F1RST Tecnologia – Santander

DATA MASTERS CASE – ENGENHARIA DE MACHINE LEARNING:

DOCUMENTAÇÃO TÉCNICA E DE SOLUÇÃO

Treinamento e Deploy de Modelos de Deep Learning utilizando Databricks + Spark + TensorFlow + MLFlow em ambiente *Cross Cloud* – Microsoft Azure e Google Cloud Plataform

Marcos Vinícius Lisboa Melo – T780496

São Paulo, 2022

1. INTRODUÇÃO

Esta documentação tem como objetivo contemplar técnica e funcionalmente a construção e implementação do Case para a Certificação interna **Data Masters** com referência a carreira de ***Machine Learning Engineer***. Nos capítulos a seguir estarão descritos, em detalhes, desde a motivação de uso das tecnologias mencionadas no título como também a construção técnica necessária para atingir o resultado pretendido de treinar modelos utilizando **TensorFlow** com a estratégia distribuída do **Spark** e realizar o *deploy* destes modelos em outro ***Workspace Databricks*** de um provedor de Cloud diferente, utilizando dos recursos de MLOps disponíveis no **MLFlow**. Também estarão listadas na antepenúltima seção propostas de melhoria para que efetivamente possa se incorporar o desenvolvido ao pipeline de Modelos do Santander.

Vale ressaltar que a leitura desta documentação também pode ser feita por meio do README disponível no repositório criado para este case, juntamente com todos os códigos fontes necessários: https://github.com/lisboavini/data_masters/.

2. MOTIVAÇÃO

O case segue, conforme a instrução de execução, os seguintes cenários de implementação:

A5-) Servir um modelo escolhido do Kaggle de Imagem;

B2-) Demonstre como seria o pipeline de implantação de um modelo implementado;

C1-) Demostre por que uma ferramenta de orquestração de modelo pode ajudar nas implantações de modelos;

D2-) Demonstre a importância dos pipelines de DEVOPs para a implantação de modelos.

A motivação deste case, se dá em dois pontos principais: **demonstrar e instrumentar o treinamento de modelos de Deep Learning, especificamente modelos de *Convolutional Neural Network* (CNN)** para Visão Computacional**, utilizando o TensorFlow (*spark-tensorflow-distributor)* alinhado com o recurso de distribuição de tarefas do Spark (*MirroredStrategyRunner)*** e em sequência **o *deploy* de modelos de ML utilizando o fluxo do MLFlow, *cross-workspaces* e *cross-cloud* –** reforçando a proposta de estarmos sempre agnósticos a um único provedor, **simulando o real fluxo de deploy do banco de um *workspace* de Sandbox para um *workspace* de esteira produtiva**. Além disso, **validar a possibilidade de**

fazer deploy dos mesmos modelos multi-cloud, possibilitando manter os serviços críticos de forma reduntante, neste case foram utilizados os seguintes provedores de Cloud: Microsoft Azure ® e Google Cloud Plataform ®.

A escolha destes provedores foi motivada pela disponibilidade de recursos, nestes foi possível, por meio de parceria, criar workspaces com alguns recursos (créditos) para execução dos clusters sem dificuldades. A ideia core deste case, é que, o desenvolvido aqui, seja facilmente acoplável aos pipelines existentes do banco, podendo ser um potencializador dos nossos negócios e ferramentas, principalmente retirando a necessidade de realizar uma implantação de MLFlow paralela, podendo aproveitar os recursos já disponíveis nativamente na plataforma. Importante destacar que este case possibilita validar uma possível utilização da Databricks no fluxo de vida completo de um modelo de Machine Learning.

3. ARQUITETURA DE SOLUÇÃO (*BluePrints*)

Como proposta de arquitetura de solução, tem-se uma arquitetura híbrida Azure-GCP. Em ambos os provedores é instanciando uma *workspace* Databricks com disponibilidade para criação de clusters conforme a necessidade. É proposto para solução a utilização da *workspace* Azure como Sandbox para treinamento e registro do modelo de CNN, utilizando clusters com recurso de GPU e a biblioteca do TensorFlow como auxiliadora para o uso de imagens/matrizes.

