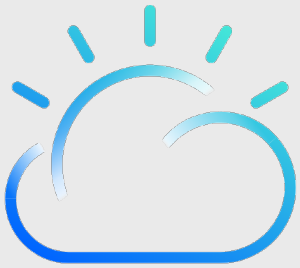


IBM Cloud

Hands-on Workshop

IBM Developer Advocacy

Módulo: Data Science – Jupyter Notebook con Machine Learning y App.

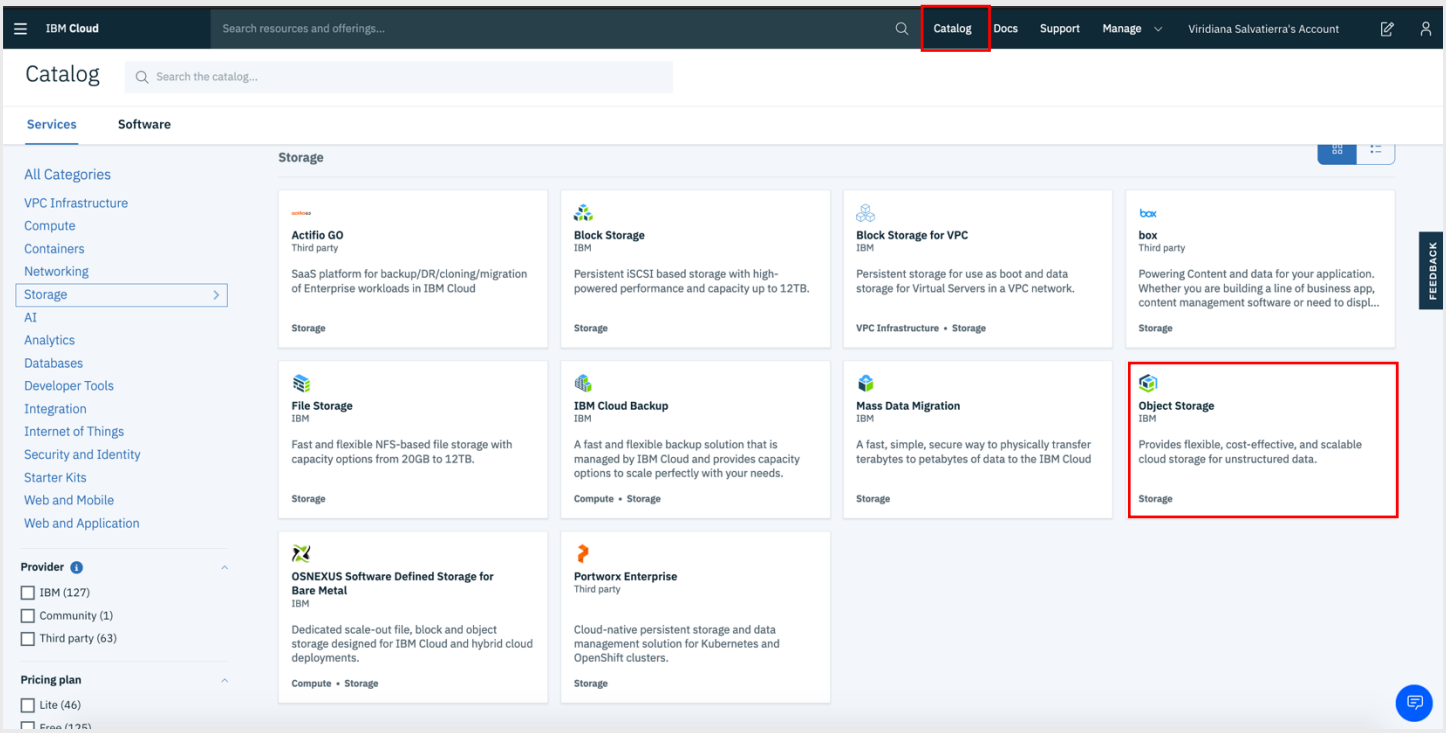
Agenda

0. Prework:

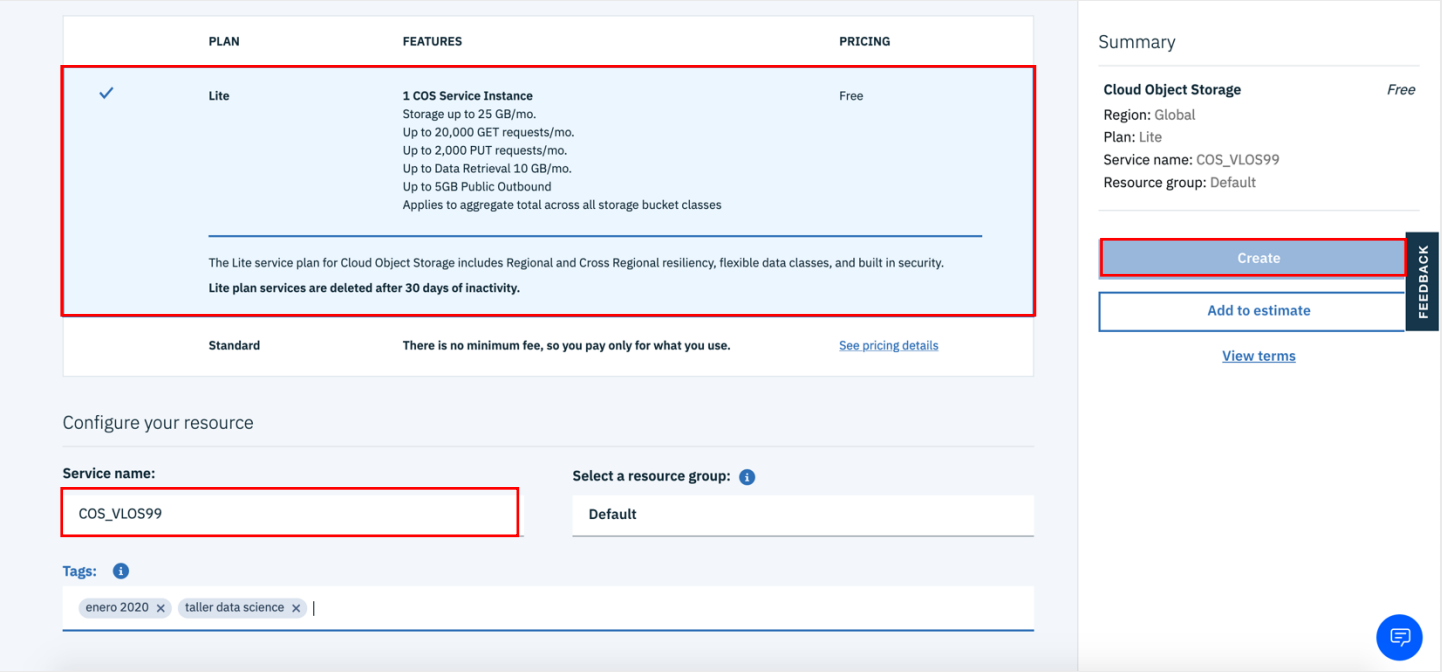
- Tener instalado github Desktop o git cli.
- Tener instalado el CLI de IBM Cloud (Si tienes problemas con este paso, da clic aquí para descargar la versión “STANDALONE”).
- Descargar el repositorio del siguiente enlace: https://github.com/ibmdevelopermx/CHURN-app-Jupyter-Notebook-y-WS.

1. Crear una instancia de Cloud Object Storage.

1. Realizamos login a IBM Cloud http://cloud.ibm.com
2. Seleccionamos “**Catalog**” (Catálogo) en el menú superior.
3. En las categorías de la izquierda, seleccionamos “**Storage**”.
4. Seleccionamos el servicio de “**Object Storage**”.

