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 respository explanation file

Semaphore Pipeline Github Project

This project is published under name: **freelancer-atamic.** This project configure the pipeline used to **test several Jupyper 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 respository 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** includging 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

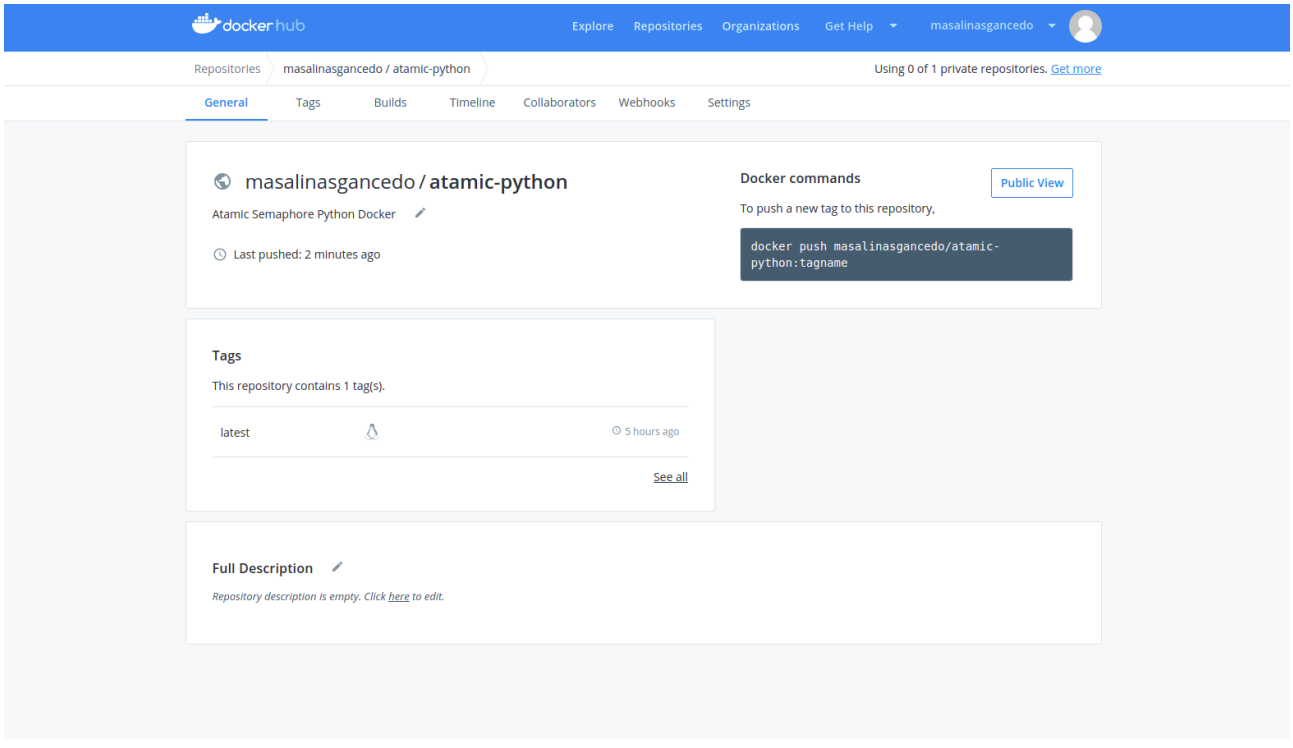
3. **Testing&Notifications**

   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



Semaphore Pipeline Error Object

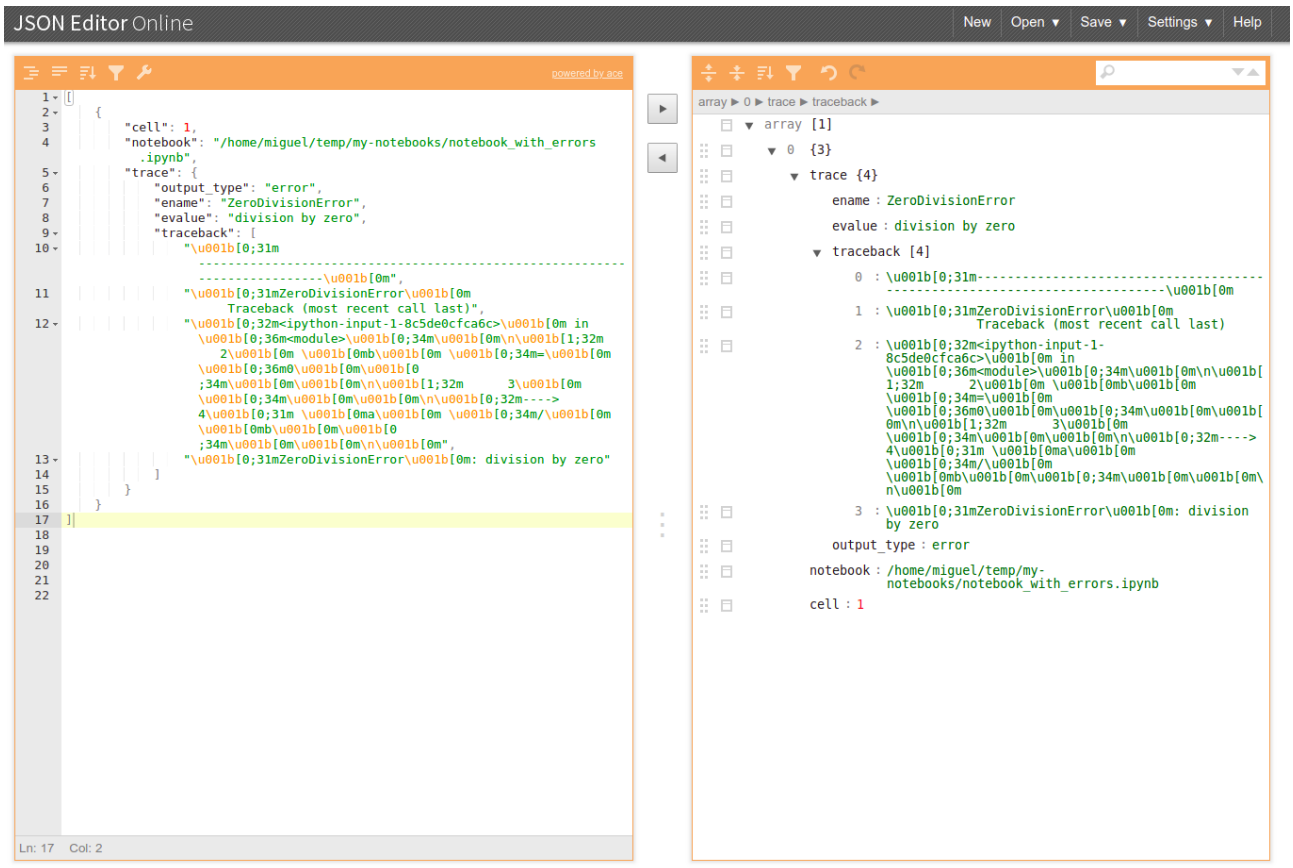
The errors tested in the pipeline are log in real time in the pipeline from Semaphore and attached like a strime to a message notify to a Slack account. From this account the owner od the channel could download the errors.

An example coulld 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[0mb\u001b[0m \u001b[0;34m=\u001b[0m \u001b[0m\u001b[0;36m0\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m      3\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m----> 4\u001b[0;31m \u001b[0ma\u001b[0m \u001b[0;34m/\u001b[0m \u001b[0mb\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m",
            "\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/



The format of the JSON is a collection o JSON objects. The structure of each object is:

- **ename**: Error type name from by python
- **evalue**: Error Type description from by python
- **traceback**: array with all stack trace from by python from by python
- **output**_type: ‘error’ . In future version could me add other tested messages like warn, etc
- **notebook**: is the notebook when the error was triggered
- **cell**: cell inside notebook where the error was triggered

Semaphore Pipeline Secrets

We must configure theses **secrets** before deploy our pipeline under the **called atamic-secrets**. This name will be used where configured the agent sercrets of the pipeline.

