
CERN

DB On Demand 2.0 (alpha)

Ignacio Coterillo, Ruben Gaspar Aparicio
mailto: ignacio.coterillo@cern.ch

28/09/2015

Table of Contents

Introduction

Service evolution
- Service evolution
- DB On Demand statistics

Current architecture
- Simplified architecture graph

Instance management
- The DBOD-core framework
- Objectives
- Generics
- Codebase comparison
- Hosting on GitHub

Testing
- Unit testing
- Functional testing
- Why don't we use Jenkins?

New Architecture
- Simplified New architecture
- Generics
- DBOD WS API

Conclusions

From this talk submission

- The presentation would cover the **ongoing** efforts and struggles we are facing in our attempt to modernize an existing codebase with large portions of legacy code in the Database on Demand project
- Some characteristics of our team that can probably be common around CERN:
  - Small team
  - Easy to have experts owning sections of code
  - Lots of code was written by eventual people not longer around

Table of Contents

Introduction

Service evolution
- Service evolution
- DB On Demand statistics

Current architecture
- Simplified architecture graph

Instance management
- The DBOD-core framework
- Objectives
- Generics
- Codebase comparison
- Hosting on GitHub

Testing
- Unit testing
- Functional testing
- Why don't we use Jenkins?

New Architecture
- Simplified New architecture
- Generics
- DBOD WS API

Conclusions

Service origins

- The service was originally specified with a single simple objective:
  To manage **single instance MySQL databases**
- The original implementation tried to make use of as many pre-existent components as possible:
  - Good: Easier and faster to kick-start the project
  - Bad: Inheriting technical debt
- Service requirements continued to change over time

Service evolution

- **2011 Q?** IT-DB hosts MySQL databases for the Drupal service
- **2011 Q4**: Started operations as a service intended for managing single instance **MySQL** databases
- **2012 Q2**: Added **Oracle** instances
- **2013 Q2**: Added **HA Cluster** instances
- **2013 Q4**: Added **PostgreSQL** instances
- ... Infrastructure migration (puppetization)
- **2015 Q4**: **DB cloning**
- **2016 Q?**: Replication? New DB types?