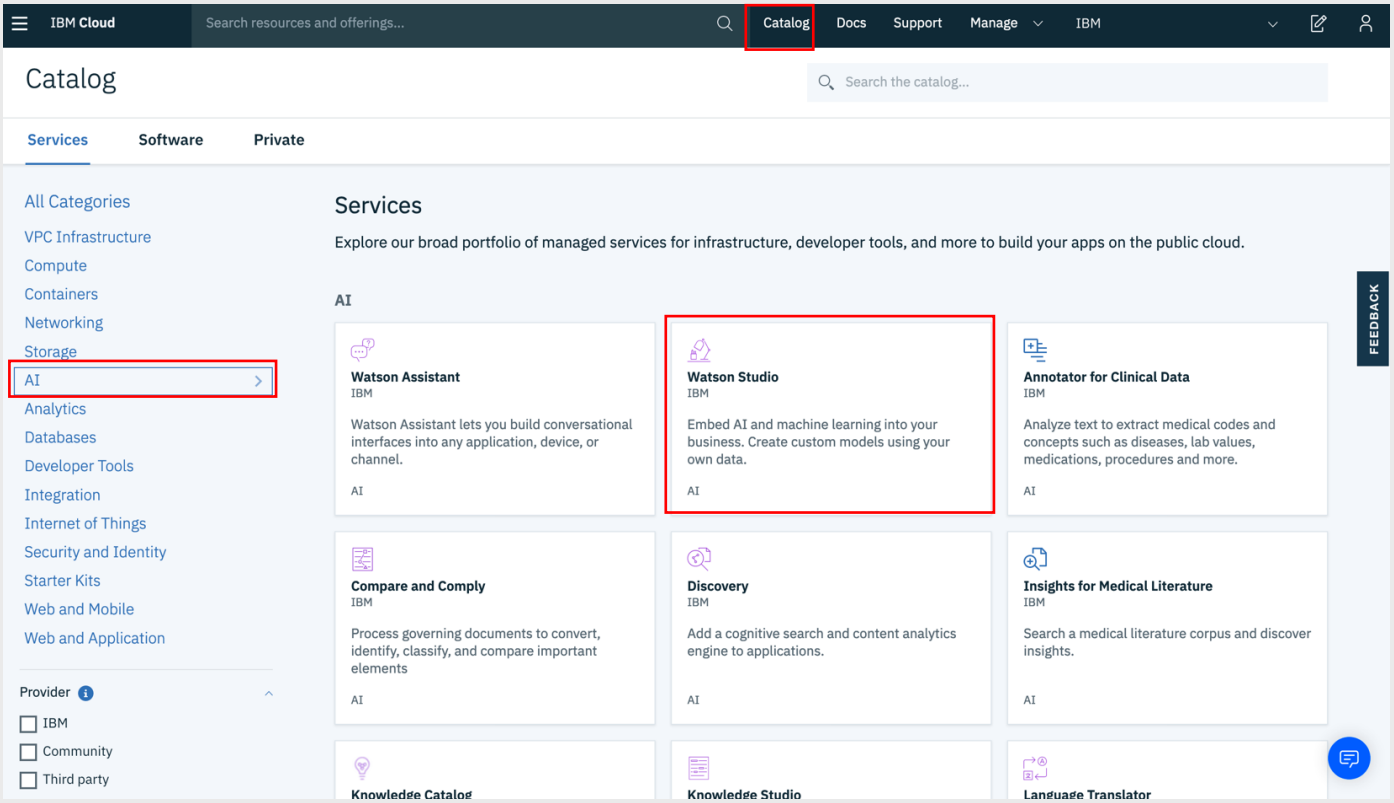


5. Seleccionamos el plan “**Lite**”, le damos un nombre a nuestra instancia de servicio y damos clic en “**Create**”.

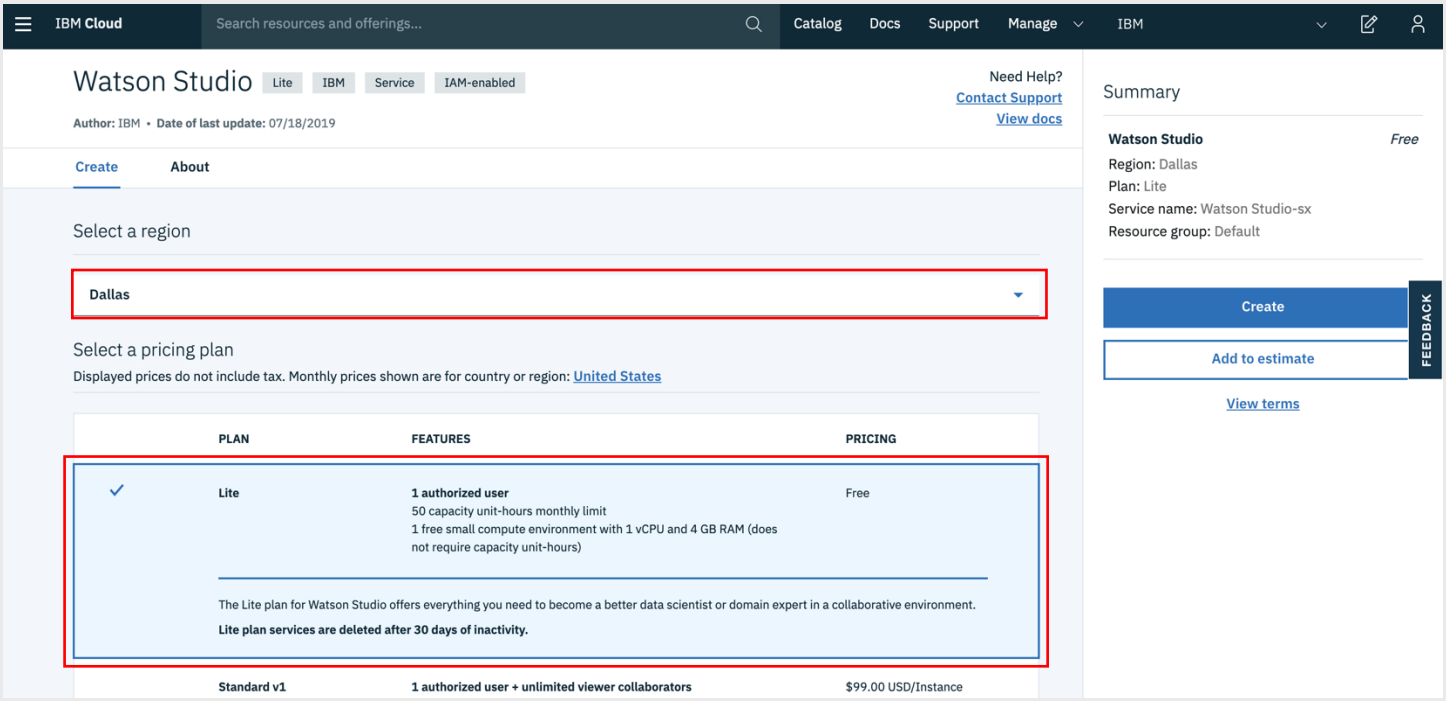


2. Crear una instancia de Watson Studio:

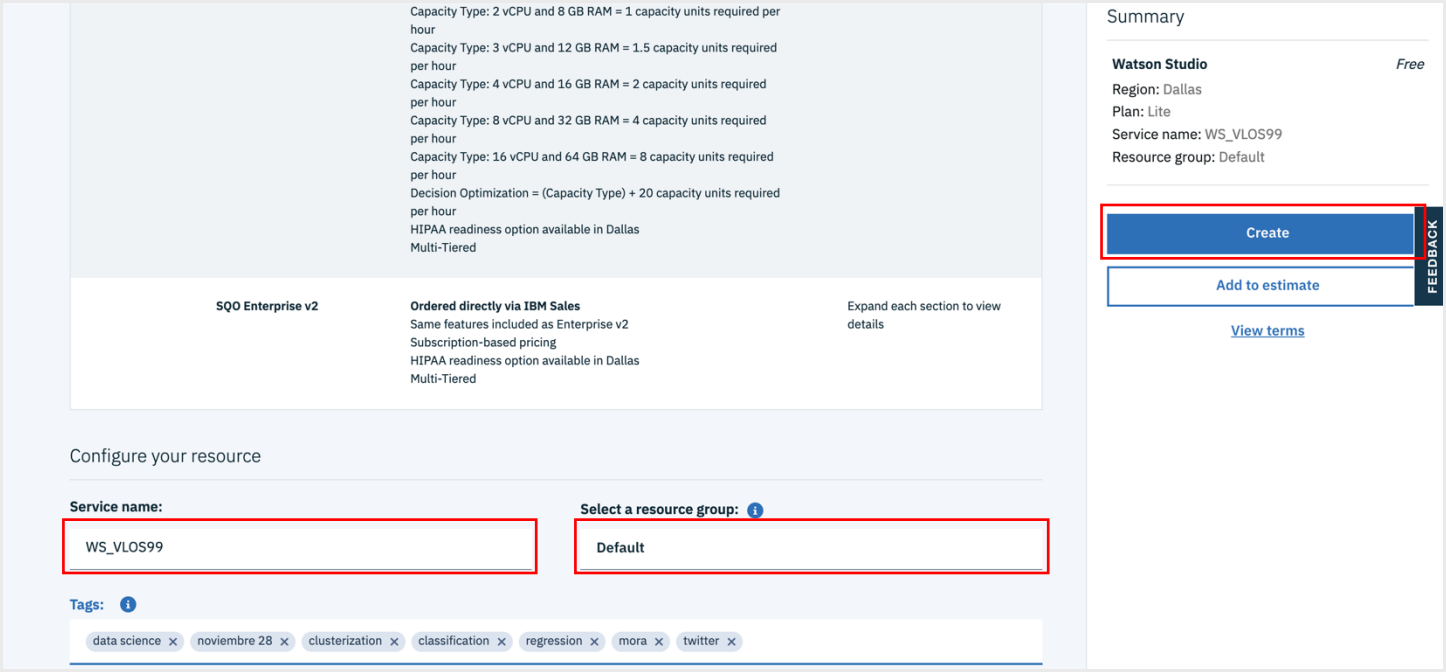
1. Realizamos login a IBM Cloud
   http://cloud.ibm.com
2. Seleccionamos “**Catalog**” (Catálogo) en el menú superior.
3. En las categorías de la izquierda, seleccionamos “**AI**”.
4. Seleccionamos el servicio de “**Watson Studio**”



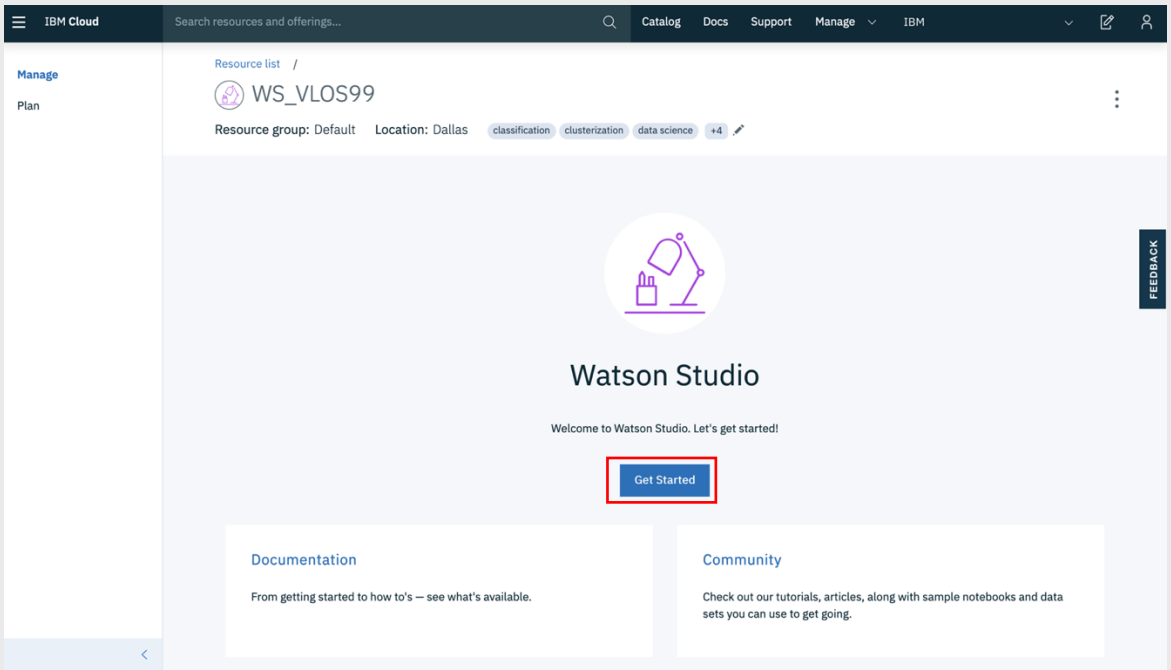
5. Una vez dentro del servicio, seleccionamos la región “**Dallas**”.
6. Seleccionamos el plan “**Lite**”.



7. En la parte inferior de la página, escribir el “**Nombre de la instancia del Servicio**”. (Es recomendable agregar etiquetas o “**tags**”)
8. En grupo de recursos, dejamos el valor “**Default**”.
9. Dar clic en “**Create**”.

