



# DEVOPS E INTEGRAÇÃO CONTÍNUA

AULA 6



Prof. Mauricio Antonio Ferste



## CONVERSA INICIAL

À medida que as organizações buscam aprimorar a eficiência, a agilidade e a confiabilidade de suas operações de desenvolvimento e infraestrutura, a adoção de práticas DevOps na nuvem tornou-se uma peça central para atingir esses objetivos. Neste cenário dinâmico, a Amazon Web Services (AWS) e o Terraform emergem como pilares fundamentais para a implementação bem-sucedida de DevOps, proporcionando um ambiente flexível, escalável e automatizado. Trata-se de uma cultura que promove a colaboração estreita entre essas duas disciplinas, com ênfase na automação, entrega contínua, integração contínua e monitoramento contínuo. A adoção de práticas DevOps visa acelerar o ciclo de vida de desenvolvimento e implementação, permitindo respostas rápidas às mudanças e melhorando a confiabilidade operacional.

A combinação de práticas DevOps com AWS e Terraform oferece uma série de benefícios. A automação proporciona ciclos de desenvolvimento mais rápidos, a orquestração eficiente da infraestrutura reduz erros operacionais, e a escalabilidade proporcionada pela nuvem AWS permite atender às demandas variáveis dos negócios. A cultura de colaboração e a visão unificada proporcionam maior visibilidade e transparência em todo o processo de desenvolvimento e operações.

Nesta jornada, exploraremos como a sinergia entre DevOps, AWS e Terraform possibilita a criação de ambientes de nuvem altamente eficientes, ágeis e confiáveis. Vamos mergulhar nas práticas, ferramentas e estratégias que impulsionam a transformação digital, capacitando as organizações a enfrentar os desafios modernos de forma inovadora e eficaz.

## TEMA 1 – DEVOPS E SERVIÇOS DE NUVEM

DevOps e serviços de nuvem são duas áreas interconectadas que desempenham papéis cruciais na modernização e eficiência das operações de TI. DevOps é uma cultura e conjunto de práticas que busca integrar o desenvolvimento de software e as operações, promovendo a colaboração e automação para acelerar a entrega de software. Isso envolve a implementação de pipelines de CI/CD, automação de infraestrutura e uma abordagem orientada por colaboração e agilidade. Por outro lado, os serviços de nuvem oferecem uma infraestrutura escalável e flexível para suportar as práticas DevOps. Plataformas



de nuvem, como AWS, Azure e Google Cloud, disponibilizam recursos como máquinas virtuais, armazenamento e serviços gerenciados, permitindo que as equipes de DevOps provisionem, automatizem e dimensionem ambientes de maneira eficiente. A nuvem também facilita a implementação de práticas como Infraestrutura como Código (IaC) e facilita a colaboração entre equipes distribuídas. Juntas, DevOps e serviços de nuvem capacitam as organizações a inovarem rapidamente, melhorar a qualidade do software, e responder de forma ágil às demandas do mercado, promovendo uma abordagem moderna e eficiente para o desenvolvimento e operação de sistemas de TI.

## 1.1 AWS

A Amazon Web Services (AWS) destaca-se como uma líder global no campo dos serviços de nuvem, fornecendo uma ampla gama de recursos para atender às diversas necessidades de computação em nuvem de organizações em todo o mundo. A infraestrutura oferecida pela AWS é conhecida por sua escalabilidade, confiabilidade e segurança, proporcionando um ambiente propício para a inovação e crescimento eficiente de empresas de todos os portes.

O serviço Amazon EC2 (Elastic Compute Cloud) é essencial para hospedar aplicativos e máquinas virtuais, oferecendo capacidade computacional escalável na nuvem. O Amazon S3 (Simple Storage Service) destaca-se como um serviço de armazenamento de objetos altamente durável e escalável, ideal para armazenar dados não estruturados, como imagens e vídeos.

A AWS Lambda introduz a ideia de computação serverless, permitindo a execução de código sem a necessidade de gerenciar servidores, escalando automaticamente conforme necessário. Para gerenciamento de bancos de dados relacionais, o Amazon RDS (Relational Database Service) fornece uma solução gerenciada que suporta diversos sistemas de banco de dados.

A Amazon VPC (Virtual Private Cloud) permite a criação de redes privadas virtuais isoladas na nuvem, proporcionando controle total sobre o ambiente de rede. O AWS Elastic Beanstalk simplifica o desenvolvimento e a implantação de aplicativos, gerenciando automaticamente a infraestrutura subjacente.

Para banco de dados NoSQL, o Amazon DynamoDB é altamente escalável e de baixa latência. A AWS CloudFormation permite a criação e gerenciamento de recursos da AWS usando código, seguindo a abordagem de



Infraestrutura como Código (IaC). O Amazon SNS (Simple Notification Service) oferece mensagens push para dispositivos e endpoints distribuídos, enquanto o AWS Glue facilita a preparação e a carga de dados entre data stores.

Em resumo, a AWS continua a ser a escolha principal para organizações que buscam uma variedade de serviços de nuvem confiáveis e inovadores. Seja para infraestrutura, armazenamento, banco de dados, computação serverless ou ferramentas de desenvolvimento, a AWS fornece um ecossistema robusto, capacitando as empresas a alcançarem seus objetivos na era da computação em nuvem.

## 1.2 Microsoft Azure

Microsoft Azure é uma plataforma de nuvem abrangente, oferecendo uma ampla variedade de serviços para atender às diversas necessidades de desenvolvimento, hospedagem e gerenciamento de aplicativos em nuvem. Seja para infraestrutura, soluções avançadas de inteligência artificial ou armazenamento de dados, o Azure fornece uma base sólida para empresas de todos os tamanhos e setores.

Um dos serviços principais do Azure é o Azure Virtual Machines (VMs), que proporciona máquinas virtuais escaláveis e flexíveis. Essas VMs podem ser configuradas com imagens pré-definidas ou personalizadas conforme as necessidades do usuário. Além disso, o Azure App Service é uma plataforma totalmente gerenciada, ideal para construir, implantar e escalar aplicativos web e móveis, suportando diversas linguagens de programação e integração contínua.

Para armazenamento de objetos não estruturados, o Azure Blob Storage oferece escalabilidade e eficiência, enquanto o Azure SQL Database fornece um serviço de banco de dados relacional totalmente gerenciado com alta disponibilidade e escalabilidade automática.

O Azure também se destaca em soluções para contêineres, com o Azure Kubernetes Service (AKS), que proporciona um ambiente gerenciado para implantação, gerenciamento e dimensionamento de contêineres utilizando Kubernetes.

