *Ansible* (adaptado de SINGH, 2020)

6.2.1 INSTALAR O *ANSIBLE*

Para instalar o *Ansible* no Debian (SOUZA, 2020) são realizados quatro passos, conforme está sendo mostrado nas próximas figuras: 1- baixar as atualizações e informações dos pacotes de todos os repositórios configurados (Figura 33); 2- instalar um pacote com *softwares* de propriedades comum (Figura 34), onde tem o *Python*, por exemplo; 3- adicionar o repositório do *Ansible* (Figura 35); 4- e instalar o *Ansible* (Figura 36).

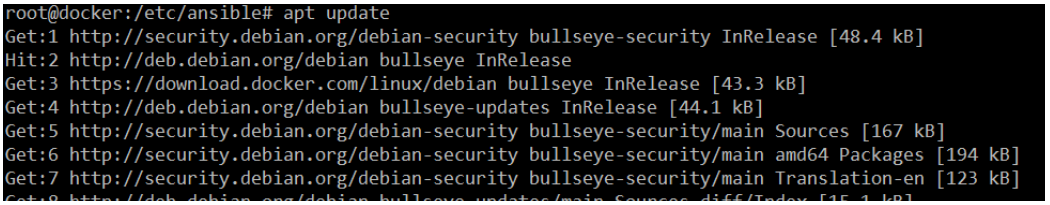

Figura 33 - Baixar as atualizações e informações

```
root@docker:/etc/ansible# apt install software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
software-properties-common is already the newest version (0.96.20.2-2.1).
```

Figura 34 - Instalar um pacote com *softwares* de propriedades comum

```
root@docker:/etc/ansible# apt-add-repository --yes --update ppa:ansible/ansible
^CTraceback (most recent call last):
  File "/usr/bin/apt-add-repository", line 122, in <module>
    shortcut = shortcut_handler(line)
  File "/usr/lib/python3/dist-packages/softwareproperties/SoftwareProperties.py", line 84
    ret = factory(shortcut)
  File "/usr/lib/python3/dist-packages/softwareproperties/ppa.py", line 393, in shortcut_
    return PPAShortcutHandler(shortcut)
  File "/usr/lib/python3/dist-packages/softwareproperties/ppa.py", line 356, in __init__
    info = get_ppa_info(self.shortcut)
  File "/usr/lib/python3/dist-packages/softwareproperties/ppa.py", line 327, in get_ppa_i
    ret = get_ppa_info_from_lp(user, ppa)
```

Figura 35 - Adicionar o repositório do *Ansible*

```
root@docker:/etc/ansible# apt install ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ansible is already the newest version (2.10.7+merged+base+2.10.8+dfsg-1).
0 upgraded, 0 newly installed, 0 to remove and 69 not upgraded.
```

Figura 36 - Instalar o *Ansible*

Ao final da instalação se desejar saber se está funcionando corretamente poderá usar o comando que está na figura abaixo para verificar a conectividade, isto é, se estiver na mesma máquina, porém se estiver em máquina diferente é só inserir o IP do servidor que deseja testar, estando com a mensagem *Success* significa que está certo e que pode começar a configuração:

```
root@docker:/etc/ansible# ansible localhost -m ping
localhost | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Figura 37 - Verificar a conectividade do *Ansible*

6.2.2 Passos para atuar

Os passos para atuar com o *Ansible* é: i. ativar a chave do *SSH* com o servidor que será configurado; ii. criar um arquivo *YAML*; iii. criar o inventário de *hosts*; iv. usar o comando `(ansible-playbook -i [arquivo_inventário] [arquivo_playbook])`, para executar todas as tarefas descritas no arquivo.

