On Python 3, DIRACOS, and other FAQs

Chris Burr

DIRAC User's Workshop 2022 ○ 9th May 2022

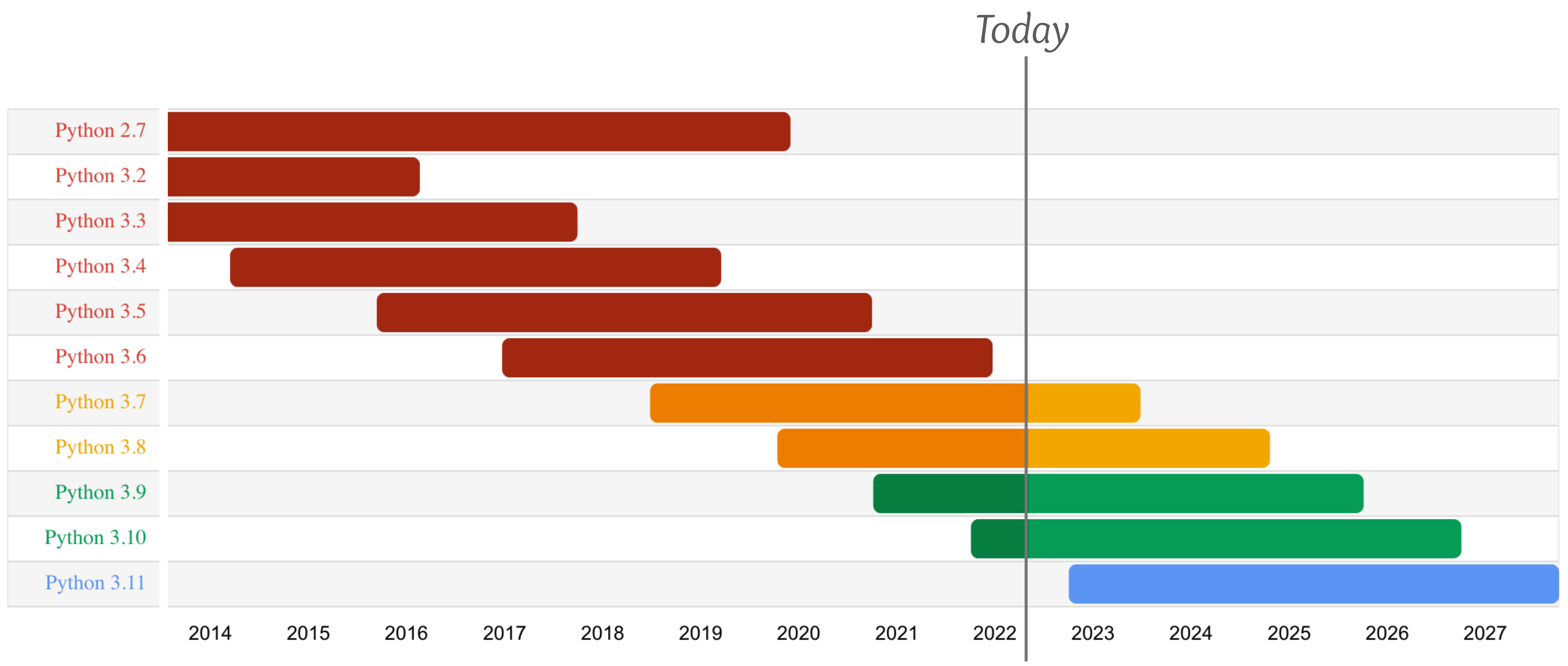
Introduction

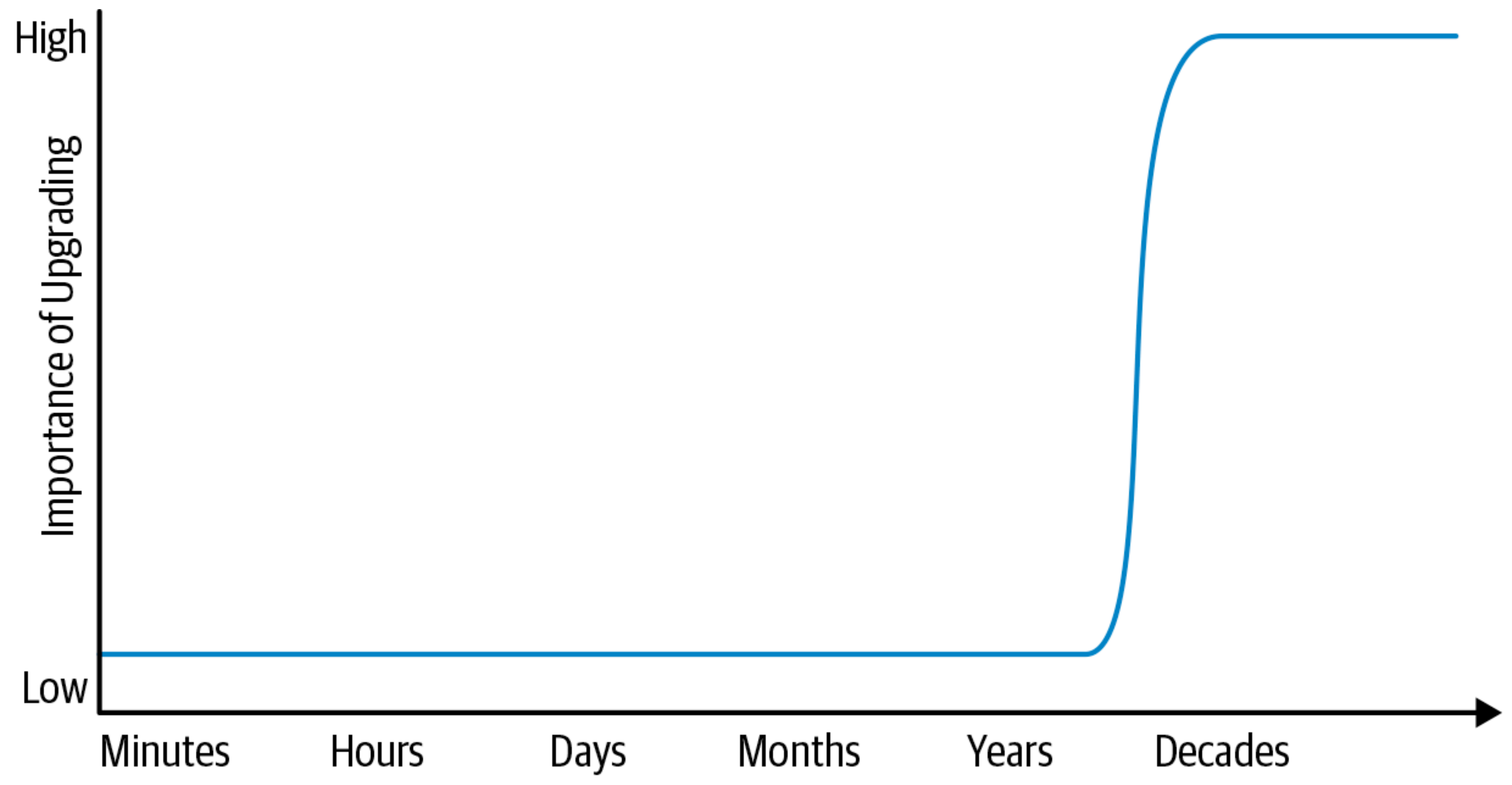
- What is happening with Python 3, DIRACOS 2 and DIRAC distribution?
- How to migrate to Python 3-based DIRAC releases?
  - Prepare any extensions
  - Moving servers
  - What's changed?
- General developer tooling

Python 3 and DIRACOS2

The sunset of Python 2

- Python 2 reached end of life on 31st December 2019
  - 2 years and 4 moths ago
- Python 2 now represents less than 5% of PyPI downloads
- “Python 3” no longer exists, it’s just called “Python”





What does this mean for DIRAC?

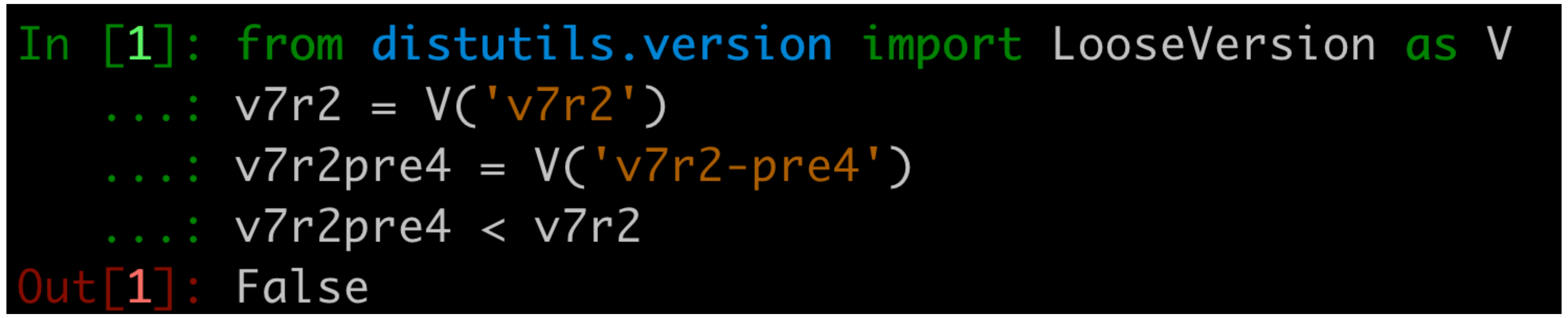
- Lots of minor modernisations and fixes
  - Modernising naturally results in a code base compatible with both versions
  - DIRAC Extensions also need to be updated
  - See: https://dirac.readthedocs.io/en/latest/DeveloperGuide/Python3Migration/index.html
- Using DIRAC remains the same, except…
- The migration is accompanied by a change of mindset
  - DIRAC is no longer the center of the universe
  - Continue to distribute a “tarball” with DIRAC’s dependencies but also support other options (i.e. DIRACOS)
  - The client should be usable alongside other software

What about Python 3.10+

- Upgrading Python 3.x versions is much simpler than Python 2.7 -> 3.x
  - Most backward incompatibly changes happen in layers below DIRAC
  - Most issues are easy to find
  - Especially when going one step at a time
- Python version comes with DIRACOS
  - DIRACOS2 releases are tested against every supported DIRAC release
  - We strongly encourage extensions to have automated tests

