



GENERIC CROSS PLATFORM  
SOLUTION FOR GRID TOOLS

CHEP 2019

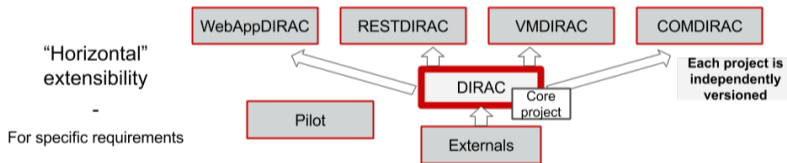
**Christophe Haen, Marko Petric, Ben Couturier**



Adelaide, 4-8 November 2019

# DIRAC

- ▶ Grid middleware used by many VOs (see #173)
- + WMS, DMS, etc
- + Extensible horizontally and vertically
- + Comes with all its dependencies



“Vertical”  
extensibility

-

Community driven



# Why?

- ▶ DIRAC requires many external library dependencies
  - + Python libraries
  - + Middleware
  - + Server Side Tools
- ▶ Historically managed by two independent packages



# Externals and LCGBundle

- ▶ Python, standard binary libraries, grid tools (gfal2, arc, ...)
- ▶ Pre-compiled for several platforms
- ▶ Different versions / composition for client and server
- ▶ Composed by manipulation/picking of existing LCG-Application-Area releases done by CERN EP-SFT and manual compilation

# The issues

- ▶ **Difficult testing environment**
  - ▶ **Externals User vs. Externals Server vs. LCGBundle**
- ▶ **Slow release process**
  - ▶ **Dependent on LCG-AA release cycle**
- ▶ **Substantial manual intervention by maintainers needed**
- ▶ **Extension or customisation of builds is hard**
- ▶ **Technical issues:**
  - ▶ **Reliance on non-system versions**
  - ▶ **Dependance on the GCC version of respective LCG**
  - ▶ **Python version newer than the native system → binary incompatibility**
  - ▶ **Stdlib with/without C++11 support**

# Shopping list

- ▶ **Portable**
  - ▶ At least SLC6 and CC7
- ▶ **Reproducible**
- ▶ **Extensible**
  - ▶ VO can have their own extension
- ▶ **Testable**
  - ▶ Unit and integration tests, nightlies, RC
- ▶ **Not a one man show**
  - ▶ Documented, automatized as much as possible

# DIRACOS

- ▶ **Single package list**
- ▶ **Recompiled all from SRPM**
- ▶ **All the dependencies are → pulled down to but excluding glibc**
- ▶ **Same packages for clients, servers and multiple platforms**
- ▶ **Solves the binary incompatibility of using a different python version from the native system**
- ▶ **Based on Fedora Mock and yum repo**
- ▶ **Extensible setup design**
- ▶ **Ideal for agile development cycle**

# Fedora Mock

- ▶ **Tool for building packages**
- ▶ **Creates chroots and builds packages in them**
- ▶ **Used for all Fedora builds (underlying Koji and Copr)**



# Grammar

```
{
  "rpmBuild": {
    "opt1": 1,
    "package Groups": [
      {
        "name": "pkgGroup1",
        "opt1": 2,
        "packages": [
          {
            "name": "pkg1",
            "opt2": "x"
          },
          {
            "name": "pkg2",
            "opt1": 3
          }
        ]
      }
    ]
  }
}
```

- ▶ **Functionality to cope with special cases**
- ▶ **Patching:**
  - ▶ certain RPM spec requires tweaking
  - ▶ simply add diff to patch folder
- ▶ **Routines for pre, post and instead actions**
- ▶ **List of Python packages handled via list in json configuration**

# Current Status

- ▶ Repo integrated into DIRACGrid
  - ▶ <https://github.com/DIRACGrid/DIRACOS>
- ▶ Current release: v1r4
- ▶ Available opt-in as from DIRAC v6r21
- ▶ **Default from DIRAC v7**
- ▶ Tests OK with SLC6, CC7, fedora and ubuntu
  - ▶ **Official support only for SLC6 and CC7**
- ▶ Bundle size 277 MB
  - ▶ Mostly distributed via CVMFS

# How to build?

## ▶ Generating of bundles made easy

### 1. Prerequisites on SLC6:

- ▶ `yum install -y mock rpm-build fedora-packager createrepo python-pip`
- ▶ `pip install diracos`

