Luis Alberto Cruz Amaya
12 de agosto de 2019

# Justificación de arquitectura

Dado el problema presentado era necesario contar con una arquitectura completa, así que supuse una arquitectura a gran escala del sitio web así como del modulo para hacer el calculo de precios.

Primero que nada, la arquitectura del sitio web es simple pero efectiva, pues es conveniente tener un API en ECS primero que nada por el precio. Si elegimos por ejemplo una instancia de tipo t3.medium con un esquema de maquinas de EC2 montadas sobre un Docker el precio sería de 35.712 USD al mes por instancia, suponiendo  que tengamos un load balancer con una instancia en cada lado el precio por el API sería de 71.424 USD mensuales.

Después el hacer que todos los get del API consuman de SQL hace sumamente rápidas las consultas y nos permite que la base de datos este siempre disponible para consultas y actualizaciones, ayudándonos a que nunca tengamos el problema de no mostrar información. En este caso propuse SQS al ser un servicio de Amazon escalarle y sumamente económico (0.40 USD por millón de solicitudes) aunque también podría usarse un Redis, memcache, o similares.

Una de las partes principales o mas bien lo que hace la magia en todo este asunto es el Job de AWS Glue, (Paso 3 y 9) pues nos permite ahorrar mucho trabajo, en el caso del código no puse este demo ya que la parte de programación es sumamente sencilla, lo mas complicado es su configuración en la consola de Amazon, mas que nada por la parte de permisos, y tener que hacer el modelo da la base de datos de nuevo pero como un catalogo de Glue, de esta manera el listener para dispara los updates en SQS puede ser un update o un create en RDS.

El otro job de AWS Glue (Ejemplo en el pseudo código) lo único que tiene que hacer es obtener los datos y mandar a llamar a las funciones correspondientes para después actualizar los datos, la parte de la actualización esta omitida ya que como funciona Glue basta con que cuando termine de hacerse el trabajo en paralelo de pyspark se pasen los datos al catalogo y Glue hace la inserción o actualización de manera automática y una vez ms esto va del lado de a configuración

en la consola de Amazon, y la actualización y reflejo de precios y demás lo hacer el otro Job que configuramos antes.

Respecto a las funciones, consideré que la mas compleja es el calculo de los indicadores de la zona, pues el calculo del precio por metro cuadrado dependerá de los indicadores de la zona, estado de la propiedad, “lujo” (Niveles de acabado, tipos de piso, etc) servicios con los que se cuenta, y cada uno de estos tiene u valor, solo es cuestión de tomar las variables y hacer las operaciones.

Una parte divertida puede ser el calculo del tipo de cambio de USD a MXN, ya que básicamente depende de cuanto se quiera gastar, hay endpoints de paga que en el supuesto de que se actualice cada hora pueden costar 10 USD mensuales, o buscar en las implementaciones de instituciones de bovino y usar sus endpoints.

Dado el problema que se presento y las variables que se tenían esta es una de las mejores soluciones que puedo presentar, pues el costo de hacer esto sería aproximadamente de 85 USD (Donde la mayoría del presupuesto se va en el API que no esta explicado aquí) si se omite la parte de ECS y el API esta en lambdas el precio disminuiría considerablemente a cerca de 20 UDS.

La parte de proponerlo en ECS es debido a que hasta hace un par de años era popular hacer API’s en Flash, java Spring, o similares y migrar esto hacía lambdas no suele ser una tarea sencilla, aunque depende mucho de la infraestructura del API, pero si se esta implementando en micro servicios sería una manera mas eficiente de hacer todo y el diagrama se podría simplificar aun mas.