




Machine Learning em Produção

S01E02

Rodrigo Ekstein

Agenda

- 
- **Recap da última Aula**
 - **Notebook é o fim ?**
 - **Formato YAML x Formato JSON**
 - **Infraestrutura como código**
 - **Diferentes tipos de IAC**
 - **Hands-on**
 - **Docker**

Notebook é o Fim




Quando terminamos de desenvolver nosso modelo em um jupyter notebook , é o fim do nosso desenvolvimento ?

O que devemos fazer para chegarmos na etapa de disponibilização do nosso modelo ?

Como saímos de um Jupyter Notebook e damos o próximo passo ?

Formato YAML



→ Formato de arquivos utilizados para serialização/codificação de alguma informação e/ou dados de forma legível ao ser humano.

→ Este formato de arquivo foi inspirado em uma antiga e conhecida linguagem de marcação/serialização, o **XML**.

→ Um dos principais pontos que levaram a criação deste formato de arquivos é sua legibilidade.


→ sintaxe simples e de fácil entendimento, o YAML torna-se uma alternativa acessível e tecnicamente de fácil implementação.

→ Estrutura baseada em indentação.

→ Mais legível que JSON e XML.

→ Diversas linguagens se beneficiam deste padrão de arquivo.

Formato YAML



Exemplo de YAML

→ Podemos escrever da forma que quisermos respeitando as regras do formato

→ Criamos os atributos necessários para cada elemento

→ Criamos vínculos de Pessoas com amigos

```
1  pessoa:
2      nome: Fulano
3      sobrenome: Cicrano
4      idade: 45
5      data_nascimento: 1974-05-13 09:25:55
6      masculino: true
7      # listas
8      hobbies:
9          - 'andar de bicicleta'
10         - patinar
11         - nadar
12     # Os membros da lista são colocados entre colchetes e separados por vírgulas.
13     amigos:
14         - nome: "João"
15           idade: 25
16         - nome: "Frederico"
17           idade: 26
18     # Para que o YAML leia mais de uma linha é necessário adicionar o símbolo (>)
19     descricao: >
20         Aqui tem algumas características do fulano,
21         ele possui 45 anos e tem alguns amigos.
22         Ele adora esportes.
23
```

Formato YAML



Comparando a legibilidade JSON x YAML

```
1  pessoa:
2    nome: Fulano
3    sobrenome: Cicrano
4    idade: 45
5    data_nascimento: 1974-05-13 09:25:55
6    masculino: true
7    # listas
8    hobbies:
9      - 'andar de bicicleta'
10     - patinar
11     - nadar
12     # Os membros da lista são colocados entre colchetes e separados por vírgulas.
13   amigos:
14     - nome: "João"
15       idade: 25
16     - nome: "Frederico"
17       idade: 26
18   # Para que o YAML leia mais de uma linha é necessário adicionar o símbolo (>)
19   descricao: >
20     Aqui tem algumas características do fulano,
21     ele possui 45 anos e tem alguns amigos.
22     Ele adora esportes.
23
```

```
1  {
2    "funcionario": {
3      "nome": "João",
4      "idade": 30,
5      "sexo": "Masculino",
6      "profissao": "Programador",
7      "dependente": {
8        "nome": "Maria",
9        "sexo": "Feminino",
10       "comentario": "Atribuicao de dependente"
11      },
12     "endereco": {
13       "rua": "Rua dos abacateiros",
14       "numero": "999",
15       "comentario": "atribuicao de endereco"
16     }
17   }
18 }
```

