MBA EM **ENGENHARIA DE SOFTWARE**

Infra As Code (IAC)

Professor Leonardo Lima

MBA USP ESALQ

A responsabilidade pela idoneidade, originalidade
e licitude dos conteúdos didáticos apresentados é do professor.

MBA USP ESALQ

Apresentação - Acadêmica

- PUC Minas - Especialização, Gestão em Engenharia (2025)
- IPT/USP - Mestrado em Engenharia de Computação (2018)
- FGV - MBA em Administração de Empresas (2011)
- IFRN - 2o/3o graus (1999/2004)

Apresentação - Profissional

- Sodexo / Pluxee
- Creditas
- DELL
- Walmart.com
- Globo.com
- Yahoo!

Apresentação - Contatos

- Linkedin: linkedin.com/in/leonardoml/
- Blog: infraascode.com.br
- Newsletter: engineeringmanager.com.br

Disciplina

Encontros

- 21/8 das 19h – 23:00h (ao vivo)
- 28/8 das 19h – 23:00h (ao vivo)

Aulas

- Tópicos 1-2: Apresentação do curso e Fundamentos teóricos
- Tópicos 3-4: Fundamentos e uso de ferramentas no contexto de IaC
- Tópicos 5-6: Configuração de serviços e servidores
- Tópicos 7-8: Gerenciamento serviços com IaC
- Tópicos 9-10: Governança de infraestruturas e futuro do IaC

Apresentação dos Temas

- Apresentação do curso e Fundamentos teóricos, apresenta os conceitos fundamentais de IaC, evolução histórica, benefícios operacionais e além de trazer práticas e fluxos de trabalho.

- Fundamentos de uso de ferramentas no contexto de IaC, monstra como utilizar ferramentas essenciais como Terraform, Ansible e técnicas de provisionamento e gerenciamento de infraestrutura.

- Configuração de serviços e servidores, continuamos dos fluxos de configuração de servidores e serviços por meio de código, implementar estratégias de imutabilidade e idempotência.

Apresentação dos Temas

- Gerenciamento de serviços com IaC, Objetivo: Implementar testes automatizados para infraestrutura, validar conformidade e estabelecer práticas de qualidade e segurança.

- Governança de infraestruturas e futuro do IaC: Estruturar governança para grandes organizações, definir modelos operacionais para equipes e explorar tendências futuras da IaC.

Fundamentação Teórica

O QUE É INFRA AS CODE?

Fundamentação Teórica

O QUE É INFRA AS CODE?

Infraestrutura como Código (IaC) é a prática de definir e gerenciar recursos de infraestrutura de TI por meio de arquivos de configuração legíveis por humanos.

Fundamentação Teórica

QUAL É A RAZÃO DE UTILIZAR IAC?

Fundamentação Teórica

- QUE TRABALHO?

- QUAL É O NOSSO TRABALHO?

- O QUE REALMENTE FAZEMOS?

Fundamentação Teórica

PRODUTO

1. Aquilo que é produzido; resultado da produção.
2. Aquilo que é produzido para venda no mercado.
3. Resultado de um trabalho ou de uma atividade.

Fundamentação Teórica

PRODUTO

Produto, em administração e marketing, é um conjunto de atributos, tangíveis ou intangíveis, constituído através do processo de produção, para atendimento de necessidades reais ou simbólicas, e que pode ser negociado no mercado, mediante um determinado valor de troca, quando então se converte em mercadoria.

Referência: Wikipédia (Produto Marketing)

Fundamentação Teórica

Fundamentação Teórica

- **QUE TRABALHO?**

Fazer a mesma coisa sempre?

Só faço o meu trabalho.

- **QUAL É O NOSSO TRABALHO?**

Fazer uma classe JAVA, Python.
Fazer uma função de validação?

- **O QUE REALMENTE FAZEMOS?**

Fazemos produtos para pessoas.
Realizamos atividades para aumentar a competitividade da empesa que trabalhamos.

Fundamentação Teórica

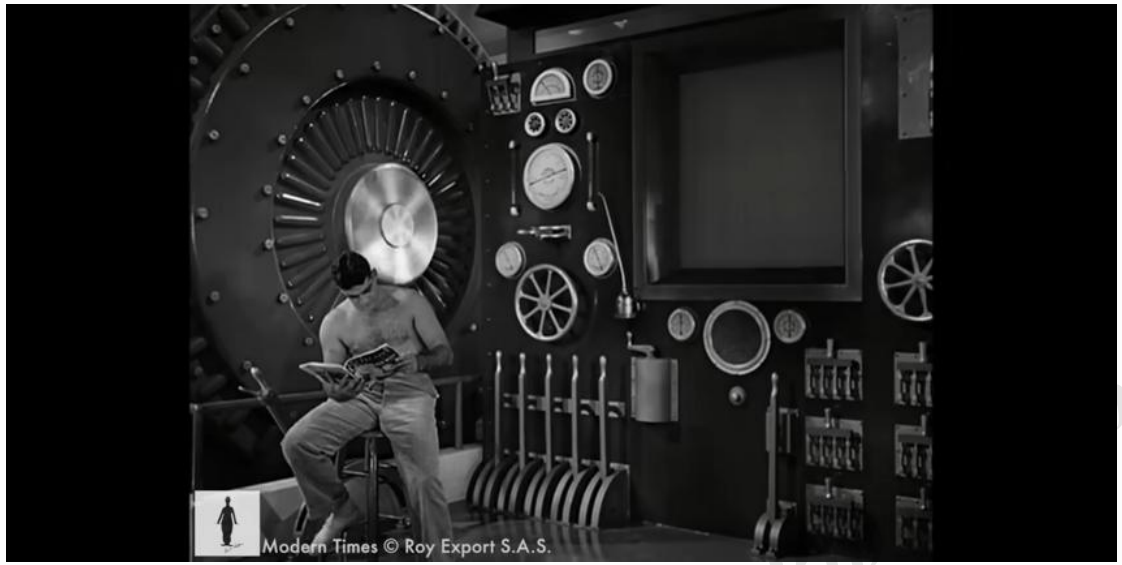
PRODUTOS DE QUALIDADE

Fundamentação Teórica

CONSISTÊNCIA

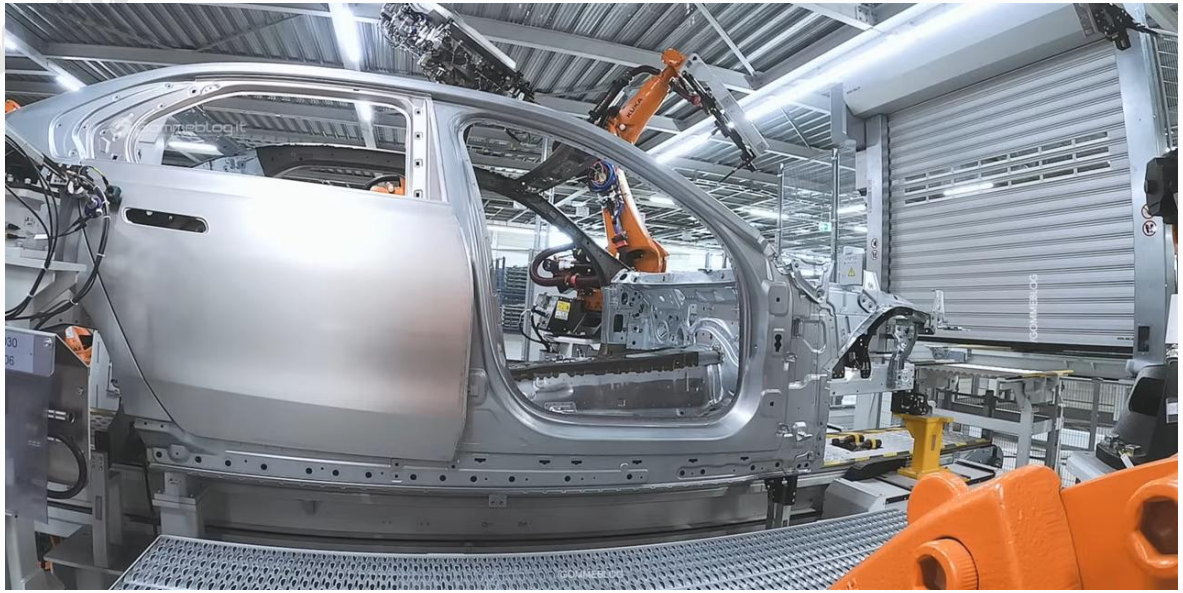
Fundamentação Teórica

CONSISTÊNCIA



Charlie Chaplin. 2019. **Charlie Chaplin - Factory Scene - Modern Times (1936).** Disponível no Youtube: https://www.youtube.com/watch?v=6n9ESFJTnHs

Gommeblog – Tecnologia e Desempenho Automotivo. 2024. **Inside BMW's Factory: How the New 7 Series is Manufactured.** Disponível no Youtube: https://www.youtube.com/watch?v=Kge89TAJGf0



Fundamentação Teórica

BUSCA PELA EXCELÊNCIA

No Sistema Toyota de Produção (TPS) está profundamente enraizada na filosofia de melhoria contínua, no respeito pelas pessoas e na eliminação de desperdícios.

Kaizen é uma abordagem sistemática de melhoria contínua nos processos, produtos, serviços e na cultura organizacional, que envolve todos os colaboradores, desde a alta liderança até o chão de fábrica.

Factory Window. 2024. **How Millions of Heineken Bottles are Made in a Factory | Heineken Factory Process.** Disponível no Youtube: https://www.youtube.com/watch?v=qWgy1uHIxwI

Fundamentação Teórica

Porque utilizar IaC?

- Os processos manuais de configuração de infraestrutura introduzem variabilidade humana que compromete a consistência da entrega de um trabalho.
- Aumenta a qualidade do trabalho realizado.
- Abordagem sistemática de melhoria contínua nos processos de entrega de software.

Fundamentação Teórica

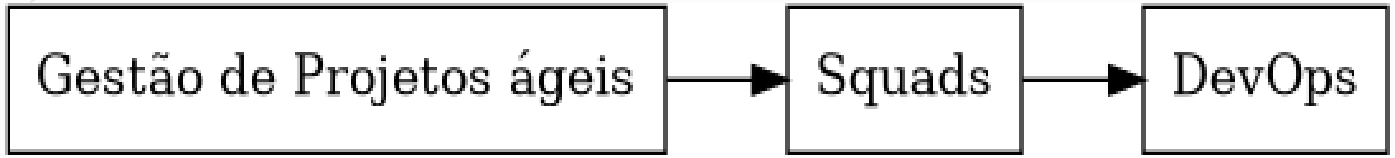
Contextualização histórica anos ~2000

- As empresas compravam servidores físicos.
- Cada servidor era único, com configurações feitas por meio de comandos diretos no sistema.
- O tempo para disponibilizar um novo ambiente podia levar dias ou semanas.

