

# Overview

The Cloud Natural Language API lets you extract entities from text, perform sentiment and syntactic analysis, and classify text into categories. In this lab, we'll focus on text classification. Using a database of 700+ categories, this API feature makes it easy to classify a large dataset of text.

## What you'll learn

- Creating a Natural Language API request and calling the API with curl
- Using the NL API's text classification feature
- Using text classification to understand a dataset of news articles

## What you'll need

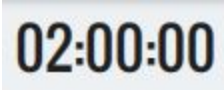
- A Google Cloud Platform Project
- A Browser, such [Chrome](#) or [Firefox](#)

# Setup and Requirements

## Qwiklabs setup

For each lab, you get a new GCP project and set of resources for a fixed time at no cost.

1. Make sure you signed into Qwiklabs using an **incognito window**.

2. Note the lab's access time (for example,  and make sure you can finish in that time block.


There is no pause feature. You can restart if needed, but you have to start at the beginning.


3. When ready, click  .


4. Note your lab credentials. You will use them to sign in to Cloud Platform Console.

Open Google Console

**Caution:** When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

**Username**  
google2876526\_student@qwiklabs.n 

**Password**  
TG959yrKDX 

**GCP Project ID**  
qwiklabs-gcp-0855e773352d3560 

[New to labs? View our introductory video!](#)

5. Click **Open Google Console**.
6. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

If you use other credentials, you'll get errors or **incur charges**.

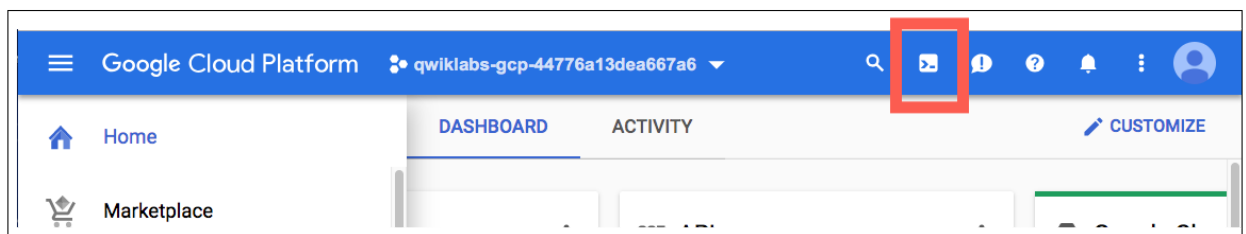
7. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you are finished with the lab or want to restart it. This clears your work and removes the project.

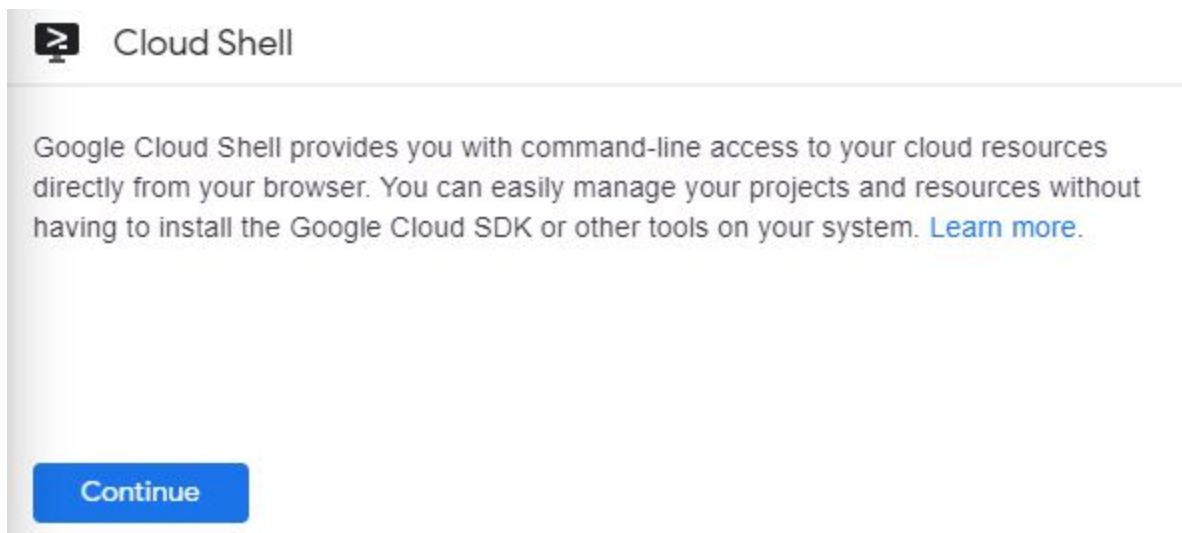
# Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Google Cloud Shell provides command-line access to your GCP resources.

