ID Alumno: 129280

MBDM Módulo 2 y 3

ARQUITECTURA BICIMAD

26 DE ENERO DE 2019

Contenido

Arquitectura de datos BiciMAD

Introducción

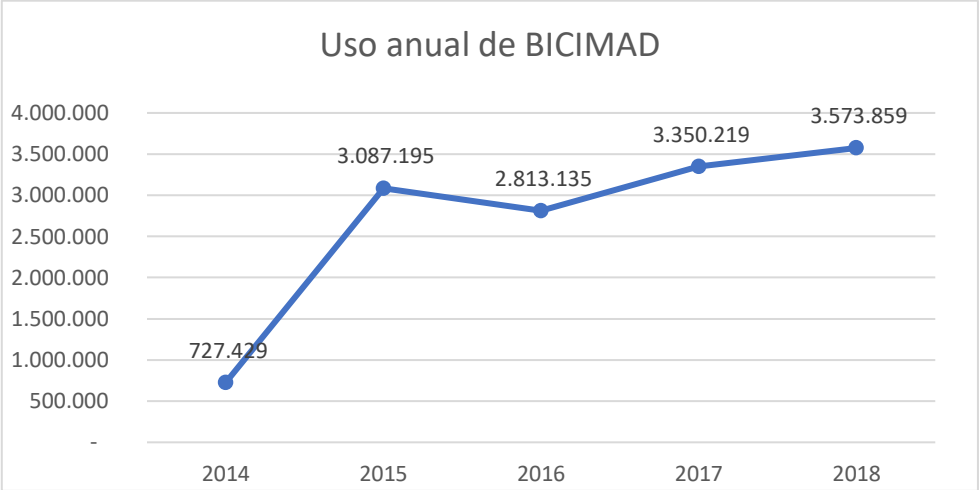
El propósito de este trabajo es diseñar la arquitectura *big data* para BiciMAD. Para la realización del mismo, se analizará en un principio, la fuente y estructura de los datos, posteriormente se plantearán los casos de uso en los que se van a emplear estos datos y por último se irán describiendo los distintos factores que componen la arquitectura: su distribución, las tecnologías seleccionadas, sus características y la integración entre ellas.

¿Qué es BICIMAD?

BiciMAD es el sistema de alquiler público de bicicletas de la ciudad de Madrid. El servicio se ofrece en toda la ciudad a través de bases fijas con bicicletas que cualquier persona puede alquilar pagando por el tiempo de utilización; se quita la bicicleta de una base, se utiliza por el tiempo necesario y luego se devuelve en la misma o en otra diferente. Cuenta con 165 bases, con un total de 2028 bicicletas. BiciMAD tiene asimismo, una aplicación móvil donde los usuarios pueden consultar la disponibilidad de cada base, así como dónde dejarla y, dependiendo de la disponibilidad de las mismas, se le puede aplicar un descuento al precio de uso.

Este servicio nació en 2014 promoviendo la utilización de transportes no contaminantes y eficientes. Las bicicletas cuentan con motor eléctrico y a través de los años, su uso fue creciendo. En su año de lanzamiento, se registraron un total de 727.429 usos, mientras que para 2018 la cifra fue de 3.573.859, un 391% más. En cuanto al uso promedio diario, en 2018 fue de 9.791 usuarios, un 7% más que el año anterior. Para el abastecimiento de las bases, existen seis

camiones que recorren la ciudad cargados de bicicletas y reabastecen el stock de estas.[1]



Usabilidad del servicio de BiciMAD desde sus orígenes hasta la actualidad.[2]

Fuente de datos

Para el servicio mencionado, podemos encontrar las siguientes fuentes y sus respectivos datos:

- Estaciones BiciMAD: En la web oficial, se puede encontrar una API abierta al público que permite consultar información sobre el estado actual de cada una de las estaciones. La información de esta fuente se encuentra en formato *Json* y entre los indicadores que contiene, se pueden destacar los siguientes:
    - *ID*: código único de la base.
    - *Latitude:* latitud de la base en formato WGS84.
    - *Longitude:* longitud de la base en formato WGS84.

[1] Entrevista al director de tecnología de EMT, Madridiario: https://www.madridiario.es/457601/comision-investigacion-bicimad-sistema-sigue-siendo-mejorable

[2] Gráfico de elaboración propia en base a datos abiertos del ayuntamiento de Madrid: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=6d8bdae2be63c410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD

    - *Name*: nombre de la base.
    - *Light:* grado de ocupación de la base, siendo 0 ocupación baja, 1 ocupación alta y 2 ocupación media.
    - *Activate*: estación activa o no activa.
    - *No_available*: disponibilidad de la estación.
    - *Total_bases*: lugares totales de la estación.
    - *Dock_bikes*: número de bicicletas ancladas.
    - *Free_bases*: número de espacios libres.
    - *Reservations_count*: número de reservas activas.

