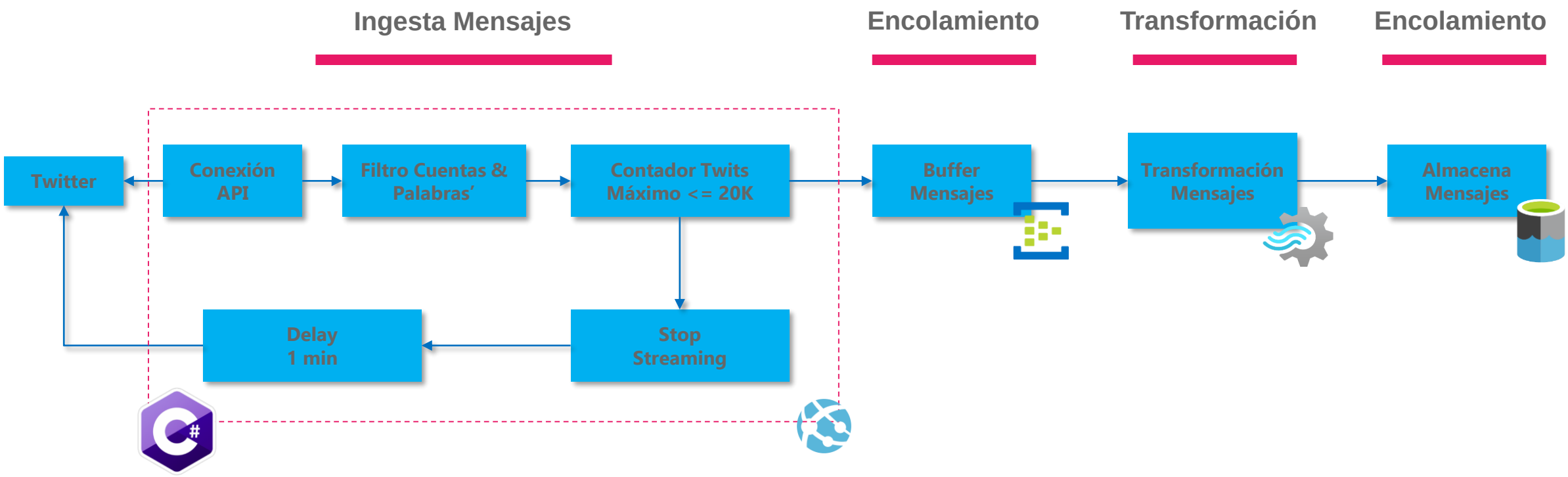
# Webscrapping – Twitter
## Seguridad

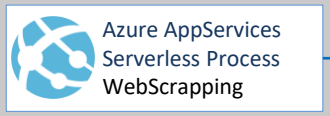
# Arquitectura de la Solución

Streaming Mensajes



# Arquitectura de la Solución

## Streaming Mensajes

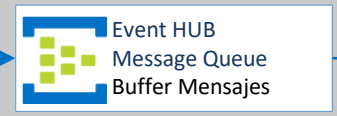



**Contexto**

Desarrollo construido en C# que permite el acceso a API Twitter para extraer mensajes de canales particulares y producir eventos.

**Costos**

1 x 730 Horas = 54.75 USD / Mes

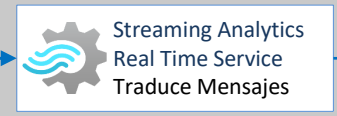



**Contexto**

Receptor y encolador de eventos que se producen en el exterior asociados a twitter a partir de aplicaciones externas.

**Costos**

1 x 730 Horas = 22.18 USD / Mes




**Contexto**

Consumidor y transformador de eventos que son almacenados en grupos de consumo dentro de Event Hub.

**Costos**

1 x 730 Horas = 80.30 USD / Mes




**Contexto**

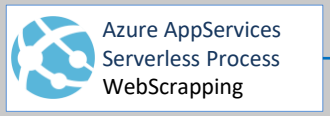
Storage que permite el almacenamiento de mensajes de acuerdo a los datos que se envían a través de Streamig Analytcs.

**Costos**

10 GB = 51.29 USD / Mes

# Arquitectura de la Solución

## Streaming Mensajes – Configuración




**Configuración**

Se cuentan con variables de:

- Key Access para Twitter
- Gestión de Twits
- Gestion de Delay

**Tier**

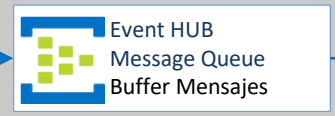
B1
100 total ACU
1.75 GB memory
A-Series compute equivalent

**Input**

Mensajes API REST Twitter

**Output**

Json por Twits




**Configuración**

Se cuenta con un name space denominado como devicehub en el cual se depositan los eventos y son agregados al consumer group “consumergroupdevicehub”

**Tier**

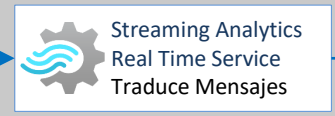
Standard
Throughput Units Min 3 max 10

**Input**

Eventos en formato Json

**Output**

Eventos en formato Json




**Configuración**

Se cuenta con un input denominado como “inputdevicehub” y un output denominado como “outputscwebscraping” para la salida de datos.