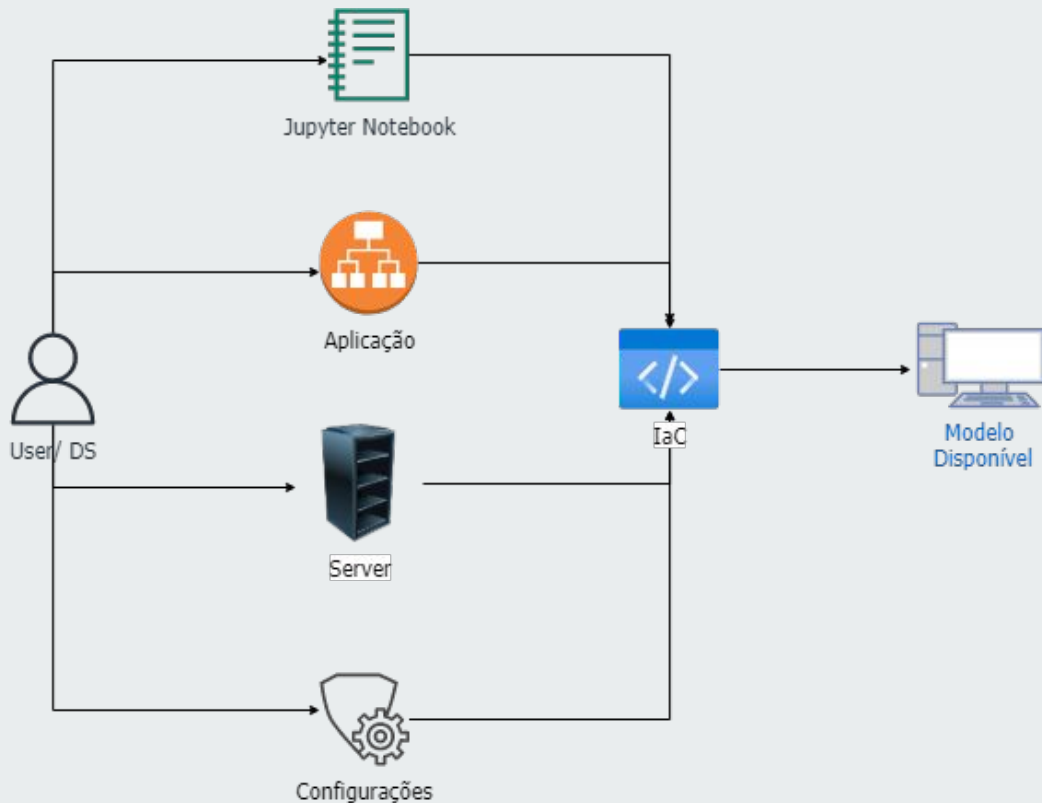
Infraestrutura como Código



→ No diagrama ao lado temos o cenário em que o cientista/desenvolvedor coordena diferentes recursos durante o desenvolvimento do modelo.

→ Configurações do modelo, jupyter notebook, Servidores de arquivo, etc.

→ Devemos refletir todos esses steps através de um código que represente a infraestrutura.



Diferentes tipos de IaC



Cloud Formation

→ Solução de IaC da AWS

→ Baseada no Formato YAML

→ Provisionamento dos recursos através de arquivos YAML

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  Parametro1:
    Type: String
  Parametro2:
    Type: String
  Parametro3:
    Type: String
Resources:
  Model:
    Type: AWS::SageMaker::Model
    Properties:
      ExecutionRoleArn:
        Ref: Parametro1
      PrimaryContainer:
        Image:
          Ref: Image
        Mode: "SingleModel"
        ModelDataUrl:
          Ref: Parametro2
  EndpointConfig:
    Type: AWS::SageMaker::EndpointConfig
    Properties:
      ProductionVariants:
        - InitialInstanceCount:
            Ref: Parametro1
          InstanceType:
            Ref: Parametro2
          ModelName:
            Fn::GetAtt: Parametro3
          VariantName:
            Ref: Project
          InitialVariantWeight: 1
  Endpoint:
    Type: AWS::SageMaker::Endpoint
    Properties:
      EndpointConfigName: !GetAtt EndpointConfig.EndpointConfigName
```



Sagemaker
Model



Sagemaker
Model



Sagemaker
EndpointConfig

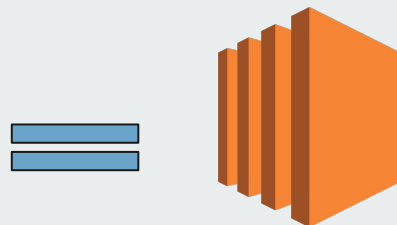
Diferentes tipos de IaC



Terraform

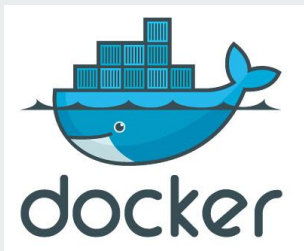
- Solução de IaC da empresa Hashicorp
- Baseada no Formato HCL
- Provisionamento dos recursos através de arquivos .tf
- open source