Fundamentação Teórica

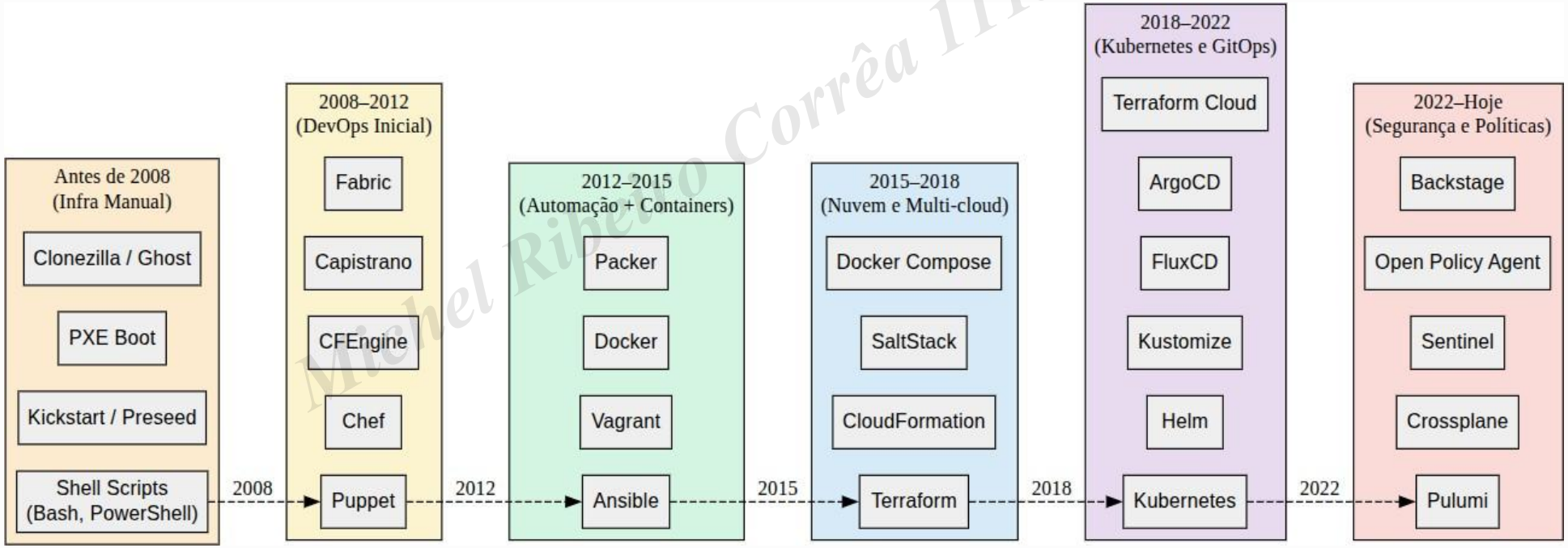
Contextualização histórica anos ~2000 e ~2009

- Popularização da virtualização como VMware.
- As empresas começaram a automatizar partes do processo, criando máquinas virtuais.
- A partir de 2008, com o surgimento do movimento DevOps, a integração entre times de desenvolvimento e operações.
- Popularização do conceito de Infrastructure as Code (IaC).



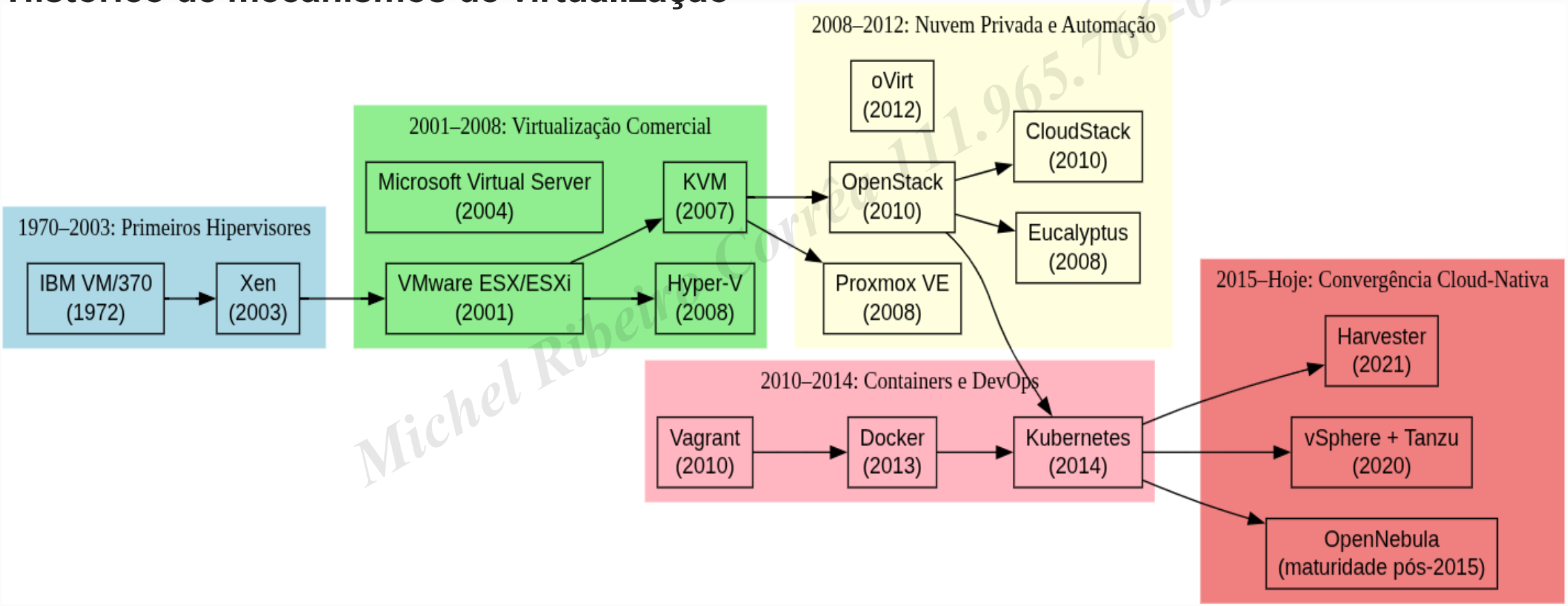
Fundamentação Teórica

Hoje, o uso de IaC é uma prática comum em empresas modernas que procuram aumentar sua qualidade de entrega. A infraestrutura é escrita, testada e versionada como o código da aplicação, possibilitando reprodutibilidade, escalabilidade e rastreabilidade.



Fundamentação Teórica

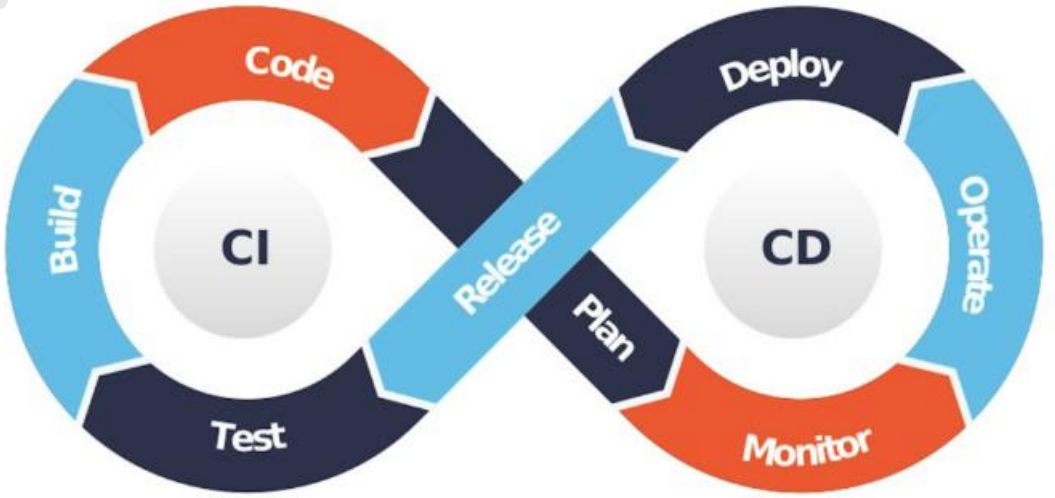
Histórico de mecanismos de virtualização



Fundamentação Teórica

Ciclo de desenvolvimento de software

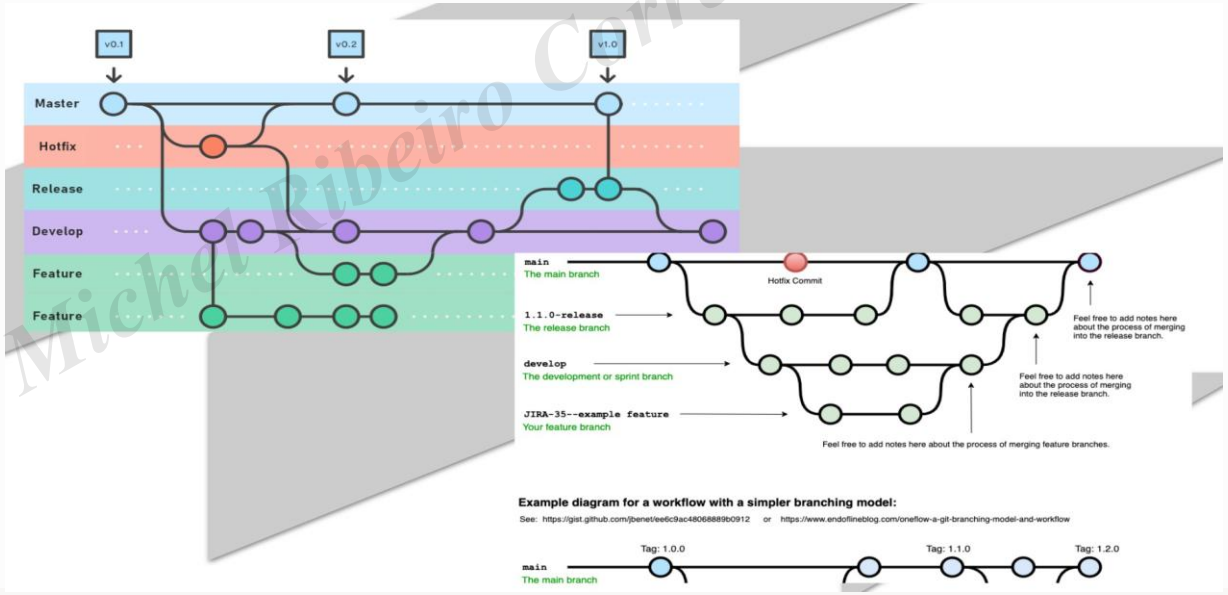
- O ciclo de desenvolvimento de software, representado de build, test, deploy e release.
- Prática de entrega contínua e integração contínua (CI/CD).
- Permite tratar configurações e provisionamento de ambientes com as mesmas práticas de versionamento.