DB On Demand instances evolution

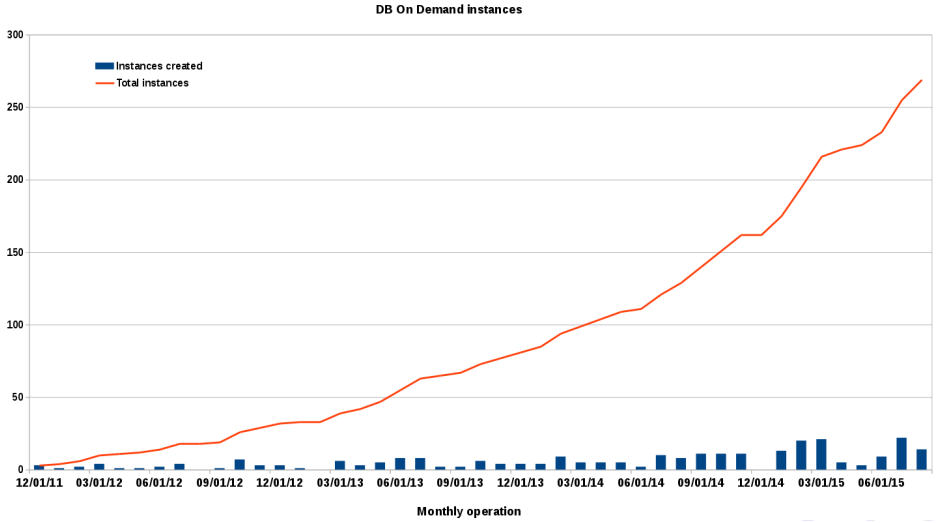


Table of Contents

Introduction

Service evolution
- Service evolution
- DB On Demand statistics

Current architecture
- Simplified architecture graph

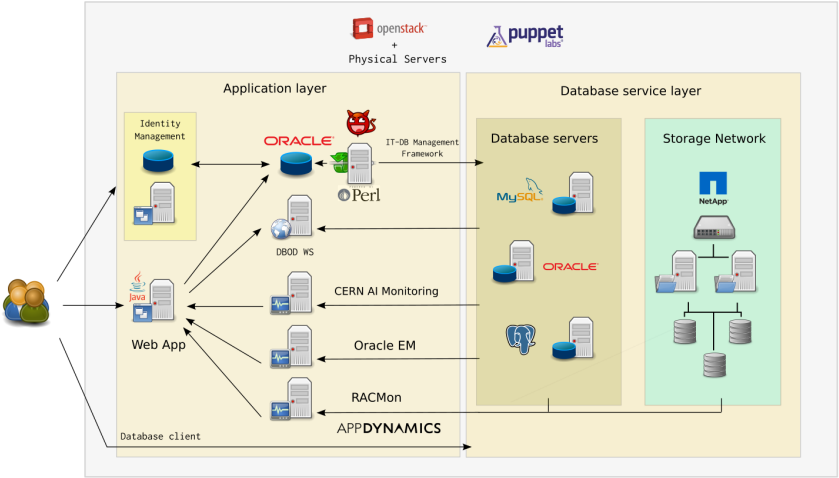
Instance management
- The DBOD-core framework
- Objectives
- Generics
- Codebase comparison
- Hosting on GitHub

Testing
- Unit testing
- Functional testing
- Why don't we use Jenkins?

New Architecture
- Simplified New architecture
- Generics
- DBOD WS API

Conclusions

Simplified architecture



Simplified architecture graph

Explained

- A Java web application exposes a high level managing interface for the service users/instance owners
- A daemon process continuosly polls the service database scouting for new jobs to execute using Syscontrol, an IT-DB internally developed management framework
- At the lowest level operations are executed by the **instance management** code which is distributed as part of the Syscontrol framework.

Problem

- Data is fragmented
- Data accesses needs to be duplicated in several places/languages

Table of Contents

Introduction

Service evolution

- Service evolution
- DB On Demand statistics

Current architecture

- Simplified architecture graph

Instance management

- The DBOD-core framework
- Objectives
- Generics
- Codebase comparison
- Hosting on GitHub

Testing

- Unit testing
- Functional testing
- Why don't we use Jenkins?

New Architecture

- Simplified New architecture
- Generics
- DBOD WS API

Conclusions

Instance management

Towards the DBOD-core framework

Legacy code

- Our managing code re-used heavily from old IT-DB projects librairies. Specially from the **Backup&Recovery** project and the original **Drupal** databases managing code.
- The codebase is a continuosly growing collection of Perl scripts with:
  - Almost duplicated configuration files
  - Fragmented configuration source

Problem

Specially hard to add support for models deviating from the single-instance concept such as **HA Clusters, Master/slave replication sets, DB Clones** or **Sharded** databases

Instance management

Towards the DBOD-core framework

DBOD-core framework

- Consolidates code from previously 3rd party libraries
- Consolidates all static configuration in one single file
- Consolidates access to all dynamic parameters using the DBOD API
- Modernize methods (e.g: operations logging)

Instance management

Towards the DBOD-core framework

Updating the codebase

- Targeting a more up to date Perl distribution
  - We distribute a full Perl 5.20.2 environment created and managed with **Perlbrew**, as an RPM package
- Modern Perl: Using **Perl::Moose** as a foundation

- Static code analysis and standards with **Perlcritic**

Instance management

Codebase comparison

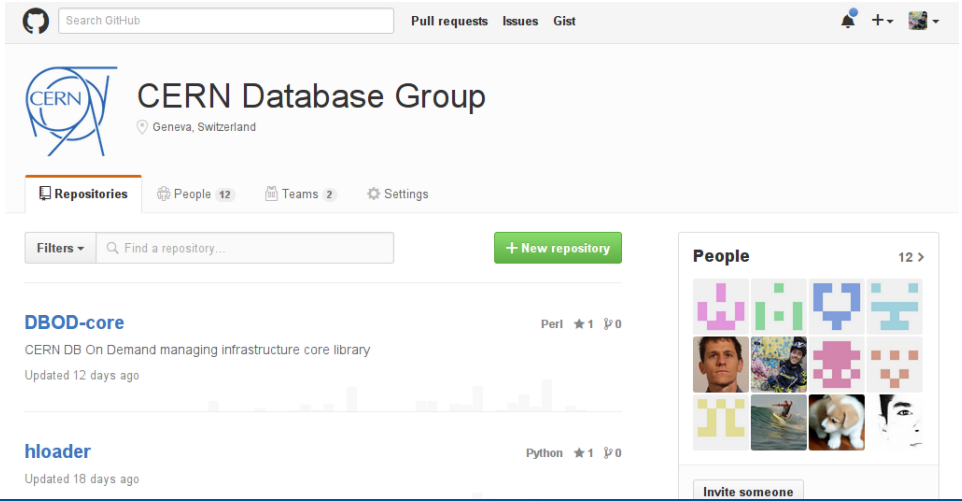
Current implementation

- 88 Perl files $\sim$ 15 KLOC
- 19 Bash scripts $\sim$ 1.2 KLOC
- 58 Support files: configuration, templates, SQL

DBOD-core

- 44 Perl files $\sim$ 6 KLOC (80% functionality implemented)
- 24 Support files, now Puppet managed

Hosted on GitHub



Why GitHub?