**Tier**

Standard
Streaming Units 3

**Input**

Mensajes API REST Twitter

**Output**

Json por Twits




**Configuración**

Se cuenta con una instancia en Datalake Gen2 con nombre “scwebscraping” un contenedor de registro denominado como “outputstreaming” en el que se depositan archivos.

**Tier**

10 GB

**Input**

Mensajes API REST Twitter

**Output**

Json por Twits

# Componentes de la Solución

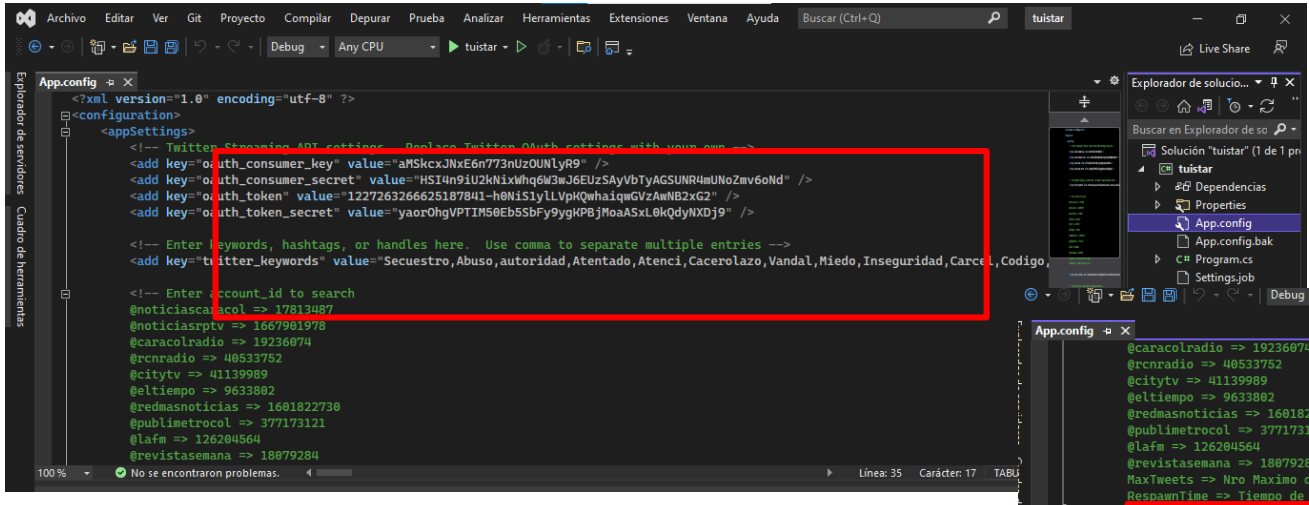
## Extracción Mensajes Twitter

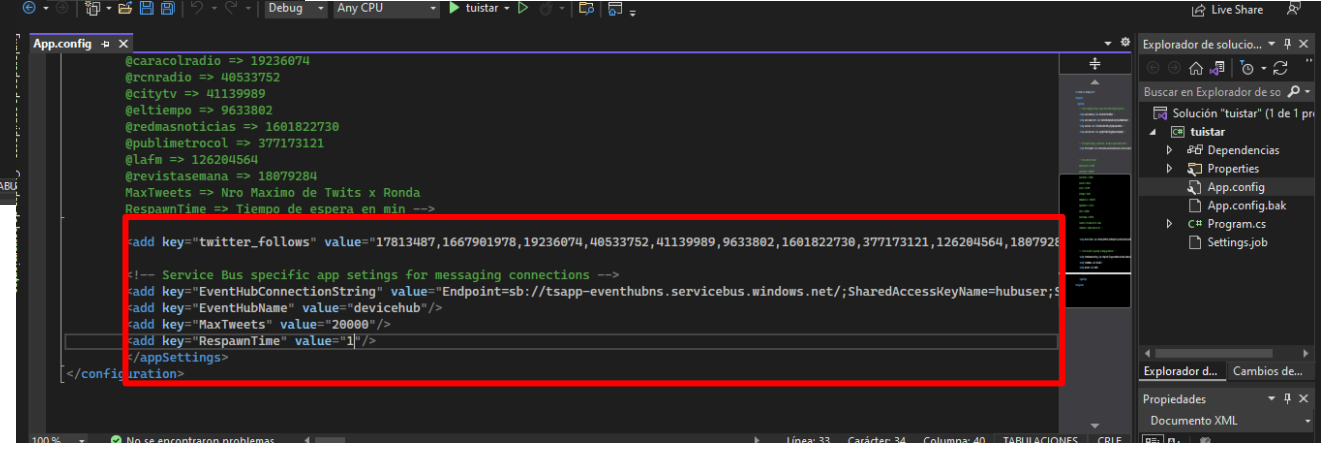
Se ha desarrollado el código de extracción que permite controlar mediante key el acceso a la api de twitter mediante las siguientes variables:

