



Project Number 675728

## D2.2 – First Release of Workflow Blocks and Portals

*WP2: Portable Environments for Computing and Data Resources*



Copyright© 2015-2018 The partners of the BioExcel Consortium



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

## Document Information

<b>Deliverable Number</b>	D2.2
<b>Deliverable Name</b>	First Release of Workflow Blocks and Portals
<b>Due Date</b>	2016-12-31 (PM14)
<b>Deliverable Lead</b>	IRB
<b>Authors</b>	Adam Hospital (IRB), Anna Montras (IRB), Josep Lluís Gelpí (BSC), Daniele Lezzi (BSC), Rosa M. Badia (BSC), Steven Newhouse (EMBL-EBI), Jose A. Dienes (EMBL-EBI), Pau Andrio (BSC), Stian Soiland-Reyes (UNIMAN), Darren J. White (UEDIN), Adam Carter (UEDIN), Emiliano Ippoliti (Juelich), Adrien Melquiond (UU), Bert de Groot (MPG)
<b>Keywords</b>	Cloud Computing, HPC, Tools, Workflows
<b>WP</b>	WP2
<b>Nature</b>	Report
<b>Dissemination Level</b>	Public
<b>Final Version Date</b>	2016-12-30
<b>Reviewed by</b>	Erwin Laure (KTH), Bert de Groot (MPG), Alexandre Bonvin (UU)
<b>MGT Board Approval</b>	2016-12-29

## Document History

<b>Partner</b>	<b>Date</b>	<b>Comments</b>	<b>Version</b>
IRB	2016-12-03	First draft	0.1
BSC	2016-12-12	BSC infrastructure added	0.2
UEDIN	2016-12-19	Content for UC1 added.	0.3
IRB	2016-12-22	BSC, UNIMAN, Juelich, EMBL-EBI, UEDIN, UU and MPG partners contributions added	0.4
IRB	2016-12-23	First D2.2 draft distributed for review	0.5
IRB	2016-12-29	Comments from the PMB review addressed Conclusions & Future Work section added	0.6

## Executive Summary

This deliverable presents a description of the workflow's development process that will be followed by BioExcel partners, from the development, test and verification of the workflows to the final deployment and benchmarking. Software practices behind this process are also presented, demonstrating the strong connection of BioExcel with the Elixir bioinformatics infrastructure, and in particular, with the tools and interoperability platforms.

The current state of development of the computational infrastructures designed to store and deploy computing and data resources of BioExcel is also presented. Two main infrastructures, a development one (BSC) and a production one (EMBL-EBI) have been already set up, and are described in this document.

The first workflow prototype designed and implemented in BioExcel (*Model Protein Mutants*) is introduced. Steps followed during the development of this prototype are described and linked to the workflow's development process presented in the first part of the document. An overview of the different workflows proposed from the project pilot use cases and presented in the D2.1 is also exposed.

## Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>6</b>
<b>2</b>	<b>WORKFLOWS DEVELOPMENT</b>	<b>7</b>
2.1	WORKFLOW'S DEVELOPMENT PROCESS	7
2.1.1	DEVELOPMENT, TEST & VERIFICATION	7
2.1.2	ACCEPTANCE AND DEPLOYMENT	7
2.1.3	BENCHMARKING	7
2.2	WORKFLOW'S BUILDING BLOCKS INTEROPERABILITY	8
2.2.1	AUTOMATIC DISCOVERY (REGISTER)	8
2.2.2	MODULARITY (INTEROPERABILITY)	9
2.2.3	DESCRIPTION AND PROVENANCE	10
<b>3</b>	<b>BIOEXCEL COMPUTATIONAL INFRASTRUCTURES AND PORTAL</b>	<b>13</b>
3.1	BSC CLOUD INFRASTRUCTURE (DEVELOPMENT)	13
3.2	EMBL-EBI BioExcel PORTAL (PRODUCTION)	14
3.2.1	SINGLE SIGN ON	15
3.2.2	SERVICE DEFINITION AND DEPLOYMENT	16
<b>4</b>	<b>BIOEXCEL WORKFLOW BLOCKS</b>	<b>19</b>
4.1	TRANSVERSAL WORKFLOWS: MODEL PROTEIN MUTANTS AS A CASE EXAMPLE	19
4.1.1	MODEL PROTEIN MUTANTS: OVERVIEW	19
4.1.2	MODEL PROTEIN MUTANTS: RUN	22
4.1.3	MODEL PROTEIN MUTANTS: BENCHMARKING	25
4.1.4	MODEL PROTEIN MUTANTS: DEPLOYMENT	25
4.1.5	MODEL PROTEIN MUTANTS: TESTING & VERIFICATION	27
4.2	PILOT USE CASES WORKFLOWS	27
4.2.1	PILOT USE CASE 1: GENOMICS	28
4.2.2	PILOT USE CASE 2: FREE ENERGY SIMULATIONS OF BIOMOLECULAR COMPLEXES	30
4.2.3	PILOT USE CASE 3: MULTI-SCALE MODELING OF MOLECULAR BASIS FOR ODOR AND TASTE	32
4.2.4	PILOT USE CASE 4: BIOMOLECULAR RECOGNITION	33
4.2.5	PILOT USE CASE 5: VIRTUAL SCREENING	34
<b>5</b>	<b>CONCLUSIONS &amp; FUTURE WORK</b>	<b>36</b>
<b>6</b>	<b>REFERENCES</b>	<b>37</b>

## 1 Introduction

The roadmap for the initial setup of the BioExcel infrastructure consists of the deployment of a set of common software blocks to perform most commonly demanded operations, as gathered from the Use Case analysis (see D2.1). This is based in a bottom-up building approach starting from the individual operations already available to lead to “transversal workflow units”, higher level operations that were considered general needs for the different use cases. Although the basic functionality is available and stable, building such units requires solving interoperability issues, deploying them in the selected compute infrastructures, and eventually setting them up with the most appropriate workflow managers. To this end, the Cloud infrastructure available at the Barcelona Supercomputing Center (BSC) is a first logical solution (see section 3.1 for technical details). BSC’s cloud is being used for initial deployment, verification, and testing, before the tools are made available through the BioExcel’s Portal (section 3.2). At the main portal, users will be able to access production resources from within the project consortium (e.g. cloud or HPC), community resources (e.g. PRACE or Elixir Compute Platform) or commercial cloud resources (e.g. Amazon Web Services, Google Cloud Platform, Microsoft Azure). Additionally, proof-of-concept complete workflows, like the generation of a MD ensemble for a series of known protein variants, or virtual-screening analysis, are being deployed. Such complete workflows will serve as demonstrators of the capabilities of the software architecture and also as a guide for users when building new functionalities. Lessons learned during this initial roadmap, will be applied during the integration of the remaining BioExcel tools.

## 2 Workflows development

One of the main duties of BioExcel WP2 is to provide easy access to computing and data resources through a range of workflow environments. The first step in this direction was a thorough analysis of the state-of-the art of portable environments for computing (D2.1). In this study, a set of workflow managers, computational infrastructures and data resources were identified. Besides, an extensive catalogue of tools supported by BioExcel partners were also collected and proposed as the main units (building blocks) to be used in the development of workflows for our pilot use cases. A set of pipelines were described and analyzed, focusing on the building blocks needed, interoperability between them, and possible technological gaps found in their development.

The next step towards our goal is the generation of the first workflow prototypes (transversal workflow units, T2.2) following this analysis.

### 2.1 Workflow's Development Process

The software development process of biomolecular research workflows can be divided into four main steps:

#### 2.1.1 Development, Test & Verification

The workflow is assembled, tested and verified in the development computational infrastructure (BSC testbed, see section 3.1). High-level tests and data set examples to verify the correctness of the workflow results will be produced together with the pipeline code (T2.3). The Common Workflow Language [1] (CWL) will be used to describe workflows, and standard to follow data provenance (PROV; <https://www.w3.org/TR/prov-overview>) will be taken into account.

#### 2.1.2 Acceptance and Deployment

Once the workflow is accepted, it will be made accessible on the production computational infrastructure through the BioExcel Portal (EMBL-EBI final portal, see section 3.2). Depending on the portable environment for computing used (VM, Docker container, HPC, etc.; see D2.1) different technologies will be applied (see section 3.2.2) and accessed through the BioExcel Portal at EMBL-EBI.

#### 2.1.3 Benchmarking

The final step in the workflow development will be a benchmarking to measure the performance. Benchmarking will be mainly run in the production phase, although it could be occasionally run in the development phase, especially when the workflow is run in HPC

environments, as the BSC infrastructure is directly coupled to the MareNostrum supercomputer.

## 2.2 Workflow's Building Blocks Interoperability

Although the catalogue of tools collected in D2.1 is extensive and covers a large part of the computational biomolecular field, most of the individual tools are not ready to be used in an interconnected complex pipeline. A fully interoperable scenario requires a number of components: Common data ontologies providing both machine-readable data type definitions and semantics; service registries allowing easy discovery; and a process management software. BioExcel will follow here the initiatives from Elixir[2], with projects like EDAM ontology[3] (a product of the effort made in the EMBRACE project[4]), BioXSD[5], or the Elixir Tools Catalogue (bio.tools)[6, 7] that seek to configure a fully interoperable environment for Bioinformatics.