Já para a GCP, a *workspace* utilizada trata-se de um cluster sem GPU, pois não é necessário o recurso gráfico para otimização do deploy desta aplicação, considerando um cenário no qual a API Rest tem uma quantidade de requisições limitadas. Poderia ser criado um cluster com GPU caso fosse necessário que a aplicação respondesse com maior performance ou para um número maior de chamadas simultâneas. A Figura 1 mostra o desenho completo da solução *end-to-end*.

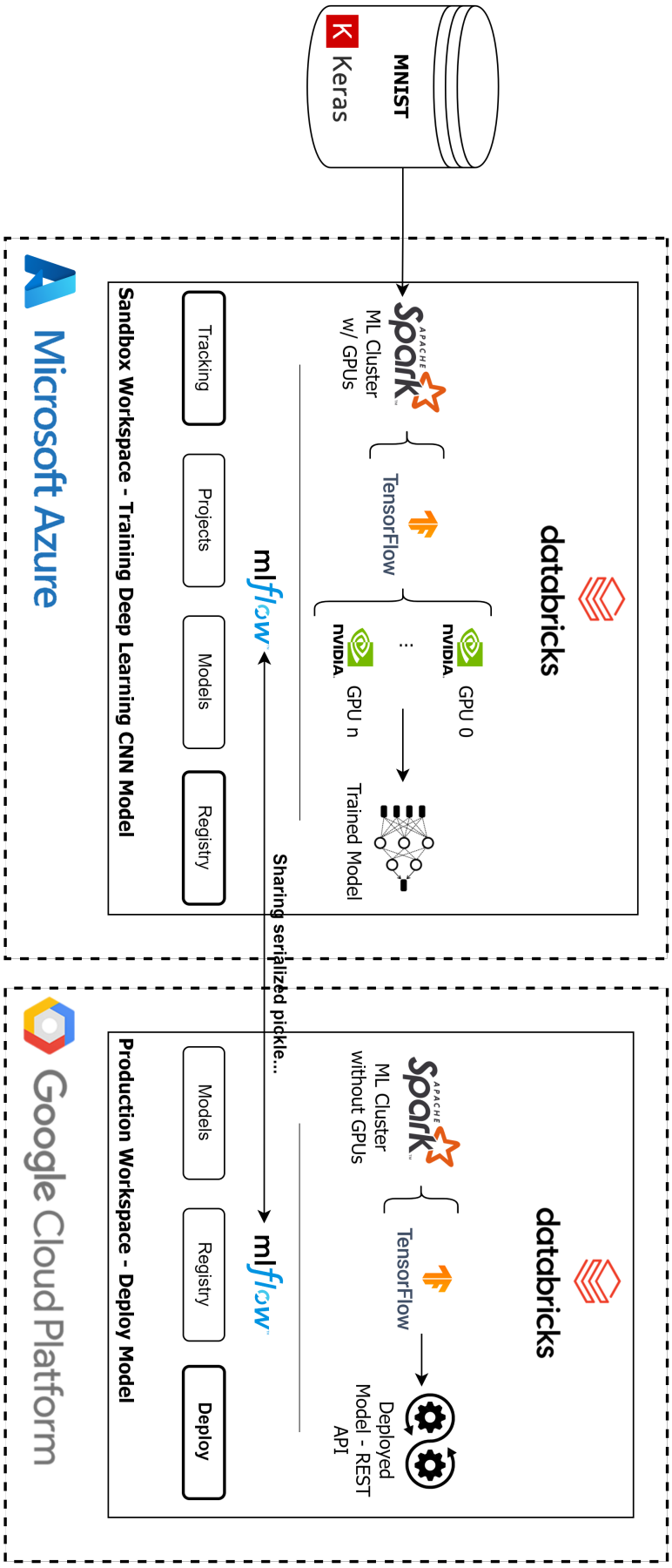


Figure 1 - BluePrint Solução

4. ARQUITETURA TÉCNICA/DADOS

Tecnicamente é possível destacar alguns pontos principais e fundamentais para que a solução proposta funcione de maneira correta e eficiente. Neste aspecto é importante destacar o modo de funcionamento específico dos três componentes-chaves propostos: ***spark-tensorflow-distributor, MirroredStrategyRunner*** e **MLFlow API.** Como também, as definições técnicas de Infra para cada ambiente cloud.