```
provider "aws" {  
  region = "us-west-2"  
}  
  
resource "aws_instance" "exemplo" {  
  ami = "ami-0021005f0cbcd6aaa"  
  instance_type = "t2.micro"  
}
```



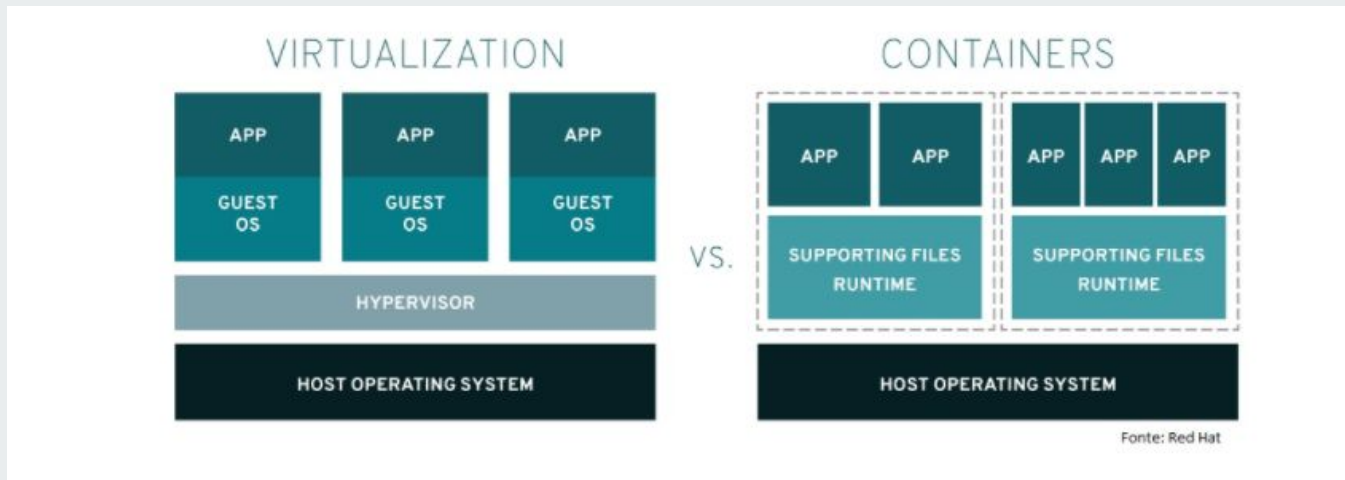
Instância EC2

Docker



- Plataforma Open Source
- Ambientes Isolados
- Facilita Administração e Criação de Novos Ambientes
- Portabilidade
- Otimização de Recursos

Docker x Maquina Virtual



→ Maior consumo de recursos para virtualização
Toda aplicação carrega um sistema operacional separado.

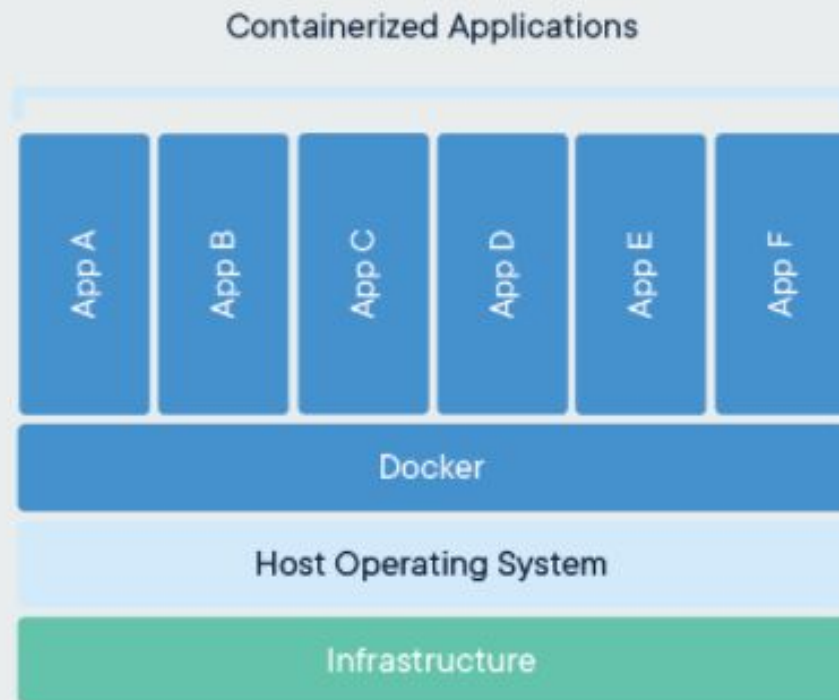
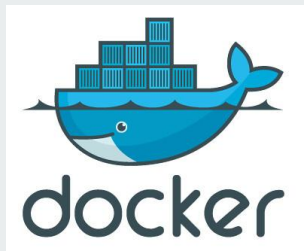
→ Difícil replicação entre ambientes

→ Não há necessidade de vários SO's.

→ Otimização de Recursos

→ Isolamento de Processos

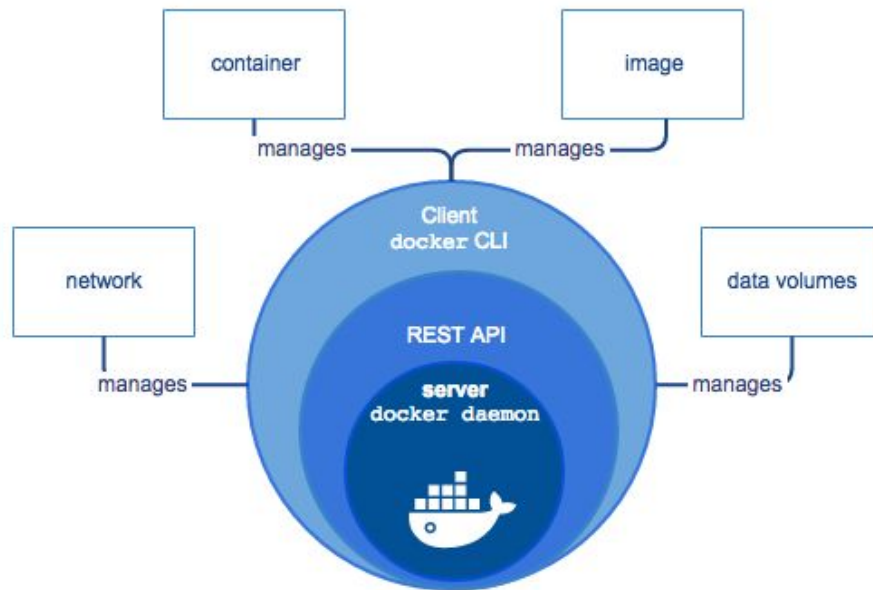
