EPJ Web of Conferences **295**, 04018 (2024) https://doi.org/10.1051/epjconf/202429504018
CHEP 2023

DIRAC

current, upcoming and planned capabilities and technologies

Federico Stagni[1,*], *Alexandre* Boyer[1,**], *Andrei* Tsaregorodtsev[2,***], *Andrii* Lytovchenko[****], *André* Sailer[1,†], *Christophe* Haen[1,‡], *Christopher* Burr[1,§], *Daniela* Bauer[3,¶], *Simon* Fayer[3,‖], *Janusz* Martyniak[3,**], and *Cédric* Serfon[4,††]

[1]CERN, EP Department, European Organization for Nuclear Research, Switzerland
[2]Aix Marseille University, CNRS/IN2P3, CPPM, Marseille, France
[3]Imperial college, London, UK
[4]Brookhaven National Laboratory, NY, USA

Abstract. DIRAC is the interware for building and operating large scale distributed computing systems. It is adopted by multiple collaborations from various scientific domains for implementing their computing models. DIRAC provides a framework and a rich set of ready-to-use services for Workload, Data and Production Management tasks of small, medium and large scientific communities having different computing requirements. The base functionality can be easily extended by custom components supporting community specific workflows. DIRAC is at the same time an aging project, and a new DiracX project is taking shape for replacing DIRAC in the long term. This contribution will highlight DIRAC's current, upcoming and planned capabilities and technologies, and how the transition to DiracX will take place. Examples include, but are not limited to, adoption of security tokens and interactions with Identity Provider services, integration of Clouds and High Performance Computers, interface with Rucio, improved monitoring and deployment procedures.

1 Introduction

DIRAC [1] is a software framework that enables communities to interact with distributed computing resources. It builds a layer between users and resources, hiding diversities across computing, storage, catalog, and queuing resources. DIRAC has been adopted by several HEP and non-HEP experiments' communities [2], with different goals, intents, resources and workflows: it is experiment agnostic, extensible, and flexible [3]. A single DIRAC service

*e-mail: federico.stagni@cern.ch
**e-mail: alexandre.franck.boyer@cern.ch
***e-mail: atsareg@in2p3.fr
****e-mail: lytovchenko@cppm.in2p3.fr
†e-mail: andre.philippe.sailer@cern.ch
‡e-mail: christophe.denis.haen@cern.ch
§e-mail: christopher.burr@cern.ch
¶e-mail: daniela.bauer@imperial.ac.uk
‖e-mail: simon.fayer05@imperial.ac.uk
**e-mail: janusz.martyniak@imperial.ac.uk
††e-mail: cserfon@bnl.gov

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

can provide a complete solution for the distributed computing of one, or multiple collaborations. The Workload Management System (WMS) provides a transparent, uniform interface for managing computing resources and complex workflows. The Data Management System offers several tools for data handling operations. DIRAC puts special emphasis on large scale data production and dataset management.

Under the DiracGrid umbrella we host, on GitHub[1], a few interconnected software projects, all released under the GPLv3 license. DiracGrid projects are publicly documented, and host discussions on GitHub. A yearly user workshop and weekly open developers meetings gather together users and experts. The project(s) consists of about six core programmers, and a dozen contributing developers. For simplicity and historical reasons, we will often refer to DiracGrid simply as DIRAC.

The DIRAC consortium was established in 2014 as a representing body for the development and maintenance of the DiracGrid software. The consortium comprises of a small set of active members, each of which nominates a representative, while consortium members elect every second year a Coordinator, and a Technical Coordinator. Institutes that are part of the consortium engage in the maintenance and in the promotion of the DiracGrid software.

This paper mainly highlights new DiracGrid capabilities and functionalities. For a more generic description of DIRAC as a whole please refer to [4].

This paper is organized as follows: section 2 briefly explains the history and future of DiracGrid; section 3 highlights the features of DIRAC major version v8.0; in section 4 the writers explain the future development steps; finally, section 5 gives some final remarks.