Fundamentação Teórica

Ciclo de desenvolvimento de software - Versionamento

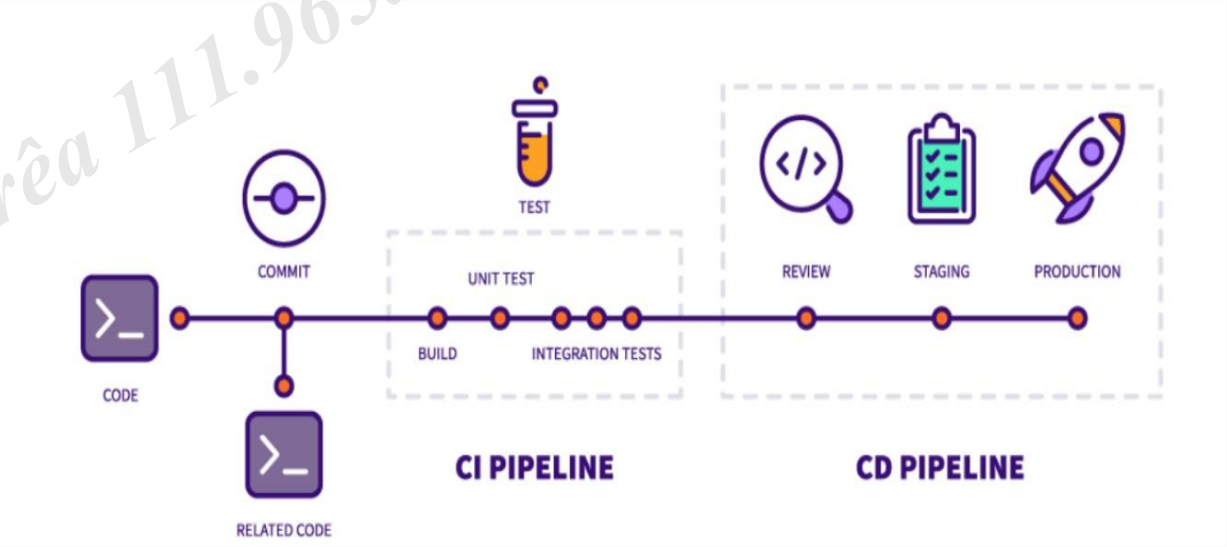
- Facilita a colaboração entre desenvolvedores, possibilita o trabalho simultâneo.
- Qualquer erro possa ser revertido com segurança para uma versão anterior.



Fundamentação Teórica

Ciclo de desenvolvimento de software - Pipelines de CI/CD

- Pipelines de CI/CD são fluxos automatizados que integram, testam e implantam mudanças de código com rapidez e segurança.
- Substituem tarefas manuais sujeitas a erros, como rodar testes ou copiar arquivos, garantindo consistência e rastreabilidade. Na prática, a CI valida.
- Garante ambientes consistentes em todas as etapas e reduz o risco operacional.



Fundamentação Teórica

Ciclo de desenvolvimento de software – Repetibilidade

- Permite recriar ambientes inteiros de forma idêntica.
- Acelera recuperação de desastres
- Facilita a criação de ambientes de teste

Fundamentação Teórica

Ciclo de desenvolvimento de software - Idempotência

- Idempotência garante que múltiplas execuções do mesmo código produzam o mesmo resultado, independente do estado inicial do sistema, eliminando efeitos colaterais indesejados.

$$f(f(x)) = f(x)$$

Intervalo

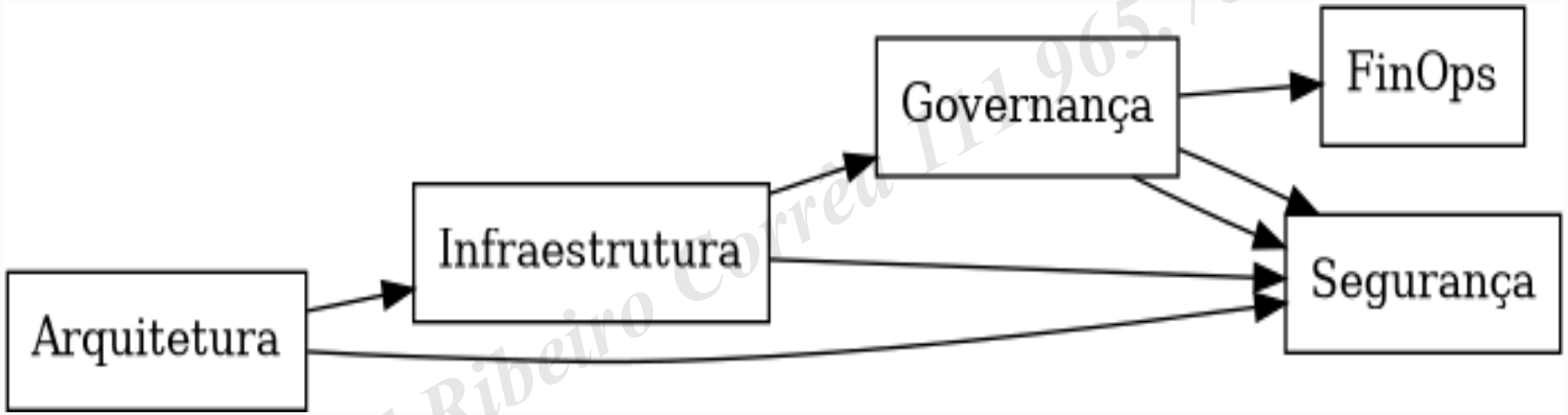
15 minutos

FUNDAMENTOS DE USO DE FERRAMENTAS

NO

CONTEXTO DE IAC

Fundamentos e Contexto de IAC

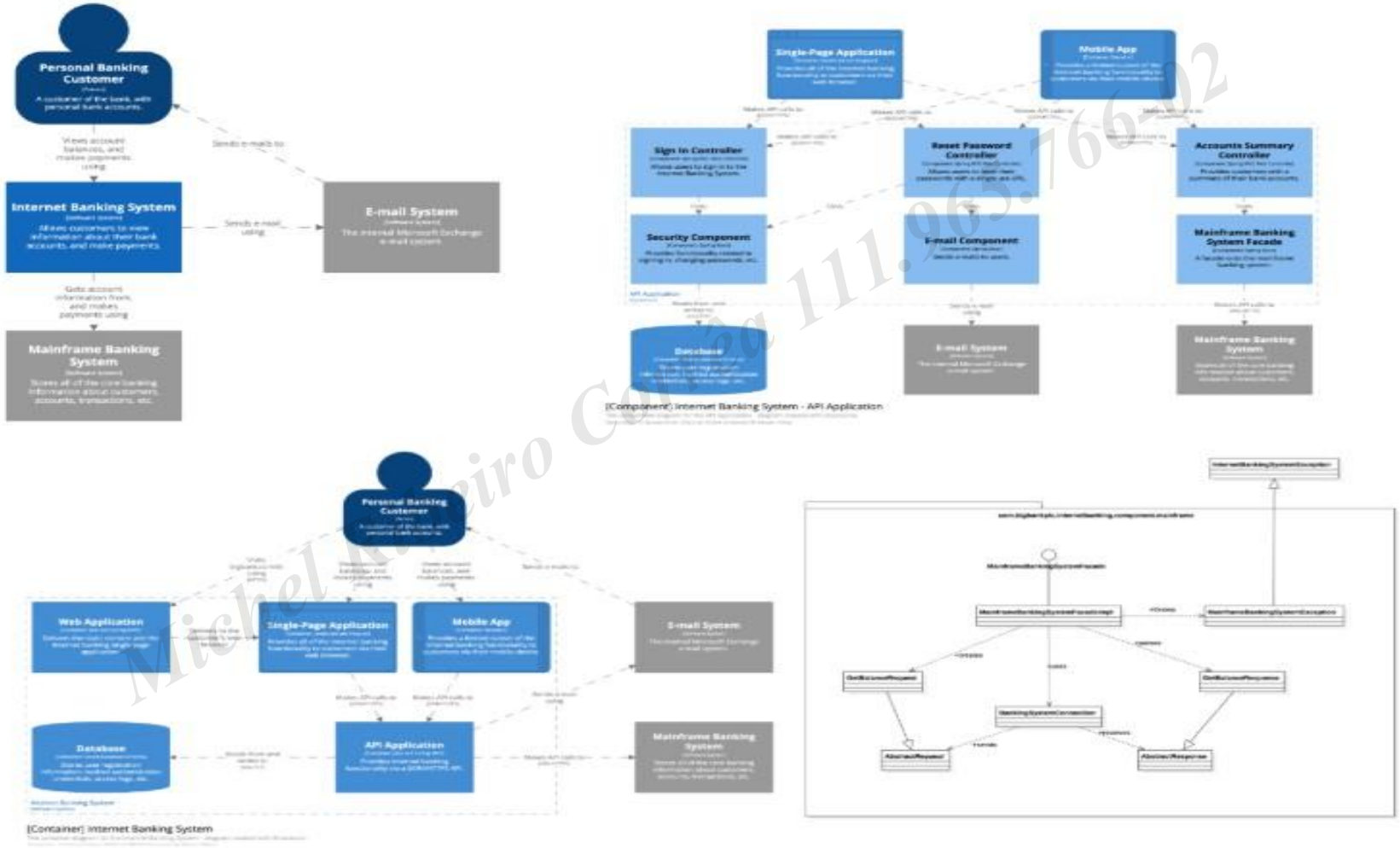


Fundamentos e Contexto de IAC

- **Arquitetura:** É nesse estágio que se definem os padrões técnicos e os componentes que suportarão as soluções.
- **Infraestrutura:** executa essas decisões por meio de automação, buscando escalabilidade e disponibilidade.
- **Governança:** Governa os recursos sobre quem pode fazer o quê, onde e quando.
- **Segurança:** Atua de forma transversal, é obrigatório estar presente desde a arquitetura.
- **FinOps:** Faz a gestão financeira da nuvem. Em ambientes dinâmicos, é essencial entender o custo real das decisões arquiteturais e operacionais.

