https://learning.oreilly.com/videos/building-ai-applications/9780135973462

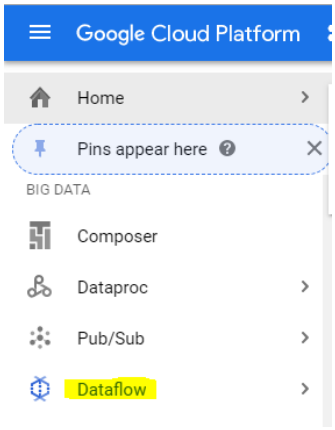
Lesson 1 – Shell tutorial

```
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ gcloud
        ERROR: (gcloud) Command name argument expected.
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ gcloud auth list
        Credentialed Accounts
        ACTIVE ACCOUNT
        * a.siddikov2@gmail.com
        To set the active account, run:
        $ gcloud config set account `ACCOUNT`
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ gcloud config list project
      [core]
      project = introduction-ai-291400
      Your active configuration is: [cloudshell-164]
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ echo $HOME
        /home/a_siddikov2
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ env
        SHELL=/bin/bashUSE_CLOUD_SDK_PYTHON3=true…
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ ls -al
      total 60
      drwxr-xr-x 7 a_siddikov2 a_siddikov2  4096 Oct  3 00:22 .
      drwxr-xr-x 4 root        root         4096 Oct  3 00:15 ..
      -rw------- 1 a_siddikov2 a_siddikov2   240 Oct  3 04:28 .bash_history
      -rw-r--r-- 1 a_siddikov2 a_siddikov2   220 Apr 18  2019 .bash_logout
      -rw-r--r-- 1 a_siddikov2 a_siddikov2  3564 Sep 29 21:45 .bashrc
      drwxr-xr-x 3 a_siddikov2 a_siddikov2  4096 Oct  3 00:22 .cache
      drwxr-xr-x 3 a_siddikov2 a_siddikov2  4096 Sep 29 21:26 .config
      drwxr-xr-x 2 a_siddikov2 a_siddikov2  4096 Oct  3 00:15 .docker
      -rw-r--r-- 1 a_siddikov2 a_siddikov2   807 Apr 18  2019 .profile
      -rw-r--r-- 1 a_siddikov2 a_siddikov2   913 Oct  3 04:22 README-cloudshell.txt
      drwxr-xr-x 6 a_siddikov2 a_siddikov2  4096 Oct  3 04:22 .theia
      drwxr-xr-x 3 a_siddikov2 a_siddikov2  4096 Oct  3 00:22 .yarn
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ touch foo.py
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ vim foo.py
   [click i to insert]
   def foo ():
        return 1
   [click esc to exit insert]
   :wq
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ gcloud app create
      Creating App Engine application in project [introduction-ai-291400] and region [us-
         central]....done.
      Success! The app is now created. Please use `gcloud app deploy` to deploy your
         first app.
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ git clone
   https://github.com/googlecloudplatform/python-docs-samples
   Cloning into 'python-docs-samples'...
        remote: Enumerating objects: 33, done.
        remote: Counting objects: 100% (33/33), done.
        remote: Compressing objects: 100% (22/22), done.
        remote: Total 39804 (delta 13), reused 22 (delta 8), pack-reused 39771
        Receiving objects: 100% (39804/39804), 60.35 MiB | 26.26 MiB/s, done.
        Resolving deltas: 100% (22225/22225), done.
a_siddikov2@cloudshell:~ (introduction-ai-291400)$ cd python-docs-
   samples/appengine/standard_python3/hello_world
a_siddikov2@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world
   (introduction-ai-291400)$ virtualenv venv
      created virtual environment CPython3.7.3.final.0-64 in 875ms
        creator CPython3Posix(dest=/home/a_siddikov2/python-docs-
         samples/appengine/standard_python3/hello_world/venv, clear=False, global=False)
```

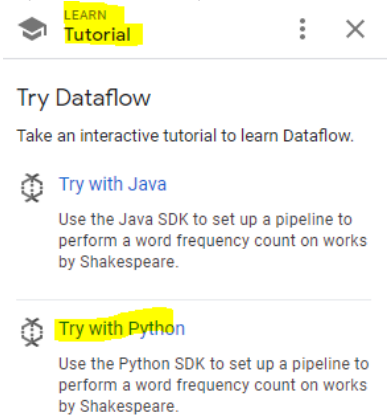
```
        seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle,
         via=copy, app_data_dir=/home/a_siddikov2/.local/share/virtualenv)
          added seed packages: pip==20.2.2, setuptools==49.6.0, wheel==0.35.1
        activators
         BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,
         XonshActivator
a_siddikov2@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world
   (introduction-ai-291400)$ source venv/bin/activate
(venv) a_siddikov2@cloudshell:~/python-docs-
   samples/appengine/standard_python3/hello_world (introduction-ai-291400)$ pip install -
   r requirements.txt
(venv) a_siddikov2@cloudshell:~/python-docs-
   samples/appengine/standard_python3/hello_world (introduction-ai-291400)$ python
   main.py
[if esc does not work, CTL+c]
(venv) a_siddikov2@cloudshell:~/python-docs-
   samples/appengine/standard_python3/hello_world (introduction-ai-291400)$ gcloud app
   deploy
      ╔═══════════════════════════════════════════════════════════════╗
      ╠═ Uploading 704 files to Google Cloud Storage                  ═╣
      ╚═══════════════════════════════════════════════════════════════╝

      File upload done.
      Updating service [default]...done.
      Setting traffic split for service [default]...done.
      Deployed service [default] to [https://introduction-ai-291400.uc.r.appspot.com]

      You can stream logs from the command line by running:
        $ gcloud app logs tail -s default

++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++
```

Lesson 2 - Building ETL Pipelines, Extract Transform Load.

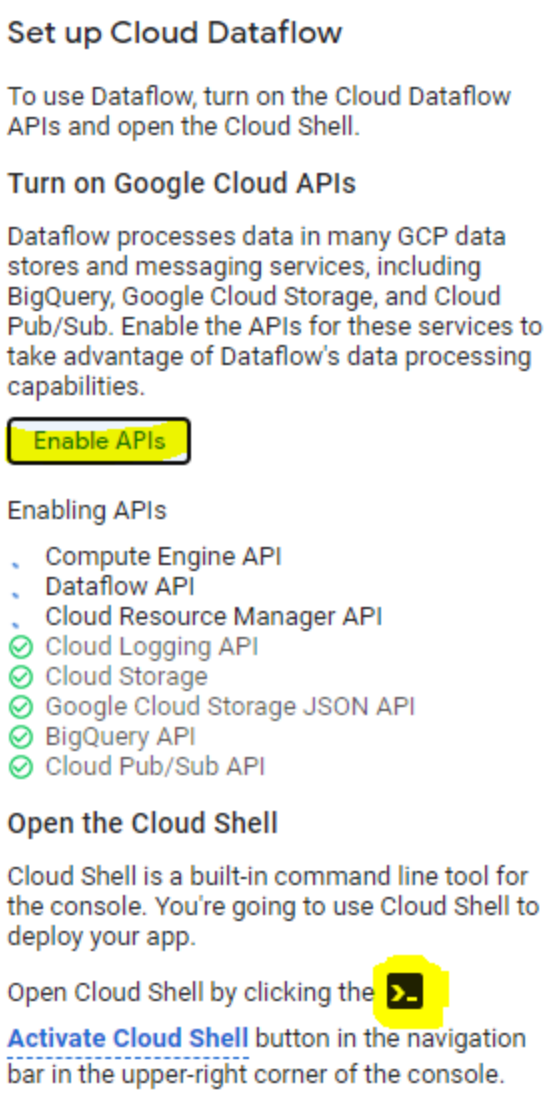
- ETL Overview
    1. Cloud Approach
        - **Cloud Function** and it is server less: Python code can be executed by the following triggers
            - HTTP protocol
            - Google Cloud storage
            - Event in our system
        - **Batch data flow operation**: we will build pipeline which would run at some periodic interval.
        - **Google App Engine (GAE)** and it is server less: it can run CRON jobs. We can use our existing application such as Python skills and schedule it to run. And all we need to do is develop a CRON YAML file.
    2. Manual Approach
        - Non-cloud native approach – manual approach: install Hadoop, Jenkins.
- Data Flow Demo
    - Sign into the GCP
    - Locate Dataflow



- Try tutorial with Python



- Dataflow Word Count Tutorial Introduction: In this tutorial, you'll learn the basics of the Cloud Dataflow service by running a simple example pipeline using the Apache Beam Python SDK. This pipeline will show you the basics of reading a text file from Google Cloud Storage, counting the number of unique words in the file, and finally writing the word counts back to Google Cloud Storage. To see what code we will be running today, you can visit the Apache Beam GitHub repository's example wordcount. Dataflow pipelines are either batch (processing bounded input like a file or database table) or streaming (processing unbounded input from a source like Cloud Pub/Sub). The example in this tutorial is a batch pipeline that counts words in a collection of Shakespeare's works. Before you start, you'll need to check for prerequisites in your Cloud Platform project and perform initial setup.
- Project setup: Google Cloud Platform organizes resources into projects. This allows you to collect all the related resources for a single application in one place. Select a project or create a new one.

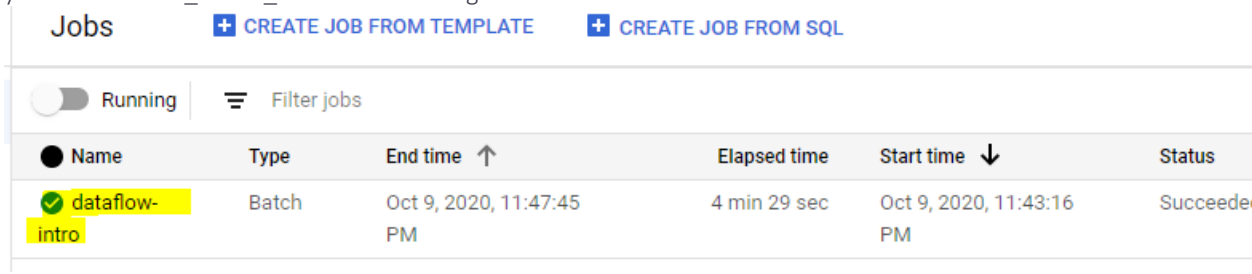


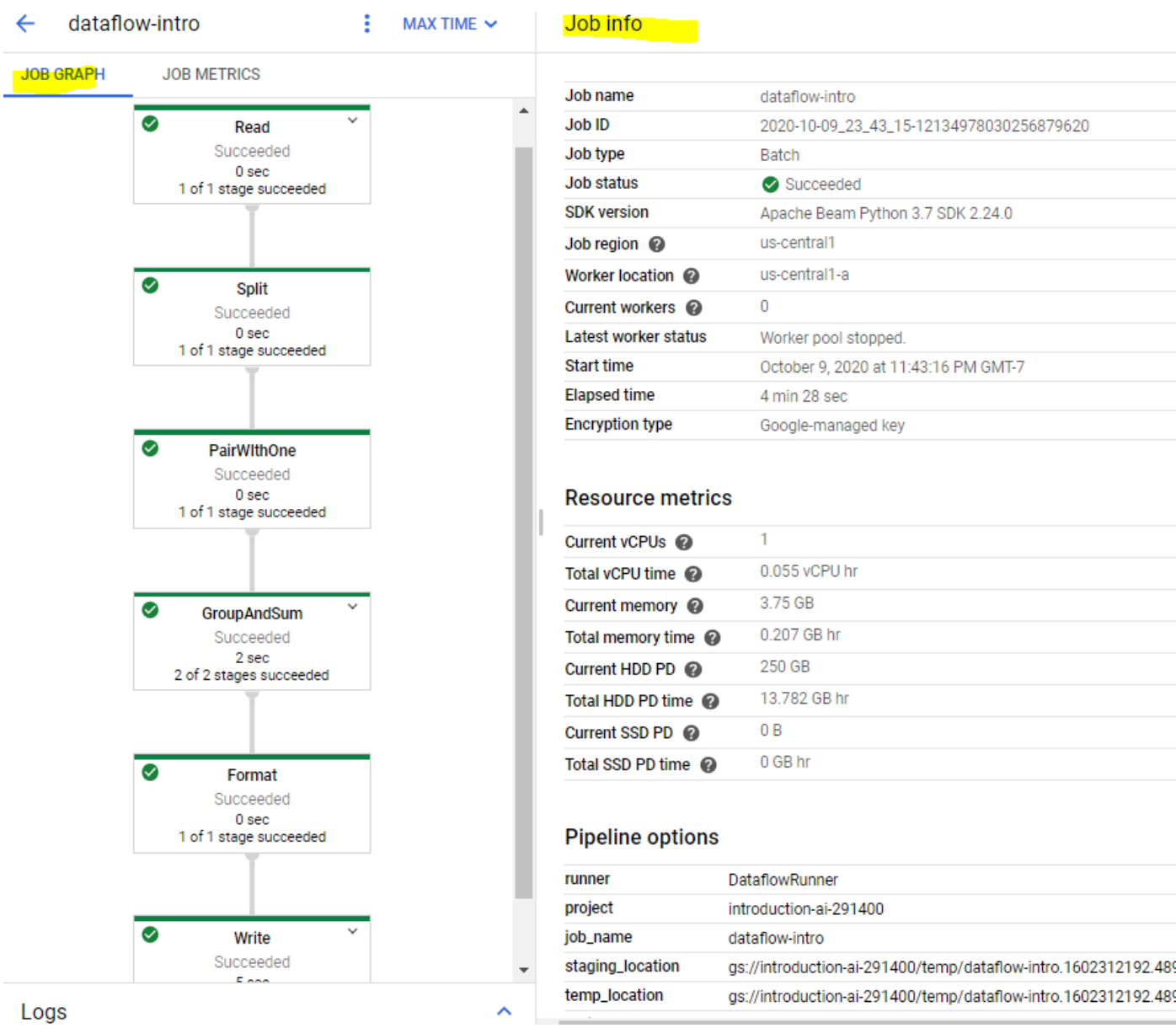
- Install Cloud Dataflow samples on Cloud Shell. Dataflow runs jobs written using the Apache Beam SDK. To submit jobs to the Dataflow Service using python, your development environment will require Python, the Google Cloud SDK, and the Apache Beam SDK for Python. Additionally, Cloud Dataflow uses pip3, python's package manager, to manage SDK dependencies, and virtualenv to create isolated Python environments. This tutorial uses a Cloud Shell that has python and pip3 already installed. If you prefer, you can do this tutorial on your local machine.
- Install virtualenv and activate a Python virtual environment
    - Install virtualenv version 13.1.0 or above if it is not installed already.
      `pip3 install --upgrade virtualenv -user`
    - Create a Python virtual environment
      `python3 -m virtualenv env`
    - and activate it.
      `source env/bin/activate`
- Download the samples and the Apache Beam SDK for Python using the pip3 command. In order to write a Python Dataflow job, you will first need to download the SDK from the repository. When you run this command, pip3 will download and install the appropriate version of the Apache Beam SDK.
  `a_siddikov2@cloudshell:~ (introduction-ai-291400)$ pip3 install --quiet apache-beam[gcp]`
- Set up a Cloud Storage bucket
    - Cloud Dataflow uses Cloud Storage buckets to store output data and cache your pipeline code.
    - Run gsutil mb
    - In Cloud Shell, use the command gsutil mb to create a Cloud Storage bucket.
      `gsutil mb gs://introduction-ai-291400`
    - For more information about the gsutil tool, see the documentation.

- **Create and launch a pipeline:** In Cloud Dataflow, data processing work is represented by a pipeline. A pipeline reads input data, performs transformations on that data, and then produces output data. A pipeline's transformations might include filtering, grouping, comparing, or joining data. The code for this example is located in the Apache Beam GitHub repository.
    - **Launch the pipeline on the Dataflow Service:** Use python to launch your pipeline on the Cloud Dataflow service. The running pipeline is referred to as a job.

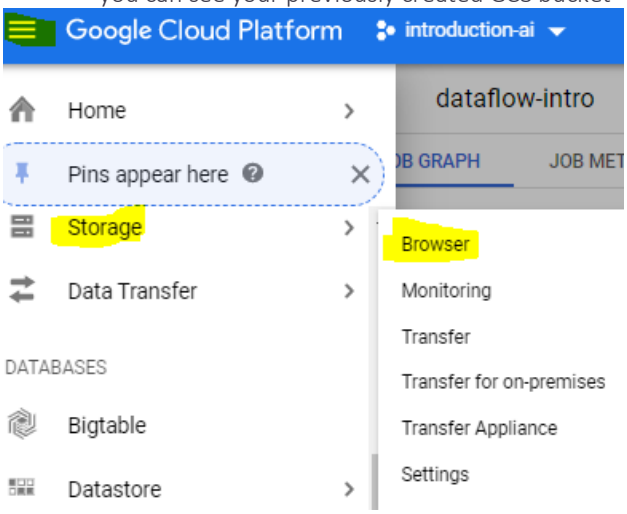
      ```
      (env) a_siddikov2@cloudshell:~ (introduction-ai-291400)$ python3 -m apache_beam.examples.wordcount --project introduction-ai-291400 --runner DataflowRunner --temp_location gs://introduction-ai-291400/temp --output gs://introduction-ai-291400/results/output --job_name dataflow-intro --region us-central1
      ```

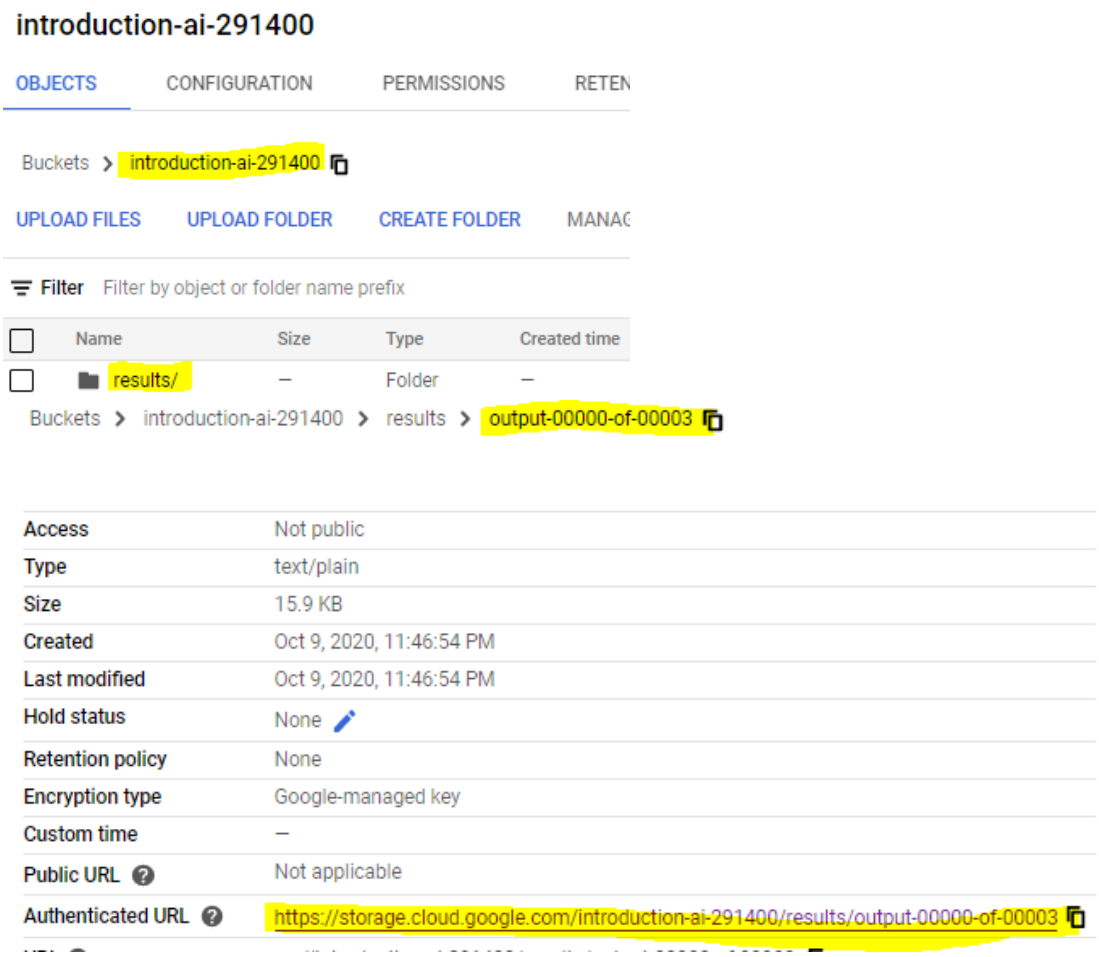
        - project is the GCP project.
        - runner is the specific execution engine to use to run your pipeline. The DataflowRunner uses the Dataflow Service as the execution engine.
        - temp_location is the storage bucket Cloud Dataflow will use for the binaries and other data for running your pipeline. This location can be shared across multiple jobs.
        - output is the bucket used by the WordCount example to store the job results.
        - job_name is a user-given unique identifier. Only one job may execute with the same name.
        - region specifies a regional endpoint for deploying your Dataflow jobs.
    - **Your job is running**: Congratulations! Your binary is now staged to the storage bucket that you created earlier, and Compute Engine instances are being created. Cloud Dataflow will split up your input file such that your data can be processed by multiple machines in parallel. You can move to the next section when you see the "JOB_STATE_RUNNING" message in the console.



