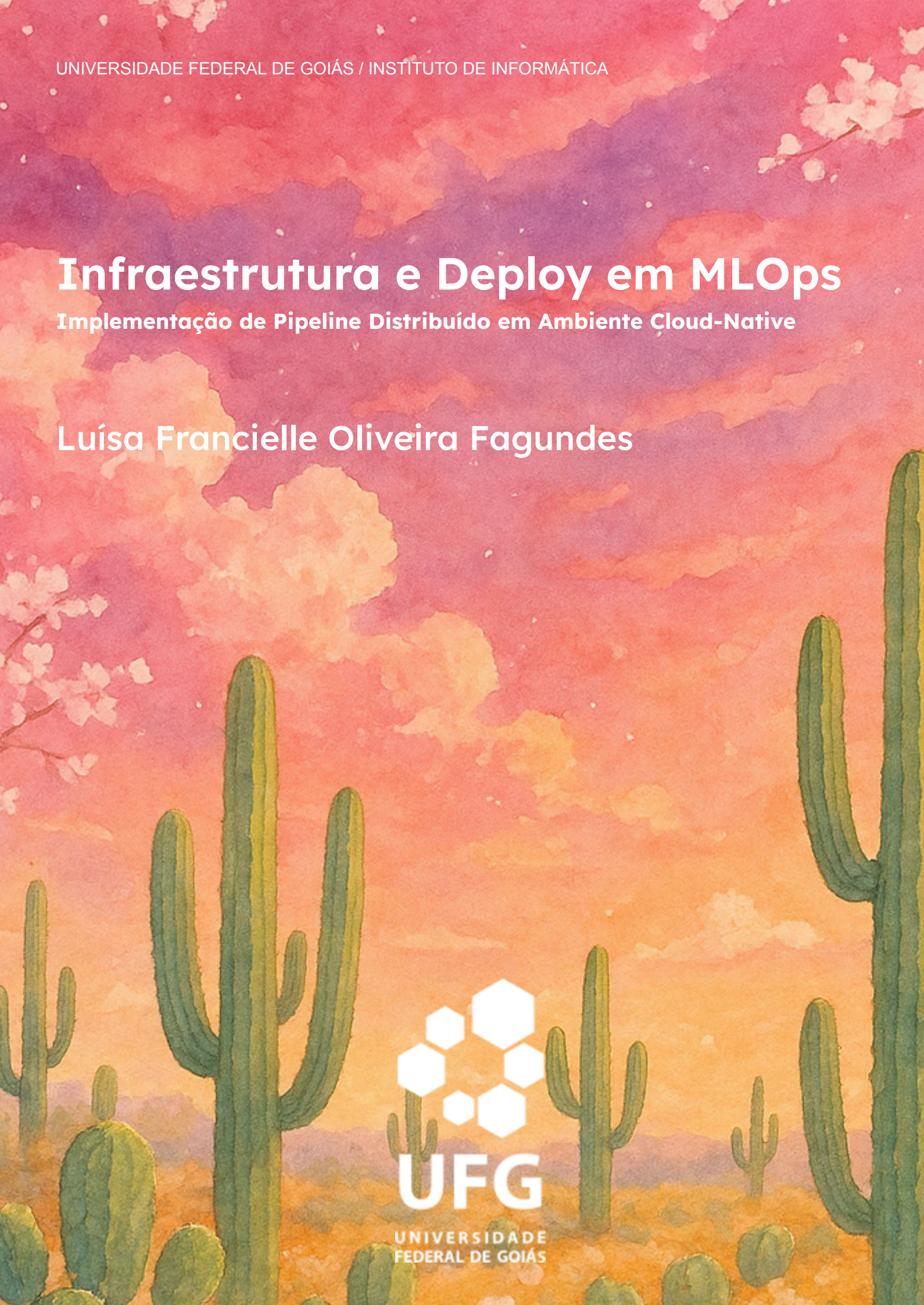
UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Infraestrutura e Deploy em MLOps

Implementação de Pipeline Distribuído em Ambiente Cloud-Native

Luísa Francielle Oliveira Fagundes



UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

LUÍSA FRANCIELLE OLIVEIRA FAGUNDES

Infraestrutura e Deploy em MLOps

Implementação de Pipeline Distribuído em Ambiente Cloud-Native

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): LUÍSA FRANCIELLE OLIVEIRA FAGUNDES

Título do trabalho: Infraestrutura e Deploy em MLOps

Implementação de Pipeline Distribuído em Ambiente Cloud-Native

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Luísa Francielle Oliveira Fagundes**, **Discente**, em 04/02/2026, às 17:19, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 13/03/2026, às 11:38, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5956682** e o código CRC **09D33B2A**.

Referência: Processo nº 23070.005517/2026-12 SEI nº 5956682

LUÍSA FRANCIELLE OLIVEIRA FAGUNDES

Infraestrutura e Deploy em MLOps

Implementação de Pipeline Distribuído em Ambiente Cloud-Native

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Orientador: Prof. Dr. Fernando Marques Federson

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

FAGUNDES, LUÍSA FRANCIELLE OLIVEIRA
Infraestrutura e Deploy em MLOps [manuscrito]: Implementação de Pipeline Distribuído em Ambiente Cloud-Native / LUÍSA FRANCIELLE OLIVEIRA FAGUNDES. - 2025.
78 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Mlops. 3. Ambiente Cloud-native.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

LUÍSA FRANCIELLE OLIVEIRA FAGUNDES

Infraestrutura e Deploy em MLOps

Implementação de Pipeline Distribuído em Ambiente Cloud-Native

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Prof. Dr. Sávio Salvarino Teles de Oliveira
(INF-UFG)

LUÍSA FRANCIELLE OLIVEIRA FAGUNDES

Infraestrutura e Deploy em MLOps

Implementação de Pipeline Distribuído em Ambiente Cloud-Native

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **MLOps**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: Inteligência artificial; MLOps; Ambiente cloud-native.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **MLOps**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: Artificial intelligence; MLOps; Cloud-native environment.

Goiânia

2025

Minha Jornada

Luísa Francielle Oliveira Fagundes

Especialista em: MLOps

MINHA JORNADA

Nome: **Luísa Francielle Oliveira Fagundes**

Especialidade: **MLOps**

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

Minha jornada começou na **Semana 1**, quando iniciei a busca pela área em que eu iria me tornar especialista. A experiência adquirida na disciplina de Processamento de Dados Massivos, cursada durante o Bacharelado, foi fundamental para despertar meu interesse por ***Machine Learning Operations (MLOps)***. Antes de definir um foco de estudo, busquei compreender de onde essa área surgiu, quais desafios motivaram seu desenvolvimento e como ela se tornou um conjunto de práticas fundamentais para levar modelos à produção em escala. Para isso, iniciei a leitura de artigos e blogs técnicos, conforme registrado no **Apêndice 1**, o que me permitiu construir uma base teórica mais consistente sobre o propósito do MLOps e sobre as práticas que estruturam o desenvolvimento, integração e operação de modelos em ambientes reais. Ao final dessa etapa exploratória, defini que meu foco seria **Infraestrutura e Deploy de Modelos**.

A partir da definição da área de interesse, nas **Semanas 2 e 3**, desenvolvi uma revisão bibliográfica inicial que envolveu a seleção e a leitura de artigos científicos, *surveys*

[1] Dez Semanas, entre setembro de 2025 e dezembro de 2025.

e outros materiais relevantes sobre o tema. O objetivo central dessa etapa foi compreender com maior precisão o que a fase de *Deployment* abrange dentro do ciclo de vida de modelos de *Machine Learning*, identificando seus principais desafios, práticas e componentes. Durante esse processo, elaborei dois resumos teóricos que sintetizam os conceitos estudados, disponíveis no **Apêndice 2**. Paralelamente, iniciei, em conjunto com duas colegas que também escolheram o domínio de MLOps, a construção de um glossário com os termos mais recorrentes da área, disponível no **Apêndice 3**. Também organizei em uma planilha, igualmente presente no **Apêndice 3**, todas as leituras realizadas. Essa sistematização permitiu visualizar com maior clareza em quais etapas do *pipeline* cada conceito, desafio ou solução se insere, contribuindo para a consolidação gradual do entendimento sobre o funcionamento e a complexidade do Ciclo de Vida de modelos de *Machine Learning*.

Nas **Semanas 4 e 5**, foquei na compreensão da infraestrutura necessária para desenvolver e operar modelos de Machine Learning. Nesse período, organizei e salvei os *links* e materiais técnicos que serviriam de referência para esse processo. Além disso, avancei nos estudos com foco em três eixos principais: (1) qualidade de serviço, (2) trade-offs de arquitetura e (3) mecanismos de observabilidade e confiabilidade. Essa etapa teve como finalidade entender quais requisitos de infraestrutura precisam ser considerados antes do desenvolvimento de um modelo, incluindo SLIs, SLOs e SLAs relacionados a latência, *throughput* e disponibilidade, além de requisitos não funcionais como custo, escalabilidade, segurança, compliance e portabilidade. Também estudei práticas de resiliência essenciais para ambientes de produção, como *exponential backoff*, *circuit breaker*, *autoscaling*, uso de instâncias *preemptible* ou *spot* e estratégias de *rollback*. O estudo de caso realizado nesse período teve função formativa, permitindo aplicar esses conceitos em um cenário controlado para visualizar, de forma prática, como decisões de infraestrutura influenciam todas as etapas do Ciclo de Vida de um modelo. Tanto o documento do estudo, quanto os links utilizados encontram-se, respectivamente, nos **Apêndices 5 e 4.**

Nas **Semanas 6 e 7**, aprofundei o estudo de técnicas e estratégias de *deploy* de modelos, avançando na compreensão de como diferentes arquiteturas de implantação

sustentam soluções em produção. Nesse período, investiguei as principais modalidades utilizadas nesse processo, incluindo operações em *batch*, em tempo real, em *streaming* e em dispositivos *edge*, analisando como cada abordagem se adequa a diferentes requisitos de latência, volume de dados e disponibilidade. Também estudei estratégias que permitem atualizar sistemas sem interrupções, como *blue green deployment*, *canary deployment* e *shadow deployment*, compreendendo seus objetivos, vantagens e limitações. Além desses aspectos conceituais, explorei boas práticas para construção de ambientes escaláveis e confiáveis em nuvem, entendendo como serviços gerenciados contribuem para desempenho, robustez e controle operacional. Ademais, realizei uma busca direcionada por artigos e exemplos que me ajudassem a identificar possibilidades de implementação para a etapa prática. O objetivo era compreender quais abordagens poderiam servir de referência para um experimento próprio. E esse levantamento exploratório, organizado no **Apêndice 6**, guiou a definição do caminho a seguir na próxima etapa da minha Residência.

Nas **Semanas 8, 9 e 10**, avancei para a etapa de aplicação prática, iniciando a adaptação de um fluxo local de aprendizado geoespacial para execução completa na nuvem utilizando o *Google Cloud Platform (GCP)*. Com base no estudo teórico desenvolvido nas **Semanas** anteriores, tomei como referência a arquitetura do *framework* “InstaGeo”, cuja organização integrada de dados, processamento e modelagem se mostrou adequada para consolidar os conceitos aprendidos ao longo da Residência. A partir dessa estrutura, mapeei cada parte do *pipeline* para os serviços equivalentes do GCP, configurando os recursos necessários para viabilizar sua execução em ambiente distribuído. Essa adaptação envolveu a preparação dos dados geoespaciais, incluindo organização e armazenamento no *Cloud Storage*, além da construção dos contêineres responsáveis pelas etapas de processamento. Durante esse processo, explorei práticas essenciais de infraestrutura em nuvem, como criação de ambientes isolados, configuração de permissões, definição de políticas de segurança e ajustes de rede, vivenciando também desafios reais, como limitações de armazenamento, expansão de discos e provisionamento de recursos adicionais para garantir o andamento adequado das tarefas. Na continuidade dessa etapa, explorei a execução distribuída das tarefas do *pipeline*, incluindo o treinamento dos modelos, a criação de imagens de execução e a configuração dos serviços responsáveis por orquestrar *workloads* em nuvem. Trabalhei com recursos gerenciados para automatizar partes do processo e

estruturar a execução das atividades de maneira mais organizada e reprodutível. Ao longo desse período, também organizei um repositório dedicado para registrar os passos realizados, além de escrever tutoriais detalhados que documentam essa adaptação e que podem ser consultados no **Apêndice 10.** Essa experiência me permitiu compreender como *pipelines* originalmente locais podem ser migrados e executados de maneira escalável, resiliente e padronizada em ambientes *cloud-native*, consolidando a visão de como dados, processamento, treinamento e implantação se articulam dentro do ecossistema do GCP.

