# Divolte Workshop: Exercise 1

In this exercise we're going to start up our toy application and generate some clicks. We'll then have a quick look at them.

## Step 1: Start the Stack

The stack uses *Docker Compose* to run a bunch of services locally. If you're on a Mac or using Linux we have some convenience scripts.

To build and start the stack:

```
% ./refresh
```

The first time you start everything up it might be quite slow, but subsequent updates tend to be quicker.

When it's finished you should be able to browse to http://localhost:9011/ and see our application. But we're not quite ready yet...

## Step 2: Initialize the Catalogue

Initially our shop doesn't have any catalogue data. To load the data:

```
% ./load-data
```

While the catalogue is being loaded a bunch of stuff is printed, but it should end with something like:

```
Summary:
        2689 times status code 200
        19 times status code 400
```

Browsing the shop should now have content.

## Step 3: Add the Divolte Tag

At the moment our web shop looks nice but doesn't include the Divolte tag. To add it in:

1. Open up: `webapp/templates/base.html`
   This is the base template that is used to render every page on the site.

2. Around line 20 is a disabled `<script src=…></script>` block for Divolte. Change this so it reads:

```
<!-- Divolte Collector -->
<script src="{{ config.DIVOLTE_URL }}"></script>
```

In this case we have a configuration variable ( `DIVOLTE_URL` ) that contains the location of Divolte Collector.

1. Restart the web application:

   ```
   % ./refresh
   ```

2. In your browser, reload the shop. Check that `divolte.js` is being loaded.

The base tag is now included in the web application. Try doing things like:

- Adding things to your basket;
- Previewing images;
- Completing the purchase. (You might not want to have too many items in your basket!)

## Step 4: Examine the Clickstream

Now that the tag is present and you've been clicking around the site, there should be some clickstream data to examine.

1. The Divolte container is saving its data to a volume mounted on `/data`. Every 30 seconds a new file appears there, so long as some clicks have been received. To see the Avro files:

   ```
   % docker-compose run divolte ls -l /data
   ```

2. If you wish, you can see the content of the events in the clickstream using a utility script in the container.

   ```
   % docker-compose run divolte show-avro
   ```

For those familiar with Avro: this is just a wrapper around the `avro-tools tojson` command.

## Things to Think About

- What are the various fields in the events?
- Does navigating back to a previous page cause a new event?
- What is the value of the `firstInSession` field?
- What happens if you clear your cookies?