



# DB On Demand 2.0 (alpha)

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

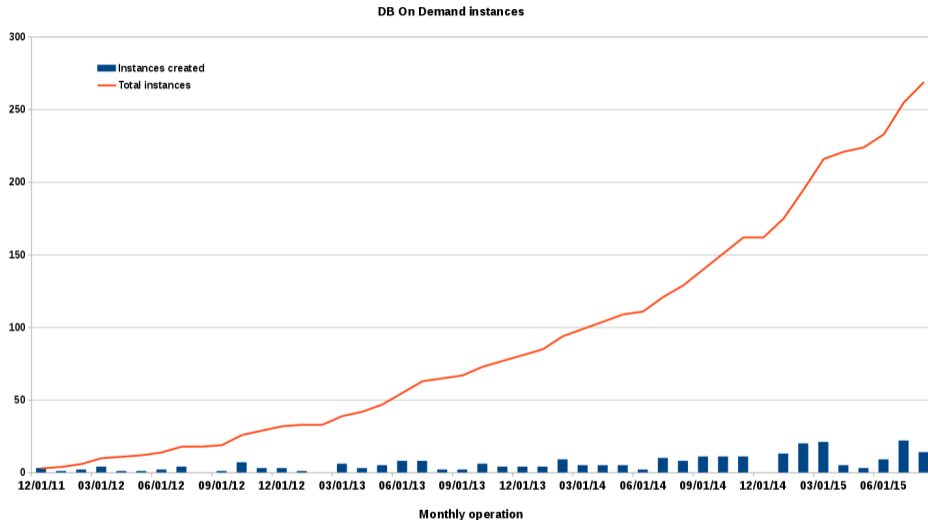
# Service origins

- ▶ 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

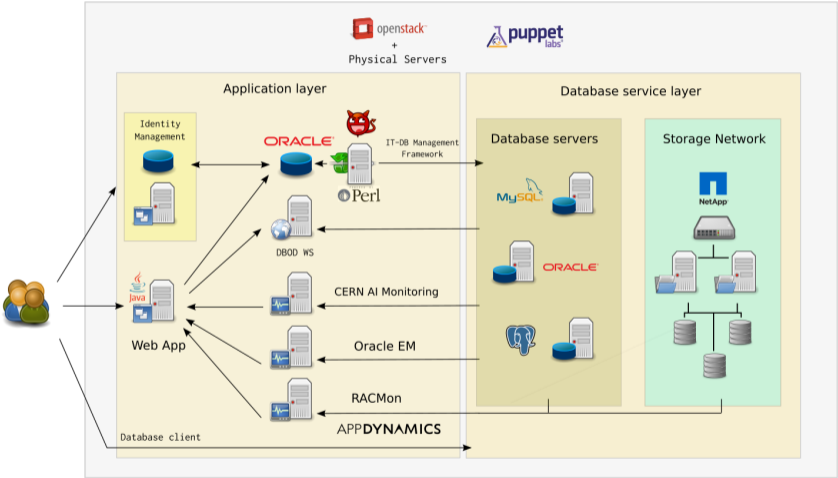
Simplified New architecture

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 continuously 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 libraries. Specially from the **Backup&Recovery** project and the original **Drupal** databases managing code.
- ▶ The codebase is a continuously 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

## Codebase comparison

### Current implementation

- ▶ 88 Perl files ~ 15 KLOC
- ▶ 19 Bash scripts ~ 1.2 KLOC
- ▶ 58 Support files: configuration, templates, SQL

### DBOD-core

- ▶ 44 Perl files ~ 6 KLOC (80% functionality implemented)
- ▶ 24 Support files, now Puppet managed



# Hosted on GitHub

The screenshot shows the GitHub profile page for the CERN Database Group. At the top, there is a search bar with the GitHub logo and the text "Search GitHub". To the right of the search bar are links for "Pull requests", "Issues", and "Gist". Further right are notification, user, and organization icons. The main header features the CERN logo and the text "CERN Database Group" with a location pin indicating "Geneva, Switzerland". Below this is a navigation bar with "Repositories", "People 12", "Teams 2", and "Settings". A search bar for repositories is present with a "Filters" dropdown and a "+ New repository" button. The repository list shows "DBOD-core" (Perl, 1 star, 0 forks, updated 12 days ago) and "hloader" (Python, 1 star, 0 forks, updated 18 days ago). A "People" section on the right shows a grid of 12 avatars, with the first one being a real person's photo. At the bottom right of the profile area are icons for a menu, refresh, search, and share, along with an "Invite someone" button.



