Universidade Federal de Ouro Preto
Escola de Minas
Engenharia de Controle e Automação

Hugo Oliveira Rocha

DevSecOps: Infraestrutura de Nuvem como Código (IaC) e a Orquestração de Contêineres via Pipelines CI/CD com Segurança.

Ouro Preto
2026

Hugo Oliveira Rocha

DevSecOps: Infraestrutura de Nuvem como Código (IaC) e a Orquestração de Contêineres via Pipelines CI/CD com Segurança.

Trabalho apresentado ao Colegiado do Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro(a) de Controle e Automação.

Orientador: Prof. Dr. Tiago Garcia De Senna Carneiro
Coorientador: Prof. Me. João Carlos Vilela de Castro

Ouro Preto
2026

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Hugo Oliveira Rocha

DevSecOps: Infraestrutura de Nuvem como Código (IaC)
e a Orquestração de Contêineres via Pipelines CI/CD com Segurança

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheiro de Controle e Automação

Aprovada em 27 de julho de 2026.

Membros da banca

Dr. Tiago Garcia de Senna Carneiro - Orientador (Universidade Federal de Ouro Preto)
Me. João Carlos Vilela de Castro - Coorientador (Universidade Federal de Ouro Preto)
Dr. Joubert de Castro Lima - (Universidade Federal de Ouro Preto)
Dr. Rodrigo César Pedrosa Silva - (Universidade Federal de Ouro Preto)

Tiago Garcia de Senna Carneiro, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 29/07/2026

Documento assinado eletronicamente por **Tiago Garcia de Senna Carneiro**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 30/07/2026, às 01:04, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1147229** e o código CRC **1344C4EF**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

"A ambição de reimaginar como a equipe oferece seus serviços está ligada à automação."

— Jha et al., 2023.

Resumo

Este projeto propôs a implementação e validação de uma base processual, ferramental e cultural de DevSecOps no laboratório de pesquisa e capacitação de software Terralab, vinculado ao Departamento de Computação da Universidade Federal de Ouro Preto (UFOP). O modelo clássico de desenvolvimento de software frequentemente gera segregação entre as equipes de desenvolvimento e operações, resultando em interrupções, lentidão e falhas. Para mitigar esses problemas e elevar a maturidade do processo, a pesquisa baseou-se em um estudo de caso focado na reestruturação e migração do provedor de nuvem do laboratório utilizando uma abordagem *greenfield*. A nova infraestrutura foi provisionada e gerenciada integralmente via Infraestrutura como Código (IaC), incorporando *pipelines* de Integração e Entrega Contínuas (CI/CD) com ferramentas de segurança automatizadas (SAST) e práticas de *Site Reliability Engineering* (SRE) para o monitoramento de SLIs e SLOs. A adoção deste modelo visou agilidade operacional, aumento na frequência de *deploys* e conformidade de segurança proativa (*shift-left*) , consolidando uma cultura de responsabilidade compartilhada e melhoria contínua.

Palavras-chave: DevSecOps. Infraestrutura como Código. CI/CD. Computação em Nuvem. SRE.

Abstract

This project proposed the implementation and validation of a procedural, tooling, and cultural DevSecOps baseline in the Terralab software development laboratory, affiliated with the Federal University of Ouro Preto (UFOP). The classic software development model often creates silos between development and operations teams, resulting in disruptions, delays, and failures. To mitigate these issues and elevate process maturity, the research was based on a case study focused on the restructuring and migration of the laboratory's cloud provider using a greenfield approach. The new infrastructure was provisioned and managed entirely via Infrastructure as Code (IaC), incorporating Continuous Integration and Continuous Delivery (CI/CD) pipelines with automated security tools (SAST) and Site Reliability Engineering (SRE) practices for monitoring SLIs and SLOs. The adoption of this model aimed to promote operational agility, increased deploy frequency, proactive security compliance (shift-left), and a reduction in Mean Time to Recovery (MTTR), thereby consolidating a culture of shared responsibility and continuous improvement.

Keywords: DevSecOps. Infrastructure as Code. CI/CD. Cloud Computing. SRE.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

CI/CD	*Continuous Integration and Continuous Delivery* (Integração Contínua e Entrega Contínua)
Embrapa	Empresa Brasileira de Pesquisa Agropecuária
Fiocruz	Fundação Oswaldo Cruz
GCP	*Google Cloud Platform*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IaC	*Infrastructure as Code* (Infraestrutura como Código)
INPE	Instituto Nacional de Pesquisas Espaciais
IP	*Internet Protocol*
KPI	*Key Performance Indicator* (Indicador-Chave de Desempenho)
MFA	*Multi-Factor Authentication* (Autenticação Multifator)
MTTR	*Mean Time to Recovery* (Tempo Médio de Recuperação)
RAM	*Random Access Memory* (Memória de Acesso Aleatório)
RDP	*Remote Desktop Protocol*
SaaS	*Software as a Service* (Software como Serviço)
SAST	*Static Application Security Testing* (Teste Estático de Segurança de Aplicações)
SLA	*Service Level Agreement* (Acordo de Nível de Serviço)
SLI	*Service Level Indicator* (Indicador de Nível de Serviço)
SLO	*Service Level Objective* (Objetivo de Nível de Serviço)
SRE	*Site Reliability Engineering* (Engenharia de Confiabilidade de Sítios)
SSH	*Secure Shell*
TLS/SSL	*Transport Layer Security / Secure Sockets Layer*

UFOP	Universidade Federal de Ouro Preto
VPN	*Virtual Private Network* (Rede Privada Virtual)

Sumário

1 Introdução

Nos ambientes tradicionais de desenvolvimento de software, os times de desenvolvimento e operações trabalham isoladamente. Cada um deles trabalhava com objetivos diferentes: o time de desenvolvimento entregava o código para a equipe de operações, que, por sua vez, se preocupava em manter a disponibilidade, a confiabilidade e o desempenho. Essa disparidade entre os times frequentemente causava problemas como interrupções, um time culpando o outro, ciclos de desenvolvimento lentos, falta de padronização, falhas, estresse interno e insatisfação por parte do consumidor (JHA *et al.*, 2023). A solução para esses problemas foi criar um ambiente de colaboração entre esses dois times dando início à cultura DevOps (LEITE *et al.*, 2019). O DevOps visa um ciclo de desenvolvimento rápido enfatizando automação e promovendo colaboração entre Desenvolvimento e Operações. Uma evolução natural desse modelo é o DevSecOps, que visa integrar a segurança em todas as partes do processo de desenvolvimento (RAJAPAKSE *et al.*, 2022), bem como nas operações.

1.1 Justificativa

A adoção de práticas DevSecOps tem se tornado essencial para organizações e laboratórios de pesquisa que buscam agilidade operacional sem comprometer a qualidade e a segurança em seus processos de desenvolvimento de *software* (KATHIRESAN, 2022). Nesses ambientes, um dos principais desafios reside em garantir a evolução contínua dos serviços mantendo alta disponibilidade e resiliência, mesmo diante de fluxos de trabalho colaborativos com alta rotatividade de membros (como estudantes em formação).

Nesse contexto, este trabalho justifica-se pela necessidade de estabelecer uma base sólida e automatizada de infraestrutura que previna a introdução de vulnerabilidades de segurança e falhas operacionais em ambiente de produção. A implementação de Infraestrutura como Código (IaC) integrada a esteiras de CI/CD e práticas de Engenharia de Confiabilidade de Sítios (SRE) viabiliza a padronização, rastreabilidade e auditabilidade dos processos, analisando quantitativamente os impactos da cultura DevSecOps na redução da superfície de ataque e na estabilização dos recursos computacionais.

1.2 Objetivo

Um time de DevSecOps é composto por desenvolvedores e operadores de software que dispõem de um conjunto de métodos e ferramentas para aumentar a eficiência e

qualidade do processo de desenvolvimento. Jha *et al.* (2023) definem sete princípios fundamentais do DevOps (Seção 3.4), que serão apresentados como os pilares do DevOps.

O objetivo principal deste trabalho foi desenvolver, implantar e avaliar uma base processual, ferramental e cultural sólida de DevSecOps no laboratório Terralab. Para viabilizar a mensuração prática e quantitativa deste objetivo central, foram definidos os seguintes objetivos específicos:

- Reduzir o número de portas administrativas expostas publicamente.
- Automatizar o processo de construção e implantação (*deploy*) das aplicações por meio de contêineres.
- Reduzir a quantidade de tentativas de conexões não autorizadas que alcançam os servidores internos de produção.
- Definir e acompanhar Objetivos de Nível de Serviço (SLOs) primários, como a disponibilidade (*Uptime*) da infraestrutura.
- Consolidar a separação de responsabilidades e promover a percepção de segurança transversal na equipe (responsabilidade compartilhada).

O desenvolvimento deste trabalho foi norteado pelos 7 pilares explicados a seguir.

1.2.1 Pilar 1: Ações orientadas para o cliente

O feedback do cliente é essencial para o desenvolvimento de software, um time de DevOps deve sempre investir em estratégias e serviços que garantem o maior grau de satisfação por parte do usuário.

1.2.2 Pilar 2: Considere opções com um objetivo final

Desenvolvedores precisam produzir software com a mentalidade de fazer algo que vai ser útil para o consumidor, não basta adotar uma tecnologia ”moderna”ou um processo ”ágil”se isso não contribuir para o resultado esperado (pelo usuário).

1.2.3 Pilar 3: Responsabilidade do início ao fim

As equipes devem assumir responsabilidade integral pelo ciclo de vida completo do software. No contexto DevSecOps, isso se traduz em uma atuação abrangente que vai desde o desenvolvimento até a operação e segurança da infraestrutura. O engenheiro de DevSecOps, seguindo este princípio, gerencia todo o fluxo de valor tecnológico da organização. Suas responsabilidades incluem:

- A construção e manutenção do pipeline de Integração Contínua e Entrega Contínua (CI/CD - do inglês, Continuous Integration and Continuous Delivery)
- A gestão da infraestrutura como código (IaC - do inglês, Infrastructure as Code)
- A segurança contínua da cadeia de suprimentos de software
- O monitoramento proativo dos ambientes de produção - Observabilidade

1.2.4 Pilar 4: Equipes autônomas multifuncionais

Conforme estabelecido por Jha *et al.* (2023), o quarto princípio do DevOps defende a formação de equipes auto suficientes e multidisciplinares. No contexto DevSecOps, essa abordagem se expande para incluir competências em segurança em todas as fases do ciclo de desenvolvimento. Enquanto no desenvolvimento clássico um profissional poderia se especializar apenas em desenvolvimento backend ou administração de sistemas, o modelo DevSecOps exige uma visão sistêmica de todo o fluxo de valor, capacidade de trabalhar em múltiplas camadas tecnológicas e habilidade para tomar decisões autônomas considerando desenvolvimento, operações e segurança.

