INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE - CÂMPUS PASSO FUNDO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RUAN DELATORRE MOTTA

IMPLEMENTAÇÃO E ADOÇÃO DE CULTURA E PRÁTICAS DEVOPS UTILIZANDO INFRAESTRUTURA COMO CÓDIGO E COMPUTAÇÃO EM NUVEM

Prof. Me. Élder Bernardi

PASSO FUNDO
2023

RUAN DELATORRE MOTTA

IMPLEMENTAÇÃO E ADOÇÃO DE CULTURA E PRÁTICAS DEVOPS UTILIZANDO INFRAESTRUTURA COMO CÓDIGO E COMPUTAÇÃO EM NUVEM

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense, Campus Passo Fundo, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Élder Bernardi

PASSO FUNDO

2023

DEDICATÓRIA

Dedico este trabalho de conclusão de curso à mulher extraordinária que sempre foi minha fonte inesgotável de inspiração e apoio, minha querida mãe. Seu amor incondicional, sabedoria e incentivo constante foram pilares fundamentais para minha jornada acadêmica. Mãe, este trabalho é dedicado a você, como expressão sincera da minha gratidão e reconhecimento pela luz que você trouxe à minha vida.

RESUMO

Este trabalho aborda a cultura DevOps, uma abordagem que visa modernizar empresas através da integração entre desenvolvimento e operações, visando entregas mais ágeis. Originado das palavras "Desenvolvimento" e "Operações", essa metodologia engloba pessoas, ferramentas e processos para viabilizar métodos ágeis, reduzindo os intervalos entre desenvolvimento e operações. O projeto propõe demonstrar as vantagens do DevOps, implementando um ambiente automatizado na AWS com infraestrutura como código. O objetivo é evidenciar os benefícios dessa cultura, destacando a relevância para empresas de TI, explorando pilares como boas práticas, serviços gerenciados em nuvem e a contratação de Engenheiros DevOps. O trabalho busca promover a compreensão prática e teórica da cultura e seu impacto positivo.

O trabalho prático demonstrou de forma eficaz a transição gradual de um ambiente manual para um ambiente automatizado, destacando a aplicação bem-sucedida da IaC na plataforma AWS. A implementação foi dividida em duas etapas, começando com a criação manual dos recursos essenciais para integração e entrega contínua. Posteriormente, utilizando *pipelines* previamente configurados, a infraestrutura completa para microsserviços foi implementada, evidenciando a agilidade e eficiência proporcionadas pela automação. Os resultados obtidos refletem a eficácia ao automatizar todo o ciclo de vida de desenvolvimento e implantação de infraestrutura. A arquitetura proposta e colocada em execução demonstra a aplicação prática dos princípios DevOps, integrando desenvolvimento e operações de maneira fluida.

Palavras-chave: DevOps. AWS. Iac. Pipelines.

ABSTRACT

This paper discusses the DevOps culture, an approach aimed at modernizing companies through the integration of development and operations, with a focus on achieving more agile deliveries. Derived from the words "Development" and "Operations," this methodology encompasses people, tools, and processes to enable agile methods, reducing the gaps between development and operations. The project aims to demonstrate the advantages of DevOps by implementing an automated environment on AWS with infrastructure as code. The goal is to highlight the benefits of this culture, emphasizing its relevance to IT companies, exploring pillars such as best practices, cloud-managed services, and the hiring of DevOps Engineers. The work seeks to promote both practical and theoretical understanding of the culture and its positive impact.

This work addresses the DevOps culture, an approach aimed at modernizing companies through the integration of development and operations, with the goal of achieving more agile deliveries. Originating from the words "Development" and "Operations," this methodology encompasses people, tools, and processes to enable agile methods, reducing the gaps between development and operations. The project aims to demonstrate the advantages of DevOps by implementing an automated environment on AWS with Infrastructure as Code (IaC). The objective is to highlight the benefits of this culture, emphasizing its relevance for IT companies and exploring pillars such as best practices, cloud-managed services, and the hiring of DevOps Engineers. The work seeks to promote both practical and theoretical understanding of the culture and its positive impact.

Keywords: DevOps. AWS. Iac. Pipelines.

LISTA DE FIGURAS

Figura 1 – Crescimento da utilização de Computação em nuvem por empresas . . . 11
Figura 2 – Ciclo DevOps 13
Figura 3 – Pipeline utilizando a integração entre os serviços da AWS 14
Figura 4 – Arquitetura implementação DevOps AWS 22
Figura 5 – Template CloudFormation CI/CD parte 1 24
Figura 6 – Template CloudFormation CI/CD parte 2 25
Figura 7 – Buildspec.yaml (AWS CodeBuild) 26
Figura 8 – Resultado Stack CloudFormation 27
Figura 9 – Template CloudFormation Microsserviços parte 1 28
Figura 10 – Template CloudFormation Microsserviços parte 2 30
Figura 11 – Execução do AWS CodePipeline 31
Figura 12 – Visualização do Microsserviço através do AWS ELB 32

LISTA DE ABREVIATURAS E SIGLAS

AWS - Amazon Web Services

ECR - Elastic Container Registry

ECS - Elastic Container Service

VPC - Virtual Private Cloud

CI/CD - Continuous Integration/Continuous Delivery

IaC - Infrastructure as Code

ELB - Elastic Load Balancer

DNS - Domain Name System

IAM - Identity Access Management

SUMÁRIO

1 **INTRODUÇÃO** 5
2 **REFERENCIAL TEÓRICO** 9
2.1 **DEVOPS** 9
2.1.1 **DevOps utilizando Computação em nuvem** 10
2.2 **INTEGRAÇÃO E ENTREGA CONTÍNUA (CI/CD)** 12
2.2.1 **Ciclo DevOps** 12
2.2.2 **AWS CodePipeline, CodeBuild e CodeDeploy** 13
2.3 **VERSIONAMENTO DE CÓDIGO COM GIT** 15
2.3.1 **AWS CodeCommit** 15
2.4 **CONTAINERS E VERSIONAMENTO DE IMAGENS DOCKER** 15
2.4.1 **AWS Elastic Container Registry** 16
2.5 **MICROSSERVIÇOS** 16
2.5.1 **AWS Elastic Container Service** 17
2.6 **MONITORAMENTO, OBSERVABILIDADE E REGISTRO DE LOG** 17
2.6.1 **AWS CloudWatch** 18
2.7 **INFRAESTRUTURA COMO CÓDIGO (IAC)** 19
2.7.1 **CloudFormation** 20
3 **ESTUDO DE CASO IMPLEMENTAÇÃO** 21
3.1 **IMPLEMENTAÇÃO DE PRÁTICAS DEVOPS UTILIZANDO IAC NA NUVEM DA AWS** 22
3.2 **IMPLEMENTAÇÃO DE PIPELINES CI/CD** 23
3.3 **IMPLEMENTAÇÃO DE INFRAESTUTURA PARA MICROSSERVIÇOS** . . . 27
4 **CONSIDERAÇÕES FINAIS** 33
REFERÊNCIAS 34

1 INTRODUÇÃO

Com base na busca pela modernização das empresas e pela adoção de ferramentas e tecnologias mais atuais surgiu a cultura DevOps com intuito de tornar as entregas dos produtos mais ágeis. O termo DevOps é uma combinação das palavras "Desenvolvimento" e "Operações", sendo originário do inglês "Development" e "Operations". Essa união de pessoas, metodologias, ferramentas e processos tem como objetivo viabilizar métodos ágeis reduzindo os intervalos entre o desenvolvimento e operações.

Adotar essa cultura não significa apenas o trabalho adjunto entre as equipes, mas sim a adoção de todos os pilares que compõem essa metodologia. A automatização de processos por exemplo é uma das práticas realizadas para implementação dessa estrutura, automatizar o processo de CI/CD que consiste na integração/entrega contínua de códigos computacionais é extremamente necessário para garantir um desenvolvimento mais rápido e ágil.

Este projeto tem como objetivo demonstrar para as empresas quais são as vantagens em utilizar a metodologia e práticas DevOps, Para demonstração de como pode ser utilizado na prática, o trabalho tem como propósito a implementação de um ambiente automatizado utilizando infraestrutura como código para provisionamento dos recursos na nuvem, demonstrando as vantagens e quais os benefícios de adotar essa metodologia em um ambiente corporativo.

O objetivo do trabalho é demonstrar para pequenas e grandes empresas quais são as vantagens em utilizar a metodologia e práticas DevOps dentro da AWS (Amazon Web Services), uma das principais plataformas de computação em nuvem. Para que isso seja possível, serão abordados os pilares necessários para viabilização desse modelo de trabalho, a cultura de boas práticas e a utilização de serviços gerenciados da nuvem para utilização de Integração e entrega contínua, infraestrutura como código, microsserviços, versionamento de código, monitoramento e registro de *log*.

O trabalho tem por objetivo prático:

a) Implementar uma infraestrutura automatizada, utilizando os princípios da cultura DevOps e Infraestrutura como código dentro da nuvem a partir da utilização do serviço Cloud Formation.

b) Relatar a importância e vantagem da utilização da cultura DevOps pelas pequenas e grandes empresas de TI. Como definição desta metodologia será abrangida a explicação de todos os pilares que descrevem a cultura, estrutura, objetivos, automações usadas, estratégias, comparações, colaboração e compartilhamento de informações. Mostrar quais os benefícios de utilizar a nuvem para implementar essa cultura, utilizando serviços da Amazon Web Services (AWS) e implementando um ambiente totalmente gerenciável.
c) Apresentar como pode ser relevante para as empresas a contratação de Engenheiros DevOps, quais benefícios eles podem trazer, como podem ser utilizados, quais são seus conhecimentos e habilidades.

