Desafio para time de PLD

Contratos inteligentes ETH

Introdução

Imagine que você precise vender uma casa. Este é um processo complicado e assustador, que envolve muita documentação, comunicação com diferentes empresas e pessoas, além de altos níveis de riscos. É por isso que a maioria absoluta dos vendedores de residências decide procurar um corretor imobiliário, que cuida de toda a papelada, negocia a propriedade e age como um intermediário quando as negociações começam, supervisionando o negócio até que ele seja fechado.

Além disso, a imobiliária fornece um serviço de caução, que é especialmente útil em tais transações, pois as somas envolvidas normalmente são muito grandes e você não pode realmente confiar totalmente na pessoa com quem estará lidando. Entretanto, após o sucesso do acordo, os agentes do vendedor e do comprador compartilharão cerca de sete por cento do preço de venda como comissão. Isso equivale a uma perda financeira substancial para o vendedor.

É numa situação como essa em que os contratos inteligentes podem realmente ser viáveis e efetivamente revolucionar uma indústria inteira, tornando todo o processo muito menos oneroso. Talvez o mais importante, eles resolveriam um problema de confiança. Os contratos inteligentes funcionam com um princípio "se-então", o que significa que a propriedade da casa só será repassada ao comprador quando a quantia de dinheiro acordada for enviada ao sistema.

Eles também funcionam como serviços de custódia, o que significa que tanto o dinheiro quanto o direito de propriedade serão armazenados no sistema e distribuídos para as partes participantes exatamente ao mesmo tempo. Além disso, a transação é testemunhada e verificada por centenas de pessoas, portanto a entrega sem falhas é garantida. Como a confiança entre as partes não é mais um problema, não há necessidade de um intermediário. Todas as funções que um agente imobiliário faz podem ser pré-programadas em um contrato inteligente, ao mesmo tempo em que economizam ao vendedor e ao comprador quantias consideráveis de dinheiro.

Fonte:
https://cointelegraph.com.br/ethereum-for-beginners/what-are-smart-contracts-a-beginners-guide-to-automated-agreements

O desafio

Estamos interessados em utilizar os dados dos contratos inteligentes que estão sendo criados em uma aplicação fictícia. Para isso, precisamos migrar os dados de todos os contratos existentes que foram criados para um banco **Postgres** que será utilizado por esta aplicação. Estes dados precisam ser migrados em uma periodicidade **D-1**, ou seja, precisamos migrar, no dia de hoje, todos os contratos que foram criados no dia de ontem.

Os dados de contratos inteligentes estão disponíveis em um dataset **público e gratuito** da Google. Este dataset fica dentro do **BigQuery**, banco de dados voltado para analytics, no seguinte endereço:

- **Projeto**: “bigquery-public-data”
- **Dataset**: “crypto_ethereum”
- **Tabela**: “tokens”

Para acessar estes dados você precisará criar uma conta gratuita na **Google Cloud Platform** (GCP) onde é disponibilizado 1TB de consultas gratuitamente todo mês. Como os dados de contratos inteligentes tem em torno de 30MB isso deveria ser o suficiente.

O time de dados decidiu seguir com a utilização do **Airflow** como solução para controlar o processo de migração destes dados. Para garantir que não terão problemas com escalabilidade e disponibilidade, decidiram também que será utilizado o Kubernetes como gerenciador de containers.

O que você precisa fazer?

- Subir um Airflow versão 2.6 ou superior;
    - Baixe e execute o projeto mwaa-local-runner seguindo as instruções.
- Criar uma DAG que lê os dados da tabela tokens(BQ), e cria uma tabela tokens no Postgres:
    - Além dos campos existentes, criar um campo adicional chamado **block (varchar)**, concatenando as colunas block_timestamp, block_hash e block_number. Use pandas ou outro package análogo.
    - A ingestão é incremental e baseada na coluna **block_timestamp, inserindo apenas os dados do dia anterior à execução da DAG.**
- Subir um banco **Postgres** local em um cluster kubernetes para receber os dados;
- Configurar a DAG para executar diariamente;
    - Executar às 08:00 UTC, com data inicial 01/09/2023;
    - Cada task precisa estar configurada para 3 retentativas;
    - Para as Tasks, utilize preferencialmente o KubernetesPodOperator;
    - Pode ser usada qualquer imagem docker com Python 3.10;
- Versionar todo o código das DAGs e do Airflow em um repositório **git**;
- Executar a **DAG** manualmente e verificar os dados chegando no banco **Postgres**;

Bônus e observações:

- Evite gravar os dados diretamente no postgresql com inserts. Gere os dados em arquivo e use copy.
- Deixar definido no README de que forma o Airflow e o banco podem ser executados, para que possamos executar a DAG localmente. Descreva de forma objetiva os passos necessários no desenvolvimento da solução.

Critérios de conclusão do desafio

- Para que possamos considerar o desafio entregue, precisamos que você disponibilize o código em algum serviço de git (ex.: Github, Gitlab, Bitbucket, etc) público ou privado (dando acesso aos revisores). Não serão considerados commits após a data de entrega do desafio;
- Demonstrar, via print screen do log da interface web do Airflow, que a DAG está executando corretamente e migrando dados;
- Demonstrar, via print screen de uma consulta no banco, que temos dados migrados no Postgres