1.2.5 Pilar 5: Melhoria Contínua

O mercado está sempre mudando e requer adaptação contínua. No contexto do DevSecOps é esperado a melhoria contínua da infraestrutura sempre com ênfase na segurança e velocidade do processo de desenvolvimento. Experimentação é também uma parte importante do processo, permitindo o aprendizado com os erros e desenvolvimento em um ambiente seguro.

1.2.6 Pilar 6: Automatize o máximo que puder

”A capacidade de automatizar o ciclo de desenvolvimento de software (incluindo entrega contínua e alocação de recursos dinamicamente) e todo o cenário de nuvem, construindo plataformas de virtualização baseadas em contêineres é o que permite que a estrutura seja versionada e considerada como código. A ambição de reimaginar como a equipe oferece seus serviços está ligada à automação”(JHA *et al.*, 2023).

1.2.7 Pilar 7: Segurança em todas as etapas

Embora Jha *et al.* (2023) não incluam explicitamente a segurança como um princípio isolado, o DevSecOps amplia essa estrutura ao tornar a segurança transversal ou um sétimo pilar. Isso implica incorporar práticas como ’Shift Left Security’ e governança proativa desde a especificação e projeto do sistema.

2 Revisão Bibliográfica

A adoção de DevSecOps como modelo de mediação entre desenvolvimento, operações e segurança exige uma base teórica sólida, que abranja desde seus princípios básicos até as práticas e ferramentas consolidadas no mercado. Esta seção apresenta os conceitos essenciais que embasam a implementação proposta para o Terralab, além de um panorama dos trabalhos relacionados que demonstram a aplicabilidade do modelo em contextos similares.

2.1 Conceitos básicos

No contexto de DevSecOps, é fundamental compreender uma série de conceitos inter-relacionados que sustentam a integração entre desenvolvimento, operações e segurança, que foram definidos por Bass, Weber e Zhu (2015). A infraestrutura em nuvem (Cloud Computing) oferece a base para a entrega de recursos computacionais escaláveis e flexíveis, permitindo que organizações acessem servidores, armazenamento e serviços de software sob demanda. Essa infraestrutura é frequentemente gerenciada por meio de pipelines, fluxos automatizados que coordenam desde a escrita do código até sua implantação em produção, incorporando práticas como integração contínua (CI) e entrega contínua (CD). A CI assegura que alterações de código sejam verificadas automaticamente, enquanto o CD garante que o software esteja sempre pronto para ser liberado de forma segura e eficiente. A comunicação entre sistemas na nuvem, geralmente, depende de componentes como servidores *web*, responsáveis por processar solicitações no protocolo HTTP e entregar conteúdo a usuários, e proxies, que atuam como intermediários para otimizar desempenho e segurança. Um *proxy* reverso, em particular, é essencial para balancear carga de tráfego e proteger servidores internos, complementando soluções como redes privadas virtuais (VPN do inglês, Virtual Private Network), nas quais recursos computacionais em sua zona militarizada sejam acessíveis somente por dispositivos e usuários credenciados. Para garantir a portabilidade e consistência das aplicações, os contêineres surgem como uma solução de virtualização leve, encapsulando código e dependências em ambientes isolados e replicáveis. A observabilidade do sistema é reforçada por ferramentas de monitoramento e logging, que coletam métricas em tempo real e registros de execução, permitindo a identificação proativa de falhas. Esses dados são frequentemente consolidados em dashboards, interfaces visuais que sintetizam indicadores-chave de desempenho (KPIs) para tomada de decisão ágil. Paralelamente, a automação de testes reduz a dependência de intervenção manual, ampliando a cobertura e a frequência de verificações de segurança e funcionalidade. **A Engenharia de Confiabilidade de Sítios (SRE do inglês Site**

Reliability Engineering) integra princípios de engenharia de software às operações, priorizando a confiabilidade e a resiliência dos sistemas. Ao equilibrar inovação e estabilidade, o SRE opera em sinergia com o DevSecOps, utilizando automação e métricas rigorosas para garantir que a infraestrutura suporte tanto a agilidade do desenvolvimento quanto a robustez exigida pela segurança contínua.

2.2 DevOps e DevSecOps

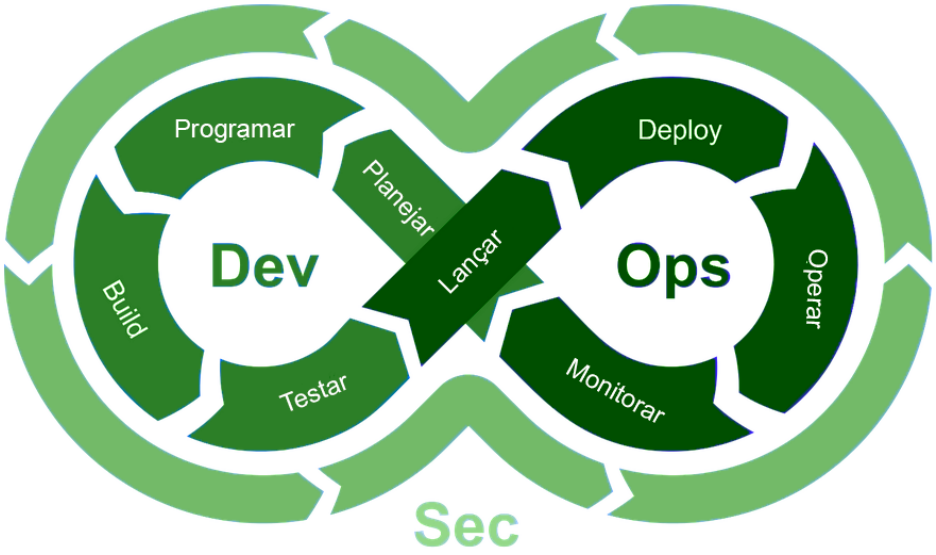
O DevOps é uma cultura e um conjunto de práticas de engenharia de *software* que une duas áreas fundamentais da tecnologia: o Desenvolvimento (*Dev*), responsável por escrever o código e criar os aplicativos, e as Operações (*Ops*), responsáveis por manter esses aplicativos no ar e funcionando perfeitamente (LEITE *et al.*, 2019).

Na prática, o DevOps funciona como uma linha de montagem moderna e altamente sincronizada. Os profissionais dessas duas áreas trabalham juntos como uma única equipe colaborativa durante todo o ciclo de vida do *software*, desde o planejamento da ideia até a entrega final para o usuário (JHA *et al.*, 2023). O objetivo principal é garantir que a entrega do produto seja rápida, contínua e de alta qualidade.

Para que essa cultura funcione, ela se apoia fortemente na automação (JHA *et al.*, 2023). São criadas "linhas de montagem" que testam o código dos programadores e o publicam na internet sem a necessidade de intervenção manual repetitiva. Isso elimina gargalos, reduz falhas humanas e permite que o sistema seja atualizado de forma constante e segura (JHA *et al.*, 2023).

Essa união de colaboração e automação prepara o terreno para um nível de maturidade ainda maior: o **DevSecOps** (RAJAPAKSE *et al.*, 2022). Nele, a Segurança (*Sec*) é inserida diretamente nessa cultura, garantindo que a proteção do sistema seja tratada como uma responsabilidade de todos e automatizada em todas as etapas de construção do *software* (RAJAPAKSE *et al.*, 2022). Este ciclo contínuo, com a segurança integrada em cada fase do processo, encontra-se ilustrado na Figura 1.

Figura 1 – O Loop Infinito do DevSecOps



Fonte: Elaborado pelo autor.

2.2.1 DevSecOps vs. SRE

SRE assim como DevSecOps é uma abordagem para solucionar os problemas causados pela separação de desenvolvimento e operações e alcançar entregas mais rápidas, mas diferente do DevOps, o SRE depende de engenheiros de confiabilidade dentro do time de desenvolvimento que também tem experiência em operações para resolver problemas de comunicação (BEYER *et al.*, 2016). Um dos pilares do SRE é a definição clara de metas de serviço (SLI, SLO, SLA), que ajudam a medir e garantir a confiabilidade. Esses conceitos são frequentemente confundidos, mas possuem funções distintas.

SLI (Service Level Indicator) reflete o desempenho de um serviço, SLO (Service Level Objective) define o nível desejado de confiabilidade e SLA (Service Level Agreement) é um acordo formal com clientes ou acionistas sobre consequências se os SLOs não forem atingidos. DevOps e SRE são complementares, o primeiro visa entregar software de forma rápida e colaborativa e o segundo mantê-lo estável e confiável em produção.

2.3 Trabalhos Correlatos

A seleção dos trabalhos correlatos apresentados nesta seção baseou-se em um levantamento bibliográfico realizado em bases acadêmicas (como *Google Scholar*, *IEEE Xplore* e *ACM Digital Library*). Foram utilizados termos de busca como *"DevSecOps"*, *"CI/CD Security"*, *"Infrastructure as Code"* e *"SRE in Cloud Migration"*. Como critérios de inclusão, priorizaram-se revisões sistemáticas da literatura e estudos empíricos recentes

que abordassem os desafios técnicos e organizacionais da integração entre desenvolvimento, operações e segurança, servindo de fundamentação e alinhamento metodológico para a intervenção realizada no Terralab.

2.3.1 Integração e Comunicação entre Dev, Sec e Ops

Analisando a dimensão colaborativa do modelo, Rajapakse *et al.* (2022) destacaram a importância da integração contínua entre as equipes de Dev, Sec e Ops, abordando o alinhamento de papéis, canais de comunicação e a automação de ferramentas. O estudo demonstra que práticas como *Shift-left*, integração via *ChatOps* e a formação de equipes híbridas são decisivas para reduzir atritos e agilizar a resposta a incidentes.