Na figura abaixo está sendo mostrado como criar um arquivo *YAML* simples de instalação do *Apache*, o qual é um servidor web:

```
192.168.1.10 - PuTTY
  GNU nano 5.4
---
#playbook
- name: Criando servidores web
  hosts: all

  tasks:
    #tasks 1
    - name: instalar apache
      apt: name=apache2 update_cache=yes state=latest
```

Figura 38 - Criar um arquivo YAML

A próxima figura também é de criação de um arquivo YAML, porém tem mais tarefas especificadas em sua descrição, são elas: criação de diretório e instalação de alguns pacotes:

```
---
#defaults:
#playbook
  - name: Criando servidores web
    hosts: all


    tasks:
    #tasks 1
    - name: criar diretorio
      file: path={{item}} state=directory mode=0755 owner=nginx
      with_items:
        - /var/www
        - /var/www/html
      notify: Restart nginx
      when: ansible_distribution == 'RedHat' or ansible_distribution == 'CentOS'
    #tasks 2
    - name: instalando pacotes essenciais
      apt: name= "{{ packages }}"
      vars:
        packages:
          - vim
          - git
```

Figura 39 - Arquivo YAML com mais tarefas

Na próxima figura tem um exemplo de arquivo de inventário com variáveis de tipo de conexão, tipo de usuário que será conectado e a senha para login na máquina que será configurada:

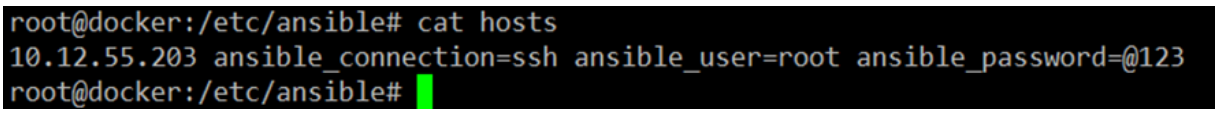

Figura 40 - Criar o inventário

A figura a seguir mostra as tarefas sendo executadas (APÊNDICE C):

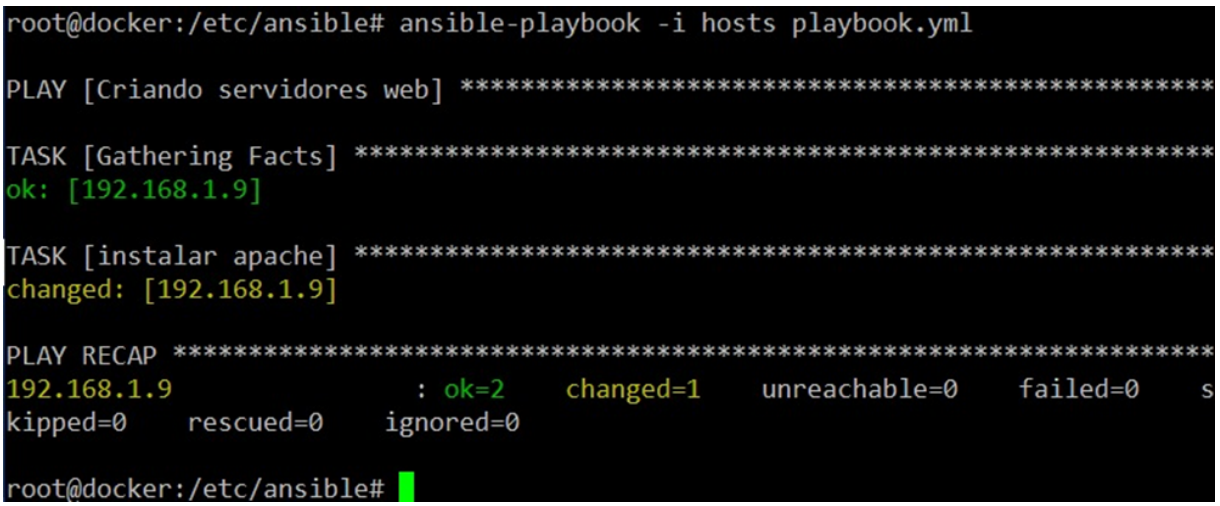

Figura 41 - Executar as tarefas

7 TRABALHOS FUTUROS

A construção desse trabalho mostrou os seus primeiros passos em relação a automação de criação e manutenção de servidores através das tecnologias específicas, *Docker* e *Ansible*, foi possível demonstrar como realizar essas tarefas. Na perspectiva de tornar esse projeto em um processo contínuo a sugestão é usar um software que tenha como finalidade a organização de todos os contêineres gerados, onde cada contêiner é referente a uma parte do mesmo sistema, para que juntos eles possam construir um sistema completo, ou seja, orquestrar os contêineres.

