Terraform e CDK: Uma análise comparativa entre ferramentas de gerência de infraestrutura

João Victor Tadeu Chaves Frois[1], **Lucas Gabriel Padrão Rezende**[1]

[1]Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
30140-002 – Belo Horizonte – MG – Brasil

{jfrois,lgprezende}@sga.pucminas.br

Abstract. *The selection of infrastructure tools is crucial to the success of development teams, as is the choice of programming language for a specific project. However, differences in the efficiency of support tools for managing infrastructure as code are not clear. This lack of clarity makes it difficult to choose the best alternative for each context, making it relevant to compare them, such as Terraform and the AWS Cloud Development Kit. Thus, the purpose of this study is to identify the differences between the tools in terms of ease of use, scalability, maintainability and performance. Initially, searches are carried out in official sources to obtain relevant information about the tools. In addition, two controlled experiments are conducted in a software development environment, using quantitative and qualitative metrics for analysis. It is expected to obtain insights about the efficiency of the tools.*

Resumo. *A seleção de ferramentas de infraestrutura é crucial para o sucesso das equipes de desenvolvimento, assim como a escolha da linguagem de programação para um projeto específico. Porém não está claro as diferenças de eficiência das ferramentas de suporte para o gerenciamento de infraestruturas como código. Essa falta de clareza, dificulta a escolha da melhor alternativa para cada contexto, tornando relevante a comparação das mesmas, como exemplo o Terraform e o AWS Cloud Development Kit. Dessa forma, o propósito deste estudo é identificar as diferenças entre as ferramentas em relação à facilidade de uso, escalabilidade, manutenibilidade e desempenho. Inicialmente, são realizadas pesquisas em fontes oficiais para obter informações relevantes sobre as ferramentas. Além disso, são conduzidos dois experimentos controlados em um ambiente de desenvolvimento de software, utilizando métricas quantitativas e qualitiativas para as análises. Espera-se obter insights sobre a eficiência das ferramentas.*

Bacharelado em Engenharia de *Software* - PUC Minas
Trabalho de Conclusão de Curso (TCC)

Orientador de conteúdo (TCC I): Cleiton Tavares - cleitontavares@pucminas.br
Orientador acadêmico (TCC I): Laerte Xavier - laertexavier@pucminas.br
Orientador do TCC II: (A ser definido no próximo semestre)

Belo Horizonte, 18 de junho de 2023.

1. Introdução

A cultura *DevOps* se refere a um conjunto de práticas e técnicas que visam a entrega rápida e confiável de *software* [Kim et al. 2018]. A automação, um dos pilares da cultura *DevOps*, propõe a infraestrutura como código (IaC, do inglês *Infrastructure as Code)*, uma técnica de gerenciamento de infraestruturas em que seus recursos são definidos e gerenciados via arquivos de configuração e códigos [Parnin et al. 2019]. Portanto, a gestão de IaC se encaixa no contexto de *DevOps* como uma prática fundamental para a entrega rápida e confiável de *software* em ambientes de *cloud*[1] por meio da automação de seus *scripts* [Patni et al. 2020]. A gestão desses ambientes tem se tornado relevante na Engenharia de *Software*, dada a necessidade de gerenciar seus recursos de maneira eficiente e escalável [Chappell 2008].

Estudos anteriores avaliam o uso das infraestruturas como código no processo de desenvolvimento de software. Por exemplo, eles evidenciam *code smells* inseridos durante o desenvolvimento de *scripts* de IaC [Alonso et al. 2023] e abordam a utilização das ferramentas de IaC para facilitar a implantação de um modelo arquitetural [Sandobalín et al. 2020]. Além desses estudos, ferramentas para gestão de infraestrutura foram propostas [Rahman et al. 2019]. Contudo, poucos estudos se propõem a comparar a aplicabilidade de diferentes ferramentas e identificar seus cenários de uso recomendados. Sendo assim, **o problema tratado neste trabalho é a escassez de clareza sobre a eficiência entre diferentes ferramentas de suporte para o gerenciamento de infraestruturas como código**.