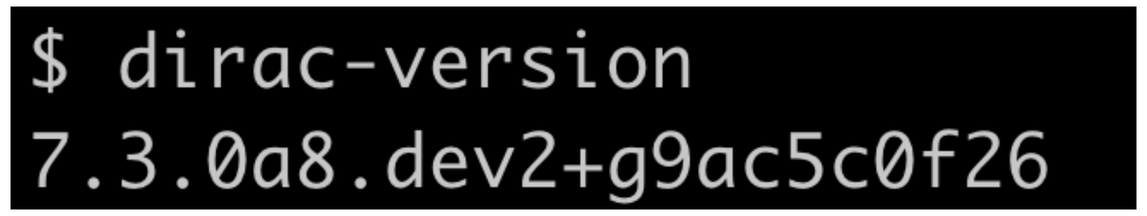
Versioning

- The vXrYpZ-style versioning is completely non-standard
  - Almost every tool will fail to do things like “upgrade to latest v7r2”



```
In [1]: from distutils.version import LooseVersion as V
   ...: v7r2 = V('v7r2')
   ...: v7r2pre4 = V('v7r2-pre4')
   ...: v7r2pre4 < v7r2
Out[1]: False
```

- Python has a standard for this: PEP-440
  - Compatible with the standards that almost everybody uses for version numbers
  - Several useful extensions, mostly for development version numbers
  - Example: 2 git commits after v7.3.0a8, commit hash is `9ac5c0f26`



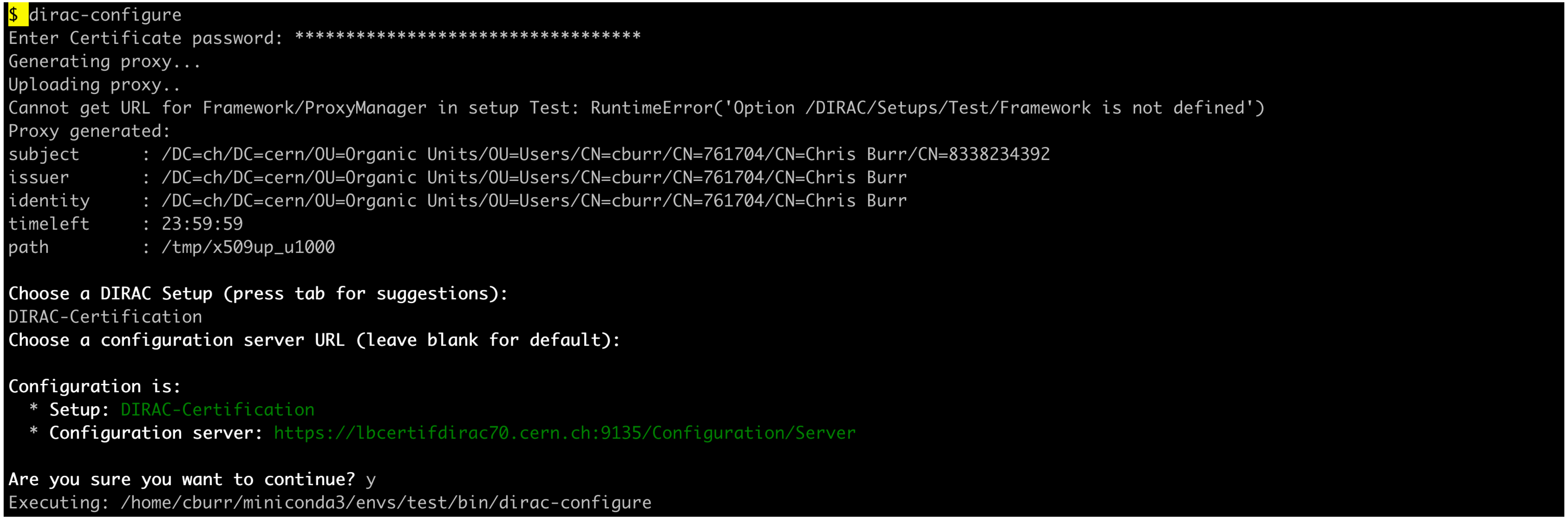
```
$ dirac-version
7.3.0a8.dev2+g9ac5c0f26
```

- Use `setuptools_scm` to automatically set version numbers using git metadata

- Map “pre” to alpha for now (i.e. v7r3-pre6 is 7.3.0a6)

Configuration

- ➤ From DIRAC 7.3: `dirac-configuration` can be used without arguments
- ➤ Just* need to add your setup/URL in: `DIRAC.__init__:extension_metadata`

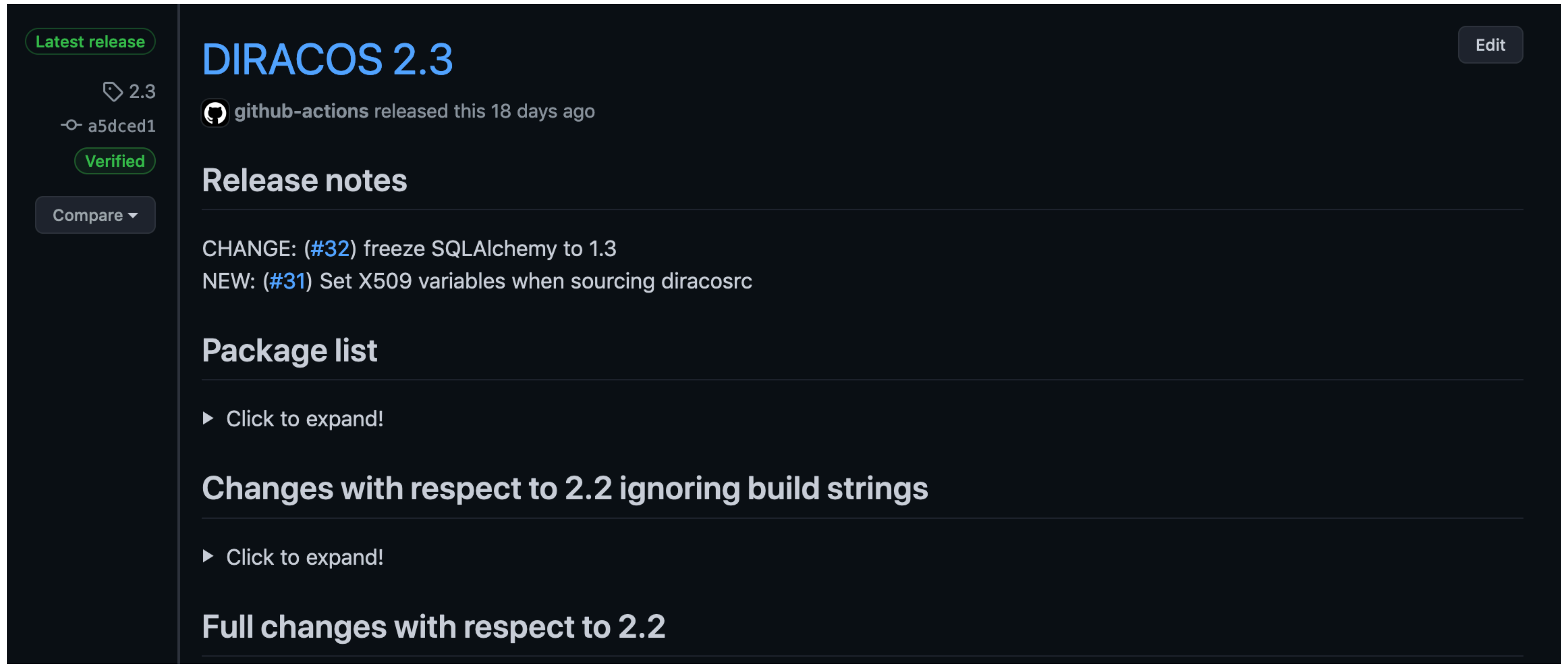


```
$ dirac-configure
Enter Certificate password: **********************************
Generating proxy...
Uploading proxy..
Cannot get URL for Framework/ProxyManager in setup Test: RuntimeError('Option /DIRAC/Setups/Test/Framework is not defined')
Proxy generated:
subject      : /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=cburr/CN=761704/CN=Chris Burr/CN=8338234392
issuer       : /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=cburr/CN=761704/CN=Chris Burr
identity     : /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=cburr/CN=761704/CN=Chris Burr
timeleft     : 23:59:59
path         : /tmp/x509up_u1000

Choose a DIRAC Setup (press tab for suggestions):
DIRAC-Certification
Choose a configuration server URL (leave blank for default):

Configuration is:
  * Setup: DIRAC-Certification
  * Configuration server: https://lbcertifdirac70.cern.ch:9135/Configuration/Server

Are you sure you want to continue? y
Executing: /home/cburr/miniconda3/envs/test/bin/dirac-configure
```

**Assuming you don't have an extension*

DIRACOS 2

- DIRACOS(1) is CentOS 6 based
  - Needs updating, end of life was 30th November 2020
- DIRACOS 2 similar model to DIRACOS 1
  - Relocatable binary distribution containing everything needed for servers, clients and pilots
- Repository: https://github.com/DIRACGrid/DIRACOS2
  - Releases hosted with GitHub releases, no more issues with https://diracos.web.cern.ch



DIRACOS 2.3

github-actions released this 18 days ago

Release notes

CHANGE: (#32) freeze SQLAlchemy to 1.3
NEW: (#31) Set X509 variables when sourcing diracosrc