Considerando tudo o que vivenciei ao longo desta Jornada, gostaria de registrar que ela contribuiu de maneira muito significativa para a minha formação profissional. Iniciei o processo com muitas dúvidas sobre qual caminho seguir e não me sentia plenamente segura quanto à escolha do meu foco de especialização. No entanto, conforme avancei nos estudos, organizei minhas leituras e comecei a trabalhar de forma mais prática com MLOps, percebendo uma crescente identificação com a área. Senti, pela primeira vez, que estava de fato me encontrando dentro de um campo que une desafios técnicos, aprendizagem contínua e um propósito claro. Essa trajetória reforçou não apenas meu interesse por MLOps, mas também a convicção de que a aprendizagem contínua é um dos aspectos mais gratificantes de atuar na área da Inteligência Artificial.

Por fim, gostaria de agradecer, primeiramente, à minha família, minha mãe Elenísia, meu pai João Gildo e meu irmão João Lucas, que sempre foi minha base e meu maior apoio nos momentos em que precisei, acreditando e incentivando este meu sonho. Agradeço também ao professor Fernando Federson por todos os ensinamentos, por caminhar ao nosso lado ao longo de todo o Bacharelado e por me incentivar a nunca “perder o brilho”. Estendo meus agradecimentos ao professor Sávio Teles, responsável por despertar em mim a paixão pela área que fundamenta este Trabalho de Conclusão de Curso. Não posso deixar de agradecer aos meus colegas de turma, aos meus amigos e aos amigos do Pequi Mecânico, que tornaram essa vivência mais leve e prazerosa, sempre prezando pelo apoio mútuo e pelo crescimento conjunto. Agradeço, ainda, a todos os professores que desempenharam papel essencial na minha formação e na trajetória que me conduziu até

este momento, em especial aos professores Ricardo Franco, Telma Woerle, Anderson Soares, Lucas Araújo e Bruno Brandão.

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 4 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante a primeira Semana, foi estudado o conceito de **MLOps**, com o objetivo de compreender sua fundamentação, sua origem e o que ele abrange. Para isso, foi realizada uma busca por artigos na área, conforme documentado neste arquivo: Documento auxiliar da Semana 01

A partir desse estudo, concluiu-se que:

- MLOps não surgiu exatamente de DevOps, como muitos acreditam, mas foi fortemente inspirado por ele.
- DevOps nasceu da necessidade de integrar desenvolvimento (Dev) e operações (Ops) de software, reduzindo o atrito entre equipes, automatizando deploys e garantindo entregas contínuas.
- MLOps (Machine Learning Operations) surgiu posteriormente, como uma adaptação e extensão dos princípios de DevOps para projetos de Inteligência Artificial e Machine Learning.

Embora aproveite conceitos do DevOps — como integração contínua, entrega contínua, monitoramento e automação de pipelines — o MLOps precisa lidar com desafios exclusivos do ciclo de vida de modelos de ML, tais como:

- Gerenciamento de Dados
- Experimentação e Gestão de Modelos
- Pipeline de ML
- CI/CD para ML
- Infraestrutura e Deploy de Modelos

- Monitoramento e Observabilidade
- Automação de Re-Treinamento

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Na próxima Semana, será realizado um estudo com o intuito de aprofundar os conhecimentos sobre **Infraestrutura e Deploy de Modelos**, por meio da busca e análise de artigos que permitam compreender melhor o tema.

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Documento auxiliar da Semana 01

Inicialmente, foi realizada uma leitura exploratória sobre o tema, utilizando a ferramenta **Gemini** no modo "*Deep Research*". Essa busca resultou na produção do seguinte arquivo: Origem e Conceitos do MLOps

A leitura desse material serviu como ponto de partida para o aprofundamento no assunto. Durante a análise, os principais trechos e conceitos foram destacados com um marca-texto, o que facilitou a identificação dos conteúdos mais relevantes e a organização das ideias que seriam trabalhadas nas etapas seguintes.

Ademais, com o intuito de obter um conteúdo mais sólido sobre o tema, foi realizada a leitura do artigo **Machine Learning Operations (MLOps): Overview, Definition, and Architecture**, disponível em: https://arxiv.org/pdf/2205.02302 (Acesso em 02/09/2025).

A principal conclusão desse artigo é que o paradigma de MLOps (Operações de Aprendizado de Máquina) surge como solução para o desafio crítico de levar projetos de *machine learning* da fase de prova de conceito (PoC) para a produção, etapa em que muitos projetos falham. A pesquisa evidencia que, enquanto a academia concentrou esforços na construção de modelos, a operacionalização de sistemas complexos de ML no mundo real foi negligenciada, resultando em fluxos de trabalho frequentemente gerenciados de forma manual.

Por meio de um estudo de método misto — que incluiu revisão de literatura, análise de ferramentas e entrevistas com especialistas —, o trabalho busca esclarecer o termo MLOps ao identificar seus quatro aspectos principais: princípios, componentes, papéis e arquitetura. Com base nesses pilares, o artigo propõe uma definição holística, fornecendo um entendimento comum capaz de auxiliar pesquisadores e profissionais a estruturar projetos de ML bem-sucedidos no futuro.

Além disso, também foi realizada a leitura do artigo **"MLOps: A Taxonomy and a Methodology"**, disponível em https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9792270 (Acesso em 02/09/2025).

Esse trabalho apresenta uma revisão abrangente da literatura sobre MLOps e propõe tanto uma taxonomia para organizar os trabalhos existentes quanto uma metodologia prática para implementação de projetos de *machine learning* em produção. Os autores destacam que,

apesar do avanço da inteligência artificial e do aumento do volume de dados disponíveis, ainda persiste uma lacuna significativa entre a pesquisa acadêmica e a aplicação industrial, muitas vezes devido à ausência de processos padronizados.

Para enfrentar esse desafio, o artigo define um pipeline composto por dez etapas — desde a compreensão do problema de negócio até a sustentabilidade —, incorporando práticas de integração e entrega contínuas (CI/CD), monitoramento, explicabilidade e eficiência energética. O objetivo central é sistematizar conceitos e oferecer diretrizes claras que possibilitem a transição eficaz de modelos de ML da fase experimental para soluções escaláveis e industrializáveis.

APÊNDICE 2

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 10 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante essa segunda Semana, foi realizado um estudo para compreender o ciclo de vida de um modelo de Machine Learning, com o objetivo de identificar de forma mais clara o que a etapa de **Deployment** abrange.

Após esse entendimento inicial, foram aprofundados **conceitos fundamentais da área de MLOps**, com foco em infraestrutura e boas práticas relacionadas ao processo de deploy de modelos. Esse estudo permitiu consolidar uma base teórica mais ampla, necessária para dar prosseguimento às próximas etapas.

Todos as anotações das aulas, vídeos e artigos lidos estão presentes no documento auxiliar da Semana 2 , que sintetiza as reflexões e análises realizadas.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana, pretende-se dar continuidade ao estudo dos conceitos fundamentais de MLOps. O objetivo é compreender como esses elementos **sustentam as práticas de MLOps**, possibilitando maior integração entre desenvolvimento, automação e gestão de modelos.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Documento auxiliar da Semana 02

Inicialmente, foi assistido algumas aulas gratuitas no "DataCamp" com o título **"MLOps Fundamentals"** com o intuito de entender o que abrangia a área de Deployment.

Anotações das aulas:

O MLOps é aplicado ao que chamamos de ciclo de vida do aprendizado de máquina, que inclui tudo, desde o design e o desenvolvimento até a manutenção do aprendizado de máquina na produção.

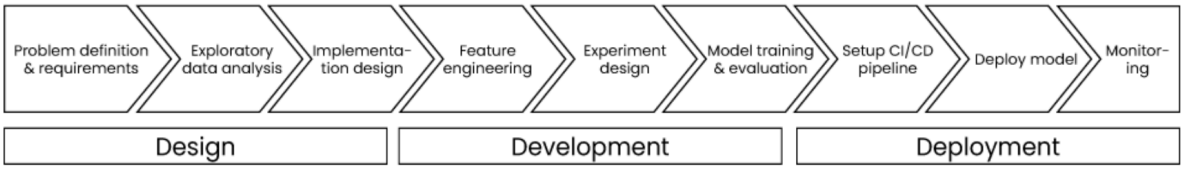


Figura 1. Imagem retirada da aula "MLOps Fundamentals".

Na fase de Design:

- Contextualizamos o problema
- Avaliamos o valor agregado do uso do aprendizado de máquina
- Requisitos de Negócio
- Estabelecer métricas-chaves -> Permite monitorar o progresso de forma eficaz
- Garantir o processamento de dados de alta qualidade -> Essencial para construir um modelo robusto

Fase de Development:

- Desenvolver modelos de Machine Learning
- Experimentar diferentes dados, algoritmos e hiperparâmetros
- Treinar modelos e avaliar performances
- Modelo pronto para implementação

Fase de Deployment (onde o modelo encontra o mundo real):

- Integrar o modelo de Machine Learning nos negócios
- Deploying do modelo em produção -> Podemos criar um micro serviço em torno do nosso modelo, permitindo fácil acesso e escalabilidade
- Monitoramento da performance -> Detectar desvios de dados e receber alertas se as previsões do modelo começaram a piorar

Perguntas que devem ser sempre presentes:

- **O projeto ainda é viável?**
- **Está entregando valor?**

Continuous Integration (CI) e Continuous Deployment/Delivery (CD)

Ademais, estudei sobre o conceito de CI e CD em um vídeo no Youtube e fiz algumas anotações:

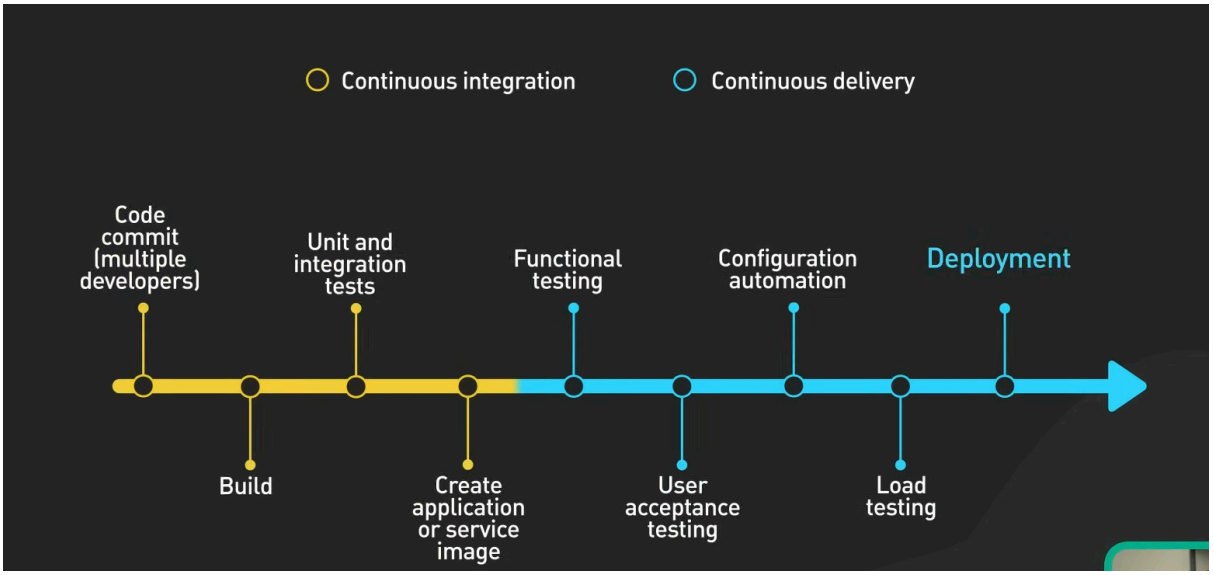


Figura 2. Imagem retirada do vídeo "CI/CD In 5 Minutes | Is It Worth The Hassle: Crash Course System Design".