Arquitetura como Código

C4 Model



Arquitetura como Código

C4 Model

- **Contexto:** Apresenta o sistema como um todo, seus usuários e outros sistemas com os quais interage o público alvo são os times de negócio, stakeholders, times externos.

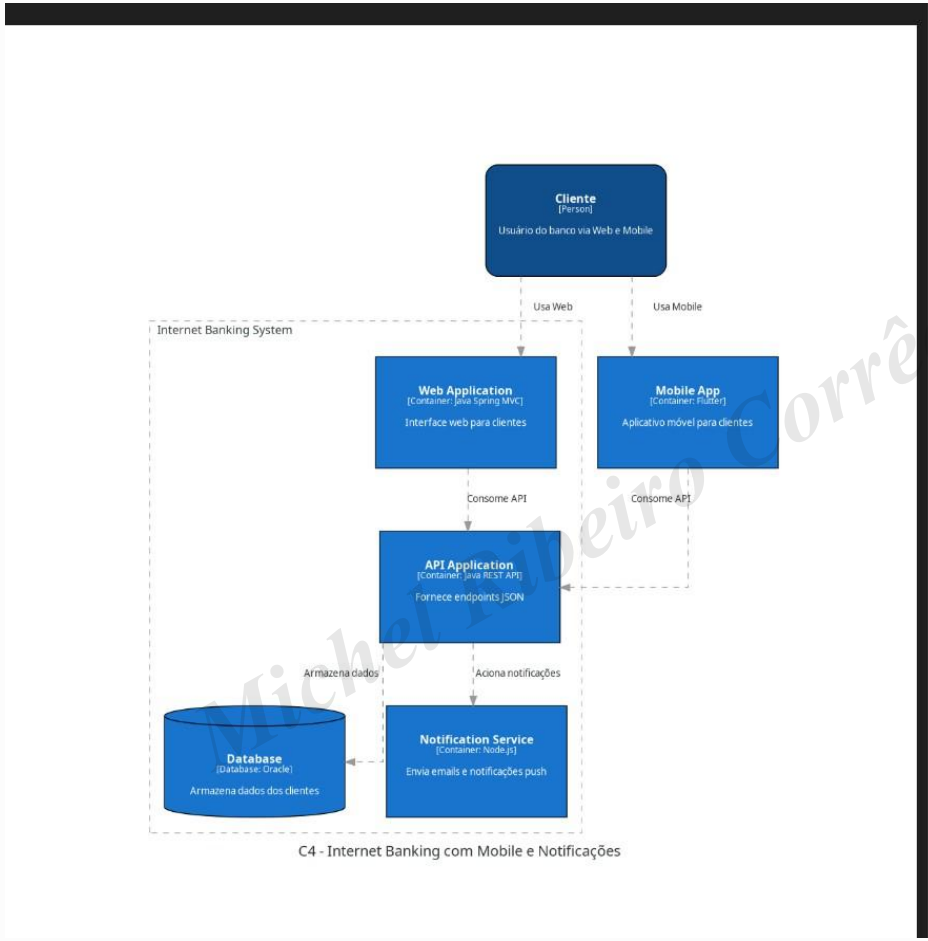
- **Container:** Os contêineres identificam dentro do contexto do sistema o: front-end, APIs, bancos de dados, serviços de background, essa visão é para os desenvolvedores, arquitetos e operadores do sistema.

- **Componente:** Apresenta os componentes internos de cada container (módulos, serviços, pacotes), o público de interesse desta visão são os desenvolvedores e arquitetos.

- **Código:** Mostra detalhes do código-fonte, como classes, métodos, arquivos, essa visão é destinada para os desenvolvedores.

Arquitetura como Código

Contexto



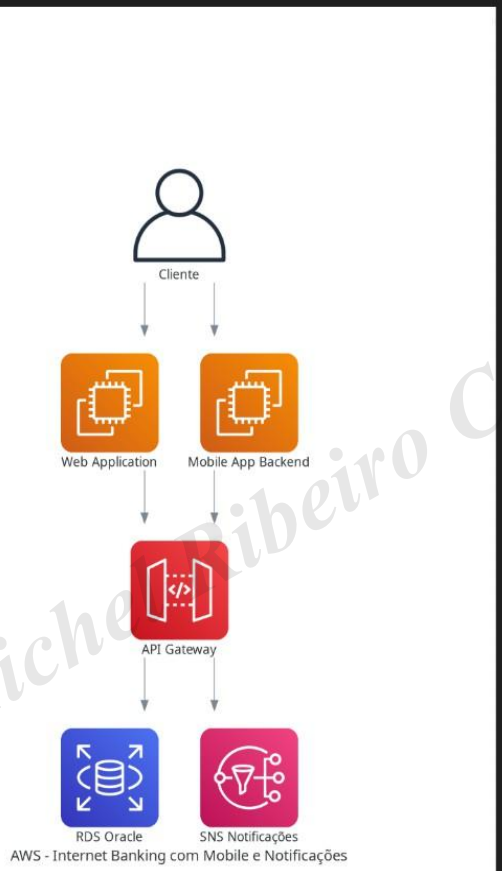
C4 - Internet Banking com Mobile e Notificações

```python
from diagrams import Diagram
from diagrams.c4 import Person, Container, Database, SystemBoundary, Relationship

with Diagram("C4 - Internet Banking com Mobile e Notificações", direction="TB"):
    customer = Person("Cliente", "Usuário do banco via Web e Mobile")

    with SystemBoundary("Internet Banking System"):
        webapp = Container(
            "Web Application",
            "Java Spring MVC",
            "Interface web para clientes"
        )

        api = Container(
            "API Application",
            "Java REST API",
            "Fornece endpoints JSON"
        )

        database = Database(
            "Database",
            "Oracle",
            "Armazena dados dos clientes"
        )

        notification_service = Container(
            "Notification Service",
            "Node.js",
            "Envia emails e notificações push"
        )

    mobileapp = Container(
        "Mobile App",
        "Flutter",
        "Aplicativo móvel para clientes"
    )

    customer >> Relationship("Usa Web") >> webapp
    customer >> Relationship("Usa Mobile") >> mobileapp

    webapp >> Relationship("Consome API") >> api
    mobileapp >> Relationship("Consome API") >> api

    api >> Relationship("Armazena dados") >> database
    api >> Relationship("Aciona notificações") >> notification_service
```

Arquitetura como Código

Container

aws_-_internet_banking_com_mobile_e_notificações.png



C4-ex-01-aws.py > ...

```python
from diagrams import Diagram
from diagrams.aws.compute import EC2
from diagrams.aws.database import RDS
from diagrams.aws.mobile import APIGateway
from diagrams.aws.ml import Personalize
from diagrams.aws.management import Cloudwatch
from diagrams.aws.integration import SNS
from diagrams.aws.general import User


with Diagram("AWS - Internet Banking com Mobile e Notificações", direction="TB"):
    customer = User("Cliente")

    webapp = EC2("Web Application")
    mobileapp = EC2("Mobile App Backend")
    api = APIGateway("API Gateway")
    database = RDS("RDS Oracle")
    notification = SNS("SNS Notificações")

    customer >> webapp
    customer >> mobileapp

    webapp >> api
    mobileapp >> api

    api >> database
    api >> notification

```

Arquitetura como Código

Desafios de Arquitetura

1. Realizar os exercícios de Arquitetura 01
2. Realizar os exercícios de Arquitetura 02
3. Realizar os exercícios de Arquitetura 03
4. Realizar os exercícios de Arquitetura 04
5. Aprofundar os conhecimentos na lib C4 Model
   https://c4model.com/

Intervalo

15 minutos

Infraestrutura como Código

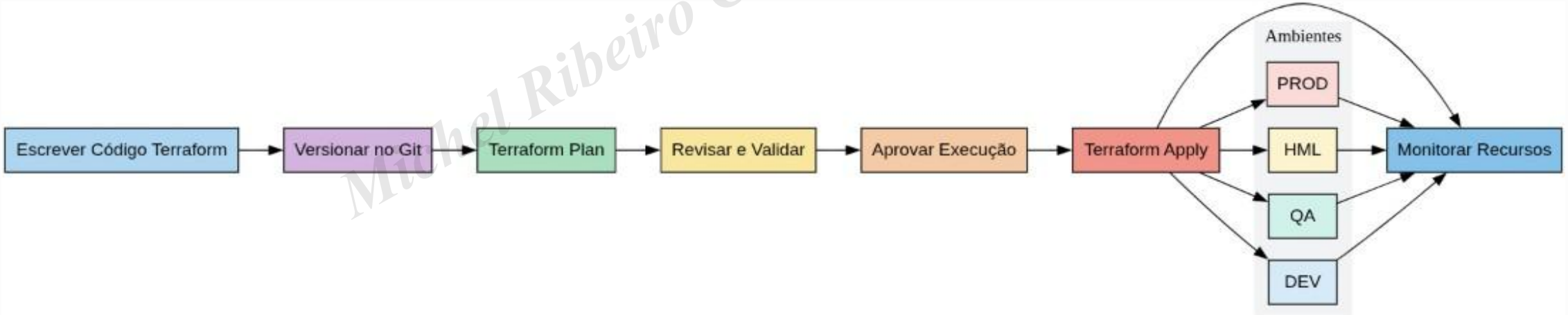
Modelo de Trabalho Antigo

- No passado, criar infraestrutura era uma atividade manual.
- Impossível reproduzir o ambiente de forma idêntica e gerava dependência de pessoas.

Infraestrutura como Código

Modelo de Trabalho Atual

- Versionar toda a definição do ambiente.
- Aplicar as mudanças de forma controlada e revisar o histórico de alterações.



Infraestrutura como Código

Modelo de Trabalho Atual

- **Reprodutibilidade:** o mesmo código Terraform pode ser executado várias vezes para criar ambientes idênticos.
- **Idempotência:** aplicar o código Terraform repetidamente não altera o estado se nada foi modificado.
- **Construção por blocos:** a infraestrutura é descrita em blocos reutilizáveis como módulos, recursos e variáveis.
- **Evolutividade:** a infraestrutura pode ser modificada de forma incremental mantendo controle sobre as mudanças.