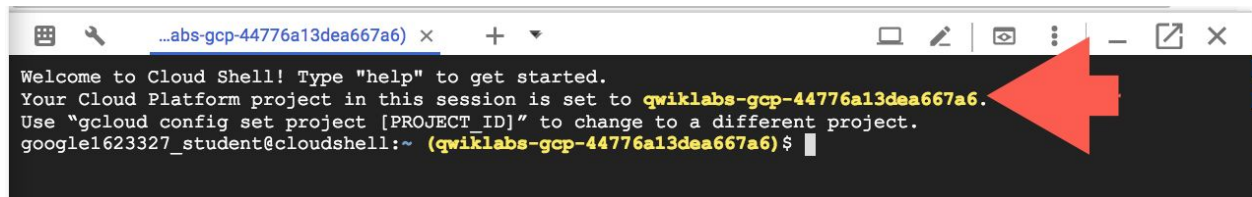
1. In GCP console, on the top right toolbar, click the Open Cloud Shell button.



2. Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT\_ID*. For example:



```
...abs-gcp-44776a13dea667a6) x + -
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6) $
```

**gcloud** is the command-line tool for Google Cloud Platform. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

Output:

```
Credentialed accounts:
- <myaccount>@<mydomain>.com (active)
```

Example output:

```
Credentialed accounts:
- google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

Output:

```
[core]
project = <project_ID>
```

Example output:

```
[core]
project = quiklabs-gcp-44776a13dea667a6
```

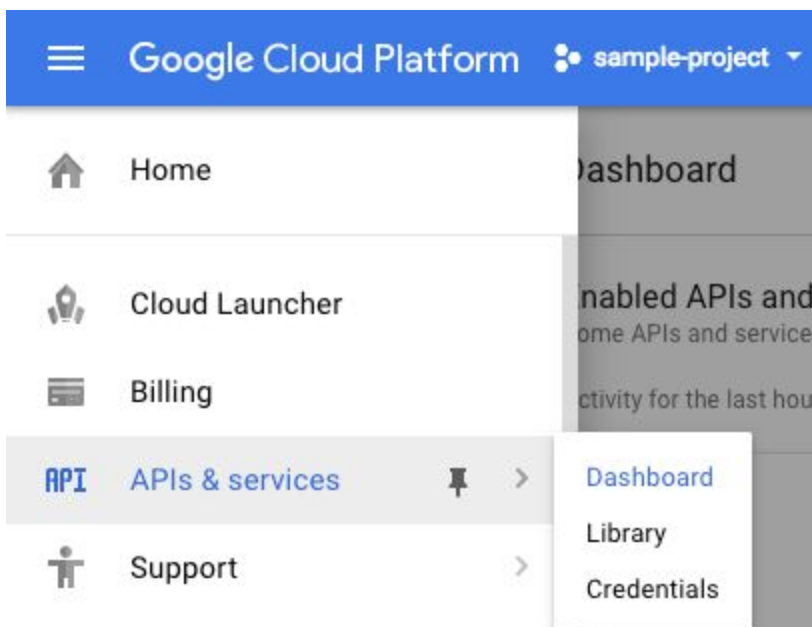
Full documentation of **gcloud** is available on [Google Cloud gcloud Overview](#) .

## Task 1: Confirm that the Cloud Natural Language API is enabled

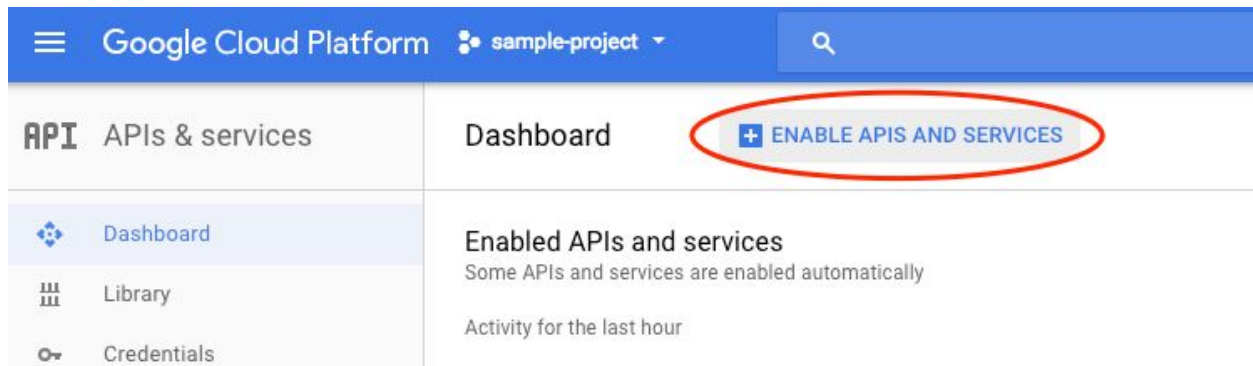
1. Click the menu icon in the top left of the screen.



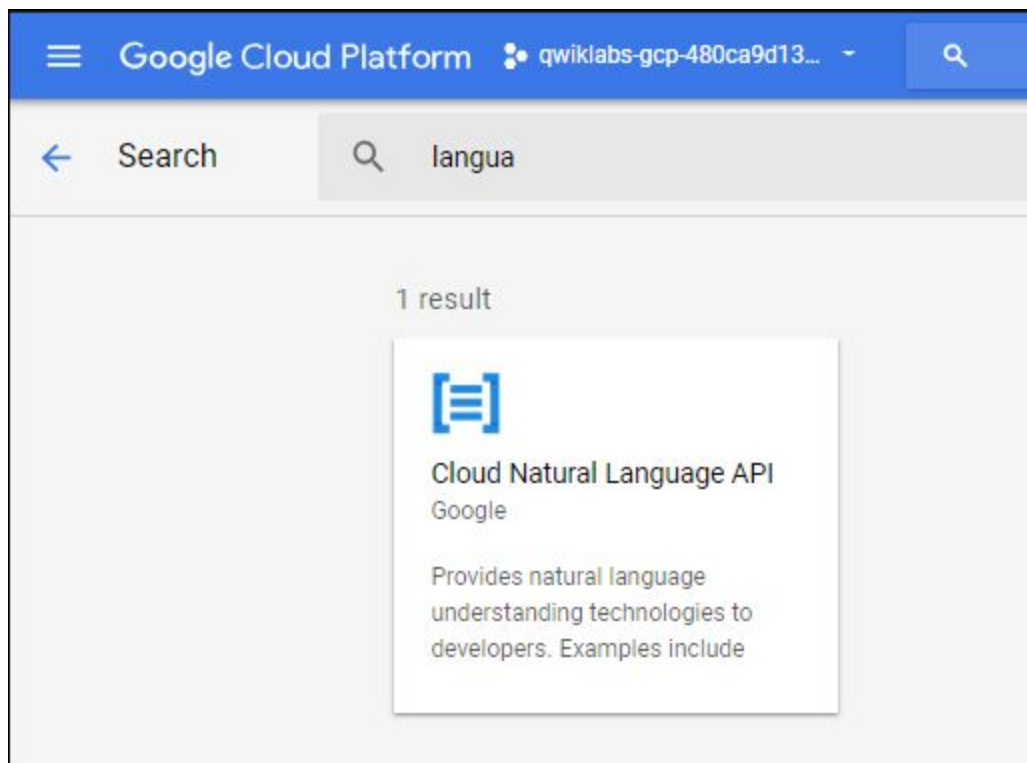
2. Select **APIs & services** > **Dashboard**.



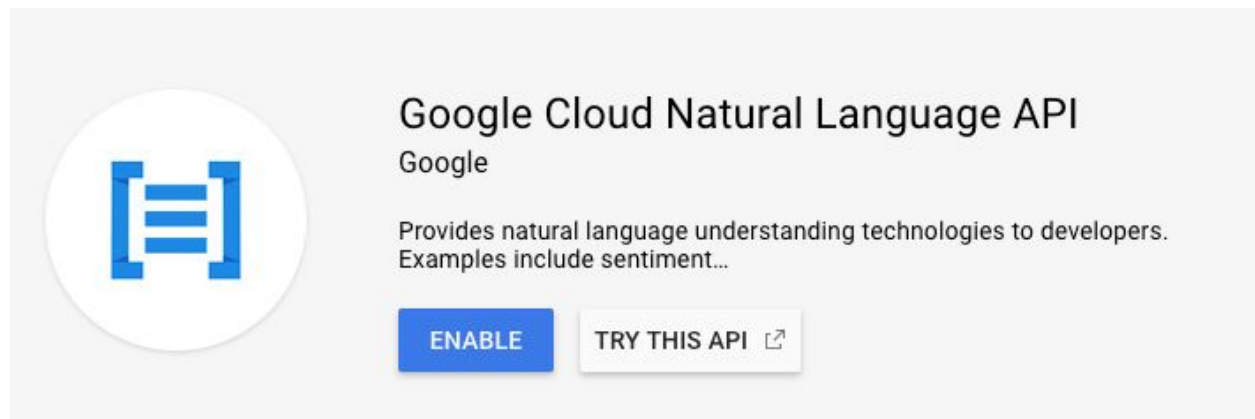
3. Click **Enable APIs and services**.



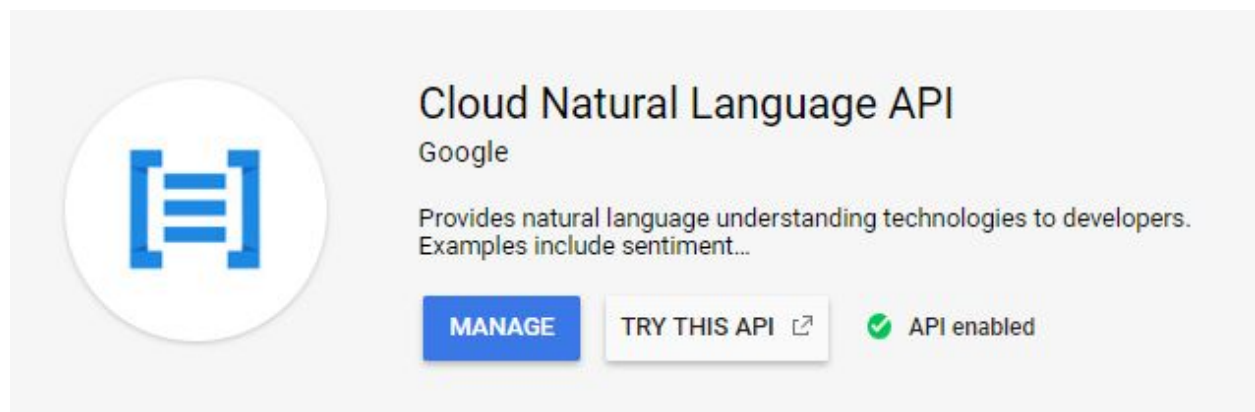
4. Then, search for "language" in the search box. Click **Google Cloud Natural Language API**:



5. If the API is not enabled, you'll see the **Enable** button. Click **Enable** to enable the Cloud Natural Language API:



When the API is enabled, GCP displays API information as follows:

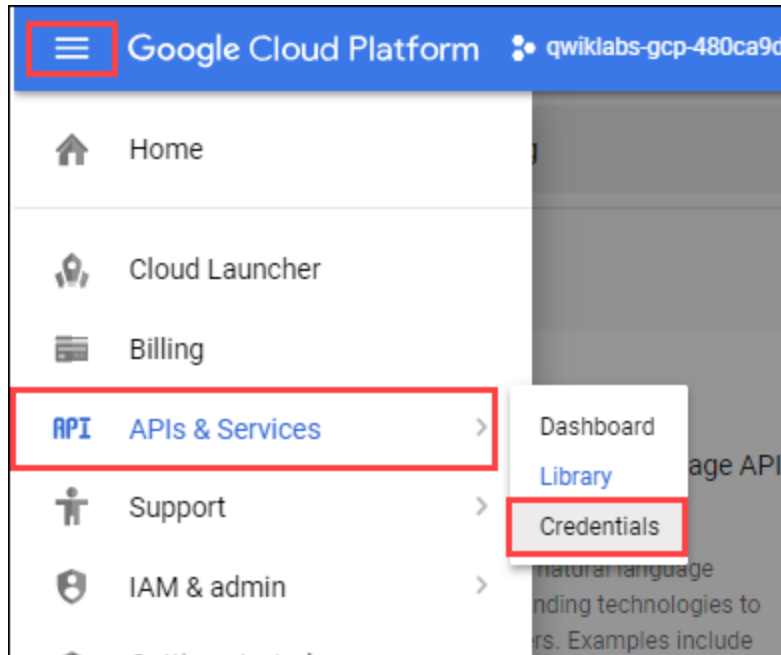


## Task 2: Create an API Key

Since you're using `curl` to send a request to the Natural Language API, you need to generate an API key to pass in the request URL.



1. To create an API key, in your Console, click **Navigation menu > APIs & services > Credentials**:



2. Then click **Create credentials**:
3. In the drop down menu, select **API key**:

[+ CREATE CREDENTIALS](#)

 [DELETE](#)

### API key

Identifies your project using a simple API key to check quota and access

### OAuth client ID

Requests user consent so your app can access the user's data

### Service account

Enables server-to-server, app-level authentication using robot accounts

### Help me choose

Asks a few questions to help you decide which type of credential to use

- Next, copy the key you just generated. Then click **Close**.

Now that you have an API key, save it to an environment variable to avoid having to insert the value of your API key in each request.

- In Cloud Shell run the following. Be sure to replace `<your_api_key>` with the key you just copied.

