



MANUAL

# ETL EN NUBE AWS - AZURE

UN NUEVO ENTORNO PARA NUESTROS DATOS

VALENTINA ARIZA - LAURA LÓPEZ

ETL es un proceso que nos permite poner a disposición los datos que están almacenados, extrayéndolos de múltiples fuentes y transformándolos en datos útiles para la limpieza, la transformación y, por último, la obtención de información.

En este manual se explica detalladamente como implementar la siguiente arquitectura de ETL en las nubes Azure y AWS (Fig. 1)

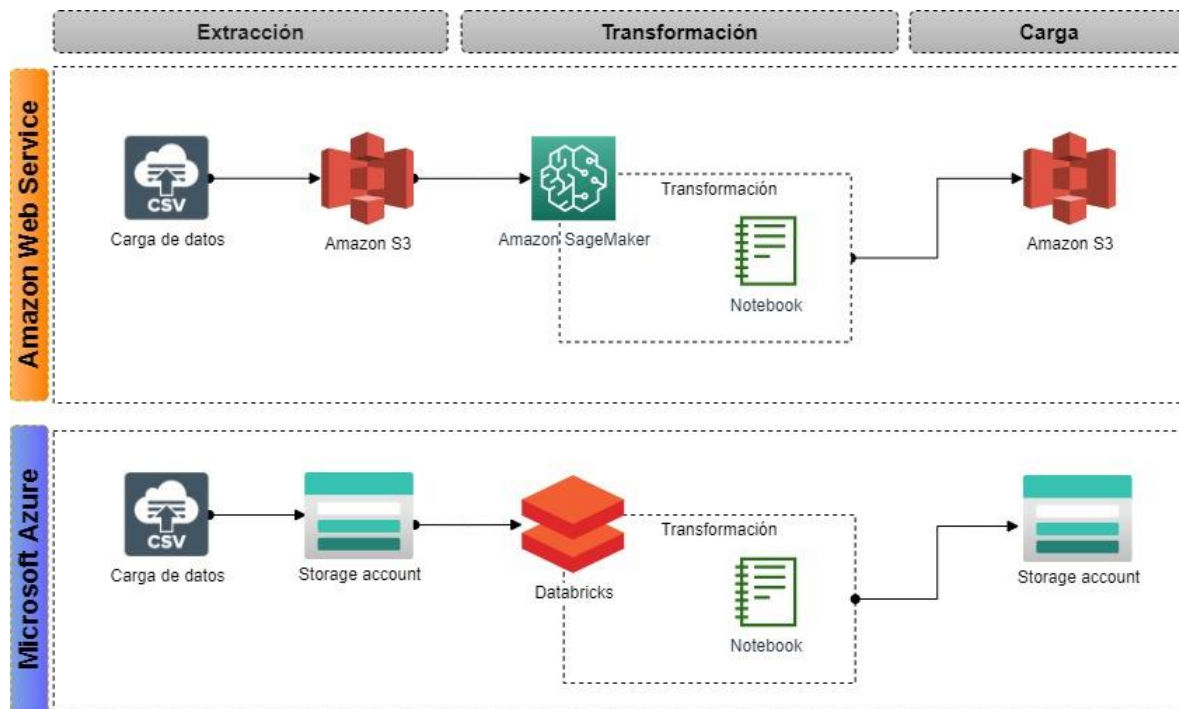


Fig. 1 Arquitectura ETL en nube.

Para este ejercicio práctico usaremos la base de datos de Billboard Hot 100 obtenida de [Kaggle](https://www.kaggle.com/datasets/andrewbriand/billboard-hot-100).

Billboard Hot 100 es la lista de discos estándar de la industria de la música en los Estados Unidos para canciones, publicada semanalmente por la revista Billboard. Las clasificaciones de las listas se basan en las ventas, la reproducción de radio y la transmisión en línea en los Estados Unidos.

Cada semana, Billboard publica la lista "The Hot 100" de canciones que fueron tendencia en ventas y difusión durante esa semana. Este conjunto de datos es una colección de todos los gráficos "The Hot 100" publicados desde su creación en 1958 hasta 2021.

# Implementación de ETL en AWS

## Extracción:

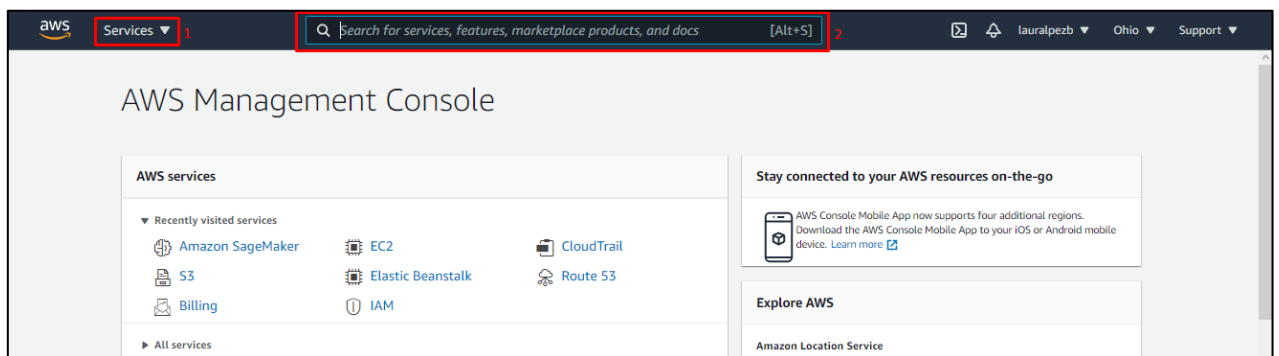
Para esta fase utilizamos el servicio S3 de AWS:

“Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento.”

Una vez ingresemos a nuestra cuenta de AWS podemos implementar esta fase de la siguiente manera:

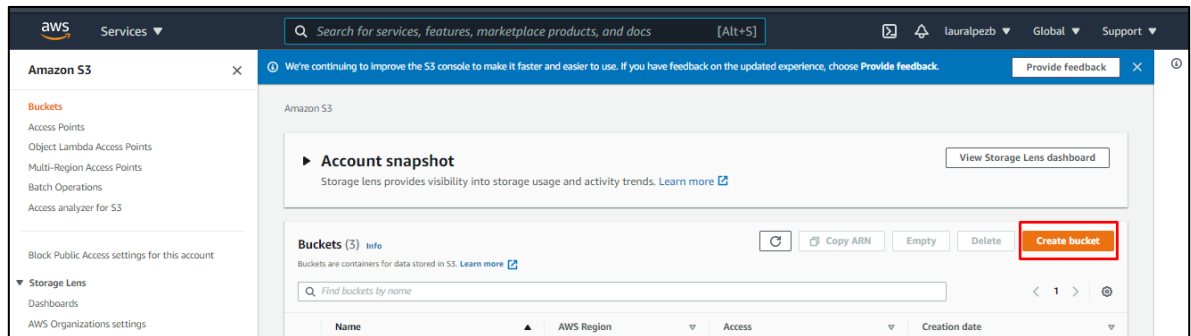
1. Buscamos el servicio S3 en AWS.

Tenemos dos alternativas para buscar el servicio que queremos utilizar:



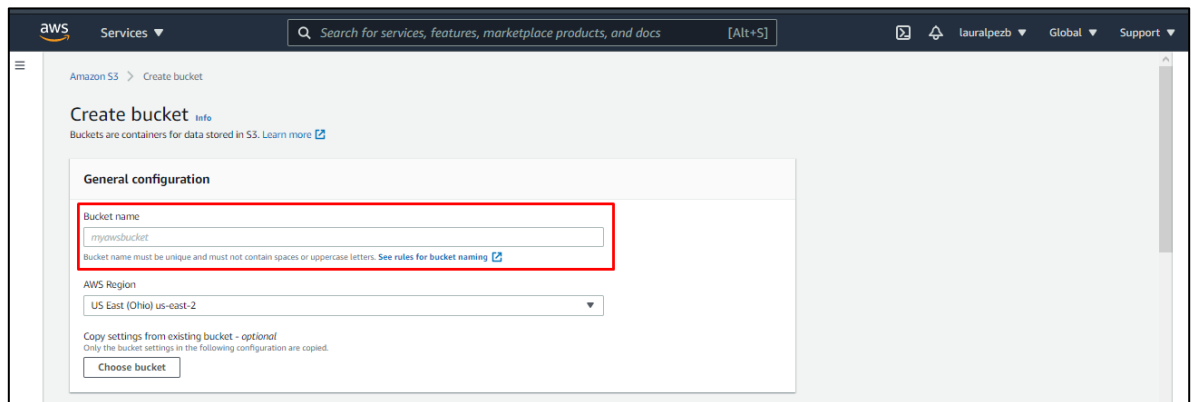
2. Una vez en el servicio seleccionamos *Create bucket* donde almacenaremos nuestros datos.