A sugestão é implementar o *kubernetes* (KUBERNETES, 2022), o qual é um gerenciador de *clusters* que contêm *hosts* para executar aplicações em contêiner, esse *software* tem seu código aberto que automatiza a instalação, o dimensionamento e a gestão de aplicações em contêineres e é um *software* responsável por orquestrar os contêineres.

Existem outros *softwares* que podem ser responsáveis por orquestrar os contêineres, como por exemplo, *Swarm*, *Terraform* e *Rancher*, esse último trabalha em junto com o *Kubernetes*, citado anteriormente.

8 CONCLUSÃO

O desenvolvimento do presente estudo possibilitou um conhecimento mais avançado no *Docker*, implementando na prática a criação de contêineres e imagens, a persistência dos dados nos contêineres, a execução do *Docker* em outra máquina, através do *SSH* (*Secure Shell*) e a instalação do servidor *Web*, através da criação de arquivos *dockerfile*..

Para automatizar e gerenciar o ambiente que será criado, houve um estudo mais aprofundado na ferramenta de infraestrutura por código, o *Ansible*, em suas funcionalidades e códigos para a obtenção de um melhor resultado.

Referências

ABDALLA, Gabriel. Curso Docker Unleashed, 2018. Disponível em: https://docker-unleashed.readthedocs.io. Acesso em: 04 jun. 2022.

ANSIBLE. Collection Index, 2022. Disponível em: https://docs.ansible.com/ansible/latest/collections/index.html. Acesso em: 10 nov. 2022.

CLOUD, Google. Contêineres no Cloud. Disponível em: https://cloud.google.com/containers. Acesso em: 15 dez. 2022.

DOCKER, Docs. Docker Desktop. Disponível em: https://docs.docker.com/desktop/. Acesso em: 09 set. 2022.

FARIAS, Roberto. Se torne um especialista em automação com Ansible na 4Linux!, 23 mar. 2022. Disponível em: https://blog.4linux.com.br/se-torne-um-especialista-em-automacao-com-ansible-na-4linux/. Acesso em: 17 out. 2022.

HERBST, Nikolas R.; KOUNEV, Samuel; REUSSNER, Ralf. Elasticity in Cloud Computing: What It Is, and What It Is Not, 2013. Disponível em: https://sdq.kastel.kit.edu/publications/pdfs/HeKoRe2013-ICAC-Elasticity.pdf. Acesso em: 15 dez. 2022.

KUBERNETES. Documentação do Kubernetes, 20 fev. 2022. Disponível em: https://kubernetes.io/docs/home/. Acesso em: 08 jun. 2022.

NETFLIX. Titus. Disponível em: https://netflix.github.io/titus/. Acesso em: 15 dez. 2022.

ORACLE. Guia de administração do sistema: Gerenciamento de recursos do Oracle Solaris Containers e Oracle Solaris Zones. Disponível em: https://docs.oracle.com/cd/E38904_01/html/820-2978/zones.intro-1.html. Acesso em 15 dez. 2022.

RED HAT. Containers e máquinas virtuais (VMs), Rio de Janeiro, 15 jan 2020. Disponível em: https://www.redhat.com/pt-br/topics/containers/containers-vs-vms. Acesso em: 02 jun. 2022.

RED HAT. Noções básicas do Ansible, 02 nov. 2020. Disponível em: https://www.redhat.com/pt-br/topics/automation/learning-ansible-tutorial. Acesso em: 05 jun. 2022.