```
export API_KEY=<YOUR_API_KEY>
```

## Task 3: Classify a news article

Using the Natural Language API's `classifyText` method, you can sort text data into categories with a single API call. This method returns a list of content categories that apply to a text document. These categories range in specificity, from broad categories like `/Computers & Electronics` to highly specific categories such as `/Computers & Electronics/Programming/Java (Programming Language)`. A full list of 700+ possible categories can be found [here](#).

We'll start by classifying a single article, and then we'll see how we can use this method to make sense of a large news corpus. To start, let's take this headline and description from a New York Times article in the food section:

*A Smoky Lobster Salad With a Tapa Twist. This spin on the Spanish pulpo a la gallega skips the octopus, but keeps the sea salt, olive oil, pimentón and boiled potatoes.*

1. In your Cloud Shell environment, create a `request.json` file with the code below. You can either create the file using one of your preferred command line editors (nano, vim, emacs) or use the Cloud Shell code editor:



2. Create a new file named `request.json` and add the following:

```
{  
  "document": {
```

```
"type": "PLAIN_TEXT",
"content": "A Smoky Lobster Salad With a Tapa Twist. This spin on the
Spanish pulpo a la gallega skips the octopus, but keeps the sea salt, olive
oil, pimentón and boiled potatoes."
}
```

3. Now, you can send this text to the Natural Language API's `classifyText` method with the following `curl` command:

```
curl
https://language.googleapis.com/v1/documents:classifyText?key=${API_KEY}" \
-s -X POST -H "Content-Type: application/json" --data-binary @request.json
```

Look at the response:

Output (do not copy)

```
{ categories:
[
{
name: '/Food & Drink/Cooking & Recipes',
confidence: 0.85
},
{
name: '/Food & Drink/Food/Meat & Seafood',
confidence: 0.63
}
]
```

The API returned 2 categories for this text:

- /Food & Drink/Cooking & Recipes
- /Food & Drink/Food/Meat & Seafood

The text doesn't explicitly mention that this is a recipe or even that it includes seafood, but the API is able to categorize it. Classifying a single article is cool, but to really see the power of this feature, let's classify lots of text data.

## Task 4: Classifying a large text dataset

To see how the `classifyText` method can help us understand a dataset with lots of text, you'll use this [public dataset](#) of BBC news articles. The dataset consists of 2,225 articles in five topic areas (business, entertainment, politics, sports, tech) from 2004 - 2005. A subset of these articles are in a public Google Cloud Storage bucket. Each of the articles is in a `.txt` file.

To examine the data and send it to the Natural Language API, you'll write a Python script to read each text file from Cloud Storage, send it to the `classifyText` endpoint, and store the results in a BigQuery table. BigQuery is Google Cloud's big data warehouse tool - it lets you easily store and analyze large data sets.

1. To see the type of text you'll be working with, run the following command to view one article (`gsutil` provides a command line interface for Cloud Storage):

