Olá!

Chegamos na etapa do teste prático para a vaga de **Engenheiro de Dados**.
Neste momento, esperamos que você seja capaz de resolver um desafio baseado em um cenário proposto, para que possamos avaliar você tecnicamente.

Entregável

Esperamos receber o seu desafio da seguinte forma:

- O código do seu projeto precisa estar disponível no Github (precisa ser público);
- Se você utilizar algum recurso de infraestrutura para processamento (por ex: cluster EMR, Lambda, etc), nos mostre como foi sua estratégia de provisionamento;
- Se você utilizar notebooks, nos mostre sua linha de raciocínio, contando a história dos seus passos;
- Se o processo utilizar múltiplas etapas, explique a arquitetura envolvida para que possamos entender melhor o fluxo;

Lembre-se: não há solução certa ou errada. Existem diversas maneiras diferentes de alcançar o mesmo objetivo e o importante neste desafio, é conseguirmos avaliar sua linha de raciocínio, a clareza do seu código, o nível de organização dos seus projetos e sua criatividade.

Critérios de avaliação

1. Manutenibilidade;
2. Simplicidade;
3. Testabilidade;
4. Documentação;

Se tiver qualquer dúvida é só perguntar, ligar, entrar em contato ou enviar sinais de fumaça - estaremos à disposição.

Um abraço e sucesso,
Time Pismo

Cenário

Considere um fluxo onde diversas aplicações emitem eventos como resultado do seu processamento. Um pipeline é responsável por consumir estes eventos e disponibilizá-los, de tempos em tempos, como arquivos em um diretório [1].

Payload de exemplo de um evento salvo:

```
{
  "event_id": "3aaafb1f-c83b-4e77-9d0a-8d88f9a9fa9a",
  "timestamp": "2021-01-14T10:18:57",
  "domain": "account",
  "event_type": "status-change",
  "data": {
    "id": 948874,
    "old_status": "SUSPENDED",
    "new_status": "ACTIVE",
    "reason": "Natus nam ad minima consequatur temporibus."
  }
}
```

Considerações

- Todos os eventos respeitam um **contrato base**, contendo os campos `event_id`, `timestamp`, `domain`, `event_type` e `data`:
  - O campo `event_id` representa um identificador único de cada evento;
  - O campo `timestamp` representa a data/hora de geração do evento;
  - A combinação dos campos `domain` + `event_type` representam um único tipo de evento;
  - O campo `data` corresponde ao **payload do evento** e seu formato é livre, a ser definido por cada aplicação que o produz;
  - Dentro do objeto `data` existe um campo `id` que representa o identificador único da entidade, conforme exemplos abaixo:

    - Quando o `domain` for igual a `account`, representa o identificador de uma conta;
    - Quando o `domain` for igual a `transaction`, representa o identificador de uma transação;
- Existem tipos diferentes de eventos (combinação `domain + event_type`) misturados nos arquivos;
- Existe a possibilidade de existirem eventos duplicados nos arquivos;

O desafio

Dado o contexto acima, queremos que você desenvolva uma solução capaz de consumir um conjunto de arquivos contendo uma amostra de eventos [2] e separá-los em diretórios [1] distintos por cada tipo evento.

Observações técnicas importantes

1. Use Python;
2. O formato de saída dos arquivos deve ser parquet;
3. Para o caso de eventos duplicados, somente a última versão deve ser mantida;
4. No diretório [1] de saída, deve ser utilizado um particionamento por data do evento (ano, mês, dia);

Dicas úteis

Aqui vão algumas dicas que podem lhe ajudar no desafio:

- Spark é um bom candidato para o trabalho, mas nem tudo precisa ser feito dentro dele;
- O conteúdo do campo `data` precisa estar presente no resultado final, mas não precisa ser um *`struct`*;

Notas

[1] Considere como diretórios de origem, temporário e de destino, um local acessível pelo seu script, independente da arquitetura escolhida. Pode ser um caminho na estação local, um bucket S3, um mapeamento no cluster Hadoop, etc. O importante é que seu script possua acesso a ele e realize o processamento.

[2] A amostra disponibilizada foi criada baseada em um cenário real em nosso ambiente, porém todos os dados gerados representam uma simulação criada com a biblioteca Faker (https://faker.readthedocs.io).