- Camiones de reabastecimiento: información de los camiones y su ubicación geográfica, proveniente de la API de Google Maps, también en formato *Json*:
    - *IdCamión*: código identificativo.
    - *Longitude*: longitud actual del camión.
    - *Latitude*: latitud actual del camión.
    - *Stock*: bicicletas disponibles.
    - *Near_stations*: id de estaciones cercanas en un radio de 3km.

Todos estos datos indicados, son los que serán utilizados en el presente trabajo, cuyo objetivo es determinar su tratamiento según su estructura y periodicidad de uso.

Casos de uso

Antes de comenzar a plantear la arquitectura, resulta necesario describir y entender el uso que tendrán estos datos, por ello en este apartado se pretende dar foco a la utilización de los datos anteriormente planteados.

Se pueden dividir en dos casos de uso consecutivos:

1. Cuadros de mando

Para comenzar a entender el negocio y tener un tablero donde poder controlar cómo funciona el servicio, se creará un cuadro de mando indicando los principales KPIs (*Key performance indicator*):

- % Estaciones sin utilizar.
- % Estaciones sin bicicletas disponibles.
- Bicicletas que están siendo usadas.
- Reservas programadas para los próximos 5 minutos.
- Camiones activos.
- Duración media de uso de bicicleta.
- Código Postal de mayor demanda.
- Código Postal de menor demanda.

Este cuadro de mando debería tener un ingreso de datos cercano al tiempo real, con segundos de demora como máximo, para poder tener un control real de lo que sucede y debe ser presentado en una plataforma dinámica, estática y coherente.

2. Logística de camiones

Teniendo en cuenta la flota disponible de camiones, la demanda previamente descripta de bicicletas, y la gran cantidad de bases que hay distribuidas en la ciudad, se creará una solución para la logística de reparto y recolección de bicicletas.

Una solución posible consiste en montar una aplicación móvil para los choferes de los camiones, donde se le indique la próxima estación a la que debe acudir, el recorrido que debe tomar y cuántas bicicletas debe anclar o retirar. Una vez realizada la entrega, toca el botón de OK en su aplicación y se le indica la siguiente parada.

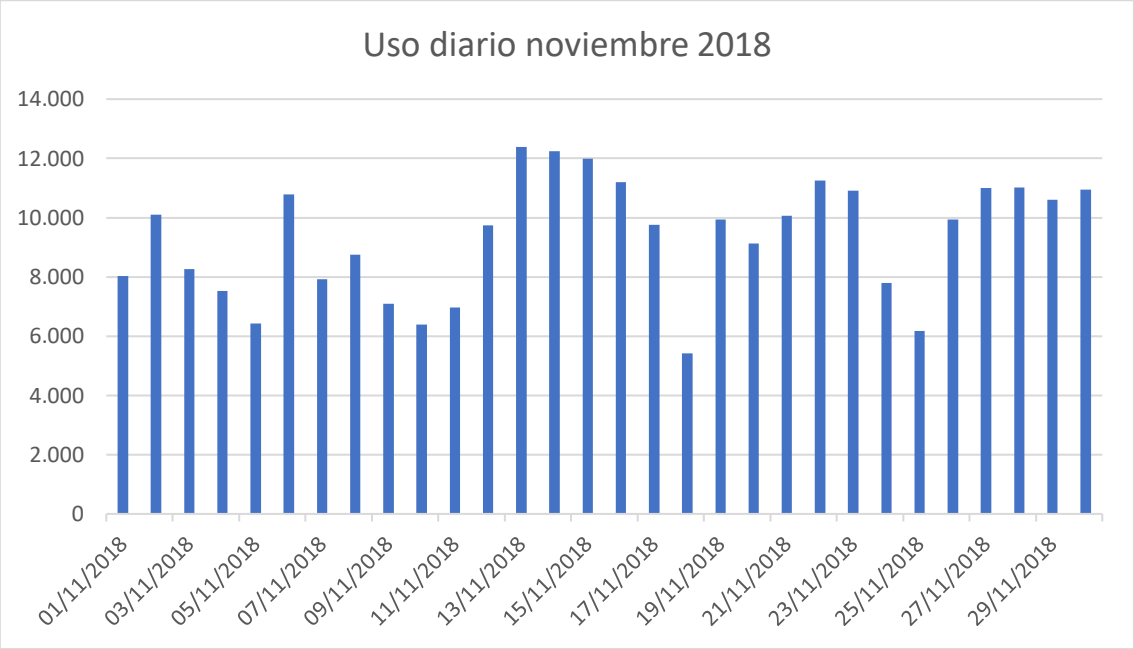
Esto logrará una gestión más eficiente del recorrido de los camiones, una mayor disponibilidad de bicicletas en las bases y la posibilidad de anticiparse a picos de demanda.

