Hello CanDB in 15 minutes!

What we're setting up

A simple multi-canister greeting application using the Web3 CanDB framework.

The application will create users, storing both their username and a display name (attribute) in different actor partitions. These partitions are based on the group they've selected to join.

(Yes, using multi-canister scalable data storage for a simple greeting example is overkill, but hopefully it gets the point across 🙂).

Dependencies (before beginning)

- NodeJS & npm - https://nodejs.org/ (v18.x recommended)
- Vessel (Motoko's package manager) - https://github.com/dfinity/vessel#getting-started
    - Note: if unable to install CanDB packages via vessel, make sure you have your ssh-key configured through GitHub and added to your ssh-agent
- **Important:** Before beginning, make sure that you've accepted GitHub access to the CanDB and candb-client-typescript repositories

Let's follow the steps below to get set up:

1) Clone hello-candb

At this point let's clone the hello-candb repository sample project. This will contain a Motoko backend, which is located in the src/ directory, and the CanDB frontend, which is located in the frontend/ directory.

```
git clone git@github.com:canscale/hello-candb.git
```

Enter the project you've just cloned

```
cd hello-candb
```

2) Deploy the backend

a) Start DFX

> **Note**: CanDB requires Motoko compiler versions >= 0.7.6. An easy way to pull this down is with a `dfx upgrade`

First, stop any local background dfx server you may have running (`dfx stop` or kill the process running on port 8000), and then from the same directory but a fresh terminal let's start a fresh dfx server

```
dfx start --clean
```

b) Deploy the Index Canister

To deploy the index canister, run

```
dfx deploy index
```

The Index Canister is responsible for spinning up new actor canister partitions.

c) Create a shell HelloService canister

```
dfx canister create helloservice
```

This is required in order to generate declarations for our helloservice canister to be used by the frontend in step 3a)

Next, we'll get the frontend piece set up so that from it we can trigger the Index Canister to spin up and interact with our HelloService actor canister partitions.

3) Start up the frontend

a) Copy backend canister declarations

In order for the frontend to talk to the backend, we need to copy over the backend canister candid declaration files. From the root hello-candb directory run

```
npm run refresh-declarations
```

b) Install dependencies

Let's navigate to the frontend directory and install its code dependencies

```
cd frontend; npm install
```

c) Start up your frontend

```
npm run start
```

Now you can navigate to localhost:8080 to interact with the hello world dapp!

You should see the following UI (see image below)

Hello CanDB!

Below is a sample CanDB app in which one can create users in separate "groups".

To accomplish this, a unique **partition key** (PK) is used in order to **partition**, or separate the data associated with each unique "group" name.

Creating a user inserts it into the currently selected group **partition**, and getting a user fetches it from the currently selected group partition.

Selected Group (**Partition**): 8 YEAR GANG

Get a User from the 8 Year Gang group

Set username to greet: GET USER GREETING

Greeting response:

Create a User in 8 Year Gang group

Set username to create:

Set displayName:

CREATE USERNAME

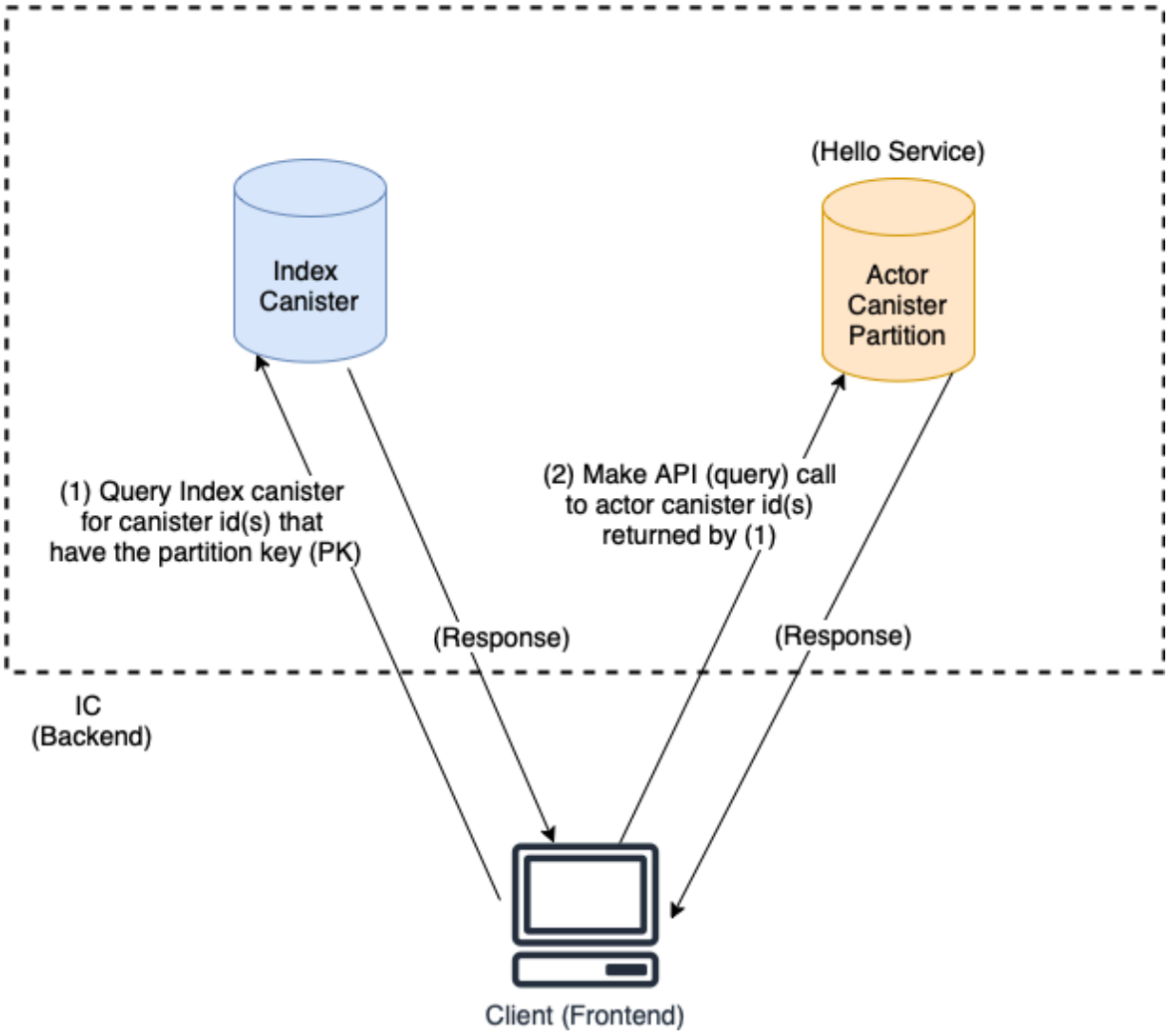
d) Play around with the app

Use the “Create a user” inputs and button to create a user in the 8 year gang group. After creating your first user, navigate to the terminal window holding your dfx server to view the following lines, showing that you’ve dynamically created a canister with a partition key of `group#eightyr`

```
[Canister rrkah-fqaaa-aaaaa-aaaaq-cai] creating new hello service canister
```

```
with pk=group#eightyr
[Canister rrkah-fqaaa-aaaaa-aaaaq-cai] new hello service
canisterId=qoctq-giaaa-aaaaa-aaaea-cai
```

Now with the same group partition key selected, use the “Get a User” input and button to get the user you just created by username. This is what’s happening under the hood



Now try changing the group from 8 year gang to IC Maximalist and try re-fetching the same user. This user will not exist as they were only created in the 8 year gang group, which has a partition key of `group#eightyr`.

Try creating another user now in the IC Maximalist group. Notice in your dfx server log how this first user that is created creates a new canister in the IC Maximalist group, which has a partition key of `group#icmaxi`.

4) Have fun with auto-scaling

The previous example deploys an application with an auto-scaling limit of 200MB worth of data. If you want to play around and test auto-scaling without inserting 200MB worth of data, let's have some fun.

a) Adjust the auto-scaling sizeLimit

From the root of the directory in src/index/IndexCanister.mo, navigate to the createHelloServiceCanister method, which is responsible for spinning up new canisters.

In this method, comment out the following line which scales out a partition based on a heap size of 200MB:

```
sizeLimit = #heapSize(200_000_000); // Scale out at 200MB
```

And then two lines below uncomment this line, which will now scale out a partition every 3 entities that are inserted.

```
sizeLimit = #count(3); // Scale out at 3 entities inserted
```

b) Deploy a fresh index canister

Now let's deploy a fresh version of our index canister to see the difference

```
dfx deploy -m reinstall index
```

When prompted with `Do you want to proceed? yes/No`, type **yes** to continue with the reinstall.

> **Note:** CanDB provides functionality that allows one to upgrade the scaling options of an already deployed application preserving your data (no fresh reinstall required), but that's for another tutorial!

c) Top up your index canister to get it ready for auto-scaling!

This command (for local development only) will top up your index canister with 10T cycles

```
dfx ledger fabricate-cycles --canister index
```

d) Auto-scale with your application

Navigate to localhost:8080 and use “Create a user” to start creating new users into your application. Notice that after the first user, a new canister will be spawn for the `group#eightyr` partition key and be created for every 3 users you create!

Inserting a maximum of 3 users with their display name into a single canister is a huge waste of space for a single canister, but hopefully you can now start to see how seamlessly CanDB auto-scales with your application data.