- oauth_consumer_key
- oauth_consumer_secret
- oauth_token
- oauth_token_secret

Adicionalmente, se cuenta con dos variables que permiten controlar el flujo de mensajes que se consumen desde twitter que son:

- MaxTweets: Numero máximo de twits a extraer
- RespawnTime: Tiempo de espera entre cada Batch de envío de twits.



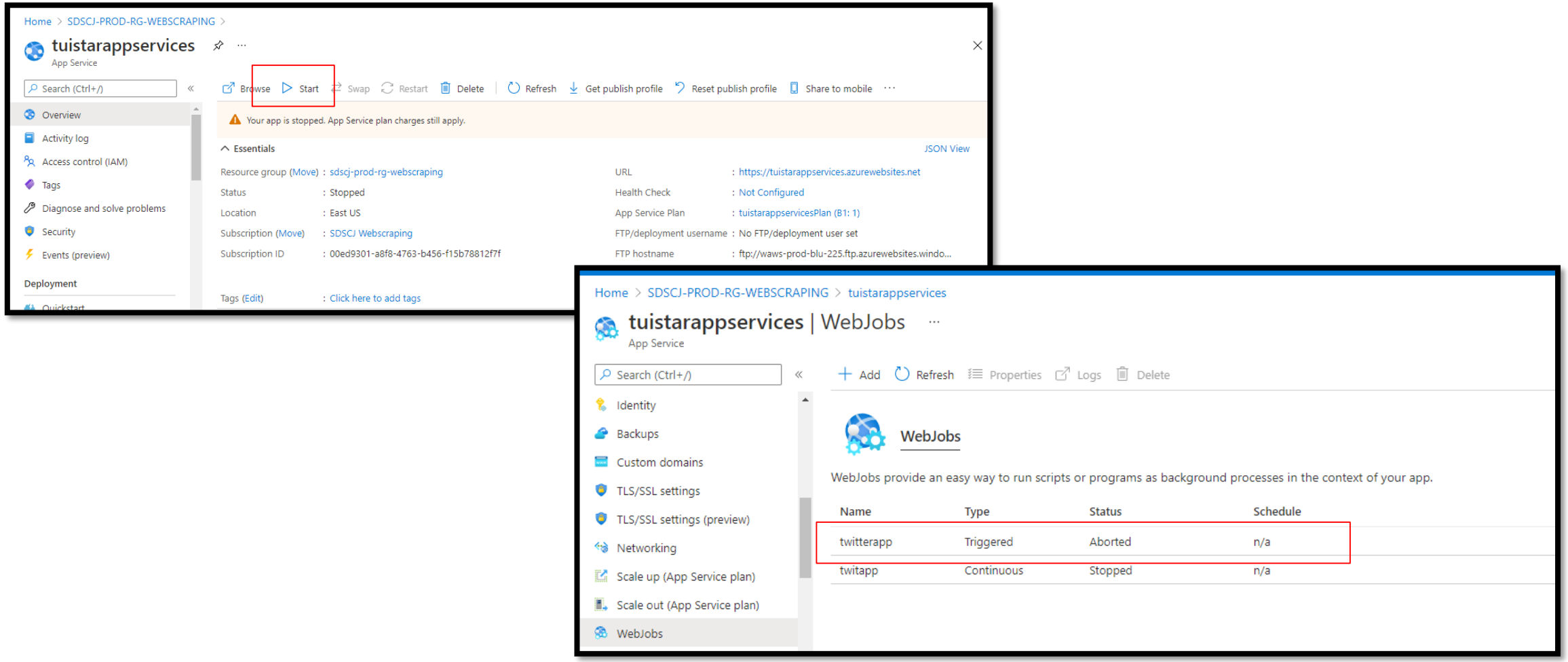




https://developer.twitter.com/en/docs/authentication/oauth-1-0a/authorizing-a-request

# Componentes de la Solución

## Despliegue App Services

Se ha desplegado la aplicación desarrollada en C# sobre el servcio de App service con el que cuenta Azure con el nombre “tuistarappservices” y dentro de esta se ejecuta un webJob con nombre “twitterapp” que puede ser programado en cron.