2 Brief history and future of DiracGrid projects

DIRAC was started around the year 2002 as an LHCb [5] project for running LHCb MonteCarlo productions. Originally written in `bash`, two years later "DIRAC2" was re-written in Python [6] for interfacing with EDG (European DataGRID) [7]. Between 2006 and 2007 the code went through a full refurbishment, and the DISET [8] protocol powering DIRAC's services was first created. This version was dubbed "DIRAC3" [9] and, in its core structure, is still the foundation of current DIRAC versions. In 2008 and 2009 a new large code reshuffling led to DIRAC becoming open source (its code was made available under open licence in 2009), and its first extensions were created. Further developments led to multi-VO support and several other adaptations [10].

DIRAC is an example of a project evolving from an experiment specific to a general-purpose one. The pilot based architecture of the WMS, first introduced by DIRAC, is now adopted by all the LHC experiments and multiple grid infrastructures. DIRAC is also a rare example of an efficient complex solution, with both WMS and DMS (Data Management System) at a scale. During the 20 years of the project lifetime it received contributions from more than a hundred developers, not including the specific extensions.

DiracGrid developers recognise that, in its current production incarnation, the project presents a few issues: it is complex, with high entrance bar, it is somewhat cumbersome to deploy, it is late on "standards", it has an old-ish design, and it is not very developer-friendly. For this and other technical reasons DiracGrid developers started, in 2023, a new project, dubbed DiracX[2]. DiracX is "the neXt DIRAC incarnation", and some of the features originally planned for DIRAC are currently being implemented in DiracX, and removed from DIRAC features list. The transition between DIRAC and DiracX will be handled with care and support will be offered to all existing DiracGrid installations.

[1] https://github.com/DIRACGrid
[2] https://github.com/DIRACGrid/diracx

DiracX will leverage existing battle-tested off-the-shelf technologies. DiracX wants to attract new users and developers, provide easy interoperability and standards-based interfaces. Stability is a must. Multi-VO support will be there from the get-go.

This paper does not enter into DiracX technical details, as some of them are still under discussions, but gives information about which functionalities will be present in DiracX that will not be implemented in DIRAC. It also explains, grosso modo, how we think the transition to DiracX will look like.

3 DIRAC v8.0 release

Release v8.0 of DIRAC is a major one. We devote this section to the changes added specifically to this release.

3.1 Transition to Python 3 and to standard Python packaging

Release v8.0 of DIRAC dropped the Python 2 compatibility. The DiracGrid package stack was based, for Python 2 and up to release v7r3 (the last release that was compatible with Python 2) on the custom stack depicted in figure 1. The installation of DIRAC and its related packages was done through a custom script (`dirac-install`) that was taking care of fetching the appropriate versions of the required software. Tarballs were created for every new version and hosted on project-specific web servers. The DIRACOS package [11] was used for collecting all packages upon which DIRAC client and server installations depended. DIRAC release deployments relied on the environment variable `$PYTHONPATH`, a possible cause of issues at specific sites. Extensions had to comply with the same way of working and creating releases, thus adding to the complexity.

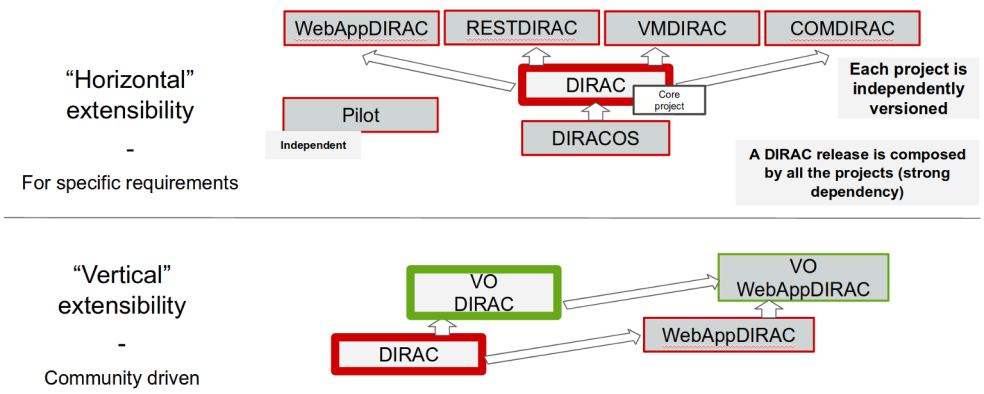


Figure 1: Python 2 DiracGrid software stack

For Python 3 compatible releases, DiracGrid started adopting the standard `pip` package manager. The PyPI[3] repository now hosts all DiracGrid packages, including extensions of DIRAC and other DIRAC-related Python packages. Since `Pip` is only for Python packages, and DIRAC installations make use of also non-Python packages, Python 3 releases make use of the DIRACOS2 installer, which in turn is based on the conda [12] binary package and environment manager. In parallel, some extension packages have been absorbed within the