2. REFENCIAL TEÓRICO

Nesse capítulo serão apresentados os conceitos fundamentais relacionados ao tema.

2.1 DEVOPS

Durante muito tempo as organizações do segmento de tecnologia da informação estão enfrentando grandes desafios, com o lançamento de novas tecnologias a todo momento destaca-se a necessidade de se adequar constantemente às exigências que o mercado traz, buscando melhor atendimento de seus clientes e fornecendo soluções inovadoras cada vez mais rápidas (LIMA, 2021). A partir da missão de endereçar essas necessidades surge a ideia central do DevOps que as equipes de desenvolvimento e operações devem trabalhar juntas como uma única entidade, ao invés de entidades separadas com objetivos e metas diferentes. Essa abordagem enfatiza a importância da comunicação, automação e compartilhamento de responsabilidades para alcançar um ciclo de desenvolvimento mais ágil.

A principal meta do DevOps é melhorar a integração entre os desenvolvedores e a equipe de infraestrutura. Essas funções são frequentemente submetidas a pressões por parte das empresas. Os desenvolvedores são responsáveis por entregar valor por meio de funcionalidades e aplicações, enquanto a equipe de infraestrutura precisa garantir a estabilidade dos serviços. O DevOps busca superar essa dicotomia ao promover uma cultura de colaboração e compartilhamento de responsabilidades. Ele incentiva as equipes a trabalharem juntas desde o início do ciclo de desenvolvimento, em vez de se isolarem em suas respectivas áreas. Isso envolve a adoção de práticas de automação, integração contínua e entrega contínua, onde as alterações no código são testadas e implantadas em ambientes de produção de forma rápida e segura (BRANCO, 2020).

Com a implementação do DevOps, as duas equipes podem passar a trabalhar juntas de uma forma mais eficiente melhorando a produtividade e participando do ciclo de vida completo desde o desenvolvimento até a infraestrutura da aplicação, com uma comunicação frequente de forma a melhorar a qualidade do serviço que é entregue aos clientes (BRANCO, 2020).

A cultura deve ser vista como um acelerador do desenvolvimento das empresas, que se baseia em metodologias ágeis permitindo aumento da performance e da

produtividade organizacional do departamento de TI, reduzindo custos no ciclo de vida e melhorando a eficiência operacional com um maior alinhamento de negócio entre equipes de desenvolvimento e operações (FORSGREN; HUMBLE; KIM, 2018).

Existem várias vantagens resultantes da adoção do DevOps, como a velocidade de resolução dos problemas, os benefícios culturais que contribuem para a melhoria da comunicação e para a criação de laços mais fortes entre os colaboradores de diferentes equipes, os benefícios de negócios relacionados ao cliente final e a velocidade de entrega de valor (SOUSA, 2019). Esses diversos benefícios são o resultado da aplicação de princípios e práticas como versionamento de código, integração e entrega contínua, monitoramento e registro de *log* das aplicações, utilização de microsserviços e infraestrutura como código.

A computação em nuvem pode ser uma aceleradora para uma empresa passar a usufruir desses processos e práticas, com serviços gerenciados e provisionamento sob demanda de forma imediata. .

2.1.1 DevOps utilizando Computação em nuvem

Um desafio para as empresas é buscar um padrão para implementar o DevOps, a partir dessa necessidade plataformas baseadas em Nuvem disponibilizam um conjunto de serviços flexíveis que foram criados com intuito de permitir que as empresas criem e distribuam produtos de forma rápida e com maior segurança utilizando práticas DevOps dentro da Nuvem. O objetivo desses serviços é simplificar o provisionamento e o gerenciamento da infraestrutura, a implantação do código do aplicativo e a automação do lançamento de novas versões do software. Sobre essa perspectiva com a computação em nuvem podemos implementar todas as práticas e processos necessários para adoção dessa cultura dentro de um único ecossistema e usufruir dos benefícios centralizando toda a infraestrutura e o desenvolvimento em uma única plataforma.

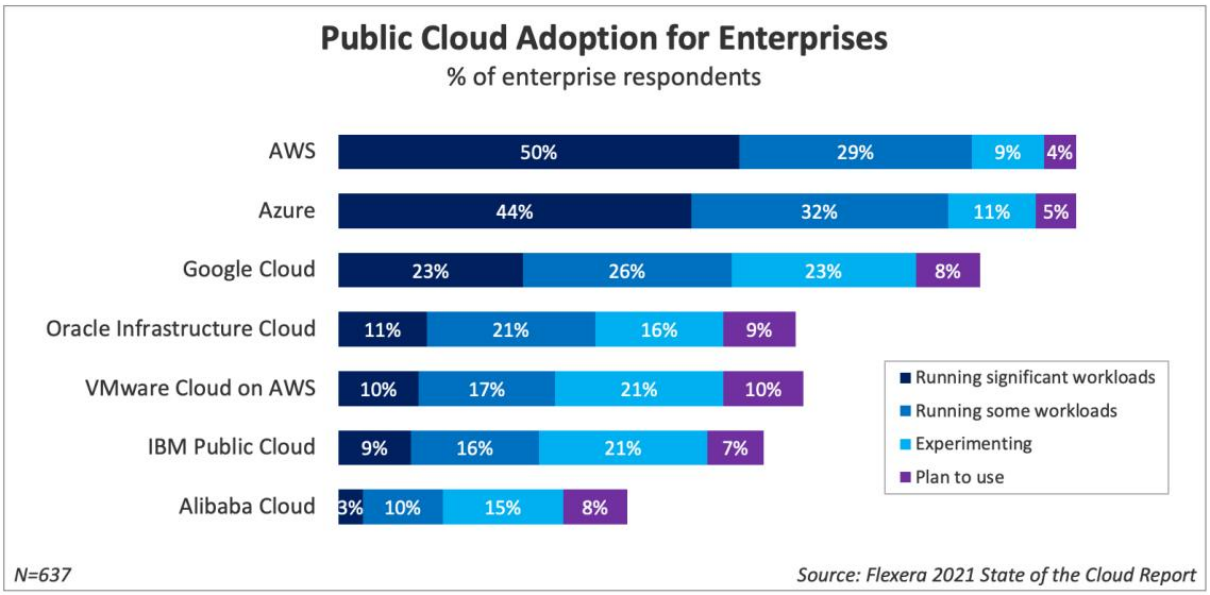
Com a enorme variedade de tecnologias disponíveis surge também a necessidade de estudo para utilização e configuração destas. Sendo assim as empresas do ramo de tecnologia necessitam de profissionais com seu foco voltado para soluções relacionadas a DevOps. Diante disso surge o Engenheiro DevOps, seu papel dentro de uma corporação além de implementar as práticas dessa cultura é propor melhorias e motivar a utilização de ferramentas da nuvem que facilitam o

crescimento das empresas, também deve possuir habilidades interpessoais para criar um ambiente colaborativo entre as equipes de desenvolvimento (BUFFA, 2021).

A adoção da nuvem é uma oportunidade de transformação, garantindo que as perspectivas estejam alinhadas com as necessidades do negócio. Essa adoção já é uma realidade entre diversas empresas de tecnologia de diferentes ramos, segundo o levantamento "Infosys Cloud Radas 2021", feito pela empresa global de consultoria e serviços digitais Infosys com companhias de 12 setores nos Estados Unidos, Alemanha, França, Austrália, Nova Zelândia e Reino Unido. O estudo aponta que a adoção de tecnologias da nuvem podem agregar US$ 414 bilhões anuais de lucro a essas empresas (INFOSYS, 2021).

Também podemos observar na Figura 1 diante da pesquisa "State of the cloud Report" (FLEXERA,2021) realizada pela empresa americana de soluções de TI Flexera o crescimento da utilização de soluções da nuvem, onde se destaca a Amazon Web Services.

Figura 1 - Crescimento da utilização de Computação em nuvem por empresas



Fonte: FLEXERA, 2021.

2.1 INTEGRAÇÃO E ENTREGA CONTÍNUA

Antes dessa prática existir, era comum os desenvolvedores perderem bastante tempo antes de publicar suas alterações em uma versão nova do código. A integração contínua é uma prática de desenvolvimento onde os desenvolvedores

integram suas alterações em um repositório central utilizando versionamento de código com o objetivo de investigar erros rapidamente e dar visualização do código para toda a equipe. Essa prática reduz o tempo necessário para validar e juntar as novas versões do código (BRANCO, 2020).

A Integração contínua exige que o projeto seja compilado a cada mudança realizada. Caso ocorra algum erro nesse processo, deve ser interrompido imediatamente de forma automatizada e a equipe de desenvolvimento pode investigar o erro sem que a aplicação fique indisponível.

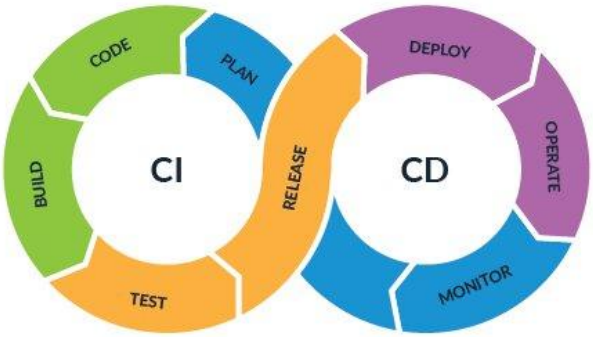
Com a entrega contínua cada alteração de código é criada, testada e publicada em um ambiente, diferente da integração o objetivo é atualizar automaticamente um software sem aprovação explícita (AWS, 2023c). Dessa forma pode-se combinar esses padrões no que chamamos de *pipelines* que se resumem em um conjunto de processos automatizados que permitem aos desenvolvedores e profissionais de operações trabalharem de maneira coesa na criação e implementação de novos código em um ambiente (HALL, 2023).