For this, the BioExcel project will be presented as a use case in the Elixir Excelerate Interoperability Platform where it has already contributed to *Task 5.3: "Bring Your Own Data (BYOD) & Capacity Building Workshops"* with the organization within the BioExcel WP2 of the first *Bring Your Own Workflow (BYOW)* workshop in Barcelona (BioExcel: workflow training for computational biomolecular research, 20-21 October 2016), where experts in the field of Workflow Managers were joined together to share their expertise with researchers.

The different steps designed to follow this initiative, which are being currently tested in the first workflow prototype (see section 4.1), are described in the next sections.

### 2.2.1 Automatic discovery (register)

The state-of-the-art technology that BioExcel is planning to use in its final portal (see section 3.2) requires an automatic discovery and retrieval of tools to be installed in the deployed computational infrastructure. Elixir's tools and data services registry bio.tools is the repository chosen to register our list of building blocks collected in D2.1. bio.tools uses a controlled vocabulary to describe input and output data for the tools registered: EDAM (EMBRACE Data and Methods)[3]. EDAM is a simple ontology of well established, familiar concepts that are prevalent within bioinformatics, including types of data and data identifiers, data formats, operations and topics. EDAM provides a set of terms with synonyms and definitions organized into an intuitive hierarchy for convenient use.

The first step towards having all the information needed in bio.tools is registering our catalogue of tools in the repository, identifying them by a BioExcel tag. That first step was already done, and the list of BioExcel tools can be easily retrieved from bio.tools: <https://bio.tools/?q=bioexcel> (currently 34 entries within the BioExcel collection are registered). Next step is the curation of

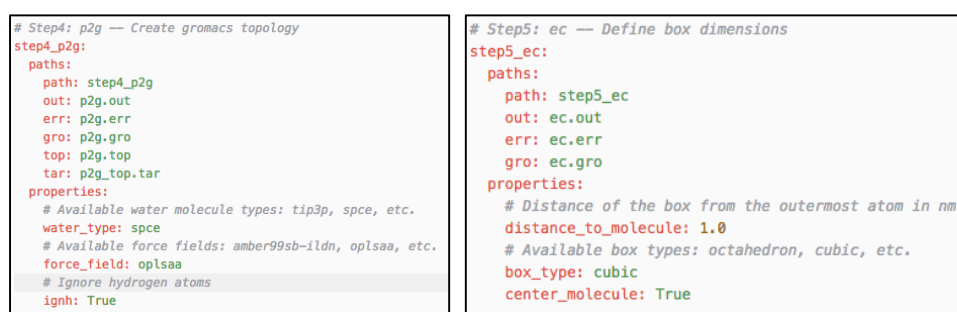


the EDAM ontology representing the input and output data of our tools. There is a large bias in the EDAM ontology towards genomics data. In order to have a good description of data accepted and generated by our tools, EDAM needs therefore to be enriched and updated. Ontology data describing macromolecular structure, flexibility and dynamics will be integrated in EDAM during the next year. This is another tight link between Elixir and BioExcel, where our work will have a great impact on the Elixir Tools platform.

### 2.2.2 Modularity (interoperability)

The software model designed to be used in BioExcel workflow components consists of a series of building blocks organized as a library of modules, encapsulating the necessary functionalities. These modules will be generated as configurable Python modules wrapping the original software. Interaction with the underlying software will be managed through command line execution, or, when appropriate, through a specific Python API provided by the software. This ensures that the original software can be kept untouched, minimizing installation and configuration issues. Besides, parallelization strategies already available in such applications can also be used when appropriate. In general terms, wrappers will expose tasks and their dependencies, such that the underlying computational infrastructure can optimize their execution. Our task-based strategy for parallelism is commonly used in a number of runtime environments for high-performance computing (for example, see the RADICAL Pilot project[8, 9] or the Extasy project[10]).

Configuration of the modules will be made through YAML/JSON scripts (see an example in Figure 2.1). Wrappers will take care of interpreting configuration scripts and translate their settings to the execution command line. This configuration strategy will allow maintaining a stable interface even in the eventual modification of the internal applications, and will hide this complexity to the users. Default configuration schemas will be provided, in a way that non-expert users could execute software with a set of recommended, default settings.



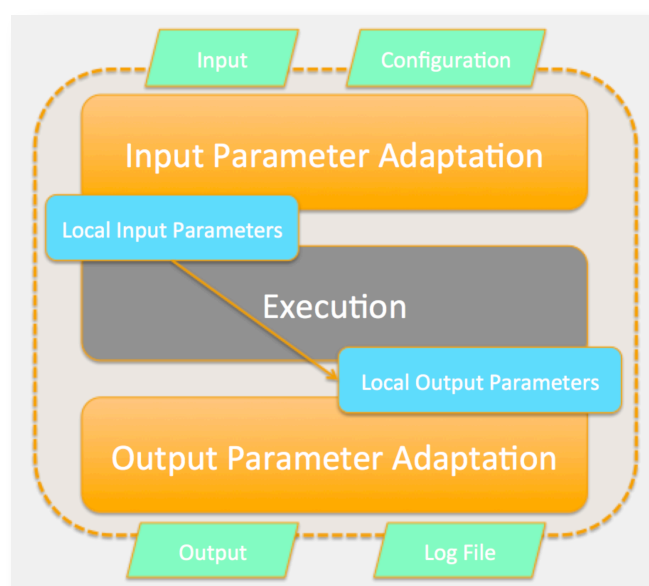
```
# Step4: p2g — Create gromacs topology
step4_p2g:
  paths:
    path: step4_p2g
    out: p2g.out
    err: p2g.err
    gro: p2g.gro
    top: p2g.top
    tar: p2g_top.tar
  properties:
    # Available water molecule types: tip3p, spce, etc.
    water_type: spce
    # Available force fields: amber99sb-ildn, oplsa, etc.
    force_field: oplsa
    # Ignore hydrogen atoms
    ignh: True

# Step5: ec — Define box dimensions
step5_ec:
  paths:
    path: step5_ec
    out: ec.out
    err: ec.err
    gro: ec.gro
  properties:
    # Distance of the box from the outermost atom in nm.
    distance_to_molecule: 1.0
    # Available box types: octahedron, cubic, etc.
    box_type: cubic
    center_molecule: True
```

**Fig. 2.1 – Example of configuration files in BioExcel workflow's components**

The functionality of software blocks should be kept to single operations, typically with a minimum set of input data items (input data + configuration data), and a single output data item (always accompanied by a log data item),

thus maximizing the modularity, flexibility and interoperability. Input data will be internally converted into local input parameters. The tool will then be executed with these input parameters, and local output data obtained will be similarly transformed to interoperable output data items (see Fig 2.2). Moreover, it can be foreseen that some sets of operations will be usually performed as a block. In those cases, it is reasonable to build higher-level blocks, including a more complex pipeline, made itself from the combination of simpler blocks. These pipelines will be organized in the same way and will offer a similar interface and configuration strategies than the simpler blocks.



**Fig. 2.2 – Modular scheme used for the design of BioExcel workflow's components**

The use of these Python modules allows the definition of any complex workflow just as a Python script that will call component modules as internal tasks. Workflows could be configured using the same YAML/JSON procedure as with the modules, in a way that a single file will manage the configuration of the individual modules and the complete workflow. Also, thanks to this modular software model, single components and entire workflows can be controlled with different workflow managers, as the model is completely workflow manager agnostic. A first test in this direction has been done with the Model Protein Mutants workflow prototype, using PyCOMPSs[11] and Galaxy[12] workflow managers (see section 4.1.2).

### 2.2.3 Description and Provenance

The Common Workflow Language (CWL)[1] is a specification for describing analysis workflows and tools that are portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high performance computing (HPC) environments. CWL is an

independent community-led effort, with implementations being developed for more than 9 workflow engines, 3 of which are already meeting conformance tests.

CWL is designed to meet the needs of data-intensive science, such as Bioinformatics, Medical Imaging, Astronomy, Physics, and Chemistry, with a particular focus on pipelines of command line tools executed on cloud and cluster infrastructure. While CWL recommends using Docker to containerize and distribute the executable tools in the corresponding version, it also supports other ways to identify and install tools, as is needed in HPC systems.

The CWL community is also actively participating in the GA4GH Containers and Workflows team – GA4GH (Global Alliance for Genomics and Health) brings together over 400 leading institutions working in healthcare, research, disease advocacy, life science, and information technology (<https://genomicsandhealth.org/>).

CWL is being adopted by the Elixir project as the standard language to describe workflows, and BioExcel wants to contribute to this by describing all of our workflows with CWL.

From a point of view of BioExcel users, the benefits of CWL interoperability are twofold: Firstly, CWL provides a structural description of command line tools (e.g. input/output file types, parameters and options) that are reusable across workflow engines. It is therefore that BioExcel has pushed <http://bio.tools/> to support direct links to CWL tool descriptions, which enable programmatic discovery and usage of such tools in a variety of systems.