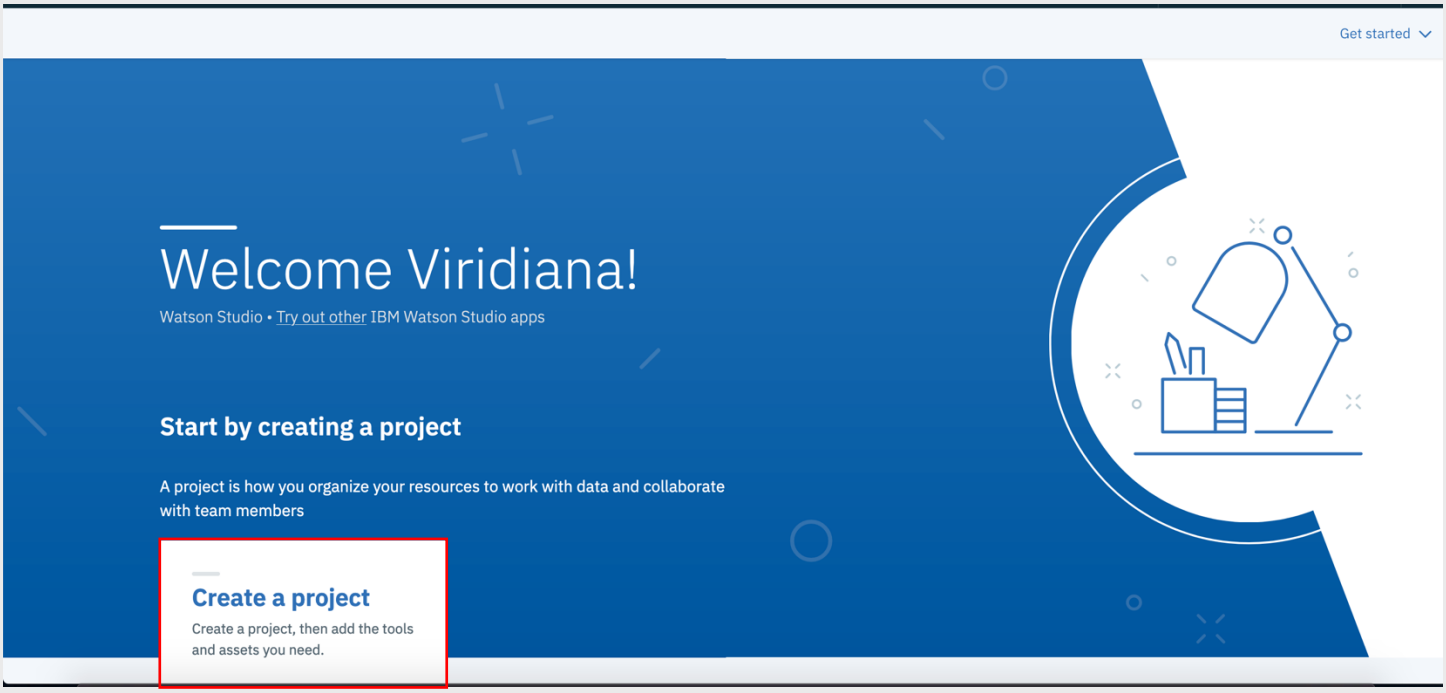


10. En la pestaña que se despliegue, damos clic en “**Get Started**” de la pestaña “**Manage**”.

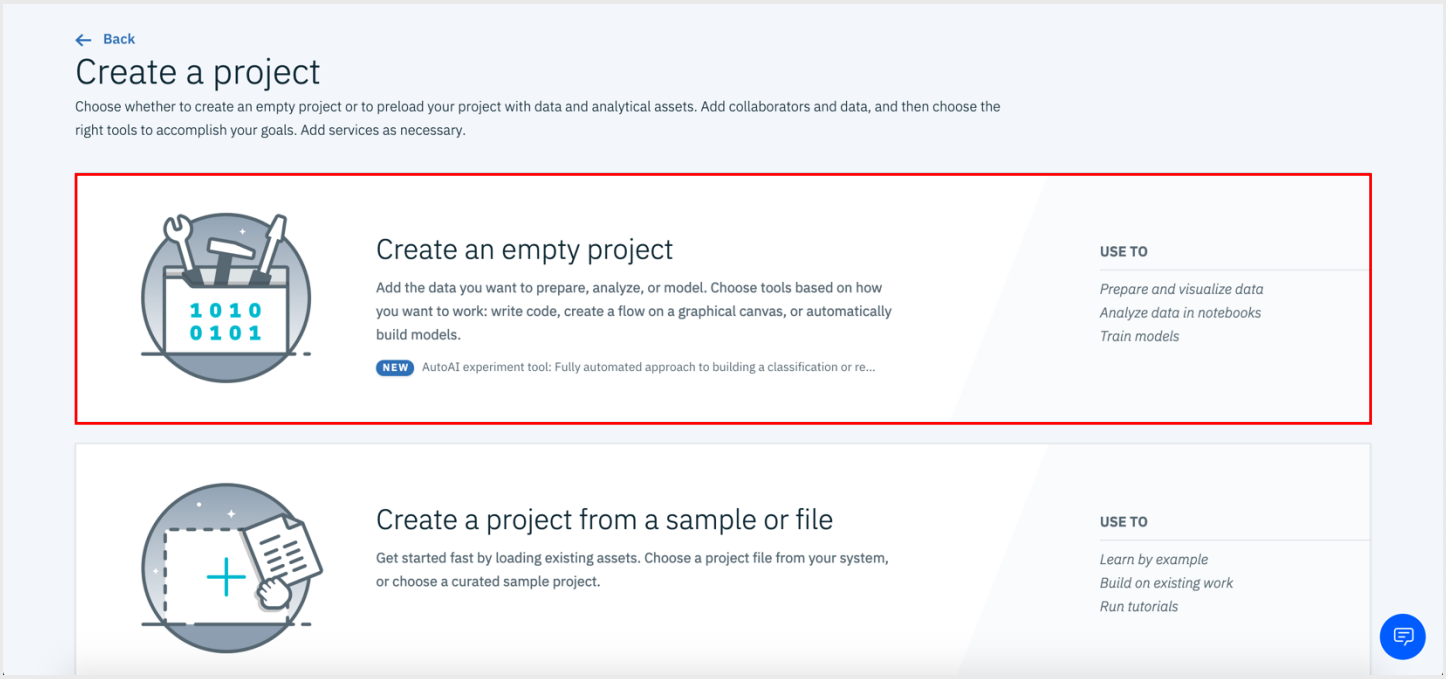


3. Crear un proyecto:

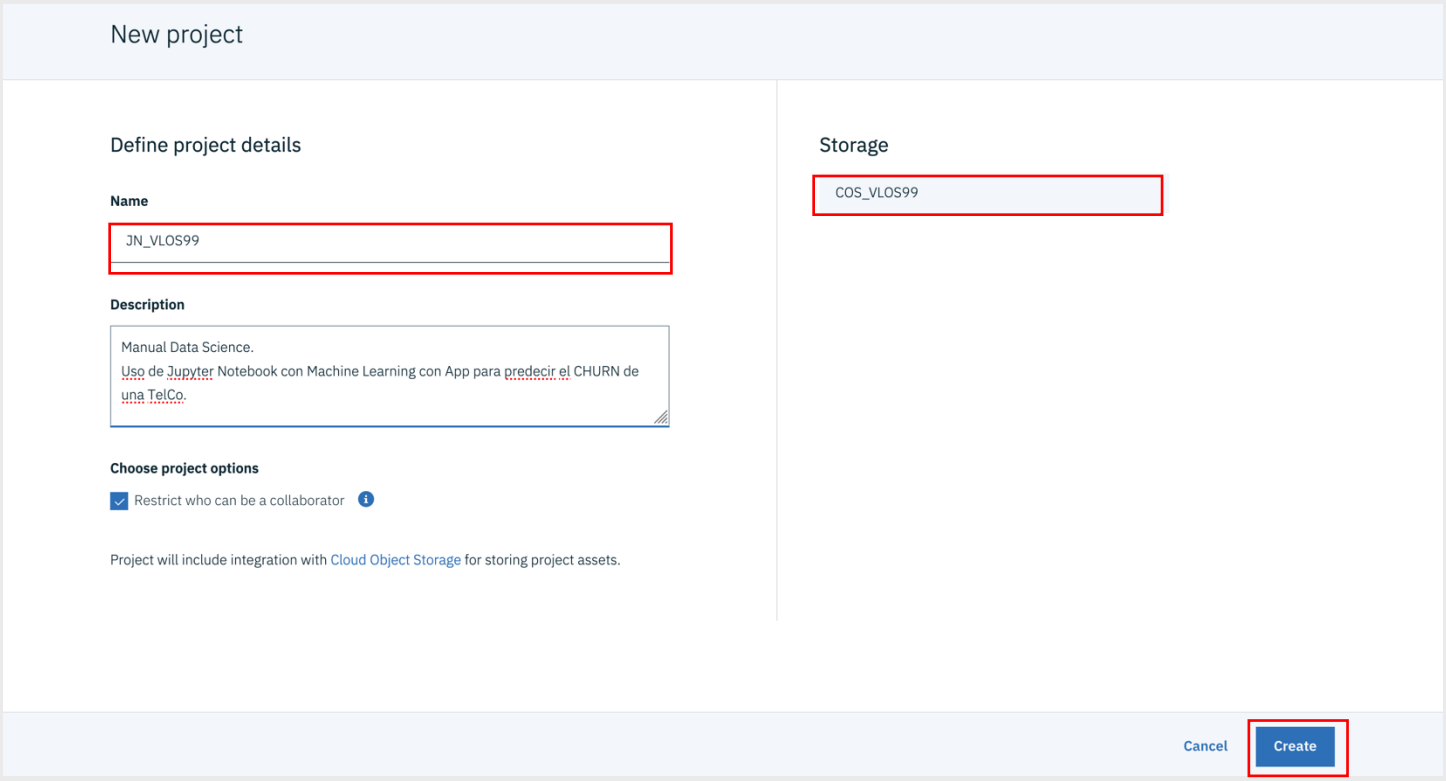
1. Una vez en la página principal de nuestra instancia, damos clic en “**Create a project**”.



2. Seleccionamos la opción “**Create an empty project**”.

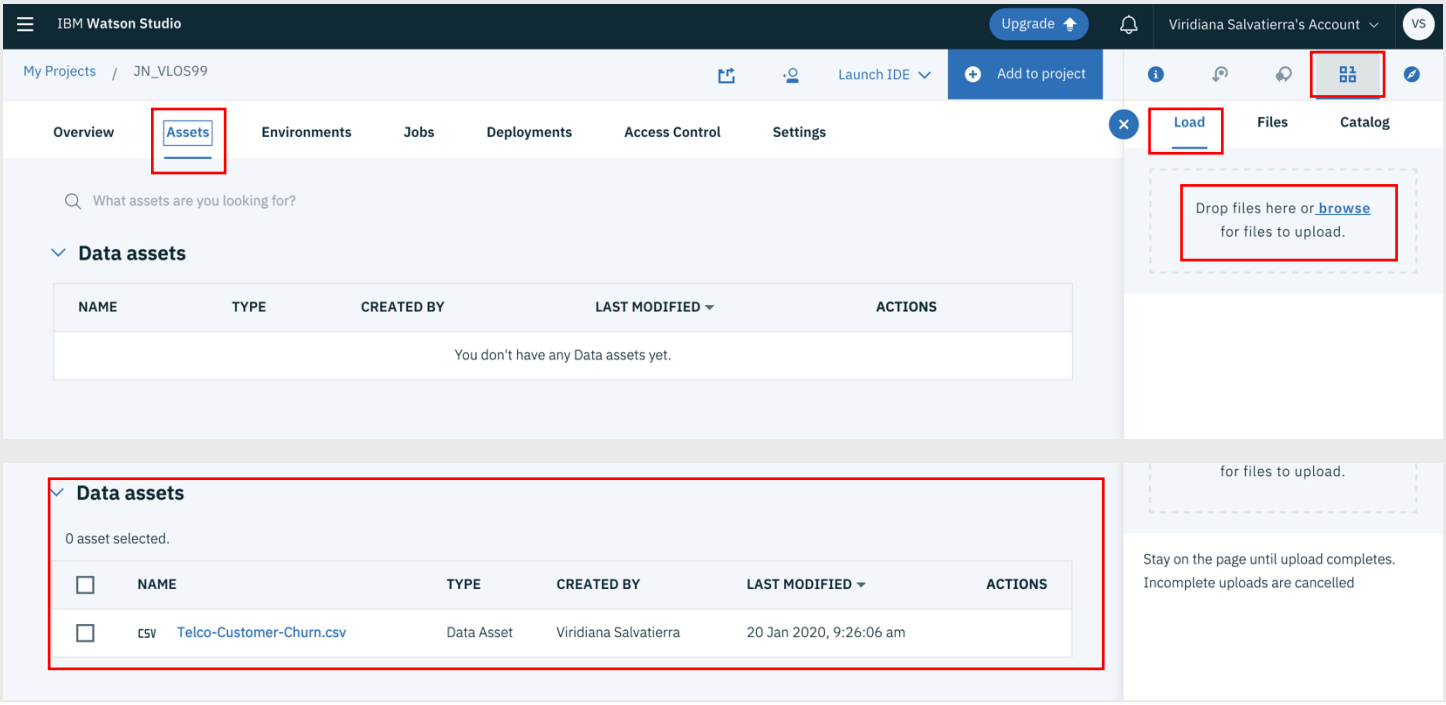


3. Ahora, le damos un nombre al proyecto, comprobamos que nuestra instancia de “**Cloud Object Storage**” sea correcta y damos clic en “**Create**”.



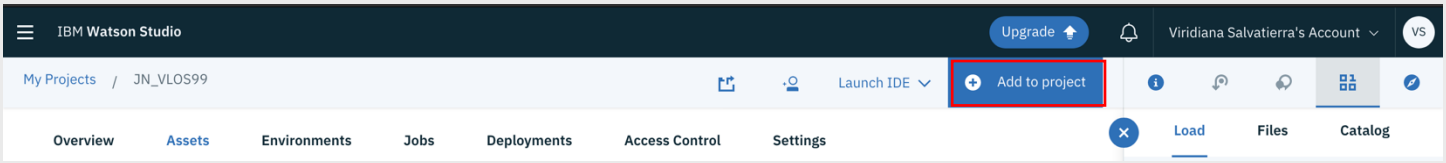
4. Cargar un *data asset*:

1. Una vez dentro del proyecto que acabamos de crear, damos clic en "**Assets**". Del lado derecho, en el apartado de "**Find and add data**", seleccionamos "**Browse**" o arrastramos y soltamos el archivo "**Telco-Customer-Churn.csv**" de la carpeta "**Data**" del repositorio que recién clonamos.

