

The French Science Center for SVOM experiment

Andrea Formica - Irfu CEA/Saclay
Henri Louvin, Jean Paul Le Fevre, Lea Jouvin, Maxime Bocquier
On behalf of FSC SVOM team



Outline

- **Svom experiment**
 - French collaboration
 - Data flow: from satellite raw data to science products
- **FSC in the SVOM Ground Segment**
 - Global architecture of SVOM ground segment
- **FSC architecture**
 - Services, APIs and data model
- **Ecosystem**
 - Continuous integration and deployment



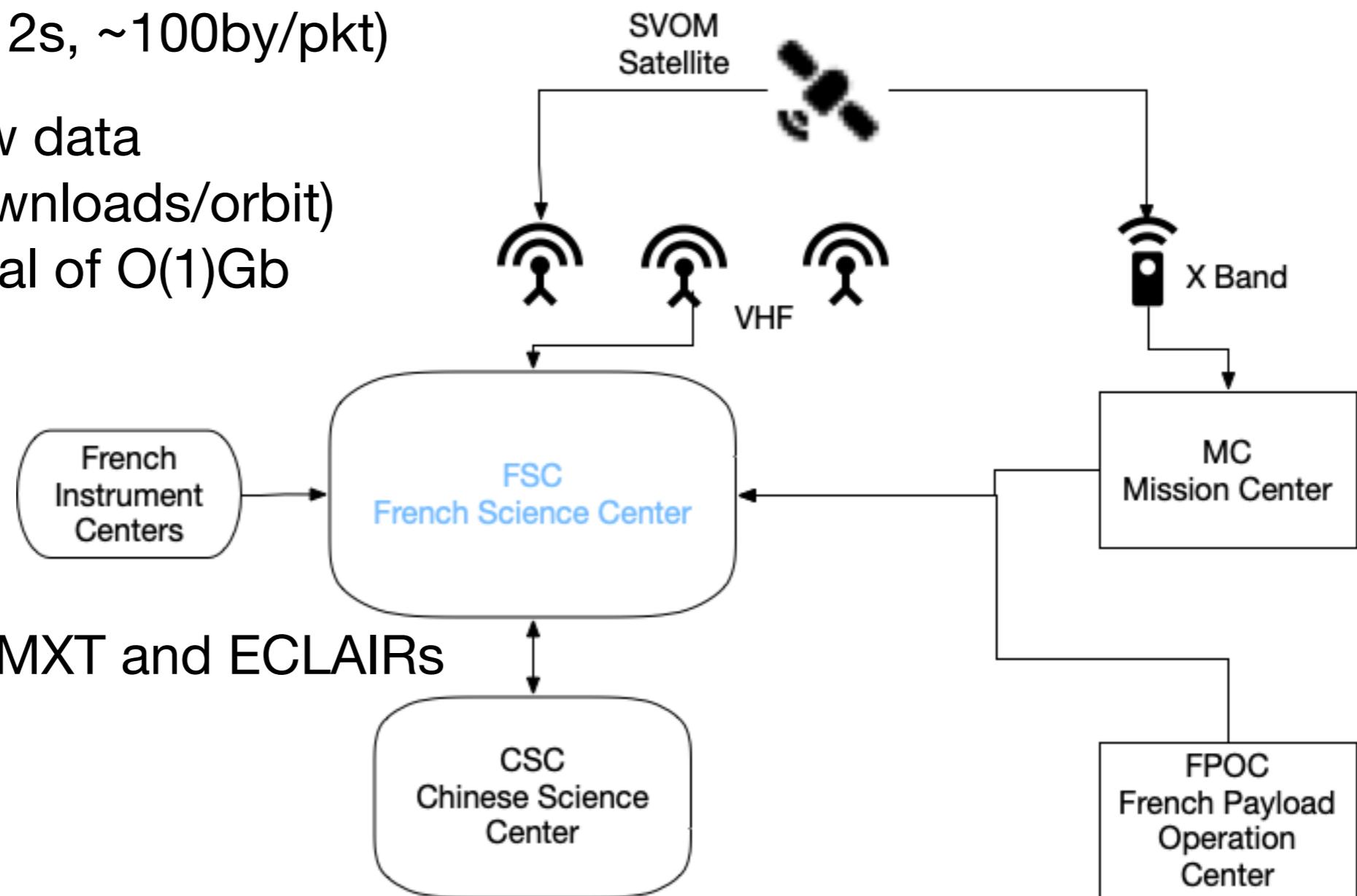
French Collaboration

- **Laboratories**
 - Cnes, Lam, ObAS, APC, LUPM, IAP, Gepi, Irap, CPPM, IJCLab, Irfu (Dap, Dedip)
- **Irfu Dedip**
 - Management, architecture, development of common services, integration....
- **Irfu Dap**
 - For the Ground segment contributions: A.Sauvageon, K.Tazhenova, R.Bouteloup, T.Sadibekova
 - Interfaces and low level data formats, developments of pre-processing pipelines, instrument center for MXT,...