Em termos de inteligência artificial, os Azure Cognitive Services oferecem um conjunto robusto de APIs e serviços, possibilitando a incorporação de recursos de visão, fala, linguagem natural e pesquisa em aplicativos. Além disso,



o Azure Active Directory (AD) garante autenticação segura e controle de acesso para aplicativos e recursos do Azure.

Para desenvolvimento contínuo e operações, o Azure DevOps oferece uma suíte completa de ferramentas e serviços, abrangendo todo o ciclo de vida de desenvolvimento e entrega. Adicionalmente, o Azure Virtual Network permite a criação de redes privadas e seguras na nuvem, incluindo opções para conexão segura com redes locais.

Com serviços especializados para Internet das Coisas (IoT), análise de big data (Azure Synapse Analytics) e uma série de outros recursos, o ecossistema do Azure proporciona a flexibilidade e a escalabilidade necessárias para apoiar a inovação e a transformação digital nas organizações que optam por migrar para a nuvem.

### 1.3 Outras tecnologias e nuvens

Além das plataformas mencionadas anteriormente, existem outras abordagens de nuvem que oferecem serviços e soluções para atender às diversas necessidades de empresas e desenvolvedores. Vamos explorar algumas dessas alternativas:

- Google Cloud Platform (GCP): a Google Cloud Platform é uma opção robusta para computação em nuvem, oferecendo uma variedade de serviços, desde computação e armazenamento até aprendizado de máquina e análise de dados. O GCP é conhecido por sua forte presença em soluções de dados e análise.
- IBM Cloud: a IBM Cloud oferece uma ampla gama de serviços, incluindo infraestrutura como serviço (IaaS), plataforma como serviço (PaaS) e software como serviço (SaaS). Destaca-se por suas soluções de nuvem híbrida, integração com inteligência artificial e blockchain.
- Oracle Cloud: a Oracle Cloud é focada em fornecer serviços de nuvem escaláveis, especialmente para aplicações corporativas e bancos de dados. Oferece uma variedade de serviços, incluindo computação, armazenamento, banco de dados, análise e soluções específicas da indústria.
- Alibaba Cloud: conhecida como a Alibaba Cloud ou Aliyun, é a plataforma de nuvem líder na China e uma das maiores em todo o mundo. Oferece



uma ampla gama de serviços, incluindo computação, armazenamento, banco de dados, análise de big data e inteligência artificial.

- DigitalOcean: a DigitalOcean é uma plataforma de nuvem simplificada, popular entre startups e desenvolvedores individuais. Ela se destaca pela simplicidade na implementação e gerenciamento de máquinas virtuais, além de serviços adicionais, como armazenamento e redes.
- VMware Cloud: a VMware oferece soluções de nuvem híbrida, permitindo que as empresas integrem seus ambientes de nuvem e locais. A VMware Cloud fornece uma infraestrutura consistente que abrange ambientes locais, nuvem privada e pública.
- Red Hat OpenShift: o Red Hat OpenShift é uma plataforma Kubernetes empresarial que simplifica o desenvolvimento, implantação e gerenciamento de aplicativos em contêineres. É uma escolha popular para empresas que buscam orquestração de contêineres em ambientes de nuvem.
- Heroku: Heroku é uma plataforma como serviço (PaaS) que simplifica o desenvolvimento e a implantação de aplicativos. É conhecida por sua abordagem amigável ao desenvolvedor e suporte a várias linguagens de programação.

Essas são apenas algumas das diversas opções disponíveis no mercado de computação em nuvem. Cada plataforma possui suas próprias características distintas, atendendo a diferentes casos de uso e requisitos específicos. A escolha da abordagem de nuvem dependerá das necessidades particulares de cada organização, considerando fatores como escalabilidade, segurança, custo e recursos oferecidos.

## **TEMA 2 – INFRAESTRUTURA COMO CÓDIGO COM TERRAFORM**

A Infraestrutura como Código (IaC) revoluciona a maneira como as organizações gerenciam e provisionam recursos de infraestrutura, e o Terraform emerge como uma ferramenta essencial nesse cenário dinâmico. Ao invés de configurações manuais e processos tradicionais, a IaC trata a infraestrutura como se fosse código de software, proporcionando vantagens como automação, consistência e escalabilidade. Terraform, uma plataforma líder em IaC, destaca-se pela sua simplicidade e eficácia. Com sua linguagem declarativa, os usuários



podem descrever a infraestrutura desejada em um formato legível e compreensível, permitindo a criação e gestão de ambientes complexos com facilidade. Esta introdução explorará como a combinação da filosofia da IaC com o Terraform possibilita a criação, modificação e versionamento de infraestrutura de maneira ágil, proporcionando uma base sólida para a implementação de práticas modernas de desenvolvimento e operações.

## 2.1 Infraestrutura como código

As empresas modernas operam em ambientes dinâmicos e em constante evolução, exigindo uma abordagem eficiente para gerenciar infraestrutura de maneira rápida, consistente e repetível. Uma solução para esse desafio é a adoção de ferramentas de Infrastructure as Code (IaC), que permitem gerenciar infraestrutura por meio de arquivos de configuração, em vez de interfaces gráficas.

### 2.1.1 Terraform

O Terraform, desenvolvido pela HashiCorp, é uma ferramenta de IaC, agnóstica a provedores de nuvem, que possibilita a definição de recursos e infraestrutura em arquivos de configuração legíveis por humanos. Ele gerencia o ciclo de vida da infraestrutura, proporcionando vantagens significativas sobre a gestão manual:

- Multiplataforma: o Terraform pode gerenciar infraestrutura em diversas plataformas de nuvem.
- Linguagem de configuração legível: sua linguagem declarativa de configuração facilita a rápida escrita de código de infraestrutura.
- Controle de versão: permite a commitagem de configurações em controle de versão para colaboração segura.
- Rastreamento de mudanças: utiliza um estado para rastrear alterações nos recursos durante as implantações.
- É agentless: não precisa de instalação de agentes para funcionar.
- Open: possui uma versão free e uma versão paga.



### 2.1.2 Como o Terraform Funciona

O Terraform interage com plataformas de nuvem e serviços por meio de plugins chamados providers. Existem mais de 1.000 providers disponíveis, abrangendo serviços como AWS, Azure, Google Cloud, Kubernetes, GitHub, Splunk e DataDog. Caso não encontre o provider desejado, é possível criar seu próprio.

A padronização do fluxo de trabalho de implantação é alcançada por meio de módulos, que são composições reutilizáveis de recursos provenientes de diferentes providers. A linguagem declarativa do Terraform descreve o estado desejado da infraestrutura, permitindo que os providers calculem automaticamente dependências entre recursos.

