UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Graduação em Ciência da Computação

Weked dos Anjos Curvel

Ensino de Redes de Computadores em Ambiente Virtual Baseado em Conteinerização de Aplicações

São Luís - MA

2022

Weked dos Anjos Curvel

Ensino de Redes de Computadores em Ambiente Virtual Baseado em Conteinerização de Aplicações

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Curso de Graduação em Ciência da Computação

Universidade Federal do Maranhão

Orientador: Mário Antonio Meireles Teixeira

Dr. em Ciência da Computação - UFMA

São Luís - MA

2022

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Curvel, Weked dos Anjos.
Ensino de Redes de Computadores em Ambiente Virtual Baseado em Conteinerização de Aplicações / Weked dos Anjos Curvel. - 2022.
72 f.

Orientador(a): Mário Antonio Meireles Teixeira.
Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís - MA, 2022.

1. Ambientes Virtuais de Ensino. 2. Docker. 3. Educação em Computação. 4. Redes de Computadores. I. Teixeira, Mário Antonio Meireles. II. Título.

Weked dos Anjos Curvel

Ensino de Redes de Computadores em Ambiente Virtual Baseado em Conteinerização de Aplicações

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. São Luís - MA, 11 de fevereiro de 2022:

Mário Antonio Meireles Teixeira
Dr. em Ciência da Computação - UFMA
Orientador
Universidade Federal do Maranhão

Antônio Abreu Batista Junior
Dr. em Ciência da Computação - UFABC

Rafael Fernandes Lopes
Dr. em Engenharia Elétrica - UFCG

São Luís - MA
2022

Dedicado aos meus Pais, por sempre acreditarem em mim.

Agradecimentos

Inicialmente gostaria de agradecer a Deus por ter me permitido experienciar muitos momentos importantes durante esta empreitada acadêmica, momentos bons e ruins em que pude aprender e crescer como pessoa, mesmo reconhecendo minhas próprias limitações.

A minha família que se manteve ao meu lado durante a árdua caminhada onde seus desejos sinceros de perseverança e motivação, me mantiveram resiliente. Um agradecimento especial para a minha companheira Anne que me acolheu nos momentos difíceis levantando meu astral, me dando o suporte necessário para continuar seguindo e que foi o porto seguro para onde eu sempre podia retornar.

Aos meus pais por serem minhas inspirações de vida, com os quais aprendi que a persistência dignifica o indivíduo. Por todo o carinho, motivação e principalmente por seus ensinamentos que moldaram meu caráter e me impulsionam a melhorar constantemente.

Ao Professor Mário Meireles agradeço toda a paciência, orientação e dedicação, suas instruções foram cruciais para a produção deste trabalho. Sua postura e conhecimento são inspiradores.

"História, nossas histórias
Dias de luta, dias de glória"

(Chorão - Charlie Brown Jr.)

Resumo

A consolidação do conhecimento na área de redes de computadores caminha ao lado das atividades práticas de fixação. Por esse motivo, as instituições de ensino necessitam construir e manter laboratórios que favoreçam tais metodologias. Contudo, os custos de implantação, manutenção e atualização destes ambientes podem ser altos e restritivos. Diversas ferramentas propõem a simulação de laboratórios virtuais de redes, possuindo a capacidade de reproduzir diversos dispositivos utilizando apenas um computador, entretanto demandam de complexas etapas de configuração. Este trabalho apresenta uma alternativa para a construção de ambientes virtuais voltados ao ensino de redes, fazendo uso da ferramenta Docker para gerar contêineres pré-configurados, capazes de emular o funcionamento de dispositivos computacionais. Além disso, é disponibilizado um repositório no GitHub contendo roteiros e laboratórios prontos para serem baixados e executados em qualquer computador com Docker.

Palavras-chave: Redes de computadores, Docker, ambientes virtuais de ensino, educação em computação.

Abstract

The consolidation of knowledge in the area of computer networks goes hand in hand with practical fixing activities. For this reason, educational institutions need to build and maintain laboratories that favor such methodologies. However, the costs of deployment, maintenance and updating of these environments can be high and restrictive. Several tools propose the simulation of virtual network laboratories, having the capacity to reproduce several devices using only one computer, however they demand complex configuration steps. This paper presents an alternative for the construction of virtual environments aimed at teaching networks, making use of the Docker tool to generate pre-configured containers capable of emulating the operation of computing devices. In addition, a repository is made available on GitHub containing scripts and labs ready to be downloaded and run on any computer with Docker.

Keywords: Computer networks, Docker, virtual teaching environments, computer science education.

Lista de ilustrações

Lista de abreviaturas e siglas

API	*Application Programming Interface*
AUFS	*Advanced multi-layered Unification FileSystem*
AVA	*Ambiente Virtual de Aprendizagem*
ELAI	*Agente Inteligente adaptável ao Nível de Expertise dos Estudantes*
HTTP	*Hypertext Transfer Protocol*
IA	*Inteligência Artificial*
IP	*Internet Protocol*
LXC	*Linux Containers*
NAT	*Network Address Translation*
PaaS	*Plataforma como um Serviço*
REST	*Representational State Transfer*
SELixux	*Security-Enhanced Linux*
SQL	*Structured Query Language*
TCP	*Transmission Control Protocol*
VM	*Virtual Machines*
VMM	*Virtual Machine Monitor*
WSL	*Windows Subsystem for Linux*
YAML	*YAML Is not Markup Language*

Sumário

1 Introdução

O cenário de ensino nas instituições do país enfrentam constantes dificuldades no que diz respeito aos cursos de redes de computadores, dificuldades às quais refletem diretamente nas disciplinas que compõem o círculo de conhecimento destes. A aplicação de exercícios de fixação baseados em aplicações imagéticas dos fundamentos abordados em sala de aula são, em muitos casos, um reflexo da ausência de recintos laboratoriais direcionados à prática do conhecimento adquirido. Apesar de valorosas, as atividades paliativas infelizmente tendem a produzir um aprendizado incompleto sobre os desafios que englobam os processos da comunicação digital e internet.

Por sua vez, quando o cenário anterior é observado do ponto de vista do ensino a distância, tais fatores são agravados, algo que decorre da maior abstração produzida por esse tipo de ambiente, levando em consideração que não há uma presença física demarcada por um professor ou monitor de disciplina capaz de orientar na etapa de aprendizagem do discente. Tanto neste cenário quanto no anterior, pode ser perceptível uma capacidade insuficiente ou inferior do processo de aquisição do conhecimento na área de redes de computadores por parte dos alunos submetidos a uma destas realidades.

Dessa forma, para que ocorra o desenvolvimento completo das habilidades técnicas fundamentais que complementam o eixo formativo dos estudantes de computação, bem como as noções requeridas para uma avaliação satisfatória das mesmas, as atividades práticas demonstram-se essenciais para a consolidação do conhecimento. Ainda que, em grande parte das situações os alunos não disponham de espaço físico devidamente equipado ou ainda por horários incompatíveis devido às altas demandas de alunos pertencentes a instituição de ensino (FERREIRA et al., 2013).

Ao longo do tempo, a execução de aulas práticas exercidas em laboratórios recebeu certa notoriedade. Para o modelo de ensino atual, a recorrência ao uso de tal mecanismo como recurso de ensino, e consequentemente, aprimoramento dos alunos de redes de computadores, tornou-se algo comum. As aulas produzidas em laboratórios adequados viabiliza a apresentação das principais problemáticas que envolvem a elaboração, correção e manutenção de uma rede computadores, assim como, apresenta ao discente o comportamento dos principais equipamentos utilizados, fornecendo assim experiências prévias valiosas a serem aplicadas futuramente em sua carreira (GARCIA et al., 2016).

Apesar de ideais, laboratórios direcionados à atividades específicas necessitam de diversas instalações para acomodar os equipamentos, além de possuírem altos custos para serem implementados e mantidos, portanto, as instituições tendem a manter laboratórios práticos de uso mais geral voltados às disciplinas de computação. Sendo assim, (CARISSIMI,

2008) pontua em seu trabalho "Virtualização: da teoria a soluções", que a virtualização seja uma alternativa viável para as práticas laboratoriais, capaz de contornar os principais problemas enfrentados por laboratórios de uso geral. Já que estes por sua vez tendem a limitar as capacidades administrativas ofertadas a um aluno, como a instalação de novos sistemas operacionais, acesso a configurações de sistema, particionamento de disco, entre outros recursos necessários para que haja um melhor entendimento das atribuições que um administrador de redes possui.

A virtualização de ambientes, por sua vez, permite a execução de diversos SOs, a realização de tarefas que exigem permissões administrativas irrestritas, bem como, as experimentações necessárias para que haja uma melhor fixação por parte do aluno. Tudo isso ocorre de forma isolada e sem comprometer o funcionamento do sistema preexistente na máquina laboratorial de uso geral.

De forma a propor um repositório contendo laboratórios práticos virtualizados e direcionados ao ensino e aprendizagem de redes de computadores, que destina-se tanto aos cursos de graduação, quanto aos tecnológicos e de nível técnico. Este trabalho faz uso de técnicas de virtualização baseadas em contêineres, levando em consideração as significativas vantagens oferecidas por este tipo de abordagem, em detrimento das perspectivas que fomentam o uso de ferramentas baseadas em hipervisores.

Este trabalho tem como foco o uso do Docker, sendo este por sua vez uma ferramenta livre para a criação e gestão de contêineres. Essa ferramenta é capaz de criar ambientes virtualizados portáteis propícios à execução de softwares, algo que assemelha-se a utilização de máquinas virtuais, possuindo também uma interface de comunicação baseada nos protocolos de rede TCP/IP.

O presente estudo não se detém a somente apresentar a ferramenta Docker como um gestor de contêineres funcional e versátil, mais que isso propõe-se a atestar a viabilidade da mesma como acessório para produção de laboratórios práticos voltados ao ensino nos cursos de redes de computadores, além de disponibilizar um repositório público no GitHub contendo utensílios acessíveis que possam propiciar a professores e alunos uma alternativa para ministrar e praticar suas aulas laboratoriais em qualquer lugar portando apenas uma computador com o Docker e internet, tal conjunto foi aqui denominado como NetEnsina Docker. O trabalho foi implementado utilizando Docker Containers, onde os ambientes experimentais foram criados usando como base uma imagem do Linux Ubuntu 18.04 alinhado a ferramentas direcionadas a inspeção e manutenção de redes do ambiente Linux.

A motivação que culminou na escolha de contêineres neste trabalho diz respeito a versatilidade que este tipo de tecnologia tem apresentado ao longo do tempo, sendo amplamente aplicada na virtualização de ambientes completos para a produção de aplicações e fornecimento de serviços, além de possuir nas suas versões mais recentes compatibilidade multiplataforma capaz de proporcionar a execução não somente em

sistemas baseados em UNIX, mas também no Windows através da ferramenta WSL 2, permitindo assim que os laboratórios possam ser utilizados em qualquer sistema operacional desde que haja conexão com a internet e o Docker instalado.

Os laboratórios virtuais são compostos por dois ou mais Docker Containers derivados de uma Docker Image contendo as configurações pré estabelecidas que simulam máquinas virtuais mais simplificadas contendo apenas os recursos necessários para que sejam efetuados os experimentos, e coordenados por um arquivo Docker Compose. Estes contêineres são baseados na distribuição Ubuntu do Linux, sendo capazes de receber interação por parte de um usuário através de seu terminal em modo texto. O grande diferencial destas unidades virtuais está no fato de possuírem uma inicialização rápida uma vez que o container tenha sido criado, e por se tratar de um imagem baseada em Linux, todos os comandos utilizados neste tipo de sistema podem ser executados nos laboratórios do NetEnsina Docker.

Além dos arquivos Dockerfile para a criação dos contêineres, os laboratórios foram produzidos de maneira a seguir uma estrutura de testes pré definidos. Sendo assim, foram concebidos roteiros didáticos de maneira que o conteúdo abordado possa ser melhor assimilado pelo estudante. Os roteiros permitem ainda que seja desnecessário o conhecimento prévio de Docker, onde o passo a passo necessário para executar os contêineres contendo as máquinas emuladas está contido. Dessa maneira, é possível criar toda uma estratégia de aprendizagem à medida que os conteúdos abordados pela disciplina vão avançando, propiciando uma dinamicidade entre teoria e prática no aprendizado de redes.

Este trabalho disponibiliza um repositório de dados no GitHub, com o intuito de organizar e servir os arquivos e roteiros para diversos tipos de laboratórios virtuais, que poderão ser utilizados nas práticas laboratoriais das disciplinas de redes de computadores. O acesso ao repositório pode ser feito de forma online por meio de computadores com acesso à Internet, e a execução dos laboratórios pode ser feita mediante a ferramenta Docker previamente instalada.

Pretende-se com a produção deste trabalho contribuir de maneira significativa a disseminação e acesso ao conhecimento de redes de computadores, atestando a eficácia da ferramenta Docker como um acessório valoroso para o cenário atual de ensino de computação, aumentando assim a acessibilidade à ferramentas de estudo outrora desconhecidas ou restritas.

Para uma melhor compreensão sobre o presente documento, o mesmo está organizado da maneira que se segue: o Capítulo 2 tratará da fundamentação teórica contendo os temas relevantes acerca do tema, onde inicialmente é abordado o ensino de redes de computadores, os conceitos por trás da conteinerização e demais assuntos pertinentes à implementação. No Capítulo 3 por sua vez, estão descritos os trabalhos com os quais este se relaciona. O Capítulo 4 ocupa-se em abordar a ferramenta Docker apresentando as

principais características utilizadas na produção deste trabalho. O Capítulo 5 apresenta o NetEnsina Docker, melhor detalhando seus componentes, funcionalidades e as vantagens de uso de seus resultados. Por fim, o Capítulo 6 traz as principais conclusões acerca deste trabalho, e discorre sobre as possibilidades para trabalhos futuros para a continuação da pesquisa. É possível ainda, encontrar anexados ao fim deste documento exemplos de modelos de laboratórios utilizando Docker.

2 Fundamentação Teórica

2.1 Ensino de Redes

Um estudo realizado no Instituto Federal do Rio Grande do Norte sobre a taxa de evasão dos alunos do curso de licenciatura em informática constatou que um pouco mais de 55% dos alunos abandonaram o curso no período que consistia de 2010 à 2013. Outro dado interessante é que da taxa total de evasão, aproximadamente 26% a quarta maior dentre as informações presentes na pesquisa, consistia daqueles discentes que não conseguiram acompanhar o conteúdo de disciplinas específicas do curso, como a de algoritmos. Um dado que atesta a necessidade da constante busca por inovações nos métodos de ensino, de maneira a suprir a dificuldade em capatar o interesse dos alunos mantendo-os motivados, além da investigação de didáticas que consigam auxiliar os problemas da absorção do conhecimento (MORAIS PAULEANY S.; JUNIOR, 2015).

Ao longo do tempo os ambientes educacionais, de um modo geral, tendem a constantemente buscar melhorias para seus conteúdos ofertados, onde, geralmente os professores são incentivados a buscar inovações em suas didáticas, de maneira que consigam ministrar aulas mais atrativas e produtivas. Para estes últimos há uma rotina familiar na pesquisa por métodos mais eficazes de atrair a atenção dos alunos. A revisão constante do planejamento didático, inserção de novas ferramentas ou inovações tecnológicas no meio de ensino, são exemplos recorrentes.

O momento atual vivenciado pelas salas de aulas, professores e alunos, é a de uma integração mais rápida e natural com as tecnologias e seus avanços, algo que corrobora para que os profissionais da educação tenham um incentivo a mais no momento de buscar compreender e integrar de forma mais consistente essas ferramentas às aulas ministradas.

A forma com que a tecnologia avança, se modifica e cada vez mais se estabelece nos diversos meios da sociedade, o que inclui também os ambientes de ensino. Aqueles diretamente envolvidos tendem a ter um comportamento cotidiano que se adapta ao meio em que se inserem (COUTINHO, 2006). Consequentemente, os profissionais expostos a um ambiente em constante modificação, serão levados a rotineiramente buscar compreender tais mecanismos, de maneira a obter a melhor utilidade das ferramentas tecnológicas existentes no momento em que se encontram (RICOY MARíA CARMEN;COUTO, 2011).

A necessidade por qualificação de profissionais capazes de produzir e reproduzir conhecimento no cenário da computação e suas tecnologias, culminou no surgimento de disciplinas, cursos de graduação e especialização direcionados a esse objetivo. De forma não necessariamente subsequente surgiram ramificações, onde assuntos mais específicos

do ensino de redes puderam ser agregados, cursos de redes de computadores, segurança e tecnologia da informação, são exemplos.

(FRANÇA; SILVA; AMARAL, 2013) em sua pesquisa "Despertando o interesse pela ciência da computação: Práticas na educação básica", pontuam que além da já constatada importância do ensino de informática e suas disciplinas relacionadas na esfera do ensino superior em cursos como o de ciência da computação, sistemas de informação, engenharia da computação e afins, também existem benefícios capazes de serem extraídos na aplicação de tais conhecimentos no ensino fundamental e médio. Os mesmos defendem ainda que, o ensino e aprendizado da Ciência da Computação, na Educação Básica, ainda se revelam por meio de ações embrionárias. Frisando que, práticas educacionais mais condizentes com o momento tecnológico e científico atual podem ser melhor alcançadas desde que ocorra com um trabalho de desenvolvimento, divulgação e principalmente de discussão sobre o assunto.

De acordo com (BEZERRA; SILVEIRA, 2011), as Licenciaturas em Computação têm em sua gênese a intenção de formar profissionais qualificados para atuar no ensino de Computação e Informática no nível médio, além de serem aptos a atuarem no segmento de mercado que envolve a Informática na Educação de maneira plena.

2.2 A Prática no Ensino de redes

O uso de atividades práticas como recurso de ensino no ambiente educacional acadêmico é uma realidade, e em grande parte dos casos torna-se uma componente essencial para a fixação do conteúdo teórico diante da complexidade e abstração que envolvem determinados temas. Para os cursos de tecnologia, como os de informática, há uma certa recorrência ao uso de tal artifício, dado que grande parte dos conceitos abordados requer uma implementação prática. Sujeito a estas condições encontra-se um dos ramos mais afetados, o ensino de Rede de Computadores.

A área de conhecimento que compreende Redes de Computadores engloba conceitos que são difíceis de serem compreendidos pelos alunos quando não são também abordados de maneira prática. Assuntos como: Segurança da informação, comunicação entre dispositivos, compartilhamento de recursos, tipos de topologias física e lógica são alguns dos exemplos. Para tanto necessita-se de ambiente capaz de prover uma estrutura ideal direcionada para que tais elementos de conhecimento possam ser melhor absorvidos, sendo nesse momento que os laboratórios práticos têm o seu real impacto sobre aprendizado.

Tais dificuldades são enfrentadas por outras disciplinas no exercício de sua produção de conhecimento. Contudo, em Redes de computadores, a implantação de ambiente prático se torna bem mais custosa. Isso decorre do fato de que, os recursos exigidos por essa disciplina não ficam delimitados ao uso de computadores como ferramentas práticas. Para

a implementação de um laboratório de redes, outros recursos de hardware computacional são requisitados, como: Hubs; switches; cabos de redes; pontos de acesso, roteadores, entre outros. Com tantas peças chave necessárias, os recursos financeiros demandados para a implementação e posteriormente manutenção desses ambientes são elevados, algo que nem sempre está disponível na maioria das instituições de ensino. Quando o cenário é observado através da ótica de uma instituição pública é possível que a situação se agrave, o que certamente pode ocasionar um comprometimento da qualidade do ensino ofertado, algo que por sua vez repercute na formação dos alunos.