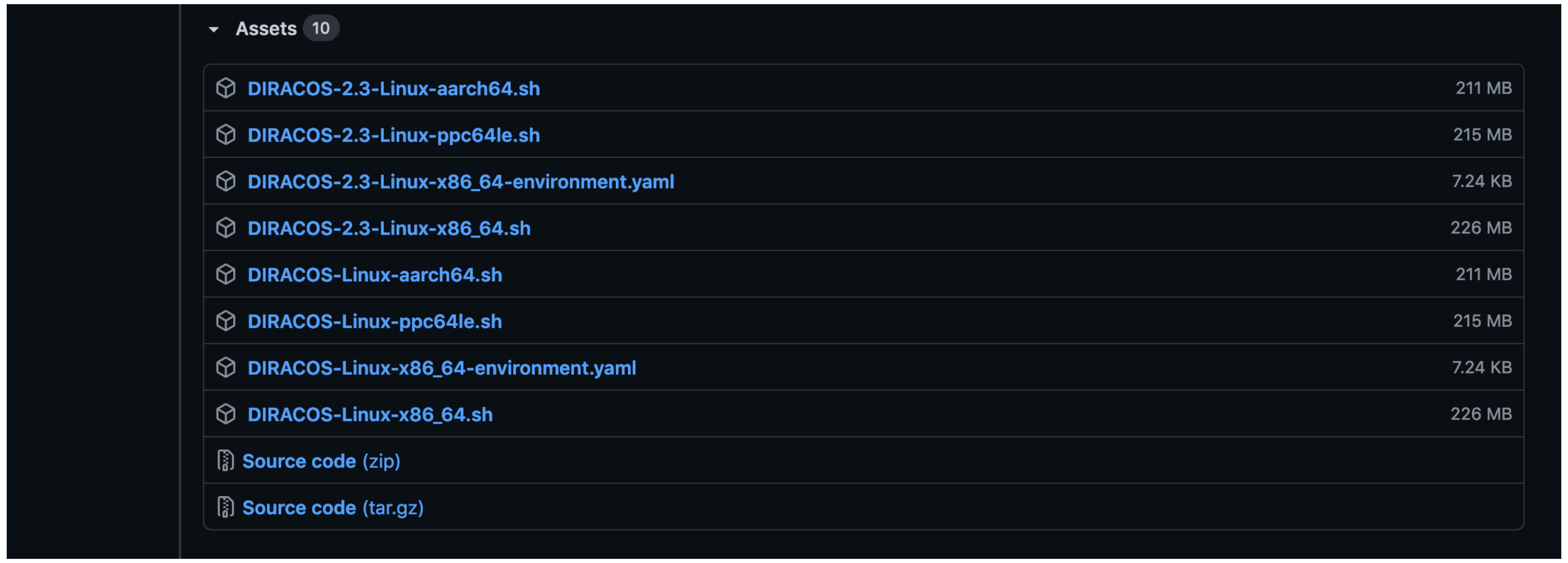
Package list

▸ Click to expand!

Changes with respect to 2.2 ignoring build strings

▸ Click to expand!

Full changes with respect to 2.2



Assets 10

DIRACOS-2.3-Linux-aarch64.sh	211 MB
DIRACOS-2.3-Linux-ppc64le.sh	215 MB
DIRACOS-2.3-Linux-x86_64-environment.yaml	7.24 KB
DIRACOS-2.3-Linux-x86_64.sh	226 MB
DIRACOS-Linux-aarch64.sh	211 MB
DIRACOS-Linux-ppc64le.sh	215 MB
DIRACOS-Linux-x86_64-environment.yaml	7.24 KB
DIRACOS-Linux-x86_64.sh	226 MB
Source code (zip)	
Source code (tar.gz)	

What is Conda

- ***Language agnostic*** package manager (Python, C++, R, Julia, Rust, Go, Java, Ruby, Fortran, …)
- Multi platform (Linux, macOS, Windows)
- Multi architecture (i386, x86_84, aarch64, ppc64le, partially s390x)
- Provides "environments" which are self contained sysroots in a folder
  - No admin privileges required
  - Easy to preserve long term
- Easily switch between Python versions, compilers and other packages

CONDA-FORGE

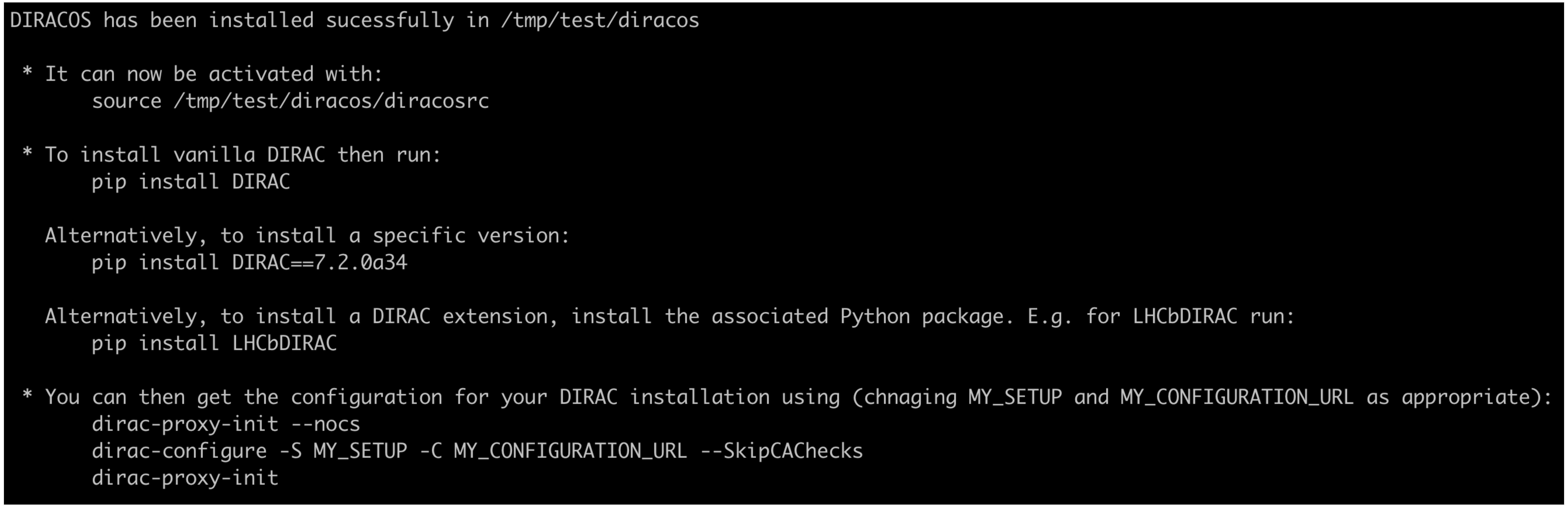
- Community maintained collection of conda packages
- Over 17,900 packages available and rapidly growing
- Over 3,900 maintainers
- Over 350,000,000 package downloads each month
- Fiscally sponsored project of NumFOCUS
- Includes everything user facing (vim/curl/findutils/htop/...)

- Should work on almost any Linux machine
  - Requires glibc 2.17 or later (CentOS 7, Ubuntu 13.04, Fedora 20, OpenSUSE 13, Debian 8)
- Changes:
  - Uses conda-forge as the source of the packages (via constructor)
  - Self-extracting executable instead of a tarball - only relocatable at install time
  - Support for aarch64 and ppc64le
  - Much faster builds (~3 minutes vs ~3 hours)

- `dirac-install` is no longer used for Python 3 based installations

```
$ curl -LO https://github.com/DIRACGrid/DIRACOS2/releases/latest/download/DIRACOS-Linux-x86_64.sh
$ bash DIRACOS-Linux-x86_64.sh [-p /path/for/installation]
```

- Then prints instructions for how to proceed:



```
DIRACOS has been installed sucessfully in /tmp/test/diracos

 * It can now be activated with:
       source /tmp/test/diracos/diracosrc

 * To install vanilla DIRAC then run:
       pip install DIRAC

   Alternatively, to install a specific version:
       pip install DIRAC==7.2.0a34

   Alternatively, to install a DIRAC extension, install the associated Python package. E.g. for LHCbDIRAC run:
       pip install LHCbDIRAC

 * You can then get the configuration for your DIRAC installation using (chnaging MY_SETUP and MY_CONFIGURATION_URL as appropriate):
       dirac-proxy-init --nocs
       dirac-configure -S MY_SETUP -C MY_CONFIGURATION_URL --SkipCAChecks
       dirac-proxy-init
```

DIRACOS Extensions

- Currently don't foresee a need for DIRACOS extensions
- Extensions are just Python packages
  - Any other python packages can be added as a dependencies
- If the need arises, will likely be a complete copy of DIRACOS 2
  - Just change construct.yml to contain different packages

How to migrate to Python 3?

- Extensions should exist on PyPI as Python packages
  - Change to a src-style layout (see slide 7)
  - Create pyproject.toml and setup.cfg files
  - Tag, build sdist and bdist with "python -m build" then upload to PyPI with twine
- WebApp extensions are compiled as part of the Python build process
  - Installs of unreleased versions with pip should work
  - Uses docker or singularity to get the web app compiler
  - See: `dirac-webapp-packaging`
- For complete examples refer to LHCbDIRAC and LHCbWebDIRAC

Package layout for extensions

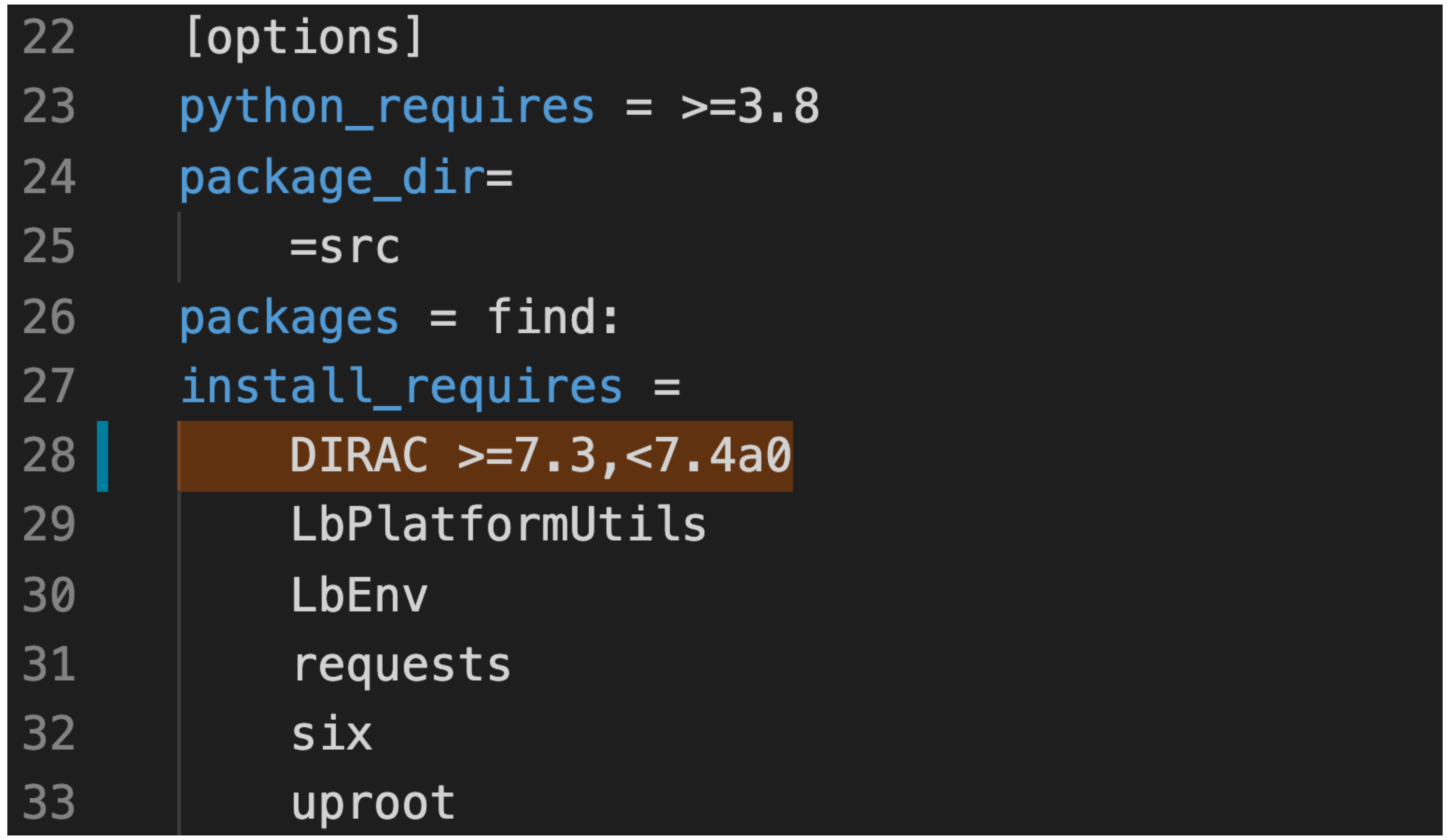
- The layout prior to DIRAC 7.1 is fairly unique
  - Requires the repository checkout folder is “DIRAC”
  - Mixes package code with other things that are in repository (CI config, linters, …)
- Most packages have now moved to a “src layout”

```
DIRAC
    – .git/
    – tests/
    – src/
        – DIRAC/
            – __init__.py
            – Core/
    – README.rst
```