Características de la arquitectura

Tipo de distribución

Para comenzar a plantearnos la arquitectura, es primordial decidir en un principio, en qué tipo de distribución va a ser montada. Existen tres tipos de distribuciones: física, *cloud* e híbrida. Hoy en día, la distribución física y la *cloud* están a un nivel muy similar, donde los servicios y tecnologías que se ofrecen, en ambas están niveladas. La decisión para esta selección radica, básicamente, en la infraestructura que se posee y en la continuidad que tendrá el flujo de datos.

Por lo tanto, a continuación analizaremos un gráfico de los usos diarios de BiciMAD:



3

Como podemos ver en este caso de noviembre del 2018, el uso a lo largo de los días es disparejo, existe diferencia de hasta cuatro mil bicicletas al día. Con esto,

[3] Gráfico de elaboración propia en base a datos abiertos del ayuntamiento de Madrid: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=6d8bdae2be63c410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD

sumado a un análisis predictivo, se podría determinar que no es necesario tener todos los días los seis camiones activos y como consecuencia, los datos que ingresarían a nuestras bases de datos serían menores.

Tomando en cuenta que el flujo de datos no es demasiado grande, y que probablemente existan picos y valles en la demanda, resulta más conveniente crear la arquitectura en un entorno *cloud* o en la nube. De esta forma, se usaría el dinero en infraestructura y computación dependiendo de la demanda de cada momento, siendo flexibles ante sus cambios.

Para el segundo caso de uso, la opción más viable es integrar Google Maps en nuestra aplicación y consultar datos de geolocalización, por lo que utilizar Google Cloud Platform como entorno *cloud*, indicaría la óptima viabilidad. Esta plataforma nos proveería de herramientas de ingesta, procesado, almacenamiento y explotación de datos. Asimismo, contaríamos con la escalabilidad necesaria, la tolerancia a fallos, la seguridad resuelta y por último, el beneficio de que se integraría a la perfección con la API de Google Maps.

Ingesta

Para comenzar el flujo de datos, en un primer lugar necesitamos una capa de ingesta, con el fin de programar tareas de consulta hacia las APIs que nos proveen de información. Para esta etapa, la herramienta seleccionada es Kafka.

Kafka es una herramienta de mensajería que canaliza múltiples flujos de datos a través de un sistema unificado, siendo capaz de transportar trescientos mil mensajes por segundo. Tiene además escalabilidad horizontal, y sirve como amortiguador entre productor y consumidor de datos, sin perder los registros en caso de caída de los servidores[4]. Kafka funciona principalmente con logs, pero

[4] Web de grupo CESA, Orellana-Bracamante, Daniel. Junio 2017: http://www.grupocesa.com/index.php/2017/06/15/kafka-que-es-y-por-que-deberiamos-estar-interesados-en-el/

además, es posible trabajar con formato *Json* y, en caso necesario, serializar y deserializar la información.[5]

Además, se utilizará *PubSub*, similar a *Kafka* y disponible en Google Cloud Platform, para la ingesta de los datos de la API de Google Maps, dado que se integra perfectamente con los productos de la misma empresa. Funciona también en tiempo real, es escalable a cientos de millones de mensajes por segundo y trabaja de forma distribuida como Kafka.[6]

Para integrar Kafka dentro de la plataforma de Google Cloud, se hará uso del módulo Compute Engine, en el cual se monta una máquina virtual Linux y dentro de ella se le instala Kafka, a través de donde se pueden ejecutar todos los *Jobs* que se requiera.[7]

Como orquestador se utilizará Apache Nifi, quién se define como encargado de movilizar datos entre sistemas no conectados con una velocidad real-time. Nifi es fácil de utilizar y tiene una muy buena usabilidad con APIs, realizando conexiones, configurando credenciales y disponibilizando la información en herramientas de ingesta o almacenamiento como Kafka y HDFS.[8] Teniendo en cuenta esto, la herramienta nos serviría para programar las consultas a las APIs y depositarlo en Kafka.

