Update on Development and Deployment of Django Backend for Rucio OpInt

Cedric Sefron
Mario Lassnig
Dilaksun Bavarjan
Mayank Sharma
Panos Paparrigopoulos
Boris Vasilev
Alessandro Di Girolamo
Rucio OpInt: what are we talking about

- **Data Management Operations**
  - In ATLAS during the past several years there are at least 3 Full Time Equivalent (FTE) dedicated to DDM Ops.
  - ....Plus several shifters monitoring the distributed computing infrastructure and ticketing sites in case of obvious failures.
  - We know a lot can be automated.

- **What we want to do:**
  - Details in 10th June Dilaksun and Cedric work in [https://indico.cern.ch/event/818950/](https://indico.cern.ch/event/818950/) and last month tech discussion [https://indico.cern.ch/event/838714/](https://indico.cern.ch/event/838714/)
  - Basic idea: take the FTS (asynchronous transfers) success and errors, categorize them, find correlation, provide "suggestions" to "someone" on what to do.
    - Tool in place, send mattermost messages on sites failing w/o GGUS ticket already open, i.e. suggest shifters to submit ticket. Feedback on action taken yet to be recorded (learning)
  - If we build the "right" system, we have tremendous possibilities of improvements:
    - E.g. extending to job errors
Moving to Django

Last discussion: https://docs.google.com/document/d/1Qt3twQ6xzpuLUlgLzKlrl8vxRy-uYhерьNdJOV0/edit

Out of the box functionality:

- Django admin
- Django is designed to protect against common threats such as XSS attacks and code injections.
- Django ORM requires zero config to begin, easy to manage migrations, db queries etc.

Django-REST-framework provides easy way to build a REST API which is exactly what we need.
Current OpenShift+OpenStack Deployment

**URL:** [http://rucio-opint.web.cern.ch/](http://rucio-opint.web.cern.ch/)

**OpenStack**

rucio_loader_cron : to fetch logs from influxdb/elasticsearch and update the production database.

**OpenShift**

Django backend ported by Panos that reads the production database and exposes it to the frontend.
Issues with Current Deployment

Sensitive information was available in a Git private repo (DB keys, ElasticSearch API_KEY)

Cron for fetching monitoring logs did not run inside OpenShift Pods.

Pods run Django’s development server.

- WSGI applications are recommended to be run in production.
- Gunicorn enables to do that.

Our custom entrypoint did not run in production. Manual steps were needed to reproduce the expected state of the pods.
Long term maintainability Goals

Development environment

1. Having an easy to setup development environment to make the involvement of more developers to the project easier.
2. Provide a postgres db backend locally and populate it with a sample of real data.

Deployment/ Operational Overhead

1. Minimizing operational work to make maintainability easier.
2. Easily reproducible deployment. If needed can be migrated easily to another platform.
3. Connect to backend to the DB on Demand production db easily
Deployment Alternatives

OpenStack IaaS + Nginx + Gunicorn + Django
- Deploying to OpenStack will require us to build a VM and configure a lot of things ourselves (firewalld, load-balancing etc.).
- Requires people to be involved to maintain the servers if and when things go wrong.

OpenShift PaaS + Gunicorn + Django
- OpenShift uses Kubernetes inside which means we don’t have to worry about the load-balancing.
- Easy to maintain development and production environment. Just write 2 Dockerfiles.
Last few weeks..

Resolve issues with current deployment.

Implement the long term maintainability goals.

OpenShift was picked as the deployment platform.

Get comfortable with Okd (OpenShift’s community distribution for Kubernetes)
Analysis of Current Deployment

**Dockerfile**: sets up Django, requirements, mounts code directory

**Entrypoint**: makes migrations, set environment variables, configuring and starting cron

The Dockerfile and entrypoint we created in our Git repository were ignored in OpenShift deployment.

- The [python template available](#) in the OpenShift console uses its own Dockerfile (Builder image) and entrypoint script (also called *s2i scripts*) which takes care of running management commands for collecting static files, making migrations and starting the server.
The Solution

URL:  http://oc-opint-test.web.cern.ch/

Public Git repo independent of secrets:  https://github.com/maany/rucio-opint-django-backend

Setting up Dev Environment:

- Clone the repository
- docker-compose up --build (sets up django and postgres containers)

Pushing code to Production:

- Push code to production branch (at present, it is master. I'll change it after the meeting)
- Setting up Okd takes care of making all secrets available to the containers and connecting to the production db.

Cron inside containers:

- The containers use supercronic: https://github.com/aptible/supercronic
OpenShift Pipeline

Source-to-Image (S2) Developer Workflow
Current OpenShift Pipeline

Generation of images for application containers

Builder Images (Okd template) + Source Code (GitHub) = Image for container to be deployed

Customization Options

- **Assemble script**: Executed by OpenShift’s s2i tool to inject custom configuration while building an image for the containers to be deployed for a project
- **Run script**: Executed by OpenShift when starting the containers
OpenShift Pipeline

Source-to-Image (S2) Developer Workflow

Custom builder image with custom scripts for assemble and run operations
Current OpenShift Pipeline

URL: http://oc-opint-test.web.cern.ch/

Custom Builder Image: https://github.com/maany/rucio-opint-django-s2i-test/blob/master/Dockerfile

Custom Template: https://github.com/maany/rucio-opint-django-s2i-test/blob/master/mytemplate.yml

Assemble: https://github.com/maany/rucio-opint-django-s2i-test/blob/master/s2i/bin/assemble

Run: https://github.com/maany/rucio-opint-django-s2i-test/blob/master/s2i/bin/run
What’s left?

- Need to update Readme.md for developers.
- Add a production branch for the Git repo for django backend + Update the webhook in Okd.
- Add Instructions for initial deployment using s2i and command line.
- Parameterize the template for Okd so custom deployments can be made by anyone.
- Rollout this repository to the current production deployment.
- Incorporate the frontend in the same Okd project as the backend
Questions?