PySpark Application

Appling a PySpark model on a web site

Export PMML file from PySpark model

- Import libraries into your PySpark cluster
  - jpmml-sparkml (version varies with Spark version!)
    - Import this Java library using maven
  - pyspark2pmml
    - Import from PyPi
- Within a PySpark notebook
  - Create a model
    - The input DataFrame for the ML pipeline (training data) must have the fields you will want as input to your model when the application run
      - The arguments that will be passed to your model
      - The values should be Double or Integer
  - Save the model as a PMML file
  - Download the PMML file and place with Python files for your app

Workspace

Workspace
- Shared
- Users
- 1. Introduction
- 2. Data operations
- 3. ML 1
- 4. ML 2
- 5. Pipelines
- 6. Internals
- 7. Application
- Answers
- Archive
- bs4
- DG_examples
- jpmml-sparkml-1.5.8
- M1
- M2Student
- pyspark2pmml
- Tips

7. Application
- 1. PMML demo

1. PMML demo (Python)

Cmd 1

Download the data

```
%sh
wget https://raw.githubusercontent.com/wrobinson-gsu/bdUtils/master/data/StrokeData.csv
```

```
--2020-04-20 15:57:06--  https://raw.githubusercontent.com/wrobinson-gsu/bdUtils/master/data/StrokeData.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.52.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.52.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2853518 (2.7M) [text/plain]
Saving to: 'StrokeData.csv.2'

     0K .......... .......... .......... .......... ..........  1% 3.99M 1s
    50K .......... .......... .......... .......... ..........  3% 8.00M 0s
   100K .......... .......... .......... .......... ..........  5% 37.2M 0s
   150K .......... .......... .......... .......... ..........  7% 10.1M 0s
   200K .......... .......... .......... .......... ..........  8% 27.1M 0s
   250K .......... .......... .......... .......... .......... 10%  128M 0s
   300K .......... .......... .......... .......... .......... 12%  166M 0s
   350K .......... .......... .......... .......... .......... 14%  162M 0s
   400K .......... .......... .......... .......... .......... 16% 9.53M 0s
   450K .......... .......... .......... .......... .......... 17%  209M 0s
   500K .......... .......... .......... .......... .......... 19%  222M 0s
   550K .......... .......... .......... .......... .......... 21% 85.2M 0s
```

Command took 0.33 seconds -- by wrobinson@gsu.edu at 4/20/2020, 11:57:05 AM on db6.4

Cmd 2

Read the data

```
from pyspark.sql.types import DoubleType, StringType, StructType, StructField

schema = StructType([ StructField("gender", StringType(), True),
                      StructField("age", DoubleType(), True),
```

Model Scoring using regression

Model score for stroke is 0.0

Feature	Value
gender	0.0
age	0.0
BMI	0.0
heart_disease	0.0
stroke	0.0
smoking_history	0.0
hypertension	0.0
diabetes	0.0

Update

Read the PMML model in Python

```
def get_model():
    '''
    Read PMML model from file, and read the input parameter names for the model.
    :return: model and its parameter names
    '''
    global model, params
    if model is None:
        # On Windows, path to the file
        if os.name == 'nt':
            model_path = os.path.join("//U:/dev/docker/flask-pmml/flask_app/",
                                      "model.pmml")
            print("loading model from ", model_path)
            model = Model.fromFile(model_path)
        # In Docker, python runs in the app directory
        else:
            model = Model.fromFile("model.pmml")
        params = {fn: 0.0 for fn in model.dataDictionary.fieldNames}
    return model, params
```