---

[5] Consume JSON Messages From Kafka Using Kafka-Python's Deserializer, Kumar,Mukesh. Nov 2017: https://medium.com/@mukeshkumar_46704/consume-json-messages-from-kafka-using-kafka-pythons-deserializer-859f5d39e02c

[6] Web oficial de GCP: https://cloud.google.com/pubsub/?hl=es-419

[7] Apache Kafka Foundation Course - Installing Kafka in Google Cloud: https://www.learningjournal.guru/courses/kafka/kafka-foundation-training/kafka-in-gcp/

[8] Pull data from Twitter and push data to Elasticsearch using Apache NiFi, Vivas, Joakim. Nov 2017: https://www.theninjacto.xyz/Pull-data-Twitter-push-to-Elasticsearch/

Procesamiento

Los datos extraídos de las API yacientes en Kafka y PubSub, deberán pasar por un procesamiento antes de ser almacenados en la base de datos. Este procesamiento se hará mediante DataFlow, herramienta de Google similar a Spark que permite la transmisión de datos en tiempo real y *batch* mediante lotes, con base de Apache Beam.[9] Esta tecnología funciona tanto con PubSub como con Kafka.[10] Según estudio de mercado realizado por InfoWorld en 2016, DataFlow es hasta 5 veces más rápido que Spark y lo destaca su autoescalabilidad que permite el procesamiento con más de mil *cores*.[11]

Almacenamiento

Una vez procesados los datos, irán a Cloud Storage para ser almacenados, otra plataforma de Google que permite el almacenamiento distribuido de grandes cantidades de datos. Funciona como un *datalake*, similar a HDFS y su costo es muy bajo.

Para el almacenamiento de datos agregados, se hará uso de Google Big Query, una base de datos creada para almacenar Big Data y hacer analítica sobre los datos. Estructura la información de modo que se pueda extraer mediante consultas SQL en un tiempo mínimo de reacción y se cobra sólo por su uso. Para el primer caso de uso, los KPI serán almacenados en esta base de forma agregada y se irán actualizando.

[9] Web oficial de GCP: https://cloud.google.com/dataflow/?hl=es

[10] Web oficial de GCP: https://cloud.google.com/blog/products/gcp/apache-kafka-for-gcp-users-connectors-for-pubsub-dataflow-and-bigquery

[11] Google Cloud Dataflow vs. Apache Spark: Benchmarks are in, Oliver, Andrew. Mayo 2016: https://www.infoworld.com/article/3064728/analytics/google-cloud-dataflow-vs-apache-spark-benchmarks-are-in.html

Explotación

En esta etapa, es donde se enriquece la información y se les agregan valor a los datos ya almacenados. La explotación se hará de formas distintas dependiendo del caso de uso:

1. Para la generación y proyección de los KPI, se hará uso de la herramienta Tableau: plataforma de visualización de datos con conector disponible a Google BigQuery. Con Tableau online se puede combinar su velocidad y facilidad de uso con la agilidad de BigQuery, dando como resultado informes generadores de valor, los cuales se actualizan constantemente y están a disponibles en cualquier dispositivo.[12] Se podría haber elegido Google Data Studio, también disponible en GCP (Google Cloud Platform), pero se encuentra en modo beta y todavía no dispone de muchas características. Los KPIs se generan mediante medidas dentro de Tableau y es la plataforma la que genera la transformación del dato, almacenando su fórmula en ella.
2. Como se mencionó anteriormente, se debe generar un modelo de camino crítico para los camiones reabastecedores de bicicletas. Para ello, se hará uso de la herramienta Colab, se trata de una incorporación de Jupyter dentro de la nube de GCP que tiene además la posibilidad de compartir los *notebooks* realizados y trabajarlos en equipo. En esta herramienta, en forma *batch* se extraerán los datos de BigQuery al principio del día y dependiendo de la demanda histórica y la predicción del clima, se hará una predicción del uso de bicicleta para ese día, actualizando esta información cada 2 horas y almacenándola nuevamente en BigQuery. Luego, se extraerán los datos nuevamente de BigQuery y en forma *batch* (cada vez que el camión haya terminado una entrega), determinando la posición actual del camión, sus estaciones en un radio de 3km y la urgencia de reabastecimiento de cada una de ellas: ponderación entre lugares libres y predicción de ocupación para la próxima hora. Con lo que

