

Ambiente de Dados e Operações - DataOps

DevOps, IaC e CloudFormation

Professor MSc. Fernando Sousa

DevOps

- Desenvolvimento + Operações
 - Conjunto de práticas e ferramentas que combinam o desenvolvimento de aplicações (dev) com operações de TI (ops)
 - União de pessoas, processos e tecnologias para fornecer valor ao cliente continuamente
- O objetivo é diminuir o tempo do ciclo de vida das aplicações
 - Mais entregas menores em menos tempo
- A entrega contínua e de alta qualidade são conceitos que caminham junto de DevOps
- É um complemento às metodologias ágeis

DevOps

- Possibilita que diversas funções desempenhadas no ciclo de desenvolvimento de aplicações atuem de forma coordenada e colaborativa para criar aplicações confiáveis e de qualidade
 - Desenvolvedores
 - Analistas
 - Engenheiros
 - Equipe de dados
 - Segurança
 - Operações
 - Qualidade

DevOps - Vantagens

Responder melhor e mais rápido às necessidades dos clientes

Entregar soluções com mais qualidade e menos erros

Entregar soluções mais confiáveis

Entregar mais soluções menores, em menos tempo

Diminuir tempo do ciclo de entrega

Cumprir metas empresariais mais rápidas

Alto desempenho da equipe

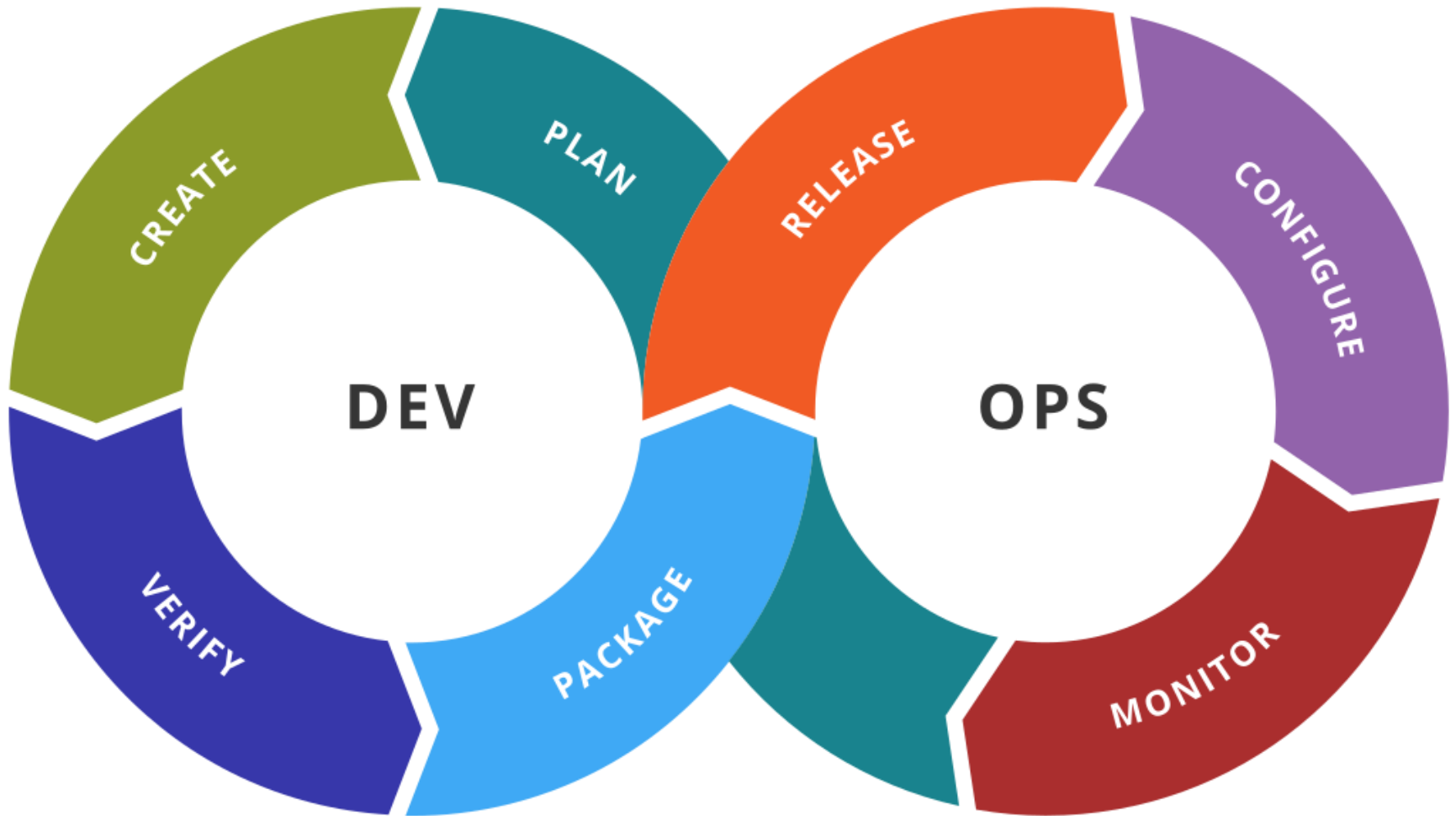
Maior satisfação do cliente

Aumenta da colaboração e produtividade

DevOps

- DevOps é representado por um ciclo que envolve 7 fases:
 - Planejamento
 - Criação/desenvolvimento
 - Verificação/testes
 - Preparação para lançamento (pacote)
 - Lançamento
 - Configuração do ambiente
 - Monitoramento
- Ao final de um ciclo, outro começa com novas funcionalidades

DevOps



Planejamento

- Primeiro passo para entrega de valor de negócio
- Definir as necessidades e requisitos da aplicação
- Planejar as próximas fases
- Envolve diversas pessoas da equipe
 - Desenvolvedores
 - Arquitetos/Engenheiro/Cientista/Analista de dados
 - Infraestrutura
 - Gestores
 - Proprietário do negócio