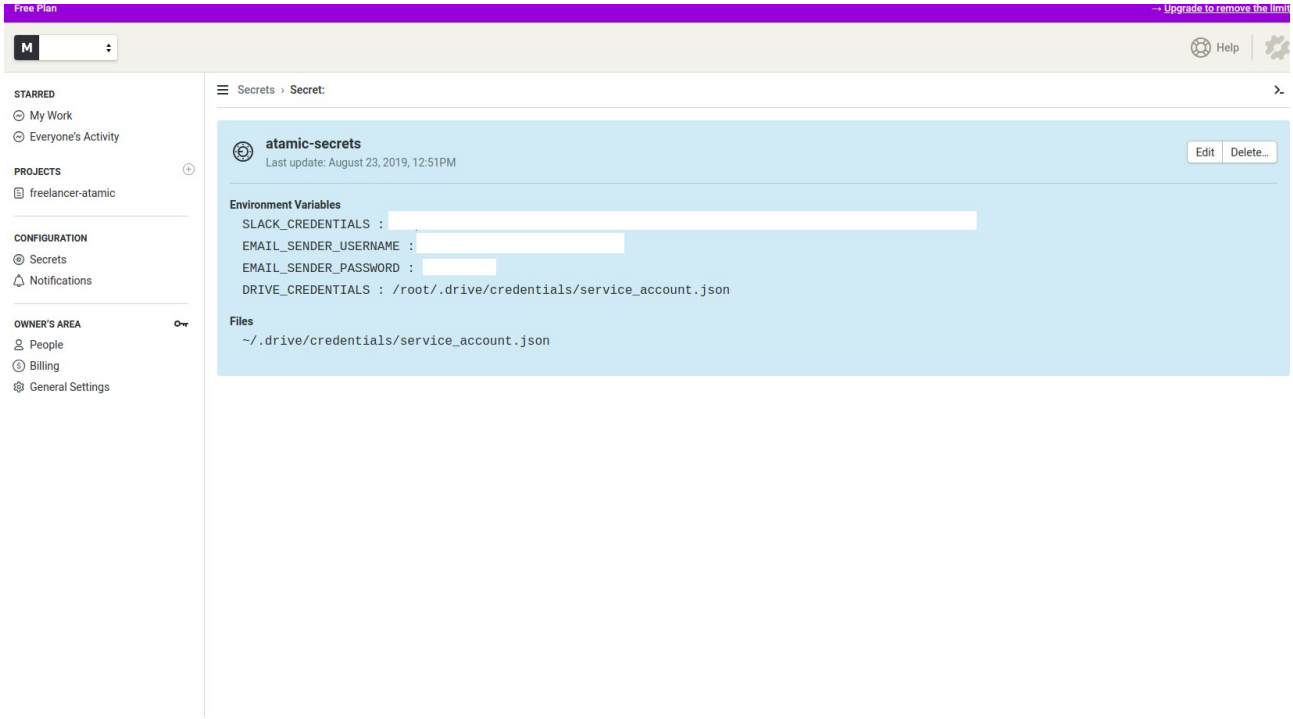
- **SLACK_CREDENTIALS**: this secret configure the Token used to access Slack service and notify events. Example:
  - **SLACK_CREDENTIALS** = <SLACK_TOKEN_ID>
- **EMAIL_SENDER_USERNAME**: this secret configure the GMAIL user account to notify events
  - **EMAIL_SENDER_ USERNAME** = <GMAIL_USERNAME_ACCOUNT>

- **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 f**ile.

- **trace**: is a boolean parameter to track all the secrets and atributes under Semaphore in real time. Values: 0 or 1. Defaul: 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 atribute 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:

- **Tittle**: Testing Atamic 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[0mb\u001b[0m \u001b[0;34m=\u001b[0m
\u001b[0;36m0\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[1;32m
3\u001b[0m \u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0;32m----> 4\u001b[0;31m
\u001b[0ma\u001b[0m \u001b[0;34m/\u001b[0m
\u001b[0mb\u001b[0m\u001b[0;34m\u001b[0m\u001b[0m\n\u001b[0m",
            "\u001b[0;31mZeroDivisionError\u001b[0m: division by zero"
         ]
      }
   }
]
```

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

- **Tittle**: 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: