



## Goal

Use the Kafka REST Proxy through an Oracle APEX client application.

This Oracle APEX sample App, [available on Github](#), includes a simple KAFKA producer and a KAFKA consumer. It can be used to get more familiar with the produce/consume process and the *commit* features subtleties.

About the sample demo, let's assume a collection of devices spreaded in several cities and the ability given to any operator to insert a manual message in the stream or polling events from a given topic.

Contenu [Afficher](#)

## Prerequisites

Either install an on-premise [Apache KAFKA](#) cluster, or use a docker image or subscribe to [Confluent platform](#).

If on-premise KAFKA installation, one must install, at least, the [community version of Confluent REST Proxy](#) and setup TLS in order to make calls from a free tiers APEX instance. (In case of apex.oracle.com, it's possible to call a http endpoint instead https) cf [Oracle rules](#).

The [APEX application is available on github](#). Export has been made with a version 23.2.

## Installation of Kafka

For a single Broker, on a Linux server, follow the links:

- [Download Apache Kafka](#)

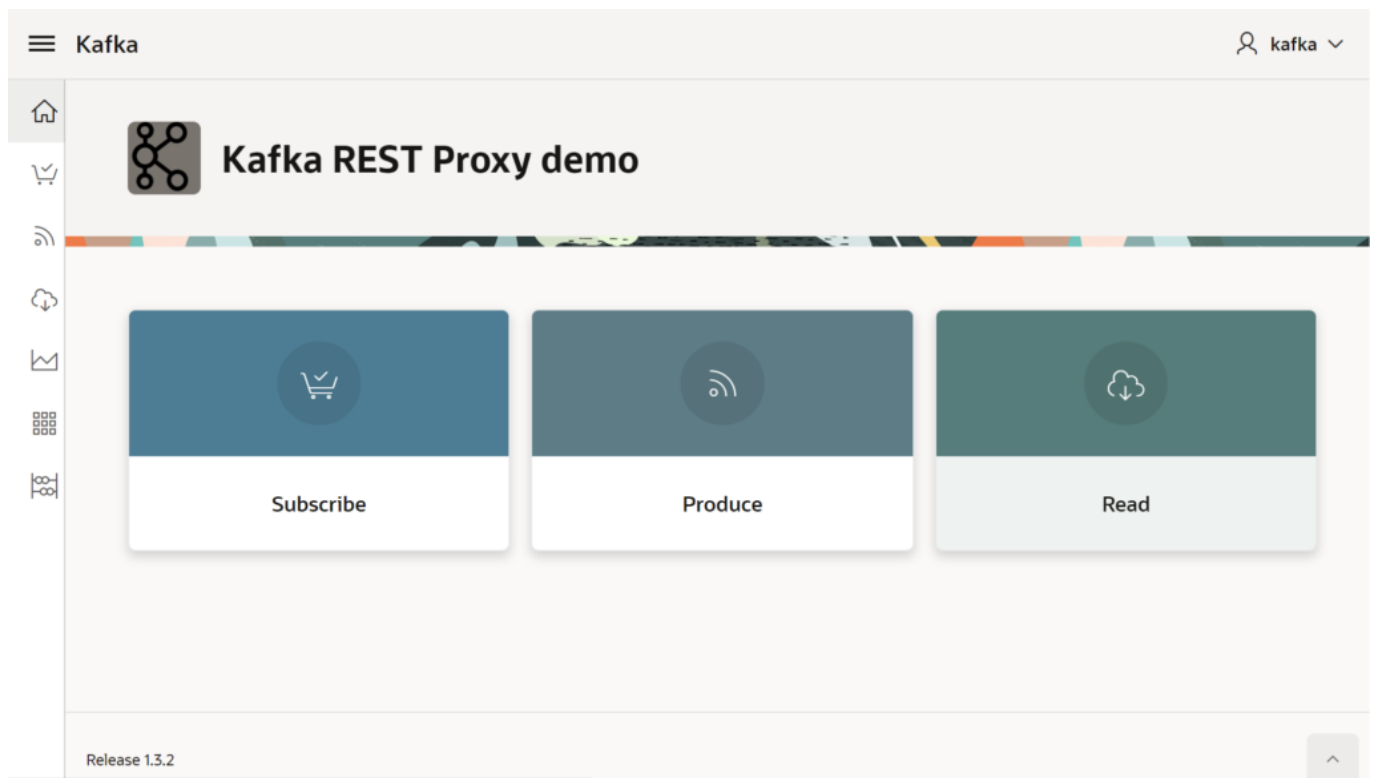


- [Install Kafka](#)
- [Install REST Proxy](#)
- Create start/Stop scripts (cf appendices)

About setup TLS for the REST Proxy, read for instance the very good [post from Ken Coenen](#) and get infos related to openssl and keystore and adapt the file etc/kafka-rest/kafka-rest.properties.

Read [REST Proxy Security](#) and adapt ssl.client.authentication.

## Description of application



A regular APEX app, named Kafka, relies on a package (KAFKA\_PKG) which wraps calls to the REST Proxy. The material is [available from Github](#).

Import it in a Oracle APEX instance  $\geq 23.2$ .



During import process, set the REST Proxy endpoint and accept installation of supporting objects

Launch Kafka app, jump in *setup* option, check the *default consumer name* (ie: patrick) and the *Consumer Group name* (default: *my\_json\_consumer\_group*).

The producer menu option proposes to add only one message or a batch of ten records based on the content of VILLES table. That can be changed in the package KAFKA\_PKG.