### 2. Compile all RPMs

- ▶ `dos-build-all-rpms config/diracos.json`

### 3. Generate requirements.txt with versions

- ▶ `dos-fix-pip-versions config/diracos.json`

### 4. Compile the python modules

- ▶ `dos-build-python-modules config/diracos.json`

### 5. Pull dependencies and tar

- ▶ `dos-bundle config/diracos.json`

# How to extend? Python

## ▶ Trivial configuration file

```
{  
  "extensionName": "LHCb",  
  "diracOsVersion": "v1r4",  
  "version": "master",  
  "pipRequirements": "lhcb_requirements.txt"  
}
```

## ▶ dos-build-extension lhcbdirac.json



# How to extend? RPM

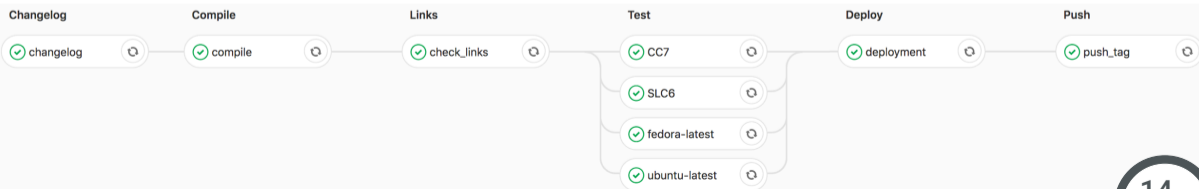
- ▶ **Modify config/diracos.json and add your desired SRPM**

```
{
  "name": "gfal2",
  "packages": [
    {
      "src": "https://diracos.web.cern.ch/diracos/SRPM/dcap-2.47.12-4.el6.src.rpm",
      "name": "dcap"
    },
    {
      "comment": "used by gfal2",
      "src": "https://diracos.web.cern.ch/diracos/SRPM/gtest-1.5.0-5.el6.src.rpm",
      "name": "gtest",
      "buildOnly": true
    },
    {
      "src": "__PATH_TO__/SRPM/package-version.el6.src.rpm",
      "name": "MyPackage"
    }
  ]
}
```

- ▶ **All SRPM archived at <http://diracos.web.cern.ch>**

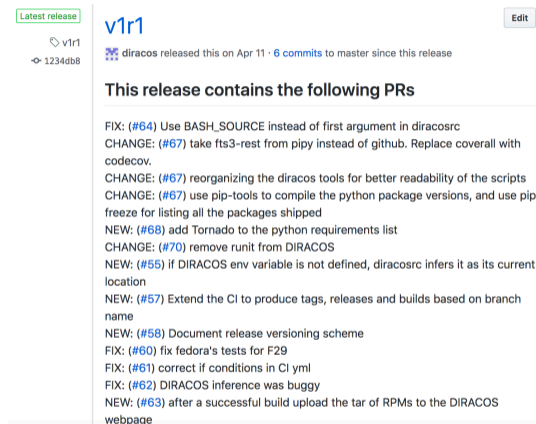
# What about testing?

- ▶ **DIRACOS equipped with working CI** (gitlab mirror at CERN)
- ▶ **repo build for every merge a posteriori**
- 1. **Check if there are any broken symbolic links**
- 2. **Check if there are any absolute symbolic links**
  - ▶ Whitelist some broken links that are inherent to some packages
- 3. **Try to import all python packages and see if dependencies can be resolved**
- 4. **Run some CLI tools to check working (e.g. gfal-stat)**
- ▶ **Still many tests missing**



# Release procedure

- ▶ Fully integrated into CI
- ▶ Based on branch name
- ▶ Create build only for testing
- ▶ **Create Release**
  - ▶ aggregate changes
  - ▶ make release build
  - ▶ run tests
  - ▶ compile list of versions
  - ▶ prepare release notes
  - ▶ make tag
  - ▶ publish on GitHub



The screenshot shows a GitHub release page for the tag `v1r1`. At the top left, there is a green box labeled "Latest release". Below it, the tag `v1r1` is shown with a link to the commit `1234db8`. To the right, the text "diracos released this on Apr 11 · 6 commits to master since this release" is displayed. An "Edit" button is visible in the top right corner. The main content is titled "This release contains the following PRs" and lists several pull requests with their descriptions:

- FIX: (#64) Use BASH\_SOURCE instead of first argument in diracosrc
- CHANGE: (#67) take fts3-rest from pipy instead of github. Replace coverall with codecov.
- CHANGE: (#67) reorganizing the diracos tools for better readability of the scripts
- CHANGE: (#67) use pip-tools to compile the python package versions, and use pip freeze for listing all the packages shipped
- NEW: (#68) add Tornado to the python requirements list
- CHANGE: (#70) remove runit from DIRACOS
- NEW: (#55) if DIRACOS env variable is not defined, diracosrc infers it as its current location
- NEW: (#57) Extend the CI to produce tags, releases and builds based on branch name
- NEW: (#58) Document release versioning scheme
- FIX: (#60) fix fedora's tests for F29
- FIX: (#61) correct if conditions in CI yml
- FIX: (#62) DIRACOS inference was buggy
- NEW: (#63) after a successful build upload the tar of RPMs to the DIRACOS webpage

# Release procedure

- ▶ Fully integrated into CI
- ▶ Based on branch name
- ▶ Create build only for testing
- ▶ **Create Release**
  - ▶ aggregate changes
  - ▶ make release build
  - ▶ run tests
  - ▶ compile list of versions
  - ▶ prepare release notes
  - ▶ make tag
  - ▶ publish on GitHub

## Included versions of packages

===== RPM packages =====

autoconf-2.63-5.1.el6.noarch.rpm  
automake-1.11.1-4.el6.noarch.rpm  
boost-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-date-time-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-devel-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-filesystem-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-graph-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-iostreams-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-math-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-program-options-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-python-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-regex-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-serialization-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-signals-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-static-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-system-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-test-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-thread-1.41.0-28.el6.py27.usc4.x86\_64.rpm  
boost-wave-1.41.0-28.el6.py27.usc4.x86\_64.rpm



# Release procedure

- ▶ Fully integrated into CI
- ▶ Based on branch name
- ▶ Create build only for testing
- ▶ **Create Release**
  - ▶ aggregate changes
  - ▶ make release build
  - ▶ run tests
  - ▶ compile list of versions
  - ▶ prepare release notes
  - ▶ make tag
  - ▶ publish on GitHub

==== Python packages ====

```
astroid==1.6.6
atomicwrites==1.3.0
attrs==19.1.0
autopep8==1.3.3
backports-abc==0.5
backports.functools-lru-cache==1.5
backports.shutil-get-terminal-size==1.0.0
certifi==2019.3.9
chardet==3.0.4
CMRESHandler==1.0.0
codecov==2.0.15
configparser==3.7.4
coverage==4.5.3
cx-Oracle==7.1.2
cyclerr==0.10.0
decorator==4.4.0
docopt==0.6.2
docutils==0.14
elasticsearch==6.3.1
elasticsearch-dsl==6.3.1
```

- ▶ Automatic self documentation of releases
- ▶ Substantial amount of documentation in repo (~ 700 lines)
- ▶ There is always room for improvement

## DIRACOS

pipeline passed

DIRACOS aims at bringing in one archive all the dependencies needed by DIRAC.

- Principle
  - How it works
    - Bootstrap issue
    - About EPEL repository
    - About links
  - Supported platforms
    - Trick
- Building RPMs
  - Patching the sources
  - Altering the normal build workflow
  - Caching mechanism
  - Adding an RPM package
- Python packages
  - Adding a python package
- Generate a new diracos
  - Initial setup
  - Configuration files
  - Building everything
  - Build the python modules
  - Bundle DIRACOS
  - Get your bundle
  - Test It !
- Configuration Grammar
  - rpmBuild section
    - Package and PackageGroup
      - Mandatory parameters
      - Optional parameters
  - Other sections and options
- Troubleshoot
  - Error `Requires: python(abi) = 2.6`
  - Build is failing for broken rpm dependencies

# Summary

- ▶ Reached production level quality
  - ▶ DIRAC certification performed with DIRACOS
  - ▶ Already in production (LHCb, ILC, FranceGrilles)
- ▶ Minimised size of bundle to reasonable size
- ▶ In sync with an agile development cycle
- ▶ Conda in v2? (see #488 *Sustainable packaging for end users with conda*)