O processo de CI/CD cuida da construção, teste e implantação de novo código na produção. A Integração Contínua (CI) é uma prática consolidada e amplamente utilizada. Em poucas palavras, consiste no uso de automação para que as equipes possam integrar alterações de código em um repositório compartilhado de forma rápida, frequente e segura.

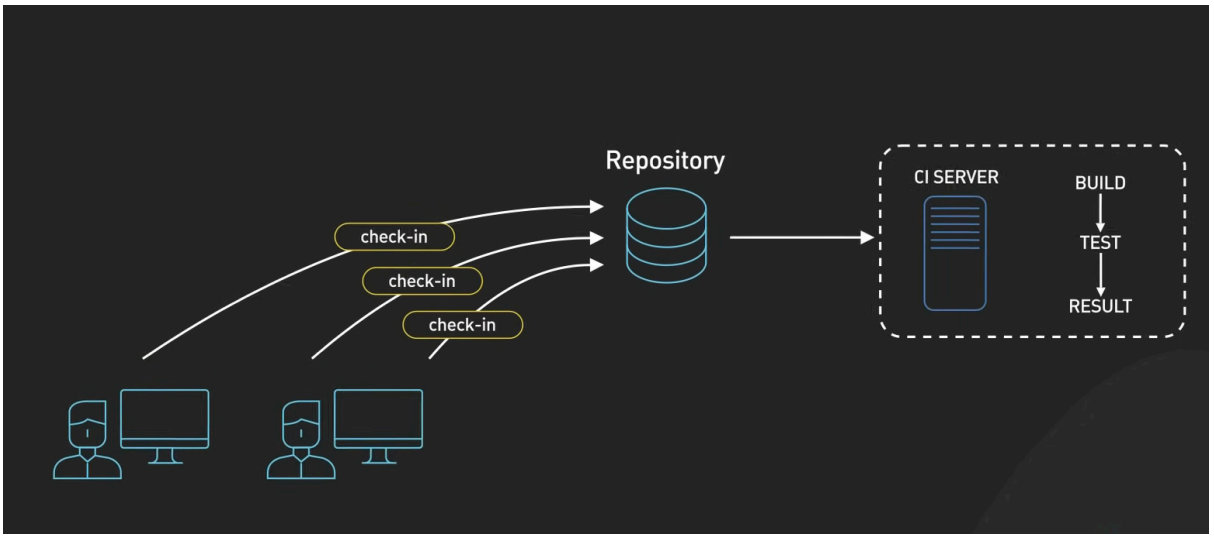


Figura 3. Imagem retirada do vídeo "CI/CD In 5 Minutes | Is It Worth The Hassle: Crash Course System Design".

A cada confirmação, um servidor de CI aciona um fluxo de trabalho automatizado que executa uma série de tarefas para verificar se a alteração é segura para ser integrada à branch principal. O bom funcionamento desse processo depende da existência de um conjunto robusto de testes. No entanto, não é trivial manter um conjunto de testes com cobertura suficiente e que não seja instável. Uma alta cobertura de teste geralmente demora mais para ser executada, o que afeta a produtividade do desenvolvedor.

Continuous Deployment/Delivery (CD):

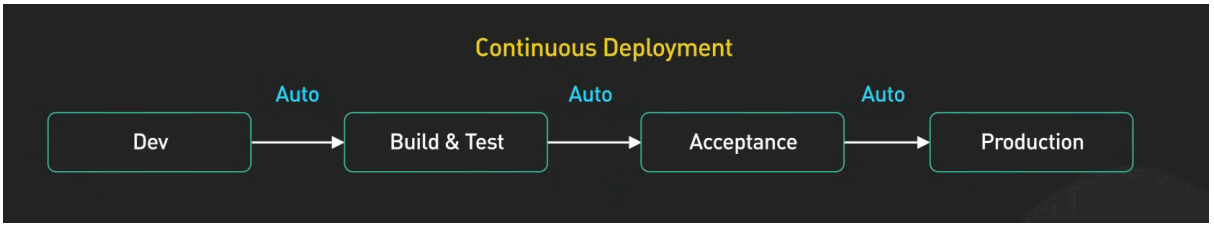


Figura 4. Imagem retirada do vídeo "CI/CD In 5 Minutes | Is It Worth The Hassle: Crash Course System Design".

A prática de CD em sua forma plena é mais complexa e menos comum do que a CI. Embora exista, sua adoção ainda é limitada. Muitas equipes acabam aplicando CD apenas em cenários mais simples, geralmente em sistemas sem estado, como camadas de API ou servidores web.

É comum também encapsular novos recursos em *feature flags* (sinalizadores de recursos), de forma a separar a implantação do código da ativação das funcionalidades. Essa

prática permite que a equipe desative rapidamente um recurso problemático sem a necessidade de reverter todo o código.

Outra abordagem recorrente em sistemas de grande escala é a implementação canário, utilizada especialmente em produtos com centenas de milhões de usuários. Nessa estratégia, o novo código é liberado inicialmente para um subconjunto restrito de usuários avançados e colaboradores que têm maior tolerância a riscos e interesse em experimentar novidades. Assim, é possível validar o comportamento do sistema em um ambiente real, reduzindo o impacto potencial caso surjam falhas, uma vez que o alcance inicial do problema permanece limitado.

Ferramentas: Kubernetes -> Argo CD

Resumindo:

A Integração Contínua (CI) consiste em integrar alterações de código ao repositório compartilhado de forma frequente e automatizada, com o suporte de pipelines que executam testes e verificações de qualidade. Seu objetivo é identificar erros rapidamente e garantir que cada alteração esteja pronta para ser incorporada ao projeto com segurança.

A Entrega Contínua (CD), por sua vez, estende esse processo ao automatizar também a disponibilização das alterações em ambientes de produção. Embora mais complexa e menos comum do que a CI, a CD busca tornar a implantação previsível e segura. Entre as práticas associadas, destacam-se o uso de *feature flags*, que permitem ativar ou desativar funcionalidades sem reverter código, e a implementação canário, que libera novas versões para um subconjunto restrito de usuários antes da distribuição ampla.

Orquestração

A orquestração de ML refere-se ao processo de gerenciamento e coordenação das várias tarefas e fluxos de trabalho envolvidos no ciclo de vida do aprendizado de máquina, desde a preparação de dados e treinamento de modelos até a implantação e monitoramento. Envolve a integração de várias ferramentas e plataformas para agilizar as operações, automatizar tarefas repetitivas e garantir uma colaboração perfeita entre cientistas de dados, engenheiros e equipes de operações. Ao usar estruturas de orquestração, as organizações podem melhorar a reprodutibilidade, a escalabilidade e a eficiência, permitindo-lhes gerenciar pipelines complexos de aprendizado de máquina e melhorar a qualidade geral dos modelos em

produção. Isso garante que os modelos sejam consistentemente atualizados e mantidos, facilitando a rápida iteração e adaptação à mudança de dados e necessidades de negócios.

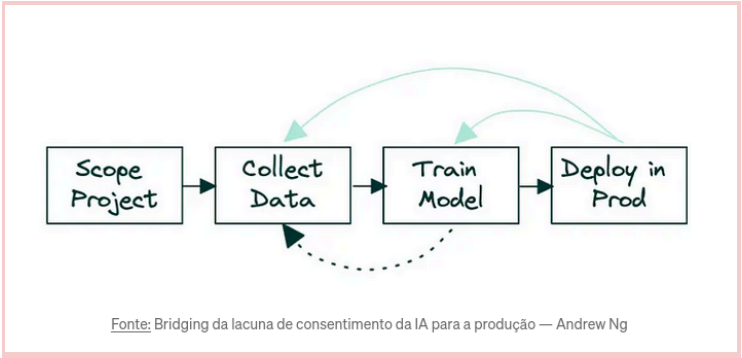
ML Observability

Leitura do artigo no medium sobre o assunto:
https://medium.com/at-the-front-line/ml-observability-hype-or-here-to-stay-acef064ff843

Josh Wills descreve essa falha de forma bastante sucinta:

> *"Por definição, quando estamos implantando modelos em produção, não podemos antecipar e especificar adequadamente todo o comportamento que esperamos ver. Se pudéssemos fazer isso, escreveríamos código para fazê-lo, não nos preocupávamos em fazer aprendizado de máquina, então, o inesperado é um dado."*

O monitoramento e a observabilidade na produção são, portanto, uma necessidade, não uma opção e é necessário um processo iterativo para o ML.



Fonte: Bridging da lacuna de consentimento da IA para a produção — Andrew Ng

Modelo de Desempenho Degradada em Produção

Há uma série de razões pelas quais os modelos de ML podem falhar na produção - desde Training-Prod Skew, onde seu modelo faz previsões precisas como uma prova de conceito, mas tem um desempenho ruim imediatamente quando colocado em produção - até Degenerate Feedback Loops onde as saídas do seu sistema de ML são posteriormente usadas como entradas para o mesmo sistema e causam consequências não intencionais que ampliam o viés no modelo.

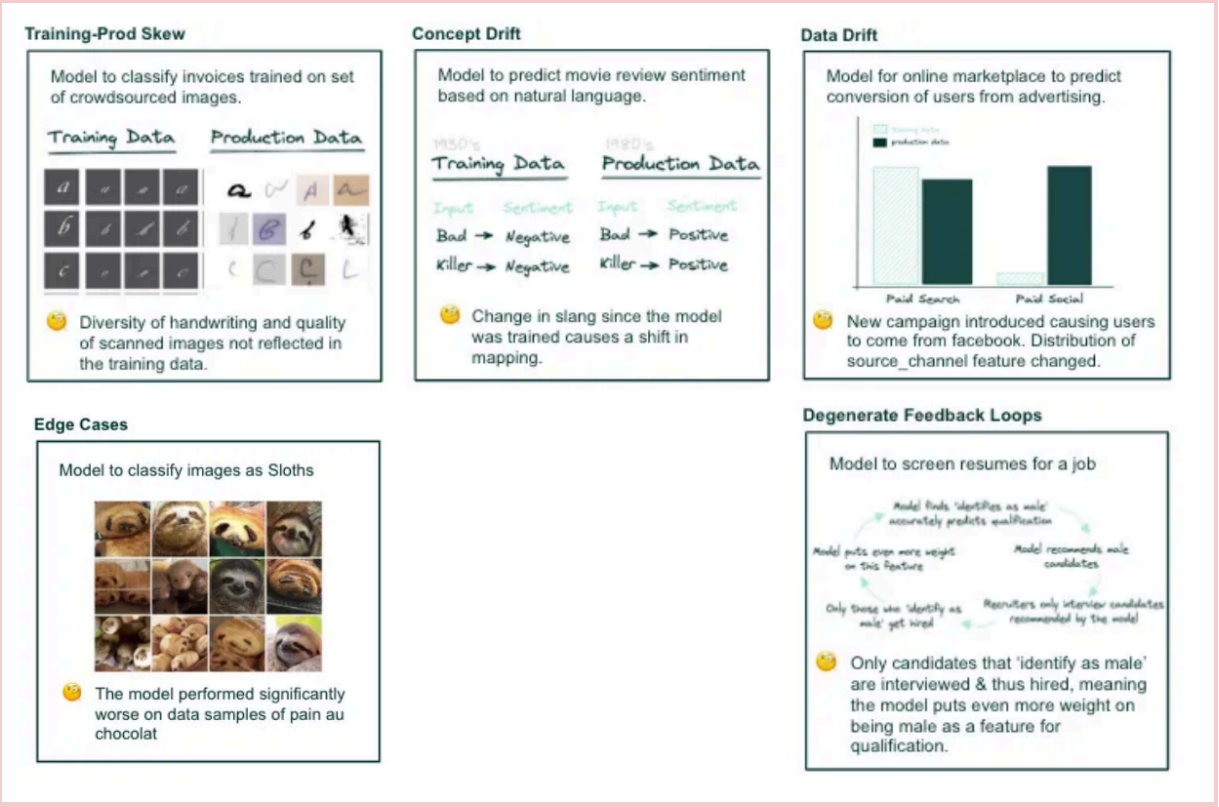


Figura 5. Imagem retirada do artigo “ML Observability — Hype or Here to Stay?”