### 2.1.3 Fluxo de implantação do Terraform

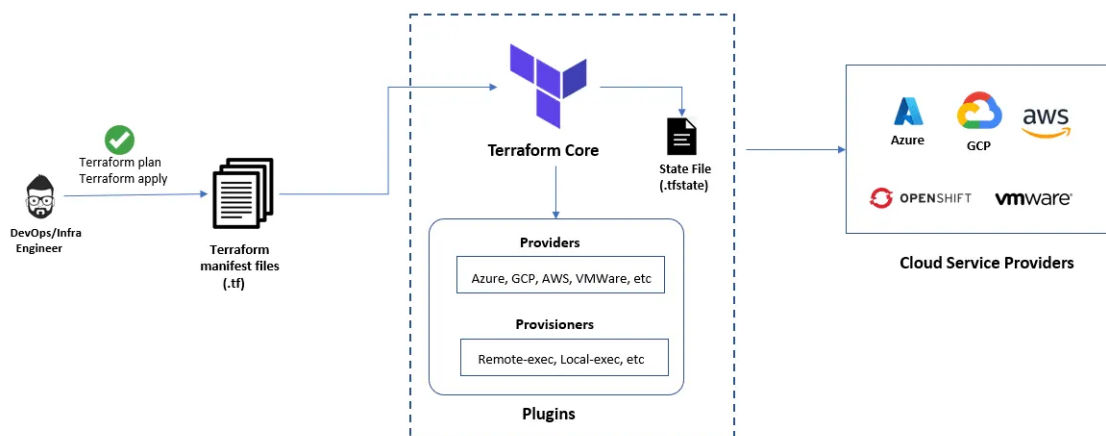
1. Escopo: identificar a infraestrutura necessária para o projeto;
2. Autoria: escrever a configuração da infraestrutura desejada;
3. Inicialização: instalar os plugins necessários para gerenciar a infraestrutura;
4. Planejamento: visualizar as alterações propostas pelo Terraform;
5. Aplicação: executar as mudanças planejadas.

### 2.1.4 Rastreamento da infraestrutura

O Terraform mantém um arquivo de estado que atua como fonte de verdade para o ambiente real. Esse arquivo é essencial para determinar as alterações necessárias na infraestrutura para corresponder à configuração desejada.



Figura 1 – Terraform Architecture



Fonte: Terraform, 2023.

### 2.1.5 Colaboração

O Terraform facilita a colaboração ao permitir o uso de backends de estado remotos. Ao utilizar o Terraform Cloud (gratuito para até cinco usuários), é possível compartilhar o estado de maneira segura, fornecer um ambiente estável para a execução do Terraform e evitar condições de corrida ao realizar alterações simultâneas.

Conectar o Terraform Cloud a sistemas de controle de versão como GitHub e GitLab automatiza propostas de alterações quando configurações são commitadas. Isso possibilita o gerenciamento de mudanças na infraestrutura por meio de controle de versão, similar à gestão de código de aplicativos.

### 2.1.6 Exemplo prático

Vamos criar um exemplo prático de uso do Terraform para provisionar uma instância EC2 (Elastic Compute Cloud) na AWS. Certifique-se de ter o Terraform instalado em sua máquina antes de começar.

### 2.1.7 Configurar suas credenciais AWS

Antes de começar, certifique-se de ter suas credenciais AWS configuradas. Você pode configurá-las executando o comando `aws configure` em seu terminal e inserindo suas informações quando solicitado.



## 2.1.8 Crie um diretório de trabalho

Crie um diretório para o seu projeto Terraform:

```
mkdir terraform-aws-example
cd terraform-aws-example
```

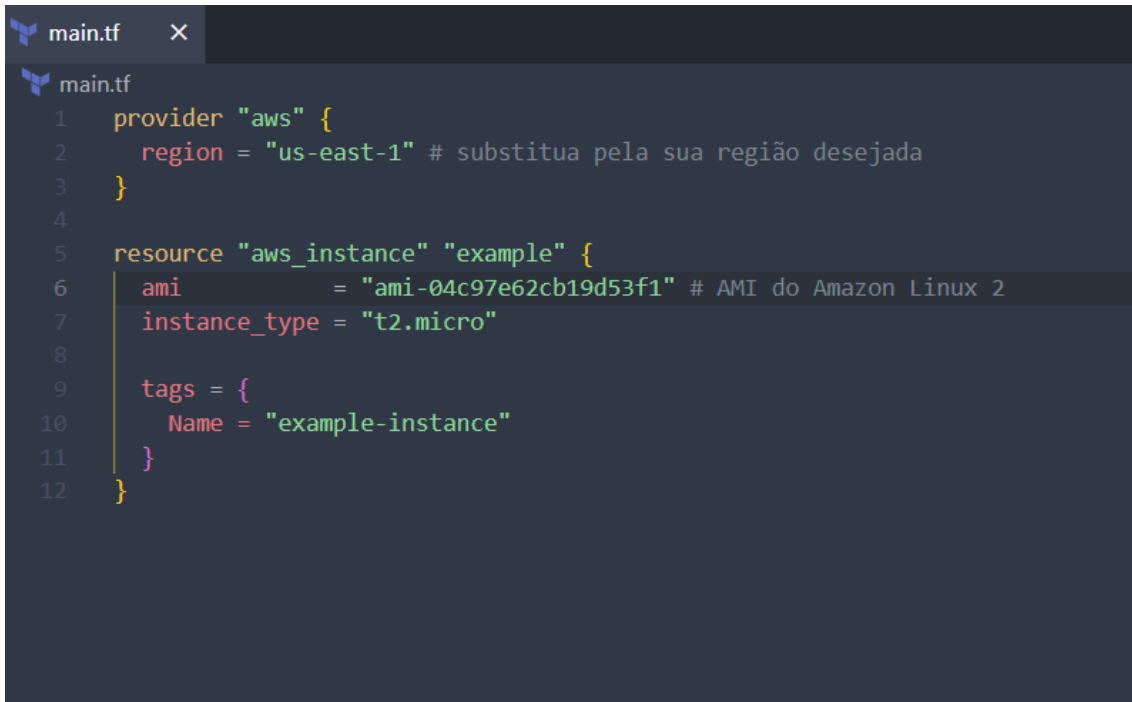
## 2.1.9 Crie um arquivo de configuração do Terraform

Crie um arquivo chamado main.tf no diretório que você acabou de criar e adicione o seguinte código:

```
provider "aws" {
  region = "us-east-1" # substitua pela sua região desejada
}
resource "aws_instance" "example" {
  ami          = "ami-0c55b159cbf1f0" # AMI do Amazon Linux
  instance_type = "t2.micro"
  tags = {
    Name = "example-instance"
  }
}
```

Este código define um provedor AWS e uma instância EC2. Certifique-se de substituir a região e a AMI conforme necessário. Existe uma galeria de AMIs disponível na AWS.