A seleção de ferramentas de gerenciamento de IaC é crucial para o sucesso de qualquer equipe de desenvolvimento, assim como a escolha da linguagem de programação certa para um projeto específico. Da mesma maneira que existem estudos comparativos entre diferentes linguagens de programação [Bogner and Merkel 2022, Ray et al. 2014], é importante haver estudos comparativos entre ferramentas de provisionamento e gerenciamento de infraestruturas. Nesse sentido, se torna relevante a comparação entre ferramentas de gestão de IaC como o Terraform, ferramenta mais eficiente para implantações *Multicloud* [Gupta et al. 2021], e o *AWS Cloud Development Kit* (CDK), que abstrai o gerenciamento de infraestruturas na principal *cloud*, *Amazon Web Services* (AWS), utilizada atualmente [Vogels 2023]. Resolver o problema é fundamental para que as equipes de desenvolvimento possam tomar decisões mais assertivas e garantir o sucesso dos projetos na *cloud*.

Tendo em vista o contexto apresentado, o objetivo deste trabalho é **avaliar a eficiência do Terraform e do AWS CDK por meio de uma análise comparativa**. Para atingir esse objetivo, são propostos os seguintes objetivos específicos: (i) avaliar a abstração do CDK e Terrform para criação de infraestruturas a partir de desenvolvedores com menos experiência; (ii) avaliar, a partir de desenvolvedores experientes, a escalabilidade e manutenibilidade do Terraform e do CDK; e (iii) avaliar o desempenho do Terraform e do CDK na criação de recursos de infraestrutura na AWS.

Como resultado deste estudo, espera-se obter uma análise comparativa entre o Terraform e o AWS CDK. Com isso, pretende-se obter uma caracterização das ferramentas,

[1] Ambientes de *cloud* são ambientes que abstraem, agrupam e compartilham recursos escaláveis em uma rede. Locais onde as aplicações são executadas [RedHat 2023].

identificando suas diferenças em termos de facilidade de uso e eficiência. Além disso, espera-se evidenciar os cenários mais propícios à utilização de cada uma das ferramentas através das informações coletadas. As métricas utilizadas para a realização dessa análise incluem o tempo de execução das ferramentas, vulnerabilidades encontradas, facilidade de aprendizado, dentre outros fatores relevantes para a seleção e implantação das ferramentas de por parte das equipes de desenvolvimento.

O restante deste trabalho está dividido em quatro seções. A Seção 2 apresenta a fundamentação teórica sobre o *DevOps*, a infraestrutura como código, e as ferramentas utilizadas e o ambiente de *cloud* escolhido. A Seção 3 descreve os principais trabalhos relacionados. A Seção 4 detalha os materiais e métodos utilizados neste estudo.

2. Fundamentação Teórica

Esta seção fornece uma visão geral sobre *DevOps* e infraestrutura como código, bem como as ferramentas Terraform e *AWS Cloud Development Kit* (AWS CDK). O entendimento desses conceitos, tecnologias e práticas é fundamental para a compreensão do foco principal do trabalho realizado.

2.1. DevOps

O conceito de *DevOps* está focado na construção de uma cultura colaborativa entre equipes que são historicamente separadas em: equipes de desenvolvimento e equipes de operações [Loukides 2012]. Essa abordagem foi criada por P. Debois em 2008 na *Agile Conference*, onde foram propostas ideias e processos que visavam criar uma cultura de integração e colaboração entre os times, com o intuito de prover agilidade às demandas de clientes e solucionar empecilhos históricos entre essas equipes [Debois 2008].

Os times de desenvolvimento eram muitas vezes forçados a lançar novas versões de forma rápida, devido a prazos apertados, fazendo com que questões como à manutenibilidade ou estabilidade não fossem priorizadas. Já os times de operações, responsáveis por introduzir as novas versões, não conseguiam garantir a estabilidade e a execução dos produtos, o que ocasionava a criação de processos formais complicados que resultavam em menor colaboração entre as equipes [Dörnenburg 2018].

Sendo assim, a cultura proposta por Debois atua como um facilitador, principalmente no contexto de metodologias ágeis, graças ao foco em entregas contínuas de forma rápida, segura e colaborativa [Kim et al. 2018]. Dessa forma, o *DevOps* é uma resposta ao mercado ao possibilitar a implementação de mudanças contínuas em um *software*, fazendo com que os produtos sejam fluidos e passíveis de adaptações dentro de seu ciclo de desenvolvimento [Schlossnagle 2017]. Como consequência, o *software* desenvolvido é implantado de forma confiável o mais rápido possível [Mala 2019].

2.2. Infraestrutura como Código

Infraestrutura como código (IaC, do inglês *Infrastructure as Code)* é um conceito chave do *DevOps* que envolve a automação de processos de gerenciamento de infraestrutura por meio do uso de código [Ibrahim et al. 2022]. Na prática, isso significa que as configurações de infraestrutura, incluindo servidores, redes, bancos de dados, *firewalls* e outros componentes, são definidas e gerenciadas por meio de código. Essa abordagem permite que as equipes de *DevOps* gerenciem a infraestrutura de maneira mais efi-

ciente e escalável, eliminando a necessidade de intervenções manuais e reduzindo a probabilidade de erros humanos [Dalla Palma et al. 2022]. Ao codificar a infraestrutura, as configurações podem ser versionadas, rastreadas e facilmente reproduzidas, o que significa que as equipes podem implementar atualizações e correções com maior rapidez e confiança.

Além disso, a IaC permite que as equipes de *DevOps* criem ambientes de desenvolvimento e teste mais facilmente, o que ajuda a acelerar o ciclo de desenvolvimento. A infraestrutura como código também oferece maior transparência e colaboração entre as equipes de desenvolvimento e operações. Isso ocorre porque os desenvolvedores podem entender melhor a infraestrutura subjacente e as operações podem se envolver mais cedo no processo de desenvolvimento [Sandobalín et al. 2020]. Como exemplos de plataformas de infraestrutura como código podemos mencionar o Terraform e o CDK.

O Terraform [2] é uma ferramenta de orquestração de ambientes *cloud* desenvolvido pela Hashi Corp. Seu principal objetivo é aprimorar o fluxo de implantação de recursos na *cloud* por meio de seus módulos, uma estrutura predefinida onde para cada recurso há um módulo específico a ser utilizado para sua criação [Gupta et al. 2021]. Além disso, o Terraform possui um recurso de planejamento que mostra as mudanças que devem ser implementadas na infraestrutura com base no código que será executado [HashiCorp 2023]. O fluxo de trabalho principal do Terraform consiste em três estágios: Gravação, Planejar e Aplicar.

A *AWS Cloud Development Kit* [3] oferece uma abstração de alto nível para o *AWS CloudFormation*, permitindo que desenvolvedores usem linguagens de programação populares, como *TypeScript*, *JavaScript*, *Python*, *Java*, *.NET* e *Go*, para definir e implantar infraestrutura na AWS [Vogels 2023]. Essa abstração de alto nível possibilita que equipes de desenvolvimento criem templates de infraestrutura de forma mais rápida e com menos esforço, fornecendo flexibilidade e controle para personalizar a implantação conforme as necessidades específicas do projeto [Coulter 2023]. Além disso, a AWS CDK possui uma estrutura de teste incorporada, que permite que os desenvolvedores simulem a implantação, validem os recursos de infraestrutura e executem testes automatizados, garantindo que as definições de infraestrutura estejam funcionando corretamente antes da implantação na AWS [Lovchikova 2023].

3. Trabalhos Relacionados

Os trabalhos relacionados discutidos nesta seção abrangem a implementação dinâmica e ágil, a aplicação de testes para garantir sua confiabilidade e a mensuração da qualidade da infraestrutura.

O primeiro trabalho relacionado é Dalla Palma et al. (2020) que tem como objetivo identificar e catalogar métricas de qualidade de software em códigos de infraestrutura, em *Ansible*. Diante disso, com a intenção de estabelecer uma base sólida para a avaliação da qualidade do IaC, os autores definiram 46 métricas agrupadas em três categorias distintas: (i) métricas orientadas a objetos que podem ser portadas para Ansible e IaC em geral, (ii) métricas que foram investigadas em trabalhos anteriores sobre IaC e que podem

[2]https://developer.hashicorp.com/terraform?product$_intent = terraform$

[3]https://docs.aws.amazon.com/pt$_br/cdk/v2/guide/home.html$

ser portadas para o Ansible, e (iii) métricas relacionadas às melhores e más práticas no Ansible. Nesse contexto, os resultados obtidos evidenciaram que as métricas empregadas na avaliação da qualidade do código de software podem ser aplicadas com sucesso na avaliação da qualidade do código da infraestrutura, incluindo a identificação de *smells*. Portanto, embora tenha sido criado para o *Ansible*, esse estudo tem o potencial de enriquecer a metodologia deste trabalho ao aplicar algumas das métricas do catálogo apresentado para avaliar a infraestrutura em Terraform e em AWS CDK.

Gupta (2021) tem como intuito avaliar a eficácia do uso das ferramentas *Ansible* e Terraform na implantação da arquitetura *Hadoop* em um ambiente em *cloud*. Para isso, os autores implementaram *scripts* de automação utilizando as ferramentas de IaC, além de realizar testes de carga para avaliar a escalabilidade e o desempenho da arquitetura implantada. Diante disso, os resultados mostraram que as tecnologias utilizadas possibilitaram a implantação rápida, eficiente e escalável da arquitetura *Hadoop*. Desse modo, embora essa pesquisa aborde contextos arquiteturais diferentes, é relevante para o artigo em questão, pois fornece informações e práticas importantes sobre o gerenciamento de infraestrutura, apresentando diferentes abordagens e ferramentas que podem ser úteis, além da aplicabilidade dos testes para verificar a escalabilidades das infraestruturas. Dessa forma, o estudo conduzido por Gupta se diferencia deste por não compara ferramentas de infraestrutura distintas, e sim as utilizas em cunjunto.

Sokolowski e Salvaneschi (2023) discute as melhores práticas e abordagens para garantir a confiabilidade das infraestruturas. Para garantir a confiabilidade, os autores provem o uso de uma ferramenta de teste automatizada *ProTI*, baseada em testes de unidade. Essa ferramenta, é um *fuzzer* para programas em Pulumi [4] que adota técnicas modernas para testar programas IaC em diferentes configurações. Por conseguinte, os resultados evidenciaram uma diminuição da carga de trabalho manual e um aumento da confiança nos resultados, proporcionando um *feedback* rápido e abrangente sobre as infraestruturas. Concluindo, a relação entre este trabalho e o presente estudo reside no processo de automatização das infraestruturas, porem serão utilizadas as ferramentas AWS CDK e Terraform para a automatização ao invez do Pulumi.

Bogner e Merkel (2022) tem o objetivo de comparar a qualidade do *software* entre aplicações *JavaScript* e *TypeScript* no *GitHub*. Para alcançar esse objetivo, os autores selecionaram uma amostra de 604 aplicativos com mais de 16 milhões de linhas de código, 575.699 *commits* e 214.075 problemas analisados. Considerando as métricas como tamanho do código, complexidade ciclomática e número de defeitos, os resultados mostraram que aplicações *TypeScript* tendem a ter menos defeitos e menor complexidade ciclomática do que aplicações *JavaScript*. Portanto, diferente deste trabalho que avalia linguagens de programacao, esse estudo presente pretende avaliar plataformas de IaC e suas implementações.

Parnin et al. (2019) auxiliam na prevenção de práticas inseguras de codificação em *scripts* de IaC. Os autores por meio da análise de 1.726 scripts IaC identificaram sete *security smells* e desenvolveram a ferramenta Security Linter for Infrastructure as Code *scripts* (SLIC) para realizar análise estática. Os resultados revelaram a presença

[4]Plataforma de automação de infraestrutura baseada em código que permite que os usuários criem, implantem e gerenciem infraestrutura em várias nuvens usando linguagens de programação de abstrações de *clound*

de 21.201 ocorrências de *security smells*, incluindo 1.326 senhas diretamente codificadas no código. Essas descobertas são valiosas para melhorar a qualidade do código, mitigar vulnerabilidades e garantir a eficiência e segurança na gestão da infraestrutura. Além disso, essas descobertas oferecem *insights* importantes para a pesquisa atual na detecção de práticas inseguras de codificação em scripts de IaC, servindo como um referencial para a criação de padrões sólidos de IaC.