- View the output: Now that your job has run, you can explore the output files in Cloud Storage.
    - Go to the Cloud Storage page
        - Open the menu on the left side of the console.
    - Then, select the Storage section, and click on Browser. You can verify that you are on the correct screen if you can see your previously created GCS bucket "introduction-ai-291400".

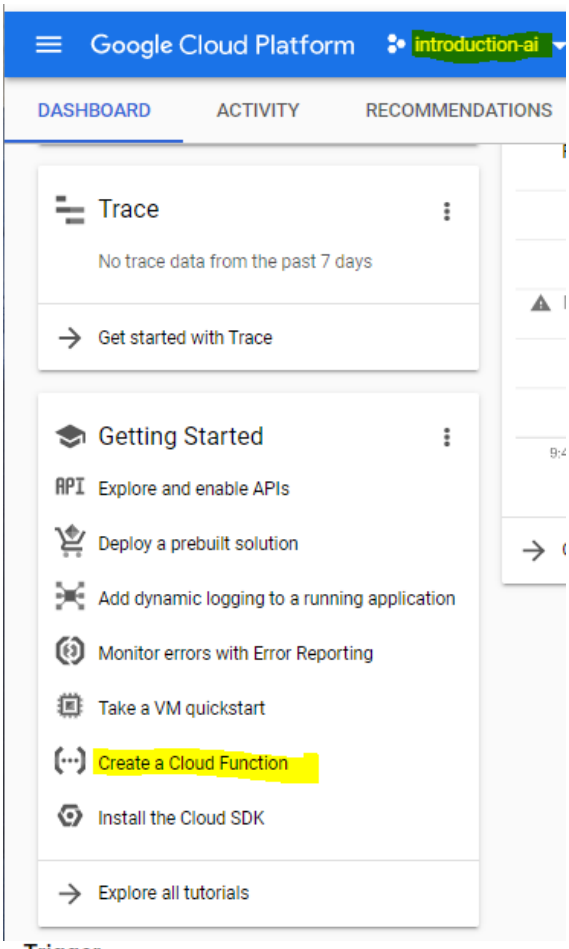




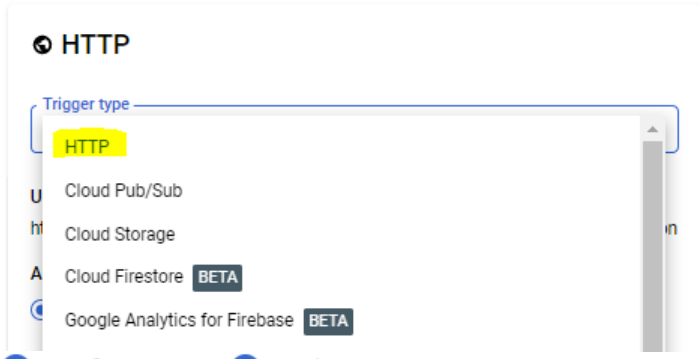

```
bow: 3
scurvy: 1
Importune: 1
forked: 1
embossed: 1
import: 1
day: 7
never: 26
profess: 3
covering: 1
Stain: 1
rare: 1
sing: 2
. . .
```

- This is a really powerful automated ETL pipeline where it can run periodically like once a day, once a week, etc. This really takes advantage of the power of the cloud which is that everything lives inside of the Google Cloud data center and it's able to provision machines to scale up to the job and also automatically provision them to go back down.

- Cloud Functions - serverless movement of data
  - Google Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment



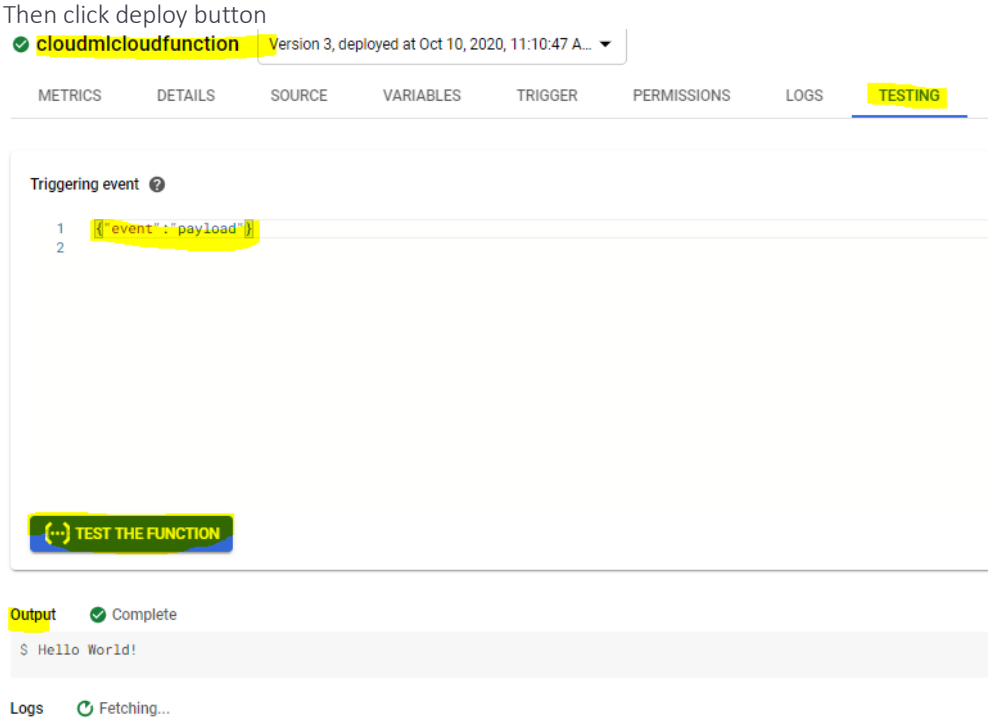
Trigger




```
import wikipedia
def hello_world(request):
    """Responds to any HTTP request.
    Args:
        request (flask.Request): HTTP request object.
    Returns:
        The response text or any set of values that can be turned into a
        Response object using
        `make_response <http://flask.pocoo.org/docs/1.0/api/#flask.Flask.make_response
>`.
```

```
    """
    request_json = request.get_json()
    result = wikipedia.summary("google", sentences=1)
    if request.args and 'message' in request.args:
        return request.args.get('message')
    elif request_json and 'message' in request_json:
        return request_json['message']
    else:
        return result
```

- Then click deploy button



- Google App Engine (GAE) CRON Jobs

Use App Engine as a cron job to schedule jobs and do ETL operations by creating a cron.yaml file.
https://cloud.google.com/appengine/docs/standard/python3/scheduling-jobs-with-cron-yaml

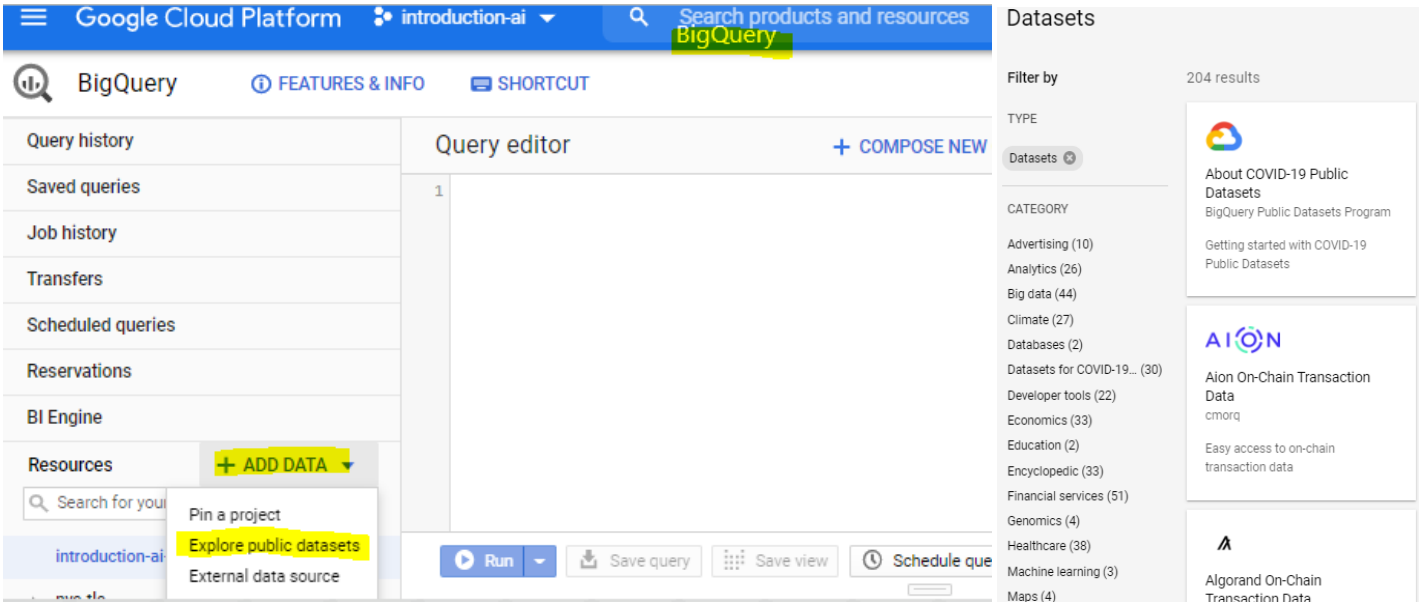
- The following is an example cron.yaml file:

```
cron:
- description: "daily summary job"
  url: /tasks/summary
  schedule: every 24 hours
- description: "monday morning mailout"
  url: /mail/weekly
  schedule: every monday 09:00
  timezone: Australia/NSW
- description: "new daily summary job"
  url: /tasks/summary
  schedule: every 24 hours
  target: beta
```

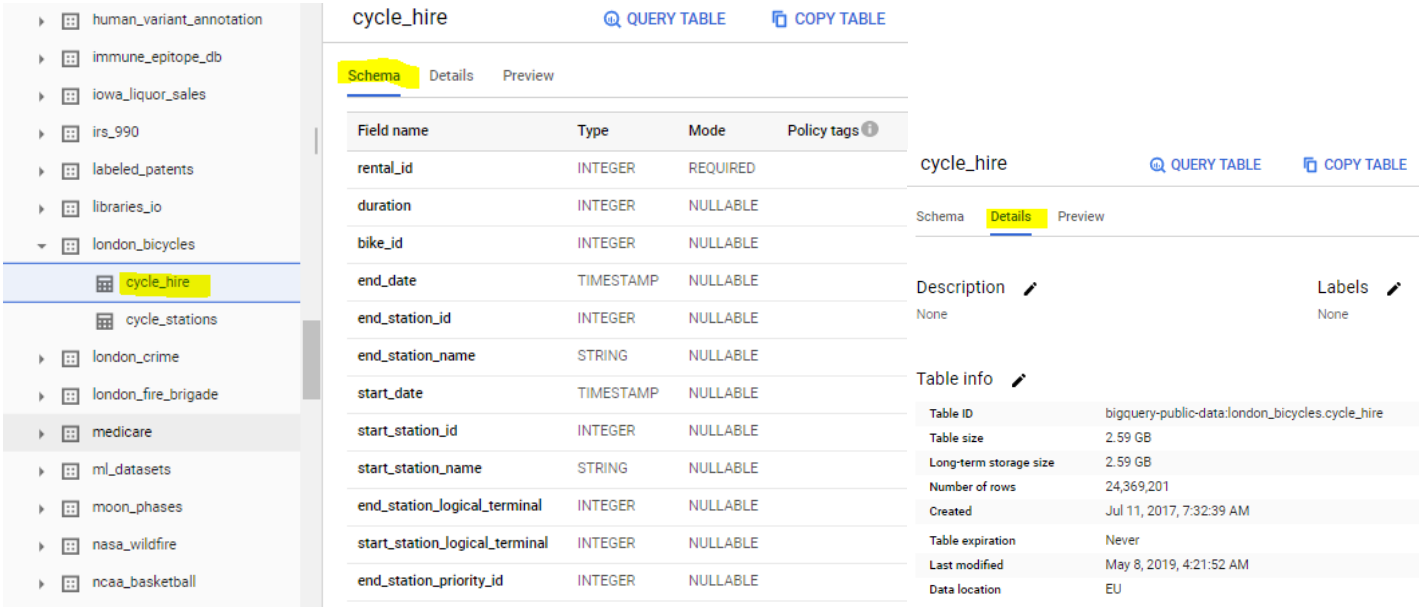
Lesson 3 – Use ML prediction on BigQuery

- Introduction to Big Theory

Search BigQuery from *search products and resources* and go to *add data,* and click *explore public datasets*. I search for London Bicycle Hires and click view dataset.



When I located the cycle hire, I can explore the data: schema, detail, and preview. It has about 2.6GB of data.



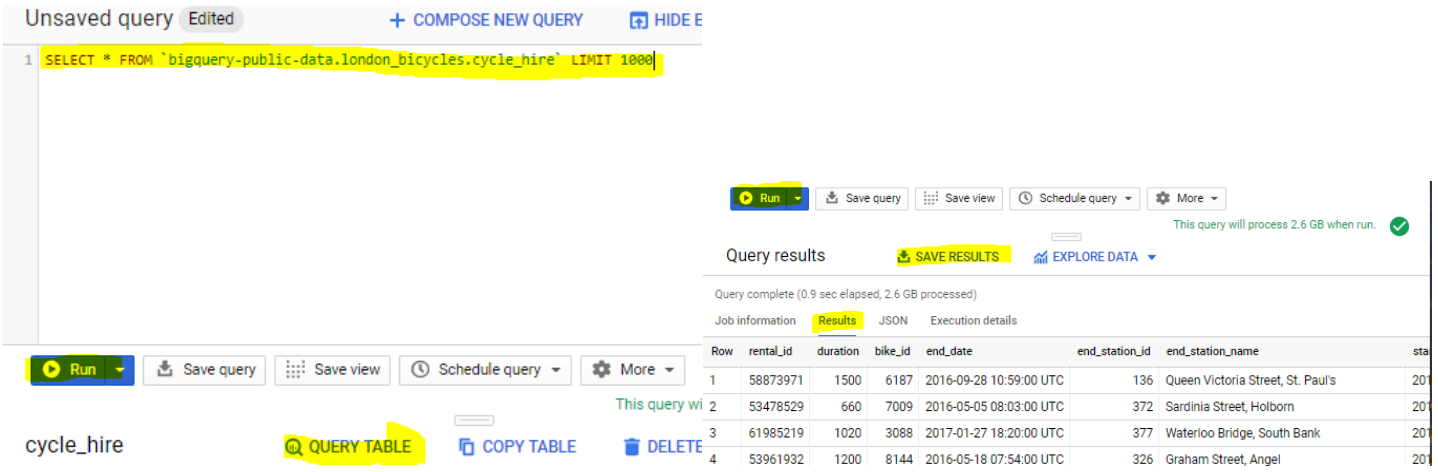
cycle_hire

QUERY TABLE COPY T

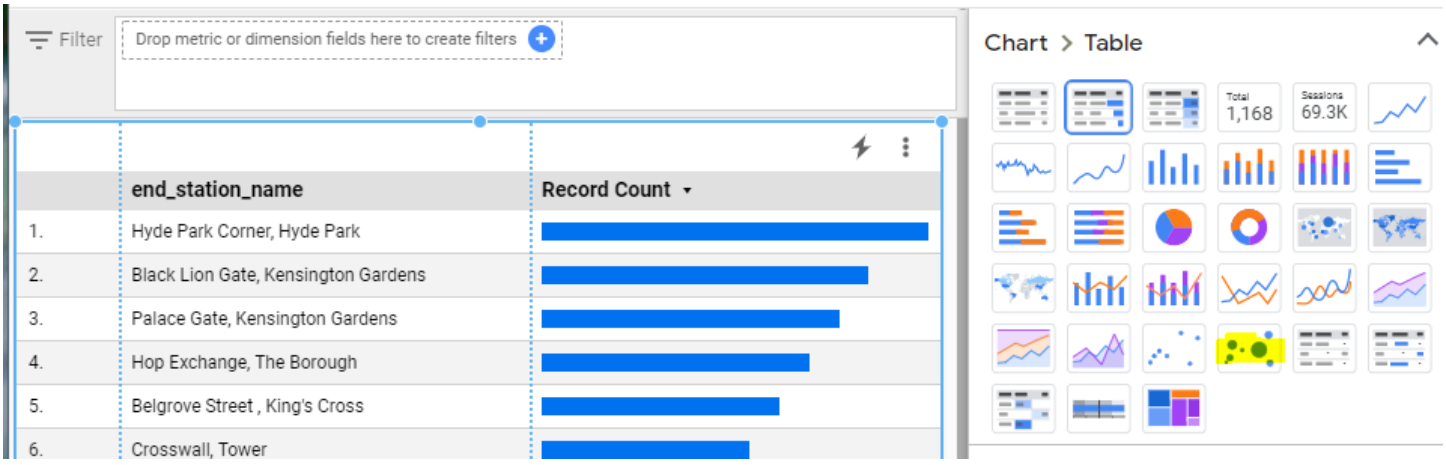
Schema Details Preview

Row	rental_id	duration	bike_id	end_date	end_station_id	end_station_name	start_date	start_station_id	start_station_name	end_station_logical_terminal
1	47469109	3180	7054	2015-09-03 12:45:00 UTC	111	Park Lane , Hyde Park	2015-09-03 11:52:00 UTC	300	Serpentine Car Park, Hyde Park	null
2	46915469	7380	3792	2015-08-16 11:59:00 UTC	407	Speakers' Corner 1, Hyde Park	2015-08-16 09:56:00 UTC	407	Speakers' Corner 1, Hyde Park	null
3	65899423	2040	3038	2017-06-09 18:30:00 UTC	165	Orsett Terrace, Bayswater	2017-06-09 17:56:00 UTC	579	Queen Street 2, Bank	null
4	64280726	2280	10868	2017-04-22 10:14:00 UTC	553	Regent's Row , Haggerston	2017-04-22 09:36:00 UTC	519	Teviot Street, Poplar	null
5	59235489	2340	7183	2016-10-09 04:31:00 UTC	100	Albert Embankment, Vauxhall	2016-10-09 03:52:00 UTC	612	Wandsworth Rd, Isley Court, Wandsworth Road	null
6	55248935	2160	7619	2016-06-26 07:26:00 UTC	465	Pitfield Street North,Hoxton	2016-06-26 06:50:00 UTC	459	Gunmakers Lane, Old Ford	null
7	43015438	5400	2779	2015-04-27 22:50:00 UTC	772	Binfield Road, Stockwell	2015-04-27 21:20:00 UTC	157	Wright's Lane, Kensington	null
8	61090882	1980	1710	2016-12-15 21:16:00 UTC	487	Canton Street, Poplar	2016-12-15 20:43:00 UTC	564	Somerset House, Strand	null
9	47150505	2400	6748	2015-08-22 17:15:00 UTC	291	Claverton Street, Pimlico	2015-08-22 16:35:00 UTC	248	Triangle Car Park, Hyde Park	null
10	54154089	3000	12290	2016-05-23 22:03:00 UTC	785	Aquatic Centre, Queen Elizabeth Olympic Park	2016-05-23 21:13:00 UTC	785	Aquatic Centre, Queen Elizabeth Olympic Park	null