Essa prática melhoraria a produtividade do desenvolvedor, permitindo investigação de erros e bugs mais rapidamente, distribuindo atualizações com maior frequência e automatizando o processo de lançamento de software. A implementação desses processos se torna muito mais simples com serviços gerenciados prontos para utilização, sem a necessidade de provisionamento de ferramentas.

2.2.1 Ciclo DevOps

O Ciclo do DevOps ilustra processos que podem ser implementados na integração e entrega contínua de códigos, esse processo se resume em fases necessárias para o desenvolvimento de software e entrega de uma versão do produto. A Figura 2 representa todas as fases do desenvolvimento de uma aplicação. O processo de desenvolvimento de software geralmente começa com o planejamento, seguido pela fase de codificação.

Figura 2 - Ciclo DevOps CI/CD



Fonte: PARASOFT, 2021.

A equipe de desenvolvimento é responsável por executar as tarefas de implementação, enquanto a equipe de negócios define os requisitos a serem atendidos. As fases subsequentes, que incluem *build*, *test*, *release* e *deploy*, são frequentemente automatizadas por meio de práticas de integração contínua e entrega contínua. Nesses estágios, o código desenvolvido anteriormente é compilado, testado e instalado em diferentes ambientes, até chegar ao cliente final.

a) *Build:* Fase *que* envolve a compilação do código e a realização de testes iniciais, como testes unitários.
b) *Test: O*nde são realizados testes unitários, de aceitação, integridade, desempenho e não funcionais. Esses testes validam aspectos como código, serviços, escalabilidade, segurança e disponibilidade, entre outros.
c) *Release:* Fase diretamente relacionada ao *deploy*, sendo que a *release* envolve disponibilizar ao público um conjunto de alterações.
d) *Deploy:* É a execução do processo de instalação em um ambiente específico.

As duas fases finais são de operações e monitoramento, nas quais as equipes garantem a integridade e a manutenção das aplicações. Além disso, métricas são coletadas constantemente para orientar os próximos desenvolvimentos, visando garantir uma melhoria contínua (SOUSA, 2019). Por meio desse ciclo, com feedback constante em cada fase, é possível entregar software de maior valor ao cliente final.

2.2.2 AWS CodePipeline, CodeBuild e CodeDeploy

Dentro da nuvem torna-se muito mais fácil configurar os processos de Integração/Entrega contínua, com serviços gerenciados. Para compilação podemos

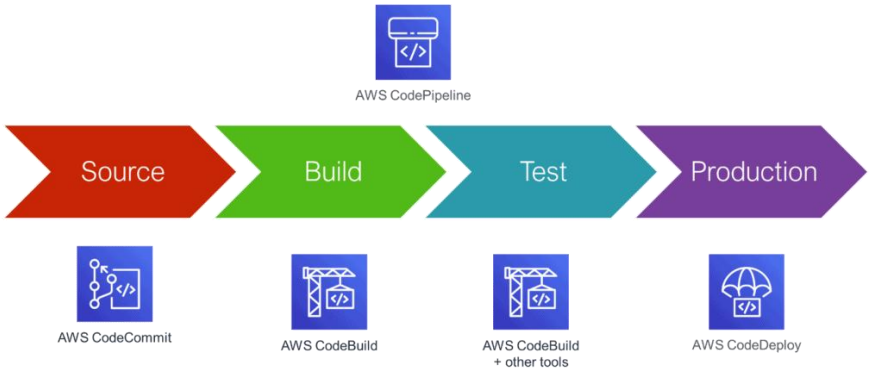
utilizar dentro da nuvem o AWS CodeBuild, sem a necessidade de gerenciar e escalar servidores de compilação, basta especificar a localização do código de origem e o serviço executa scripts para compilar, testar e empacotar o código (AWS, 2023a).

Eliminando a necessidade de operações manuais, a implementação de aplicativos pode ser feita em diversos serviços de computação utilizando o AWS CodeDeploy. Esse serviço oferece controle centralizado das alterações por meio de interface e protege seu serviço contra o tempo de inatividade monitorando a saúde da implementação. Também é possível visualizar histórico e definir notificações a níveis de eventos que informam o status de todos os lançamentos, aumentando a confiabilidade e velocidade do processo de entrega de software (AWS, 2023g).

Com o AWS CodePipeline, é viável integrar esses serviços e automatizar a criação de *pipelines* de lançamento para fornecer atualizações ágeis e confiáveis para aplicações e infraestruturas. Ele proporciona uma interface gráfica intuitiva, permitindo que os usuários criem, configurem e gerenciem pipelines juntamente com todas as suas etapas e ações. Essa interface simplifica a visualização e modelagem dos fluxos de trabalho do processo de liberação, tornando o gerenciamento do ciclo de vida de desenvolvimento mais acessível e eficiente. Visualizar o progresso e históricos de execução pela interface é outro benefício do serviço, permitindo revisão e acompanhamento das execuções, incluindo horários de início e término (AWS, 2023h).

Pode-se observar a representação de um pipeline e a integração desses serviços na Figura 3.

Figura 3 - Pipeline utilizando a integração entre os serviços da Amazon Web Services



Fonte: GAUR, 2022.

2.3 VERSIONAMENTO DE CÓDIGO COM GIT

Podemos definir versionamento de código como o gerenciamento de versões diferentes de um código, normalmente utilizada no desenvolvimento de software. Essa técnica traz como principal vantagem permitir que várias pessoas de uma mesma equipe trabalhe junto em um conjunto de arquivos ao mesmo tempo, onde cada um tem sua versão dos arquivos e no final das alterações é colocada junto das versões dos demais colegas (BATAGINI, 2023).

O versionamento de código pode ser utilizado com Git, uma ferramenta de controle de versão descentralizada altamente eficiente, que possibilita uma colaboração organizada entre desenvolvedores na construção de projetos de código (BATAGINI, 2023). Segurança e controle de histórico motivam a utilização do Git, avaliar no detalhe o que foi mudado por cada colaborador entre as versões e fornecer backup do código indiretamente são benefícios muito relevantes no desenvolvimento.

2.3.1 AWS CodeCommit

O AWS CodeCommit é um serviço de controle de código totalmente gerenciado, seguro e escalável onde é possível provisionar sob demanda repositórios baseados em Git (AWS, 2023i). A utilização desse recurso parte da facilidade de acesso e integração, sendo muito benéfico em um ambiente totalmente dentro da nuvem utilizá-lo em conjunto com os serviços de Integração e Entrega contínua referenciados no capítulo 2.2 e na Figura 3, onde em um pipeline podemos integrar a origem dos códigos a partir do AWS Code Commit.

2.4 CONTAINERS E VERSIONAMENTO DE IMAGENS DOCKER

Containers são unidades de software leves e independentes que encapsulam uma aplicação, suas dependências e configurações necessárias para executar de forma consistente em diferentes ambientes. Eles oferecem isolamento, portabilidade e eficiência, permitindo que os desenvolvedores empacotem todo o ambiente de execução de uma aplicação em uma única unidade, conhecida como imagem de container. Essas imagens é a representação do aplicativo contendo o código da aplicação, bibliotecas, frameworks e outras dependências, garantindo que a aplicação funcione consistentemente em qualquer ambiente que suporte containers

(MICROSOFT, 2023).

Para executar um microsserviço, é criado um container referenciando uma imagem específica. Geralmente essas imagens são baseadas em Docker, um projeto de software livre para automatizar implantações de aplicativos autossuficientes que podem ser executados na nuvem ou localmente (MICROSOFT, 2023).

A cada atualização do aplicativo uma nova imagem é lançada, as versões da aplicação são definidas pelas tags que são concedidas a essas imagens. O versionamento de imagens parte da necessidade de adquirir resiliência mantendo todas as versões das diversas aplicações de uma empresa em um mesmo local, para isso utilizamos repositórios de imagens.

2.4.1 AWS Elastic Container Registry

No mundo das aplicações em containers, gerenciar e implantar imagens de container é um aspecto fundamental do processo de desenvolvimento e implantação. Para atender a essa necessidade, a amazon oferece o AWS ECR, um serviço gerenciado de registro de imagens de container que é seguro, escalável e confiável. Com o Amazon ECR, desenvolvedores e organizações podem armazenar, gerenciar e implantar facilmente imagens de container em sua infraestrutura dentro da nuvem (AWS, 2023d).

A escalabilidade do Amazon ECR é uma vantagem significativa para organizações com cargas de trabalho em container em crescimento. O serviço dimensiona automaticamente para atender às demandas crescentes de armazenamento e recuperação de imagens, garantindo que as imagens estejam prontamente disponíveis e acessíveis para suas aplicações e processos de implantação. Seja com um número pequeno de imagens de container ou um repositório vasto, o serviço pode lidar com a escala e o volume das necessidades de gerenciamento de imagens.

2.5 MICROSSERVIÇOS

Conforme definido por Stoiber e citado por (LUCIO, 2017) um microsserviço é um componente de software autônomo que, juntamente com outros, compõem uma aplicação maior. Ao dividir o aplicativo em unidades menores, cada parte pode ser

implantada e escalada de forma independente. Além disso, cada microsserviço pode ser desenvolvido por equipes distintas, utilizando diferentes linguagens de programação, e pode ser testado individualmente. Essa abordagem proporciona flexibilidade e facilita a manutenção e evolução do sistema como um todo.