Figura 2 – Arquivo principal de configuração AWS

A screenshot of a code editor window titled 'main.tf'. The editor shows the following Terraform configuration code:

```
1 provider "aws" {
2   region = "us-east-1" # substitua pela sua região desejada
3 }
4
5 resource "aws_instance" "example" {
6   ami           = "ami-04c97e62cb19d53f1" # AMI do Amazon Linux 2
7   instance_type = "t2.micro"
8
9   tags = {
10     Name = "example-instance"
11   }
12 }
```

Fonte: AWS, 2023.

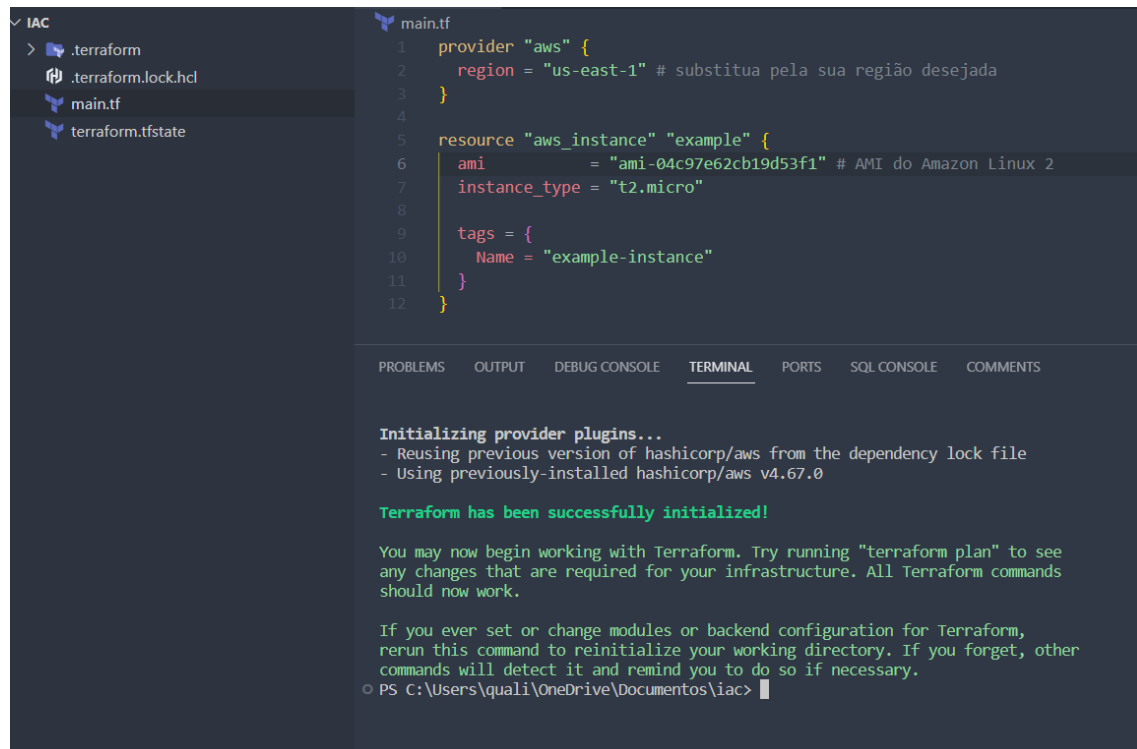
### 2.1.10 Inicialize e aplique as configurações

No mesmo diretório do arquivo main.tf, execute os seguintes comandos:

```
terraform init
```

Este comando inicializa o diretório de trabalho, baixando os plugins necessários.

Figura 3 – Configuração AWS



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory named 'IAC' containing files: '.terraform', '.terraform.lock.hcl', 'main.tf', and 'terraform.tfstate'. The code editor shows the content of 'main.tf' with the following Terraform configuration:

```
1 provider "aws" {
2   region = "us-east-1" # substitua pela sua região desejada
3 }
4
5 resource "aws_instance" "example" {
6   ami           = "ami-04c97e62cb19d53f1" # AMI do Amazon Linux 2
7   instance_type = "t2.micro"
8
9   tags = {
10     Name = "example-instance"
11   }
12 }
```

Below the code editor, there is a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE COMMENTS

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v4.67.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
PS C:\Users\quali\OneDrive\Documents\iac>
```

Fonte: AWS, 2023.

### 2.1.11 Resultado do comando

Este comando mostra o planejamento para o provisionamento que será feito através do Terraform.

```
terraform plan
```

Figura 4 – Configuração Terraform

```
+ enable_resource_name_dns_a_record    = (known after apply)
+ enable_resource_name_dns_aaaa_record = (known after apply)
+ hostname_type                        = (known after apply)
}

+ root_block_device {
  + delete_on_termination = (known after apply)
  + device_name           = (known after apply)
  + encrypted             = (known after apply)
  + iops                  = (known after apply)
  + kms_key_id            = (known after apply)
  + tags                  = (known after apply)
  + throughput            = (known after apply)
  + volume_id             = (known after apply)
  + volume_size           = (known after apply)
  + volume_type           = (known after apply)
}
```

**Plan:** 1 to add, 0 to change, 0 to destroy.

Fonte: Terraform, 2023.

### 2.1.12 Aplicando o planejamento

```
terraform apply
```

Este comando aplica as configurações definidas no arquivo main.tf. Será necessário confirmar a ação digitando “yes” quando solicitado.



Figura 5 – Instância sendo criada

```
+ encrypted      = (known after apply)
+ iops           = (known after apply)
+ kms_key_id     = (known after apply)
+ tags           = (known after apply)
+ throughput     = (known after apply)
+ volume_id      = (known after apply)
+ volume_size    = (known after apply)
+ volume_type    = (known after apply)
}
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.

Enter a value: yes

aws\_instance.example: Creating...  
aws\_instance.example: Still creating... [10s elapsed]  
aws\_instance.example: Creation complete after 15s [id=i-01f23e3f339352463]

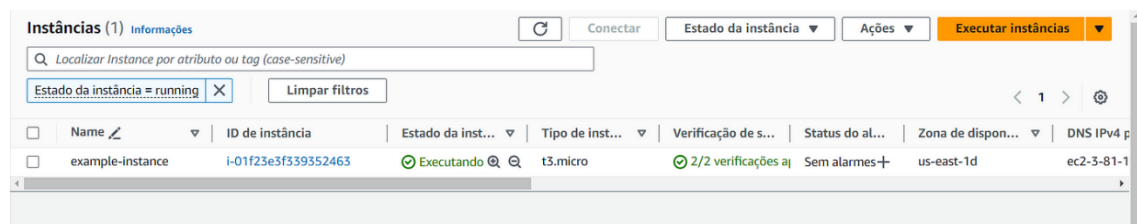
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Fonte: AWS, 2023.