General Architecture and data flows

VHF: “Real time” workflow (40 stations, 1pkt / 2s, ~100by/pkt)

XBAND: full raw data workflow (2 downloads/orbit)
~1Kb/pkt, a total of O(1)Gb



French payload: MXT and ECLAIRs

Chinese payload: VT and GRM

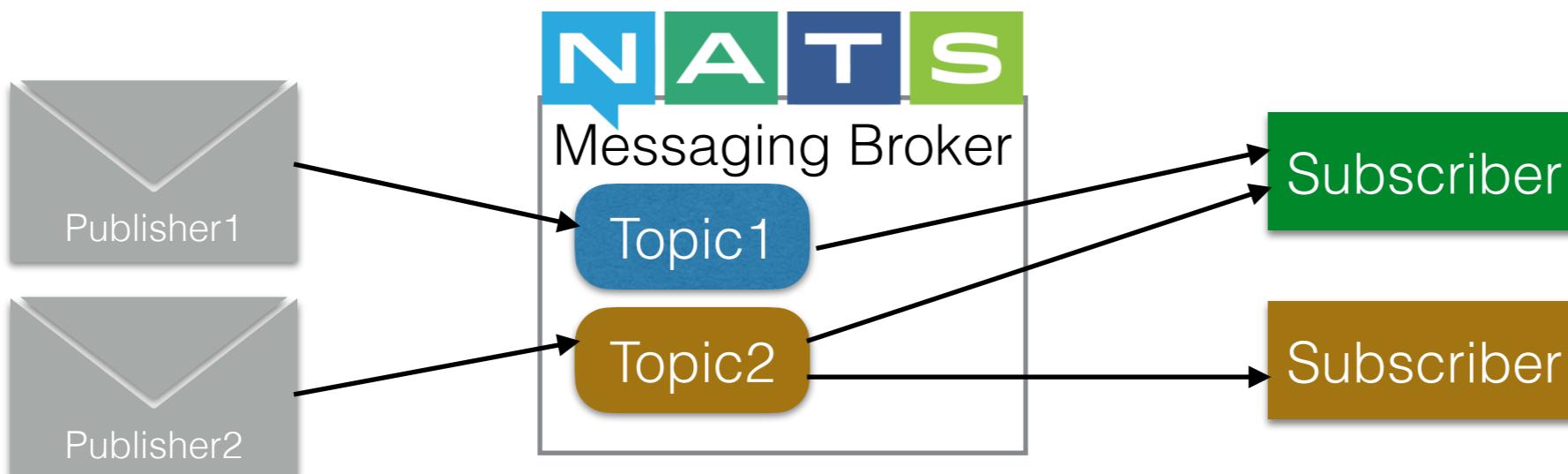


Architecture principles

- SOA (service oriented architecture)
 - Define small services dedicated to specific tasks
 - DB management (for satellite data, science products, etc)
 - Data processing (algorithm processing data at different level)
 - Orchestration of services (based on data availability triggers activity of other services)
- Architecture and Protocols
 - REST API : services can retrieve and store information using dedicated REST APIs (...other services) which are described in ad hoc documents. This type of exchange is in general synchronous.
 - Messaging : services may be alerted of new data arrival (satellite, calibration or other...) via messaging protocol. This type of exchange is asynchronous.

Messaging systems

- NATS (Jetstreaming)
 - A light and fast messaging system
- VOEvent
 - ▶ COMET: receive messages from outside community and send gamma ray burst alerts for internal SVOM entities
- RabbitMq (MQTT)
 - ▶ Exchange information between FSC and CSC (China)



