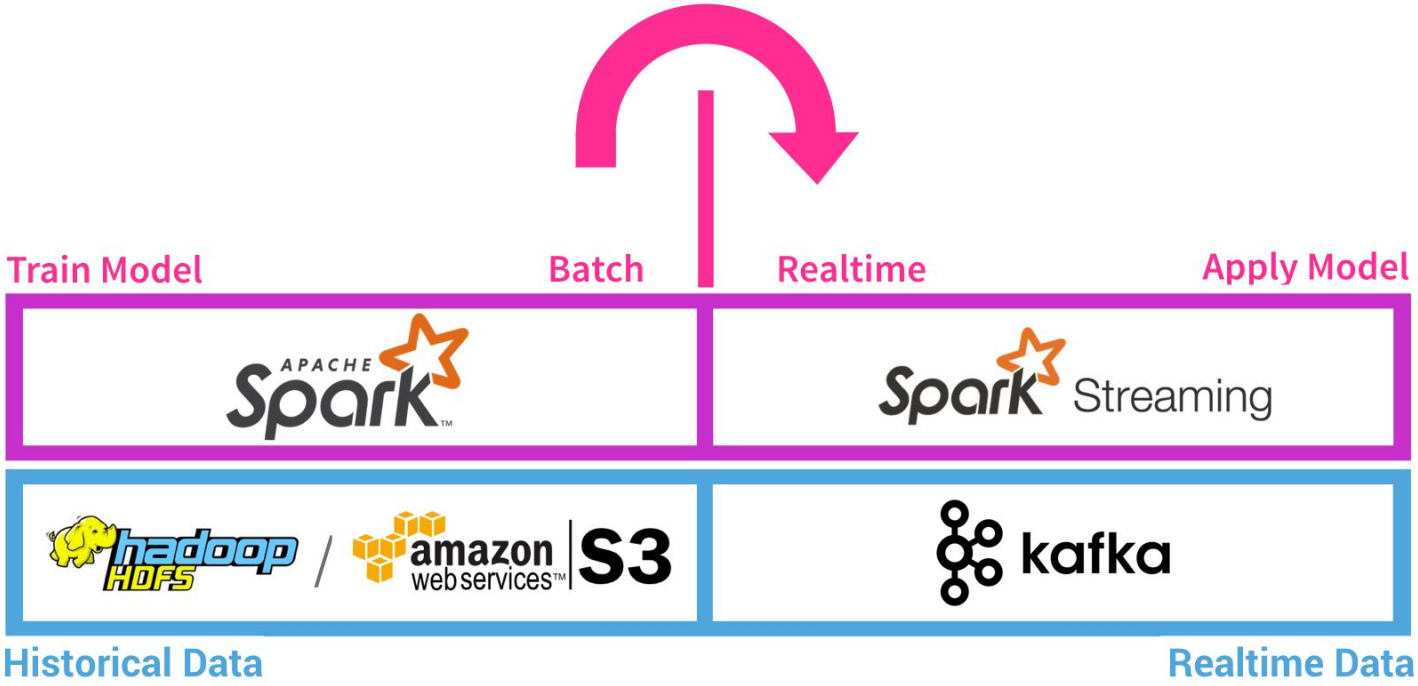
Práctica Big Data

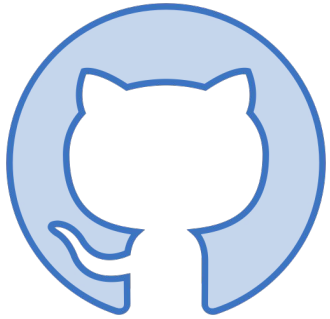


Andrés Muñoz Arcentales

2022

Repositorio

https://github.com/ging/practica_big_data_2019



Basado en

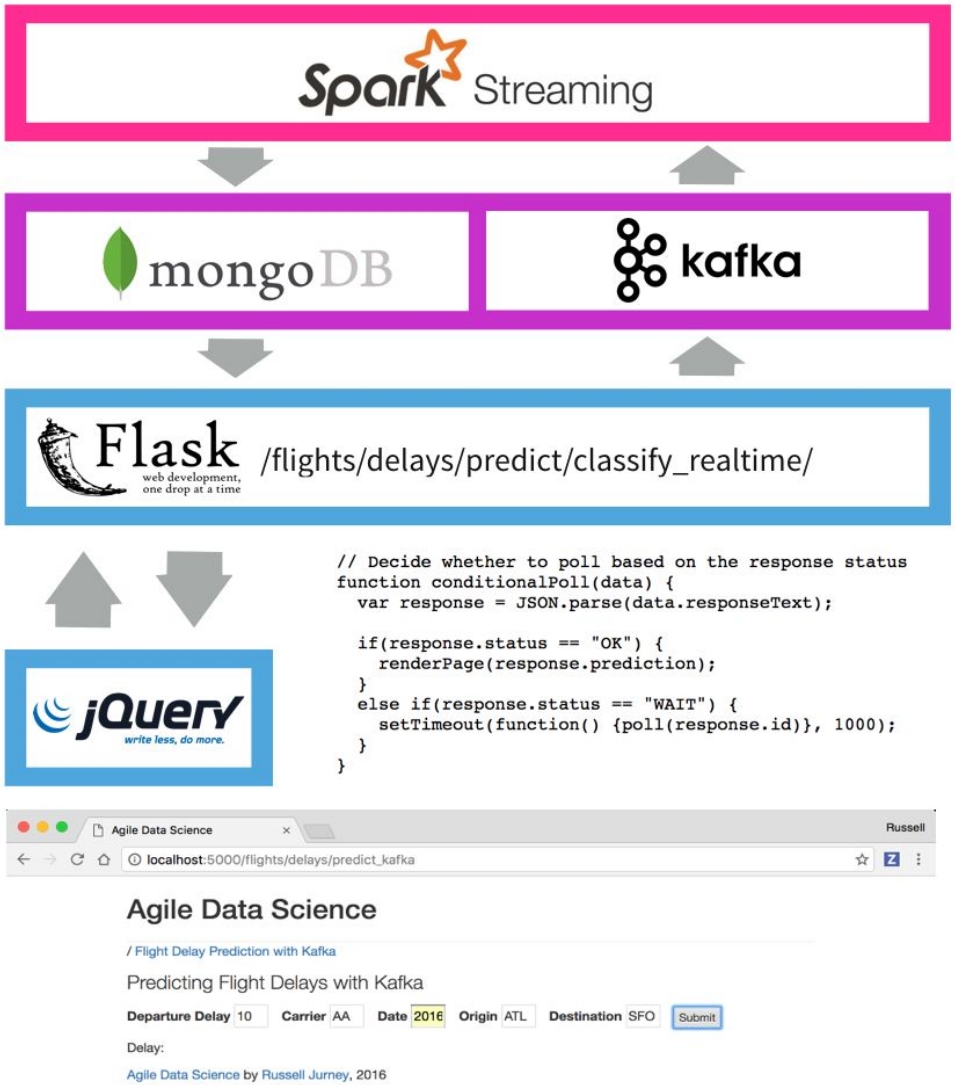
https://github.com/rjurney/Agile_Data_Code_2

Predicción de retraso de vuelos

- Tenemos un dataset que contiene información de vuelos pasados, incluyendo si han salido con retraso o no.
- A partir de esta información, queremos predecir si va a haber retrasos en un vuelo futuro.
- Para ello, entrenamos un modelo predictivo basado en el algoritmo **RandomForest** utilizando los datos que tenemos de vuelos antiguos.
- Tenemos que desplegar una arquitectura completa que nos permita, utilizando el modelo predictivo que hemos creado, realizar predicciones en tiempo real para nuevos vuelos.

Proceso

1. **Descargar** los datos de vuelos pasados
2. **Entrenar** modelo de Machine Learning utilizando los datos de vuelos
3. **Desplegar el job** de Spark que predice el retraso de los vuelos usando el modelo creado
4. Por medio de una interfaz web, el usuario introducirá los datos del vuelo a predecir, que se enviarán al servidor web de **Flask**
5. El servidor web enviará estos datos al job de predicción a través de **Kafka**
6. El job realizará la predicción y la guardará en **Mongo**
7. La interfaz web está constantemente haciendo **polling** para comprobar si se ha realizado ya la predicción
8. En caso afirmativo, se muestra la **predicción** en la interfaz



