OAK RIDGE National Laboratory

Web monitor: installation and maintenance

Mathieu Doucet

Oak Ridge National Laboratory

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Plan for the next few weeks

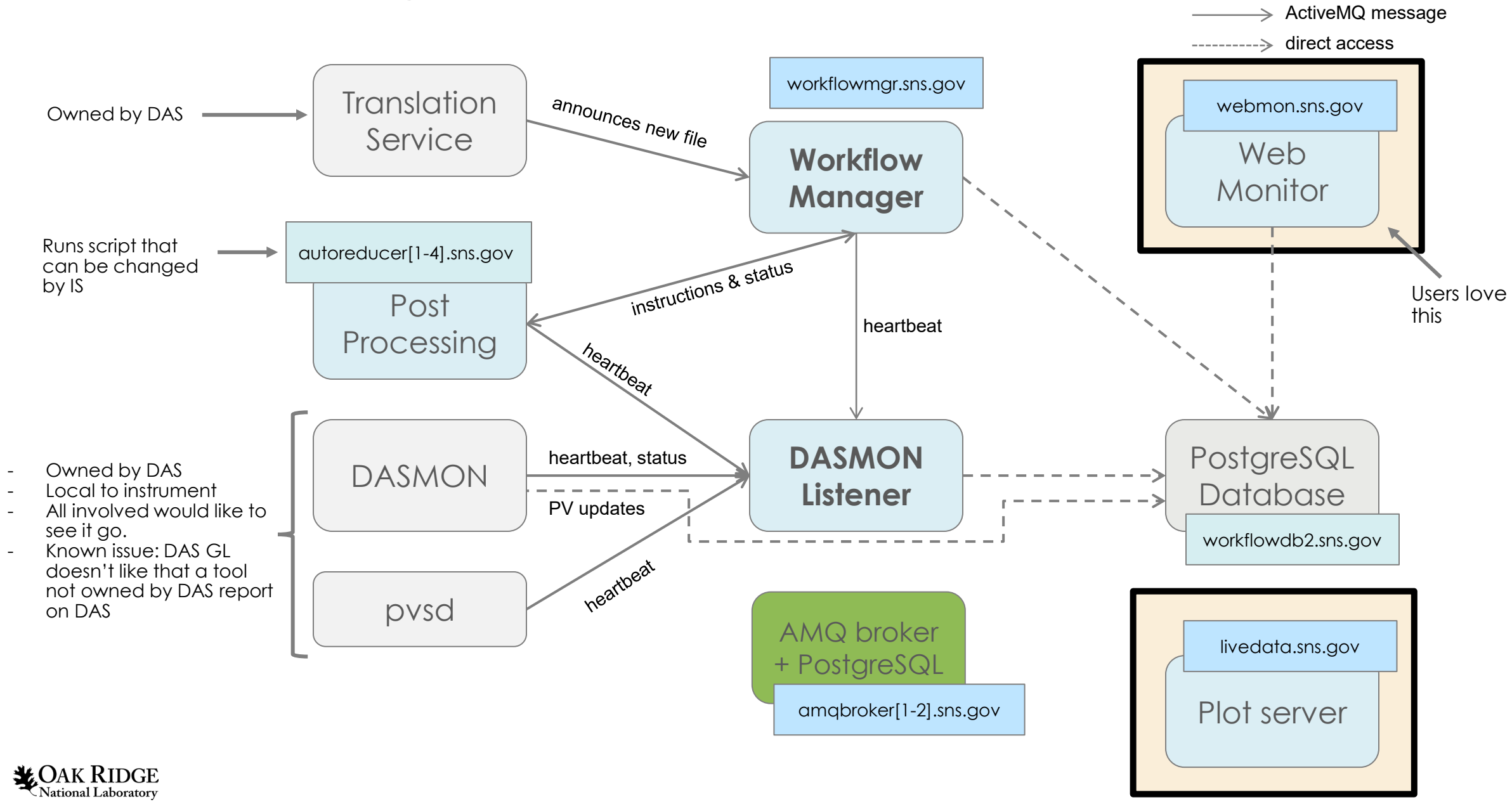
Test environment:

New RHEL8 machines are being set up so we can install them together

Topics to cover:

1. General overview
2. Workflow manager and DASMON listener – Installation & maintenance
3. **Web monitor – Installation and maintenance [this presentation]**
4. Autoreduction service – Installation and maintenance
5. Autoreduction setup through webmon – how-to and future vision
6. The IHC call – when things go wrong & recovery strategies
7. Vision for the future – what I would do differently

Post-Processing Architecture

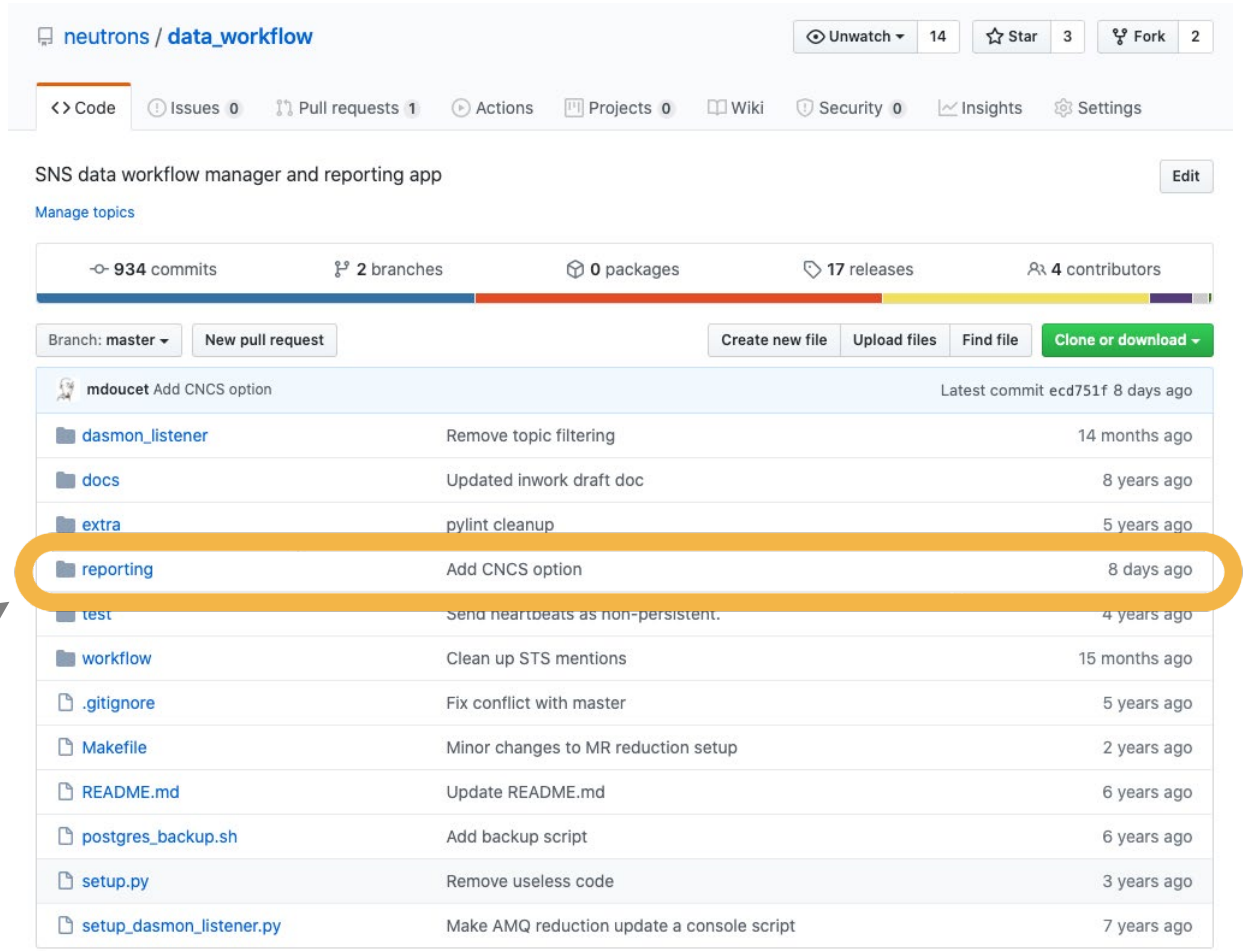


Installation

https://github.com/neutrons/data_workflow

- The web monitor offers a view on the workflow manager's data. It shares a database with the workflow manager.
- The dasmon listener and DAS also write instrument information to the same DB so the monitor can report instrument status.
- The database is hosted on `workflowdb2.sns.gov`
- The monitor is also used to set up AR and reprocess runs. For this reason it also connects to the AMQ brokers to submit requests.
- … more on the AR configuration in a couple of weeks.

Web monitor application code



Web monitor configuration

local_settings.py

```
from django_auth_ldap.config import LDAPSearch, PosixGroupType
import ldap

[ database info here ]


[ LDAP info here ]

ALLOW_GUESTS = True
GRAVATAR_URL = "https://www.gravatar.com/avatar/"
ALLOWED_DOMAIN = ('ornl.gov','sns.gov')

LIVE_PLOT_SECRET_KEY=”xxx"
FITTING_URLS = {'ref_l': 'https://reflectivity.sns.gov/fit/ref_l/$run_number'}

CATALOG_URL = 'https://oncat.ornl.gov'
CATALOG_ID = ”xxxxx"
CATALOG_SECRET = ”xxxxxx"

FACILITY_INFO = {'hb2c': 'HFIR','cg1d': 'HFIR', 'hb2a': 'HFIR', 'hb2b': 'HFIR',
'hb3a': 'HFIR', 'cg2': 'HFIR', 'cg3': 'HFIR’}

[ AMQ info here ]