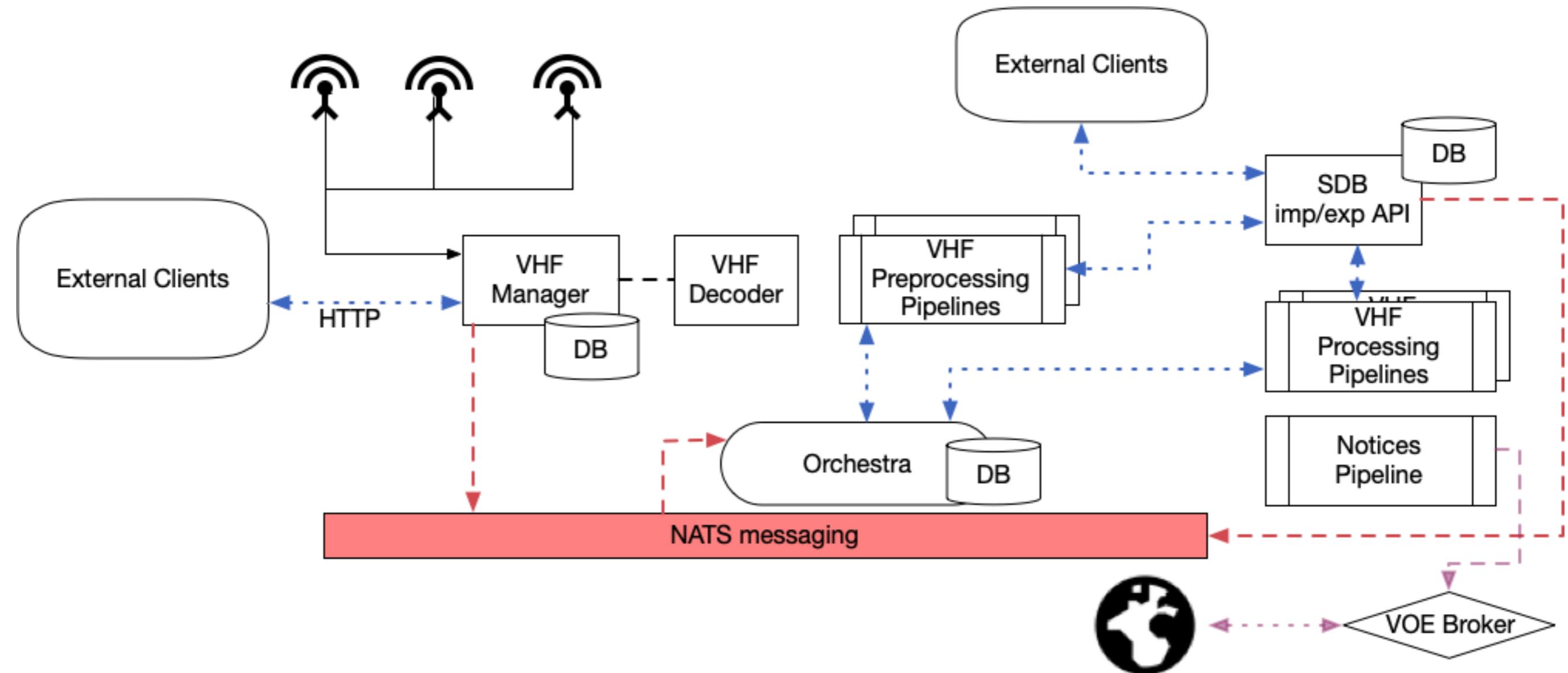


Technologies overview

- Multi-language
 - Official support for Python and Java. Interoperability guaranteed by architectural choices (REST and messaging)
 - C/C++ code is supported essentially for on board satellite libraries used at FSC
- Python
 - Flask (web server for REST)
 - Django (in ICs)
 - astropy, pandas, ...
- Java
 - Spring Boot (undertow as web server, JAX-RS, JPA)
- XML, JSON, FITS
 - XML is used by Cnes to define the raw data formats (home made code generator for decoding libraries in python)
 - JSON: format of all exchanges inside FSC
 - FITS: science product
- Web UIs
 - Javascript : Angular, React