2.3.2 Automação e Integração Contínua de Segurança em Pipelines CI/CD

Em uma revisão sistemática seguida de análise empírica, Kathiresan (2022) concluiu que a integração contínua de segurança ao longo do ciclo de vida de desenvolvimento é essencial para melhorar a segurança, qualidade e agilidade. Destacou automação, monitoramento contínuo e pipelines CI/CD como fatores decisivos.

2.3.3 Transformação Cultural e Valores Fundamentais do DevOps

Em sua pesquisa, Jha *et al.* (2023) realizaram uma revisão sistemática sobre a cultura DevOps, enfatizando que o modelo não se resume à adoção de ferramentas ou automação, mas constitui predominantemente uma transformação cultural. Os autores destacam cinco valores fundamentais (colaboração, automação, mensuração, compartilhamento de informação e uso de serviços web) e propõem um *framework* conceitual para auxiliar a implementação prática desses valores nas organizações. Além disso, identificam que um dos maiores desafios reside na mudança da mentalidade das equipes, exigindo transparência e comunicação contínua.

2.3.4 Desafios Técnicos e Culturais na Adoção do Modelo DevOps

Um dos trabalhos mais abrangentes sobre DevOps que oferece uma análise sistemática da literatura e da prática em múltiplos contextos organizacionais. O estudo sintetiza os principais conceitos, benefícios esperados, desafios enfrentados e práticas adotadas por empresas que buscam implementar o modelo. Dentre os achados, os autores ressaltam que DevOps é, ao mesmo tempo, uma mudança técnica e cultural, e que sua implementação exige alterações nos processos, ferramentas e, principalmente, no comportamento das equipes (LEITE *et al.*, 2019).

2.3.5 Desafios Práticos na Implementação de Infraestrutura como Código (IaC)

Aprofundando-se na adoção prática da automação, Santos *et al.* (2018) apresentam um estudo empírico sobre os desafios de transpor a teoria do DevOps para a prática através da Infraestrutura como Código (IaC). Os autores destacam que, embora a IaC traga benefícios inquestionáveis para a reprodutibilidade e escalabilidade dos ambientes, a curva de aprendizado e a necessidade de padronização arquitetural impõem barreiras significativas. Este estudo reforça a importância de abordagens estruturadas, como a utilização de contêineres e repositórios centralizados para o código de infraestrutura, metodologias estas que foram diretamente aplicadas na reestruturação do ambiente em nuvem do Terralab.

2.3.6 Integração de Princípios SRE para Resiliência Preditiva

No contexto da confiabilidade de sistemas, Vanama (2023) exploram a integração dos princípios de *Site Reliability Engineering* (SRE) em arquiteturas corporativas visando alcançar resiliência preditiva. O trabalho demonstra que a implementação de ferramentas de observabilidade e a definição rigorosa de métricas são essenciais para antecipar falhas antes que estas impactem o usuário final. As conclusões deste artigo fundamentam a decisão arquitetural adotada no presente estudo de caso, onde a criação de uma plataforma centralizada de *dashboards* e o monitoramento proativo dos indicadores de serviço se mostraram cruciais para a evolução da maturidade operacional e segurança do laboratório.

3 Metodologia e Desenvolvimento

Esta pesquisa tem natureza aplicada, realizada por meio de um estudo de caso no laboratório de desenvolvimento de software Terralab. A abordagem adotada visou solucionar problemas práticos relacionados à disparidade entre times de desenvolvimento e operações, buscando integrar a segurança como um elemento transversal ao processo. Para alcançar os objetivos propostos, o estudo foi conduzido seguindo etapas estruturadas de diagnóstico, implementação e validação, fundamentadas nos princípios de DevSecOps e SRE.

3.1 Caracterização do Objeto de Estudo: Terralab

O Terralab é o laboratório para capacitação e pesquisa em desenvolvimento de *software*, lotado no Departamento de Computação da Universidade Federal de Ouro Preto (UFOP). O laboratório desenvolve projetos científicos e tecnológicos majoritariamente na interface entre Geoinformática (INTERNATIONAL..., 2005) e modelagem ambiental (ENVIRONMENTAL..., 1997), atendendo a demandas de pesquisa de diversas instituições parceiras, como o Instituto Nacional de Pesquisas Espaciais (INPE), a Fundação Oswaldo Cruz (Fiocruz) e a Empresa Brasileira de Pesquisa Agropecuária (Embrapa).

Entre os produtos tecnológicos mais relevantes decorrentes dessas parcerias destacam-se o simulador de sistemas terrestres TerraME (`www.terrame.org`), o arcabouço de modelagem de mudanças de uso e cobertura do solo LUCCME (`luccme.ccst.inpe.br`) e o modelo computacional de emissões de gases de efeito estufa INPE-EM (`inpe-em.ccst.inpe.br`). Atualmente, a equipe trabalha no desenvolvimento do TerraPlanner (`www.terraplanner.org`), um arcabouço de serviços em nuvem voltados para Inteligência Espacial.

O principal desafio operacional do laboratório é garantir que esses serviços permaneçam disponíveis 24 horas por dia, 7 dias por semana, com elevado desempenho e segurança, mesmo considerando que as alterações nos códigos-fonte são frequentemente realizadas por recursos humanos em capacitação (estudantes de graduação e pós-graduação).

3.2 Arquitetura e Ferramentas

A infraestrutura tecnológica proposta baseou-se nos conceitos de Computação em Nuvem (Cloud Computing), garantindo acesso a recursos computacionais escaláveis e flexíveis, os quais foram provisionados e gerenciados integralmente através de Infraestrutura como Código (IaC). Para a operacionalização do modelo DevSecOps, o fluxo de trabalho

iniciou-se em um sistema centralizado de versionamento e repositório de código, integrado a um pipeline de CI/CD responsável por automatizar a coordenação, a verificação de qualidade e a implantação contínua das alterações. As aplicações e suas dependências foram encapsuladas utilizando soluções de virtualização leve e contêineres, assegurando a portabilidade e a consistência entre os ambientes de desenvolvimento, homologação e produção. Na camada de rede, o gerenciamento de tráfego e a proteção dos servidores internos contra acessos diretos são intermediados por proxies reversos e balanceadores de carga, enquanto o acesso seguro e criptografado da equipe aos recursos internos foi garantido por uma Rede Privada Virtual (VPN). Por fim, todo esse ecossistema foi acompanhado por uma plataforma de observabilidade que unificou ferramentas de monitoramento e logging, permitindo a coleta de métricas em tempo real e a consolidação dos dados em dashboards para embasar a tomada de decisão.

3.3 Diagnóstico do Processo Atual

Nesta etapa, foi realizado um levantamento detalhado do fluxo de trabalho atual das equipes do Terralab. O objetivo foi identificar os gargalos de comunicação e os pontos de atrito entre desenvolvimento e operações que causam os problemas de interrupções e falta de padronização citados na introdução. Foram coletados dados quantitativos e qualitativos sobre o ciclo de desenvolvimento atual, estabelecendo uma linha de base (baseline) para comparações futuras. Esse diagnóstico permitiu mapear onde a aplicação dos pilares de DevSecOps, como a responsabilidade do início ao fim, era mais crítica.

3.4 Projeto de Adequação

Com base no levantamento e no diagnóstico do processo até então vigente, foi elaborado um projeto de adequação estratégico norteado por uma Análise de Lacunas (*Gap Analysis*). Este processo teve como objetivo central confrontar o cenário legado do laboratório com o estado ideal almejado, o qual foi estritamente fundamentado nos pilares metodológicos do DevSecOps e nas práticas de Engenharia de Confiabilidade de Sítios (SRE).

A análise identificou deficiências estruturais críticas nos eixos tecnológico, cultural e de segurança corporativa, servindo como a principal base de justificativa para a priorização das etapas de reestruturação arquitetural.

3.4.1 Relatório de *Gap*

O mapeamento das discrepâncias entre o estado operacional anterior e a nova arquitetura resultou na elaboração do Relatório de *Gap*. As principais lacunas identificadas

e as respectivas propostas de adequação estrutural e ferramental estão consolidadas na Tabela 1.

Tabela 1 – Relatório de *Gap*: Comparativo entre o Cenário Legado e o Estado Desejado

Dimensão	**Cenário Legado (As-Is)**	**Estado Desejado (To-Be)**	**Lacuna (*Gap*) e Adequação**
Infraestrutura e Redes	Configuração manual e isolada de servidores, acarretando em falta de padronização.	Infraestrutura imutável e integralmente gerenciada via Infraestrutura como Código (IaC).	Implementar o provisionamento automatizado baseado em contêineres e *scripts* IaC.
Integração e Entregas	Múltiplas integrações isoladas com processos de *deploy* lentos, manuais ou parcialmente roteirizados.	Automação de ponta a ponta garantindo alta frequência de *deploys* com base em repositórios centralizados.	Criar esteiras de Integração e Entrega Contínuas (CI/CD) acionadas via controle de versão.
Segurança e Acessos	Acessos administrativos com portas expostas; testes de segurança feitos tardiamente.	Proteção de borda rigorosa e auditoria proativa de segurança desde a concepção (*Shift-Left* via SAST).	Exigir túnel VPN, ocultar portas via *Bastion Host* e embutir varreduras no *pipeline*.
Observabilidade	Resolução puramente reativa a incidentes operacionais; ausência de indicadores de integridade em tempo real.	Monitoramento proativo orientado a dados, controlando a redução do MTTR e o acompanhamento de SLIs e SLOs.	Adotar uma plataforma de observabilidade contínua suportada por *dashboards* centralizados.

Fonte: Elaborado pelo autor.

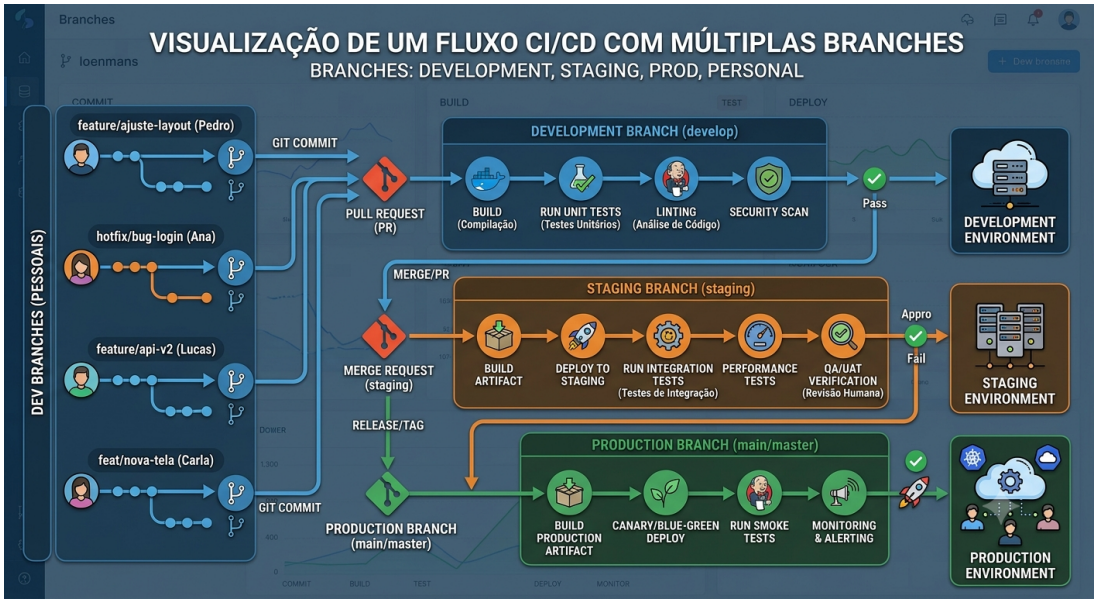
A partir do reconhecimento estruturado destas lacunas, a execução do projeto de adequação consolidou a adoção de uma abordagem *greenfield*. Esta decisão garantiu que o ambiente do laboratório não herdassem as dívidas técnicas levantadas, permitindo que a nova nuvem fosse arquitetada ”do zero”, sanando nativamente todos os apontamentos do relatório de *gap*.

3.5 Implementação do Pipeline CI/CD

Com base na arquitetura definida na seção 3.2, a construção do pipeline de CI/CD acontece antes da configuração do ambiente de nuvem . O fluxo é estruturado a partir de repositórios Git, que atuaram como a fonte única de verdade para o código-fonte e para as configurações da infraestrutura. Qualquer alteração submetida a esses repositórios aciona automaticamente o pipeline, seguindo uma abordagem incremental onde a automação é priorizada, conforme o sexto pilar do DevOps. Durante a etapa de Integração Contínua, o código é compilado e empacotado na forma de imagens de contêineres, que consistem em artefatos de software autossuficientes contendo as aplicações e todas as suas dependências

embutidas. Ferramentas de testes automatizados são integradas diretamente nesse fluxo, analisando não apenas o código, mas também realizando varreduras nas próprias imagens geradas, garantindo que a segurança seja tratada de forma transversal e não apenas no final do processo. Além disso, a infraestrutura é gerenciada como código (IaC) para assegurar a reprodutibilidade exata dos ambientes baseados nessas imagens validadas. Esse fluxo é ilustrado na Figura 2.

Figura 2 – Diagrama de fluxo automatizado do pipeline de CI/CD



Fonte: Imagem gerada por Inteligência Artificial (Gemini, Google), 2026.

3.6 Definição de Métricas e Observabilidade

Para a validação do modelo proposto, foram definidos indicadores de desempenho baseados nas práticas de *Site Reliability Engineering* (SRE), visando mensurar objetivamente a confiabilidade, o desempenho e a segurança do novo ambiente.

A coleta de dados foi estruturada através de agentes exportadores instalados nos servidores, responsáveis por extrair métricas de baixo nível do sistema operacional. Esses dados brutos foram continuamente raspados (*scraped*) e armazenados em um banco de dados de séries temporais, formando a base de cálculo para os Indicadores de Nível de Serviço (SLIs). Dentre as métricas monitoradas, destacaram-se:

- **Disponibilidade (*Uptime*):** Estabelecido como o principal SLI operacional, refletindo a proporção de tempo em que os serviços estiveram responsivos e acessíveis para os usuários. O Objetivo de Nível de Serviço (SLO) associado visava manter a alta disponibilidade contínua exigida pelos projetos do Terralab.

- **Saturação e Tráfego de Rede:** O monitoramento do tráfego de dados de entrada e saída permitiu definir SLIs de utilização de banda, garantindo que o SLO de capacidade da rede não fosse estrangulado durante picos de acesso.
- **Métricas de Segurança (Auditoria de *Logs*):** No contexto do DevSecOps, a quantificação de *logs* de tentativas de acesso não autorizado atuou como um SLI de segurança vital. O SLO estabelecido foi garantir uma redução drástica na taxa de ataques de força bruta, validando o isolamento da rede promovido pelo *Bastion Host*.

Essa instrumentação automatizada possibilitou o acompanhamento do estado de saúde da infraestrutura em tempo real, permitindo verificar empiricamente a melhoria na resiliência do sistema e validando a eficácia da nova arquitetura.

3.7 Estudo de Caso: Reestruturação e Migração de Nuvem

A validação prática do modelo foi realizada no contexto da migração de provedor de nuvem do Terralab. Aproveitando que o time de DevOps, embora já constituído, encontrava-se em fase inicial de estruturação, a mudança de provedor foi utilizada como uma oportunidade estratégica para reconstruir a infraestrutura "do zero". Esta estratégia caracterizou-se como uma abordagem greenfield, que, no contexto da engenharia de software, refere-se ao desenvolvimento e implantação em um ambiente totalmente novo, livre de restrições, dependências ou dívidas técnicas impostas por sistemas legados (SOMMERVILLE, 2011). Mais do que uma simples atualização tecnológica, esse cenário de reconstrução atuou como o principal vetor para a consolidação da cultura DevSecOps no laboratório. Ao realizar todo o provisionamento do novo ambiente estritamente via Infraestrutura como Código (IaC), a equipe internalizou o conceito de responsabilidade compartilhada e colaboração contínua, rompendo os tradicionais silos isolados entre desenvolvimento, operações e segurança. Essa mudança de paradigma cultural garantiu que os pilares de automação e a mentalidade de segurança proativa (shift-left) fossem enraizados na própria fundação da nova arquitetura, evitando a replicação de vícios processuais de gestões anteriores.

3.8 Observabilidade e Monitoramento de Servidores

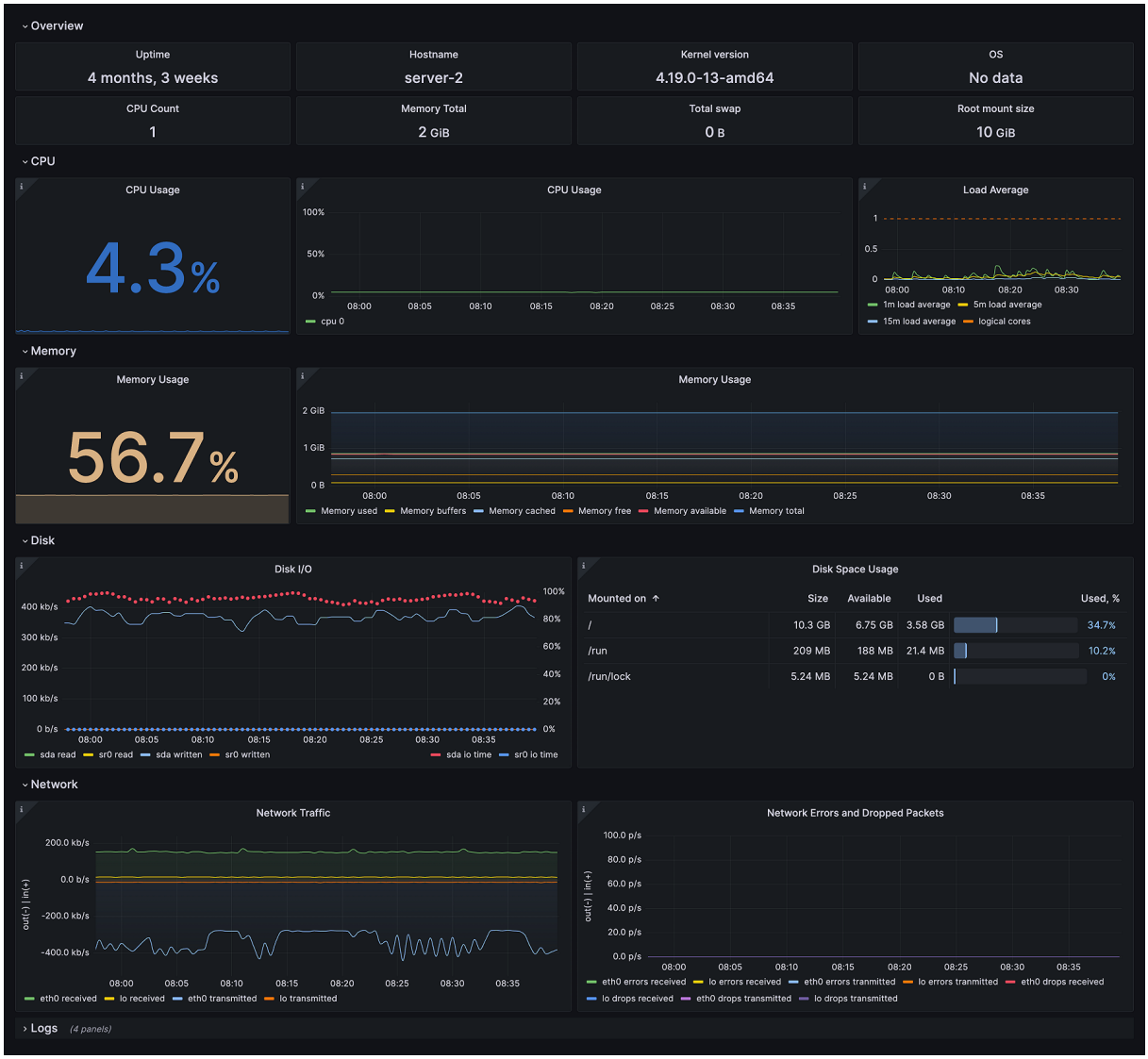
O primeiro passo prático do estudo de caso consistiu na estruturação de uma plataforma centralizada de observabilidade. A implementação prévia desta ferramenta foi fundamental para estabelecer uma linha de base (*baseline*) do ambiente legado, permitindo a coleta de métricas e a comparação direta de desempenho, disponibilidade e segurança antes e depois das intervenções propostas.

Visando manter o controle estrito sobre os dados operacionais e evitar a exposição de informações sensíveis da arquitetura a serviços de terceiros (*SaaS*), optou-se por hospedar a ferramenta de criação de *dashboards* interativos internamente. Esta solução visual foi implantada diretamente em um dos servidores da própria nuvem do laboratório.

O *dashboard* de monitoramento foi configurado para consumir e unificar dados de séries temporais provenientes de todos os contêineres e máquinas da infraestrutura. Através de painéis visuais customizados, a equipe passou a ter acesso imediato a indicadores vitais de saúde dos servidores, como o consumo de processamento (CPU), uso de memória RAM, saturação de disco e tráfego de rede. A Figura 3 é uma demonstração de uma ferramenta popular para essa aplicação.

Além das métricas de infraestrutura de baixo nível, o painel consolidou os Indicadores de Nível de Serviço (SLIs) das aplicações. Ter essa visibilidade inicial permitiu que as operações do Terralab fizessem a transição de um modelo de resposta reativa para um gerenciamento de confiabilidade orientado a dados, comprovando empiricamente a eficácia das mudanças de infraestrutura e segurança que se seguiram.

Figura 3 – Exemplo de painel de visualização (*dashboard*) de uma plataforma de observabilidade



Fonte: Grafana Labs

3.9 Infraestrutura e Automação

Com a plataforma de observabilidade e a linha de base já estabelecidas, o provisionamento do novo ambiente em nuvem foi integralmente gerenciado utilizando o conceito de Infraestrutura como Código (IaC). É fundamental distinguir, neste contexto, a camada de infraestrutura em nuvem da camada de aplicação.

Os recursos computacionais brutos da nuvem, como a criação de redes virtuais (VPCs), regras de *firewall*, alocação de instâncias e discos gerenciados, foram provisionados por meio de ferramentas declarativas de IaC. O estado dessa infraestrutura passou a ser

armazenado de forma centralizada, garantindo a idempotência das execuções e mitigando o desvio de configuração (*drift*). Qualquer plano de alteração estrutural passou a exigir revisão humana antes da aplicação, garantindo total auditoria.

Na camada de aplicação, adotou-se a conteinerização (utilizando tecnologias padrão de mercado) para garantir a padronização e o isolamento dos serviços. Embora a ferramenta de composição de contêineres adotada atue primariamente na coordenação de manifestos em um único nó — não oferecendo recursos avançados de orquestradores de *cluster*, como autorrecuperação em múltiplos nós (*self-healing*) ou reescalonamento dinâmico —, ela foi suficiente para o escopo e a escala atual do laboratório. Atuando como um mecanismo de "automação de implantação"e orquestração leve, ela controlou declarativamente as réplicas, volumes e redes internas, integrando-se perfeitamente ao processo automatizado.

Um dos pilares estruturais dessa automação foi a adoção abrangente da conteinerização para todos os serviços do laboratório. Utilizando tecnologias padrão de mercado, como por exemplo, Docker e Docker Compose, a equipe garantiu que cada aplicação fosse empacotada com todas as suas dependências e variáveis de ambiente em manifestos declarativos. Essa padronização mitigou a clássica disparidade entre os ambientes de desenvolvimento e produção.

Paralelamente, foi configurada uma plataforma de automação de Integração e Entrega Contínuas (CI/CD) que atua em total sinergia com esses contêineres. Este *pipeline* tornou-se o elemento central da cultura DevSecOps proposta. Ele foi programado para interpretar os arquivos do Docker, orquestrando automaticamente a execução, o *build* e o *deploy* dos contêineres diretamente nos servidores sempre que uma nova alteração é aprovada.

Além da implantação automatizada, o *pipeline* também orquestra a execução de testes estáticos de segurança (SAST) no código e nas próprias imagens dos contêineres. Essa integração forçou a validação da conformidade antes de qualquer liberação para o ambiente de produção, alinhando a equipe com a cultura de *Shift-Left Security*.

3.10 Arquitetura de Segurança: Bastion Host, VPN e Proxy Reverso

Para mitigar a exposição dos recursos críticos à internet pública, a topologia de rede foi fundamentalmente redesenhada, segmentando os componentes em zonas de segurança com funções estritamente distintas. O acesso direto de administração aos servidores internos foi completamente bloqueado.

O acesso administrativo à infraestrutura passou a ser intermediado unicamente por um *Bastion Host* (servidor de salto). Dessa forma, a equipe de desenvolvedores e operadores do Terralab passou a ter a obrigatoriedade de estabelecer uma conexão através

de uma Rede Privada Virtual (VPN), que funciona como a única porta de entrada segura para a rede administrativa. Somente após essa autenticação e inseridos na rede virtual, o acesso lógico ao *Bastion Host* era liberado.

De forma paralela, e em uma zona de segurança completamente isolada do tráfego administrativo, o fluxo de dados legítimo proveniente dos usuários finais (*web*) passou a ser recepcionado por um proxy reverso (ou *gateway*). Atuando como a porta de entrada pública exclusiva para as requisições externas (nos protocolos HTTP e HTTPS), o proxy reverso intercepta o tráfego das aplicações, assume a responsabilidade pela terminação dos certificados de criptografia (TLS/SSL) e roteia as chamadas internamente para os respectivos contêineres.

Essa segregação arquitetural, representada de forma simplificada e lógica na imagem 5 (agrupando funções de borda por questões de abstração visual), ocultou a verdadeira topologia física da rede e os endereços IP dos serviços internos. Na arquitetura efetivamente implantada, a separação destas zonas de segurança garante que as aplicações finais nunca fiquem expostas diretamente à internet pública e que a superfície de ataque seja minimizada, em conformidade com as diretrizes de confidencialidade do laboratório.

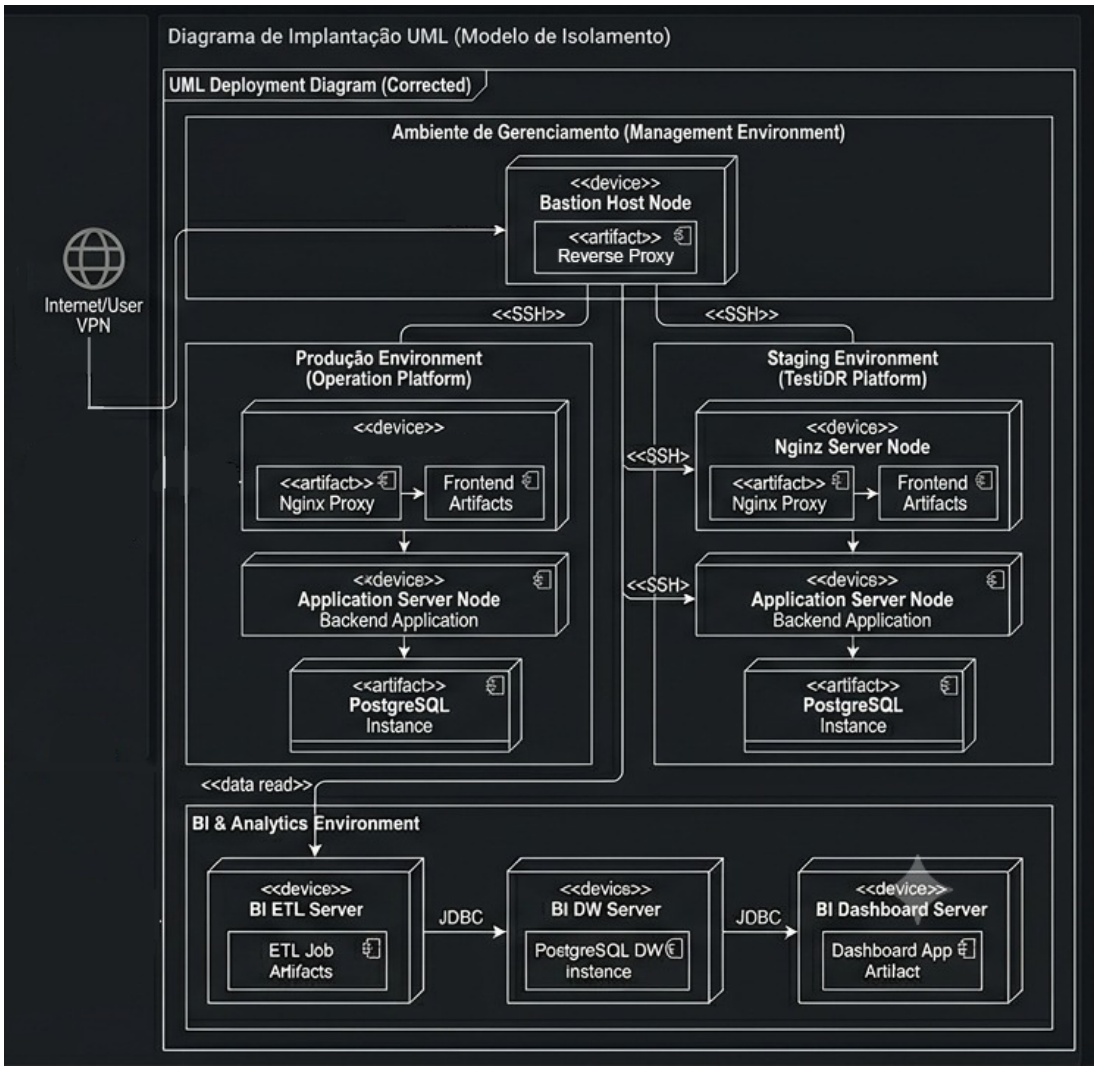
A fim de ilustrar a sinergia entre a conteinerização e a estratégia de rede adotada, o fragmento de código e a imagem 5 a seguir apresenta um exemplo estrutural simplificado utilizando o *Docker Compose* (`docker-compose.yaml`). Neste cenário declarativo, o contêiner atuando como proxy reverso (NGINX) é o único serviço que mapeia portas para o meio externo, enquanto a aplicação web permanece blindada, acessível exclusivamente através da rede interna do *Docker*:

Figura 4 – (`docker-compose.yml`)

```
version: '3.8'

services:
proxy-reverso:
image: nginx:latest
ports:
- "80:80"
- "443:443"
volumes:
- ./nginx.conf:/etc/nginx/nginx.conf:ro
depends_on:
- aplicacao-web
networks:
- rede-interna

aplicacao-web:
image: terralab/app-planejamento:latest
expose:
- "8080" # Porta exposta estritamente para a rede interna do
    Docker
networks:
- rede-interna

networks:
rede-interna:
driver: bridge
```

Fonte: Elaborado pelo autor.

