FSC Architecture

Learning from Data Challenges

A.Formica

Outlines

- Architecture overview
  - Principles : SOA in the cloud
  - FSC cloud Architecture during Data Challenges
- FSC cloud: the integration site infrastructure
  - From git to the swarm
- FSC services architecture description
  - An example for DC1….
- Conclusions

Architecture principles

SOA

- (Micro) Service Oriented Architecture
  - Agile architecture, both in terms of development process and testing process (suits with geographically distributed community of developers)
  - Single components are “autonomous”
    - They are deployed in docker containers
- How to integrate the software produced ?
  - From last year we started Data Challenges…we are still learning but some better view of the architecture and the process is now possible

Naming conventions

- Services
  - Software components which provide in general data management capabilities (data classification, formatting, storage, triggering actions on data availability …)
  - They interact with DBs (which are also services)
- Pipelines
  - Software components which provide in general science products generation
  - They interact with the services to deal with inputs and outputs
  - They are in general “called” by services

Communication Protocols

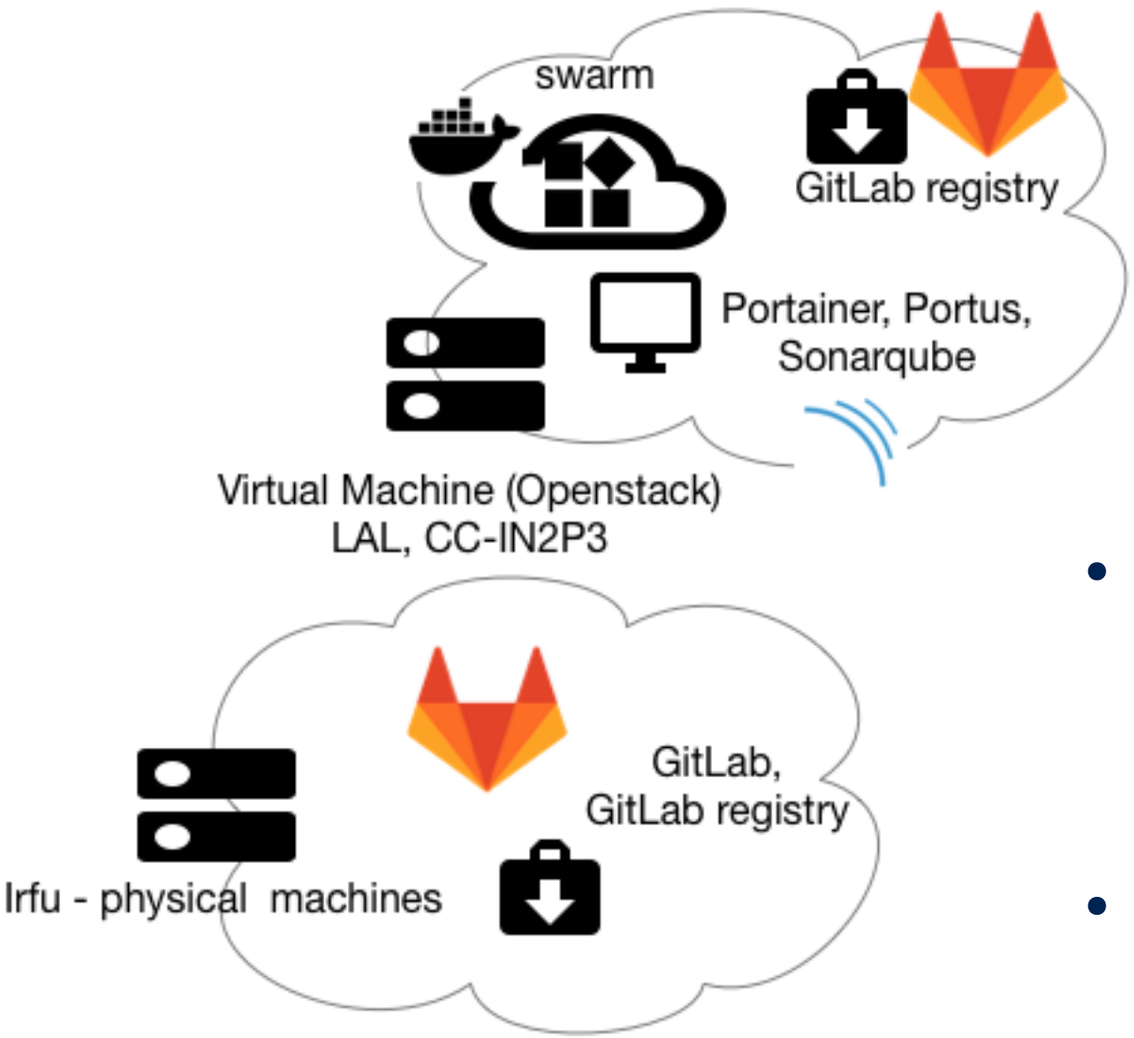
- Peer to peer connections: REST (HTTP)
  - Services and pipelines provide to external clients a REST API
    - configuration, data management, monitoring, science products generation, calibrations, ….
- Broadcast connections: Messaging (NATS)
  - Every service provides listeners and/or publishers to specific message queues
    - trigger further processes, send logging and monitoring informations…