### 2.1.13 Verifique a instância criada na AWS

Acesse o console da AWS ou utilize a CLI da AWS para verificar se a instância EC2 foi criada com sucesso.

Figura 6 – Servidor mostrando as instâncias AWS rodando



Fonte: AWS, 2023.

Console EC2 com a instância que acabamos de criar.

### 2.1.14 Destrua os recursos criados

Depois de verificar que tudo está funcionando conforme esperado, você pode destruir os recursos para evitar custos contínuos. No mesmo diretório, execute:

```
terraform destroy
```

Este comando removerá todos os recursos que foram criados.

Figura 8 – Verificando a destruição ou remoção das máquinas

```
Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

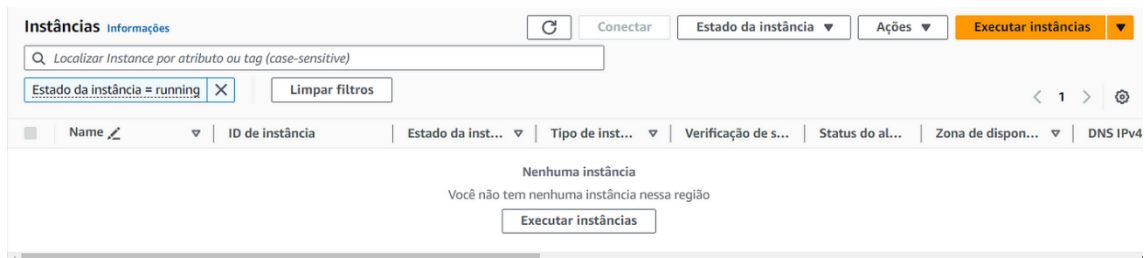
aws_instance.example: Destroying... [id=i-01f23e3f339352463]
aws_instance.example: Still destroying... [id=i-01f23e3f339352463, 10s elapsed]
aws_instance.example: Still destroying... [id=i-01f23e3f339352463, 20s elapsed]
aws_instance.example: Still destroying... [id=i-01f23e3f339352463, 30s elapsed]
aws_instance.example: Still destroying... [id=i-01f23e3f339352463, 40s elapsed]
aws_instance.example: Still destroying... [id=i-01f23e3f339352463, 50s elapsed]
aws_instance.example: Destruction complete after 52s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\quali\OneDrive\Documentos\iac>
```

Fonte: AWS, 2023.

### 2.1.15 Verificando se a instância foi destruída

Figura 9 – Verificando se a instância foi destruída



Fonte: AWS, 2023.

Este é apenas um exemplo simples para começar. O Terraform oferece recursos poderosos para gerenciar infraestrutura, e você pode expandir e personalizar conforme necessário para atender aos requisitos específicos do seu projeto. Certifique-se de revisar a documentação oficial do Terraform para AWS para explorar mais recursos e opções disponíveis.



### 2.1.16 Próximos passos

Agora que você está familiarizado com os conceitos fundamentais do Infrastructure as Code e do Terraform, está pronto para escrever suas próprias configurações de infraestrutura.

### 2.1.17 Conclusão

A adoção do Terraform para automação de infraestrutura proporciona uma abordagem eficiente para enfrentar os desafios de ambientes dinâmicos e em constante evolução. Ao interagir com diversas plataformas de nuvem através de plugins chamados providers, o Terraform utiliza uma abordagem declarativa para descrever o estado desejado da infraestrutura, permitindo a automática gestão de dependências entre recursos. A colaboração é facilitada pelo uso de backends de estado remotos, como o Terraform Cloud, proporcionando um ambiente estável para execução e prevenindo condições de concorrência durante alterações simultâneas. Ao familiarizar-se com os conceitos fundamentais de Infrastructure as Code e Terraform, você está preparado para escrever suas próprias configurações de infraestrutura, embarcando em uma jornada de automação eficiente e consistente.

## TEMA 3 – ESTRATÉGIAS DE ESCALABILIDADE E GRUPOS DE DIMENSIONAMENTO AUTOMÁTICO

As estratégias de escalabilidade e os grupos de dimensionamento automático desempenham papéis cruciais na eficiência operacional de sistemas distribuídos em ambientes de nuvem. A escalabilidade, tanto horizontal quanto vertical, é essencial para atender às demandas flutuantes de tráfego, permitindo que os recursos se ajustem dinamicamente às necessidades do sistema. Os grupos de dimensionamento automático, por sua vez, são mecanismos inteligentes que automatizam o processo de provisionamento e desprovisionamento de recursos em resposta a variações na carga de trabalho. Ao compreender e implementar estratégias eficazes de escalabilidade, juntamente com grupos de dimensionamento automático, as organizações podem otimizar a utilização de recursos, garantir alta disponibilidade e manter um ambiente operacional ágil e resiliente.





### 3.1 Escalabilidade

A escalabilidade em nuvem é um conceito central que se refere à capacidade de um sistema, aplicativo ou infraestrutura se ajustar dinamicamente para lidar com variações na carga de trabalho. Na computação em nuvem, a escalabilidade é fundamental para garantir que os recursos computacionais possam ser expandidos ou reduzidos conforme necessário, proporcionando flexibilidade para enfrentar demandas flutuantes. Essa capacidade de dimensionamento elástico permite otimizar o desempenho, assegurando que os recursos estejam alinhados de maneira eficiente com a demanda em constante mudança. A escalabilidade em nuvem é essencial para lidar com cenários dinâmicos, garantindo eficiência operacional, economia de custos e disponibilidade consistente dos serviços oferecidos na infraestrutura em nuvem.

### 3.2 O que é Auto Scaling?

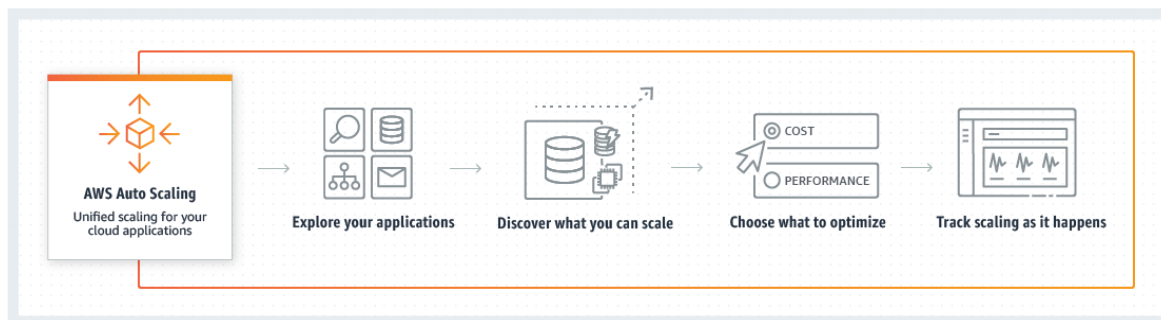
Os Grupos de Dimensionamento Automático (ASGs) da AWS representam uma solução eficaz para a rápida gestão e escalabilidade de um conjunto homogêneo de instâncias do EC2, compartilhando a mesma configuração. Desenvolvidos para ajustar automaticamente o número de instâncias em resposta a alterações na demanda ou conforme políticas predefinidas, os ASGs asseguram a manutenção constante do número desejado de instâncias em operação. Essa automação é fundamental para preservar a disponibilidade de aplicativos e lidar de maneira eficiente com flutuações nas cargas de trabalho.