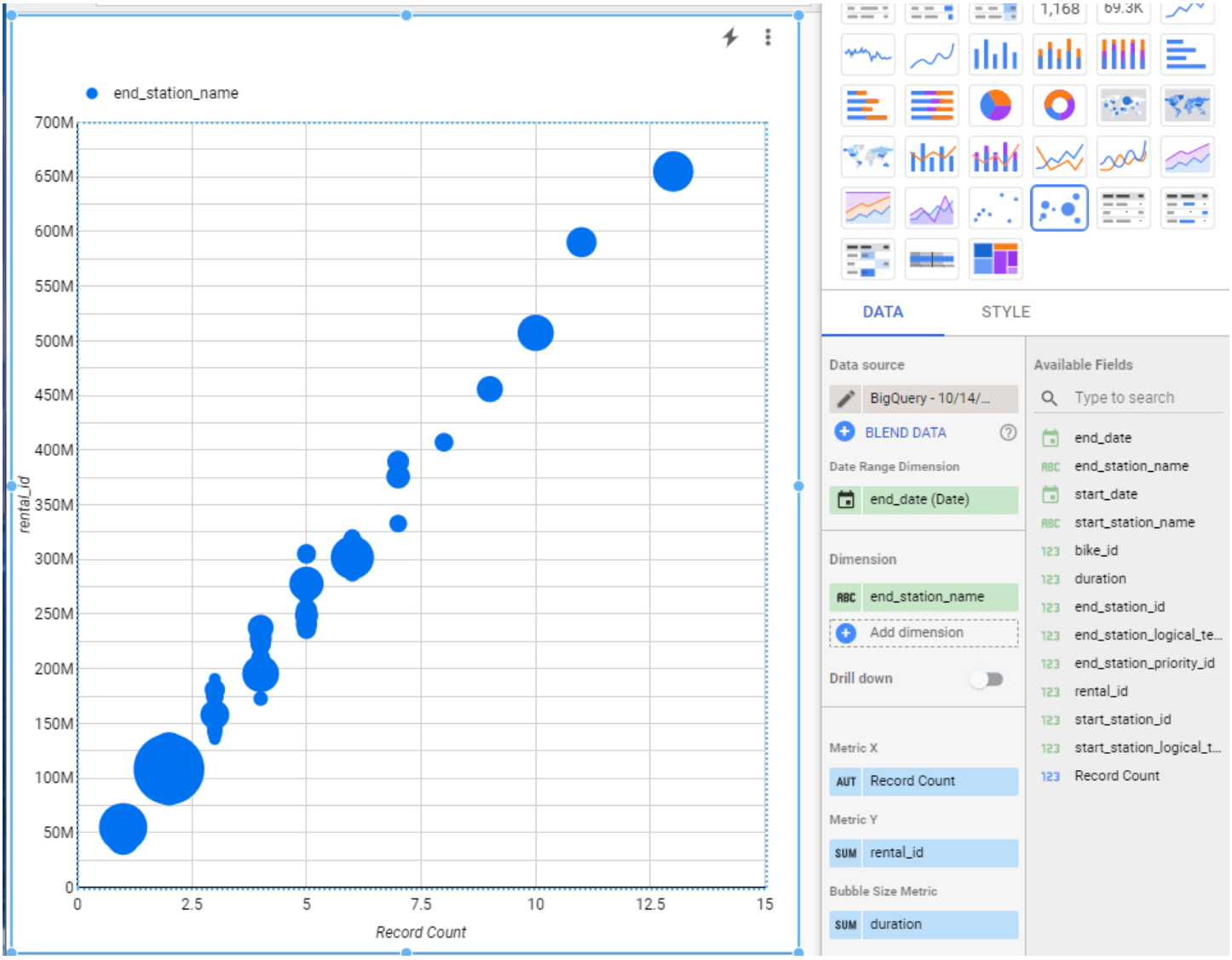
When we click the "Query Table," it will automatically generate the SQL query with a limit of 1000. We can select either all fields by inserting a star or specific field from the schema. Click the run button to run the SQL query and see the results. We can save the results as CSV (Google sheet or Excel), JSON file, and BigQuery table. We can also explore it with the data studio by clicking the "explore data" button.



If you get an "Oops…Not able to connect to your data" error message, go back and click the "explore data" button again. You will keep getting this error message until it pops up in the user agreement window. Once you agree with the terms and conditions, the error message will go away, and you will connect your data.



You can visualize your data by dragging and dropping the available fields to the x and y-axis and dimensions. It is very similar to the Tableau dashboard.

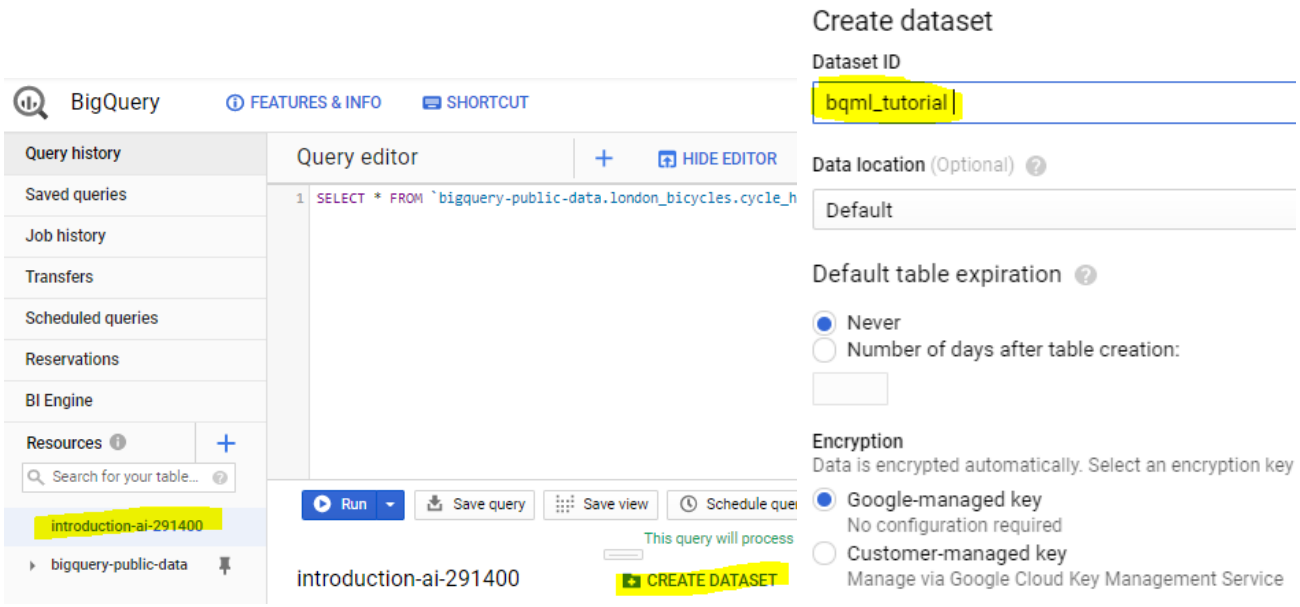


- Create supervised ML predictions with BigQuery

We will follow to the BigQuery ML documentation: https://cloud.google.com/bigquery-ml/docs/bigqueryml-natality
In this tutorial, you use the natality sample table to create a model that predicts the birth weight of a child based on the baby's gender, the length of the pregnancy, and demographic information about the mother. The natality sample table contains information about every birth in the United States over a 40 year period.

We need to follow those steps:

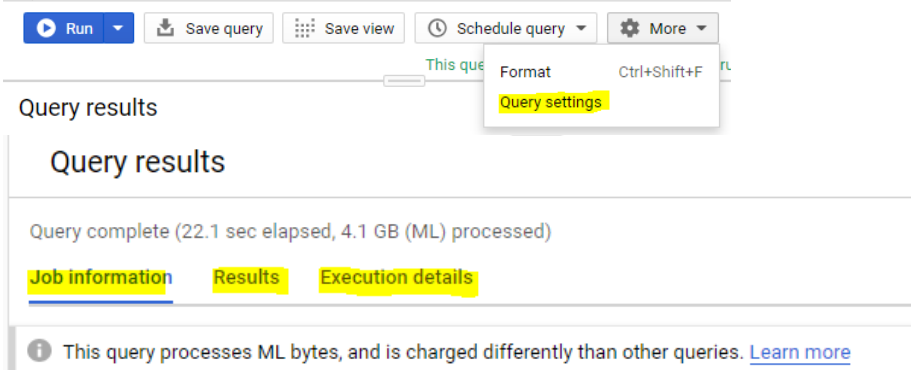
- Enable the API
- Choose the appropriate existing project
- Click "go to credentials."
  - Which API are you using? BigQuery API
  - Are you planning to use this API with App Engine or Compute Engine? Yes, I'm using one or both
  - Click the "What credentials do I need?" button and then click the "Done" button
- Now we need to "create dataset." In the navigation panel, in the Resources section, click your project name, then click Create dataset, and enter bqml_tutorial to Dataset ID.



- Copy-paste the following SQL

```
#standardSQL
CREATE MODEL `bqml_tutorial.natality_model`
OPTIONS
  (model_type='linear_reg',
    input_label_cols=['weight_pounds']) AS
SELECT
  weight_pounds,
  is_male,
  gestation_weeks,
  mother_age,
  CAST(mother_race AS string) AS mother_race
FROM
  `bigquery-public-data.samples.natality`
WHERE
  weight_pounds IS NOT NULL
  AND RAND() < 0.001
```

- Click Query settings to know more about it. We can also navigate to job information, results, and execution details

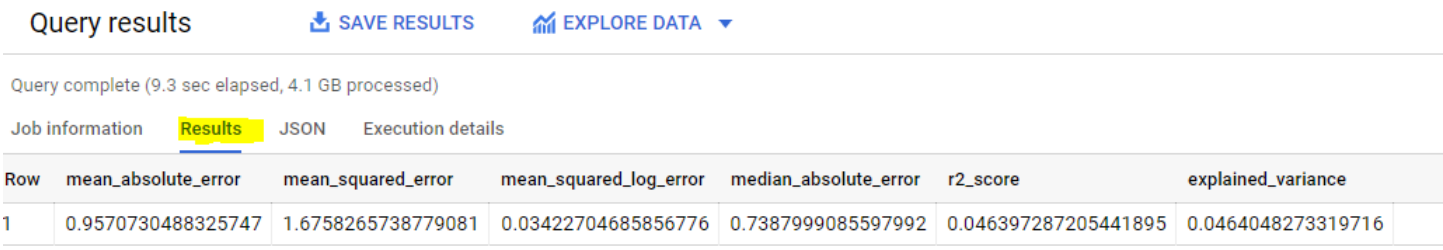


- Now we need to compose a new query and paste the following SQL code to evaluate the model.

```
#standardSQL
SELECT
```

```
  *
FROM
  ML.EVALUATE(MODEL `bqml_tutorial.natality_model`,
    (
    SELECT
      weight_pounds,
      is_male,
      gestation_weeks,
      mother_age,
      CAST(mother_race AS STRING) AS mother_race
    FROM
      `bigquery-public-data.samples.natality`
    WHERE
      weight_pounds IS NOT NULL))
```

Once we run the query, we can see the diagnostic information about how well we're able to actually predict. And how much the baby's weight is actually determined by the model.



Row	mean_absolute_error	mean_squared_error	mean_squared_log_error	median_absolute_error	r2_score	explained_variance
1	0.9570730488325747	1.6758265738779081	0.03422704685856776	0.7387999085597992	0.046397287205441895	0.0464048273319716

- After evaluating the query, we can compose a new query to run prediction

```
#standardSQL
SELECT
  predicted_weight_pounds
FROM
  ML.PREDICT(MODEL `bqml_tutorial.natality_model`,
    (
    SELECT
      is_male,
      gestation_weeks,
      mother_age,
      CAST(mother_race AS STRING) AS mother_race
    FROM
      `bigquery-public-data.samples.natality`
    WHERE
      state = "WY"))
```

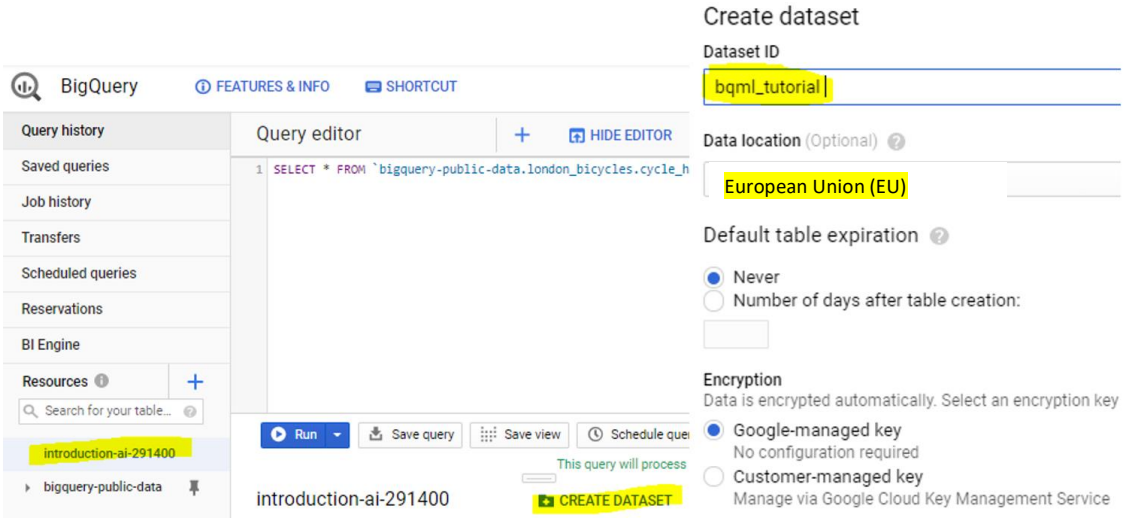
We can see the predicted values below by going to the results tab.

Row	is_male	gestation_weeks	mother_age	mother_race	predicted_weight_pounds
1	FALSE	29	42	1	7.437333591
2	TRUE	40	42	1	7.74759245
3	FALSE	42	40	1	7.752917823
4	TRUE	40	41	1	7.312649055
5	TRUE	39	42	1	7.576402535
6	TRUE	99	13	1	7.504205376
7	TRUE	41	15	1	7.523258532
8	TRUE	35	15	1	7.622907914
9	FALSE	41	41	1	7.684179207
10	TRUE	38	15	1	7.534459517
11	TRUE	42	15	1	7.817922778
12	TRUE	37	15	1	7.908678516
13	TRUE	35	40	1	7.278279747
14	FALSE	40	40	1	7.650085018
15	FALSE	39	15	1	7.720415346
16	TRUE	99	15	2	7.462643864
17	TRUE	34	14	1	7.693238242

- Create unsupervised learning: K-Means clustering with BigQuery

BigQuery ML supports unsupervised learning. You can apply the k-means algorithm to group your data into clusters. Unlike supervised machine learning, which is about predictive analytics, unsupervised learning is about descriptive analytics. It's about understanding your data so that you can make data-driven decisions. We will follow the following tutorial: https://cloud.google.com/bigquery-ml/docs/kmeans-tutorial

  - We need to create a new dataset: bqml_tutorial. For **Data location**, choose the **European Union (EU)**. The London Bicycle Hires public dataset is stored in the EU multi-region location. Your dataset should be in the same location.



Run the following query. This query extracts data on cycle hires, including start_station_name, and duration and joins it against station information, including distance-from-city-center. Then, it computes attributes of the station in station stats, including the average duration of rides and the number of trips, and passes through the station attribute distance_from_city_center. This query uses the WITH clause to define subqueries. The query also uses the ST_DISTANCE and ST_GEOGPOINT BigQuery GIS functions.

```
WITH
 hs AS (
 SELECT
   h.start_station_name AS station_name,
   IF
   (EXTRACT(DAYOFWEEK
     FROM
       h.start_date) = 1
```

```
      OR EXTRACT(DAYOFWEEK
      FROM
        h.start_date) = 7,
      "weekend",
      "weekday") AS isweekday,
    h.duration,
    ST_DISTANCE(ST_GEOGPOINT(s.longitude,
        s.latitude),
      ST_GEOGPOINT(-0.1,
        51.5))/1000 AS distance_from_city_center
  FROM
    `bigquery-public-data.london_bicycles.cycle_hire` AS h
  JOIN
    `bigquery-public-data.london_bicycles.cycle_stations` AS s
  ON
    h.start_station_id = s.id
  WHERE
    h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP)
    AND CAST('2016-01-01 00:00:00' AS TIMESTAMP) ),
  stationstats AS (
  SELECT
    station_name,
    AVG(duration) AS duration,
    COUNT(duration) AS num_trips,
    MAX(distance_from_city_center) AS distance_from_city_center
  FROM
    hs
  GROUP BY
    station_name )
SELECT
  *
FROM
  stationstats
ORDER BY
  distance_from_city_center ASC
```

Query complete (4.8 sec elapsed, 1.2 GB processed)

Job information | Results | JSON | Execution details

Row	station_name	duration	num_trips	distance_from_city_center
1	Borough Road, Elephant & Castle	1349.318091187026	7523	0.12623965466425408
2	Webber Street , Southwark	823.8790560471979	8136	0.16402063786209384
3	Great Suffolk Street, The Borough	844.3554874545104	10442	0.19366718830977991
4	LSBU (Borough Road), Elephant & Castle	1297.835314091682	7068	0.25790299799917293
5	Harper Road, Borough	822.6867469879517	1660	0.30630610581879264
6	Harper Road, The Borough	1083.9731285988482	4168	0.30630610581879264

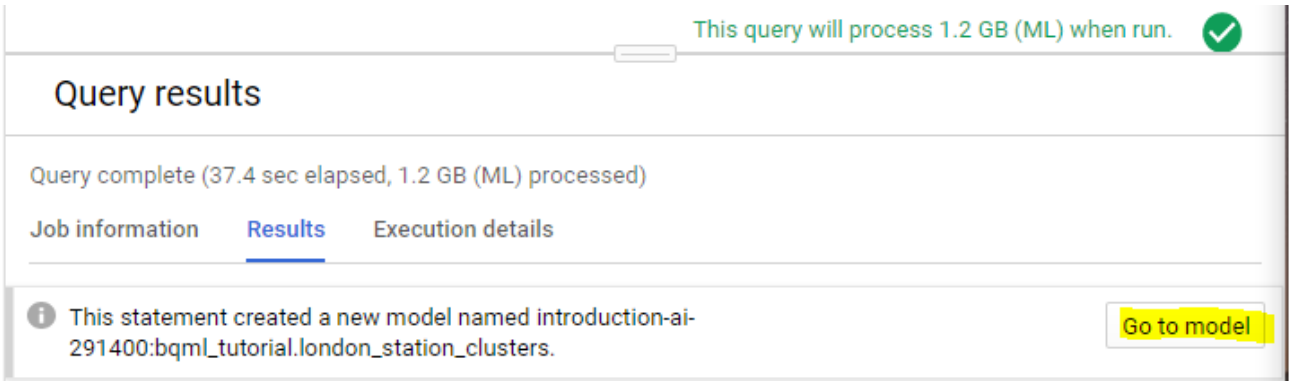
- Now that you have examined your training data, the next step is to create a k-means model using the data.