Introdução ao Ansible

O **Ansible** é uma ferramenta open source, escrita em Python, que permite gerenciar servidores remotamente.

Introdução ao Ansible

O que o Ansible pode fazer ?

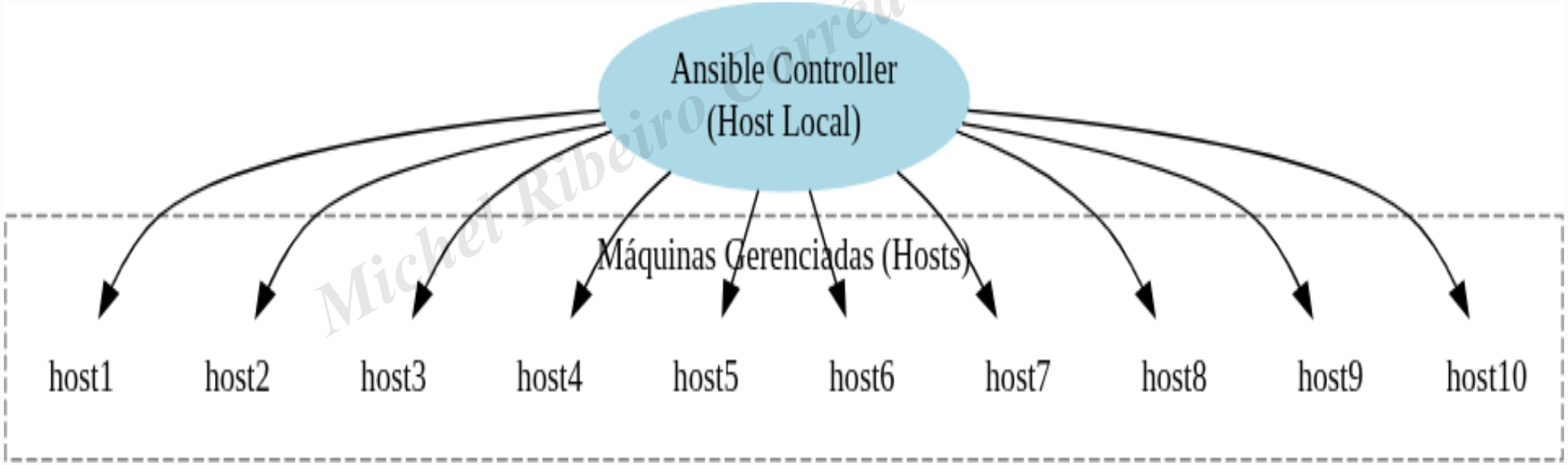
O que significa gerenciar servidores remotamente?

- Instalar e atualizar pacotes;
- Configurar arquivos de sistema;
- Reiniciar serviços;
- Criar e gerenciar usuários;
- Garantir configurações de segurança;
- Copiar arquivos e templates;
- Orquestrar mudanças em dezenas ou centenas de servidores de forma simultânea.

Introdução ao Ansible

Componentes básicos do Ansible?

- Control Node, de onde os comandos são executados.
- Managed Nodes, são os servidores que serão gerenciados.
- Inventory, lista os Managed Nodes.



Introdução ao Ansible

Componentes básicos do Ansible?

- Módulos: Blocos de código que realizam tarefas específicas.
- Playbooks: Arquivos que descrevem o que o Ansible deve fazer: associam hosts + tarefas.
- Roles: Estrutura organizada para reutilização de código.
- Templates: Arquivos que permitem gerar configurações dinâmicas com variáveis.

Introdução ao Ansible

Ansible Modules

Os modules no Ansible são blocos de funcionalidade que realizam tarefas específicas nos servidores gerenciados.

```
- name: Copia um arquivo
  ansible.builtin.copy:
    src: /tmp/index.html
    dest: /var/www/html/index.html
```

Introdução ao Ansible

Ansible Modules

O módulo file gerencia permissões, criação ou remoção de arquivos e diretórios:

```
- name: Garante que o diretório exista
    ansible.builtin.file:
    path: /var/log/app
    state: directory
    mode: '0755'
```

Introdução ao Ansible

Atividade

Desafio 01 de Ansible

Introdução ao Ansible

Desafios de Ansible

1. Realizar os exercícios de Ansible 01
2. Realizar os exercícios de Ansible 02
3. Realizar os exercícios de Ansible 03
4. Realizar os exercícios de Ansible 04
5. Aprofundar os conhecimentos dos módulos Ansible

Intervalo

15 minutos

Introdução ao Terraform

O **Terraform** é uma ferramenta de infraestrutura como código que permite criar, alterar e gerenciar recursos. Toda definição de ambientes fica registrada em arquivos de configuração que podem ser versionados.

Introdução ao Terraform

O que o Terraform pode fazer ?

- Criar recursos de infraestrutura;
- Gerenciar redes;
- Criar e anexar armazenamento;
- Configurar variáveis e parâmetros;
- Criar múltiplos recursos dinamicamente;
- Consultar dados de provedores;
- Alterar/destruir infraestrutura.

Introdução ao Terraform

Comandos principais do Terraform:

- `terraform init` - Inicializa o diretório de trabalho e baixa os plugins necessários.
- `terraform plan` - Mostra as ações que serão executadas para atingir o estado desejado.
- `terraform apply` - Aplica as mudanças descritas no plano ao ambiente.
- `terraform destroy` - Remove todos os recursos gerenciados pelo código.

Introdução ao Terraform

Comandos principais do Terraform:

- `terraform validate` - Verifica se a sintaxe dos arquivos de configuração está correta.
- `terraform fmt` - Formata os arquivos de configuração de forma padronizada.
- `terraform show` - Exibe detalhes sobre o estado atual dos recursos.
- `terraform output` - Mostra os valores de saída definidos no código.

Introdução ao Terraform

Terraform módulo

Um módulo Terraform é um conjunto de arquivos que define recursos reutilizáveis e organizados para provisionar infraestrutura. Ele pode ser usado localmente ou compartilhado entre projetos. A estrutura básica de um módulo inclui os seguintes arquivos e diretórios:

- main.tf: arquivo principal, onde os recursos são definidos.
- variables.tf: define as variáveis de entrada que o módulo recebe.
- outputs.tf: define os valores de saída que o módulo retorna.
- terraform.tfvars (opcional): define valores para variáveis.
- providers.tf (opcional no módulo, usado principalmente no root): define o provedor, como AWS, Azure, GCP.
- README.md (opcional): documentação do módulo.

Introdução ao Terraform

Criação de Recursos

Criando uma VM na AWS

```
resource "aws_instance" "web" {
    ami    = "ami-12345678"
    instance_type = "t2.micro"
}
```

Introdução ao Terraform

Criação de Recursos

Criando 3 VMs na AWS

```
resource "aws_instance" "web" {
  count    = 3
  ami      = "ami-12345678"
  instance_type = "t3.micro"
}
```

Introdução ao Terraform

Criando, Alterando, Destruindo

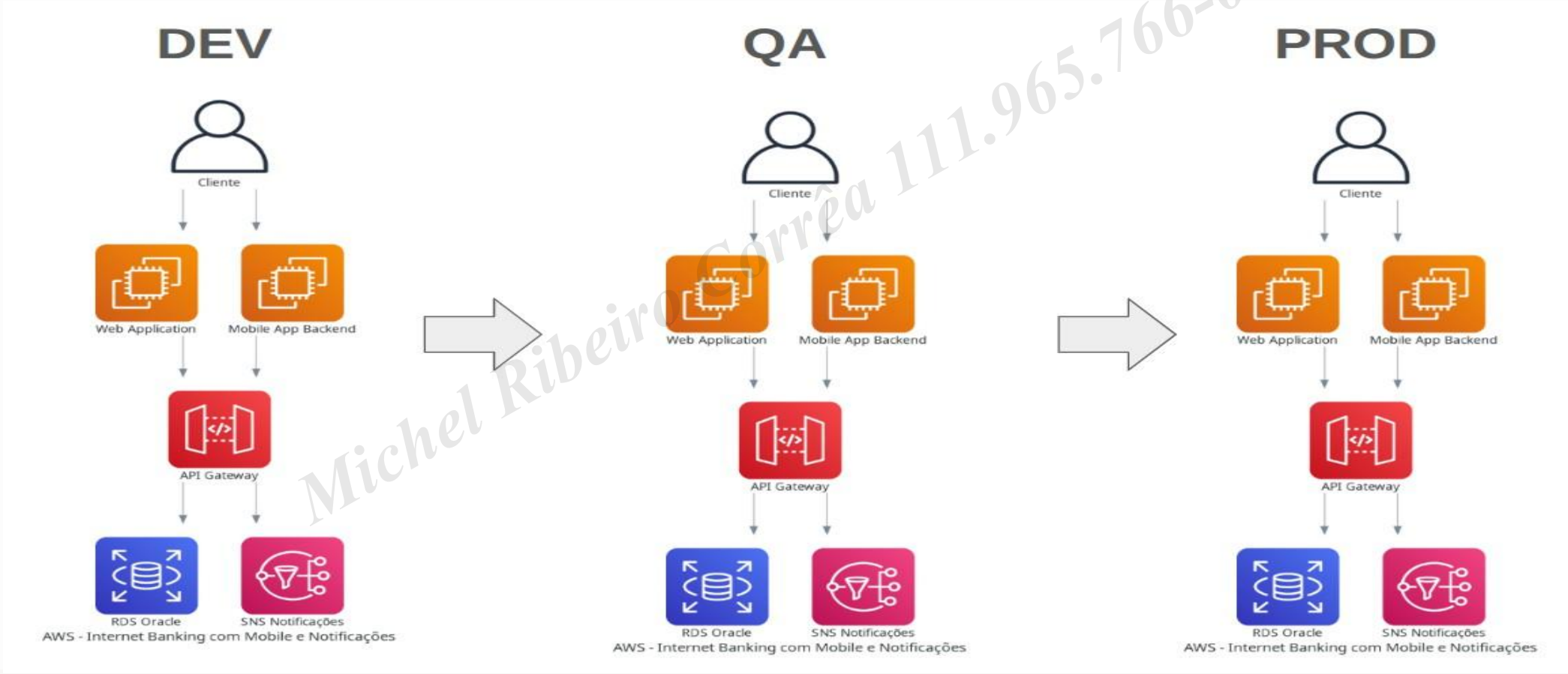
```
$ terraform plan
$ terraform apply
$ terraform destroy
```