Interfaces

- REST APIs and Message types contents
  - interface definition is essential because this is what a client should use
  - standard OpenApi for REST service specification (Swagger)
  - describe **API** and **data model** of the exchanged content in JSON (or YAML) format

FSC cloud, a general description

FSC infrastructure

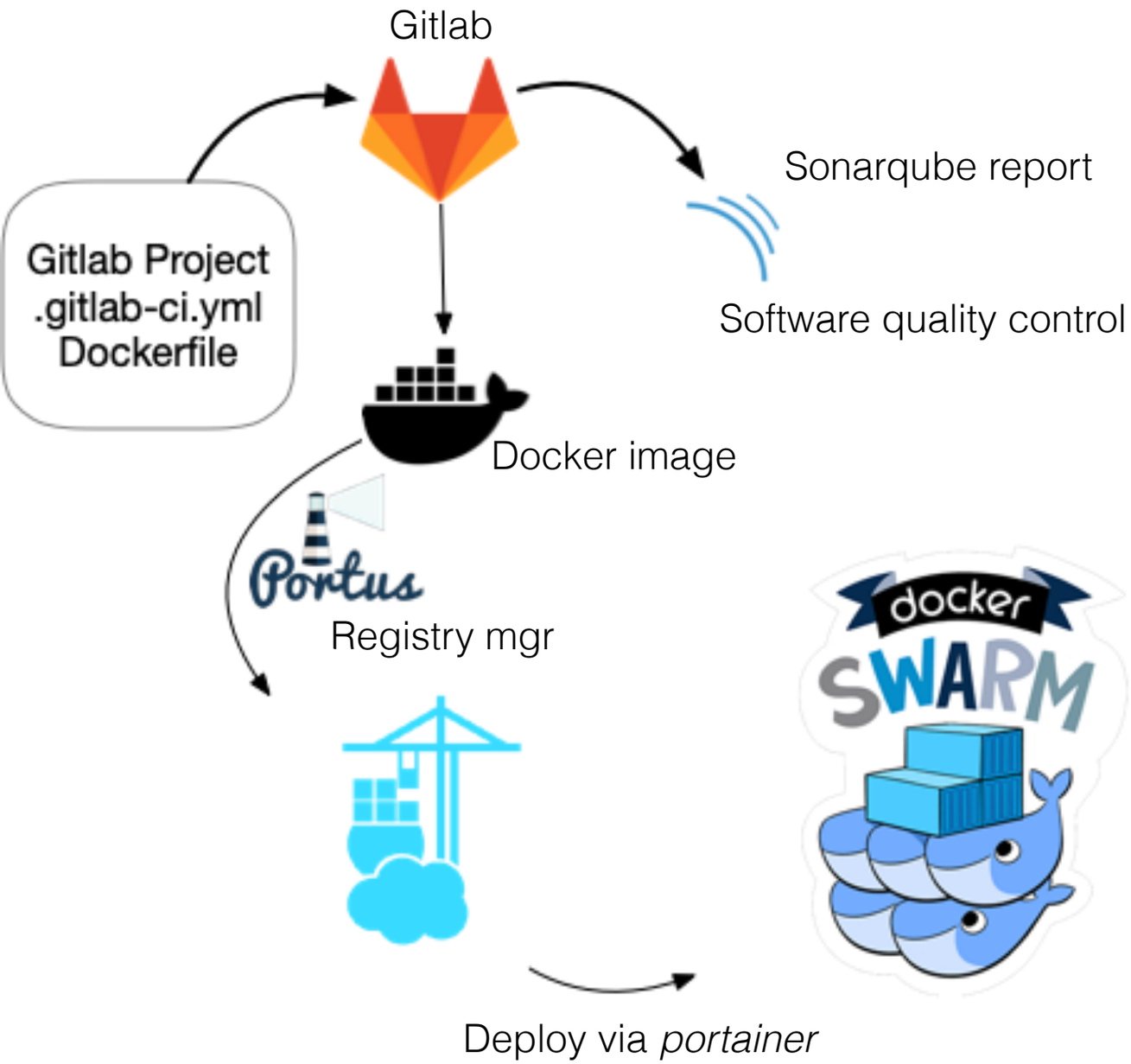


- Development environment
  - experts (from all labs) use the gitlab server to version their software projects
  - Each project has its own “Dockerfile”: a recipe to package it in a container
  - Docker images produced can be stored in Svom gitlab registry from where they can later be retrieved
- Svom deployment project