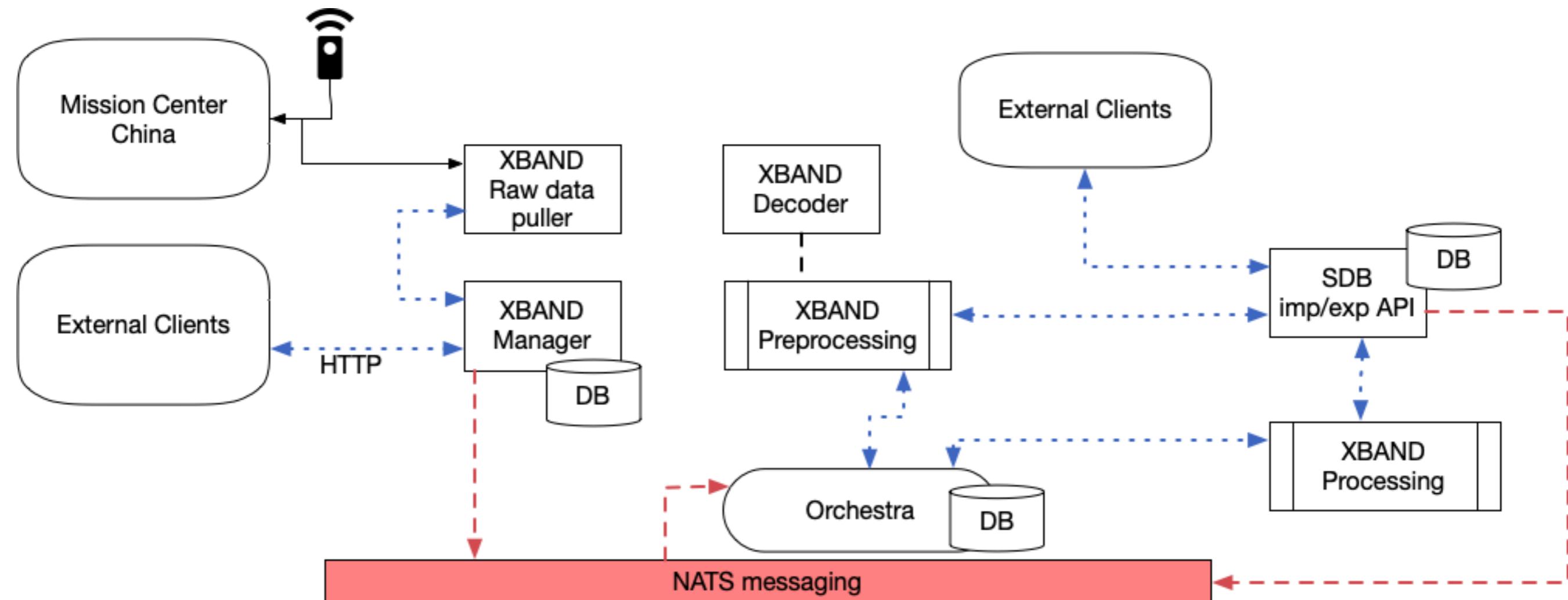


Data flow: VHF

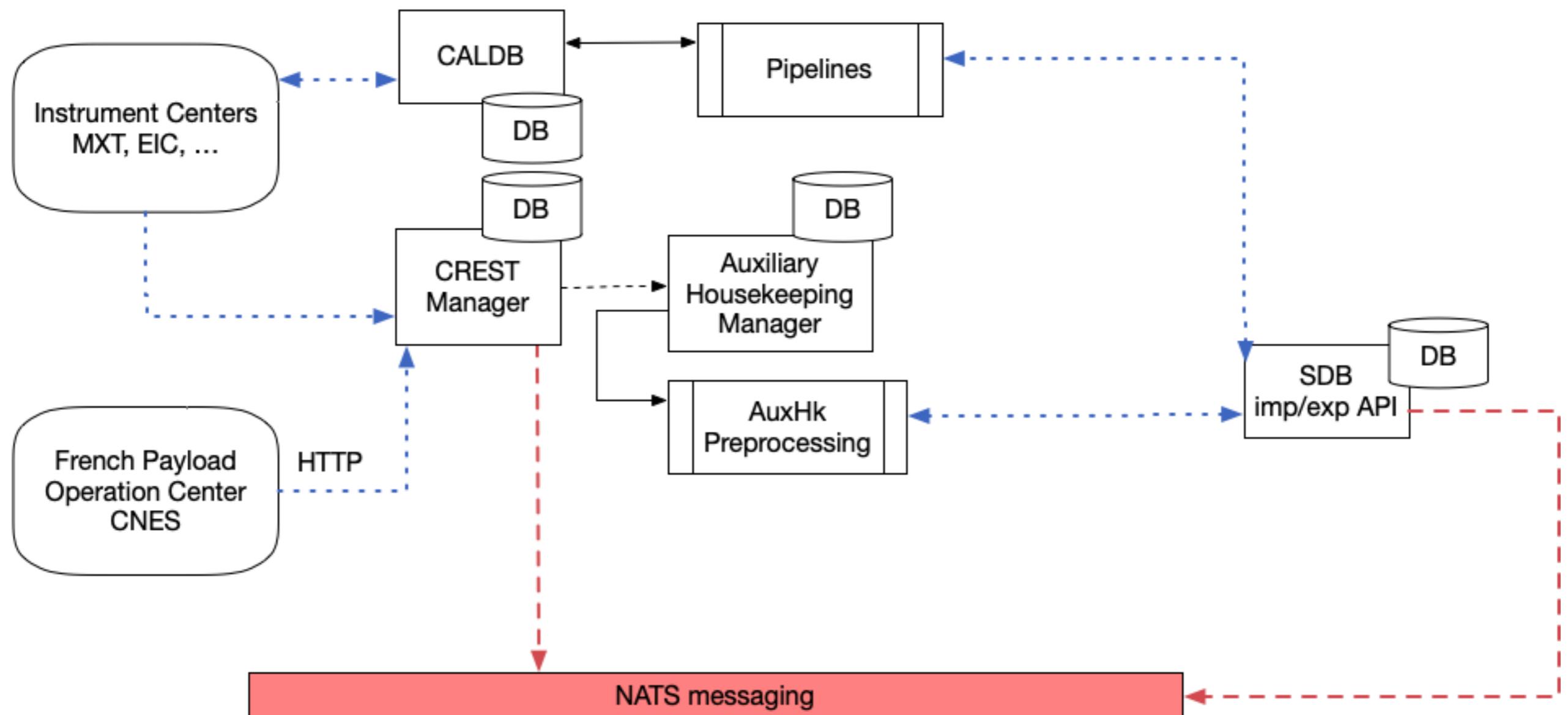




Data flow: XBAND



Common services



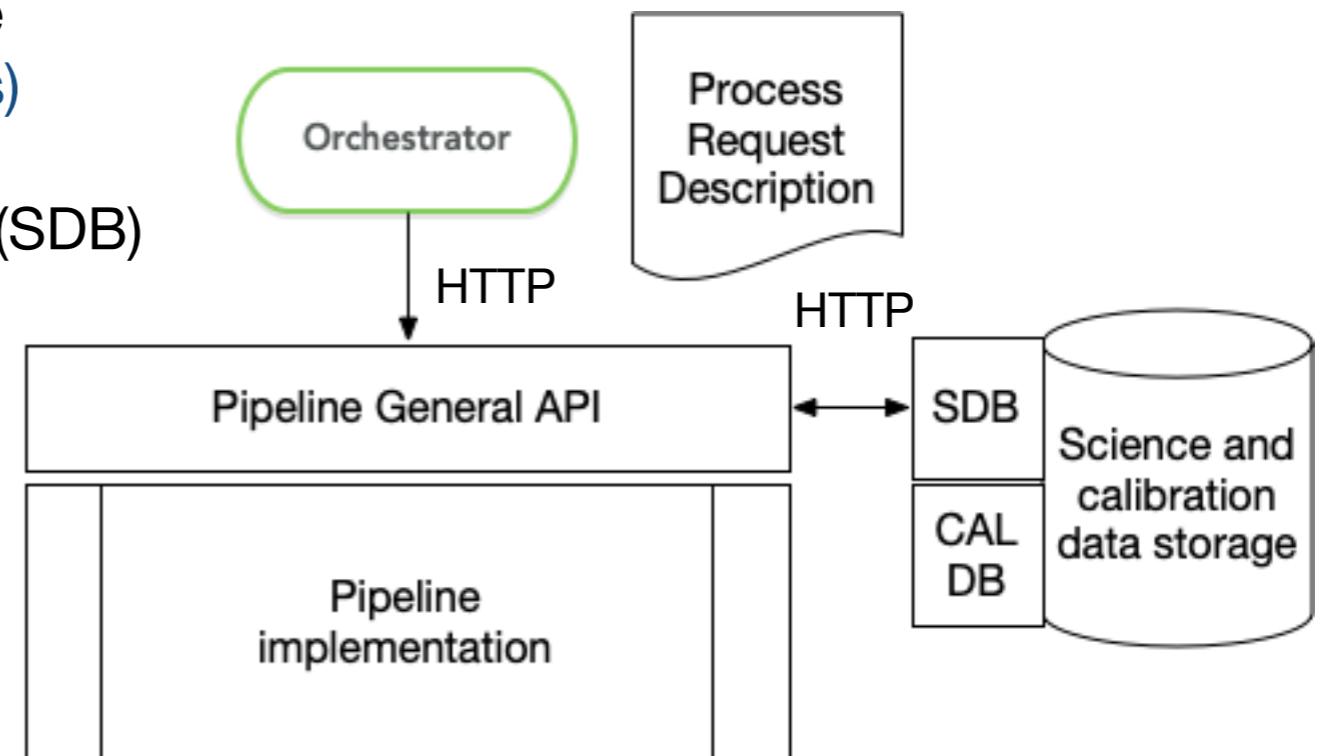
Pipeline architecture

Definition of a generic API for each pipeline

Definition of data model to trigger the pipeline

Processing Request (mandatory parameters)

Pipeline only contacts the Science DataBase (SDB) and the Calibration DB (CALDB)



The API definition is useful so that orchestrator can start a processing requests once specific input products which are needed by each pipeline have been registered into the SDB.

To announce available new products the NATS messaging system is used.



API and services

- API is a contract

- In an environment with a large number of services having different roles it is important to establish in a clear manner the contract that each service expose
- In Svom we were not too strict in this approach, but a certain number of our services have a well documented contract.
- We instead have been more strict in documenting the formats of the JSON objects we use when we exchange information (via messaging and http)

- What do we need to describe ?

- SOA architecture

- Define the functions available via REST: address of the system (URL) and the signatures
MyObject fun1(SomeType a, SomeOtherType b)
In a web URL we can pass parameters :
 - via Path (**xxx/{name}/resources**)
 - via Query (**xxx/resources?name=somename**)
 - or directly in the body of the request (typically used for insertions and updates)
 - The same URL can be declined in different functions by using a different HTTP method (POST,PUT,GET,DELETE,...)