1. Training–Production Skew

- **O que é:** Diferença entre os dados usados no treinamento e os dados reais em produção.
- **Exemplo da imagem:** O modelo foi treinado para reconhecer números/ letras com um certo estilo de escrita (dados crowdsourced), mas no mundo real aparecem caligrafias muito diferentes ou imagens com qualidade ruim.

- **Problema:** O modelo não generaliza bem porque não viu essa variedade durante o treino.

2. Concept Drift

- **O que é:** A relação entre entrada e saída muda com o tempo.
- **Exemplo da imagem:** Nos anos 1980, a palavra *killer* podia ser interpretada como algo “negativo”, mas hoje em dia pode significar “ótimo” (positivo).

- **Problema:** O modelo perde acurácia porque o conceito que ele aprendeu deixa de refletir a realidade.

3. Data Drift

- **O que é:** Mudança na **distribuição dos dados de entrada** em produção.
- **Exemplo da imagem:** O modelo de marketplace foi treinado com dados de usuários vindos do *Paid Search* (busca paga), mas depois uma campanha fez os usuários virem do *Paid Social* (Facebook).
- **Problema:** O modelo não performa bem porque a proporção das features mudou, mesmo que o conceito em si não tenha mudado.

4. Edge Cases

- **O que é:** Casos raros ou inesperados que o modelo não aprendeu a lidar.
- **Exemplo da imagem:** O modelo classifica bichos-preguiça em fotos. Porém, quando aparece um pão *pain au chocolat* (que visualmente se parece com uma preguiça), ele erra bastante.
- **Problema:** O modelo é enganado por exemplos incomuns que lembram o que foi treinado.

5. Degenerate Feedback Loops

- **O que é:** O modelo reforça seus próprios vieses ao longo do tempo.
- **Exemplo da imagem:** Um modelo de triagem de currículos favorece candidatos do sexo masculino. Ele recomenda mais homens, que são entrevistados e contratados. Isso retroalimenta o viés, porque o modelo "aprende" que ser homem é um fator de qualificação.
- **Problema:** Cria ciclos viciados e injustos, que pioram com o tempo.

ML Observability como solução

O ML Observability – *o conjunto de processos e ferramentas necessárias para manter um modelo saudável na produção* – surgiu como uma disciplina para ajudar os profissionais de ML com os desafios descritos acima. Existem quatro tipos de sinais que você pode monitorar para determinar a integridade do seu modelo, cada um vindo com um trade-off em termos de quão informativa é a métrica e como é fácil de obter.

Métricas do modelo: São o padrão-ouro, comparando previsões com resultados reais. Porém, os rótulos costumam estar ausentes ou atrasados (ex.: prever inadimplência, mas só saber anos depois). Quando não há rótulos disponíveis, usam-se métricas de proxy.

Métricas de negócios: Avaliam o impacto do modelo nos KPIs da empresa. É útil transformar os "medos" das partes interessadas em métricas monitoráveis (ex.: taxa de rejeição injusta em concessão de crédito).

Entradas do modelo: Mais fáceis de monitorar, mas quedas no desempenho das features não significam necessariamente falha do modelo. Há risco de "fadiga de alertas" (muitos alarmes irrelevantes), por isso devem ser usadas para depuração e suporte ao monitoramento de saídas.

Desempenho do sistema: Refere-se ao monitoramento clássico de software (latência, disponibilidade, uso de recursos).

"ML Observability ajuda a manter a qualidade do modelo a longo prazo - mais modelos são, portanto, implantados em produção - isso cria uma maior necessidade de ferramentas de observação."

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 18 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante esta Semana, finalizei os estudos dos conceitos fundamentais de MLOps e organizei um Glossário de termos na área de MLOPS, documento compartilhado com Danielle e Maria Eduarda, para preenchimento colaborativo.

Realizei a leitura do artigo *"Challenges in Deploying Machine Learning: a Survey of Case Studies"* (29 páginas, 2022), destacando pontos relevantes. Em seguida, organizei os destaques em uma planilha: Leitura: Challenges in Deploying Machine Learning: a Survey of Case Studies

Além disso, iniciei a leitura do artigo "A Multivocal Review of MLOps Practices, Challenges and Open Issues" (45 páginas, 2024).

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Finalizar a leitura do artigo *"A Multivocal Review of MLOps Practices, Challenges and Open Issues"*, registrando os principais pontos na planilha de apoio. Em seguida, buscar novos surveys com foco em deploy de modelos, organizar uma lista de leituras complementares, iniciar a análise desses materiais e começar a levantar as principais ferramentas utilizadas para o processo de deploy.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Glossário de termos na área de MLOPS:

Termo	Definição
CI (Continuous Integration)	Processo de integração contínua de código e componentes de ML. Garante que alterações em scripts, pipelines ou configurações sejam testadas e validadas de forma automática, prevenindo falhas na colaboração entre equipes.
CD (Continuous Delivery/Deployment)	Prática de entregar e implantar modelos e pipelines de forma contínua e confiável em ambientes de produção. Permite atualizações rápidas e seguras, reduzindo o tempo
CT (Continuous Training)	Extensão das práticas de CI/CD para o aprendizado de máquina. Automatiza o reprocessamento e re-treinamento de modelos sempre que novos dados estão disponíveis ou quando ocorre drift, garantindo que o modelo permaneça atualizado e relevante.
CM (Continuous Monitoring)	Monitoramento constante de dados e modelos em produção, avaliando métricas técnicas e de negócio, além de detectar problemas como *drift* e vieses.
Orquestração	Gerenciamento e coordenação das várias tarefas e fluxos de trabalho envolvidos no ciclo de vida do aprendizado de máquina, desde a preparação de dados e treinamento de modelos até a implantação e monitoramento.
ML Observability	Conjunto de práticas, ferramentas e métricas voltadas para entender, monitorar e diagnosticar o comportamento de modelos de ML em produção.

Orquestração de Containers	É o processo de automatizar o gerenciamento, a implantação, a escala e a comunicação de aplicações empacotadas em containers.
Experiment Tracking	Processo de monitorar e registrar experimentos de ML, armazenando metadados como hiperparâmetros, métricas, modelos gerados e ambiente computacional. Garante reprodutibilidade, facilita comparações entre execuções e auxilia na escolha do melhor modelo.
Model Registry	Repositório central para versionar, organizar e gerenciar modelos de ML, permitindo controle de ciclo de vida, rastreabilidade e implantação estruturada em produção.
Data Lineage	Rastreamento da origem, transformação e uso dos dados ao longo do pipeline de ML. Permite entender de onde vêm os dados, como foram processados e onde são aplicados, garantindo transparência, auditoria e confiabilidade nos experimentos e modelos.
Feature Store	Repositório centralizado para armazenar, versionar e compartilhar *features* usadas em modelos de ML. Facilita a reutilização, mantém consistência entre treino e produção e acelera o desenvolvimento de novos modelos.
Explainability (XAI – Explainable Artificial Intelligence)	Conjunto de técnicas que tornam modelos de ML interpretáveis e transparentes, permitindo que especialistas entendam e confiem nas decisões dos algoritmos. Towards MLOps: A Framework and Maturity Model
GitOps	Extensão de DevOps que usa repositórios Git como "fonte de verdade" para infraestrutura e deployment, permitindo que

	alterações sejam rastreadas e aplicadas automaticamente.
Shadow Deployment	Estratégia de deployment em que um novo modelo roda em paralelo ao modelo atual, mas sem impactar usuários finais, servindo apenas para testes.
Canary Deployment	Implantação gradual de um novo modelo em uma fração controlada dos usuários, reduzindo riscos de falhas em larga escala. *Toward an Open Source MLOps Architecture*.
Technical Debt (Dívida Técnica)	Compromissos assumidos ao priorizar velocidade sobre qualidade na implementação, que podem aumentar custos de manutenção futura. Hidden Technical Debt in Machine Learning Systems
Maturity Model (Modelo de Maturidade)	Estrutura que descreve níveis de evolução na adoção de MLOps, desde fluxos manuais (nível inicial) até pipelines totalmente automatizados com monitoramento e re-treinamento. Towards MLOps: A Framework and Maturity Model, John, Olsson & Bosch, 2021
Compliance	Adequação de sistemas e pipelines a normas legais e regulatórias (ex.: GDPR, HIPAA), garantindo governança, privacidade e auditabilidade.
A/B Testing	Método experimental que compara duas ou mais versões de um modelo/sistema para avaliar impacto em métricas técnicas ou de negócio.
Blue-Green Deployment	Mantêm-se dois ambientes de produção idênticos: "Blue" (com a versão atual) e "Green" (com a nova versão). O tráfego é direcionado para o ambiente Blue. Quando a nova versão no Green é validada, o tráfego é todo redirecionado para ele. Se algo der

	errado, é fácil e rápido reverter para o ambiente Blue.
Rolling Deployment	A nova versão do modelo é gradualmente introduzida, substituindo as instâncias da versão antiga uma a uma, até que todas estejam atualizadas. Isso permite uma transição suave e sem tempo de inatividade (downtime).
Big Bang (ou Recreate)	A versão antiga é desligada e a nova é ligada de uma vez. É a abordagem mais simples, porém mais arriscada, pois qualquer problema na nova versão afeta todos os usuários.
Federated learning	É uma abordagem de aprendizado de máquina distribuído em que os dados permanecem nos dispositivos ou organizações que os geraram, sem necessidade de centralização; em vez disso, cada nó treina localmente um modelo e envia apenas atualizações de parâmetros (como gradientes ou pesos) para um servidor central, que agrega os resultados e atualiza o modelo global. Esse processo garante maior privacidade, já que os dados brutos não são compartilhados, melhora a eficiência ao reduzir a transferência de dados e permite personalização em contextos locais. No entanto, apresenta desafios como a heterogeneidade dos dados, altos custos de comunicação e riscos de segurança. É amplamente aplicado em áreas como teclados inteligentes de smartphones, saúde e sistemas financeiros.
Dataset Shift	Mudança na distribuição dos dados de entrada usados pelo modelo em produção em relação aos dados de treinamento. Isso faz com que o modelo enfrente padrões diferentes dos que aprendeu, podendo reduzir sua performance.

Concept Drift	Alteração na relação entre as variáveis de entrada (features) e o alvo (label) ao longo do tempo. Diferente do dataset drift, que é mudança só na distribuição dos dados, aqui muda o significado do que o modelo deve aprender. Mesmo que os dados de entrada não mudem muito, a regra que liga entrada e saída pode mudar, exigindo re-treino ou adaptação do modelo.
Ataques Adversariais	Técnicas usadas para enganar modelos de aprendizado de máquina, inserindo pequenas perturbações nos dados de entrada (muitas vezes imperceptíveis para humanos) que levam o modelo a tomar decisões erradas.
Interpretabilidade	Capacidade de compreender e explicar como um modelo de machine learning chega às suas previsões ou decisões. Permite identificar quais variáveis ou fatores influenciaram o resultado, sendo essencial para auditoria, transparência, confiança e uso ético de modelos.
Data Lake	É um repositório centralizado que armazena **grandes volumes de dados em seu formato bruto**, sem necessidade de estrutura pré-definida.
Data Warehouse	É um sistema usado para armazenar e analisar **dados já tratados, limpos e estruturados**, otimizados para consultas e relatórios.
Latência	Tempo entre pedido e resposta, medidos por percentis
AutoML	Automação de etapas críticas de ML (pré-processamento, seleção de features,

	tuning de hiperparâmetros, seleção de modelos, ensembling).
KaizenML	Foco na melhoria contínua de todo o ciclo de vida do ML.
Drift Detection	Processo de identificar se ocorreu uma mudança significativa na distribuição dos dados ao longo do tempo. Tem como objetivo detectar quando um modelo pode perder desempenho devido a alterações no ambiente ou nos dados.
Drift Localization	Processo de identificar em quais regiões do espaço de dados a mudança ocorreu. Busca determinar onde exatamente o drift acontece, permitindo ações mais direcionadas, como re-treinar o modelo apenas para segmentos afetados.
Drift Explanations	Processo de tornar compreensível o fenômeno do drift, explicando como e por que a mudança ocorreu. Envolve indicar quais variáveis foram mais impactadas, quais padrões se alteraram e como isso afeta o comportamento do modelo.
Exponential backoff	Ao falhar uma chamada, você espera e tenta de novo com intervalos que dobram (1s, 2s, 4s…), com limite e jitter.
Jitter	Pequena aleatoriedade
Circuit breaker	Um “disjuntor” que abre quando a taxa de erro/latência explode; enquanto aberto, bloqueia novas chamadas e retorna rápido (fallback).