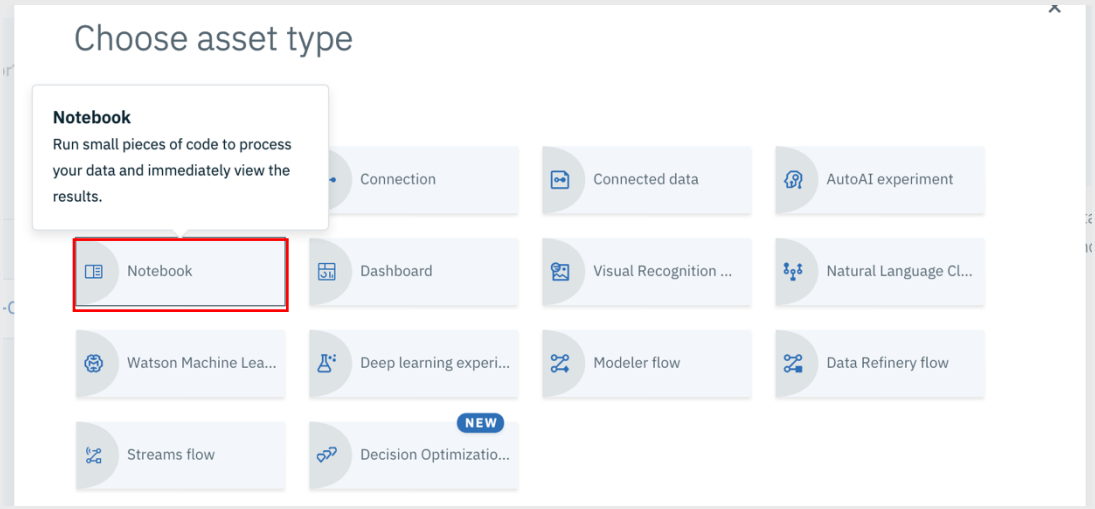


5. Importar una Jupyter Notebook:

1. Damos clic en “**Add to project**”.

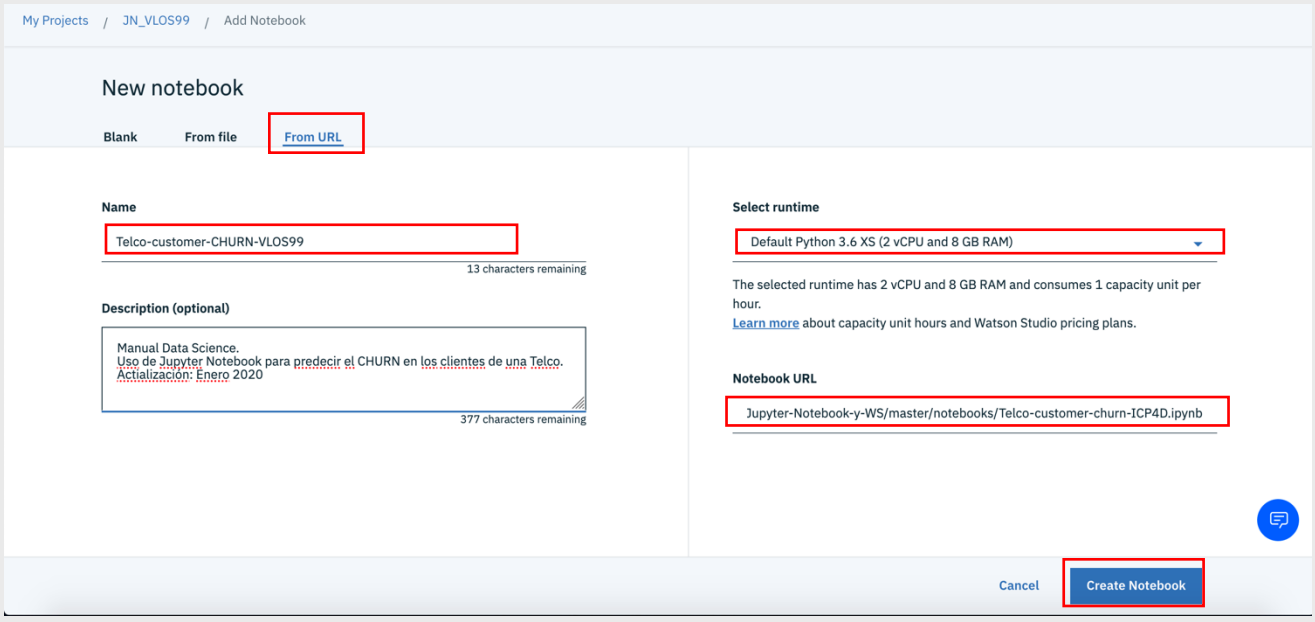


2. Elegimos “**Notebook**” como “**Asset Type**”.



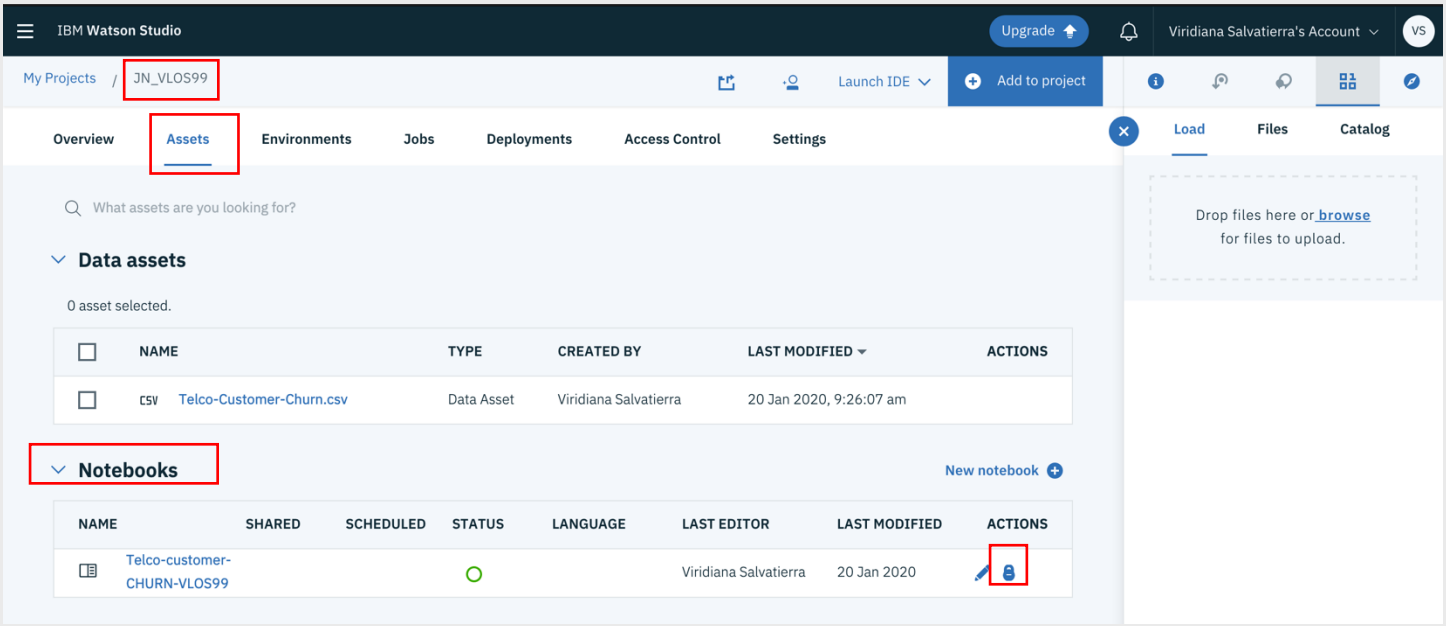
3. Ahora seleccionamos la pestaña “**From URL**”. Le damos un nombre original a nuestra Jupyter Notebook, elegimos la opción “**Python 3.6**” en “**Enviroment**” y en URL pegamos la siguiente dirección: https://raw.githubusercontent.com/ibmdevelopermx/CHURN-app-Jupyter-Notebook-y-WS/master/notebooks/Telco-customer-churn-ICP4D.ipynb

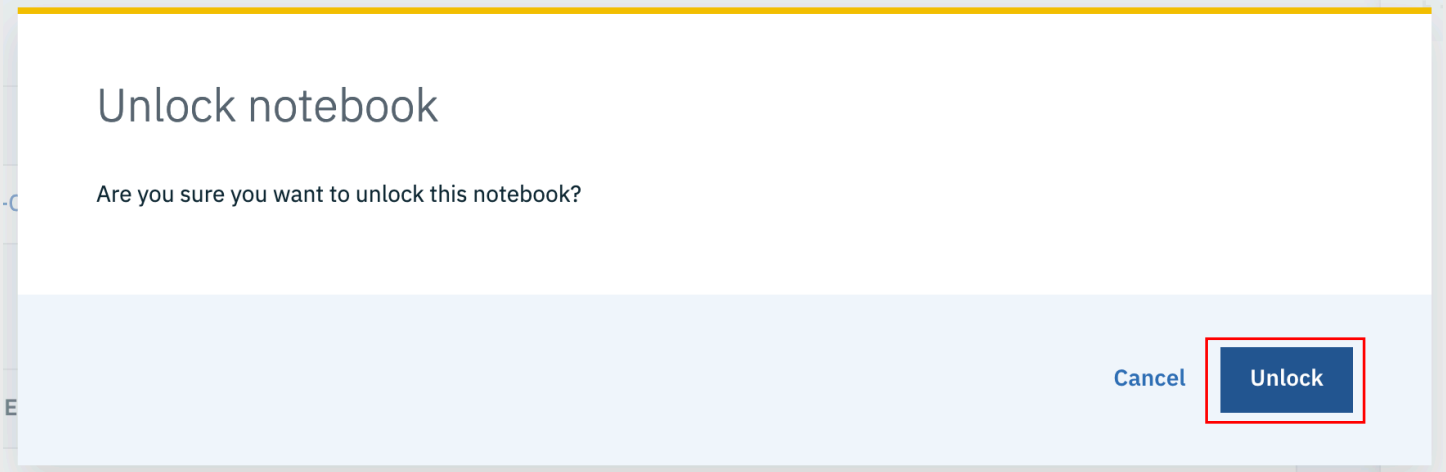
   Y damos clic en “**Create Notebook**”.



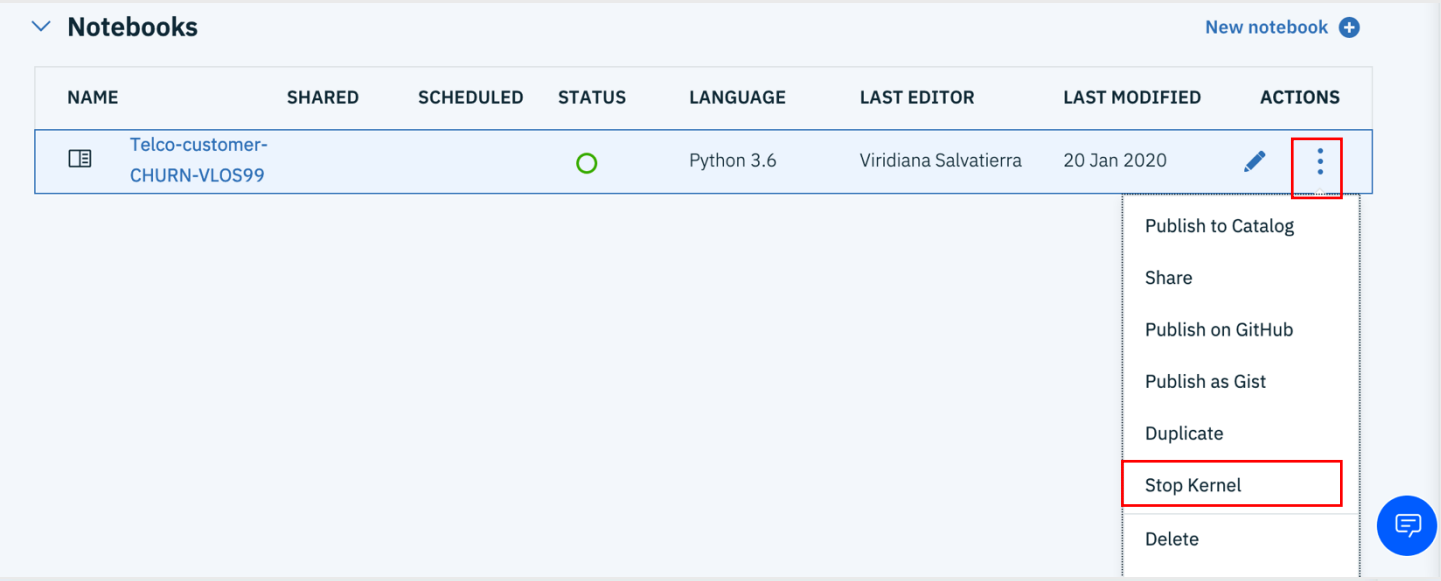
4. Una vez que cargue nuestra Jupyter Notebook, podemos empezar a correr las celdas.

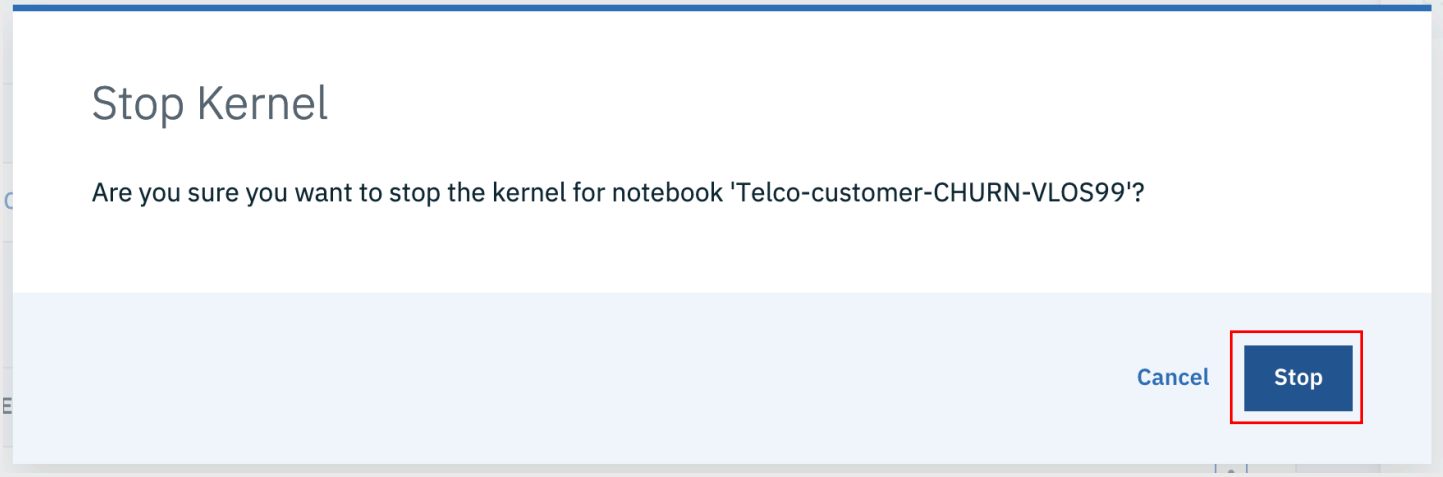
5. Tras correr la primera, regresamos a nuestro proyecto. Seleccionamos la pestaña "**Assets**" y en el apartado "**Notebooks**", buscamos nuestro notebook. Damos clic en el candado "**Unlock notebook**" y en la ventana emergente, damos clic en "**Unlock**".





6. Ahora damos clic en el menú de “hamburguesa” (aparece al pasar el puntero sobre él). Y seleccionamos la opción “**Stop Kernel**”. En la ventana emergente, damos clic en “**Stop**”. (Esto es para no gastar los recursos de memoria asignados a esta *NOTEBOOK*).

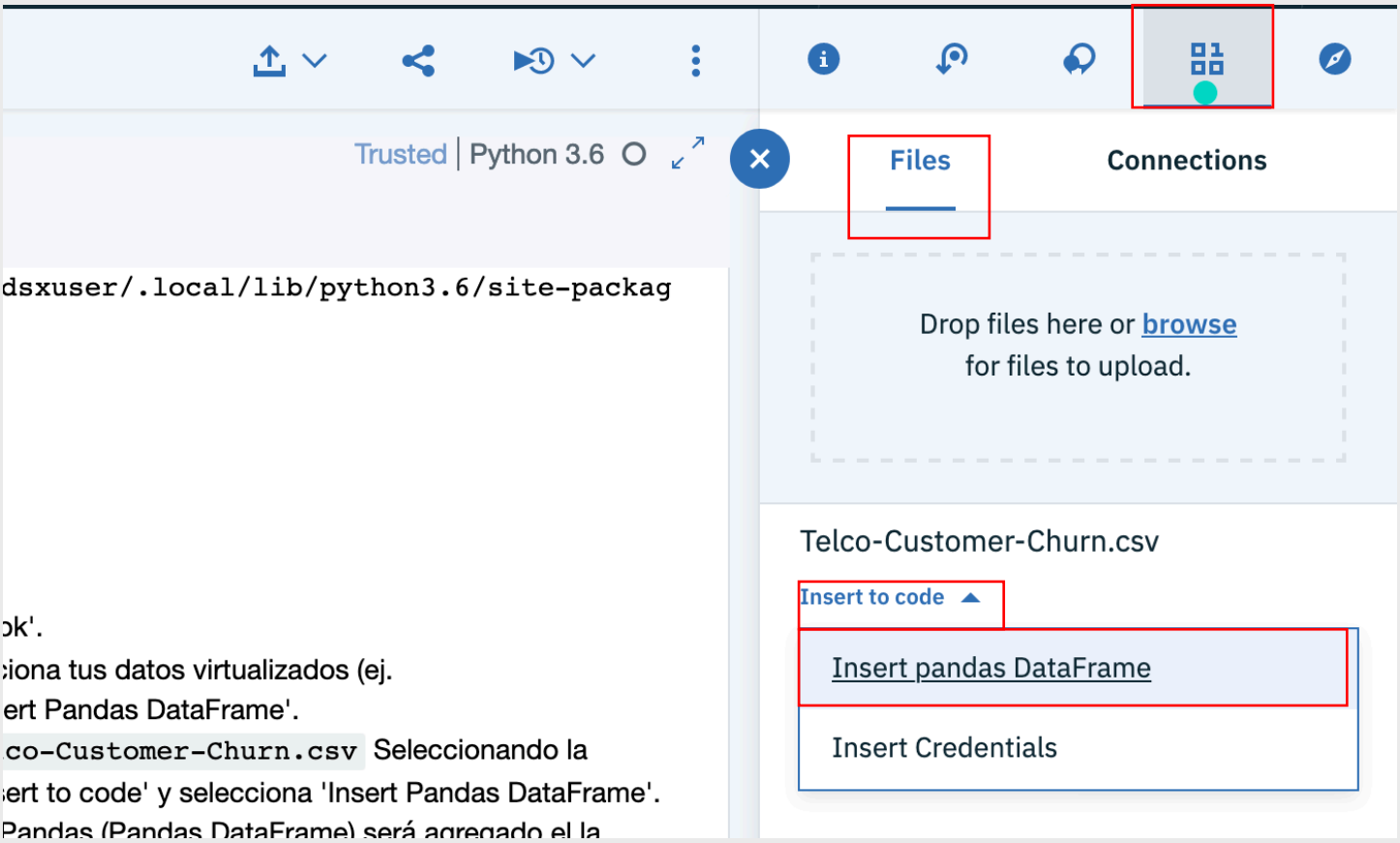




6. Cargar un *data asset* a nuestra *Notebook*:

1. Damos clic en nuestra “Notebook”. Una vez dentro de ésta, damos clic en el lápiz para editar.

2. Damos clic en la celda del paso 2. Ahora seleccionamos el ícono “01/00” “**Find and add data**” del lado superior derecho de nuestra Notebook. Da clic en la pestaña “**Files**”, damos clic en “**Insert to code**” y seleccionamos la opción “**Insert pandas DataFrame**”.