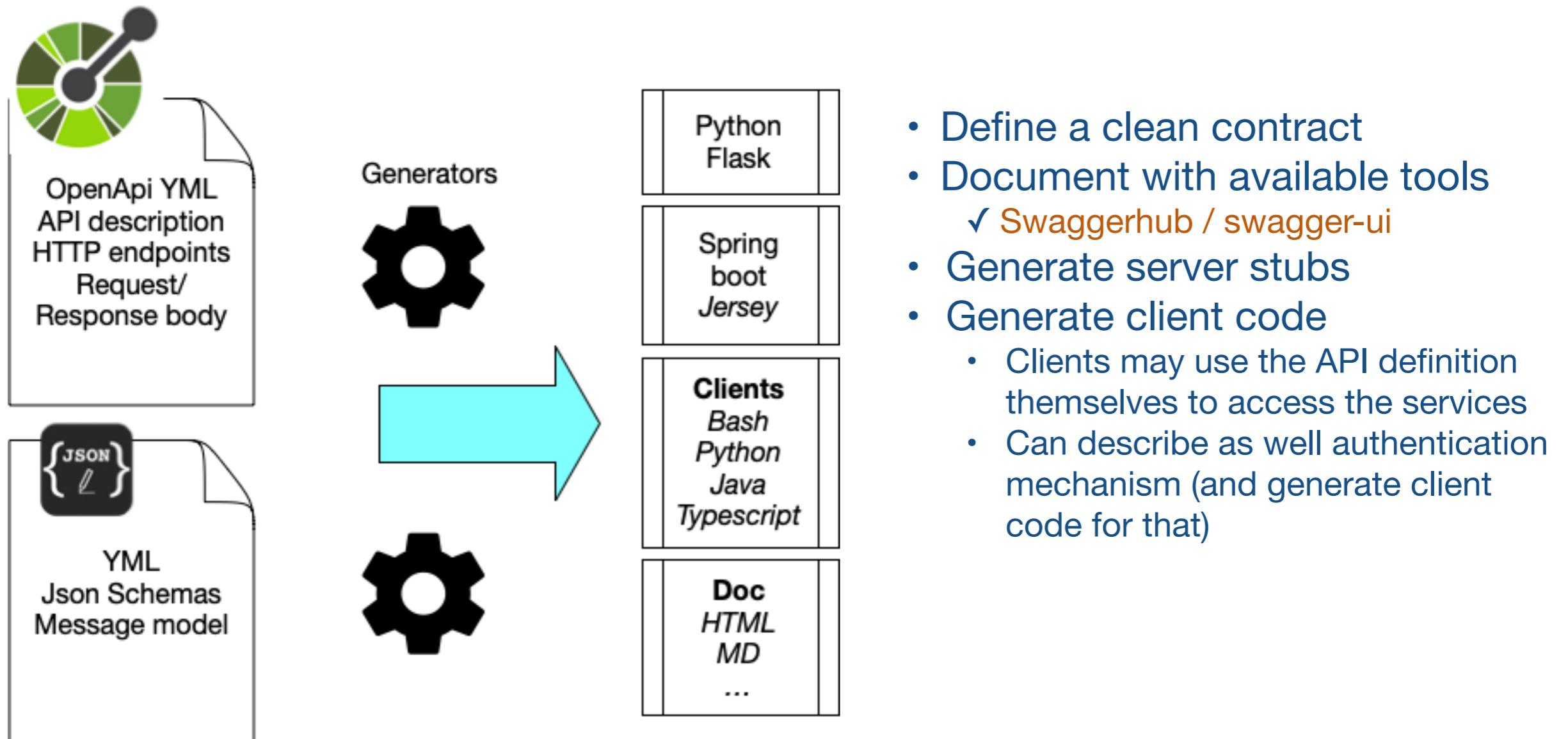
Example of REST API

Property	Value	Description
URL	http://host:port/api/vhfmgr/apids	Return a list of APIDs from the VHF system
HTTP verb	GET	Make a GET request to the server
Query Param	name [=someapidname]	Define parameters in the URL
Response data	{ "name" : "someapidname", "apid" : 123, "description" : "a description" }	The JSON response (an object, may be containing a list of other resources)
Response code	200	The HTTP response code; 200 means request was successful
CURL example	<code>curl http://host:port/api/vhfmgr/ apids?name=someapidname</code>	An example for curl command with query parameter



API specifications

A large part of our APIs are described via OpenApi





OpenApi examples: endpoints

```
paths:
  /apids:
    get:
      tags:
        - apids
      summary: Finds a packet APID list.
      description: "This method allows to perform search
                    arguments: bv={searchpattern}.\
      operationId: listPacketApids
      parameters:
        - name: by
          in: query
          description: "by: the search pattern {none}; s
                        , description, name]"
        schema:
          type: string
          default: none
        - name: page
          in: query
          description: 'page: the page number {0}'
          schema:
            type: integer
            format: int32
            default: 0
```

```
post:
  tags:
    - stations
  summary: Saves VhfGroundStation
  description: Create new station
  operationId: createVhfGroundStation
  requestBody:
    description: 'A json string that is used to construct a
                  vhfgroundstationdto
    object: { name: xxx, ... }'
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/VhfGroundStationDto'
    application/xml:
      schema:
        $ref: '#/components/schemas/VhfGroundStationDto'
  required: true
responses:
  201:
    description: successful operation
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/VhfGroundStationDto'
      application/xml:
        schema:
          $ref: '#/components/schemas/VhfGroundStationDto'
```

Describe URLs and their parameters, inputs and outputs



OpenApi examples: components

```
VhfDecodedPacketDto:  
  description: data container for json format, inherit from generic  
  vhfbasepacket  
  allOf:  
    - $ref: '#/components/schemas/VhfBasePacket'  
    - type: object
```

```
VhfResponseDto:  
  type: object  
  properties:  
    size:  
      type: integer  
      format: int64  
    datatype:  
      type: string  
      format:  
        type: string  
    page:  
      $ref: "#/components/schemas/RespPage"  
    filter:  
      $ref: '#/components/schemas/GenericStringMap'
```

A component is like an object: you define the attributes and you can reference other objects inside them.

The OpenApi allow for a basic description of inheritance.