  - A special project is used to gather other important information for the deployment : the configuration, secrets and volumes for the services, deployment files
- Swarm administration tools
  - Several services are in place to speed up the deployment and check the status of running containers (*portainer*)

FSC infrastructure /2

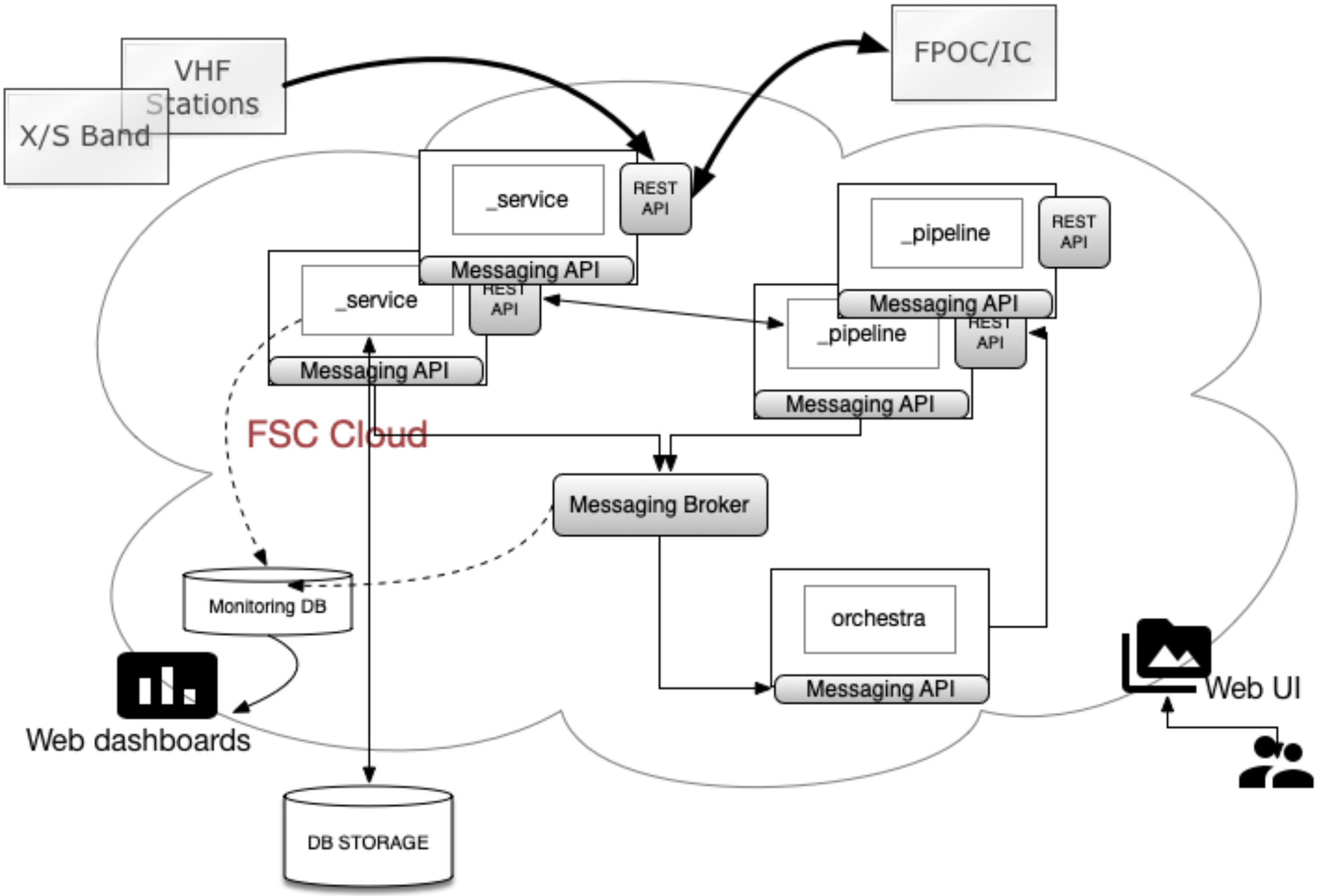
- Infrastructure @ integration site
  - LAL Orsay delivered a set of VMs for our initial tests
    - About 10 VMs (2-4 cores with 4-8 GB RAM)
  - LAM, LAL and IRFU : setup basic orchestration services
    - docker swarm created with a subset of VMs, Portainer and Portus web ui to administer stacks and images/registries