Além dos custos envolvidos, os equipamentos mencionados são componentes físicos que à medida que a tecnologia avança e se reinventa, tais aparatos tendem a tornar-se obsoletos, necessitando de atualizações com o passar do tempo para que se mantenha o aprendizado nivelado com as inovações tecnológicas contemporâneas. Para contornar tais empecilhos, alguns professores têm utilizado como recurso alternativo para complementar a teoria que envolvem essas disciplinas, softwares específicos voltados a virtualização com o intuito de criar ambientes computacionais virtualizados, para que assim seus alunos também tenham acesso a aulas práticas. Este trabalho, por sua vez, busca propor uma alternativa para a prática laboratorial no ensino de Redes de computadores através de ambientes virtualizados fazendo o uso de contêineres como principal recurso de virtualização, procurando atestar a viabilidade da técnica para tal objetivo.

Para que haja uma melhor compreensão sobre onde os contêineres se encaixam nas técnicas de virtualização, a seguir alguns conceitos atrelados a este tema são abordados. Há ainda, uma seção dedicada a discorrer sobre os conceitos que cercam os contêineres.

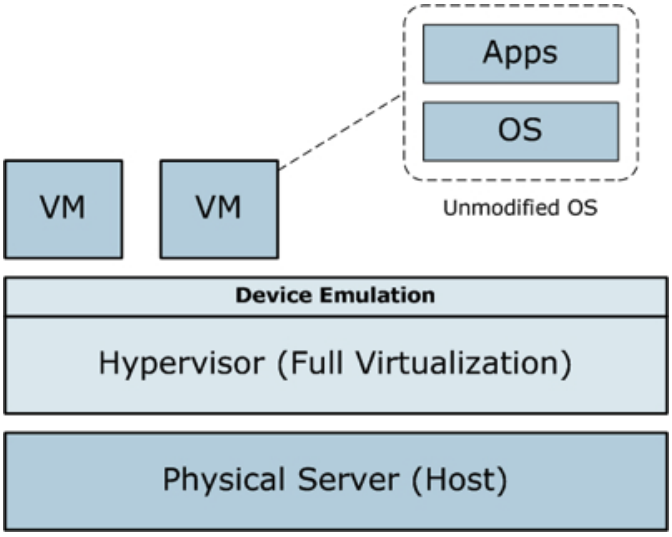
2.3 Virtualização Total

Na virtualização total um sistema operacional completo e sem alterações é executado em uma abstração de hardware subjacente a outro, algo que ocorre sobre uma camada de software responsável por monitorar as requisições, a VMM (*Virtual Machine Monitor*), ou ainda, hypervisor. Apesar de denotar uma técnica que traz os principais benefícios da virtualização, a exemplo de ambientes isolados e portáveis, a virtualização total possui alguns problemas atrelados.

A compatibilidade com diversos dispositivos é um desses problemas enfrentados pelo VMM, então o uso de dispositivos genéricos seria a principal maneira de contorná-lo. Em adição, o hypervisor deve ser o responsável por testar todas as instruções executadas pelo sistema operacional visitante antes de serem repassadas para o hardware, já que o mesmo funciona de maneira que o SO hospedeiro sequer tenha conhecimento de sua execução. O terceiro e principal problema, é gerado pela concorrência entre os sistemas operacionais virtualizados que originalmente não são projetados para este tipo de cenário e

então demandam recursos de paginação para funcionarem corretamente em seus ambientes, o que culmina em uma queda de desempenho das VMs (MATTOS, 2008). A Figura 1 apresenta a estrutura presente na virtualização total.

Figura 1 – Estrutura da Virtualização Total



Fonte: (JERSAK et al., 2014)

Devido à totalidade do trabalho despendido para capturar e efetuar a tradução das instruções vindas do sistema operacional virtualizado, esse seria o fator crucial para a redução de desempenho da técnica de virtualização total (JERSAK et al., 2014).

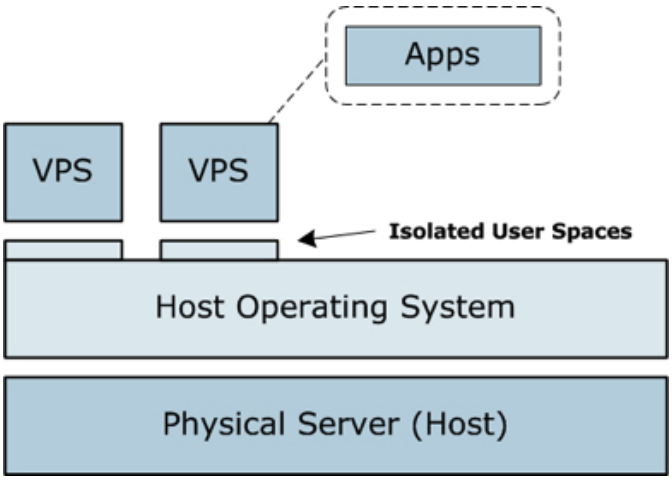
2.4 Paravirtualização

A paravirtualização é a abordagem alternativa à virtualização total. A ideia principal por trás desta técnica é principalmente a de contornar os principais problemas enfrentados na implementação de sua antecessora.

Neste modelo de virtualização, o sistema operacional é modificado de maneira a transferir suas requisições de execução de instruções diretamente ao VMM ou hypervisor. Deste modo, sempre que o SO visitante necessite executar uma operação que altere o estado atual do sistema, a mesma não precisa ser interpretada, traduzida ou testada pelo hipervisor, denotando um ganho significativo de desempenho em relação a virtualização total (MATTOS, 2008). A Figura 2 representa a estrutura da paravirtualização.

A fim de evitar problemas com os dispositivos, a paravirtualização elimina o uso de drivers genéricos, e implementa o uso de outros virtuais projetados para a própria máquina virtual. Desta maneira há uma exclusão da limitação de uso da capacidade total destes dispositivos nestes ambientes, onde há um distanciamento da forma de operação nativa do

Figura 2 – Estrutura da Paravirtualização



Fonte: (JERSAK et al., 2014)

SO hospedeiro, mesmo que mantenha-se um nível de semelhança próximo ao hardware físico (BERTOCHI; BELLEZI, 2016).

2.5 Conteinerização

2.5.1 Conceitos

O conceito de containers dentro da computação não é recente, tendo sido apresentado anteriormente na virtualização de sistemas. Contudo, a visão inicial sobre conteinerização de aplicações é apresentada com a introdução da ferramenta chroot no Unix, que posteriormente culmina no LXC (*Linux Containers*).

Apesar de contêineres constantemente serem confundidos com máquinas virtuais, os mesmos tratam-se de unidades mais compactas que agregam geralmente a aplicação a ser executada, bibliotecas e arquivos necessários para seu funcionamento. Além de não apresentarem um sistema operacional completo em seu interior, o que permite a sua independência em relação ao *host* e consequentemente maior portabilidade (RED HAT, 2020).

Há ainda dois conceitos distintos atrelados à conteinerização de aplicações. No que tange a desenvolvedores, se trata de um aglomerado que contém a aplicação, os recursos necessários para a sua correta execução, distribuível e facilmente instalável em diferentes ambientes. Já para o cenário que compreende a manipulação e operação de contêineres, são aplicações altamente portáveis que possuem o seu funcionamento em nível de *Kernel* com o sistema hospedeiro (JUNIOR; LIMA, 2015).

Os contêineres têm tido um papel fundamental na produção de novas ferramentas,

plataformas e serviços. Algo que tem aumentado constantemente após o surgimento e popularização do gerenciador de contêineres Docker, estando o mesmo presente nos principais sistemas operacionais do mercado, datacenters e plataformas de nuvem. Sendo já utilizado por diversos serviços de diferentes segmentos de mercado, a exemplo do PayPal, eBay, Spotify, entre outros. Na sequência, contêineres e o conceito de aplicações conteinerizadas são melhor explorados.

2.5.2 Containers

É verdade que a virtualização baseada em hipervisores trouxe grandes vantagens sobre a subutilização dos sistemas computacionais, proporcionando um melhor aproveitamento dos recursos de hardware disponibilizados por servidores com o intuito de fornecer serviços de maneira isolada entre si e com um desempenho satisfatório. Contudo, a evolução das tecnologias costuma cobrar mecanismos de segurança que consigam limitar cada vez mais o acesso de usuários e aplicações a recursos críticos, os mantendo em ambientes controlados e voltados às necessidades de suas demandas.

Com o intuito de contornar os problemas gerados pelo distribuição de recursos, no ano 2000 surge o FreeBSD Jail, que viabilizou a subdivisão de um sistema baseado em FreeBSD em diversos outros sistemas menores. As *Jails*, como foram denominadas essas divisões, eram ambientes seguros e destinados ao compartilhamento entre diversos usuários por intermédio de um administrador de sistemas, sendo capazes de limitar o acesso aos recursos da máquina que mantém o serviço de "celas" (RED HAT, 2018).

No ano seguinte, visando aprimorar o Linux com uma tecnologia que se assemelhava ao FreeBSD, surgiu o Vserver com a primícia de executar diversos servidores virtuais que forneciam, aos usuários e aplicações, espaços controlados, suficientemente seguros e isolados (LINUX VSERVER, 2013). Foi a partir da visão explorada no Vserver que posteriormente surgiria o conceito de virtualização baseada em contêineres.

De posse das formalizações geradas com o projeto de Jacques Gélinas (Vserver), outros desenvolvedores puderam agregar duas importantes ferramentas disponibilizadas pelo *kernel* Linux, o *cgroups* e *systemd*. Estas por sua vez viabilizaram a separação eficaz dos ambientes criados no sistema operacional, isso ocorre pois, uma é responsável por controlar e limitar o acesso aos recursos por parte de um processo ou grupo, e a outra é responsável por inicializar e configurar os espaços de usuário direcionados a estes mesmos processos.

- *Cgroups*: Basicamente trata-se de uma ferramenta fornecida pelo *kernel* Linux capaz de controlar e limitar o uso de recursos de sistema como tempo de CPU, memória, largura de banda, ou ainda um conjunto destes, por um dado processo ou grupo composto pelos mesmos.

- *Systemd*: Consiste em um conjunto de blocos básicos de construção para um sistema Linux, sendo assim, é capaz de iniciar todo o sistema a partir de si mesmo fornecendo ainda um gerenciador para o sistema.

Em 2002 o *kernel* Linux adicionou outra *feature* crucial para a tecnologia que mais tarde se converteria nos contêineres LXC, o *namespaces*. Este por sua vez funciona viabilizando a criação e manipulação de diferentes contextos no sistema, permitindo com que as características globais de cada um tornem-se isoladas entre si, ou seja, um novo ambiente de rede pode ser criado e este por sua vez é isolado do ambiente físico e dos demais contextos criados. Desta forma, o novo contexto possuirá novos endereços lógicos e físicos, além de que tráfego e regras de *firewall* que não podem ser enxergadas por nenhum outro contexto presente no sistema, o que resulta no isolamento do ambiente (EDUARDO, 2017).

Portanto, é com este recurso que a tecnologia alcança o conceito de container, e a partir do *kernel* Linux 2.6.26 de 2008 integra uma API para o gerenciamento de contêineres que resulta no LXC, que concede ao cenário de virtualização um novo nível (CANONICAL, 2017), agora sendo capaz de isolar os processos em unidades virtuais que fornecem todos o recursos necessários para o seu correto funcionamento sem a necessidade de um hipervisor intermediando as interações com o sistema, algo que assemelha-se ao FreeBSD Jail, contudo, possuindo maior segurança e isolamento entre os diferentes contêineres (JIANG; SONG, 2015).

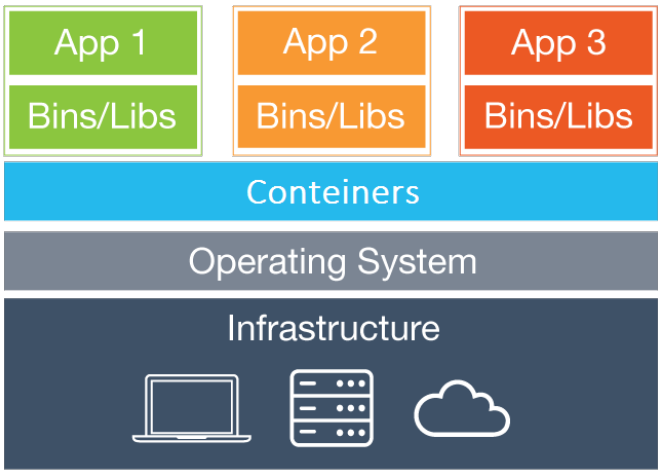
Por se tratar de uma tecnologia de virtualização, os Linux *containers* também possuem como uma de suas características principais a portabilidade. Desta maneira é possível utilizar um mesmo contêiner em diferentes hosts, desde que os mesmos possuam a tecnologia presente em seu *kernel*.

Em sequência a implementação do LXC no Linux tendo como finalidade complementar o uso de contêineres no sistema, surge o LXD. Este por sua vez, nada mais é do que um daemon, capaz de auxiliar na execução e gerenciamento dos contêineres. Além disso, quase simultaneamente ao surgimento do LXC, um novo projeto para gerência de contêineres e que inicialmente utiliza como base a própria ferramenta disponibilizada pelo Linux é apresentado ao público, o Docker.

2.5.2.1 Funcionamento

Sabendo que contêineres LXC nada mais são do que um tipo de virtualização executada a nível de SO, ou seja, é permitido pelo sistema que vários processos sejam instanciados de maneira isolada entre si através de ferramentas do *kernel* na mesma máquina. Dessa forma, segue o funcionamento de uma aplicação conteinerizada no Linux.

Figura 3 – Estrutura de virtualização com containers.



Fonte: (WE LOVE TEACH, 2016)

É possível executar um app de modo isolado em um contêiner a partir da ferramenta do *kernel*, *namespaces*, que como citado anteriormente, envolve a aplicação em um novo contexto que isola suas configurações globais do restante do sistema. De modo a definir um nível a mais de isolamento para o contexto em que se encontra a aplicação. É feito o uso pelo gerenciador de contêineres o *chroot*, de modo a evitar que conteúdos fora do container sejam acessados de forma indevida (MACHADO, 2017).

Em sequência, para que o contêiner não tenha acesso de maneira desregrada aos recursos fornecidos pelo sistema, é aplicado o *cgroups*. Dessa forma, o processo é alocado em um grupo com uma hierarquia baseada no tipo de recurso consumido pelo mesmo. Desta maneira os recursos podem ser limitados aos grupos e melhor monitorados pelo sistema.

Por fim o SELinux entra em ação executando o gerenciamento das autorizações de aplicações, fazendo a verificação a cada chamada de sistema e executando as restrições de acordo com o usuário e domínio. O *apparmor* é executado em simultâneo, o mesmo pode aplicar restrições à uma aplicação sem levar em consideração o usuário que a iniciou, portanto, essa é uma garantia a mais para que as aplicações sigam as configurações de recursos impostas à seus respectivos grupos (MACHADO, 2017).

2.5.3 Contêineres e Virtualização

Contêineres e VMs são ambos tomados como tecnologias de virtualização, pois seu objetivo final é reproduzir o comportamento de um ambiente computacional que propicie a execução de tarefas inerentes a este tipo de sistema, possuindo claras diferenças entre suas implementações.

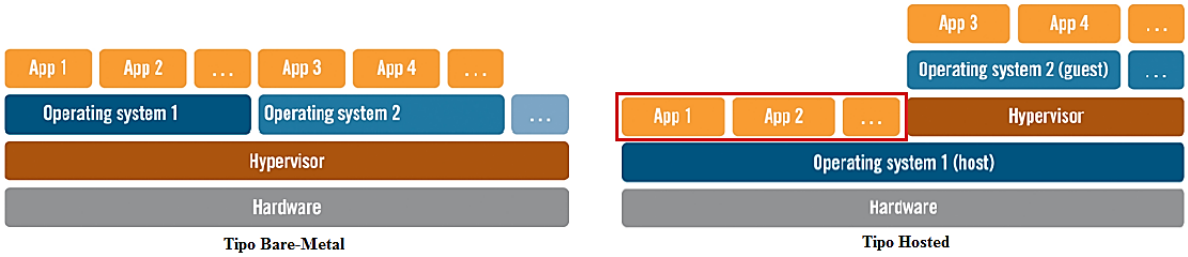
A virtualização, quando utilizada, permite uma melhor otimização de uso do

hardware computacional, onde os recursos do dispositivo são repartidos entre unidades virtuais menores de forma a suprir as necessidades de cada uma delas. Estas unidades são comumente denominadas na literatura de máquinas virtuais, ou somente VMs. Estas VMs tem como objetivo ofertar um ambiente que possa transparecer características presentes em dispositivos físicos, viabilizando a instalação de sistemas operacionais completos e distintos entres as unidades virtuais, permitindo também recursos de instalação de softwares e ofertando serviços de rede (CARISSIMI, 2008).

Virtualização baseada em VM tem como principal característica a reprodução de um sistema computacional completo, ou seja, o conceito que serve de alicerce para a tecnologia é a emulação do hardware para que o software seja executado posteriormente em uma nova camada. Nesta técnica o hardware do host é abstraído e um novo é definido possuindo quantidade de memória, armazenamento, interfaces de rede e etc. A abstração por sua vez é executada por uma camada de software denominada hipervisor instalado no sistema hospedeiro. Levando em consideração o fato anterior, existem duas implementações distintas de virtualização baseada em hipervisores (MACEDO; SANTOS, 2014):

- Virtualização tipo 1 - *Bare-metal*: Trata-se da virtualização onde o hipervisor é instalado diretamente no hardware do hospedeiro, tendo suas interações levadas diretamente à máquina física, o que torna este tipo de técnica independente de um sistema operacional, sendo gerida única e exclusivamente pelo hipervisor.
- Virtualização tipo 2 - *Hosted*: Na virtualização do tipo 2, o hipervisor é instalado sob um sistema operacional, executando assim uma camada de abstração entre o SO e as VMs. Tomando por base que o sistema operacional do hospedeiro continua funcionando e gerenciando os processos gerados por sua camada de aplicação, é possível em virtualização do tipo *hosted* a troca de informação entre as máquinas emuladas e o SO hospedeiro através desta camada de aplicação.

Figura 4 – Tipos de Virtualização baseada em Hipervisores.



(*a*) (*b*)

(a) Estrutura de hipervisor do tipo bare-metal (b) Estrutura de hipervisor do tipo hosted Fonte: (MACEDO; SANTOS, 2014).

Apresentados anteriormente neste trabalho, os contêineres tratam-se de unidades virtuais capazes de reunir todos os recursos necessários para o funcionamento de um serviço ou processo, podendo ser estes, seus códigos, bibliotecas, dependências e recursos de desenvolvimento, por exemplo. E como o mesmo funciona junto ao sistema operacional hospedeiro se servindo das suas funções de *kernel*, não há necessidade de criação de uma nova instância de servidor ou um serviço direcionado ao seu propósito, algo que teoricamente funciona como um mecanismo de encaixe, não prejudicando qualquer outra tarefa executada pelo hospedeiro.

