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.

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 block 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

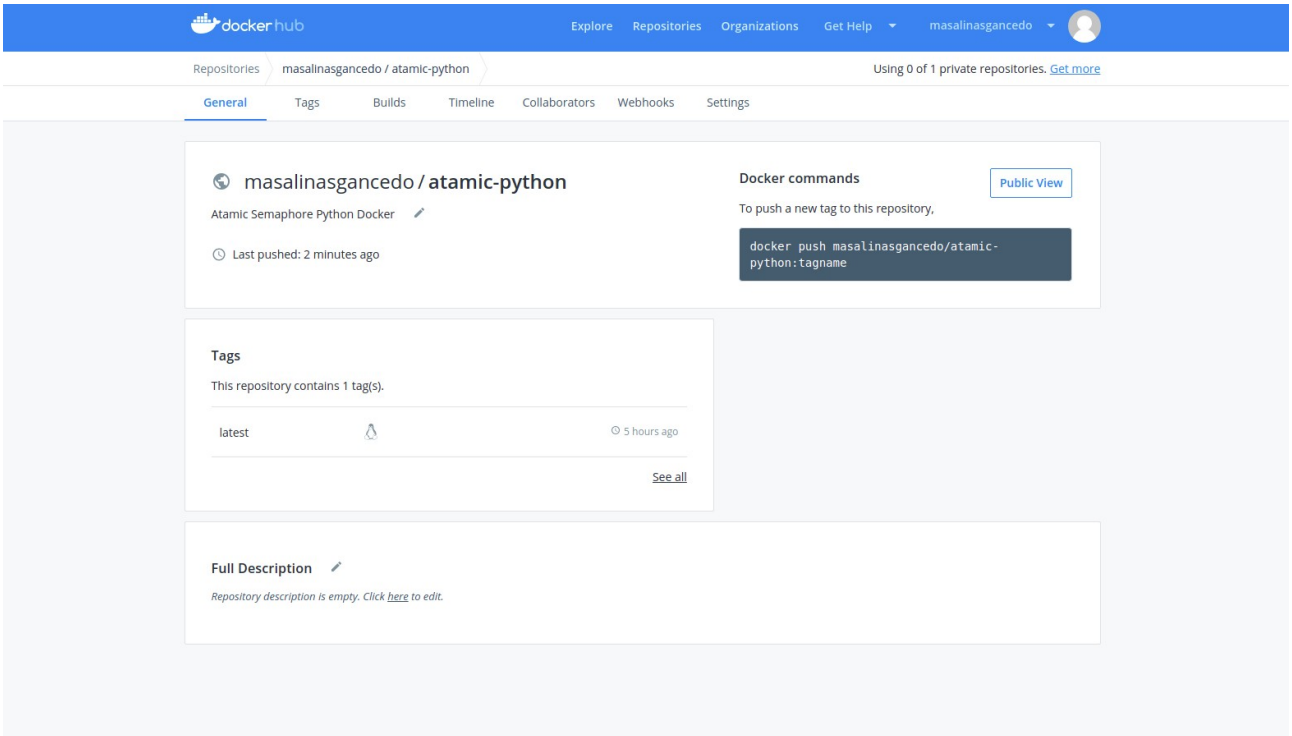
3. **Testing&Notifications**

   In this step Semaphore execute other block where 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.

Actually the trigger that execute the pipeline is when any source from the github projetc attached to the pipeline change.

