The Graph RBBID

1. instalando o Graph CLI

Caso o yarn não esteja instalado na máquina, abra um terminal e execute o comando:

npm install --global yarn

Para o uso do the Graph é preciso ter o Graph CLI instalado na máquina, caso o Graph CLI não esteja instalado na máquina, execute a instalação em um terminal através do comando:

yarn global add @graphprotocol/graph-clim

Importante: Para este tutorial, consideramos que os comandos acima não foram usados dentro de um contêiner. Caso isso tenha ocorrido, é necessário ler o apêndice após o passo 3

2. Inicie um nó de gráfico local

Para iniciar um nó do the graph, é preciso baixar o projeto pré-configurado, para isso, escolha um diretório pelo terminal e execute o comando:

git clone https://github.com/graphprotocol/graph-node/

Após clonar o projeto, acesse a pasta /graph-node/docker. Lá você vai encontrar o docker-compose.yml, dentro desse arquivo é preciso mudar a linha ethereum: 'mainnet:http://host.docker.internal:8545'.

Essa linha, especifica a rede blockchain ('http://host.docker.internal:8545') que o the graph precisa acessar e o nome da rede (mainnet).

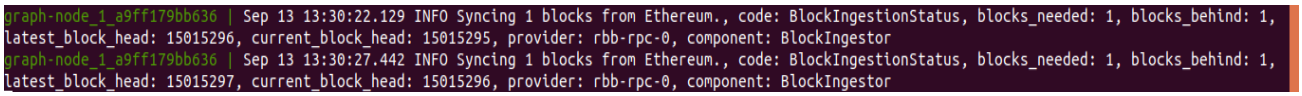
Obs: o nome da rede é escolhido de forma arbitrária, para o nosso exemplo, vamos chamá-la de *rbb*. Já a rede, é o Ip/porta, de um writer node rbb.

Essa linha deve ficar da seguinte maneira:

ethereum: 'rbb:http://host.docker.internal:8545'

Dado que tudo acima já esteja configurado, acesse a pasta /graph-node/docker pelo terminal e levante os containers executando o comando:

docker-compose up



O docker-compose up criará 3 containers, são eles:

1. graphprotocol/graph-node
2. postgres
3. ipfs/go-ipfs

Você pode verificar a criação dos containers com o comando docker ps.

Obs: caso o graph-node não começe a ler os blocos após o docker-compose up. Apague esses containers, remova a pasta data em /graph-node/docker e volte a rodar o docker-compose up.

3. Criando um projeto usando Graph cli

Abra um **outro terminal**, escolha um diretório e execute o seguinte comando:

graph init

Rodando o graph init no terminal, serão feitas uma série de perguntas.

O tipo do projeto: selecione hosted-service

O nome do subgraph: devia-se colocar o nome de um github aqui, mas para o exemplo em questão isso não será necessário, basta colocar *rbb/id*

Nome do diretório que será criado: pode ser escolhido qualquer nome

Ethereum network: pode ser escolhido qualquer rede, pois esse valor será alterado mais à frente

Contract address: Compile o arquivo flat.sol de caminho .../rbb-identificacao/Back-Blockchain/contracts/flat.sol e informe o endereço obtido.

Caminho para o seu ABI: informe o seguinte caminho a partir do diretório que contém o RBBID .../rbb-identificacao/Back-Blockchain/build/contracts/RBBRegistry.json

Contract Name: Contract

Após o passo anterior, um projeto pré-configurado será criado. Substitua os arquivos do projeto pelos seguintes arquivos de mesmo nome do repositório do RBBID através do caminho /theGraph/id

theGraph/id/schema.graphql
theGraph/id/subgraph.yaml
theGraph/id/src/mapping.ts

4. Configurando Graph cli

Para finalizar, acesse o arquivo subgraph.yaml (que se encontra no diretório do projeto criado), adicione em network o nome colocado no docker-compose.yml no passo 2, essa linha deve ficar da seguinte maneira:

network: rbb

5. Executar o sistema

Pelo terminal, entre no diretório do projeto criado e execute os seguintes comandos:

yarn codegen (instala dependências e gera tipos para as ABIs)

yarn create-local (faz o subgraph funcionar localmente alocando o nome do subgraph no graph node)

yarn deploy-local (faz o deploy do subgraph no graph node local) será necessário informar o valor do atributo specVersion do arquivo subgraph.yaml

Ao final da execução do comando yarn deploy-local, haverá um campo com o atributo Queries (HTTP) e seu valor como no exemplo abaixo:

Subgraph endpoints:
Queries (HTTP): http://localhost:8000/subgraphs/name/rbb/id
Subscriptions (WS): http://localhost:8001/subgraphs/name/rbb/id

Copie o valor deste atributo queries (HTTP), acesse esse link pelo browser e comece a fazer as consultas.

Consulta de exemplo:

```
{
	accounts(first:100) {
		id
		addr
		RBBId
		CNPJ
		responsible
		reason
		hashProof
		dateTimeExpiration
	}
}
```

Apêndice

Esse passo só é necessário caso tenho feito o passo 1 dentro de um conteiner

Agora é preciso acessar a pasta id, onde pode ser encontrado o arquivo package.json. Nele vamos editar as seguintes linhas.

"create-local": "graph create --node http://localhost:8020/ rbb/id", "remove-local": "graph remove --node http://localhost:8020/ rbb/id",
"deploy-local": "graph deploy --node http://localhost:8020/ --ipfs http://localhost:5001 rbb/id"

Mude o localhost:8020 para o ip do graph-node e o localhost:5001 para o ip do ipfs.

Você pode achar o ip do ipfs e do graph-node usando o docker inspect “contêiner id”

OBS: o contêiner id pode ser encontrado usando o docker ps

No final da saída, você encontra o ip em “IPAddress”

Referencias

https://medium.com/intech-conseil-expertise/create-your-graph-node-to-query-complex-data-from-blockchain-via-graphql-6f08fbd494c5

https://thegraph.com/docs/developer/quick-start