LAN: Plataforma Web Baseada em Containers para Aprendizagem Prática da Linha de Comando Linux

Matheus A. Lima
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Macau – RN, Brasil
mathlegeferi@gmail.com

Charles H. Santos
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN)
Macau – RN, Brasil
charles.santos@ifrn.edu.br

Resumo

O processo de ensino e aprendizagem de Sistemas Operacionais como meio para administração de recursos como serviços de rede frequentemente enfrenta desafios ligados à complexidade e eficiência de manter laboratórios informatizados. Enquanto plataformas de nuvem podem atenuar esses problemas, o uso de provedores públicos pode resultar em custos de operação elevados. Por outro lado, o uso de tecnologias de nuvem privada tem potencial de proporcionar às instituições de ensino uma ampliação na oferta de ambientes virtuais para uso pedagógico. Com base nisso, este trabalho propõe o *Linux Access Node*, que utiliza mecanismos baseados em *containers* que oferecem o acesso remoto a terminais em linha de comando para o ensino de sistemas operacionais Linux. Como resultado principal, foi realizada uma implementação de um protótipo da ferramenta em ambiente *web* a ser disponibilizada publicamente, demonstrada por meio de tutorial de utilização.

Palavras-chave: laboratórios virtuais, containers, linha de comando, Linux, computação em nuvem

1 Introdução

O desenvolvimento de tecnologias de educação digitais para ambientes remotos de aprendizagem permitiu o aprimoramento de ensino nas mais variadas áreas do conhecimento. Variados conceitos que seriam de difícil compreensão se tornaram mais acessíveis por meio da integração de recursos de multimídia e de simulação, que auxiliam o processo de assimilação por meio de visualizações de definições abstratas em cenários reais [4]. Na área de Ciência da Computação, os Ambientes Virtuais de Aprendizagem (AVA) ainda devem incorporar meios para que os alunos sejam capazes de aplicar os conhecimentos adquiridos em diversos tópicos específicos como algoritmos e sistemas distribuídos [14] [18].

No entanto, a aplicação de tecnologias digitais no ensino enfrenta diversos desafios na sua concepção, devido à necessidade da realização de adaptações nos componentes curriculares, na infraestrutura de informática e na capacitação dos professores [5]. Além disso, a falta de inclusão digital em diversas comunidades e a ausência de equipamentos nas residências são fatores que impedem um aprendizado ótimo fora da sala de aula. Especificamente componentes curriculares relacionados à computação, como desenvolvimento de software, sistemas operacionais e redes de computadores, muitas aplicações possuem requisitos mínimos de operação na qual os alunos dessas comunidades não têm condições de suprir [3].

Entre as soluções habilitadoras que permitem um ambiente de aprendizagem de baixo custo no âmbito de comunidades de baixa renda está a computação em nuvem. Este paradigma de comunicação permite que os recursos de aprendizagem estejam localizados em servidores remotos, de tal maneira a permitir que sejam acessados em qualquer dispositivo com acesso à Internet. Nesse sentido, a materialização de plataformas de *e-learning* em ambientes de *cloud* faz com que esta seja uma solução popular de ensino, proporcionando mais flexibilidade no processo de aprendizagem [2].

Ao adotar e-learning sobre o paradigma de computação em nuvem, é necessário fazer diversas considerações sobre o modelo de infraestrutura a ser utilizado. Entre essas considerações está a escolha entre *cloud* pública, onde plataformas como Google, Amazon e Meta disponibilizam seus servidores para hospedagem dos recursos, e *cloud* privada, em que as instituições empregam seus próprios servidores para provimento de serviços [10].

Um dos principais fatores para esta escolha é o custo associado à aquisição e operação dos artefatos computacionais. O modelo público consiste, geralmente, em sistemas de *leasing* (aluguel), na qual o pagamento é realizado à medida que os recursos forem utilizados. Por outro lado, ao adotar uma *cloud* privada, é necessário levar em consideração o custo de aquisição e manutenção dos equipamentos, assim como desgaste de peças e desvalorização dos ativos adquiridos [8].