```
gsutil cat gs://cloud-training-demos-text/bbc_dataset/entertainment/001.txt
```

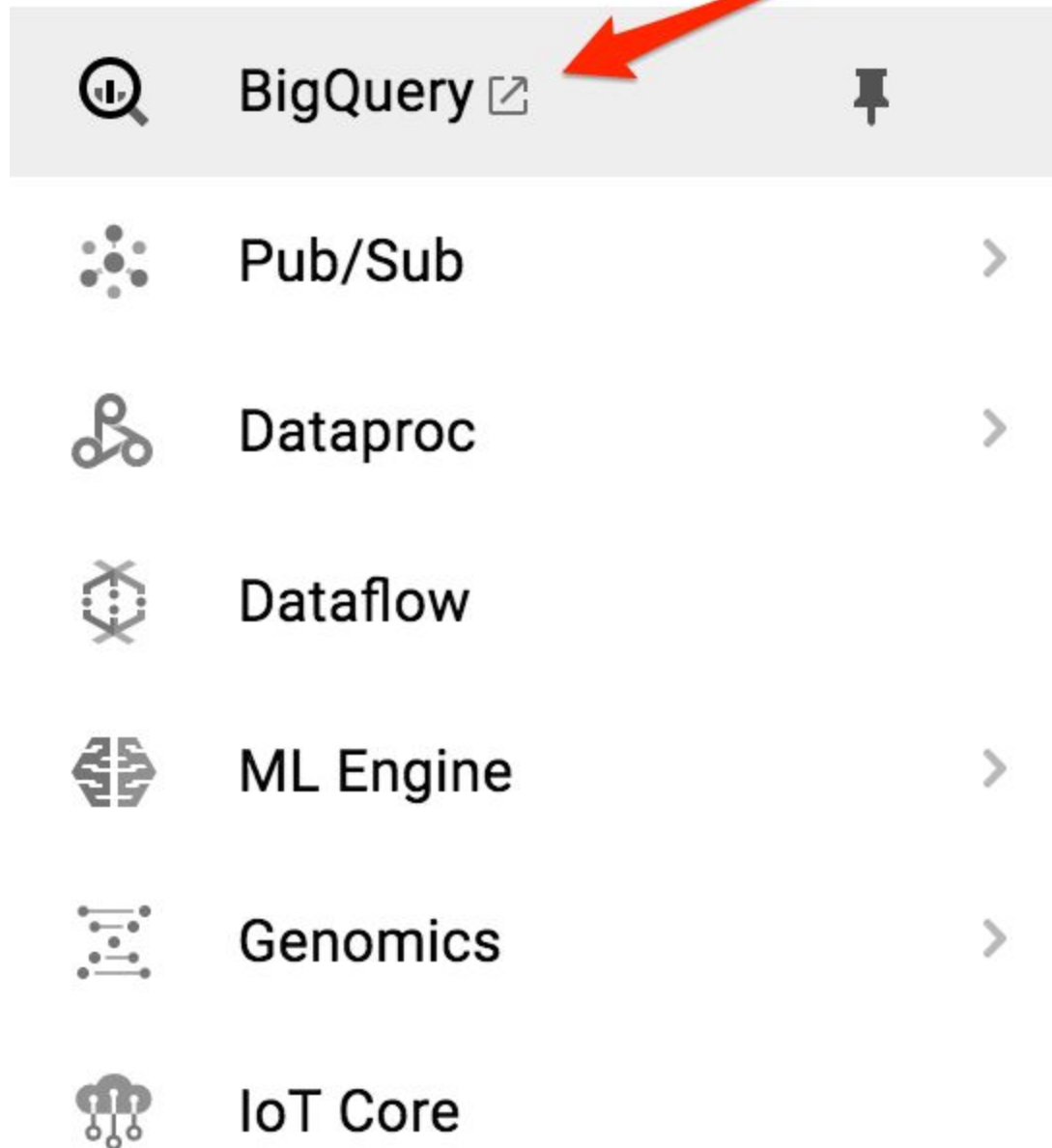
Next you'll create a BigQuery table for your data.

## **Task 5: Creating a BigQuery table for our categorized text data**

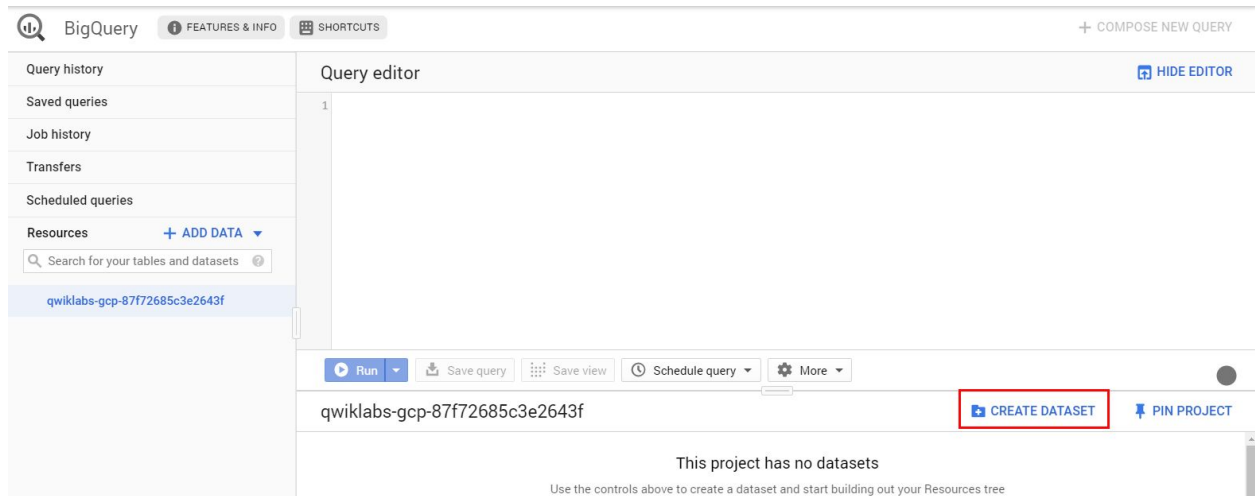
Before sending the text to the Natural Language API, you need a place to store the text and category for each article.

1. Navigate to the BigQuery under Big Data in the Console Menu. Accept the welcome notice when launching BigQuery.

## BIG DATA



2. Then click on the name of your project, then **Create dataset**:



3. For **Name**, type `news_classification_dataset`
4. Click **Create dataset**.
5. Click on the name of the dataset, then select **Create table**. Use the following settings for the new table:
  - Create From: **empty table**
  - Name your table **article\_data**
  - Click **Add Field** and add the following 3 fields: **article\_text** with type **STRING**, **category** with type **STRING**, and **confidence** with type **FLOAT**.



## Create table

### Source

Create table from:

Empty table ▼

### Destination

Project name

qwiklabs-gcp-881767e307f47f86 ▼

Dataset name

news\_classification\_dataset ▼

Table type ?

Native table ▼

Table name

article\_data

### Schema

☐ Edit as text

Name	Type	Mode	
article_text	STRING ▼	NULLABLE ▼	×
category	STRING ▼	NULLABLE ▼	×
confidence	FLOAT ▼	NULLABLE ▼	×
<div>+ Add field</div>			

### Partition and cluster settings

Partitioning: ?

No partitioning ▼

Clustering order (optional): ?

Clustering order determines the sort order of the data. Clustering can only be used on a partitioned table, and works with tables partitioned either by column or ingestion time.

Comma-separated list of fields to define clustering order (up to 4)

Advanced options ▼

Create table

Cancel

6. Click **Create Table**.

The table is empty right now. In the next step you'll read the articles from Cloud Storage, send them to the Natural Language API for classification, and store the result in BigQuery.

## Task 6: Classifying news data and storing the result in BigQuery

Before writing a script to send the news data to the Natural Language API, you need to create a service account. This will be used to authenticate to the Natural Language API and BigQuery from a Python script.

1. First, back in Cloud Shell, export the name of your Cloud project as an environment variable. Replace `<your_project_name>` with the **GCP Project ID** found in the **CONNECTION DETAILS** section of the lab.