  You can create and train a k-means model using the CREATE MODEL statement with the option model_type=kmeans. The following query adds a CREATE MODEL statement to the previous query and removes the id fields in the data.

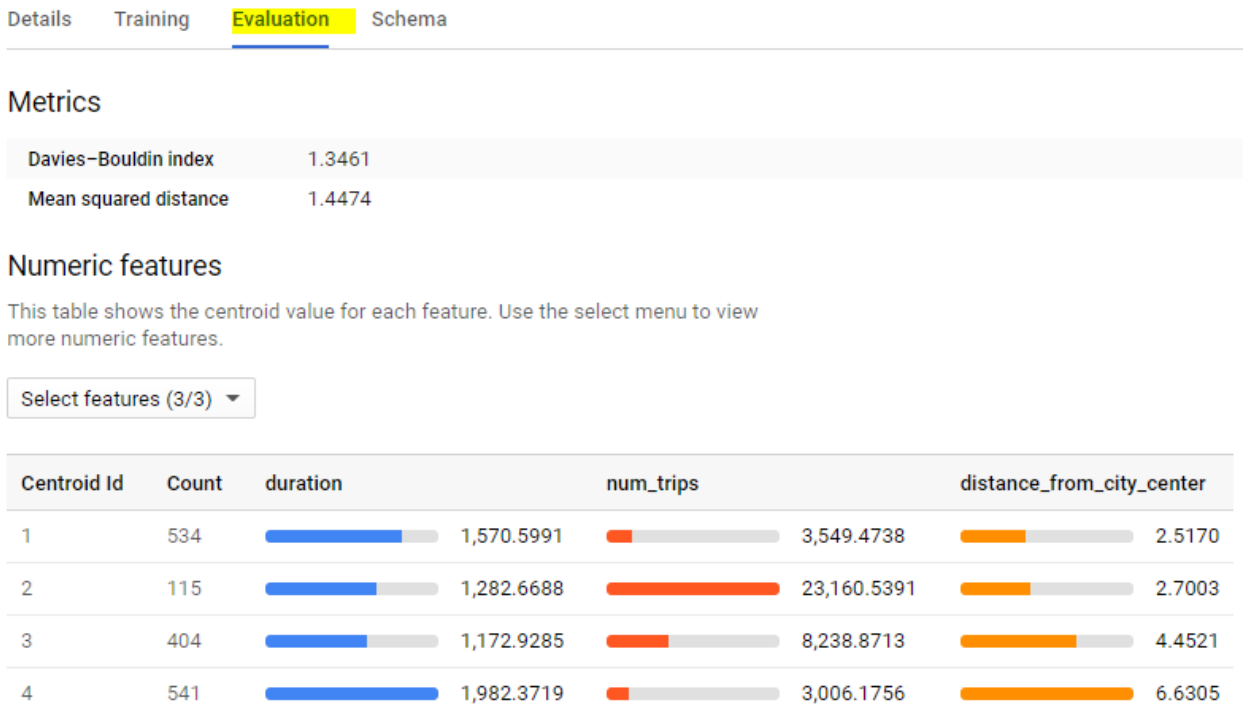
  The CREATE MODEL statement specifies the desired number of clusters — four. In the SELECT statement, the EXCEPT clause excludes the station_name column because station_name is not a feature. The query creates a unique row per station_name, and only the features are mentioned in the SELECT statement. If you omit the num_clusters option, BigQuery ML will choose a reasonable default based on the total number of rows in the training data.

```
CREATE OR REPLACE MODEL
  bqml_tutorial.london_station_clusters OPTIONS(model_type='kmeans',
    num_clusters=4) AS
WITH
  hs AS (
  SELECT
    h.start_station_name AS station_name,
  IF
    (EXTRACT(DAYOFWEEK
      FROM
        h.start_date) = 1
      OR EXTRACT(DAYOFWEEK
      FROM
        h.start_date) = 7,
      "weekend",
      "weekday") AS isweekday,
    h.duration,
    ST_DISTANCE(ST_GEOGPOINT(s.longitude,
        s.latitude),
      ST_GEOGPOINT(-0.1,
        51.5))/1000 AS distance_from_city_center
  FROM
    `bigquery-public-data.london_bicycles.cycle_hire` AS h
  JOIN
    `bigquery-public-data.london_bicycles.cycle_stations` AS s
  ON
    h.start_station_id = s.id
  WHERE
    h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP)
    AND CAST('2016-01-01 00:00:00' AS TIMESTAMP) ),
  stationstats AS (
  SELECT
    station_name,
    isweekday,
    AVG(duration) AS duration,
    COUNT(duration) AS num_trips,
    MAX(distance_from_city_center) AS distance_from_city_center
  FROM
    hs
```

```
  GROUP BY
    station_name, isweekday)
SELECT
  * EXCEPT(station_name, isweekday)
FROM
  stationstats
```



Click the **Evaluation** tab. This tab displays visualizations of the clusters identified by the k-means model. Under **Numerical features**, bar graphs display up to 10 of the most important numerical feature values for each centroid. You can select which features to visualize from the drop-down menu.



Run prediction. This query uses the REGEXP_CONTAINS function to find all entries in the station_name column that contain the string "Kennington". The ML.PREDICT function uses those values to predict which clusters would contain those stations.

```
WITH
  hs AS (
  SELECT
    h.start_station_name AS station_name,
```

```
    IF
    (EXTRACT(DAYOFWEEK
      FROM
        h.start_date) = 1
      OR EXTRACT(DAYOFWEEK
      FROM
        h.start_date) = 7,
      "weekend",
      "weekday") AS isweekday,
    h.duration,
    ST_DISTANCE(ST_GEOGPOINT(s.longitude,
        s.latitude),
      ST_GEOGPOINT(-0.1,
        51.5))/1000 AS distance_from_city_center
  FROM
    `bigquery-public-data.london_bicycles.cycle_hire` AS h
  JOIN
    `bigquery-public-data.london_bicycles.cycle_stations` AS s
  ON
    h.start_station_id = s.id
  WHERE
    h.start_date BETWEEN CAST('2015-01-01 00:00:00' AS TIMESTAMP)
    AND CAST('2016-01-01 00:00:00' AS TIMESTAMP) ),
  stationstats AS (
  SELECT
    station_name,
    AVG(duration) AS duration,
    COUNT(duration) AS num_trips,
    MAX(distance_from_city_center) AS distance_from_city_center
  FROM
    hs
  GROUP BY
    station_name )
SELECT
  * EXCEPT(nearest_centroids_distance)
FROM
  ML.PREDICT( MODEL bqml_tutorial.london_station_clusters,
    (
    SELECT
      *
    FROM
      stationstats
    WHERE
      REGEXP_CONTAINS(station_name, 'Kennington')))
```

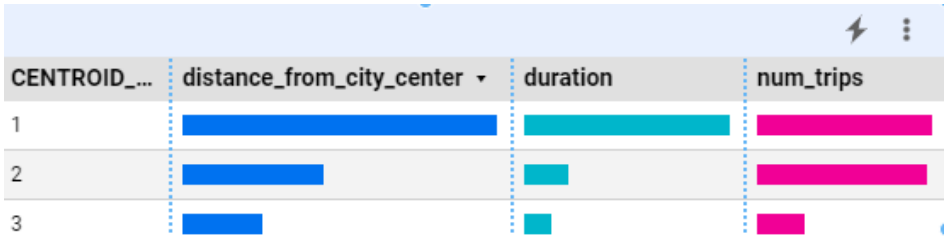
Query results SAVE RESULTS EXPLORE DATA

Query complete (1.3 sec elapsed, 1.2 GB processed)

Job information | Results | JSON | Execution details

Row	CENTROID_ID	station_name	duration	num_trips	distance_from_city_center
1	1	Kennington Road Post Office, Oval	2096.259541984733	7074	1.84603345094448
2	1	Cleaver Street, Kennington	1231.1580148317169	5259	1.4967922765165333
3	1	Doddington Grove, Kennington	1532.1560975609761	9225	1.468140527379382
4	2	Kennington Lane Rail Bridge, Vauxhall	1144.5931083593778	25277	2.175032834765301
5	3	Kennington Oval, Oval	1325.5939126952348	12485	2.0831341271372983

We can also visualize the results by clicking the explore data



Lesson 4 – Use AutoML

- Overview

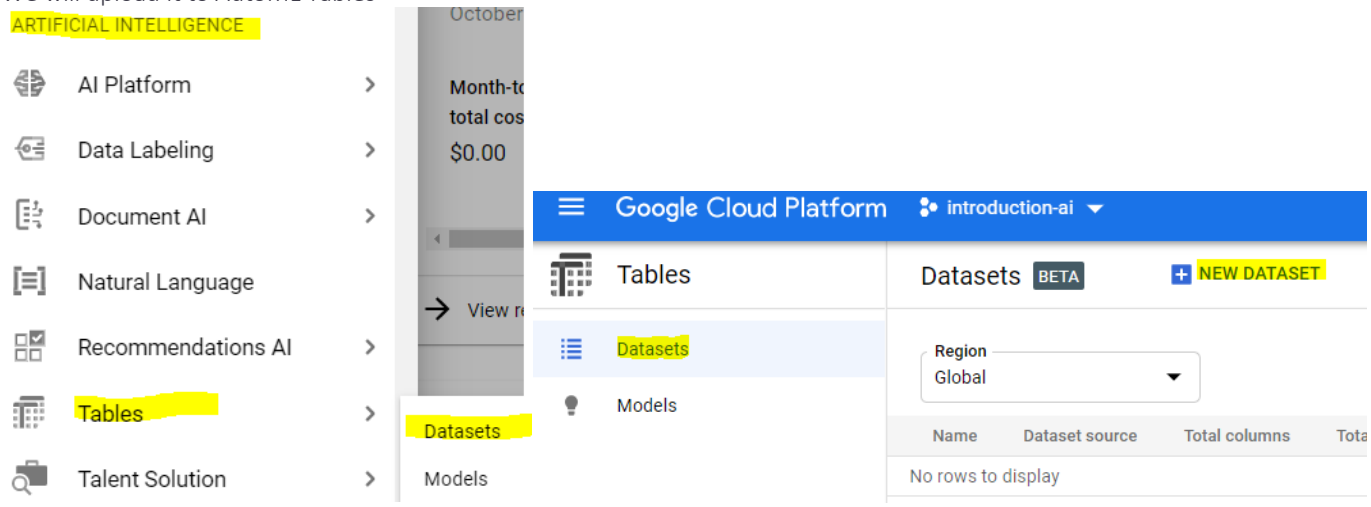
  We will go through this tutorial: https://cloud.google.com/vision/automl/docs/beginners-guide

- Use AutoML Vision

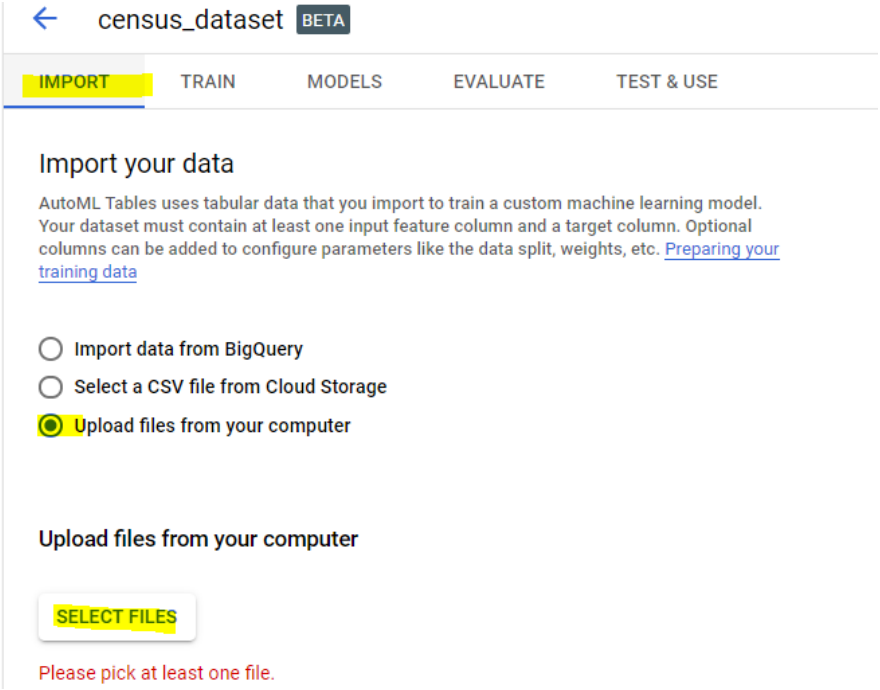
  Please look at the next video

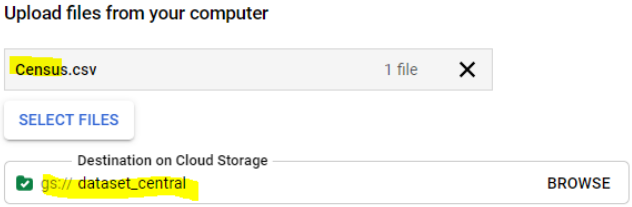
- Use AutoML Tables

  Once we download the census income dataset: https://archive.ics.uci.edu/ml/datasets/Census+Income
  We will upload it to AutoML Tables

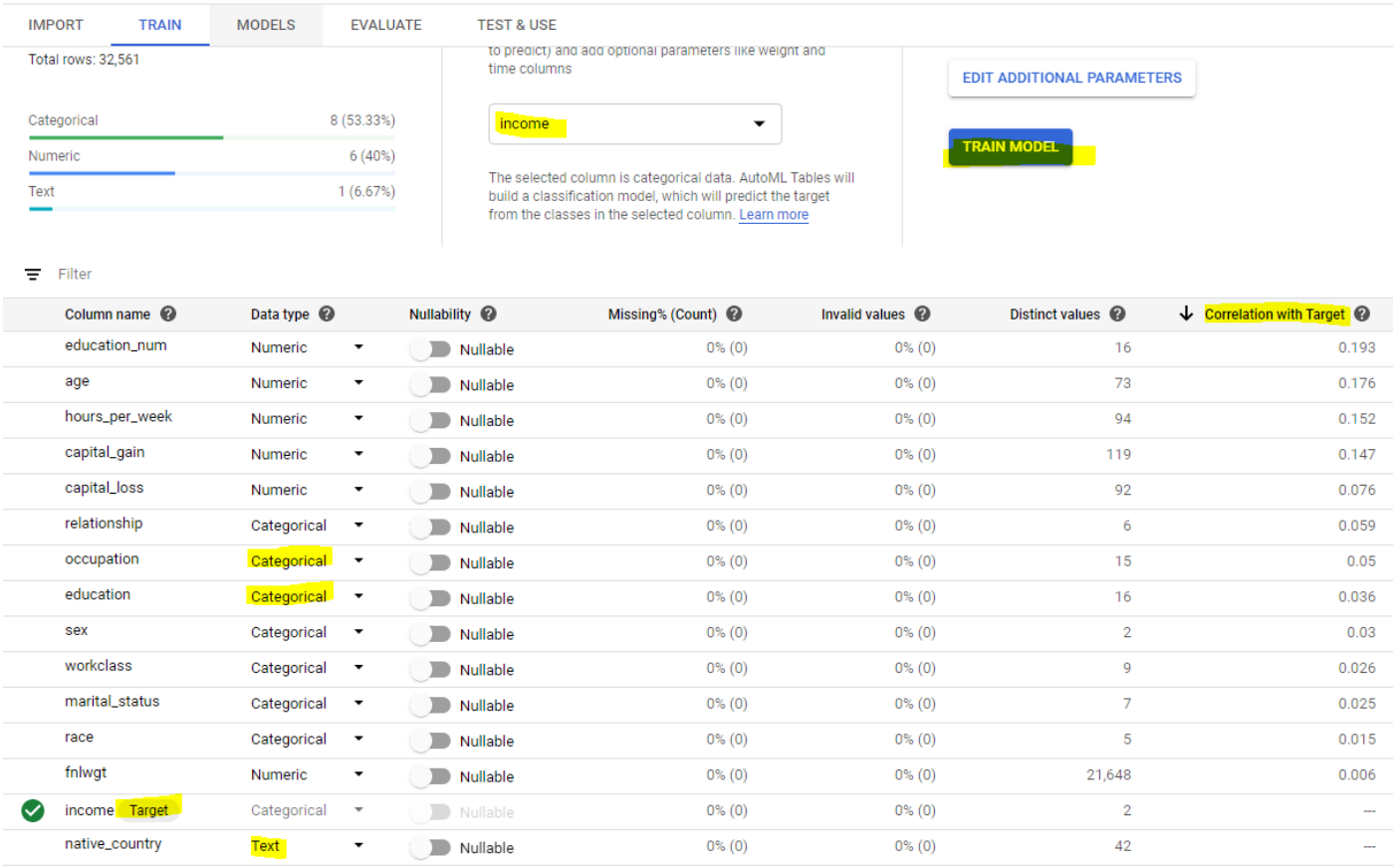


I can import data from my computer or cloud storage (bucket). I chose import csv file from my computer and selected appropriate bucket to upload the file.





Make sure the headers do not have space or dash. You need to convert them to underscores.



So, we can specify the how many node hours we need to train and what fields to use for prediction.

Train your model

Model name *
census_dataset_20201014040710

Training budget

Enter a number between 1 and 72 for the maximum number of node hours to spend training your model. If your model stops improving before then, AutoML Tables will stop training and you'll only be charged for the actual node hours used. Training budget doesn't include setup, preprocessing, and tear down. These steps usually don't exceed one hour total and you won't be charged for that time. Training pricing guide

Budget *
1 maximum node hour

Input feature selection

By default, all other columns in your dataset will be used as input features for training (excluding target, weight, and split columns).