“Un bucket es un contenedor de objetos. Un objeto es un archivo y cualquier metadato que describa ese archivo.”



3. Configuramos nuestro bucket y para ello ingresamos el nombre que le queremos poner.

El nombre debe ser único y no debe contener espacio ni letras en mayúsculas.



En este paso también es importante que tengamos en cuenta la región donde crearemos el bucket. Trata de elegir una que este cerca de tu zona.

Las siguientes opciones las dejaremos con el valor por defecto.

4. Una vez tengamos las configuraciones de nuestro bucket, seleccionamos *Create bucket*.

☒ Disable  
☐ Enable

Tags (0) - optional

Track storage cost or other criteria by tagging your bucket. [Learn more](#)

No tags associated with this bucket.

Add tag

Default encryption

Automatically encrypt new objects stored in this bucket. [Learn more](#)

Server-side encryption  
☒ Disable  
☐ Enable

► Advanced settings

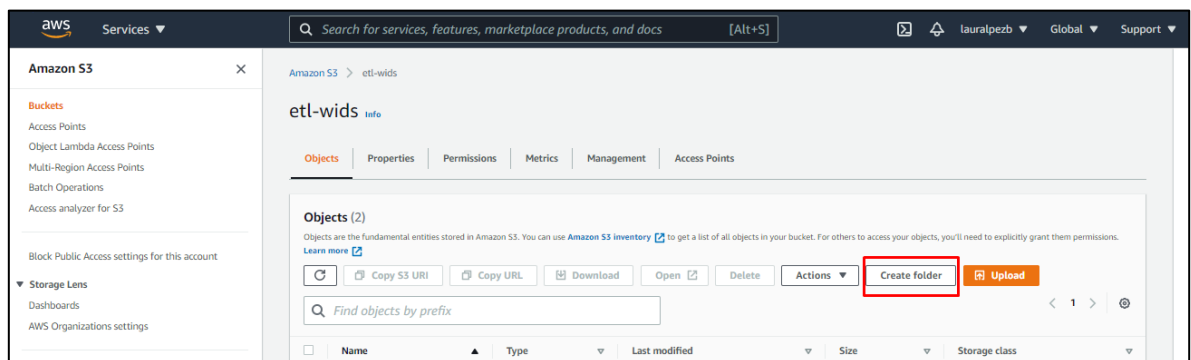
After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

CancelCreate bucket

*¡Listo! Hemos creado nuestro primer bucket.*

- Para tener un orden en nuestro bucket vamos a crear dos carpetas, una para los datos crudos y otra para almacenar los datos luego de la transformación.

Seleccionamos *Create folder*.



6. Agregamos el nombre de nuestra carpeta y seleccionamos al final *Create folder*.

Amazon S3 > etl-wids > Create folder

## Create folder Info

Use folders to group objects in buckets. When you create a folder, S3 creates an object using the name that you specify followed by a slash (/). This object then appears as folder on the console. [Learn more](#)

**Your bucket policy might block folder creation**

If your bucket policy prevents uploading objects without specific tags, metadata, or access control list (ACL) grantees, you will not be able to create a folder using this configuration. Instead, you can use the [upload configuration](#) to upload an empty folder and specify the appropriate settings.

### Folder

Folder name

Enter folder name /

Folder names can't contain "/". See [rules for naming](#)

### Server-side encryption

The following settings apply only to the new folder object and not to the objects contained within it.

Server-side encryption

☒ Disable

☐ Enable

Cancel **Create folder**

Recuerda que necesitamos dos carpetas para almacenar nuestros datos antes y después de la transformación.

Amazon S3 > etl-wids

## etl-wids Info

**Objects** | Properties | Permissions | Metrics | Management | Access Points

### Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

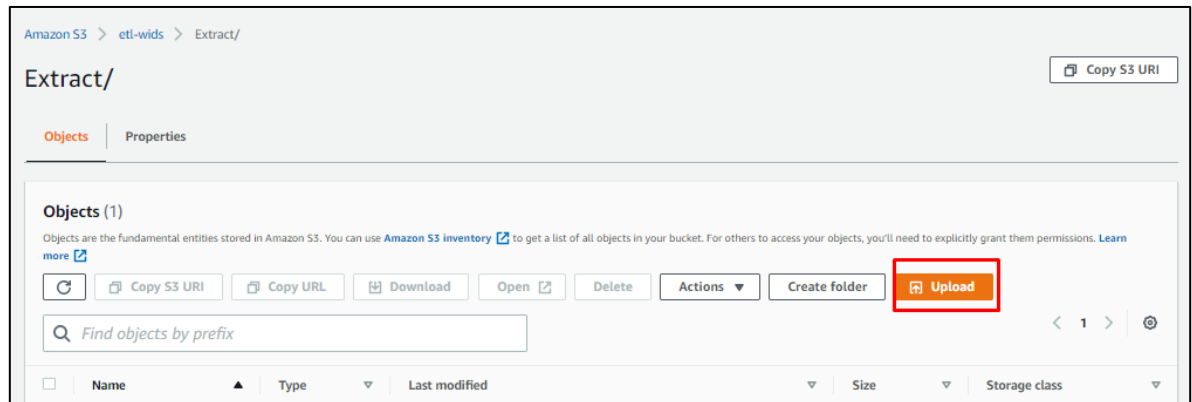
[Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

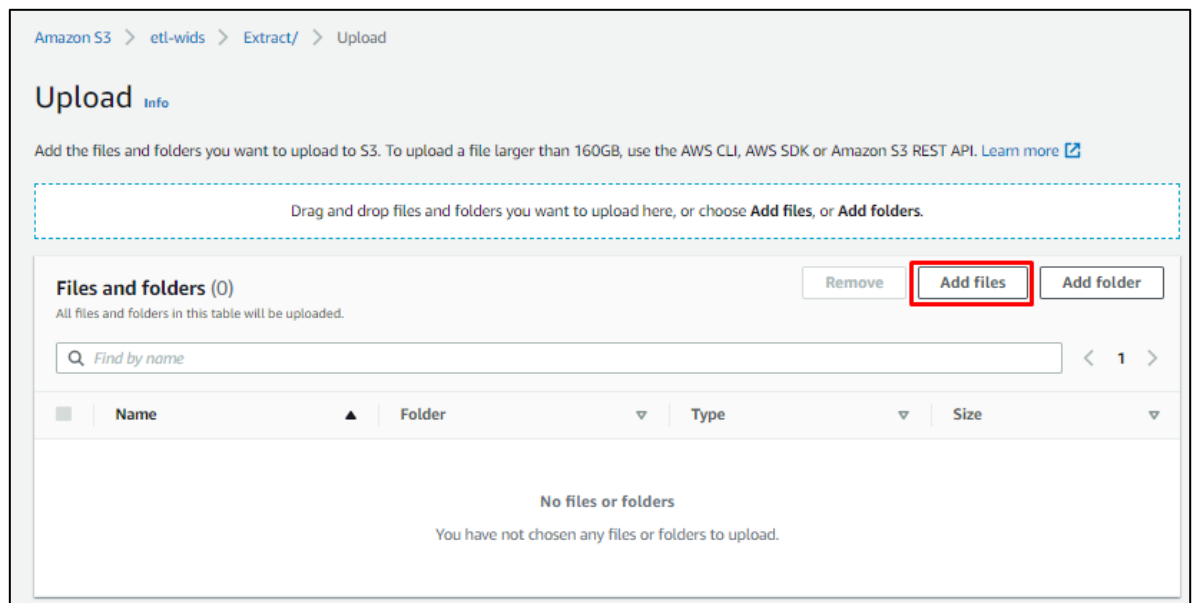
<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">Extract/</a>	Folder	-	-	-
<input type="checkbox"/>	<a href="#">Load/</a>	Folder	-	-	-

7. Ingresamos a la carpeta donde vamos a almacenar los datos crudos (*Extract*) y subimos el archivo con los datos.

Seleccionamos *Upload*.