Autoscaling	Escala automaticamente a quantidade de instâncias conforme carga para manter SLOs.
Throughput	Taxa sustentada de processamento, é importante para definir a capacidade de aguentar pico sem estourar a latência.
Políticas de rollback	Regras para voltar rápido à versão anterior quando o novo deploy degrada métricas.
SLO (Service Level Objective)	lvo/limite da qualidade (ex.: p95 < 200 ms em 99% do mês).
SLI (Service Level Indicator)	Métrica medida (p95, AUC, PSI, 5xx) usada para verificar SLO.
SLA (Service Level Agreement)	Compromisso contratual externo derivado de SLOs.
Burn rate (de erro)	Quão rápido você "queima" o orçamento do SLO; usado para alertas rápido/lento.
Multi-armed bandits	Roteamento adaptativo de tráfego conforme performance online.
Batching vs. Streaming	Processamento em lotes vs. em fluxo/tempo real para deploy/monitoria.

Anotações do artigo "Challenges in Deploying Machine Learning: a Survey of Case Studies"

Seção/Tema	Conteúdo	Notas/Observações
Fluxo de Trabalho de Implantação	Adota-se o fluxo descrito por Ashmore et al., dividido em quatro estágios: gerenciamento de dados, aprendizagem do modelo, verificação do modelo e implantação do modelo. Esses estágios podem ocorrer em paralelo e se retroalimentar.	A escolha desse fluxo permite classificar desafios de forma mais refinada. O artigo ressalta que não é cronograma rígido, mas abstração útil.
Gerenciamento de Dados	A qualidade dos dados é central para o sucesso do ML. Problemas comuns aparecem em coleta, pré-processamento, aumento e análise de dados.	Polyzotis et al. mostram que essa etapa consome muito tempo e energia inesperados.
Gerenciamento de Dados – Coleta	Encontrar e entender quais dados existem é desafiador, especialmente em grandes organizações. Muitas vezes dados estão dispersos, em logs ou nem são armazenados.	Caso do Twitter: serviços separados armazenavam partes diferentes da mesma entidade, dificultando o rastreamento.
Gerenciamento de Dados – Pré-processamento	A integração de múltiplas fontes com esquemas e convenções diferentes é complexa. Exemplo do Firebird (Corpo de Bombeiros de Atlanta), que precisou unir 12 bases distintas e corrigir dados espaciais inconsistentes.	Integração e limpeza consomem muito esforço humano antes mesmo do modelo ser treinado.
Gerenciamento de Dados – Aumento	A ausência de rótulos é um dos maiores obstáculos. Problemas vêm do volume, da falta de	Exemplos: tráfego de rede (difícil rotular pacotes), imagens médicas (escassez de especialistas). Surge o

	especialistas e da baixa variância dos dados. Rotulagem imprecisa prejudica setores sensíveis como saúde.	“Final Percent Challenge”, onde pequenos erros são inaceitáveis.
Gerenciamento de Dados – Análise	Detectar vieses e mudanças inesperadas de distribuição é essencial, mas faltam ferramentas robustas de perfilagem de dados e visualização.	Pesquisa da Microsoft mostra que problemas de dados são a principal razão para duvidar da qualidade de projetos de ML.
Aprendizagem do Modelo	A fase mais estudada academicamente, mas ainda com muitas barreiras práticas em seleção, treinamento e ajuste de hiperparâmetros.	O número de submissões à NeurIPS ilustra o crescimento, mas persistem desafios de custo, interpretabilidade e privacidade.
Aprendizagem – Seleção de Modelos	Na prática, modelos simples muitas vezes são preferidos pela interpretabilidade, rapidez de implantação e restrições de hardware. Deep learning é usado quando há muitos dados, mas enfrenta barreiras computacionais.	Casos: AirBnB simplificou rede neural para conseguir implantar; Europa Clipper usou PCA por limitações da nave; bancos preferem árvores de decisão por interpretabilidade.
Aprendizagem – Treinamento	O custo computacional e ambiental do treinamento é elevado. Modelos como BERT custam de 50 mil a 1,6 milhão de dólares para treinar. Impacto ambiental é comparável às emissões de carros em toda a vida útil.	Além do custo, há riscos à privacidade: ataques de inferência de pertencimento revelam se dados individuais foram usados. Soluções incluem privacidade diferencial, criptografia homomórfica e aprendizagem federada.
Aprendizagem – Hiperparâmetros	Ajuste de hiperparâmetros é caro, exige múltiplos treinos e cresce exponencialmente com a dimensionalidade. Técnicas	Problema adicional: falta de conhecimento para definir espaços de busca realistas. Em dispositivos embarcados, é preciso ajuste ciente do

	como Hyperband ou Bayes não resolvem tudo.	hardware.
Verificação do Modelo	Essencial para garantir robustez, casos de borda e requisitos de negócio. Dividida em codificação de requisitos, verificação formal e testes.	Reforça que acurácia não basta: métricas de negócio e justiça devem ser incluídas.
Verificação – Requisitos	Exemplo da Booking.com: 150 modelos implantados, mas métricas proxy (cliques) não se traduziam em conversão. Métricas devem alinhar modelagem, engenharia e negócios.	Define-se também monitoramento contínuo com base nessas métricas.
Verificação – Formal	Setores regulados, como o bancário, exigem estruturas de risco de modelos de ML. Reguladores publicam diretrizes que obrigam testes extensivos para garantir qualidade.	A verificação formal no ML muitas vezes se traduz em aderir a normas regulatórias.
Verificação – Testes	Simulações são usadas quando testar no mundo real é caro ou perigoso (ex.: veículos autônomos, RL), mas não substituem totalmente. Pequenas diferenças entre simulação e realidade podem causar falhas.	Caso da ISS: modelo treinado em simulação não conseguiu lidar com condições reais. É preciso validar também os dados continuamente para evitar erros ocultos.
Implantação do Modelo	Aplicar princípios de DevOps no ML gera o campo de MLOps/AIOps. Três passos principais: integração, monitoramento e atualização.	Os desafios vão além do software comum: falta de telemetria, rótulos e boas práticas consolidadas.

Implantação – Integração	Benefícios da reutilização de dados e modelos, como no caso do Pinterest que unificou embeddings em vez de manter três modelos separados. Problemas comuns incluem glue code, pipeline jungles e dívida de configuração.	Exemplo do Pinterest: ao unificar embeddings, simplificou pipelines, reduziu esforço de engenharia e melhorou a performance nas tarefas internas.
Implantação – Integração	Benefícios da reutilização de dados e modelos (ex.: Pinterest unificou embeddings). Problemas comuns incluem glue code, pipeline jungles e dívida de configuração.	Recomenda-se colaboração estreita entre engenheiros e pesquisadores, com revisão de código conjunta e versionamento compartilhado.
Implantação – Monitoramento	Definir métricas de dados e modelos a monitorar ainda é um problema em aberto. Há risco de ciclos de feedback, onde o modelo influencia sua própria entrada. Detecção de outliers é essencial, mas difícil por falta de rótulos.	Exemplo: O sistema policial de intervenção precoce nos EUA precisou desenvolver verificações do zero, pois ferramentas prontas não atendiam.
Implantação – Atualização	Modelos precisam ser atualizados, mas o concept drift compromete o desempenho. Pode ser súbito (crise financeira) ou gradual (dados marinhos, manutenção de máquinas).	CD (Continuous Delivery) em ML é mais complexa por envolver código, modelo e dados. Também há risco de perder compatibilidade com usuários humanos — atualizações podem minar a confiança.
Aspectos Transversais – Ética	ML pode reproduzir vieses ocultos nos dados e gerar impactos sociais negativos (ex.: justiça criminal, DRM, reconhecimento facial). É preciso responsabilidade	Exemplos: risco de uso indevido de EMBERS; gender bias em traduções e datasets de face; preocupação com autoria em artes criadas por IA.

	contínua no ciclo de vida.	
Aspectos Transversais – Lei	Leis e regulações variam por país e muitas vezes não acompanham a velocidade da IA (GDPR, FDA, leis de proteção de dados em saúde).	Exemplo: caso DeepMind + NHS que precisou rever acordo por falha em compliance.
Aspectos Transversais – Confiança de Usuários	Confiar em ML exige engajamento dos usuários finais, transparência, explicabilidade e interfaces adequadas.	Síntese: confiança não se constrói só com interpretabilidade. Projetos como Sepsis Watch investiram em comunicação, accountability e UX.
Aspectos Transversais – Segurança	Ataques como data poisoning, model stealing e inversion podem comprometer modelos em produção.	Exemplos: bot Tay (Microsoft) manipulado; ataques de extração em APIs de Google/Amazon; risco de violar GDPR com inversion.
Soluções – Ferramentas/Serviços	Plataformas e libs ajudam em rotinas de deploy, monitoramento e labeling.	Exemplos: MLflow, TFX, Snorkel, AutoML, Alibi Detect.
Soluções – Abordagens Holísticas	Precisamos de práticas além de ferramentas: datasheets para datasets, model cards, DVC, arquiteturas orientadas a dados, frameworks de readiness levels.	Essas propostas tentam padronizar e reduzir o improviso, mas exigem mudança cultural e tempo de adoção.

APÊNDICE 4

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 24 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta Semana, finalizei a leitura do artigo *"A Multivocal Review of MLOps Practices, Challenges and Open Issues"* (45 páginas, 2025), e adicionei uma página com resumo à planilha de acompanhamento dos artigos (Resumos dos artigos lidos durante o processo).

Além disso, iniciei minha pesquisa para compreender as ferramentas de MLOps disponíveis e definir aquelas que serão exploradas nas próximas semanas. A partir dessa análise e reflexão, defini o direcionamento principal da pesquisa: desenvolver um **pipeline de MLOps no Google Cloud Platform (GCP)**, estruturado de forma generalizável e aplicável a diferentes domínios. Como estudo de caso, implementarei o pipeline para **ingestão, classificação e análise de dados de gastos financeiros**.