3. Hecho esto, corremos la segunda celda. Nos daremos cuenta de que se despliega una tabla según los parámetros de la celda.

```
df_data_2 = pd.read_csv(body)
df_data_2.head()
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	T
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

4. Antes de correr la tercera celda, debemos asegurarnos de que la variable “n” en “df_data_n” sea igual en la segunda y tercera celdas. Ejemplo: en mi caso, el resultado

del proceso de la celda 2, dio como resultado “df_data_2”, por lo que en la tercera celda, “df” debe tener asignado el valor “df_data_2”.

```
df_data_2 = pd.read_csv(body)
df_data_2.head()
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	T
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

Usaremos la convención de nombramiento de Pandas 'df' para nuestro marco de datos (DataFrame). Asegúrate de que la celda inferior use el mismo nombre para el marco de datos(DataFrame) usado en la celda superior. Para el archivo cargado de forma local, debe verse como: 'df_data_1' o 'df_data_2' o 'df_data_x'. Para el caso de los datos virtualizados, debe verse como: data_df_1 o data_df_2 o data_df_x.

```
In [3]: # Para datos virtualizados:
        # df = data_df_1

        # Para carga local:
        df = df_data_2
```

5. A continuación, seguimos ejecutando las celdas hasta llegar al apartado 3.0 “Crear un modelo”.

7. Crear un modelo:

1. En esta sección, se realiza la separación de datos de prueba y de entrenamiento.

3.0 Crear un modelo

```
In [ ]: from pyspark.sql import SparkSession
        import pandas as pd
        import json

        spark = SparkSession.builder.getOrCreate()
        df_data = spark.createDataFrame(df)
        df_data.head()
```

3.1 Dividir los datos en conjuntos de entrenamiento y de prueba.

```
In [ ]: spark_df = df_data
        (train_data, test_data) = spark_df.randomSplit([0.8, 0.2], 24)

        print("Número de registros para entrenamiento: " + str(train_data.count()))
        print("Número de registros para evaluación: " + str(test_data.count()))
```

2. En el apartado 3.5, se crea la “*pipeline*” y el modelo utilizando el algoritmo clasificador “Random Forest”.

3.5 Crear una 'pipeline', y ajustar un modelo usando 'RandomForestClassifier'.

Montar todas las etapas a una 'pipeline'. No esperamos una regresión lineal limpia, así que usaremos 'RandomForestClassifier' para encontrar el mejor árbol de decisión para nuestros datos.

```
In [44]: classifier = RandomForestClassifier(featuresCol="features")

pipeline = Pipeline(stages=[si_gender, si_Partner, si_Dependents, si_PhoneService, si_MultipleLines, si_InternetSe
                            si_TechSupport, si_StreamingTV, si_StreamingMovies, si_Contract, si_PaperlessBilling,
                            classifier, label_converter])

model = pipeline.fit(train_data)
```

```
In [45]: predictions = model.transform(test_data)
evaluatorDT = BinaryClassificationEvaluator(rawPredictionCol="prediction")
area_under_curve = evaluatorDT.evaluate(predictions)

#La evauación predeterminada es 'areaUnderROC'
print("areaUnderROC = %g" % area_under_curve)
```

```
areaUnderROC = 0.694581
```

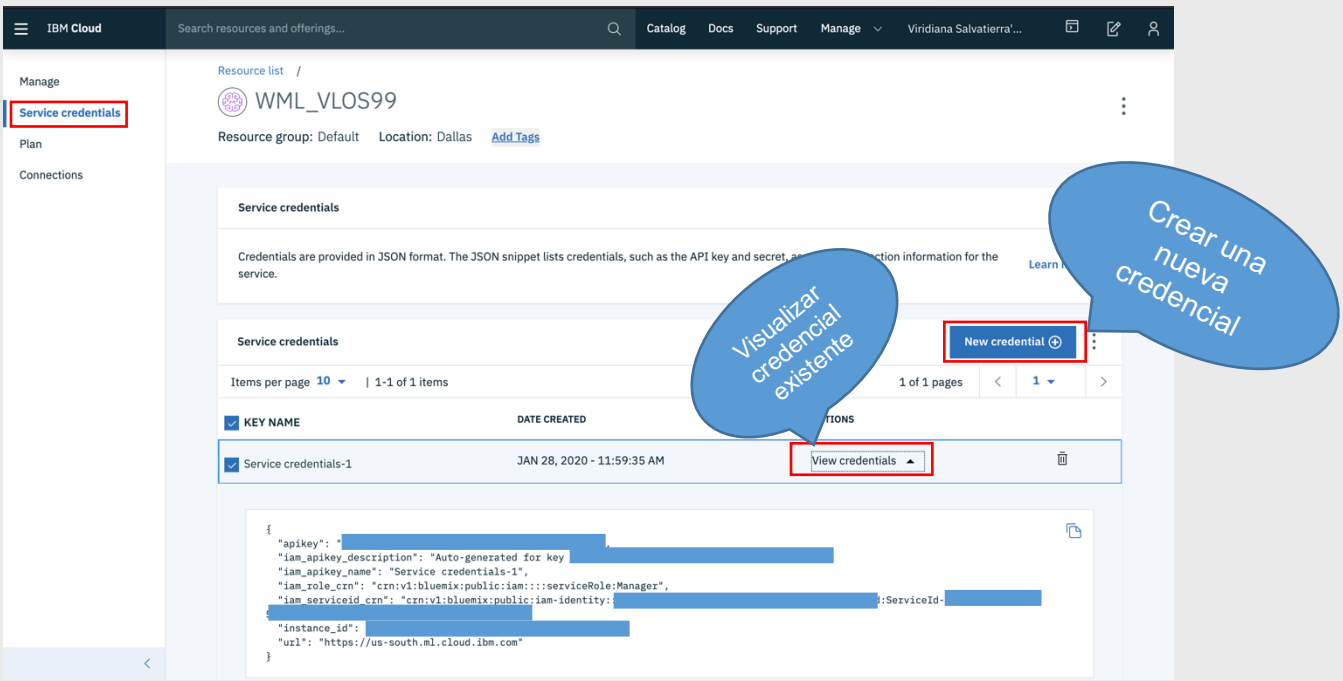
8. Guardar el modelo y probar los datos:

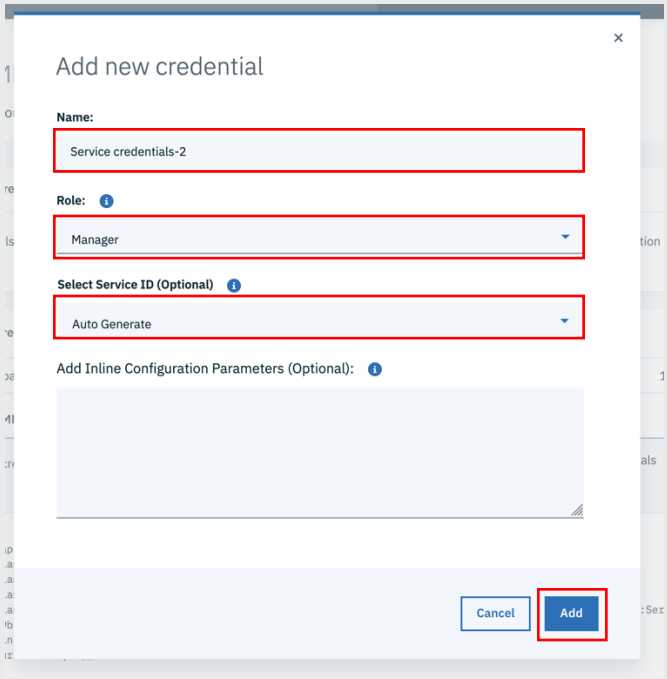
1. Lo primero que debemos hacer, es darle un nombre fácilmente identificable a nuestro modelo. Podemos usar nuestras iniciales y un número o la fecha.

Agrega un nombre único para el MODEL_NAME.

```
In [47]: MODEL_NAME = "JN_VLOS_99-Enero-21-2020"
```

2. En las credenciales, cambiamos el “**url**”, la “**apikey**” y la “**instance_id**” por tus propias credenciales, para encontrarlas, en la página del servicio de “**Watson Machine Learning**”, damos clic en el apartado de “**Service Credentials**” y desplegamos la que tengamos generada o creamos una nueva credencial.

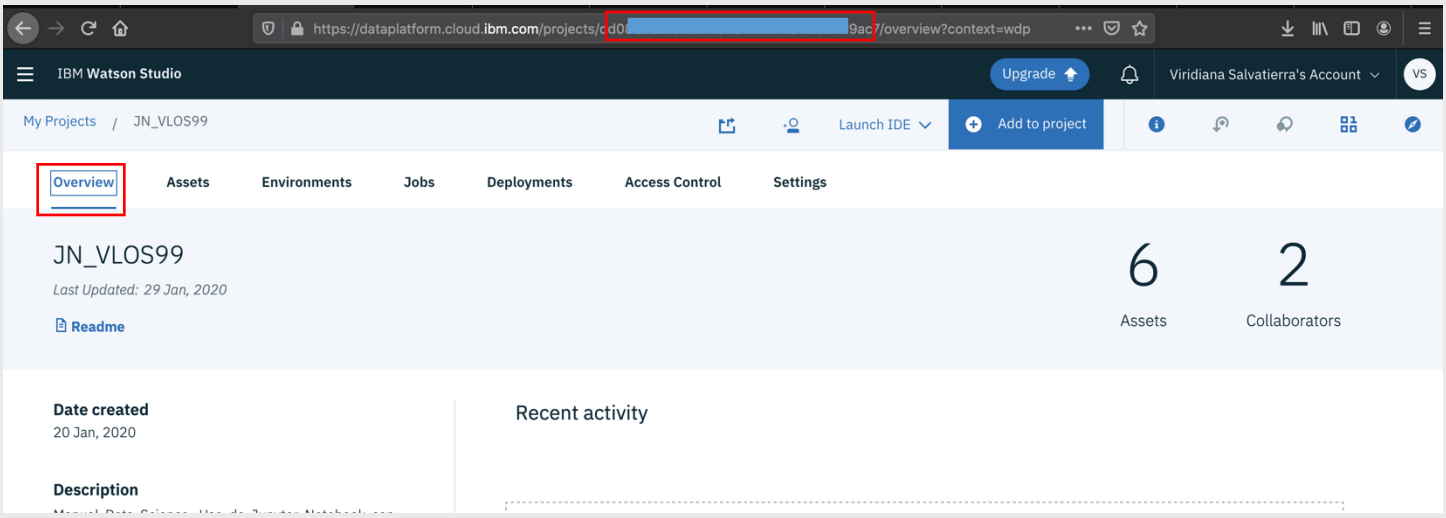




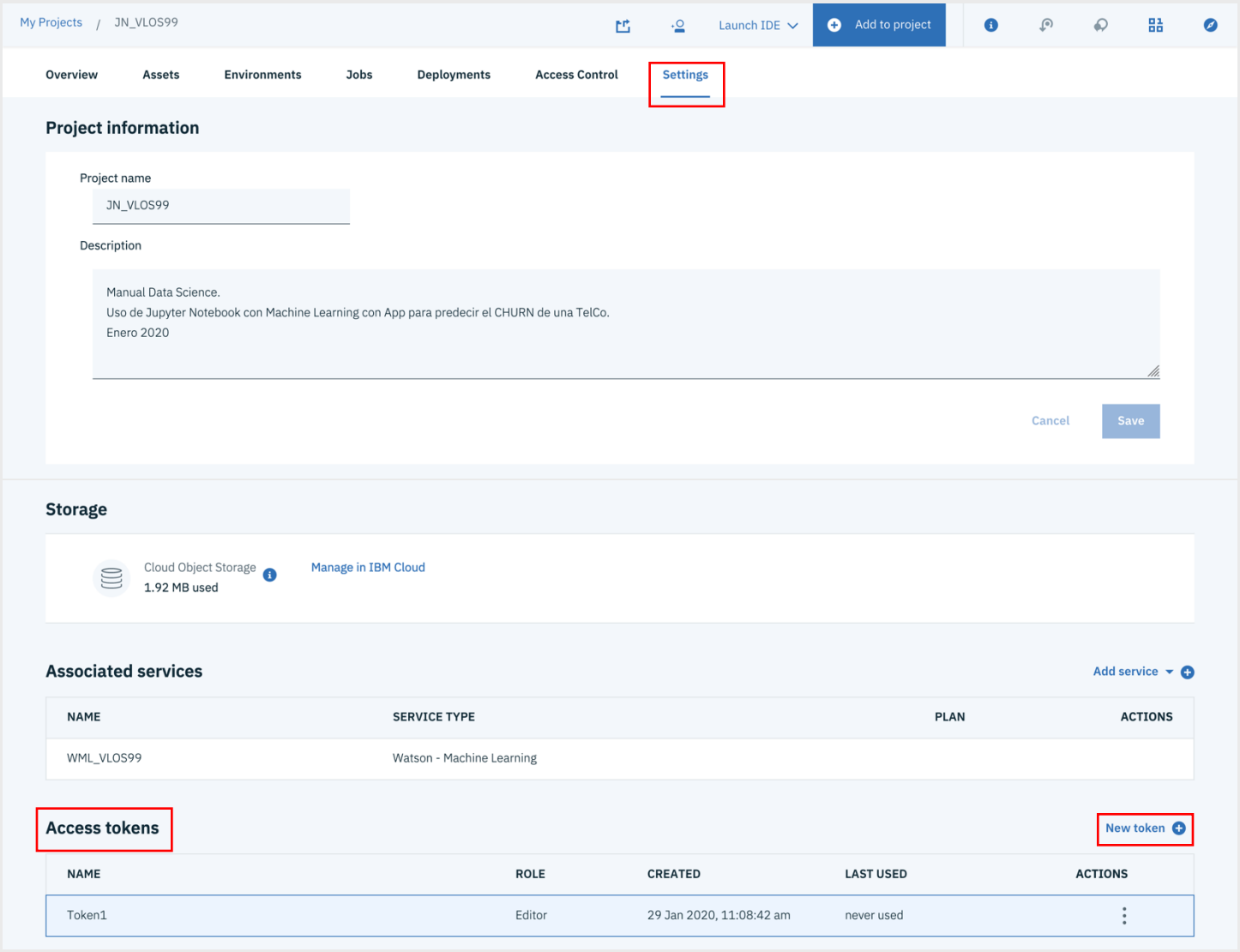
4.1 Guardar el modelo en 'ICP4D local Watson Machine Learning'.

```
In [115]: from watson_machine_learning_client import WatsonMachineLearningAPIClient


wml_credentials = wml_credentials = {
                   "url": "https://us-south.ml.cloud.ibm.com",
                   "apikey":"[redacted]",
                   "instance_id": "[redacted]"
                  }
client = WatsonMachineLearningAPIClient(wml_credentials)
```

