Requirements

To start locally, clone the repository to your local machine.

This project requires the following to be installed locally on your machine:

- Python virtual environment (3.11 and above)
- Terraform
- Airbyte OSS
- AWS Cloud provider
- Stripe Test account
- Duckdb (installed in the requirements.txt)
- DBT core (installed in the requirements.txt)

Installation & Setting up

Setting up Python

You can download Python here. Activate the Python environment by following the steps below:

```
python3 -m venv .venv
source .venv/bin/activate
pip3 install -r requirements.txt
```

The above will create a Python virtual environment and install the requirements to set up this project.

Setting up Terraform

You can install Terraform by following the link Terraform

Setting up Airbyte

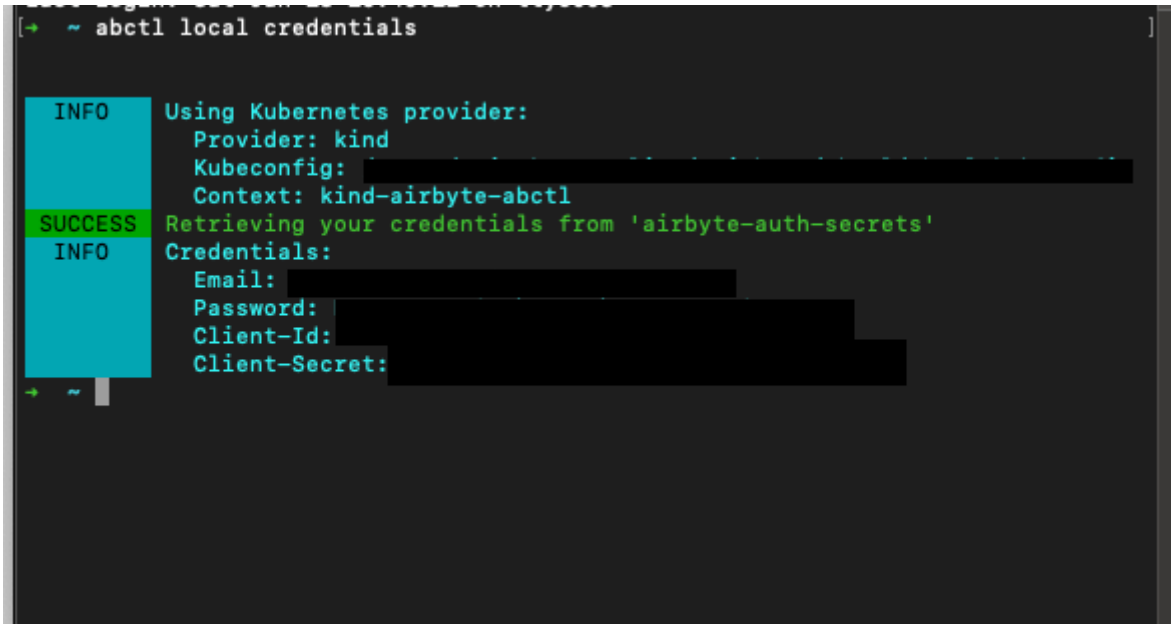
Since this is a proof-of-concept, we are using Airbyte open source installation. You can install it by following this link: Airbyte.

If you run the steps above successfully, you can access your Airbyte local instance at localhost:8000/

You will need to generate credentials to log in to Airbyte instance running locally, and you can do this by running the following:

```
abctl local credentials
```

You should see the following:



You will need to copy the Credentials as they will be useful in the later steps.

Fill in the following credentials in the iac/dev.tfvars file

```
aws_access_key_id =
aws_secret_access_key =
airbyte_client_id = <value of Airbyte Credentials: Client-ID>
airbyte_client_secret = <value of Airbyte Credentials: Client-Secret>
airbyte_server_url = "http://localhost:8000/api/public/v1"
airbyte_workspace_id = <value of Airbyte workspace-ID>
airbyte_user_name = <value of Airbyte Credentials: Email>
airbyte_password = <value of Airbyte Credentials: Password>
stripe_account_id =
stripe_api_key =
```

Setting up AWS

This project assumes you have access to AWS and can generate an `AWS_ACCESS_KEY_ID`, and an `AWS_SECRET_ACCESS_KEY`. You can set up and create these keys by following the link here. Copy them and keep them somewhere safe, as you will need them in the following steps).