Dev cycle and CI



- Development cycle
  - Update software and push into gitlab server
  - Continuous integration starts
    - compile, test, provide sonarqube report, create docker image and push it to Svom docker registry
  - Deployment file
    - All informations for the deployment are available in a specific git project (svom-deployment)
  - This file is used directly by the experts to deploy their stack (a set of docker containers) into the swarm (via portainer web server)

Cloud architecture (DC)



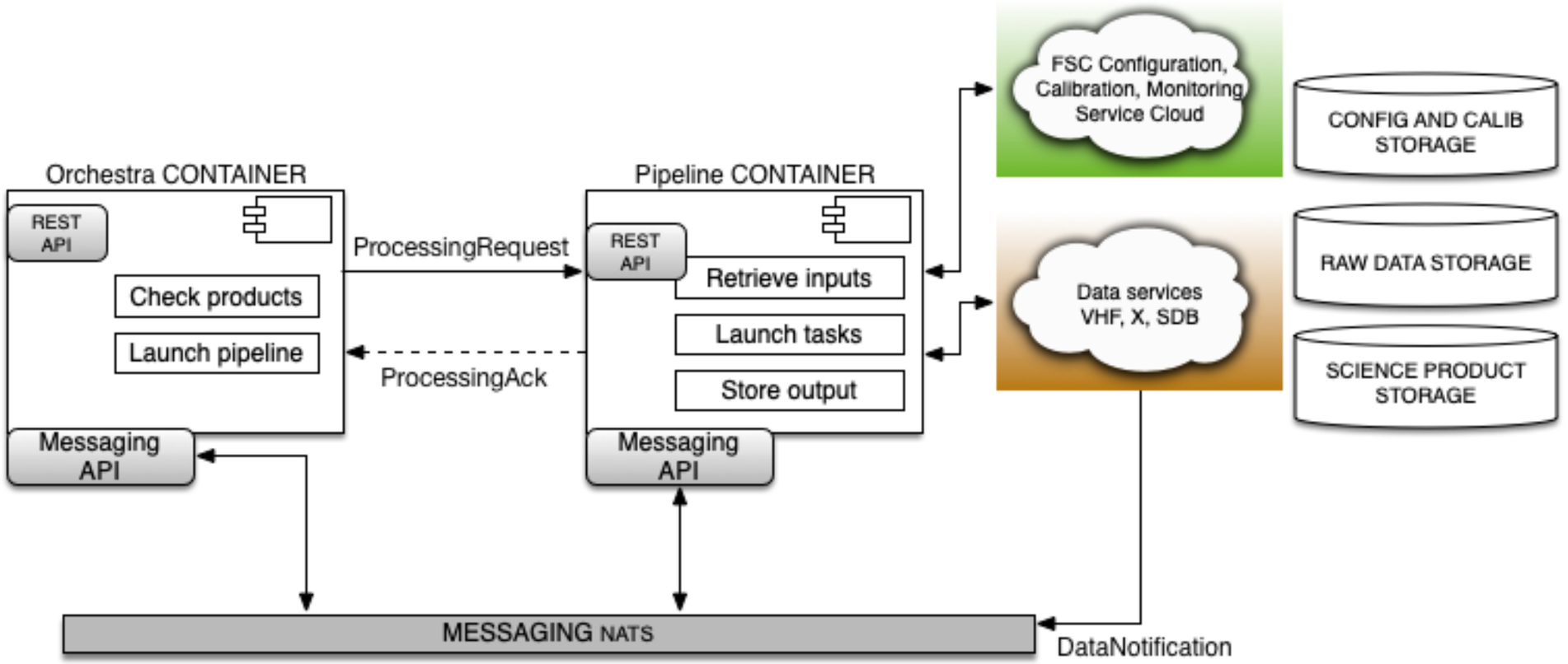
Low level services

- Postgres DB
  - Now running in a docker (temporary)
  - Will be installed outside the swarm on some ad hoc VM
  - It hosts ALL needed schemas (VHF packets DB, X-band packets DB, crest DB, ScienceDB, …)
- InfluxDB
  - Running in a container
  - Time series DB for logging services “life”, containers CPU etc.
- Messaging NATs (broker)
  - Running in a container
  - It is accessed by almost ALL services / pipelines
- Monitoring
  - Running in a container
  - Web UI with Grafana dashboards filled via InfluxDB queries