- It’s “ugly” but has many subtle benefits, especially when running tests
- For a full justification: https://hynek.me/articles/testing-packaging/

Inverting the dependency

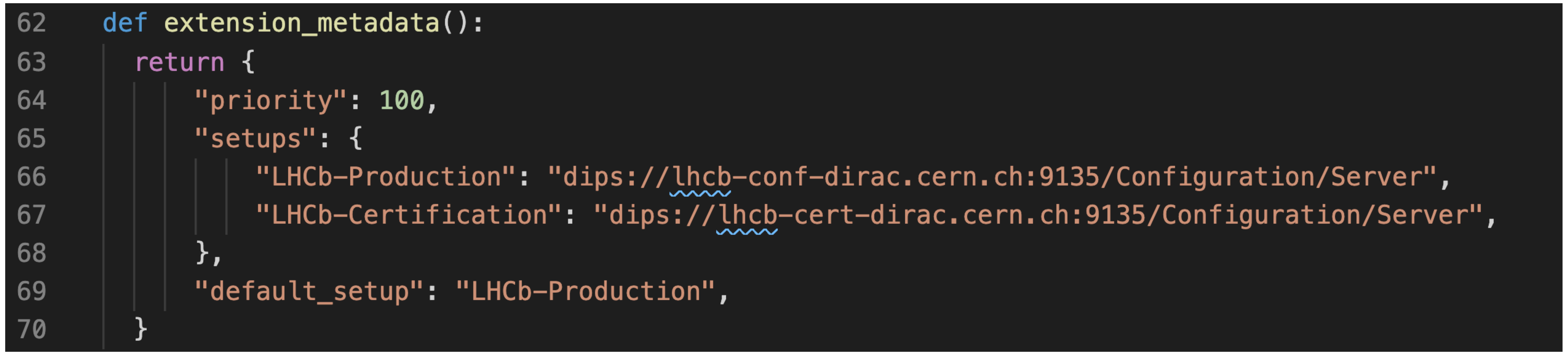
- ➤ Invert how the dependencies are handled
  - ➤ Old way: Install DIRAC with MyDIRAC
  - ➤ New way: Install MyDIRAC, vanilla DIRAC is included automatically
- ➤ No need for releases.cfg, just setuptools metadata in each release



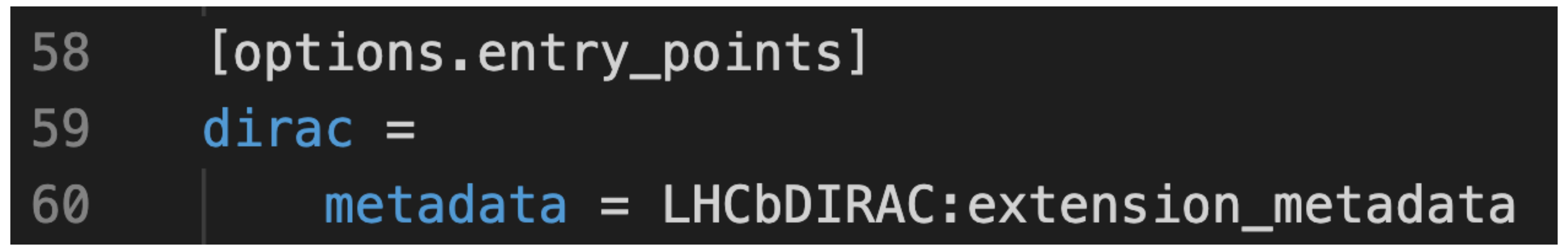
```
[options]
python_requires = >=3.8
package_dir=
    =src
packages = find:
install_requires =
    DIRAC >=7.3,<7.4a0
    LbPlatformUtils
    LbEnv
    requests
    six
    uproot
```

DIRAC Extensions

- Extensions make themselves known to using a "dirac" "entrypoint"
  - https://setuptools.readthedocs.io/en/latest/userguide/entry_point.html
  - https://amir.rachum.com/blog/2017/07/28/python-entry-points/



```python
def extension_metadata():
    return {
        "priority": 100,
        "setups": {
            "LHCb-Production": "dips://lhcb-conf-dirac.cern.ch:9135/Configuration/Server",
            "LHCb-Certification": "dips://lhcb-cert-dirac.cern.ch:9135/Configuration/Server",
        },
        "default_setup": "LHCb-Production",
    }
```



```ini
[options.entry_points]
dirac =
    metadata = LHCbDIRAC:extension_metadata
```

- Care is also needed if you defined any `dirac-xxx-yyy` command line scripts
  - https://dirac.readthedocs.io/en/latest/DeveloperGuide/AddingNewComponents/DevelopingCommands/index.html

- Ordering in DIRAC/Extensions in the CS is no longer needed

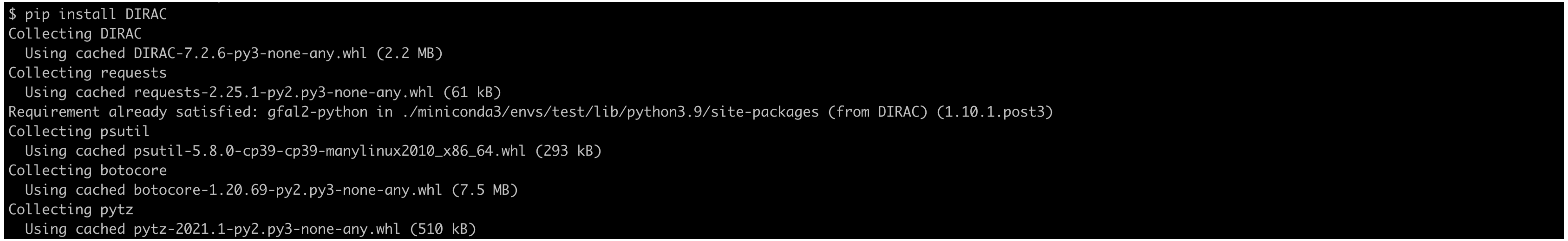
Updating

- See the release notes for DIRAC v7r2 and v7r3 for details
- **While running DIRAC v7r2:** start using Python 3 clients and pilots
  - Can be done on a per-CE basis
- **While running DIRAC v7r3:** Migrate servers one-by-one
  - Modify the bashrc as described in the v7r3 release notes
  - Specify a new-style version number when updating using the system administrator
  - Monitor the logs for issues
  - Can roll back by renaming the “old” symlink to “pro” and restarting
- LHCbDIRAC has been stably Python 3-only since December 2021

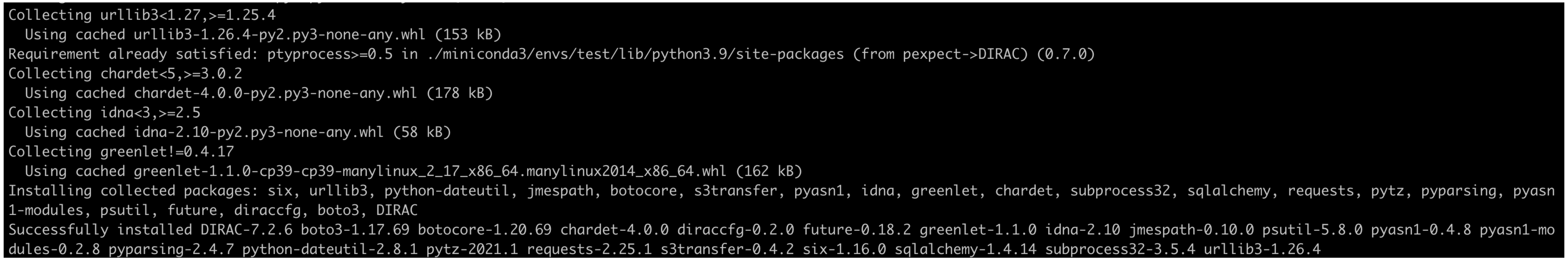
Other distribution channels

Pip

- Python packages should normally be installed using pip from PyPI
  - Sometimes hidden from the user (e.g. Conda packages of Python code are normally built using pip)
- Installing Python 3 DIRAC inside DIRACOS is done using pip



