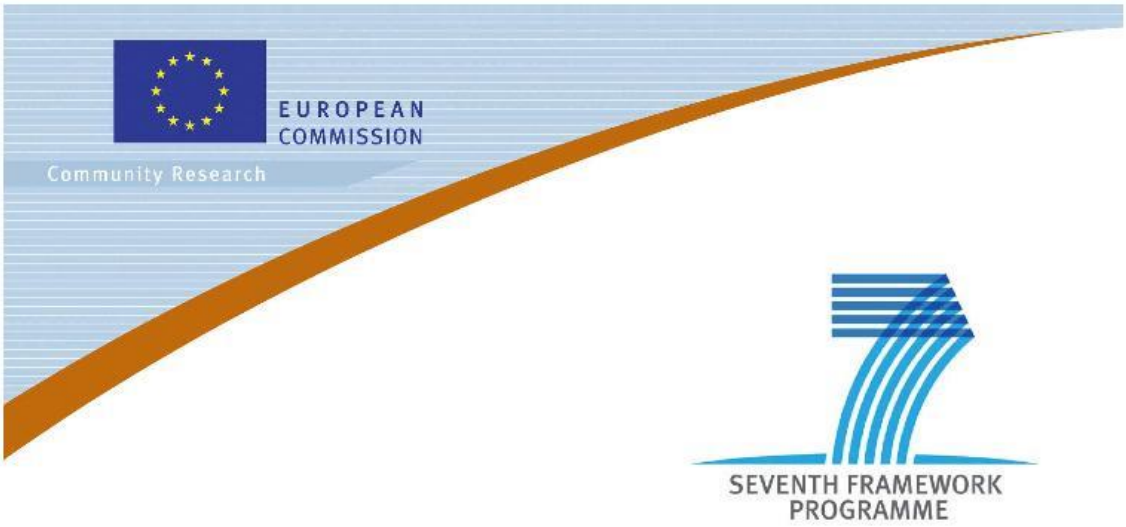

Private Public Partnership Project (PPP)

Large-scale Integrated Project (IP)

D.11.9.2 Report on FIWARE Dockerization activities

Project acronym: FI-Core

Project full title: Future Internet - Core

Contract No.: 632893

Strategic Objective: FI.ICT-2011.1.7 Technology foundation: Future Internet Core Platform

Project Document Number: ICT-2013-FI-632893-WP11-D.11.9.2

Project Document Date: 2016-12-31

Deliverable Type and Security: PU

Authors: José Manuel Cantera (TID), Alberto Martín (Bitergia)

Executive Summary

This report outlines the activities performed by FIWARE aimed at creating Docker containers. Docker containers wrap a piece of software in a complete filesystem that contains everything needed to run. This guarantees that the software will always run the same, regardless of its environment. The usage of Docker improves the overall FIWARE friendliness, allowing developers to instantiate FIWARE GEs, incarnated by lightweight Docker containers, following a very convenient, portable and flexible approach.

In a nutshell, the activities reported by this document have consisted of the elaboration and refinement of the “Docker Guidelines”, including technical support to all the teams involved, the management of the Docker Hub account for FIWARE and the development of *Dockerfiles* for the different GEri Docker images. The activities conducted are briefly described together with the results obtained. Finally, conclusions and future work are outlined.

Table of Contents

Executive Summary 2
Table of Contents 3
1 Introduction 4
1.1 Introduction 4
1.2 Target audience 4
1.3 Related readings 4
2 Activities and Results 5
2.1 Development of the Docker guidelines 5
2.2 FIWARE Docker Design Principles 7
2.3 Implementation of Docker Images for GEris 9
2.4 Management of the Docker hub account 11
3 Conclusions and future work 12
Appendix .- FIWARE Docker Guidelines 13

1 Introduction

1.1 Introduction

This report outlines the activities performed by FIWARE with regards to the creation and packaging of GEris as Docker images. The availability of Docker images improves FIWARE friendliness, allowing developers to instantiate FIWARE GEs, incarnated by lightweight Docker containers, following a very convenient, flexible and portable approach.

The activities reported by this document have consisted of:

- elaboration and refinement of the "Docker Guidelines", so that there is consistency and coherency across FIWARE.
- provision of technical support to the FIWARE GEri teams when developing their Docker images.
- Implementation and testing of recipes for the different GEri Docker images.
- management of a Docker Hub organization account as the main publication point of Docker artefacts for FIWARE.