```
export PROJECT=<your_project_name>
```

2. Then run the following commands from Cloud Shell to create a service account:

```
gcloud iam service-accounts create my-account --display-name my-account
```

```
gcloud projects add-iam-policy-binding $PROJECT
--member=serviceAccount:my-account@$PROJECT.iam.gserviceaccount.com
--role=roles/bigquery.admin
gcloud iam service-accounts keys create key.json
--iam-account=my-account@$PROJECT.iam.gserviceaccount.com
export GOOGLE_APPLICATION_CREDENTIALS=key.json
```

Now you're ready to send the text data to the Natural Language API!

To do that, write a Python script using the Python module for Google Cloud. You can accomplish the same thing from any language, there are many different cloud client libraries.

3. Create a file called `classify-text.py` and copy the following into it.

Replace `YOUR_PROJECT` with your **GCP Project ID** (**NOTE:** Please leave the single quotes around the Project ID value in place).

```
from google.cloud import storage, language, bigquery

# Set up our GCS, NL, and BigQuery clients
storage_client = storage.Client()
nl_client = language.LanguageServiceClient()
# TODO: replace YOUR_PROJECT with your project id below
bq_client = bigquery.Client(project='YOUR_PROJECT')

dataset_ref = bq_client.dataset('news_classification_dataset')
dataset = bigquery.Dataset(dataset_ref)
table_ref = dataset.table('article_data') # Update this if you used a
different table name
table = bq_client.get_table(table_ref)

# Send article text to the NL API's classifyText method
def classify_text(article):
    response = nl_client.classify_text(
        document=language.types.Document(
            content=article,
            type=language.enums.Document.Type.PLAIN_TEXT
```

```

    )
    )
    return response

rows_for_bq = []
files = storage_client.bucket('cloud-training-demos-text').list_blobs()
print("Got article files from GCS, sending them to the NL API (this will take
~2 minutes)...")

# Send files to the NL API and save the result to send to BigQuery
for file in files:
    if file.name.endswith('txt'):
        article_text = file.download_as_string()
        nl_response = classify_text(article_text)
        if len(nl_response.categories) > 0:
            rows_for_bq.append((str(article_text),
str(nl_response.categories[0].name),
str(nl_response.categories[0].confidence)))

print("Writing NL API article data to BigQuery...")
# Write article text + category data to BQ
errors = bq_client.insert_rows(table, rows_for_bq)
assert errors == []

```

Now you're ready to start classifying articles and importing them to BigQuery.

#### 4. Run the following script:

```
python3 classify-text.py
```

The script takes about two minutes to complete, so while it's running let's discuss what's happening.

**Note:** If you get an error while executing `python3 classify-text.py`, it might be the case the cloud shell get disconnected. In order to fix that, please export your environment variables by running the below commands then re-run the `python3 classify-text.py` command.

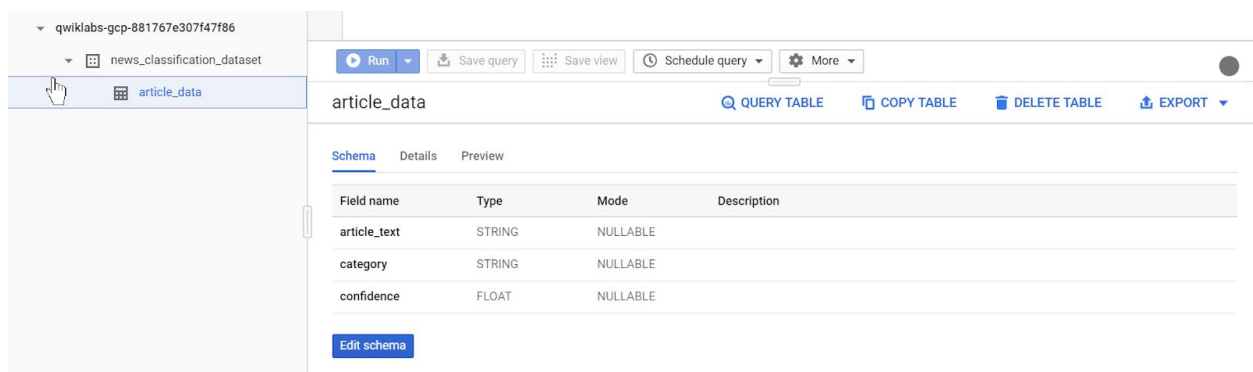
```
export PROJECT= (GCP PROJECT ID) export
GOOGLE_APPLICATION_CREDENTIALS=key.json
```

We're using the google-cloud [Python client library](#) to access Cloud Storage, the Natural Language API, and BigQuery. First, a client is created for each service; then references are created to the BigQuery table. `files` is a reference to each of the BBC dataset files in the public bucket. We iterate through these files, download the articles as strings, and send each one to the Natural Language API in our `classify_text` function. For all articles where the Natural Language API returns a category, the article and its category data are saved to a `rows_for_bq` list. When classifying each article is done, the data is inserted into BigQuery using `insert_rows()`.

Note: The Natural Language API can return more than one category for a document, but for this lab you're only storing the first category returned to keep things simple.

When the script has finished running, it's time to verify that the article data was saved to BigQuery.

5. In BigQuery, navigate to the `article_data` table in the BigQuery tab and click **Query Table**:



6. Edit the results in the **Unsaved query** box, adding an asterisk between SELECT and FROM:

```
SELECT * FROM `news_classification_dataset.article_data`
```

7. Now click **Run**.

You will see your data when the query completes. Scroll to the right to see the category column.

The category column has the name of the first category the Natural Language API returned for the article, and confidence is a value between 0 and 1 indicating how confident the API is that it categorized the article correctly. You'll learn how to perform more complex queries on the data in the next step.

## Task 7: Analyzing categorized news data in BigQuery

First, see which categories were most common in the dataset.



1. In the BigQuery console, click **Compose New Query**.
2. Enter the following query:

```
SELECT  
  category,  
  COUNT(*) c  
FROM  
  `news_classification_dataset.article_data`
```

```
GROUP BY
category
ORDER BY
c DESC
```

3. Now click **Run**.

You should see something like this in the query results:

Query results			 SAVE RESULTS ▼	 EXPLORE IN DATA STUDIO
Query complete (1.1 sec elapsed, 0 B processed)				
Job information <u>Results</u> JSON   Execution details				
Row	catagory	c		
1	/Arts & Entertainment/Movies	60		
2	/News/Politics	41		
3	/Business & Industrial	30		
4	/Sports/Individual Sports/Track & Field	25		
5	/Sports/Individual Sports	25		

4. If you wanted to find the article returned for a more obscure category like /Arts & Entertainment/Music & Audio/Classical Music, you could run the following query:

```
SELECT * FROM `news_classification_dataset.article_data`
WHERE category = "/Arts & Entertainment/Music & Audio/Classical Music"
```

5. To get only the articles where the Natural language API returned a confidence score greater than 90%, run the following query:

```
SELECT
  article_text,
  category
FROM `news_classification_dataset.article_data`
WHERE cast(confidence as float64) > 0.9
```

To perform more queries on your data, explore the [BigQuery documentation](#).

BigQuery also integrates with a number of visualization tools. To create visualizations of your categorized news data, check out the [Data Studio quickstart](#) for BigQuery.