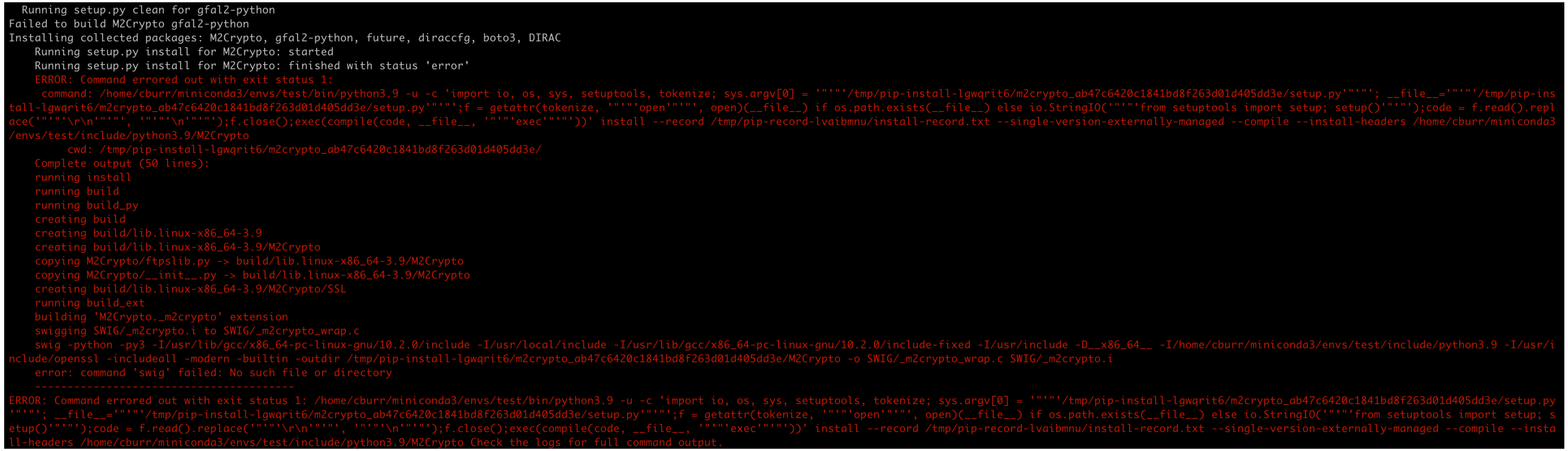
```
$ pip install DIRAC
Collecting DIRAC
  Using cached DIRAC-7.2.6-py3-none-any.whl (2.2 MB)
Collecting requests
  Using cached requests-2.25.1-py2.py3-none-any.whl (61 kB)
Requirement already satisfied: gfal2-python in ./miniconda3/envs/test/lib/python3.9/site-packages (from DIRAC) (1.10.1.post3)
Collecting psutil
  Using cached psutil-5.8.0-cp39-cp39-manylinux2010_x86_64.whl (293 kB)
Collecting botocore
  Using cached botocore-1.20.69-py2.py3-none-any.whl (7.5 MB)
Collecting pytz
  Using cached pytz-2021.1-py2.py3-none-any.whl (510 kB)
```



```
Collecting urllib3<1.27,>=1.25.4
  Using cached urllib3-1.26.4-py2.py3-none-any.whl (153 kB)
Requirement already satisfied: ptyprocess>=0.5 in ./miniconda3/envs/test/lib/python3.9/site-packages (from pexpect->DIRAC) (0.7.0)
Collecting chardet<5,>=3.0.2
  Using cached chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting greenlet!=0.4.17
  Using cached greenlet-1.1.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (162 kB)
Installing collected packages: six, urllib3, python-dateutil, jmespath, botocore, s3transfer, pyasn1, idna, greenlet, chardet, subprocess32, sqlalchemy, requests, pytz, pyparsing, pyasn
1-modules, psutil, future, diraccfg, boto3, DIRAC
Successfully installed DIRAC-7.2.6 boto3-1.17.69 botocore-1.20.69 chardet-4.0.0 diraccfg-0.2.0 future-0.18.2 greenlet-1.1.0 idna-2.10 jmespath-0.10.0 psutil-5.8.0 pyasn1-0.4.8 pyasn1-mo
dules-0.2.8 pyparsing-2.4.7 python-dateutil-2.8.1 pytz-2021.1 requests-2.25.1 s3transfer-0.4.2 six-1.16.0 sqlalchemy-1.4.14 subprocess32-3.5.4 urllib3-1.26.4
```

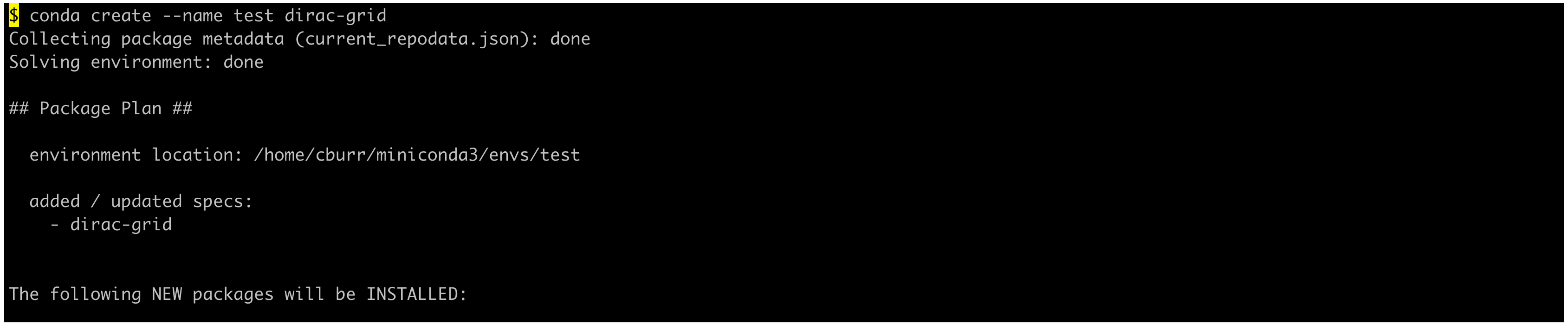
But that’s not the whole story…

- Pip is only for Python packages
  - Can’t be used to ship things like `voms` (essential for generating valid X509 proxies)
- Also requires the package to be uploaded to PyPI “correctly”
  - M2Crypto and gfal2 currently don’t provide pre-compiled binaries (known as “bdists” or “wheels”)
- Impossible to set things like `X509_*` environment variables

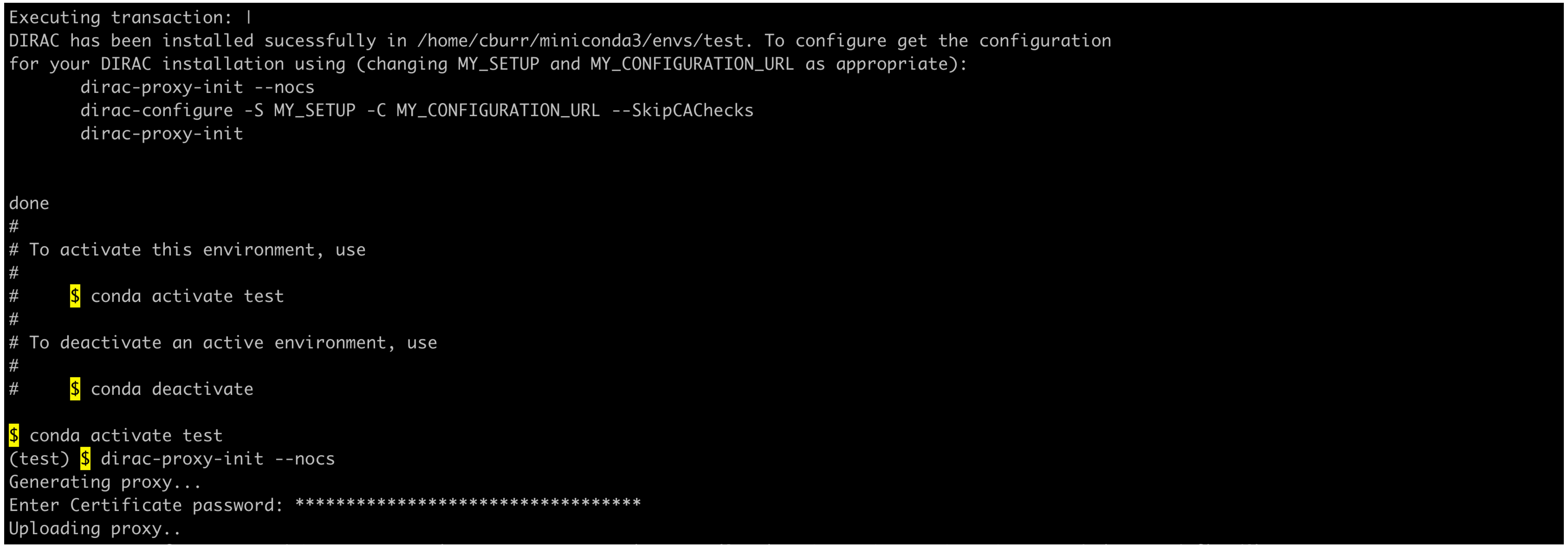


```
  Running setup.py clean for gfal2-python
Failed to build M2Crypto gfal2-python
Installing collected packages: M2Crypto, gfal2-python, future, diraccfg, boto3, DIRAC
    Running setup.py install for M2Crypto: started
    Running setup.py install for M2Crypto: finished with status 'error'
    ERROR: Command errored out with exit status 1:
     command: /home/cburr/miniconda3/envs/test/bin/python3.9 -u -c 'import io, os, sys, setuptools, tokenize; sys.argv[0] = '"'"'/tmp/pip-install-lgwqrit6/m2crypto_ab47c6420c1841bd8f263d01d405dd3e/setup.py'"'"'; __file__='"'"'/tmp/pip-ins
tall-lgwqrit6/m2crypto_ab47c6420c1841bd8f263d01d405dd3e/setup.py'"'"';f = getattr(tokenize, '"'"'open'"'"', open)(__file__) if os.path.exists(__file__) else io.StringIO('"'"'from setuptools import setup; setup()'"'"');code = f.read().repl
ace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' install --record /tmp/pip-record-lvaibmnu/install-record.txt --single-version-externally-managed --compile --install-headers /home/cburr/miniconda3
/envs/test/include/python3.9/M2Crypto
         cwd: /tmp/pip-install-lgwqrit6/m2crypto_ab47c6420c1841bd8f263d01d405dd3e/
    Complete output (50 lines):
    running install
    running build
    running build_py
    creating build
    creating build/lib.linux-x86_64-3.9
    creating build/lib.linux-x86_64-3.9/M2Crypto
    copying M2Crypto/ftpslib.py -> build/lib.linux-x86_64-3.9/M2Crypto
    copying M2Crypto/__init__.py -> build/lib.linux-x86_64-3.9/M2Crypto
    creating build/lib.linux-x86_64-3.9/M2Crypto/SSL
    running build_ext
    building 'M2Crypto._m2crypto' extension
    swigging SWIG/_m2crypto.i to SWIG/_m2crypto_wrap.c
    swig -python -py3 -I/usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/include -I/usr/local/include -I/usr/lib/gcc/x86_64-pc-linux-gnu/10.2.0/include-fixed -I/usr/include -D__x86_64__ -I/home/cburr/miniconda3/envs/test/include/python3.9 -I/usr/i
nclude/openssl -includeall -modern -builtin -outdir /tmp/pip-install-lgwqrit6/m2crypto_ab47c6420c1841bd8f263d01d405dd3e/M2Crypto -o SWIG/_m2crypto_wrap.c SWIG/_m2crypto.i
    error: command 'swig' failed: No such file or directory
    ----------------------------------------
ERROR: Command errored out with exit status 1: /home/cburr/miniconda3/envs/test/bin/python3.9 -u -c 'import io, os, sys, setuptools, tokenize; sys.argv[0] = '"'"'/tmp/pip-install-lgwqrit6/m2crypto_ab47c6420c1841bd8f263d01d405dd3e/setup.py
'"'"'; __file__='"'"'/tmp/pip-install-lgwqrit6/m2crypto_ab47c6420c1841bd8f263d01d405dd3e/setup.py'"'"';f = getattr(tokenize, '"'"'open'"'"', open)(__file__) if os.path.exists(__file__) else io.StringIO('"'"'from setuptools import setup; s
etup()'"'"');code = f.read().replace('"'"'\r\n'"'"', '"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' install --record /tmp/pip-record-lvaibmnu/install-record.txt --single-version-externally-managed --compile --insta
ll-headers /home/cburr/miniconda3/envs/test/include/python3.9/M2Crypto Check the logs for full command output.
```