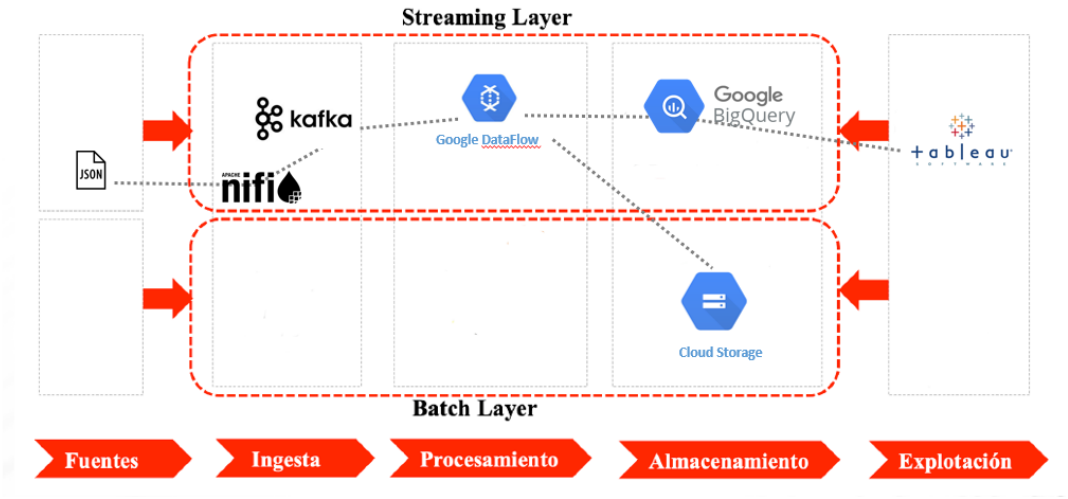
[12] Google BigQuery y Tableau: prácticas recomendadas, Jeff Feng: https://www.tableau.com/es-es/learn/whitepapers/google-bigquery-tableau-best-practices

se podrá devolver a la aplicación, la base seleccionada como próximo destino. Para aumentar el intervalo de confianza de la predicción de demanda, se le va a incluir predicción del tiempo diario extraída de la API de DarkSky, la cual nos provee información detallada del tiempo y es gratuita hasta 1000 consultas por día.[13]

Diseño final y conclusiones

Diseño final

Tomando el primer caso de uso, la arquitectura se puede definir de la siguiente forma:

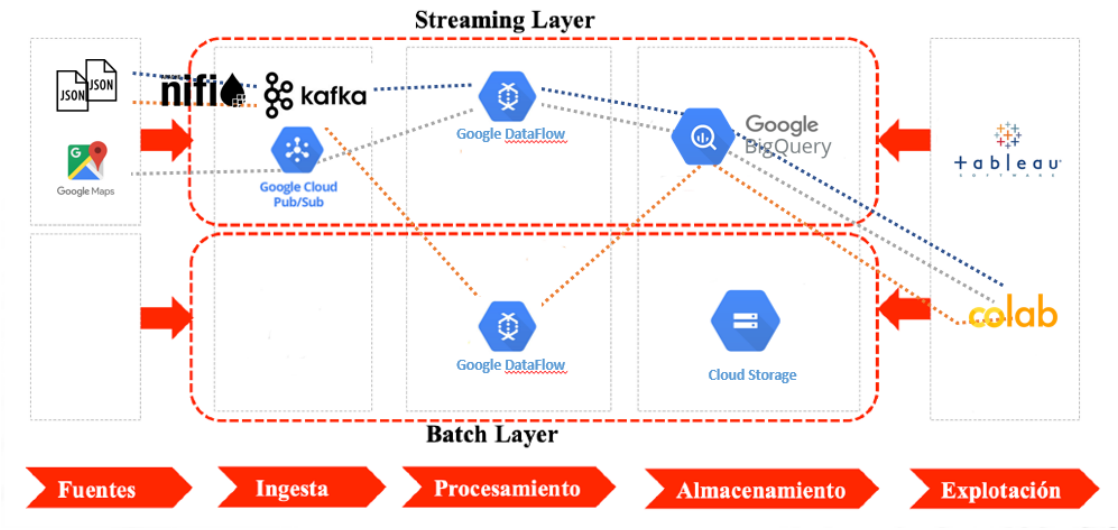


El objetivo de esta primera arquitectura es tener los KPI en tiempo real, para lo mismo, tendremos a Kafka, montado en una máquina virtual, consultando a los datos que le proveerá Nifi de la API de BiciMAD. Luego, los procesaría DataFlow almacenándolos en BigQuery, para que Tableau los consulte rápidamente y, con sus fórmulas almacenadas, calcule los KPI con el flujo de datos entrante. Esto crearía una corriente de datos constante, en la que el dato la recorrería en menos de un segundo. Además, DataFlow estará enviando una réplica de los datos a

[13] Web oficial de DarkSky: https://darksky.net/dev

Cloud Storage, con el objetivo de almacenarlos y tener información histórica del uso.