For each of the points mentioned above, the technical approach followed is briefly described together with the results obtained. Finally, conclusions and future work are outlined.

1.2 Target audience

This document is intended to technical managers and FIWARE Community members who want to know more about the Docker activities performed by the FIWARE initiative.

1.3 Related readings

To know more about Docker, Docker Hub, etc. please visit https://www.docker.com/.

Related FIWARE deliverables are D.11.1.3 FIWARE Tour Guide.

2 Activities and Results

2.1 Development of the Docker guidelines

The work started by providing a full wiki page including the guidelines to be followed by each GEri team when creating Docker images: https://wiki.fiware.org/Docker. For the reader's convenience these guidelines have been included as an appendix at the end of this document.

The guidelines are structured around *must*, *should* and *may* actions, corresponding, respectively, to mandatory, recommended and optional practices to be put in place by GEri owners. In fact, they describe detailed information on what is needed for Docker images, including several requirements such as documentation (README files), version tagging, where to place Docker files, or even how images should be built.

Particularly all the GEris must include, at least, three different labels for their Docker images:

- *latest*: It corresponds to the latest stable release of a GEri.
- *develop*: It corresponds to the latest build (latest code snapshot) of the GEi. It might not be stable.
- *<release n>*: one tag per relevant and active stable release. The name of the tag should correspond to the name assigned to the release in Github.

An example of the implementation of these conventions can be found at https://hub.docker.com/r/fiware/orion/tags/

Once the documentation was delivered, a follow-up process started, helping and supporting teams to debug images and to follow strictly the guidelines provided. For instance, as a result of these activities all the GEris include Docker badges/links as part of their main README file. These badges inform about the number of downloads of the corresponding image, providing metrics about the popularity of each GEri.

Figure 1.- Example of Docker labels

(https://github.com/Fiware/context.Orion/)

In addition, all the GEris incorporate a 'Docker' folder containing all the Docker-related artefacts, containing a *README* file which describes how to use the corresponding Docker image. The figure below shows how this was implemented for Orion Context Broker, one of the most popular GEris.

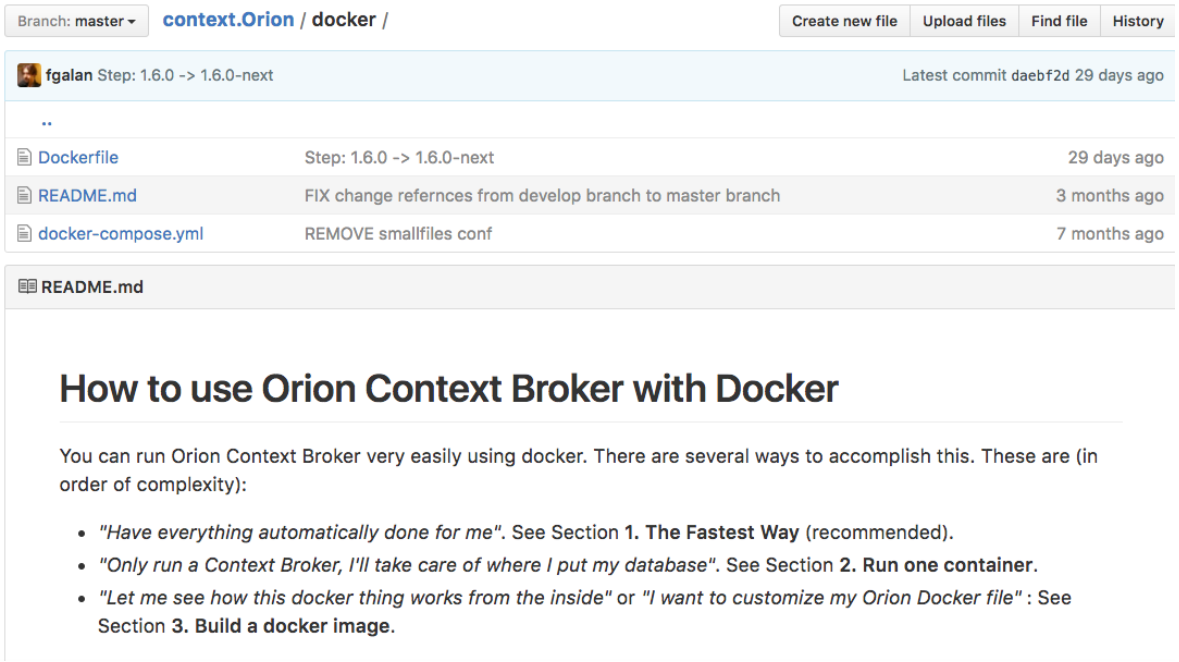

Figure 2.- Example of a Docker README file
(https://github.com/Fiware/context.Orion/tree/master/docker)