Figura 5 – Diagrama simplificado da arquitetura de segurança da rede isolada via *Bastion Host* e VPN



Fonte: Adaptada de uma imagem gerada por IA (Google Gemini, 2026)

4 Resultados e Discussões

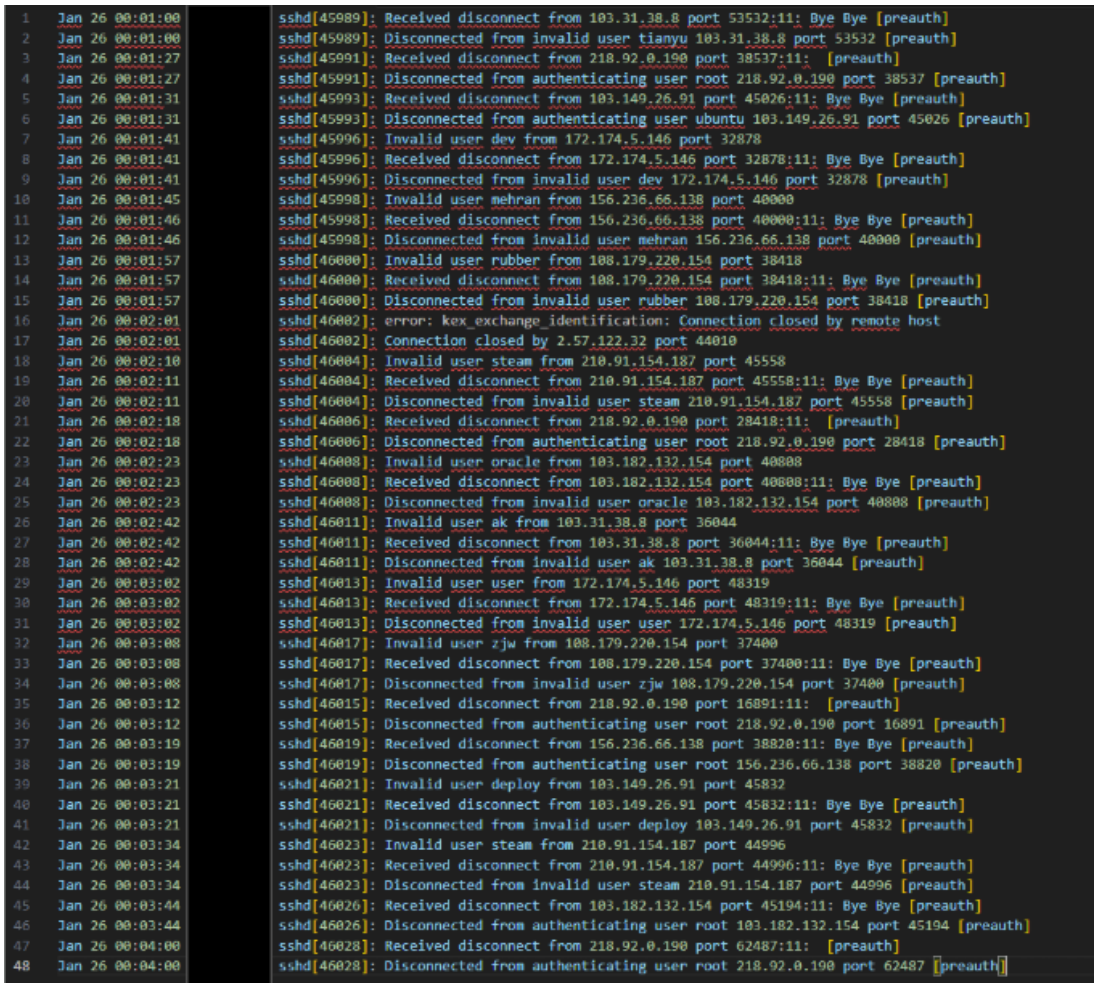
Após a implementação da nova arquitetura e a consolidação das práticas de IaC e CI/CD, os dados gerados continuamente pela plataforma de observabilidade possibilitaram analisar objetivamente o impacto da reestruturação. As métricas coletadas comprovaram o atingimento dos objetivos em dois eixos fundamentais: Segurança e Confiabilidade.

4.1 Segurança e Proteção da Infraestrutura

A mudança paradigmática na forma como o acesso administrativo era concedido resultou na maior conquista de segurança deste trabalho. Ao expurgar portas administrativas públicas da nuvem e forçar a utilização do túnel VPN em conjunto com o *Bastion Host*, a superfície de exposição foi controlada com precisão.

Para ilustrar a gravidade do cenário legado, a Figura 6 apresenta um extrato real do monitoramento de *logs* de um dos servidores antes da reestruturação. É possível observar um constante bombardeio de ataques automatizados de força bruta, caracterizado por requisições de IPs desconhecidos tentando estabelecer conexões SSH com credenciais genéricas e usuários inválidos (como `root`, `ubuntu` e `dev`).

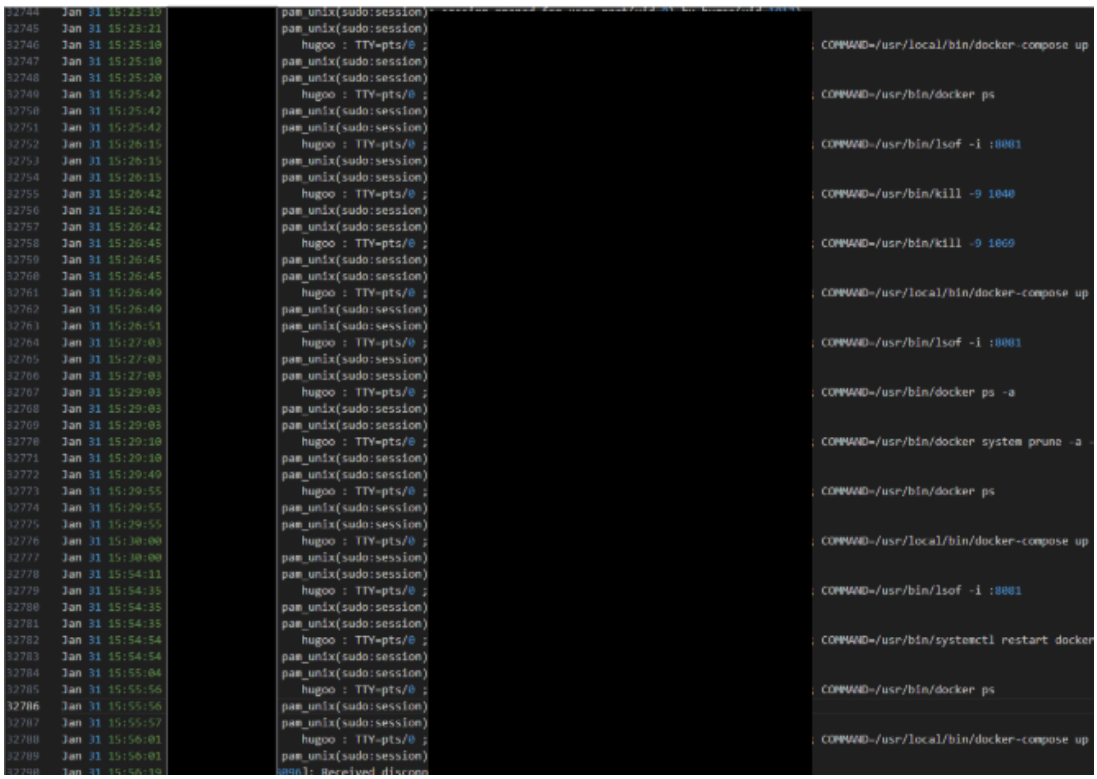
Figura 6 – Extrato de *logs* evidenciando sucessivas tentativas de acesso não autorizado via protocolo SSH



Fonte: Elaborado pelo autor.

Em contrapartida, a eficácia da nova arquitetura de proteção pode ser visualmente atestada na Figura 7. O extrato ilustra o monitoramento do mesmo servidor após a adoção do *Bastion Host* e a restrição de acesso via VPN.

Figura 7 – Extrato de *logs* após a reestruturação, evidenciando acessos legítimos, rastreáveis e focados na orquestração de contêineres



Fonte: Elaborado pelo autor.

Diferente do cenário de bombardeio evidenciado anteriormente, a nova configuração demonstra um ambiente controlado e rastreável. O tráfego de auditoria passou a registrar exclusivamente sessões administrativas legítimas, nas quais usuários internos, devidamente autenticados e com privilégios escalonados controlados (*sudo*), executam comandos pertinentes à manutenção da infraestrutura, como o gerenciamento de processos e a orquestração via `docker` e `docker-compose`.