4.1 *spark-tensorflow-distributor*

O *spark-tensorflow-distributor* é um pacote native de TensorFlow de código aberto que auxilia os desenvolvedores de modelos de IA (cientistas de dados e engenheiros de ML) a distribuir o treinamento dos modelos TF em clusters Spark. Abaixo tem-se um desenho de arquitetura técnica que explica em mais detalhes o funcionamento do pacote.

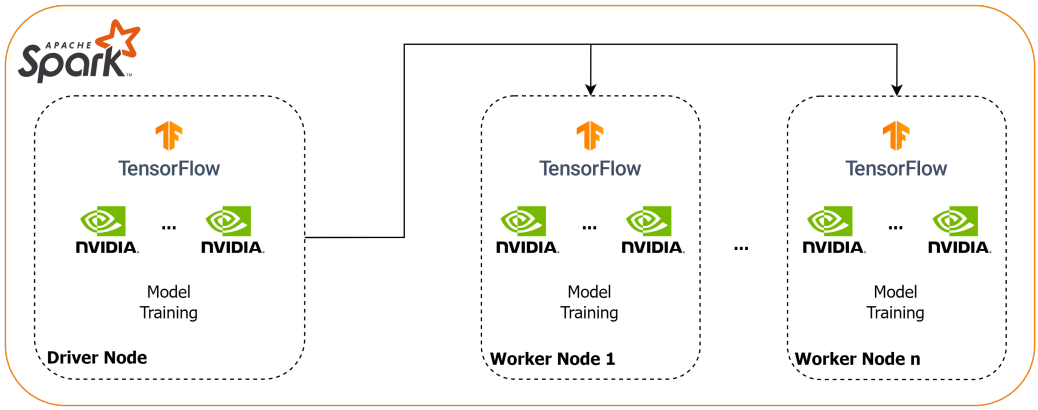


Figure 2 - Arquitetura Técnica TensorFlow Distributed

4.2 *MirroredStrategyRunner*

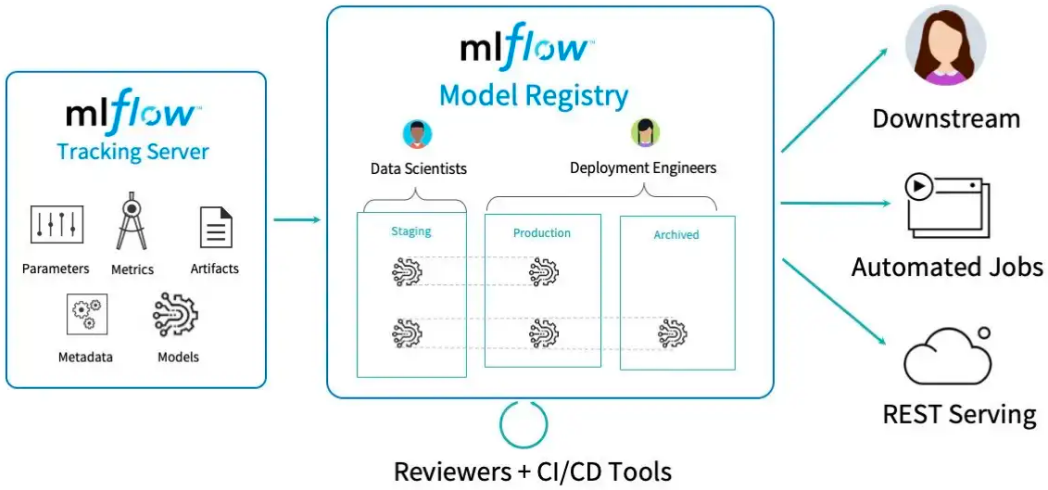
Dentro do pacote mencionado no tópico anterior é possível encontrar um método chamado *MirroredStrategyRunner()*, que dentro de si encapsula as funções necessárias para definir o número de slots, tipo de treino e quantidade de GPUs utilizadas. Ele tem a função embarcada de realizar as conexões e divisão de paralelismo entre os nodes, e é o principal componente do pacote de distribuição.