[3]https://pypi.org/

DIRAC code, resulting in the Python 3 DiracGrid software stack shown in figure 2. A few other packages (not depicted in figure 2) have been instead extracted from DIRAC code, for ease of installation and sharing.

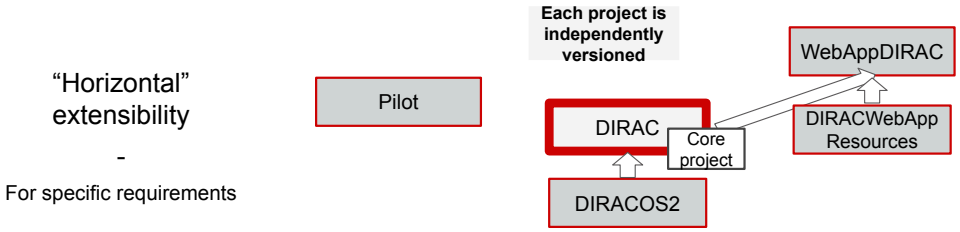


Figure 2: Python 3 DiracGrid software stack

The developers largely modernized the code with the help of automated tools for updating the python syntax. Support for platforms `ppc64le` and `aarch64` (in addition to the more common `x86_64`) have also been added. DIRACOS2 is at the time of writing (Q3 2023) based on Python 3.11.

3.2 Tokens support

Token-based authentication and authorization has been a de-facto industry security standard for several years now. Grid communities are pushed to migrate from X.509 certificates and proxies [13] to tokens, defined as JSON Web Tokens (JWT) [14] to be provisioned over OpenID Connect (OIDC) [15] and OAuth2 [16] workflows.

Commonly used Grid technologies like computing elements HTCondorCE [17] and ARC-CE [18] pushed versions of their software accepting tokens as well as X.509 certificates and proxies as an authentication and authorization base technology. HTCondor version 9.0 went as far as removing support for proxies in their pre-built binaries. Bodies like EGI (European Grid Infrastructure) and WLCG (Worldwide LHC Computing Grid) worked in parallel towards the adoption of such standards. Since July 2017, the WLCG Authorization WG[4] has been looking into how authentication and authorization technologies used on WLCG could evolve towards taking advantage of federated identities and standards commonly used in industry and academia, in particular through the use of JWT [19]. This resulted in recommendations and timeline documents [20].

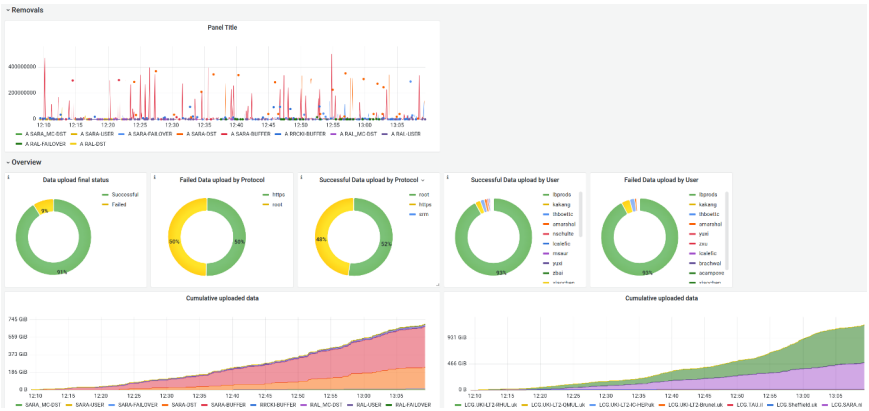
DIRAC version v8.0 rationalizes many aspects related to authentication, authorization, tokens and OAuth2 support. Specifically, it adds support for new Identity Providers (IAM [21] and EGI CheckIn [22]), and support for submitting pilots using access tokens obtained via a `client_credentials` flow from a token provider.