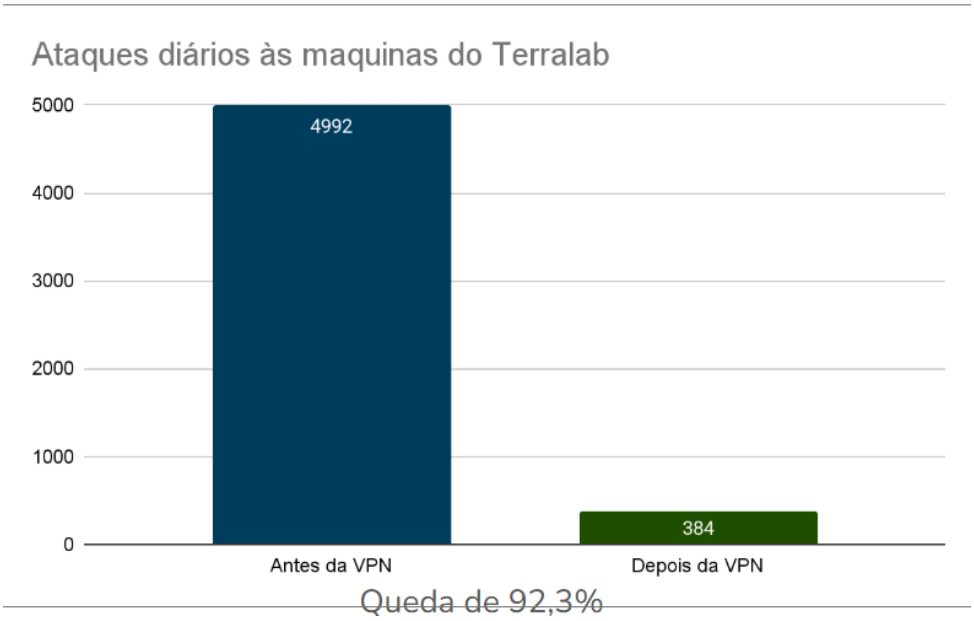
A justaposição desses dois cenários comprova empiricamente a estabilização do ambiente. A supressão quase total do ”ruído”gerado por agentes maliciosos externos atesta que o encapsulamento da rede atendeu perfeitamente ao objetivo de reduzir a superfície de ataque, consolidando a segurança como alicerce estrutural das operações do laboratório.

Uma pesquisa comparativa focada nos *logs* de auditoria e monitoramento de rede, englobando o período anterior e posterior à intervenção, demonstrou resultados empíricos substanciais. Observou-se uma redução de 92,3

4.1.1 Análise da Superfície de Ataque Residual

Para quantificar o impacto visualizado nos *logs* de auditoria, os dados de conexões bloqueadas foram consolidados, permitindo uma comparação do volume de ameaças antes e depois da reestruturação arquitetural. O resultado dessa medição está ilustrado na Figura 8.

Figura 8 – Comparativo de tentativas de ataques diarios de força bruta antes e após a adoção da VPN



Fonte: Elaborado pelo autor.

Conforme demonstrado, a transição para a nova infraestrutura resultou em uma queda expressiva de 92,3% nos eventos de varredura e nas tentativas de conexões não autorizadas diárias nas instâncias internas do Terralab.

No entanto, do ponto de vista da segurança cibernética, a redução no registro destas ocorrências não significa, necessariamente, que a quantidade de atacantes diminuiu. Mas sim que o tráfego nocivo deixou de alcançar e onerar os servidores de produção. Os eventos de varredura continuam atingindo a borda externa da rede (como o *gateway* da VPN e os *firewalls*), evidenciando a existência de uma superfície de ataque residual.

Essa margem residual de 7,7% ocorre porque a arquitetura de *Bastion Host* e VPN exige que ao menos uma porta lógica permaneça acessível à internet pública para processar as requisições de autenticação legítimas da equipe, mantendo-se, assim, como alvo de *scanners* automatizados. O ganho fundamental do modelo DevSecOps implementado reside na contenção perimetral, atestando o conceito de Defesa em Profundidade (*Defense in Depth*).

O ganho fundamental do modelo DevSecOps implementado, contudo, reside no conceito de Defesa em Profundidade (*Defense in Depth*). O atrito malicioso deixou de ocorrer diretamente nos servidores de banco de dados e aplicações lógicas, que agora operam em redes isoladas, e passou a ser contido exclusivamente na borda da rede. Esse servidor funciona como um perímetro fortificado (*hardened*), preparado estruturalmente para descartar acessos inválidos.

O reconhecimento dessa vulnerabilidade inerente na porta de entrada evidencia que a segurança é um processo contínuo (pilar 5 do DevOps). Esse cenário abre precedentes para trabalhos e melhorias futuras na infraestrutura do laboratório, como a adoção de Autenticação Multifator (MFA), implementação de bloqueios dinâmicos de IP (como o *Fail2Ban*) e evolução para o modelo de *Zero Trust Network Access* (ZTNA).

Uma evolução arquitetural proposta para trabalhos futuros envolve o uso de técnicas de *Cyber Deception* (Engano Cibernético). Essa abordagem consistiria na expansão do modelo atual para uma estrutura contendo dois *Bastion Hosts* dispostos em série. O primeiro servidor da cadeia, posicionado na linha de frente do tráfego, atuaria exclusivamente como um *Honeypot* (pote de mel) de alta interação. Esse ambiente emularia serviços e portas legítimas da nuvem, mas seria um *sandbox* isolado contendo apenas informações falsas, credenciais inválidas e dados inúteis para a organização.

O verdadeiro *Bastion Host*, responsável por conceder o acesso à rede interna e aos contêineres de produção, ficaria posicionado em uma segunda camada, oculto e acessível apenas por caminhos mapeados que o atacante desconhece.

Dessa forma, caso um invasor consiga transpor a defesa inicial da nuvem, ele comprometerá a máquina isca. Ao invés de paralisar as operações reais, o atacante perderá tempo explorando dados fabricados enquanto a plataforma de observabilidade do laboratório registra silenciosamente suas Táticas, Técnicas e Procedimentos (TTPs).

Estudos aplicados à arquitetura em nuvem, como a pesquisa de Moura (2014), confirmam que o uso integrado de *Honeypots* em ambientes virtualizados aumenta significativamente a proteção autonômica dos servidores de produção. A isca não apenas absorve e isola o impacto da intrusão, como fornece inteligência acionável à equipe de operações para bloquear a ameaça de forma preditiva.

4.2 Confiabilidade e Métricas de SRE

A avaliação da confiabilidade do ambiente baseou-se nos preceitos de Engenharia de Confiabilidade de Sítios (SRE). Contudo, a análise dos Indicadores de Nível de Serviço (SLIs) de disponibilidade (*Uptime*) no cenário legado revelou uma limitação metodológica clássica dos sistemas de monitoramento estritamente internos (*Whitebox Monitoring*).

No cenário anterior à reestruturação, os agentes exportadores e a ferramenta de coleta de métricas (como o *Prometheus*) operavam localmente nas próprias máquinas. Durante os episódios de ataque e sobrecarga de requisições de *login*, o esgotamento dos recursos de rede fazia com que o servidor bloqueasse a comunicação externa, tornando os serviços inacessíveis para os usuários finais. No entanto, do ponto de vista do agente interno, os processos do sistema operacional permaneciam ativos, registrando um *SLI* de *Uptime* continuamente elevado (com médias que não caíam abaixo de 98%), o que mascarava a real indisponibilidade do serviço.

A partir desse diagnóstico, constatou-se que as métricas tradicionais de *Uptime* interno eram insuficientes para refletir a saúde real da aplicação sem o suporte de testes de disponibilidade de ponta a ponta (*Blackbox Monitoring*). Por essa razão, a validação empírica do sucesso da transição para a cultura DevSecOps e da migração para a nova nuvem concentrou-se no acompanhamento direto da saturação de CPU, do tráfego de rede perimetral e na auditoria de *logs* de acesso.

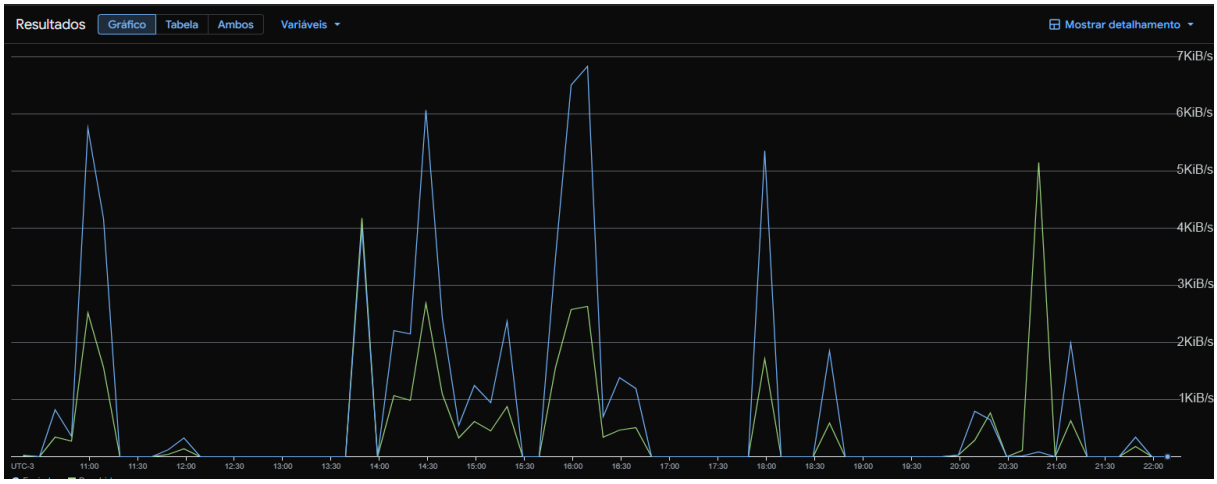
Foi definido, internamente, um Objetivo de Nível de Serviço (SLO) de disponibilidade (*Uptime*) alvo de 99,9% para os serviços críticos, passando a ser acompanhado ativamente. Contudo, na janela de avaliação prática deste estudo, os resultados quantitativos mais expressivos concentraram-se nas métricas de estabilidade das instâncias isoladas. A análise, mesmo durante períodos em que as aplicações não estavam em uso ativo, forneceu um panorama empírico incontestável sobre a redução da superfície de varredura e o ganho de estabilidade física.

4.2.1 Análise do Tráfego de Rede e Isolamento

Para atestar o isolamento da rede promovido pela implementação do *Bastion Host* e da VPN, foram comparadas as métricas de tráfego de rede (*Network Traffic*) de um servidor em estado ocioso antes e depois da intervenção.