4.3 MLFLOW API - PYFUNC

O MLflow tem como intuito oferecer suporte ao ciclo de vida de projetos de ML, mas também disponibiliza uma API para diminuir alguns desafios comuns, como:

compartilhamento de artefatos, reprodutibilidade do modelo, etc. MLflow é uma ferramenta de ciclo de vida de aprendizado de máquina de código aberto, que facilita o gerenciamento do fluxo de trabalho para treinamento, rastreamento e produção de modelos de machine learning.

O MLFlow foi organizado para funcionar com as bibliotecas e estruturas de aprendizado de máquina mais recentes e utilizadas no mercado atualmente. Na imagem abaixo pode-se observar o funcionamento arquitetural como um todo do fluxo de MLOps.



O *flavour python_function* serve como uma interface de modelo padrão para modelos Python do MLflow. Espera-se que qualquer modelo MLflow Python seja carregável como um modelo *python_function*.

Além disso, o módulo *mlflow.pyfunc* define um formato de sistema de arquivos genérico para modelos Python e fornece utilitários para salvar e carregar desse formato. O formato é independente no sentido de que inclui todas as informações necessárias para que qualquer pessoa possa carregá-lo e usá-lo. As dependências são armazenadas diretamente com o modelo ou referenciadas por meio de um ambiente Conda.

O módulo *mlflow.pyfunc* também define utilitários para criar modelos *pyfunc* personalizados usando estruturas e lógica de inferência que podem não estar incluídas nativamente no MLflow.

4.4 INFRAESTRUTURA AZURE

Caso seja necessário realizar a criação de um Workspace Databricks em um recurso Azure, vide referência com passo-a-passo: https://learn.microsoft.com/en-us/azure/databricks/scenarios/quickstart-create-databricks-workspace-vnet-injection.

Foi definida para implementação em uma worksapce Databricks instanciada na Azure uma estrutura com dois clusters, um cluster de apoio para evitar o custo excessivo em tarefas triviais e um cluster de treinamento de modelos, munido de GPU Tesla. A tabela abaixo contém a descrição detalhada dos clusters e os correspondentes recursos criados.

Clusters

Name	Type	RAM	Cores	Workers	GPU
data_masters_nc12_2_gpu	Standard_NC6s_v3	112 GB	6	2-8	Tesla P100
data_masters_standard_4_cores	Standard_DS3_v2	14 GB	4	2-8	-

Para armazenamento do token de conexão entre workspaces foi criado escopo **data_masters** juntamente com a secret/key **data_master_sandbox**.

4.5 INFRAESTRUTURA GCP

Caso seja necessário realizar a criação de um Workspace Databricks de um recurso Google, vide referência com passo-a-passo: https://docs.gcp.databricks.com/administration-guide/account-settings-gcp/workspaces.html.

Para implementação na workspace Databricks GCP, apenas é necessário um único cluster para teste e *deploy* do modelo via MLFlow na forma de API Rest. Este cluster não necessita, obrigatoriamente, de GPU, o modelo pode ser deployado em um cluster apenas com CPU, e a depender da necessidade por velocidade e processamento da aplicação a ser utilizada, pode ser utilizado o recurso de placas gráficas para acelerar o desempenho.

Clusters

Name	Type	RAM	Cores	Workers	GPU
data_masters_e2_8_cores_deploy	e2-standard-8	32 GB	8	2-4	-

Importante salientar também que foi utilizado neste workspace o recurso de serving via API de modelos, que realiza o deploy utilizando o modelo registrado no MLFlow invocando as APIs padrão, neste caso em específico do TFX para deploy de um modelo Tensorflow.

Para armazenamento do token de conexão entre workspaces foi criado escopo **data_masters_gcp** juntamente com a secret/key **data_master_deploy**.

