

# Desenvolvimento Contínuo para Aprendizagem de Máquina

---

Advanced Institute for Artificial Intelligence – AI2

<https://advancedinstitute.ai>

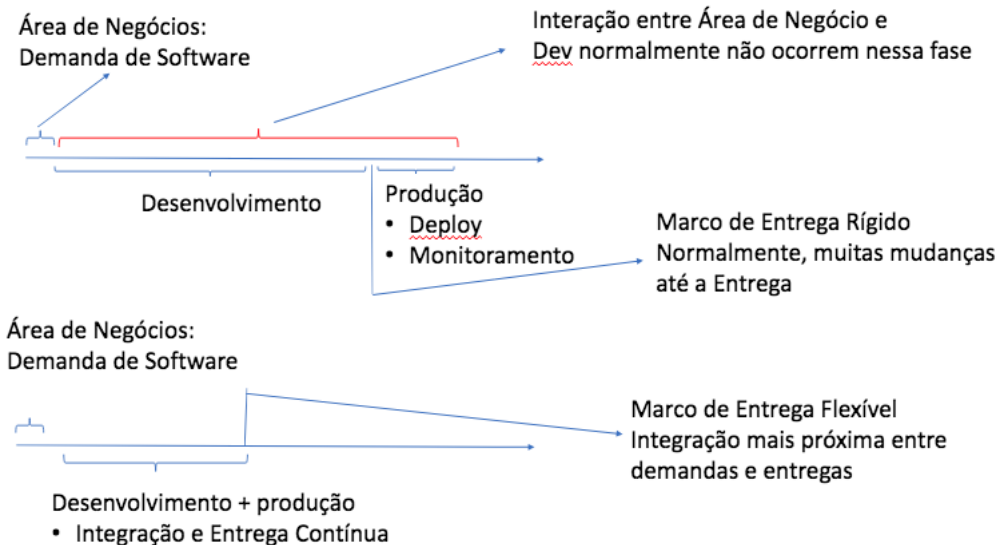
## Agenda

- ☐ Integração e Entrega Contínua
- ☐ Fluxo de Trabalho de Aplicações de Aprendizagem de Máquina
- ☐ Introdução ao Modelo de Microserviços
- ☐ Container

# Desenvolvimento Contínuo

- ❑ Integração contínua (CI): desenvolvedor continuamente integra seu código (normalmente uma ramificação do código principal) com o código principal
  - Controle de versão
  - Build automatizado
  - Check in frequente
  - Suíte de testes abrangente
- ❑ Entrega contínua (CD): processo automatizado que permite colocar uma nova versão do código principal em produção e desfazer o processo caso ocorra problemas, de forma simples
  - Colocar em produção automaticamente
  - Automação de uso de containers
  - Integração contínua é um pré-requisito
  - Normalmente um processo semi-automático
- ❑ Montagem (Deploy) Contínua: todo código que entra na master automaticamente é colocado em produção

# Desenvolvimento Contínuo



Integracao e Entrega Contínua representam uma flexibilizacao de entrega de software em marcos pré-definidos

- ☐ Maior sinergia entre demandantes de software e processo de desenvolvimento de software
- ☐ Desenvolvimento de software pode ser feito de modo mais granular
- ☐ Cada nova funcionalidade pode ser desenvolvida e implantada sem impactar o sistema como um todo
- ☐ Permite melhorar o atendimento de expectativas da área de negócios, que pode ter novas funcionalidades disponíveis em curtos espaços de tempo
- ☐ Processo auto controlado, que permite organizar o trabalho, mantendo a liberdade de cada colaborar de utilizar suas próprias ferramentas

Como alcançar tais níveis de automação?

- ☐ Profissional TI dedicado a colocar sistemas diversos em produção
  - Pouco ou nenhum conhecimento da estrutura de tais sistemas
  - Desenvolvedores do sistema possuem pouco ou nenhum conhecimento quanto a montagem de tais sistemas
- ☐ Barreira entre desenvolvimento e produção
  - Dependencias normalmente desconhecidas
  - Complexidade das etapas desconhecidas
- ☐ Integração/Entrega contínua é uma forma de quebra tal barreira
  - Colocar um sistema em producao, ou uma parte dele, é um processo conhecido pelo desenvolvedor e TI
  - Processo claro, planejável e conhecido usando especificações comuns