```

The DB settings are shared between the workflow app and the monitor app.

A number of places are available to write configs.

You could just write a local_settings.py file.

`data_workflow/reporting/reporting_app/settings.py`

imports

`data_workflow/workflow/database/settings.py`

imports (optional, not in repo)

`data_workflow/workflow/local_settings.py`

Deploying the web monitor

- Runs on `webmon.sns.gov`. A new test node is available on `webmondev.sns.gov`
- `sudo /usr/sbin/deploy-webmonitor [path to code]`
- Once deployed, you just need to restart apache
- Logs are in /var/log/httpd

Installation for a bare machine:

```
> sudo yum install mod_wsgi (with python 2.7)
> sudo chown -R apache /var/www

- Install easy_install
> wget https://bootstrap.pypa.io/ez_setup.py -O - | sudo python
> sudo easy_install django==1.7
> sudo easy_install django_auth_ldap
> sudo yum install python-devel

- postgresql installation
> sudo yum install postgresql postgresql-devel postgresql-server
> sudo yum install postgresql-libs postgresql-contrib
> sudo easy_install psycopg2

- Make sure port 5432 on workflowdb.sns.gov is reachable from webmon.sns.gov

- Modify reporting/apache/apache_django_wsgi.conf as needed, then copy it to /etc/httpd/conf.d

> cd /var/www/workflow/app; python manage.py createcachetable webcache
```

data_workflow/reporting/apache/apache_django_wsgi.conf

```
Alias /static/ /var/www/workflow/static/

<Directory /var/www/workflow/static>
  Order deny,allow
  Allow from all
</Directory>

WSGIScriptAlias /workflow /var/www/workflow/app/reporting_app/reporting.wsgi
WSGIPythonPath /var/www/workflow/app

<Directory /var/www/workflow/app/reporting_app>
   Order allow,deny
   Allow from all
</Directory>
```