DIRAC will not significantly further expand the use of security tokens technologies: works in this direction will be done within DiracX.

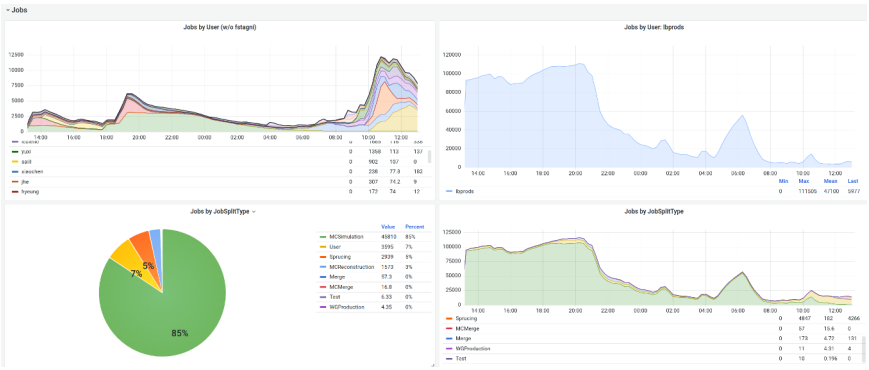
3.3 Monitoring with and for DIRAC

The monitoring of DIRAC is currently based on two back-ends: MySQL and ElasticSearch. MySQL is there mostly for historical reasons, as DIRAC's own accounting system fully relies on it. ElasticSearch (and OpenSearch, as long as these two will be compatible one with

[4]https://twiki.cern.ch/twiki/bin/view/LCG/WLCGAuthorizationWG



(a) LHCb DIRAC's DMS operations visualized through grafana dashboard



(b) LHCb DIRAC's WMS operations visualized through grafana dashboard

Figure 3: Grafana dashboards for monitoring DIRAC WMS and DMS operations

the other) has instead been the back-end of choice for what some time ago DIRAC developers called the monitoring system. Both of them are tightly coupled with the DIRAC web framework, which provides a visualization and plotting interface based on Matplotlib [23].

On one side, we recognise that DIRAC's MySQL accounting back-end is highly custom and difficult to maintain and/or visualise (hence the DIRAC's Web accounting application). On the other side, adding to the existing DIRAC's plotting capabilities is tedious and time consuming. At present some projects (especially Grafana [24]) offer highly usable visualisation tools, for data stored on several back-ends. Based on these considerations, DIRAC v8 adds to the data types monitor-able through an Elastic/OpenSearch back-end. The list includes, but is not limited to, Pilots and Data Operations monitoring. DIRAC v8 also adds definitions of dashboards for Kibana and Grafana.

In figure 3 we show real-life examples of using Grafana dashboards for monitoring the LHCb DIRAC production setup.

3.4 DIRAC and Rucio

DIRAC provides, since long time, a File Catalog plugin for transparently interfacing DIRAC Data Management with Rucio's catalog. Synchronization agents are there to make sure that the configuration is reflected between the two. DIRAC and Rucio are used in production by the Belle2 collaboration, with at least two more communities looking to adopt a similar schema. Developments in this area include support for Rucio metadata. DiracX and Rucio plan to provide flawless communication through inter operable tokens.

4 Future releases, and the road to DiracX

DiracX is a complete rewrite of the DIRAC code, with DiracX having DIRAC as one of its dependencies. DiracX comes with a new deployment model, and a new Web application. DiracX shares with DIRAC its databases (MySQL, Elastic/OpenSearch), making the transition from DIRAC to DiracX easy to handle. Once a DiracX service is ready, a legacy adaptor is added to transition from DIRAC service to a DiracX one. Integration tests are already running. Communities currently using extensions of DiracGrid projects will need to code an extension to DiracX, and add similar legacy adaptors to their services (if any).