2.2 FIWARE Docker Design Principles

Docker images are generated by means of Dockerfiles, which contain a set of instructions intended to generate and/or install all the required dependencies for a software to run. In addition, a *Dockerfile* contains all the instructions that allow to run the involved service and to expose interface ports to the outside world. As a result, with a few simple lines of code a Docker image can be easily generated.

Generally speaking, there are two different strategies when creating Dockerfiles:

- Install the target software from official repositories (*apt-get install, yum install, npm install*, *pip install*, etc.), ensuring that the target image contains all the runtime dependencies.
- Build from sources, which means that all the source code is downloaded from Github (from a master branch or a release branch) and then it is compiled, built and configured. If the build process depends on the presence of a certain toolchain, then such toolchain will have to be installed and managed by the corresponding *Dockerfile*.

The "installation strategy" has the advantage of being very simple but not very flexible in terms of version management or hacking possibilities. An example of a Dockerfile which follows this strategy is the one corresponding to Wirecloud.

The "build from sources" strategy is very flexible as it allows developers to use images which contain the latest version of the software. However, it is much more difficult to maintain and it requires a proper management of the image. For instance, it is normally a good practice to remove all the software packages (toolchain) used for building a GEri. As a result, the size of the resulting image will decrease, making it easier to be managed. An example of a Dockerfile which follows the "build from sources" strategy is the one corresponding to Orion Context Broker.

On the other hand, a design principle applied by FIWARE, with regards to Docker, is that all the images have to be modularized properly (lightweight images). This means, for instance, that if a GEri uses a database server for the persistence layer, such database server should not be included in the GEri image. In other words, an image should include all their internal dependencies (libraries, web servers, etc.) to run, but not any external dependency (i.e. any software artefact which can be incarnated as a separated server process).

For instance, the Orion Context Broker image does not include MongoDB (the NoSQL database use by Orion to store context data). MongoDB is available through a different Docker image. This means that users of the FIWARE Orion image have to link at instantiation time the Orion container with a Mongo DB container. This process can be done manually but, fortunately, the Docker toolchain includes a utility called docker-compose which allows to orchestrate the execution of multiple Docker containers that have dependencies between them. The configuration of the orchestration to be performed by *docker-compose* is done through a docker-compose.yml file, which declares which are the Docker images involved, how they should be run and how they are linked together.

To make FIWARE developers lives easier, each GEri must not only provide a Dockerfile, but also a *docker-compose* file which allows to instantiate the GEri container together with any of their dependencies. As a result, developers can obtain an up and running instance of the concerned GEri together with all the supporting containers needed to operate.

Below there is a listing of the *docker-compose.yml* file provided by Orion Context Broker. It can be observed that the *.yml* file allows to declare what are the images involved, and that there is a link between the Orion and the MongoDB containers. Finally, the command line instruction for running Orion indicates what is the node (container) which hosts the MongoDB database needed.

```
4   mongo:
5     image: mongo:3.2
6     command: --nojournal
7
8   orion:
9     image: fiware/orion
10    links:
11      - mongo
12    ports:
13      - "1026:1026"
14    command: -dbhost mongo
```

Figure 3.- Orion Context Broker docker-compose.yml
https://github.com/Fiware/context.Orion/blob/master/docker/docker-compose.yml

2.3 Implementation of Docker Images for GEris

The FIWARE "dockerization" activities started with the creation of Docker images for the GEris used by the FIWARE Tour Guide Application. Initially, the images generated followed the "installation strategy". Once this milestone was achieved, a second iteration with the image was done to improve its performance (i.e., removed unused packages needed for the installation, directories clean-up and image size optimization, among others). A last testing step was done in several GEs to ensure that all of them follow all the steps described in the Docker Guidelines previously described. Afterwards, README files were provided with instructions on how to use the images or how to build them from scratch. Finally, those images were offered to the GEris developers who wanted to use them, with a successful result for many of the GEris. After that, the maintenance of the images was left to each GEri team.