RESEARCHGATE. ResearchGate. Disponível em: https://www.researchgate.net/figure/Linux-container-architecture_fig2_324616594. Acesso em: 10 jun. 2022.

SAGAR. How to SSH into Docker Containers [Step-by-Step], 27 ago. 2021. Disponível em:https://adamtheautomator.com/ssh-into-docker-container/#Setting_up_an_OpenSSH_Server_and_Connecting_with_a_Dockerfile. Acesso em: 14 jul. 2022.

SIMIC, Sofija. How to SSH into a Running Docker Container and Run Commands, 24 out. 2019. Disponível em: https://phoenixnap.com/kb/how-to-ssh-into-docker-container. Acesso em: 09 jul. 2022.

SINGH, Sumit. Ansible Configuration Management Tool, 6 ago 2020. Disponível em: https://k21academy.com/devops-foundation/ansible-playbook-galaxy-tower/. Acesso em: 09 nov. 2022.

SOUZA, Vinicius. Primeiros passos com Ansible, 15 mar. 2020. Disponível em: https://ironlinux.com.br/primeiros-passos-com-ansible/. Acesso em: 14 out. 2022.

TECMUNDO. Containers: como usar essa tecnologia no Brasil, 22 out 2020. Disponível em: https://www.tecmundo.com.br/software/205624-containers-usar-tecnologia-brasil.htm. Acesso em: 05 jun 2022.

TECNISYS. Qual é a diferença entre uma Imagem Docker e um Container?, 21 jul. 2021. Disponível em: http://www.tecnisys.com.br/noticias/2021/qual-e-a-diferenca-entre-uma-imagem-docker-e-um-container. Acesso em: 28 jul. 2022.

GLOSSÁRIO

Clusters: é um termo dado a um sistema que conecta uma série de computadores em uma rede para que eles trabalhem de maneira conjunta no processamento de tarefas complexas, resolução de programas ou integridade de dados.

Commit: tornar um conjunto de alterações permanentes.

Contêiner: é um processo (Figura 2) que executa uma imagem. (TECNISYS, 2021)

Escalável: é quando uma empresa alcança grandes resultados com baixo investimento e recursos.

Hardware: é um termo técnico que se refere a parte física de computadores ou outros sistemas eletrônicos.

Hosts: o seu significado na tradução é hospedeiro, na informática podemos acrescentar que é qualquer dispositivo que esteja conectado a uma rede.

Imagem: é um arquivo (Figura 2) que contém todas as dependências necessárias para executar um processo, exemplos: arquivos de biblioteca, arquivos no sistema de arquivos, pacotes instalados, recursos disponíveis e processos em execução. (TECNISYS, 2021)

Inventário: onde estão os endereços das máquinas que serão acessadas e configuradas.

Kernel: é o componente central do sistema operacional no computador, o núcleo.

Linux: é um termo usado para se referir a sistemas operacionais que utilizam o *kernel* desenvolvido pelo programador Linus Torvalds.

Microsserviços: o qual consiste em dividir seus serviços e funcionalidades em diversos pedaços, com isso a arquitetura de uma aplicação se torna muito mais versátil e funcional.

Módulos ou *tasks*: são tarefas que serão executadas de maneira granular, ou seja, executa uma tarefa por vez, seguindo a ordem proposta pelo administrador.

NGINX: Servidor web.

Orquestrar: orquestrar normalmente é associado à música, mas na área de tecnologia da informação ele significa gerenciar ou coordenar tarefas de automação para obterem eficiência no resultado.

***Play*:** é um conjunto de módulos que serão executados sequencialmente.

***Playbook*:** é um arquivo YAML que contém um ou mais plays, ou também pode ser definido como a descrição das tarefas.

Python: é uma linguagem de programação de fácil entendimento.

Runtime: é a execução do projeto.