Ambos os modelos podem ser eficientes a depender dos requisitos exigidos para a infraestrutura. Entretanto, em se tratando do ensino de sistemas operacionais, especialmente aqueles voltados para serviços de rede, muitos recursos utilizados podem ser custosos em todos os cenários devido aos requerimentos computacionais de rede, memória e processamento. Um exemplo de recurso são as máquinas virtuais (VM, do inglês *Virtual Machine*), que, ao emular um sistema operacional completo, consomem grande parte da capacidade computacional dos servidores ao isolá-las a nível de

hardware [16]. Isso significa que a capacidade reservada não pode ser utilizada por outras aplicações.

Como alternativa, soluções baseadas em *containers*, que isolam recursos a nível de aplicação, fornecem um ambiente de desenvolvimento com baixos requisitos de operação, tendo em vista que todos os elementos compartilham um único sistema operacional [16]. Esta tecnologia pode auxiliar especificamente no ensino de administração de sistemas, em que os alunos são estimulados a configurar serviços de rede em um sistema operacional, tipicamente Linux.

Nesse contexto, este trabalho visa abordar possíveis desafios sobre a adoção de um sistema cloudificado para ensino em administração de sistemas, como autenticação e disponibilização de uma plataforma para a realização de práticas, além do isolamento de recursos entre os usuários. Com isso em mente, foi desenvolvido o Linux Access Node (LAN), uma aplicação *cloud* de baixo custo computacional que fornece acesso a terminais Linux de forma remota. Esses ambientes têm a finalidade de serem uma solução viável para a problemática discutida e, para isso, a ferramenta utiliza *containers* para que seu custo de operação não impeça seu uso.

A implementação inicial do LAN consiste em um protótipo desenvolvido com a finalidade de implantar as funcionalidades básicas previstas para a ferramenta. A partir disso, busca-se estabelecer uma primeira validação que permitirá, por meio de sistemas colaborativos, um aperfeiçoamento no emprego das técnicas de fornecimento do ambiente de aprendizado proposto. Como resultado, são disponibilizados tutoriais de uso e o código-fonte para a ferramenta.

Dessa forma, a estrutura deste artigo é organizada da seguinte maneira: a Seção 2 apresenta os trabalhos relacionados e discute suas limitações frente às necessidades de ambientes educacionais, a Seção 3 descreve detalhadamente a plataforma LAN, incluindo sua arquitetura, tecnologias utilizadas e fluxo de funcionamento, a Seção 4 demonstra o funcionamento da ferramenta por meio de uma demonstração e, por fim, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2 Trabalhos Relacionados

Para atenuar os desafios mencionados na Seção 1, foi constatado que o laboratório virtual deve apresentar tais características: baixo custo operacional, propósito pedagógico, sessões sem limite de tempo e ambientes persistentes. Diversas soluções exploram a virtualização de ambientes Linux. No entanto, parte das aplicações disponíveis não é voltada ao contexto educacional e, entre aquelas que possuem esse foco, é comum encontrar limitações que inviabilizam um uso contínuo em sala de aula, como sessões temporárias.

2.1 Play With Docker

O Play With Docker é uma ferramenta oficial do Docker [7] que permite a interação com *containers* Linux com propósito educacional. Embora seja adequado para aprendizado inicial e livre a aplicação não é focada no ensino de Linux, mas sim de Docker. Ademais, ela também apresenta restrições importantes, como tempo de acesso limitado e a capacidade de processamento é reduzida, o que compromete sua utilização em cenários educacionais contínuos.

2.2 LabEx

O LabEx é um site educacional que oferece cursos de computação acompanhados de laboratórios interativos [11]. Neste ambiente, os alunos podem praticar diretamente em ambientes Linux remotos, porém com funcionalidades limitadas. Em seu plano gratuito, a ferramenta permite a criação de apenas três máquinas virtuais por dia, cada uma com tempo de uso restrito. Além disso, seu sistema apresenta limitações de desempenho, não suportando aplicações mais pesadas.