Essa prática conciliada a integração e entrega contínua, versionamento de código e versionamento de imagens trazendo ainda mais agilidade às equipes. Ao adotar microsserviços, é possível tratar as etapas do *pipeline* de forma isolada, o que agiliza a identificação e correção de erros sem afetar o sistema como um todo. Além disso, os microsserviços permitem trabalhar com várias equipes simultaneamente, garantindo que cada funcionalidade esteja funcionando corretamente. Isso resulta em uma experiência do usuário aprimorada, proporcionando um melhor desempenho e qualidade ao sistema (BARROS, 2022).

2.5.1 AWS Elastic Container Service

O AWS ECS é um serviço de orquestração de containers totalmente gerenciado que simplifica o processo de implantação, gerenciamento e escalabilidade de aplicativos em containers na AWS.

Com o ECS, você pode executar containers facilmente, sem a necessidade de gerenciar a infraestrutura subjacente, permitindo que você se concentre no desenvolvimento e execução de seus aplicativos (TECHTARGET, 2023). Com o Amazon ECS, os desenvolvedores podem extrair as imagens do Amazon ECR e executar em containers, dimensionando o aplicativo e gerenciando continuamente a disponibilidade.

O ECS também oferece recursos avançados de escalabilidade e balanceamento de carga. Você pode dimensionar automaticamente seus serviços de container com base na demanda, garantindo que seus aplicativos estejam sempre disponíveis e respondendo às necessidades de tráfego.

2.6 MONITORAMENTO, OBSERVABILIDADE E REGISTRO DE LOG

Com o aumento da complexidade dos sistemas de TI, é cada vez mais comum ocorrerem erros e falhas que ultrapassam a compreensão dos desenvolvedores. Por esse motivo, o monitoramento e a observabilidade se tornaram atividades essenciais

e complementares para garantir a qualidade e o bom funcionamento de um sistema, bem como dos serviços conectados à ele (PEREIRA, 2022).

O monitoramento refere-se à coleta sistemática de dados e métricas relacionadas ao desempenho e ao estado dos componentes do sistema. Isso inclui métricas como a utilização de recursos, a latência, o tráfego de rede, a taxa de erros e outros indicadores relevantes. O monitoramento é normalmente realizado por meio de ferramentas de monitoramento especializadas, que coletam e exibem os dados em painéis e relatórios que permitem aos administradores e desenvolvedores acompanhar a saúde do sistema em tempo real.

A observabilidade, por outro lado, vai além do monitoramento tradicional. Ela se concentra na capacidade de compreender o comportamento interno do sistema, entender o fluxo de dados, rastrear transações e identificar a causa raiz de problemas quando eles ocorrem. A observabilidade busca fornecer insights sobre o funcionamento interno dos componentes do sistema, permitindo uma análise mais profunda e uma melhor compreensão dos eventos que ocorrem durante a execução do serviço.

Para alcançar uma observabilidade efetiva, é comum utilizar técnicas como a geração de registros de *logs* detalhados e o rastreamento de solicitações. Os logs detalhados registram informações sobre eventos e ações específicas, permitindo uma análise posterior em caso de problemas. O rastreamento de solicitações envolve o acompanhamento de uma solicitação específica à medida que ela atravessa os diferentes componentes do sistema, facilitando a identificação de gargalos e falhas em potencial.

2.6.1 AWS CloudWatch

O Amazon CloudWatch é um serviço de monitoramento e observabilidade da AWS para os recursos da nuvem e as aplicações que executadas na AWS. Você pode usar o Amazon CloudWatch para coletar e rastrear métricas, coletar e monitorar arquivos de log e definir alarmes. Pode-se utilizar painéis automatizados em tempo real para otimizar a infraestrutura e manutenção de aplicações (AWS, 2023e).

O CloudWatch permite que se armazene registros de aplicativos e sistemas em um local centralizado. Você pode enviar registros diretamente para o CloudWatch

Logs, isso facilita a busca, a análise e o monitoramento de registros, permitindo identificar problemas, depurar aplicativos e rastrear eventos específicos.

Pode-se também utilizar dentro desse serviço recursos avançados de alarmes, que permitem definir condições específicas com base nas métricas monitoradas, podendo acionar notificações. Isso possibilita uma resposta rápida a eventos importantes, como um aumento repentino na carga de um servidor ou uma falha em um serviço crítico.

2.7 INFRAESTRUTURA COMO CÓDIGO (IAC)

Segundo Buchanan a ideia de Infraestrutura como Código, ou modelagem de infraestrutura com código, foi estimulada pelo sucesso da integração e entrega contínua. O DevOps comprovou a produtividade fazendo utilização dessa prática. A automação que esses fluxos de trabalho trouxeram para o desenvolvimento de software ajudou a reduzir a nova complexidade da administração de sistemas em nuvem (BUCHANAN, 2023).

Infraestrutura como código pode ser definida como um processo de gerenciamento de recursos de TI que aplica práticas recomendadas do DevOps ao gerenciamento de serviços de infraestrutura na nuvem, esses serviços podem ser máquinas virtuais, balanceadores de carga, banco de dados ou qualquer recurso disponível dentro das soluções da nuvem (BUCHANAN, 2023). Sendo uma abordagem que permite criar, provisionar e gerenciar infraestrutura por meio de arquivos de configuração e scripts, em vez de configurá-la manualmente. Com o IaC, a infraestrutura é tratada como código e pode ser versionada, testada e implementada de forma consistente e automatizada.

A ideia por trás da IaC é aplicar práticas de desenvolvimento de software, como controle de versão, testes automatizados e integração contínua, à infraestrutura de TI. Em vez de configurar servidores, redes e outros recursos manualmente, os engenheiros DevOps definem sua infraestrutura desejada em um arquivo de configuração, utilizando linguagens específicas, como YAML ou JSON. Esse arquivo descreve os recursos necessários, suas propriedades e dependências.

Ao adotar essa prática, existem várias vantagens significativas como:

- **Menor tempo de produção/lançamento no mercado:** Agiliza o fornecimento de infraestrutura, permitindo uma entrega mais rápida para desenvolvimento, teste e

produção. Também facilita o ajuste de escala e desativação da infraestrutura quando necessário.

- **Melhor consistência, menor desvio de configurações:** Evita desvios de configuração, garantindo que todos os ambientes mantenham a mesma configuração, reduzindo problemas de implementação, vulnerabilidades de segurança e riscos de conformidade.
- **Desenvolvimento mais rápido e eficiente:** Simplifica o fornecimento de infraestrutura, acelerando todas as fases do ciclo de vida de entrega de software.
- **Reprodutibilidade:** Como a infraestrutura é definida por código, ela pode ser reproduzida facilmente em diferentes ambientes, como desenvolvimento, teste e produção. Isso garante consistência e evita discrepâncias entre as configurações.
- **Proteção contra rotatividade:** Assegura que o conhecimento de provisionamento permaneça com a organização, mesmo quando especialistas deixam a empresa, evitando a necessidade de reconstruir processos.
- **Documentação viva:** O código de infraestrutura serve como documentação viva do ambiente. As configurações e dependências estão claramente definidas nos arquivos, tornando mais fácil entender e auditar a infraestrutura.
- **Integração com práticas DevOps:** O IaC é um componente chave da abordagem DevOps, permitindo a automação da infraestrutura e a colaboração entre equipes de desenvolvimento e operações.
- **Controle de versão:** A IaC permite versionar o código da infraestrutura, assim como qualquer outro código de software. Isso facilita o rastreamento das mudanças, o trabalho em equipe e a reversão para versões anteriores, se necessário.

Oferecendo uma abordagem moderna e eficiente para o provisionamento e gerenciamento de infraestrutura. Ao tratar a infraestrutura como código, é possível alcançar automação, consistência, escalabilidade e flexibilidade, tornando a implantação e a manutenção de sistemas mais ágeis e confiáveis (IBM, 2023).

2.7.1 AWS CloudFormation

O AWS CloudFormation é uma ferramenta que permite modelar e provisionar recursos dentro da AWS de forma rápida e consistente, tratando a infraestrutura

como código. Com o CloudFormation, é possível descrever os recursos desejados e suas dependências em um modelo, permitindo iniciar e configurar esses recursos como uma pilha. Em vez de gerenciar os recursos individualmente, é possível criar, atualizar e excluir uma pilha inteira como uma unidade única, facilitando o gerenciamento da infraestrutura (AWS, 2023f).

Ao criar uma pilha no CloudFormation, o serviço se encarrega de provisionar e configurar automaticamente os recursos definidos no modelo, como microsserviços do Amazon ECS, repositórios do AWS Code Commit, registros do Amazon ECR e muitos outros serviços da AWS. O CloudFormation gerencia as dependências entre os recursos e garante a ordem correta de criação ou atualização.

Uma vez que uma pilha tenha sido criada, você pode usar o CloudFormation para atualizar ou excluir a pilha de forma fácil e controlada. O serviço permite o versionamento dos modelos e acompanha as alterações feitas na infraestrutura ao longo do tempo.

Podendo se beneficiar desse serviço para implementar práticas DevOps, o CloudFormation é uma ferramenta poderosa para automatizar a criação e o gerenciamento de recursos de infraestrutura na AWS, proporcionando uma abordagem escalável, consistente e controlada para a implantação de aplicativos e ambientes na nuvem.

No próximo capítulo, será detalhado a execução dessas etapas apresentando como a implementação bem-sucedida impacta positivamente as práticas DevOps, utilizando o AWS CloudFormation como uma ferramenta central nesse processo.