Também organizei uma seção com links úteis que servirão de apoio para o andamento do estudo: Links úteis

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana, pretendo:

- Aprofundar o estudo das ferramentas de MLOps disponíveis no GCP.
- Selecionar e documentar as ferramentas que serão utilizadas no pipeline.
- Estruturar o desenho inicial da arquitetura do pipeline para ser usada como guia de estudos para as próximas Semanas.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Anotações do artigo "Multivocal Review of MLOps Practices, Challenges and Open Issues"

Práticas de MLOps		
Área	**Práticas identificadas**	**Exemplos / Detalhes**
Automação do ciclo de vida	- Versionamento de dados e modelos - Pipelines reprodutíveis - Testes de modelos - Validação de dados	MLflow, DVC e Pachyderm para versionamento; testes automatizados de performance e fairness; validação contínua da qualidade dos dados
CI/CD para ML	- Automação de treinamento - Automação de validação - Automação de deployment	Jenkins, GitHub Actions, GitLab CI/CD; integração com Kubernetes e Kubeflow; retraining automático quando novos dados chegam
Monitoramento em produção	- Rastreamento de performance - Detecção de drift de dados/conceitos - Alertas e logging	Evidently AI, Prometheus, Grafana; monitoramento de métricas como precisão, recall, latência e uso de recursos
Governança e conformidade	- Gestão de metadados - Auditoria - Rastreabilidade de experimentos	Model cards, datasheets para datasets; ML Metadata Store (TFX); logs auditáveis para compliance (LGPD, GDPR, HIPAA)
Colaboração interdisciplinar	- Integração de perfis técnicos e não técnicos - Definição de papéis claros	Times multifuncionais (cientistas de dados + DevOps + engenheiros de software + product owners); uso de ferramentas colaborativas (Jira, Confluence)
Gestão do ciclo completo	- Planejamento de reuso de modelos - Políticas de descontinuação	Reaproveitamento de modelos pré-treinados (transfer learning); definição de critérios para aposentar modelos
Desafios de MLOps		
Categoria	**Desafios principais**	**Exemplos / Impactos**

Dados	- Qualidade - Versionamento - Segurança	Dados incompletos ou enviesados impactam fairness; necessidade de versionar dados junto com código; compliance de privacidade
Escalabilidade	- Volumes massivos de dados - Modelos cada vez maiores	Treinamento distribuído (Spark, Ray, Horovod); desafios de custo em cloud computing
Automação	- Testes de ML pouco padronizados - Integração parcial com DevOps	Diferença entre testes de software (unitários) e testes de ML (baseados em dados); gaps em pipelines
Monitoramento e manutenção	- Concept drift - Data drift - Falta de alertas adequados	Modelos degradam silenciosamente ao longo do tempo; impacto direto em aplicações críticas (ex: saúde, finanças)
Reprodutibilidade	- Diferenças de ambiente - Falta de controle total do pipeline	Modelos não reproduzem resultados em produção; dependências conflitantes entre bibliotecas
Organizacional/cultural	- Barreiras entre times - Resistência a novas práticas	Silos de dados e conhecimento; falta de cultura DevOps/Agile em times de ciência de dados
Custo e ROI	- Dificuldade em justificar investimentos - Gestão de recursos	Dilema entre custo de GPU/infraestrutura e ganhos de acurácia; pouco entendimento de ROI por stakeholders
Questões em aberto		
Tema	**Pontos a serem explorados**	**Detalhes / Perspectivas**
Arquiteturas em MLOps	Definição de padrões universais	Hoje coexistem frameworks diversos (TFX, MLflow, Kubeflow, SageMaker), mas sem consenso
Métricas de qualidade	Métricas específicas para ML em produção	Além de acurácia: métricas de estabilidade, robustez, fairness e impacto de negócio
Ética e	Integrar fairness,	Modelos precisam ser auditáveis e

explicabilidade	interpretabilidade e accountability	explicáveis para contextos regulados (ex: saúde, finanças, governo)
Educação e capacitação	Profissionais híbridos com skills em ML, DevOps e engenharia de software	Programas de treinamento ainda fragmentados; necessidade de cursos interdisciplinares
Ferramentas e padronização	Fragmentação de mercado e baixa interoperabilidade	Dificuldade de migrar modelos entre plataformas; falta de APIs e formatos comuns
Segurança de ML (MLSecOps)	Proteção contra ataques adversariais, manipulação de dados e vazamento de modelos	Áreas como adversarial ML e model stealing ainda pouco integradas em práticas de MLOps
Sustentabilidade	Consumo energético de grandes modelos impacto ambiental	Adoção de green AI; métricas de eficiência energética como parte do pipeline
Governança de dados	Políticas de uso responsável de dados privacidade e compliance	Necessidade de pipelines auditáveis e alinhados a legislações como LGPD e GDPR

Links úteis

Repositório	O que oferece / destaques	Como pode ajudar
https://github.com/GoogleCloudPlatform/mlops-on-gcp?utm_source=chatgpt.com	Vários exemplos/casos para ML engineering no GCP (workshops, padrão de pipeline, serving, skew, etc).	Usar como "base de referência" para estruturas de pastas, scripts, práticas de infra.
https://github.com/GoogleCloudPlatform/vertex-pipelines-end-to-end-samples?utm_source=chatgpt.com	Template completo para pipelines com Vertex AI, terraform, componentes reutilizáveis, deploy, batch prediction etc.	Pegar pipelines prontos e adaptar ao meu domínio.
https://github.com/GoogleCloudPlatform/mlops-wit	Exemplo end-to-end usando TFX / Vertex, com notebooks que	Posso ver como montar o ciclo completo com notebooks

h-vertex-ai?utm_source=chatgpt.com	mostram dataset, pipeline, monitoramento etc.	controlando etapas.
https://github.com/statmike/vertex-ai-mlops?utm_source=chatgpt.com	Workflows completos (treino, deploy, monitoramento, exemplos múltiplos) para Vertex AI.	Inspiração e código reutilizável para estrutura de pipeline e modulo de monitoramento.
https://github.com/GoogleCloudPlatform/automlops?utm_source=chatgpt.com	Ferramenta que gera arquiteturas de MLOps, provisiona infra + pipelines já integradas com CI/CD, etc.	Posso estudar para comparar com minha arquitetura e ver como “automatizar o automlops”.
https://github.com/bregman-arie/devops-exercises/tree/master	Repositório com perguntas referente a área.	Revisar conceitos e aprender bash.
https://www.cloudskillsboost.google/paths/1283	Curso de treinamento do GCP.	Aprender o pipeline de MLOps dentro da arquitetura do Google.

APÊNDICE 5

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 1 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante a Semana, avancei nos estudos de **MLOps**, com foco em três eixos principais: **qualidade de serviço**, **trade-offs de arquitetura** e **observabilidade e confiabilidade**.

Explorei como projetar sistemas que atendam a requisitos de desempenho (SLIs, SLOs e SLAs relacionados a latência, throughput e disponibilidade), ao mesmo tempo em que consideram requisitos não funcionais como **custo**, **escala**, **segurança**, **compliance** e **portabilidade**. Também aprofundei práticas que fortalecem a resiliência dos sistemas, como *exponential backoff*, *circuit breaker*, *autoscaling*, uso de instâncias *preemptible/spot* e políticas de *rollback*.

O **estudo de caso aplicado** teve papel complementar, servindo como exercício prático em um cenário *near real-time* para reforçar os conceitos abordados.

- Documentação do estudo: Documento auxiliar da Semana
- Esboço da arquitetura inicial do estudo de caso: Canva

Além disso, aprofundei o entendimento sobre **AutoML** e **KaizenML**, conceitos recorrentes nas pesquisas. Para isso, li o capítulo 5 do livro *Practical MLOps: Operationalizing Machine Learning Models* (44 páginas) e realizei uma aplicação prática de exemplo no **GCP**. Também produzi um **desenho simplificado da arquitetura da aplicação** para apoiar a compreensão: Aplicação - Canva.

Por fim, novos conceitos identificados foram incorporados ao **Glossário de MLOps**, que segue em evolução contínua e conjunta como instrumento de apoio ao aprendizado: Glossário de termos na área de MLOPS

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Na próxima Semana, vou estudar sobre **governança e gestão de dados** e **estratégias de deployment**. Também pretendo ler o capítulo 9 *"MLOps for GCP"* do livro *Practical MLOps: Operationalizing Machine Learning Models*.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Desenho simplificado da arquitetura da aplicação prática sobre AutoML e KaizenML:

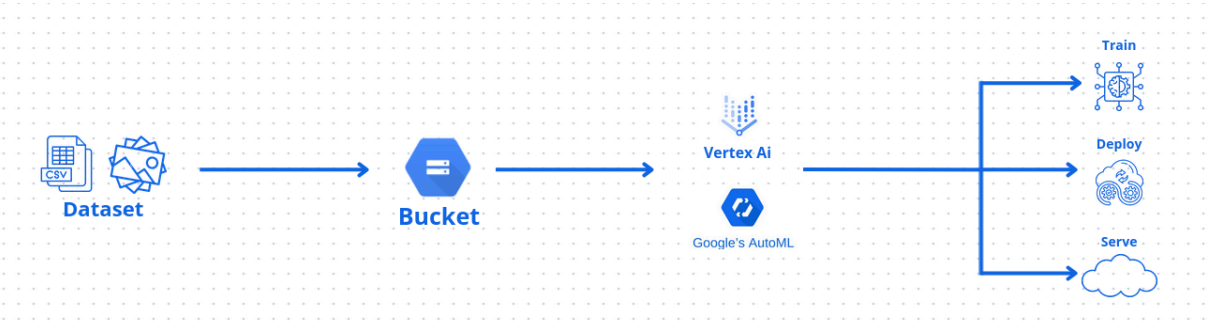


Figura 6. Desenho simplificado da arquitetura da aplicação prática sobre AutoML e KaizenML

Documento auxiliar da Semana 05

Objetivo e requisitos

- Qual é o **caso de uso** (batch, streaming, near-real-time)?
- Quais **SLAs** e **SLOs**? (latência, throughput, disponibilidade)
- Quais **requisitos não funcionais** dominam? (custo, escala, segurança, compliance, portabilidade)

Dados (ingestão, storage, formato, qualidade)

- Quais fontes? Volume diário? Picos?
- **Formato** preferido (Parquet/Avro/JSON) e **particionamento**?
- Onde armazenar "verdade": **BigQuery** (analítico serverless) vs **Cloud Storage** (data lake) vs **Spanner/SQL** (transacional)?
- Estratégia de **esquema** (evolução/compatibilidade) e **lineage**?
- Data Quality: regras, validação e bloqueios de pipeline?

GCP: Pub/Sub (stream), Dataflow (ETL/ELT), Dataproc (Spark), BigQuery (warehouse), Dataplex (governança), Cloud Storage (lake).

Features e versionamento

- Como **definir/servir** features: online x offline?
- **Versionar** features, datasets e modelos?

GCP: Vertex **Feature Store**, BigQuery para offline, Redis/AlloyDB/Feature Online Store para online; **Vertex ML Metadata**.

Treinamento (compute, imagens, reprodutibilidade)

- Treino **serverless** vs **Kubernetes**?
- CPU/GPU/TPU? **Spot/Preemptible** vs on-demand?
- Como empacotar ambiente (Docker), fixar seeds, **data & code snapshot**?

GCP: **Vertex AI Training** (custom/AutoML), **TPU/GPU**, **Artifact Registry**, **Cloud Build**.

Orquestração e CI/CD

- Pipelines declarativos com **cache** e **retries**?
- CI/CD de **dados** e **modelo** (testes, lint, unit/integration e e2e)?

GCP: **Vertex AI Pipelines** (Kubeflow), **Cloud Build/GitHub Actions**, **Cloud Composer** (Airflow) se precisar DAGs multi-serviço.

Registro, aprovação e promoção

- Onde **registrar modelos** e controlar **stages** (Staging→Prod)?
- Gates manuais/automáticos (AUC mínima, bias checks)?

GCP: **Vertex Model Registry**, **Evaluation**, **Model Monitoring** + políticas de promoção.

Servir/Deploy (latência, custo, escalabilidade)

- **Online** (REST/stream) vs **batch** (offline scoring)?
- Autoscales? Multi-region? A/B, canary, shadow?

GCP: **Vertex Online Prediction**, **Batch Prediction**, **Cloud Run** (serverless HTTP), **GKE** (fine-grained), **API Gateway/Cloud Endpoints**.

Observabilidade & responsividade

- Métricas de infra (CPU/RAM), app (latência/erro), **ML** (drift, data skew, qualidade)?
- Alertas, SLOs, rotas de incidentes?

GCP: **Cloud Monitoring/Logging**, **Error Reporting**, **Vertex Model Monitoring** (drift, skew), **BQ Audit Logs**.

Segurança e compliance

- **IAM** mínimo necessário, **VPC-SC**, chaves/segredos, KMS?
- Dados sensíveis: criptografia, DLP, masking, regiões de dados?