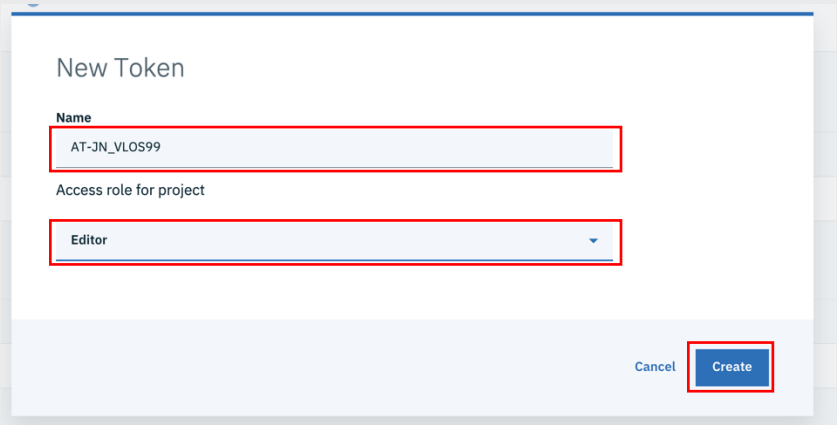
3. Corremos la celda siguiente.
4. En las siguientes dos celdas (4.2 y 4.3), cambiamos los valores “**project_id**” y “**Access_Token**” por nuestros valores. Para obtener el “**Project ID**”, simplemente vamos al “**Overview**” de nuestro projecto y copiamos, del “**URL**” dicho ID.



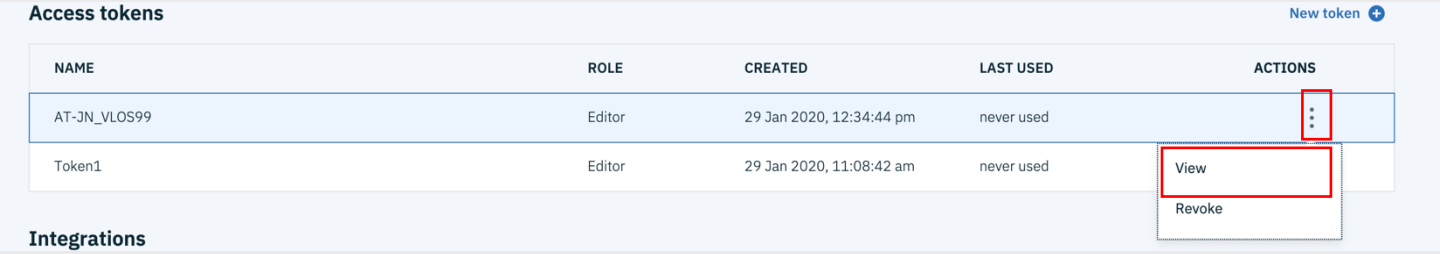
5. Ahora, para obtener el “**Access Token**”, vamos a la pestaña de “**Settings**”, buscamos el apartado de “**Access Token**” y damos clic en “**New Token**”.



6. Para crear nuestro “**Token**”, le vamos a dar un nombre que identifiquemos fácilmente, y en el apartado de “**Access role for project**”, seleccionamos la opción de “**Editor**” y damos clic en “**Create**”.



7. Hecho esto, vamos a dar clic en el menú de “hamburguesa” (aparece al pasar el mouse sobre él) y seleccionamos la opción “**view**”.

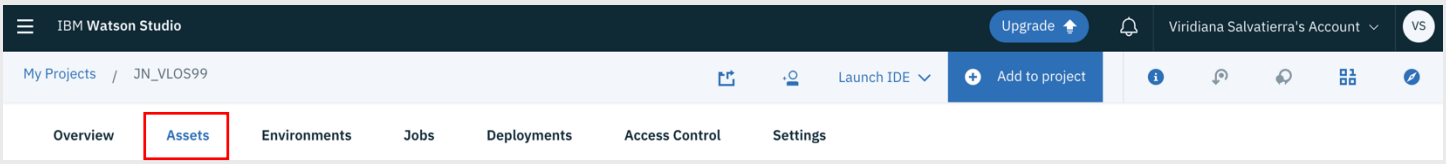


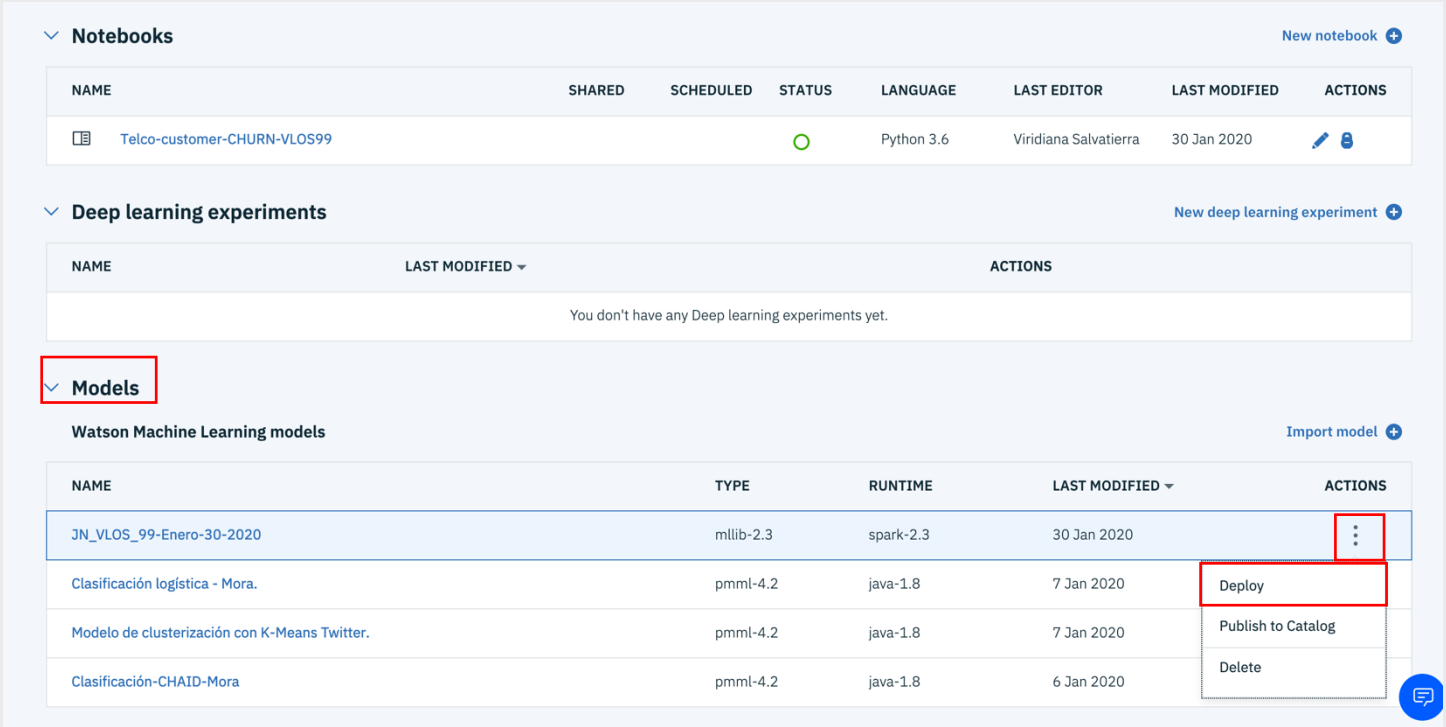
8. Ahora copiamos el valor del token y lo reemplazamos en nuestra Jupyter Notebook, así como con el valor de nuestro "**project ID**" en las celdas de los apartados **4.2** y **4.3** de nuestra "Jupyter Notebook"

9. Hecho esto ya habrás terminado de crear tu modelo.

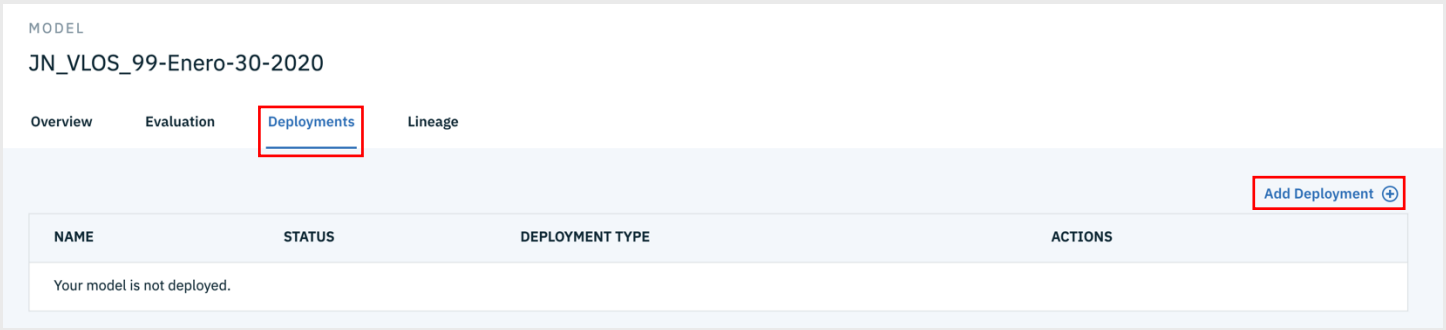
9. Desplegar el modelo:

1. Para realizar nuestro despliegue, vamos a nuestro proyecto en watson studio, y en la pestaña "**Assets**", buscamos el apartado "**Models**". Ahí debemos poder ver el modelo que recién creamos. Damos clic en el menú de "hamburguesa" (aparece al pasar el "mouse" sobre él) y seleccionamos la opción. "**Deploy**".

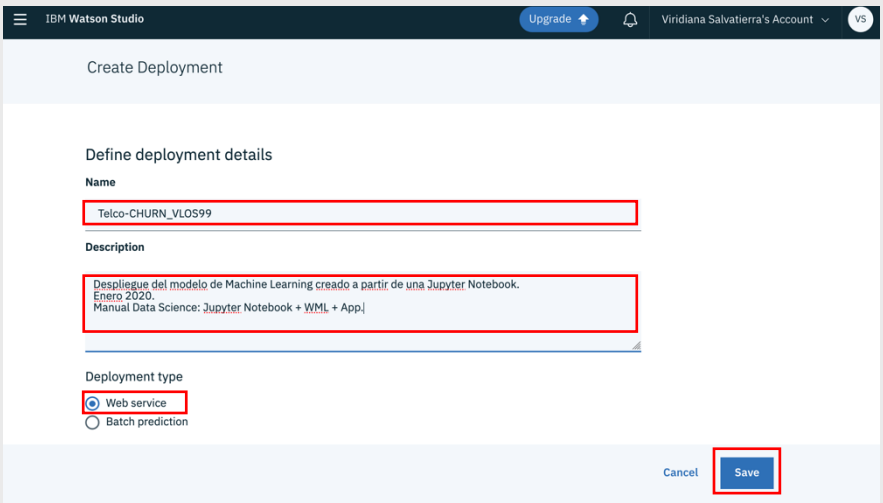




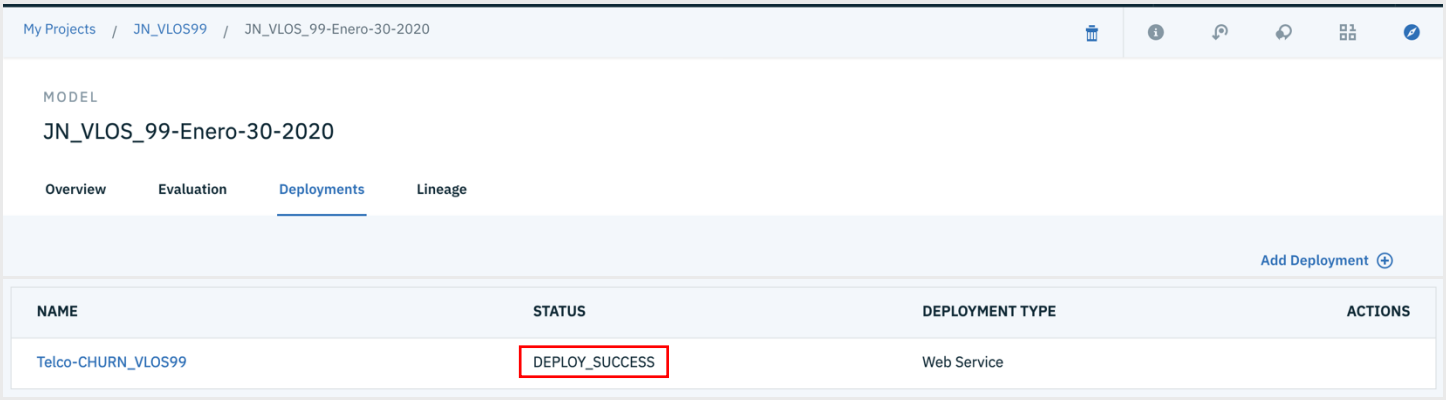
2. Una vez que hemos sido redireccionados a la página del modelo, en la pestaña "**Deployments**", damos clic en "**Add Deployment**".