Work in the CWL community includes exposing existing Galaxy tools as CWL; the Galaxy Tool Shed <https://toolshed.g2.bx.psu.edu/> already contains ~4300 tool descriptions in the bioinformatics domain. The addition of BioExcel's CWL descriptions of biomolecular tools therefore will fit into a large ecosystem and could be utilized in a multitude of workflow systems.

Secondly, as CWL workflows are portable across multiple workflow engines, BioExcel-produced software block workflows and use case workflows could be used in multiple workflow managers. This should enable flexibility for consumers to choose their preferred workflow engine based on their infrastructure, computational needs and expertise.

In December 2016, in collaboration with the Elixir Accelerate project, we had a two-day meeting with CWL community manager Michael Crusoe, the EMBL-EBI Protein Families team led by Rob Finn, the Ensembl team, as well as MG-RAST developers led by Folker Meyer from Argonne Labs. This highlighted the requirement to run the same CWL workflow on multiple engines and a mix of in-house and public cloud compute infrastructure, even crossing institutional barriers.

While scientific workflows primarily aid systematic execution and coordination of computational analysis and data transformation, they also

function as structured documentation and a historic trace of how new results have been generated and with which tools. This is important for publishing, archival and verifiability purposes, but also for reuse and human understanding of scientific methods[13].

To that end, BioExcel will keep a focus on provenance of workflow executions, utilizing our existing work on W3C PROV (<https://www.w3.org/TR/prov-overview>), Research Objects (<http://researchobject.org/>) and workflow provenance of large, complex datasets[14], while migrating towards using the Common Workflow Language as an executable workflow trace, Docker for tool containerization and bio.tools identifiers of workflows and tools.

We have actively collaborated with the US Food and Drug Administration (FDA), which have shown strong interest in using CWL and Research Objects as part of NIH BioCompute objects for tracking provenance of workflow executions in personalized medicine between healthcare providers across the US. Our engagement with the George Washington HIVE team (led by Raja Mazumder) and the High-throughput Sequencing Computational Standards for Regulatory Sciences (HTS-CSRS) project (<https://hive.biochemistry.gwu.edu/htscsrs>) means we have been invited to present at the FDA HTS-CSRS workshop in 2017. (<https://hive.biochemistry.gwu.edu/htscsrs/workshop>).

This work is overlapping with our engagement with developers of the Galaxy workflow system (Bjorn Grüning, Freiburg and Jeremy Goecks, George Washington), who, beyond the Galaxy CWL integration, are also keen to generate and share portable workflow provenance using Research Objects. We will therefore pursue this effort further in 2017.

### 3 BioExcel Computational Infrastructures and Portal

As stated in the D2.1 roadmap, BSC's cloud will be used for initial deployment, verification, and testing, before being transferred and made available at the BioExcel Portal (EMBL-EBI). The BioExcel Portal will provide access to computational infrastructures both within the project (extensively described in the following sections) and resources accessible to partners outside the project.

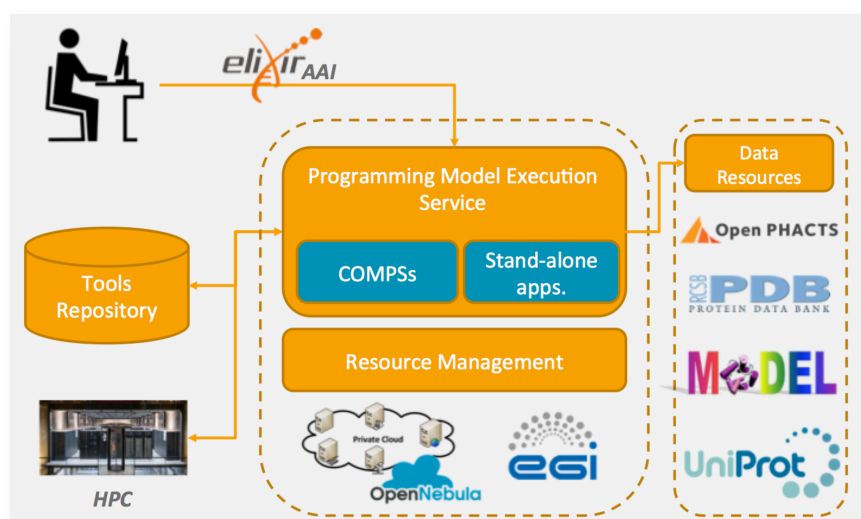
#### 3.1 BSC Cloud Infrastructure (development)

BSC hosts a cloud infrastructure designed to perform small-scale analysis, and to develop advanced workflow management (Fig. 3.1). The main characteristics of the platform are:

1. A virtualization system, based on *OpenNebula*, to control the underlying hardware infrastructure. Applications are run in virtual machines (VMs) that are instantiated dynamically following the requirements of the analysis workflow.
2. Workflows are defined in the syntax of the COMPSs programming model. COMPSs is able to discover implicit parallelism in the pipelines, and hence, execute otherwise serial operations with an optimal use of a parallel environment. COMPSs workflows can be defined using Java, C++, or Python. The Python binding of COMPSs (PyCOMPSs) is adapted to the architecture of software building blocks developed in BioExcel (see section 2.2). COMPSs has been adapted to control the virtualization layer, making it transparent to the user, and also allowing to execute the same workflow in a series of environments, from single workstations, to HPC or grid/cloud facilities.
3. Applications where the use of COMPSs would not be advisable can be also executed in their native environment, exploiting already existing parallelism if available.
4. Complex applications are stored in the system as a collection of pre-packed virtual machines that include the application itself and the necessary software environment. Virtual machines developed here are fully compatible with most common cloud infrastructures, as the EGI FedCloud, and could be accessed through the BioExcel Portal and deployed automatically in a cloud environment through EMBL-EBI's Cloud Portal being developed within BioExcel.
5. Access to the system is made through the Programming Model Enactment Service (PMES). PMES offers an OGF-BES (Basic Execution Service) compliant web service, accessible through WS clients (like Taverna), and also through a Java API. A Web Dashboard allows for a full control of the infrastructure. The Dashboard is useful for small analysis and for

development. A Galaxy interface interacting with the PMES web service allows integrating the infrastructure applications in Galaxy workflows.

6. Data management within the infrastructure can be done at different levels. PMES is able to manage directly data from users' premises using http or ftp protocols. For limited amounts of data, the infrastructure provides an http based personal data manager, where input and output data can be stored. Finally, for larger scale executions, standard ssh based access to a shared storage is provided. Services to download data from public repositories (PDB, Uniprot, Ensembl), or specific ones (OpenPhacts, MoDEL), are also available.

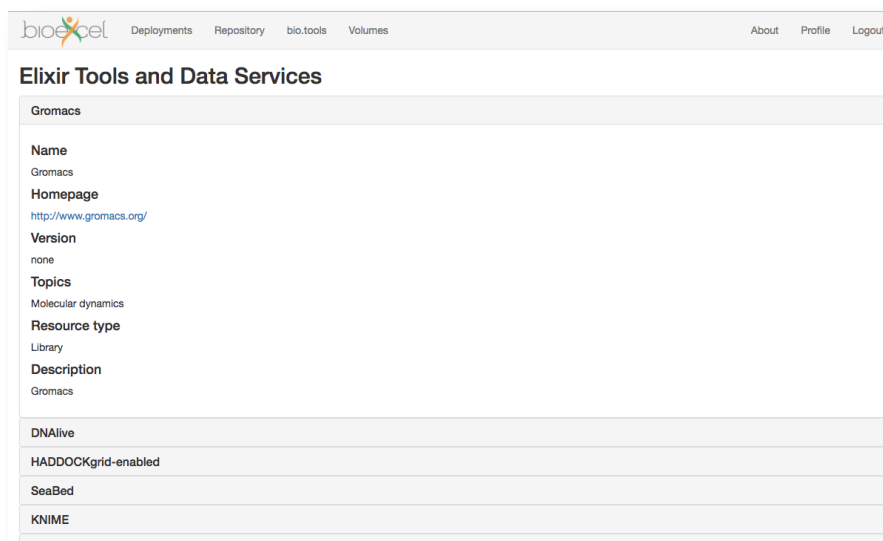


*Fig. 3.1 – BioExcel development computational infrastructure (BSC Testbed)*

### 3.2 EMBL-EBI BioExcel Portal (production)

The BioExcel Portal will present to researchers a list of life-science software supported by the project, which is obtained from the BioExcel tagged entries in the Elixir Tools Registry. From the BioExcel Portal a user will be able to select the service directly (if it is offered online as a service), find the HPC centres where it is already installed and available for use and how they can access the software, or to retrieve VMs from a repository (currently planned to be the EGI Applications Database but other solutions may be supported) and deploy the virtual machine or container across different types of cloud infrastructure through the EBI Cloud Portal (being supported by BioExcel) onto a cloud provider.

Additional applications are provided as configuration scripts, so they can be deployed on vanilla VMs. Right now, GitHub is being used as an application definition and sharing platform, opening the door to collaborative development and versioning of appliances.