For the `iac/dev.tfvars`fill in the `aws_access_key_id` and `aws_secret_access_key` you copied from the above.

`iac/dev.tfvars`

```
aws_access_key_id = <value of AWS_ACCESS_KEY_ID>
aws_secret_access_key = <value of `AWS_SECRET_ACCESS_KEY>
airbyte_client_id = <value of Airbyte Credentials: Client-ID>
airbyte_client_secret = <value of Airbyte Credentials: Client-Secret>
airbyte_server_url = "http://localhost:8000/api/public/v1"
airbyte_workspace_id = <value of Airbyte workspace-ID>
```

```
airbyte_user_name = <value of Airbyte Credentials: Email>
airbyte_password = <value of Airbyte Credentials: Password>
stripe_account_id =
stripe_api_key =
```

There is a `test.env` file that contains the following, and they need to be filled out with the key and secret key you obtained from the IAM page

Kindly leave the `AWS_BUCKET_NAME` blank for now, as it will be auto-generated for you after running provisioning the services with Terraform.

`test.env`

```
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_REGION="eu-west-1"
AWS_BUCKET_NAME=
```

After filling in the `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` and your `AWS_REGION` in the `test.env` **rename it** to `.env` (Yes, still leave the `AWS_BUCKET_NAME` empty for now)

Setting up Stripe

You can sign up with Stripe using the link here.

You'd need a Stripe API key and a Stripe account ID for a test account. Kindly follow this link to generate a Stripe API key docs.stripe.com/keys?locale=en-GB#create-api-secret-key for test mode, and to get your Stripe account ID you can get it from your dashboard.stripe.com/settings/user . You can follow these steps here if you are unsure.

Provisioning resources (Cloud + Airbyte)

There is a file called `dev.tfvars` found in the `iac` directory.

You will need to fill in the values in your `iac/dev.tfvars` as they will be required for provisioning the resources:

Assuming you've been following the steps above, it should look like this:

`iac/dev.tfvars`

```
aws_access_key_id = <value of AWS_ACCESS_KEY_ID>
aws_secret_access_key = <value of `AWS_SECRET_ACCESS_KEY>
airbyte_client_id = <value of Airbyte Credentials: Client-ID>
airbyte_client_secret = <value of Airbyte Credentials: Client-Secret>
airbyte_server_url = "http://localhost:8000/api/public/v1"
airbyte_workspace_id = <value of Airbyte workspace-ID>
```

