**Pessoa de Infra**

1. **Cria os Data Lakes no S3**
   * **Exemplo: buckets nome-do-projeto-raw (para dados brutos), nome-do-projeto-processed (se quiser separar dados limpos) e nome-do-projeto-models (para armazenar .h5).**
2. **Cria um evento EventBridge para “trigar” EC2 com MLflow na chegada do arquivo no Data Lake /raw**
   * **Assim que novos CSVs ou arquivos chegam em raw/, o EventBridge chama uma instância EC2 (ou tarefa Fargate) onde a aplicação de Feature Store será executada (limpeza/categorização).**
3. **Cria um repositório no GitHub para a aplicação de Engenharia de Feature Store**
   * **Exemplo: github.com/MinhaOrg/feature-store-engineering.**
4. **Cria um pipeline para Engenharia de Feature Store (com testes)**
   * **Em GitHub Actions (ou outra ferramenta), que faz build do container (Python, Feast, MLflow, etc.) e roda testes unitários.**
5. **Cria EC2 ou Fargate para rodar o container (ou EC2) de Engenharia de Feature Store com MLflow e Feast**
   * **Poderá ser um ECS Fargate sob demanda: quando o evento dispara, sobe o contêiner, processa os dados e grava no DynamoDB.**
   * **Se preferir EC2 fixo, configure IAM Roles para acesso S3, DynamoDB, etc.**
6. **Cria observabilidade para aplicação de Engenharia de Feature Store**
   * **Logs (CloudWatch Logs), métricas de execução (ex.: tempo de processamento), alarmes (caso falhe o processo).**
7. **Cria o DynamoDB para repositório de FeatureStore**
   * **Tabelas específicas para usuários com histórico e para cold start (ou uma única tabela com partição userId e tipo).**
8. **Cria um repositório no GitHub para a aplicação de Treinamento (com teste unitário)**
   * **Exemplo: github.com/MinhaOrg/model-training.**
9. **Cria EC2 ou Fargate para rodar o container (ou EC2) de treinamento dos modelos**
   * **O container (ou script) acessa S3 e/ou DynamoDB para ler dados de treino e gera .h5.**
10. **Cria observabilidade para aplicação de Treinamento**
    * **Logs no CloudWatch, monitorando CPU/Mem durante o treino.**
11. **Cria S3 para armazenar os .h5 do modelo / cada pasta tem seu modelo**
    * **Pode ser o bucket nome-do-projeto-models, com subpastas por versão ou tipo de modelo (ex.: modelX/v1, modelX/v2).**
12. **Cria um evento no EventBridge para quando o modelo .h5 é salvo em uma das pastas do modelo, “triga” a Lambda que salva metadados**
    * **Detecta “ObjectCreated” no bucket nome-do-projeto-models.**
13. **Cria uma Lambda para fazer o commit no repositório (metadados) no GitHub, dizendo que existe um novo .h5**
    * **Ex.: atualiza um arquivo models\_metadata.csv ou gera um commit automático no repositório, sinalizando a nova versão do modelo.**
14. **Cria um workflow no GitHub onde rodará um pipeline para deploy de uma nova inferência usando o novo .h5, nesse pipeline haverá testes**
    * **Ao receber o aviso (commit), o pipeline baixa o .h5 e gera o contêiner de inferência.**
    * **Roda testes unitários/integrados (endpoint /predict etc.).**
15. **Cria um EC2 ou Fargate para receber o container do deploy do GitHub que contém a inferência com o novo .h5 na AWS Fargate ou EC2**
    * **Pode ser um serviço ECS com Load Balancer expondo a API de inferência.**
16. **Cria observabilidade para aplicação de Inferência**
    * **Logs no CloudWatch, métricas de latência, alarmes de erro 4xx/5xx.**

**Pessoa Engenheiro de Dados**

1. **Limpar e categorizar os dados CSVs da pasta “Itens”, separando por regional, nacional ou internacional / e subcategorias**
   * ***(Pelo que vi, a maioria é regional, mas tem subcategorias... seria interessante usar algum ML para categorizar a notícia, como Bag of Words, TF-IDF, Word2Vec, Bert, transformer... e os dados que têm categoria na URL podem servir de treinamento para esses modelos.)***
   * **Isto pode ser feito na aplicação de Engenharia de Feature Store, usando Python + bibliotecas de NLP.**
2. **Criar FeatureStore para as pessoas que têm histórico (pasta treino)**
   * **Para cada cliente, armazenar quais categorias foram acessadas, data e qual notícia regional ele acessou (possível detecção automática da região).**
   * **Esses dados podem ser armazenado em csv, servindo para recomendações.**
3. **Criar o FeatureStore para Cold Start(pasta treino)**
   * **Obter todos os dados de histórico dos primeiros acessos (categorias e datas) daqueles que foram “marinheiros de primeira viagem”.**
   * **Esses dados podem ser armazenado em csv, servindo para recomendações quando não há histórico do userId ainda.**
4. **Na inferência, a ideia é obter o userId**
   * **Se achar esse userId na featureStore de cliente que tem histórico, carrega as features (categorias, data, regiões acessadas) e o modelo usa essas features para recomendar notícia.**
   * **Se o cliente for cold start, obtém dados da FeatureStore de Cold Start, recomendando a categoria de notícia com data mais atual (considerando comportamento de novos usuários naquele período).**
   * ***(Exemplo: em 2020, cold start acessava mais notícias de saúde por conta da COVID; em 2025, pode ser política internacional, etc.)***
5. **Usar Python com Feast e MLflow**
   * **Feast para gerenciar offline/online features (DynamoDB + S3).**
   * **MLflow para versionar modelos (logar métricas) e gerar .h5.**