Software: é um termo técnico que se refere a parte lógica dos computadores.

***Tasks* ou módulos:** são tarefas que serão executadas uma de cada vez, seguindo a ordem proposta pelo administrador.

Unix: é um termo usado para se referir a sistemas operacionais que utilizam o *kernel* desenvolvido pela empresa The Open Group.

Virtualização em nível de sistema operacional: é um tipo de virtualização cuja técnica permite que um sistema ou uma aplicação dele possa ser operado dentro de outro sistema operacional.

APÊNDICE A – Diferença entre Imagem e Contêiner

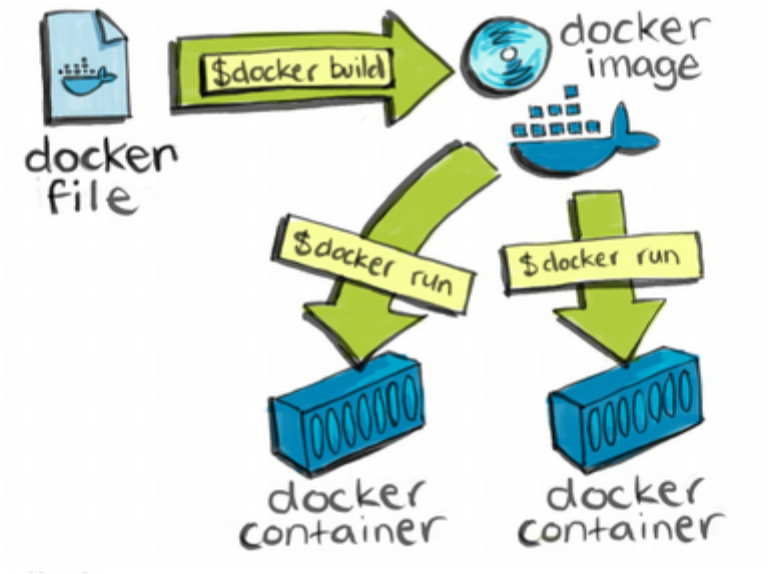

Figura 42 - Diferença entre imagem e contêiner. (TECNISYS, 2021)

O *dockerfile* na figura acima é um arquivo texto que tem todos os comandos para criar e executar um contêiner.

A primeira seta (\$*docker build*) indica que está sendo criado uma nova imagem no seu sistema operacional.

Imagem é uma representação estática do aplicativo ou do serviço e de sua configuração e dependências.

A segunda e a terceira seta (\$*docker run*) mostram que a imagem acima deles está executando os contêineres com todas as dependências necessárias.