O funcionamento dos Grupos de Dimensionamento Automático é orientado por políticas específicas de dimensionamento, que estipulam as condições para incrementar ou reduzir o grupo. Essas condições podem ser baseadas em métricas como a utilização da CPU, o tráfego de rede ou outras medidas personalizadas. Por exemplo, é possível criar uma política que aumente o número de instâncias quando a utilização da CPU atingir 80%, garantindo recursos adequados para lidar com a demanda do aplicativo.

Para implementar Grupos de Dimensionamento Automático de forma eficaz, é imperativo compreender detalhadamente os requisitos de dimensionamento do aplicativo. Isso envolve a determinação do número de instâncias necessário para atender à demanda média e a adaptação do número

de instâncias para acomodar picos de demanda. A monitorização do desempenho do aplicativo em testes de carga desempenha um papel crucial nesse processo, permitindo a definição precisa das métricas utilizadas nas políticas de dimensionamento.

Figura 9 – Esquema genérico de máquinas da Amazon



Fonte: Amazon, 2023.

Tanto a AWS quanto o Terraform oferecem recursos robustos para implementação eficiente. Com a AWS, o Auto Scaling permite ajustar dinamicamente a capacidade dos recursos computacionais, como instâncias EC2, para atender às variações na demanda de aplicativos. Utilizando políticas de Auto Scaling, é possível definir condições específicas para escalar automaticamente os recursos para cima ou para baixo com base em métricas como uso de CPU, tráfego de rede ou latência.

O Terraform, por sua vez, oferece suporte nativo para a configuração de grupos de Auto Scaling, permitindo que você defina e gerencie políticas de escalabilidade como código. Isso significa que você pode especificar o número mínimo e máximo de instâncias, bem como as métricas de monitoramento a serem usadas para escalar automaticamente o grupo. Ao combinar a AWS e o Terraform, é possível implementar uma infraestrutura escalável e resiliente, capaz de lidar com flutuações na carga de trabalho de forma automática e eficiente.

## TEMA 4 – ARQUITETURAS SERVERLESS E FAAS (FUNÇÕES COMO SERVIÇO)

Estamos vivendo o período da Indústria 4.0, em um cenário tecnológico em constante evolução. As arquiteturas serverless e FaaS (Funções como Serviço) emergem como paradigmas inovadores, redefinindo a maneira como desenvolvemos e implementamos serviços na era da computação em nuvem.



Contrariando a abordagem tradicional de provisionamento constante de recursos, as arquiteturas serverless propõem uma mudança fundamental ao eliminar a necessidade de gerenciamento direto da infraestrutura.

Nesse contexto, o modelo FaaS se destaca como uma implementação específica, permitindo a execução de código sob demanda, sem a preocupação com a manutenção de servidores subjugados. Esta introdução explorará os princípios centrais por trás das arquiteturas serverless e do paradigma FaaS. Abordaremos como essas abordagens revolucionam a eficiência, escalabilidade e agilidade no desenvolvimento de aplicações, destacando o potencial de otimização de custos e a capacidade de se adaptar dinamicamente às necessidades variáveis de carga de trabalho. Vamos adentrar no universo serverless, onde as preocupações com infraestrutura cedem lugar à execução de funções independentes, desencadeando uma nova era na arquitetura de software.

#### **4.1 Arquiteturas SERVERLESS**

As arquiteturas representam uma revolução no paradigma de desenvolvimento de software, introduzindo uma abordagem que elimina a necessidade de gerenciamento direto de servidores por parte dos desenvolvedores. Contrariando o modelo convencional, no qual os recursos são provisionados de forma contínua, o serverless adota uma perspectiva onde a infraestrutura é completamente gerenciada pelos provedores de nuvem.

Na essência dessa arquitetura, as aplicações são construídas em torno de eventos e funções. As funções, unidades independentes de execução de código, são acionadas sob demanda em resposta a eventos específicos, como solicitações HTTP, atualizações de banco de dados ou até mesmo eventos de terceiros. Essa abordagem modular e orientada por eventos permite que os desenvolvedores se concentrem exclusivamente na lógica de negócios, sem se preocupar com a infraestrutura subjacente.

Os benefícios das arquiteturas serverless incluem escalabilidade automática, otimização de custos, alta disponibilidade e uma abordagem pay-as-you-go, onde os desenvolvedores pagam apenas pelo tempo de execução efetiva das funções. No entanto, é crucial entender as limitações e considerar cuidadosamente os casos de uso ideais para esse modelo, pois nem todas as aplicações se beneficiam igualmente do paradigma serverless. Com sua



capacidade de simplificar o desenvolvimento e acelerar a entrega de software, as arquiteturas serverless representam uma evolução significativa na forma como concebemos e implementamos soluções na nuvem.

## **4.2 FAAS (Funções como Serviço)**

FaaS, ou Funções como Serviço, é um componente essencial das arquiteturas serverless, proporcionando uma abordagem inovadora para o desenvolvimento e execução de código na nuvem. Nesse modelo, os desenvolvedores dividem suas aplicações em funções independentes e granulares, que são acionadas em resposta a eventos específicos, sem a necessidade de gerenciar a infraestrutura subjacente.

A característica distintiva do FaaS é sua capacidade de executar código sob demanda. Cada função é autossuficiente e encapsula uma única tarefa ou operação, permitindo que seja executada de maneira independente em resposta a eventos, como requisições HTTP, atualizações de banco de dados ou notificações externas. Isso não apenas simplifica o desenvolvimento, permitindo que os desenvolvedores foquem em unidades modulares de lógica de negócios, mas também oferece benefícios notáveis em termos de escalabilidade automática e eficiência de custos.