GCP: **IAM**, **Secret Manager**, **Cloud KMS**, **VPC-SC**, **DLP**, **Org Policies**.

Custos e otimizações

- Onde estão os **drivers de custo** (armazenamento, consulta BQ, GPUs, egress)?
- Estratégias: **partition/cluster** no BQ, **preemptible**, **autoscaling**, **caching** de pipeline, **reservations/committed use**?

GCP: **BQ slots**, **Recommenders**, **Budgets & Alerts**.

Confiabilidade e continuidade

- Backups, **DR** (RPO/RTO), multi-zona/região, **infra as code**?

GCP: **Terraform/Deployment Manager**, **Multi-region buckets/BQ**, **GKE regional**, runbooks.

Governança e risco de modelo

- Explainability, fairness, auditoria de decisões?
- Política de **lifecycle** (expirar modelos/dados), retenção, trilhas de auditoria.

GCP: **Vertex Explainable AI**, **Model Monitoring**, **Audit Logs**, **Dataplex** (catálogo & políticas).

Batch, Streaming e Near-Real-Time

Batch (lote):

Processa um conjunto de dados de uma vez, em janelas (ex.: a cada hora/dia).
Latência típica: minutos -> horas.
Quando usar: relatórios, reprocessamento, treinamento periódico, custos baixos e previsíveis.
GCP típico: Cloud Storage/BigQuery + Dataflow (batch) ou Dataproc + Vertex **Batch** Prediction.

Streaming (tempo real contínuo):

Dados chegam evento a evento e são processados assim que entram.
Latência típica: segundos (até sub-segundos).
Quando usar: alertas, filas de eventos, fraudes, dashboards "vivos".
GCP típico: Pub/Sub (ingestão) + Dataflow **streaming** + BigQuery streaming inserts + Vertex Online Prediction.

Near-Real-Time (quase em tempo real):

Responde **bem rápido**, mas aceita um pequeno atraso (ex.: 5–60s) para **agregar/emparelhar** dados.
Latência típica: dezenas de **segundos** (às vezes 1–2 min).
Quando usar: atualizações frequentes com **pequena** tolerância a atraso (ex.: enriquecer OCR com regras, atualizar score de risco recente).
GCP típico: ainda Pub/Sub + Dataflow, mas com janelas/buffers curtos; BigQuery com "micro-batch"; Vertex Online/Async.

Como decidir (regra prática):

- Se a **resposta** pode esperar → **Batch**.
- Se precisa reagir **agora** a cada evento → **Streaming**.
- Se pode esperar **alguns segundos** para ganhar **contexto/qualidade** → **Near-Real-Time**.

Latência, Throughput, Disponibilidade:

Latência (tempo por requisição):

- **O que é:** tempo do pedido até a resposta.
- **Por que importa:** é a sensação de "rápido/lento" do usuário. Pior vista pela cauda (p95/p99).
- **Como medir:** p50, p95, p99 end-to-end e por etapa (upload, OCR, classificador, persistência).
- **No GCP:** Cloud Monitoring (SLO de latência), Vertex Endpoint metrics, Cloud Trace.
- **Regra prática:** defina um latency budget (ex.: total 30s → OCR 15s, classif. 5s, I/O 5s, fila 5s).

Throughput (quantidade por tempo):

- **O que é:** taxa sustentada de processamento (req/s no online; itens/min no batch/stream).
- **Por que importa:** capacidade de aguentar pico sem estourar a latência.
- **Como medir:** média e picos (burst). Testes de carga: sustentado vs. explosões 5×.
- **No GCP:** autoscaling (Cloud Run/Vertex/GKE), limites por SKU, filas Pub/Sub.
- **Regra prática:** aumente throughput com paralelismo e autoscaling, sem quebrar o SLO de latência**.**

Disponibilidade (tempo operante com qualidade):

- **O que é:** % do tempo em que o serviço responde com sucesso e dentro do alvo de latência.
- **Por que importa:** traduz confiabilidade percebida. (99.5%, 99.9% etc.)
- **Como medir:** razão de requests 2xx dentro do SLO de latência / total, por janela (5m, 1h, mês).
- **No GCP:** SLOs de disponibilidade no Cloud Monitoring + alertas; multi-zona/região se o negócio exigir.
  Regra prática: disponibilidade conta junto com latência — resposta lenta demais deve não contar como "disponível".

Como eles se relacionam:

- Throughput alto sem capacidade → fila cresce → latência sobe → cai a disponibilidade efetiva.
- Melhorar a latência às vezes reduz throughput (sem batching) ou aumenta custo (mais instâncias).
- Aumentar disponibilidade (multi-zona, instâncias quentes) geralmente ↑ custo.

SLI, SLO e SLA:

SLI - Service Level Indicator: É o indicador que você mede. Ex.: Latência p95, taxa de erro, disponibilidade.

SLO - Service Level Objective: É o objetivo/meta para o SLI. Ex.: "p95 <= 1,8s no mês".

SLA - Service Level Agreement: É o acordo/contrato externo (com cliente) baseado nos SLOs, geralmente com penalidades ou créditos se não cumprir.

Requisitos não funcionais:

Custo:

- **O que é:** quanto você gasta para treinar/rodar/armazenar/transferir.
- **Por que importa:** define viabilidade e ROI.
- **Como medir:** custo por 1.000 docs, custo por hora, custo por requisição.
- **Alavancas no GCP:** preemptible/spot, autoscaling, BigQuery partition/cluster, Cloud Run `min/max instances`, batch vs online, compressão e redução de tokens (se usar API externa), budgets/alerts.

Escala (Scalability):

- **O que é:** capacidade de crescer (mais dados/usuários) mantendo SLOs.
- **Por que importa:** picos e crescimento orgânico.
- **Como medir:** throughput sustentado e em burst, uso de CPU/GPU/memória, fila/tempo de espera.

- **Alavancas no GCP:** autoscaling (Cloud Run/Vertex/GKE), Pub/Sub + Dataflow (fan-out), sharding por partição/tenant, micro-batching em GPU, paralelismo no Dataflow/Dataproc.

Segurança:

- **O que é:** confidencialidade, integridade, disponibilidade (CIA) dos dados e serviços.
- **Por que importa:** risco operacional e legal.
- **Como medir:** cobertura de IAM (princípio do mínimo privilégio), criptografia em trânsito/repouso, rotação de segredos, resultados de pentest.
- **Alavancas no GCP:** IAM refinado, VPC/SVPC, Private Service Connect, Secret Manager, Cloud KMS, VPC-SC, DLP (detecção/mascara de PII), audit logs.

Compliance

- **O que é:** aderência a normas (LGPD, HIPAA, PCI, SOC 2…).
- **Por que importa:** auditorias, multas, reputação.
- **Como medir:** evidências/auditoria por controle (quem acessou o quê e quando), políticas de retenção, data residency.
- **Alavancas no GCP:** regiões conformes, Dataplex (catálogo/políticas), Policy Controller/Org Policies, Audit/Access Transparency, tabelas BQ com políticas de colunas/row-level security, registros de lineage.

Portabilidade

- **O que é:** facilidade de migrar/adaptar para outro provedor/ambiente.
- **Por que importa:** evitar lock-in e manter opções de custo.
- **Como medir:** % de componentes gerenciados proprietários, tempo/esforço estimado de migração.
- **Alavancas no GCP:** padronizar em **containers** (Artifact Registry + Cloud Run/GKE), Kubeflow/Vertex Pipelines com componentes portáveis, Terraform para IaC, formatos abertos (Parquet/Avro), abstrair provedores em código (drivers, interfaces).

APÊNDICE 6

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 9 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Relembrando, na primeira Semana defini que meu foco de estudo dentro de MLOps seria **Infraestrutura e Deploy de Modelos**.

Sendo assim, nas últimas Semanas, concentrei-me principalmente na **Infraestrutura**, buscando compreender como projetar e prover um ambiente robusto, escalável e confiável para dados, treinamento de modelos e execução de pipelines, além de explorar algumas das ferramentas utilizadas nesse contexto.

Nesta Semana, para finalizar essa etapa, realizei a leitura do Capítulo 9 "MLOps for GCP", do livro "*Practical MLOps: Operationalizing Machine Learning Models*". Essa leitura foi essencial para aprofundar meu entendimento sobre as ferramentas do Google Cloud Platform (GCP) aplicadas à prática de MLOps. Realizei também os exercícios e atividades práticas sugeridos, documentando todo o processo para futuras consultas: Registro no Milanote.

Agora, avançando para a etapa de ***Deploy***, iniciei uma pesquisa mais aprofundada sobre as principais abordagens de implantação de modelos, na qual identifiquei as seguintes modalidades:

- Deploy em Batch (Offline)
- **Deploy em Tempo Real (Online)**
- **Deploy em Streaming (ou Near Real-Time)**
- Deploy Embarcado ou no Edge (On-Device)

Além das formas de servir modelos, investiguei também **estratégias de atualização** em produção sem indisponibilidade, como:

- Blue-Green Deployment

- Canary Deployment
- Shadow Deployment

Utilizando o GPT no modo agente, busquei exemplos práticos replicáveis de *Deploy em Tempo Real* ou *Deploy em Streaming* que incorporassem algumas dessas técnicas de atualização. Também realizei a pesquisa com o Deep Research do Gemini, utilizando o mesmo prompt, para ampliar a diversidade das referências encontradas. A partir dos resultados obtidos, fiz uma triagem inicial e selecionei os estudos mais promissores para uma análise mais detalhada, disponíveis neste arquivo: Documento auxiliar da Semana 06

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Nas próximas Semanas, o objetivo será selecionar uma dessas aplicações e replicar a prática, aprofundando o aprendizado em estratégias de deploy e atualização contínua de modelos em produção.

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

Estarei em viagem entre os dias 11 e 19/10 para participar da Robótica 2025, representando o Pequi Mecânico. Por esse motivo, esta será uma Semana atípica, e utilizei a expressão "Próximas Semanas" ao me referir ao planejamento das atividades, já que não terei controle total sobre a rotina durante esse período.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Resumo do Documento auxiliar da Semana 06

O Documento Auxiliar da Semana 06 fala sobre práticas de deploy em tempo real e pipelines de streaming com atualização contínua, destacando estratégias que permitem evoluir modelos de machine learning sem interrupção de serviço. O primeiro texto aborda exatamente esse conteúdo, apresentando uma visão geral das arquiteturas de streaming com Kafka e Flink, explicando como funcionam os componentes essenciais do pipeline e mostrando como técnicas como blue-green e canary garantem zero downtime, inclusive quando há estado envolvido. Esse resumo também enfatiza a importância de savepoints do Flink e discute como a migração para ambientes cloud, como GCP, pode ser estruturada.

Além disso, o Documento Auxiliar da Semana 06 também traz exemplos práticos e replicáveis de deploy. Ele mostra como ferramentas como KServe, Iter8 e Seldon Core podem ser usadas para realizar canary rollout, blue-green deployment, A/B testing e shadow deployment em clusters Kubernetes. Esses exemplos demonstram como dividir tráfego entre versões, validar desempenho com SLOs, usar observabilidade para decisões automatizadas e aplicar técnicas semelhantes em pipelines de streaming com Kafka e microserviços. Tudo isso reforça a ideia central da semana: atualizar modelos em produção de forma segura, gradual e sem downtime.

