Information system evolution at the French National Network of Seismic Survey (BCSF-ReNaSS)

Fabien Engels, Marc Grunberg
fabien.engels@unistra.fr, marc.grunberg@unistra.fr

INTRODUCTION

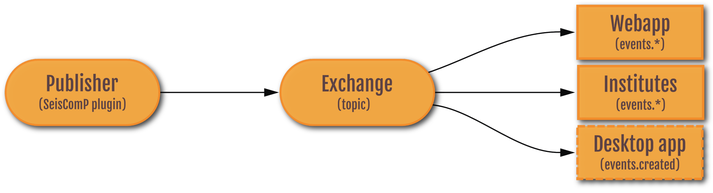
During last years, our observatory had to update its information system. This poster introduces some of services deployed and the new workflow for realtime detection of seismic events.

A SEISMOLOGICAL FRAMEWORK

One of the big step was to switch to SeisComP3. This software offers many features : data acquisition (based on seedlink), processing (realtime detection) and interactive analysis (picker app). To know what's going on during processing, it's possible to subscribe to different kind of messaging group through Python plugins. So with few lines of code we can easily know when an earthquake is detected and trigger actions.

ADD A POSTMAN TO THE SYSTEM

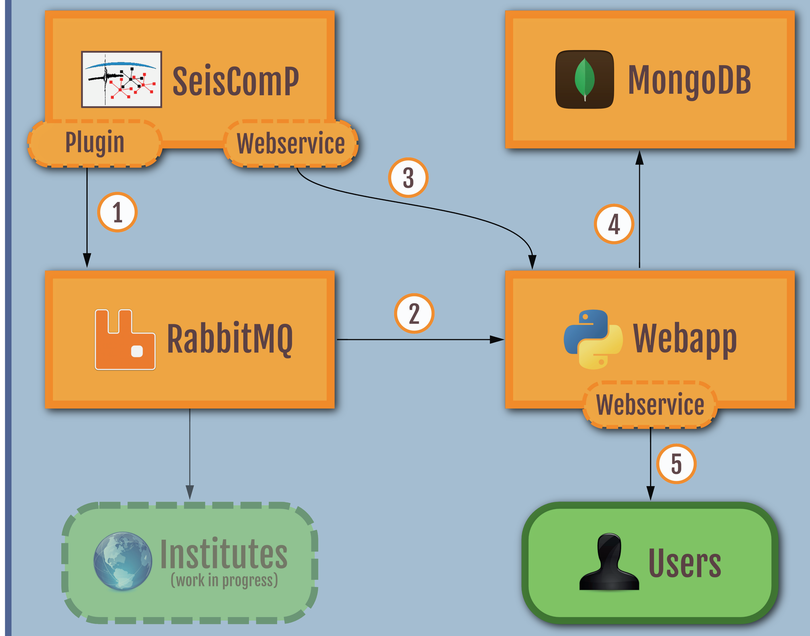
RabbitMQ is a message broker, it brings agility by decoupling the detection system from the rest of the system. The idea is to propagate information by pushing messages to a set of clients : Scripts, Notification applications (desktop & smartphone) and in midterm, we would like to offer a direct access to RabbitMQ to institutes who wants to get events in realtime.



Messages are based on GeoJSON which is easily readable by any language. Even if GeoJSON is a standard, there is nothing defined about metadata, like url or magnitude properties. As USGS already uses this format, we decided to use the same name for properties. These latters are stored in queues waiting to be consumned by clients. Queues can be permanent or volatile and receive only creation or update messages depending of client usage.

A NEW WORKFLOW

We aim to publish new events to make them available through a FDSN Webservice as soon as possible. To reach this goal, the locations flow was utterly rethink. When an earthquake occured, steps from database to webservice are these followings :



1. The SeisComP plugin create a message in RabbitMQ
2. Python retrieve RabbitMQ message
3. Then retrieve event data using SeisComP FDSN webservice
4. Data are dumped in a MongoDB database
5. Users retrieve the catalog using a FDSN Event webservice

SQL OR NOSQL, WHY NOT BOTH ?

As our SeisComp instance use PostGreSQL as backend. It ensures data integrity using SQL advantages. But due to SeisComP database modeling, it often difficult to query the catalog. Dumping data (keeping QuakeML modelization) in a MongoDB database allow us to solve this issue. We also take advantages of NoSQL for datamining usage like Map Reduce algorithms and performance gains.

ONE LANGUAGE TO REPLACE THEM ALL

Having different kind of projects create the necessity to rationalize languages used. Due to its versatility, Python became the main language. It already had a good reputation in many domains like web development (Flask & Django frameworks), dataviz (matplotlib) or datamining (scipy, mongoengine). Thanks to Obspy library, it becames even good for seismology tasks. This library makes our lifes easier when it comes to manipulate files like GSE2 or QuakeML or to acquire and processing waveforms (arclink, miniseed supports).

PUPPET MASTER

We also evolved the way we deploy application by creating a devops friendly environment based on Linux Container (LXC), Puppet and Fabric. Having a homogenous operating system environment permits the usage of containers (LXC) instead of virtualization (KVM). These containers are not longer directly manipulated, they are using Puppet recipes that desbribes procedures to install services.
Internal application are deployed using a Python library called Fabric, it allows to describe installation and update procedures in a single Python file. So a developper is able to control how his application should be deployed.

WORK IN PROGRESS

We are working on improving our seismic catalog quality and to add missing features on our website (waveforms plotting, static maps). Another work is to consolidate data discremination in order to provide them to the incoming RESIF web portal.

AFFILIATION

BCSF-ReNaSS, UMS 830
Université de Strasbourg
5 rue René Descartes, F-67084 Strasbourg Cedex, France