14 feature columns *
All columns selected

Summary

Model type: Binary classification model

Data split: Automatic

Target: income

Input features: 14 features

Rows: 32,561 rows

Advanced options

TRAIN MODEL CANCEL

Models

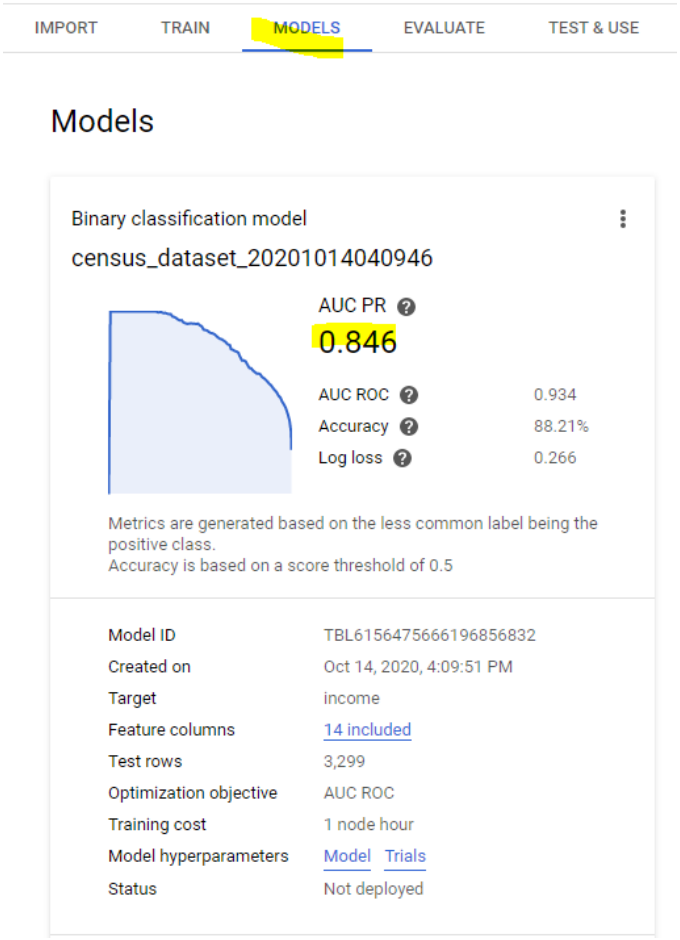
census_dataset_20201014040946

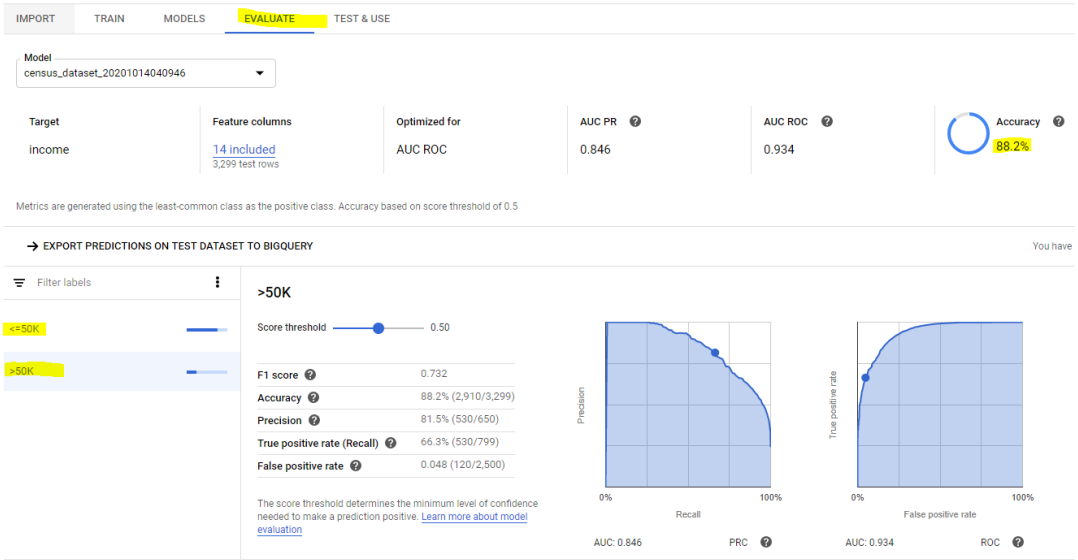
Training may take several hours. This includes node training time as well as infrastructure set up and tear down, which you aren't charged for.

You will be emailed once training completes.

Node hour training

CANCEL





Lesson 5 – Use AI Platform

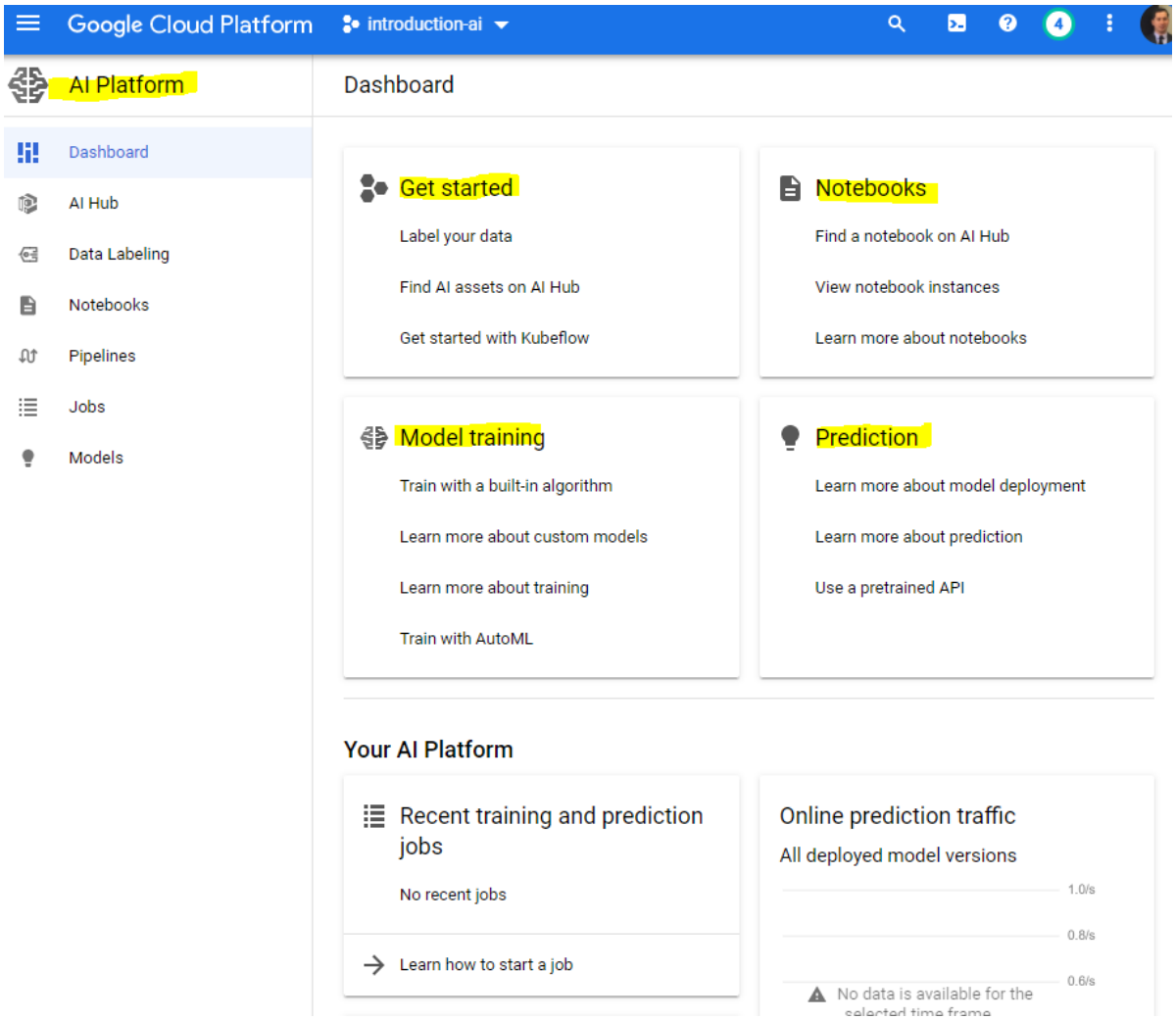
- Explore AI APIs

  The cloud vendors are developing sophisticated AI APIs. You can outsource some of those APIs to the cloud. If you off-load it to the cloud platform, you don't have to be worried about the accuracy of the model. You can focus on another part of the problem. this is really tapping into this concept of comparative advantage, which means, you should focus on what you do best. If someone, another organization, another person, can do something at a cheaper cost, you should pay that person to do it for you so you can focus on the thing you're uniquely qualified to do.

  - NLP
    - Entity extraction (organization, person, etc.)
    - Sentiment analysis (negative, positive, neutral)
  - Vision

        - AutoML
        - Vision API
    - Translate

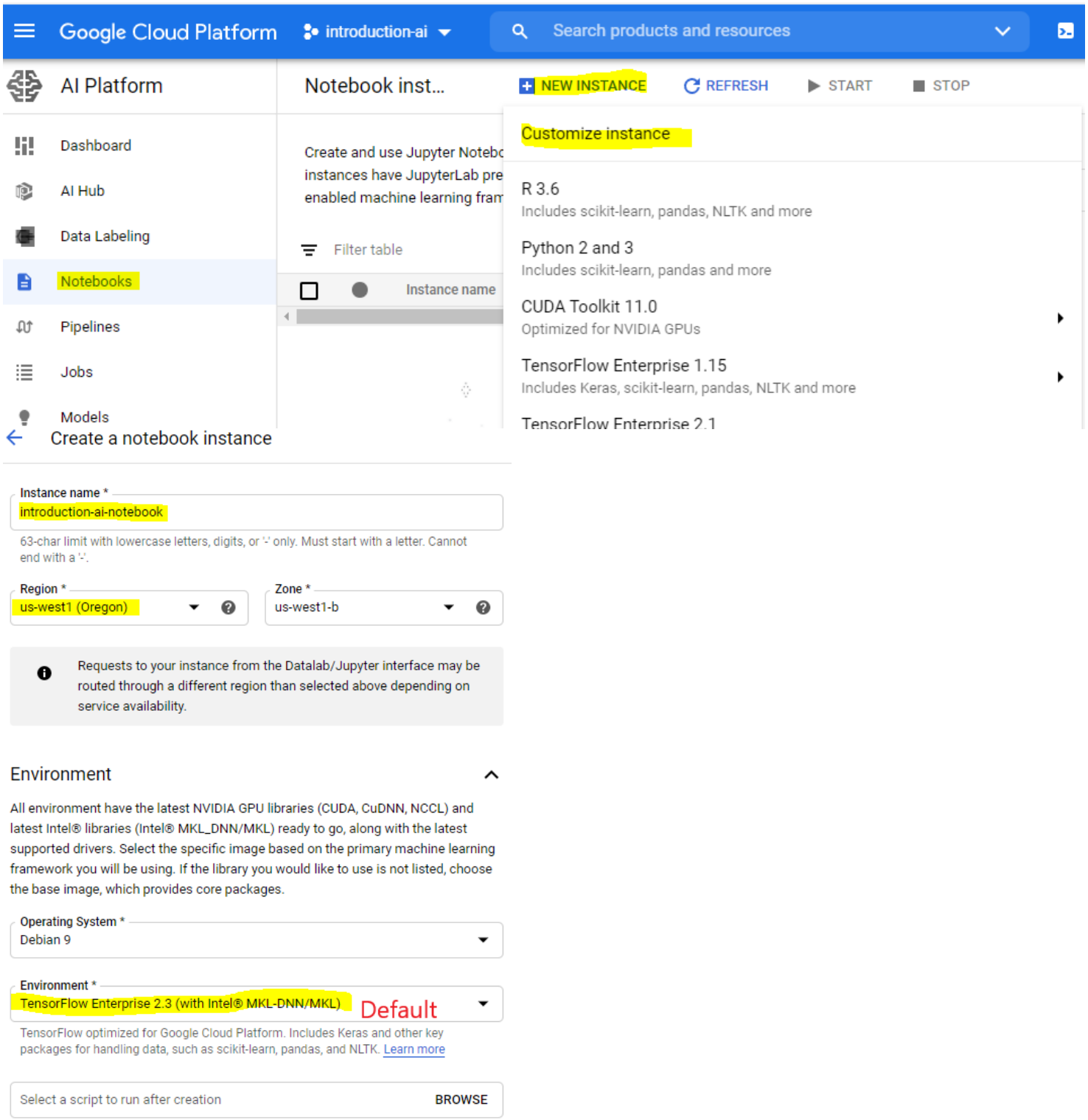
- Use Notebooks for Data Science explorations



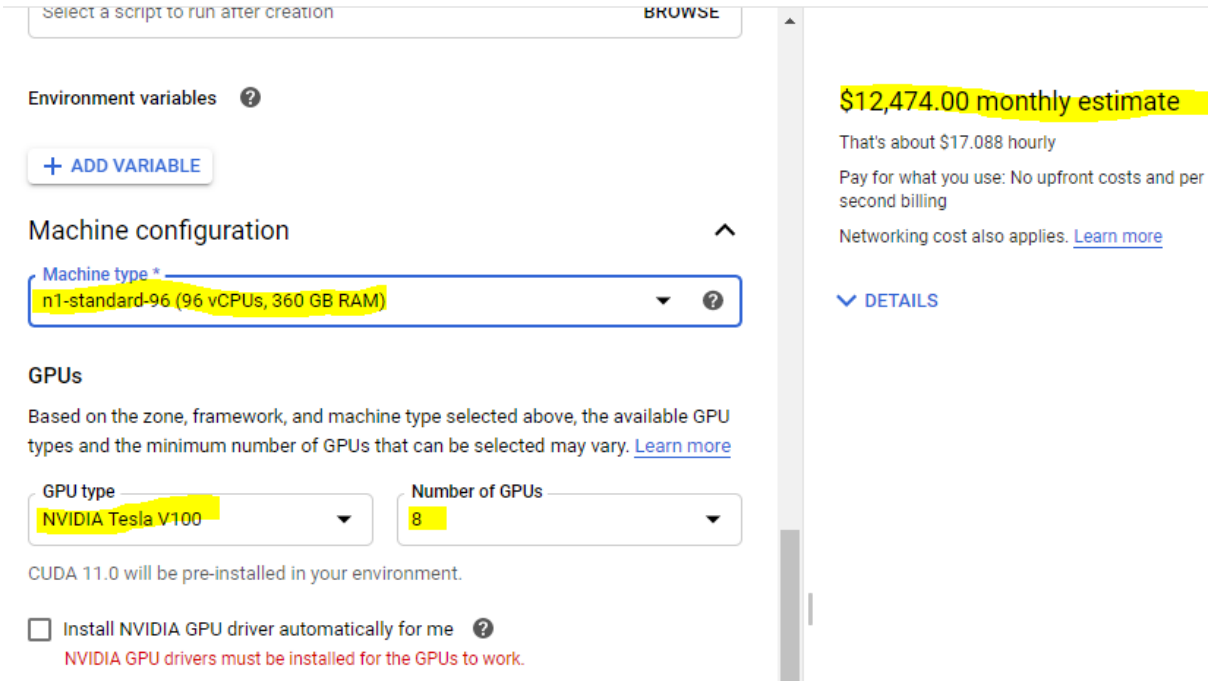
https://github.com/GoogleCloudPlatform/cloudml-samples/blob/master/notebooks/scikit-learn/OnlinePredictionWithScikitLearnInCMLE.ipynb

- Predict with AI Platform

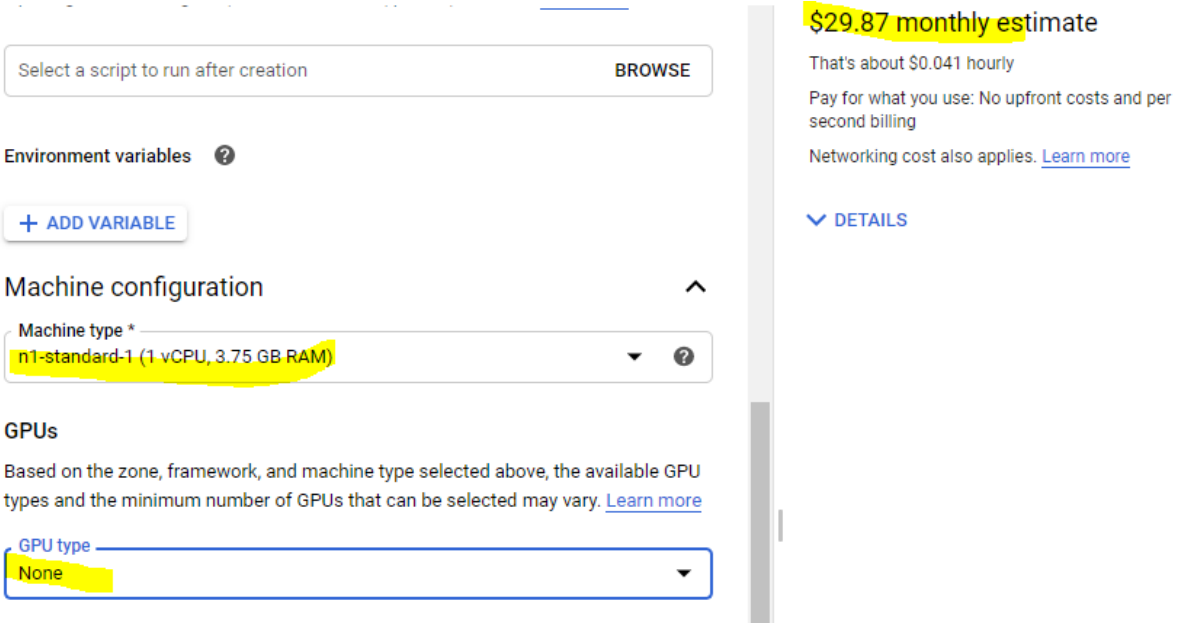
We select a customize interface because we need to tune it a little bit.