- We wanted to make the project OSS
  - Following Transfer Technology Office indications (Licensing, Copyright, etc)
- GitHub is de facto standard hosting solution for OSS
- Better visibility for the project
  - Extra value for contributors (Technical students, PJAS's, Fellows,...)
- Easier to collaborate with entities external to CERN

Table of Contents

Introduction

Service evolution
- Service evolution
- DB On Demand statistics

Current architecture
- Simplified architecture graph

Instance management
- The DBOD-core framework
- Objectives
- Generics
- Codebase comparison
- Hosting on GitHub

Testing
- Unit testing
- Functional testing
- Why don't we use Jenkins?

New Architecture
- Simplified New architecture
- Generics
- DBOD WS API

Conclusions

Testing

At the moment

All testing is manual!

...

Testing

Unit testing

- Helpful during development
- Problematic to mock DB operations
- Travis-CI free for GitHub based OSS projects

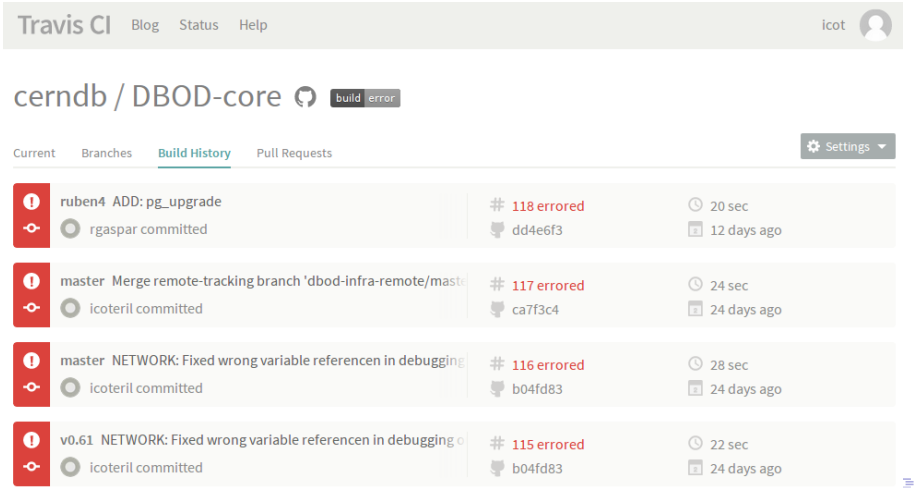
Functional testing

- Our priority
- A VM hosts sample DB instances ef each type of our offered databases
- We use **Rundeck** to run sets of our infrastructure operations on these test instances

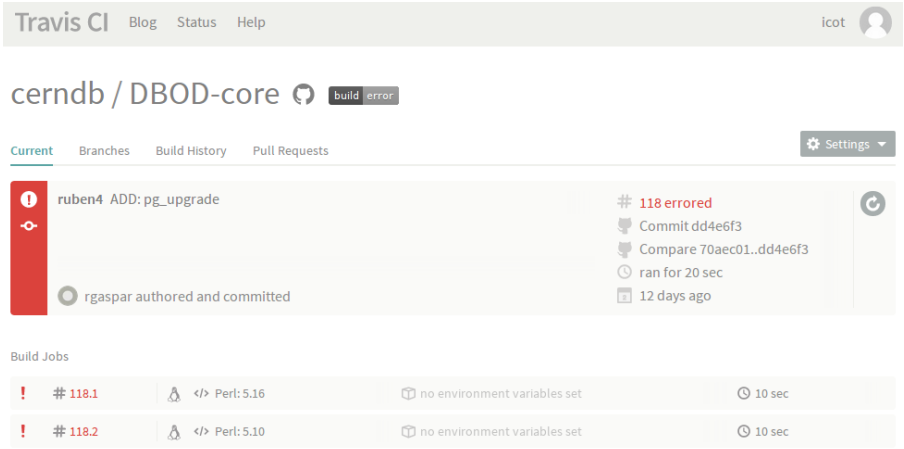
Unit testing: Travis CI

```
 1 [No Name]  1 w/c/d/.travis.yml
language: perl
perl:
    - "5.16"
    - "5.10"

before_script:
    - cpanm Config::General
    - cpanm -n Devel::Cover::Report::Coveralls
    - cpanm -n JSON
    - cpanm -n REST::Client
    - cpanm -n MIME::Base64

after_script:
    - cover -test -report coveralls

~
~
```