As a result of this work, a full list of GEri images is available under the FIWARE organization in Docker Hub (see below). This means, every GE available in FIWARE can now be used or tested in an isolated environment, just by typing few commands and making life easier to everyone in the community who wants to use or to contribute to FIWARE. Besides, it helps to use more than a GEri at a time, without any networking configurations by using the *docker-compose* tool. An example of this approach is the "FIWARE Tour Guide Application" that involves several GEs being orchestrated for the same application.

Below there is a table that summarizes the most salient FIWARE Dockerfiles produced at the time of writing this report:

GEri Name	Dockerfile / Docker image
Orion	https://github.com/Fiware/context.Orion/blob/master/docker/Dockerfile
CKAN	https://github.com/Fiware/context.Ckan/tree/master/contrib/docker
Kurento	https://hub.docker.com/r/fiware/stream-oriented-kurento/
Proton	https://github.com/Fiware/context.Proton/blob/master/docker/Dockerfile
Cygnus	https://hub.docker.com/r/fiware/cygnus/
IDAS	https://github.com/Fiware/iot.IoTagent-UL/blob/master/Dockerfile
IoT Discovery	https://github.com/Fiware/iot.Discovery/tree/master/docker
IoT Broker	https://hub.docker.com/r/fiware/iotbroker/
Cepheus	https://github.com/Fiware/iot.Cepheus/blob/master/docker/Dockerfile
SpagoBI	https://github.com/Fiware/apps.SpagoBI/tree/master/docker
Wirecloud	https://hub.docker.com/r/fiware/wirecloud/
Business API Ecoystem	https://hub.docker.com/r/fiware/business-api-ecosystem/
Keyrock	https://hub.docker.com/r/fiware/idm/
Wilma	https://hub.docker.com/r/fiware/pep-proxy/
AuthzForce	https://hub.docker.com/r/fiware/authzforce-ce-server/
GIS Data Provider	https://hub.docker.com/r/fiware/gisdataprovider/tags/
Policy Manager	https://hub.docker.com/r/fiware/bosun-cloto/ https://hub.docker.com/r/fiware/bosun-facts/
Ofnic	https://hub.docker.com/r/fiware/ofnic/
FIWARE Tour Guide Application	https://github.com/Fiware/tutorials.TourGuide-App/tree/develop/docker/images

2.4 Management of the Docker hub account

One of the requirements stated by the "Docker Guidelines" is to make all the GEri images available in Docker Hub. Docker Hub is a cloud-based registry service which allows to link to code repositories, build and test images, etc. In fact, under the organization created on Dockerhub for FIWARE, is where all the GEri developers have made their Docker images available for the community.

TID and Bitergia are in charge of managing the Docker Hub account for FIWARE, ensuring that all the GEri owners publish their images properly. The figure below shows a screenshot of the Docker hub account home page, including details about some of the images offered by the different FIWARE GEris.