Contêiner é uma imagem em execução na máquina onde está sendo usado o *Docker*.

APÊNDICE B – ARQUITETURA DO *DOCKER*

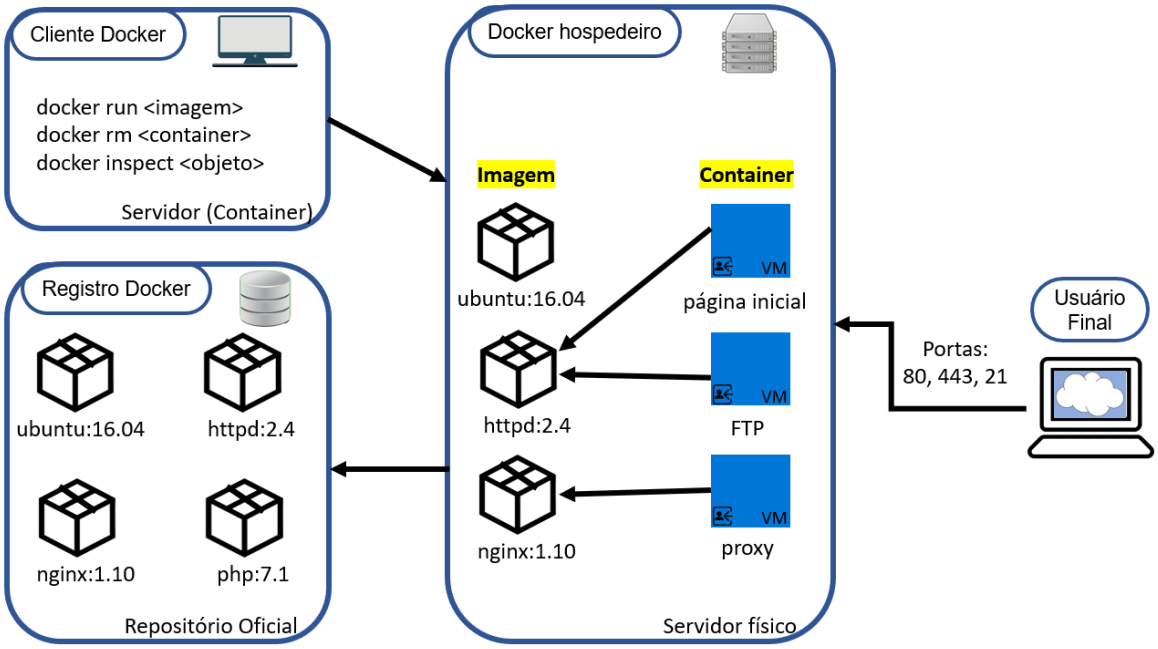

Figura 43 - Arquitetura do *Docker*.

O *Docker* trabalha com o cliente, que é o servidor virtual, ou melhor, é o próprio contêiner, ele que é responsável por executar, remover e inspecionar, ou seja, fornecer informações detalhadas sobre o *Docker*.

O cliente está diretamente conectado ao hospedeiro, ele quem vai requerer a solicitação da imagem ou objeto, irá buscar essa imagem no registro oficial, guardando em sua memória e por fim irá executar o contêiner.

O usuário final, através das portas de saída disponíveis, acessará o sistema criado naquele servidor.

APÊNDICE C – RESUMO *ANSIBLE*

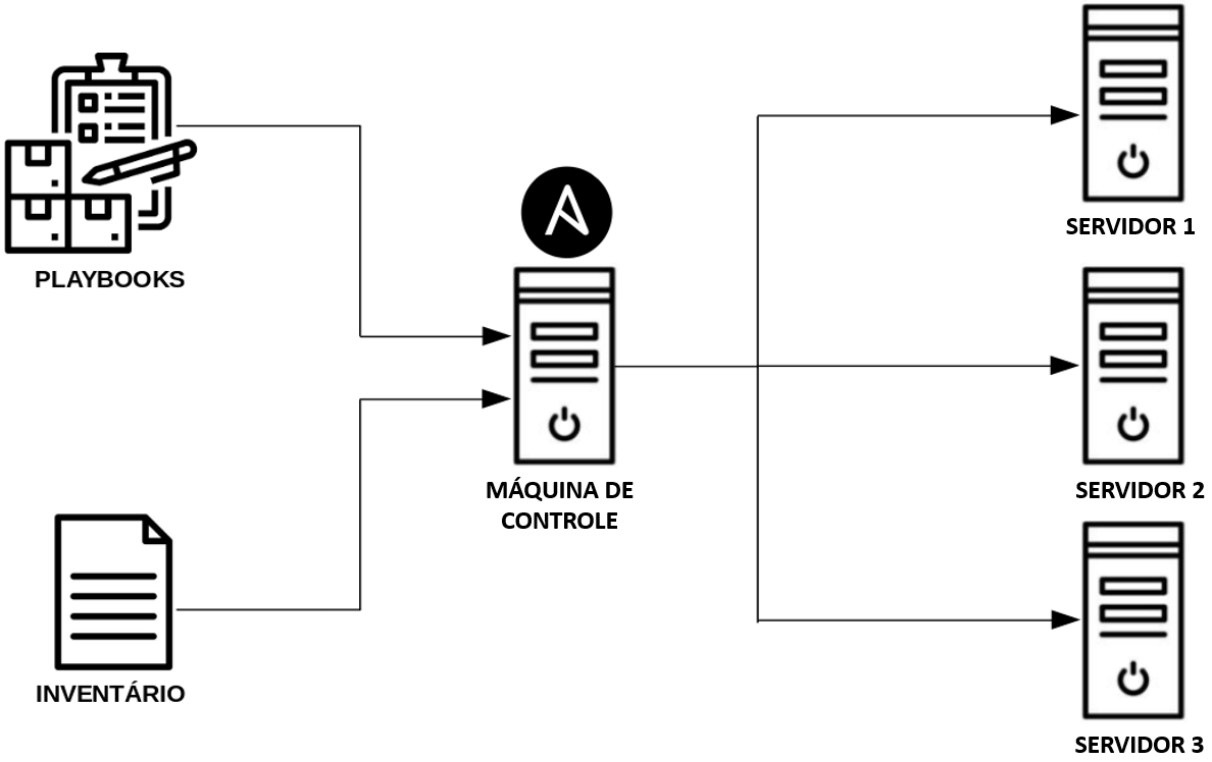

Figura 44 - Resumo *Ansible*. (adaptado de FARIAS, 2022)

O *Ansible* se utiliza de uma máquina de controle, a qual é responsável por configurar os servidores. Como o *Ansible* foi criado para trabalhar em grande escala, a máquina de controle precisa de um arquivo de inventário (Figura 45), onde tem todos os *IPs* dos servidores que serão configurados com a finalidade de se conectar através do *SSH*.