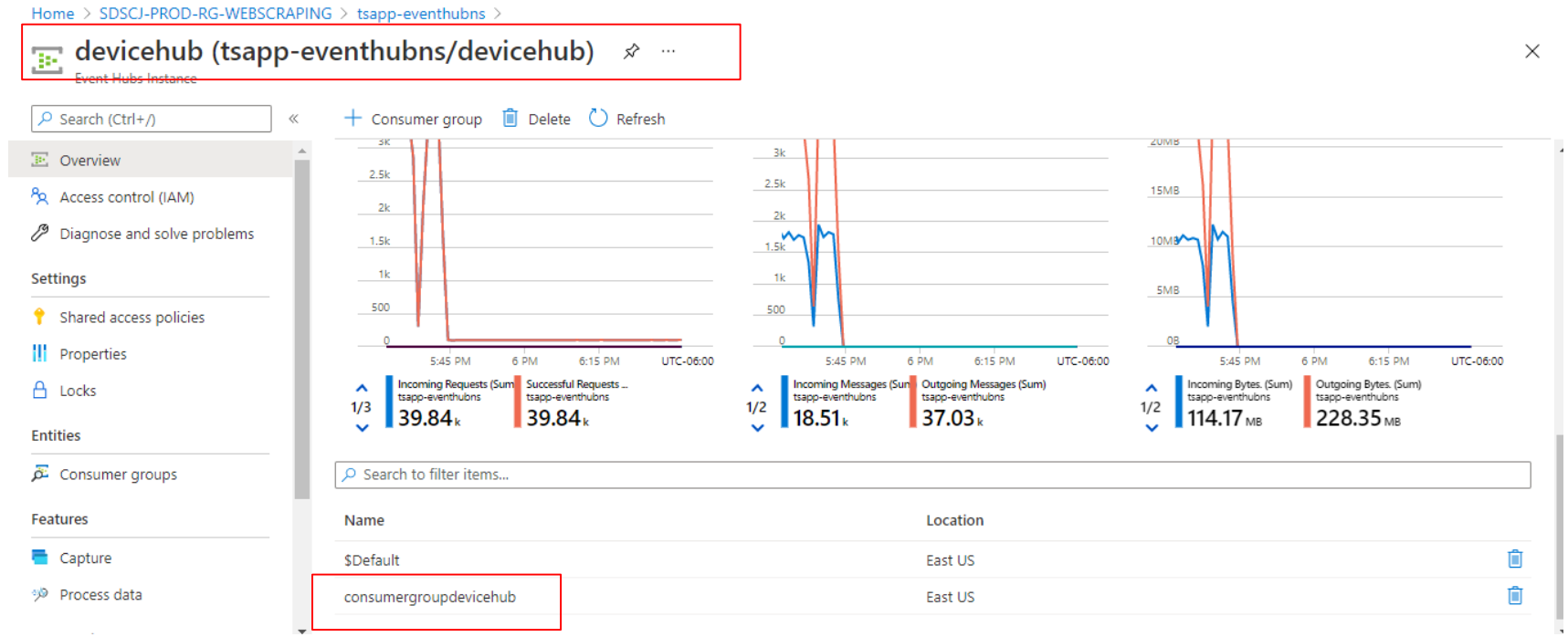



https://portal.azure.com/#@b096652d-23db-4751-aa7e-04b22767aec3/resource/subscriptions/00ed9301-a8f8-4763-b456-f15b78812f7f/resourceGroups/sdscj-prod-rg-webscraping/providers/microsoft.web/sites/tuistarappservices/webJobs

# Componentes de la Solución

## Despliegue Event Hub

Se ha configurado el componente de Event Hub para que reciba peticiones con un throuputh de 3 Unidades mínimas y un crecimiento máximo de hasta 10 unidad de consumo. Adicionalmente se cuenta con un namespace denominado como “tsapp-eventhubns” que es quien se encarga de almacenar los mensajes que se producen desde el app service a través del event hub con nombre “devicehub” y disponiblizarlos a través del grupo de consumo con nombre “consumergroupdevicehub”.