Criação/desenvolvimento

Construir, codificar e configurar o processo de desenvolvimento

Codificar pesando em qualidade (testes) e desempenho

Construir o pacote para publicação

Gerar uma versão candidata

Verificação

Garantir a
qualidade da
versão da
aplicação

Mantar a
qualidade quando
novas versões são
criadas

Implantar a versão
com a mais alta
qualidade

Teste de regressão

Teste de aceitação

Análise de
segurança e
vulnerabilidade

Análise de
desempenho

Teste de
configuração

Automação de
testes

SPC

Preparação para lançamento (pacote)

Feita quando a versão está pronta para publicação

- Staging ou pré-produção

Atividades:

- Aprovações
- Configuração do pacote
- Acionar gatilhos de lançamento
- Lançar o pacote em staging

Lançamento



Agendar
lançamento



Orquestrar
aplicações



Provisionar
recursos



Implantar versão
em produção

Configuração do Ambiente



Tarefa da equipe de operações



Provisionamento e configuração de infraestrutura adicional



Provisionamento e configuração da aplicação

Monitoramento

Monitorar a execução da aplicação

Identificar problemas específicos

Avaliar o impacto para o usuário final

Monitorar infraestrutura para encontrar gargalos

Verificar experiência do usuário

Usar métricas e estatísticas de produção (SPC)

The Periodic Table of DevOps Tools (V4.2)

digital.ai

CollabNet Version One, Kelsa Labs, Arxan, Numerify & Expertise
are now Digital.ai

Fonte: https://digital.ai/sites/default/files/pictures/2020-06/Digital.ai_Periodic-Table-of-DevOps.pdf

Práticas de DevOps

CI/CD (Integração
Contínua e
Entrega Contínua)

Controle de
versão

Desenvolvimento
ágil

Microserviços

Monitoramento
contínuo

Gerenciamento
de configuração

Infraestrutura
como um código

CI/CD (Integração Contínua e Entrega Contínua)

Integração contínua (CI – Continuous Integration)

- Juntar as alterações no código em um repositório central frequentemente
- Executar testes automatizados
- Objetivos
 - Encontrar erros mais rapidamente
 - Melhorar qualidade
 - Reduzir tempo para validar e publicar

Entrega contínua Implantação Contínua

(CD – Continuous Delivery e Continuous Deployment)

- Alterações são criadas, testadas e preparadas automaticamente para liberar para produção
- Expansão de CI
- Gera um pacote pronto para implantação, devidamente testado
- Se o processo estiver maduro o suficiente, pode-se adotar a Implantação contínua (aplicar as alterações em produção)

CI/CD (Integração Contínua e Entrega Contínua)



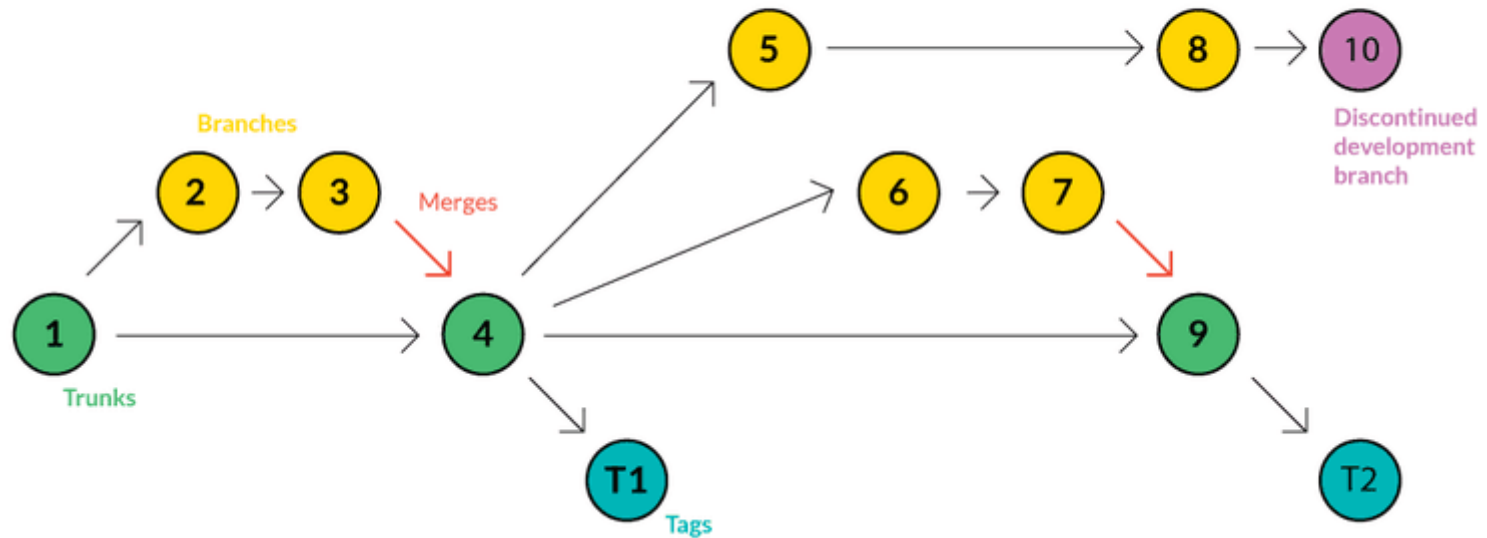
<https://www.redhat.com/pt-br/topics/devops/what-is-ci-cd>

Controle de versão

- Gerenciar códigos por meio de versões
 - Acompanhar revisões e histórico de alterações
 - Facilitar revisão e recuperação
- Sistemas de controle de versão
 - Git
- Colaboração de vários membros da equipe
 - Divisão das tarefas
 - Trabalhar sem interferir no trabalho dos outros
- Mesclar alterações, gerenciar conflitos e reverter alterações
- Ajuda no CI/CD e IaC

Controle de versão

What is “version control”?