Intervalo

15 minutos

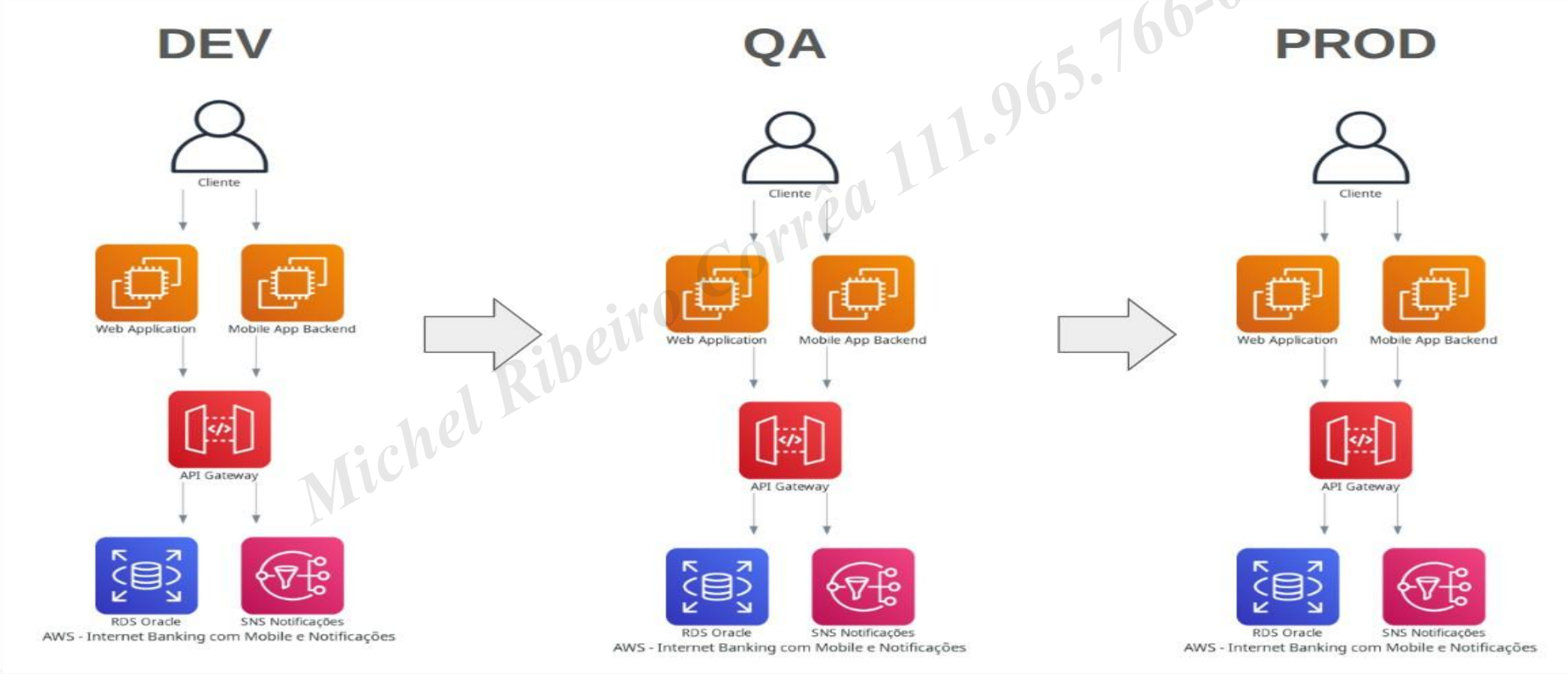
Terraform Múltiplos Ambientes

Construindo DEV, QA e PROD



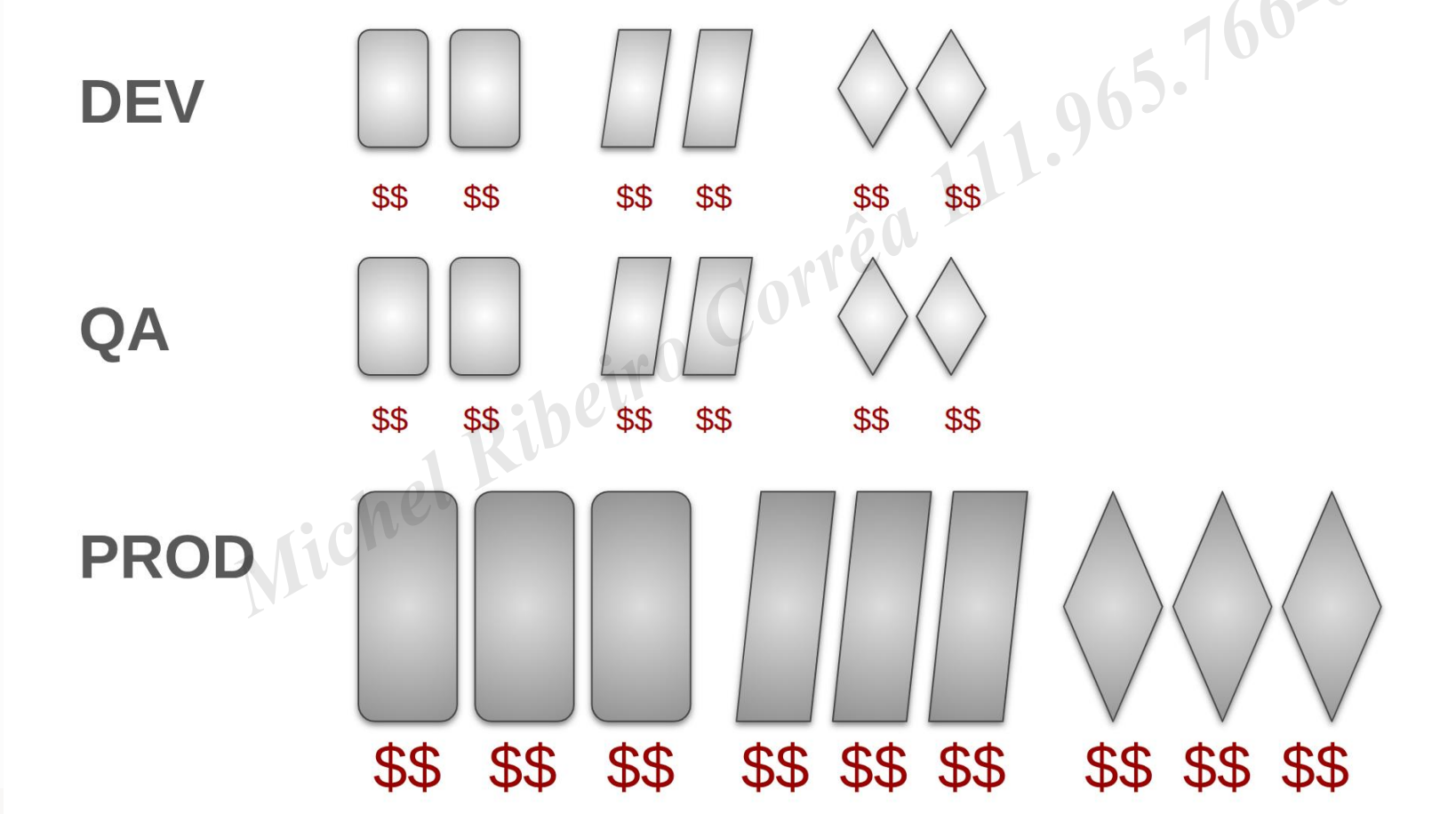
Terraform Múltiplos Tamanhos

Construindo DEV, QA e PROD



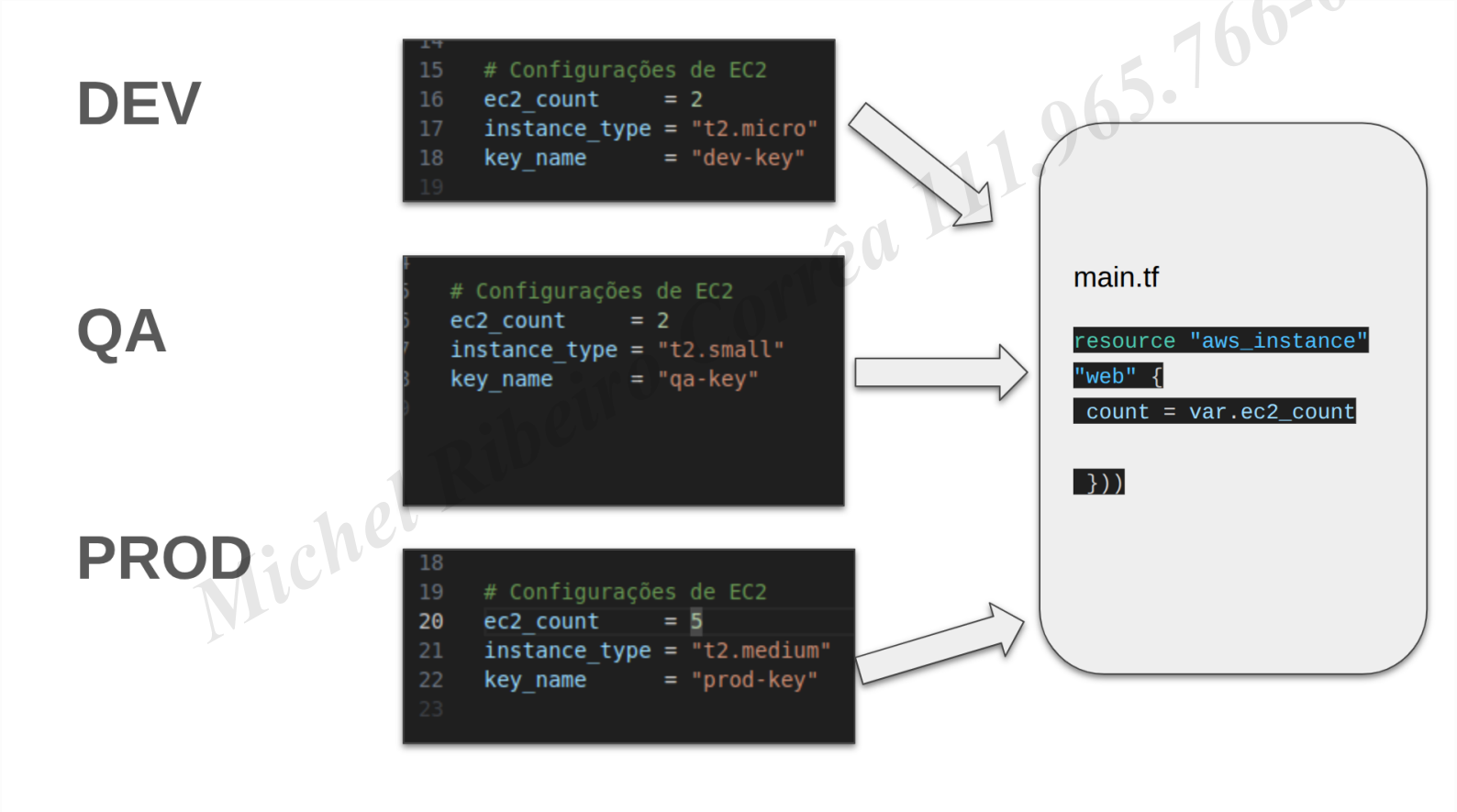
Terraform Múltiplos Custos

Construindo DEV, QA e PROD



Terraform Múltiplos Custos

Construindo DEV, QA e PROD



Intervalo

15 minutos

Governança como Código

Governança como Código representa a evolução natural dos processos de TI, onde políticas, regras de compliance e controles de segurança são definidos por meio de código versionado e executado automaticamente.