3. ESTUDO DE CASO IMPLEMENTAÇÃO

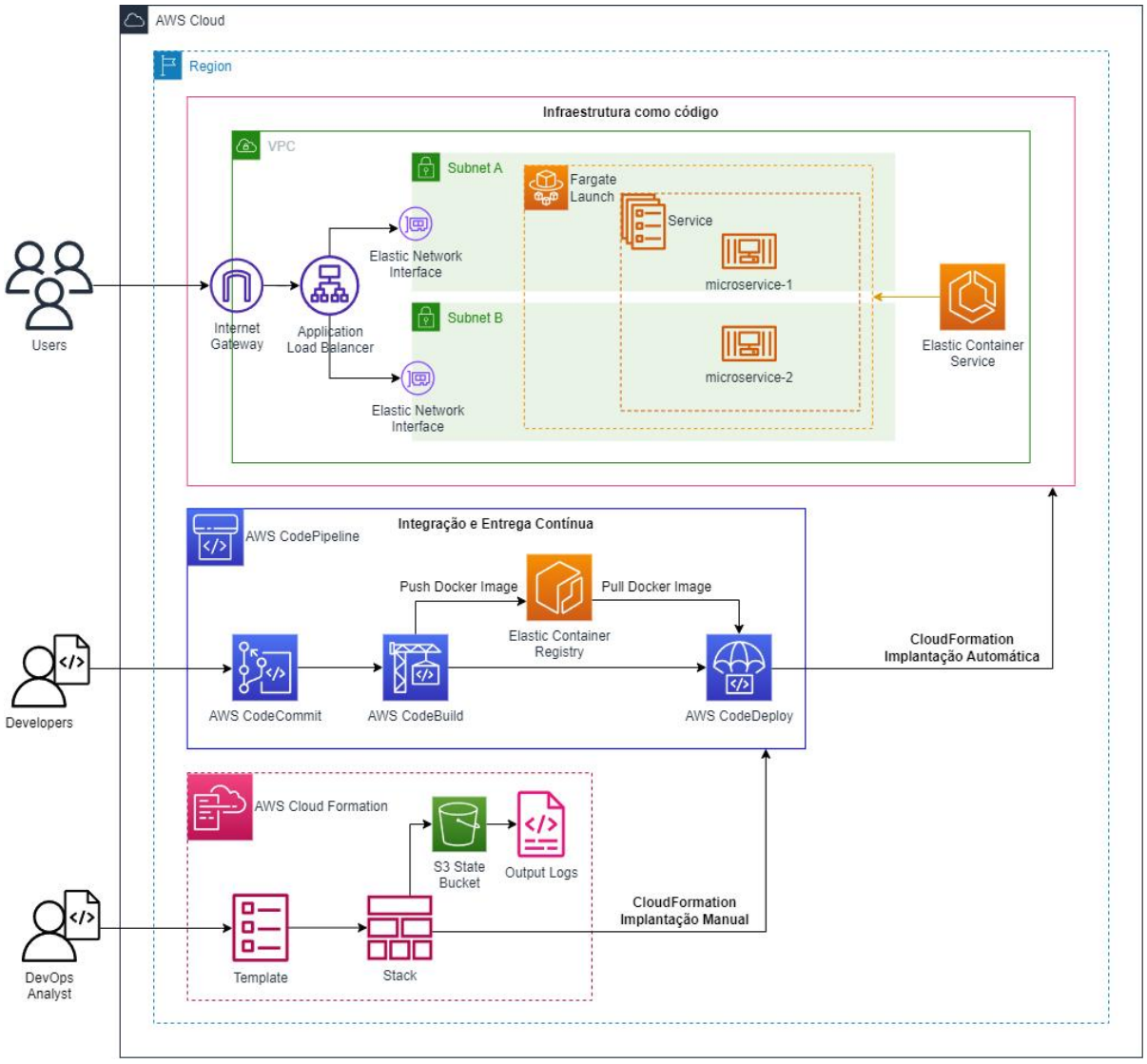
Para demonstrar de maneira prática como a Infraestrutura como código pode acelerar a adoção de práticas DevOps, nesse capítulo será apresentado a definição dos objetivos e o processo de implementação. O projeto buscou não apenas teorizar sobre a cultura DevOps, mas demonstrar sua aplicação prática por meio da implementação de um ambiente automatizado na nuvem, utilizando o AWS CloudFormation. A transição gradual de uma criação manual para uma abordagem automatizada evidenciou a escalabilidade e eficácia da automação, destacando como a Infraestrutura como Código acelera a adoção das práticas DevOps.

3.1 Implementação de práticas DevOps utilizando IaC na Nuvem da AWS

O processo de implementação será dividido em duas etapas. A primeira etapa é composta pela criação manual dos recursos essenciais para integração e entrega contínua. Posteriormente, aproveitando os pipelines que foram previamente configurados, realizaremos a implementação de uma infraestrutura completa para microsserviços. Essa abordagem prática demonstra a transição gradual de um ambiente manual para um ambiente automatizado, destacando a eficácia do AWS CloudFormation na automação de todo o ciclo de vida de desenvolvimento e implantação de aplicativos. O desenho representado na Figura 4 demonstra a arquitetura que será colocada em execução.

A arquitetura consiste em uma base para implantar e escalar microsserviços em conteiners. A infraestrutura é configurada em uma VPC com duas sub-redes públicas em zonas de disponibilidade distintas. Um cluster ECS é criado para executar contêineres Fargate, definidos por uma tarefa com suas configurações e imagem Docker especificadas nos parâmetros. Um balanceador de carga gerencia o tráfego, com grupos de segurança para controle de acesso. A VPC é conectada à Internet por meio de um Internet Gateway, e as tarefas ECS têm um papel IAM para permissões.

Figura 4 - Arquitetura implementação DevOps AWS



Fonte: Própria, 2023.

Essa arquitetura representa uma maneira eficiente de automatizar todo o processo de desenvolvimento, teste e implantação de microsserviços na AWS, garantindo uma entrega contínua e confiável de seus aplicativos. Além disso, ela proporciona flexibilidade e escalabilidade para atender às crescentes demandas de sua aplicação.

3.2 Implementação de pipelines CI/CD

O AWS CloudFormation oferece a flexibilidade de ser utilizado manualmente por meio da console de gerenciamento da Nuvem. Nesse processo, é possível criar uma pilha a partir de um template predefinido. Esse template contém o código que

descreve os recursos e configurações desejados. Ao criar a pilha, inicia-se o processo de provisionamento, no qual o AWS CloudFormation automatiza a implementação dos recursos conforme especificado no código do template. Esse método permite uma abordagem intuitiva e visual para gerenciar e orquestrar a infraestrutura na nuvem, facilitando a criação e atualização de ambientes complexos de maneira eficiente e consistente, na Figura 5 está representado no código os recursos que serão necessários para implementação do pipeline, são eles:

- **ECRRepository**: Repositório para armazenar imagens Docker.
- **S3Bucket:** Bucket para armazenar informações e logs de build.
- **CodePipeLineExecutionRole:** Função IAM para ser usada pelo AWS CodePipeline, concede permissões administrativas à função.
- **CodeBuildExecutionRole:** Função IAM para ser usada pelo AWS CodeBuild.
- **CloudformationExecutionRole:** Função IAM para ser usada pelo AWS CloudFormation.

Figura 5 - Template CloudFormation CI/CD parte 1

```yaml
AWSTemplateFormatVersion: 2010-09-09
Description: Template CloudFormation para criação dos recursos necessário para pipeline de Integração e Entrega Contínua.

Parameters:
  Stage:
    Type: String
    Default: dev
  ContainerPort:
    Type: Number
    Default: 3000

Resources:
  ################## Deploy Infraestrutura CICD #################
  # Criação de Repositório ECR para armazenar imagens Docker
  ECRRepository:
    Type: AWS::ECR::Repository
    Properties:
      RepositoryName: !Join ['-', ['ecr-repository' ,!Ref Stage]]

  # Criação de Bucket S3 - logs CodeBuild
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: !Join ['-', ['bucket-artifacts', !Ref Stage,]]

  # Criação da Access Role para o CodePipeLine
  CodePipeLineExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ['-', ['CodePipelineExecutionRole',!Ref Stage,]]
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: codepipeline.amazonaws.com
            Action: 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AdministratorAccess'

  # Criação da Access Role para o CodeBuild
  CodeBuildExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ['-', ['CodeBuildExecutionRole',!Ref Stage,]]
      AssumeRolePolicyDocument:
        Statement:
          Effect: Allow
          Principal:
            Service: codebuild.amazonaws.com
          Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AdministratorAccess

  # Criação da Access Role para o Cloudformation
  CloudformationExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !Join ['-', ['CloudFormationExecutionRole',!Ref Stage,]]
      AssumeRolePolicyDocument:
        Statement:
          - Effect: Allow
            Principal:
              Service: cloudformation.amazonaws.com
            Action: 'sts:AssumeRole'
      ManagedPolicyArns:
        - 'arn:aws:iam::aws:policy/AdministratorAccess'
```

Fonte: Própria, 2023.

A partir desse código é criado na sequencia os recursos que compõem as fases do Pipeline, representado na Figura 6, sendo:

- **CodeCommitRepository:** Repositório Git para armazenar o código fonte.
- **BuildProject:** Projeto para realizar a compilação.
- **CodePipeLine:** Pipeline com três estágios (*Source*, *Build* e *Deploy*). Os estágios são configurados para interagir com os serviços AWS CodeCommit, CodeBuild e CloudFormation. O estágio de *Deploy* cria ou atualiza uma *stack* CloudFormation para implantar os microsserviços ECS.