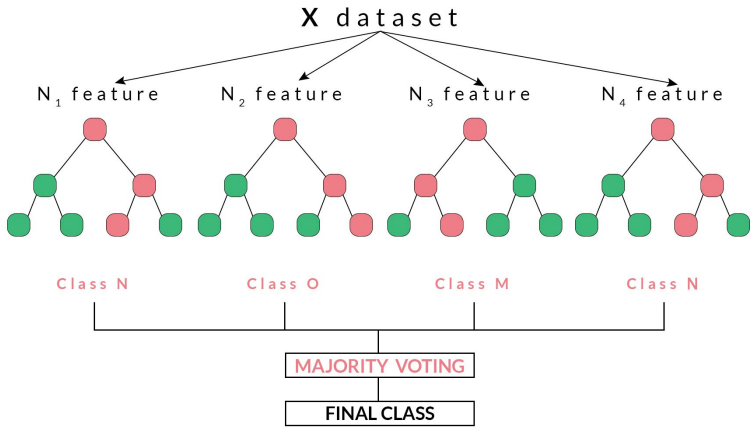
Los datos de vuelos

- El dataset contiene datos del 90-95% de los vuelos con origen en EE.UU desde 2015, publicados en el *Bureau of Transportation Statistics*.
- Algunos campos relevantes:
  - **FlightDate:** Fecha del vuelo
  - **Carrier:** Aerolínea
  - **FlightNum:** Número de vuelo
  - **Origin:** Aeropuerto de origen
  - **Dest:** Aeropuerto de destino
  - **DepDelay:** Retraso inicial

```
2015,1,1,1,4,2015-01-01,"AA",19805,"AA","N787AA","1",12478,...,31703,"JFK", ...
2015,1,1,2,5,2015-01-02,"AA",19805,"AA","N795AA","1",12478,...,31703,"JFK", ...
2015,1,1,3,6,2015-01-03,"AA",19805,"AA","N788AA","1",12478,...,31703,"JFK", ...
```

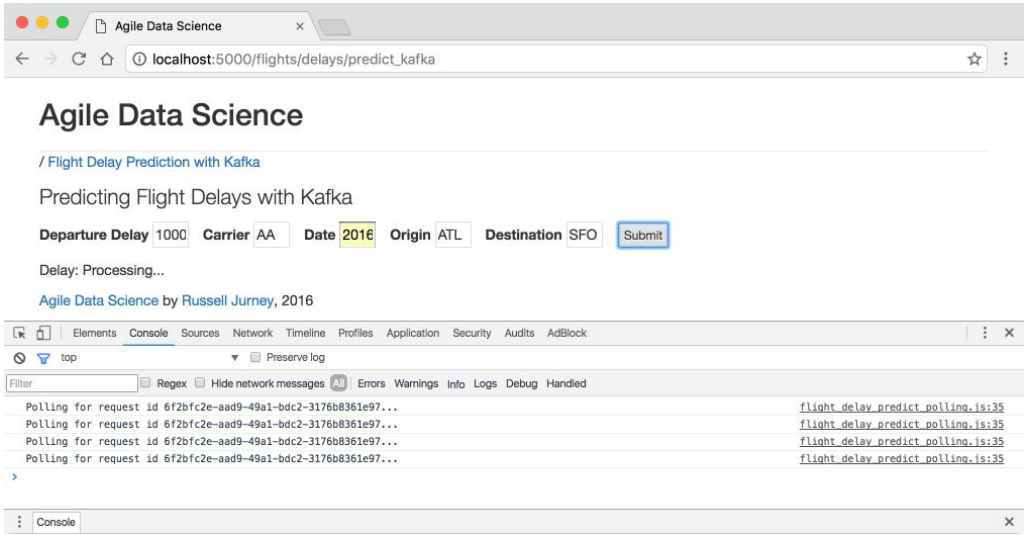
Entrenar el modelo

- Los datos de los vuelos los vamos a emplear para entrenar un modelo predictivo basado en el algoritmo **RandomForest**.
- Todo el proceso de entrenamiento lo vamos a realizar en batch utilizando **PySpark**.
- Como resultado vamos a obtener un modelo que para un nuevo vuelo dado, va a predecir si va a tener o no retraso.