Display the model

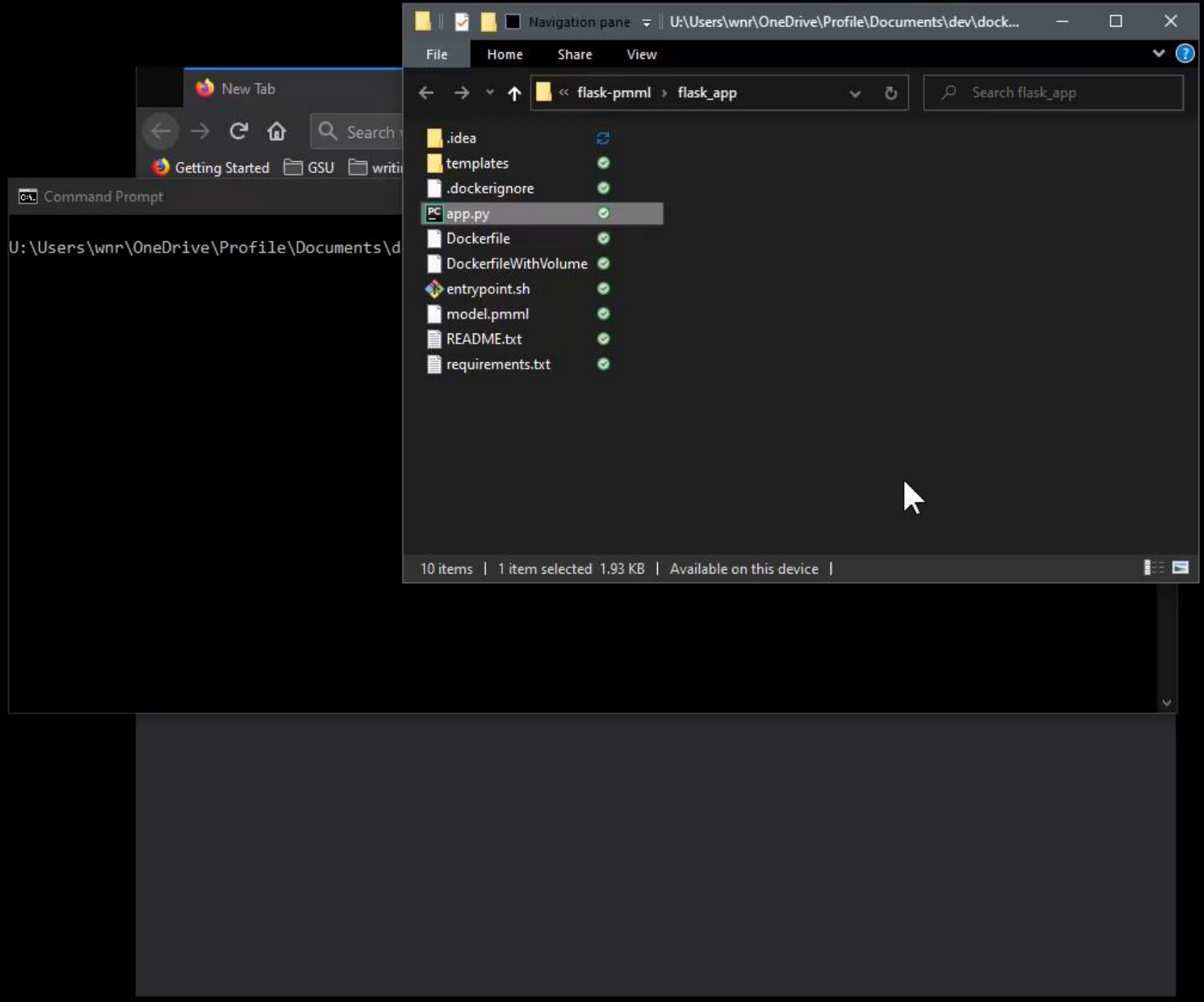
```python
@app.route('/model', methods=['POST', 'GET'])
def model_update():
    '''
    Generates HTML using the model parameters, prediction label name, and model name.
    :return: the HTML template for the model.
    '''
    global model, params
    model, params = get_model()
    score = 0.0
    if request.method == 'POST':
        params = request.form
        prediction = model.predict(params)
        score = prediction["prediction"]
    param_pairs = zip(params.keys(), params.values())
    return render_template("model.html", parameters=param_pairs, score=score,
                           label=model.targetName, model=model.functionName)
```

```html
<title>Model Score</title>
<div class=page>
  <h1>Model Scoring using {{ model }}</h1>

  <h2>Model score for {{ label }} is {{ score }}</h2>

  <form action="/model" method="post">
  <table>
      <tr>
          <th>Feature</th>
          <th>Value</th>
      </tr>
      {% for item in parameters %}
          <tr><td>{{item[0]}}</td>
              <td><input name="{{item[0]}}" value="{{item[1]}}"></td>
      </tr>
  {% endfor %}
  </table>
  <input type="submit" value="Update">
  </form>
```

Run in Docker on Windows

PMML in Python on Windows Docker

- Ensure Docker is started
- In shell
  - docker build -t pmmlserver:1.0 .
  - docker run -p:5000:5000 -d pmmlserver:1.0
- Open browser to
  - localhost:5000


