Dummy Cell Phones Factory

Production line simulation software

This document provides information about the architecture and use of the software that implements the simulation of the production line built for research purposes at University of Milan and described in the paper *Maggi, Federico, et al. "Smart factory security: A case study on a modular smart manufacturing system.".*

# Overview

The software simulates the production as it is described in the mentioned paper, so it is not only the architecture of the production line, but also the purpose, that is borrowed from the mentioned paper. For each device described in the paper, a Docker container exists in its simulation. The mobile device is simulated through a real Android app (*coming soon*) ready to be installed on any Android device and inclusive of source code. Additional Docker containers are created for auxiliary modules such as the orders systems, the magazine (warehouse), the relational database, the OpenTelemetry chain (Jaeger, Prometheus), the Apache Flume instance. A Postman collection is available [here](https://drive.google.com/file/d/1oS1VjQg44WszE6sK3FtDTP0J0yRb_vjM/view?usp=drive_link) that contains the requests that have to be done for creating the whole system from scratch on top of Docker by using [this](https://github.com/mircosoderi/State-of-the-art-Artifacts-for-Big-Data-Engineering-and-Analytics-as-a-Service) software framework available opensource on GitHub. The associated Postman environment is available [here](https://drive.google.com/file/d/1zcB4vLxj7pEG1iIDJ8H9zi_s4D66aGH9/view?usp=drive_link). The resulting system is summarised in a [spreadsheet](https://docs.google.com/spreadsheets/d/1LC_Os6TlAevcXZdfBE2-_r2kVKoXtiq1K1GVbhp5jpQ/edit?usp=drive_link) that contains a listing of the containers, including the domain names they use to refer to each other, the ports they expose, the Docker Images, the environment variables, and links to the Docker Volume backups to be loaded into the different containers for them to accomplish to their respective tasks. This document is meant to facilitate the porting of the ready-to-use system to platforms other than Docker, for example Kubernetes. The Node-RED applications are reconfigurable: the single parameters or the whole implementation can be changed via API requests with immediate effect. The documentation available in the [GitHub repository](https://github.com/mircosoderi/State-of-the-art-Artifacts-for-Big-Data-Engineering-and-Analytics-as-a-Service) and the [Postman collection](https://drive.google.com/file/d/1oS1VjQg44WszE6sK3FtDTP0J0yRb_vjM/view?usp=drive_link) will help you understand how that can be done.

# Getting Started

Once the system is created, add orders and items to the magazine, then start the production. Examples of how to do this are in the [TEST] API requests that you can find in the DCPF-Orders, DCPF-Magazine, and DCPF-ExecSystem sections of the [Postman collection](https://drive.google.com/file/d/1oS1VjQg44WszE6sK3FtDTP0J0yRb_vjM/view?usp=drive_link). Connect to the Web interface of the conveyor belt (for example <https://localhost:1882/ui>) to see the components flowing and being worked and assembled along their journey from the loader to the press, which is the last step. Similarly, most of the DCPF containers have a Web interface associated thanks to which you can see when they are working, and on what they are working. Looking at the conveyor belt, you will see that items will stop time by time, which is when they are worked by the PLC/sensor/actuator system they are in front of. After they have been worked on by the press, the single components cease to exist on the conveyor belt and the corresponding finished product is displayed. It can be a smallcellphone or a largecellphone depending on the order and the corresponding components that have been loaded on the belt and worked. Use the API exposed by DCPF-ConveyorBelt to remove the finished product from the belt, and use the API exposed by DCPF-Magazine to put the finished product in the magazine. Putting finished products in the magazine is important because when you call the delivery API on DCPF-Orders, it checks to see if all finished products are in the magazine.

# Self-signed certificates

All Node-RED instances expose their Web interface, and then also their APIs, over HTTPS, but the certificates in use are self-signed, so you will receive an alert from your browser when you attempt to connect to the Web interface of any node. You will typically need to hit “Advanced” at the bottom of the warning message, and then the link that allows you to go ahead to the Web interface even if “it is not secure”.

# Telemetry

All DCPF containers have an OpenTelemetry node in their Configuration flow. The node is configured by means of the OTEL\_EXPORTER, OTEL\_SERVICE, and OTEL\_PREFIX environment variables. For demo purposes, a Jaeger instance has been created on the same Docker instance where the whole system runs, and the OTEL\_EXPORTER points to that. At creation time, the Jaeger instance is configured to rely on a Prometheus instance also created on the Docker instance where the whole system runs. It’s easily foreseen that actual configurations in the production environment will be different. Also, Node-RED logs are exported to a demo instance of Apache Flume. This export is configurable through the environment variables FLUME\_LEVEL, FLUME\_HOST, FLUME\_PORT, FLUME\_METRICS, FLUME\_AUDIT. Having a look at the Node-RED settings files in the DCPF containers is the best way for understanding what each variable controls. You can add Node-RED debug nodes anywhere you need in the implementation, and configure them to output to console, and you will have the event sent to Apache Flume, in addition to those that are automatically generated by Node-RED. The flume support is not available on Kubernetes (see below); it can be used only on Docker.

# See also

Important information for the deployment of the simulation on Kubernetes, instead of Docker, can be found in the [Dummy Cell Phones Factory on Kubernetes - README File](https://docs.google.com/document/d/1BpDf6tR0ouuhQ4JNFNOaxIyuFW0wWMNae2xvlh5e7to/edit?usp=drive_link). The document also contains a pretty detailed description of the simulation: how to get started, what the different components do, how you can monitor the “production”, and what you should do after that you have made any changes to the implementation of any of the software components to ensure that the deployment and service files can be effectively used for the deployment of the simulation on a different Kubernetes cluster without losing the changes that have been made.

# Contacts

Mirco Soderi, PhD (he/him/his)  
[#StartsWithAName](https://www.startswithaname.net/) Mirco (first n., 🔊 **miracle**), Soderi (🔊 **so**rry - **de**spite - **ri**thm)  
[Data Science Institute (DSI)](https://dsi.nuigalway.ie/) – Postdoctoral Researcher  
[CoSE EDI Committee](https://www.universityofgalway.ie/science-engineering/edi/cose-edi/) – Researchers Representative  
[University of Galway](https://www.universityofgalway.ie/)  
Phone: +353894916707  
Working hours: Mon – Fri, 9:00 – 17:00, [Irish time](https://en.wikipedia.org/wiki/Time_in_the_Republic_of_Ireland)  
Office: [DSI Building](https://goo.gl/maps/386as6vwNamwNj9u7), 1st Floor, South Wing, Desk 225