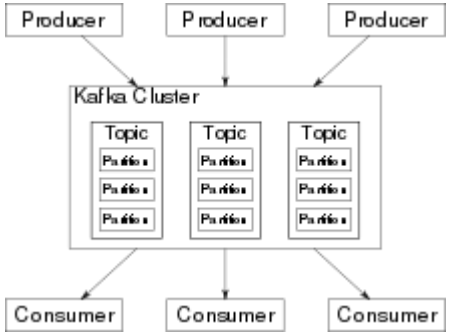
Servidor web

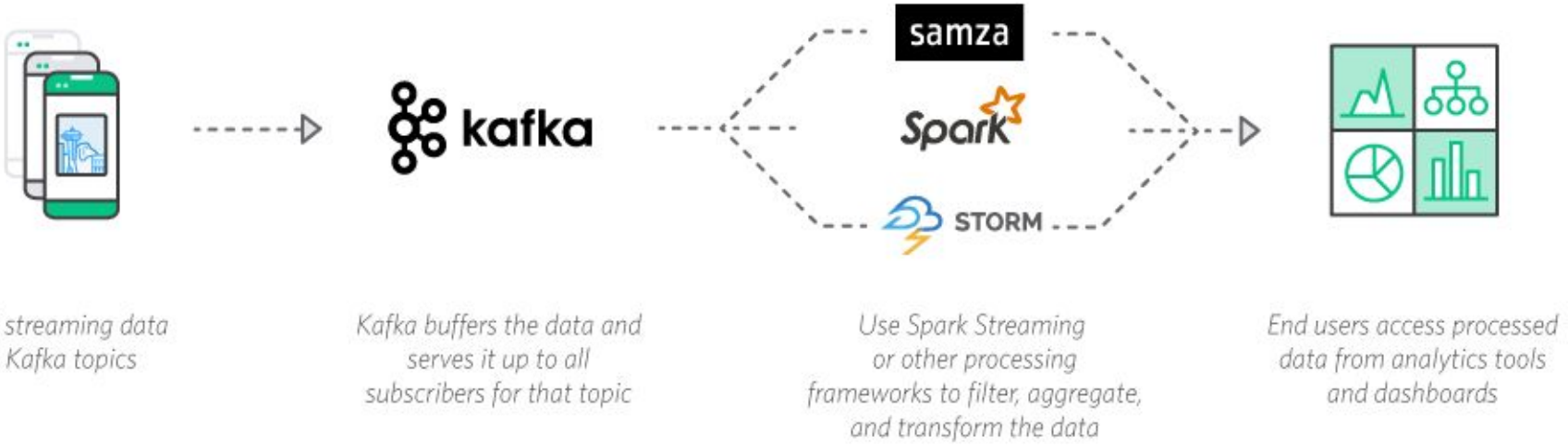
- Para facilitar que el usuario pueda introducir el vuelo para el que quiere realizar una predicción, se emplea una interfaz web.
- El servidor web está implementado con **Flask**, un microframework de **Python** para desarrollar servicios web sencillos.



Envío de datos en tiempo real

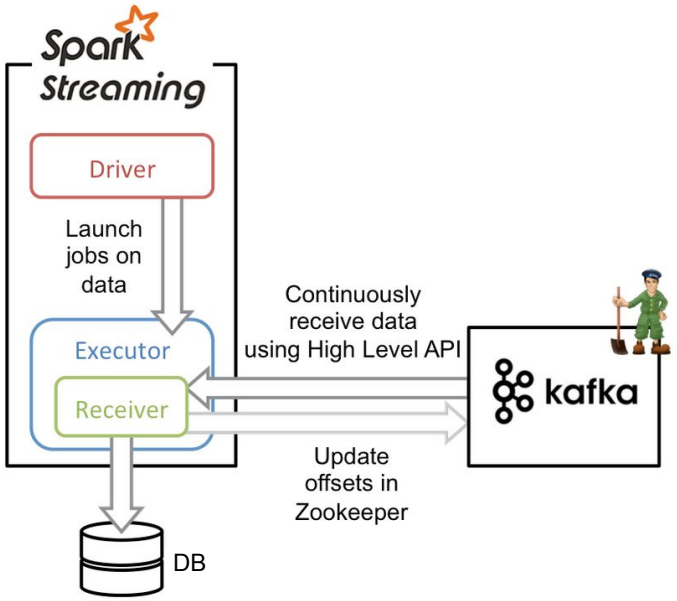
- Para comunicar el servicio web con el job de predicción utilizamos **Kafka**.
- **Kafka** es una herramienta de Apache para crear pipelines de streaming de datos en tiempo real.