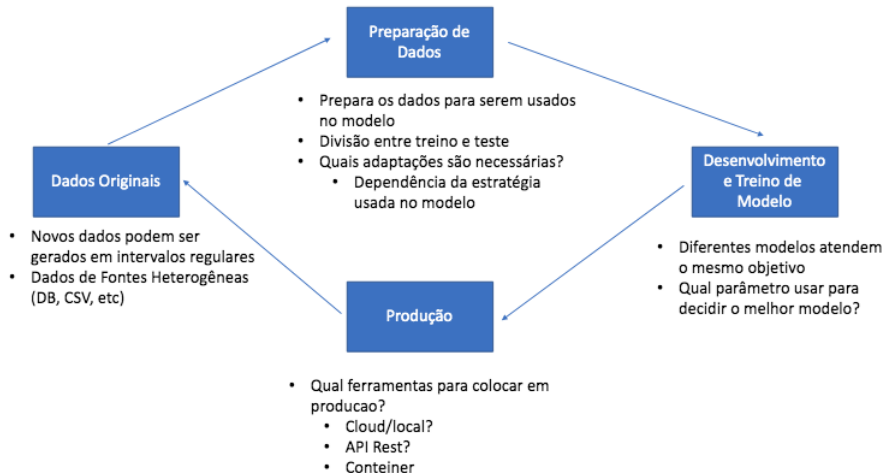
- ❑ OPs Contração que mostra unificação de áreas com a parte operacional de colocar em produção
- ❑ DEVOPs: prática de software que unifica o desenvolvimento de software (Dev) e a operação de software (Ops)
- ❑ Tecnologias diversas como Git, Jenkins, tecnologias de teste automatizados, usando por equipes compostas por desenvolvedores e TI
- ❑ As mesmas práticas de DEVOps tem sido também se popularizado para desenvolvimento de sistemas de aprendizagem de máquinas:
  - CD4ML
  - DATAOPs
  - MLOPs

Sistemas de aprendizagem de máquina (Machine Learning), são tipos de sistemas que seguem uma estrutura clara

- ☐ Organização dos dados de entrada
- ☐ Visualização dos dados
- ☐ Desenvolvimento de um modelo
- ☐ Avaliação do modelo
- ☐ Colocar modelo em produção
- ☐ Receber requisições para predição



# Desenvolvimento Contínuo



Melhorias contínuas podem ser feitas em aplicações de Aprendizagem de Máquina considerando diferentes aspectos:

- ☐ Organizar os dados usados para treino que são gerados diariamente
- ☐ Adaptar modelos para utilização de dados complementares
- ☐ Comparação entre modelos implementados de acordo com diferentes abordagens
- ☐ Implementar modelos que abordam o problema de outras maneiras
- ☐ Adaptações quanto a forma de integração do modelo com outras aplicações
- ☐ Adequações para melhorar o desempenho do treino e predição

Desenvolver software rápido e com qualidade é desafiador.

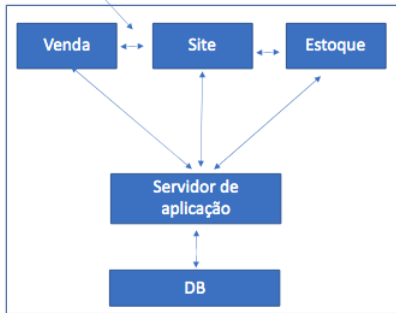
- O nível de maturidade da abordagem de entrega contínua exige:
  - Reuso de software
  - Abstração de ferramentas de software para integrar e colocar em produção uma funcionalidade
- Diversas iniciativas buscam atender tais desafios:
  - Divisão em múltiplas camadas
  - Componentes
  - Orientação a serviços (SOA)
  - Microserviços

Microserviço: abordagem para desenvolver sistemas

- ☐ Funcionalidades do sistema dividida em unidades independentes
- ☐ Unidades podem ser montadas de modo independente
- ☐ Acoplamento ocorre apenas por troca de mensagens

# Desenvolvimento Contínuo

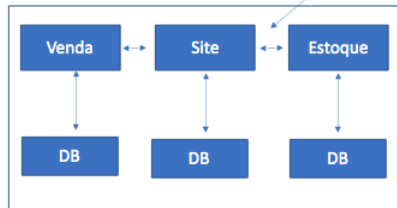
Comunicação via API  
(ferramenta específica)



**Monolito**

- Risco de Bug em módulo ser propagado
- Sistema inteiro deve ser montado após alteração de um módulo

Comunicação via web  
(Independência de ferramenta)



**Microserviço**

- Maior complexidade:
  - Desenvolvimento
  - Monitoramento
    - Sistema pode ser montado em diversos nós, distribuídos pela Internet

No contexto de aprendizagem de máquina, microserviços facilitam integração

- Modelos podem ser montados e integrados sem conhecimento da base ou regras de negócio
- O fluxo para treino de um modelo pode ser desenvolvido como funcionalidades independentes
  - Separação entre as etapas permite melhorar o modelo continuamente
  - Melhor forma de preparar os dados
  - Proposição de novos modelos
- Ferramentas para desenvolvimento de microserviços:
  - Flask biblioteca para Microserviço
  - Json protocolo para troca de dados