**Fig. 3.2 - The BioExcel Portal User Interface linking Elixir bio.tools**

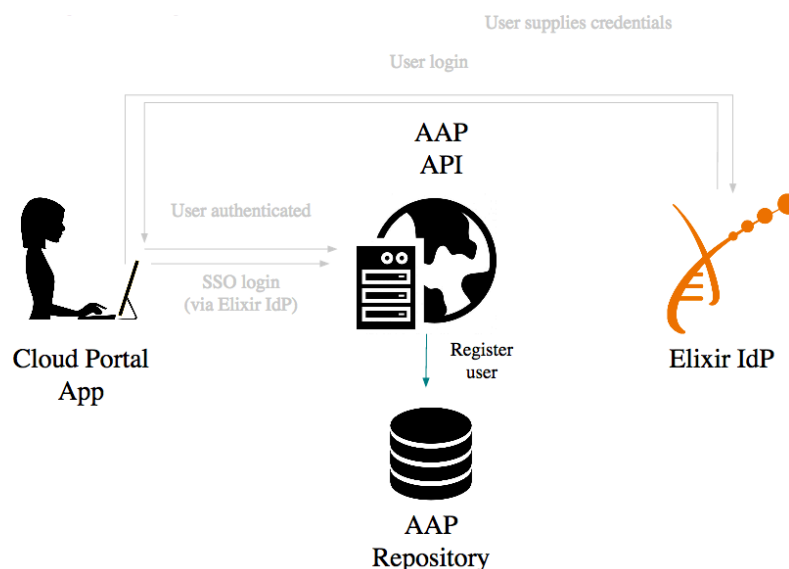
The BioExcel and EBI Cloud Portals make good use of two Elixir resources. The first is the Elixir Authentication and Authorisation Infrastructure (AAI), which is used as an identity provider in order to provide Single Sign On capabilities to the EBI Cloud Portal users. The second is the use of the Elixir application registry (bio.tools) in order to provide an application storefront to the BioExcel portal users (Fig. 3.2). By doing so, BioExcel benefits from a well-known directory of biomedical applications, while directing users to the usage of the EBI Cloud Portal within the BioExcel project. The EMBL-EBI Technology Science Integration (TSI) team has been collaborating closely with the bio.tools development team in order to bring new features and data representations to better represent the kind of applications and users that the BioExcel project will serve.

The overall architectural design of the EBI Cloud Portal has advanced considerably over the last year including the implementation of a REST API for programmatic access. We plan to initiate UX research, maximising the benefits of a common functional back end in order to serve different user communities with different needs.

### 3.2.1 Single Sign On

One of the strengths of the EBI Cloud Portal is the possibility of using a single account for authentication purposes. Elixir is used as an identity provider in order to provide Single Sign On capabilities to the EBI Cloud Portal users. EMBL-EBI has developed an Authentication Authorisation, and Profile service (AAP) as a way to centralise different identity interactions, including those required by the EBI Cloud Portal. One clear benefit of this, apart from being able to use a single account for sign on purposes, is the possibility of sharing permissions across multiple BioExcel, EMBL-EBI and Elixir related resources.

Users will be represented as unique across all of them, experiencing them as part of a single ecosystem (Fig. 3.3).



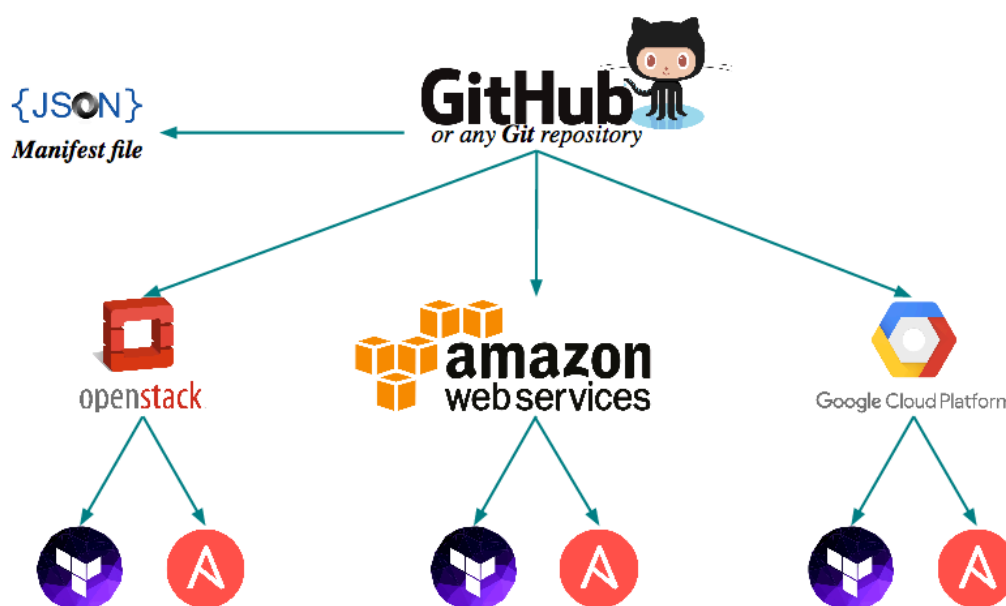
*Fig. 3.3 – BioExcel Single Sign On using Elixir*

### 3.2.2 Service definition and deployment

In order to have services ready for multi-cloud environments (i.e. compatible with different cloud providers) BioExcel services are represented to the EBI Cloud Portal using two technologies:

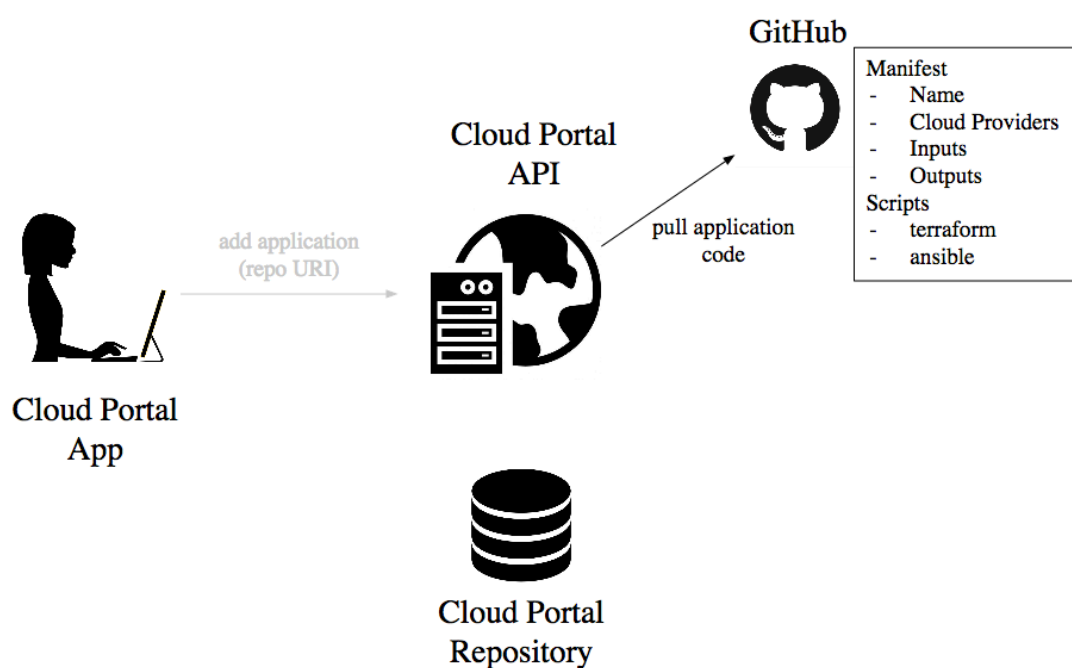
- **Terraform** is used to define the infrastructure the service needs to deploy into the cloud provider. This technology exposes a common language that can be translated into multiple specific cloud providers.
- **Ansible** is used to configure the infrastructure. By definition deals with infrastructure and therefore it is even more independent from the cloud provider.





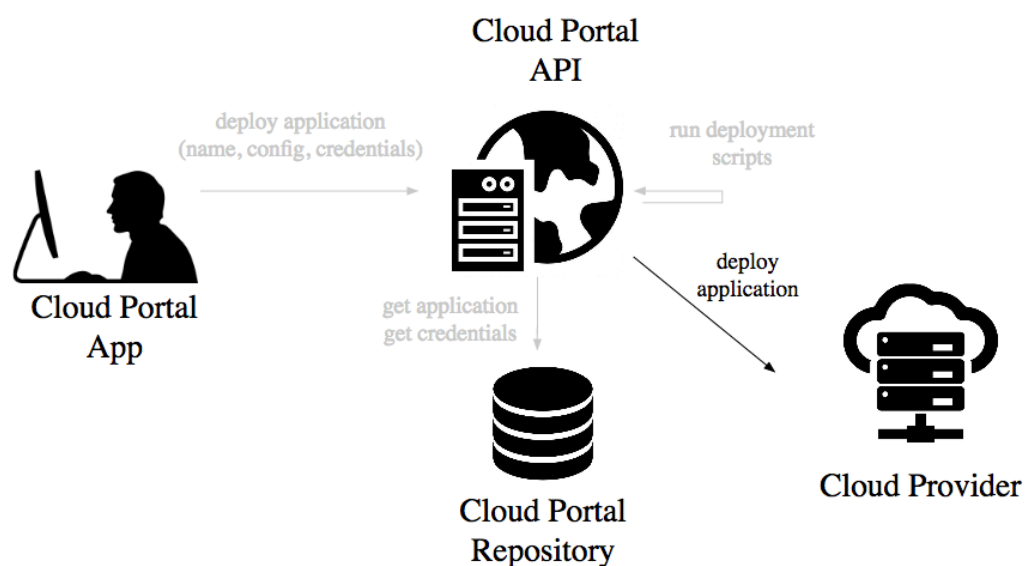
**Fig. 3.4 – EBI Cloud Portal application definition structure**

The EBI Cloud Portal consumes services defined by these two technologies in order to deploy a service into the Cloud Provider that the BioExcel users choose. Additionally, the whole service is described by a JSON-based manifest file, where the service developer can describe it in terms of its inputs, outputs, and additional metadata (including the set of supported cloud providers) (Fig. 3.4). All these scripts and manifest are consumed by the portal in order to install the new service as part of the EBI Cloud Portal repository of applications (Fig. 3.5).