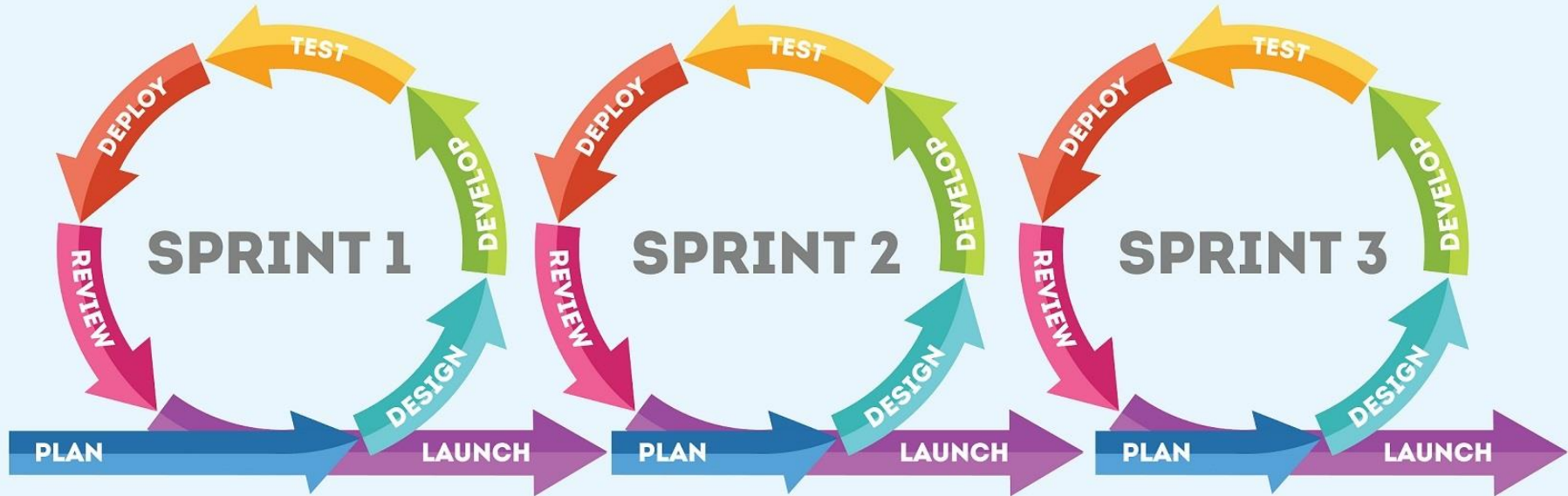
<https://webinerds.com/version-control-systems-keep-your-code-in-order/>

Desenvolvimento ágil

- Abordagem de desenvolvimento de aplicações
 - Enfatiza a colaboração
 - Comentários do cliente/usuário
 - Adaptabilidade a mudança por ciclos curtos de versões
- Fornecer mudanças e melhorias contínuas aos clientes
- Cliente é parte do processo
 - Coletar feedback
 - Aprender com os comentários
 - Ajustar de acordo com as necessidades
- Utiliza Kanban e Scrum

Desenvolvimento ágil

AGILE



<https://www.soldevelo.com/blog/is-agile-always-the-best-solution-for-software-development-projects/>

Microserviços



Criar aplicação composta por pequenos serviços



Cada serviço executa seu próprio processo



Comunicação por uma interface
(regras) bem definidas

APIs baseadas em HTTP



Cada serviço deve ter uma finalidade única



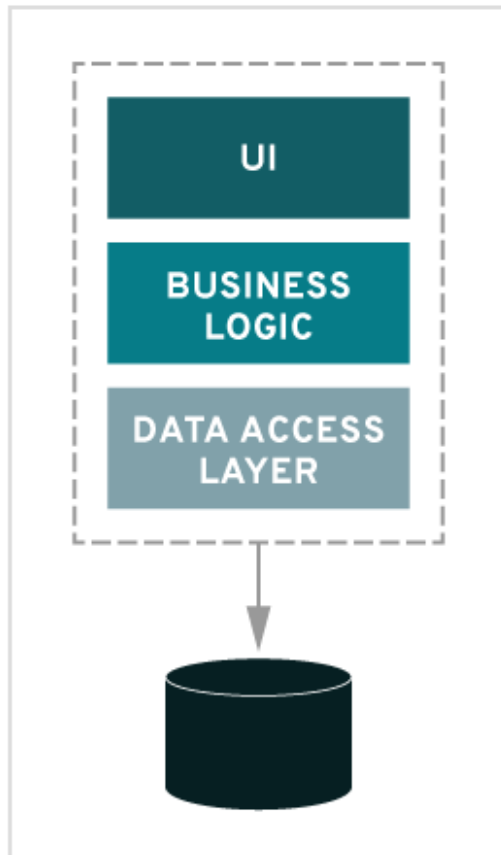
Independente da linguagem de programação