2.3 Amazon Web Services

A AWS (Amazon Web Services) oferece diversos serviços e funcionalidades, incluindo o uso de *containers* Linux eficientes e com persistência [1]. No entanto, essa aplicação é feita para ambientes profissionais, não sendo adequada para contextos educacionais, visto que apresenta uma curva de aprendizado acentuada. Para realizar tarefas como criação e o gerenciamento de instâncias, *containers* e redes virtuais, é necessária a aquisição conhecimentos técnicos e familiaridade com sua infraestrutura em nuvem. Além disso, sua versão gratuita oferece tempo e recursos (capacidade de processamento, armazenamento e memória) limitados, impedindo seu uso para atividades acadêmicas.

2.4 Container-Based Virtual Laboratory (CVL)

Esta proposta apresenta desenvolvimento de um laboratório virtual baseado em *containers* Linux, aplicado no ensino de cibersegurança [17]. Cada estudante tem acesso remoto a um ambiente isolado composto por múltiplos *containers* interligados, que simulam uma rede corporativa. Nesse ambiente, é possível configurar políticas de *firewall*, monitorar tráfego e realizar testes de segurança diretamente via terminal.

A solução foi projetada para ser de baixo custo e facilmente escalável, utilizando recursos de virtualização leve do Docker. O estudo mostrou alta aceitação dos alunos quanto à usabilidade e desempenho, reforçando a viabilidade do uso de *containers* em contextos educacionais.

2.5 Discussão

Os itens a seguir sumarizam os trabalhos relacionados nesta seção:

Resumo comparativo:
- Play With Docker: gratuito, mas com sessões temporárias e baixo desempenho.
- LabEx: paga, porém com versão gratuita que possui fortes limitações de tempo e desempenho.

- AWS: oferece persistência e escalabilidade, porém é complexa e paga.
- CVL: gratuito e persistente, mas restrito a alunos da instituição.

Com base na análise dos trabalhos relacionados, percebe-se que os serviços gratuitos geralmente têm grandes limitações, enquanto as plataformas pagas mostram-se inviáveis para instituições com recursos reduzidos. Diante disso, sistemas privados como o CVL vêm sendo desenvolvidos como alternativas, destacando o potencial de sistemas containerizados em plataformas de ensino. Nesse contexto, o *Linux Access Node* se apresenta como solução para o aprendizado da linha de comando do Linux, possibilitando que estudantes tenham acesso a uma plataforma virtual acessível para interação remota.

3 Linux Access Node

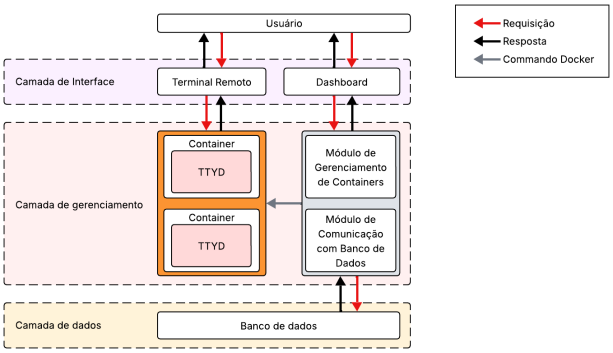

Figura 1. Arquitetura do Linux Access Node

O *Linux Access Node* (LAN) é um sistema desenvolvido com o intuito de fornecer acesso remoto a terminais Linux em um ambiente educacional acessível e de baixo custo, voltado à administração de sistemas operacionais. Seu objetivo é disponibilizar um ambiente remoto adaptado a instituições com recursos limitados. A aplicação se baseia em *containers* para treinamento e experimentação, sendo disponibilizados remotamente via interface *web*.

Os *containers* utilizados no LAN são baseados em um modelo que usa a tecnologia *TTYD (Teletypewriter Daemon)* [20], um programa de código aberto que permite compartilhar um terminal Linux por navegadores *web*. Além disso, o modelo de *containers* é baseado na distribuição Ubuntu do Linux, assim permitindo que os alunos inicializem as sessões de terminal remotamente.

Para essa finalidade, o LAN integra as funcionalidades necessárias para fornecer uma interface para o usuário, gerenciamento das sessões remotas e dos dados utilizados. Sua arquitetura funcional é apresentada na Figura 1, que contém os componentes da ferramenta, organizados nas camadas que integram suas principais funções.