8. Seleccionamos *Add files* y subimos nuestro archivo de datos.



9. Una vez subido el archivo seleccionamos *Upload* para almacenarlo en la carpeta.

**Files and folders** (1 Total, 17.3 MB)

All files and folders in this table will be uploaded.

Find by name

<input type="checkbox"/>	Name	Folder	Type	Size
<input type="checkbox"/>	charts.csv	-	application/vnd.ms-excel	17.3 MB

**Destination**

Destination

s3://etl-wids/Extract/

► **Destination details**

Bucket settings that impact new objects stored in the specified destination.

► **Permissions**

Grant public access and access to other AWS accounts.

► **Properties**

Specify storage class, encryption settings, tags, and more.

Cancel **Upload**

**¡Listo!** Tenemos nuestro archivo en S3.

Amazon S3 > etl-wids > Extract/

**Extract/**

Copy S3 URI

Objects Properties

**Objects (1)**

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	charts.csv	csv	September 4, 2021, 17:40:40 (UTC-05:00)	17.3 MB	Standard

**¡Listo!** Hemos terminado la primera fase de ETL.



## Transformación:

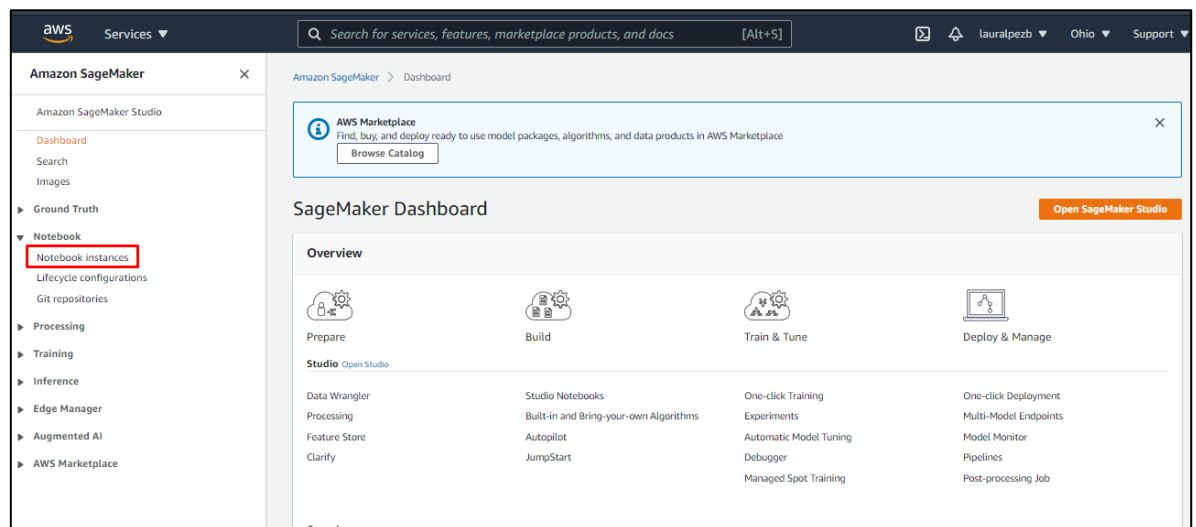
Para esta fase utilizamos el servicio SageMaker de AWS:

“Amazon SageMaker es un servicio de aprendizaje automático completamente administrado. Con SageMaker, los desarrolladores y los analistas de datos pueden crear y perfeccionar modelos de aprendizaje automático de forma rápida y sencilla y, a continuación, implementarlos directamente en un entorno alojado listo para su uso. Incluye una instancia de bloc de notas de creación de Jupyter integrada para obtener acceso de manera sencilla a sus orígenes de datos y poder realizar estudios y análisis sin tener que administrar servidores”.

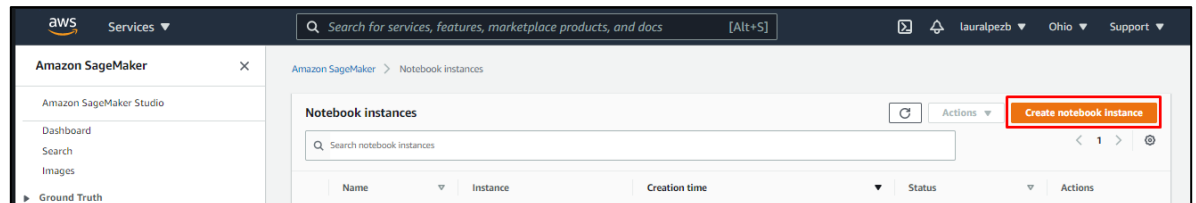
Luego de cargar nuestros datos buscamos el servicio SageMaker y realizamos la siguiente configuración:

1. Creamos una instancia notebook para integrar Jupyter notebooks.

Seleccionamos *Notebook instances*.



2. Seleccionamos *Create notebook instance*



3. Ingresamos el nombre que queremos ponerle a nuestra instancia. En las demás opciones podemos dejar el valor por defecto.

Amazon SageMaker > Notebook instances > Create notebook instance

## Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

### Notebook instance settings

Notebook instance name  
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type  
ml.t2.medium

Elastic Inference [Learn more](#)  
none

**Amazon SageMaker Notebook Instance is ending its standard support on Amazon Linux AMI (AL1). [Learn more](#)**

Platform identifier [Learn more](#)  
notebook-ml1-v1

► Additional configuration

4. En la sección de permisos creamos un nuevo rol IAM en *Create a new role*, para otorgar los permisos necesarios al usar la instancia.

### Permissions and encryption

**IAM role**  
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

Create a new role

Enter a custom IAM role ARN

Use existing role

Encryption key - optional  
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption

Para nuestro caso de ejemplo podemos dejar las opciones por defecto y seleccionamos *Create rol*.

**Create an IAM role**

Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role you create.

The IAM role you create will provide access to:

- ☒ S3 buckets you specify - optional
  - ☒ Any S3 bucket  
Allow users that have access to your notebook instance access to any bucket and its contents in your account.
  - ☐ Specific S3 buckets  
  
Comma delimited. ARNs, "\*" and "/" are not supported.
  - ☐ None
- ☒ Any S3 bucket with "sagemaker" in the name
- ☒ Any S3 object with "sagemaker" in the name
- ☒ Any S3 object with the tag "sagemaker" and value "true" [See Object tagging](#)
- ☒ S3 bucket with a Bucket Policy allowing access to SageMaker [See S3 bucket policies](#)

5. Finalmente seleccionamos *Create notebook instance*.

► **Network** - optional

► **Git repositories** - optional

► **Tags** - optional

**¡Listo! Tenemos nuestra instancia.**

6. Una vez creamos la instancia debemos iniciarla, para ello seleccionamos la instancia y en *Actions* seleccionamos *Start*.