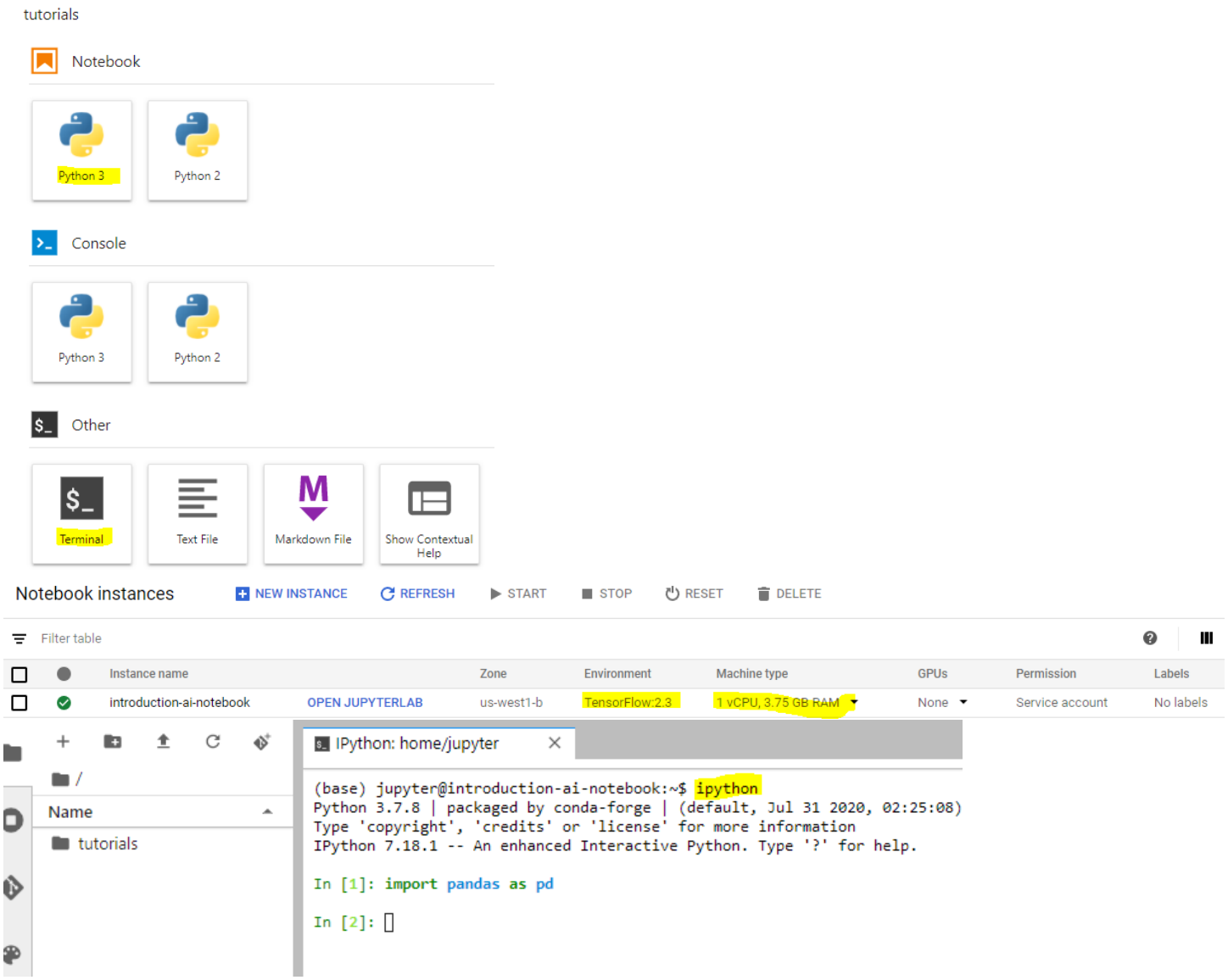


Depending on our machine type and GPU type selection, the monthly estimated cost gets very expensive. There are high CPU, high memory, and ultra-memory. So, there's incredibly expensive machines that could almost bankrupt you. You want to be very careful about not selecting a machine like that. You also have GPU as well. The cost will go up even higher if I select a very expensive GPU as well as maximum number of GPUs.



The following machine and GPU configuration give us a pretty reasonable price – about $30 a month.





Do you want to train something that can extract entities?
Or do you want to use off-shelve APIs and mix it with your Python code to get solutions much quicker? So, we need to create solutions that solve customers' problems quickly, and that's really how the AI platform plays a role in the future.

```
# Imports the Google Cloud client library
from google.cloud import language
from google.cloud.language import enums
from google.cloud.language import types

result = "Bill Gates was born in Seattle"
client = language.LanguageServiceClient()
document = types.Document(
      content=result,
      type=enums.Document.Type.PLAIN_TEXT)
entities = client.analyze_entities(document).entities
print(str(entities))
```

```
[1]: # Imports the Google Cloud client library
from google.cloud import language
from google.cloud.language import enums
from google.cloud.language import types
```

Entity Extraction

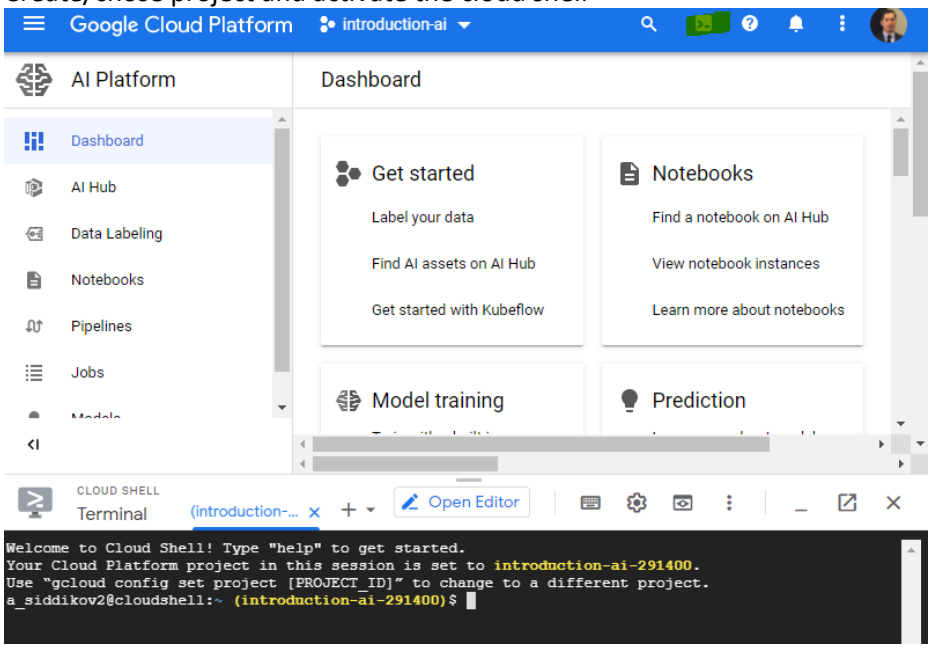
```
[2]: result = "Bill Gates was born in Seattle"
client = language.LanguageServiceClient()
document = types.Document(
content=result,
        type=enums.Document.Type.PLAIN_TEXT)
entities = client.analyze_entities(document).entities
print(str(entities))
```

```
[name: "Bill Gates"
type: PERSON
metadata {
  key: "mid"
  value: "/m/017nt"
}
metadata {
  key: "wikipedia_url"
  value: "https://en.wikipedia.org/wiki/Bill_Gates"
}
salience: 0.6612948775291443
mentions {
  text {
    content: "Bill Gates"
    begin_offset: -1
  }
  type: PROPER
}
, name: "Seattle"
type: LOCATION
metadata {
  key: "mid"
  value: "/m/0d9jr"
}
metadata {
  key: "wikipedia_url"
```

Lesson 6 – Build an Analytics Application from Scratch

- Create a new Analytics application from scratch

Create/chose project and activate the cloud shell



Let's verify if I'm in a right project:

```
$ gcloud projects describe introduction-ai-291400
createTime: '2020-10-03T00:14:23.349Z'
lifecycleState: ACTIVE
name: introduction-ai
projectId: introduction-ai-291400
projectNumber: '381877330557'
```

if you needed to switch from one project to another project you also could run this command

```
$ gcloud config set project introduction-ai-291400
Updated property [core/project].
```

You can also toggle back and forth and work on different projects by running this gcloud config set project. The two main commands you'd want to use gcloud project describe make sure you're on the right project and gcloud config set project to ensure that you're switching to a different project.
Now that you've verified that you're on the correct project, you can go ahead and create a new app engine application.

```
$ gcloud app create
```

Now that we've gone through and created a new application, we can clone the sample source code that Google provides.

```
$ git clone git clone https://github.com/GoogleCloudPlatform/python-docs-samples
```

CD it to the correct directory where it actually has an example of a Python 3 flask app

```
$ cd python-docs-samples/appengine/standard_python3/hello_world/

a_siddikov2@cloudshell:~/python-docs-samples/appengine/standard_python3/hello_world
(introduction-ai-291400)$ ls
app.yaml main.py
main_test.py
requirements-test.txt
requirements.txt
```

Let's go ahead and now create a virtual environment. This isolates my system, which I can create multiple projects and not worry about conflicting packages.

```
$ virtualenv venv
```

I can activate that virtual environment by typing the following code.

```
$ source venv/bin/activate
```

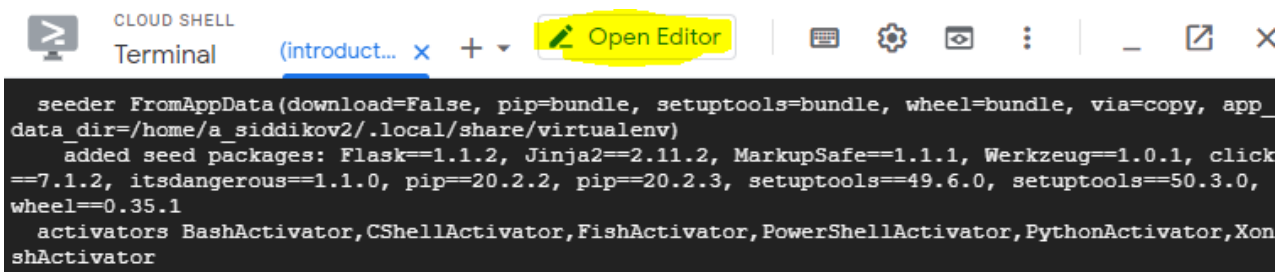
I can verify if it's working; I can type in which python and see that it's using a Python inside my virtual environment.

```
$ which python
```

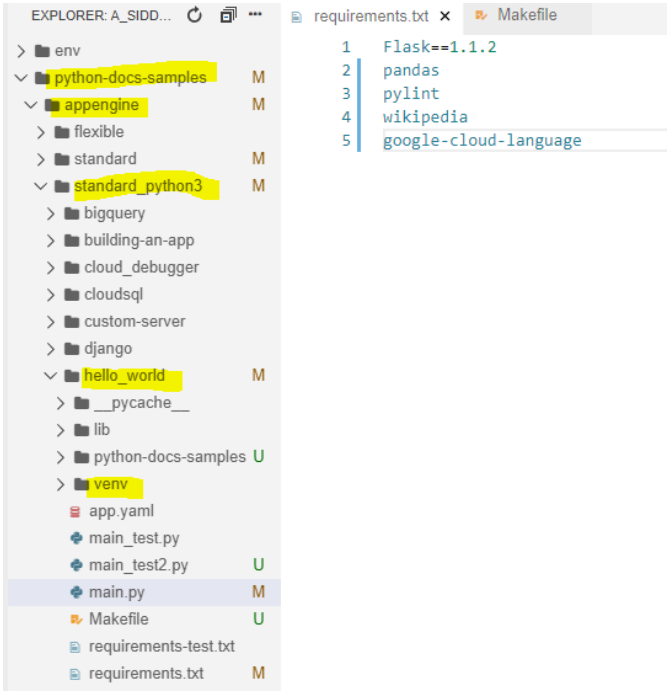
- Deploy an Analytics application

Now we need to click the "open editor" to locate the hello world package. We can go back to the terminal and install missing packages such as Flask inside of my virtual environment (use "-r"):

```
$ pip install -r requirements.txt
```

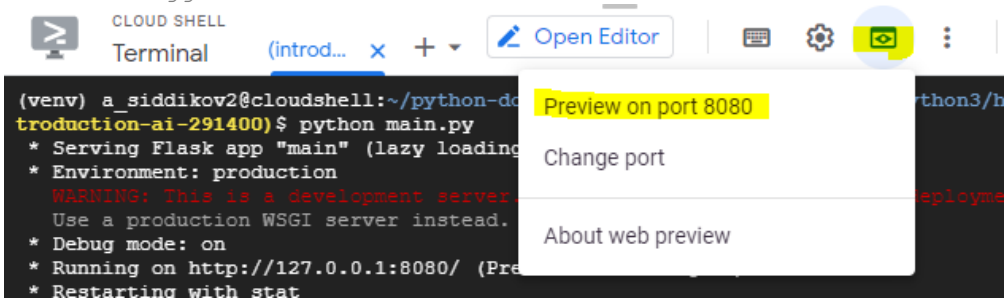


/home/a_siddikov2/python-docs-samples/appengine/standard_python3/hello_world

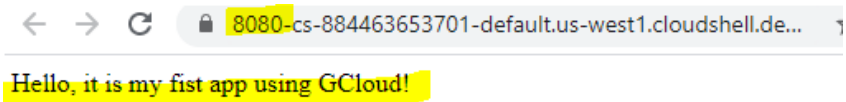


If I run python main.py, this will run Flask locally.

```
(venv) a_siddikov2@cloudshell:~/python-docs-
samples/appengine/standard_python3/hello_world (introduction-ai-291400)$ python
main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 266-730-147
```



I can click web preview and view it on port 8080. We can kill the Flask app by clicking control and C.



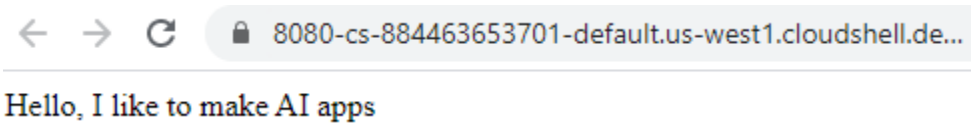
Hello, it is my fist app using GCloud!

We can replace the code which is inside of main.py with the following one:

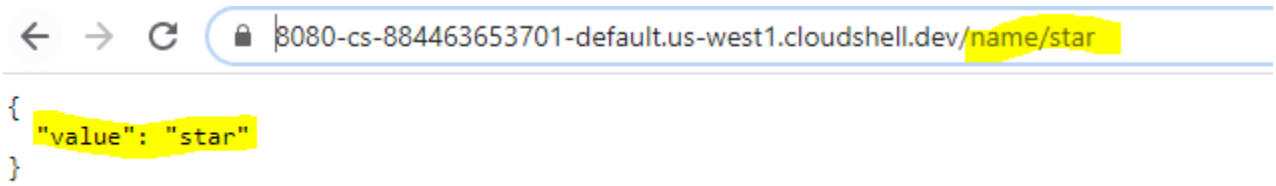
```
from flask import Flask
from flask import jsonify

app = Flask(__name__)
```

```
@app.route('/')
def hello():
    return 'Hello, I like to make AI apps'

@app.route('/name/<value>')
def name (value):
    val = {"value": value}
    return jsonify(val)

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True )
```

When I run the above Flask, I get the following results.

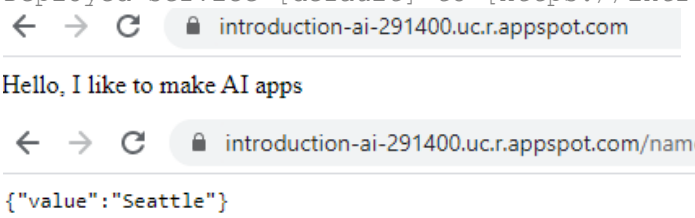


We can pass the "star" parameter into the name function and print out its value. When I type the "star," a Flask app itself intercepts it and converts it to a dictionary, which then gets jsonafied and returned to the users.



Now we will bundle up everything that's in this directory and deploy this Flask app. This deployment will run inside the Google production environment.

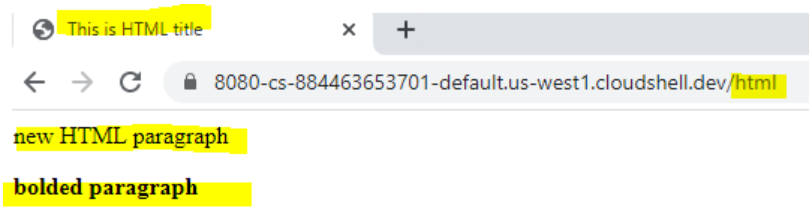
```
$ gcloud auth login
$ gcloud app deploy
Do you want to continue (Y/n)?  Y
Deployed service [default] to [https://introduction-ai-291400.uc.r.appspot.com]
```



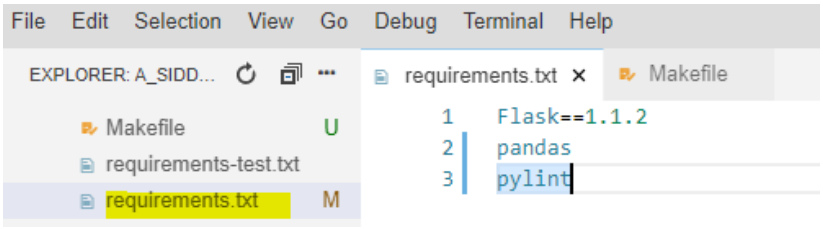
- Enhance an Analytics allocation

I can also add HTML code inside of triple code

```
@app.route('/html')
def html():
    return """
    <title> This is HTML title </title>
    <p> new HTML paragraph </p>
    <p><b> bolded paragraph </b></p>
    """
```



Add pandas and pylint to the requirements text

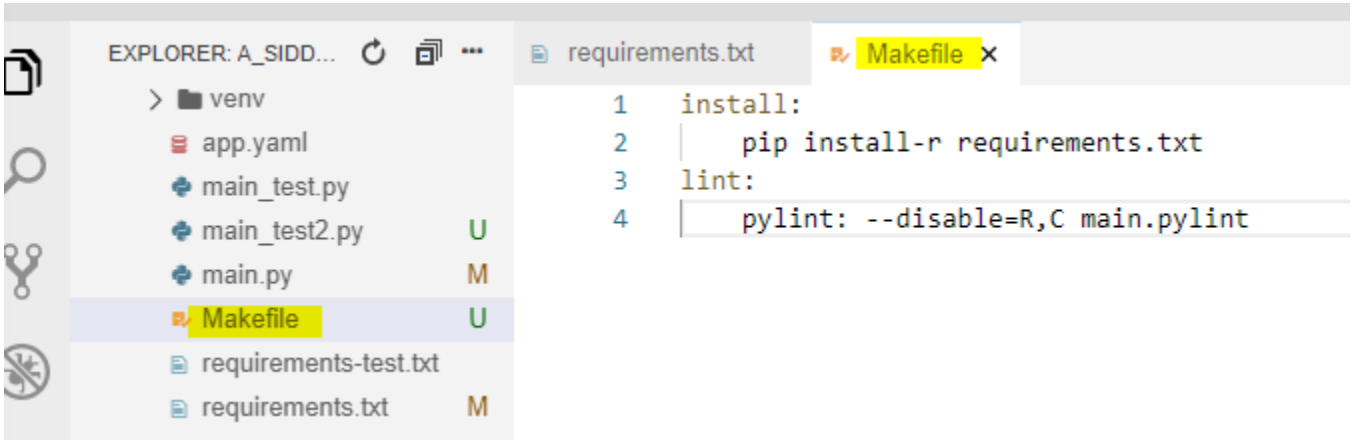


I can Makefile and put various useful commands that I could use throughout the course of working with my project.

```
$ touch Makefile
```

Toggle to "open edit"; click "tab size: 4" right bottom corner and click "indent using tabs." Copy and paste the following codes to the Makefile. Running Pylint ($ make lint) helps us to locate the error.

```
install:
    pip install -r requirements.txt
lint:
    pylint --disable=R,C main.py
```



I do not have to type the "`pip install -r requirements.txt`" command in anymore; it just does it for me. Now I can type make install, and it's going to look inside my requirements file and see that a new package has been selected and installed.

```
$ make install
$ make lint
```

Add the following code to the main.py:

```
import pandas as pd

@app.route('/pandas')
def pandas_df():
    df = pd.read_csv("https://raw.githubusercontent.com/noahgift/sugar/master/data/education_sugar_cdc_2003.csv")
    return jsonify(df.to_dict())
```

8080-cs-884463653701-default.us-west1.cloudshell.dev/pandas


```
{
  "<High school": {
    "0": "47.1 (37.8\u201356.5)",
    "1": "40.4 (30.9\u201350.7)",
    "2": "38.5 (34.2\u201343.0)",
    "3": "27.8 (22.4\u201333.9)",
    "4": "45.6 (36.4\u201355.2)",
    "5": "50.7 (44.3\u201357.0)",
    "6": "49.2 (40.0\u201358.5)",
    "7": "45.2 (40.9\u201349.7)",
    "8": "53.4 (48.6\u201358.1)",
    "9": "60.0 (53.3\u201366.5)",
    "10": "40.7 (35.2\u201346.5)",
```