Facilita a implantação

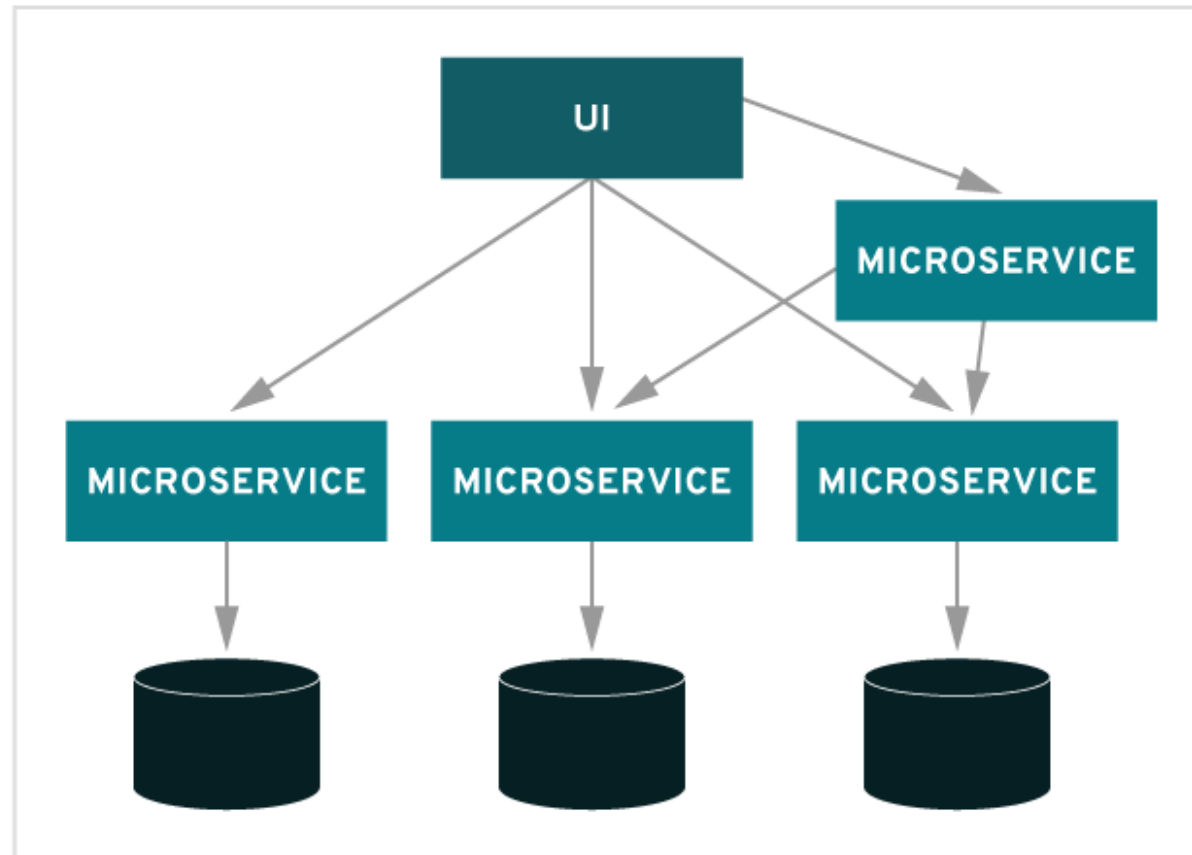
Microserviços

MONOLITHIC



VS.

MICROSERVICES

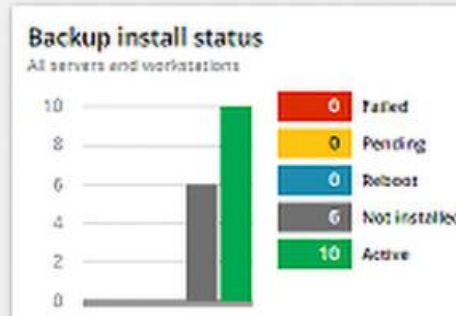
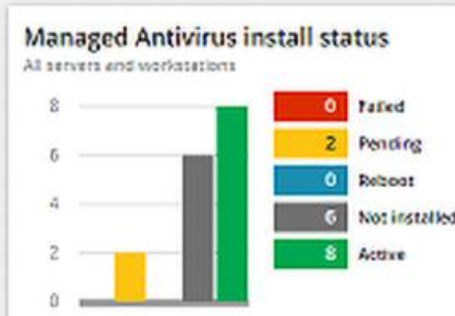
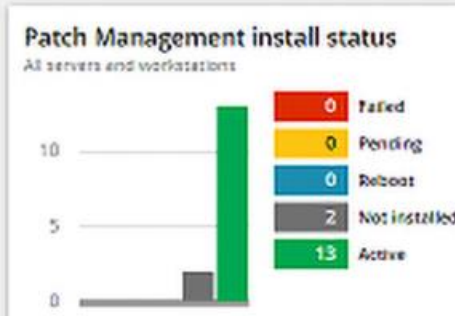
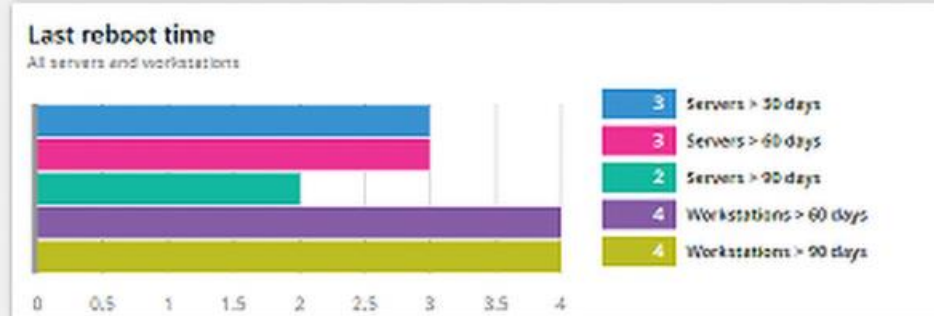
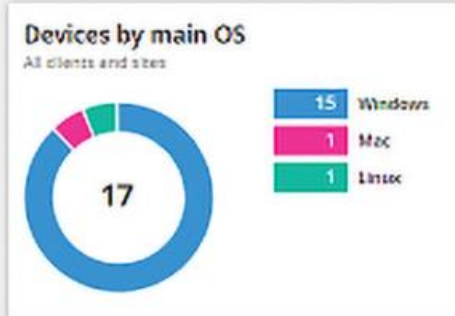
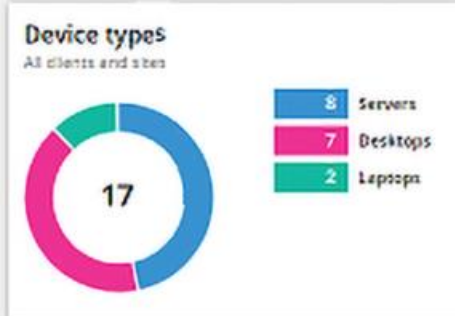
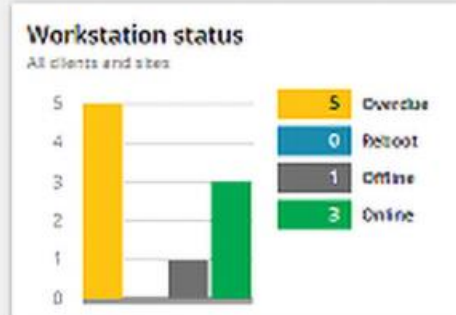
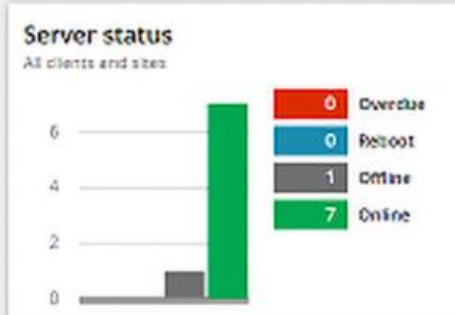