Amazon SageMaker > Notebook instances

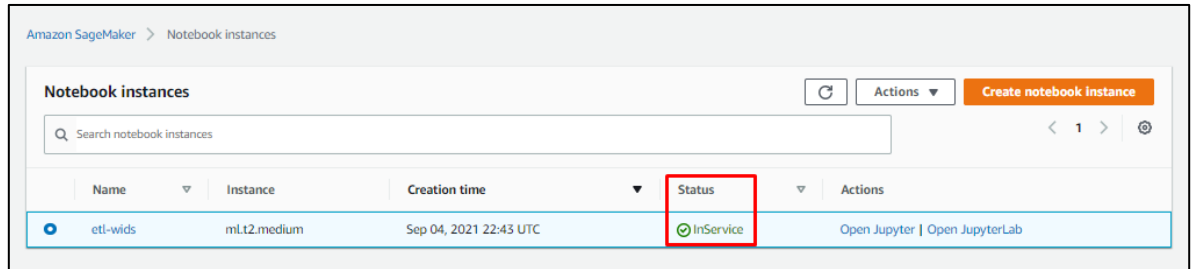
**Notebook instances**

Search notebook instances

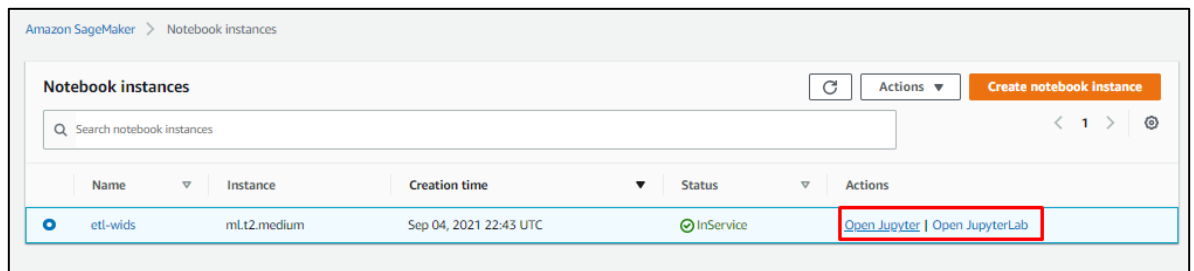
Name	Instance	Creation time	Status	Actions
etl-wids	ml.t2.medium	Sep 04, 2021 22:43 UTC	Stopped	<input checked="" type="radio"/> <b>Start</b> Open Jupyter Open JupyterLab Stop Update settings Add/Edit tags Delete

Este proceso puede tardar unos segundos.

Para verificar que la instancia está iniciada validamos el campo *Status*, debe estar en *InService*.

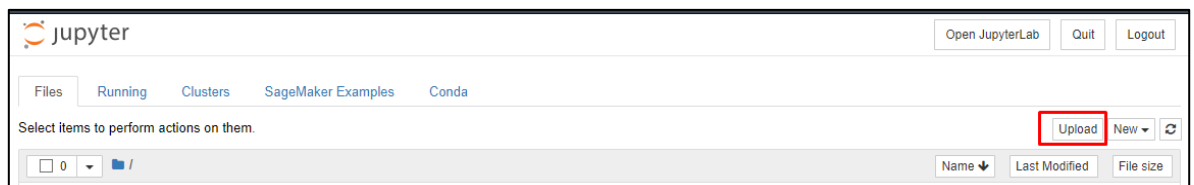


7. Ingresamos a la instancia usando Jupyter o JupyterLab.

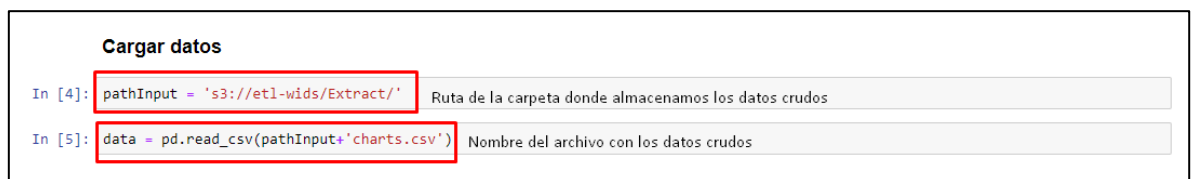


En esta ocasión usaremos Jupyter.

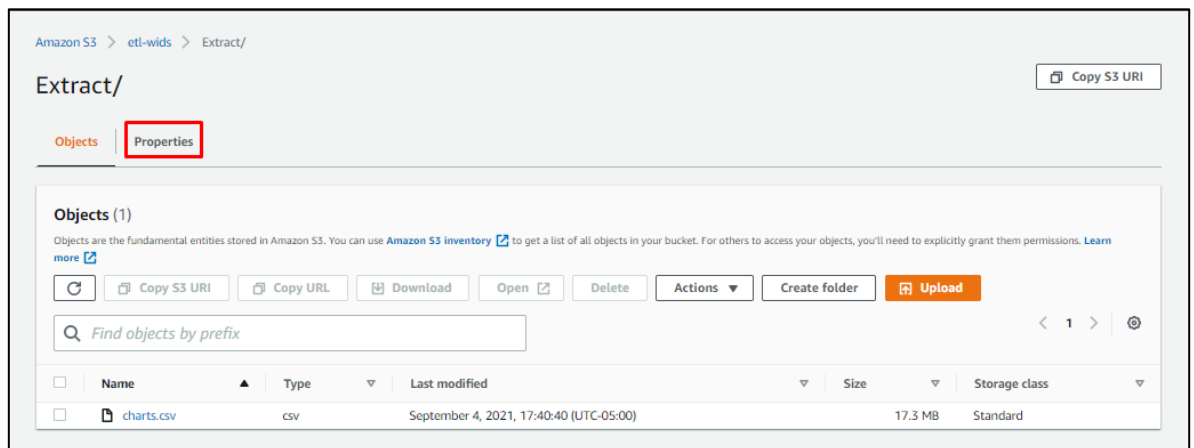
8. Cargamos el Jupyter notebook *transform.ipynb* que podemos descargar desde el repositorio de GitHub.



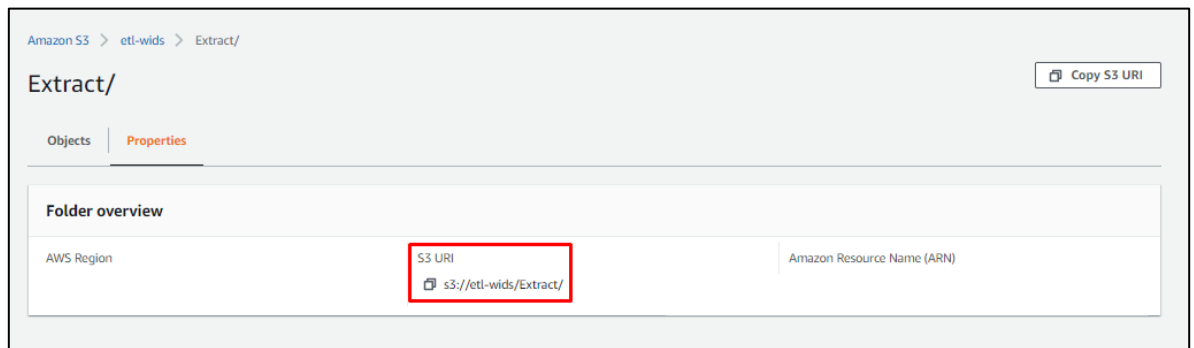
9. Ingresamos al notebook y validamos que las rutas para leer y almacenar nuestros datos están correctas.



Para validar la ruta podemos ir a S3, ingresar al bucket que creamos y a la carpeta donde se encuentra el archivo. Estando en esa ubicación seleccionamos *Properties*.



Copiamos la ruta que aparece en la sección *S3 URL* y esa será la ruta que pondremos en el notebook para leer el archivo.



- Corremos el notebook (hasta la línea antes de guardar datos) para realizar las transformaciones correspondientes a nuestros datos (cada parte del notebook cuenta con su explicación).



**¡Listo!** Hemos terminado la segunda fase de ETL.

## Carga:

Para esta fase utilizamos el servicio S3 de AWS:

Una vez realizamos las transformaciones a nuestros datos vamos a almacenar los datos procesados para que podamos usarlos en otras fases del proyecto como análisis o implementación de modelos.