Installing DIRAC with conda



```
$ conda create --name test dirac-grid
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/cburr/miniconda3/envs/test

  added / updated specs:
    - dirac-grid


The following NEW packages will be INSTALLED:
```

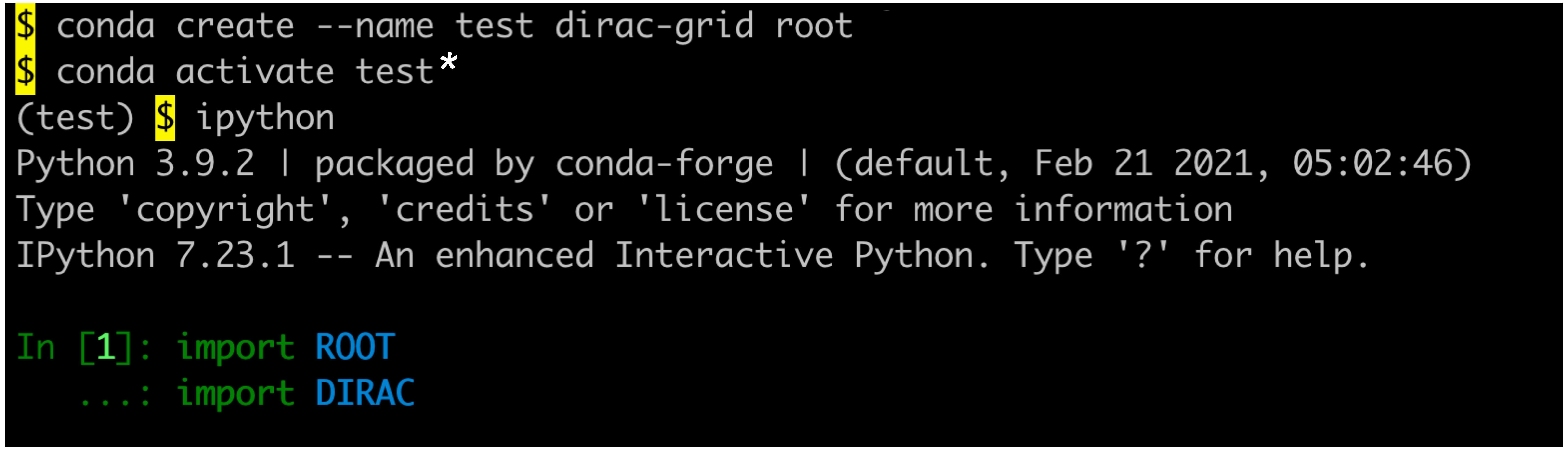


```
Executing transaction: |
DIRAC has been installed sucessfully in /home/cburr/miniconda3/envs/test. To configure get the configuration
for your DIRAC installation using (changing MY_SETUP and MY_CONFIGURATION_URL as appropriate):
       dirac-proxy-init --nocs
       dirac-configure -S MY_SETUP -C MY_CONFIGURATION_URL --SkipCAChecks
       dirac-proxy-init


done
#
# To activate this environment, use
#
#     $ conda activate test
#
# To deactivate an active environment, use
#
#     $ conda deactivate

$ conda activate test
(test) $ dirac-proxy-init --nocs
Generating proxy...
Enter Certificate password: **********************************
Uploading proxy..
```

Full instructions at: https://github.com/DIRACGrid/DIRAC/#install

Installing DIRAC with conda

- ➤ Works on macOS and almost any Linux
- ➤ Can easily install things alongside DIRAC, e.g.



```
$ conda create --name test dirac-grid root
$ conda activate test*
(test) $ ipython
Python 3.9.2 | packaged by conda-forge | (default, Feb 21 2021, 05:02:46)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.23.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import ROOT
   ...: import DIRAC
```

**The dirac-configure step was skipped*

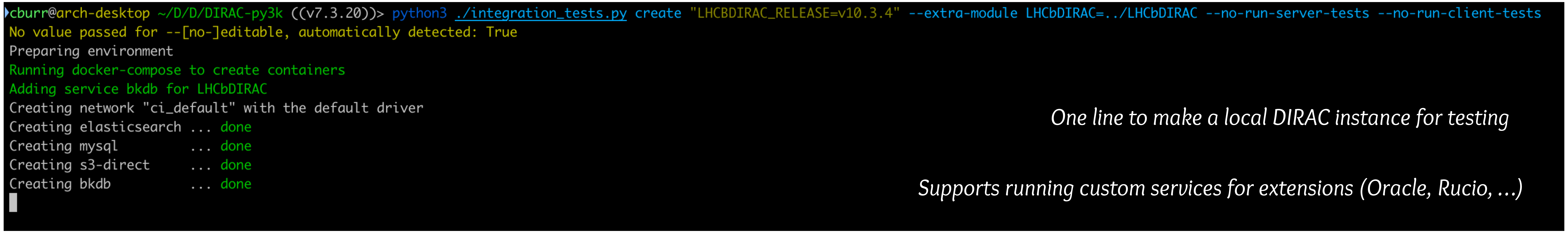
An aside: conda vs mamba vs micromamba vs anaconda

- Conda-forge has grown very large, conda sometimes struggles to manage
- Mamba is an alternative implementation with a faster dependency solver
  - Will replace the default Conda solver eventually
  - Latest version of Conda supports using mamba by passing "`--experimental-solver=libmamba`"
  - For details see: A faster Conda for a growing community
- Micromamba is a small standalone binary that is mostly-compatible
  - Used internally by DIRACOS2
  - Very helpful as a small/fast alternative for CI and containers
- Anaconda Inc. provides the commercial Anaconda and Miniconda installers
  - Not compatible with DIRAC, should make a new environment with Conda-forge
  - Or use one of the free Miniforge installers

TLDR: If Conda is slow, replace `conda install/create` **with** `mamba install/create`

Developer tooling

- Robust testing requires a multi-levelled approach:
  - **Linting:** Statically detect obvious errors
  - **Basic tests:** Relatively fast test to ensure individual components work
  - **Integration tests:** Make a full DIRAC setup and test as much as possible
  - **"Certification hackathons":** Catch hard to automate issues (VOMS, CEs, SEs)



```
cburr@arch-desktop ~/D/D/DIRAC-py3k ((v7.3.20))> python3 ./integration_tests.py create "LHCBDIRAC_RELEASE=v10.3.4" --extra-module LHCbDIRAC=../LHCbDIRAC --no-run-server-tests --no-run-client-tests
No value passed for --[no-]editable, automatically detected: True
Preparing environment
Running docker-compose to create containers
Adding service bkdb for LHCbDIRAC
Creating network "ci_default" with the default driver
Creating elasticsearch ... done
Creating mysql         ... done
Creating s3-direct     ... done
Creating bkdb          ... done
```

One line to make a local DIRAC instance for testing

Supports running custom services for extensions (Oracle, Rucio, …)

https://dirac.readthedocs.io/en/latest/DeveloperGuide/CodeTesting/index.html#running-integration-tests-locally

Code formatting

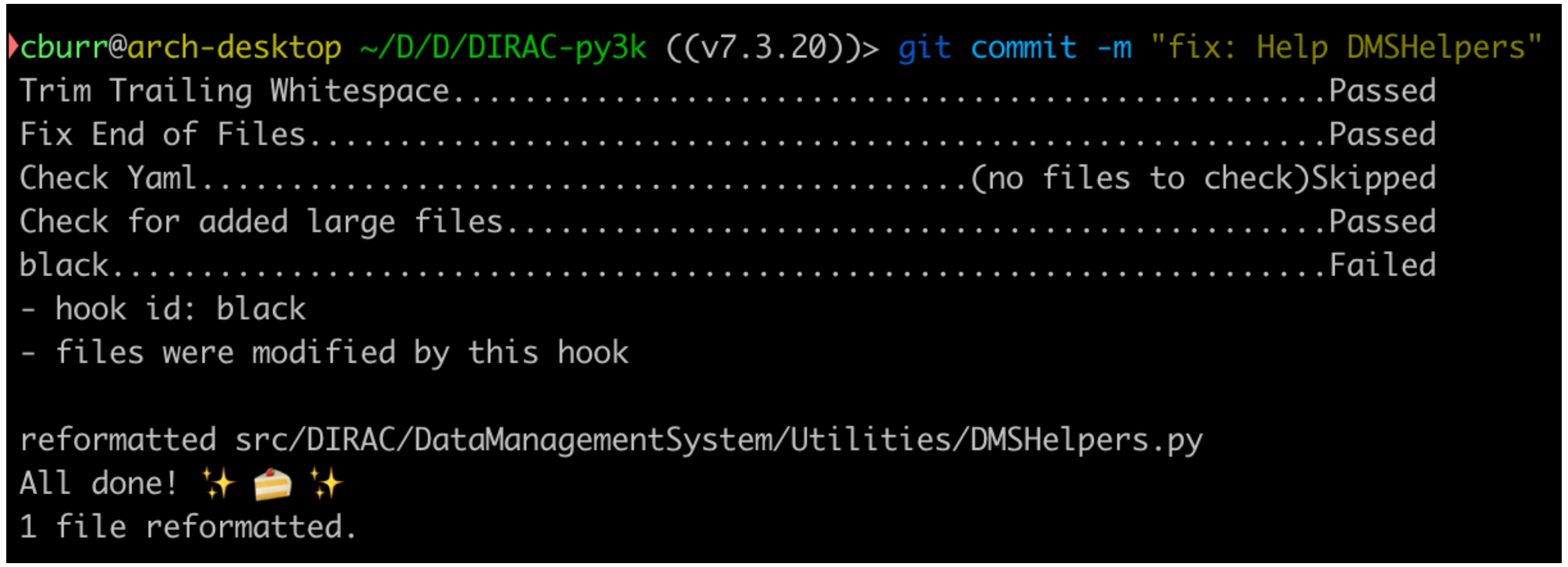
- All active DIRAC branches are now formatted with black
  - Deterministic automated code formatting
  - Style optimised to be version control friendly
  - Very widely used