Governança como Código

Políticas como Código

- Restrições de tipo e tamanho de recursos
- Obrigatoriedade de tags
- Proibição de recursos públicos
- Restrições de localização geográfica
- Compliance de segurança

Governança como Código

Políticas como Código

- Compliance contínuo (Continuous Compliance)
- Evidências automáticas
- Redução de esforço manual

Governança como Código

Desafios de Governança

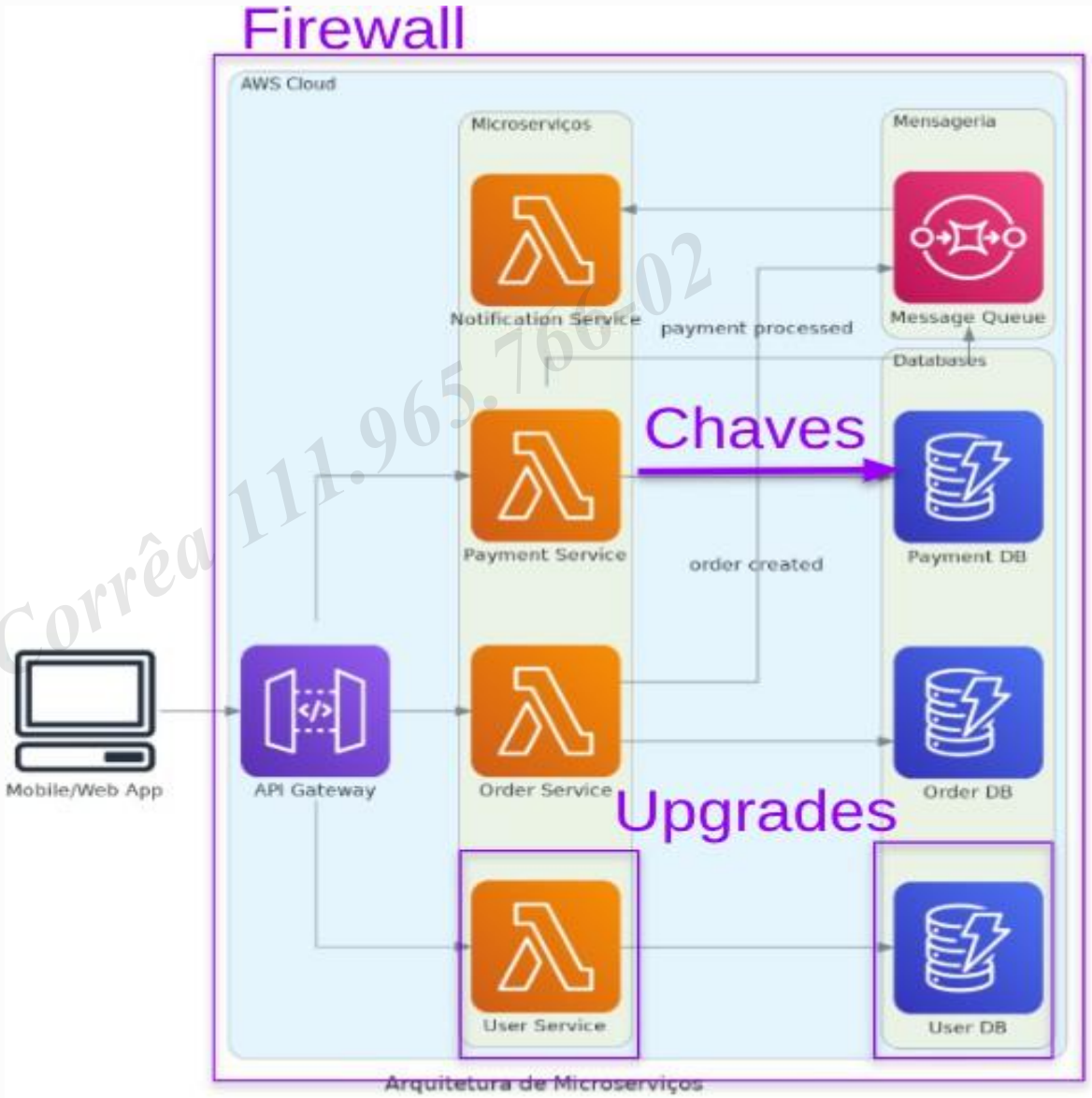
- Pesquisar sobre os frameworks de governança ITLv3 e COBIT

Segurança como Código

Segurança como código é a prática de tratar controles de segurança como parte do mesmo processo de automação da infraestrutura.

Segurança como Código

- Grupos de segurança e regras de firewall.
- Aplicação de políticas de ciclo de vida de chaves.
- Automação de patches e atualizações de sistema.
- Atualização de componentes.



Segurança como Código

- SAST (Static Application Security Testing): Analisa o código-fonte repouso para identificar vulnerabilidades antes da execução.

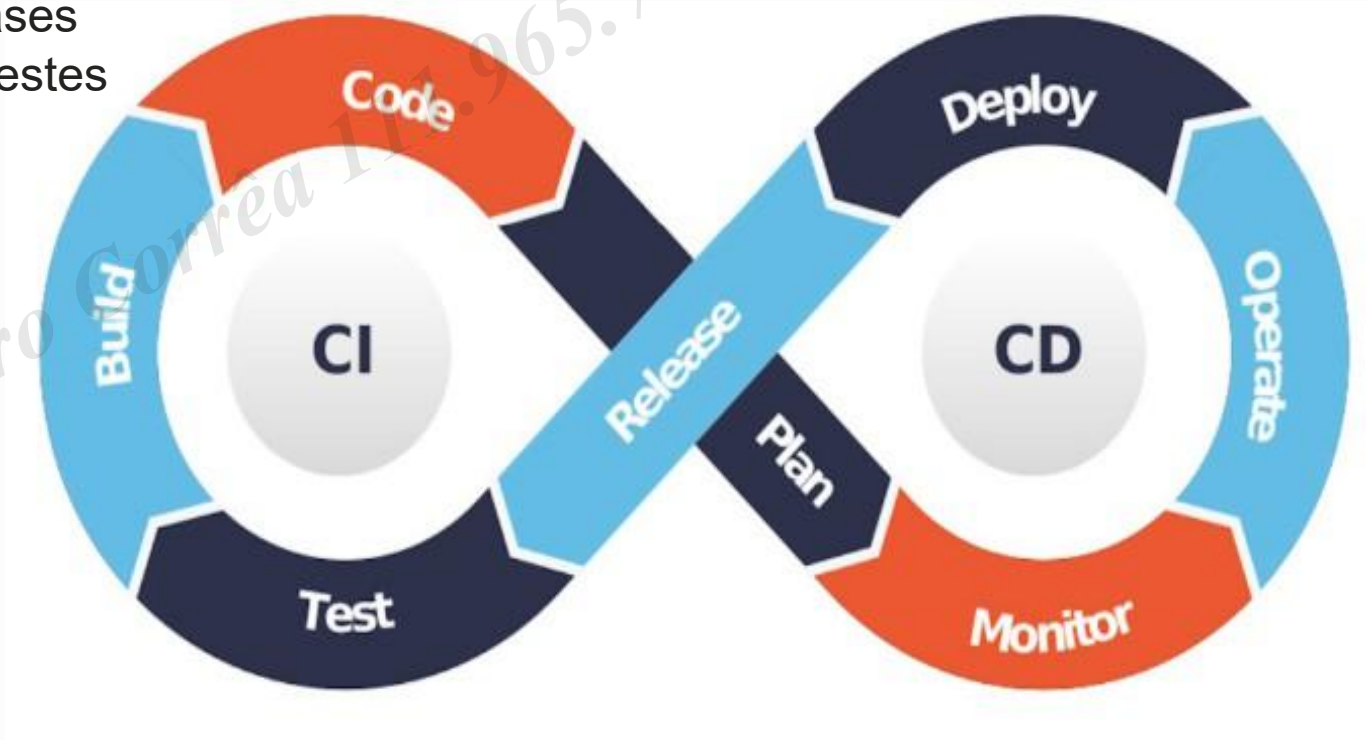
- DAST (Dynamic Application Security Testing): Testa a aplicação em execução, simulando ataques externos para encontrar falhas exploráveis.

- IAST (Interactive Application Security Testing): Monitora a aplicação em execução de dentro dela, combinando técnicas de SAST e DAST para detectar vulnerabilidades em tempo real.

Segurança como Código

Shift-Left Security

- Prática de integrar segurança desde as fases iniciais do desenvolvimento, antecipando testes e correções para antes da produção.



Segurança como Código

- Compliance as Code
- Auditoria
- Conformidade Automatizada

Segurança como Código

Desafio Segurança como código

- Pesquisar o funcionamento da ferramenta SonarQube, Snyk e Trivy.
- Pesquisar técnicas de Shift-Left aplicada ao fluxo de CI/CD.
- Pesquisar sobre o padrão SOC2 e PCI-DSS.

FinOps

FinOps tem como objetivo principal gerenciar custos de nuvem com o objetivo claro de reduzir desperdícios e gerar redução de custos.

FinOps

Visibilidade

- Monitoramento centralizado: dashboards com consumo por serviço, conta e região.
- Detalhamento granular: custo por recurso, tag, projeto ou ambiente.
- Alertas e previsões: identificar desvios antes que se tornem surpresas na fatura.
- Transparência entre áreas: permitir que engenharia, produto e finanças vejam os mesmos números.

FinOps

Alocação de Custos

- Tagueamento consistente: uso de chaves como Owner, Environment, Project para rastrear gastos.