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

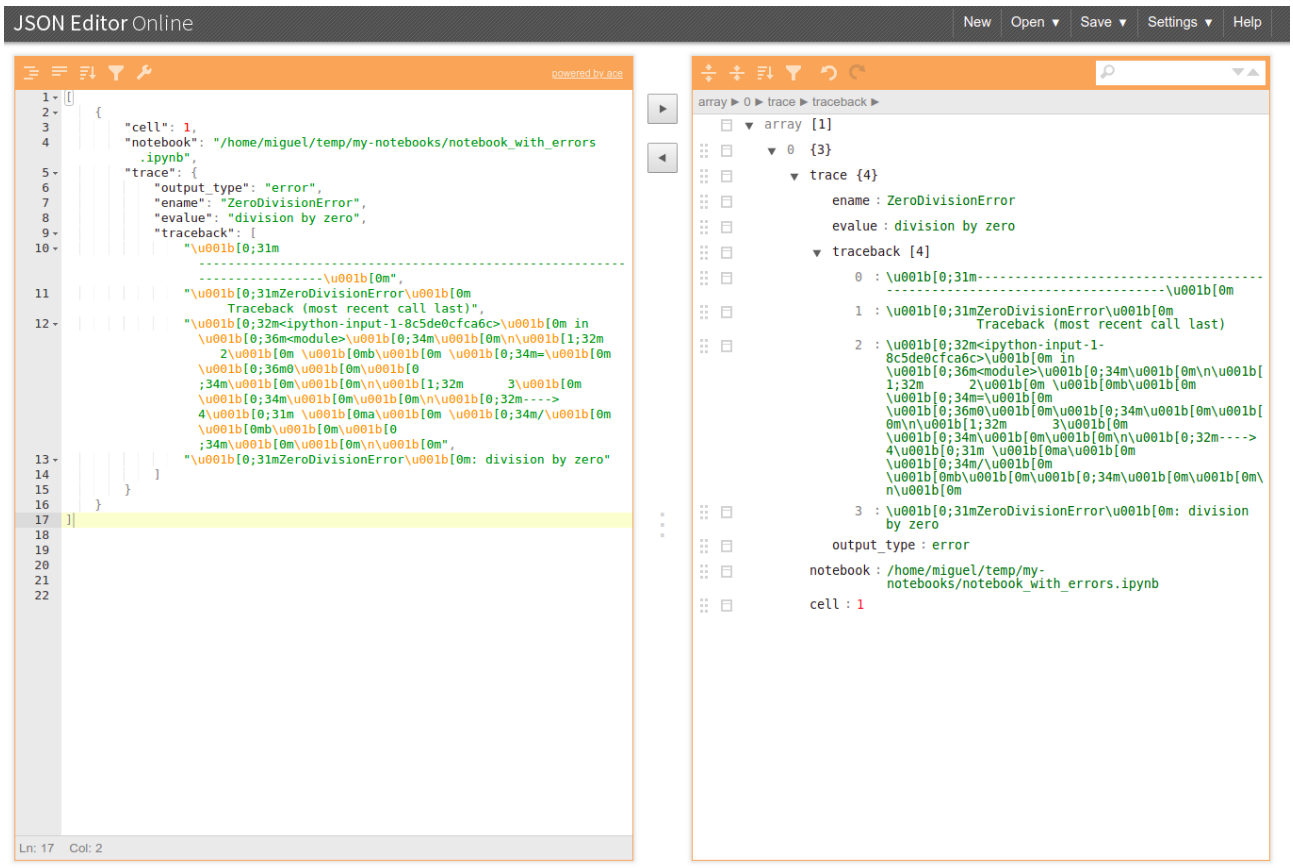
Pipeline Error Payload

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

https://jsoneditoronline.org/

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[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[0;31mZeroDivisionError\u001b[0m: division by zero"
         ]
      }
   }
]
```



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

Pipeline Secrets

Before deploy our pipeline we must create the secrets for the Slack API TOKEN to notifie any message generated for the Python Tester Service. This action we could made from Semaphore UI or from shell using the Semaphore CLI. If use the CLI we must execute this command:

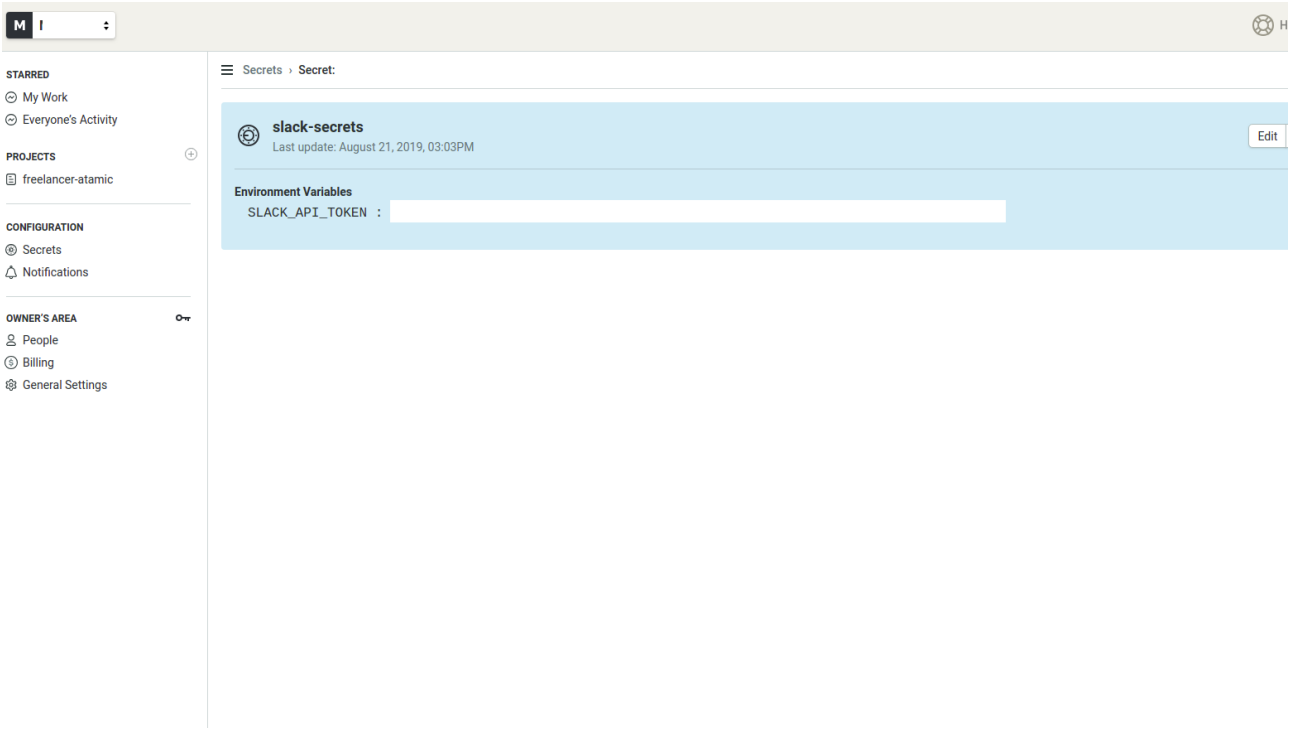
sem create secret <SECRET_NAME> -e <TOKEN_ID_1>=<TOKEN_VALUE_2> ... -e <TOKEN_ID_N>=<TOKEN_VALUE_N>

Where:

- SECRET_NAME: name of the secret. Example: slack-secrets

- TOKEN_ID_N: N name of the secret token included in previous SECRET_NAME.
  Example SLACK_API_TOKEN
- TOKEN_VALUE_N: N value of the secret token included in previous SECRET_NAME.
Example AAABBBCCC