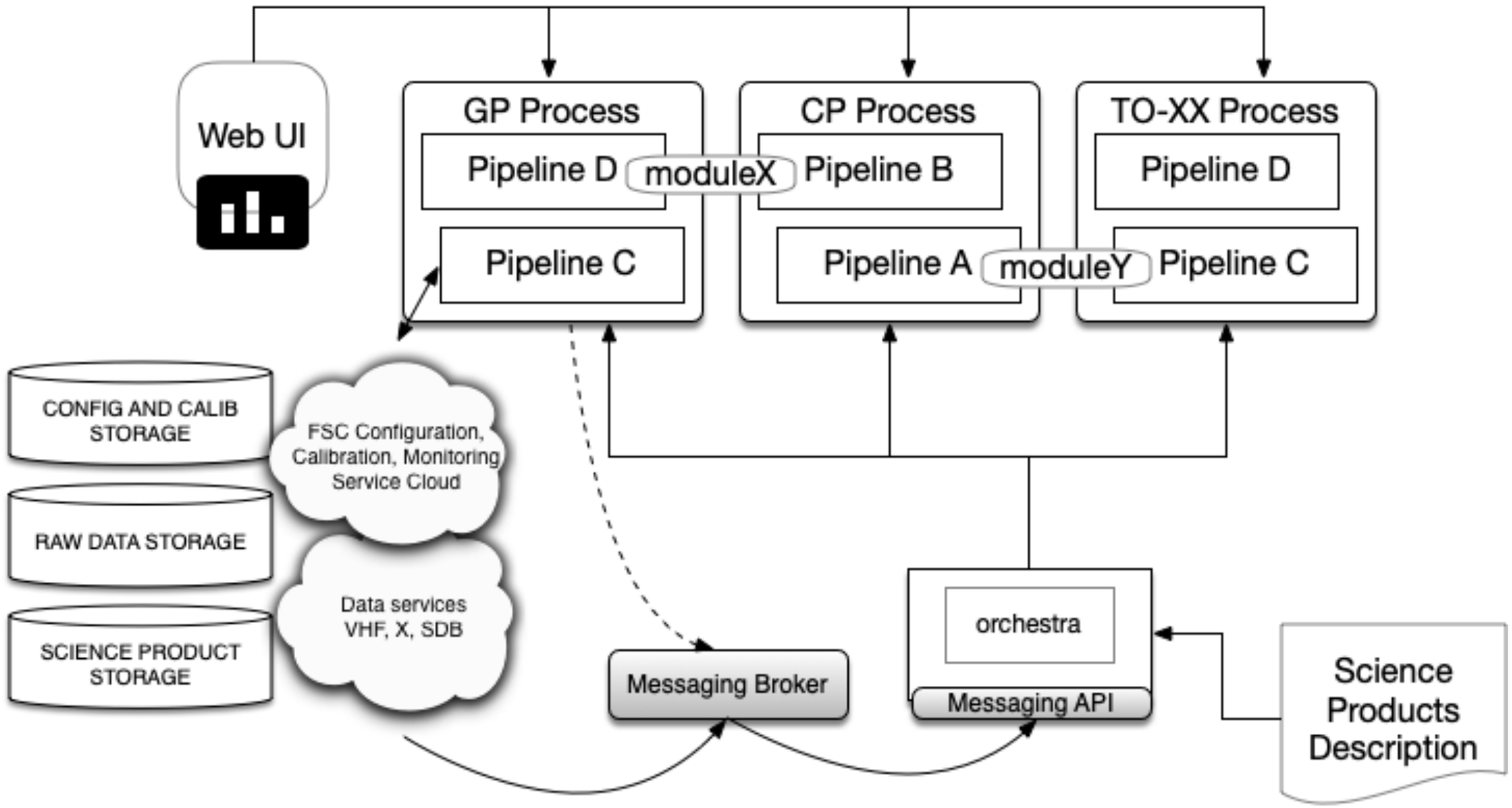
General services

- General services: used by many other services…
  - *Configuration*: store time varying configuration for services in a simple ascii format (work-plans etc…) **(DC)**
  - **User management**: store user related information, can be used later on for authentication and authorization
  - **Calibration service**: store time varying files from instrument centers, prepare CALDB environment for common access from pipelines **(DC)**
  - **Logging and monitoring**: log and monitor information on the status of the other services (not “physic” monitoring) **(DC)**

Generic pipeline



Processes inside FSC



Future milestones

- Several system tests occurring in 2020
  - FSGS internal interfaces (IC-FPOC-FSGS)
  - FSGS external interfaces (which Chinese centers)
  - Several tests using simulated data that shall then be processed (mock) and exchanged between centers in France and China : x-band sequence, vhf sequence, …
- Generation / simulation data (GRB sequence)
  - Several test cases will need a certain level of simulated data both in VHF and X-Band formats. The effort is on going already for DC1.
- Preparation using DC1
  - We consider that DC1 is a first step to be able to face future system tests occurring during 2020

Conclusions

- Architecture principles and tools
  - REST, Messaging (NATs) and OpenAPI
  - Started the validation of these principles using some prototypes (VHF, MXT, GFT + raw data simulation)
- Integration
  - Gitlab-ci / Sonarqube / Swarm : fully automatized chain is in good shape, should be partially ready for DC1 (at least for some services)
- Deployments for DC1
  - Focus on few use cases for VHF and X-Band data: most of the code is ready but need now further testing to guarantee that all communication between services and pipelines are working
  - All tests are a preparation to system tests that are foreseen in 2020
  - Monitoring is still preliminary (grafana/influxDB/…telegraf…)