Figura 6 -Template CloudFormation CI/CD parte 2

```
  CodeCommitRepository:
    Type: "AWS::CodeCommit::Repository"
    Properties:
      RepositoryName: devops-iac-microservice-frontend
      RepositoryDescription: Repositório utilizado para implementação de microsserviços utilizando CICD.

  BuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Artifacts:
        Type: CODEPIPELINE
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: aws/codebuild/standard:7.0
        ImagePullCredentialsType: CODEBUILD
        PrivilegedMode: True
        Type: LINUX_CONTAINER
        EnvironmentVariables:
          - Name: ECR_REPOSITORY_URI
            Value: !Join [ ".", [ !Ref "AWS::AccountId", "dkr.ecr", !Ref "AWS::Region", !Join [ "/", [ "amazonaws.com", !Ref
"ECRRepository" ] ] ] ]
      Name: !Join ['-', ['build-project', !Ref Stage, 'ecs']]
      ServiceRole: !Ref CodeBuildExecutionRole
      Source:
        Type: CODEPIPELINE
        BuildSpec: buildspec.yml

  CodePipeLine:
    Type: AWS::CodePipeline::Pipeline
    DependsOn: S3Bucket
    Properties:
      ArtifactStore:
        Location: !Join ['-', ['bucket-artifacts', !Ref Stage,]]
        Type: S3
      Name: !Join ['-', ['pipeline', !Ref Stage, 'ecs']]
      RestartExecutionOnUpdate: False
      RoleArn:
        Fn::GetAtt: [ CodePipeLineExecutionRole, Arn ]

      Stages:
        - Name: Source
          Actions:
          - Name: Source
            ActionTypeId:
              Category: Source
              Owner: AWS
              Provider: CodeCommit
              Version: 1
            Configuration:
              RepositoryName: devops-iac-microservice-frontend
              BranchName: dev
            RunOrder: 1
            OutputArtifacts:
              - Name: source-output-artifacts

        - Name: Build
          Actions:
          - Name: Build
            ActionTypeId:
              Category: Build
              Owner: AWS
              Version: 1
              Provider: CodeBuild
            OutputArtifacts:
              - Name: build-output-artifacts
            InputArtifacts:
              - Name: source-output-artifacts
            Configuration:
                ProjectName: !Ref BuildProject
            RunOrder: 1

        - Name: Deploy
          Actions:
          - Name: Deploy
            ActionTypeId:
              Category: Deploy
              Owner: AWS
              Version: 1
              Provider: CloudFormation
            InputArtifacts:
              - Name: source-output-artifacts
              - Name: build-output-artifacts
            Configuration:
                ActionMode: CREATE_UPDATE
                Capabilities: CAPABILITY_NAMED_IAM
                ParameterOverrides: !Sub |
                  {
                    "ImageURI" : { "Fn::GetParam" : ["build-output-artifacts", "imagedefinitions.json", "ImageURI"] },
                    "Stage":  "${Stage}",
                    "ContainerPort": "${ContainerPort}"
                  }
                RoleArn:
                  Fn::GetAtt: [ CloudformationExecutionRole, Arn ]
                StackName: devops-aws-iac-microservices
                TemplatePath: source-output-artifacts::cloudformation/iac-microservices.yml
            RunOrder: 1
```

Fonte: Própria, 2023.

O arquivo buildspec.yml (representado na Figura 7) exemplifica as fases de pré *build*, *build* e pós *build* utilizada em nosso projeto do AWS CodeBuild, ele

desempenha um papel fundamental no processo de automação da compilação e construção de imagens Docker. Esse arquivo é uma configuração do CodeBuild que especifica as fases e comandos a serem executados durante o pipeline de compilação. Essencial para a integração do CodeBuild com o Amazon ECR e o Docker, garantindo uma compilação bem-sucedida e o upload das imagens geradas. Ele automatiza a autenticação no repositório ECR, constrói a imagem Docker a partir do Dockerfile, adiciona tags relevantes e, por fim, realiza o *push* dessas imagens para o repositório. O arquivo imagedefinitions.json é criado como um artefato, contendo informações sobre a URI da imagem e sua tag associada, possibilitando o uso dessas informações em etapas subsequentes do pipeline, como no deploy usando o AWS CloudFormation. Este arquivo exemplifica as boas práticas de CI/CD ao automatizar tarefas cruciais no processo de desenvolvimento e implantação de aplicações baseadas em containers.

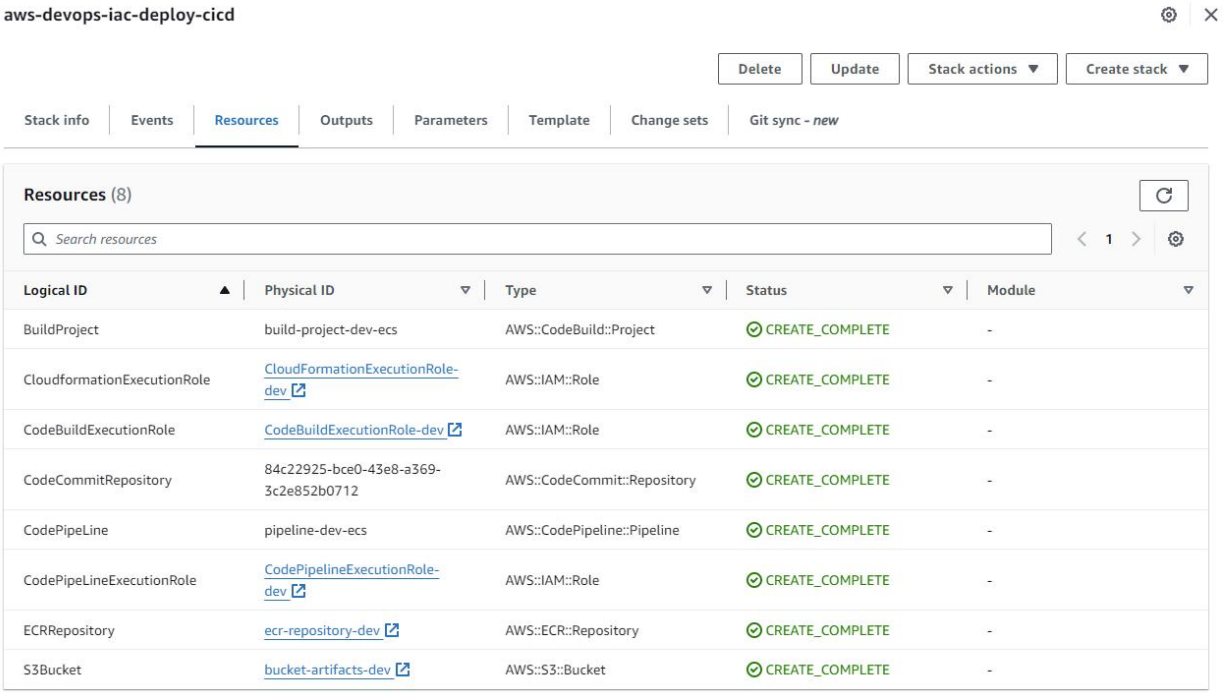
Figura 7 - Buildspec.yaml (AWS CodeBuild)

```yaml
version: 0.2
phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --username AWS --password-stdin
$ECR_REPOSITORY_URI
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $ECR_REPOSITORY_URI:latest .
      - docker tag $ECR_REPOSITORY_URI:latest $ECR_REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $ECR_REPOSITORY_URI:latest
      - docker push $ECR_REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '{"ImageURI":"%s:%s"}' $ECR_REPOSITORY_URI $IMAGE_TAG > imagedefinitions.json
artifacts:
  files:
    - imagedefinitions.json
```

Fonte: Própria, 2023.

Após a execução manual desse template utilizando o CloudFormation, é possível acessar cada recursos criados no console da AWS, pode ser observado no resultado da execução da *stack* cada *resource* e seu status de implementação, conforma representa a imagem abaixo.

Figura 8 - Resultado Stack CloudFormation



Fonte: Própria, 2023.

3.3 Implementação de infraestrutura para Microsserviços

A execução da infraestrutura para os microsserviços será conduzida pelo pipeline, conforme definido anteriormente no capítulo 2.8.1. Quando ocorre uma publicação no repositório configurado, o CodePipeline é acionado automaticamente, iniciando o processo de criação da infraestrutura necessária para dar suporte aos microsserviços no ambiente ECS (Amazon Elastic Container Service). Este pipeline, previamente estabelecido, automatiza integralmente o fluxo de execução do AWS CloudFormation. Ao detectar alterações no código hospedado no repositório, especialmente na pasta "cloudformation/iac-microservices.yaml", o pipeline inicia a execução do CloudFormation. Esse processo abrange todas as etapas cruciais, desde a configuração inicial da VPC (Virtual Private Cloud) e das sub-redes até a elaboração do balanceador de carga e a configuração dos serviços ECS associados.

Inicialmente, o pipeline cria um cluster ECS, fornecendo uma plataforma robusta para a execução dos microsserviços. Em seguida, estabelece uma VPC com duas sub-redes públicas, garantindo a conectividade e permitindo o acesso à internet através de um *Internet Gateway* e Tabelas de rotas. Está representado na Figura 9 a definição desses recursos.