## Sending Messages

The screenshot shows the Kafka REST Proxy interface. At the top, there is a header with a hamburger menu icon, the text 'Kafka', and a search icon with the text 'kafka'. Below the header, there is a sidebar with navigation icons. The main content area shows a 'Topic' dropdown menu set to 'demo' and a 'Several Messages' button. Below this, there are six message cards arranged in a 3x2 grid. Each card has a city name, a code, a number, and a 'Send Message' button. The cards are: 1. Paris (PA, 2243833), 2. Marseille (MA, 850726), 3. Lyon (LY, 484344), 4. Toulouse (TO, 441802), 5. Nice (NI, 343304), and 6. Nantes (NA, 284970).

Application offers following features:

- Listing existing topics
- Creating/deleting a consumer instance and subscribing to one topic
- Consuming records from an offset.



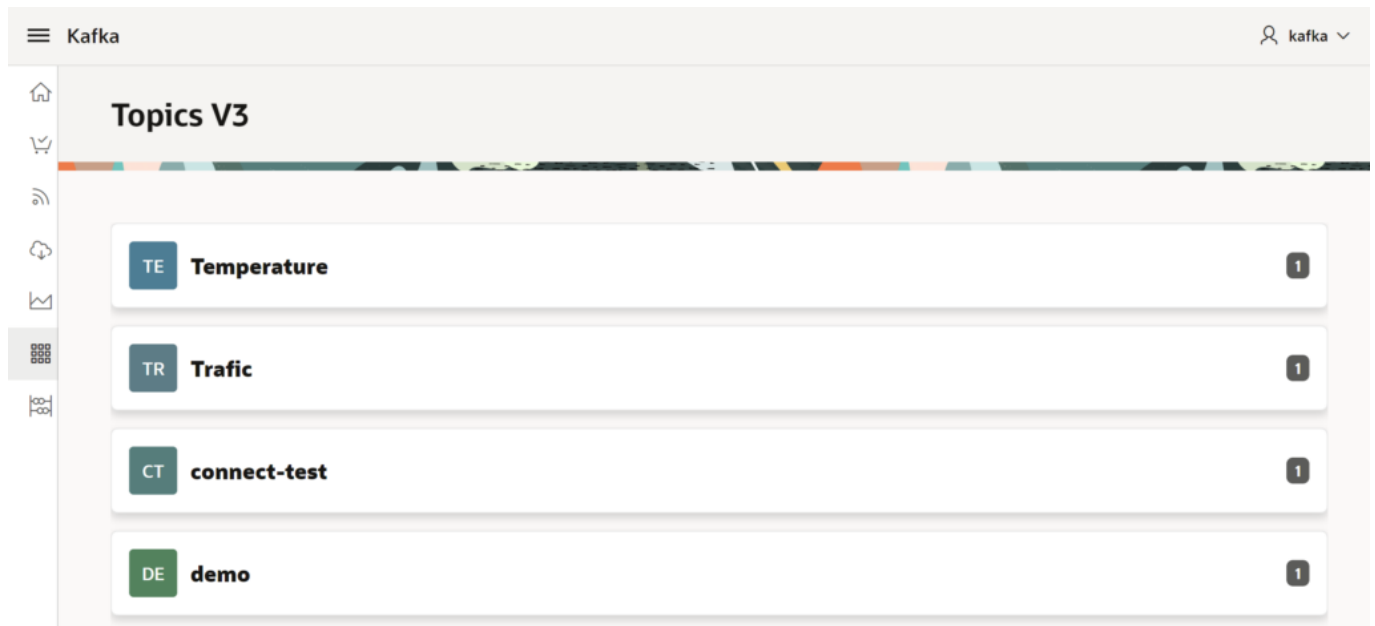
The records page relies on a data source and the other actions are implemented in a dedicated PLSQL package : KAFKA\_PKG. This package is embeded as a supporting object in the APEX application.

The screenshot shows the 'Subscribe' page in the Kafka REST Proxy APEX application. The page has a header 'Kafka' and a user profile 'kafka'. The main content area is titled 'Subscribe' and contains a 'Subscription' form. The form has three input fields: 'Consumer Name', 'Subscribe to' (with a dropdown menu showing 'demo'), and 'Auto offset reset' (with a dropdown menu showing 'earliest'). There is also an 'Auto commit' toggle switch. A 'New Instance' button is in the top right of the form, and a 'Delete Instance' button is in the bottom right. The footer shows 'Release 1.3.2'.

## Notes about the consumer instance

When creating a new consumer instance in a consumer group, the max iddle session is set at the server side around 4 minutes. That means that we have to poll regularly, otherwise, we must re-create a new consumer instance. The sample application doesn't catch this situation, but there is a page which draws a chart on a regular basis and that prevents a too long idle time.

## List of existing Topics



Clicking on a topic entry gives the lags between the last commit point and the last entry.

## Appendices

### Scripts for starting and stopping Apache KAFKA

It's strangely tricky to start Kafka at boot, for obscure reasons of permissions, even as root ...

I didn't want to dig in these details, not important in my context.

So I just mention two scripts to launch manually the needed modules.

Because I used Kafka with Zookeeper, the first one starts *zookeeper*, then a Kafka server in background.

(Another option is to use Kafka with KRaft)

The second script launches REST Proxy. We can choose to let it in foreground or as a daemon.



Copied from

<https://stackoverflow.com/questions/34512287/how-to-automatically-start-kafka-upon-system-startup-in-ubuntu>

These following scripts are available on the Github repository.

- Start/Stop/Status Zookeeper and Kafka
- Start/stop/Status REST Proxy

## Author



[Patrick](#)

GPM Factory