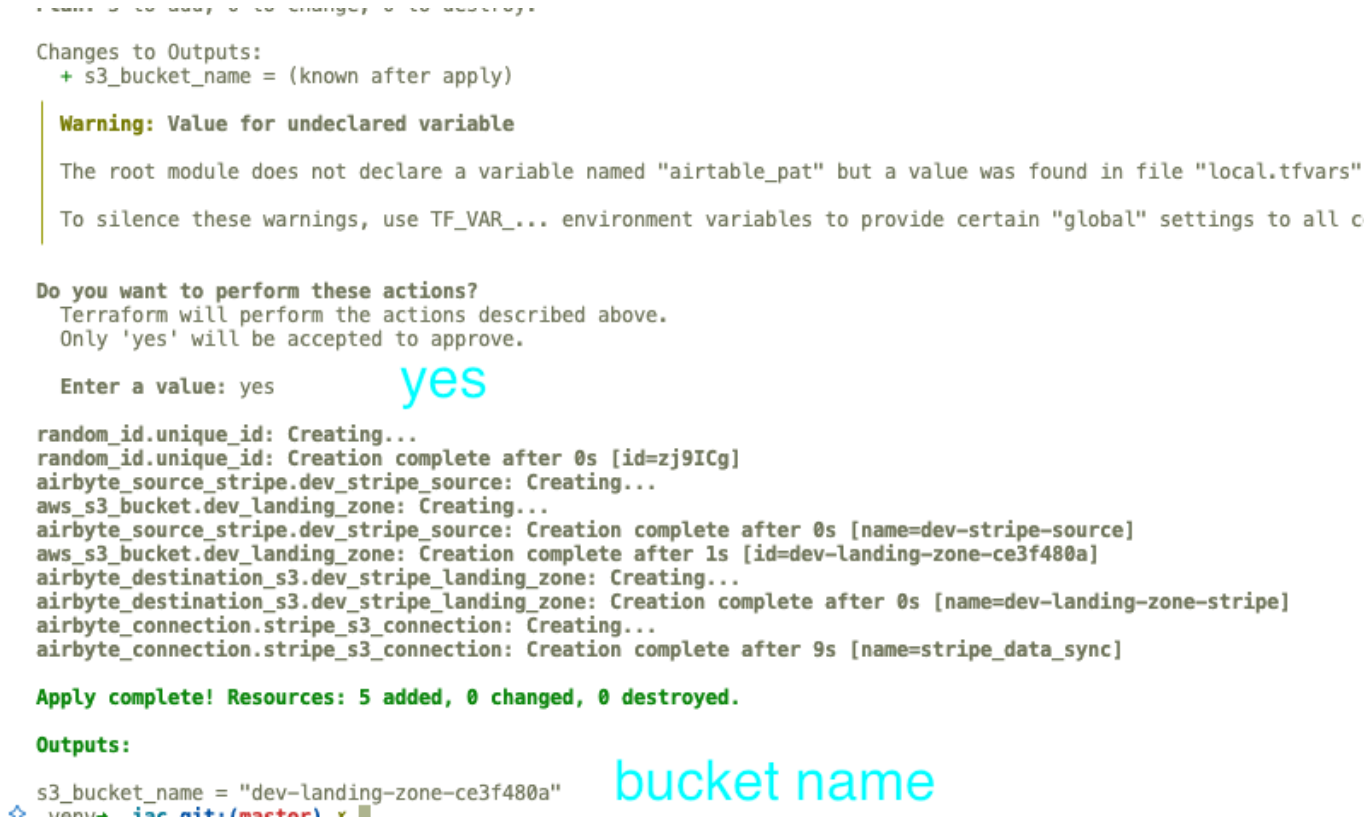
```
airbyte_user_name = <value of Airbyte Credentials: Email>
airbyte_password = <value of Airbyte Credentials: Password>
stripe_account_id = <value of Stripe account ID>
stripe_api_key = <value of Stripe api key>
```

Assuming you have filled in the above, you can start provisioning the resources on AWS; you can run this step in your terminal.

```
cd iac
terraform apply -var-file="dev.tfvars"
```

And type in `yes` when prompted.

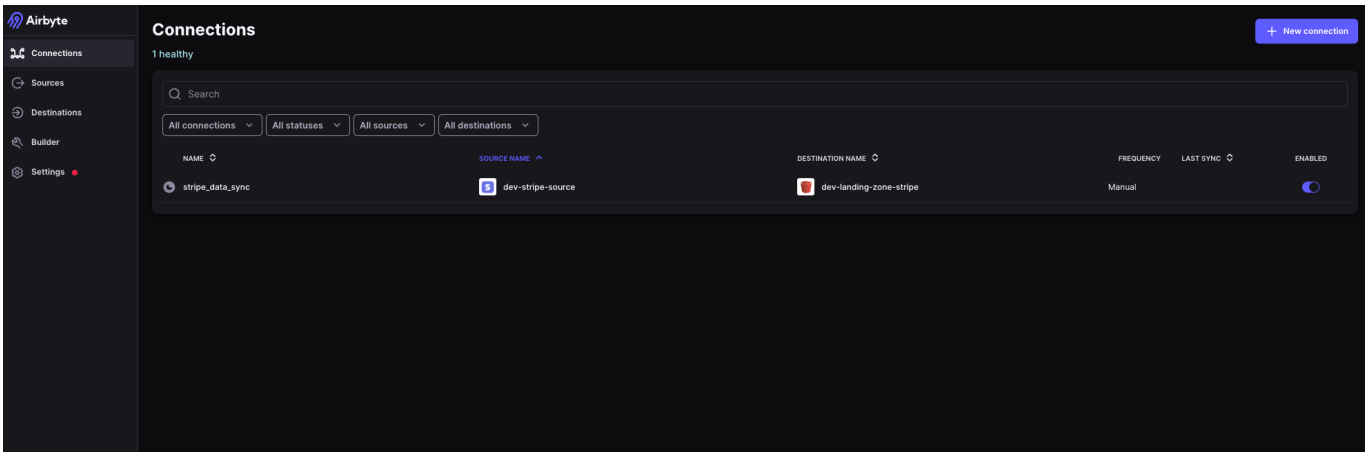
If everything is successful, you should see `s3_bucket_name` an output after successfully creating the resources. Kindly copy the value generated and put that in your `.env` file (the one you renamed) as the value for `AWS_BUCKET_NAME`