Os dois parágrafos acima oferecem uma visão condensada dos principais conceitos, ferramentas e estratégias discutidos na Semana 06, funcionando como um guia rápido para entender o conteúdo central do material. Para consultar todos os tutoriais detalhados, manifestos Kubernetes, códigos, exemplos completos e explicações aprofundadas, o documento está disponível na íntegra pelo link: Documento auxiliar da Semana 06

Registo no Milanote

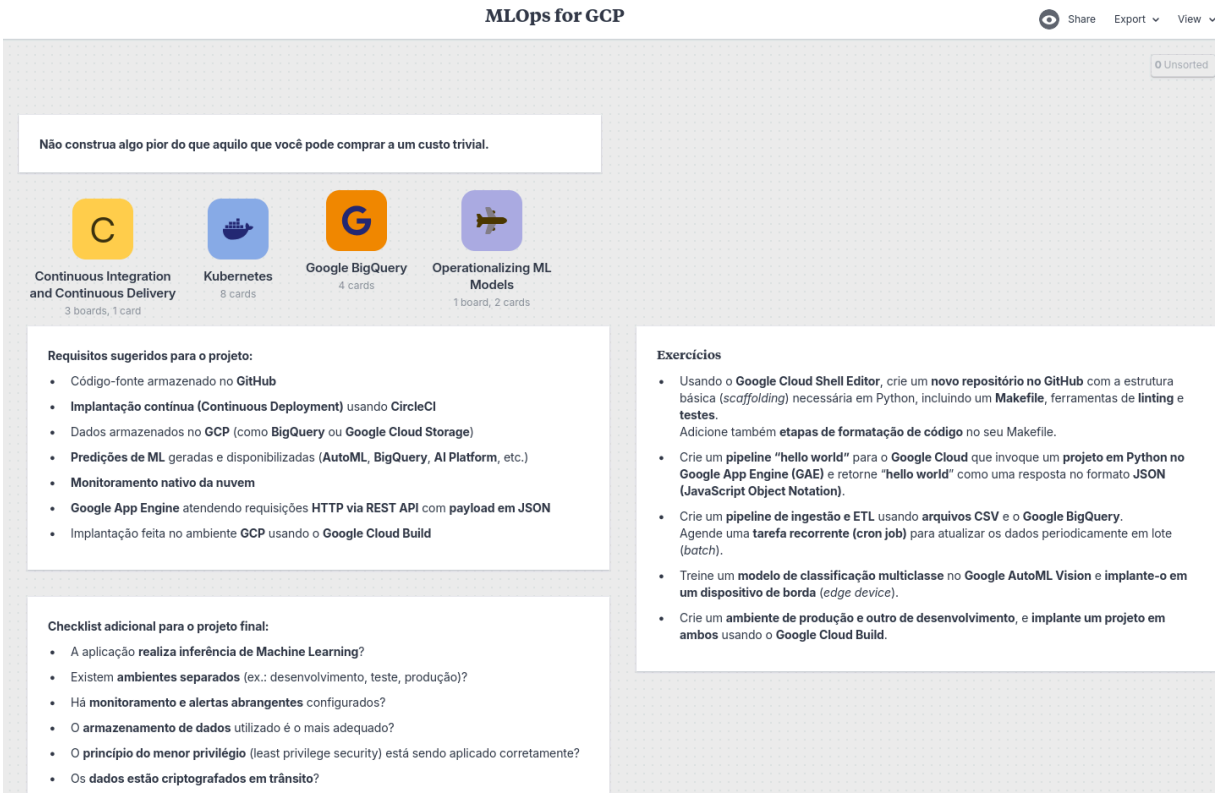


Figura 7. Tela principal do Milanote com as anotações da Semana

O mural no Milanote é organizado com vários cards sobre MLOps no Google Cloud Platform. Os tópicos estão distribuídos em blocos que podem ser clicados e abertos individualmente, facilitando a navegação entre temas como CI/CD, Kubernetes, BigQuery e operacionalização de modelos. O mural funciona como um painel visual onde cada card leva a anotações mais detalhadas, listas e checklists utilizados no planejamento do pipeline em GCP. Pode ser consultado na íntegra pelo link: https://app.milanote.com/1V6el71bSEnpbX?p=QPfrJHOBkat

APÊNDICE 7

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 16 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta Semana, dei continuidade à etapa de **Deploy de Modelos**, com foco em consolidar o aprendizado sobre as principais estratégias de implantação e aplicá-las a um caso prático dentro do **Google Cloud Platform (GCP)**.

Após o levantamento teórico realizado na Semana anterior, escolhi o artigo **"InstaGeo: Compute-Efficient Geospatial Machine Learning from Data to Deployment"** como estudo de caso para aplicar os conceitos de MLOps na prática. Esse trabalho propõe um pipeline completo — do preparo de dados geoespaciais ao deploy de modelos otimizados por destilação — o que o torna ideal para explorar técnicas de implantação e escalabilidade em nuvem.

A partir dessa referência, realizei o mapeamento arquitetural do InstaGeo para os serviços do GCP, identificando correspondências entre suas etapas e componentes da plataforma, além disso preparei os dados geoespaciais necessários para o experimento, realizando o processamento inicial, limpeza e organização dos arquivos em **Cloud Storage**, seguindo o padrão de entrada esperado pelo pipeline.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Na próxima Semana, pretendo dar continuidade à etapa prática de Deploy, avançando na automação do pipeline de MLOps no GCP. O foco será integrar as etapas de preparo de dados, treinamento e deploy em um fluxo orquestrado com o Vertex AI Pipelines, garantindo reprodutibilidade e rastreabilidade em todo o processo.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 8

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 22 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na Semana passada, foi concluída a configuração da infraestrutura inicial no **Google Cloud Platform (GCP)**, que servirá como base para o ambiente de execução do estudo. Nessa etapa, foram criados os principais recursos necessários, como o **Artifact Registry** (repositório para armazenar imagens de containers), a **Service Account** (conta de serviço usada para autenticação segura entre os componentes) e os **Buckets** (espaços de armazenamento na nuvem).

Também foi realizada a **containerização** e **execução** do componente **chip_creator**, responsável pelo processamento inicial dos dados. Após empacotar o componente em um container, ele foi enviado ao Artifact Registry e configurado para execução automática no **Cloud Run**, concluindo assim a fase de dados do pipeline.

Nesta Semana, foi iniciada a fase voltada ao **treinamento dos modelos**, denominados "professor" e "aluno", utilizando o **Vertex AI**, plataforma do GCP voltada a soluções de inteligência artificial. O objetivo desta etapa é **automatizar o pipeline de treinamento**, permitindo que os modelos sejam treinados, avaliados e registrados automaticamente no **Model Registry** (repositório central de modelos do Vertex AI).

Essa fase está em andamento e busca tornar o ciclo de treinamento totalmente automatizado, modular e escalável, integrando todas as etapas do processo dentro do ecossistema do Google Cloud.

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Na próxima Semana, o foco será dar continuidade ao desenvolvimento do pipeline no Vertex AI, avançando para a etapa de Deploy.

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 9

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 5 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luísa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante a Semana, na fase de treinamento no Vertex AI, foi avaliado o custo de execução do modelo "professor", que possui cerca de 96 milhões de parâmetros. A estimativa indicou um custo entre US$ 40 e US$ 60 por execução, considerando instâncias com GPU A100 e tempo médio de treino de aproximadamente 10 horas. Esse valor tornaria a etapa financeiramente inviável para os recursos disponíveis. Assim, optei por não realizar o treinamento completo, limitando-me a compreender o processo e avançar diretamente para a etapa de Deploy.

Na fase de Deploy, foi realizada a **implantação completa do InstaGeo-E2E-Geospatial-ML**, framework open source da InstaDeep voltado ao aprendizado de máquina geoespacial. O objetivo foi preparar e executar o ambiente completo da aplicação no **Google Cloud**. Esse ambiente, chamado de full-stack, reúne todos os componentes necessários para o funcionamento do sistema, desde a interface visual (frontend), passando pela camada de processamento de dados (backend) e pelos workers responsáveis por tarefas automáticas, até o Redis, que armazena informações temporárias para acelerar a execução. Todo o conjunto foi implantado dentro de uma instância de VM no **Compute Engine**.

Inicialmente, foi criada a instância de VM "residencia", configurando automaticamente os serviços do **Compute Engine (compute.googleapis.com)**, as políticas do **VM Manager** e a proteção de dados associada ao disco de inicialização, conforme registrado no log de operações do GCP. Em seguida, foram instalados e ativados o **Docker** e o **Docker Compose**, necessários para orquestrar os contêineres do InstaGeo.

Durante o primeiro deploy, ocorreu uma **falha por falta de espaço em disco**, já que a instância possuía apenas 10 GB de armazenamento. Para corrigir o problema, o disco foi expandido para

30 GB por meio do Console do GCP, seguido da instalação do pacote **cloud-guest-utils** e da execução dos comandos `growpart` e `resize2fs`, que permitiram redimensionar o sistema de arquivos e utilizar todo o novo espaço disponível.

Após o redimensionamento, foi criada uma **regra de firewall** para permitir conexões seguras via **Cloud Identity-Aware Proxy (IAP)**, com origem no intervalo `35.235.240.0/20` e porta 22. Essa configuração viabilizou o acesso à VM diretamente pelo Console do GCP, garantindo segurança e conectividade durante a operação.

O deploy do InstaGeo foi concluído com sucesso, e o sistema agora opera em uma máquina virtual no Google Cloud, pronta para processar dados geoespaciais e realizar experimentos de modelagem. A implantação segue o formato de **Deploy em Batch (Offline)**, em que o processamento ocorre em lotes de dados sob demanda, assegurando organização, reprodutibilidade e integração com os serviços já configurados do GCP.

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Na próxima Semana, pretendo organizar o repositório (https://github.com/luisafrancielle/residencia-mlops) com os códigos utilizados, e, como ainda restam créditos após o Deploy, realizar um teste de treinamento do modelo “aluno”.

Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 10

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 12 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Luisa Francielle Oliveira Fagundes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Relembrando, na primeira Semana defini que meu foco seria me especializar em **MLOps**. Com o tempo, afunilei meu interesse para **Infraestrutura e Deploy de Modelos**. Ao longo das Semanas, li artigos e papers para compreender os fundamentos, principais conceitos e técnicas de deploy.

Como Estudo de Caso, escolhi o **framework open-source InstaGeo**, apresentado em um paper que descrevia um pipeline completo de aprendizado geoespacial. O projeto tinha escopo local, então meu objetivo foi **migrar o pipeline para o Google Cloud Platform (GCP)**, com ênfase na etapa de **deploy**.

Com essa aplicação, pude consolidar o entendimento de ferramentas e conceitos-chave, como:

- **Containerização:** Docker e Artifact Registry.
- **Infraestrutura serverless:** Cloud Run e Cloud Functions.
- **Orquestração e automação:** Vertex AI Pipelines e Kubeflow.
- **Treinamento e deploy escalável:** Vertex AI Training, Endpoints, Batch Prediction e instâncias de VM.
- **Monitoramento e análise de custo-performance:** comparação entre modelos destilados e originais quanto à latência, consumo de recursos e custo operacional.

Durante a última Semana, foquei em preparar a execução do treinamento do modelo *"aluno"* no **Vertex AI**, etapa essencial para completar o pipeline do InstaGeo. Nesse processo, configurei o projeto no Google Cloud Platform, habilitei as APIs necessárias e preparei o ambiente de execução com Docker e Artifact Registry. Consegui construir a imagem Docker localmente com sucesso, incluindo todas as dependências geoespaciais e de GPU, mas o *push* para o Artifact

Registry foi interrompido devido às limitações do Cloud Shell, que encerrou a sessão antes do upload completo. Por conta disso, o treinamento no Vertex AI ainda não pôde ser executado.

Uma alternativa encontrada é refazer o processo utilizando o **Cloud Build**, que realiza o build e upload diretamente na nuvem, evitando interrupções e permitindo concluir o pipeline de forma estável.

Por fim, ao organizar o repositório criado para registrar os passos e códigos desenvolvidos ao longo das Semanas, percebi que seria mais interessante compartilhar esse processo em formato de artigos no Medium. Dessa forma, escrevi por aqui: medium.com/@luisafrancielle

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Escrita do Trabalho de Conclusão de Curso.

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

Uma observação que gostaria de fazer é que, durante a primeira Semana, eu ainda tinha muitas dúvidas sobre qual área gostaria de me especializar. E, ao longo das Semanas, depois de escolher MLOps e começar a trabalhar com isso, senti uma grande satisfação por finalmente me encontrar em algo. Hoje, com certeza, sei muito mais do que sabia no início, porém ainda tenho muito o que aprender, mas isso é o divertido.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Artigo publicado no Medium:

Figura 8. Artigo no Medium “Como criar um pipeline de Machine Learning no GCP com o InstaGeo”