Easy App Deployment Environment

Fog and Cloud Computing 2021 - Group 19
Claudio Facchinetti <claudio.facchinetti@studenti.unitn.it>
Matteo Franzil <matteo.franzil@studenti.unitn.it>

June 1, 2021

1 Introduction

This document outlines the modifications applied to the final project from group 19 for the FCC-21 course. Each section mirrors the ones from the draft, highlighting the differences and the issues found during the deployment.

2 Sample Application

The sample application was successfully Dockerized and updated to the latest Java and Tomcat versions. However, due to time constraints and the fact that it wasn't the central part of the project, more advanced features such as PDF generation and QR code generation were not fully tested and are not guaranteed to be functional. Additionally, email interaction has not been implemented. All the other features described in the GitHub repo (https://github.com/mfranzil/centodiciotto) are functional. In order to test them, you can use the following credentials (they should be enough):

```
PATIENT:
ottavio.longhena@gmail.com
ottavio.longhena

HEALTH SERVICE:
ssp.treviso@gmail.com
servizio.sanitario.treviso
```

3 Platform as a Service

The PaaS side was implemented successfully. Everything is available through the `eval` user. The **web server** has 3 replicas (scalable), the **database** has a single pod and is not scalable (since `psql`) would need extra work for supporting distributed instances.

Being a test infrastructure, we decided to implement a single Ingress listening on port `80`, and the web server must be contacted using the `$MASTER_IP`. It is anyway available on port `30130`, and the database on port `30230`. These ports should never be exposed to an eventual web-facing IP.

We created an automated script that deploys everything (bar the eval cluster) on one go. This includes the restarting of the Docker daemon: we need to make the certificate of the Docker Registry available and then reboot it.

4 Infrastructure as a Service

In the end, we decided not to implement a DNS resolver at all. It would have unnecessarily complicated all addressing.

The local Docker Registry was deployed as a Debian Buster instance on Nova. Due to constraints on the resources on PAAS-19, we still chose to deploy the registry on IAAS-19 and we did it by:

- installing Docker on this machine
- pulling from DockerHub the official registry image
- mounting the image as a container with port 8081 exposed to the rest of the NVM

While this may seem redundant, this still decouples the Kubernetes deployments from the rest of the components, and provides more resources to every instance in the project.

The Resource Server was instead deployed using the official Swift API and is therefore available in the Object Storage section of the web portal. REST API requests are used for contacting it. We tried to use a Java library for addressing it, but it was too outdated so we settled on using old-fashioned HTTP requests, given the limited amount of requests made by the app itself.