Durante a execução de um contêiner o que se faz é destacar recursos do *host* através das ferramentas incorporadas como o *cgroups* citado anteriormente, e faz uso das bibliotecas de *kernel* utilizadas pelos ambientes executados no interior do container. Como todo o recurso necessário já está presente no contêiner - como bibliotecas e códigos em versões específicas - a portabilidade dos mesmos torna-se mais prática (ZHUANG et al., 2017).

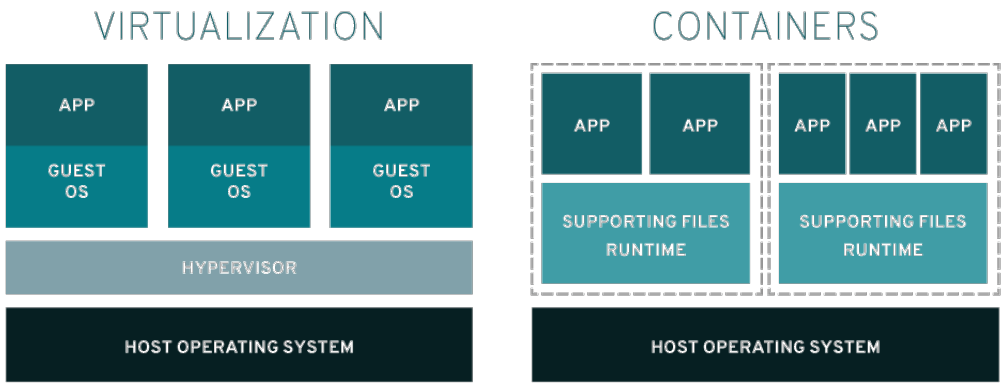
O autor (JUNIOR; LIMA, 2015) apresenta dois pontos de vista que se destacam quando contêineres são a ferramenta de estudo, a visão compartilhada pelos desenvolvedores e a compartilhada pelos administradores de sistemas:

- Para os desenvolvedores: Dado um ambiente executado em um container funcional e tendo como fato que estes por sua vez são itens isolados entre si, são pertinentes os testes com outras versões de bibliotecas e dependências eximindo uma real preocupação com a "quebra" da estabilidade da aplicação original. Contudo, tendo em vista que um ambiente é totalmente funcional com os recursos que já possui, o mesmo irá sempre funcionar em qualquer situação, ainda que seja transposto de um dispositivo para outro. Não haverá a necessidade de preocupações com dependências perdidas, pacotes ou outros problemas recorrentes no processo de implantação de uma aplicação ou serviço. Isso é possível graças à característica de apenas leitura dos contêineres, o que consiste em imagens leves e portáteis, possuindo um inicialização rápida e capacidade de execução em qualquer plataforma que possui *Kernel* Linux.
- Para os administradores: Elimina os riscos produzidos por VMs como os custos, implantação, portabilidade e performance, sendo por sua vez ambientes mais consistentes se levadas em consideração as etapas de produção de um produto de software, desenvolvimento, teste, produção e ambiente do cliente. Sendo itens que interagem diretamente com o *Kernel* do sistema, eliminam a necessidade de virtualizar um sistema completo, tornando a tecnologia mais leve e reproduzível em qualquer dispositivo, viabilizando, ao processo de desenvolvimento, mais eficiência.

Contêineres, no que diz respeito à implementação e implantação em um ambiente produtivo, leva uma série de vantagens, e se destaca das máquinas virtuais. A principal

diferença certamente está na estabilidade e facilidade de uso da tecnologia, onde uma série de configurações complexas não são requeridas, além de fornecer um desempenho superior e menos custoso em relação à tecnologia de virtualização baseada em hipervisor.

Figura 5 – Estrutura da Virtualização baseada em hipervisores e contêineres.



(*a*) (*b*)

(a) Possui uma camada de software intermediaria entre o sistema host e as maquinas virtuais, por onde são feitas as requisições. (b) Armazenam microserviços ou aplicações, agrupadas com todos os recursos necessarios para seu funcionamento. Fonte: (RED HAT, 2020)

Contudo, apesar de um contêiner não ser semelhante a uma máquina VM por diversos motivos (Figura 5), a mesma não se trata de uma técnica de virtualização antagonista e sim complementar, podendo ser utilizada em conjunto com as VMs que por possuírem SOs completos sendo executados, podem facilmente executar contêineres a fim de produzir um maior nível de segurança.

2.5.4 Aplicações Conteneirizadas

Uma aplicação conteinerizada de maneira mais geral, trata-se de uma peça de software aglutinada em uma unidade portátil contendo código, bibliotecas e dependências, possuindo capacidades plenas para a sua reprodução em outro ambiente computacional propício à execução de contêineres (FERNANDES, 2018).

As imagens são apresentadas na seção 4.3, elas desempenham um papel crucial em tecnologias de virtualização que baseiam seus contêineres em imagens. São as imagens que carregam a característica de partilha de informações acerca de uma aplicação ou serviço no ambiente Docker, por exemplo. Portanto, é partindo de uma imagem que se obtém o contêiner de uma, ou um conjunto de aplicações. Seja um servidor *web*, um banco de dados, entre outros (BICCA, 2015).

O objetivo por trás deste conceito é o da simplificação do meio de execução do processo, já que em um contêiner é possível trocar informações e recursos utilizando o

mínimo de um sistema operacional funcional, algo que se mostra vantajoso em relação a implementações semelhantes utilizando VMs.

Aplicativos conteinerizados são melhores utilizados quando empregados como micro serviços, ou ainda, como aplicações distribuídas. Isso é verdade pois, um rápido exercício mental comprova a capacidade de isolamento dos contêineres, já que cada agrupamento de aplicações consegue funcionar de forma independente das outras (LEADCOMM, 2020). Caso haja necessidade de comunicação entre as unidades, a própria tecnologia de contêineres deve ser capaz de oferecer interfaces próprias que se comportam como meio de propagação dos dados, sendo um bom exemplo a interface de rede nativa do gerenciador de containers Docker, melhor abordado posteriormente neste trabalho.

Mesmo sendo muito semelhantes às máquinas virtuais em alguns aspectos, os contêineres de aplicações conseguem apresentar grandes vantagens em relação a VM 's levando em consideração sua implementação. A economia de recursos do host com certeza é um dos mais notáveis, algo que pode ser melhor percebido à medida que o número de unidades aumenta e a gestão de recursos consegue manter ótimo funcionamento na máquina hospedeira, proporcionando a todos os serviços igual operação entre si, algo que geralmente não acontece com o uso de VM's (GAO et al., 2017).

Um aplicativo conteinerizado não é algo imutável definido em um único momento de sua criação. Estando baseada em imagens como o Docker é possível sim que um outro desenvolvedor percebendo a necessidade de acréscimo de uma nova ferramenta ou biblioteca, possa incluí-la. A imagem é tão somente uma base para um serviço a ser implementado, podendo ser facilmente modificada no ato da criação do container de aplicação. O que testifica a característica de flexibilidade das aplicações conteinerizadas.

3 Trabalhos Relacionados

3.1 Ferramentas de interação em ambientes educacionais mediados por computador

o O trabalho produzido por (PRIMO, 2001) apresenta em sua totalidade uma visão analítica sobre o aspecto da importâcia da interação para a produção de conhecimento em um indivíduo, sobretudo no aprendizado derivado do uso de tecnologias como e-mail, lista de discussão, chat, ICQ, fórum, site, dentre outras ferramentas passíveis de aplicação ao ensino mediado por dispositivos computacionais em relação à classificação das interações, que podem ser do tipo mútua e reativa.

Inicialmente o texto apresenta uma discussão hipotética entre alguém que defende o ensino praticado em ambientes online e um funcionário do Instituto Universal Brasileiro. O suposto diálogo gira em torno dos benefícios alcançados pela informática no ramo do ensino a distância, onde o funcionário rebatia as informações apresentadas com relatos que atestam a eficácia da companhia para a qual trabalha em fornecer os mesmos serviços sem a necessidade de um meio digital para tal. A ideia em um primeiro momento do texto é denotada através da própria sugestão do autor, promover uma "polêmica inicial". Concluindo assim, que as tecnologias devem agregar aos métodos de ensino, algo que é atestado no trecho: "uma prática educacional antiga e um método ultrapassado com nova roupagem: uma 'anciã maquiada'." (PRIMO, 2001)

Apesar da importância do trecho anterior para a contextualização da ideia geral buscada pelo autor, os conceitos que são apresentados na sequência, de forma a conectar-se com a narrativa, embasam a discussão apresentada ao longo de todo o texto: conhecimento e interação. O autor (PRIMO, 2001) toma os conceitos de interação de Piaget para formalizar a relação do indivíduo com o conhecimento mediado por ferramentas computacionais, conceitos estes que valorizam a interação entre sujeito e objeto. Contudo, é importante atentar-se que a produção de conhecimento piagetiana decorre do fato que o sujeito afeta o objeto e vice-versa.

A partir da classificação das interações que ocorrem em sequência, o autor se propõe a analisar as ferramentas citadas, definindo-as como sendo de um dos dois tipos (mútua ou reativa). Em certas situações o mesmo comenta sobre aplicabilidades que favoreçam o ensino demarcando usos que agregam pouco ou nada a este cenário, seja nos métodos a distância ou tradicionais. O autor (PRIMO, 2001) enfatiza o fato de que alunos e professores deslumbram-se com os determinismos de ferramentas que fomentam apenas o clique e linearidade das interações.

Ao fim conclui-se que, não há a intenção de se subestimar ou mesmo deixar de lado o potencial oferecido pelo aprendizado que é produzido na interação reativa, ou seja, o conhecimento contido em ferramentas que não se modificam com a ação sofrida por um indivíduo, como um artigo presente em uma página web sobre um assunto relevante ao usuário, por exemplo. O que é pretendido pelo autor, é a valorização das interações mútuas, que prezam pela cooperação e dialógica, onde, usuários interagem com os conteúdos e entre si. Logo, a combinação de sites ricos em informações e ferramentas para o debate entre os participantes podem motivar ambientes férteis para a construção do conhecimento através da interação (PRIMO, 2001). A construção do conhecimento a partir de uma interação rica, é objetivada por este trabalho que permeia o auxílio de professores e alunos como ferramenta de ensino.

3.2 Ambiente Virtual Imersivo para ensino em Redes de Computadores: uma proposta usando Agentes Inteligentes

O trabalho produzido por (HERPICH et al., 2014) apresenta um ambiente de aprendizado virtual imersivo e interativo para a prática de Redes de Computadores assistida por uma inteligência artificial (IA). O conceito é permitir que os alunos que se utilizem da ferramenta, possam absorver os conteúdos de uma forma mais natural através dos cenários pensados para o ambiente.

A modelagem do projeto por sua vez, consiste na programação e implantação de um agente inteligente denominado ELAI (Agente Inteligente adaptável ao Nível de Expertise dos Estudantes), desenvolvido para se adaptar ao conhecimento prévio do usuário, que por sua vez entra em contato com com a IA (Figura 6), através de uma representação sua baseada em avatares. Desta maneira, o agente inteligente ofertará ao aluno conteúdos personalizados e condizentes com suas habilidades em redes.

O texto de (HERPICH et al., 2014) agrega no que diz respeito ao conceito de continuidade e evolução das Tecnologias de Informação e Comunicação (TICs) aplicadas ao ambiente educacional, que tendem a possuir novos requisitos e aplicabilidades ao longo do tempo, onde os ambientes virtuais imersivos aparecem como uma nova tendência das pesquisas voltadas ao aprendizado de indivíduos através de meios digitais interativos.

A pesquisa citada concluiu que, diante do provimento de diferentes recursos aos estudantes, foi possível observar como resultado preliminar a potencialidade existente no emprego de agentes inteligentes na educação juntamente aos ambientes virtuais imersivos. Agregado a isso, está o fato destes serem cientes dos contextos de seus usuários, tornando possível a adaptação de conteúdos, materiais e atividades no MV (Mundos Virtuais) às preferências dos estudantes (HERPICH et al., 2014).

Figura 6 – Usuário Interagindo com ELAI.

Fonte: (HERPICH et al., 2014)

3.3 Utilização de ambientes virtualizados para ensino de servidores de redes de computadores

O trabalho de (GARCIA et al., 2016) procura estudar a viabilidade da produção e implementação de ambientes virtuais com o intuito de fomentar a atividade prática laboratorial direcionada ao ensino de redes, capazes de suprir a demanda das necessidades enfrentadas pelas instituições de ensino com laboratórios compartilhados por diversos alunos e disciplinas, de modo a não perturbar a estrutura física destes recintos. O mesmo leva em consideração ainda, a importância de tal etapa para o eixo formativo de profissionais capacitados na área de redes de computadores.

As aulas práticas devem colocar o aprendiz em contato com eventos reais que ocorrem no mundo das redes, permitindo que ele vivencie experiências que lhe possibilitem passar de mero espectador como, por exemplo, um aluno que apenas assiste às aulas (ensino baseado em simples transmissão de informações), para um participante colaborador que experimenta, delimita o problema, e através da convivência com a situação real, raciocina, deduz, interage com colegas, com o material instrucional e com o professor e transforma-se de assistente a agente motriz do seu próprio aprendizado (GARCIA et al., 2016).

Para validar o trabalho em questão, os pesquisadores envolvidos produziram dois testes distintos elaborados em cursos voltados ao ensino de Redes de Computadores, sendo o primeiro voltado ao ensino de um servidor de *proxy*, e o segundo de um servidor *web*. Ambos foram realizados em instituições distintas, também utilizando equipamentos distintos em cada situação. Para a implementação dos laboratórios nos dois cenários, fizeram o uso da ferramenta de virtualização baseada em hipervisor, VirtualBox.

Atualmente a técnica de virtualização vem ganhando destaque na Tecnologia da Informação (TI) sendo muito utilizada em servidores de redes, desta forma aproveitando

melhor os recursos da máquina disponível. Porém sua utilização não se limita a este ambiente, pois as máquinas virtuais podem ser utilizadas em diversos setores das organizações, até mesmo por usuários domésticos e universidades em ambientes de testes (GARCIA et al., 2016). A pesquisa pode concluir que, a utilização de máquinas virtuais se mostrou muito eficaz nos testes realizados, pois atendeu todas as necessidades de configuração e instalação de sistemas operacionais que possibilitasse o objetivo do curso (GARCIA et al., 2016).

Semelhante ao trabalho produzido por (GARCIA et al., 2016), o presente trabalho se propõe a apresentar uma solução baseada em virtualização para a produção de laboratórios interativos para o aprendizado de Redes de Computadores. Contudo, utiliza virtualização baseada em contêineres para obter as vantagens atreladas a este tipo de técnica, fato este responsável pela escolha das tecnologias empregadas nesta pesquisa.

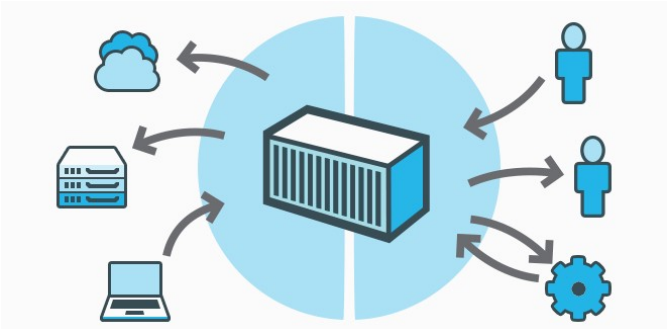
4 O Gerenciador de Containers Docker

Em 2013 uma apresentação promovida pela companhia dotCloud realizada na Python Developers no estado da Califórnia, mais especificamente na cidade de Santa Clara e tendo como palestrante o CEO e fundador da empresa Solomon Hykes, compartilharam em 15 de março o conceito inicial do Docker com a comunidade mundial de tecnologia (JUNIOR; LIMA, 2015).

A principal motivação da dotCloud para a criação da ferramenta era uma melhor gestão das plataformas de serviços ofertados pela empresa, em contrapartida levando em consideração a base tecnológica que incorporava o Docker, a tecnologia LXC proporcionaria a substituição das máquinas virtuais até então utilizadas, por contêineres. Como reflexo, haveria uma melhora das condições para os desenvolvedores executarem tarefas mais direcionadas a cada aplicação no processo de produção de um artefato de software, e também no processo de implantação, tornando-o menos trabalhoso, rápido e simples (VITALINO; CASTRO, 2016).

O Docker trata-se de uma ferramenta para gerenciamento de unidades virtuais com capacidade de agrupar o itens necessários para o desenvolvimento e replicação de aplicações em diversos ambientes, facilitando os esforços no âmbito da produção de software. Essas características culminaram em um alto interesse da comunidade de tecnologia, algo que posteriormente foi respondido com a conversão do projeto da dotCloud em open source, onde toda uma comunidade dedicada à ferramenta, contribui e desenvolve-se em seu entorno, produzindo novas features, e ou corrigindo bugs (RED HAT, 2018).

Figura 7 – Docker - Visão Geral



Fonte: (GOMES, 2015)

O Docker proporciona uma quantidade diversa de vantagens em sua utilização. Em uma primeira observação é possível notar um baixo uso dos recursos disponibilizados pelo sistema no qual a ferramenta é utilizada, algo que para as VMs, onde ocorre a virtualização de sistemas operacionais completos, representa um enorme ganho de desempenho ao passo

que culmina em maior oferta de serviços em um mesmo dispositivo. Esta última, por sua vez, é possível graças à forma de isolamento produzido pelos contêineres, que ocorre em nível de processo (GAO et al., 2017). Além disso, duas outras características são muito importantes para todo o conceito de funcionamento da ferramenta, são elas (JUNIOR; LIMA, 2015):

- Portabilidade: é capacidade de executar uma determinada aplicação anteriormente produzida com as ferramentas de conteinerização oferecidas pelo Docker, em qualquer *host* que também possua a plataforma.
- Previsibilidade: decorre do fato que as interfaces proporcionadas pelo Docker são padronizadas e as interações com *host* são previsíveis, o que exime o host docker do gerenciamento do conteúdo executado no interior dos contêineres, ou mesmo do conhecimento sobre qual unidade virtualizada está executando em um dado momento, que imprime ao *host* uma visão sob os containeres apenas a nível de processo.