5. DATASET E ARQUITETURA CNN

Foi utilizado o conhecido dataset chamado MNIST - *Modified National Institute of Standards and Technology*, composto por cerca de 70.000 imagens de algarismos numéricos escritos a mão e pode ser utilizado de forma aberta para o desenvolvimento de modelos de reconhecimento de números utilizando técnicas de visão computacional.

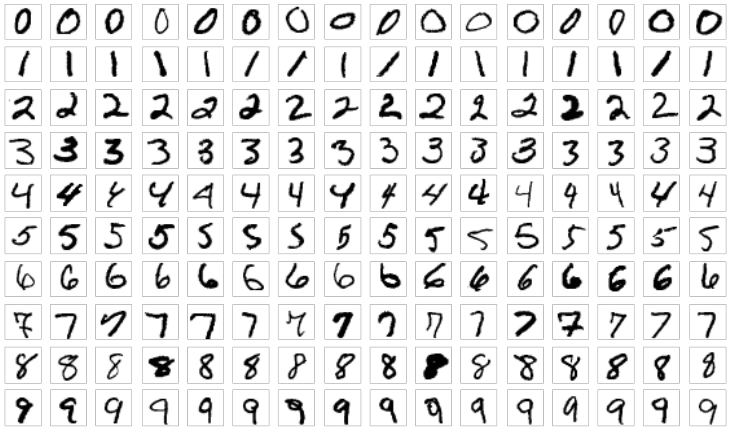


Figure 3 - Amostras do MNIST (https://en.wikipedia.org/wiki/MNIST_database)

O objetivo deste case em específico não consiste no treinamento de um modelo com a melhor acurácia possível, falhas são toleráveis se tratando da predição, logo, a arquitetura selecionada para a rede não foi de grande complexidade, uma vez que, apenas era necessário validar o treinamento de uma CNN, com n camadas ocultas.

Para tanto, foi utilizada uma rede com 3 camadas Conv2D, com ativação relu, e nas camadas de saída duas camadas densas, uma com ativação relu e a camada de classificação softmax para 10 classes.

O learning_rate setado foi de 0.001, utilizando um treino com 50 épocas e 100 steps por época. A métrica logada e avaliada foi apenas acurácia, uma vez que o enfoque não era a qualidade do modelo apresentado.

6. TREINAMENTO DATABRICKS AZURE

Foi construído um cenário de Sandbox dentro de uma *workspace* Databricks Azure, para a construção de modelos de Deep Learning. A escolha do provedor Azure neste cenário específico foi em decorrência dos recursos disponíveis neste ambiente, e a liberdade para o consumo de créditos, acordado previamente.

O cenário aqui implementado consistiu no consumo do MNIST, mencionado na sessão anterior, e utilizando-se uma Rede Convolucional (CNN) com poucas camadas (diminuindo o tempo e recursos para treinamento), para validar o cenário de implantação com um modulo funcional.

Para treinamento utilizando os recursos de GPU e distribuição de tarefas entre os workers existem 4 modos de treinamento conforme detalhados nos tópicos a seguir.

6.1 TREINAMENTO SINGLE NODE

O treinamento Single Node consiste na modalidade mais simples de treino, na qual o recurso é completamente alocado no driver, se esse possuir uma GPU a mesma pode ser utilizada para acelerar o treinamento, entretanto a paralelização de tarefas ocorrerá somente dentro das *threads* da GPU.

6.2 TREINAMENTO DISTRIBUIDO

O treinamento distríbuido permite a utilização mais de um nó para paralelização da execução das tarefas de treinamento, podendo multiplexar o treinamento de vários modelos simultaneamente, ou até mesmo de um único modelo cross-workers.