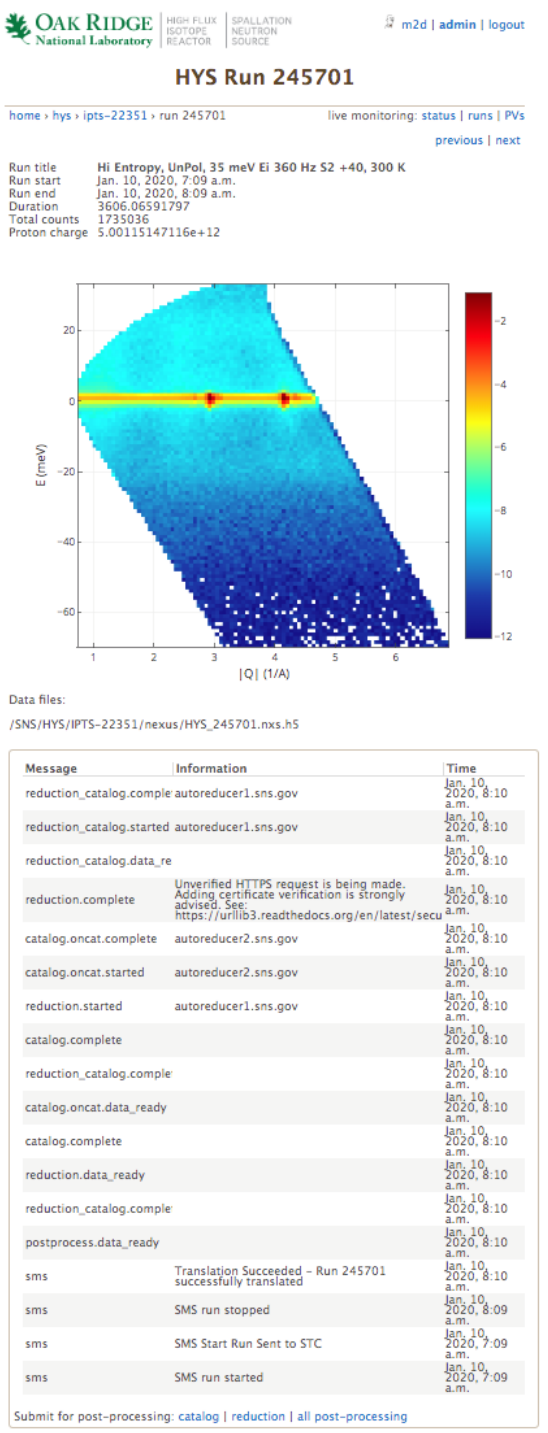
Web monitor features

Info from ONCat

HTML from livedata.sns.gov

AR workflow AMQ log

Some tasks can be requested [sends AMQ message]



livedata.sns.gov

https://github.com/neutrons/live_data_server

- Runs on `livedata.sns.gov`. A new test node is available on `livedatadev.sns.gov`
- This application is only used to store/read html data in a table. It doesn't hold critical data.
- Read access is done by comparing hash with a secret key.
- Same deal with the local_settings, but it's mostly empty.
- There's a postgres DB running but not accessible from outside the machine.
- Same type of configuration as the web monitor, but just run `make install`
- Once deployed, you just need to restart apache

Installation for a bare machine:

```
- postgresql installation
  > sudo yum install postgresql postgresql-devel postgresql-server
  > sudo yum install postgresql-libs postgresql-contrib
  > sudo yum install pgadmin3

- postgres setup, performed as postgres user
  > sudo su - postgres
  > initdb
  > pg_ctl -D /var/lib/pgsql/data -l /var/lib/pgsql/server.log start
  > createuser livedata -W [pwd = XXX]
  > createdb -O livedata livedata_db
  > exit

- install django [prod version runs 1.9.6]
> sudo yum install mod_wsgi
> sudo yum install mod_ssl
> sudo chown -R apache /var/www
> wget https://bootstrap.pypa.io/get-pip.py -O - | sudo python
> wget https://bootstrap.pypa.io/ez_setup.py -O - | sudo python
> sudo pip install django
> sudo yum install python-devel


> sudo pip django_auth_ldap

> sudo pip install psycopg2

# https://github.com/ottoyiu/django-cors-headers
> sudo pip install django-cors-headers
> sudo chmod a+r /var/log/httpd
> sudo chmod a+x /var/log/httpd

> Create user livedata to be used by auto reduction to post data
```