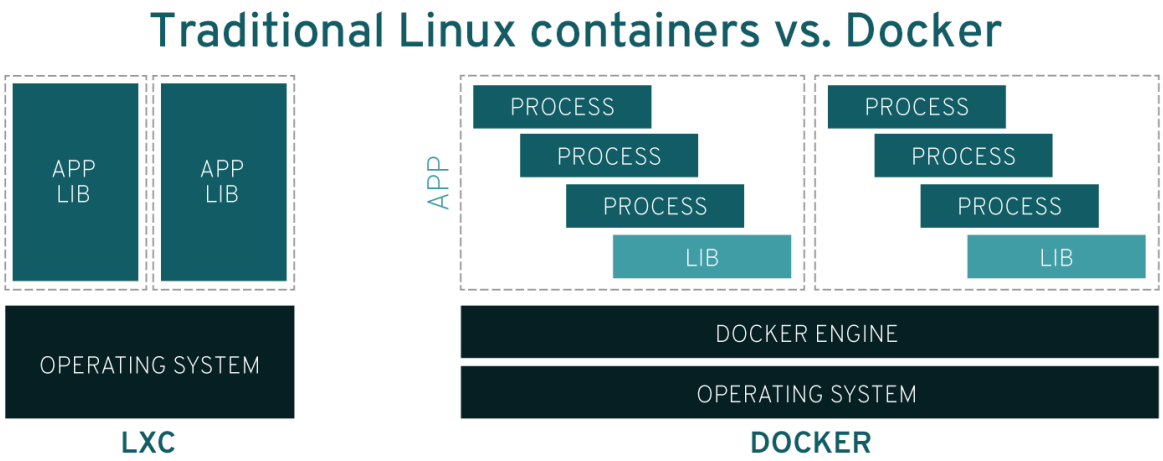
Levando em consideração que o intuito inicial da criação do Docker seria um aumento significativo na facilidade com a qual fornecia-se serviços. Atualmente implementar uma PaaS (*Platform as a Service*) utilizando a plataforma alcançou este objetivo fundamental, já que com poucas etapas de configuração é possível criar por exemplo, um servidor *web* com Apache ou mesmo com Nginx, conectado a um serviço de armazenamento que faz uso de um banco de dados SQL. Ambientes padronizados como os citados, podem ser facilmente importados da nuvem pública do docker conhecida como Docker *Hub*, que armazena uma grande quantidade de imagens contendo os mais diversos recursos (DOCKER INC., 2021).

No momento da escrita deste documento a ferramenta de gestão de containers Docker pode ser executada apenas em hosts com arquitetura de 64 bits e que possuem kernel Linux, são exemplos de sistemas com suporte ao Docker: o Linux, o MacOs e Windows. Este último por sua vez, ganhou suporte por volta de 2019 quando a Microsoft incluiu à build do Windows 10 o WSL 2, que trata-se de virtualização alinhada a um *kernel* linux para fornecer uma compatibilidade com a execução de binários da plataforma.

4.1 Características

Como citado anteriormente, o Docker se trata de uma ferramenta capaz de gerenciar unidades virtuais contendo um aglomerado de informações necessárias para o funcionamento de uma aplicação ou serviço. Sendo assim, quando esta tecnologia ainda dava seus “primeiros passos”, de fato a implementação por trás dos contêineres se tratavam diretamente do LXC presente no Linux. Com a expansão do projeto a tecnologia sofisticou-se e tornou-se totalmente independente (JUNIOR; LIMA, 2015).

Figura 8 – Conteineres Linux - Containeres Docker



Fonte: (RED HAT, 2021)

Atualmente os contêineres em Docker possuem uma divergência muito clara no funcionamento em relação ao LXC tradicional, que tende a interpretar uma aplicação inteira como um único processo, enquanto que o sistema de gerência do Docker tende a interpretar a aplicação como vários processos separados, fornecendo ferramentas para tal objetivo. A Figura 8 ilustra tal comportamento.

Do ponto de vista de desempenho, os LXC 's apresentam desempenho satisfatório nos cenários em que as aplicações são mais leves. Contudo, em demandas mais robustas e de desenvolvimento, por exemplo, a abordagem de execução granular tende a apresentar uma maior vantagem. É importante frisar, que para o *host*, um contêiner no final será apenas um processo (RED HAT, 2021).

Ainda sobre o cenário de desenvolvimento, o Docker destaca-se por sua característica de portabilidade mencionada na Seção 4, que consiste no "transporte" de uma dada aplicação ou serviço que opera de forma completa e funcional de *host* para outro (JUNIOR; LIMA, 2015). Este transporte não ocorre com o uso direto dos contêineres, mas sim, através das imagens.

As imagens Docker são construções que empacotam um conjunto de informações previamente adquiridas e agregadas a um arquivo de imagem. Portanto, é a partir de uma imagem que o Docker criará e posteriormente executará um container fazendo uso destas informações. O Docker *Hub* é um repositório público mantido pela companhia que faz a gestão da ferramenta. Ele é responsável por armazenar uma grande quantidade de imagens prontas contendo desde apenas sistemas de arquivos base, até estruturas mais complexas que culminam em um serviço, como o já citado Nginx (DOCKER INC., 2021).

Capaz de executar um bom isolamento entre *host* e aplicações, os Docker *containers* se assemelham bastante ao funcionamento das VMs, contudo, sendo artefatos mais leves e

portáteis. Além disso, o Docker possui a capacidade de adicionar ou remover funcionalidades dos contêineres apenas recompilando a imagem com a característica desejada, algo que se assemelha bastante às ferramentas de controle de versão.

Em resumo, o Docker oferece uma experiência inovadora ao desenvolvimento de produtos de software, tendo capacidade suficiente para executar aplicações robustas, algo crucial no fornecimento de serviços.

4.2 Estrutura

O Docker nasceu com o intuito de atender a necessidade de gerenciar a implantação de aplicativos e serviços de maneira distribuída e isolada entre si. Sendo assim, a plataforma alcançou seus objetivos fomentando um cenário onde a conteinerização tornou-se mais simples, previsível e robusta.

Os contêineres produzidos por esta ferramenta são artefatos de execução leve, padronizados e isolados do *host*, algo que é possível devido ao funcionamento dos contêineres, que de um modo geral empacotam componentes, aplicações e suas dependências em um mesmo ambiente. Tudo isso permite que um ecossistema utilizando Docker para serviços, e ou, aplicações seja capaz de resistir às falhas de máquina ou processos.

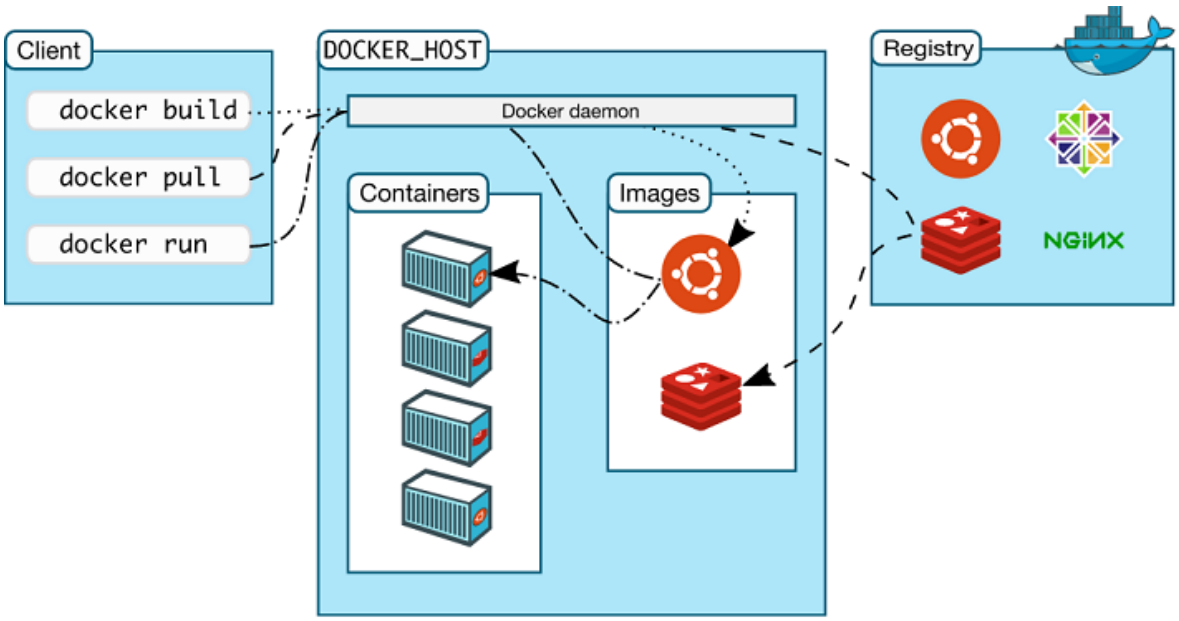
A ferramenta apresenta características muito semelhantes ao que se pretende alcançar quando se faz uso de VMs. Contudo, estas geram uma série de problemas contornados pelo Docker, como o uso de espaço em disco, a facilidade de manutenção, entre outros. Apesar de no funcionamento se apresentar ao usuário e *host* com o comportamento de uma máquina virtual, a estrutura de funcionamento de um contêiner é bem diferente.

A literatura e documentação do Docker tende a subdividi-lo em 4 (quatro) partes distintas entre si, mais que se relacionam no processo de criar e gerenciar contêineres, sendo elas (DOCKER INC., 2021):

- Docker *daemon*: É o responsável por receber os comandos do Docker *client* que são enviadas através de interfaces de linha de comando, ou ainda, por uma API *REST*. É no Docker *Host* que todas as interações de contêineres ocorrem. Sendo assim, é o Docker *daemon* (dockerd) que "escuta"as solicitações e efetua a gerência dos Docker *objects*. Além do *client*, pode ser conectado a outros *daemons* para efetuar tarefas de gerenciamento dos contêineres no *host*. Vale ressaltar que o Docker *host* pode ser local ou remoto.

- Docker *client*: Trata-se do principal meio de interação entre usuário e Docker para que uma determinada ação possa ser executada no *host*. Isso ocorre através de comandos como o *docker run*, que executa um container por exemplo. Eles são enviados ao

Figura 9 – Estrutura do Docker



Fonte: (DOCKER INC., 2021)

dockerd que fica ativamente aguardando instruções a serem executadas no *host*. O *cliente* pode ainda comunicar-se com diferentes *daemons*.

- Docker *registries*: O intuito por trás de um Docker *registry* é basicamente o de armazenar Docker *images*. A Docker Inc. responsável por manter o projeto, disponibiliza um repositório público de imagens intitulado Docker *Hub*, sendo este por sua vez, o repositório padrão onde o Docker serve-se das imagens para a construção dos contêineres no *host*, podendo ser alterado para um repositório local, por exemplo. Um usuário é capaz de construir, salvar e distribuir imagens com outrem ao fazer uso de um *registry*, além de ser capaz de utilizar o comando *docker push* para enviar uma imagem construída diretamente a um registro configurado no Docker *registries*.

- Docker *objects*: O uso do Docker manipula e produz uma série de artefatos sendo estes imagens, contêineres, redes, volumes, *plug-ins* entre outros, a estes dá-se o nome de Docker *objects*. As imagens e contêineres são particularmente importantes, pois são aqueles manipulados de forma direta ao utilizar comandos como do *docker build* (constrói uma imagem) e o "docker run" (executa um container).

A Figura 9 apresenta a estrutura base do Docker. É perceptível que há uma interação direta entre o *client* e docker *host*, isso ocorre pois, o conceito dessa organização é servir contêineres através da nuvem sendo estes por sua vez previamente armazenados em um servidor, culminando assim em um modelo cliente-servidor.

4.3 Imagens em Docker

Uma imagem no Docker trata-se de um objeto que antecede a fase de criação de um contêiner no *host*. Sendo assim, são as imagens que carregam com sigo as instruções necessárias para a criação de um contêiner contendo as características descritas no arquivo de configuração *dockerfile*. É possível então definir uma imagem como sendo um arquivo de caracterização, que armazena em seu interior as informações sobre a construção de um Docker *container*.

O conteúdo de uma imagem é apenas do tipo *read-only*. Isso decorre do fato de que esta por sua vez é um modelo para a criação e execução de um contêiner. Isto implica que, na confecção de uma imagem todos os itens requeridos para o funcionamento da aplicação como bibliotecas, configurações, e softwares adicionais devem estar nas versões compatíveis com o serviço ou processo que se deseja executar posteriormente (DOCKER INC., 2021).

O processo de criação de uma imagem ocorre primariamente com o já citado Dockerfile, que trata-se de um arquivo escrito utilizando uma sintaxe simples em formato de instruções, que se assemelham às utilizadas no Docker *client*. Algo que frequentemente acontece no processo de criação de imagens é a utilização de outras preexistentes, então executa-se uma personalização adicionando os itens que compõem a imagem esperada. Por exemplo, para se criar um servidor web apache é possível no dockerfile solicitar que a imagem do Ubuntu em sua versão 18.04 seja utilizada como sistema de arquivos base, e em seguida apenas adicionar uma instrução que coleta o apache. A Figura 10 ilustra o exemplo.

Figura 10 – Dockerfile básico para criação de uma imagem com Apache.

```
1  #DEFININDO A IMAGEM BASE
2  FROM ubuntu:18.04
3  LABEL maintainer 'Weked A. Curvel <weked.curvel@gmail.com>'
4
5  RUN mkdir netensina && \
6  #efetuando o update da imagem.
7      apt-get -y update && \
8  #efetuando o instalação do Apache.
9      apt-get install -y apache2
```

Fonte: acervo do autor.

As imagens Docker baseiam-se em um sistema de camadas para a sua construção, o AUFS. No exemplo anterior, a imagem Ubuntu 18.04 será adicionada como uma camada à imagem, assim como o comando para a aquisição do apache também cria uma nova camada. A principal vantagem deste modelo de construção é que se houver necessidade de remover, alterar, ou adicionar alguma instrução do Dockerfile apenas a camada correspondente será alterada, sem a necessidade de recompilar toda a imagem.

A compilação de uma imagem é feita a partir da execução do comando *"docker build -t nomedaimagem ."* (Figura 11). A instrução build do Docker iniciará cada comando descrito no *dockerfile*, permitindo assim que cada camada seja adicionada a imagem. Já o argumento *-t*, permite que uma tag seja adicionada a imagem para fins de identificação e posteriormente uma melhor manipulação do arquivo. O último argumento, representado no exemplo por um ponto descreve a localização do arquivo *dockerfile*, vale ressaltar que este não deve possuir extensão ou outro nome.

Figura 11 – Execução comando *docker build*.

```
PS C:\DockerTest> docker build -t ubuntu-apache .
[+] Building 184.4s (4/5)
 => [internal] load build definition from Dockerfile                                                    2.5s
 => => transferring dockerfile: 307B                                                                    0.5s
 => [internal] load .dockerignore                                                                       1.8s
 => => transferring context: 2B                                                                         0.3s
 => [internal] load metadata for docker.io/library/ubuntu:18.04                                         5.9s
 => CACHED [1/2] FROM docker.io/library/ubuntu:18.04@sha256:c2aa13782650aa7ade424b12008128b60034c795f25456e8eb552  0.0s
 => [2/2] RUN mkdir netensina &&     apt-get -y update &&     apt-get install -y apache2               175.5s
 => => # Ign:6 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages
 => => # Get:7 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5 kB]
 => => # Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 Packages [1344 kB]
 => => # Get:9 http://archive.ubuntu.com/ubuntu bionic/multiverse amd64 Packages [186 kB]
 => => # Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [796 kB]
 => => # Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/restricted amd64 Packages [796 kB]
```

Fonte: acervo do autor.

4.4 Contêineres em Docker

Em Docker os conceitos tradicionais de contêineres apresentados anteriormente, que os definem como sendo ambientes simplificados, isolados e altamente portáveis, são aplicáveis. Entretando, Docker pode referir-se a diversas outras formalizações, como a comunidade *open source* que constantemente aprimora as tecnologias utilizadas; A empresa criada para manter e persistir de maneira segura o trabalho desenvolvido pela comunidade; E por fim, a ferramenta de conteinerização de mesmo nome capaz de criar, gerenciar e executar contêineres. Este último por sua vez, é o principal item abordado ao longo deste trabalho (RED HAT, 2021).

A viabilidade dos contêineres em Docker deriva do tipo de virtualização aplicada na ferramenta, que é executada em nível de sistema operacional, ou seja, o gerenciador faz uso do *kernel* Linux e seus recursos. *Cgroups*, por exemplo, trata-se de um alocador de recursos do sistema operacional para os grupos de processos que estão sendo executados em um dado momento. São recursos como esses extraídos do *kernel*, que proporcionam independência na execução dos processos em um container Docker (RED HAT, 2018).

Sendo assim, contêineres independem do ambiente proporcionado pelo *host*, já que os mesmos possuem a capacidade de executar diversas aplicações e processos separadamente. Deste modo, a visibilidade do usuário, processos e aplicações se assemelham no que tange o ambiente apresentado pelo Docker *container*, onde os mesmos têm a percepção de estarem isolados em uma máquina e que este por sua vez é um sistema computacional completo e

autossuficiente, desconhecendo que há somente uma partição abstrata de um processo no *host*.

O Docker viabiliza diversas maneiras para a manipulação dos contêineres, a principal delas é o docker *client*, que é principalmente utilizado através de terminais utilizando linha de comando. O docker possui uma sintaxe padronizada e de fácil assimilação, onde a estrutura é iniciada pela instrução docker seguida de uma ou mais instruções e argumentos.

O client permite que vários comandos possam ser utilizados direcionados a docker *containers*, sendo o mais básico deles o *docker container run*. Este comando permite que um container seja criado suprimindo uma série de outros comandos como o *create* e o *start*, ou mesmo instruções para conectar o container a uma rede. O comando run permite a adição de um nome para o container, e necessita que uma imagem seja definida como base. A Figura 12 ilustra o comando *docker run*.

Figura 12 – Execução comando *docker container run*.

```
PS C:\DockerTest> docker container run -it -d --name ubuntu ubuntu:18.04
c588a7f7a66c1e1182b97bfff881b7a6b24c87cf656c8591d7031a5b28207381
PS C:\DockerTest> docker container ls
CONTAINER ID   IMAGE          COMMAND   CREATED          STATUS          PORTS     NAMES
c588a7f7a66c   ubuntu:18.04   "bash"    18 seconds ago   Up 14 seconds             ubuntu
PS C:\DockerTest>
```

Fonte: acervo do autor.

Outro comando muito utilizado e certamente um dos mais importantes para a gestão de contêineres, é o *docker container exec*. Este por sua vez, é utilizado para acessar um docker *container* que esteja sendo executado. A partir dele é possível manipular o terminal de uma unidade, e efetuar procedimentos de configuração internos, por exemplo. A figura 13 ilustra a utilização do comando *docker exec*.

Figura 13 – Execução comando *docker container exec*.

```
PS C:\DockerTest> docker container exec -it ubuntu /bin/bash
root@c588a7f7a66c:/#
root@c588a7f7a66c:/#
root@c588a7f7a66c:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:5 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 Packages [26.8 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [2544 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 Packages [11.3 MB]
19% [7 Packages 477 kB/11.3 MB 4%] [6 Packages 971 kB/2544 kB 38%]
```

Fonte: acervo do autor.

4.5 Redes no Ambiente Docker

À primeira vista, a interface de rede utilizada pelos contêineres no Docker tendem a ser semelhantes às redes de computadores convencionais, levando em consideração a sua

finalidade, a comunicação entre dispositivos (no Docker, entre contêineres). Contudo, o conceito de rede no Docker melhor define-se como uma abstração do modelo convencional de rede de computadores.

Inicialmente o Docker fornece três redes distintas entre si, cada uma apresentando as próprias funcionalidades, limitações e drive's atribuídos. Sendo elas, *bridge*, *none* e *host* (DOCKER INC., 2021). Tendo em vista que este trabalho faz uso recorrente das redes Docker para a produção dos laboratórios, as mesmas serão descritas na sequência.

Figura 14 – Redes padrão do Docker

```
PS C:\> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
3daddfe01417   bridge    bridge    local
19ca6761e03a   host      host      local
ecc1078553e9   none      null      local
PS C:\>
```

Fonte: acervo do autor.

4.5.1 Rede Bridge

O Docker fornece uma gama de praticidades no momento da criação de um contêiner, geralmente várias informações mais específicas podem ser omitidas obtendo-se ainda assim um container funcional. A rede *brigde*, se trata da rede padrão fornecida pelo Docker na execução do comando *run*, quando este é executado com a ausência de um comando mais explícito para a interface de rede.