- Advantages
  - Code looks consistent
  - Write code however you like and black will fix it
  - Large scale refactoring is much easier

"Any colour you like."

pre-commit

- Framework for managing pre-commit hooks
  - Scripts that run when you execute “git commit”
- Allows black to be ran automatically before each commit
- Can be extended to run other checks or formatters (pyupgrade?)



```
cburr@arch-desktop ~/D/D/DIRAC-py3k ((v7.3.20))> git commit -m "fix: Help DMSHelpers"
Trim Trailing Whitespace.................................................Passed
Fix End of Files.........................................................Passed
Check Yaml...........................................(no files to check)Skipped
Check for added large files..............................................Passed
black....................................................................Failed
- hook id: black
- files were modified by this hook

reformatted src/DIRAC/DataManagementSystem/Utilities/DMSHelpers.py
All done! ✨ 🍰 ✨
1 file reformatted.
```

Commit message checking

- Commit messages to DIRAC are now required to match

```
^(docs|feat|fix|refactor|style|test|sweep)( ?\(.*\))?: .+$
```

- **Why:** Maintain a cleaner git history, especially for debugging
- Messages should be written relative to what is in the target branch
  - "fix" and "refactor" aren't for fixing/refactoring your PR
  - If the PR is broken, amend the commit and force push

Sweeping

- Applying changes to multiple release branches is hard
  - Previously managed by merging to higher branches when making releases
  - Error prone, tedious and focuses the work on a single person
- Now use what we call “sweeping”
- When a PR is opened a label is added indicating where it should be swept
  - Manually add the “sweep:ignore” label if it shouldn’t be swept to higher release branches
- When a PR is merged it is “cherry-picked” on to the upper branch
  - If this fails an issue is opened with instructions for how to fix it manually

Wrap up…

Roadmap

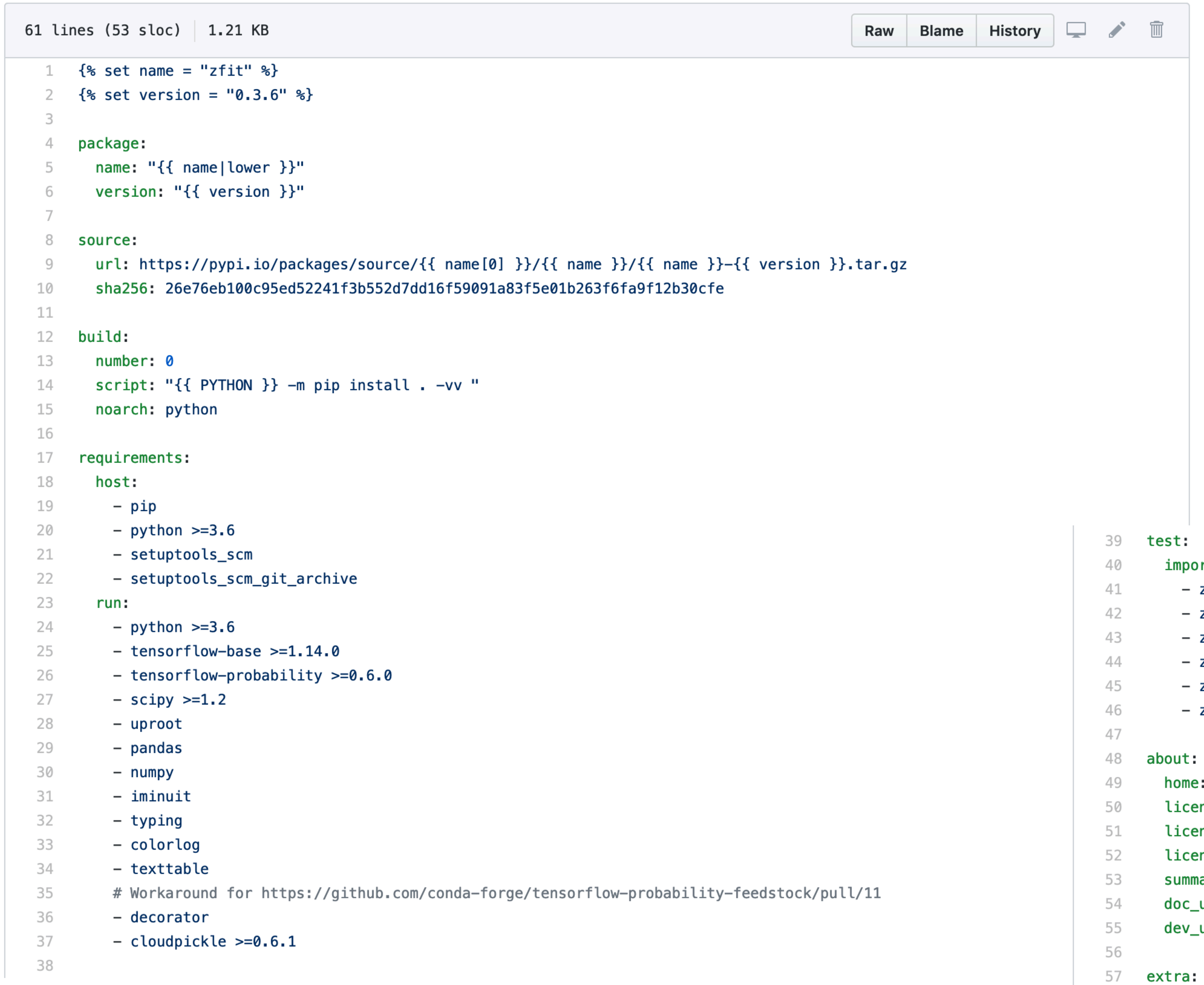
- ➤ DIRAC 7.2 (approaching end of life)
  - ➤ Support for Python 3 client installations (not default)
- ➤ DIRAC 7.3 (already available)
  - ➤ Support for Python 3 server installations
- ➤ DIRAC 8.0 (releasing very soon)
  - ➤ Python 3 only

- What's new?
  - dirac-install.py is deprecated and client installations are now much more flexible
  - Support for ppc64le and aarch64
- Anybody with an DIRAC extension should update it (ask for guidance)
  - And also add integration tests! (Even if you just run the DIRAC ones for now)

Questions?

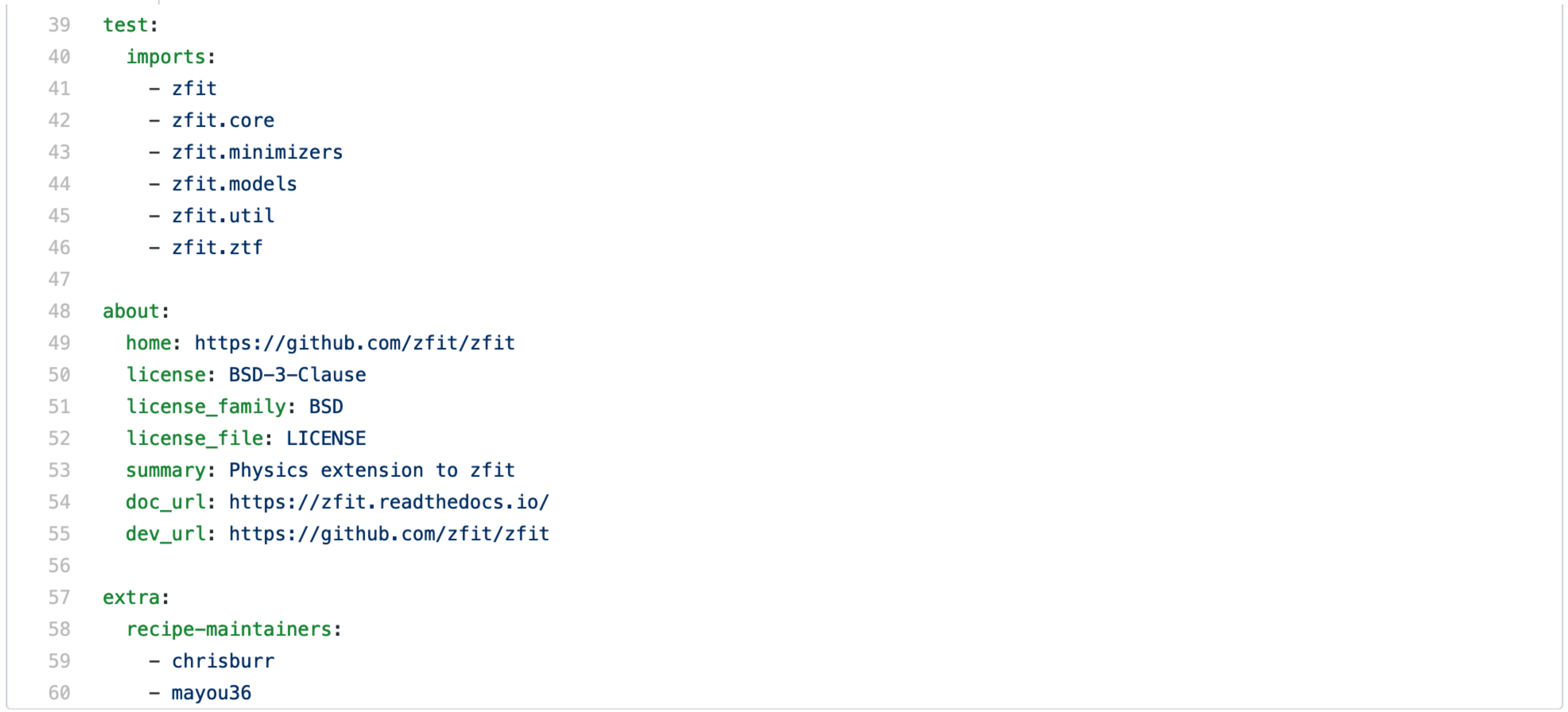
How does conda-forge work?