**Fig. 3.5 – BioExcel portal consuming a service definition**

Finally, some services are provided as already packaged VMs. This differs from the previous model in the sense that by using Terraform and Ansible, we deploy infrastructure, configure it, and install software. By deploying a previously generated VM, we deploy a single infrastructure unit with all the required software already available inside of it (Fig. 3.6). Many bio.tools applications are provided this way so it is important that the EBI Cloud Portal can service them.



**Fig. 3.6 – BioExcel portal deploying a service/application**

## 4 Bioexcel Workflow Blocks

### 4.1 Transversal Workflows: Model Protein Mutants as a case example

Following the initial analysis of the use cases (UCs) and transversal units, we have identified a series of operations that appear in several of the cases (D2.1, Table 6.2). The initial work in Tasks 2.1 and 2.2 is to generate integrated workflows covering such functionalities and deploy them in the most appropriate environments. Components for such workflows will be adapted from the existing tools and orchestrated as interoperable building blocks, to be used in the chosen environment, using the Python-based modules library presented in section 2.2.2.

In this deliverable we describe the first transversal workflow prototype developed in BioExcel: Model Protein Mutants. This workflow allowed us to test, for the first time, the suitability of the software model chosen (section 2.2) and the development and production computational infrastructures (section 3). This workflow was chosen to be our first prototype because it contains building blocks and functionalities that will be also needed in several of the use cases presented in D2.1, as for example remote data access or MD simulation setup and running. Although this is a workflow performing a very specific operation, it provides a complete example and usage guidelines to demonstrate the procedure to build workflows using BioExcel tools. Source code (<https://github.com/bioexcel/pymdsetup>) and a VM (<http://inb.bsc.es/bioexcelova/bioexcel21.ova.zip>) providing it are available to download, and have been already used as a tutorial in the first BioExcel workshop on workflow management (see section 4.1.3).

#### 4.1.1 Model Protein Mutants: Overview

Model Protein Mutants is a pipeline with biological interest that was proposed in the project Description of Action (Task 2.2). It can be described as an automated protocol to generate structures for protein variants detected from genomics data. Structures will be prepared and analysed using Molecular Dynamics (MD) simulations.

The pipeline receives a Uniprot id as input, and it automatically retrieves all the information needed to model the structures for the different annotated mutations. It then prepares and runs MD simulations for each of the systems, thus obtaining static information (an ensemble of modeled structures for each of the protein variants) and also dynamic information (trajectories for each of the protein variants), which can be then used in a comparative study. The pipeline uses data from public repositories, but can be easily modified to accept user-provided specific datasets.

The workflow is divided in 5 main parts (Figure 4.1):

- Sequence Analysis

In the first step of the pipeline, the protein sequence, together with its possible variants and isoforms are recovered from the input id (using remote data access, from IRB or EMBL-EBI servers), resulting in a set of variants, which can be converted to a set of FASTA sequences.

- Structure Analysis

In the second step of the pipeline, protein structures were recovered from PDB database (again using remote data access) for each of the sequences. If no structure is found, it will be modeled (if possible) using a tool (yet to be decided).

- Structure Mutants

In the third step of the pipeline, the set of mutated structures are generated using tools such as MDWeb[15]. Mapping between genomic sequences and structures is obtained through remote data access.

- Molecular Dynamics

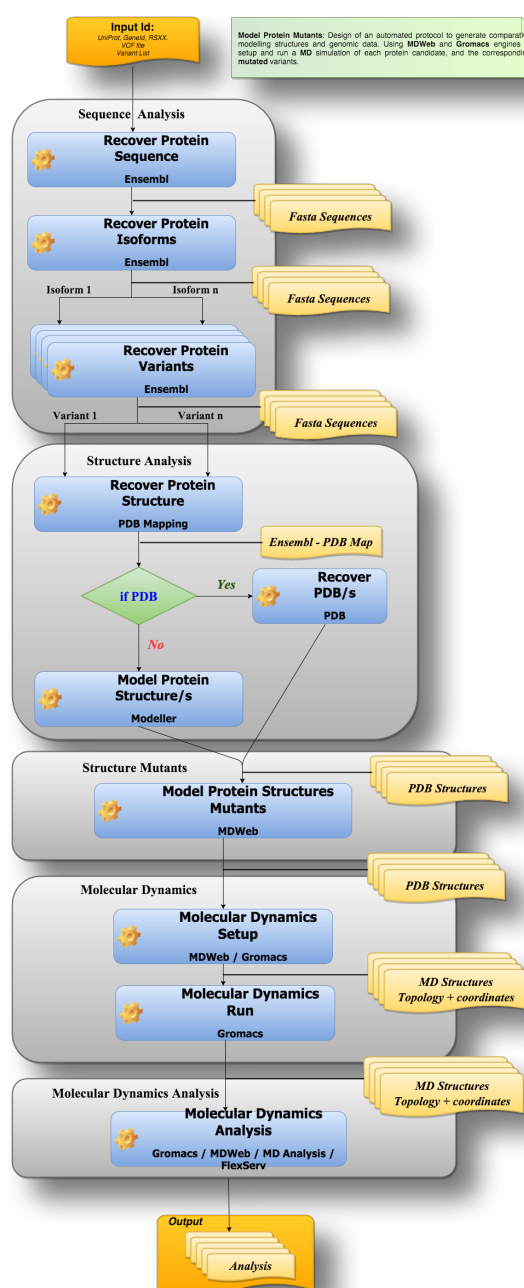
In the fourth step of the pipeline, all the generated structures are prepared until they are suitable to be used as input files for a MD simulation. This is a computationally expensive and complicated step that can be seen as another workflow by itself. It implies a series of steps, such as solvating the system, adding counterions to neutralize the charge, warming up the system to a desired temperature, equilibrating the entire system, etc. All the steps are performed using the MD set of components from our library of modules (PyMDSetup, see section 4.1.2) extracted from MDWeb and MDMoby web services[15], translated to the software model previously described, and updated to be used with the last version of GROMACS package[16].

Once the structures are prepared, the MD simulations are launched. As the simulations are the most computationally expensive part of the whole pipeline, this part of the workflow is the one that can be exploited by workflow managers such as PyCOMPSs, as it is able to manage parallel computations automatically discovering the dependencies between them (see section 4.1.2).

- MD Analysis

In the fifth and last part of the workflow, trajectories from the different simulations are gathered together and used as basis for a

comparative analysis. This step can be expanded to any possible and interesting analysis of MD trajectories, such as clustering of structures, flexibility properties and conformational changes, etc. By now, the prototype implemented is just computing a Root Mean Square deviation (RMSd) between all the different trajectories, to easily identify mutations causing large deviations in the simulations.



**Fig. 4.1 – First Transversal Workflow Prototype: Model Protein Mutants**

### 4.1.2 Model Protein Mutants: Run

The Model Protein Mutants workflow is implemented in a Python code that can be found in the BioExcel GitHub repository (<https://github.com/bioexcel>). The software model used in its implementation (see section 2.2.2) allows us to run the workflow either as a serial pipeline (single processor) or as a parallel pipeline (many processors), and more interestingly, it can be easily integrated with different workflow managers (workflow agnostic). To illustrate this capability, the workflow prototype has been integrated in two different workflow managers: PyCOMPSs[11] and Galaxy[12]. The first one, PyCOMPSs, is able to automatically split the pipeline in different threads, according to the number of different mutants (simulations) needed, and manage thread dependencies. The latter one, Galaxy, has a graphical interface and is widely used by the genomic community, but is not as extended in the structural community. Finally, the workflow can be also run in HPC facilities such as MareNostrum supercomputer in the BSC. A preliminary benchmark has been compiled with all the available environments. Further information about these points is included in the next sections.