- Enable functionality in an Analytics application

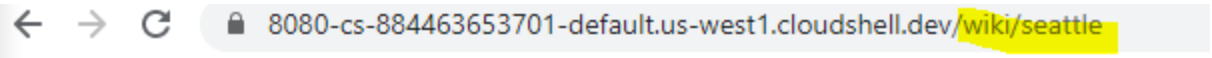
  Now we can add Wikipedia library to Flask app

requirements.txt Makefile

```
1  Flask==1.1.2
2  pandas
3  pylint
4  wikipedia
```

```
$ ipython
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:935: UserWarning: Attempting to work in a v
irtualenv. If you encounter problems, please install IPython inside the virtualenv.
  warn("Attempting to work in a virtualenv. If you encounter problems, please "
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.18.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import wikipedia

In [2]: result = wikipedia.summary("seattle", sentences=10)

In [3]: result
Out[3]: 'Seattle ( (listen) see-AT-əl) is a seaport city on the West Coast of the United States. It is the seat of
King County, Washington. Seattle is the largest city in both the state of Washington and the Pacific Northwest regi
on of North America. According to U.S. Census data released in 2019, the Seattle metropolitan area\'s population st
ands at 3.98 million, making it the 15th-largest in the United States. In July 2013, Seattle was the fastest-growin
g major city in the United States and remained in the top five in May 2015 with an annual growth rate of 2.1%. In J
uly 2016, Seattle was again the fastest-growing major U.S. city, with a 3.1% annual growthrate.Seattle is situated
on an isthmus between Puget Sound (an inlet of the Pacific Ocean) and Lake Washington. It is the northernmost large
 city in the United States, located about 100 miles (160 km) south of the Canadian border. A major gateway for trad
e with Asia, Seattle is the fourth-largest port in North America in terms of container handling as of 2015.The Seat
tle area was inhabited by Native Americans for at least 4,000 years before the first permanent European settlers. A
rthur A. Denny and his group of travelers, subsequently known as the Denny Party, arrived from Illinois via Portlan
d, Oregon, on the schooner Exact at Alki Point on November 13, 1851. The settlement was moved to the eastern shore
of Elliott Bay and named "Seattle" in 1852, in honor of ChiefSi\'ahl of the local Duwamish and Suquamish tribes.'
```

```
import wikipedia
@app.route('/wiki/<company>')
def wiki_route(company):
    result = wikipedia.summary(company, sentences=10)
    return result
```



Seattle ((listen) see-AT-əl) is a seaport city on the West Coast of the United States. It is the sea
largest city in both the state of Washington and the Pacific Northwest region of North America
2019, the Seattle metropolitan area's population stands at 3.98 million, making it the 15th-large
the fastest-growing major city in the United States and remained in the top five in May 2015 w
Seattle was again the fastest-growing major U.S. city, with a 3.1% annual growth rate.Seattle i
(an inlet of the Pacific Ocean) and Lake Washington. It is the northernmost large city in the Ur
south of the Canadian border. A major gateway for trade with Asia, Seattle is the fourth-largest
handling as of 2015.The Seattle area was inhabited by Native Americans for at least 4,000 yea
Arthur A. Denny and his group of travelers, subsequently known as the Denny Party, arrived fi
schooner Exact at Alki Point on November 13, 1851. The settlement was moved to the eastern
1852, in honor of Chief Si'ahl of the local Duwamish and Suquamish tribes.

```
from flask import Flask
from flask import Flask
from flask import jsonify
import pandas as pd
import wikipedia

app = Flask(__name__)

@app.route('/')
def hello():
    return 'Hello, I like to make AI apps'

@app.route('/name/<value>')
def name (value):
    val = {"value": value}
    return jsonify(val)

@app.route('/html')
def html():
    return """
    <title> This is HTML title </title>
    <p> new HTML paragraph </p>
    <p><b> bolded paragraph </b></p>
    """
@app.route('/pandas')
def pandas_df():
    df = pd.read_csv("https://raw.githubusercontent.com/noahgift/sugar/master/data/
education_sugar_cdc_2003.csv")
    return jsonify(df.to_dict())

@app.route('/wiki/<company>')
def wiki_route(company):
    result = wikipedia.summary(company, sentences=10)
    return result

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8080, debug=True )
```

[4]:
```
# Imports the Google Cloud client library
from google.cloud import language
from google.cloud.language import enums
from google.cloud.language import types
import wikipedia
```

Entity Extraction

[6]:
```
#result = "Bill Gates was born in Seattle"
result = wikipedia.summary("seattle", sentences=10)
client = language.LanguageServiceClient()
document = types.Document(
content=result,
        type=enums.Document.Type.PLAIN_TEXT)
entities = client.analyze_entities(document).entities
print(str(entities))
```

```
[name: "Seattle"
type: LOCATION
metadata {
  key: "mid"
  value: "/m/0d9jr"
}
metadata {
  key: "wikipedia_url"
  value: "https://en.wikipedia.org/wiki/Seattle"
}
salience: 0.8071296811103821
mentions {
  text {
    content: "Seattle"
    begin_offset: -1
  }
  type: PROPER
}
mentions {
  text {
    content: "seaport city"
    begin_offset: -1
  }
  type: COMMON
}
mentions {
  text {
    content: "Seattle"
    begin_offset: -1
```

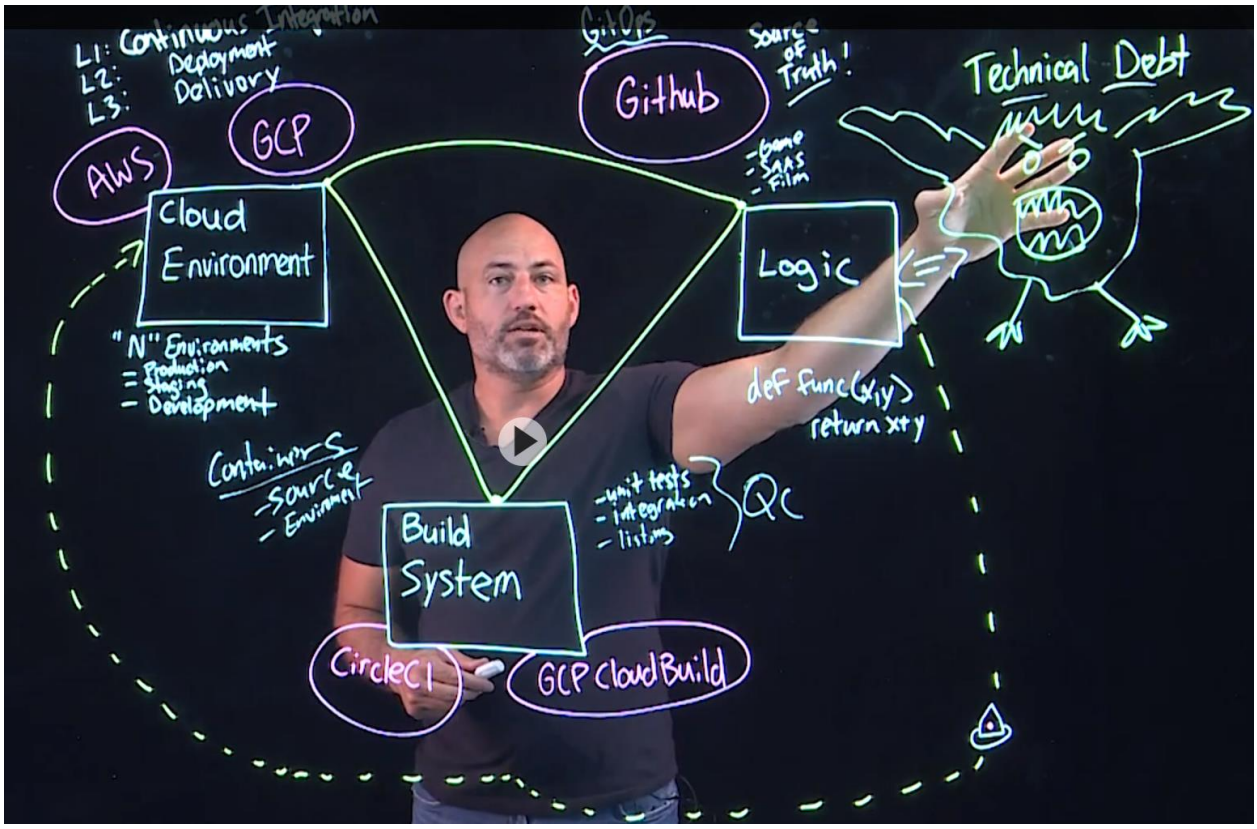
Lesson 7 – Using Build Systems and Containers

- Architect continous deployment systems

The important concept in here is continuous integration, continuous deployment, and continuous delivery.

- Continuous integration gives you the ability to constantly test your source code. And so, you can do continuous integration just locally on your laptop. You can have a make file, you can have linting, you can have testing.
- Continuous deployment: you integrate your code that with containers or a configuration management system, and you can constantly test and deploy your software into production.
- Continuous delivery is the next level up. Which is that you're constantly deploying your software that's quality controlled, but it doesn't have to be deployed. For a business reason, you may choose not to deploy, but you have a production quality deployment that's constantly being generated.

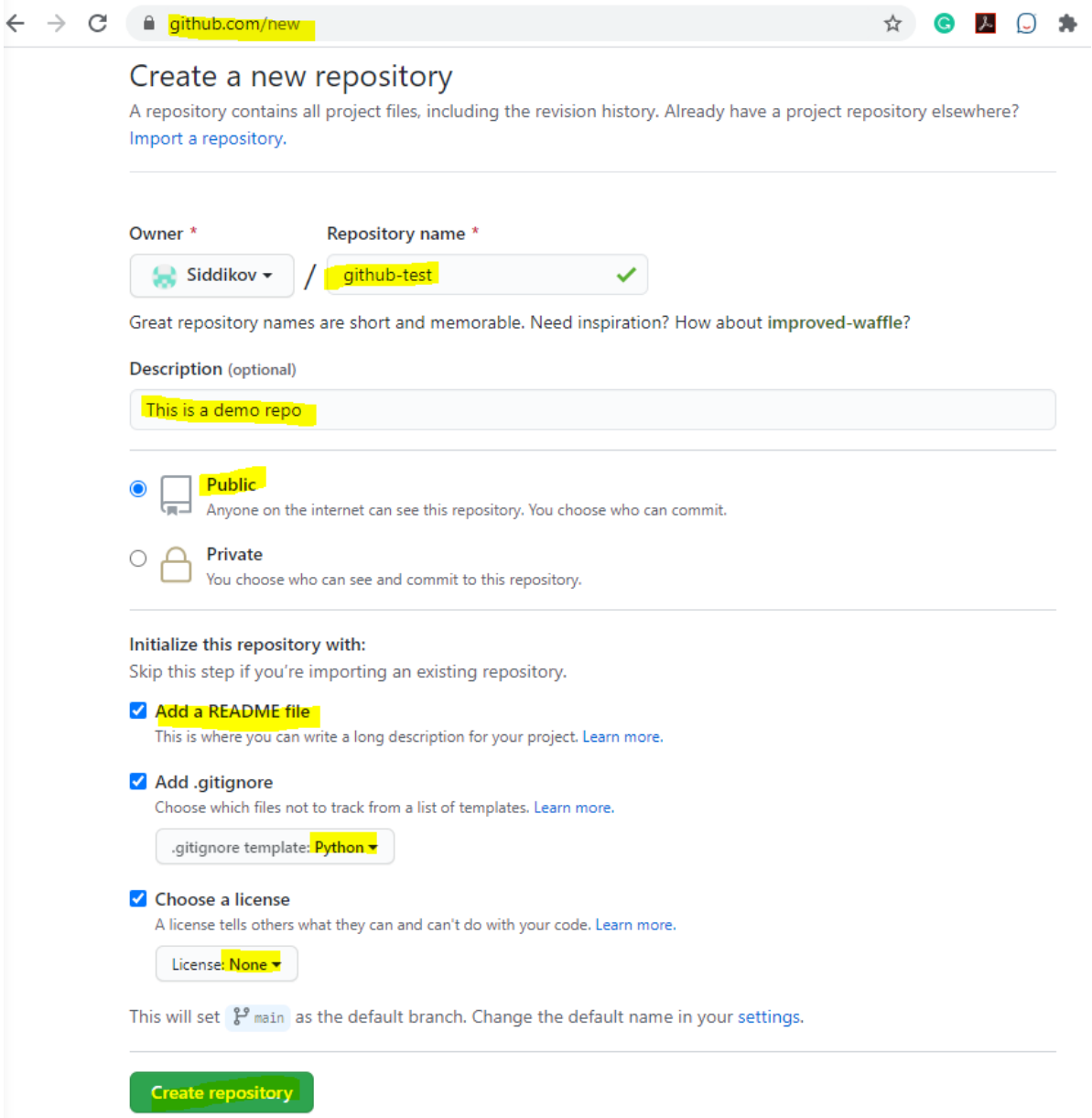
Github becomes a source of truth where you have access to the logic (prototype), all the testing, and all of the things that you need to deploy your software (entire application, the containers, the configuration management).
Logic, build system, cloud environment, and the whole thing together allows you to do continuous integration, deployment, and delivery. Please see the graph below.



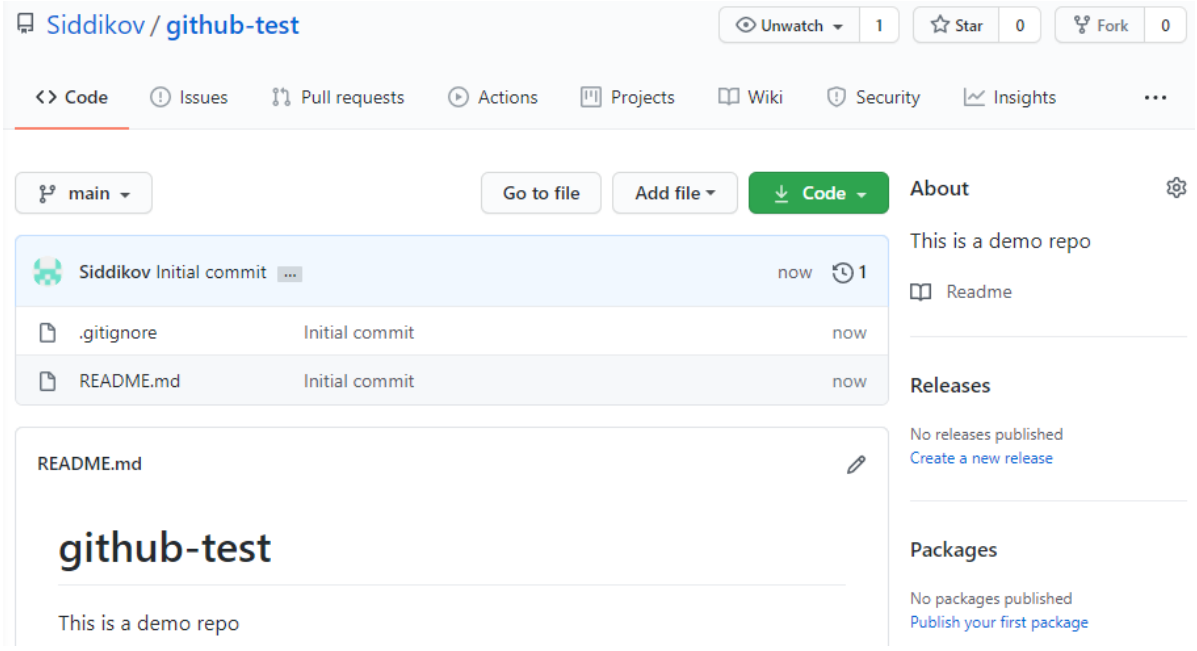
- Configure your repository in GitHub

Here is the step by step instructions to create a repository in GitHub

  - README: You can add a README file to your repository to tell other people why your project is useful, what they can do with your project, and how they can use it.
  - GitIgnore: You can create a gitignore file in your repository's root directory to tell Git which files and directories to ignore when you make a commit. To share the ignore rules with other users who clone the repository, commit the gitignore file into your repository. We will select Python and it gets rid of basically garbage files like files that I don't want keep track of.
  - License: it is not a bad idea especially with an opensource project to select some kind of a license.



Here is what we see after creating the new GitHub repository.



Now we need to go to the GCP cloud shell and connect our GitHub repository. To communicate with GitHub via an encrypted protocol, we need to type the followings to the GCP shell command:

```
$ ssh-keygen -t rsa
And click return key four times
```

- ssh-keygen is a standard component of the Secure Shell (SSH) protocol suite found on Unix, Unix-like and Microsoft Windows computer systems used to establish secure shell sessions between remote computers over insecure networks, through the use of various cryptographic techniques. The ssh-keygen utility is used to generate, manage, and convert authentication keys.
- RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission. It is also one of the oldest.

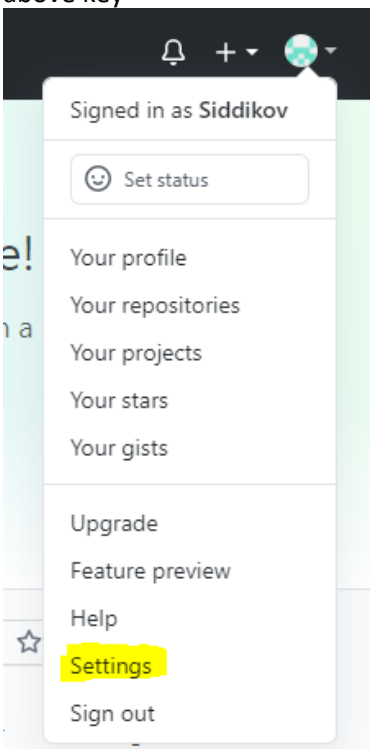
ssh-keygen command options	description
-b bits	Specifies the number of bits in the key to create. The default length is 3072 bits (RSA) or 256 bits (ECDSA).
-C comment	Provides new comment.
-p	Requests changing the passphrase of a private key file instead of creating a new private key.
-t	Specifies the type of key to create.
-o	Use the new OpenSSH format.
-q	quiets ssh-keygen. It is used by the /etc/rc file while creating a new key.
-N	Provides a new Passphrase.
-B	Dumps the key's fingerprint in Bubble Babble format.
-l	Dumps the key's fingerprint in SHA-2 (or MD5) format.