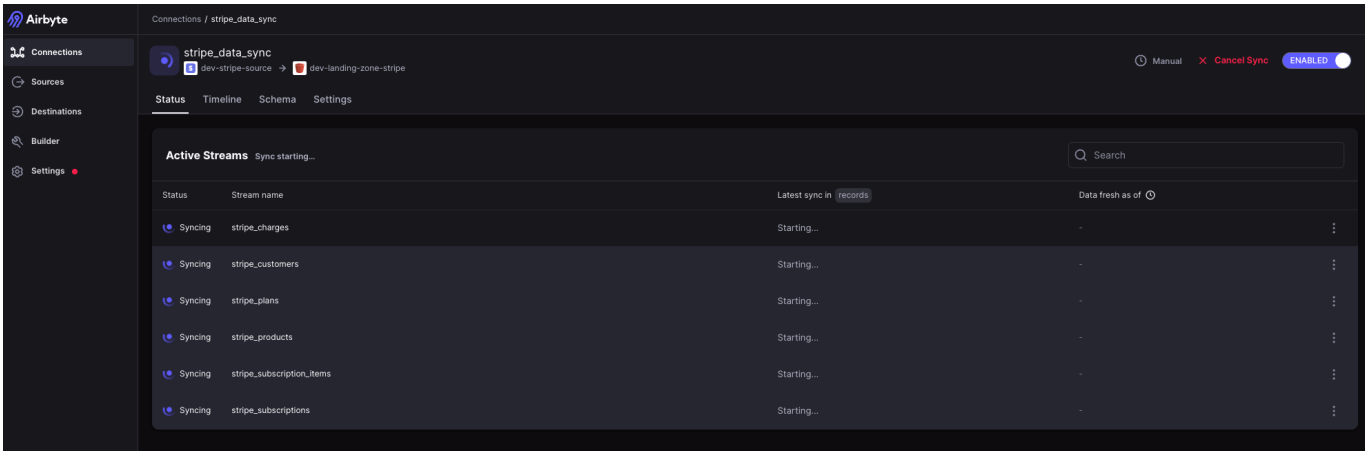
The code above changes the directory into the `iac` folder containing our terraform. Running `terraform apply -var-file="dev.tfvars"` provisions the following:

- An S3 bucket, which we will use as `landing zone` to save the Stripe data that we regularly ingest using Airbyte.
- A Stripe source in Airbyte which represents our connection to the Stripe data source, which contains the data we want to ingest from Stripe.
- An S3 destination in Airbyte which represents in Airbyte which points to our S3 landing zone
- An Aibyte connection which syncs data from the Stripe source to the S3 destination

If successful, in Airbyte, you should see the following:



The next step is to trigger a sync manually by clicking on the connection and selecting `sync now`



DBT

In a new terminal or in the same terminal, from the project's root directory, source the environmental variables defined in your .env by running the following:

```
source ./.env
```

If you forgot to rename the `test.env` to `.env` file, no worries, you can run this instead

```
source ./test.env
```

Next, you will need to build the models using dbt-core by running the following

```
cd data-models
dbt run
```

Issues

- If you notice that Terraform is not working for you, double-check to make sure you are running the command from within the `iac` folder
- I get this error message `Runtime Error in model stg_stripe_customers (models/staging/stg_stripe_customers.sql) HTTP Error: HTTP GET error on '/?encoding-type=url&list-type=2&prefix=stripe%2Fstripe_customers%2FDATE%3D' (HTTP 404) 20:48:00` . This could mean that you have not synced data into your S3 bucket or that the bucket name for your S3 bucket is wrong
-