6.2.1 TREINAMENTO DISTRIBUIDO: LOCAL MODE

6.2.2 TREINAMENTO DISTRIBUIDO: DISTRIBUTED MODE

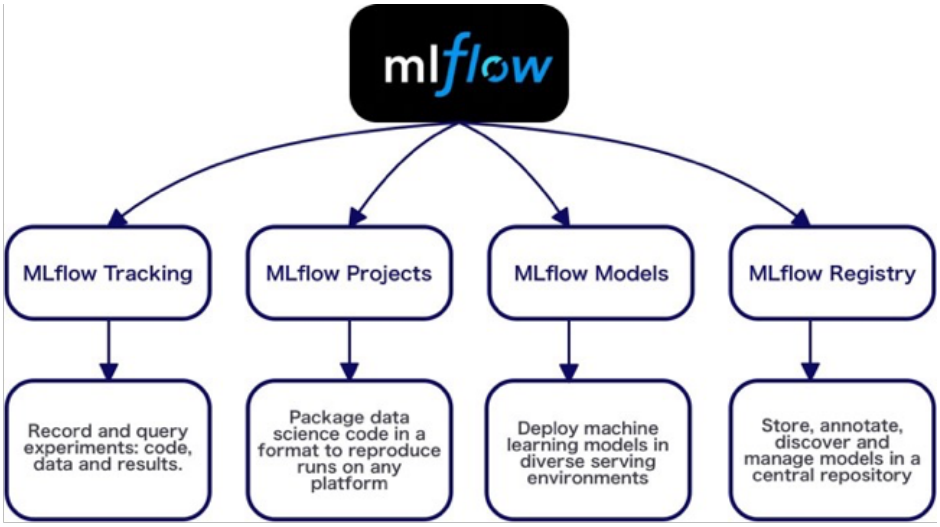
6.2.3 TREINAMENTO DISTRIBUIDO: CUSTOM MODE

7. MODEL LOGGING AND REGISTRY VIA MLFLOW

Nativamente embarcado a solução da Databricks existe um serviço de MLFlow gerenciado, este serviço contempla todas as funções já conhecidas do MLFlow e disponibiliza uma camada gráfica de UI na plataforma. Além disso, esse serviço nativamente é configurado para armazenar as informações pertinentes ao modelo dentro de um diretório no DBFS da workspace.

Podem ser encontradas informações detalhadas de como realizar cada uma das etapas de modelos dentro do ambiente Databricks no link: https://docs.databricks.com/mlflow/models.html.

É possível utilizar todas as funções disponíveis do MLFlow fazendo o import da biblioteca e utilizando os recursos de log, tracking, models e registry da ferramenta, conforme exemplificados na imagem abaixo.



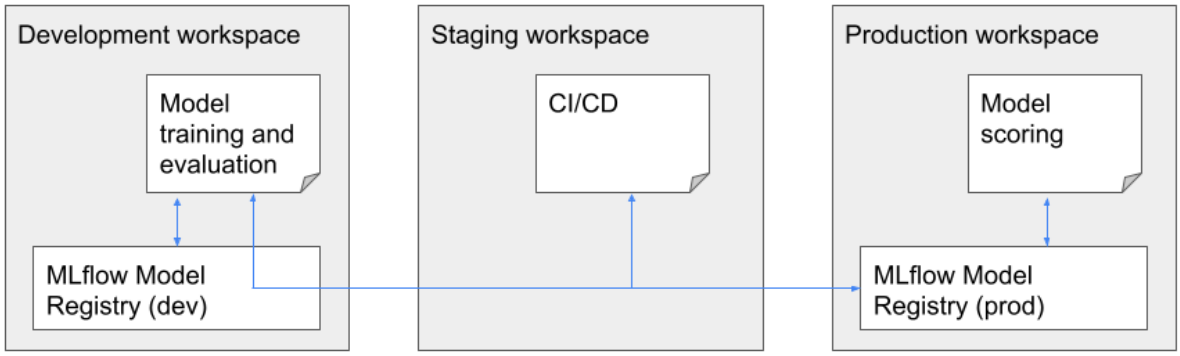
A abordagem aqui descrita fez uso do componente de autolog do MLFlow, já que o foco não era detalhar ao máximo os registros dentro do experimento, mas sim, validar todo o pipeline criado. E após isso realizar o registry do modelo, podendo ser diretamente pela UI ou através de comando em código. Detalhes são facilmente observados dentro dos notebooks de treino e deploy, os quais, respectivamente, logam e registram e após carregam e servem.