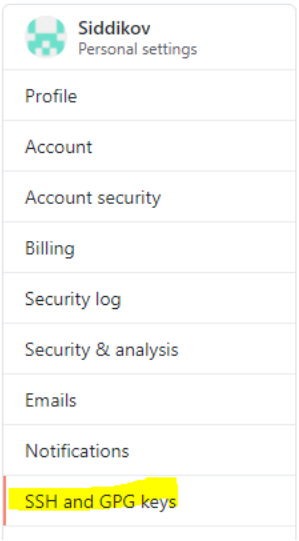
```
a_siddikov2@cloudshell:~ (msds-498-group-project)$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/a_siddikov2/.ssh/id_rsa):
Created directory '/home/a_siddikov2/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/a_siddikov2/.ssh/id_rsa.
Your public key has been saved in /home/a_siddikov2/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:WxPgMcneeOX8Ix6CUaB7ny7WRH9brRRmiz3HlcSGSqI a_siddikov2@cs-884463653701-default-boost-b5khd
The key's randomart image is:
+---[RSA 2048]----+
|       .=o    o  |
|       oo=.... + |
|      ..o+++. o .|
|       E+ +oo +..|
|      . S=o. * +o|
|       ..+oo= O =|
|         .ooo = B |
|          o.. . o  |
|         . ..      |
+----[SHA256]-----+
a_siddikov2@cloudshell:~ (msds-498-group-project)$
```

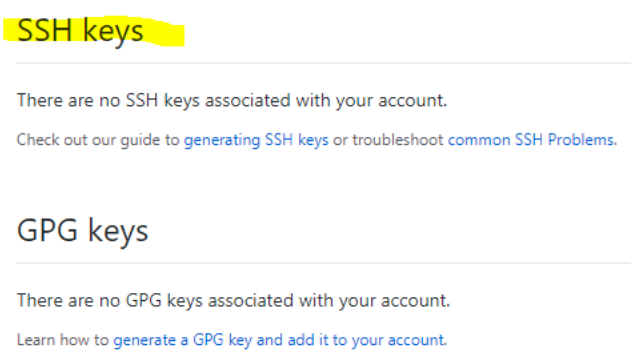
~/ (tilde-slash) refers to your home directory and. (dot) refers to the current working directory

```
a_siddikov2@cloudshell:~ (msds-498-group-project)$ cd ~/.ssh
a_siddikov2@cloudshell:~/.ssh (msds-498-group-project)$ ls
  id_rsa id_rsa.pub
a_siddikov2@cloudshell:~/.ssh (msds-498-group-project)$ cat id_rsa.pub
  ssh-rsa
  AAAAB3NzaC1yc2EAAAADAQABAAABAQDd6cKGj4XIlrjXRZv/WaW6D4K2PcyJin4shG61U5DaipWVEk/iE1
  rIAkh0Ap3LsYTZ0Enn7+OJG1nJjQhrEZr2TWvkBxy4LuPmQlLtGG6qCLybI82JHRqM0a8MZccZlZBPOa9L
  xHjaTlAmK+HV9RmdbG+mh5e1Dzm4hpc1hm3NzaYE0wuXpRaf5w9lgs0SVMhhuowTzTT5991RodlpOsjdhw
  585xmWwRsy4wb23fffV5ix8sBW1F45EpbkaOibHjpoNxN5Fyxxq4P9hHFYUAbtyZOTPi29iCoWHurEkyBD
  YeJfC0zQ57ViJJe662ArFAto/irZuG1GiPfHLXp+paHV a_siddikov2@cs-884463653701-default-
  boost-b5khd
```

Now we need to go to the GitHub → settings → “SSH and GPG keys” → New SSH key → copy and paste the above key







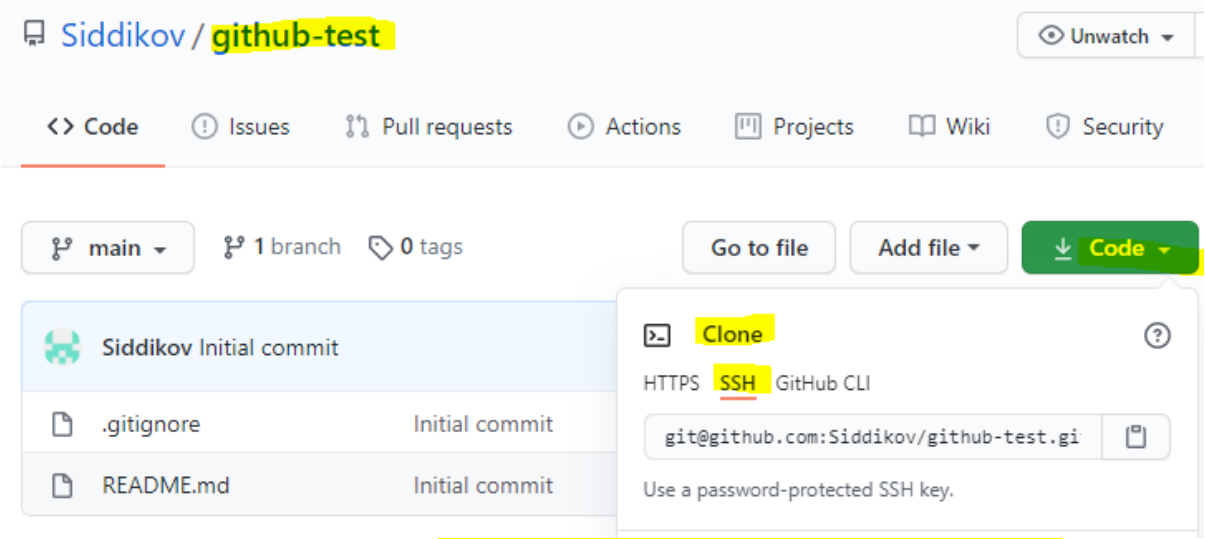
Title

gcp-cloud-key-demo

Key

ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDd6cKGj4XIlrjXRZv/WaW6D4K2PcyJin4shG61U5DaipWVEk/iE1rlAkh0Ap3
LsYTZ0Enn7+OJG1nJjQhrEZr2TWvkBxy4LuPmQILtGG6qCLybI82JHRqM0a8MZccZIZBPOa9LxHjaTlAmK+HV9RmdbG
+mh5e1Dzm4hpc1hm3NzaYE0wuXpRaf5w9Igs0SVMhhuowTzTT5991RodIpOsjdhw585xmWwRsy4wb23fffV5ix8sBW
1F45EpbkaOibHjpoNxN5Fyxxq4P9hHFYUAbtyZOTPi29iCoWHurEkyBDYeJfC0zQ57ViJJe662ArFAto/irZuG1GiPfHLXp+
paHV a_siddikov2@cs-884463653701-default-boost-b5khd

Now I need to go back to the github-test and then clone with ssh



Copy and past the GitHub SSH code (`git@github.com:Siddikov/github-test.git`) to the GCP shell

```
a_siddikov2@cloudshell:~ (msds-498-group-project)$ mkdir src
a_siddikov2a_siddikov2@cloudshell:~ (msds-498-group-project)$ cd src
a_siddikov2@cloudshell:~/src (msds-498-group-project)$ git clone
git@github.com:Siddikov/github-test.git
  Cloning into 'github-test'…
```

To make an easy way to keep track of our virtual environment, we need to create the same a virtual environment name as the project name in our home directory (~/ tilde slash).

```
a_siddikov2@cloudshell:~/src (msds-498-group-project)$ virtualenv ~/.github-test
a_siddikov2@cloudshell:~/src (msds-498-group-project)$ source ~/.github-
test/bin/activate
(.github-test) a_siddikov2@cloudshell:~/src (msds-498-group-project)$ pwd
  /home/a_siddikov2/src
```

[OPTIONAL STEP] Here is how we can change the directory and source the virtual environment automatically. Basically, creating a shortcut to the virtual environment (`source ~/.github-test/bin/activate`).

Click the “open editor” → File → Open → scroll down and click the “.bashrc” and then add this code

```
#alias
alias github-test="cd ~/src/github-test && source ~/.github-test/bin/activate"
```

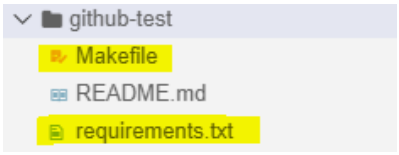
requirements.txt | Preview README.md | .bashrc | Makefile

```
    fi
fi
source /google/devshell/bashrc.google
#alias
alias github-test="cd ~/src/github-test && source ~/.github-test/bin/activate"
```

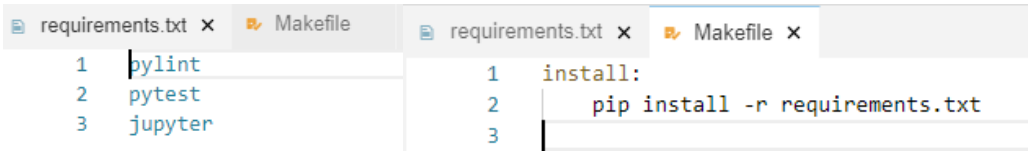
- Configure makefile in Google Cloud Shell

We need to create the Makefile and requirements by running the following command

```
(.github-test) a_siddikov2@cloudshell:~/src/github-test (msds-498-group-project)$
touch Makefile && touch requirements.txt
```



Please refer the lesson 6 how to create Makefile and requirements.txt. We will automate the dependencies.




```
$ make install
```

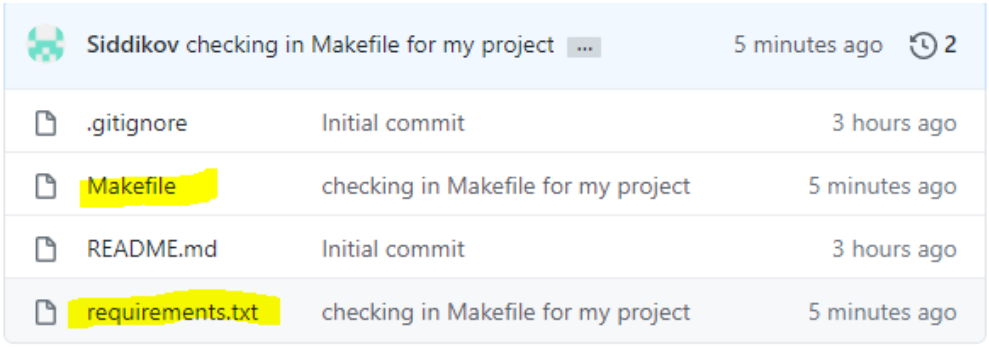
We can check what files we have in GitHub

```
(.github-test) a_siddikov2@cloudshell:~/src/github-test (msds-498-group-project)$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Makefile
        requirements.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
$ git status
$ git add Makefile
$ git add requirements.txt
$ git commit -m "checking in Makefile for my project"
$ git push
```

```
(.github-test) a_siddikov2@cloudshell:~/src/github-test (msds-498-group-project)$ git commit -m "checki
ng in Makefile for my project"

[main 87354d8] checking in Makefile for my project
 2 files changed, 5 insertions(+)
 create mode 100644 Makefile
 create mode 100644 requirements.txt
(.github-test) a_siddikov2@cloudshell:~/src/github-test (msds-498-group-project)$ git push
Warning: Permanently added the RSA host key for IP address '192.30.255.113' to the list of known hosts.
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 437 bytes | 437.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:Siddikov/github-test.git
   40eb6d5..87354d8  main -> main
```

Now I can check the GitHub. I can see that now there are Makefile and requirements.txt files.



- ## Build your project in CircleCI

  It is imperative to automate the stages of package development. Simultaneously, ensure that if an updated code or a new function has been added, nothing breaks. That's the point of using the continuous integration/ Deploy and Delivery platforms — CI/CD pipeline. The pipeline ensures everything is working in the flow according to the following chart:

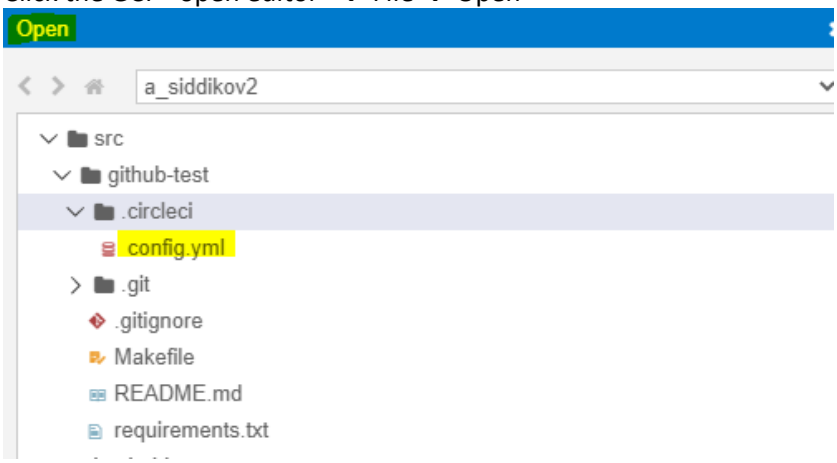
  build →Test → Merge → Auto release to public repo → auto deployments

  There are plenty of hosted platforms that used to the CI/CD pipelines, to name a few:
  - CircleCI
  - Travis
  - GitLab CI
  - Jenkins and many others..etc.

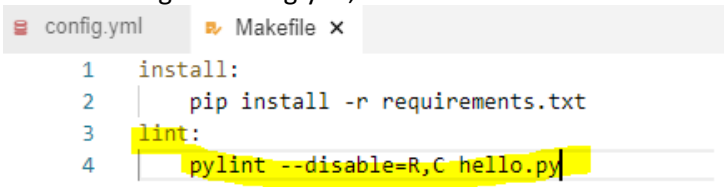
Once I login CircleCi with GitHub credentials, I need to go to GCP and add YML file

```
(.github-test) a_siddikov2@cloudshell:~/src/github-test (msds-498-group-project)$
mkdir -p .circleci && touch .circleci/config.yml
```

Click the GCP "open editor" → File → Open



Before editing the config.yml, we should add the lint to our Makefile:



Click config.yml, then copy and paste the code by visiting the following website:
https://github.com/noahgift/functional_intro_to_python/blob/master/.circleci/config.yml

```
# Python CircleCI 2.0 configuration file
#
# Check https://circleci.com/docs/2.0/language-python/ for more details
#
version: 2
jobs:
  build:
    docker:
      # specify the version you desire here
      # use `-browsers` prefix for selenium tests, e.g. `3.6.1-browsers`
      - image: circleci/python:3.6.3-stretch

      # Specify service dependencies here if necessary
      # CircleCI maintains a library of pre-built images
      # documented at https://circleci.com/docs/2.0/circleci-images/
      # - image: circleci/postgres:9.4

    working_directory: ~/repo

    steps:
      - checkout

      # Download and cache dependencies
      - restore_cache:
          keys:
          - v1-dependencies-{{ checksum "requirements.txt" }}
          # fallback to using the latest cache if no exact match is found
          - v1-dependencies-

      - run:
          name: install dependencies
          command: |
            python3 -m venv venv
            . venv/bin/activate
            make install
      - save_cache:
          paths:
            - ./venv
          key: v1-dependencies-{{ checksum "requirements.txt" }}

      # run lints!
      - run:
          name: run lint
          command: |
            . venv/bin/activate
            make lint
```

Then I need to create the hello.py

```
(.github-test) a_siddikov2@cloudshell:~/src/github-test (msds-498-group-project)$
touch hello.py
```

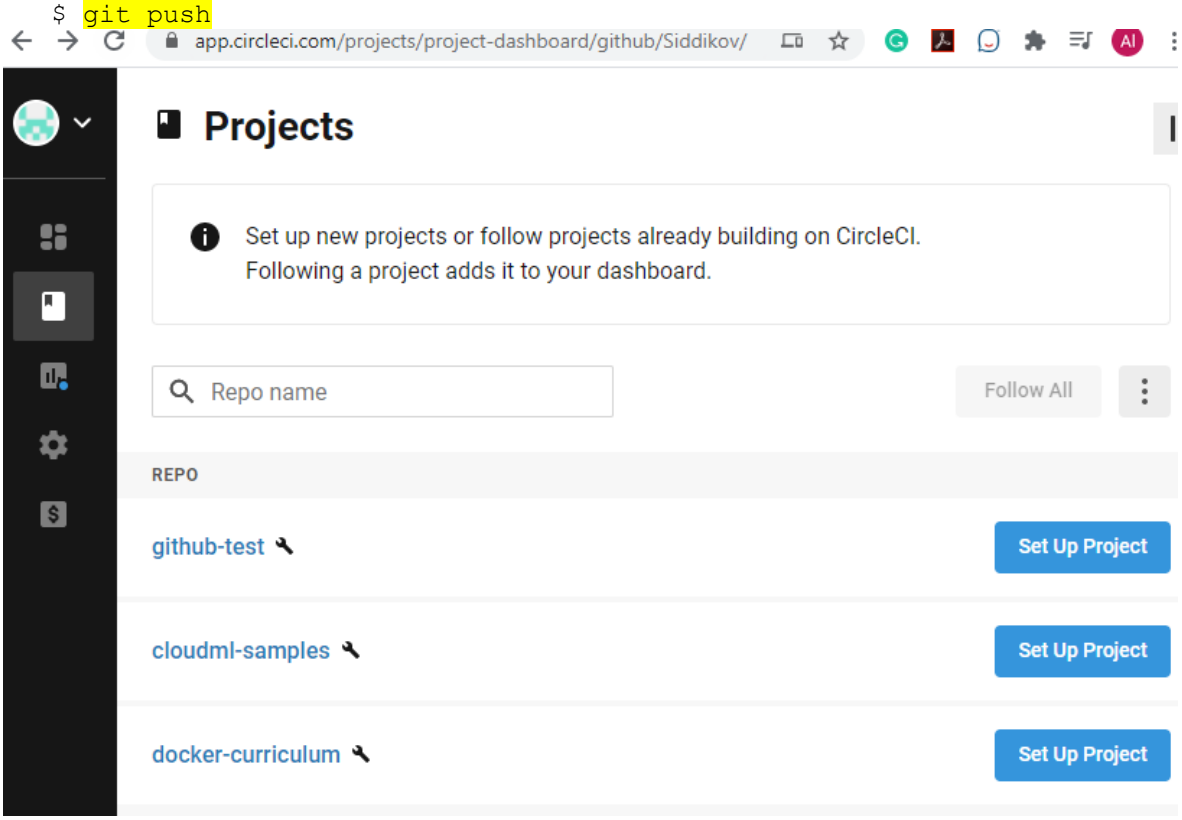
Open the editor, locate the hello.py file and add any code.

config.yml | hello.py | Makefile

```
1  def test():
2      return "lint worked"
```

Go back to shell and type the followings one by one:

```
$ git status
$ git add Makefile
$ git add .circleci
$ git add hello.py
$ git commit -m "setting up circleci"
$ git push
```



To get the updated code from GitHub, use git pull.

```
$ git pull
```

Other Resources:

/home/a_siddikov2/python-docs-samples/appengine/standard_python3/hello_world

https://www.earthdatascience.org/courses/intro-to-earth-data-science/open-reproducible-science/bash/bash-commands-to-manage-directories-files/

Intro to earth data science
ctbook

UNITS

1. INTRODUCTION TO OPEN
PRODUCIBLE SCIENCE
ORKFLOWS
CHAPTER 1 OPEN REPRODUCIBLE
IENCE WORKFLOWS
About Open Science
Open Science Tools
Project Management Best
ractices
CHAPTER 2 USE BASH TO
ANIPULATE FILES
Introduction to Bash
Bash Commands
CHAPTER 3 JUPYTER FOR PYTHON
Intro to Jupyter
Jupyter Notebook For Python

Lesson 2. Bash Commands to Manage Directories and Files

Jenny Palomino, Leah Wasser

ON THIS PAGE

- LEARNING OBJECTIVES
- HOW TO RUN BASH COMMANDS IN THE TERMINAL
- USEFUL BASH COMMANDS TO MANAGE DIRECTORIES AND FILES
- PRACTICE YOUR BASH SKILLS

Learning Objectives

- Run `Bash` commands to complete the following tasks:
  - print the current working directory (`pwd`)
  - navigate between directories on your computer (`cd`)
  - create new directories (`mkdir`)
  - print a list of files and subdirectories within directories (`ls`)
  - delete files (`rm`) and directories (`rm -r`)
  - copy files (`cp`) and directories (`cp -r`) to another directory
  - easily create new files using a single command (`touch`)

How to Run Bash Commands in the Terminal

Unix: https://www.ohio.edu/mechanical/programming/lab1/basic_UNIX.html
https://www.git-tower.com/learn/git/ebook/en/command-line/appendix/command-line-101/