3. Ahora, le damos un nombre a nuestro despliegue, después, en "**Deployment type**", seleccionamos la opción "**Web service**" y damos clic en "**Save**".

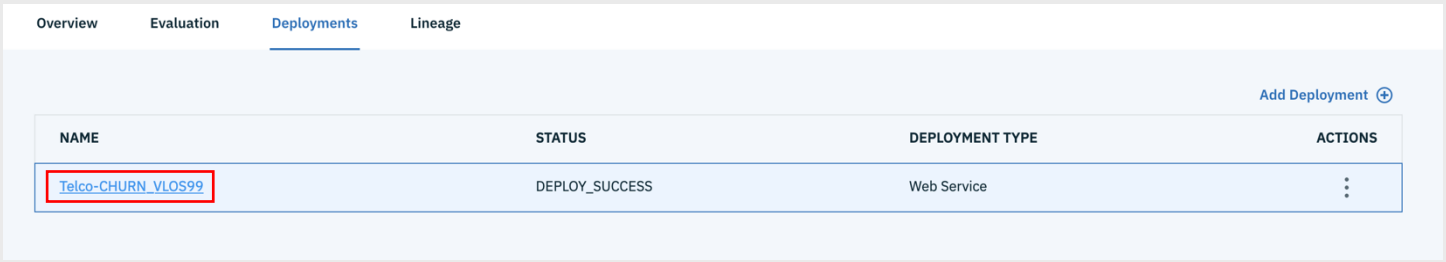


4. Ahora podemos visualizar nuestro despliegue dentro de la página de nuestro modelo, en el apartado “**Deployments**”. Cuando el “status” cambie a “**DEPLOY_SUCCESS**”, nuestro modelo habrá sido desplegado exitosamente. (Si no cambia el “status” en un par de minutos, te recomendamos refrescar la página.)

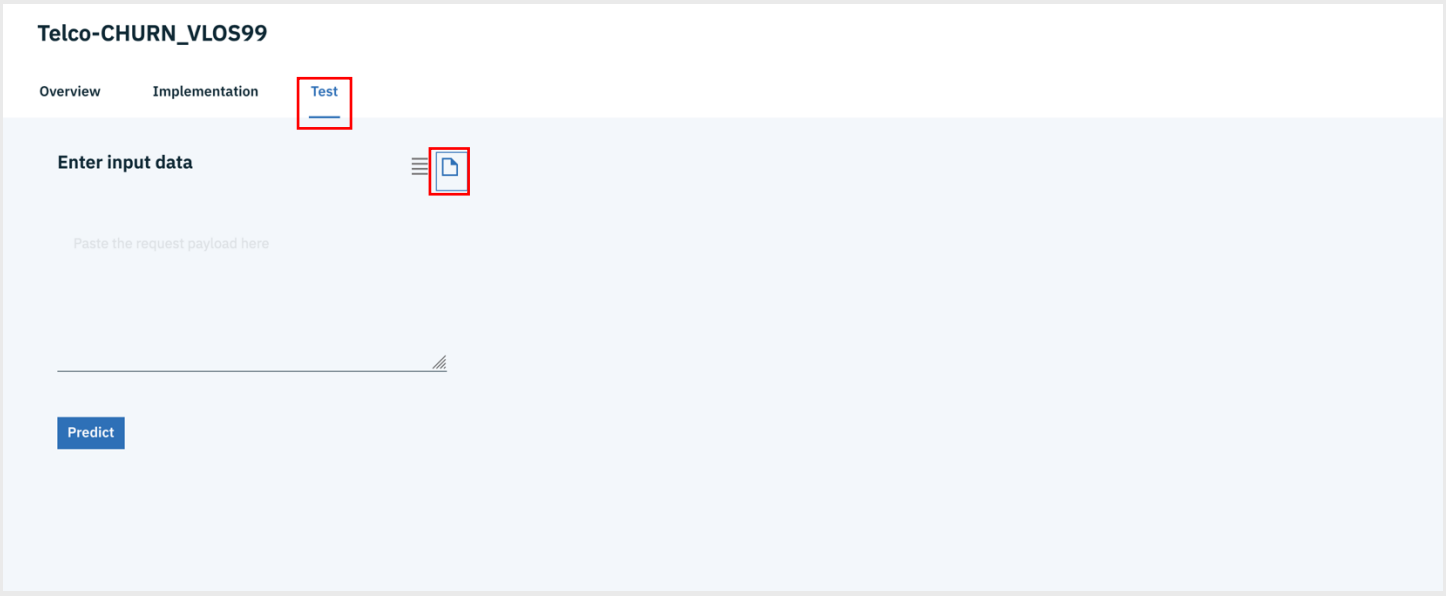


10. Probar el modelo:

1. Una vez que desplegamos el modelo, damos clic en el nombre de nuestro despliegue.

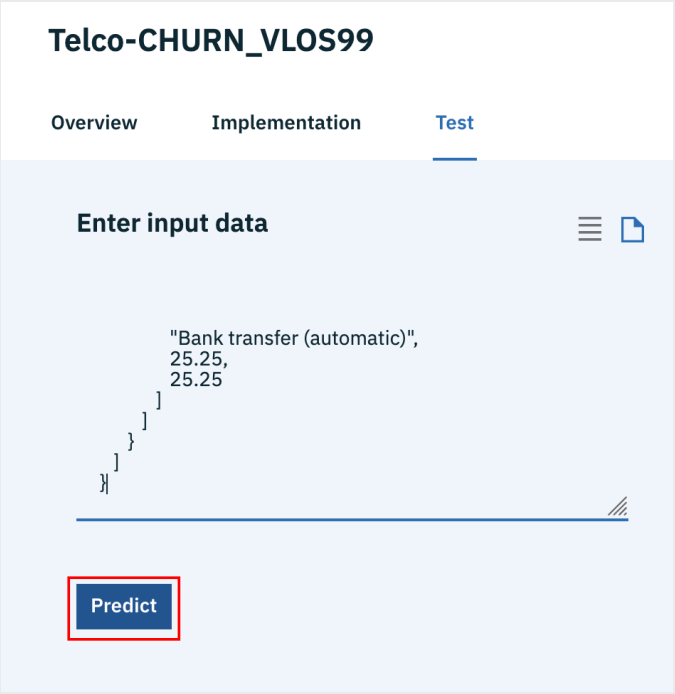


2. Una vez dentro de la página de nuestro despliegue, damos clic en la pestaña “**Test**” y seleccionamos la opción “**Provide imput data as JSON**”(📄).



3. Ahora, en el apartado “**Paste the request payload here**”, pegamos lo siguiente:

```
{
   "fields":[
      "gender",
      "SeniorCitizen",
      "Partner",
      "Dependents",
      "tenure",
      "PhoneService",
      "MultipleLines",
           "InternetService",
           "OnlineSecurity",
           "OnlineBackup",
           "DeviceProtection",
           "TechSupport",
           "StreamingTV",
           "StreamingMovies",
           "Contract",
           "PaperlessBilling",
           "PaymentMethod",
           "MonthlyCharges",
           "TotalCharges"
     ],
     "values":[
        [
           "Female",
           0,
           "No",
           "No",
           1,
           "No",
           "No phone service",
           "DSL",
           "No",
           "No",
           "No",
           "No",
           "No",
           "No",
           "Month-to-month",
           "No",
           "Bank transfer (automatic)",
           25.25,
           25.25
        ]
     ]
  }
```

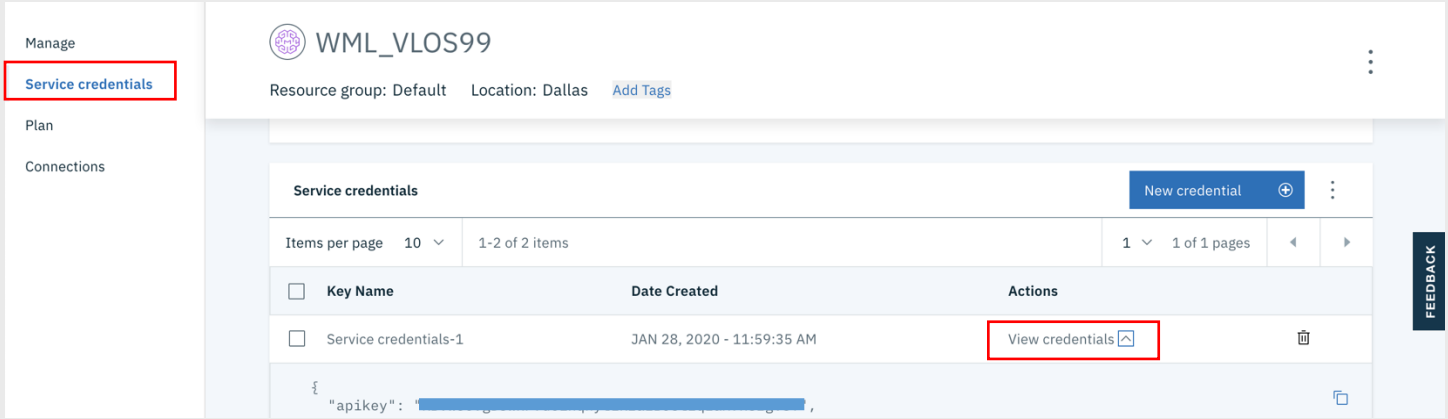


Y damos clic en “**Predict**”.

{"fields":["gender","SeniorCitizen","Partner","Dependents","tenure","PhoneService","MultipleLines","InternetService","OnlineSecurity","OnlineBackup","DeviceProtection","TechSupport","StreamingTV","StreamingMovies","Contract","PaperlessBilling","PaymentMethod","MonthlyCharges","TotalCharges"],"values":[["Female",0,"No","No",1,"No","No phone service","DSL","No","No","No","No","No","No","Month-to-month","No","Bank transfer (automatic)",25.25,25.25]]}

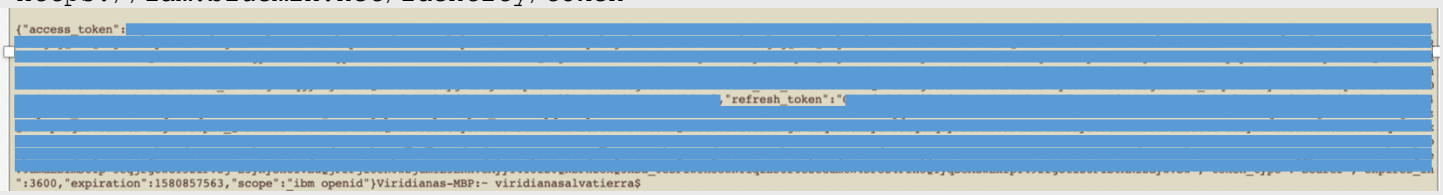
4. Ahora, Podemos visualizar el resultado de nuestra predicción en formato JSON. Al final, debemos poder ver un “**Yes**” o un “**No**” como valor correspondiente al “**CHURN**”.

5. Para poder probarlo realizando un “**Post**” desde la CMD, Terminal, PowerShell, etc., primero debemos obtener la apikey de nuestro servicio de “Watson Machine Learning” desde la página de nuestra instancia. En el apartado de “**Service Credentials**”, desplegamos las credenciales creadas anteriormente y copiamos la apikey.