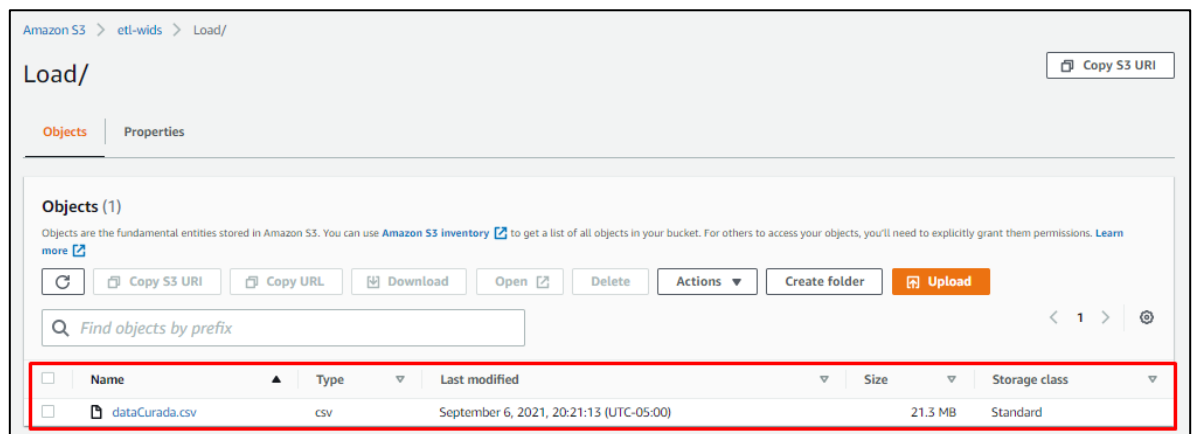
1. Guardamos nuestros datos procesados en la ruta de la carpeta que creamos en S3.

```
Guardar datos

In [31]: pathOutput = 's3://etl-wids/Load/'

In [32]: data.to_csv(pathOutput+'dataCurada.csv')
```

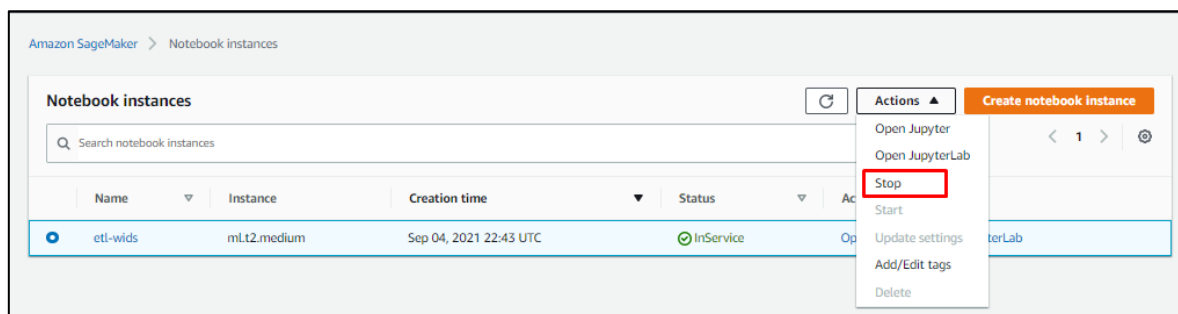
2. Regresamos a S3 para validar que los datos están almacenados.



**¡Listo!** Hemos realizado todos los pasos de ETL en AWS.

## Recomendación 💡

Recuerda apagar tu instancia notebook si no la vas a usar. Para ello puedes seleccionar la instancia y en *Actions* seleccionar *Stop*.



Si te interesa conocer más sobre este tema puedes visitar los siguientes enlaces:

- AWS S3:  
<https://aws.amazon.com/es/s3/>  
[https://docs.aws.amazon.com/es\\_es/AmazonS3/latest/userguide/creating-bucket.html](https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide/creating-bucket.html)
- AWS SageMaker:  
[https://docs.aws.amazon.com/es\\_es/sagemaker/latest/dg/gs-setup-working-env.html](https://docs.aws.amazon.com/es_es/sagemaker/latest/dg/gs-setup-working-env.html)
- Esta no es la única forma para implementar ETL en AWS, existen otras alternativas y servicios que te permitirán hacer este proceso, como por ejemplo AWS Glue. Puedes encontrar más información en el siguiente enlace:  
<https://aws.amazon.com/es/glue/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc>

# Implementación de ETL en Azure

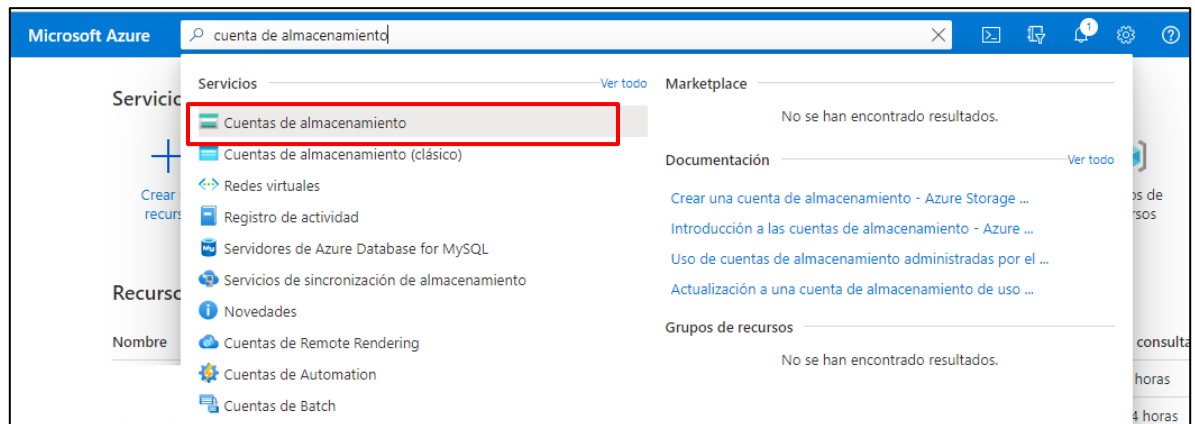
## Extracción:

Para esta fase usamos el servicio Storage Account de Azure, específicamente Azure Data Lake Storage Gen2:

“Es un servicio dedicado al análisis de macrodatos basado en Azure Blob Storage, admite los tipos de cuentas de almacenamiento: De uso general estándar, v2 y Blobs en bloques Premium. Como estas funcionalidades se basan en Blob Storage, también disfrutará de un almacenamiento por niveles de bajo costo, con funcionalidades de alta disponibilidad y recuperación ante desastres”.

Una vez ingresemos al Portal de Azure, podemos implementar esta fase de la siguiente manera:

1. Buscamos el servicio *cuentas de almacenamiento*:

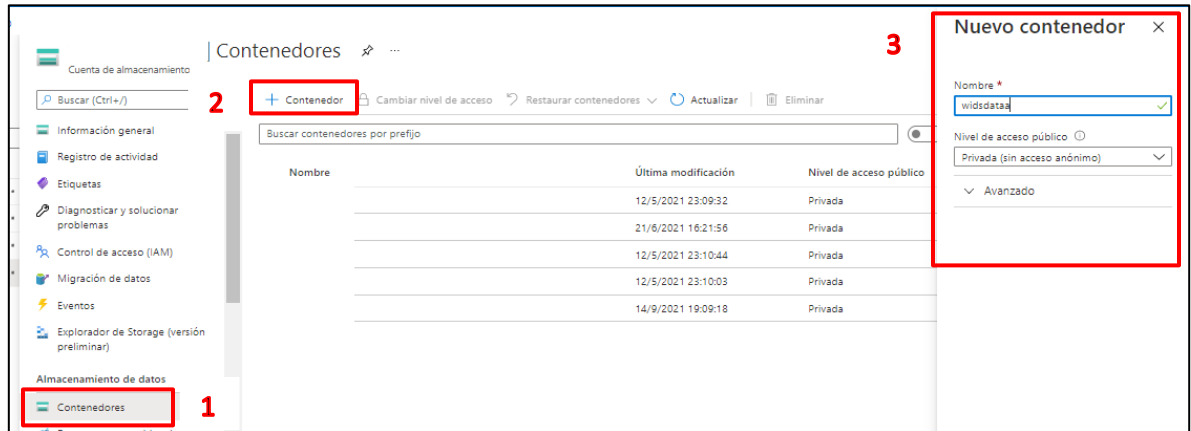




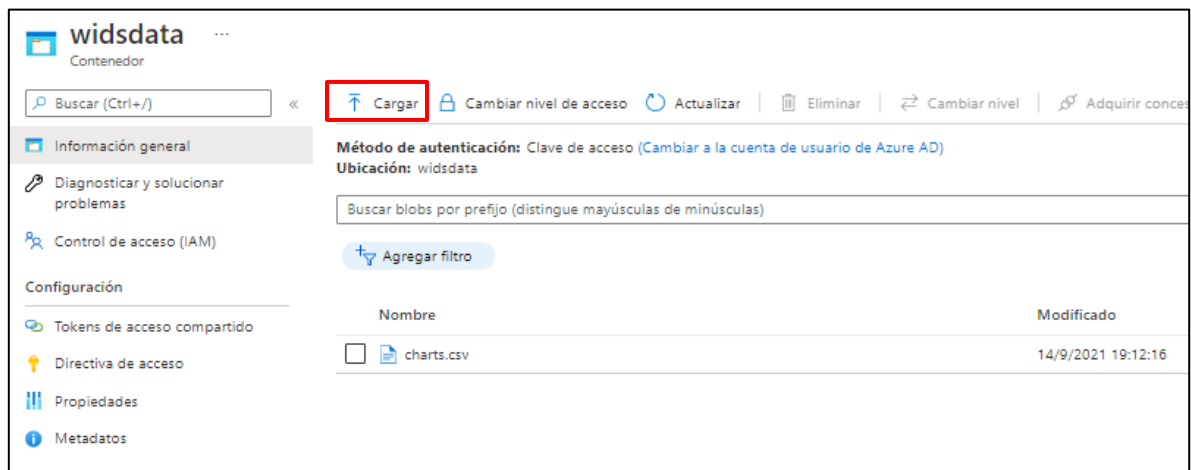
2. Una vez dentro del servicio, seleccionamos Crear

Se nos presentará la siguiente ventana en la cual configuraremos la cuenta de almacenamiento como un Data Lake Storage Gen2. En los datos básicos se debe escoger la suscripción que estamos usando, el grupo de recursos (en caso de no tener uno, seleccionamos la opción de crear nuevo) y especificar un nombre para la cuenta. Verificar que en *Rendimiento* se escoja el **Estándar**, que se refiere al uso general v2.

Las siguientes opciones se dejarán con el valor por defecto, y seleccionamos *Revisar* y *crear*. Cuando haya finalizado el proceso de creación, ingresamos al servicio y escogemos la pestaña *contenedores* (1) y luego agregar contenedor (2), esto abrirá una ventana que permitirá especificar el nombre del contenedor.



Seguidamente ingresamos al contenedor creado y mediante la opción *cargar*, cargamos nuestros datos:



Adicionalmente, regresamos a la cuenta de almacenamiento y seleccionamos Claves de acceso en *Configuración*. Copiamos el nombre de la cuenta de almacenamiento y la key1 en un editor de texto para usarlos más adelante.

**¡Listo!** Tenemos nuestros datos cargados en un Data Lake Gen2, con este paso finalizamos la primera fase de nuestra ETL, extracción.

## Transformación:

Para esta fase usamos el servicio *Azure Databricks* disponible en Azure:

“Azure Databricks es una plataforma de análisis de datos optimizada, ofrece tres entornos para desarrollar aplicaciones que consumen muchos datos: Databricks SQL, Databricks Data Science & Engineering y Databricks Machine Learning”.

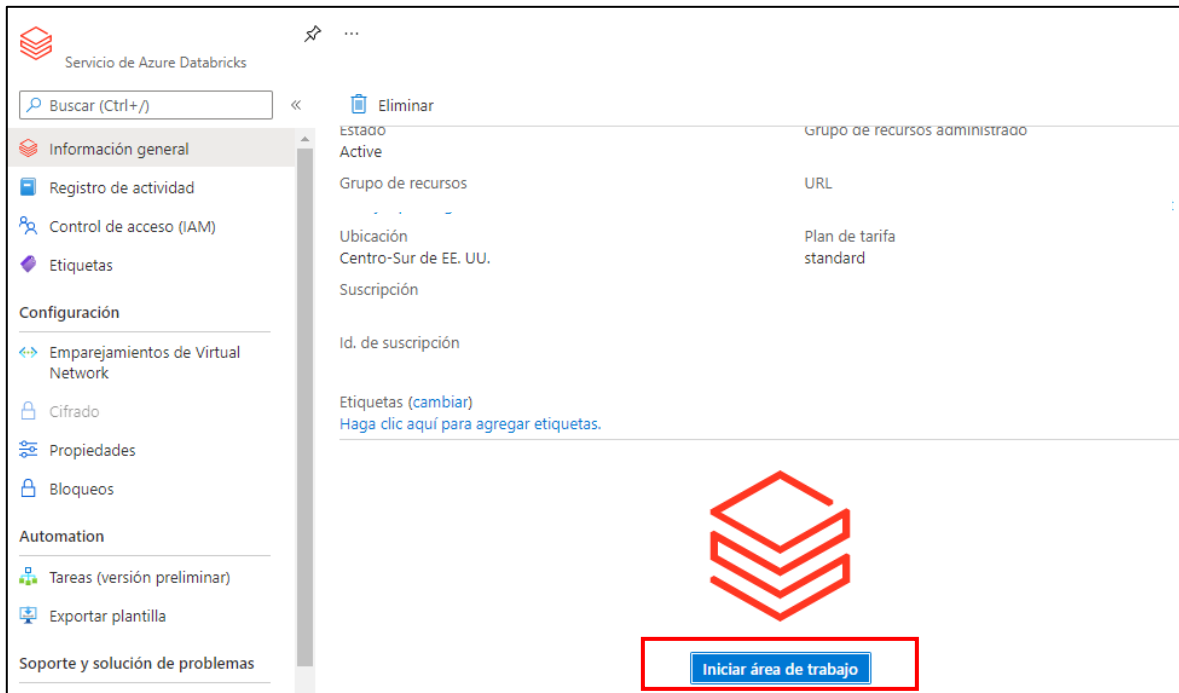
Continuando en el Portal de Azure, buscamos el servicio Azure Databricks, igual a cómo se buscó la cuenta de almacenamiento en la fase de extracción. Allí, seleccionamos *Crear*:

The screenshot shows the Azure Databricks portal interface. On the left, the 'Crear' button is highlighted with a red box. The main content area is titled 'Creación de un área de trabajo de Azure Databricks'. It features a tabbed interface with 'Datos básicos' selected. Below the tabs, there's a section 'Detalles del proyecto' with instructions to select a subscription and resource group. The 'Suscripción' and 'Grupo de recursos' fields are dropdown menus. Below them is a 'Crear nuevo' link. The 'Detalles de instancia' section includes fields for 'Nombre del área de trabajo', 'Región', and 'Plan de tarifa', each with a dropdown menu. At the bottom, there are navigation buttons: 'Revisar y crear', '< Anterior', and 'Siguiente: Redes >'. The left sidebar also includes a search bar and a list of items.

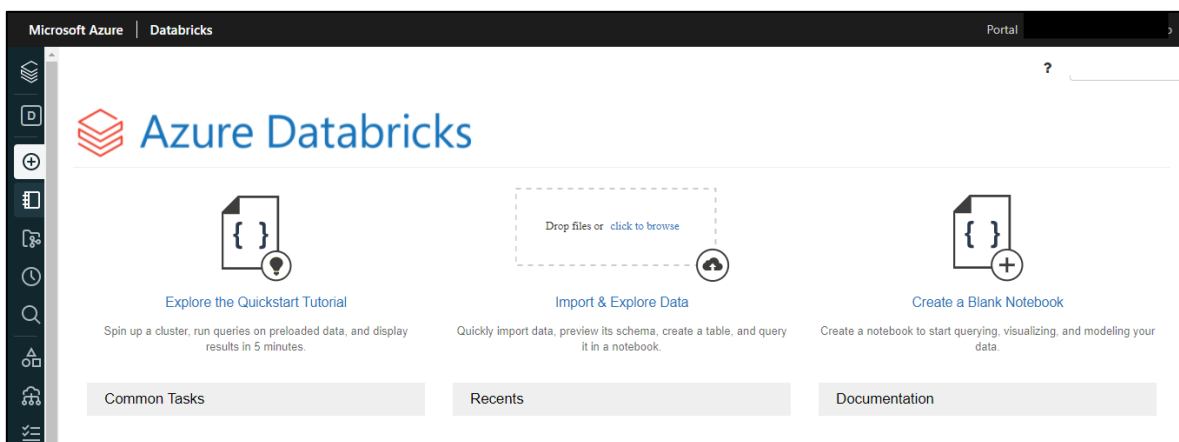
Allí proporcionaremos los siguientes valores para crear el servicio:

- **Nombre del área de trabajo:** nombre para el servicio de Databricks, ejemplo: dev-general-dbricks.
- **Suscripción:** Seleccionamos la suscripción utilizada, para esta implementación.
- **Grupo de recursos:** Seleccionamos el grupo de recursos en el cual se almacenará el servicio, en caso de no tener uno, seleccionamos la opción de crear nuevo.
- **Plan de tarifa:** Elegimos Estándar.