Esta rede do Docker ao contrário do que o seu nome sugere, imprime ao contêiner que a utiliza uma interface de rede que se conecta em modo NAT com a interface docker0 fornecida para o sistema que hospeda o Docker. Por sua vez, a rede *bridge* automatiza o fornecimento de endereços disponíveis dentro do *range* compreendido pela rede IP 172.17.0.0/16 por padrão. Outras variações, por sua vez, devem ser configuradas através do Docker *client*, levando em consideração que esses procedimentos de configuração deverão gerar outras redes com suas próprias características.

A comunicação exercida por todos os contêineres que fazem parte deste tipo de rede e das outras padronizadas pelo Docker, se trata do protocolo TCP/IP, utilizado nas redes de computadores. Ou seja, se um contêiner souber o endereço de outro participante da mesma rede, é perfeitamente possível o direcionamento de um fluxo de dados de um "contêiner de origem" para um "contêiner de destino".

Outra característica inerente a este tipo de rede, é a comunicação externa. Esta por sua vez, são mediantes as configurações do sistema que hospeda o Docker, semelhante

ao que ocorre em máquinas virtuais que se utilizam deste tipo de interface. Por exemplo as VM (*virtual machines*) do Virtualbox, que estão configuradas em modo NAT veem apenas o sistema *host* como um *gateway* para as tarefas de fluxo de dados, e para a rede a qual o *host* se encontra a VM não é visível, já que há uma camada de abstração sobre ela. Simplificando, se o *host* possui saída para a internet, consequentemente o contêiner também poderá acessá-la.

4.5.2 Rede Host

A rede Docker nativa do tipo *host*, trata-se de um tipo de interface de "conversa" direta entre o hospedeiro do sistema docker e o contêiner mapeado neste tipo de rede. E assim, como citado no tópico anterior, o protocolo aplicado na comunicação dos containers é também o TCP/IP.

A rede host possui a particularidade de ser do tipo *bridge*. Ou seja, as características que versam sobre o interfaceamento entre contêiner e *host* permitem que o elemento em questão seja uma extensão da rede à qual pertence o sistema hospedeiro, além de é claro utilizar o mesmo *range* de IP ao qual pertence o sistema *host*.

Levando em consideração as informações anteriores, contêineres atrelados a este tipo de rede não tem a capacidade de compartilhamento de portas equivalentes entre outros Docker *containers*. Por exemplo, ao executar-se dentro da rede *host* duas imagens docker que se utilizem da porta 80 para publicar um servidor *web*, após um curto período de tempo (precisamente o da inicialização do contêiner) um dos contêineres seria derrubado pelo sistema docker, pois a porta necessária para o seu funcionamento não estaria disponível. A solução, por sua vez, seria executar este segundo container em um *host* diferente.

4.5.3 Rede None

Por fim, o terceiro tipo de rede nativa do docker, é a *none*. Esta é uma rede capaz de isolar *containers*. Sendo assim, a mesma é capaz de isolar um elemento do acesso externo ao sistema docker, como também do acesso de outros contêineres.

Essa característica de isolamento é muito importante para a produção de ambientes capazes de persistir dados críticos para certos tipos de aplicações, como os sistemas de bancos de dados.

Como geralmente este tipo de rede é utilizada para containers de manipulação de fluxo de informações, costuma-se utilizar esta rede para containers do tipo volume, que para um correto funcionamento com outras aplicações devem ser devidamente mapeados.

5 NetEnsina Docker

5.1 Visão Geral

A absorção de informação para a conversão em conhecimento sólido, trata-se de uma etapa fundamental para o processo de aprendizado em qualquer área de estudo. Contudo, em cursos pertencentes ao ramo da informática as atividades práticas constituem um eixo fundamental deste processo, sobretudo em Redes de Computadores tais atividades tornam-se dispendiosas e dificultosas, algo que deriva do fato das aulas e sessões de estudo dependerem de uma infraestrutura muito mais complexas do que somente computadores. Dispositivos de rede como cabos, roteadores, *switches*, e repetidores são itens imprescindíveis para manter uma boa qualidade dos treinamentos direcionados ao ensino de Redes de computadores.

O NetEnsina Docker é uma ferramenta produzida com o intuito de auxiliar o ensino de Redes de Computadores. A ferramenta em questão, faz uso do gerenciador de contêineres Docker para implementar práticas laboratoriais virtuais nos cursos de redes, além do uso de um repositório de dados no GitHub que servirá de apoio para o professor na elaboração de um roteiro didático consistente, que poderá ser adotado ao longo das aulas práticas, e acompanhado pelos alunos.

O NetEnsina Docker se propõe a ser uma ferramenta de fácil acesso, capaz de ser executada em ambientes não tão robustos ou que demandem custos altos para sua implantação. Ele permite também que o aluno possa praticar de diferentes locais, possuindo apenas um sistema computacional com o ambiente Docker previamente instalado e uma conexão com a internet.

5.1.1 O Docker no NetEnsina

Já citado anteriormente neste documento o Docker trata-se de uma ferramenta voltada à criação e gestão de contêineres para aplicações, capazes de fornecer os recursos necessários para o seu correto funcionamento. Através dela é possível criar ambientes suscetíveis contendo bibliotecas específicas e códigos necessários sem afetar diretamente o sistema operacional da máquina que os hospeda, garantindo isolamento e segurança.

O Docker trás à luz diversas possibilidades para criação de estruturas para o fornecimento de serviços. Isso é possível pois, do ponto de vista de execução em uma dada aplicação, a estrutura fornecida será semelhante a de um hardware convencional, possuindo memória, armazenamento, comunicação em rede, dispositivos e sistema operacional. Sendo assim, cada contêiner se assemelha a uma unidade computacional completa, contudo,

virtualizada e portátil.

Sabendo que essas unidades assemelham-se a computadores convencionais que podem ser executados de forma leve em máquinas reais sem consumos expressivos de recursos de hardware, e com capacidade de escalar uma variada quantidade de contêineres com pouco ou nenhum prejuízo ao desempenho total do dispositivo que hospeda o Docker, utilizou-se os Docker *Containers*, como a base dos laboratórios virtuais práticos para o ensino de redes de computadores do NetEnsina.

O Docker viabiliza a criação de diversas unidades virtuais a partir de uma mesma imagem. Esta por sua vez, tem como função principal padronizar o tipo de atividade que os contêineres podem realizar. Desta forma, é possível criar contêineres com um conjunto de ferramentas de software próprias e assim destinando-os a servirem ao propósito de uma determinada aula prática, por exemplo. Outro uso interessante é a simulação de dispositivos de rede como roteadores, que podem ser reproduzidos através da adição de recursos para a construção de tabelas de roteamento às imagens.

O meio de comunicação fornecido pelo Docker para que os contêineres possam trocar informações entre si e o *host* é bastante completo, baseia-se nas redes convencionais e fornece diversas estruturas próprias que possuem propósitos distintos entre si, como apresentado na seção 4.5. É a Docker *Network* que viabiliza a construção das topologias empregadas nos ambientes construídos para as aulas do NetEnsina. Ela viabiliza a criação de redes próprias, proporcionando a capacidade de configurar e adicionar Docker *containers*, além de possibilitar configurações mais específicas como as de IPs, máscaras e *gateways*.

O Docker é uma ferramenta eficaz na construção de ambientes voltados ao desenvolvimento de aplicações e fornecimento de serviços, mas, é perceptível que foi pouco explorada na construção de ambientes virtuais de ensino. Tal situação pode decorrer do parcial descobrimento de todas as capacidades oferecidas pelo Docker até o momento da escrita deste documento. Onde a maior parte de seu engajamento está voltado principalmente ao fornecimento de serviços e seus aprimoramentos, ou ainda, pode ser fruto da complicada configuração exigida para a construção de ambientes virtuais mais complexos que se propõem a simular estruturas de redes. As dificuldades envolvidas nesse processo de configuração e principalmente o desconhecimento das estruturas de comunicação do Docker, podem acabar desencorajando este lado pouco conhecido e explorado da ferramenta.

O NetEnsina Docker por sua vez, busca apresentar a viabilidade da construção de ambientes virtuais voltados ao ensino de redes de computadores de maneira prática. Apresentando um repositório de dados contendo as imagens, arquivos utilizados para a configuração e construção dos laboratórios, tutorias e roteiros didáticos. Desta maneira é possível criar uma forma de padronizar os laboratórios, possibilitando assim que professores possam criar suas próprias aulas práticas e disponibilizá-las a seus alunos.

5.2 Construindo laboratórios práticos

5.2.1 Estrutura

De um modo geral as instituições de ensino lidam com um problema corriqueiro em seus ambientes educacionais, a inconstância no número de alunos que frequentam os cursos em determinados períodos. Desta maneira, as aulas práticas sofrem um golpe inevitável, seja pela subutilização dos equipamentos, por pouca demanda ou pela ausência de material suficiente para suprir uma alta presença de indivíduos no ambiente. Desta maneira torna-se extremamente complexa a tarefa de previsão de usuários, e consequentemente dos recursos a serem direcionados para os ambientes laboratoriais.

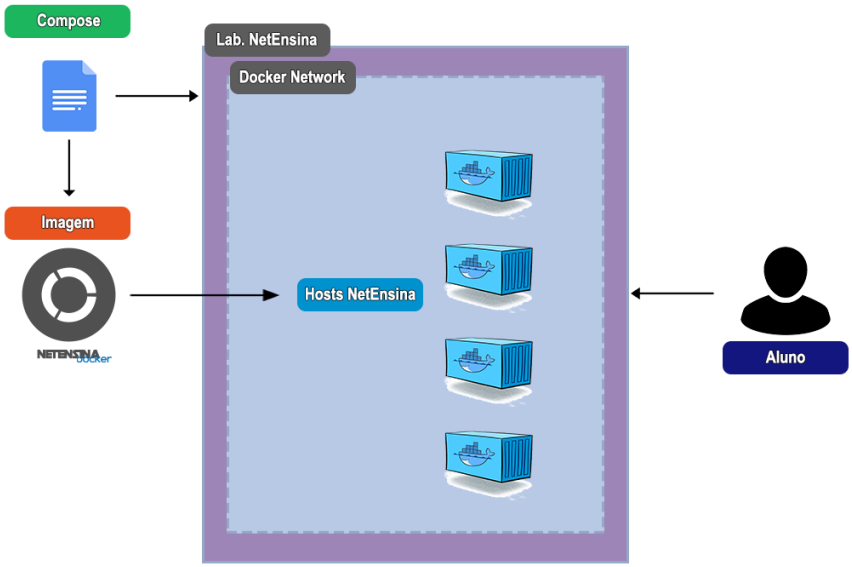
É conhecida a importância que as disciplinas práticas possuem para o eixo formativo das disciplinas de computação, sobretudo essa é uma realidade crítica quando volta-se a atenção para as disciplinas de rede, que além de exigirem computadores, necessitam de dispositivos específicos inerentes à área de atuação desses futuros profissionais, para que assim haja a qualidade mínima para o correto andamento do curso.

Por outro lado, algo que surgiu como um método eficaz capaz de suprir esse problema foi a virtualização dos ambientes de ensino através de simuladores. Assim, é possível realizar as aulas práticas de laboratórios sem a necessidade de se adquirir novos dispositivos computacionais, o que corrobora na diminuição dos custos aplicados aos ambientes direcionados a execução de aulas práticas. O NetEnsina Docker se apoia nesse propósito para fomentar a construção de ambientes virtuais que possam ser utilizados no ensino prático de redes de computadores.

Os contêineres em geral apresentam diversas vantagens quando comparados a outras metodologias de virtualização. Como citado na seção 2.5, é possível criar unidades computacionais que possuem aplicações auto suficientes com todos os recursos necessários ao seu dispor e sem afetar o *host* para tal. Desta forma, o NetEnsina Docker utiliza a ferramenta de gestão de contêineres para produzir "computadores"virtuais bem leves, com a capacidade de transparecer ao usuário e aplicações experiência similar a de um dispositivo computacional real, para que assim possam ser realizadas as práticas laboratoriais.

A estrutura com a qual os laboratórios são construídos está dividida como apresentado na Figura 15: a imagem base dos dispositivos, o arquivo de inicialização e os laboratórios gerados. Os arquivos necessários por sua vez podem ser encontrados no repositório do NetEnsina Docker no Github. Desta maneira é possível tanto para alunos quanto professores acessarem os laboratórios do NetEnsina Docker. Estes últimos por sua vez podem ainda se utilizar dos modelos disponibilizados no repositório para criarem seus próprios laboratórios.

Figura 15 – Estrutura de funcionamento do NetEnsina Docker.



Fonte: acervo do autor.

5.2.2 Dockerfile e Image

Quando refere-se à Docker *Containers*, toda uma estrutura prévia é necessária para que um serviço possa ser fornecido corretamente, seja este uma ambiente de desenvolvimento ou uma aplicação já construída. Por consequência, para cada tipo de finalidade de uso dentro de uma aplicação conteinerizada recursos como bibliotecas, arquivos e códigos necessitam estar presentes nestes ambientes. Para os laboratórios do NetEnsina Docker isto não é diferente, pois o ambiente precisa se adequar ao ensino de redes.

Os contêineres em Docker sobretudo são baseados no que denota-se como imagens, assim como citado na Seção 4.3. As imagens por sua vez são o ponto de partida para um contêiner ou um conjunto deles com características semelhantes, seja pelo sistema arquivos base, ou pelas ferramentas instaladas para dar suporte a aplicação ou serviço fornecido pelo contêiner. Contudo, imagens Docker são personalizáveis para que assim possam suprir as necessidades de cada usuário.

O Dockerfile é o arquivo que propicia à personalização das imagens. Nele é possível ao desenvolvedor escolher o sistema de arquivos que se deseja utilizar, e então adicionar as ferramentas que necessita para que o seu serviço funcione corretamente, ou seja, é possível especificar versões, adicionar conteúdos de origens do próprio desenvolvedor, como repositórios próprios ou arquivos locais.

No NetEnsina Docker, o Dockerfile foi criado de maneira para que a imagem gerada agregasse as principais ferramentas voltadas ao monitoramento de rede nos contêineres,

para que assim, os alunos pudessem analisar os dados das tarefas realizadas em suas aulas práticas como se os fizesse em dispositivos reais. Pacotes como o *tcpdumps* e *iputils* foram adicionados para esta finalidade.

Cada imagem no Docker é criada na forma de camadas, onde cada uma delas é uma sobreposição da outra iniciando pelo sistema de arquivos, e estes por sua vez funcionam como se fossem versões simplificadas de sistemas operacionais baseados em Linux, a exemplo do Debian e Ubuntu. Para o NetEnsina Docker, o sistema de arquivos escolhido para a base da imagem netensina/dockerlabs foi a do Ubuntu em sua versão 18.04, por ser considerada uma das versões mais estáveis do sistema.

Em um *Dockerfile* cada novo comando adicionado é refletido é uma nova camada na imagem final. O problema atrelado à construção de imagens com muitas camadas é o aumento da complexidade e consequentemente o tamanho do arquivo final, este último por sua vez é motivo de atenção para o NetEnsina Docker. Desta maneira garante-se que não hajam arquivos que se tornem inviáveis de serem adquiridos pelos alunos através da internet. As camadas foram adicionadas tomando o cuidado em criar uma imagem com um tamanho razoável, mas que possuísse ferramentas úteis para o estudo de redes de computadores. A Figura 16 mostra o arquivo Dockerfile do NetEnsina Docker.

Figura 16 – Dockerfile dos hosts do NetEnsina Docker.

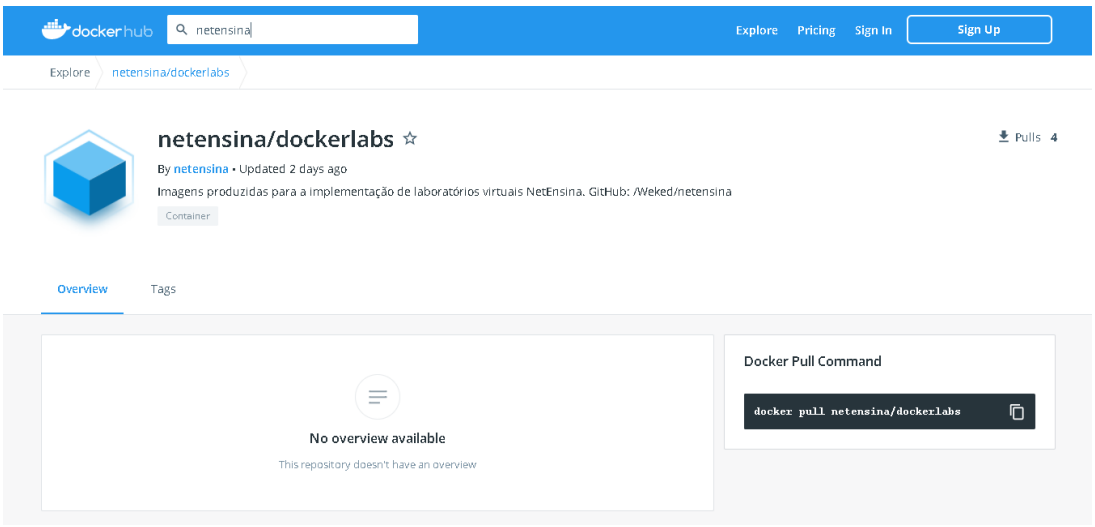
```
FROM ubuntu:18.04
LABEL maintainer 'Weked A. Curvel <weked.curvel@gmail.com>'
#Camada com os recursos comuns a todos os containeres.
#criando o workdir que ira conter o arquivos a serem compartilhados com o host.
RUN mkdir netensina && \
#efetuando o update da imagem.
    apt-get -y update && \
#instaladno os recursos de rede comuns a todos os containeres.
    apt-get install -y net-tools iputils-ping tcpdump iproute2 netcat dnsutils curl iptables nmap && \
#Definindo a camada personalizada do equipamento a ser emulado,
#contem ferramentas direcionadas à uma tarefa específica.
    apt-get install -y bridge-utils network-manager
#definindo o workdir
WORKDIR /netensina
```

Fonte: acervo do autor.

No arquivo observa-se uma linha que define o sistema de arquivos Ubuntu como citado anteriormente, em seguida, é feita a atualização do mesmo, e a próxima camada é onde é feita a instalação das ferramentas necessárias para a configuração e monitoramento de rede dentro do container. Para este Dockerfile, o propósito principal era o de criar uma imagem que pudesse ser alterada facilmente para a simulação de um determinado dispositivo ou funcionalidade de software, além de facilitar as produções de laboratórios independentes.

Uma versão da imagem criada para o NetEnsina foi disponibilizada no repositório onde é mantido este projeto, e também pode ser acessado através do repositório de imagens públicas do Dockehub.

Figura 17 – Imagem netensina/dockerlabs no repositório público Docker *Hub*.



Fonte: acervo do autor.

5.2.3 Containers

Como visto na Seção 4.4, Docker *containers* são produtos gerados a partir de uma imagem existente, é nestas unidades virtuais que uma aplicação pode ser fornecida como um serviço, pois a mesma é mantida isolada do sistema *host*, proporcionando assim do ponto de vista da aplicação e usuários um ambiente completo com todos os recursos do sistema disponíveis. No NetEnsina Docker os contêineres são tratados como unidades computacionais ou mesmo *hosts*, algo que difere do conceito de sistema *host*, onde este por sua vez refere-se ao dispositivo que hospeda a ferramenta Docker.