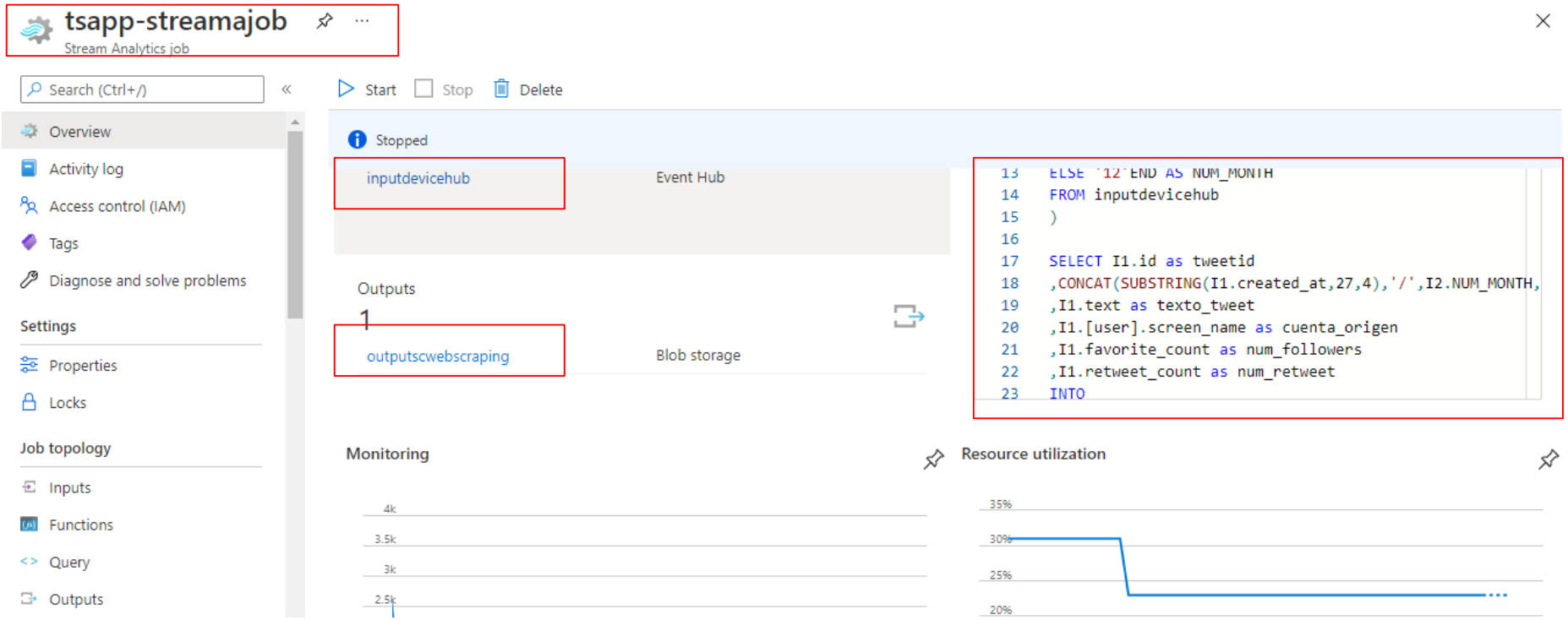

https://portal.azure.com/#@b096652d-23db-4751-aa7e-04b22767aec3/resource/subscriptions/00ed9301-a8f8-4763-b456-f15b78812f7f/resourceGroups/SDSCJ-PROD-RG-WEBSCRAPING/providers/Microsoft.EventHub/namespaces/tsapp-eventhubns/eventhubs/devicehub/overview

# Componentes de la Solución

## Despliegue Streaming Analytics

Se ha configurado el componente Azure Streaming Analytics que está recibiendo como input la extracción de los mensajes de Event Hub a través del consumer group denominado como “consumergroupdevicehub” a través del input llamado “inputdevicehub”. Como salida se ha configurado la salida con nombre “outputscwebscraping” que cuenta con una configuración que permite el envío de datos en formato “csv” hacia un storage con nombre “scwebscraping”, el cual almacena los datos en el archivo que se encuentre en ejecución dentro del contenedor “outputstreaming”.

Los campos que son extraídos por cada evento son transformados de acuerdo a lo establecido así:



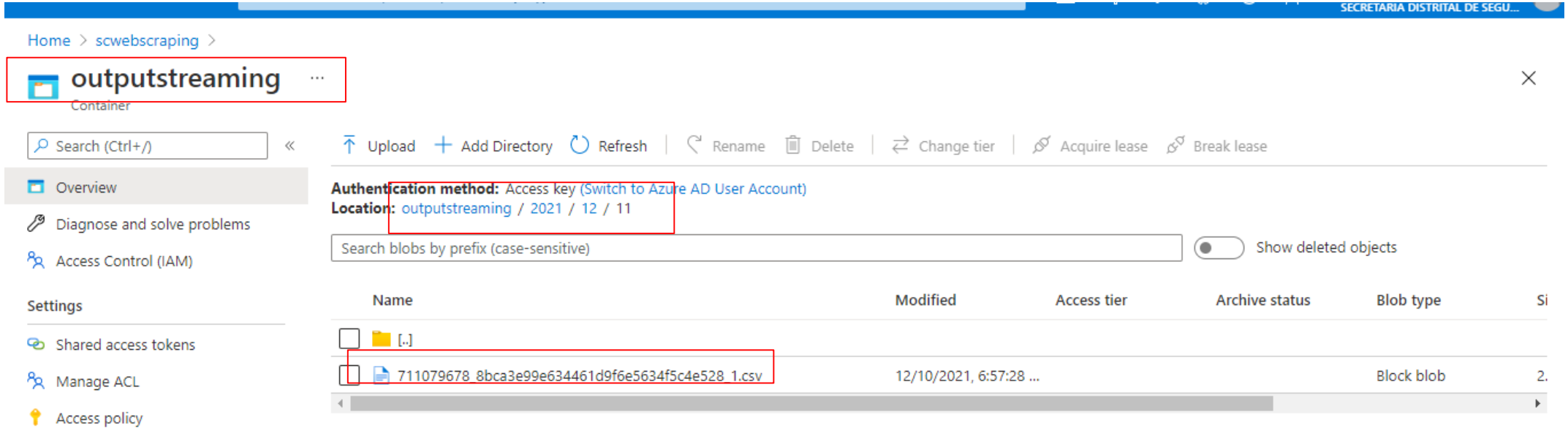

**Nota**: Se ha ajustado la salida del streamig para que guarde los datos bajo una estructura de carpetas YYYY/MM/DD

https://portal.azure.com/#@b096652d-23db-4751-aa7e-04b22767aec3/resource/subscriptions/00ed9301-a8f8-4763-b456-f15b78812f7f/resourceGroups/SDSCJ-PROD-RG-WEBSCRAPING/providers/Microsoft.EventHub/namespaces/tsapp-eventhubns/eventhubs/devicehub/overview

# Componentes de la Solución

## Despliegue Data Lake Gen 2

Se ha configurado el componente Data Lake Gen2 con una cuenta de almacenamiento llamada “scwebscraping” la cual guarda la información en el contenedor con nombre “outputstreaming” de cuerdo a los datos generados a través del proceso en streaming Anallytics.

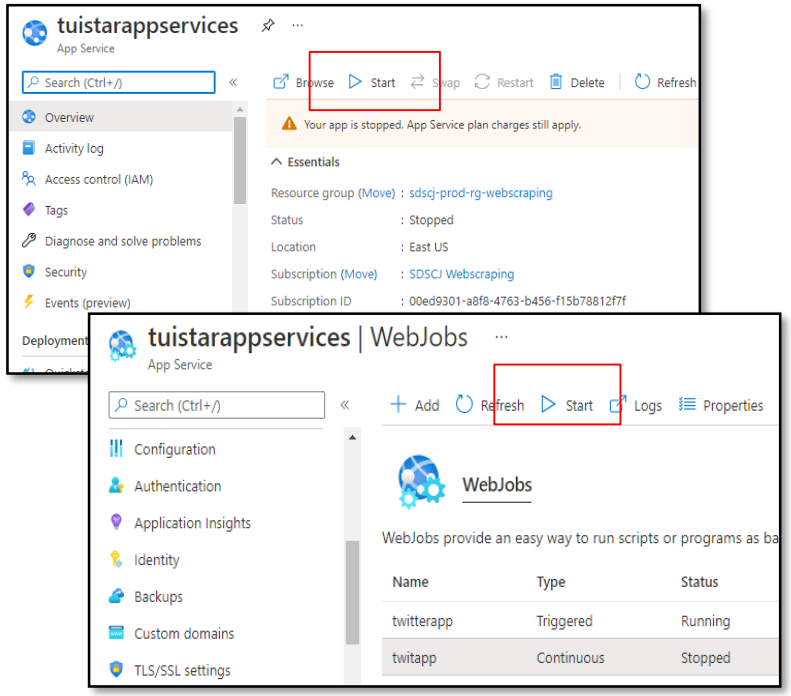



https://portal.azure.com/#blade/Microsoft_Azure_Storage/ContainerMenuBlade/overview/storageAccountId/%2Fsubscriptions%2F00ed9301-a8f8-4763-b456-f15b78812f7f%2FresourceGroups%2FSDSCJ-PROD-RG-WEBSCRAPING%2Fproviders%2FMicrosoft.Storage%2FstorageAccounts%2Fscwebscraping/path/outputstreaming/etag/%220x8D9BA5B071BABAD%22/defaultEncryptionScope/%24account-encryption-key/denyEncryptionScopeOverride//defaultId//publicAccessVal/None