3.1 Arquitetura Funcional

A arquitetura do sistema é composta por três camadas: interface web, gerenciamento de *containers* e persistência de dados. Essa separação busca proporcionar isolamento entre usuários, escalabilidade e segurança.

Na camada de interface, temos duas dois modos de interação com o usuário, denominadas "*dashboard*"e "terminal". *Dashboard* é a página *web* em que o usuário pode criar, editar e renomear seus *containers*, se comunicando, para isso, com os módulos da camada de gerenciamento. Para regulamentar o acesso, o usuário precisa criar e acessar sua conta para gerenciar seus *containers* a partir das páginas de cadastro e de *login*. Ao realizar o registro, o usuário acessa o "terminal", que representa a página em que o usuário acessa a linha de comando do *container* diretamente através do TTYD.

Na camada de gerenciamento, o "Módulo de Gerenciamento de *Containers*"é responsável por executar as instruções do usuário, como criar e excluir os ambientes, assim como executar os comandos via terminal remoto. Por sua vez, o "Módulo de Comunicação com o Banco de Dados"é responsável por buscar ou modificar informações na camada de dados a pedido da camada de interface, sendo crucial para o processo de cadastro, *login* e gerenciamento dos *containers*. Esta camada também é responsável pela criptografia das credenciais antes de as registrar no banco de dados.

A camada de dados é onde se armazenam informações essenciais para o funcionamento da aplicação como o nome dos usuários, suas senhas, os nomes dos *containers* e quais usuários são seus criadores e as suas datas de criação por meio de um banco de dados. Para atender aos requisitos do trabalho, o sistema conta com boa integração com aplicações *web*, alto desempenho e facilmente escalável.

3.2 Tecnologias utilizadas

Para a criação dos *containers*, utiliza-se o Docker, uma plataforma amplamente adotada devido à sua portabilidade, facilidade de configuração e compatibilidade com diversos sistemas [6]. Sistemas baseados em Docker apresentam elevada eficiência por adotarem um modelo de isolamento a nível de sistema operacional, no qual múltiplos contêineres compartilham o mesmo *kernel* e as mesmas camadas de imagem. Dessa forma, elimina-se a necessidade de replicar um sistema operacional completo para cada instância de virtualização, reduzindo significativamente o consumo de armazenamento e memória, acelerando os tempos de inicialização e aumentando a densidade de aplicações executadas em um único *host*. A adoção de *containers* no LAN se apoia em evidências presentes na literatura, que demonstram inicializações mais rápidas e melhor utilização de recursos quando comparados a máquinas virtuais [15].

Para o armazenamento dos dados utilizados pelo sistema, foi escolhido o Supabase. Essa plataforma oferece um conjunto robusto de serviços voltados a aplicações modernas,

incluindo autenticação nativa, banco de dados relacional baseado em *PostgreSQL* e APIs geradas automaticamente. A adoção do Supabase se justifica por sua capacidade de fornecer um ambiente seguro, escalável e de baixa complexidade operacional [19].

A aplicação foi desenvolvida utilizando Node para o *backend* e uma interface web construída com React. Essa combinação foi escolhida pela facilidade de integração com *APIs REST* e pela leveza na comunicação com o Docker e o Supabase [12, 13].

O projeto também conta com um mecanismo de instalação automatizada, implementado por meio de um Makefile, responsável por configurar dependências essenciais, preparar o ambiente de contêineres e validar possíveis erros de execução [9].

Além disso, foram adotadas práticas de segurança voltadas à proteção de dados e à prevenção de ataques web, utilizando *middlewares* específicos para controle de cabeçalhos HTTP e criptografia de informações sensíveis. Essas medidas reforçam a integridade da comunicação entre as camadas da aplicação e reduzem a superfície de ataque. No ambiente de terminal, o usuário possui acesso restrito ao seu próprio *container*, impossibilitando qualquer interação direta com o *host* do sistema e mitigando tentativas de exploração ou elevação de privilégios.