4. Materiais e Métodos

Este trabalho utiliza uma abordagem descritiva quantitativa e qualitativa para realizar uma pesquisa comparativa entre as ferramentas. Além disso, são conduzidos dois experimentos controlados nos quais são aplicadas intervenções específicas, criando um ambiente propício para análise. Nesse sentido, são coletadas métricas quantitativas como parte do processo. O objetivo principal do estudo é descrever e comparar as ferramentas com base nessas métricas, proporcionando informações objetivas e mensuráveis. Com isso, busca-se obter dados confiáveis e replicáveis que fundamentem decisões informadas acerca do uso das ferramentas.

Para realizar a comparação detalhada entre o Terraform e o AWS CDK, são utilizados métodos que combinam dados qualitativos e quantitativos. São coletados dados buscando informações sobre as ferramentas em fontes confiáveis, como documentação oficial e comunidades de usuários. São conduzidos experimentos práticos usando casos de uso reais, permitindo avaliar a eficiência das ferramentas em situações que refletem os desafios enfrentados pelos usuários no mundo real. São utilizadas estratégias de amostragem adequadas para garantir a representatividade dos experimentos e minimizar a influência de fatores externos. Esses métodos permitem identificar as diferenças entre as ferramentas e determinar qual delas é mais eficiente em cada métrica de interesse. A combinação desses métodos possibilita uma análise completa e robusta da eficiência do Terraform e do AWS CDK, sendo fundamental para a tomada de decisões informadas sobre a escolha da ferramenta mais adequada para diferentes cenários.

4.1. Coleta de Informações e Análise Comparativa

Para obter uma compreensão completa das ferramentas Terraform e AWS CDK, é necessário seguir uma abordagem estruturada, focando na identificação das limitações distintas de cada uma delas. Essa análise é fundamental para realizar comparações justas e coerentes. A pesquisa deve ser baseada em fontes confiáveis, como a documentação oficial de ambas as ferramentas, fóruns de suporte, comunidades de usuários, tutoriais e exemplos disponíveis *online*.

Durante a coleta de informações, é importante que sejam identificados os módulos e estruturas padrões recomendadas para a criação de recursos na AWS. Esses componentes pré-construídos fornecem abstrações convenientes que agilizam o desenvolvimento e fornecem uma base sólida para construir infraestruturas escaláveis e confiáveis em *cloud*. A documentação oficial de cada ferramenta é o recurso primário para obter informações sobre esses módulos e sua localização. É essencial identificar a disponibilidade de módulos pré-prontos que simplificam a criação de recursos comuns em *cloud*, a fim de saber se uma ferramenta possibilita a gerência de mais recursos que a outra.

Além disso, é fundamental coletar informações sobre as atualizações e a evolução

das ferramentas ao longo do tempo. Isso permite compreender a dinâmica do desenvolvimento e do uso das ferramentas, além de identificar novos recursos, melhorias e correções incorporados em versões mais recentes. A análise das atualizações e evoluções proporciona uma visão mais completa das funcionalidades, limitações e melhores práticas de uso do Terraform e do AWS CDK. Outro aspecto relevante é obter informações sobre a instalação e o uso básico das ferramentas. Conhecer os requisitos de sistema, o processo de instalação correto e a estrutura básica dos arquivos de configuração é fundamental para iniciar a utilização adequada do Terraform e do AWS CDK.

Nesse sentido, é importante destacar as limitações distintas de cada ferramenta, considerando aspectos como recursos não suportados, funcionalidades experimentais, restrições de escalabilidade ou compatibilidade com serviços específicos da AWS. A consulta às fontes oficiais, como a documentação, é a base confiável para obter informações sobre a arquitetura, os recursos suportados, a sintaxe, as opções de configuração e as melhores práticas de uso.

Ao seguir uma abordagem estruturada, coletando informações sobre as limitações, os módulos e estruturas padrão, as atualizações e a evolução das ferramentas, bem como a instalação e o uso básico, é possível obter uma compreensão abrangente do Terraform e do AWS CDK. Essa compreensão permite uma análise comparativa sólida das funcionalidades, das melhores práticas e dos possíveis desafios que podem surgir ao utilizar cada ferramenta. Assim, é possível auxiliar o processo de tomada de decisões ao escolher a ferramenta mais adequada para projetos específicos de gestão de infraestrutura na AWS.