Unit testing: Travis CI



Unit testing: Travis CI

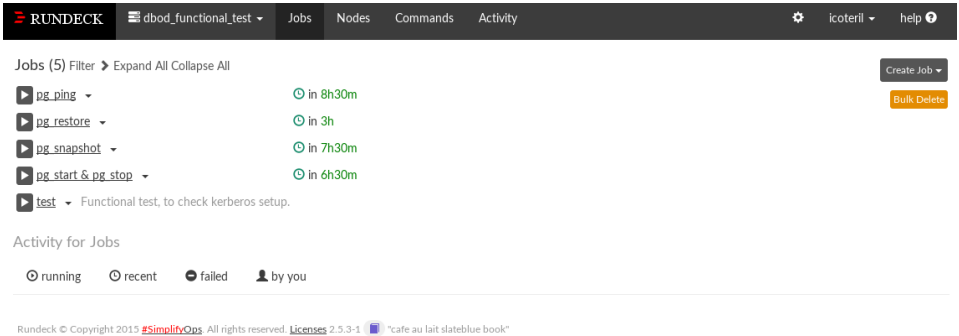


Functional testing: Rundeck

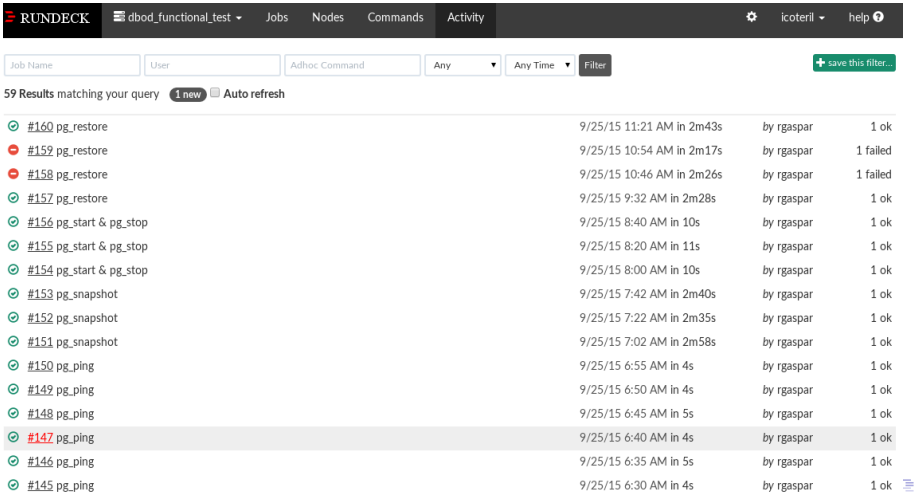
Rundeck

- Self defined as: **Job Scheduler and Runbook automation**
- Allows to define simple or complex workflows of commands
- The commands can run on local or remote nodes
- They can be triggered by its scheduler or on-demand via user iterface or API

Functional testing: Rundeck



Functional testing: Rundeck



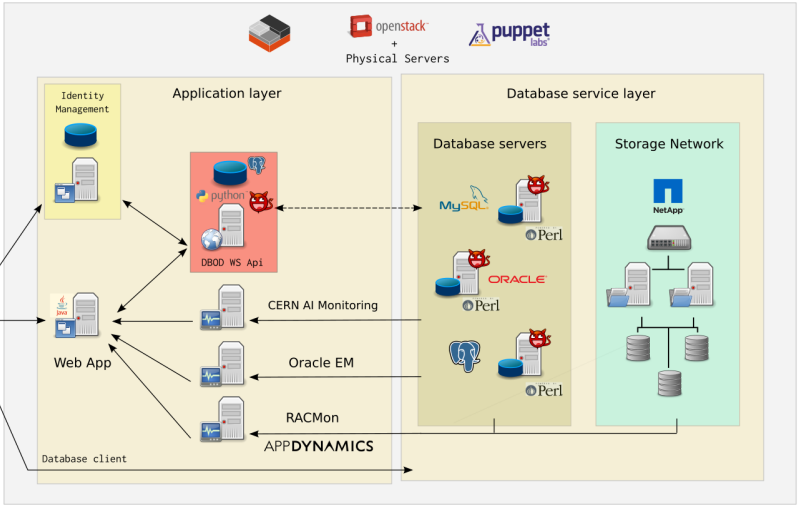
Why don't we use Jenkins?