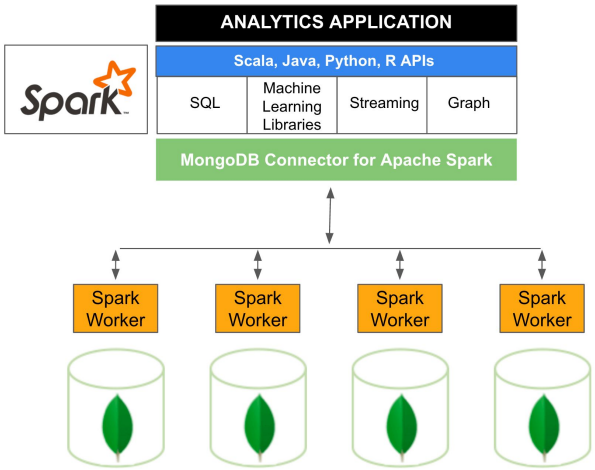
El job de predicción

- Cada vez que un usuario inicia una nueva predicción, le llega al job a través de **Kafka**.
- Para habilitar el cálculo de predicciones en tiempo real vamos a utilizar **Spark Streaming** y el modelo predictivo que hemos entrenado anteriormente.



Guardando la predicción

- Una vez que se ha calculado la predicción, ésta se guarda en la base de datos **MongoDB**.
- La aplicación web consulta continuamente la base de datos (*polling*) para comprobar si se ha calculado la predicción.
- En caso afirmativo, se muestra la predicción en la web.



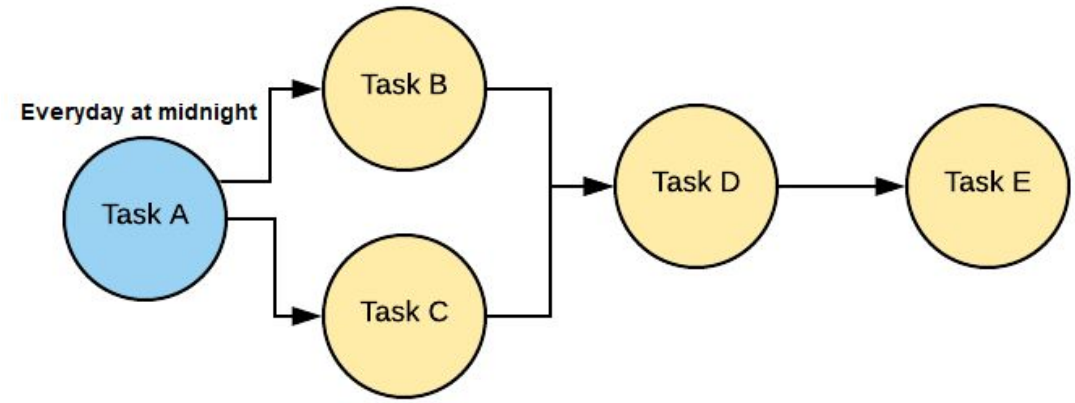
Evaluación

La evaluación se realizará mediante un examen **oral por parejas**:

4 puntos	Lograr el funcionamiento de la práctica sin realizar modificaciones
1 punto	Ejecución del job de predicción con Spark Submit en vez de IntelliJ
1 punto	Dockerizar cada uno de los servicios que componen la arquitectura completa
1 punto	Desplegar el escenario completo usando docker-compose
2 puntos	Desplegar el escenario completo usando kubernetes
1 punto	Desplegar el escenario completo en Google Cloud/AWS
2 puntos	Entrenar el modelo con Apache Airflow

Automatización y planificación de tareas

- **Apache Airflow** es una plataforma para la ejecución planificada de tareas (task scheduling)
  - Las tareas se combinan formando un grafo acíclico dirigido (DAG)
  - Las tareas son programas python
- Permite automatizar tareas, por ejemplo:
  - Eliminar de la BBDD todas las peticiones de predicción que se han realizado en el último mes
  - Reentrenar el modelo una vez a la semana añadiendo nuevos datos



Entrega

- Subir a Moodle los ficheros necesarios para ejecutar la práctica
- Incluir un Readme con las instrucciones de despliegue

Dudas y tutorías

Andrés Muñoz Arcentales joseandres.munoz@upm.es