Describe the components you use in the API methods



Publish your API: swagger hub

SMARTBEAR
SwaggerHub™

vhfmgr-api 2.1.2

PUBLISHED SYNC

Aa ☀️ Read Only

VhfDecodedPacketDto ▾ {

```
description: data container for json format, inherit from generic vhfbasepacket
```

pktformat* string

pktype* string

hashId* string

insertionTime integer(\$int64)

pktsize* integer(\$int64)

packet* string

A string representation of the packet content (can be JSON or base64 string, or RAW for hexa string)

uri string

}

VhfBinaryPacketDto ▶

VhfResponseDto ▶

VhfBurstIdSetDto ▶

VhfNotificationSetDto ▶

Last Saved: 2:10:10 pm - Sep 24, 2021

! VALID

Info Tags Servers Search apids stations vhf

GET /apids

POST /apids

GET /apids/{id}

PUT /apids/{id}

GET /stations

POST /stations

GET /stations/{id}

PUT /stations/{id}

GET /vhf

GET /vhf/stationstatus

GET /vhf/stationstatus/count

GET /vhf/rawpackets

GET /vhf/decoded

1384 description: The byte array representation of the input packet, this parameter is optional

1385 format: byte

1386

1387 VhfResponseDto:

1388 type: object

1389 properties:

1390 size:

1391 type: integer

1392 format: int64

1393 datatype:

1394 type: string

1395 format:

1396 type: string

1397 page:

1398 \$ref: "#/components/schemas/RespPage"

1399 filter:

1400 \$ref: '#/components/schemas/GenericStringMap'

1401 discriminator:

1402 propertyName: format

1403 VhfBurstIdSetDto:

1404 description: A set containing VhfBurstIdDto objects.

1405 allOf:

1406 - \$ref: '#/components/schemas/VhfResponseDto'

1407 - properties:

1408 resources:

1409 type: array

1410 items:

1411 \$ref: '#/components/schemas/VhfBurstIdDto'

1412 VhfNotificationSetDto:

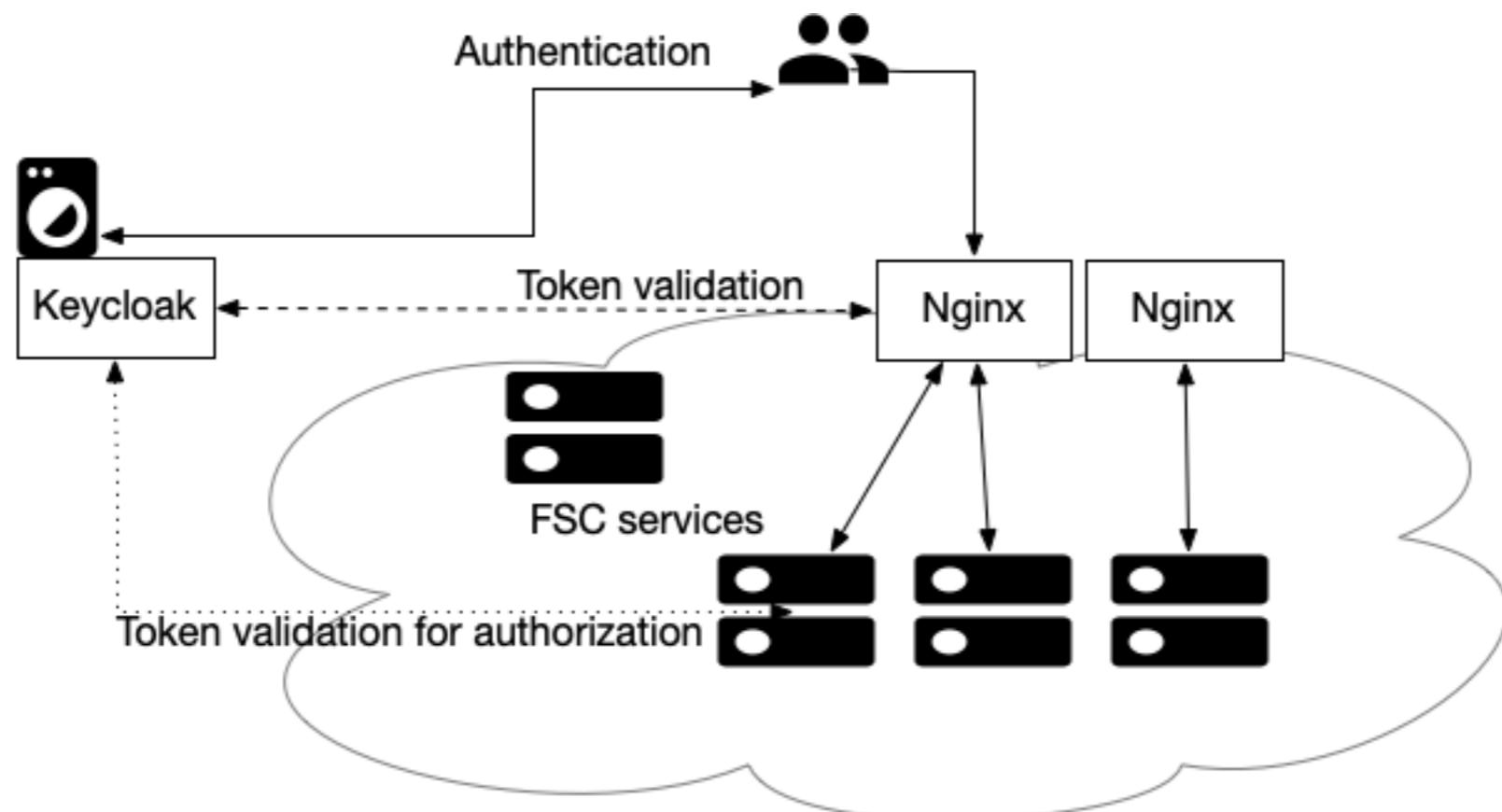
1413 description: A set containing VhfNotificationDto objects.