Los demás campos quedan con los valores por defecto. Una vez creado, nos dirigimos al servicio y escogemos la opción de *Iniciar área de trabajo*.



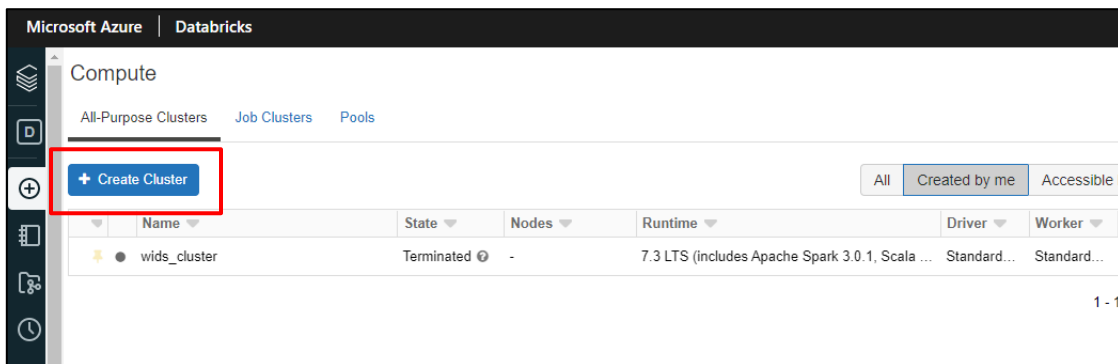
Esto nos llevará a la herramienta Databricks, aquí es donde realizaremos el proceso de transformación de los datos, en términos de seguir construyendo la arquitectura, solo cargaremos el notebook *transform.ipynb*.



Para lograr esto, primero debemos crear un cluster (en caso de tener uno funcionando, omitimos este paso).

“Un clúster de Azure Databricks es un conjunto de configuraciones y recursos informáticos en los que se ejecutan cargas de trabajo de ingeniería de datos, ciencia de datos y análisis de datos, como por ejemplo pipelines ETL”.

El cluster se crea en la pestaña *Compute* de Azure databricks, seleccionamos *Crear Cluster*.



Seguidamente, proporcionamos un nombre para nuestro cluster y escogemos la versión del Runtime 7.3 LTS y los demás campos se pueden dejar por defecto.

### Create Cluster

Cluster Name  
wids\_cluster

Cluster Mode ?  
Standard

Databricks Runtime Version ? [Learn more](#)  
Runtime: 7.3 LTS (Scala 2.12, Spark 3.0.1)

Autopilot Options  
☒ Enable autoscaling ?  
☒ Terminate after 120 minutes of inactivity ?

Worker Type ?  
Standard\_DS3\_v2 14 GB Memory, 4 Cores

Min Workers: 2 Max Workers: 8

☐ Spot instances ?

**New** Configure separate pools for workers and drivers for flexibility. [Learn more](#)

Una vez tenemos nuestro clúster, debemos acceder al data lake gen2 desde databricks, para poder transformar los datos almacenados allí. Para esto usaremos *secrets* (secretos) guardados en Azure Key vault.

Key Vault protege las claves criptográficas y otros secretos usados por las aplicaciones y los servicios en la nube

Continuando en el Portal de Azure, buscamos el servicio Azure key vault. Allí, seleccionamos *Crear*:

The screenshot shows the 'Crear almacén de claves' (Create Key Vault) page in the Azure portal. The form includes the following sections and fields:

- Suscripción \***: A dropdown menu.
- Grupo de recursos \***: A dropdown menu with a [Crear nuevo](#) link below it.
- Detalles de instancia**:
  - Nombre del almacén de claves \*** ⓘ: A text input field with the placeholder 'Escriba el nombre'.
  - Región \***: A dropdown menu showing 'Centro-Sur de EE. UU.'.
  - Plan de tarifa \*** ⓘ: A dropdown menu showing 'Estándar'.
- Opciones de recuperación**: A text block explaining that temporal deletion protection is enabled by default, allowing recovery or permanent deletion of the vault and secrets during the retention period.
- Footer**: A note stating 'Para aplicar un período de retención obligatorio y evitar la eliminación permanente de los almacenes de claves o los secretos' (To apply a mandatory retention period and avoid the permanent deletion of key vaults or secrets).
- Navigation**: Three buttons at the bottom: 'Revisar y crear' (blue), '< anterior' (grey), and 'Siguiente: Directiva de acceso >' (white with grey border).

Además de la misma suscripción y grupo de recursos que se han usado en la creación de los anteriores servicios, proporcionaremos los siguientes valores para crear el servicio:

- **Nombre:** un nombre único para el key-vault, ejemplo: wids-stg-kv.
- **Plan de tarifa:** Elegimos Estándar.

Seleccionamos revisar y crear. Luego, navegamos hasta el key vault recién creado en Azure Portal y seleccionamos *Secretos*. Después, seleccionamos **+ Generar / Importar**.

[Inicio](#) > [Almacenes de claves](#) > [videosrem-kv](#) >

## Crear un secreto ...

Opciones de carga	<div>Manual</div>
Nombre * ⓘ	<input type="text"/>
Valor * ⓘ	<input type="text" value="Escriba el secreto."/>
Tipo de contenido (opcional)	<input type="text"/>
Establecer la fecha de activación ⓘ	<input type="checkbox"/>
Establecer fecha de expiración ⓘ	<input type="checkbox"/>
Habilitado	<div><div>Sí</div><div>No</div></div>
Etiquetas	0 etiquetas

[Crear](#)

Allí proporcionaremos lo siguiente:

- **Opciones de carga:** Manual
- **Nombre:** Nombre descriptivo para la clave de su cuenta de almacenamiento.
- **Valor:** key1 que guardamos de la cuenta de almacenamiento.

Guardamos el nombre de la clave en un editor de texto para usarlo más adelante y seleccionamos **Crear**. Luego, navegamos hasta el menú *Propiedades* y copiamos el nombre DNS y el ID de recurso en un editor de texto para usarlos más adelante.

*De esta manera, la clave de acceso a nuestro data lake quedó guardada de forma segura en un key vault, y podremos usarla sin especificarla directamente.*

Ahora solo queda disponer esta clave de acceso en databricks, para eso crearemos un *secret scope* en nuestro databricks. Realizaremos lo siguiente:

- En la página de inicio de nuestro databricks, al URL le añadimos `secrets/createScope`. Esto abrirá la siguiente ventana.

← → ↻ 🔒 #secrets/createScope

Aplicaciones Gmail YouTube Traducir

Microsoft Azure | Databricks

HomePage / Create Secret Scope

## Create Secret Scope

Cancel Create

A store for secrets that is identified by a name and backed by a specific store type. [Learn more](#)

Scope Name ?

Manage Principal ?

Creator

### Azure Key Vault ?

DNS Name

https://xxx.vault.azure.net/

Resource ID

/subscriptions/xxxxxx/...