8. INTEGRAÇÃO VIA MANAGED MLFLOW

Nativamente embarcado a solução da Databricks existe um serviço de MLFlow gerenciado, este serviço contempla todas as funções já conhecidas do MLFlow e disponibiliza uma camada gráfica de UI na plataforma. Além disso, esse serviço nativamente é configurado para armazenar as informações pertinentes ao modelo dentro de um diretório no DBFS da workspace.

Uma das vantagens do serviço já ser nativo a plataforma é a facilidade de não necessitar da instanciação de um serviço paralelo, via container, do MLFlow para se utilizar todas as vantagens desta ferramenta.

Aproveitando desse fato e também da possibilidade de registrar modelos em um repositório remoto (feature disponibilizada nativamente pelo MLFlow), é possível orquestrar um *pipeline* para se realizar o treinamento e logging de um modelo em um ambiente de experimentação, sandbox, e registrá-lo remotamente em um ambiente produtivo, de deploy. Conforme ilustrado na figura abaixo.



8.1 CRIAÇÃO DE SECRETS DATABRICKS

Para utilização da API do MLFlow para acesso de modelos em um repositório remoto é necessário criar um escopo dentro do que seria a *"key vault"* do workspace local Databricks e adicionar três secrets referentes ao workspace remoto (host, token e workspace-id), isso pode ser feito de duas maneiras distintas, via Databricks CLI ou via Databricks API, para esta implantação foi utilizada o Databricks CLI. É necessário seguir o procedimento detalhado descrito na documentação que pode ser encontrada no link: Secret scopes - Azure Databricks | Microsoft Learn.

Após a correta realização do procedimento é possível acessar um repositório remoto utilizando o Client da API do MLFlow, utilizando os métodos descritos no passo-a-passo. Uma vez conectado ao repositório remoto tem-se o acesso completo as funções do MLFlow.

Na seção 4.4 e 4.5 foram descritas as *secrets* criadas em ambos os ambientes GCP e Azure para utilização dos recursos cross-workspace. Vale salientar que faz-se necessário a prévia criação do token de acesso nos workspaces que serão utilizados.

8.2 UTILIZAÇÃO DE CLIENT PARA CONEXÃO AOS ARTEFATOS REMOTOS

9. DEPLOY DATABRICKS GCP

10. TEST REQUESTS VIA API

Para testar o ambiente desenvolvido foi necessário localmente desenvolver um outro notebook responsável por montar a requisição a partir de uma imagem e via POST receber o response do modelo deployado. Seguindo a arquitetura da figura abaixo:

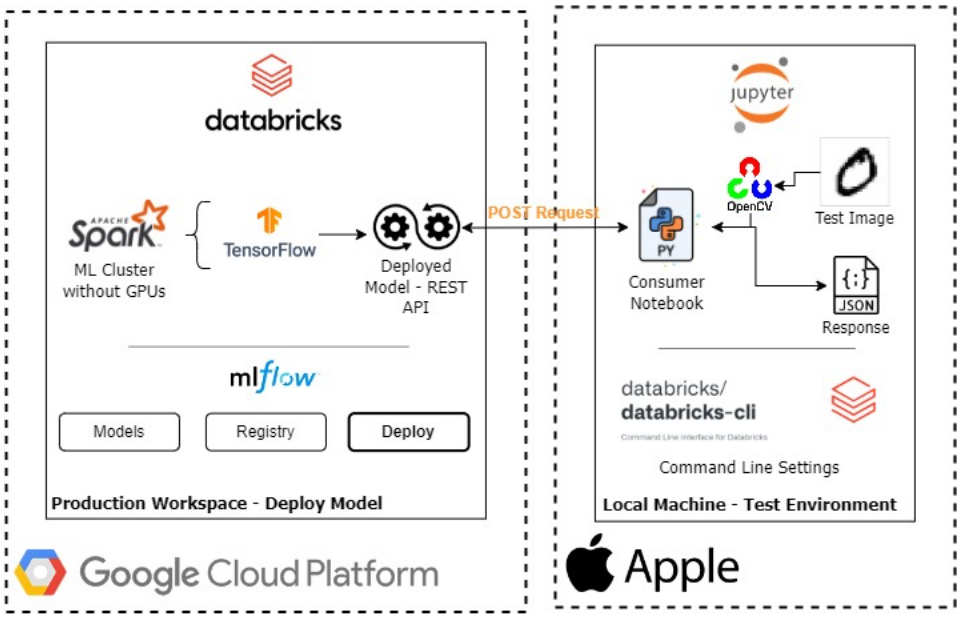


Figure 4 - Arquitetura de Requisições

Este notebook é responsável por montar a requisição no formato pré-definido na documentação do TFX, seguindo o padrão:

E após a chamada tratar o retorno (numpy array) para encontrar o maior valor que corresponde a predição do modelo nas posições de 0 a 9.

A chamada precisa conter o token para conexão com o workspace Databricks em seu header, visando sempre as boas práticas de programação e segurança o token foi criado utilizando uma váriavel de ambiente, para que o mesmo não ficasse exposto em código, a variável utilizada foi:

DATABRICKS_TOKEN_GCP

Já o payload deve seguir o formato:

'{"instances" : [image_np_array]}'

Caso mais de uma imagem seja enviada na chamada é necessário adicionar mais instances de acordo com a quantidade desejada. Detalhes podem ser consultados diretamente no notebook: *test_environment_notebook.ipynb*, no qual está descrita toda implementação e consulta na API REST. O tamanho definido para imagem de entrada é de

28x28 pixels, logo, dentro da função foi implementado o redimensionamento das imagens fornecidas pelo usuário.

11. PROPOSTA DE EVOLUÇÃO TÉCNICA

Existem oportunidades de melhoria para o desenho de solução implementado, como por exemplo, a utilização de terraform para criação dos clusters de deploy dimensionados de forma mais inteligente para a demanda que o serviço terá. É possível também explorar em mais detalhes a estratégia *custom* de treinamento possibilitando orquestrar de maneira mais eficiente os recursos necessários para um treinamento *end-to-end* de um modelo de *Deep Learning.*

Para avaliação de performance comparativa entre o uso de GPU ou apenas de CPU para treinamento seria necessário evoluir o modelo, adicionando uma maior quantidade de camadas, e épocas em seu treinamento, ganhando assim uma robusteza maior em relação a tempo de treino e acurácia. Haja vista que o foco deste case não era a evolução dentro do treinamento do modelo, e o tempo para realização era limitado, a construção do *pipeline* foi priorizada.

12. CONCLUSÕES

No desenvolvimento deste case foi possível constatar a importância de possuir as ferramentas de consumo, preparação, modelagem e deploy da forma mais agnóstica possível. Sendo possível facilmente alternar entre provedores de cloud, quando necessário, e utilizando um fluxo de MLOps/DevOps para de forma eficiente gerir todo o ciclo de vida de desenvolvimento de modelos de ML.

Ademais, para futuras abordagens em problemas de Deep Learning a estratégia de processamento distribuído do Spark e Tensorflow é altamente recomendada, podendo ser um grande catalizador da modelagem DL.

Importante constatar também que, o MLFlow mais uma vez se mostra um importantíssimo e poderosíssimo aliado do Engenheiro de Machine Learning e do Cientista de Dados, e a integração nativa do MLFlow com a plataforma Databricks permite um grande aumento de velocidade para os times de Dados realizarem o deploy dos seus pipelines.

13. REFERÊNCIAS

Site: https://learn.microsoft.com/en-us/azure/databricks/machine-learning/train-model/distributed-training/spark-tf-distributor

Site: https://learn.microsoft.com/en-us/azure/databricks/machine-learning/manage-model-lifecycle/multiple-workspaces

Site: https://www.databricks.com/blog/2016/01/25/deep-learning-with-apache-spark-and-tensorflow.html

Site: https://www.mlflow.org/docs/latest/rest-api.html