- ❑ O framework Flask permite criar servidores web seguindo a especificação WSGI (Web Server Gateway Interface), que é um padrão para o desenvolvimento de aplicações web em Python
- ❑ A idéia desse padrão é permitir a portabilidade de uma aplicação web em python entre diferentes servidores web atuando como um middleware
- ❑ Flask provê recursos para
  - Iniciar um servidor web
  - Capturar as chamadas para diferentes funcionalidades de acordo com as URLs passadas
  - Processar as requisições em um script python

Instalando flask

```
1  pip install flask
```



# Desenvolvimento Contínuo

Uma aplicação mínima em Flask precisa importar a class Flask, criar uma instância da classe flask e atender ao menos uma requisição web

O Routing define um formato de URL que deve ser capturado e direciona a chamada para uma função escrita em python

Para isso vamos salvar o código a seguir em um arquivo chamado flaskapp.py

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  app.route('/')
5  def hello_world():
6      return 'Hello, World!'
```

Para colocarmos o microserviço em produção é necessário executar dois passos:

Criar a variável de ambiente `FLASK_APP` e atribuir como valor o nome do arquivo criado (`flaskapp`)

```
1 export FLASK_APP=flaskapp
```

Executar o ambiente flask:

```
1 flask run
```

Isso indica que o servidor está no ar no local host e respondendo na porta TCP 5000. Por ser um servidor web, podemos realizar chamadas a ele por meio do navegador

O Código a seguir mostra como implementar uma requisição passando parâmetros

```
1  from flask import Flask
2  app = Flask(__name__)
3
4  app.route('/user/<username>')
5  def profile(username):
6      return username
```

- ❑ O parâmetro `@app.route()` define qual URL será tratada pelo servidor.
- ❑ Ao capturar essa URL o servidor direcionará a chamada a função declarada na próxima linhas após o `@app.route` que é a função `hello_world()`
- ❑ Essa função tem como objetivo retornar uma string com o valor "Hello, World!"
- ❑ Esse retorno é direcionado como resposta ao cliente (Nesse caso um navegador) que realizou a chamada ao servidor web, portanto vai mostrar na tela do navegador a mensagem : "Hello, World!"

- ❑ REST (Representational State Transfer): Um aplicativo Web RESTful expõe informações sobre si na forma de informações sobre seus recursos. Ele também permite que o cliente execute ações nesses recursos, como criar novos recursos (por exemplo, criar um novo usuário) ou alterar os recursos existentes (por exemplo, editar uma postagem).
- ❑ Um aplicativo REST é utilizado por meio de requisições HTTP e provê respostas do tipo HTML, XML ou Json
- ❑ É como requisitar uma URL no navegador e receber uma página HTML como resposta. A diferença é que a solicitação via REST é feita a uma aplicação e não para um arquivo estático
- ❑ Json (JavaScript Object Notation): é um formato bastante flexível para determinar a forma como as aplicações se comunicam.

O código abaixo modifica a função de buscar a idade do cliente para retornar um objeto json

```
1
2  from flask import jsonify
3
4  app.route('/busca/<username>')
5  def searchuser(username):
6      idade=0
7      with open('clientes.txt') as f:
8          for line in f:
9              l=line.split(';')
10             if (l[0] == username):
11                 idade=l[1]
12  return jsonify(cliente= username, idade= idade)
```

O Cliente pode realizar o parser das informações enviadas usando a lib json conforme o código abaixo:

```
1  import json
2
3  def buscarCliente():
4      nome = input("Digite o nome do cliente que deseja buscar: ")
5
6      contents = urllib.request.urlopen("http://127.0.0.1:5000/busca/"+nome).
          read()
7
8      print(contents)
9      cliente_idade = json.loads(contents)
10     print(cliente_idade['cliente'])
11     print(cliente_idade['idade'])
```

## Serving systems

- Ferramentas para colocar sistemas de aprendizagem de máquina em produção são normalmente chamados de serving systems
- A idéia do serving systems é criar uma interface de microserviços que receba uma requisição para uma predição e retorne a predição
- A utilização de microserviços é essencial para permitir maior flexibilidade quanto a desempenho, segurança, montagem e integração do módulo de predição em diversas aplicações



Um exemplo de Serving System em Tensorflow é o `tensorflow_model_server`

- ☐ Modelo em Keras é salvo para o disco
- ☐ Modelo é carregado por um sistema que recebe requisições REST para fazer predições

Outras ferramentas para automatizar o processo de desenvolvimento de aplicações de aprendizagem de máquina também são disponibilizadas pelo tensorflow