Monitoramento contínuo

- Visibilidade total e em tempo real das aplicações, desde a infraestrutura até as soluções entregues
 - Desempenho
 - Integridade
- Coleta de telemetria e metadados
 - Eventos de logs
- Definição de alertas
- Diminuir problemas em tempo real
- Encontrar possíveis melhorias

Dashboard: Monitoring

+ Add device | Refreshed 9:51:07 AM



Gerenciamento de configuração

- Gerenciar estado dos recursos
 - Servidores
 - Máquinas virtuais
 - Bancos de dados
 - Armazenamento
 - Redes
- Implementar alterações de maneira controlada e sistemática
- Reduzir riscos
- Acompanhar o estado do sistema e encontrar problemas
- Ajuda a operar ambientes complexos em escala
- Utilizada junto de IaC

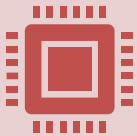
Infraestrutura como um Código



Infraestrutura como um código, do inglês *Infrastructure as a Code* (IaC) é o processo de gerenciar e provisionar recursos de TI a partir de arquivos com definições da infraestrutura (código)



Podem ser utilizados scripts que provisionam recursos ou então arquivos declarativos, que substituem o processo manual



A ideia de IaC ganhou força com a introdução da computação em nuvem

Infraestrutura como um Código

- Vantagens:
 - Reprodutibilidade
 - Versionamento
 - Diminuição de custos
 - Velocidade
 - Evitar erros do processo manual
 - Diminui tempo que o sistema fica fora
 - Aumenta confiabilidade

CloudFormation

- Serviço da AWS para provisionar serviços de TI na AWS de forma rápida e consistente utilizando IaC
- Forma fácil de provisionar recursos e serviços que estão relacionados
- Os recursos e suas dependências são declarados em um arquivo (template)
 - Tudo que está no template é provisionado e configurado junto em uma pilha (stack)
 - Criar, atualizar e remover a stack inteira, como uma unidade
- Não precisa provisionar e configurar recursos individualmente
 - CloudFormation toma conta de todo o processo

CloudFormation

- Por exemplo, provisionar uma aplicação de duas camadas:
 - API: EC2, ELB, AutoScaling
 - Banco de dados: RDS
- No template do CloudFormation são descritos os recursos que serão utilizados (EC2, ELB, AutoScaling, RDS) e como eles estão relacionados
 - O CloudFormation toma conta do resto

Anatomia do template

- Um template é um arquivo de texto formatado
 - JSON
 - YAML
- No template todas as características do recurso são descritas
 - Nome do bucket
 - VPC
 - Tamanho da instância, etc
- CloudFormation utiliza o template como um diagrama para provisionar os recursos
- Vários recursos podem ser provisionados no mesmo template
- O template pode ser alterado utilizado para atualizar os recursos

Anatomia do template - YAML

- YAML é uma linguagem para serialização de dados
- Compreensível por humanos e computadores
- Utilizada para criar arquivos de configuração
- Tem uma estrutura simples de chave: valor
 - Suporta lista
 - Suporta objetos aninhados
 - Suporta JSON
- Chave e valor são separados por dois pontos e espaço

```
nome: Fernando Sousa  
disciplina: DataOps
```


Anatomia do template

- Seções do template CloudFormation

AWSTemplateFormatVersion: "version date"

Description:

String

Metadata:

template metadata

Parameters:

set of parameters

Rules:

set of rules

Mappings:

set of mappings

Conditions:

set of conditions

Transform:

set of transforms

Resources:

set of resources

Outputs:

set of outputs

Anatomia do template - YAML

AWSTemplateFormatVersion

- Opcional
- Versão do template do CloudFormation
- Versão atual: 2010-09-09

Description

- Opcional
- Texto que descreve o template
- Deve vir após a versão do template

Metadata

- Opcional
- Informações adicionais do template

Anatomia do template - YAML

Parameters

- Opcional
- Valores enviados para o template em tempo de execução

Rules

- Opcional
- Validar parâmetros durante a execução da stack

Mappings

- Opcional
- Mapeamento de chaves e valores para especificar valores de parâmetros condicionais

Conditions

- Opcional
- Condições para controlar se um recurso será criado ou não, dependendo do ambiente

Anatomia do template - YAML

Transform

- Opcional
- Utilização de macros para processamento customizado do template

Resources

- Único elemento obrigatório
- Especifica os recursos e propriedade que serão provisionados, como uma instância EC2 ou um bucket S3

Outputs

- Opcional
- Valores que serão retornados na finalização da execução da Stack, como o nome de um bucket

Anatomia do template - YAML

- Especificação de Resources

- LogicalID:

- Identificador do recurso
 - Dever ser único no template
 - Deve conter apenas letras e números
 - Utilizado para referenciar recursos em outras partes do template

- Type:

- Tipo do recurso que será provisionado
 - Lista de recursos:

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>

- Properties:

- Propriedades do recurso
 - Depende do tipo de recurso que será provisionado

Resources:

LogicalID:

Type: Resource type

Properties:

Set of properties

CloudFormation – Exemplo Prático

- Provisionar um bucket S3

`AWSTemplateFormatVersion: "2010-09-09"`

`Description:`

`Criacao de bucket S3`

`Resources:`

`S3Bucket:`

`Type: AWS::S3::Bucket`

`Properties:`

`BucketName: mytestbucket`

- Repare na identificação dos elementos internos, como S3Bucket
 - A identificação é obrigatória
 - Cada elemento interno deve ser identificado para a correta interpretação do template

Funções Internas

- Funções implementadas pelo CloudFormation para ajudar no gerenciamento e configuração da pilha e do template
 - Referenciar recursos criados no template
 - Deixar algumas configurações dinâmicas
 - Criar recursos condicionalmente

Funções Internas - !Ref

- Referencia o LogicalID de um recurso criado no template
- Retorna um valor referente ao recurso referenciado
 - Em geral, o nome do recurso
- Por exemplo, colocar o nome do bucket criado como saída da pilha

AWSTemplateFormatVersion: "2010-09-09"

Description:

Criacao de bucket S3

Resources:

S3Bucket:

Type: AWS::S3::Bucket

Properties:

BucketName: bucket-fernandosousa

Outputs:

NomeBucket:

Description: Nome do bucket

Value: **!Ref S3Bucket**

Funções Internas - !Ref

- Saída no CloudFormation

| | | | | | | |
|---|---------|----------------------|--------|----------------|--------|-------------------------|
| Informações da pilha | Eventos | Recursos | Saídas | Parâmetros | Modelo | Conjuntos de alterações |
| Saídas (1) | | | | | | |
| <div><div>Q Saídas de pesquisa</div><div></div></div> | | | | | | |
| Chave | ▲ | Valor | ▼ | Descrição | ▼ | Nome da exportação |
| NomeBucket | | bucket-fernandosousa | | Nome do bucket | | - |

Funções Internas - !Sub

- Substitui variáveis por valores especificados no template ou valores internos
- Por exemplo, colocar no nome do bucket um valor que está no parâmetro SufixoBucket

```
AWSTemplateFormatVersion: "2010-09-09"
```

```
Parameters:
```

```
  SufixoBucket:
```

```
    Type: String
```

```
    Default: fernandosousa
```

```
Resources:
```

```
  S3Bucket:
```

```
    Type: AWS::S3::Bucket
```

```
    Properties:
```

```
      AccessControl: Private
```

```
      BucketName: !Sub bucket-${SufixoBucket}
```

Funções Internas - !Sub

- Também é possível usar variáveis padrões do CloudFormation, como o ID da conta (AWS::AccountId) e a região (AWS::Region)

```
AWSTemplateFormatVersion: "2010-09-09"
```

```
Parameters:
```

```
  SufixoBucket:
```

```
    Type: String
```

```
    Default: fernandosousa
```

```
Resources:
```

```
  S3Bucket:
```

```
    Type: AWS::S3::Bucket
```

```
    Properties:
```

```
      AccessControl: Private
```

```
      BucketName: !Sub bucket-${SufixoBucket}-${AWS::AccountId}-${AWS::Region}
```

Funções Internas - !Join

- Juntar um conjunto de valores separados por um caractere
- Por exemplo, mostrar o ARN (Amazon Resource Name) do bucket criado
 - Junta "arn:aws:s3:::" e !Ref 'S3Bucket' separados por "" (nenhum separador)

```
AWSTemplateFormatVersion: "2010-09-09"

Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      AccessControl: Private
      BucketName: !Sub bucket-fernandosousa

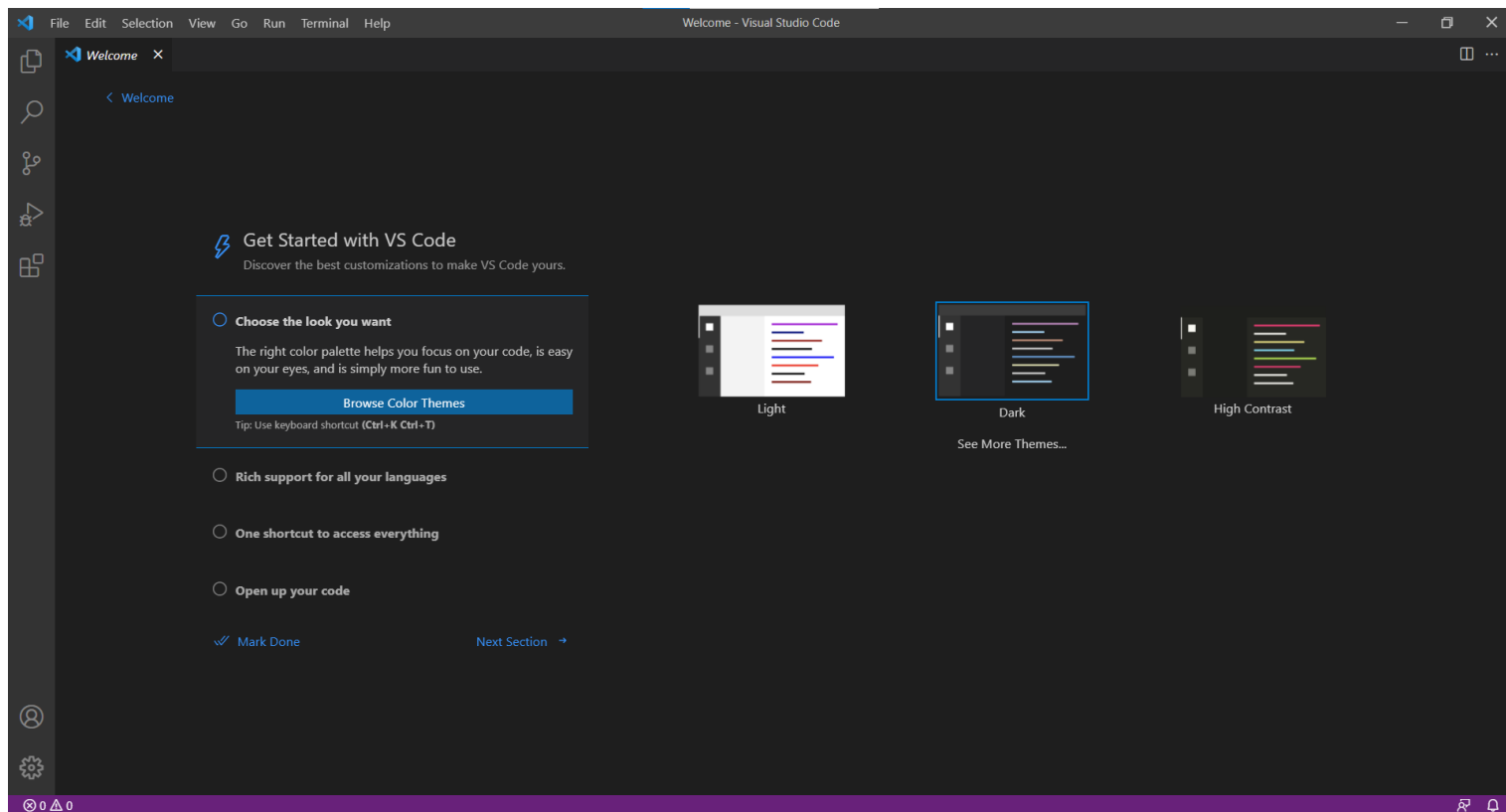
Outputs:
  ArnBucket:
    Description: Nome do bucket
    Value:
      !Join
      - ""
      - - "arn:aws:s3:::"
        - !Ref 'S3Bucket'
```

