Dynamic Ecommerce Discounts with Redpanda

Git commit with doc: 71f1bc4
HTML version

1. Introduction

This project is a companion to the Example Next.js Ecommerce Store for Snowplow.

It allows you to test this demo locally, using LocalStack, and in the AWS cloud.

Its Architecture is designed so a developer can quickly and easily set up these two environments and test the project.

2. Architecture

- The **ecommerce-nextjs-example-store** is a Next.js application that generates tracking events.
- The **stream-collector** component sends these events via Kinesis to the **[snowbridge]** component.
- The **snowbridge** component enriches these events, inserts more information (via **[enrich]** component), and sends them to [redpanda].
  - Read more about the **enrich** component here: https://docs.snowplow.io/docs/pipeline-components-and-applications/enrichment-components/enrich-kinesis/.
  - Read more about the **snowbridge** component here: https://docs.snowplow.io/docs/destinations/forwarding-events/snowbridge/.
- The **redpanda** broker is receives the events from use (aka Bethos) to apply the dynamic discounts.
- The **redpanda-connector** ...

Sequence Diagram for the Architecture:

TODO

All components in this Architecture run as Docker containers via `docker compose`:

- The Snowplow's components (**[stream-collector]**, **[enrich]**, and **[snowbridge]**) are defined in the file `compose.snowplow.yaml`.
- Redpanda's infrastructure is provided by the file `compose.redpanda.yaml`.
- The apps components (**[ecommerce-nextjs-example-store]**) are defined in the file `compose.apps.yaml`.
- The infrastructure to provide the AWS resources locally (Kinesis, DyanmoDB, etc) is created by LocalStack.
  - Read the file `compose.localstack.yaml`.
- These components and resources are created in AWS using Terraform scripts.
  - There is another document, in `docs/terraform` folder, explaining the details.

3. Prerequisites

1. Start a Ubuntu Linux (it can be running on a WSL2 environment) terminal.
2. Make sure you have docker (and docker compose) installed.
3. Clone this project with Git and cd to it.
4. Create a file `docker/.env` (from `docker/.env.sample`) and configure the AWS variables on it.

You don't need Java or Node.js configured on your machine to follow the steps below. **You only need a Bash terminal and a Docker installation.**

4. Steps (to run this application as is)

Step 1 → Start the containers

```
$ ./docker/up.sh
```

Tips:

1. You can press `Ctrl` + `C` at any time. The docker containers will remain running.
2. If there is no file `docker/.env` in the project, this script will try to locate it in a file named `../dynamic-ecommerce-discounts-with-redpanda.env` and copy it to `docker/.env`. This allows you to call `git clean -fdX` at any time you want without losing your configuration.
   a. If the file `../dynamic-ecommerce-discounts-with-redpanda.env` does not exists, it will copy the file `docker/.env.sample` to `docker/.env` and use it.
3. You can pass "services" as an argument option to this script. It will list the options you can pass to it by adding the suffix "-services":

```
$ ./docker/up.sh services
apps
localstack
redpanda
snowplow
```

4. By adding the "-services" to one of the options listed above, you will start only the services listed in the file `copose.<service>.yaml`. So, this will start only the redpanda services (services listed in `compose.redpanda.yaml`):

```
$ ./docker/up.sh redpanda-services
```

5. You can also call the script `up.sh` by using the `compose.sh` script this way:

```
$ ./docker/compose.sh up
```

Step 2 → Know the URL provided by the services

1. **LocalStack**: https://app.localstack.cloud ← localstack
2. **Redpanda**:
   a. **Internal (docker containers access)** http://localhost:9092 ← redpanda-internal
   b. **Console**: http://localhost:8080 ← redpanda-console

    i. User / password: jane / some-other-secret-password

3. **Ecommerce store**: http://localhost:3000 ← ecommerce-store
    a. It connects with **Snowplow collector** configured to run in http://localhost:9090 ← snowplow-collector

Step 3 → Browse the application pages

As expected, in the [ecommerce-store], during every page navigation, we are tracking a page view event.

