6. Analytics

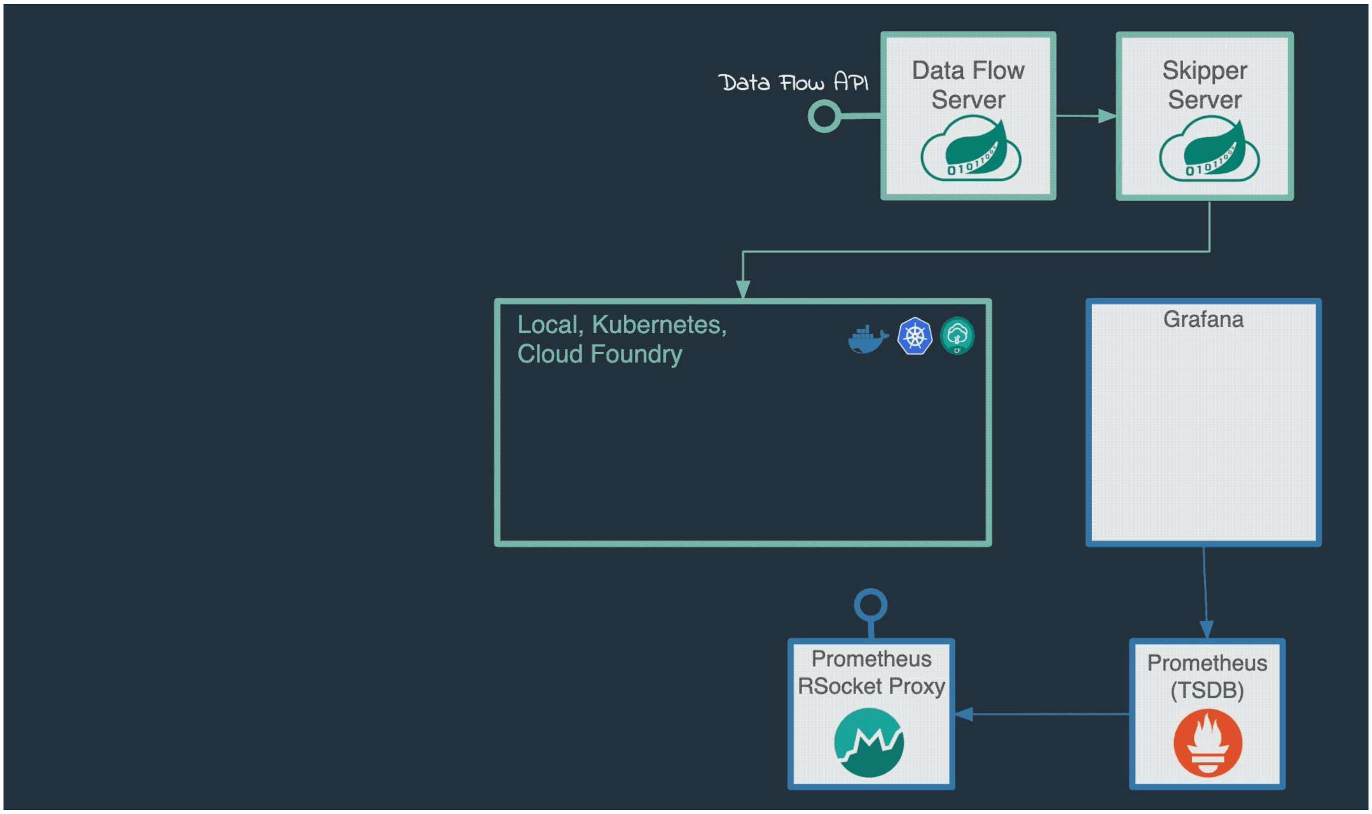
Prev Next

6. Analytics

6.1 Twitter Analytics

In this demonstration, you will learn how to build a data pipeline using Spring Cloud Data Flow to consume data from *TwitterStream*, compute analytics over data-in-transit using Analytics-Counter. Use Prometheus for storing and data aggregation analysis and Grafana for visualizing the computed data.

We will take you through the steps to configure Spring Cloud Data Flow's `Local` server.