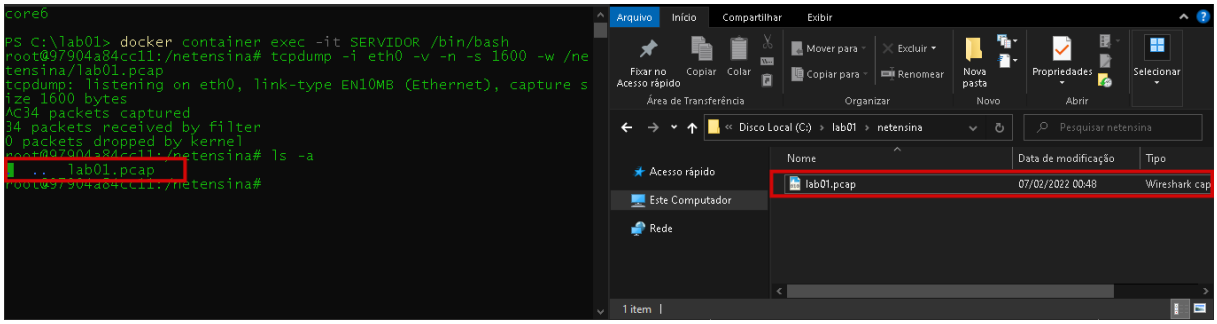
No NetEnsina os hosts são gerados a partir da imagem netensina/dockerlabs. Com isso tem-se unidades computacionais baseadas no sistema de arquivos do Linux Ubuntu 18.04, com diversos recursos de configuração e monitoramento de rede adicionados para que se possa aproximar a experiência dos alunos da de dispositivos reais.

Cada host possui as configurações de hardware padrão do Docker, onde cada dispositivo possui 2 GB de memória RAM alocada em tempo de execução sendo mapeada na memória física do sistema *host*, ou seja, a memória é alocada à medida em que a unidade virtual necessita de mais memória para seu funcionamento, possuindo ainda uma área de *swap* de 1 GB. Quanto ao processamento, cada *host* recebe a capacidade de metade dos núcleos computacionais da CPU do sistema *host*.

Os hosts possuem ainda a capacidade de troca de arquivos com o sistema *host*, isso é feito através de um mapeamento de diretórios. Cada *host* é criado com um diretório denominado NetEnsina, neste por sua vez são armazenados arquivos de diagnóstico de rede, como os gerados pela ferramenta *tcpdumps*, assim o aluno poderá posteriormente utilizar softwares para análise dos mesmos, como é o caso do Wireshark. O diretório NetEnsina

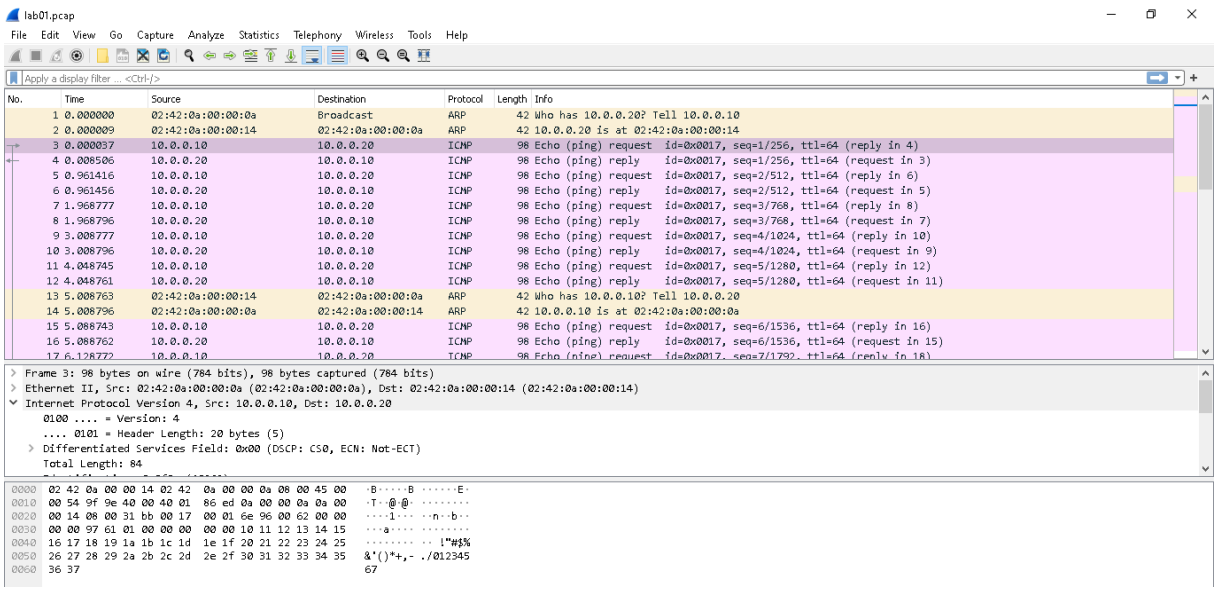
nos *hosts* é mapeado para outro de mesmo nome no sistema *host*, mais especificamente no local onde encontra-se o laboratório executado, como mostra a Figura 18.

Figura 18 – Conteiner com diretório mapeado no sistema host.



Fonte: acervo do autor.

Figura 19 – Arquivo gerado por *Host* NetEnsina aberto no Wireshark.

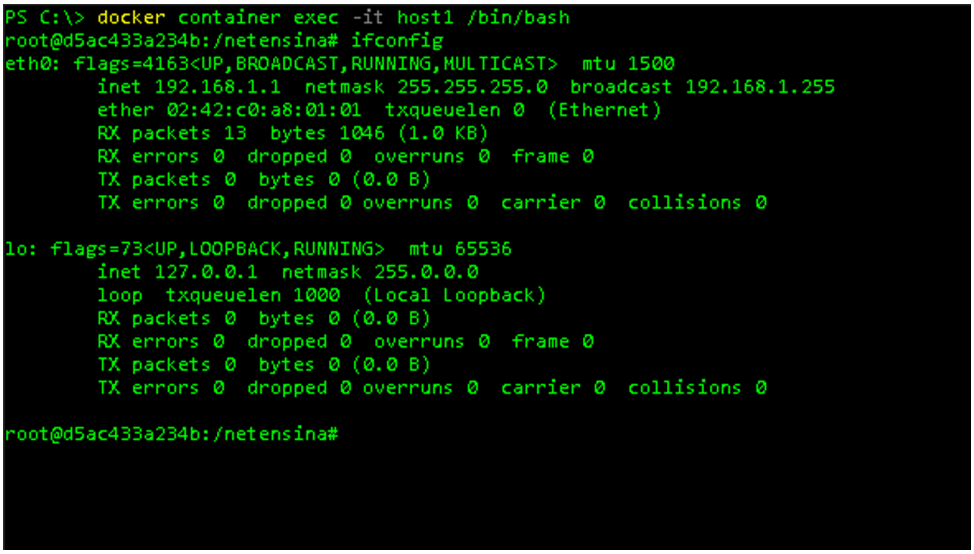


Fonte: acervo do autor.

Os *hosts* não possuem a capacidade de troca de arquivos entre si como ocorre em relação ao sistema *host*, portanto nenhum mapeamento entre eles foi definido, podendo trocar informações apenas através da Docker *Network* dependendo apenas de como está organizada a topologia de rede empregada no laboratório do qual fazem parte.

Quanto à interface, os *hosts* do NetEnsina Docker podem receber interações através de comandos de texto através do terminal Linux que possuem. Um *host* pode ser acessado através da ferramenta Docker utilizando o comando de execução *exec* citado na seção 4.4, utilizando os argumentos de interação -it, seguido do comando a ser executado, /bin/bash por exemplo executa o terminal da máquina. Desta maneira os alunos podem interagir com

Figura 20 – Host do NetEnsina sendo acessado através do terminal.



Fonte: acervo do autor.

os *hosts* e então executar as tarefas descritas nos roteiros didáticos de cada laboratório. A Figura 20 mostra como um *host* é acessado.

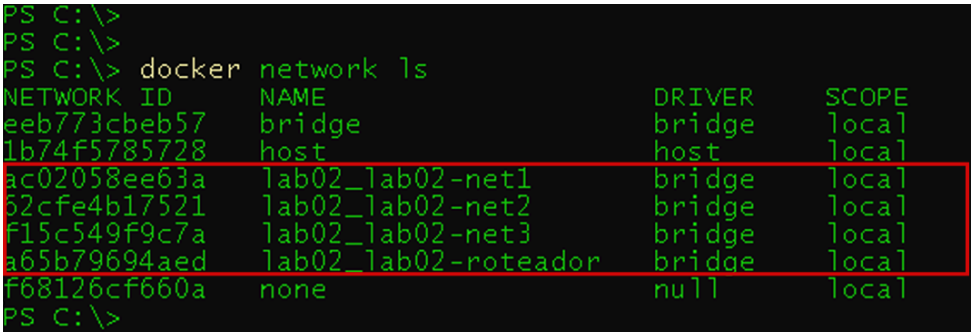
5.2.4 Redes

A comunicação entre contêineres através da Docker *Network* é de fato um dos conceitos mais importantes no momento da construção de aplicações conteinerizadas, pois assim é possível fazer com que os serviços sejam acessados por determinadas configurações de rede expondo portas ou não. Além de possibilitar que estruturas bem mais complexas e robustas possam ser construídas, onde cada container executa apenas suas próprias tarefas. O que consequentemente aumenta o isolamento, e a segurança por acessos indevidos a unidades que fornecem serviços críticos como os de bancos de dados, por exemplo. O NetEnsina Docker faz uso da Docker *Network* para produzir a comunicação entre os *hosts* da maneira que são propostos os roteiros didáticos.

Reproduzir as topologias de redes físicas utilizando contêineres é um tanto quanto complicado devido ao forte isolamento que estes por sua vez possuem. Para contornar tal problema cada configuração deve ser minuciosa, de maneira a evitar conflitos de comunicação ou mesmo o isolamento total de um contêiner.

Na Seção 4.5 apresenta-se os conceitos que circundam as redes Docker, é possível observar a existência de três redes distintas: *bridge*, *host* e *none*. A partir dessas três é permitida a criação de outras redes com características semelhantes, sendo possível ainda personalizar as configurações. No NetEnsina Docker, a rede bridge atende todas as demandas necessárias para a reprodução das topologias propostas, assim cada laboratório possui uma ou mais redes derivadas dela.

Figura 21 – Redes criadas por laboratório do NetEnsina Docker.

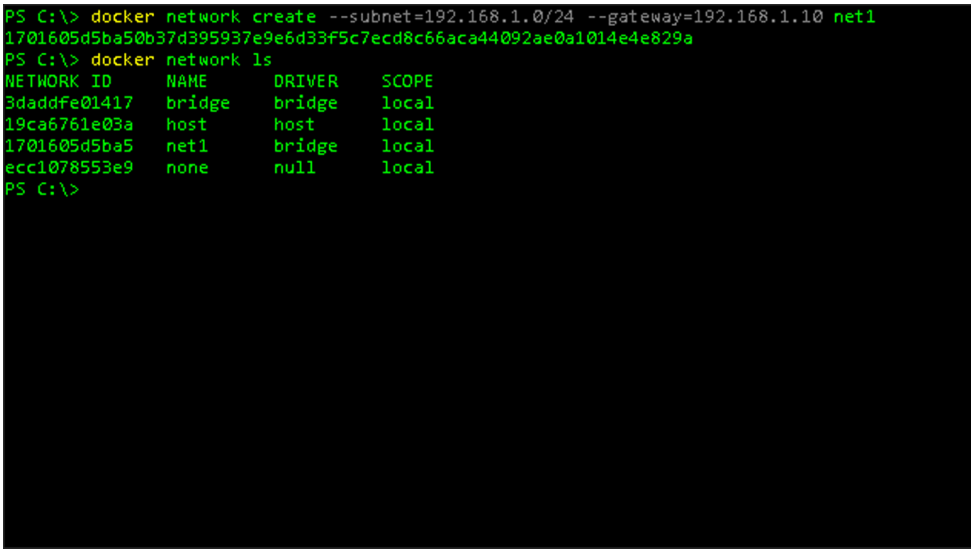


Fonte: acervo do autor.

A bridge é a rede padrão do Docker, com ela os containers recebem automaticamente as configurações básicas de IP, máscara de sub-rede e *gateway* padrão. O *driver bridge* fornece ao contêiner uma conexão com o sistema *host* através de uma NAT, sendo assim os contêineres podem receber acesso a internet também, se o sistema host estiver conectado.

Para simular as topologias de rede propostas nos roteiros, o NetEnsina utiliza o recurso de criação de *subnets* do Docker. É através das *subnets* que ocorre a personalização das configurações de rede como o IP e *gateway* padrão. Imagine por exemplo, que se queria criar um laboratório em que dois computadores trocam informações através de uma rede com IP da classe C 192.168.1.0/24, utilizando máscara 255.255.255.0 e *gateway* 192.168.1.10, com o comando de *subnet* utilizando o argumento de -ip é possível definir a faixa de endereços para os hosts, em seguida com o argumento -gateway é definido o *gateway* padrão que as unidades utilizarão, e por fim a máscara é definida pelo Docker automaticamente. A Figura 22 demonstra a criação da *subnet* citada.

Figura 22 – Criação de subnet com IP da classe C no Docker.

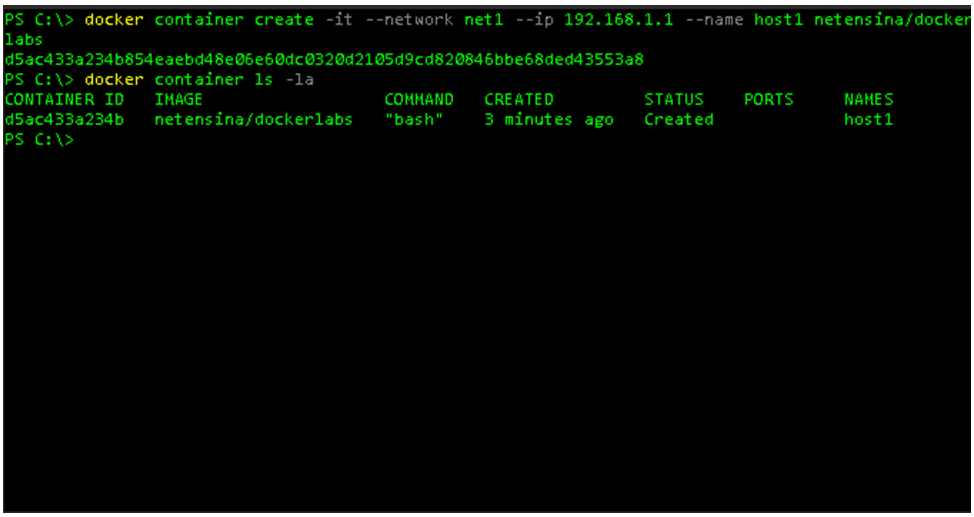


Fonte: acervo do autor.

Na construção das subnets é possível ainda definir que tipo de *driver* será usado. Com base no que foi apresentado na Seção 4.5, contudo, no NetEnsina tal configuração é omitida pois o *driver* padrão já serve muito bem ao propósito de reprodução dos roteiros didáticos. As demais configurações que possuem relação com as interfaces de rede das unidades computacionais são definidas no NetEnsina durante a criação dos contêineres. Isso ocorre pois os *hosts* utilizados nos laboratórios devem possuir IPs estáticos, dessa maneira é facilitado ao aluno acessar um *host* através do já citado comando de execução docker *exec*.

Para conectar um *host* a *subnet* do exemplo citado, no ato de criação do container é preciso especificar o nome da *subnet* à qual o mesmo irá pertencer através do comando -network, isso fará com que o Docker associe o *host* a rede em questão. Como explicado anteriormente um IP automático é atrelado ao *host*, para que isso não ocorra, no momento da criação o argumento -ip também deve ser adicionado e referenciando a um IP válido pertencente a faixa de endereços da *subnet*, como mostra a fig 23.

Figura 23 – Criação de host conectado à subnet.



Fonte: acervo do autor.

5.2.5 Docker compose

A virtualização apresentada pelas ferramentas de redes de um modo geral, tem por objetivo simular diversos dispositivos funcionando em simultâneo efetuando suas funções. Há sobretudo situações que necessitam que certos dispositivos estejam disponíveis previamente na rede para que os outros equipamentos possam efetuar sua comunicação, portanto, existem cenários que exigem prioridades de inicialização por parte da ferramenta. Tomando como base esses empecilhos, o NetEnsina foi pensado utilizando o recurso denominado Docker *Compose*, para contornar tais problemas.

O *Compose* se trata de um orquestrador de Docker containers. Com ele é possível instanciar vários contêineres em simultâneo, indicando quais são os prioritários, as redes às quais pertencem, configurações de endereços, exposição de portas, criação de redes e diversas outras funções que derivam das tarefas que podem ser realizadas manualmente com os comandos Docker. O Compose é uma ferramenta que foi incorporada nativamente ao Docker em suas últimas versões, sabendo que a ferramenta de gerência de containers sempre mantém-se atualizada, é coerente supor que o aluno deverá possuir o *Compose* disponível em sua máquina.

O Docker *Compose* de um laboratório do NetEnsina é uma arquivo escrito com uma sintaxe denominada YML, nele inicialmente define-se a versão do *Compose* a ser utilizada, e posteriormente o documento divide-se em duas partes. A primeira parte é dedicada a área de configurações mais gerais do laboratório, no caso do NetEnsina as redes são criadas neste momento, o que substitui o uso do comandos de criação de redes mencionados anteriormente, definimos o *driver*, o endereço da sub-rede e também seu *gateway*, não é necessário especificar a máscara, algo que assemelha-se ao que ocorre no comando manual. A segunda parte denominada de *services* é onde estão as configurações utilizadas pelos hosts. A Figura 24 mostra um arquivo Docker *Compose* utilizado no NetEnsina.

Figura 24 – Área de configurações gerais de um Docker Compose do NetEnsina.

```
#Versão do Compose
version: '3.9'
#Area geral
networks:
  lab02-roteador:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 200.120.1.0/24
          gateway: 200.120.1.254
  lab02-net1:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 192.168.1.0/24
          gateway: 192.168.1.254
  lab02-net2:
    driver: bridge
    ipam:
```

Fonte: acervo do autor.

Ainda sobre área *services*, é onde define-se o que cada unidade representará no laboratório. As configurações nessa etapa são feitas individualmente para cada *host*,

inicialmente a imagem netensina/dockerlabs é definida como a base de cada contêiner, e em seguida são mapeados os diretórios como mencionado na seção 5.2.3, pois é nesse diretório dentro do laboratório que estarão contidos os arquivos de diagnóstico de rede, que podem ser utilizados pelo aluno. A próxima etapa utilizada na configuração dos *hosts* é a associação a uma rede e a definição de um IP estático, então as redes definidas na primeira parte são associadas a seus respectivos *hosts*. Nesse momento, nas situações onde existem roteadores interligando várias redes, o mesmo é conectado a cada uma delas respectivamente. O próximo passo é configurar cada unidade no modo interativo atribuindo-lhes privilégios. Desta maneira o aluno poderá posteriormente acessar cada máquina através do terminal do sistema *host* e executar qualquer comando sem restrição. Ao fim, define-se a prioridade de inicialização de cada dispositivo, no exemplo dos roteadores, por exemplo, optou-se por priorizar a inicialização dos mesmos antes dos *hosts*. A Figura 25 apresenta um arquivo Docker *Compose* com as configurações dos *hosts* utilizadas no NetEnsina.