Allí, añadimos lo siguiente:

**Nombre:** nombre para el scope (guardarlo para usarlo más adelante).

**Manage Principal:** All users

**DNS:** nombre DNS de Azure Key Vault que guardamos anteriormente.

**ID de recursos:** ID de recurso de Azure Key Vault que guardamos anteriormente.

Por últimos, ejecutamos las siguientes líneas en un notebook

```
dbutils.fs.mount(  
  source = "wasbs://<nombre-contenedor>@<nombre de la cuenta de  
almacenamiento>.blob.core.windows.net",  
  mount_point = "/mnt/<nombre del blob>",  
  extra_configs = {"<conf-key>":dbutils.secrets.get(scope = "<nombre del  
scope en databricks>", key = "nombre de la key")})
```

**nombre del contenedor:** nombre del contenedor que creó dentro de la cuenta de almacenamiento.



**nombre de la cuenta de almacenamiento:** nombre con el cual creamos nuestra cuenta de almacenamiento.

**nombre del blob:** nombre del blob que se creó dentro del contenedor y donde se cargaron los datos.

**nombre del scope en databricks:** nombre del secret scope que se creó en databricks.

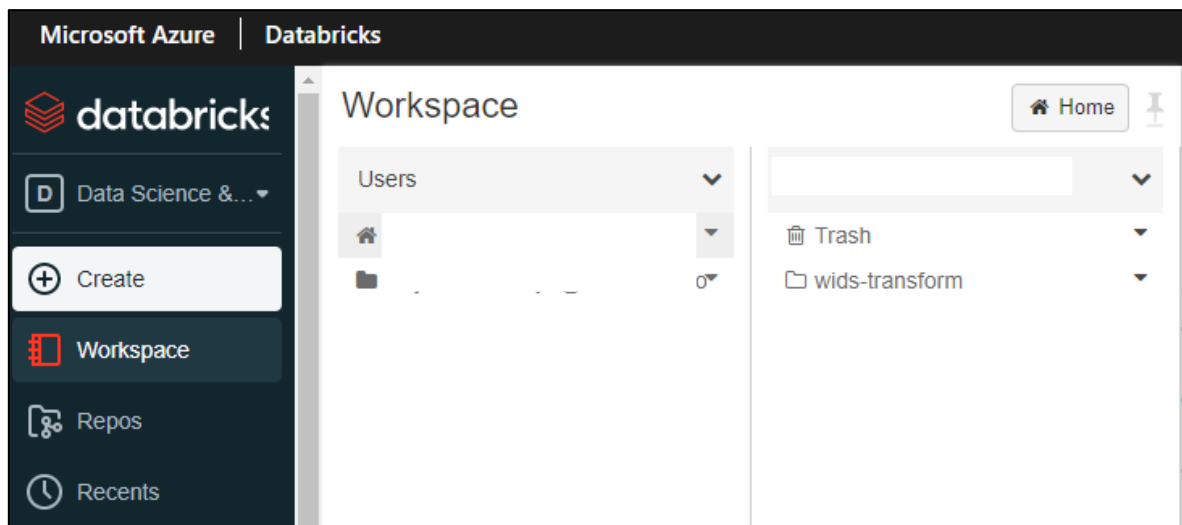
**nombre de la key:** nombre del secret creado en key vault, donde quedó guardada la llave de acceso a la cuenta de almacenamiento

**conf-key:** puede ser `fs.azure.account.key.<nombre de la cuenta de almacenamiento>.blob.core.windows.net` o `fs.azure.sas.<nombre del contenedor>.< nombre de la cuenta de almacenamiento>.blob.core.windows.net`

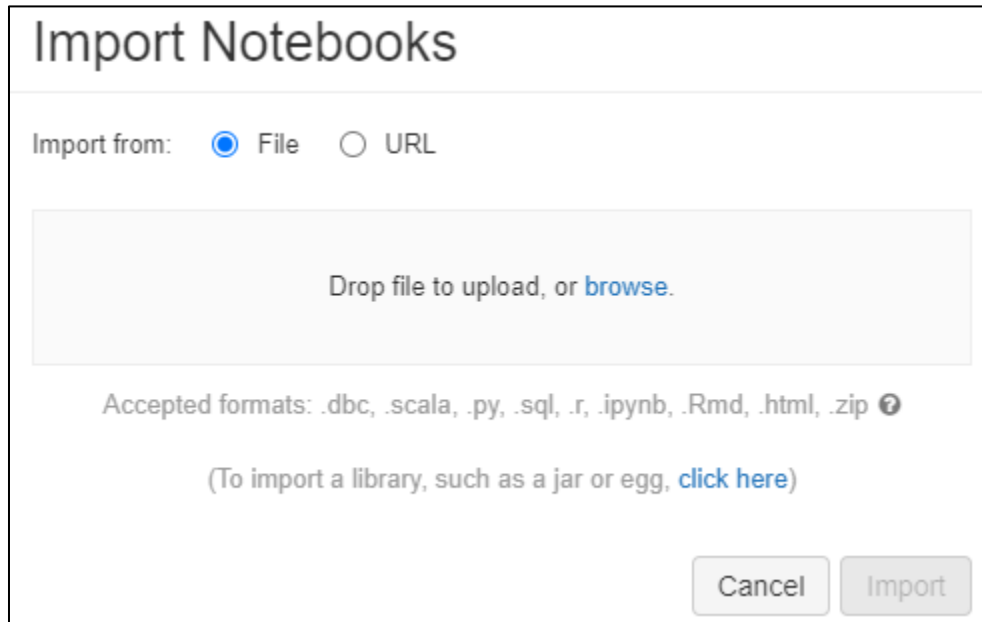
Luego ejecutamos y nos debe devolver *True*.

*¡Listo! Ya tenemos acceso a nuestros datos desde databricks.*

Continuamos a cargar nuestro notebook. Para esto, vamos a Workspace > Users > {tu\_usuario} y allí creamos una carpeta en la cual cargaremos el notebook.



Dentro de la carpeta, seleccionamos *import*, y procedemos cargar el notebook.



The image shows a dialog box titled "Import Notebooks". It has two radio buttons under "Import from:": "File" (selected) and "URL". Below this is a large light gray area with the text "Drop file to upload, or [browse](#).". Underneath that, it lists "Accepted formats: .dbc, .scala, .py, .sql, .r, .ipynb, .Rmd, .html, .zip" with a help icon. A note says "(To import a library, such as a jar or egg, [click here](#))". At the bottom right are "Cancel" and "Import" buttons.

***¡Listo! Ya tenemos nuestro notebook cargado, y podemos ejecutarlo, con este paso finalizamos la segunda fase de nuestra ETL, transformación.***



The image shows a Jupyter Notebook interface with three cells. The first cell, labeled "Cmd 1", contains a title "ETL en la nube: Un nuevo entorno para nuestros datos", a data source link "https://www.kaggle.com/dhruvildave/billboard-the-hot-100-songs", and two paragraphs of text about the Billboard Hot 100. The second cell, labeled "Cmd 2", is titled "Importar librerías". The third cell, labeled "Cmd 3", contains a code block for library imports.

```
1 #Procesamiento
2 import pandas as pd
3 import numpy as np
```

## Carga:

Una vez realizamos las transformaciones a nuestros datos vamos a almacenar los datos procesados para que podamos usarlos en otras fases del proyecto como análisis o implementación de modelos.

### Guardar datos

```
Cmd 17
1 pathOutput = '/dbfs/mnt/widsdata/'

Command took 0.02 seconds --

Cmd 18
1 data.to_csv(pathOutput+'dataCurada.csv')
```

**¡Listo!** Hemos realizado todos los pasos de ETL en Azure.

Si te interesa conocer más sobre este tema puedes visitar los siguientes enlaces:

- Azure Data Lake Storage Gen2:  
<https://docs.microsoft.com/en-us/azure/storage/blobs/create-data-lake-storage-account>
- Azure Databricks:  
<https://docs.microsoft.com/en-us/azure/databricks/scenarios/quickstart-create-databricks-workspace-portal?tabs=azure-portal>
- Clusters en Databricks:  
<https://docs.microsoft.com/en-us/azure/databricks/clusters/create>
- Acceso a Azure Datalake Gen2 desde Databricks:  
<https://docs.microsoft.com/en-us/azure/databricks/scenarios/store-secrets-azure-key-vault>