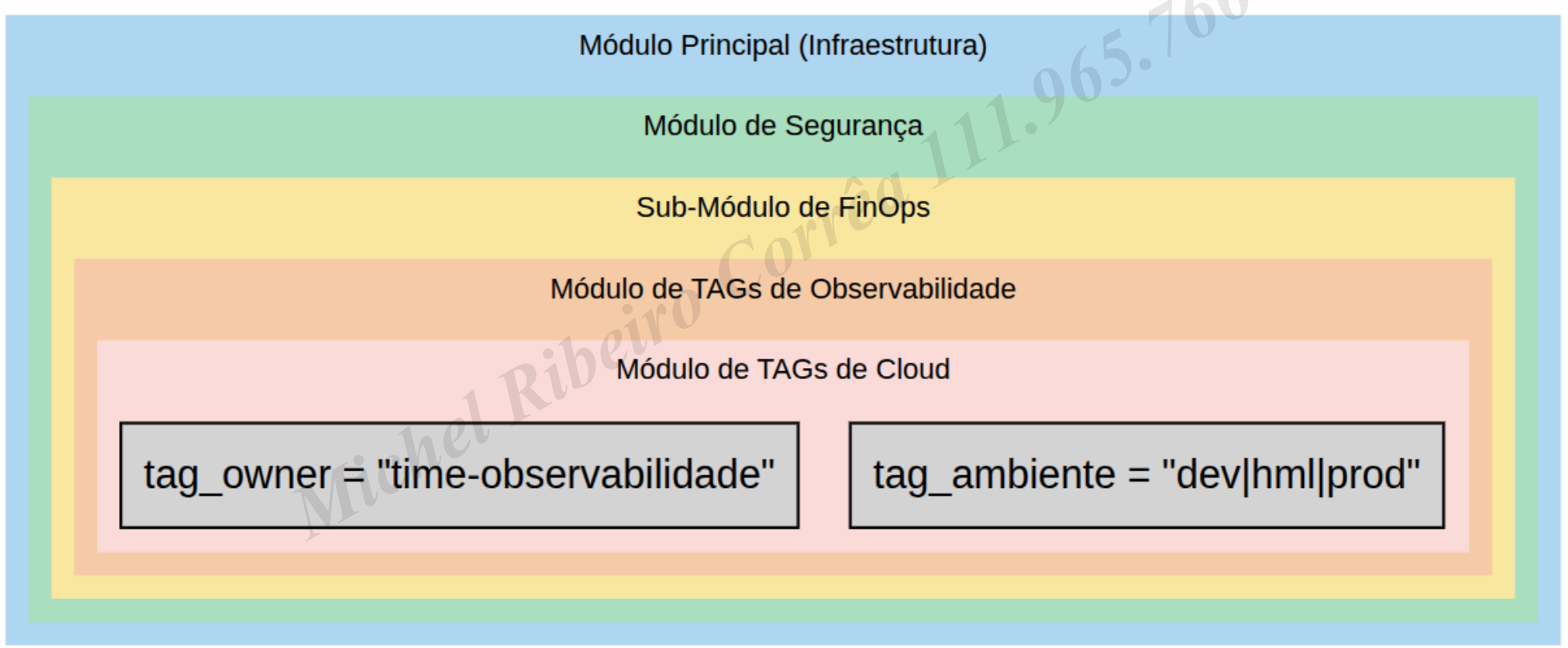
- Account separation: contas de nuvem separadas por time ou produto para facilitar rateio.

- Chargeback / Showback:
  - Showback → mostrar os custos para os times, sem cobrança interna.
  - Chargeback → repassar custos diretamente ao centro de custo responsável.

- Custos compartilhados: definir critérios para dividir recursos comuns.

FinOps

Tagueamento consistente

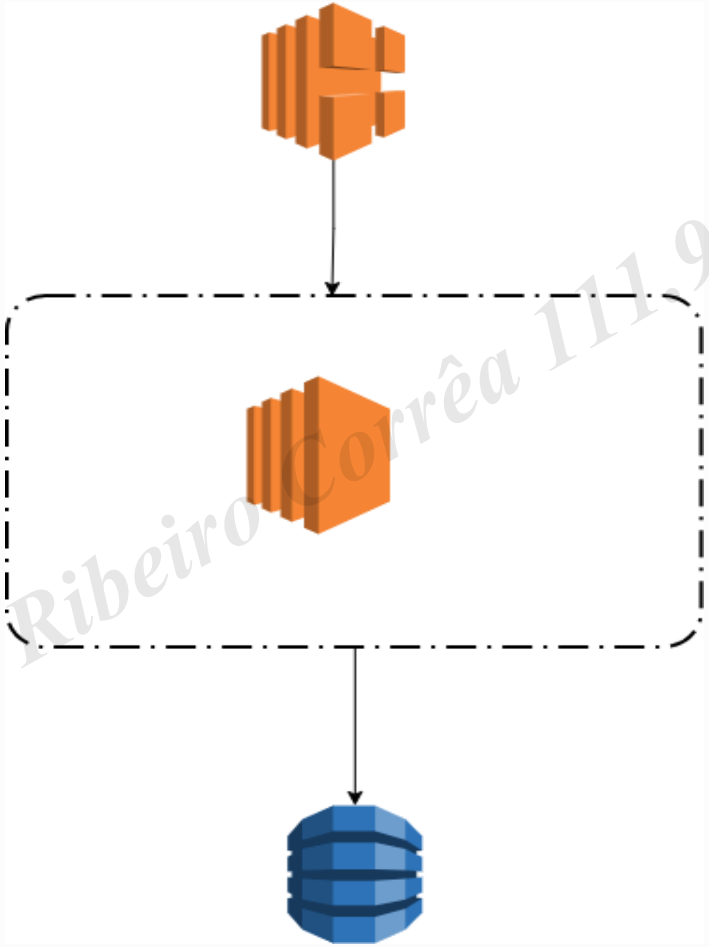


O Futuro da IaC

- Arquitetura de soluções, caminha para a integração completa entre IaC e FinOps.
- Projetar sistemas considerando performance, escalabilidade e custo.
- Engenharia de custos de cloud, sugerir componentes e otimizações para reduzir custos.
- Expertise em ferramentas de IaC como (Terraform, CloudFormation, Pulumi).
- Gestão automatizadas de ambientes das aplicações por meio de pipelines.
- IaC com AI, é um campo muito recente com muitas oportunidades para surgir.

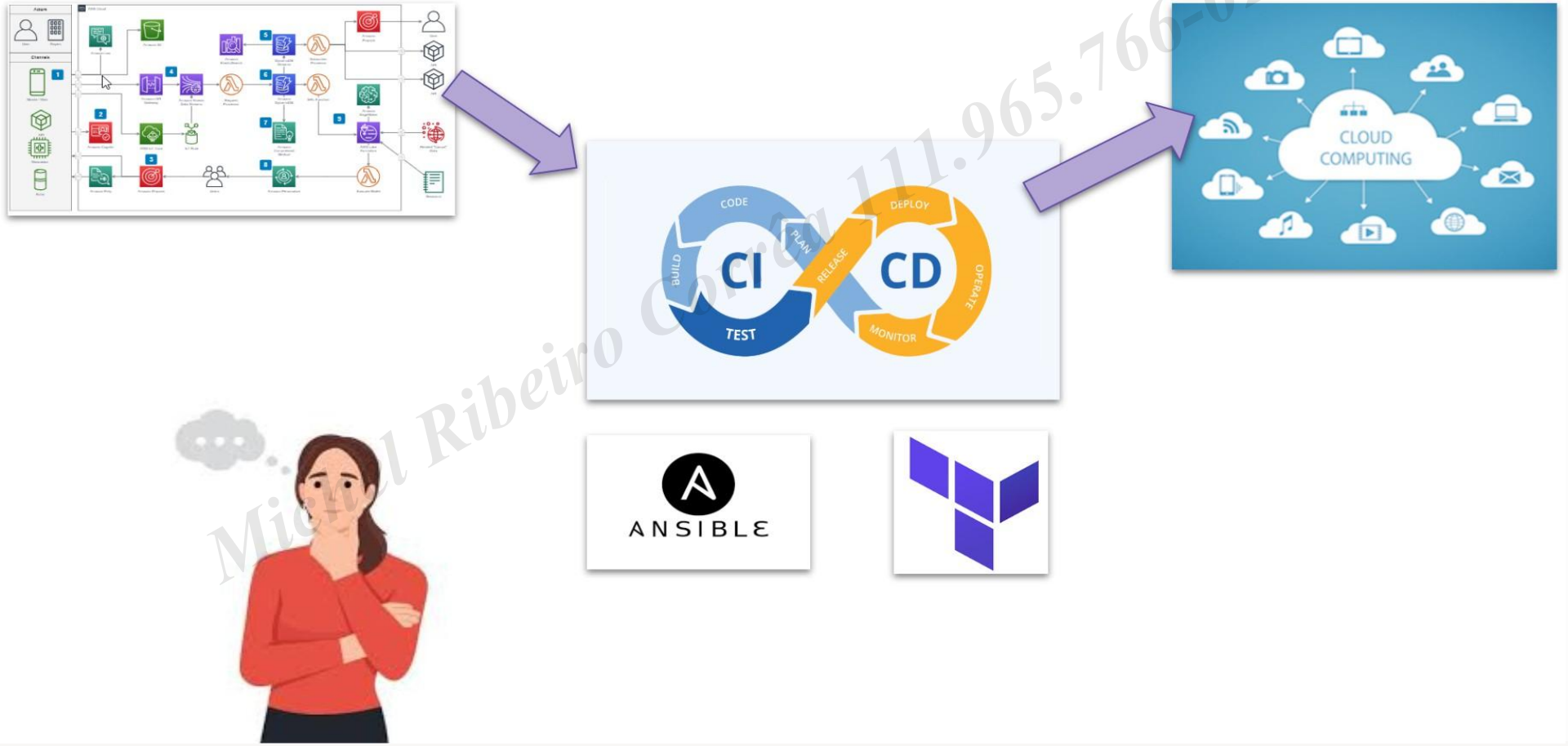
O Futuro da IaC

IaC + AI



O Futuro da IaC

IaC Automation



Obrigado!

Cara(os) alunas(os)

Desejo muita sorte em sua jornada!!

- Linkedin: linkedin.com/in/leonardoml/
- Blog: infraascode.com.br
- Newsletter: engineeringmanager.com.br

Obrigado!

Professor Leonardo Lima | linkedin.com/in/leonardoml/

MBA USP ESALQ