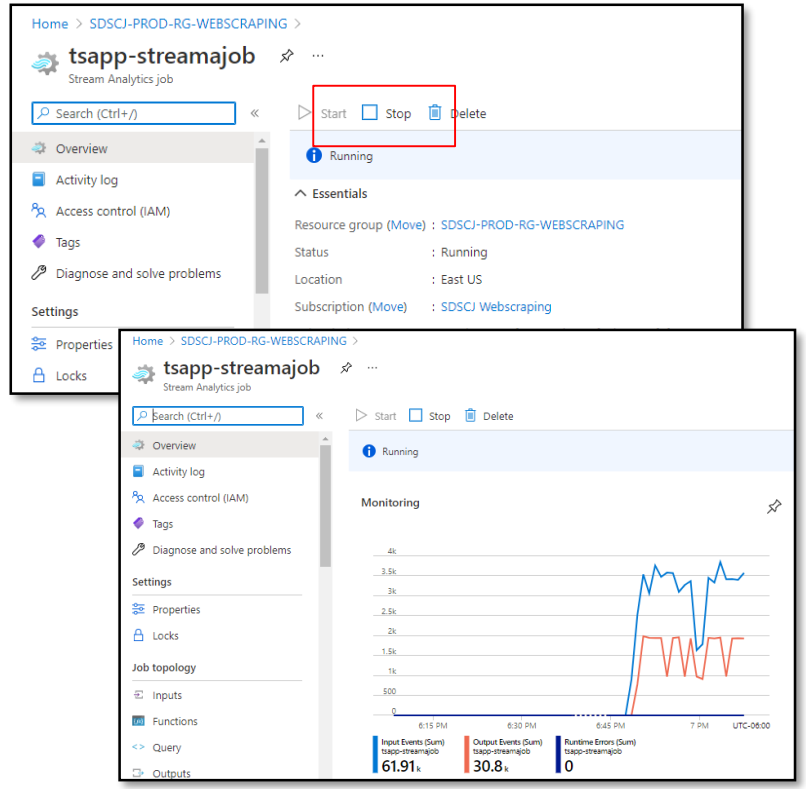
# Pasos de la Solución

Ejecución Solución

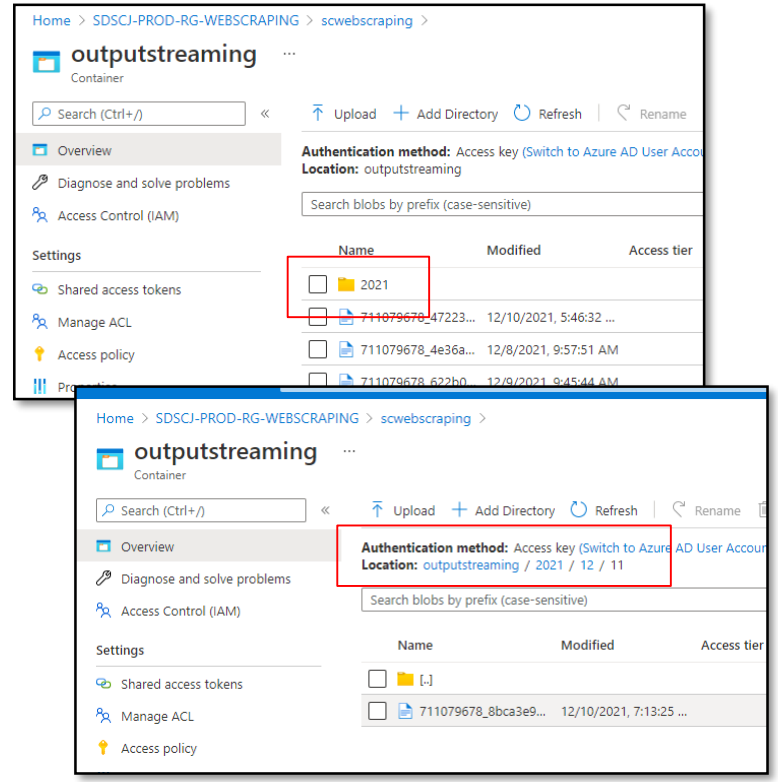
## 1 Inicio App Service




## 2 Inicio Streaming



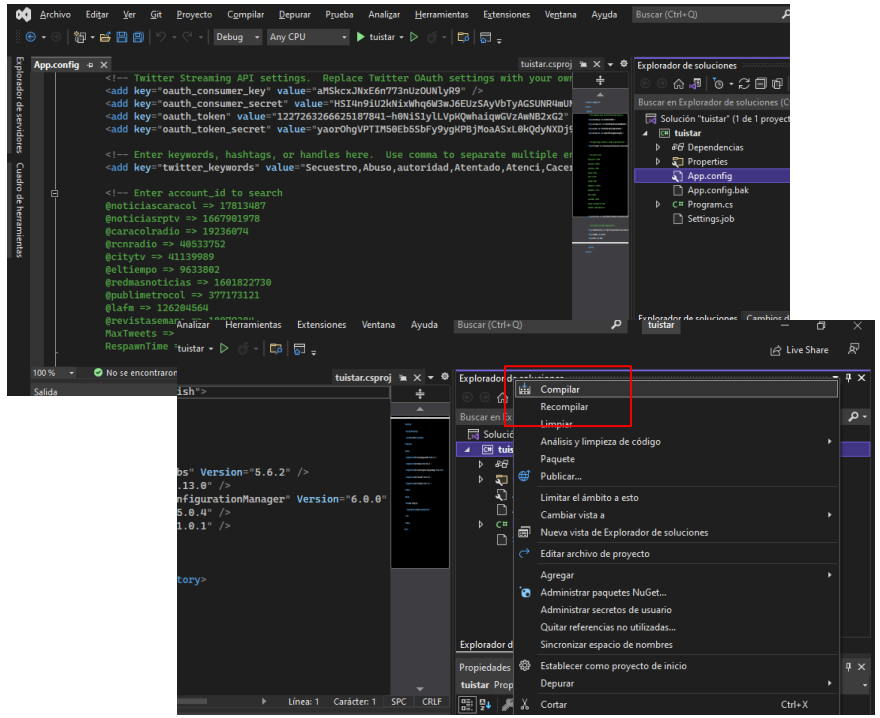

## 3 Validación Storage



# Pasos de la Solución

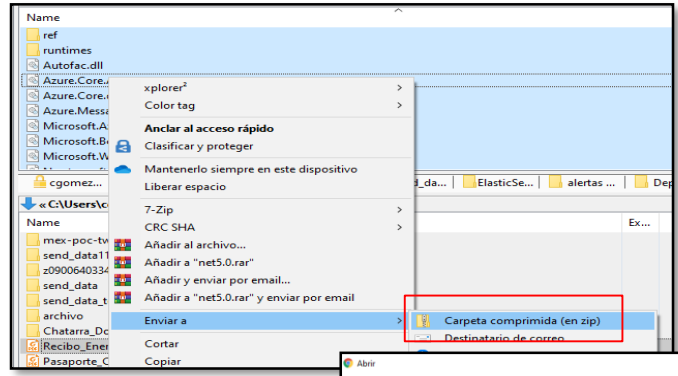
## Cambio de Palabras Tokens Variables

### 1 Cambio en Solución

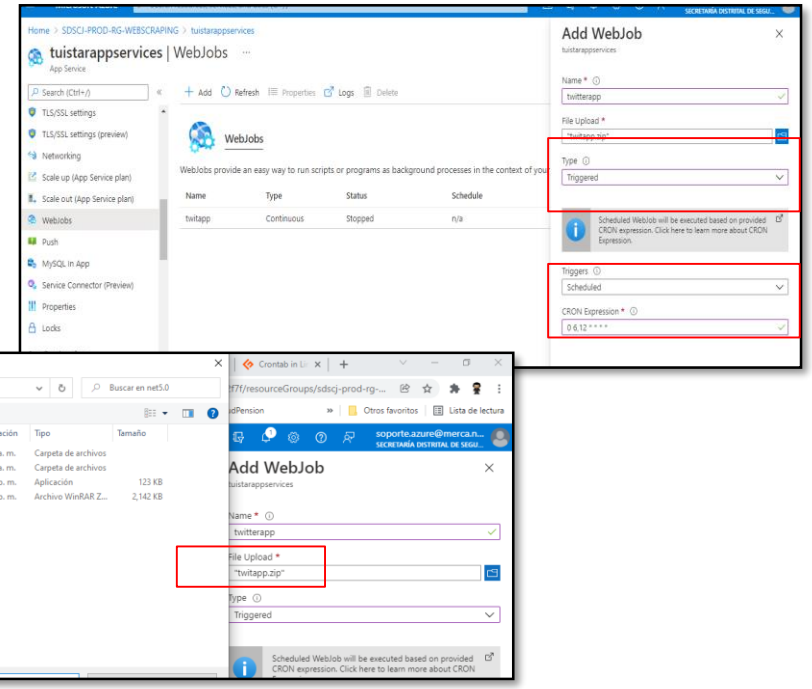



Se genera el cambio en la solución y se realizar una nueva compilación para que se generen los archivos necesarios en la ruta donde se encuentra la solución en la carpeta “bin\Debug\net5.0” para que sean comprimidos y desplegado en la nube.

### 2 Comprime y Sube Nube




### 3 Configuración WebJob




Se comprimen en zip los archivos que se encuentran en la ruta de la solución en la carpeta “bin\Debug\net5.0” y posteriormente dentro de la appservices en el apartado webjob se agrega como una nueva. En este apartado se puede definir si la app va a ser ejecutada a demanda “trigger” y si va hacer ejecutada en un periodo de tiempo programado. Ejemplo: Si se quiere ejecutar 1 una vez cada hora de 9 a. m. a 5 p. m. “0 0 9-17 * * *” o si se quiere ejecutar A las 9:30 a. m. todos los días “0 30 9 * * *” o a las 9:30 a. m. cada día de la semana “0 30 9 * * 1-5” {second} {minute} {hour} {day} {month} {day-of-week}.

https://docs.microsoft.com/es-mx/azure/azure-functions/functions-bindings-timer?tabs=csharp#ncrontab-expressions