6.1.1 Prerequisites

- A running Local Data Flow Server with enabled Prometheus and Grafana monitoring.
  On Linux/Mac, installation instructions would look like this:

  ```
  $ wget https://raw.githubusercontent.com/spring-cloud/spring-cloud-dataflow/v2
  $ wget https://raw.githubusercontent.com/spring-cloud/spring-cloud-dataflow/v2

  $ export STREAM_APPS_URI=https://dataflow.spring.io/kafka-maven-einstein

  $ docker-compose -f ./docker-compose.yml -f ./docker-compose-prometheus.yml up
  ```

  > This sample requires the 2.x (e.g. Einstein) pre-build applications! Depending on the platform (local, k8s or CF) and the binder (RabbitMQ or Kafka) one can install (or set via the `STREAM_APPS_URI` variable for local installations) apps from the following pre-build lists: (1) Kafka: `dataflow.spring.io/kafka-docker-einstein`, `dataflow.spring.io/kafka-maven-einstein`, (2) RabbitMQ: `dataflow.spring.io/rabbitmq-docker-einstein`, `dataflow.spring.io/rabbitmq-maven-einstein`.

- A running Data Flow Shell

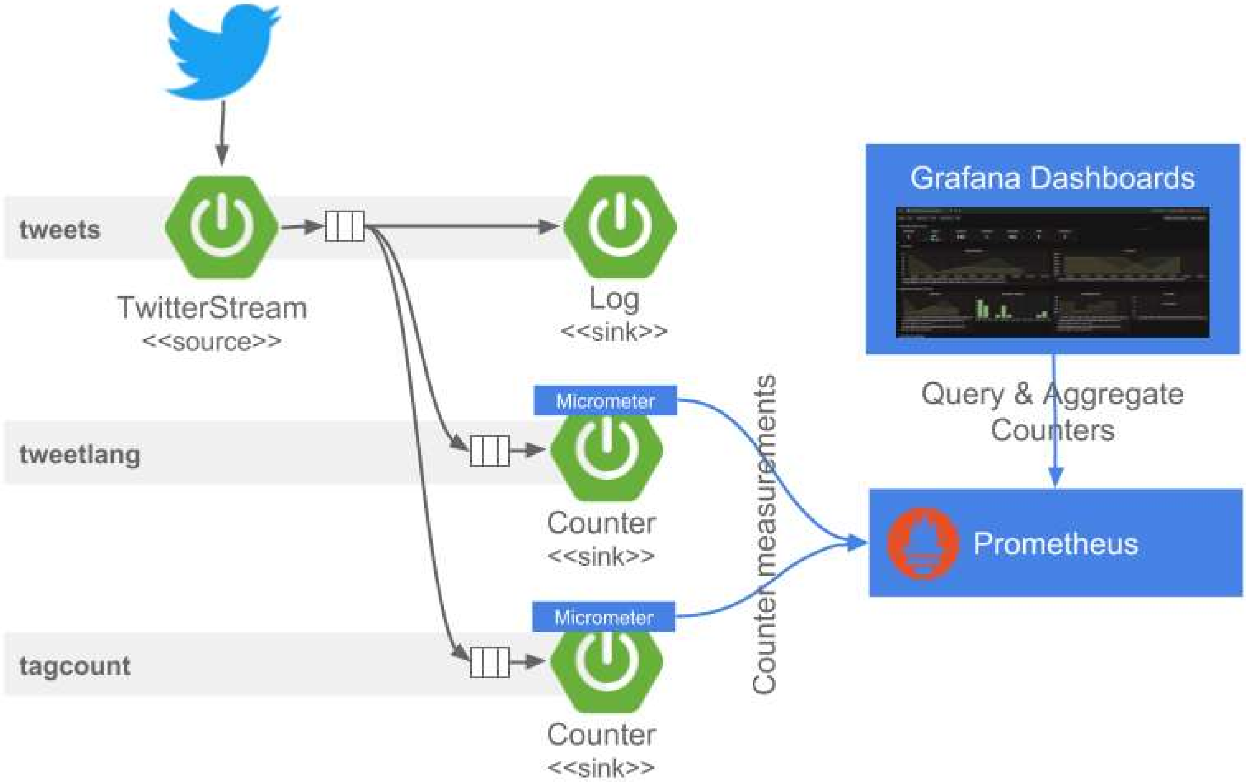
  ```
  $ wget https://repo.spring.io/release/org/springframework/cloud/spring-cloud-d
  $ java -jar spring-cloud-dataflow-shell-2.8.1.jar

  Welcome to the Spring Cloud Data Flow shell. For assistance hit TAB or type "h
  dataflow:>
  ```

