O.10 Building and installing ABINIT: a matter of collaborations

Yann Pouillon[1], Jean-Michel Beuken[2], Marc Torrent[3], Matteo Giantomassi[2], ESL Contributors[4] and EasyBuild Contributors[5]

[1] *Simune Atomistics, Donostia-San Sebastián, Spain*
[2] *Université Catholique de Louvain, Louvain-la-Neuve, Belgium*
[3] *CEA, DAM, DIF, F-91297 Arpajon, France, and Université Paris-Saclay, CEA, Laboratoire Matière en Conditions Extrêmes, 91680 Bruyères-le-Châtel, France.*
[4] *Too many to cite*
[5] *Too many to cite*

DFT software is undergoing a slow and comprehensive standardisation process, implying an increasing modularisation of the developed components, the rationalisation of their multiple interactions, as well as constant improvements in the automation and control of their build. At the same time, concepts like *Continuous Integration* and *DevOps* are progressively becoming familiar within the community.

This evolution lowers the barrier to access a myriad of use cases, from using ABINIT as a force-calculation engine in complex distributed workflows, to making it a benchmark reference when optimising pseudopotentials and basis sets for SIESTA, without forgetting calculations on large systems with wavelets provided by BigDFT. It also increases the level of confidence one can have when using all these software packages in production.

The common denominator to this expanding world of possibilities is collaboration. Better said: a multi-directional and coordinated set of collaborations. After a quick tour of these collaborations, we will explain how they influence the design of the ABINIT build system, the structure of the source code, and some of the challenges they bring to both users and developers.