Figura 25 – Área services de um Docker Compose do NetEnsina.

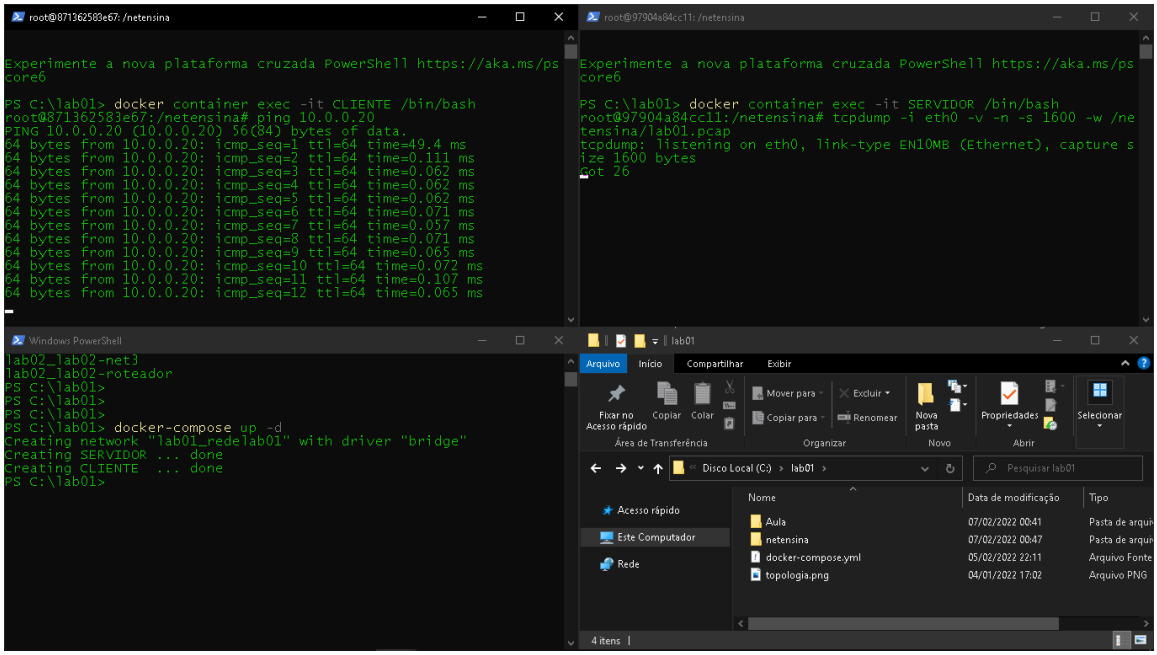
```
32                  gateway: 10.1.1.254
33  #Area Services
34  services:
35    R1:
36      image: netensina/dockerlabs:latest
37      volumes:
38        - ./netensina:/netensina
39      networks:
40        lab02-roteador:
41          ipv4_address: 200.120.1.1
42        lab02-net1:
43          ipv4_address: 192.168.1.101
44        lab02-net2:
45          ipv4_address: 192.168.2.102
46      tty: true
47      stdin_open: true
48      privileged: true
49      container_name: R1
50    R2:
```

Fonte: acervo do autor.

Poucos comandos são necessárias para se iniciar um laboratório do NetEnsina, e ao ser submetido no terminal do sistema *host* o comando *docker-compose up -d* a partir do diretório em que se encontra o laboratório, tudo o que foi configurado e especificado no arquivo *Compose* é executado gerando ao fim um serviço, que nada mais é do que um conjunto de Docker *containers* funcionando sobre o sistema *host*.

O arquivo orquestrador de contêineres de um laboratório do NetEnsina é uma parte crucial de toda arquitetura concebida, pois o mesmo assemelha-se a uma espécie de *script*, onde todos os comandos Docker podem ser representados. Com a utilização do Docker *Compose* aliado aos roteiros didáticos, o NetEnsina Docker evita que o aluno tenha a necessidade de aprender os comandos utilizados pela ferramenta para manipular

Figura 26 – Laboratório NetEnsina Docker inciado por arquivo *compose*.



Fonte: acervo do autor.

os contêineres, mantendo-se focado apenas no aprendizado de redes. Desta maneira os poucos comandos que restam para iniciar e parar os laboratórios, e acessar o terminal dos *hosts* também encontram-se contidos nos roteiros.

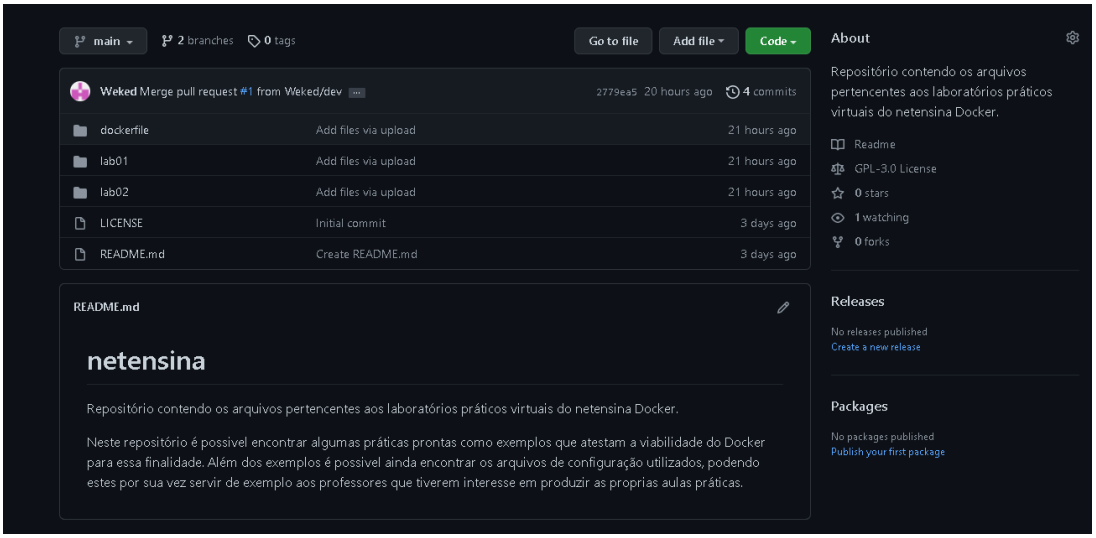
5.2.6 NetEnsina no GitHub

O NetEnsina Docker tem como principal objetivo auxiliar alunos no âmbito do aprendizado de redes de computadores, proporcionando laboratórios de prática virtualizados. Tendo em vista que a maior parcela dos indivíduos que têm contato com este tipo de área do eixo formativo de profissionais da computação estão de certa forma inseridos, ou ainda se inserindo em grupos, equipes ou projetos, que se utilizam de ferramentas para uma melhor organização dos trabalhos desenvolvidos, e que estas por sua vez tornaram-se parte fundamental na produção de produtos computacionais. Tornou-se propício para o NetEnsina Docker o uso de uma ferramenta popular e acessível de gestão de produtos de software como o GitHub.

O GitHub trata-se de uma plataforma de hospedagem de arquivos e código-fonte fundada em 2008, que possui controle de versão de software e afins, utilizando em sua estrutura interna o Git para proporcionar o versionamento. A escolha desta plataforma para a publicação, controle e atualização do NetEnsina Docker mostrou-se natural devido a facilidade apresentada pela mesma para contribuição e compartilhamento dos projetos entre as pessoas cadastrados na plataforma, possuindo ainda um uso intuitivo e amigável para novos usuários. Além disso, possui uma documentação extensa e estruturada,

com o diferencial de ser bem popular entre os estudantes, profissionais e entusiastas da computação.

Figura 27 – Repositorio público do NetEnsina Docker no GitHub.



Fonte: acervo do autor.

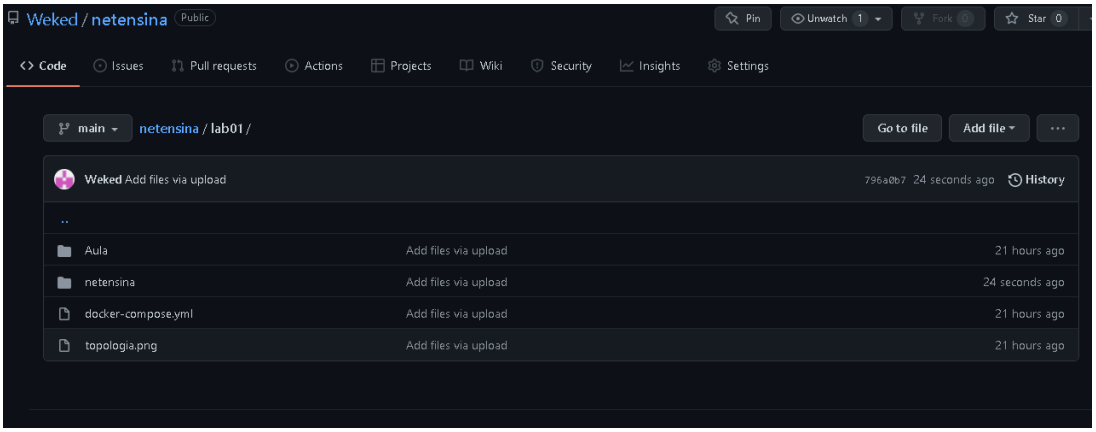
Quanto à organização dentro da plataforma GitHub, o NetEnsina Docker estrutura-se em um repositório de mesmo nome. O repositório por sua vez armazena todos os laboratórios produzidos até então. Tais laboratórios são representados por diretórios nomeados com um número representando sua ordem proposta para o aprendizado, precedido pela palavra "Lab". Cada diretório subdivide-se em outros responsáveis por armazenar os roteiros em formato PDF, a área compartilhada NetEnsina, e o Docker *Compose* responsável por orquestrar os contêineres do laboratório. Ainda é possível encontrar no GitHub uma seção que auxilia os professores na produção de seus próprios laboratórios, contendo os arquivos *dockerfile* utilizados para a construção das imagens do NetEnsina Docker.

A referida organização do repositório do NetEnsina Docker tem como objetivo fornecer ao usuário da ferramenta, todos os arquivos necessários para o aprendizado de um determinado tema sem a necessidade de busca de arquivos espalhados ao longo da internet, evitando-se assim os principais problemas enfrentados no momento de uma prática, como a falta de uma arquivo específico, ou ainda, a incompatibilidade devido a falta de uma imagem de sistema de arquivos específica, por exemplo.

5.3 Prática de redes acessível

O ensino de computação para a formação de profissionais capacitados como um todo é uma tarefa um tanto quanto custosa. Quando tal fato é inspecionado do ponto de

Figura 28 – Organização dos Laboratórios NetEnsina Docker no GitHub.



Fonte: acervo do autor.

vista do ensino de redes de computadores um dos eixos formativos da computação. Os custos envolvidos com as aulas práticas tendem a aumentar de forma rápida e considerável.

Visando compensar tais fatores, buscar introduzir ferramentas, tecnologias ou serviços que auxiliem na redução dos custos à medida que mantenha-se, ou eleve-se a qualidade do ensino prestado são fundamentais. Sendo assim, a ferramenta proposta por este trabalho tem como um de seus fundamentos a acessibilidade do ensino de redes.

Inicialmente criar uma plataforma de ensino que organizasse os arquivos necessários para os laboratórios do NetEnsina Docker, envolveria custos como a contratação de um serviço de hospedagem e domínio. Estes por sua vez, que até a escrita deste documento giravam em torno de R$ 11,00 por mês e R$ 40,00 por ano respectivamente em fornecedores do mercado brasileiro.

Com a finalidade de facilitar a contratação por parte do cliente, uma boa parte das empresas oferecem planos que englobam o pagamento anual tanto do serviço de hospedagem quanto do domínio, com preços que se iniciam por volta dos R$ 387,18 até o momento deste levantamento, variando de acordo com os recursos ofertados pelo plano.

É possível concluir que, se adotada tal ideia, no curto prazo estes preços possivelmente seriam refletidos para os usuários da plataforma, como uma maneira de suprir os gastos. Tendo em vista que alternativas a esta deveriam ser adotadas para manter o objetivo principal, a viabilidade de ferramentas e plataformas populares entre os estudantes de computação mostraram-se mais viáveis e acessíveis.

O GitHub foi escolhido por ser popular entre estudantes e profissionais da computação, e principalmente por alinhar-se com os objetivos buscados com o NetEnsina Docker. Apesar de fornecer recursos premium aos usuários que estiverem dispostos a contratar um plano, a plataforma disponibiliza a todos os cadastrados um meio gratuito de compartilhar e participar de projetos públicos. Ofertando capacidade de armazenamento suficiente para

a organização dos arquivos de roteiros e *dockerfiles*, sem esquecer de mencionar as imagens já publicadas no repositório publico do Docker *Hub*. Vale ressaltar que existem no mercado diversas outras plataformas com serviços, custos e características próprias e direcionadas a mesma finalidade.

5.4 Perspectiva de Utilização

O NetEnsina Docker destina-se à utilização nas disciplinas de redes de computadores dos cursos de computação, tornando-se assim uma ferramenta de auxílio a didática utilizada pelo docente principalmente no que tange às aulas práticas em laboratório necessárias para o eixo formativo dos alunos.

Ao aplicar a abordagem presente no NetEnsina, o professor poderá desenvolver metodologias próprias de ensino prático que favoreçam a fixação do conhecimento dos alunos em relação aos conteúdos teóricos abordados previamente em sala de aula.

A ferramenta Docker que serve de base para o funcionamento dos laboratórios do NetEnsina contribuem para o dinamismo da aulas práticas, o que decorre do fato de que uma mesma máquina pode simular diversas outras unidades computacionais que possuem as próprias interfaces e conexões, possibilitando assim, que cada computador tenha a capacidade de tornar-se um laboratório de redes.

O NetEnsina Docker favorece a prática dos laboratórios em qualquer computador que possua o Docker instalado. Dessa maneira não é mais necessário que o aluno esteja presente nos laboratórios das instituições ou mesmo ceda horários específicos do dia. Os laboratórios armazenados no repositório do Github podem ser facilmente baixados através da internet e executados com poucos comandos. Assim, a prática laboratorial pode ser realizada em qualquer hora ou lugar.

Neste trabalho, podem ser encontrados anexados dois exemplos com a finalidade de complementar o entendimento acerca do que representam os laboratórios virtuais que utilizam Docker *Containers*. Os arquivos de texto apresentam conteúdos que poderiam ser abordados em aulas iniciais das disciplinas práticas de redes de computadores. O primeiro exemplo aborda a comunicação entre dois computadores, através de seus IPs e máscaras de rede; o segundo exemplo demonstra como interligar redes distintas através da configuração manual de rotas estáticas para o encaminhamento de pacotes em dois roteadores.

6 Conclusão

As possibilidades promovidas pelo avanço tecnológico ao longo da evolução humana propiciou diversas modificações nas ações cotidianas quanto nos exercícios laborais, algo que posteriormente atingiria os sistemas educacionais vigentes. As tecnologias aplicadas ao ensino contribuíram para um aumento na produção do conhecimento e compartilhamento de informações entre alunos e professores ao longo de sua evolução. Atualmente encontram-se atrelados ao exercício do ensino dispositivos como *smartphones*, *tablets*, computadores e *datashows*, por exemplo.

Assim como as já citadas tecnologias anteriores, os ambientes virtuais de aprendizagem (AVA) surgiram como uma ferramenta contemporânea no auxílio à educação. Estes, por sua vez, têm como objetivo principal fornecer simulações bastante completas sobre salas de aulas e laboratórios encontrados no mundo real. Contudo, os AVA trazem com a sua utilização diversos benefícios, a portabilidade, onde a utilização será mediante a requisitos básicos como um dispositivo conectado à Internet por exemplo, e também fatores atrelados a economia financeira, algo que decorre do fato da exclusão de uma ambiente físico para o seu correto funcionamento.

Ao longo deste trabalho foi apresentado um ambiente virtual voltado aos cursos de ensino e disciplinas de Redes de Computadores, a fim de atestar a viabilidade destes através do uso de contêineres, mais especificamente contêineres gerenciados pela ferramenta Docker.

Altos custos de implantação e manutenção de laboratórios físicos são os principais problemas enfrentados pelas instituições de ensino que ofertam cursos de tecnologia que possuem a disciplina de Redes de Computadores em sua grade, algo que é agravado quando os olhares se voltam para as instituições pertencentes à rede pública. Tendo em vista os fatores atrelados a esta realidade parte a motivação deste trabalho, algo que pode ser viabilizado pelo uso de contêineres que virtualizam unidades de software permitindo a criação de ambientes virtuais capazes de emular dispositivos e topologias de rede funcionais, e que permitem os experimentos necessárias por partes dos alunos. Além de evitar que hajam custos adicionais, já que é feito o uso de apenas um computador.

Este documento descreve ainda o funcionamento do Docker como ferramenta gestora de contêineres e sua utilização na produção dos laboratórios virtuais, uma vez que a mesma é uma ferramenta de uso gratuito multiplataforma, o que permite a elevação na quantidade de usuários que poderão utilizá-los. Os contêineres são portáteis, logo podem ser executados em qualquer dispositivo que possua o Docker instalado, e possua acesso à Internet para aquisição dos arquivos necessários. Desta forma é possível que o aluno possa

realizar as aulas práticas de Redes de Computadores sem o auxílio de laboratórios físicos, de qualquer local e qualquer computador.

Em acréscimo, foi produzido e disponibilizado um repositório de dados no GitHub, uma das principais ferramentas utilizadas para versionamento de software utilizada por profissionais e empresas, além do uso ser amplamente difundido entre o meio acadêmico. Por intermédio deste repositório, os professores e alunos podem acessar os roteiros preparados e os arquivos necessários para efetuar a prática laboratorial descrita no mesmo. Sendo assim o repositório é um auxiliador na disposição e organização dos itens a serem utilizados nas aulas práticas. Por tratar-se de um repositório aberto, onde qualquer uma pessoa pode contribuir, os professores podem ainda produzir seus próprios roteiros e disponibilizar para seus alunos.

Como trabalhos futuros, pretende-se realizar o refinamento dos laboratórios a fim de melhorar sua eficácia durante um processo de validação da ferramenta NetEnsina Docker em sala de aula, desta maneira verificar como se dá a interação de professores e alunos com a ferramenta, atestando assim, os benefícios atrelados a seu uso no processo de ensino e aprendizagem. Outra possibilidade diz respeito à migração do NetEnsina Docker para um ambiente de nuvem, onde mais uma camada pode ser abstraída dispensando o uso da ferramenta Docker no dispositivo do usuário.

Referências

BERTOCHI, L. O.; BELLEZI, M. A. Virtualização de desktops em grandes ambientes. *Revista TIS*, v. 4, n. 2, 2016. Citado na página 21.

