Arquitetura desafio melhor envio

Aplicação que processa o arquivo logs.txt

Tecnologias utilizadas:

- Node.js
- Javascript
- Mongodb
- Sentry
- Docker
- Docker-compose

Motivos de usar essas tecnologias:

- **Node.js e Javascript:** entre as tecnologias sugeridas essa é tecnologia que mais se encaixa, pois como eu não tinha conhecimento Golang e Python que seriam ótimas ferramentas para processar os dados, então fiz em Node.js por também ser um tecnologia que tem uma boa performance.
- **Mongodb:** eu escolhi esse banco de dados por não ter uma estrutura fixa, e como são dados de logs podem existir variações para cada registro. Nesse caso se encaixa adequadamente a minha necessidade.
- **Sentry:** essa é uma serviço onde permite que você envie erros da sua aplicação e também permite criar alertas por email, slack e etc. Exemplo de uso: essa aplicação eu construí ela de forma que ela possa ser uma ferramenta de CLI, pois assim poderia processar logs constantemente basta apenas colocar um cron job para executar o comando da ferramento é assim processar os logs. Nesse cenário, caso ocorra um erro será disparado um alerta via email para o time de desenvolvimento para ser analisado.
- **Docker e Docker-compose:** são ferramentas que vão permitir rodar a aplicação de forma simples e evitar problema de incompatibilidade causado por um desenvolvedor ter um versão de um ferramento é outro desenvolvedor tem um versão mais antiga.

Fluxo de funcionamento da aplicação:

- Executa o comando para rodar a aplicação no docker
- Irá pegar o arquivo é irá ler ele como stream(leitura linha por linha) para evitar carregar todo o conteúdo do arquivo para memória é quando a acumula 50

linhas do arquivo é feito insert em batch no mongodb. O insert em batch é para melhorar a performance do script.
- Caso ocorra um erro o erro é enviado para o Sentry.
- O Sentry notifica por email sobre o problema que está ocorrendo.

Aplicação para gerar o relatório em csv

Tecnologias utilizadas:

- Node.js
- Javascript
- Mongodb
- S3
- RabbitMQ
- Docker
- Docker-compose

Motivos de usar essas tecnologias:

- **Node.js e Javascript:** entre as tecnologias sugeridas essa é tecnologia que mais se encaixa, pois como eu não tinha conhecimento Golang e Python que seriam ótimas ferramentas para processar os dados, então fiz em Node.js por também ser um tecnologia que tem uma boa performance.
- **Mongodb:** eu escolhi esse banco de dados por não ter uma estrutura fixa, e como são dados de logs podem existir variações para cada registro. Nesse caso se encaixa adequadamente a minha necessidade.
- **Docker e Docker-compose:** são ferramentas que vão permitir rodar a aplicação de forma simples e evitar problema de incompatibilidade causado por um desenvolvedor ter um versão de um ferramento é outro desenvolvedor tem um versão mais antiga.
- **S3:** é um serviço da Aws que permite armazenar arquivos é que tem alguns benefícios: não preciso me preocupar com limitações de hardware, no momento que faço o upload para s3 ele cria o link para acessar o arquivo então não preciso implementar, segurança oferecida pelo serviço, disponibilidade do serviço e a simplicidade em trabalhar com esse serviço.
- RabbitMQ: essa tecnologia foi utilizada devido eu ter que gerar relatório em csv de consultas feitas em uma coleção(tabela) que armazena logs o volume de dados pode crescer muito é eu poderia afetar a performance da api se eu coloca-se a parte de gerar o csv na api. Baseado nisso é pego os dados

necessários para uma fila no RabbitMQ onde terá um aplicação esperando para consumir esses dados, ela irá fazer a consulta no banco de dados, gerar o csv, armazenar no s3 e depois enviar por email. O porquê de RabbitMQ é não outras tecnologias: eu já conheço a tecnologia, é open source permitido que eu não fique preso a uma cloud provider e por ter muitas pessoas utilizado tem vários materiais sobre a tecnologia caso precise buscar sobre como fazer algo usando a tecnologia.

Arquiteturas utilizadas:

- **Mensageria**
- **Rest**

Motivos de usar essas arquiteturas:

- **Mensageria:** está sendo utilizada para processar tarefas pesadas de forma assíncrona.
- **Rest:** está sendo utilizada para criar o webservice, isso permite que caso apareça outras aplicações possam interagir de forma simples, não será necessário implementar a lógica para fazer o que a api já faz para cada cliente.

Fluxo de funcionamento da aplicação:

- É feita uma requisição para gerar um relatório.
- Os dados são enviados para uma fila no RabbitMQ
- Uma aplicação consome os dados, gera o relatório e envia para o email da pessoa.

Observações: devido a alguns problemas essa semana fiquei sem tempo, então irei entregar, mas alguns pontos que eu faria se tivesse tempo: testes unitários e adicionar logs na aplicação para ajudar em um cenário que precisa debugar um erro.