En el segundo caso de uso, la arquitectura sería la siguiente:



- **Google Maps API:** Pub/sub consulta directamente a la API de Google Maps en tiempo real, aprovechando la integración nativa entre ellas, y lo pasa a DataFlow, quien lo procesa y lo almacena en BigQuery. Con esto tendríamos un flujo constante de la geo posición del camión. Luego, cuando el conductor mande la petición para consultar su próxima estación de destino, Colab extrae el dato desagregado de ese camión de BigQuery, ejecuta el script con el modelo recomendando la siguiente parada y PubSub se lo devolverá a la aplicación móvil para que automáticamente se lo marque como destino de viaje.
- **DarkSky API:** La API es consultada por Nifi cada una hora. Luego Kafka lo ingesta a DataFlow, quien procesa la información y filtra los campos no necesarios en un proceso micro batch, luego se almacena en BigQuery. Colab consulta la información de BigQuery para estimar el impacto de las condiciones climatológicas en la demanda de bicicletas, y lo utiliza para el algoritmo de base destino de los camiones. Esta información luego es transportada mediante Kafka a Cloud Storage, donde se almacena el dato histórico para poder afinar el algoritmo a lo largo del tiempo.

- **BiciMAD API**: El proceso permanece igual que en el primer caso, solo que luego de ser almacenado el dato en BigQuery, Colab lo consulta para calcular su algoritmo de base destino. Luego esta información se guarda en CloudStorage, utilizando Kafka como conector, para complementar el almacenamiento histórico.

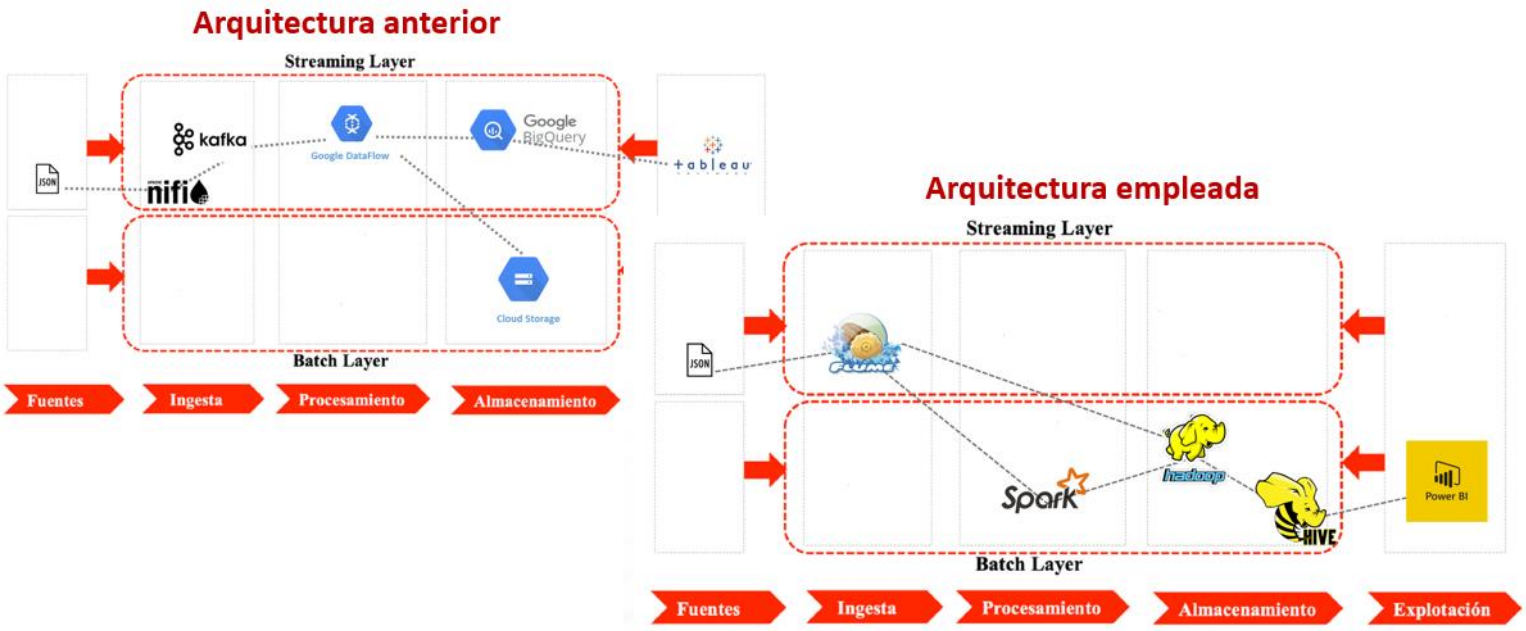
Conclusión

La arquitectura es flexible y cumple con los objetivos establecidos para realizar los casos de uso. Los puntos más destacables son: su escalabilidad, la rapidez del flujo de dato con la que está pensada y el aprovechamiento de los costos variables adaptables a la frecuencia de uso.

ANEXO I: Ingesta y pretratamiento de datos.