3.3 Metodologia de desenvolvimento

O projeto foi desenvolvido de forma incremental, com seus módulos sendo trabalhados separadamente. Essa abordagem facilitou a depuração e o aperfeiçoamento do sistema. Seguindo uma metodologia de desenvolvimento iterativa, ciclos curtos de implementação e revisão foram adotados. Antes da integração ao sistema principal, cada mudança foi devidamente planejada e testada, garantindo estabilidade entre as camadas. Os testes foram realizados manualmente, avaliando o funcionamento de cada componente e a forma como o sistema reage diante de falhas na comunicação entre as aplicações que o compõem.

As ferramentas utilizadas no processo de desenvolvimento foram o Visual Studio Code, como ambiente de programação, e o Git e GitHub, para controle de versão.

4 Demonstração

Com o objetivo de demonstrar esse fluxo e validar a aplicabilidade do LAN no ensino de sistemas operacionais, foi produzido um vídeo demonstrativo disponibilizado em plataforma online[1] . O material apresenta desde o processo de instalação do sistema até a interação prática com os ambientes Linux disponibilizados aos usuários.

4.1 Instalação da Plataforma

A instalação é realizada no *host* Linux por meio do comando `make install`. Esse procedimento engloba a atualização do sistema, instalação de ferramentas essenciais (curl, git, bibliotecas de compilação e Docker), sincronização do submódulo Git contendo a imagem utilizada pela aplicação, desativação do IPv6 no Docker para evitar conflitos de rede e construção da imagem via BuildKit. Em seguida, o instalador configura o *Node Version Manager* (NVM) e define automaticamente a versão LTS do Node.js.

Após essas etapas, a aplicação se torna acessível na rede interna por meio do comando `make run`, que inicializa o *frontend* e o *backend*. A partir disso, basta inserir o endereço IP do *host* no navegador para acessar o sistema.

A demonstração foi gravada utilizando duas máquinas virtuais idênticas: uma representando o servidor e outra o usuário. Essa separação simula o cenário de uso típico em instituições de ensino, onde o servidor é administrado pelo docente e os alunos acessam os ambientes remotamente.

4.2 Descrição da Demonstração

Na máquina virtual do servidor, o repositório oficial do projeto é obtido via GitHub[2] e, em seguida, inicia-se o processo de instalação previamente descrito. Após a configuração, a aplicação é executada e disponibilizada na rede.

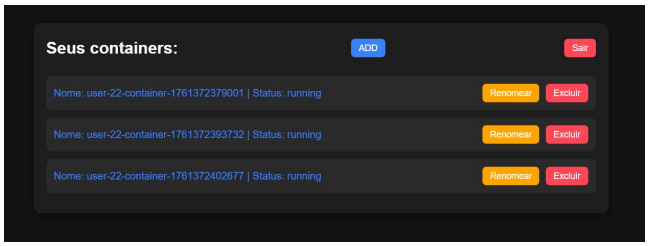

Figura 2. Dashboard

Figura 3. Terminal remoto

Na máquina virtual do usuário, o vídeo exibe o processo de registro de uma nova conta, seguido do *login* e do acesso ao *dashboard* (Figura 2). Em seguida, são criados três ambientes distintos; um deles é aberto no terminal remoto (Figura 3), demonstrando o funcionamento completo da plataforma.

[1] https://www.youtube.com/watch?v=gmrAZWN-HQo

[2] https://github.com/MatheusLima505/Linux-Access-Node

Comandos executados na VM do servidor.

```
git clone https://github.com/MatheusLima505/
    Linux-Access-Node

cd Linux-Access-Node/
make install
make run
```

5 Conclusão

A pesquisa apresentou o *Linux Access Node*(LAN), uma ferramenta desenvolvida para fornecer ambientes Linux remotos, persistentes e de baixo custo para o ensino de administração de sistemas e redes. O sistema foi construído utilizando *containers* Docker, TTYD, Supabase e um processo de instalação automatizado, permitindo implantação simples em servidores locais e operação via navegador. A demonstração prática comprovou que o fluxo de uso é intuitivo, permitindo criação, acesso e gerenciamento de ambientes sem necessidade de *hardware* dedicado.