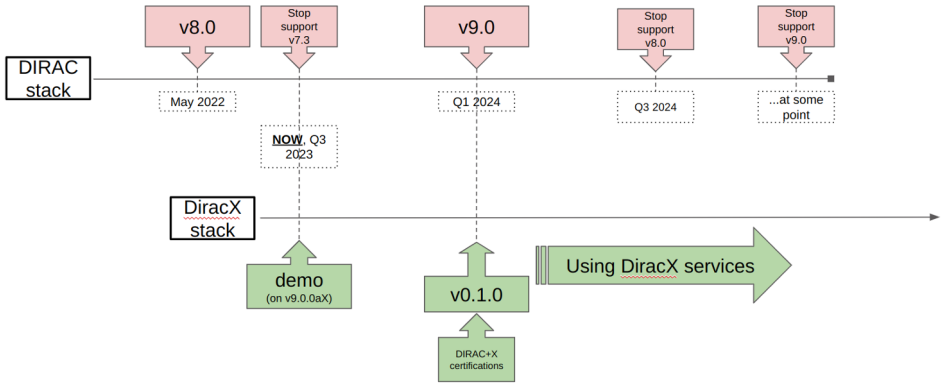


Figure 4: DIRAC and DiracX timeline. The two will need to co-exist for a long while.

At the time of writing the current production release of DIRAC is v8.0. We think that DIRAC v9.0 might be the last in the series of centrally supported DIRAC releases, or, at a minimum, the first through which a transition to DiracX will be possible. The DIRAC v9.0 release will provide several simplifications that will lead to an easier transition to DiracX. The DiracGrid Pilot will be adapted to be able to install and work also with a DiracX server.

The first of the DiracX releases (due in Q1 2024) will work together (in addition to) DIRAC v9, and will offer partial support for WMS functionalities. It will expose REST-like Web APIs and native tokens support. For its internal communications DiracX will issue its own JWT tokens and will not support X.509 proxies. Communications with those Grid resources still requiring X.509 proxies will be done through DIRAC adapter.

5 Summary and conclusions

DiracGrid components provide several functionalities (including a full-capable web portal) that are not expanded within this paper. If readers want to know more about it, DIRAC

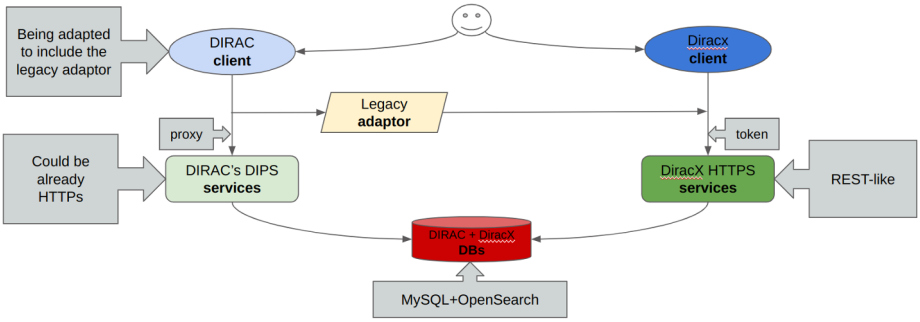


Figure 5: DiracX services will be able to read and interact with DIRAC databases. Legacy adaptors are already being developed to move from DIRAC to DiracX services.

developers and consortium members maintain a quite large documentation, which can be found in the official DIRAC documentation [25]. Users workshops are held once a year, developers meetings and hackathons are in alternating weeks. DIRAC developers hold in high regards testing and automation, as well as using de-facto technology standards.

This paper highlighted the main news about DiracGrid developments. The DIRAC line of developments introduced with DIRAC major release v8.0 three main pillars:

- Python 3 support, and dropping of Python 2 support;
- support of Identity Providers and specific usage of Tokens-based authorizations;
- expansion of monitoring capabilities.

DIRAC v8.0 ensures the possibility of fully exploiting today's grid software stack. The new DiracX project represents, from the technological point of view a break-point with respect to what DIRAC is and has been. At the same time, from a functionalities point of view we see it as natural successor of DIRAC. We are confident that current DIRAC users will be DiracX users, and that new users and developers will join the efforts of DiracX.