Ao adotar o FaaS, as organizações podem desfrutar de uma execução altamente eficiente, pagando apenas pelo tempo de execução efetiva de cada função. A escalabilidade automática permite que a infraestrutura se ajuste dinamicamente à carga de trabalho, proporcionando respostas rápidas a flutuações na demanda. No entanto, é importante considerar cuidadosamente os casos de uso ideais para FaaS, pois nem todas as aplicações se beneficiam igualmente dessa abordagem. Com a capacidade de acelerar o desenvolvimento, melhorar a eficiência operacional e promover uma arquitetura modular, o FaaS emerge como uma peça fundamental no arsenal de ferramentas para construir soluções escaláveis e ágeis na era serverless.

## **TEMA 5 – TENDÊNCIAS E FUTURO DO DEVOPS**

As tendências e o futuro do DevOps apontam para uma evolução contínua na integração e colaboração entre desenvolvimento e operações, visando aprimorar a entrega de software. Automação contínua, práticas de DevSecOps



integrando segurança ao longo do ciclo de vida do desenvolvimento, e a expansão do conceito de DevOps para além do desenvolvimento e operações tradicionais são pontos chave. Além disso, a crescente adoção de tecnologias como contêineres e orquestradores, juntamente com a ênfase em cultura, colaboração e feedback contínuo, moldam o futuro do DevOps. A escalabilidade, flexibilidade e a busca por abordagens mais ágeis e eficientes indicam uma trajetória na qual o DevOps continua a ser uma peça central na entrega de software, promovendo inovação e melhorias contínuas na forma como as equipes desenvolvem, implantam e mantêm sistemas de software.

## 5.1 Cenário DevOps

O cenário de DevOps continua a evoluir em resposta às demandas crescentes por agilidade, automação e colaboração contínua entre equipes de desenvolvimento e operações. Algumas tendências emergentes delineiam o futuro do DevOps, destacando áreas de foco que moldarão a forma como as organizações concebem, implementam e gerenciam seus processos de desenvolvimento e operações.

DevSecOps e Segurança Integrada ganham destaque como a integração da segurança em todo o ciclo de vida do desenvolvimento. A segurança é incorporada desde o início do processo de desenvolvimento, com ferramentas e práticas que garantem a identificação e correção contínuas de vulnerabilidades.

Automatização Cognitiva e IA tornam-se fundamentais, impulsionando a automação com inteligência artificial (IA) e aprendizado de máquina (ML). Isso proporciona insights avançados, detecção de padrões e automação inteligente para otimizar processos complexos.

GitOps e Infraestrutura como Código (IaC) estão mais difundidos. A abordagem GitOps, baseada em práticas de versionamento e controle de código-fonte, destaca-se. A Infraestrutura como Código continua a ser uma peça-chave, permitindo a automação de provisionamento e gerenciamento de infraestrutura.

Observabilidade e Monitoramento Contínuo crescem em ênfase, com organizações buscando ferramentas avançadas de monitoramento e análise de logs. Isso fornece insights precisos sobre o desempenho do sistema, diagnósticos rápidos e tomada de decisões informada.



Edge Computing e DevOps Distribuído se destacam com a ascensão da computação de borda. As práticas de DevOps precisam se adaptar a ambientes distribuídos, e o gerenciamento eficiente de operações em locais remotos torna-se uma prioridade.

Continuous Testing e Testes Autônomos evoluem, incorporando testes autônomos que utilizam automação avançada e inteligência artificial. Isso visa aumentar a cobertura de testes e reduzir a dependência de intervenção humana. Containers e Orquestração permanecem como uma tendência sólida, com a utilização generalizada de contêineres e orquestradores, como Kubernetes. A gestão eficiente de ambientes containerizados continua a ser uma prioridade para equipes de DevOps.

SRE (Site Reliability Engineering) ganha popularidade como uma extensão natural do DevOps. A adoção de práticas do SRE, que combina aspectos de desenvolvimento e operações para garantir confiabilidade e desempenho, destaca-se como uma tendência significativa. Estas tendências refletem a constante busca por inovação e eficiência no cenário de DevOps, moldando um futuro em que a colaboração, automação e segurança são elementos intrínsecos em todo o ciclo de vida do desenvolvimento de software.

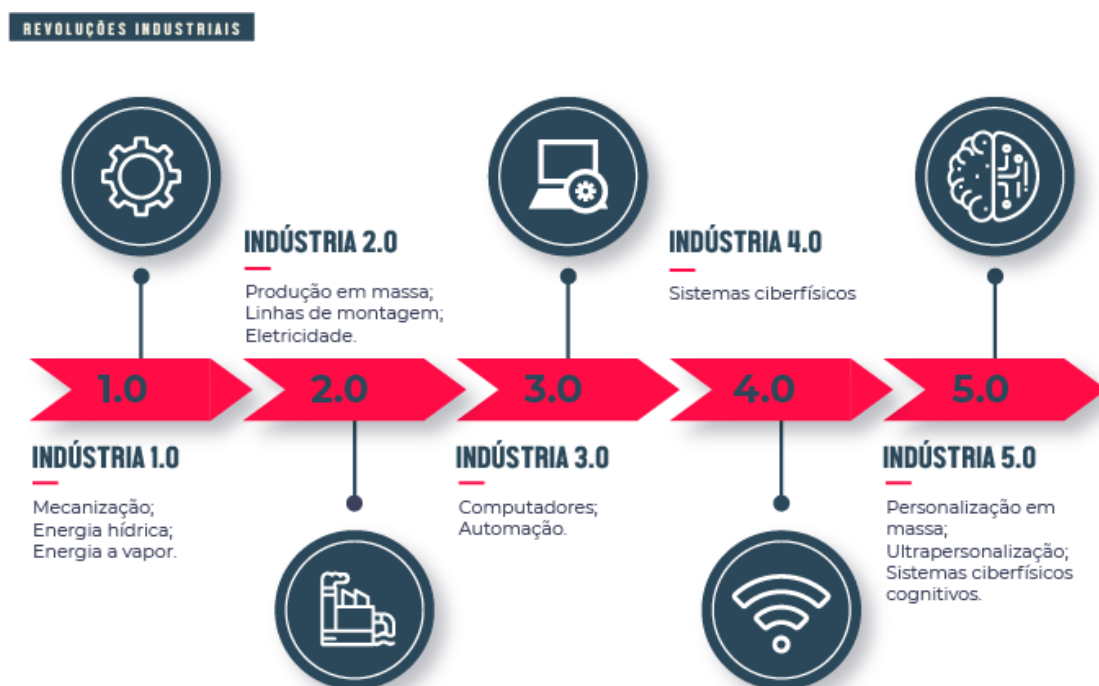
## **5.2 Pense no cenário de Indústria 5.0**

O futurismo de DevOps e a Indústria 5.0 representam uma visão empolgante e inovadora para o desenvolvimento de software e as práticas industriais. Enquanto o DevOps continua a evoluir, espera-se uma integração ainda maior entre desenvolvimento, operações e segurança, dando origem ao conceito de DevSecOps. Nesse cenário, a automação, a cultura de colaboração e a implementação de práticas ágeis se tornarão ainda mais essenciais para impulsionar a eficiência e a qualidade na entrega de software. Além disso, espera-se que o DevOps se expanda para além do desenvolvimento de software tradicional, abraçando novas áreas como a automação de processos empresariais e a gestão de infraestrutura de forma mais abrangente.