Navigation pane
U:\Users\wnr\OneDrive\Profile\Documents\dev\dock...
File
Home
Share
View
flask-pmml
flask_app
Search flask_app
.idea
templates
.dockerignore
app.py
Dockerfile
DockerfileWithVolume
entrypoint.sh
model.pmml
README.txt
requirements.txt
10 items | 1 item selected 1.93 KB | Available on this device
New Tab
Search
Getting Started
GSU
Command Prompt
U:\Users\wnr\OneDrive\Profile\Documents\d

Linux Docker on Windows

- Install Docker on WSL (Windows Subsystem for Linux)
- Optional! Generally, a better, faster environment on Windows, but requires carefully following the steps listed
  - https://nickjanetakis.com/blog/using-wsl-and-mobaxterm-to-create-a-linux-dev-environment-on-windows
  - https://nickjanetakis.com/blog/setting-up-docker-for-windows-and-wsl-to-work-flawlessly
    - Note, don't install the terminal. He no longer recommends it.
      - He does have an interesting list of tools
      - More about Pythonic coding here
    - Note, after install docker compose (pip install --user docker-compose), you will need to do:
      - source ./profile
    - This ensures that the directory .local/bin is added to the path

Build Docker Image with Cloud Build and Push to Google Container Registry

- On Google Cloud Platform (GCP)
  1. Create a project
  2. Open the Cloud Shell
  3. Upload docker application files into shell directory
  4. Select the appropriate Dockerfile
     - Simpler to NOT mount volume, but instead include files in image
  5. Google Cloud Build in Cloud Shell
     - gcloud builds submit --tag gcr.io/[PROJECT_ID]/flaskpmml .
  6. View in Container Registry

https://medium.com/google-cloud/deploy-serverless-container-google-cloud-run-68d716af7716; https://cloud.google.com/cloud-build/docs/quickstart-build

DASHBOARD ACTIVITY CUSTOMIZE

How Google Cloud is helping during COVID-19 Learn more DISMISS

Project info

Project name
cis8795-spark

Project ID
cis8795-spark

Project number
158923663093

ADD PEOPLE TO THIS PROJECT

Go to project settings

Resources

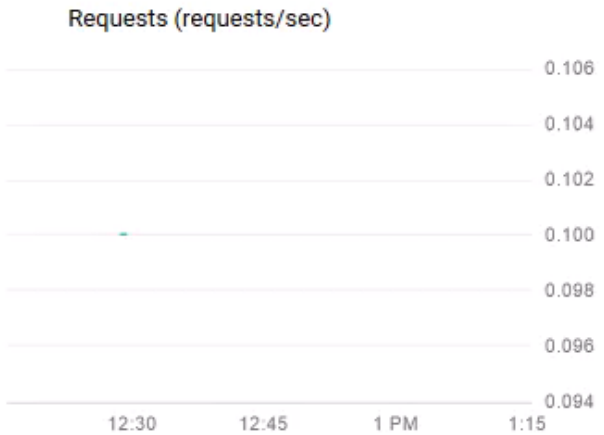
Storage
3 buckets

Trace

No trace data from the past 7 days

Get started with Stackdriver Trace

APIs



Go to APIs overview

Google Cloud Platform status

All services normal

Go to Cloud status dashboard

Billing

Estimated charges USD $0.00
For the billing period Apr 1 – 20, 2020

View detailed charges

Error Reporting

No sign of any errors. Have you set up Error Reporting?

Learn how to set up Error Reporting

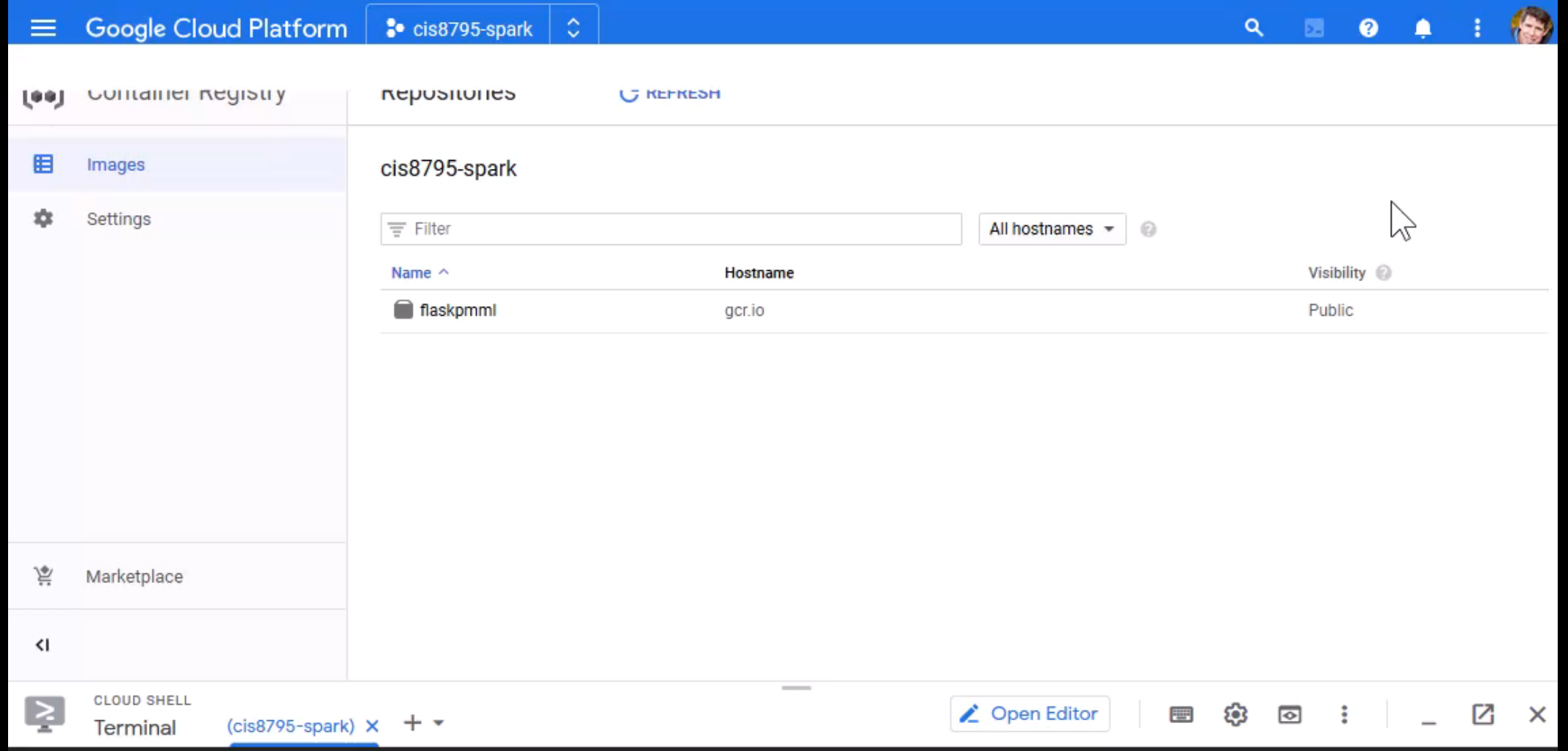
News

Keep your teams working safely with BeyondCorp Remote Access

Run Docker container in Cloud Run

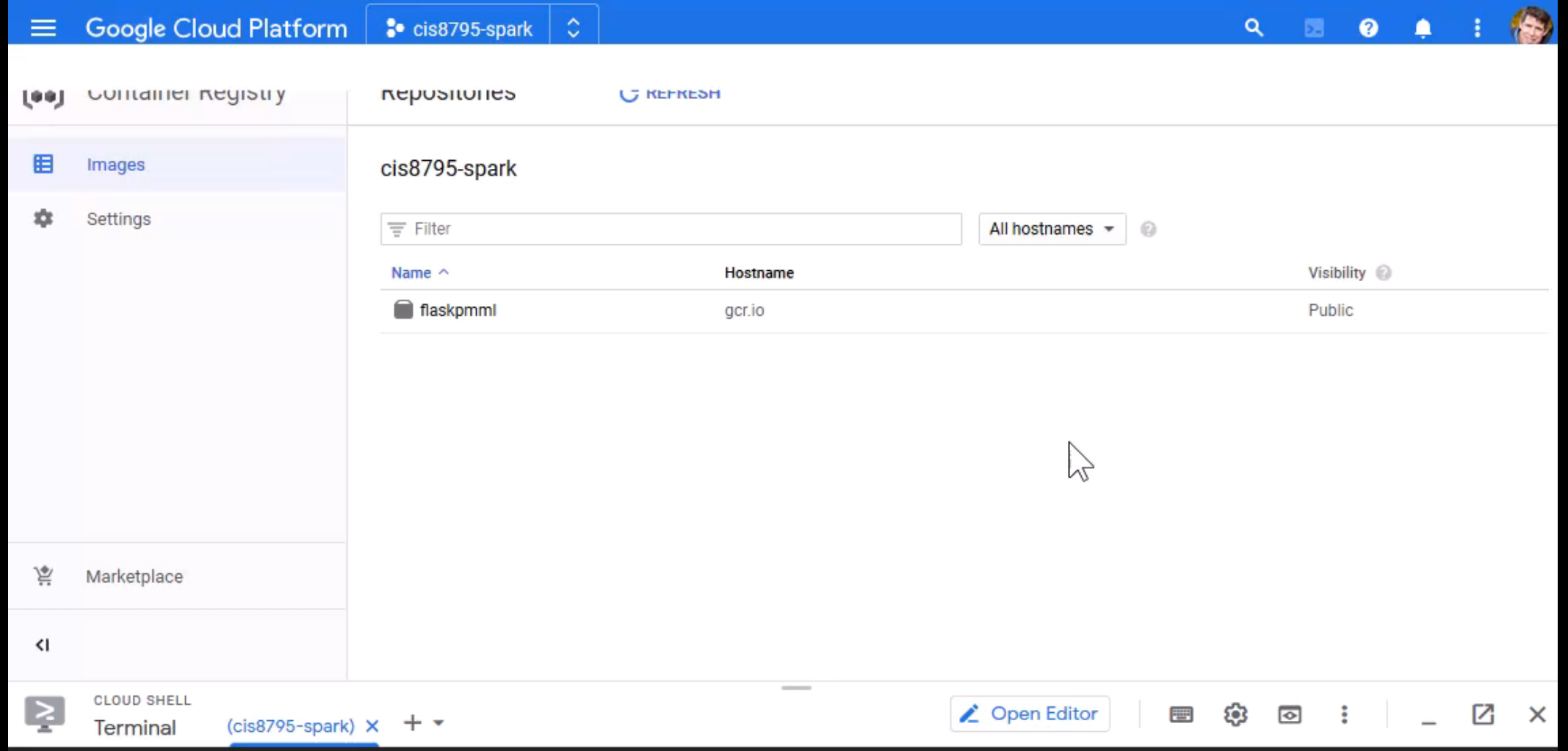
- In GCP
  - Open the Container Registry
  - Select your container
  - Select Deploy to Cloud Run

https://medium.com/google-cloud/deploy-serverless-container-google-cloud-run-68d716af7716


Google Cloud Platform
cis8795-spark
Container Registry
Repositories
REFRESH
Images
Settings
Marketplace
cis8795-spark
Filter
All hostnames
Name
Hostname
Visibility
flaskpmml
gcr.io
Public
CLOUD SHELL
Terminal
(cis8795-spark)
Open Editor
5bef08742407: Layer already exists
18499916f54d: Pushed
327cac6db23b: Pushed
805d4959c18f: Pushed
ed58cf449dc4: Pushed
latest: digest: sha256:1a8f1f82a64971dbe55328dc79e6111e203eac14e5eaf750977db533226a53de size: 2410
DONE
ID CREATE_TIME DURATION SOURCE
IMAGES STATUS
70e6778f-8bd4-4a85-9fb2-cdb4b9807a3b 2020-04-20T17:21:50+00:00 35S gs://cis8795-spark_cloudbuild/source/1587403310.1-03a56b3b977a4f558