# 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 of 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

Travis CI [Blog](#) [Status](#) [Help](#) icot 


cerndb / DBOD-core  build error


Current [Branches](#) [Build History](#) [Pull Requests](#) Settings 


 	ruben4 ADD: pg_upgrade  rgaspar committed	# 118 errored  dd4e6f3	🕒 20 sec 📅 12 days ago
 	master Merge remote-tracking branch 'dbod-infra-remote/maste  icoteril committed	# 117 errored  ca7f3c4	🕒 24 sec 📅 24 days ago
 	master NETWORK: Fixed wrong variable referencen in debugging  icoteril committed	# 116 errored  b04fd83	🕒 28 sec 📅 24 days ago
 	v0.61 NETWORK: Fixed wrong variable referencen in debugging  icoteril committed	# 115 errored  b04fd83	🕒 22 sec 📅 24 days ago



    


# Unit testing: Travis CI


Travis CI [Blog](#) [Status](#) [Help](#) icot 


cerndb / DBOD-core  build error


[Current](#) [Branches](#) [Build History](#) [Pull Requests](#) Settings 


 ruben4 ADD: pg\_upgrade # 118 errored 

 Commit dd4e6f3









 Compare 70aec01..dd4e6f3

 ran for 20 sec

 12 days ago

 rgaspar authored and committed

Build Jobs

 # 118.1	 </> Perl: 5.16	 no environment variables set	 10 sec
 # 118.2	 </> Perl: 5.10	 no environment variables set	 10 sec



# 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 interface or API

# Functional testing: Rundeck

The screenshot shows the Rundeck web interface. At the top is a dark navigation bar with the 'RUNDECK' logo, a dropdown menu for 'dbod\_functional\_test', and tabs for 'Jobs', 'Nodes', 'Commands', and 'Activity'. On the right side of the bar are icons for settings, a user profile 'icoteril', and a help icon.

Below the navigation bar, the 'Jobs' section is active, displaying 'Jobs (5) Filter > Expand All Collapse All'. On the right side of this section are two buttons: 'Create Job' and 'Bulk Delete'.

The job list contains five entries:

- `pg_ping` - in 8h30m
- `pg_restore` - in 3h
- `pg_snapshot` - in 7h30m
- `pg_start & pg_stop` - in 6h30m
- `test` - Functional test, to check kerberos setup.

Below the job list is the 'Activity for Jobs' section, which includes filters for 'running', 'recent', 'failed', and 'by you'.

At the bottom of the interface, a footer line reads: 'Rundeck © Copyright 2015 #SimplifyOps. All rights reserved. Licenses 2.5.3-1 "cafe au lait slateblue book"'. On the right side of the footer are several small navigation icons.

# Functional testing: Rundeck

The screenshot shows the Rundeck web interface. At the top, there is a navigation bar with the Rundeck logo, a dropdown menu for 'dbod\_functional\_test', and tabs for 'Jobs', 'Nodes', 'Commands', and 'Activity'. On the right side of the navigation bar, there are icons for settings, a user dropdown 'icoteril', and a help icon.

Below the navigation bar, there is a search and filter section with input fields for 'Job Name', 'User', and 'Adhoc Command', dropdown menus for 'Any' and 'Any Time', and a 'Filter' button. A green button on the right says '+ save this filter...'. Below this, it says '59 Results matching your query' with a '1 new' badge and an 'Auto refresh' checkbox.

The main content area displays a table of job execution results. The table has columns for job ID, status, execution time, user, and result. The job '#147 pg\_ping' is highlighted in grey.