Adding a new package to conda-forge



61 lines (53 sloc) | 1.21 KB

```
{% set name = "zfit" %}
{% set version = "0.3.6" %}

package:
  name: "{{ name|lower }}"
  version: "{{ version }}"

source:
  url: https://pypi.io/packages/source/{{ name[0] }}/{{ name }}/{{ name }}-{{ version }}.tar.gz
  sha256: 26e76eb100c95ed52241f3b552d7dd16f59091a83f5e01b263f6fa9f12b30cfe

build:
  number: 0
  script: "{{ PYTHON }} -m pip install . -vv "
  noarch: python

requirements:
  host:
    - pip
    - python >=3.6
    - setuptools_scm
    - setuptools_scm_git_archive
  run:
    - python >=3.6
    - tensorflow-base >=1.14.0
    - tensorflow-probability >=0.6.0
    - scipy >=1.2
    - uproot
    - pandas
    - numpy
    - iminuit
    - typing
    - colorlog
    - texttable
    # Workaround for https://github.com/conda-forge/tensorflow-probability-feedstock/pull/11
    - decorator
    - cloudpickle >=0.6.1

```



```
test:
  imports:
    - zfit
    - zfit.core
    - zfit.minimizers
    - zfit.models
    - zfit.util
    - zfit.ztf

about:
  home: https://github.com/zfit/zfit
  license: BSD-3-Clause
  license_family: BSD
  license_file: LICENSE
  summary: Physics extension to zfit
  doc_url: https://zfit.readthedocs.io/
  dev_url: https://github.com/zfit/zfit

extra:
  recipe-maintainers:
    - chrisburr
    - mayou36
```

- Create a pull request against
  https://github.com/conda-forge/staged-recipes

- Can be mostly automated using
  `conda skeleton pypi zfit`

conda-forge / phasespace-feedstock

Sponsor | Unwatch 6 | Star 0 | Fork 1

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Security | Insights

phasespace v1.0.4 #1

Edit

Merged chrisburr merged 2 commits into `conda-forge:master` from `regro-cf-autotick-bot:1.0.4` 2 days ago

Conversation 1 | Commits 2 | Checks 2 | Files changed 2 | +3 −2

regro-cf-autotic... commented 2 days ago

It is very likely that the current package version for this
Notes for merging this PR:

1. Feel free to push to the bot's branch to update this
2. The bot will almost always only open one PR per ve
   Checklist before merging this PR:

- [ ] Dependencies have been updated if changed
- [ ] Tests have passed
- [ ] Updated license if changed and `license_file` is p

Note that the bot will stop issuing PRs if more than 3 Ve
open. If you don't want to package a particular version

If this PR was opened in error or needs to be updated p
The bot will close this PR and schedule another one.

4 recipe/meta.yaml — Viewed

```diff
@@ -1,13 +1,13 @@
 {% set name = "phasespace" %}
-{% set version = "1.0.3" %}
+{% set version = "1.0.4" %}

 package:
   name: "{{ name|lower }}"
   version: "{{ version }}"

 source:
   url: https://github.com/zfit/phasespace/archive/{{ version }}.tar.gz
-  sha256: ad1b322c1c47378ec6687c1ec30d5b92101c883e96bbebea345526b21596516e
+  sha256: 18e709a27111f96276aaa1f0df073e4cefc5e764ed9551de24b345aa3cc88790

 build:
   number: 0
```

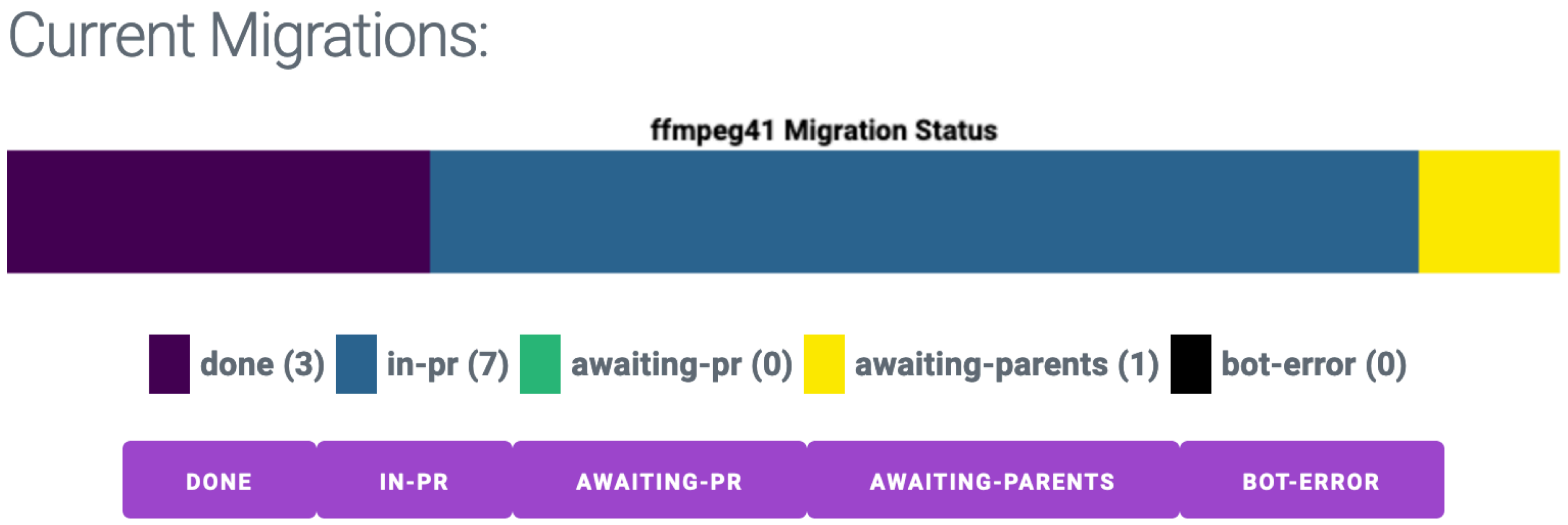
- Bots monitor for new releases
  - Even works with non-standard URLs
- Maintainers normally just have to click merge

Binary only distribution

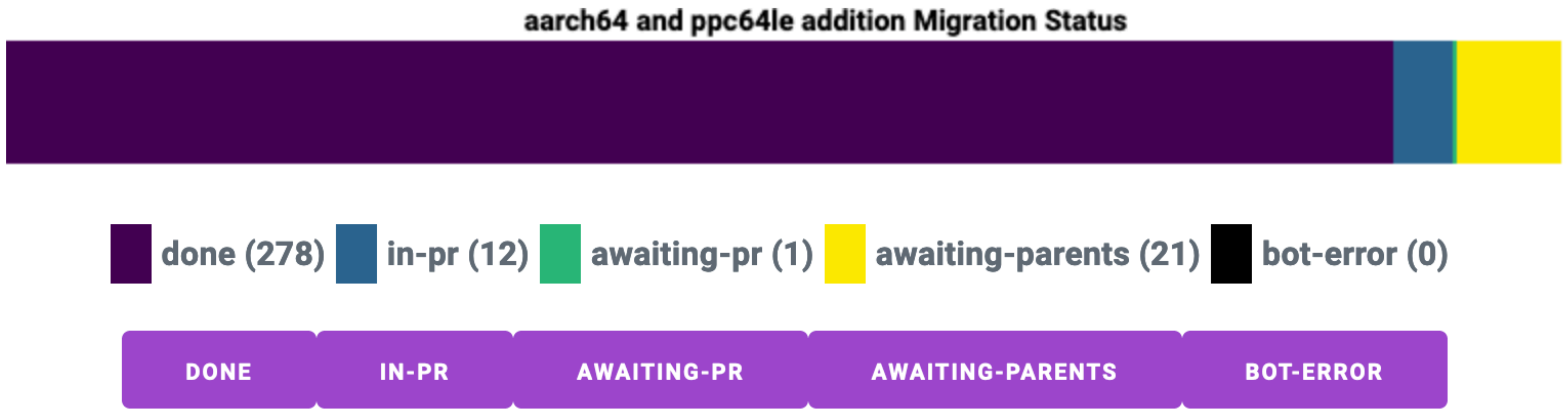
- Conda only supports installing binaries*
- Relies on the solver knowing about API/ABI compatibility
- Packages with shared libraries should specify what their ABI stability is
- Doesn't necessarily restrict what you can do
  - Variants can be used to provide a matrix of different builds
  - BLAS can be provided by netlib, mkl, blis and openblas
  - Several MPI variants
  - TensorFlow has CPU and (several) GPU variants

*Some organisations mirror the conda-forge build infrastructure for their own internal use

- What about when ABIs change? More 🤖!
- A line is added to a git repository
- Pull requests appear that rebuild packages in the correct order



- Migration is currently ongoing for ppc64le and aarch64 support
  - ROOT is included as a target



- Support for compiling CUDA with nvcc is rapidly maturing
  - Adds three additional additional targets (different driver versions)
  - Close to being fully supported by the conda-forge tooling
  - GPU variants of packages are already being added

How not to use pip

- Installing should be as simple as:

```
pip install PACKAGE_NAME
```

- Don’t use things that modify global state:

```
sudo pip install PACKAGE_NAME
```

  - Interacts poorly with system package managers
  - Can make it impossible to update or install packages using apt/yum/pacman/…

```
pip install --user PACKAGE_NAME
```

  - Normally has a higher priority in the Python search order
  - Can break other installations (e.g. use on lxplus can break your experiments software stack)

- **venv allow you to create environments from arbitrary Python installs**

Build infrastructure for conda-forge

- One repository per package ("feedstock")
- All packages are built using well known CI providers
- Currently mostly Azure Pipelines with Travis CI providing linux-ppc64le and linux-aarch64
- All managed by an external package: conda-smithy
  - Used to regenerate CI configuration for each update