

Final Report on: Pretix Redundant Operation

Filipe Pires [85122], João Alegria [85048]

Computational Infrastructures Management
Department of Electronics, Telecommunications and Informatics
University of Aveiro

June 11, 2020

Introduction

This report aims to describe the work developed for the final phase of the practical assignment of the discipline of Computational Infrastructures Management (*I*) from the Msc. degree in Informatics Engineering of the University of Aveiro at the Department of Electronics, Telecommunications and Informatics. It is assumed that the reader has knowledge about the previous report, and it is here included: the characterization of the service level agreements for each service; the description of the load balancing mechanisms amongst software components; the redundancy strategy; the listing of metrics and computational resources monitored, along with the respective defined alarms; and other relevant aspects such as horizontal scalability and component fault tolerance.

The service provided is Pretix, an online shop, box office and ticket outlet already successfully used by other service providers for conferences, festivals, exhibitions, workshops and more. The previously delivered work focused on the product presentation, distributed installation, and resource and performance analysis. All code developed is publicly accessible in our GitHub repository:

<https://github.com/FilipePires98/GIC>.

1 System Architecture

1.1 Infrastructure

...

1.2 Load Balancing

...

1.3 Redundancy

...

1.4 Scalability

...

2 Infrastructure Management

2.1 Service Level Agreements

...

2.2 Monitoring

2.2.1 Log Collection

2.2.2 Metric Collection

2.3 Alerts and Automation

...

3 Additional Remarks

3.1 Pretix Quickstart

During development, a need for a faster basic setup of Pretix emerged for debugging purposes. As many of our tests assumed the existence of organizers and events already inserted on the databases, each time these had to be cleared the objects had to be manually created before proceeding with some implementation process.

In order to respond to this need, we developed a Python script capable of completely dispose of the manual labour. This script, which we called `quickstart.py` and placed under a dedicated folder, uses Selenium (16) to create an organization inside Pretix, generate a secret API key for sending REST requests (storing it in a text file), create a default event and prepare it for load testing with Locust. It also contains configurable variables to personalize its output.

Such quickstart tool was not only found useful for the obvious advantage in development speed it brought to us, but was also considered a good help tool for any person that wishes to quickly experiment our infrastructure without having to know much about Pretix itself.

3.2 Documentation

A great portion of the code we deal with isn't of our authorship, so the documentation it contains is the documentation we get. Nevertheless, the code we developed has specific purposes and, although usually very much self explanatory due to its nature, should be easy to interpret by any person. Our goal is to ensure that what we deliver can be easily reused or even continued by other developers.

With this in mind, we made significant efforts on ensuring that all code developed by us followed a common structure with the same coding style. Also, we made sure all source files have a description of their purpose and relevant information on the first lines, and contain comments on key points of the code explaining snippets considered of greater complexity/importance.

3.3 Assignment Contributions

Regarding the work distribution amongst developers, a close-contact strategy was defined where each worked on a cluster component or piece of software according to a predefined plan. The cluster strategy and respective details were decided in conjunction, as well as the key objectives and tasks to be achieved before the final delivery deadline.

Pretix' features were explored by both developers in order to better comprehend the web platform and the REST API. Then, João focused on the deployment of our solution on the department's infrastructure while Filipe focused on creating and executing the load tests. The

analysis of the results were carefully conducted in conjunction as well, in order to maintain a common knowledge for better workflows. It is needless to say that bug and error solving was made along the development phase by both developers any time it was required. Once performance benchmarking and bottleneck identification were completed, this report and the code documentation became our primary concern, with both contributing equally.

Conclusions

...

References

1. J. P. Barraca, *GIC - Report no.2: Simple Product Operation*, University of Aveiro, 2019/20.
2. *About Pretix*, <https://pretix.eu/about/en/>. Pretix.eu, retrieved in April 2020.
3. *Welcome to Rami.io*, <https://rami.io/>. Rami.io, retrieved in April 2020.
4. *Pretix Code Repository*, <https://github.com/pretix/pretix>. GitHub, Inc., retrieved in April 2020.
5. *Welcome to pretix' documentation!*, <https://docs.pretix.eu/en/latest/>. Pretix.eu, retrieved in April 2020.
6. *Docker Homepage*, <https://www.docker.com/>. Docker Inc., retrieved in April 2020.
7. *pretix/standalone*, <https://hub.docker.com/r/pretix/standalone>. pretix, retrieved in April 2020.
8. *Meet Django*, <https://www.djangoproject.com/>. Django Software Foundation and individual contributors, retrieved in April 2020.
9. *Gunicorn Homepage*, <https://gunicorn.org/>. Gunicorn.org, retrieved in April 2020.
10. *NGinX News*, <https://nginx.org/>. NGinX.org, retrieved in April 2020.
11. *PostgreSQL: The World's Most Advanced Open Source Relational Database*, <https://www.postgresql.org/>. The PostgreSQL Global Development Group, retrieved in April 2020.
12. *About Redis*, <https://redis.io/>. Redis Labs, retrieved in April 2020.
13. *Welcome to Pgpool-II*, https://www.pgpool.net/mediawiki/index.php/Main_Page. PgPool Global Development Group, retrieved in April 2020.
14. *HAProxy: The Reliable, High Performance TCP/HTTP Load Balancer*, <https://www.haproxy.org/>. HAProxy.org, retrieved in April 2020.
15. *Locust, an open source load testing tool*, <https://locust.io/>. Locust.io, retrieved in April 2020.
16. *About Selenium*, <https://www.selenium.dev/about/>. Software Freedom Conservancy, retrieved in April 2020.
17. *Ansible*, <https://www.ansible.com/>. Red Hat, retrieved in April 2020.