From Semaphore UI



The name of the secrets and the name of the tokens inside must be used later in the semaphore pipeline to pass it like an argument to the Python Tester service to notify any messages to Slack Account associated to the token

Pipeline Notification

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

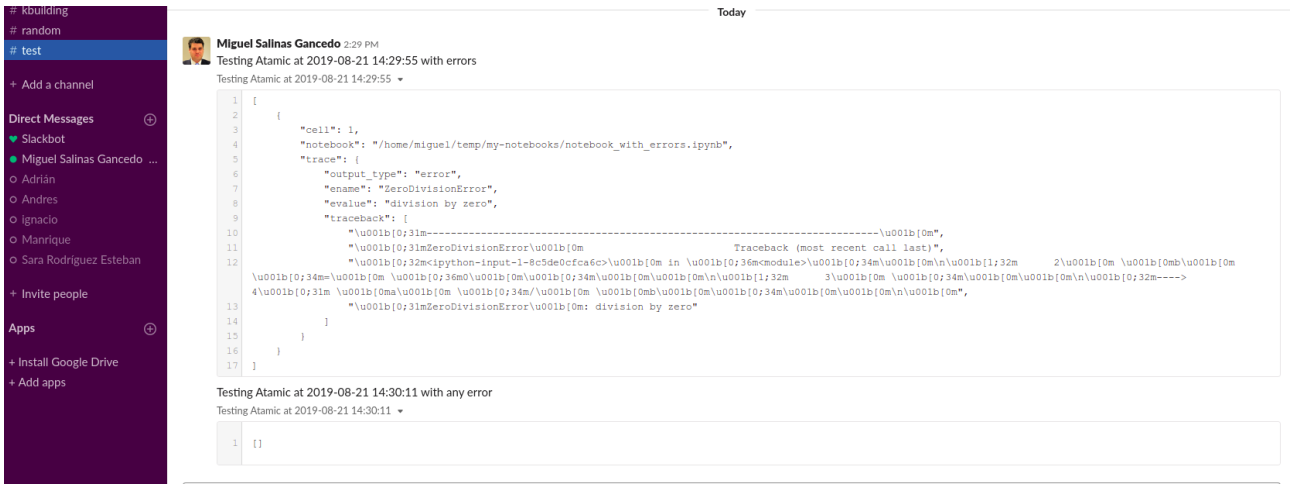
  Testing Atamic at 2019-08-21 14:29:55 with any errors

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

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

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

An example could be:


kbuilding
random
test
Add a channel
Direct Messages
Slackbot
Miguel Salinas Gancedo ...
Adrián
Andres
ignacio
Manrique
Sara Rodríguez Esteban
Invite people
Apps
Install Google Drive
Add apps
Today
Miguel Salinas Gancedo 2:29 PM
Testing Atamic at 2019-08-21 14:29:55 with errors
Testing Atamic at 2019-08-21 14:29:55
[
{
"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---------- 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"
]
}
]
Testing Atamic at 2019-08-21 14:30:11 with any error
Testing Atamic at 2019-08-21 14:30:11
[]