

Pipeline Design Manual

In this manual I will explain the flow developed to design the pipeline deploy in the cloud CD/CI service called Semaphore to test a group of Jupyter notebooks controlled by repository in Github

Docker Github Project

This github project is published under name: **freelancer-atamic-docker**. This project configure the docker image used like runtime in the Semaphore pipeline.

The files included in it are:

- **Dockerfile**: docker image configuration file
- **README**: a simple github repository explanation file

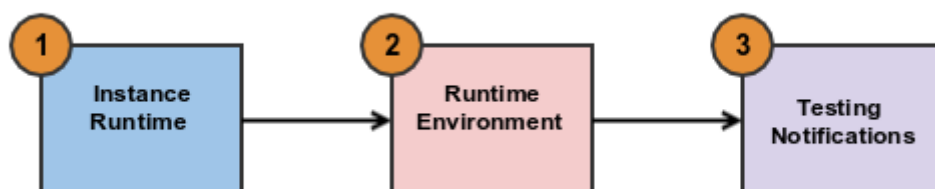
Semaphore Pipeline Github Project

This project is published under name: **freelancer-atamic**. This project configure the pipeline used to test several Jupyter Notebooks.

The files included in it are:

- **.gitignore**: ignore github configuration file.
- **envirotment.yml**: conda dependencies environmentconfiguration.
- **.semaphore/ semaphore.yml**: is the default pipeline configuration file.
- **configuration.yml**: configuration folders file where the Jupyter Notebooks are saved. By default **all notebooks are located in a folder called notebooks**.
- **notebook_tester.py**: python service to test all jupyter notebooks.
- **README**: a simple github repository explanation file.
- **Tutorial.pdf**: the filw where explain the Pipeline designed.

Semaphore Pipeline Flow



The Jobs of the unique block of the pipeline flow in Semaphore are:

1. Instance Runtime:

In this step the **Semaphone Agent** get the Custom Docker Image from Docker hub called **<GITHUB_ID>/atamic-python:latest** from your account. This image is customize from default Semaphore Python image called **semaphoreci/python:3.7.4** including the last version of conda.

2. Runtime Environment:

In this step a Semaphore configure a conda environment with all conda and pip Python packages to test the Jupyter notebooks. All dependencies are configured on file **environment.yml** used by conda

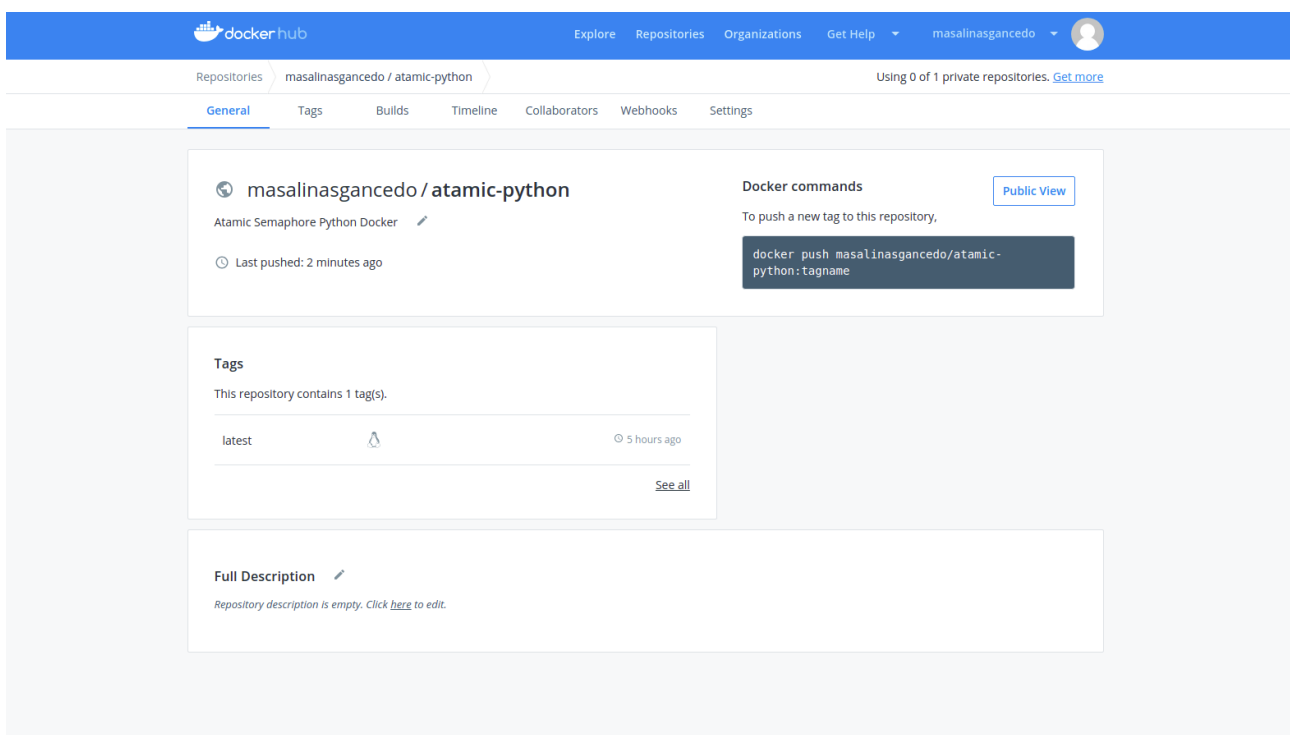
3. Testing&Notifications

In this step Semaphore execute the Python service **notebook_tester.py** using the **configuration.yml**, where define all folders where the service must test. Te input of this service are the Jupyter notebooks located in the folders configured in the previous configuration file and the result is a JSON Stream object send to an Slack Account indicating is exist error (Notebook, Cell, error payload) or not.

The Runtime Environment and Testing&Notifications are configured in a unique Semaphore Block to execute all the tasks sequentially. Actually the trigger that execute the pipeline is when any source from the github projetc attached to the pipeline change.

Semaphore Pipeline Docker Image

We create a new docker image from the Python default image from Semaphore called **semaphoreci/python:3.7.4**. I included miniconda to deploy Python dependencies and create environment for our tests. This image was published under a public account in the Docker Hub Cloud service to pull from Semaphore during pipeline runtime



The screenshot shows the Docker Hub interface for the repository 'masalinasgancedo/atamic-python'. The page includes a header with the Docker Hub logo and navigation links. The repository name is displayed prominently, along with the description 'Atamic Semaphore Python Docker'. A 'Last pushed' timestamp of '2 minutes ago' is shown. On the right, there are 'Docker commands' for pushing a new tag, including a code block with the command 'docker push masalinasgancedo/atamic-python:tagname'. Below this, a 'Tags' section lists the 'latest' tag, pushed '5 hours ago'. At the bottom, there is a 'Full Description' section with a note that the repository description is empty and a link to edit it.

Semaphore Pipeline Error Object

The errors tested in the pipeline are log in real time in the pipeline from Semaphore and attached like a strime to a message notify to a Slack account. From this account the owner od the channel could download the errors.

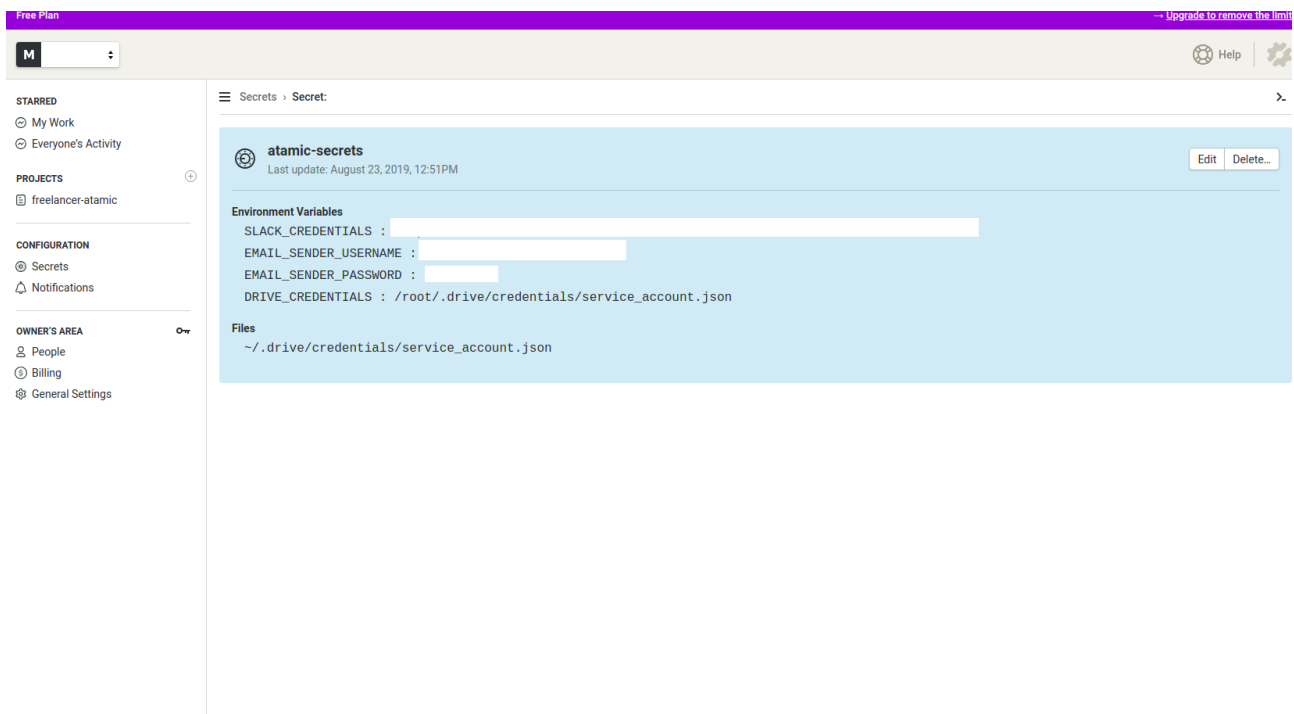
An example could be

```
[
  {
    "cell": 1,
    "notebook": "/home/miguel/temp/my-notebooks/notebook_with_errors.ipynb",
    "trace": {
      "output_type": "error",
      "ename": "ZeroDivisionError",
      "evalue": "division by zero",
      "traceback": [
        "\u001b[0;31m-----\u001b[0m",
        "\u001b[0;31mZeroDivisionError\u001b[0m          Traceback (most recent call
last)",
        "\u001b[0;32m<ipython-input-1-8c5de0cfca6c>\u001b[0m in
\u001b[0;36m<module>\u001b[0m\n\u001b[0m\n\u001b[1;32m    2\u001b[0m
\u001b[0m\n\u001b[0m\n\u001b[0m\n\u001b[0;34m=\u001b[0m
\u001b[0;36m0\u001b[0m\n\u001b[0;34m\u001b[0m\n\u001b[0m\n\u001b[1;32m    3\u001b[0m
\u001b[0;34m\u001b[0m\n\u001b[0m\n\u001b[0;32m----> 4\u001b[0m \u001b[0m\n\u001b[0m\n\u001b[0;34m/\u001b[0m
\u001b[0m\n\u001b[0m\n\u001b[0;34m\u001b[0m\n\u001b[0m\n\u001b[0m\n",
        "\u001b[0;31mZeroDivisionError\u001b[0m: division by zero"
      ]
    }
  }
]
```

The format of these errors are json easy to parse from other tools like the link

<https://jsoneditoronline.org/>

- **EMAIL_SENDER_PASSWORD:** this secret configure the GMAIL user password to notify events. Example
 - **EMAIL_SENDER_PASSWORD** = <GMAIL_PASSWORD_ACCOUNT>
- **DRIVE_CREDENTIALS:** this secret configure the path where the Python service recover the Goodle Drive credentials. This secret has other **File secret** asociated where we must attached the json credentials of our google Drive user. It's important asociate the correct path to this last file credentiales because this path is configured in **DRIVE_CREDENTIALS** and used by the service. Example:
 - **DRIVE_CREDENTIALS** = /root/.drive/credentials/service_account.json
 - **File secret** = ~/.drive/credentials/service_account.json



Variables Python Service

The Python service test all the Jupyter Notebooks. This service has several parameters o be configured under **configuration.yml** file.

- **trace:** is a boolean parameter to track all the secrets and atributes under Semaphore in real time. Values: 0 or 1. Default: 1
- **folder:** is an array of folders where the service will test. Default: './notebooks'
- **email.host:** is the SMTP host used in the email notification
- **email.port:** is the SMTP port used in the email notification
- **email.sender:** is the SMTP sender used in the email notification
- **email.receipient:** is the SMTP recipient used in the email notification
- **drive.email:** si the Goofle Drive account used to shared the files upload to service account folder

- **notification.email:** flag to active email notifications
- **notification.email:** flag to active drive notifications

Semaphore Pipeline Notification

The pipeline could notify events attaching the zip file using Gmail or Google Drive. It's configurable using an attribute called **notification.email** and **notification.drive**

The pipeline notify errors tested to a Channel of a Slack account previously created. The slack notification has this structure:

If **exist error** in the pipelin executed:

- **Title:** Testing Atomic at <DATETIME> with errors. Example

Testing Atamic at 2019-08-21 14:29:55 with errors

- **Description:** Testing Atamic at <DATETIME>

Testing Atamic at 2019-08-21 14:29:55

- **File Attached:** a example could be:

```
[
  {
    "cell": 1,
    "notebook": "/home/miguel/temp/my-notebooks/notebook_with_errors.ipynb",
    "trace": {
      "output_type": "error",
      "ename": "ZeroDivisionError",
      "evalue": "division by zero",
      "traceback": [
        "\u001b[0;31m-----\u001b[0m",
        "\u001b[0;31mZeroDivisionError\u001b[0m          Traceback (most",
        "recent call last)",
        "\u001b[0;32m<ipython-input-1-8c5de0cfca6c>\u001b[0m in",
        "\u001b[0;36m<module>\u001b[0;34m\u001b[0m\n\u001b[1;32m    2\u001b[0m \u001b[0m",
        "\u001b[0mb\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m",
        "\u001b[0;36m0\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m",
        "\u001b[0;36m3\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;32m",
        "\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;32m----> 4\u001b[0m \u001b[0;31m",
        "\u001b[0ma\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m",
        "\u001b[0mb\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m \u001b[0;34m\u001b[0m",
        "\u001b[0;31mZeroDivisionError\u001b[0m: division by zero"
      ]
    }
  }
]
```

]

If **Not exist error** in the pipelin executed:

- **Title:** Testing Atamic at <DATETIME> with any errors. Example

Testing Atamic at 2019-08-21 14:29:55 with any errors

- **Description:** Testing Atamic at <DATETIME>

Testing Atamic at 2019-08-21 14:29:55

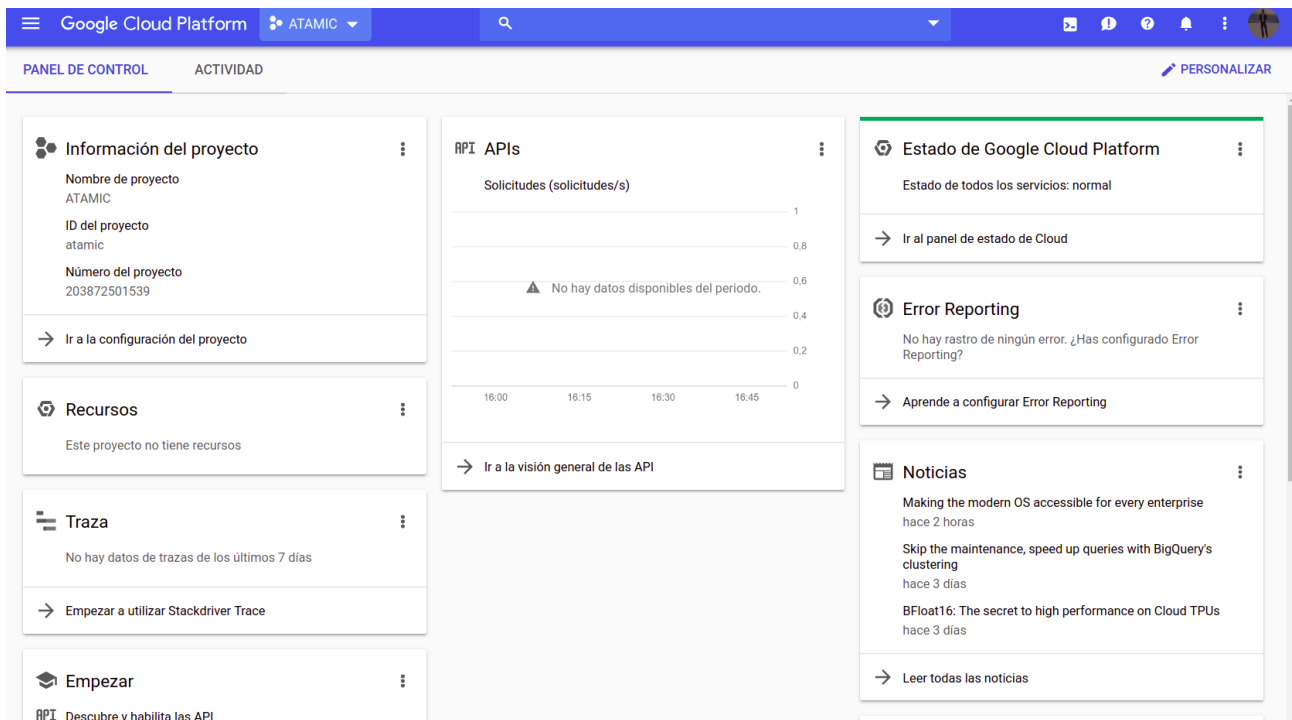
- **File Attached:** a example could be: []

An example could be:

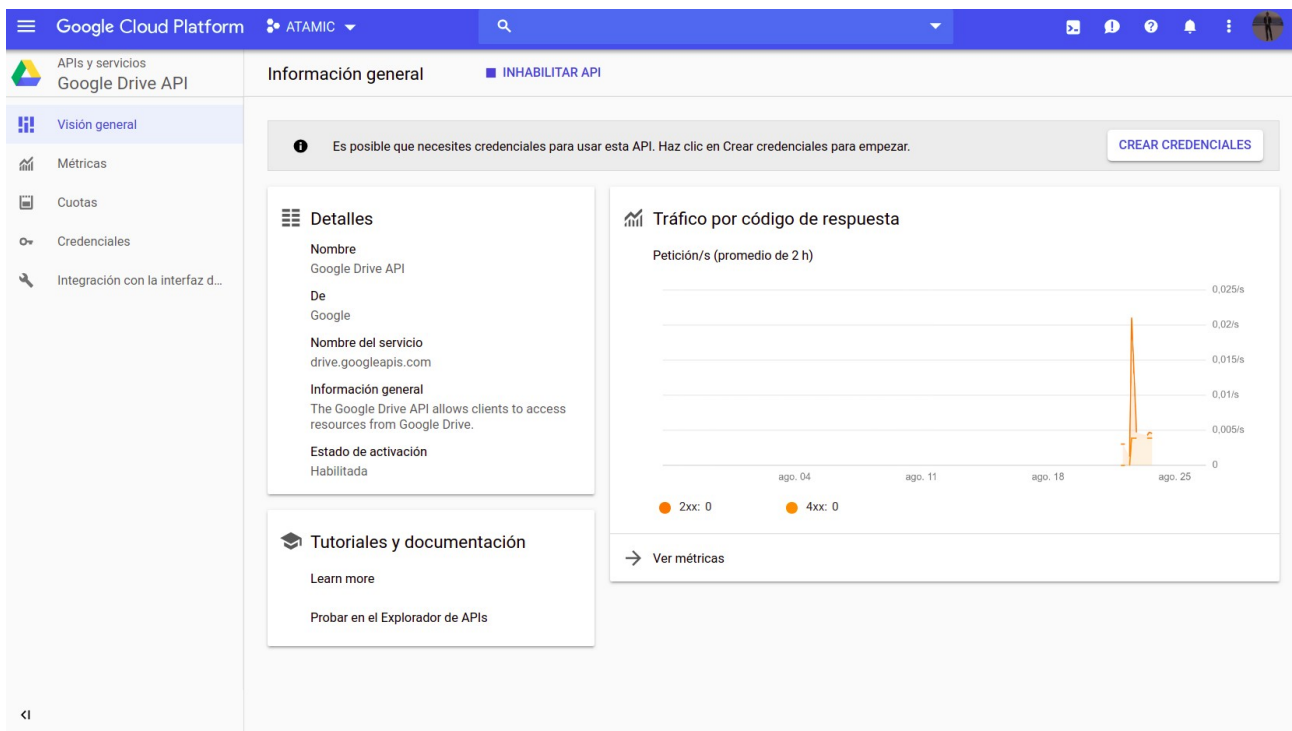
The screenshot shows a Slack interface with a channel named '# test'. A message from Miguel Salinas Gancedo, dated 2:29 PM, is displayed. The message content is 'Testing Atamic at 2019-08-21 14:29:55 with errors'. Below the text is a code block containing a JSON object representing a notebook cell with an error. The JSON object has fields for 'cell', 'notebook', 'trace', 'output_type', 'ename', 'evalue', and 'traceback'. The 'traceback' field contains a detailed Python error message: 'ZeroDivisionError: division by zero'. The error message is wrapped in a multi-line string with escape characters. The Slack interface also shows a sidebar with channel and direct message lists, and a bottom bar with app installation options.

Configure Google Drive to access throw Python Client

To configure python access to Google Drive we must create a service account under the google account where we want to upload our files. To create the service account we must access to the account of Google Cloud.



Under this account we recoment create a new project in this case called ATAMIC under we must activate the Goggle Drive API



After this we could create our service account to access to Google Drive API

Service accounts for project "ATAMIC"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. [Learn more](#)

Email	Status	Name	Actions
atamic@atamic.iam.gserviceaccount.com	Active	atamic	[Actions]

Permissions

You can allow specific users to have ownership and access to service accounts and their settings. Users with primitive project owner and project editor roles can already modify service accounts, but you might want to restrict access for some users so that they can take only specific actions against service account resources. [Learn more](#)

To assign a project role to a service account, use the IAM page.

Please select at least one resource.

In this case I created the service account called **atamic**. After create the service account Google Cloud download the credentials **service_account.json** file that represent the private key associated to the public key previously created. This file must be used by Python Google API Client to connect to the Google Drive API and upload files in Google Drive.

atamic [DELETE]

atamic

Description
Atamic Service Account

Email
atamic@atamic.iam.gserviceaccount.com

Unique ID
109446144680525066306

Disable service account
Disabling your account allows you to preserve your policies without having to delete it.

Account currently active

Enabled

SHOW DOMAIN-WIDE DELEGATION

Keys

Key ID
daf97085b9e3810e986d8c570068ecb389023d24

+ CREATE KEY

SAVE CANCEL

One important topic to know is that when the service account upload any file to Google Drive only this actor has access to the files so we must shared this file created to the google account previously used to has access to the files directly under the web and not only under the API.