4.2. Avaliando o Impacto da Abstração de Ferramentas

Neste experimento controlado, o objetivo é avaliar o impacto da abstração de ferramentas de gestão de infraestrutura, em particular, as ferramentas Terraform e AWS CDK. Dois grupos de alunos foram selecionados para participar do experimento, recebendo uma introdução geral sobre o funcionamento e o uso das ferramentas, a fim de familiarizá-los com os conceitos básicos e as funcionalidades oferecidas por cada uma delas.

O experimento é conduzido em duas etapas: na primeira etapa, um grupo de alunos utiliza o Terraform para criar recursos na AWS, seguindo as práticas e estruturas padrão recomendadas pela documentação oficial. Na segunda etapa, outro grupo de alunos realiza a mesma tarefa, porém utilizando o AWS CDK como ferramenta de gestão de infraestrutura. Durante o experimento, são coletados dados e métricas para avaliar o desempenho e a eficiência de cada grupo ao utilizar as respectivas ferramentas. Alguns dos aspectos observados incluem:

1. Tempo de implementação: É verificado o tempo necessário para a criação e configuração dos recursos na AWS utilizando cada uma das ferramentas. Isso permite comparar a eficiência e a produtividade dos grupos em suas respectivas abordagens.
2. Facilidade de uso: É avaliada a facilidade com que os alunos conseguiram compreender e utilizar as ferramentas. É possível observar se a abstração fornecida pelo AWS CDK resulta em uma curva de aprendizado mais suave ou se a sintaxe declarativa do Terraform é mais intuitiva.
3. Escalabilidade e manutenibilidade: São considerados aspectos como a capacidade de gerenciar recursos em grande escala e realizar atualizações e modificações de

forma fácil e eficiente. É possível observar se a abstração do AWS CDK proporciona uma maior flexibilidade e facilidade na gestão desses aspectos.

4.3. Avaliando a Execução dos Módulos de Infraestrutura

Para avaliar o desempenho do Terraform e do AWS CDK na criação de recursos de infraestrutura na AWS, é essencial definir um contexto que englobe as interações entre serviços e recursos, bem como desafios e limitações típicos do gerenciamento de IaC. A partir desse cenário, um plano de execução é elaborado. Nesse plano, são criados scripts replicáveis utilizando o Terraform e o AWS CDK para provisionar recursos em nuvem. Conforme a Tabela 1, é possivel ver exemplos dos recursos utilizados para a analise deste estudo.

Tabela 1. Exemplos de recursos da AWS.

Nome do Serviço	Sigla do Serviço
Amazon Elastic Compute Cloud	EC2
Amazon Simple Storage Service	S3
Amazon Elastic Container Registry	ECR

Esses scripts são desenvolvidos com foco nas estruturas e módulos padrão recomendados pelas fontes oficiais de cada ferramenta [5], garantindo uma avaliação justa e imparcial. Para garantir a replicabilidade [6] do experimento, algumas modificações são feitas em características dos recursos, já que algumas configurações excedem os limites do pacote gratuito (*Freetier*) da AWS. Um exemplo é a necessidade de ajustar o tipo de instância do EC2 para *T2.micro*. Além disso, com o objetivo de manter um ambiente controlado, planeja-se a montagem do mesmo através de código, sendo utilizadas cinco instâncias EC2. Na Tabela 2 é apresentada a configuração das instâncias utilizas para esse experimento [7].

Tabela 2. Especificações das Instâncias EC2

Tipo de Instância	T2.micro
Sistema Operacional	Ubuntu 20.04
CPU	1
RAM (GiB)	1,0

Durante o experimento descrito, são coletadas as seguintes métricas para avaliar as ferramentas:

- Quantidade de elementos de infraestrutura: É quantificado o número de elementos criados. Isso permite garantir que o código de cada ferramenta implemente uma infraestrutura idêntica, a fim de garantir uma comparação justa.