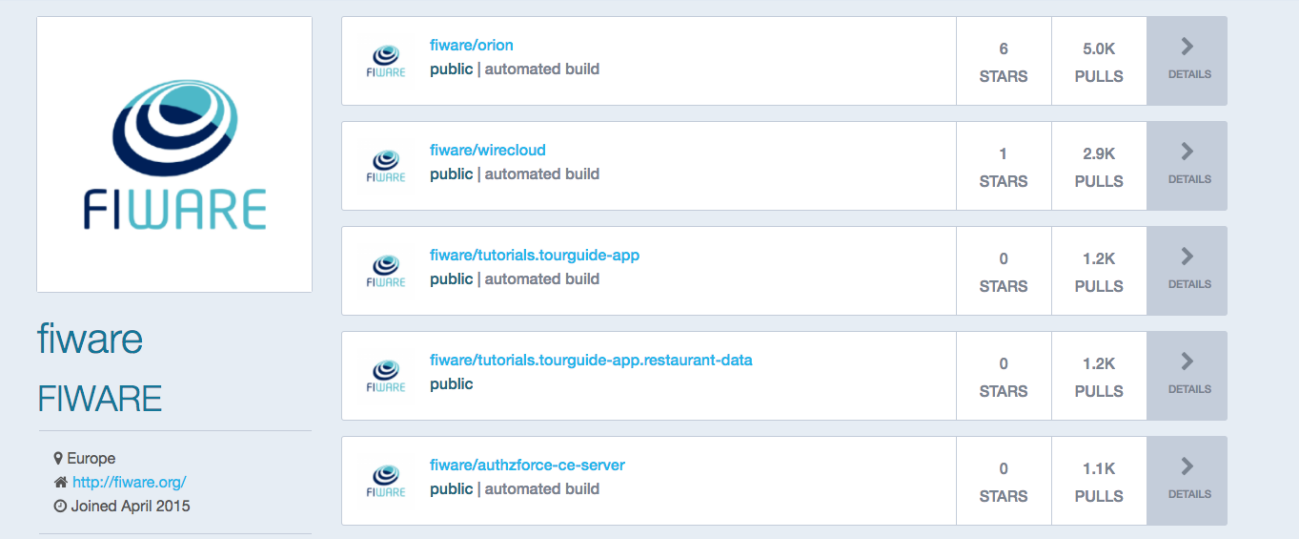

Figure 4.- Docker hub account for FIWARE (https://hub.docker.com/u/fiware/)

3 Conclusions and future work

FIWARE has made a considerable step forward by adopting the Docker technology stack. Nowadays FIWARE developers can easily instantiate a fully-fledged, portable FIWARE environment in a short period of time, regardless operating systems or programming environments.

The work coordinated by TID and Bitergia has been paramount to achieve the Docker milestone for FIWARE. The definition of clear guidelines for GEris has enabled the coordination of the different parties involved, which, in the end, has led to a coherent and consistent exercise. The strategy for creating lightweight Docker images has proved to be the right one, as it encourages modularity, and the separation of concerns, two of the fundamental principles of FIWARE. This work has been materialized, for instance, by the FIWARE Tour Guide Application tutorial, which makes extensive usage of Docker images provided by the different FIWARE GEris.

In the future, it will be convenient to have a second iteration for the different Docker images generated, ensuring that further and finer alignment is achieved. For instance, there is no complete alignment in the configuration mechanisms used by the different GEris. It would be nice to have common configuration patterns, for instance, through the usage of environment variables. This in the end will enable an easier instantiation of Docker containers on clusters hosted on the FIWARE Lab Cloud, by exploiting the capabilities of emerging tools, namely docker swarm (cluster technology for Docker).

Appendix .- FIWARE Docker Guidelines

This page summarizes Guidelines for providing Docker Containers for FIWARE GEis published on http://wiki.fiware.org/Docker.

There are three kinds of guidelines included:

- MUST Guidelines. They are mandatory and your GEi project must conform to that.
- SHOULD Guidelines. They are not mandatory but highly recommended.
- MAY Guidelines. They are nice to have.

G1.- At least one Dockerfile (hereby named as 'reference Dockerfile'), intended to GEi users, MUST be provided. The base image (Ubuntu, CentOS, etc.) for such a Dockerfile might depend on each GE.

G2.- Hacker-oriented Dockerfiles (intended to GEi developers) MAY be provided as well.

Each Docker container MUST define the following tags (present at Dockerhub):

- 'latest': It will correspond to the latest stable release of the GEi.
- 'develop': It will correspond to the latest build (latest code snapshot) of the GEi. It might not be stable.
- '<release n>': one tag per relevant and active stable release. The name of the tag will correspond to the name assigned to the release in Github.

G3.- The reference Dockerfile MUST be present under the 'docker' folder of the GEi repository

Should your GEi depend on other components (Databases, etc.) you MUST provide a docker-compose.yml file that will allow to instantiate the GEi together with its dependencies.

G4.- A README.md MUST be provided under the docker folder. Such a README MUST give instructions about how to work with the corresponding Docker container. Please bear in mind that such a README will also be included as part of the Dockerhub documentation.

G5.- A GEi's Docker MUST be published at least under the 'fiware' account on DockerHub (if possible, please avoid the term 'fiware' in your repository name, as the username already contains it). The GEi

owner will be responsible for publication and maintenance operations. The publication method will be through the 'Automatic Build Procedure' which allows to link a Github repository to a Dockerhub one (and keep them in sync). In order to get access to that account and proceed with the publication, please get in touch with José Manuel Cantera.

G6.- Dockerfiles and Dockerhub repositories MUST be linked from the FIWARE Catalogue according to the Guidelines defined by the FIWARE Catalogue.

G7.- GEi Documentation MUST have Docker references (preferably through a link to the formerly mentioned README file)

G8.- Docker containers MUST be tested before being published to the community. Error in Docker materials of a GEi will be considered as critical and MUST be fixed immediately.

G9.- Dockerfiles SHOULD follow best practices as described by https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/