As demonstrações realizadas sugerem que o uso de *containers* oferece inicialização rápida e facilidade no uso da solução, tornando o LAN adequado para instituições com infraestrutura limitada. Dessa forma, o protótipo implementado pavimenta o desenvolvimento de laboratórios virtuais para o ensino de gerenciamento de sistemas operacionais. Como trabalhos futuros, destacam-se a análise de desempenho do sistema, integração com sistemas acadêmicos, o controle automático de recursos dos containers, além de novas funcionalidades que permitirão práticas mais complexas como a configuração de serviços de redes.

Referências

[1] Inc. Amazon Web Services. 2025. *Amazon Web Services (AWS).* https://aws.amazon.com/ Acesso em: 5 nov. 2025.

[2] Jorge Ariza et al. 2021. Provisioning computational resources for cloud-based e-learning platforms using deep learning techniques. *IEEE Access* 9 (2021), 89798–89811.

[3] Cetic. 2023. Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros - TIC Domicílios 2023. https://cetic.br

[4] F. G. de B. Garbin, C. A. R. Carvajal, and A. Guilherme. 2024. TICS no contexto do ensino superior: desafios e oportunidades em tempo de quarentena. *Interfaces Científicas - Educação* 12, 2 (2024), 38–52. doi:10.17564/2316-3828.2024v12n2p38-52

[5] Jonathan Porto Galdino do Carmo et al. 2024. Utilização de tecnologias digitais na educação infantil em comunidades de baixa renda. *Revista Foco (Interdisciplinary Studies Journal)* 17, 8 (2024).

[6] Docker Inc. 2025. *Docker Documentation.* https://docs.docker.com/ Acesso em: 8 nov. 2025.

[7] Docker Playground. 2025. *Play With Docker.* https://www.docker.com/play-with-docker/ Acesso em: 5 nov. 2025.

[8] Patrick Dreher et al. 2017. Cost analysis comparing HPC public versus private cloud computing. In *Cloud Computing and Services Science: 6th International Conference, CLOSER 2016, Rome, Italy, April 23-25, 2016, Revised Selected Papers 6.* Springer International Publishing, 294–316.

[9] Free Software Foundation. 2025. *GNU Make Manual.* https://www.gnu.org/software/make/ Acesso em: 03 dez. 2025.

[10] Abhishek Gautam. 2022. Cloud computing: A review paper. *International Journal for Research in Applied Science and Engineering Technology* 10, 6 (2022), 3233–3236.

[11] Technology Limited LabEx. 2025. *Laboratórios Gratuitos de Linux, DevOps e Cibersegurança | LabEx.* https://labex.io/pt Acesso em: 23 set. 2025.

[12] Meta Platforms. 2025. *React — A JavaScript library for building user interfaces.* https://react.dev/ Acesso em: 8 nov. 2025.

[13] OpenJS. 2025. *Node.js — JavaScript Runtime Environment.* https://nodejs.org/ Acesso em: 8 nov. 2025.

[14] Ludmila Peca. 2023. E-Learning Experiences in the Computer Networks Course in Higher Education. *Intellectus* (2023), 62.

[15] Akshay Potdar, Jalpa Patel, et al. 2020. Performance Evaluation of Docker Container and Virtual Machine. *Procedia Computer Science* 171 (2020), 1417–1426.

[16] Ramalingam Saravanan. 2014. Creating a browser-based virtual computer lab for classroom instruction. In *SciPy.* 72–78.

[17] Javier Soriano, Carlos Delgado Kloos, María Ángeles Fernández, and Antonio Robles-Gómez. 2020. Students' Acceptance and Tracking of a New Container-Based Virtual Laboratory. *Applied Sciences* 10, 3 (2020), 1091. https://www.mdpi.com/2076-3417/10/3/1091 Acesso em: 5 nov. 2025.

[18] Mohammad Aman Sultani, Mohammad Nazim Kabiri, and Muhammad Wannous. 2018. Utilization of cloud technologies in building a virtual programming lab for higher education in Afghanistan work in progress. In *2018 IEEE International Conference on Applied System Invention (ICASI).* IEEE, 422–425.

[19] Supabase Inc. 2025. *Postgres Database — Features.* https://supabase.com/features/postgres-database Acesso em: 8 nov. 2025.

[20] Tao Zhu. 2025. *TTyD - Share your terminal over the web.* https://github.com/tsl0922/ttyd Acesso em: 8 nov. 2025.