- Serial execution

The modular components of the workflow are joined together in a Python script. Each one of them is invoked with the necessary input and output parameters, and then launched serially (Fig. 4.2).

```
print 'step4:  p2g ----- Create gromacs topology'
p_p2g = conf.step_prop('step4_p2g', mut)
fu.create_change_dir(p_p2g.path)
p2g = pdb2gmx.Pdb2gmx512(structure_pdb_path=p_scv.mut_pdb,
                        output_gro_path=p_p2g.gro,
                        output_top_path=p_p2g.top,
                        output_top_tar_path=p_p2g.tar,
                        water_type=p_p2g.water_type,
                        force_field=p_p2g.force_field,
                        ighn=settings.str2bool(p_p2g.ighn),
                        gmx_path=gmx_path,
                        log_path=p_p2g.out, error_path=p_p2g.err)

p2g.launch()

print 'step5:  ec ----- Define box dimensions'
p_ec = conf.step_prop('step5_ec', mut)
fu.create_change_dir(p_ec.path)
ec = editconf.Editconf512(input_gro_path=p_p2g.gro,
                        output_gro_path=p_ec.gro,
                        box_type=p_ec.box_type,
                        distance_to_molecule=float(p_ec.distance_to_molecule),
                        center_molecule=settings.str2bool(p_ec.center_molecule),
                        log_path=p_ec.out, error_path=p_ec.err, gmx_path=gmx_path,)

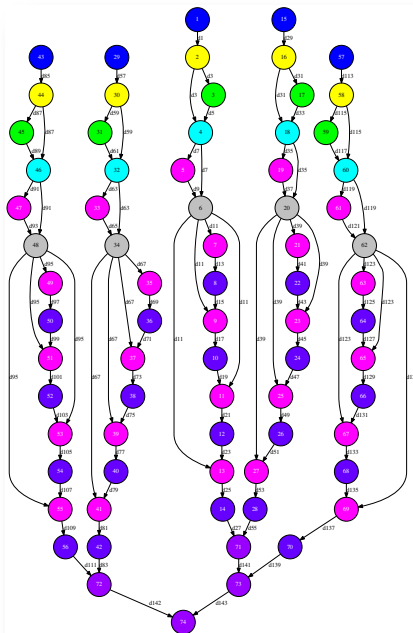
ec.launch()
```

**Fig. 4.2 – Serial execution code of the Workflow Prototype: Model Protein Mutants**

- PyCOMPSs execution

A version of the Protein Mutants workflow prepared to run with PyCOMPSs workflow manager[11] has also been written. PyCOMPSs is able to automatically

detect parallelizable parts of the pipeline, splitting the execution in different threads, and managing the possible dependencies between them. Fig 4.3 shows an example of an automatic dependency graph produced with the workflow prototype.



**Fig. 4.3 – Example of graph dependencies automatically generated by PyCOMPSs workflow manager**

Thanks to the modularity of the workflow components and the way PyCOMPSs recognizes the building blocks, the differences between the sequential workflow and the one optimized with PyCOMPSs are minimum. Only a set of directives (`@task`) on top of every module needs to be added to the script (Fig. 4.4).

```
@task(structure_pdb_path=FILE_IN, output_gro_path=FILE_OUT, output_top_path=FILE_OUT,
      output_top_tar_path=FILE_OUT, water_type=IN, force_field=IN, ighn=IN,
      log_path=FILE_OUT, error_path=FILE_OUT, gmx_path=IN)
def pdb2gmxPyCOMPSs(structure_pdb_path, output_gro_path, output_top_path,
                    output_top_tar_path, water_type, force_field, ighn,
                    log_path, error_path, gmx_path):
    """Launches the GROMACS pdb2gmx module using the PyCOMPSs library."""
    pdb2gmx.Pdb2gmx512(structure_pdb_path, output_gro_path, output_top_path,
                       output_top_tar_path, water_type, force_field, ighn,
                       log_path, error_path, gmx_path).launch()

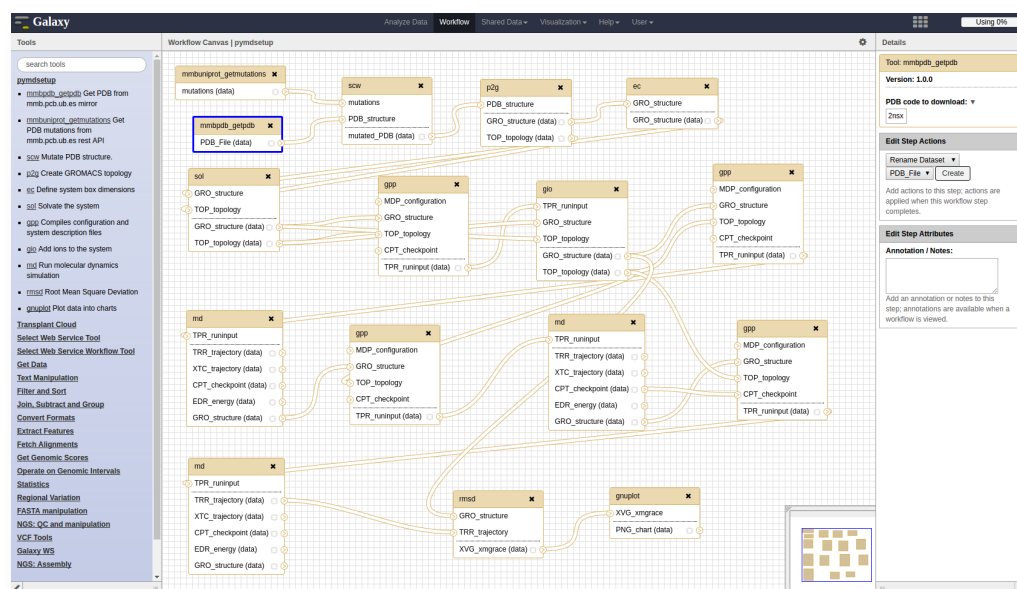
@task(input_gro_path=FILE_IN, output_gro_path=FILE_OUT, distance_to_molecule=IN,
      box_type=IN, center_molecule=IN,
      log_path=FILE_OUT, error_path=FILE_OUT, gmx_path=FILE_OUT)
def editconfPyCOMPSs(input_gro_path, output_gro_path, distance_to_molecule,
                    box_type, center_molecule,
                    log_path, error_path, gmx_path):
    """Launches the GROMACS editconf module using the PyCOMPSs library."""
    editconf.Editconf512(input_gro_path, output_gro_path, distance_to_molecule,
                        box_type, center_molecule,
                        log_path, error_path, gmx_path).launch()
```

**Fig. 4.4 – Example of the PyCOMPSs annotated building blocks**

- Galaxy execution

The Galaxy workflow manager[12] (see D2.1) is an open, web-based platform for data intensive biomedical research. Rather than building a workflow upfront, Galaxy uses a data playground approach, effectively building a workflow implicitly by applying a series of operations on the data items, keeping a history of all intermediate data items that are produced (and how they were made), making it easy to rerun parts of the workflow and share the results with others.

Galaxy is tightly integrated with a large collection of tools for genomics and sequence analysis, and is therefore popular for Next-Gen Sequencing data analysis. However, it is not so well-known in the structural biomolecular field, mainly due to its lack of tools focused on structures. In order to bring it closer to this community, and also to illustrate the possibility to use our modular tools in different workflow managers, our pipeline has also been integrated in a Galaxy platform (Fig. 4.5).



**Fig. 4.5 – Example of the Workflow prototype in the Galaxy graphical interface**

- MareNostrum execution

The workflow has been tested in an HPC environment, in particular, the MareNostrum supercomputer installed in the BSC premises. Due to its architecture MareNostrum is representative of most HPC facilities in Europe. Indeed, MareNostrum is a Tier-0 PRACE (Partnership for Advanced Computing in Europe) system. As the most usual infrastructure to run MD simulations are PRACE supercomputers, this test should be representative of the seamless deployment of the BioExcel software infrastructure at the HPC level.

The PyCOMPSs workflow manager was used to automatically control the parallelism within the pipeline.



### 4.1.3 Model Protein Mutants: Benchmarking

The workflow has been tested in the different computational infrastructures available to us, building a preliminary benchmark (the results obtained so far are shown in table 4.1). This table will be updated with new tests in the future.

The pipeline was run with the *Acid-β-Glucosidase Hydrolase* protein (PDB code: 2NSX, X-ray structure, 2.11Å resolution, 497 residues, Uniprot Id: P04062), choosing only 12 mutations, and with simulation times of 10ns per each mutated system. The different infrastructures prepared/used so far are:

- Virtual machine (1 core): Serial pipeline, Open Nebula VM deployed in the BSC testbed.
- Virtual machine (12 cores): Parallel pipeline, in one single Open Nebula VM deployed in the BSC testbed.
- BSC MareNostrum (128 cores): Parallel pipeline, managed by the new version of PyCOMPSS, presented in November 2016 (still queued).
- EGI: In preparation. We are currently working on building up a BioExcel Virtual Organization in EGI AppDB to register and deploy our VMs (see next section).

Infrastructure	Time
Virtual Machine (1 core)	19,2 hours
Virtual Machine (12 cores)	2,45 hours
BSC MareNostrum	Still queued
EGI	In preparation

**Table. 4.1 – Protein Mutants Workflow Prototype Benchmarking**

### 4.1.4 Model Protein Mutants: Deployment

The Model Protein Mutants workflow is coded in a single Python script, which uses and interconnects the modular pieces of the PyMDSetup library (building blocks).