For ecommerce interactions we track the following:

- When a customer goes to a product page we track a product view event.
- When a customer sees an internal promotion list, e.g. Homepage promotions, we track an internal promotion view event.
- When a customer clicks an internal promotion, we track an internal promotion click event.
- When a customer goes to a product list page, we track a product list view event.
- When a customer clicks a product on a product list page, we track a product list click event.
- When a customer sees a recommended product list on the product page, we track a product list view event.
- When a customer clicks on a recommended product list on the product page, we track a product list click event.
- When the customer adds a product to the cart, we track an add to cart event.
- When the customer goes to the cart page we track a checkout step event.
- When they go to the payment step, another checkout step event is tracked.
- When the customer successfully completes a transaction, we track a transaction event (triggered on the server-side but formulated with the spec of Snowplow ecommerce)

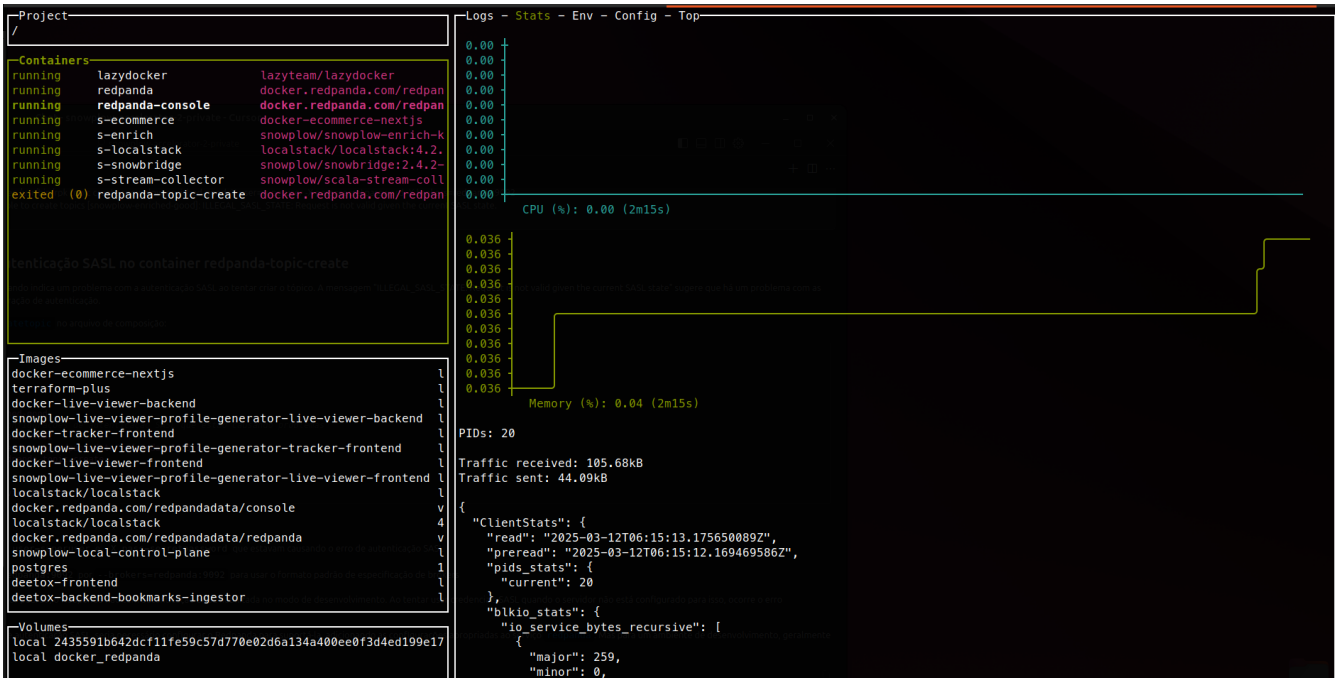
Step 4 → Access the redpanda-console and check the generated events

After browsing the , you can access the and check the generated events in the topic `snowplow-enriched-good`. See these images: redpanda-1.png, redpanda-2.png.

You can explore the data format of these events in enriched TSV format. In the scripts/raw-messages.sample directory, there are examples of the events recorded by Snowplow when they are transferred to Redpanda. These sample TSV files were created by running the script extract-snowplow-raw-messages.sh. Note: *these are not the events final format of the events sent from [snowbridge] component to [redpanda].*

Step N → (optional) Use LazyDocker to monitor the containers and logs

```
$ ./docker/compose.sh lazy
```



5. Clean up steps

Step 1 → Stop the containers

To stop all the containers, type:

```
$ ./docker/down.sh
```

Step 2 → Clean up

To remove all the containers and images, type:

```
$ ./docker/clean.sh
```

Warnings:

1. The script `clean.sh` will destroy any data generated by these containers.

6. References

LocalStack

Redpanda

- Docker Compose Labs
  - Start a Single Redpanda Broker with Redpanda Console in Docker
- Redpanda Self-Managed Quickstart
- How we engineered our CLI to improve developer productivity
- Some YouTube videos:
  - Why did Redpanda rewrite Apache Kafka? (with Christina Lin)
  - Redpanda Office Hour: HUGE rpk - Redpanda CLI update!

Redpanda Connect

- https://docs.redpanda.com/redpanda-connect/get-started/quickstarts/rpk/
- https://docs.redpanda.com/current/get-started/quick-start/

7. Demo videos