Desafio

Instruções

1. Escolha um dos cases abaixo: Twitter ou Cats API
   1.2. O Twitter, às vezes, demora para enviar as chaves de acesso. Se isso travar seu progresso, você tem a API de gatos para consumir.
2. Utilize o github ou afins para versionar seu projeto
3. Você tem até 7 dias corridos para concluir as atividades. (O Resultado deverá ser enviando após os 7 dias, independente do ponto do desenvolvimento);
   3.1. **Foque 1º em sua zona de conhecimento, e progrida o máximo que puder**. Caso não consiga concluir todas as atividades, por favor, entregue o que foi feito até a data solicitada.
4. Fique à vontade para utilizar tecnologias, frameworks e técnicas não citadas nas atividades.
5. Recomendamos a utilização do docker (http://www.docker.com) para montagem do ambiente, **mas fique à vontade para usar outras alternativas**.
   5.1 Caso opte pela utilização do docker, publique os Dockerfiles no repositório do projeto ou deixe a imagem publicada no Dockerhub.

Obs.: Para todos os itens iremos considerar a documentação como parte da entrega.

Case Twitter

1. Crie uma aplicação em Java para coletar as últimas postagens do Twitter, dada uma determinada #tag.
    a. Por padrão o Twitter disponibiliza as 100 últimas postagens.
    b. Caso não tenha 100 twittes, colete todas que vierem.
    c. Não há necessidade de coletar mais do que 100 twittes, dada um #tag.
    d. Atente-se a limites e timeouts xD

2. Use uma base de dados para armazenar as informações.

3. Colete e armazene as mensagens, na base de dados, para as #tags listadas abaixo: #openbanking, #**remediation**, #devops, **#sre**, #microservices, #**observability**, #oauth, #**metrics**, #**logmonitoring**, #**opentracing**

4. Utilizando uma linguagem de sua preferência, sumarize e grave os dados para conseguir listar as informações:
    a. Quais são os 5 (cinco) usuários, da amostra coletada, que possuem mais seguidores?
    b. Qual o total de postagens, agrupadas por hora do dia (independentemente da #hashtag)?
    c. Qual o total de postagens para cada uma das #tag por idioma/país do usuário que postou;

5. Crie APIs REST que permita o consumo dos três itens anteriores. A API deverá expor métricas de execução.

6. Crie uma coleção no postman ou insomnia para consumir as APIs criadas.

7. **Envie seus logs para uma ferramenta** de logging (exemplos: Elastic Search, Splunk, Graylog ou similar) **e crie** uma query que mostre em tempo real os eventos que acontecem na execução da API criada acima, exemplos (Warning, Erro, Debug, Info, etc).

8. **Utilize uma ferramenta de métricas para sua infraestrutura** (exemplos: Prometheus, Zabbix, cloudwatch ou similar) **e também** crie 3 dashboards que mostrem em tempo real a quantidade de execução, a latência (tempo de execução) e quantidade de erros da API criada acima.

9. **Utilize técnicas ou descreva formas de aplicar os temas qualidade e teste** na sua aplicação (teste unitário, teste regressivo, teste integrado, teste de performance, teste de resiliência etc)

10. Publique o projeto no Github e documente no README.md os itens abaixo:
    a. Documentação do projeto
    b. Documentação das APIs
    c. Documentação de arquitetura
    d. Documentação de como podemos subir uma cópia deste ambiente
    e. Manual com prints dos Logs (item 7) e os Dashboards (item 9) explicando cada um deles.

10. **Bônus. Se você chegou até aqui e quiser, pode criar um frontend para sua aplicação para facilitar a experiencia do usuário.**

Case The Cat API

1. Crie uma aplicação na linguagem que desejar para coletar as seguintes informações da API de Gatos (https://thecatapi.com/):
    a. Para cada uma das raças de gatos disponíveis, armazenar as informações de origem, temperamento e descrição em uma base de dados. (se disponível)
    b. Para cada uma das raças acima, salvar o endereço de 3 imagens em uma base de dados. (se disponível)
    c. Salvar o endereço de 3 imagens de gatos com chapéu.
    d. Salvar o endereço de 3 imagens de gatos com óculos.

2. Use uma base de dados para armazenar as informações (pode ser um banco de dados em memória como H2 ou HSQLDB)

3. Utilizando a linguagem Java, crie 4 APIs REST:
    a. API capaz de listar todas as raças
    b. API capaz de listar as informações de uma raça
    c. API capaz de a partir de um temperamento listar as raças
    d. API capaz de a partir de uma origem listar as raças

4. As APIs acima deverão expor métricas de execução.

5. Crie uma coleção no postman ou insomnia para consumir as APIs criadas.

6. **Utilizando uma ferramenta de logging** (exemplos: Elastic Search, Splunk, Graylog ou similar), crie uma query que mostre em tempo real os eventos que acontecem na execução da API criada no item 6, exemplos (Warning, Erro, Debug, Info, etc).

7. **Utilizando uma ferramenta de métricas** (exemplos: Prometheus, Zabbix, cloudwatch ou similar), crie 3 dashboards que mostre em tempo real a quantidade de execução, a latência (tempo de execução) e quantidade de erros da API criada acima.

8. **Utilize técnicas ou descreva formas de aplicar os temas qualidade e teste** na sua aplicação (teste unitário, teste regressivo, teste integrado, teste de performance, teste de resiliência etc)

9. Publique o projeto no Github e documentar em um README.md os itens abaixo:
    a. Documentação do projeto
    b. Documentação das APIs
    c. Documentação de arquitetura
    d. Documentação de como podemos subir uma cópia deste ambiente localmente
    e. Manual com prints dos Logs (item 6) e os 3 Dashboards (item 7).