  The Shell connects to the Data Flow Server’s REST API and supports a DSL for stream or task lifecycle managing.
  If you prefer, you can use the Data Flow UI: localhost:9393/dashboard, (or wherever it the server is hosted) to perform equivalent operations.
- Twitter credentials from Twitter Developers site

6.1.2 Building and Running the Demo

1. Create and deploy the following streams



The `tweets` stream subscribes to the provided twitter account, reads the incoming JSON tweets and logs their content to the log.

```
dataflow:>stream create tweets --definition "twitterstream --consumerKey=<CONS
```

To get a consumerKey and consumerSecret you need to register a twitter application. If you don't already have one set up, you can create an app at the Twitter Developers site to get these credentials. The tokens `<CONSUMER_KEY>`, `<CONSUMER_SECRET>`, `<ACCESS_TOKEN>`, and `<ACCESS_TOKEN_SECRET>` are required to be replaced with your account credentials.

The received tweet messages would have a JSON format similar to this:

```
{
  "created_at": "Thu Apr 06 15:24:15 +0000 2017",
  "id_str": "850006245121695744",
  "text": "Today we are sharing our vision for the future of the Twitter API p
  "user": {
    "id": 2244994945,
    "name": "Twitter Dev",
    "screen_name": "TwitterDev",
        "lang": "en"
  },
  "place": {},
  "entities": {
    "hashtags": [
                  {
                    "text": "documentation",
                    "indices": [211, 225]
                  },
                  {
                    "text": "GeoTagged",
                    "indices": [239, 249]
                  }
    ],
    ....
  }
}
```

The JsonPath SpEL expressions can help to extract the attributes to be analysed. For example the `#jsonPath(payload,'$..lang')` expression extracts all values of the `lang` attributes in the tweet. The Analytics Counter Sink maps the extracted values to custom Micrometer tags/dimensions attached to every measurement send. The `tweetlang` stream created below, extracts and counts the languages found in the tweets. The counter, named `language`, applies the `--counter.tag.expression.lang=#jsonPath(payload,'$..lang')` to extract the language values and map them to a Micrometer tag named: `lang`. This counter generates the `language_total` time-series send to Prometheus.

```
dataflow:>stream create tweetlang  --definition ":tweets.twitterstream > count
```

Similarly, we can use the `#jsonPath(payload,'$.entities.hashtags[*].text')` expression to extract and count the hastags in the incoming tweets. The following stream uses the counter-sink to compute real-time counts (named as `hashtags`) and the `htag` attribute in `counter.tag.expression.htag` indicate to Micrometer in what tag to hold the extracted hashtag values from the incoming tweets.

```
dataflow:>stream create tagcount  --definition ":tweets.twitterstream > counte
```

Now we can deploy the `tweets` stream to start tweet analysis.

```
dataflow:>stream deploy tweets
```

2. Verify the streams are successfully deployed. Where: (1) is the primary pipeline; (2) and (3) are tapping the primary pipeline with the DSL syntax `<stream-name>.<label/app name>`