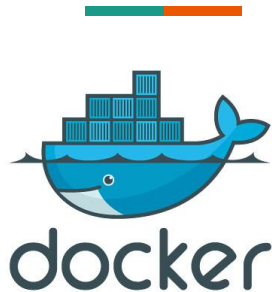
Afinal, o que são Containers ?



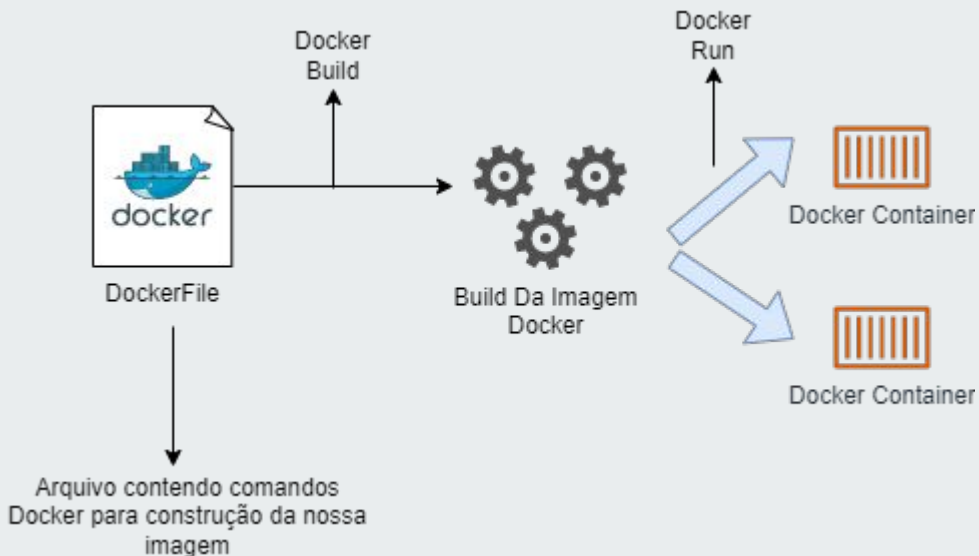
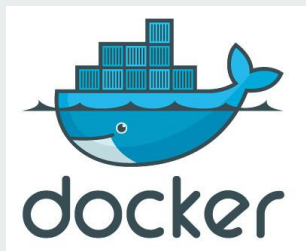
Container pode ser definido como um ambiente isolado, que contém um conjunto de processos que são executados a partir de uma imagem .

Tal imagem fornece todos os arquivos necessários .

Afinal, o que são Containers ?



Imagens



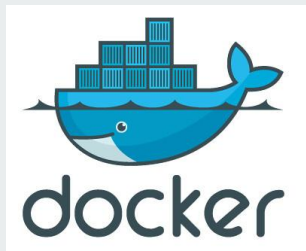
→ Imagem é um arquivo

→ Container é um processo

→ Ao invés de rodar um único arquivo binário, o container executa uma imagem.

→ Uma imagem é um pacote com um sistema de arquivos que contém todas as dependências necessárias.

DockerFile

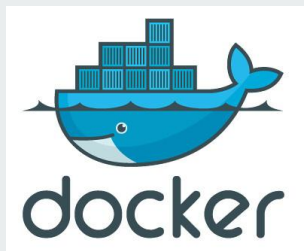


DockerFile