Figura 9 - Template CloudFormation Microsserviços parte 1

```yaml
AWSTemplateFormatVersion: 2010-09-09
Description: Template CloudFormation para criação da infraestrutura necessário para Microsserviços.

Parameters:
  Stage:
    Type: String
  ContainerPort:
    Type: Number
  ImageURI:
    Type: String

Resources:
  Cluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: !Join ['-', ['cluster-ecs', !Ref Stage]]

  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 172.10.0.0/16
      EnableDnsHostnames: True
      EnableDnsSupport: True
      Tags:
        - Key: Name
          Value: !Join ['-', ['vpc', !Ref Stage]]

  SubnetA:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 172.10.1.0/24
      VpcId: !Ref VPC
      AvailabilityZone: !Join ['', [!Ref "AWS::Region", 'a']]
      Tags:
        - Key: Name
          Value: !Join ['-', ['subnet-pub-a', !Ref Stage]]

  SubnetB:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 172.10.2.0/24
      VpcId: !Ref VPC
      AvailabilityZone: !Join ['', [!Ref "AWS::Region", 'b']]
      Tags:
        - Key: Name
          Value: !Join ['-', ['subnet-pub-b', !Ref Stage]]

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref VPC

  PublicRoute:
    Type: AWS::EC2::Route
    DependsOn: VPCInternetGatewayAttachment
    Properties:
      RouteTableId: !Ref PublicRouteTable
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref InternetGateway

  SubnetAPublicRouteAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref SubnetA

  SubnetBPublicRouteAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      RouteTableId: !Ref PublicRouteTable
      SubnetId: !Ref SubnetB

  InternetGateway:
    Type: AWS::EC2::InternetGateway

  VPCInternetGatewayAttachment:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref InternetGateway
      VpcId: !Ref VPC
```

Fonte: Própria, 2023.

A definição de tarefas ECS é configurada, especificando os recursos necessários para a execução dos containers, enquanto as *roles* IAM apropriadas são criadas para conceder permissões específicas aos serviços ECS. A infraestrutura de rede é amplamente configurada, incluindo a criação de um balanceador de carga que

distribuirá o tráfego entre os microsserviços ECS. Esse balanceador de carga é associado a um grupo de destinos e a um ou mais serviços ECS, garantindo a escalabilidade e a alta disponibilidade. Além disso, medidas de segurança são implementadas através da criação de grupos de segurança para o balanceador de carga e os containers, restringindo o tráfego conforme necessário. O pipeline também cria um ou mais serviços ECS, especificando a definição de tarefas, o número desejado de instâncias e a configuração de rede, Todas essas configurações estão representadas como código na Figura 10.

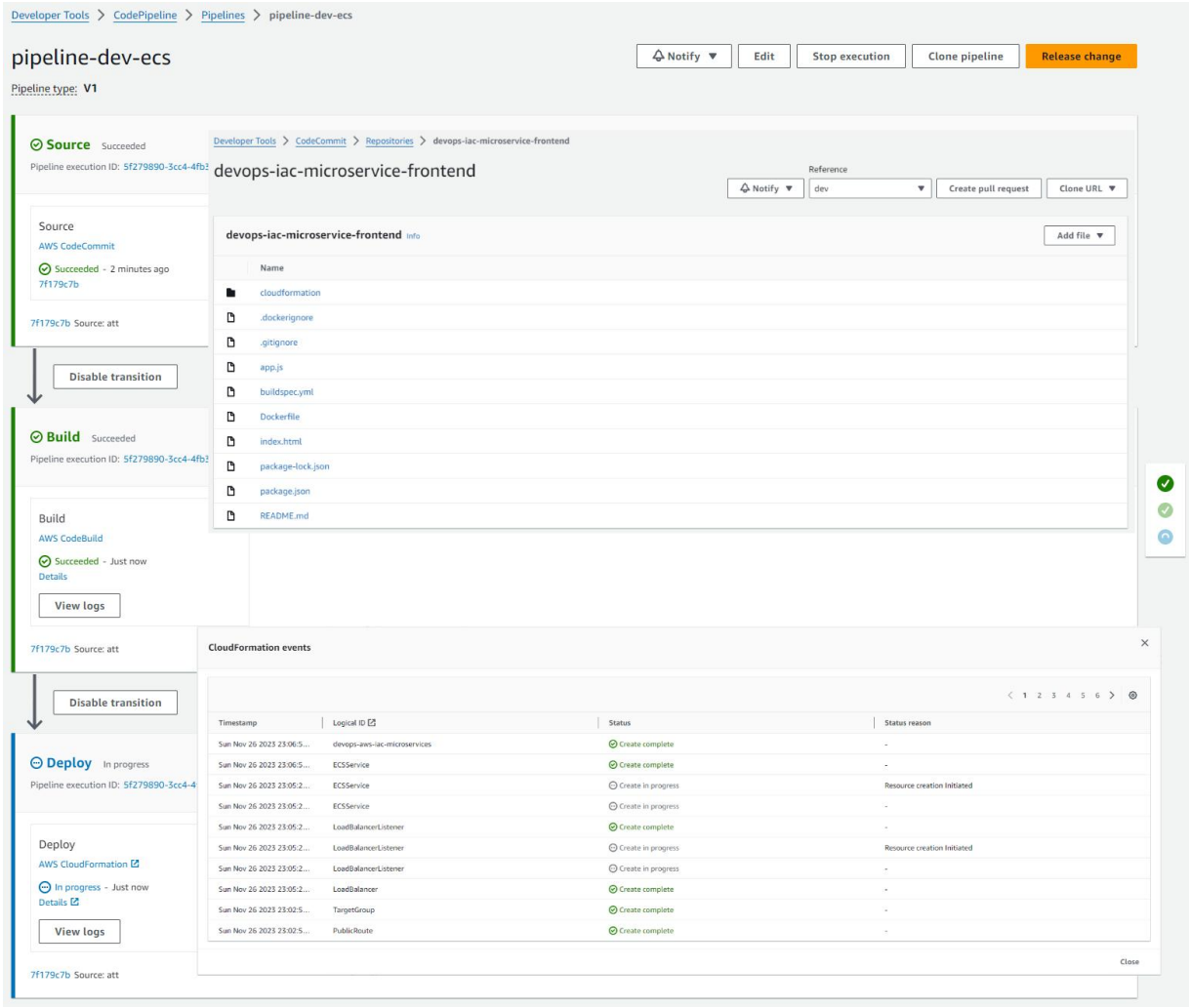
Figura 10 - Template CloudFormation Microsserviços parte 2

```yaml
ExecutionRole:
  Type: AWS::IAM::Role
  Properties:
    RoleName: !Join ['-', ['EcsTaskExecutionRole', !Ref Stage]]
    AssumeRolePolicyDocument:
      Statement:
        - Effect: Allow
          Principal:
            Service: ecs-tasks.amazonaws.com
          Action: 'sts:AssumeRole'
    ManagedPolicyArns:
      - 'arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy'

TaskDefinition:
  Type: AWS::ECS::TaskDefinition
  Properties:
    Memory: 1024
    Cpu: 512
    NetworkMode: awsvpc
    RequiresCompatibilities:
      - 'FARGATE'
    TaskRoleArn: !Ref ExecutionRole
    ExecutionRoleArn: !Ref ExecutionRole
    ContainerDefinitions:
      - Name: !Join ['-', ['container-frontend', !Ref Stage]]
        Image: !Ref ImageURI
        PortMappings:
          - ContainerPort: !Ref ContainerPort
            HostPort: !Ref ContainerPort

LoadBalancerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: !Join ['-', ['lb-sg', !Ref Stage]]
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: 80
        ToPort: 80
        CidrIp: 0.0.0.0/0

ContainerSecurityGroup:
  Type: AWS::EC2::SecurityGroup
  Properties:
    GroupDescription: !Join ['-', ['container-sg', !Ref Stage]]
    VpcId: !Ref VPC
    SecurityGroupIngress:
      - IpProtocol: tcp
        FromPort: !Ref ContainerPort
        ToPort: !Ref ContainerPort
        SourceSecurityGroupId: !Ref LoadBalancerSecurityGroup

LoadBalancer:
  Type: AWS::ElasticLoadBalancingV2::LoadBalancer
  Properties:
    IpAddressType: ipv4
    Name: !Join ['-', ['loadbalancer', !Ref Stage]]
    Scheme: internet-facing
    SecurityGroups:
      - !Ref LoadBalancerSecurityGroup
    Subnets:
      - !Ref SubnetA
      - !Ref SubnetB
    Type: application

TargetGroup:
  Type: AWS::ElasticLoadBalancingV2::TargetGroup
  Properties:
    Name: !Join ['-', ['targetgroup', !Ref Stage]]
    Port: 80
    Protocol: HTTP
    TargetType: ip
    VpcId: !Ref VPC

LoadBalancerListener:
  Type: AWS::ElasticLoadBalancingV2::Listener
  Properties:
    DefaultActions:
      - TargetGroupArn: !Ref TargetGroup
        Type: forward
    LoadBalancerArn: !Ref LoadBalancer
    Port: 80
    Protocol: HTTP

ECSService:
  Type: AWS::ECS::Service
  DependsOn: LoadBalancerListener
  Properties:
    ServiceName: !Join ['-', ['service-ecs', !Ref Stage]]
    Cluster: !Ref Cluster
    TaskDefinition: !Ref TaskDefinition
    DesiredCount: 2
    LaunchType: FARGATE
    NetworkConfiguration:
      AwsvpcConfiguration:
        AssignPublicIp: ENABLED
        Subnets:
          - !Ref SubnetA
          - !Ref SubnetB
        SecurityGroups:
          - !Ref ContainerSecurityGroup
    LoadBalancers:
      - ContainerName: !Join ['-', ['container-frontend', !Ref Stage]]
        ContainerPort: !Ref ContainerPort
        TargetGroupArn: !Ref TargetGroup
```

Fonte: Própria, 2023.