The workflow prototype is a good proof of concept for our computational infrastructures in terms of the number of software packages needed to run. The software packages currently used in the pipeline, or that will be added in forthcoming releases, are the following:

Program	Description	Version	Already Included
Gromacs [16]	MD package chosen to prepare and run MD simulations (WP1)	5.0	YES
AmberTools [17]	Used to predict and model protein side-chains conformations	16	YES
SCWRL [18]	Used to model protein residues mutations. It will be eventually changed, as this version is only free to researchers in non-profit institutions	4.0	YES
CMIP [19]	Used to predict water molecules in the most energetically favorable protein surface positions, and also to predict the most energetically favorable protonation state of ionizable residues.	2.0	TO BE IMPLEMENTED
ACPYPE [20]	Used to automatically parameterize ligands to be used in the MD simulation. It needs antechamber program from AmberTools package and OpenBabel program	1.0	TO BE IMPLEMENTED
OpenBabel [21]	Used by ACPYPE for format conversions and hydrogen addition in ligands	2.3.2	TO BE IMPLEMENTED
PyCOMPSs [11]	Workflow manager used to control the pipeline in parallel executions	2.0	YES

**Table. 4.2 – Protein Mutants Workflow software packages**

The Model Protein Mutants workflow was successfully deployed in the BSC development computational infrastructure (see section 3.1) using a Virtual Appliance (VA) that was launched in the BSC cloud infrastructure. The same VA has also been registered into the EGI AppDB. All the packages previously listed were downloaded and installed, resulting into a virtual machine ready to run the workflow. This VA running in the BSC environment was the one used in the benchmarking presented in the previous section (Table 4.1). It was also the one used in the PyCOMPSs hands-on session of the “*BioExcel: workflow training for computational biomolecular research*” (20-21 October 2016). The VA can be downloaded from the following link (note that this is a preliminary workflow, it is still being improved and will be updated soon): <http://inb.bsc.es/bioexcelova/bioexcel21.oiva.zip>

In order to test the compatibility between the development computational infrastructure and the final production one within the EMBL-EBI premises, the very same VA needs to be also tested in our preliminary final portal. For that, it needs to be first uploaded as a VA under the BioExcel Virtual Organization which is currently being defined in EGI AppDB, as part of the Elixir-Europe Virtual Organization, (<https://appdb.egi.eu/store/vo/vo.elixir-europe.org>), so that it could be retrieved and deployed from the portal, and accessed through the AAI authentication presented in section 3.2.1. A new instance of the VA will then be instantiated using the OpenStack environment in EBI.

The state-of-the-art technology proposed by EMBL-EBI partners for the final portal is also starting to be tested using this workflow prototype. Employing Ansible and Terraform technologies (see section 3.2.2), we are currently working on instantiating a VM from scratch, automatically retrieving all software needed by the workflow to run (from GitHub repositories). This highlights the importance of having all software code centralized in software repositories (like GitHub) and all tools correctly registered in a central catalogue, with appropriate links to the software.

#### 4.1.5 Model Protein Mutants: Testing & verification

This workflow has been developed using continuous integration development practices and test driven development process (Fig. 4.6). This guarantees that a set of tests is provided for each building block. In addition the whole workflow has been tested (functional tests) with a small test set of PDB IDs. The final verification of the results correctness has been performed by team members with molecular dynamics expertise.

```
class TestPdb(unittest.TestCase):
    """Unit tests for ebi_api.pdb module
    """

    def setUp(self):
        self.data_dir = opj(os.path.dirname(__file__), 'data')
        self.results = opj(self.data_dir, "temp_results")

    def tearDown(self):
        # Remove all files in the temp_results directory
        for the_file in os.listdir(self.results):
            file_path = opj(self.results, the_file)
            try:
                # Not removing directories
                if os.path.isfile(file_path) and not the_file == 'README.txt':
                    os.unlink(file_path)
            except Exception, e:
                print e

    def test_get_pdb(self):
        pdb_code = '2ki5'
        output_path = opj(self.results, 'structure.pdb')
        gold_path = opj(self.data_dir, '2ki5_gold.pdb')
        ebi_st = EbiPdb(pdb_code, output_path)
        ebi_st.get_pdb()
        with open(output_path, 'r') as out_file, open(gold_path,
                                                       'r') as gold_file:
            out_string = out_file.read()
            gold_string = gold_file.read()
            self.assertMultiLineEqual(out_string[:1000], gold_string[:1000])
```

*Fig. 4.6 – Python Unittest module example used in the Workflow prototype*

## 4.2 Pilot Use Cases Workflows

An update on the current state of the five different Pilot use cases workflows described in the DoA is presented in the next sections.

### 4.2.1 Pilot Use Case 1: Genomics

Pilot Use Case 1 (UC1) has evolved since D2.1. Previously UC1 was focusing on how BCBio<sup>1</sup> (a commonly used genomics workflow package which offers a range of best-practice pipelines for automated analysis of high throughput sequencing data) could be used within HPC infrastructure like ARCHER<sup>2</sup> or Cirrus<sup>3</sup>. New partners working within the Edinburgh Institute for Genomics and Molecular Medicine (IGMM)<sup>4</sup> have proposed a new workflow to perform rapid, large-scale variant analysis of cancer patients. The overall aim is to quickly identify variants between the normal and cancer samples of the patient, which can help inform treatment. This workflow is similar to that described in D2.1, but with more steps surrounding the core BCBio workflow.

The individual steps in this workflow are not in themselves particularly novel; it is expected however, that by increasing the performance of the workflow (by adapting it for more powerful compute resources) and by increasing the extent to which it is automated, the contexts in which it could be used would be expanded. This could increase turn-around times in whole-genome research and could potentially even lead to treatment applications.

#### 4.2.1.1 *Rapid turnaround analysis pipeline for cancer variants from high throughput sequencing data - Overview*

The proposed workflow consists of 7 main steps. Most of these involve well-known software packages, which are bundled along with BCBio in most instances.

- Sequence QC – **FastQC**<sup>5</sup>
  - Generates quality metric for FASTQ files from sequencer.
- Sequence QC Review – **Script with common checks, with manual checks performed when needed**
  - Typically performs basic checks for quality score from FastQC, most pass and go through standard trimming in the next step. Some may require more detailed manual checking.
- Read Trimming – **cutadapt**<sup>6</sup>
  - Majority of runs will be simple trimming of adaptor sequences, but some may require multiple trims for poor quality samples.
- Alignment – **bwa, samblaster or Picard MarkDuplicates, SAMTOOLS, GATK BQSR**

---

<sup>1</sup> <https://bcbio-nextgen.readthedocs.io/en/latest/>

<sup>2</sup> <http://www.archer.ac.uk/>

<sup>3</sup> <https://www.epcc.ed.ac.uk/cirrus>

<sup>4</sup> <http://igmm.ac.uk>

<sup>5</sup> <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

<sup>6</sup> <http://cutadapt.readthedocs.io/en/stable/guide.html>

- Aligns reads to reference genome, converts SAM to BAM, coordinate sort, merge reads for different lanes into single BAM file, mark duplicates and perform Base Quality Score Recalibration.
- Alignment QC – **GATK or custom script**
  - Identify areas of low read depth and collect alignment metrics.
- Small variant/structural variant and copy number calling – **Undecided**
  - Compare normal and cancer samples with reference genomes for variations.
- Small/structural variant annotation – **vcfanno, Ensembl variant effect predictor or custom script**
  - Annotate variants with information from other sources, and predict effect of these variants on genes/tumor.

The workflow can be considered as three main blocks: Sequence QC and Trimming, Alignment, and Variant Calling and annotation. Decisions are yet to be made on the use of software packages at some steps, in particular the use of BCBio to drive the workflow in general. Previous experience by our UC1 partners found that BCBio does not keep read groups from different lanes separate during the alignment stage. Why this is the case is unclear, as the packages used in the custom pipeline are used by BCBio for alignment. It is possible that the Python wrapper around those packages within BCBio does not give the same level of control, or that user configuration is not sufficiently clear. This is something that we are working with the UC1 partners to investigate. This workflow is illustrated in Figure 4.7.