Por outro lado, a Indústria 5.0 representa uma revolução na manufatura, impulsionada pela convergência entre automação avançada, inteligência artificial e colaboração homem-máquina. Nesse contexto, o DevOps desempenha um papel crucial na implementação e gerenciamento de sistemas de controle e automação industrial. A integração de práticas DevOps na Indústria 5.0 promove

uma abordagem mais ágil e eficiente para o desenvolvimento e implementação de sistemas de produção, permitindo adaptações rápidas e contínuas às demandas do mercado e às mudanças nas condições de operação.

Figura 10 – Reflexão sobre a indústria 5.0



Fonte: Furniturk Magazine Online, 2018.

Em resumo, o futurismo de DevOps e a Indústria 5.0 prometem uma sinergia poderosa, onde a agilidade, a automação e a colaboração são fundamentais para impulsionar a inovação e o progresso tanto no desenvolvimento de software quanto na manufatura industrial. Essa convergência entre tecnologia e processos industriais representa uma nova era de eficiência, flexibilidade e qualidade, onde o DevOps desempenha um papel central na transformação digital e na busca por excelência operacional.

## FINALIZANDO

No encerramento deste capítulo abrangente sobre DevOps e serviços de nuvem, é evidente que a convergência desses elementos desempenha um papel crucial na modernização e eficiência operacional de sistemas de software.

A abordagem de Infraestrutura como Código, exemplificada pelo uso do Terraform, proporciona uma base sólida para a automação e gerenciamento eficiente de recursos na nuvem. A exploração de estratégias de escalabilidade,





como os Auto-Scaling Groups, destaca a importância de ajustar dinamicamente a infraestrutura para atender às demandas flutuantes.

Além disso, a introdução das arquiteturas Serverless e FaaS evidencia uma mudança paradigmática, possibilitando o desenvolvimento de aplicações altamente escaláveis sem a necessidade de gerenciar servidores.

Ao olharmos para o futuro do DevOps, percebemos que a integração contínua, práticas de segurança incorporadas e a adoção de tecnologias emergentes delineiam uma trajetória na qual a colaboração entre desenvolvimento e operações continua a evoluir, proporcionando inovação contínua e eficiência no ciclo de vida do desenvolvimento de software. Estas tendências, impulsionadas pela busca constante por agilidade e qualidade, consolidam o DevOps como uma abordagem indispensável na era da computação em nuvem.



## REFERÊNCIAS

- CAELUM. **Test-Driven Development**. Disponível em: <<http://tdd.caelum.com.br>>. Acesso em: 17 abr. 2024.
- CARMENA, R. M. **How to teach Git**. Disponível em: <<https://rachelcarmena.github.io/2018/12/12/how-to-teach-git.html>>. Acesso em: 17 abr. 2024.
- CHACON, S.; STRAUB, B. **Pro Git**. 2. Ed. Apress, 2014.
- CUKIER, D. **DDD: Introdução a Domain Driven Design**. 2010. Disponível em: <<http://www.agileandart.com/2010/07/16/ddd-introducao-a-domain-driven-design>>. Acesso em: 17 abr. 2024.
- DEVMEDIA. **TDD: fundamentos do desenvolvimento orientado a testes**. Disponível em: <<http://www.devmedia.com.br/tdd-fundamentos-do-desenvolvimento-orientado-a-testes/28151>>. Acesso em: 17 abr. 2024.
- \_\_\_\_\_. **Test Driven Development: TDD Simples e Prático**. 2019. Disponível em: <<https://www.devmedia.com.br/test-driven-development-tdd-simples-e-pratico/18533>>. Acesso em: 17 abr. 2024.
- DHADUK, H. **How Netflix Became A Master of DevOps? An Exclusive Case Study**. Disponível em: <<https://www.simform.com/blog/netflix-devops-case-study>>. Acesso em: 17 abr. 2024.
- ELEMARJR. **BDD na prática – parte 1 – Conceitos básicos e algum código**. 2012. Disponível em: <<https://elemarjr.wordpress.com/2012/04/11/bdd-na-prtica-parte-1-conceitos-bsicos-e-algum-cdigo>>. Acesso em: 17 abr. 2024.
- ENAP – Escola Nacional de Administração Pública. **Curso O papel do DevOps na Transformação Digital dos Serviços Públicos**. 2020.
- FREECODECAMP. Disponível em: <<https://www.freecodecamp.org>>. Acesso em: 17 abr. 2024.
- FREEMAN, S.; PRYCE, N. **Desenvolvimento de software orientado a objetos, guiado por testes**. Rio de Janeiro: Alta Books, 2012.
- GASPARETO, O. **Test Driven Development**. Universidade Federal do Rio Grande do Sul. Disponível em: <<http://www.inf.ufrgs.br/~cesantin/TDD-Otavio.pdf>>. Acesso em: 17 abr. 2024.



GIT-FAST-VERSION-CONTROL. Disponível em: <<https://git-scm.com>>. Acesso em: 17 abr. 2024.

GOMES, A. F. **Desenvolvimento Ágil com Kanban**. Disponível em: <[http://www.devmedia.com.br/websys.5/webreader.aspat=6&artigo=2955&revisita=javamagazine\\_84#a-2955](http://www.devmedia.com.br/websys.5/webreader.aspat=6&artigo=2955&revisita=javamagazine_84#a-2955)>. Acesso em: 17 abr. 2024.

HORNBEEK, M. **Engeneering DevOps**. 2019.

KIM, G. et al. **The DevOps Handbook**: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. [S. l.: s. n.], 2016.

MARTIN FOWLER. Disponível em: <<https://martinfowler.com>>. Acesso em: 17 abr. 2024.

PRESSMAN, R. S.; MAXIM, B. **Software Engineering**: a practitioner's approach. 8. ed. McGraw-Hill, 2014.

ROSA, D. **6 passos para iniciar no DevOps**. 2018. Disponível em: <<https://jornadaparanuvem.com.br/6-passos-para-iniciar-na-jornada-do-devops>>. Acesso em: 17 abr. 2024.

W3C. Disponível em: <<https://www.w3schools.in/mvc-architecture>>. Acesso em: 17 abr. 2024.

WESLEY, P. **Sistemas de controle de versão - SVN e Git**. 2017.

WOLFF, E. **Microservices Flexible Software Architectures**. Leanpub, 2016.