Principales cambios

Dado que la arquitectura del módulo anterior era ambiciosa y todavía no se cuenta con los conocimientos necesarios para implementarla, se presenta una serie de cambios:



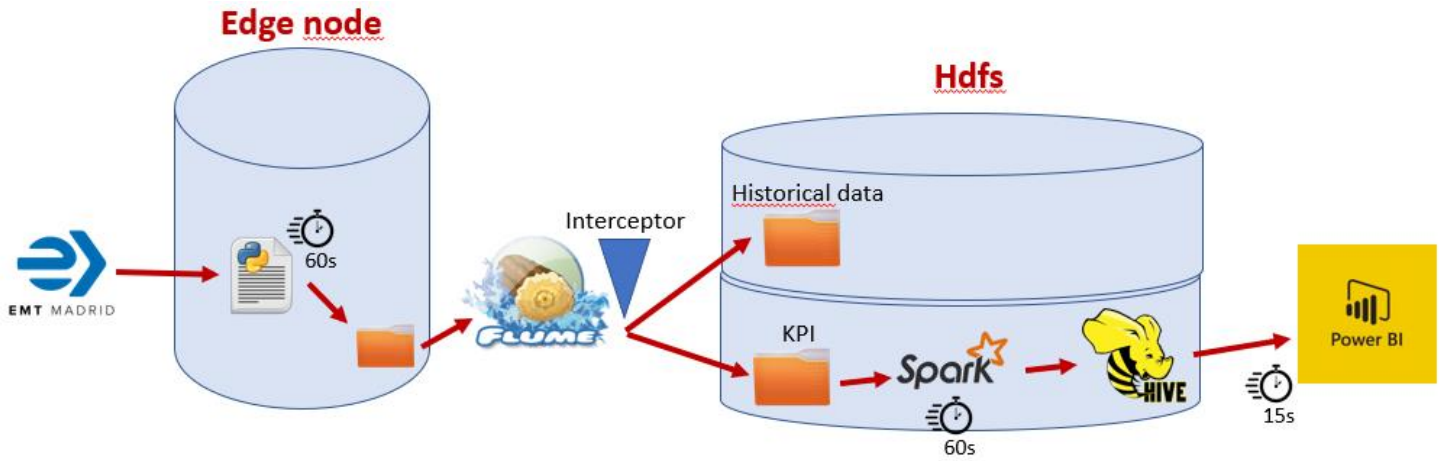
Los dos principales cambios son:

1. El proceso pasa a ser batch, ya que todavía no es posible el uso de una base de datos que posibilite la consulta y visualización instantánea de los datos.
2. Las herramientas utilizadas dejan de ser de Google Cloud, dado que todavía no se requiere el uso de los datos proporcionados por la API de Google Maps.

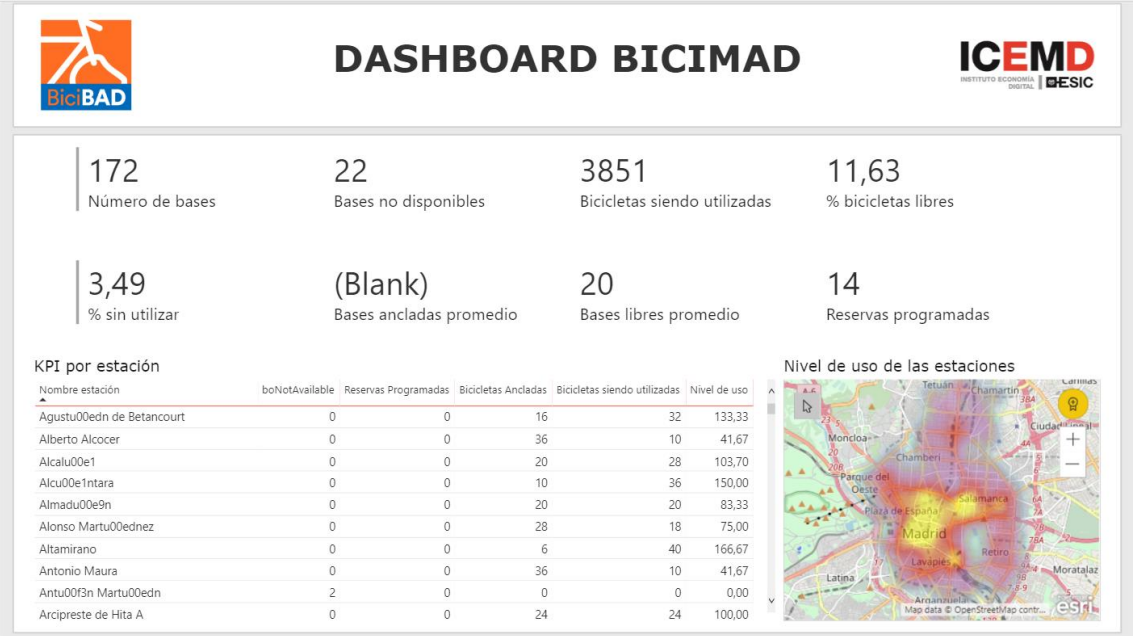
Nuevo proceso

El nuevo proceso cuenta con los siguientes pasos:

1. Un script de Python genera una llamada a la API de bicimad cada 60 segundos y guarda el resultado en un fichero json.
2. Flume escucha la carpeta donde se guardan estos ficheros con un source SpoolDir, los recoge, corrige sus defectos de escritura mediante un interceptor y los deja en dos carpetas distintas:
    a. Carpeta “kpi/” para luego ser procesados.
    b. Carpeta dinámica “historicalData/%y%m%d/%H%M/”, en donde llegan los ficheros por un *file channel* para tener un almacenamiento historificado y luego procesados cuando se lo requiera.
3. Cada 60 segundos, Spark agarra los datos de la carpeta “kpi/” y genera dos procesamientos:
    a. Fichero de KPI agregados.
    b. Fichero de KPI por estación.
4. Hive sólo interactúa creando tablas externas con los dos ficheros que genera Spark, para que luego Power BI, mediante una conexión odbc, pueda tomar estos datos y mostrarlos en forma de Dashboard.



Resultado final



Nombre estación	boNotAvailable	Reservas Programadas	Bicicletas Ancladas	Bicicletas siendo utilizadas	Nivel de uso
Agustu00edn de Betancourt	0	0	16	32	133,33
Alberto Alcocer	0	0	36	10	41,67
Alcalu00e1	0	0	20	28	103,70
Alcu00e1ntara	0	0	10	36	150,00
Almadu00e9n	0	0	20	20	83,33
Alonso Martu00ednez	0	0	28	18	75,00
Altamirano	0	0	6	40	166,67
Antonio Maura	0	0	36	10	41,67
Antu00f3n Martu00edn	2	0	0	0	0,00
Arcipreste de Hita A	0	0	24	24	100,00

Limitaciones:

1. Por problemas de permisos, el Cron para ejecutar el proceso Spark periódicamente no se puede implementar.
2. Los caracteres especiales, como las tildes en los nombres de las estaciones, aparecen como código en el dashboard final.

Próximos pasos:

1. Realizar el proceso en real-time y llevarlo a una base de datos que lo soporte.
2. Ingestar los datos de la API de DarkSky para poder hacer luego pronósticos de demanda dependiendo del clima.

Bibliografía

1. Web oficial ayuntamiento de Madrid: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=6d8bdae2be63c410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD
2. Entrevista al director de tecnología de EMT, Madridiario: https://www.madridiario.es/457601/comision-investigacion-bicimad-sistema-sigue-siendo-mejorable
3. Web de grupo CESA, Orellana-Bracamante, Daniel. Junio 2017: http://www.grupocesa.com/index.php/2017/06/15/kafka-que-es-y-por-que-deberiamos-estar-interesados-en-el/
4. Consume JSON Messages From Kafka Using Kafka-Python's Deserializer, Kumar,Mukesh. Nov 2017: https://medium.com/@mukeshkumar_46704/consume-json-messages-from-kafka-using-kafka-pythons-deserializer-859f5d39e02c
5. Web oficial de GCP: https://cloud.google.com/pubsub/?hl=es-419
6. Apache Kafka Foundation Course - Installing Kafka in Google Cloud: https://www.learningjournal.guru/courses/kafka/kafka-foundation-training/kafka-in-gcp/
7. Pull data from Twitter and push data to Elasticsearch using Apache NiFi, Vivas, Joakim. Nov 2017: https://www.theninjacto.xyz/Pull-data-Twitter-push-to-Elasticsearch/
8. Web oficial de GCP: https://cloud.google.com/dataflow/?hl=es
9. Web oficial de GCP: https://cloud.google.com/blog/products/gcp/apache-kafka-for-gcp-users-connectors-for-pubsub-dataflow-and-bigquery
10. Google Cloud Dataflow vs. Apache Spark: Benchmarks are in, Oliver, Andrew. Mayo 2016: https://www.infoworld.com/article/3064728/analytics/google-cloud-dataflow-vs-apache-spark-benchmarks-are-in.html

11. Google BigQuery y Tableau: prácticas recomendadas, Jeff Feng: https://www.tableau.com/es-es/learn/whitepapers/google-bigquery-tableau-best-practices
12. Web oficial de DarkSky: https://darksky.net/dev