1414 allOf:

1415 - \$ref: '#/components/schemas/VhfResponseDto'

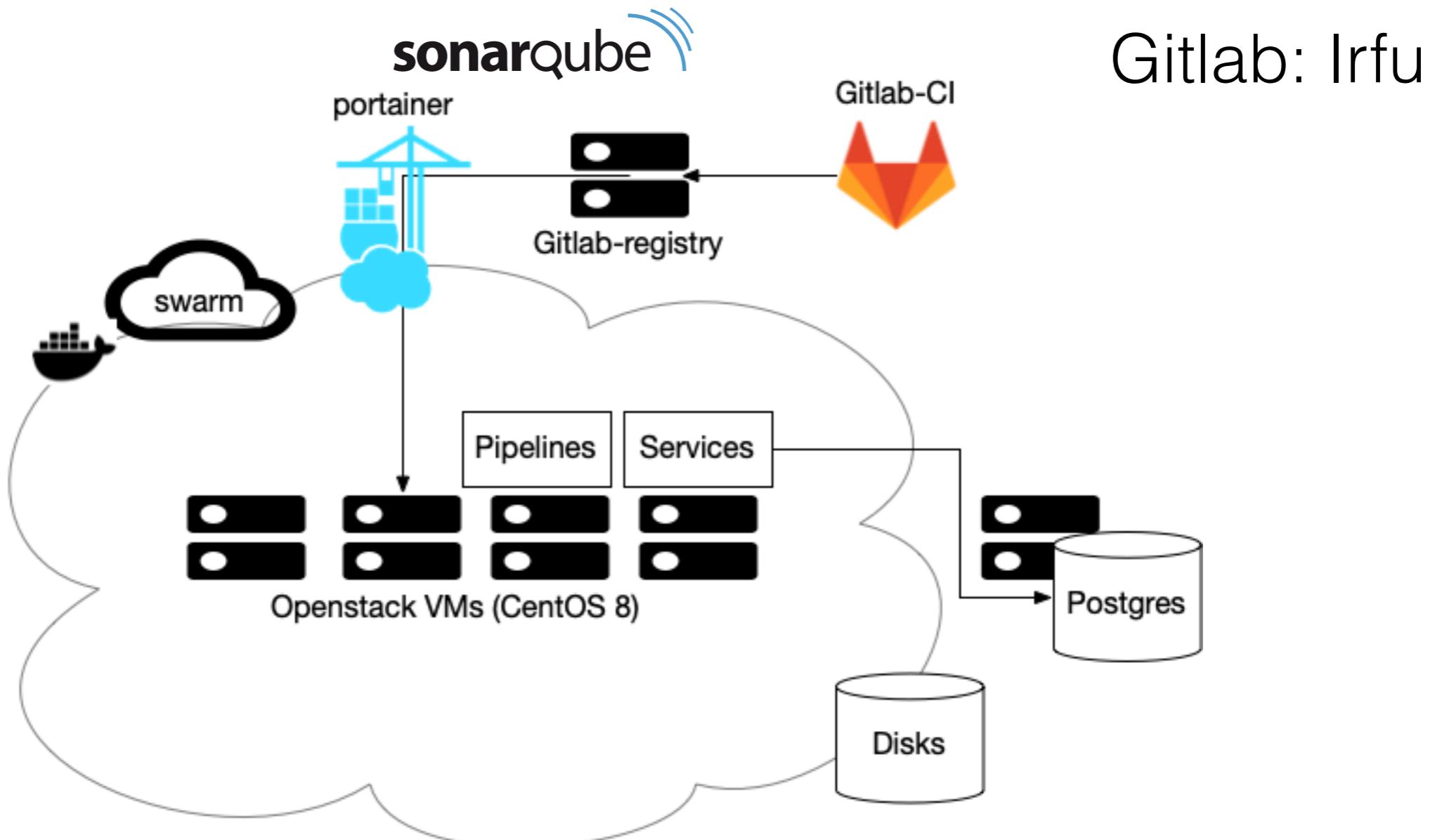


FSC Authentication





FSC deployment



Integration site: IJCLab

Production site: CC-Lyon



FSC Monitoring

- Monitoring of the infrastructure (on going)
 - ▶ Utilization of log files (graylog, grafana+telegraf,...)
- Monitoring for services
 - ▶ REST API: use the same APIs to retrieve informations on underlying storage and related data
 - ▶ In case of OpenApi description, generate typescript modules to integrate in a javascript framework (Angular, React,...)
 - ▶ Web UIs deployed for: services orchestration, VHF data, Science products generated, ...



Next steps

- Deployment @ CC
 - Explore monitoring solutions ? (ES...)
 - Kubernetes ?