Ambiente de Dev Local

- Na aula de hoje vamos configurar o ambiente de desenvolvimento nos computadores pessoais
- Para desenvolver e abrir os códigos da disciplina vamos utilizar a IDE VSCode
- Faça o download da versão apropriada para seu ambiente em <https://code.visualstudio.com/#alt-downloads>

IDE - VSCode

- Após baixar/instalar o VSCode, abra a IDE

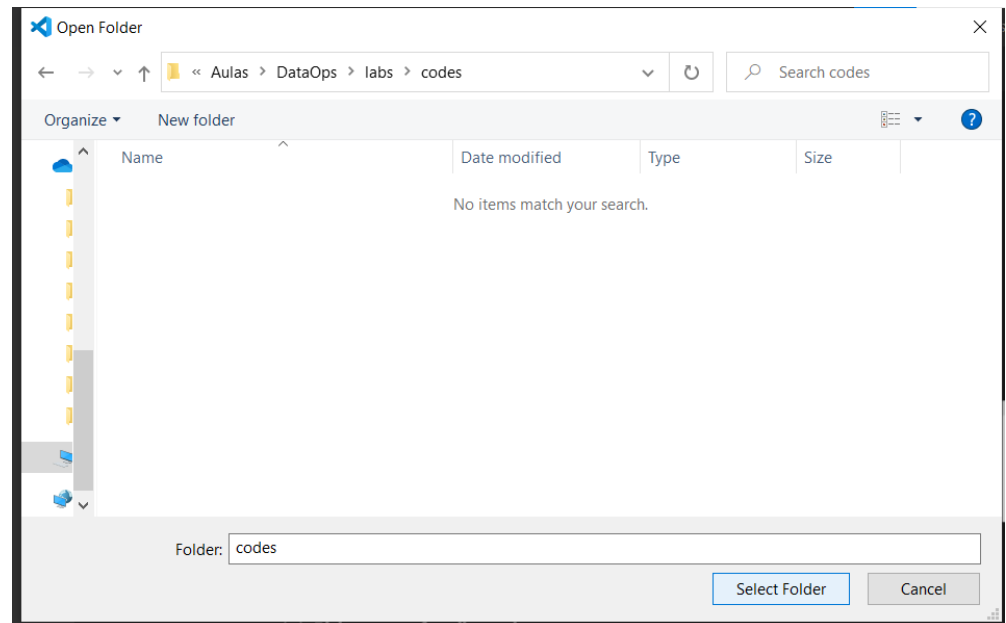
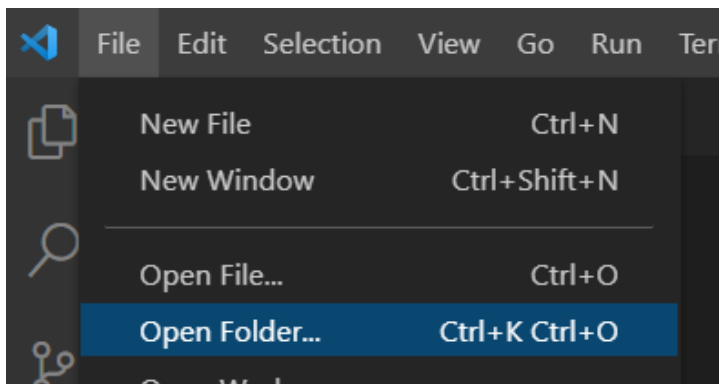


IDE – VSCode – Pasta do Projeto


- O primeiro passo é escolher uma pasta onde vai colocar os arquivos da aula
 - Use sempre a mesma pasta durante as aulas para não se perder
 - Vamos criar algumas subpastas para organizar os laboratórios durante o curso
 - Se já estiver habituado com o VSCode, pode organizar da forma que preferir