References

[1] A. Tsaregorodtsev, F. Stagni, P. Charpentier, A. Sailer, M.U. Garcia, K.D. Ciba, C. Haen, C. Burr, A. Lytovchenko, R. Graciani et al., *Diracgrid/dirac: v8.0.27* (2023), `https://doi.org/10.5281/zenodo.8249063`

[2] F. Stagni, A. Tsaregorodtsev, L. Arrabito, A. Sailer, T. Hara, X. Zhang, Journal of Physics: Conference Series **898**, 092020 (2017)

[3] S. Camarasu-Pop et al., *Exploiting GPUs on distributed infrastructures for medical imaging applications with VIP and DIRAC*, in *42nd international convention on information and communication technology, electronics and microelectronics (MIPRO 2019) Opatija, Croatia, May 20-24, 2019* (2019), pp. 190–195

[4] F. Stagni, A. Tsaregorodtsev, A. Sailer, C. Haen, EPJ Web Conf. **245**, 03035 (2020)

[5] The LHCb Collaboration, Journal of Instrumentation **3**, S08005 (2008)

[6] A. Tsaregorodtsev, V. Garonne, J. Closier, M. Frank, C. Gaspar, E. van Herwijnen, F. Loverre, S. Ponce, R. Graciani Diaz, D. Galli et al., eConf C **0303241**, TUAT006 (2003)

[7] B. Segal, L. Robertson, F. Gagliardi, F. Carminati, *Grid computing: the European Data Grid Project* (2000), `https://cds.cern.ch/record/560415`

[8] A. Casajús, R. Graciani Diaz, Journal of Physics: Conference Series **219**, 042033 (2010)
[9] A. Tsaregorodtsev, M. Bargiotti, N. Brook, A.C. Ramo, G. Castellani, P. Charpentier, C. Cioffi, J. Closier, R.G. Díaz, G. Kuznetsov et al., J. Phys.: Conf. Ser. **119**, 062048 (2008)
[10] A.F. Boyer, C. Haen, F. Stagni, D.R.C. Hill, Future Gener. Comput. Syst. **133**, 23 (2022)
[11] M. Petrič, C. Haen, B. Couturier, *DIRACOS: a cross platform solution for grid tools*, in *EPJ Web of Conferences* (EDP Sciences, 2020), Vol. 245
[12] conda-forge community, *The conda-forge Project: Community-based Software Distribution Built on the conda Package Format and Ecosystem* (2015), https://doi.org/10.5281/zenodo.4774217
[13] V. Welch, M. Thompson, D.E. Engert, S. Tuecke, L. Pearlman, *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*, RFC 3820 (2004), https://www.rfc-editor.org/info/rfc3820
[14] M.B. Jones, J. Bradley, N. Sakimura, *JSON Web Token (JWT)*, RFC 7519 (2015), https://www.rfc-editor.org/info/rfc7519
[15] OpenID Connect, *OpenID Connect*, https://openid.net/specs/openid-connect-core-1_0.html
[16] D. Hardt, *The OAuth 2.0 Authorization Framework*, RFC 6749 (2012), https://www.rfc-editor.org/info/rfc6749
[17] HTCondor Team, *Htcondor* (2023), https://doi.org/10.5281/zenodo.8230603
[18] Ould-Saada, Farid, *Arc computing element system administrator guide* (2013), https://doi.org/10.5281/zenodo.6809
[19] B. Bockelman, A. Ceccanti, I. Collier, L. Cornwall, T. Dack, J. Guenther, M. Lassnig, M. Litmaath, P. Millar, M. Sallé et al., *WLCG Authorisation from X.509 to Tokens* (2020), Vol. 245, p. 03001, 2007.03602, https://cds.cern.ch/record/2751236
[20] WLCG Authorization Working Group, *WLCG Token Transition Timeline* (2022), https://doi.org/10.5281/zenodo.7014668
[21] A. Ceccanti, E. Vianello, M. Caberletti, R. Miccoli, W. Furnell, F. Agostini, F. Giacomini, T. Dack, M. Vilaca, *indigo-iam/iam: INDIGO Identity and Access Management Service v1.8.1p1* (2023), https://doi.org/10.5281/zenodo.8113321
[22] G. La Rocca, *EGI e-infrastructure - Advanced computing services for science* (2023), https://doi.org/10.5281/zenodo.7925603
[23] J.D. Hunter, Computing in Science & Engineering **9**, 90 (2007)
[24] Grafana Labs, *Grafana documentation* (2018), https://grafana.com/docs/
[25] DIRAC, *Dirac documentation*, http://dirac.readthedocs.io/