A Figura 9 ilustra o volume de rede no cenário legado. É possível observar picos constantes e irregulares de dados recebidos (alcançando marcas em *KiloBytes* por segundo). Esse padrão é um indicativo de *scanners* automatizados e *bots* maliciosos a varrer continuamente as portas administrativas que se encontravam expostas.

Figura 9 – Tráfego de rede durante período ocioso antes da adoção da VPN (Cenário Legado)



Fonte: Elaborado pelo autor a partir do GCP *Metrics Explorer* (Fonte do eixo y aumentada).

Em contrapartida, a Figura 10 detalha o comportamento da mesma rede após a blindagem da arquitetura. O gráfico demonstra que os acessos aleatórios caíram drasticamente. Os eixos de medição reduziram a sua escala para meros *Bytes*, atestando a contenção das requisições ilegítimas na borda da nuvem, de modo que o tráfego malicioso deixou de onerar as instâncias de produção.

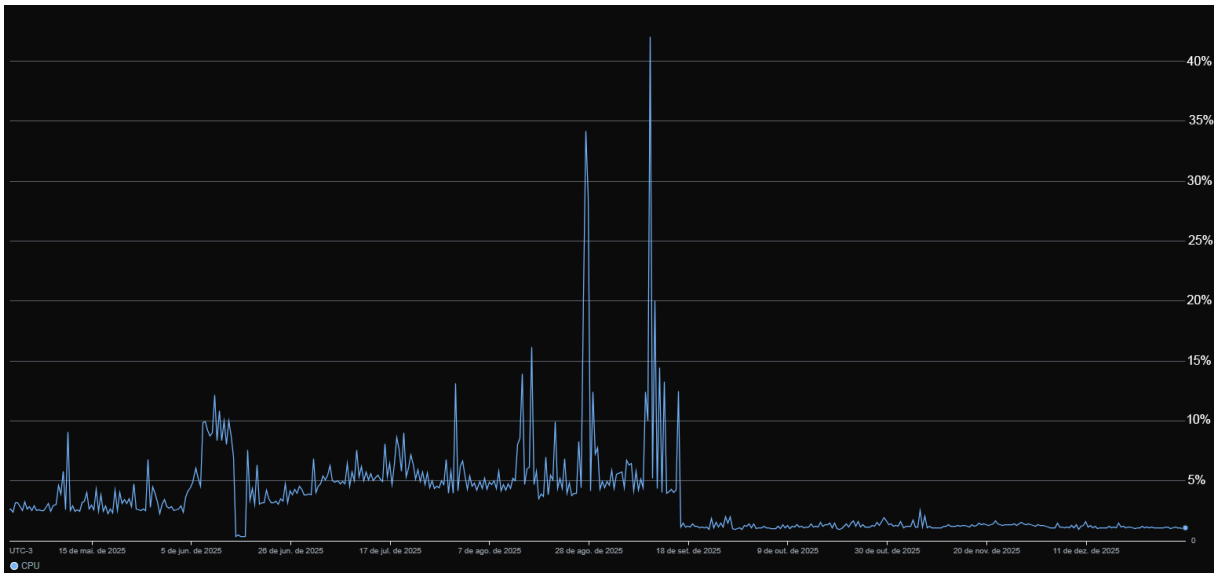
Figura 10 – Tráfego de rede estabilizado após o isolamento via *Bastion Host* e VPN

Fonte: Elaborado pelo autor a partir do GCP *Metrics Explorer* (Fonte do eixo y aumentada).

4.2.2 Estabilidade e Uso de Processamento (SRE)

Além dos ganhos em segurança, a confiabilidade inerente aos princípios de SRE impactou positivamente o consumo de recursos computacionais. A Figura 11 apresenta o histórico de utilização de CPU do servidor principal ao longo do período de transição da pesquisa.

Figura 11 – Histórico de utilização de CPU evidenciando a estabilização sistêmica pós-implementação



Fonte: Elaborado pelo autor a partir do GCP *Metrics Explorer* (Fonte do eixo y aumentada).

A linha do tempo revela claramente duas fases operacionais distintas. Na primeira metade do gráfico, sob a arquitetura legada, o processador sofria com anomalias severas e picos imprevisíveis, chegando a registar sobrecargas de até 30% em eventos não programados.

Entretanto, a partir do mês de setembro, marco cronológico da finalização da implantação da nova arquitetura, a métrica entra em absoluta estabilidade. O processamento consolidou-se numa linha constante e otimizada (operando continuamente abaixo de 5%), comprovando que a nova arquitetura eliminou o desperdício de recursos causado pela sobrecarga de ataques externos e por configurações ineficientes. Esse comportamento linear valida, na prática, a resiliência e a previsibilidade alcançadas pelas operações do laboratório.

5 Considerações Finais

Este estudo demonstrou a viabilidade e apresentou evidências práticas observáveis resultantes da adoção da cultura *DevSecOps* no laboratório Terralab. A opção por uma abordagem *greenfield* mostrou-se uma decisão arquitetural vantajosa, permitindo que a infraestrutura em *cloud* fosse desenvolvida livre das dívidas técnicas e vícios operacionais herdados do sistema legado.

A transição de processos manuais para *IaC*, com o suporte de *pipelines* de *CI/CD*, evidenciou que a automação atua como um elemento central para a integração eficiente entre as equipes de desenvolvimento e operações. A redução do isolamento entre esses setores proporcionou maior agilidade nos lançamentos e garantiu que os ambientes de homologação e produção mantivessem consistência e auditabilidade.

No âmbito de segurança (*Sec*), o isolamento da rede por meio do *Bastion Host* e a obrigatoriedade do uso de uma *Virtual Private Network* (VPN) ilustram a aplicação prática do conceito de *Shift-Left Security*. A redução de 92,3% nas tentativas de conexões não autorizadas e varreduras automatizadas atesta a eficácia dessa blindagem perimetral. Adicionalmente, a adoção de técnicas de Engenharia de Confiabilidade de Sítios (SRE), aliada à construção de um painel de observabilidade centralizado, permitiu verificar não apenas a disponibilidade (*Uptime*) do sistema, mas também a estabilização contínua do uso de recursos computacionais, reduzindo picos de sobrecarga.

Embora os resultados mostram uma expressiva evolução na maturidade operacional do laboratório, o conceito de melhoria contínua pressupõe que a infraestrutura sempre pode evoluir. A constatação empírica de uma superfície de ataque residual (7,7%) na porta de entrada da VPN abre precedentes para investigações futuras.

Como propostas para trabalhos subsequentes, sugere-se a evolução do atual modelo de defesa em profundidade para uma arquitetura baseada em *Zero Trust Network Access* (ZTNA), complementada pela adoção de Autenticação Multifator (MFA) obrigatória para todos os engenheiros. Adicionalmente, a implementação de uma arquitetura de engano cibernético (*Cyber Deception*) mediante a inserção de *Honeypots* de alta interação em paralelo ao *Bastion Host* pode fornecer inteligência preditiva à equipe do Terralab, convertendo varreduras maliciosas em dados acionáveis e isolando ainda mais o ambiente produtivo.

Referências Bibliográficas

BASS, Len; WEBER, Ingo; ZHU, Liming. *DevOps: A Software Architect's Perspective*. Upper Saddle River, NJ: Addison-Wesley Professional, 2015. Citado 1 vez na página 15.

BEYER, Betsy *et al. Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly Media, 2016. Disponível em: https://sre.google/sre-book/table-of-contents/. Acesso em: 6 fev. 2026. Citado 1 vez na página 17.

ENVIRONMENTAL MODELLING & SOFTWARE. Amsterdam: Elsevier, 1997. Mensal. ISSN 1364-8152. Citado 1 vez na página 20.

INTERNATIONAL JOURNAL OF GEOINFORMATICS. Bangkok: Association for Geoinformation Technology, 2005. Trimestral. ISSN 1686-6576. Disponível em: https://journals.sfu.ca/ijg. Acesso em: 6 fev. 2026. Citado 1 vez na página 20.

JHA, M. *et al.* From theory to practice: Understanding DevOps culture and mindset. *Cogent Engineering*, 2023. Disponível em: https://www.tandfonline.com/doi/full/10.1080/23311916.2023.2251758. Acesso em: 6 fev. 2026. Citado 9 vezes nas páginas 12–14, 16, 18.

KATHIRESAN, V. Evaluating the impact of Devsecops on software quality: A systematic review and empirical study. *World Journal of Advanced Research and Reviews*, 2022. Disponível em: https://wjarr.com/sites/default/files/WJARR-2022-0267.pdf. Acesso em: 6 fev. 2026. Citado 2 vezes nas páginas 12, 18.

LEITE, Leonardo *et al.* A Survey of DevOps Concepts and Challenges. *ACM Computing Surveys*, 2019. Disponível em: https://arxiv.org/pdf/1909.05409. Acesso em: 6 fev. 2026. Citado 3 vezes nas páginas 12, 16, 18.

MOURA, Eduardo Henrique de Carvalho. *Modelo de segurança autonômica para computação em nuvem com uso de honeypot*. 2014. Diss. (Mestrado) – Universidade Federal do Maranhão (UFMA). Disponível em: https://tedebc.ufma.br/jspui/handle/tede/516. Acesso em: 18 jun. 2026. Citado 1 vez na página 35.

RAJAPAKSE, C. *et al.* Challenges and solutions for adopting DevSecOps in software development organizations: A systematic mapping study. *Information and Software Technology*, v. 143, p. 106770, 2022. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0950584921001543. Acesso em: 6 fev. 2026. Citado 4 vezes nas páginas 12, 16, 18.

SANTOS, André L. *et al.* From Theory to Practice: The Challenges of a DevOps Infrastructure as Code Implementation, 2018. Disponível em: https://www.academia.edu/download/95657067/68261.pdf. Acesso em: 31 maio 2026. Citado 1 vez na página 19.

SOMMERVILLE, Ian. *Engenharia de software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011. Citado 1 vez na página 24.

VANAMA, Siva Kantha Rao. Integrating Site Reliability Engineering SRE Principles into Enterprise Architecture for Predictive Resilience. *International Journal of Emerging Trends in Computer Science and Information Technology*, v. 4, p. 164–170, 2023. Disponível em: https://www.ijetcsit.org/index.php/ijetcsit/article/view/514. Acesso em: 31 maio 2026. Citado 1 vez na página 19.