# Desafio ElasticSearch Stone

O desafio que propomos é elaborar uma solução de indexação event driven no **ElasticSearch**

Essa solução precisa ser capaz de escalar para suportar o seguinte cenário:

- Volume diário médio de input de dados: 20GB
- Toda consulta recebe um range de datas e o id do cliente.
- 85% das consultas se referem aos últimos 7 dias.
- 99% das consultas se referem aos últimos 30 dias.
- O período de retenção dos dados deve ser de 12 meses.
- Os eventos chegam através de um kafka

# Modelo de Dados

Temos apenas uma entidade, Transaction:

```
{
   "id": "00000000-0000-0000-0000-000000000000",
   "type": "card | pix | boleto",
   "created_at": "2022-01-01",
   "client_id": "00000000-0000-0000-0000-000000000000",
   "payer_id": "00000000-0000-0000-0000-000000000000",
   "amount": 0
}
```

Para a execução deste desafio, você pode gerar dados aleatórios.

# Requisitos de Entrega

Nesse case, esperamos que você faça**, pelo menos, a entrega do (Tier 0)** e elaborar uma definição de qual a melhor maneira de persistir esses dados no **ElasticSearch**. Os tiers seguintes são **opcionais**, ou seja, você tem liberdade para implementá-los de acordo com sua disponibilidade de tempo ou a seu critério.

## Tier 0 (obrigatório):

- Escreva uma definição de como um sistema que atenda a **todos os tiers** definidos neste desafio deve ser implementado.
    - Defina a política de indexação, shardeamento, replicação e mapping na camada de persistência.
    - Defina quais camadas e componentes esse sistema deve ter.
    - Defina como será feito a deleção de dados antigos no Elastic.

    - Tenha em mente que custo é um fator importante. Caso esse sistema vier a ser implementado, ele deve ser eficiente a nível de custo.
- O formato de entrega pode ser um arquivo PDF ou documento Google Docs.

## Tier 1 (opcional) :

- Implementar um script que envie uma massa de dados referente a um dia para o kafka
- Formato de entrega: repositório no Github com docker-compose completo para execução

## Tier 2 (opcional):

- Implementar um consumidor que leia os dados do kafka e indexe no ElasticSearch.
- Formato de entrega: repositório no Github com docker-compose completo para execução

## Tier 3 (opcional):

- Uma API de leitura no ElasticSearch com dois endpoints:
    - Listagem de transações
        - Filtros: intervalo de datas e client_id
    - Endpoint que retorne o valor total diário (soma de todos os amounts por dia), discriminando também por tipo de transação
        - Filtros: intervalo de datas e client_id

## Formato de Entrega

Para os tiers 1, 2 e 3, entregue um repositório versionado no Github com docker-compose completo para sua execução. Inclua também variáveis de ambiente básicas. Quanto mais fácil for executar o seu projeto, melhor.

# O que gostaríamos de ver e conversar sobre

- Como você toma decisões de projeto.
    - Como você escolheu as tecnologias utilizadas?
        - libs, dependencias, etc
    - Como você decidiu organizar o seu código?
    - Como você resolveu o problema proposto?
- Quão fácil é entender o seu código.
- Quão fácil é rodar o seu código.
- Como você documenta suas aplicações.
- Como você testa seu código.

# Diferenciais

- Testes automatizados são opcionais mas é muito bom ter (especialmente testes unitários.
- Documentação justificando eventuais decisões que você tomou ao longo da elaboração deste desafio.
- Ter a aplicação funcionando em algum lugar (Heroku, ElasticCloud, AWS, Digital Ocean, Google etc.).
- Ter um processo automatizado de deploy (utilizando TravisCI, CircleCI, Github Actions).
- Ter um processo de integração contínua.
- Preocupações de segurança de dados.
- Preocupações com a facilidade de manutenção e processo de desenvolvimento.

# Lembrete

- O ideal é que a sua solução cumpra todos os requisitos, mas se isso não for possível, gostaríamos de ver o seu código mesmo assim.