A implementação automatizada e orquestrada é iniciada por meio de um *commit* representativo no CodeCommit contendo código *frontend* em Node Express. Esse *commit* desencadeia a execução do CodePipeline, que, por sua vez, utiliza o CodeBuild para criar a imagem contida no repositório de código. Posteriormente, a implementação do microsserviço é realizada com o CodeDeploy. O resultado desse processo é uma infraestrutura robusta, escalável e segura para a execução eficiente dos microsserviços, aderindo às melhores práticas de arquitetura na nuvem. Os logs gerados durante a execução do pipeline proporcionam uma visão clara do progresso e permitem uma análise detalhada do processo.

Figura 11 - Execução do AWS CodePipeline



Fonte: Própria, 2023.

Por fim, podemos utilizar o DNS do AWS Elastic Load Balancer para acessar nosso microsserviço *frontend* e visualizar o resultado da aplicação em execução na web. Este endereço DNS é criado automaticamente no provisionamento do ELB e fornece uma maneira fácil de acessar a aplicação.

Figura 12 - Visualização do Microsserviço através do AWS ELB.

Fonte: Própria, 2023.

3.4 Resultados e Impactos obtidos com a implementação

Os resultados alcançados durante a transição gradual, partindo de uma criação manual, evidenciaram a escalabilidade e eficácia da automação. Isso ressaltou a maneira pela qual a Infraestrutura como Código acelera significativamente a adoção das práticas DevOps, contribuindo para um ambiente mais ágil, consistente e adaptável.

Quando as práticas DevOps não são utilizadas, as organizações enfrentam desafios como silos entre equipes de desenvolvimento e operações, atrasos na entrega de software, falta de visibilidade e dificuldades na detecção precoce de falhas. Por exemplo, sem a automação adequada, a implantação de novas funcionalidades pode ser propensa a erros manuais, levando a inconsistências e instabilidades no ambiente de produção. Em contrapartida, a implementação eficaz de práticas DevOps permite uma resposta rápida às mudanças de requisitos, maior confiabilidade nos lançamentos e, em última análise, resulta em um ciclo de vida de desenvolvimento mais ágil e eficiente.

O código utilizado para a implementação dos microsserviços está disponível em um repositório público no GitHub e pode ser acessado com a licença.

4 CONSIDERAÇÕES FINAIS

Fica evidenciado com esse trabalho que a adoção da cultura DevOps e a implementação de práticas modernas, como a Infraestrutura como Código, representam um passo significativo para a modernização e agilidade no desenvolvimento de software. A integração estreita entre desenvolvimento e operações, aliada à automação de processos, reflete diretamente na eficiência do ciclo de vida do desenvolvimento, reduzindo intervalos e promovendo entregas mais rápidas e confiáveis.

Ao explorar os benefícios oferecidos pela AWS, uma das principais plataformas de computação em nuvem, e enfatizar a importância da contratação de Engenheiros DevOps, o trabalho proporciona insights valiosos para quem busca otimizar seus processos operacionais e de desenvolvimento.

No contexto da rápida evolução tecnológica, a cultura DevOps emerge como uma abordagem fundamental para impulsionar a inovação e a competitividade. Ao final, o projeto destaca não apenas os desafios enfrentados na implementação, mas principalmente os ganhos substanciais em eficiência, qualidade e agilidade que as empresas podem alcançar ao abraçar integralmente os princípios e práticas DevOps na nuvem.

Considerando os resultados promissores da implementação de práticas DevOps e Infraestrutura como Código, abrem-se oportunidades para futuras melhorias e expansões. Uma possível implementação futura seria a incorporação de técnicas avançadas de monitoramento e análise de dados para aprimorar a detecção precoce de falhas e otimizar o desempenho do sistema. Além disso, explorar a integração de ferramentas de segurança DevSecOps poderia fortalecer a postura de segurança da infraestrutura. A continuidade da automação e a exploração de mais serviços gerenciados na nuvem, oferecidos pela AWS, também representam caminhos para aprimorar ainda mais a eficiência operacional e acelerar o desenvolvimento de novos recursos. A evolução contínua, baseada nas lições aprendidas nesta implementação, contribuirá para a construção de um ambiente de desenvolvimento e operações mais resiliente, ágil e alinhado com as demandas dinâmicas do cenário tecnológico atual.

Em conclusão, a demonstração prática da implementação desses conceitos na nuvem AWS ilustra de maneira tangível os benefícios da integração entre

desenvolvimento e operações, destacando a eficiência do ciclo de vida do desenvolvimento e a aceleração das entregas. Para aqueles que estão iniciando na área, este projeto serve como uma fonte inspiradora e educativa, proporcionando *insights* sobre as práticas modernas que impulsionam a inovação e a agilidade. Ao abraçar os princípios DevOps, os profissionais emergentes podem vislumbrar não apenas uma abordagem eficaz para enfrentar desafios operacionais, mas também uma mentalidade que promove a excelência contínua e adaptação às demandas dinâmicas do cenário tecnológico atual.

REFERÊNCIAS

AWS CodeBuild: Compile e teste código com escalabilidade contínua. Disponível em: <https://aws.amazon.com/pt/codebuild/>. Acesso em: 15 Abr. 2023.

AWS. Desenvolvimento e operações e AWS: Ferramentas e recursos de infraestrutura para profissionais de DevOps. Disponível em: <https://aws.amazon.com/pt/devops/>. Acesso em: 29 Mar. 2023.

AWS. O que significa integração contínua? Disponível em: <https://aws.amazon.com/pt/devops/continuous-integration/>. Acesso em: 15 Abr. 2023.

AWS. O que é o Amazon Elastic Container Registry? Disponível em: <https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html>. Acesso em: 15 Junho. 2023.

AWS. Recursos do Amazon CloudWatch. Disponível em: <https://aws.amazon.com/pt/cloudwatch/features/>. Acesso em: 15 Junho. 2023.

AWS. Recursos do AWS CloudFormation. Disponível em: <https://aws.amazon.com/pt/cloudformation/features/>. Acesso em: 20 Junho. 2023.

AWS. Recursos do AWS CodeDeploy. Disponível em: <https://aws.amazon.com/pt/codedeploy/features/?nc=sn&loc=2>. Acesso em: 15 Abr. 2023.

AWS. Recursos do AWS CodePipeline. Disponível em: <https://aws.amazon.com/pt/codepipeline/features/?nc=sn&loc=2>. Acesso em: 15 Abr. 2023.

AWS. Recursos do AWS CodeCommit. 2023. Disponível em: <https://aws.amazon.com/pt/codecommit/features/>. Acesso em: 07 Mai. 2023.

BARROS, L. C. A. Análise das plataformas para implementação da cultura devops: Uma visão geral dos principais prestadores de serviços em nuvem. Instituto Federal De Educação, Ciência e Tecnologia do Sertão Pernambuco, 2022. Trabalho de conclusão de curso.

BATAGINI, R. Como versionar utilizando o Git. Disponível em: <https://medium.com/biblioteca-dos-devs/como-versionar-utilizando-o-git-1f5d8fe2afcd>. Acesso em: 02 Mai. 2023.

BRANCO, M. S. Um estudo sobre devops. Universidade Estadual de Londrina, 2020. Trabalho de conclusão de curso.

BUCHANAN, I. Como a infraestrutura como código (IaC) gerencia infraestruturas complexas. Disponível em: <https://www.atlassian.com/br/microservices/cloud-computing/infrastructure-as-code>. Acesso em: 20 Junho. 2023.

BUFFA, L. H. Tornando acessível a cultura devops a pequenas empresas e startups. Pontífica Universidade Católica de Goiás, 2021. Trabalho de conclusão de curso.

FLEXERA. STATE OF THE CLOUD REPORT. [s.n.], 2021. 54 p. Disponível em: <https://lp.tufin.com/rs/769-ICF-145/images/report-cm-state-of-the-cloud-2021.pdf>.

FORSGREN, N.; HUMBLE, J.; KIM, G. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. [S.l.]: IT Revolution Press; 1st edition, 2018.

GAUR, B. How To Automate Code Deployment on AWS using CodePipeline: CodeCommit, CodeDeploy, CloudWatch. Disponível em: <https://k21academy.com/amazon-web-services/deploy-aws-codepipeline/>. Acesso em: 02 Mai. 2023.

HALL, T. Pipeline de DevOps. Disponível em: <https://www.atlassian.com/br/devops/devops-tools/devops-pipeline>. Acesso em: 15 Abr. 2023.

IBM. O que é a infraestrutura como código (IaC)? Disponível em: <https://www.ibm.com/br-pt/topics/infrastructure-as-code>. Acesso em: 15 Junho. 2023.

INFOSYS. CLOUD RADAR 2021. [s.n.], 2021. Disponível em: <https://www.infosys.com/services/cloud-cobalt/insights/cloud-radar-2021.html>.

LIMA, P. H. O. Estudo comparativo sobre as funcionalidades de integração contínua e entrega contínua das ferramentas jenkins e gitlab. Centro Universitário Christus Sistemas de InformaçãO, 2021.

LUCIO, J. P. D. Análise comparativa entre arquitetura monolítica e de microsserviços. Universidade Federal de Santa Catarina Departamento de Informática e Estatística, 2017. Trabalho de conclusão de curso.

MICROSOFT. Registros, imagens e containers do Docker. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/container-docker-introduction/docker-containers-images-registries>. Acesso em: 15 Junho. 2023.

PEREIRA, B. Observabilidade e monitoramento: o que são e como aplicá-los. Disponível em: <https://elven.works/observabilidade-e-monitoramento-o-que-sao-e-como-aplica-los/>. Acesso em: 15 Junho. 2023.