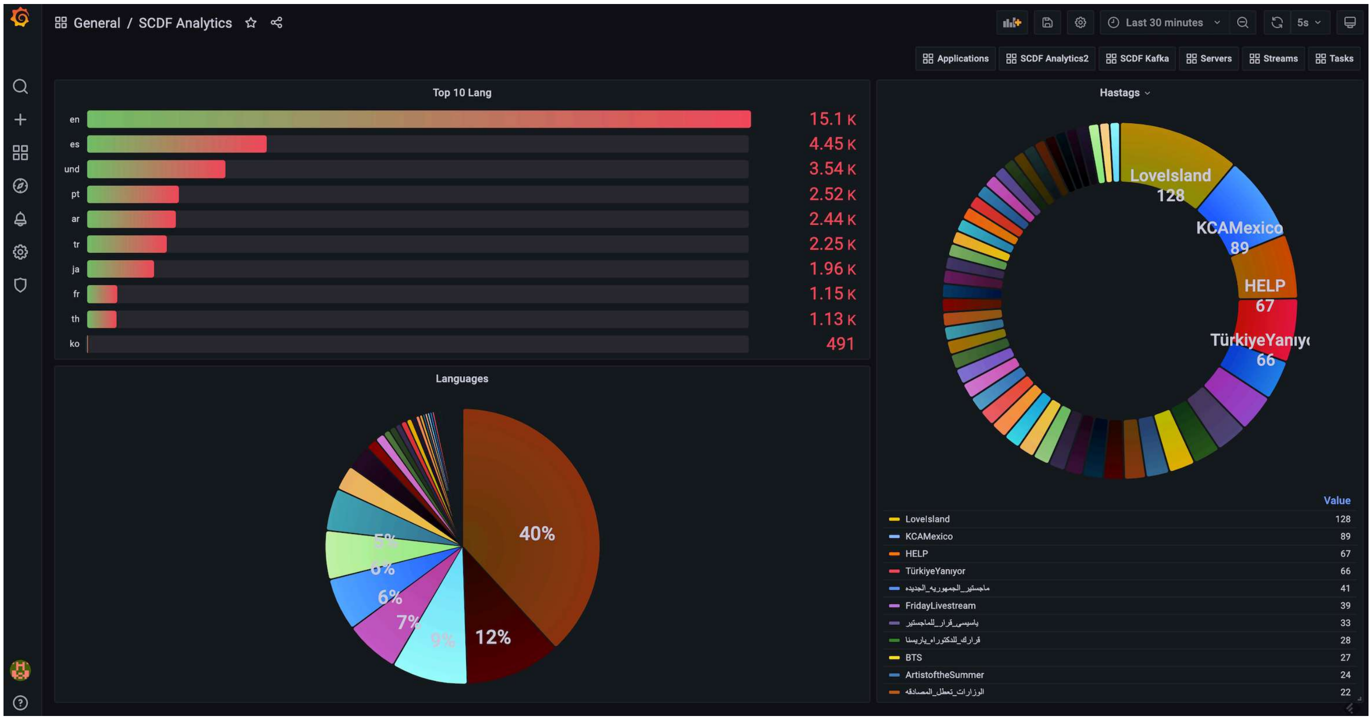
[e.x. `:tweets.twitterstream`]; and (4) is the final deployment of primary pipeline

```
dataflow:>stream list
```

3. Notice that `tweetlang.counter`, `tagcount.counter`, `tweets.log` and `tweets.twitterstream` Spring Cloud Stream applications are running as Spring Boot applications within the `local-server`.
4. Go to `Grafana Dashboard` accessible at `localhost:3000`, login as `admin`:`admin`. Import the grafana-twitter-scdf-analytics.json dashboard.

> you can import it directly using the following dashboard code: `14800`.

You will see a dashboard similar to this:



	Value
LoveIsland	128
KCAMexico	89
HELP	67
TürkiyeYanıyor	66
ماجستير_الجمهوريه_الجديده	41
FridayLivestream	39
ياسيسى_قرار_للماجستير	33
قرارك_للدكتوراه_ياريسنا	28
BTS	27
ArtistoftheSummer	24
الوزارات_تعطل_المصادقه	22

The following Prometheus queries have been used to aggregate the `lang` and `htag` data persisted in Prometheus, which can be visualized through Grafana dashboard:

```
sort_desc(topk(10, sum(language_total) by (lang)))
sort_desc(topk(100, sum(hashtags_total) by (htag)))
```

6.1.3 Summary

In this sample, you have learned:

- How to use Spring Cloud Data Flow’s `Local` server
- How to use Spring Cloud Data Flow’s `shell` application
- How to use Prometheus and Grafana with Spring Cloud Data Flow’s `Local` server
- How to create streaming data pipeline to compute simple analytics using `Twitter Stream` and `Analytics Counter` applications

Prev Next

5. Stream Launching Batch Job Home 7. Data Science