```
FROM ubuntu:18.04  
RUN apt-get update  
RUN apt-get install openjdk-8-jdk -y
```

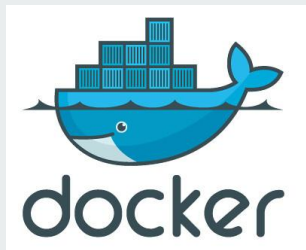
Serve como receita para construirmos nosso container e nos dá liberdade para personalizarmos da forma como julgarmos necessário.

DockerFile - Principais Comandos



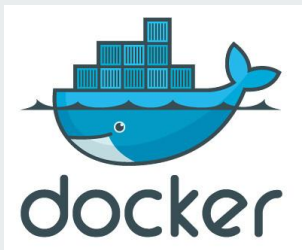
FROM	<p>Instrução obrigatória, serve como ponto de partida da nossa imagem.</p> <p>Caso eu queira utilizar uma imagem baseada em python para o meu container, basta que eu especifique para utilizar a imagem python:3.7 por exemplo</p>
RUN	<p>Instrução que pode ser executada várias vezes em nosso dockerfile, com ela, é possível definir quais os comandos a serem executados na etapa de criação das camadas.</p> <p>Cada camada pode ser reutilizada em outras imagens.</p>
CMD	<p>Parecido com o comando RUN porém ela só executa quando criamos o container e não passamos nenhuma instrução para ele.</p> <p>e caso tenhamos mais de um comando CMD no dockerfile, somente o último é executado</p>

DockerFile - Principais Comandos



ENTRYPOINT	Faz exatamente o que o CMD faz porém seus parâmetros não são sobrescritos.
ADD	Utilizado para realizar cópia de arquivos, ou até mesmo o download de arquivo contido em uma URL específica
COPY	Diferente do ADD, o comando COPY apenas transfere e copia o arquivo, não realiza downloads.

DockerHub



- Repositório centralizado para imagens Docker
- Disponibilização da sua imagem para comunidade
- Compartilhamento e gerenciamento de imagens