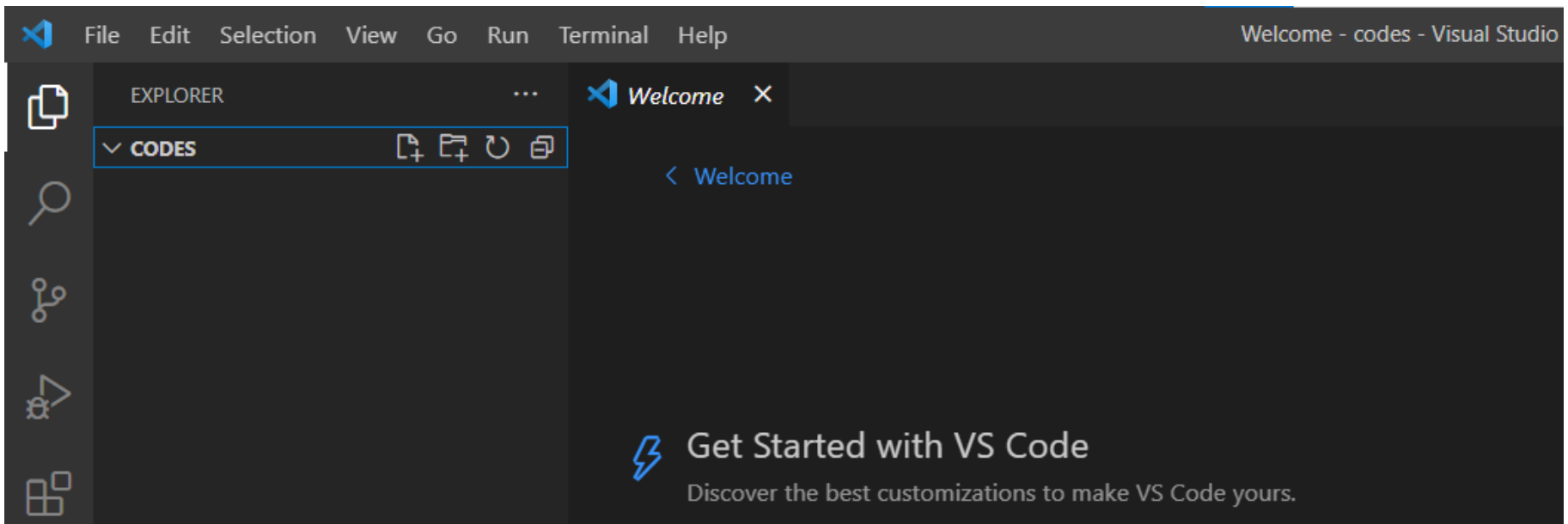
IDE – VSCode – Pasta do Projeto

- Clique no menu File → Open Folder
- Navegue até a pasta onde quer colocar seus arquivos
- Crie uma nova pasta
 - Em aula usaremos a pasta codes
- Selecione essa pasta criada




IDE – VSCode – Pasta do Projeto


- A visualização “Explorer” aparecerá ao lado esquerdo, mostrando o conteúdo da pasta (ainda vazio)
- O “Explorer” pode ser mostrado/escondido clicando na opção  ao lado esquerdo

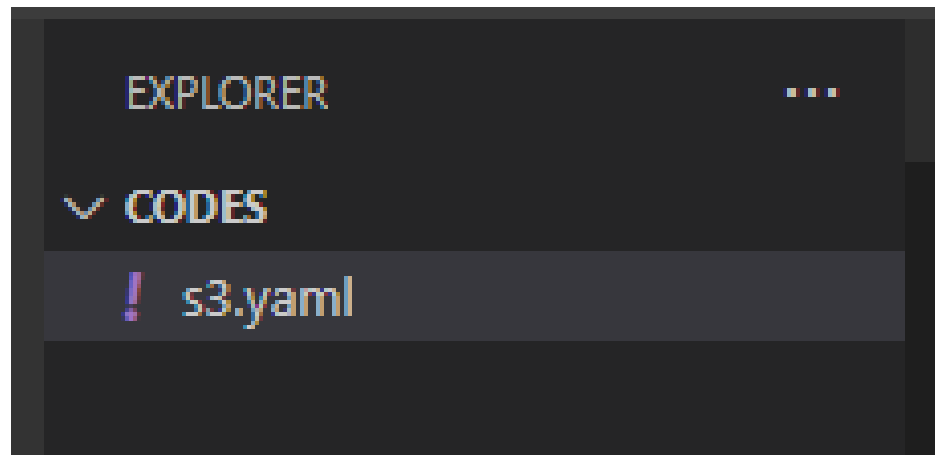


IDE – VSCode – Extensões

- Instale algumas extensões que vão facilitar a leitura e desenvolvimento dos códigos
- Clique na opção “Extensions”  ao lado esquerdo
- Procure e instale as seguintes extensões:
 - Python
 - CloudFormation

IDE – VSCode – Criar arquivo

- Para criar um arquivo na pasta clique no primeiro ícone ao lado do nome da pasta ( - New File)
- Crie um arquivo com o nome s3.yaml
 - O arquivo será aberto automaticamente



Exercício prático

- Reveja os exemplos do CloudFormation de aula e escreva um template no arquivo s3.yaml para:
 - Criar um bucket com o nome: deploy-nomesobrenome-accountID-regiao
 - nomesobrenome deve ser um parâmetro (Parameters) com seu nome e sobrenome
 - accountID é o ID da conta AWS
 - regiao deve ser a região na qual o bucket será criado
 - Deve mostrar na saída da stack o nome e o arn do bucket
- Explore o serviço CloudFormation para tentar implantar este template. Veja algumas dicas na descrição do laboratório
- Documentação:
 - https://docs.aws.amazon.com/pt_br/AWSCloudFormation/latest/UserGuide/aws-properties-s3-bucket.html
- Utilize este mesmo template para dar continuidade no laboratório

Exercício prático – Sugestão de resposta

- <https://github.com/fesousa/dataops-lab2/blob/master/code/template-s3-aula.yaml>

Lab 2

- Veja o documento do laboratório guiado no google classroom
 - DataOps - Lab 2 – CloudFormation
 - Responda ao questionário na atividade

Obrigado!

Prof. MSc. Fernando Sousa

fernando.sousa@faculdadeimpacta.com.br