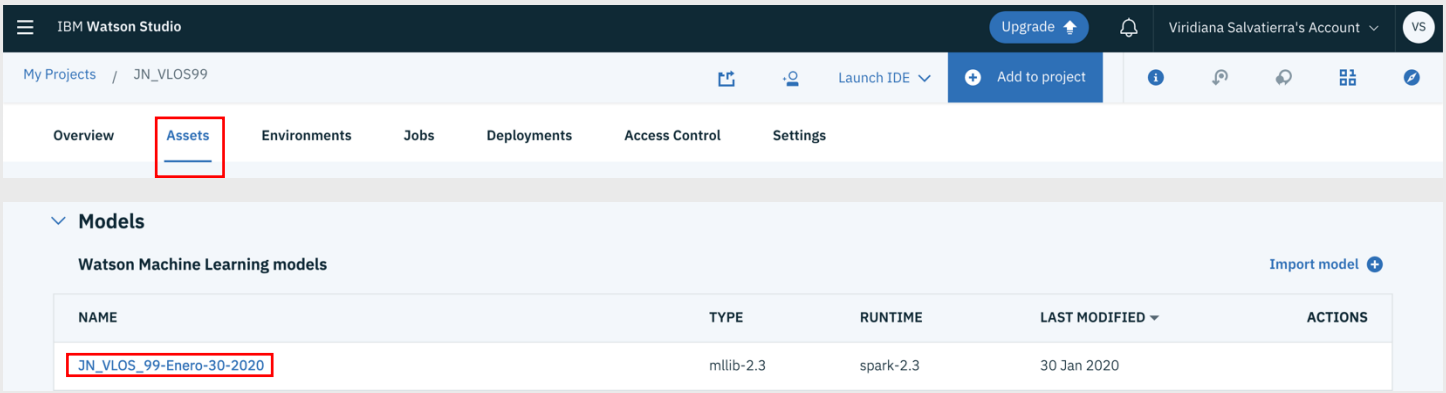


6. Ahora debemos obtener un Token pegando lo siguiente en la CMD, la Terminal, PowerShell (modo administrador).

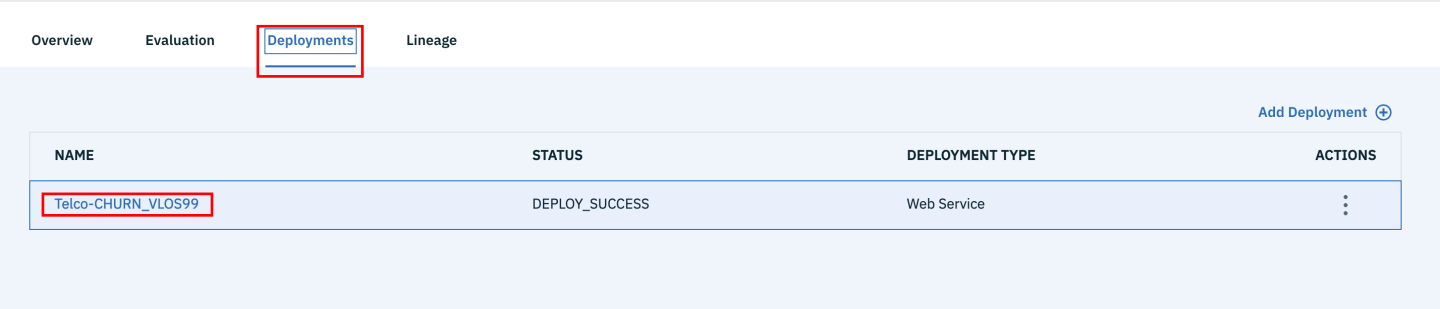
```
curl -k -X POST \
--header "Content-Type: application/x-www-form-urlencoded" \
--header "Accept: application/json" \
--data-urlencode "grant_type=urn:ibm:params:oauth:grant-type:apikey" \
--data-urlencode "apikey=(escribe aquí la clave de apikey sin espacios ni paréntesis)" \
"https://iam.bluemix.net/identity/token"
```



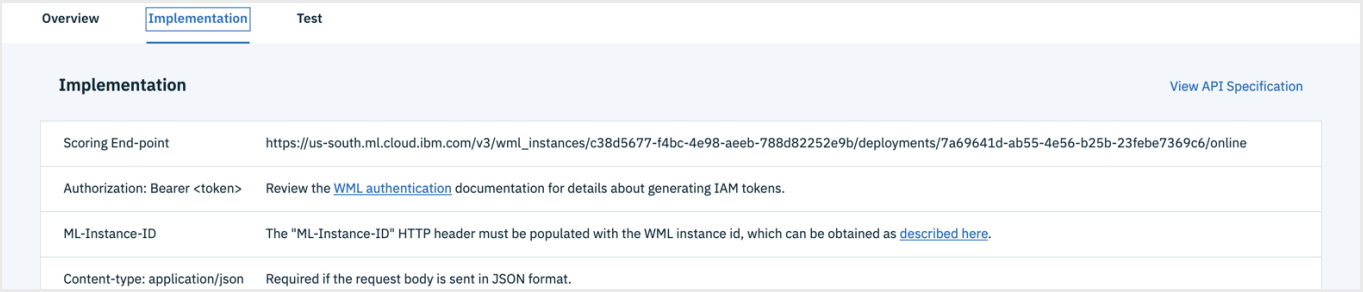
7. Ahora obtenemos nuestro URL desde la página del despliegue de nuestro proyecto. Dentro de la página de nuestro proyecto, en la pestaña de “**Assets**”, damos clic en el nombre de nuestro modelo (desde el apartado “**Models**”).



8. Una vez dentro de nuestro modelo, en la pestaña “**Deployments**”, damos clic en el mobre de nuestro despliegue.



9. Una vez dentro de muestro despliegue, copiamos la dirección de URL.



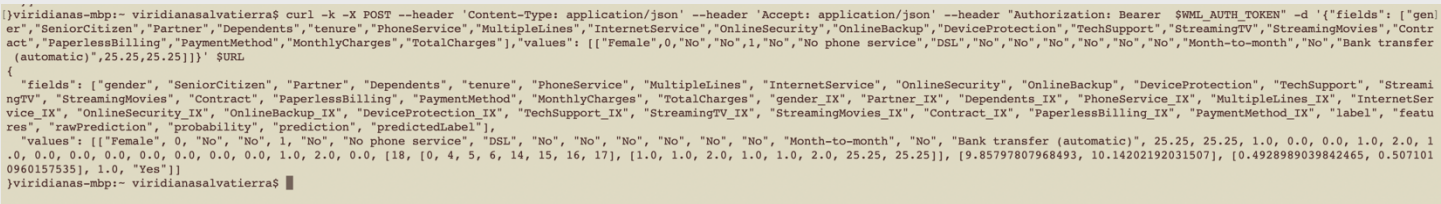
10. Ahora, reemplazamos el access token resultante y la URL en donde corresponde e introducimos los comandos siguientes en la Terminal, CMD, etc.:

```
export WML_AUTH_TOKEN=<sustituir por "access token" sin espacios ni flechas>
export URL=https://<sustituir por el URL de WML sin espacios ni flechas>
```

11. Una vez que hemos exportado estos valores, realizamos la consulta introduciendo lo siguiente en nuestra Terminal, CMD, etc.:

```
curl -k -X POST --header 'Content-Type: application/json' --header 'Accept:
application/json' --header "Authorization: Bearer  $WML_AUTH_TOKEN" -d '{"fields":
["gender","SeniorCitizen","Partner","Dependents","tenure","PhoneService","MultipleLine
s","InternetService","OnlineSecurity","OnlineBackup","DeviceProtection","TechSupport",
"StreamingTV","StreamingMovies","Contract","PaperlessBilling","PaymentMethod","Monthly
Charges","TotalCharges"],"values": [["Female",0,"No","No",1,"No","No phone
service","DSL","No","No","No","No","No","No","Month-to-month","No","Bank transfer
(automatic)",25.25,25.25]]}' $URL
```

12. Hecho esto, obtendremos el resultado de nuestra consulta como un "**YES**" o "**NO**" al final de nuestros resultados.



11. Desplegar la App:

1. Antes de empezar, es muy importante haber clonado el repositorio requerido en los prerrequisitos, además de tener instalado Python 3.
2. Accedemos a la carpeta (descomprimida) en la Terminal, CMD, etc.

```
viridianas-mbp:GitHub viridianasalvatierra$ cd CHURN-app-Jupyter-Notebook-y-WS
viridianas-mbp:CHURN-app-Jupyter-Notebook-y-WS viridianasalvatierra$ ls
CONTRIBUTING.md LICENSE         MAINTAINERS.md  README.md       data            doc             examples        flaskapp        notebooks
viridianas-mbp:CHURN-app-Jupyter-Notebook-y-WS viridianasalvatierra$
```

3. Ahora, creamos nuestro ambiente (“enviroment”).

```
python -m venv venv

  source venv/bin/activate   (Mac / Linux)
  ./venv/Scripts/activate    (Windows)
```

4. Instalamos los requerimientos de Python.

```
cd flaskapp
pip install -r requirements.txt
```

5. Ahora, editamos el archivo “env.sample”, le añadimos nuestras credenciales y lo guardamos como “.env” (**Si llegas a tener errores al correr la app, vuelve a sacar el token y cámbialo en el archivo “.env”, pues es temporal**).

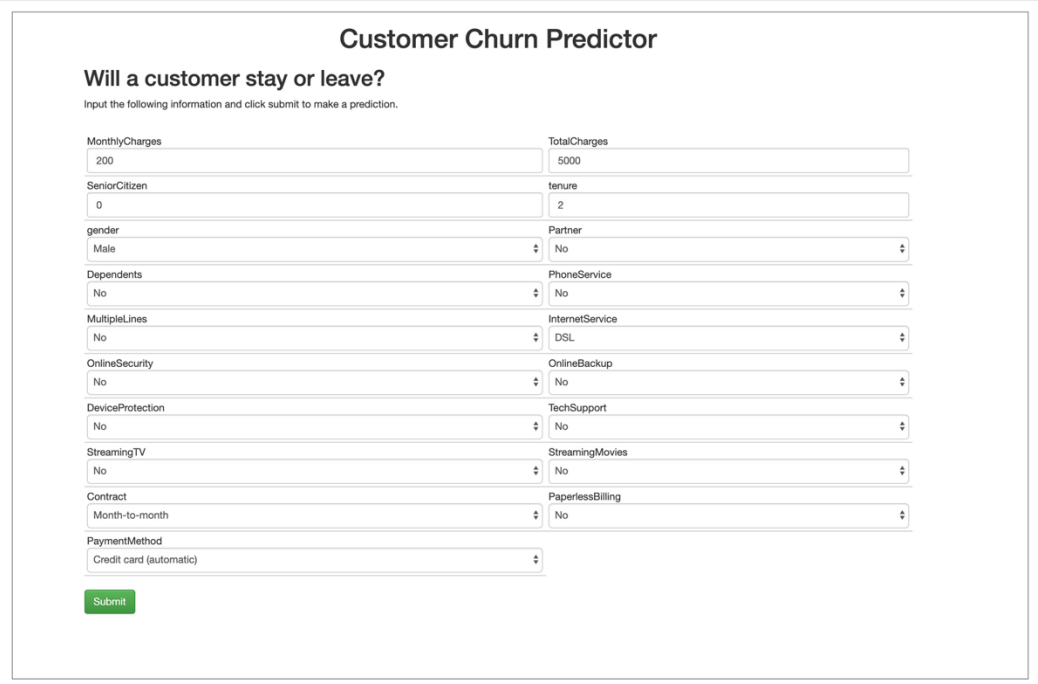
```
# Copy this file to .env.
# Edit the .env file with the required settings before starting the app.

# Required: Provide your web service URL for scoring.
# E.g., URL=https://9.10.222.3:31843/dmodel/v1/project/pyscript/tag/score

URL=(aquí pegamos el URL que obtuvimos en el paso 8.7 sin paréntesis)

# Required: Provide your web service deployment access token.
#           This TOKEN will be the part after `accessToken`. So, your
#           json string will look like:
#
{"username":"scottda","role":"Admin","permissions":["administrator","can_provis
ion","manage_catalog","virtualize_transform","access_catalog"],"sub":"scottda",
"iss":"KNOXSSO","aud":"DSX","uid":"1000331001","authenticator":"default","acces
sToken":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2 <snip>
neQ","_messageCode_":"success","message":"success"}
# The value for `TOKEN=` below will be:
#    TOKEN=eyJhbGciOi <snip> neQ

TOKEN=(aquí pegamos el token que obtuvimos en el paso 8.6 sin paréntesis)

# Optional: You can override the server's host and port here.

HOST=0.0.0.0
PORT=5000
```

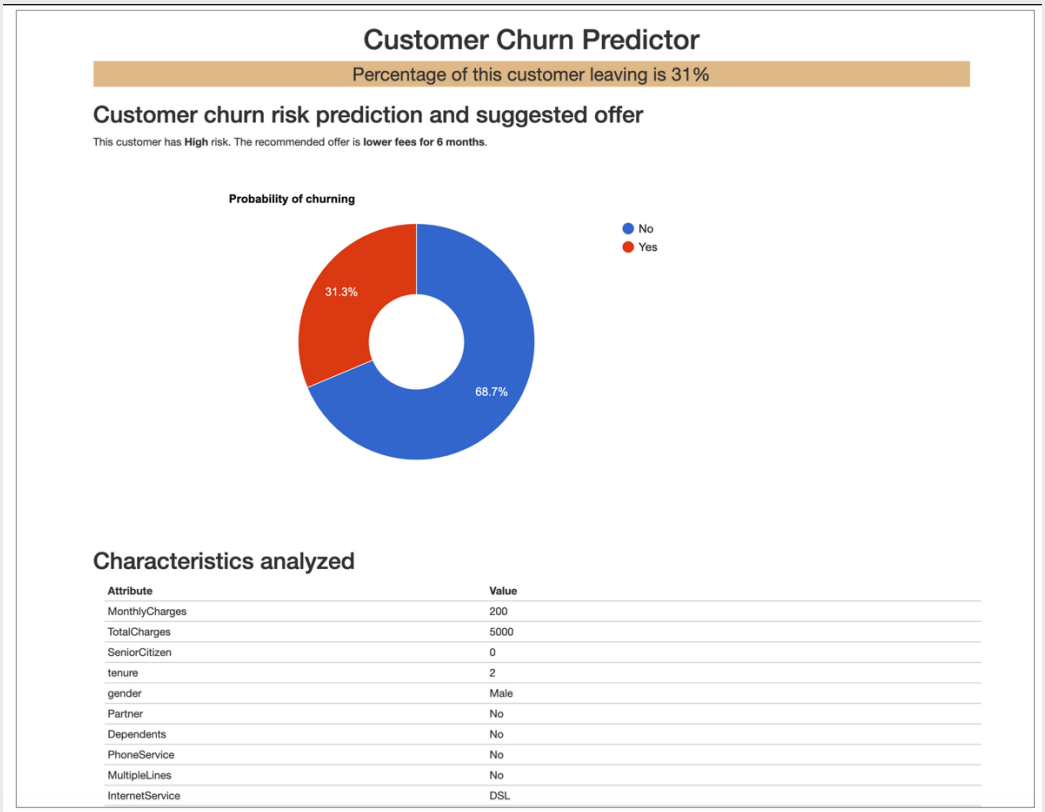
6. Una vez hecho esto, corremos nuestra app con el siguiente comando:

   ```
   python telcochurn.py
   ```

7. Ahora podemos acceder a nuestra app de forma local, pegando en el navegador lo siguiente:

   http://0.0.0.0:5000/





12. Siguientes Pasos

Te invitamos a explorar otros talleres y manuales en el siguiente enlace de Github.
https://github.com/ibmdevelopermx