- Jenkins was not offered centrally at CERN at the moment we started looking for testing/CI solutions
- Travis CI free and easy to set up GitHub hosted projects (one configuration file and you are set)
- For our use case, we found Jenkins to be cumbersome compared to Rundeck
  - SSO integration creates problems for fetching job results from remote hosts.
- We plan to use Rundeck for other tasks within the service, so we kill two birds with the same stone

Table of Contents

- Introduction
- Service evolution
  - Service evolution
  - DB On Demand statistics
- Current architecture
  - Simplified architecture graph
- Instance management
  - The DBOD-core framework
  - Objectives
  - Generics
  - Codebase comparison
  - Hosting on GitHub
- Testing
  - Unit testing
  - Functional testing
  - Why don't we use Jenkins?
- New Architecture
  - Simplified New architecture
  - Generics
  - DBOD WS API
- Conclusions

Simplified architecture



New architecture generics

New Database backend

- The database of the service is going to be migrated from Oracle to PostgreSQL (9.4.4)

Rundeck

- We plan to use Rundeck to manage execution of scheduled jobs (before managed with scheduled jobs on the Oracle database)
- Useful also for instance deletion and cleanup

DBOD WS API

Generics

- A single point of access to the service data for all components
- REST interface
- Developed in Python (using on Tornado)
- Interfaces to:
  - The service backend database
  - The Syscontrol LDAP directory and API

DBOD WS API

As of now

Currently Working

- Networking: Registration of IP-aliases/DNS name pairs
- Instance metadata: Registry of each instance configuration parameters.

Next step

Improved FIM integration

DBOD WS API

Also on GitHub

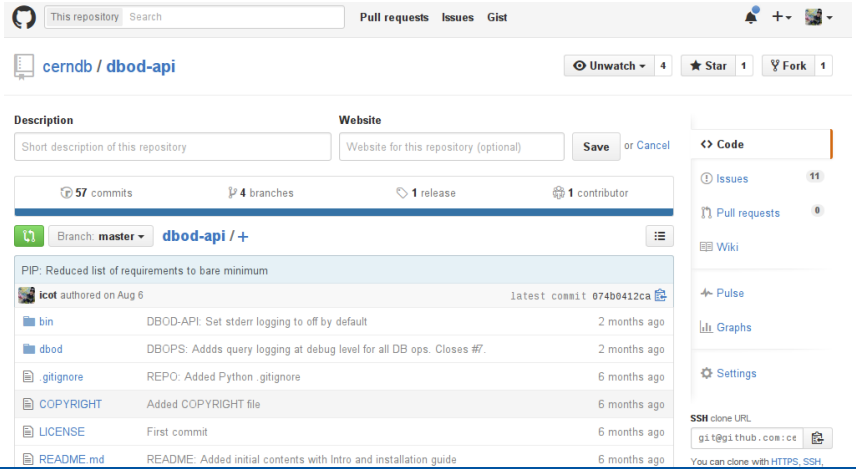


Table of Contents

Introduction

Service evolution
- Service evolution
- DB On Demand statistics

Current architecture
- Simplified architecture graph

Instance management
- The DBOD-core framework
- Objectives
- Generics
- Codebase comparison
- Hosting on GitHub

Testing
- Unit testing
- Functional testing
- Why don't we use Jenkins?

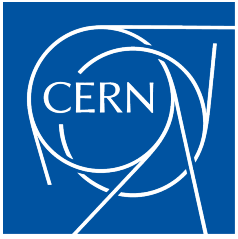
New Architecture
- Simplified New architecture
- Generics
- DBOD WS API

Conclusions

TL;DR

- We are re-implementing core pieces of our infrastructure so we can easily add new functionality in order to provide a better service to the community
  - Introducing coding standards
  - Static Analysis
  - Unit and functional testing with Travis-CI and Rundeck
- Open sourcing infrastructure components

Questions?


CERN
www.cern.ch