BEZERRA, L. N. M.; SILVEIRA, I. F. Licenciatura em computação no estado de são paulo: uma análise contextualizada e um estudo de caso. In: *XIX Workshop sobre o Ensino de Computação. Anais do CSBC*. [S.l.: s.n.], 2011. Citado na página 18.

BICCA, C. *O que é uma imagem?* 2015. Mundo Docker. Disponível em: <https://vertigo.com.br/o-que-e-conteinerizacao-de-aplicacao/>. Acesso em: 10 Out. 2021. Citado na página 27.

CANONICAL. *What's lxc?* 2017. Canonical. Disponível em: <https://linuxcontainers.org/lxc/introduction/>. Acesso em: 17 Set. 2021. Citado na página 23.

CARISSIMI, A. Virtualização: da teoria a soluções. *Minicursos do Simpósio Brasileiro de Redes de Computadores–SBRC*, v. 2008, p. 173–207, 2008. Citado 2 vezes nas páginas 14 e 25.

COUTINHO, C. *A investigação em "meios de ensino" entre 1950 e 1980: expectativas e resultados*. [S.l.]: Universidade do Minho. Instituto de Educação e Psicologia. Centro de Investigação em Educação, 2006. Citado na página 17.

DOCKER INC. *Docker Overview*. 2021. Docker Docs. Disponível em: <https://docs.docker.com/get-started/overview/>. Acesso em: 17 Set. 2021. Citado 6 vezes nas páginas 34, 35, 36, 37, 38 e 41.

EDUARDO, L. *Namespace, o que é?* 2017. Medium. Disponível em: <https://medium.com/@lets00/namespace-14c4e64d0559>. Acesso em: 17 Set. 2021. Citado na página 23.

FERNANDES, A. *O que é Conteinerização de aplicação?* 2018. Vertigo Tecnologia. Disponível em: <https://vertigo.com.br/o-que-e-conteinerizacao-de-aplicacao/>. Acesso em: 20 Set. 2021. Citado na página 27.

FERREIRA, K.; LIMA, R. de; CHAVES, J.; LIMA, M. de. Inserindo um laboratório virtual para o ensino de redes de computadores. In: *ICBL2013–International Conference on Interactive Computer aided Blended Learning, Florianópolis*. [S.l.: s.n.], 2013. Citado na página 13.

FRANÇA, R. S. de; SILVA, W. C. da; AMARAL, H. J. C. do. Despertando o interesse pela ciência da computação: Práticas na educação básica. In: *Proceedings of International Conference on Engineering and Computer Education*. [S.l.: s.n.], 2013. v. 8, p. 282–286. Citado na página 18.

GAO, X.; GU, Z.; KAYAALP, M.; PENDARAKIS, D.; WANG, H. Containerleaks: Emerging security threats of information leakages in container clouds. In: IEEE. *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. [S.l.], 2017. p. 237–248. Citado 2 vezes nas páginas 28 e 34.

GARCIA, L.; ANTUNES, F.; LARA, D.; RIBEIRO, C. P. Utilização de ambientes virtualizados para ensino de servidores de redes de computadores. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 90. Citado 3 vezes nas páginas 13, 31 e 32.

GOMES, R. *Entendendo o Docker parte 1*. 2015. Medium. Disponível em: <https://medium.com/@gomex/>. Acesso em: 20 Out. 2021. Citado na página 33.

HERPICH, F.; NUNES, F. B.; VOSS, G. B.; JARDIM, R. R.; MEDINA, R. D. Ambiente virtual imersivo para ensino em redes de computadores: uma proposta usando agentes inteligentes. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2014. v. 25, n. 1, p. 65. Citado 2 vezes nas páginas 30 e 31.

JERSAK, L. C. et al. Mapeamento de máquinas virtuais em datacenters privados visando minimizar a interferência de desempenho. Pontifícia Universidade Católica do Rio Grande do Sul, 2014. Citado 2 vezes nas páginas 20 e 21.

JIANG, K.; SONG, Q. A preliminary investigation of container-based virtualization in information technology education. In: *Proceedings of the 16th Annual Conference on Information Technology Education*. [S.l.: s.n.], 2015. p. 149–152. Citado na página 23.

JUNIOR, F. C. S.; LIMA, B. S. N. M. Tecnologia docker. 2015. Citado 5 vezes nas páginas 21, 26, 33, 34 e 35.

LEADCOMM. *Conteinerização de Aplicações*. 2020. Leadcomm. Disponível em: <https://leadcomm.com.br/2020/06/05/conteinerizacao-de-aplicacoes-app-containerization/>. Acesso em: 20 Set. 2021. Citado na página 28.

LINUX VSERVER. *Overview - Linux Vserver*. 2013. Linux Vserver. Disponível em: <http://linux-vserver.org/Overview>. Acesso em: 16 Set. 2021. Citado na página 22.

MACEDO, A. d. S.; SANTOS, C. C. G. *Hypervisor: Segurança em ambientes virtualizados*. 2014. Devmedia. Disponível em: <https://www.devmedia.com.br/hypervisor-seguranca-em-ambientes-virtualizados/30993>. Acesso em: 20 Set. 2021. Citado na página 25.

MACHADO, T. L. D. *Linux Container*. 2017. UFRJ. Disponível em: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2017_2/lxc/>. Acesso em: 17 Set. 2021. Citado na página 24.

MATTOS, D. M. F. *Virtualização*. 2008. Universidade Federal do Rio de Janeiro. Disponível em: <https://www.gta.ufrj.br/grad/08_1/virtual/Virtualizaototalepara-virtualizao.html>. Acesso em: 18 Set. 2021. Citado na página 20.

MORAIS PAULEANY S.; JUNIOR, F. C. S. S. O. S. *Um Estudo sobre a Evasão no Curso de Licenciatura em Informática do IFRN – Campus Natal – Zona Norte*. [S.l.]: WEI, 2015. Citado na página 17.

PRIMO, A. Ferramentas de interação em ambientes educacionais mediados por computador. *Educação*, v. 24, n. 44, p. 127–149, 2001. Citado 2 vezes nas páginas 29 e 30.

RED HAT. *O que é um container Linux?* 2018. Red Hat. Disponível em: <https://www.redhat.com/pt-br/topics/containers/whats-a-linux-container#hist%C3%B3ria-dos-containers>. Acesso em: 16 Set. 2021. Citado 3 vezes nas páginas 22, 33 e 39.

RED HAT. *Containers x máquinas virtuais.* 2020. Red Hat. Disponível em: <https://www.redhat.com/pt-br/topics/containers/containers-vs-vms>. Acesso em: 15 Set. 2021. Citado 2 vezes nas páginas 21 e 27.

RED HAT. *O que é Docker?* 2021. Red Hat. Disponível em: <https://www.redhat.com/pt-br/topics/containers/what-is-docker>. Acesso em: 17 Set. 2021. Citado 2 vezes nas páginas 35 e 39.

RICOY MARíA CARMEN;COUTO, M. J. V. S. *As tic no ensino secundário na matemática em portugal: a perspectiva dos professores.* [S.l.]: Revista latinoamericana de investigación en matemática educativa, Comité Latinoamericano de Matemática Educativa, v. 14, n. 1, p.95–119, 2011. Citado na página 17.

VITALINO, J. F. N.; CASTRO, M. A. N. *Descomplicando o Docker.* [S.l.]: Brasport, 2016. Citado na página 33.

WE LOVE TEACH. *Entenda o que é o LXC (Linux Containers).* 2016. We Love Teach. Disponível em: <https://weloveteach.com/site/artigo/entenda-o-que-e-o-lxc-linux-containers>. Acesso em: 17 Set. 2021. Citado na página 24.

ZHUANG, Z.; TRAN, C.; WENG, J.; RAMACHANDRA, H.; SRIDHARAN, B. Taming memory related performance pitfalls in linux cgroups. In: IEEE. *2017 International Conference on Computing, Networking and Communications (ICNC).* [S.l.], 2017. p. 531–535. Citado na página 26.

Apêndice

Apêndice A – Aula Prática 01 - Uma Rede com dois computadores

Objetivo: Conhecer o funcionamento do laboratório NetEnsina.

Figura 29 – Lab. 01 - Rede com dois computadores.

Fonte: acervo do autor.

Para executar os laboratórios do NetEnsina Docker você necessitará adquirir os arquivos previamente, para isso você pode baixá-los através do repositório do Github, procurando o diretório da aula que deseja praticar.

Sugerimos que mantenha a organização durante seus estudos, então você poderá, se assim desejar, criar uma pasta para armazenar os laboratórios. Apenas lembre-se de que as pastas do NetEnsina possuem a palavra ***lab*** em sua identificação.

Esta é uma aula inicial sobre redes, então seu conteúdo é simples e serve como um meio de familiarizar o aluno na utilização do ambiente NetEnsina Docker. Lembre-se, para iniciar a prática baixe o laboratório da aula em sua pasta pessoal de laboratórios.

- ***Iniciando o laboratório:***

Se você baixou os arquivos do laboratório através do Github Web deve possuir um arquivo compactado, se estiver utilizando Windows extraia o arquivo em sua pasta de laboratórios, em seguida clique com o botão direito em algum lugar vazio dentro

da pasta deste laboratório e abra um terminal, após isso siga para o passo 3 deste documento. Para o Linux siga os passos abaixo:

1. Abra um novo terminal e acesse sua pasta de laboratórios: **cd/home/seunome/lab**
2. Em seguida utiliza o comando de descompactação de arquivo: **tar -xf lab1.tar.gz**
3. Antes de avançar se certifique de estar dentro da pasta principal do laboratório, e localize o arquivo docker compose. Use este comando: **ls**
4. Agora digite no terminal o comando a seguir para iniciar as máquinas virtuais do laboratório: **docker-compose up -d**

   Serão inicializadas duas máquinas virtuais e conectadas a rede criada para o laboratório, elas poderão ser acessadas através de seus nomes, ***CLIENTE*** *e* ***SERVIDOR*** *respectivamente.*
5. Abra um novo terminal, e organize as duas janelas de forma que possa visualizar todas corretamente.

   - ***Vamos Praticar:***
6. Para acessar as máquinas, em cada um dos terminais execute o seguinte comando: **docker container exec -it nomedamaquina /bin/bash**.

   Lembre-se de modificar **"nomedamaquina"** pelos nomes citados anteriormente - ***CLIENTE*** e ***SERVIDOR***.
7. Na máquina **CLIENTE** e **SERVIDOR**, digite o comando **ifconfig** e observe as suas configurações de rede, elas serão importantes a seguir.
8. No **SERVIDOR** ative a ferramenta sniffer ***"tcpdump"*** através do comando: **tcpdump -i eth0 -v -n -s 1600 -w /netensina/lab01.pcap**
9. No **CLIENTE**, efetue um **ping 10.0.0.20**, observe que o **PING** retorna com sucesso de suas tentativas de comunicação. Após algum tempo, utilize as teclas **Ctrl+C** para interromper o comando.
10. Agora no **SERVIDOR** interrompa o tcpdumps com o uso das teclas **Ctrl+C**.
11. Vá até a pasta **netensina** dentro da pasta do laboratório, nela você irá encontrar o arquivo lab01.pcap, use-o para estudar no software **Wireshark**.

    Esta pasta recebe arquivos de monitoramento nas máquinas virtuais do NetEnsina, desta maneira você poderá complementar seu estudo analisando os dados contidos nos arquivos através de outros softwares como o ***Wireshark****.*

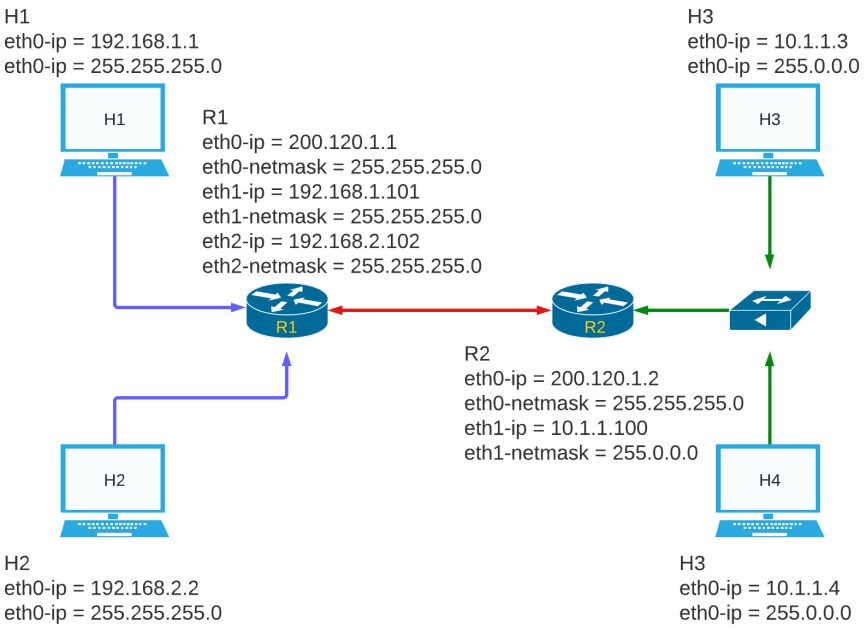
12. Para finalizar o laboratório, utilize um terminal da sua máquina REAL e execute o comando: **docker-compose stop -d**

    Espere até que todas as máquinas sejam encerradas, você perceberá que os terminais perderão o acesso à elas.

Apêndice B – Aula Prática 02 - Conectando Redes utilizando roteadores

Objetivo: Interligar redes distintas, a partir da configuração de roteadores.

Figura 30 – Lab. 02 - Conectando redes através de roteadores.



Fonte: acervo do autor.

Para executar os laboratórios do NetEnsina Docker você necessitará adquirir os arquivos previamente, para isso você pode baixá-los através do repositório do Github, procurando pelo diretório da aula que deseja praticar.

Sugerimos que mantenha a organização durante seus estudos, então você poderá, se assim desejar, criar uma pasta para armazenar os laboratórios. Apenas lembre-se de que as pastas do NetEnsina possuem a palavra ***lab*** em sua identificação.

Esta é uma aula direcionada ao aprendizado de roteadores, nela aprenderemos como configurar dois roteadores para interligarmos duas redes para viabilizarmos a comunicação

entre 04 computadores.

Obs.: Para este laboratório destacamos o uso de duas máquinas virtuais destinadas a emulação dos roteadores presentes na topologia, desta maneira teremos resultados semelhantes ao uso de roteadores virtuais.

- ***Iniciando o laboratório:***

Se você baixou os arquivos do laboratório através do Github Web deve possuir um arquivo compactado, se estiver utilizando Windows extraia o arquivo em sua pasta de laboratórios, em seguida clique com o botão direito em algum lugar vazio dentro da pasta deste laboratório e abra um terminal, após isso siga para o passo 3 deste documento. Para o Linux siga os passos abaixo:

1. Abra um novo terminal e acesse sua pasta de laboratórios: **cd /home/seunome/lab**
2. Em seguida utilize o comando de descompactação de arquivo: **tar -xf lab1.tar.gz**
3. Antes de avançar se certifique de estar dentro da pasta principal do laboratório, e localize o arquivo ***docker-compose***. Use este comando: **ls**
4. Agora digite no terminal o comando a seguir para iniciar as máquinas virtuais do laboratório: **docker-compose up -d**

   *Serão inicializadas 6 máquinas virtuais, onde **R1** e **R2** representam os dois roteadores dentro topologia apresentada na imagem no início do roteiro. Consequentemente **H1**, **H2**, **H3**, **H4** representam os hosts conectados, onde **H1** e **H2** se conectam à mesma rede do primeiro roteador **(R1)**, e **H3**, **H4** conectam-se ao segundo roteador **(R2)**.*
5. Abra novos terminais de maneira que possua quantidade suficiente para manipular todas as máquinas virtuais. Em seguida organize as janelas de maneira que possa visualizar todas corretamente.

- ***Vamos Praticar:***

6. Em cada um dos terminais execute o seguinte comando: **docker container exec -it nomedamaquina /bin/bash**, modificando ***"nomedamaquina"*** pelos nomes citados anteriormente. Assim você ganhará acesso à elas.
7. Neste momento você pode executar o comando **ifconfig** em todas as máquinas e também nos roteadores, desta maneira você poderá observar as configurações de rede dos dispositivos.

   *Você poderá observar que todas as máquinas têm interfaces de rede eth0 ativas, mas apenas os roteadores possuem interfaces adicionais, eth1 em ambos, eth2 apenas no **R1**. Isso ocorre pois o roteador **R1** se conecta às redes da máquina **H1** e **H2**.*

8. Utilize o comando **ping ipdamaquina** da máquina **H3** para H4.

   Lembre-se de substituir ***ipdamaquina*** pelos endereços IPs das máquinas.

9. Agora repita o comando **ping ipdamaquina**, só que agora da máquina **H1** para **H2**.

   *Percebe que não há resposta? Isso ocorre pois **H1** e **H2** pertencem a redes diferentes e ambas as máquinas não conhecem o caminho que leva a informação a outra. Para contornar o problema utilizaremos os roteadores.*

10. Em cada máquina cheque os caminhos conhecidos com o caminho **route**. Você irá perceber que existe rota apenas para a própria interface de rede da máquina.

    Sabendo disso, o próximo passo será demarcar as rotas por onde a informação deve ser direcionada.

11. Iniciando pelo host **H1**, use o comando: **route add default gw 192.168.1.101 dev eth0**

    *Isto adiciona um **gateway default** em **H1**, assim ele poderá redirecionar as requisições que não conhece para um determinada rota.*

12. Em **H2** utilize, **route add default gw 192.168.2.102 dev eth0**

13. Em **H3** utilize, **route add default gw 10.1.1.100 dev eth0**

14. Em **H4** utilize, **route add default gw 10.1.1.100 dev eth0**

    Com os hosts configurados podemos voltar nossa atenção para os roteadores.

15. Para **R1**, execute o comando de adição de rotas da seguinte maneira: **route add -net 10.0.0.0 netmask 255.0.0.0 gw 200.120.1.2 dev eth0**

    *Esta rota define que tudo que **R1** receba destinado a rede 10.0.0.0 seja repassado para **R2**, dessa forma passa a ser tarefa do segundo roteador lidar com essas requisições.*

16. Em **R2**, utilize o comando: **route add -net 192.168.0.0 netmask 255.255.252.0 gw 200.120.1.1 dev eth0**

    Com esta rota tudo está pronto, agora o **R2** pode enviar as requisições direcionadas às redes que **H1** e **H2** pertencem.

17. para fazer o monitoramento dos pacotes ative o ***tcpdump*** nos dois roteadores, use os seguintes comandos:

    Em **R1**: **tcpdump -i eth0 -w /netensina/lab3-R1.pcap**

    Em **R2**: **tcpdump -i eth1 -w /netensina/lab3-R2.pcap**

18. A partir do host **H4** faça um **ping** para H1. Observe que agora é possível a comunicação entre hosts de redes diferentes. Faça isso também entre **H1** e **H2** e observe os resultados. Lembre-se de encerrar o comando com **Ctrl+C** após algum tempo.

19. Encerre agora o **tcpdump** nos dois roteadores utilizando **Ctrl+C**.

20. Vá até a pasta ***netensina*** dentro da pasta do laboratório, nela você irá encontrar o arquivo **lab3-R1.pcap** e **lab3-R2.pcap**, use-o para estudar no software **Wireshark**.

21. Finalize o laboratório, utilize um terminal da sua máquina REAL e execute o comando: **docker-compose stop -d**