[5]https://registry.terraform.io/providers/hashicorp/aws/latest/docs e https://docs.aws.amazon.com/cdk/api/v2/docs/aws-construct-library.html

[6]https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2023-1-tcci-0393100-pes-lucas-padrao-e-joao-victor-frois/tree/master/Instrumentos/Codigos

[7]https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2023-1-tcci-0393100-pes-lucas-padrao-e-joao-victor-frois/tree/master/Instrumentos/OutrosInstrumentos

- Uso de recurso computacional: É medida a porcentagem de uso da memória e a porcentagem de uso da CPU. Isso possibilita a avaliação do impacto das ferramentas nos componentes computacionais.
- Tempo de execução: É calculado a média/mediana do tempo de 5 execuções para criar o elemento de infraestrutura, registrados em segundos. Isso permite avaliar a eficiência de cada uma das ferramentas na criação de recursos de infraestrutura.

Referências

Alonso, J., Piliszek, R., and Cankar, M. (2023). Embracing iac through the devsecops philosophy: Concepts, challenges, and a reference framework. *IEEE Software*, 40(1):56–62.

Bogner, J. and Merkel, M. (2022). To type or not to type? a systematic comparison of the software quality of javascript and typescript applications on github. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, pages 658–669.

Chappell, D. (2008). A short introduction to cloud platforms: An enterprise-oriented view. Technical report, David Chappell & Associates.

Coulter, M. (acessado em 31 de março de 2023). Matt coulter talks aws cdk, meeting werner vogels, silent discos.

Dalla Palma, S., Di Nucci, D., Palomba, F., and Tamburri, D. A. (2020). Toward a catalog of software quality metrics for infrastructure code. *Journal of Systems and Software*, 170:110726.

Dalla Palma, S., Di Nucci, D., Palomba, F., and Tamburri, D. A. (2022). Within-project defect prediction of infrastructure-as-code using product and process metrics. *IEEE Transactions on Software Engineering*, 48(6):2086–2104.

Debois, P. (2008). Agile infrastructure and operations: How infra-gile are you? In *Agile 2008 Conference*, pages 202–207.

Dörnenburg, E. (2018). The path to devops. *IEEE Software*, 35(5):71–75.

Gupta, M., Chowdary, M. N., Bussa, S., and Chowdary, C. K. (2021). Deploying hadoop architecture using ansible and terraform. In *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, pages 1–6.

HashiCorp (acessado em 31 de março de 2023). Introduction to terraform.

Ibrahim, A., Yousef, A. H., and Medhat, W. (2022). Devsecops: A security model for infrastructure as code over the cloud. In *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pages 284–288.

Kim, G., Humble, J., Debois, P., and Willis, J. (2018). *Manual de DevOps: como obter agilidade, confiabilidade e segurança em organizações tecnológicas*. Editora Alta Books, Rio de Janeiro, 1a ed. edition.

Loukides, M. (2012). *What is DevOps?* O'Reilly Media, 1a ed. edition.

Lovchikova, N. (acessado em 31 de março de 2023). Aws summit anz 2021 - driving a test-first strategy with cdk and test driven development.

Mala, D. J. (2019). *Integrating the Internet of Things Into Software Engineering Practices*. IGI Global, 1a ed. edition.

Parnin, C., Rahman, A., and Williams, L. (2019). The seven sins: Security smells in infrastructure as code scripts. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 164–175.

Patni, J. C., Banerjee, S., and Tiwari, D. (2020). Infrastructure as a code (iac) to software defined infrastructure using azure resource manager (arm). In *2020 International Conference on Computational Performance Evaluation (ComPE)*, pages 575–578.

Rahman, A., Mahdavi-Hezaveh, R., and Williams, L. (2019). A systematic mapping study of infrastructure as code research. *Information and Software Technology*, 108:65–77.

Ray, B., Posnett, D., Filkov, V., and Devanbu, P. (2014). A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, page 155–165, New York, NY, USA. Association for Computing Machinery.

RedHat (acessado em 22 de março de 2023). Introdução à cloud computing.

Sandobalín, J., Insfran, E., and Abrahão, S. (2020). On the effectiveness of tools to support infrastructure as code: Model-driven versus code-centric. *IEEE Access*, 8:17734–17761.

Schlossnagle, T. (2017). Monitoring in a devops world: Perfect should never be the enemy of better. *Queue*, 15(6):35–45.

Sokolowski, D. and Salvaneschi, G. (2023). Towards reliable infrastructure as code. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*, pages 318–321.

Vogels, W. (acessado em 03 de março de 2023). Werner vogels on the aws cloud development kit (aws cdk).