29e71b3b8314e9.tgz gcr.io/cis8795-spark/flaskpmml (+1 more) SUCCESS
robinson_wn@cloudshell:~/flask_app (cis8795-spark)$

Run Docker container in Google Kubernetes

- In GCP
  - Open the Container Registry
  - Select your container
  - Select Deploy to GKE
    - Ensure there is enough memory for your app


Google Cloud Platform
cis8795-spark
Container Registry
Repositories
REFRESH
Images
Settings
cis8795-spark
Filter
All hostnames
Name
Hostname
Visibility
flaskpmml
gcr.io
Public
Marketplace
CLOUD SHELL
Terminal
(cis8795-spark)
Open Editor
5bef08742407: Layer already exists
18499916f54d: Pushed
327cac6db23b: Pushed
805d4959c18f: Pushed
ed58cf449dc4: Pushed
latest: digest: sha256:1a8f1f82a64971dbe55328dc79e6111e203eac14e5eaf750977db533226a53de size: 2410
DONE
ID CREATE_TIME DURATION SOURCE IMAGES
STATUS
70e6778f-8bd4-4a85-9fb2-cdb4b9807a3b 2020-04-20T17:21:50+00:00 35S gs://cis8795-spark_cloudbuild/source/1587403310.1-03a56b3b977a4f558
29e71b3b8314e9.tgz gcr.io/cis8795-spark/flaskpmml (+1 more) SUCCESS
robinson_wn@cloudshell:~/flask_app (cis8795-spark)$