A figura abaixo retrata como é a estrutura de um arquivo de inventário, onde os endereços *IPs* podem estar sem nenhum grupo específico e quando solicitados é utilizada a nomenclatura descrita na imagem (*ungrouped*) ou com grupos que têm uma determinada característica em comum, e quando solicitados é usado o nome do grupo. Outra característica do inventário é recorrer às variáveis quando necessário, no exemplo abaixo foi declarado o tipo de conexão e a porta de saída a ser usada.

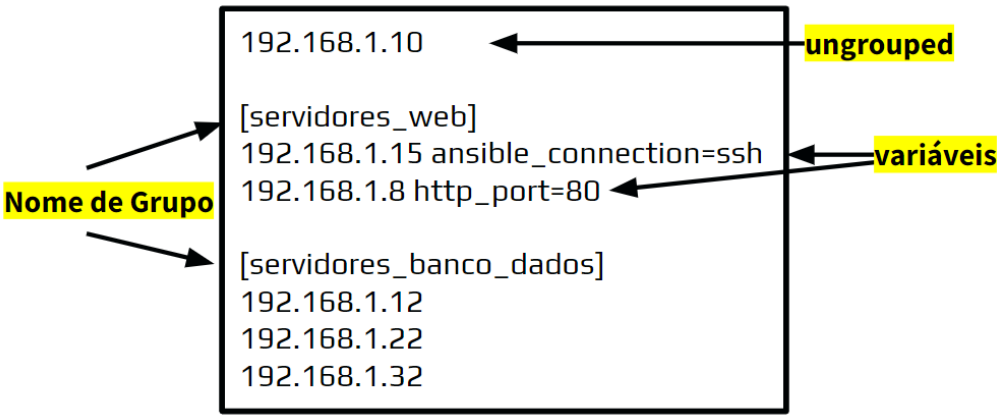

Figura 45 - Arquivo de inventário.

Outro arquivo importante para essa configuração é a *playbook* (Figura 32), a qual estará detalhando a configuração dos servidores através de tarefas a serem executadas na ordem de instrução. Este arquivo serve para documentar como foram configurados os servidores e também para evitar erros humanos quando há uma configuração em grande escala de servidores.

Portanto, com os dois arquivos (*playbook* e inventário) juntos a máquina de controle executa as tarefas nas máquinas a serem controladas (Figura 41).