Job ID	Status	Time	User	Result
#160 pg_restore	ok	9/25/15 11:21 AM in 2m43s	by rgaspar	1 ok
#159 pg_restore	failed	9/25/15 10:54 AM in 2m17s	by rgaspar	1 failed
#158 pg_restore	failed	9/25/15 10:46 AM in 2m26s	by rgaspar	1 failed
#157 pg_restore	ok	9/25/15 9:32 AM in 2m28s	by rgaspar	1 ok
#156 pg_start & pg_stop	ok	9/25/15 8:40 AM in 10s	by rgaspar	1 ok
#155 pg_start & pg_stop	ok	9/25/15 8:20 AM in 11s	by rgaspar	1 ok
#154 pg_start & pg_stop	ok	9/25/15 8:00 AM in 10s	by rgaspar	1 ok
#153 pg_snapshot	ok	9/25/15 7:42 AM in 2m40s	by rgaspar	1 ok
#152 pg_snapshot	ok	9/25/15 7:22 AM in 2m35s	by rgaspar	1 ok
#151 pg_snapshot	ok	9/25/15 7:02 AM in 2m58s	by rgaspar	1 ok
#150 pg_ping	ok	9/25/15 6:55 AM in 4s	by rgaspar	1 ok
#149 pg_ping	ok	9/25/15 6:50 AM in 4s	by rgaspar	1 ok
#148 pg_ping	ok	9/25/15 6:45 AM in 5s	by rgaspar	1 ok
#147 pg_ping	ok	9/25/15 6:40 AM in 4s	by rgaspar	1 ok
#146 pg_ping	ok	9/25/15 6:35 AM in 5s	by rgaspar	1 ok
#145 pg_ping	ok	9/25/15 6:30 AM in 4s	by rgaspar	1 ok

# 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

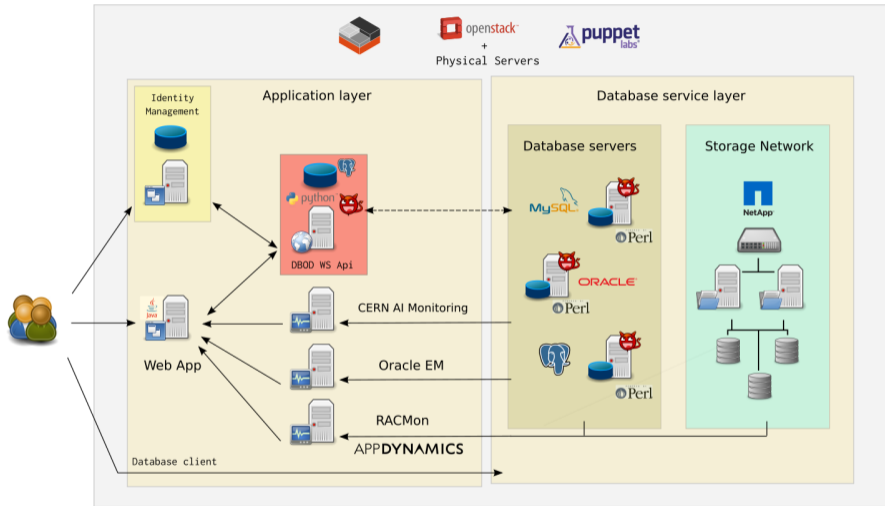
Simplified New architecture

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

The screenshot shows the GitHub interface for the repository 'cerndb/dbod-api'. At the top, there is a search bar for the repository, navigation links for 'Pull requests', 'Issues', and 'Gist', and user profile icons. Below the repository name, there are buttons for 'Unwatch' (4), 'Star' (1), and 'Fork' (1). The main content area is divided into 'Description' and 'Website' sections, both with input fields and 'Save' or 'Cancel' buttons. Below these are statistics: 57 commits, 4 branches, 1 release, and 1 contributor. A file browser shows the current branch 'master' with a '+' icon to add files. A list of files is displayed, including 'bin', 'dbod', '.gitignore', 'COPYRIGHT', 'LICENSE', and 'README.md', each with a brief description and the time since the last commit. On the right side, there is a sidebar with links for 'Code', 'Issues' (11), 'Pull requests' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. At the bottom of the sidebar, the 'SSH clone URL' is provided as 'git@github.com:ce'.

This repository Search

Pull requests Issues Gist

cerndb / dbod-api

Unwatch 4 Star 1 Fork 1

**Description** Website

Short description of this repository Website for this repository (optional) Save or Cancel

57 commits 4 branches 1 release 1 contributor

Branch: master dbod-api / +

PIP: Reduced list of requirements to bare minimum

icot authored on Aug 6 latest commit 074b0412ca

bin	DBOD-API: Set stderr logging to off by default	2 months ago
dbod	DBOPS: Adds query logging at debug level for all DB ops. Closes #7.	2 months ago
.gitignore	REPO: Added Python .gitignore	6 months ago
COPYRIGHT	Added COPYRIGHT file	6 months ago
LICENSE	First commit	6 months ago
README.md	README: Added initial contents with Intro and installation guide	6 months ago

Code

Issues 11

Pull requests 0

Wiki

Pulse

Graphs

Settings

SSH clone URL

git@github.com:ce

You can clone with HTTPS, SSH,



# 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?



[www.cern.ch](http://www.cern.ch)