#### 4.2.1.2 *Deployment and execution*

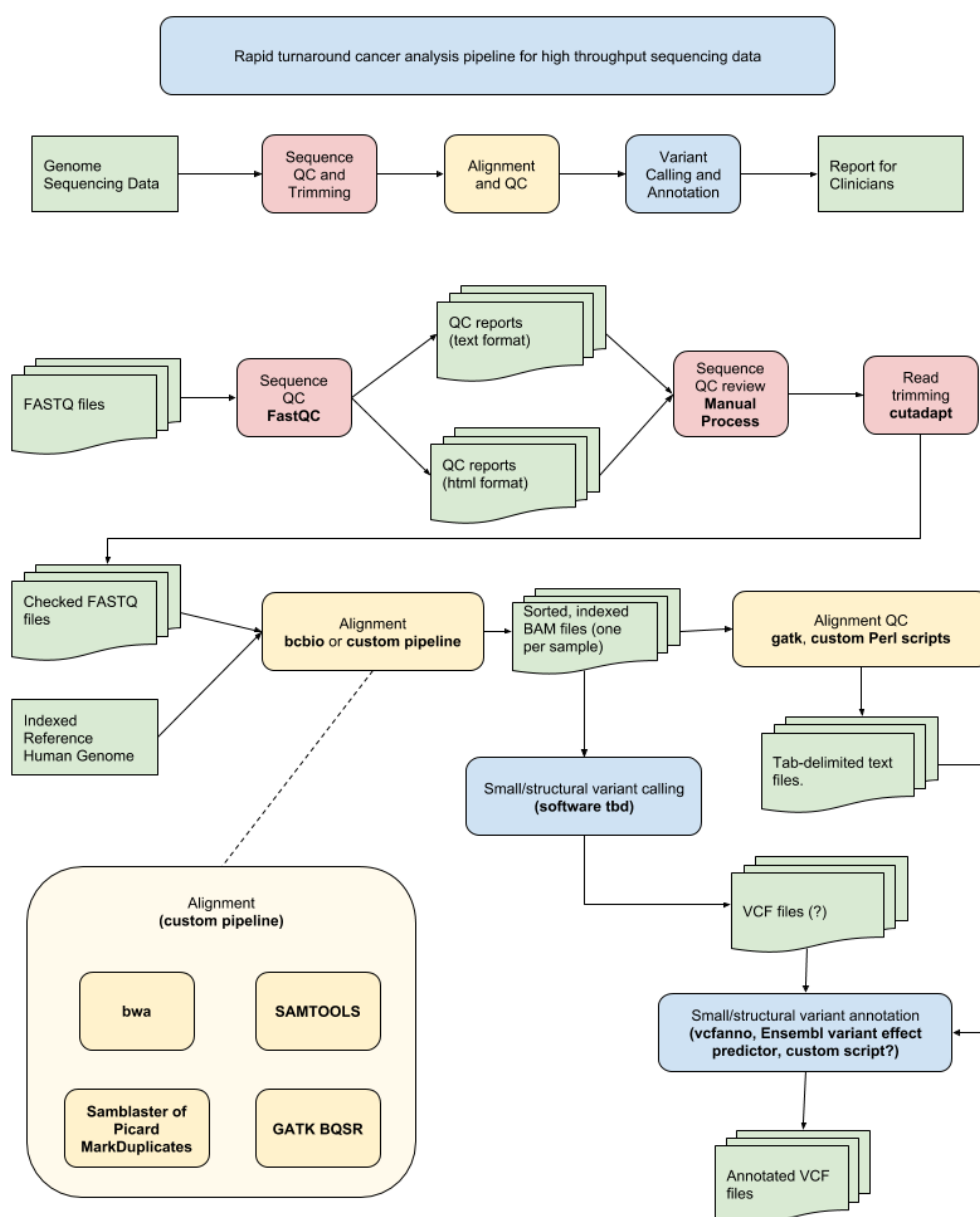
Depending on whether the issues with BCBio are overcome, we envisage the workflow running either in BCBio directly, or with a python script/wrapper around BCBio and its packages.

The use of virtual machines and/or Docker may not be ideal for this workflow. Part of our development is to investigate the application of HPC systems, such as ARCHER/Cirrus at EPCC, in genomics workflows. Typically, these systems do not directly support Docker and/or VMs.

The use of web/cloud portals may also not be suitable for this area of research, as patient data needs to be kept secure (with the exception of private clouds).

Our approach here is designed to be incremental, and we expect to follow a methodology similar to that which would be applied to optimizing an HPC code, namely to look for bottlenecks by profiling the code and then to optimize the sections which would lead to the best increase in performance. In this case, of

course, we also aim to make the building blocks of the workflow available and usable by the wider community.



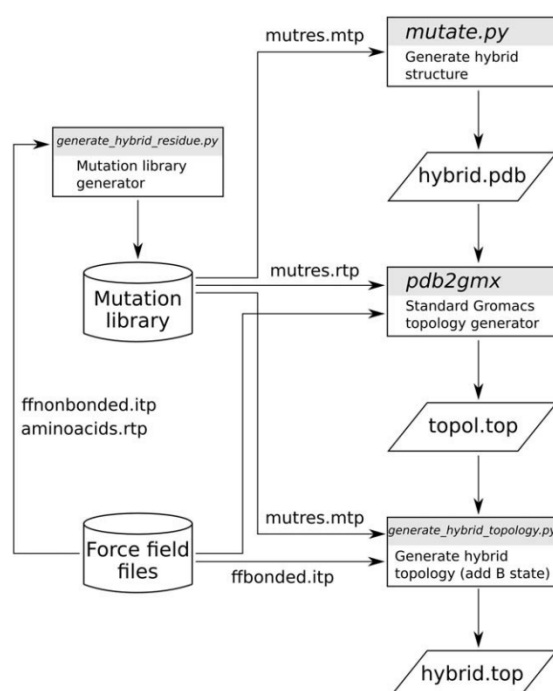
**Fig. 4.7 – Use Case 1 Workflow current state**

#### 4.2.2 Pilot Use Case 2: Free energy simulations of biomolecular complexes

Use case 2 is devoted to compute free energy simulations of biomolecular complexes through the development of a framework called pmx (formerly pymacs) to generate optimal mappings for arbitrary amino acid mutations in several modern molecular mechanics force fields and using the Gromacs MD package for carrying out the free energy simulations. A workflow pipeline has

been produced using the Python language, and has been already tested and deployed successfully (Fig 4.8).

The force field specific mutation libraries are created by *generate\_hybrid\_residue.py* using the amino acid topologies, bonded, and non-bonded parameters as defined in the force field of interest. By default pmx provides mutation libraries for five commonly used force fields. A hybrid-mutated structure is generated by *mutate.py*. The Gromacs standard tool *pdb2gmx* uses the hybrid structure to create its topology, which subsequently is processed by *generate\_hybrid\_topology.py* by adding the required parameters for the A and B states, between which the desired free energy difference is calculated.



**Fig. 4.8 – Use Case 2 Workflow current state**

The workflow in its current state is available through the pmx web server interface <http://pmx.mpibpc.mpg.de/webserver.html> (Fig. 4.9) and it will be soon accessible from the BioExcel Portal. The possibility of usage of the different blocks forming the workflow (python modules) in different pipelines will be studied during the development of the project.



pmx: generate hybrid protein structure and topology  
Computational Biomolecular Dynamics Group

pmx web server

Tripeptide DB

Instructions

Citations

Downloads

Tutorial

Changelog

Contact

Lab's website

- Structure file (.pdb):  No file chosen
- Force field selection:
  - ☐ Amber99SB\*ILDN
  - ☐ Amber99SB
  - ☐ Charmm36
  - ☐ Charmm22\*
  - ☐ OPLS AA/L
- Number of mutations:
- Perform a scan: ☐
- Select mutations:
- 1. Amino acid number:
- 1. Mutate to:
- Use pdb2gmx to assign hydrogens? ☒
- Submit the query:

WUFG-GENOMICS bioRxiv Boehringer Ingelheim

**Fig. 4.9 – pmx Web Server interface**

### 4.2.3 Pilot Use Case 3: Multi-scale modeling of molecular basis for odor and taste

Use case 3 will allow us to test the currently under development QM/MM interface of CPMD on a real size and biological interesting problem. This is understanding the mechanism how the enzyme adenylyl cyclase (AC) binding to the G-protein Gas (Gai) stimulates (inhibits) the synthesis of cyclic adenosine monophosphate (cAMP) from ATP substrate, amplifying signal transduction in the brain. A multi scale approach, in particular here a QM/MM one, is essential because the synthesis of cAMP through the so called ATP cyclization involves a chemical reaction but the entire system is too large to be treated fully at quantum level. The systems AC bound to Gas (AC:Gas) and AC:Gas in complex with an ATP strand in the reactive conformation (AC:Gas:ATP) were prepared and equilibrated with classical molecular dynamics.

Preliminary QM/MM simulations of AC:Gas and AC:Gas:ATP have been performed by using the original QM/MM interface of the CPMD code. Both the protonated (O3'H) and the deprotonated (O3') nucleophile cases have been considered. However, so far, only the deprotonated case produced stable QM/MM trajectories.

Currently, we started to perform constrained molecular dynamics of the AC:Gas:ATP in the protonated state in order to measure the free energy barrier of the proton transfer that initiate the enzymatic reaction. These simulations will be later used as reference for the comparison with the outcomes of the new QM/MM interface under development, in order to have a strong validation of the new QM/MM interface on a real biological case.



#### 4.2.4 Pilot Use Case 4: Biomolecular recognition

With this pilot use case, we aim at providing a workflow for the automated modelling of biomolecular interactions. Our HADDOCK engine (**H**igh **A**mbiguity **D**riven protein-protein **DOCK**ing[22]) lies at the center to generate models of the complexes. MD engines, such as Gromacs, will be used to both sample conformations prior to docking and to evaluate the stability of the best cluster representatives generated by HADDOCK through systematic MD simulations (post-docking).

The input data for this pilot use case is a protein, a peptide or a nucleic acids structure, either coming from a PDB code or a protein sequence (EBI, IRB APIs). Small molecule docking using SMILES information for the ligand is also an option, with potentially the possibility of estimating the binding affinity of the complex for this particular use-case.

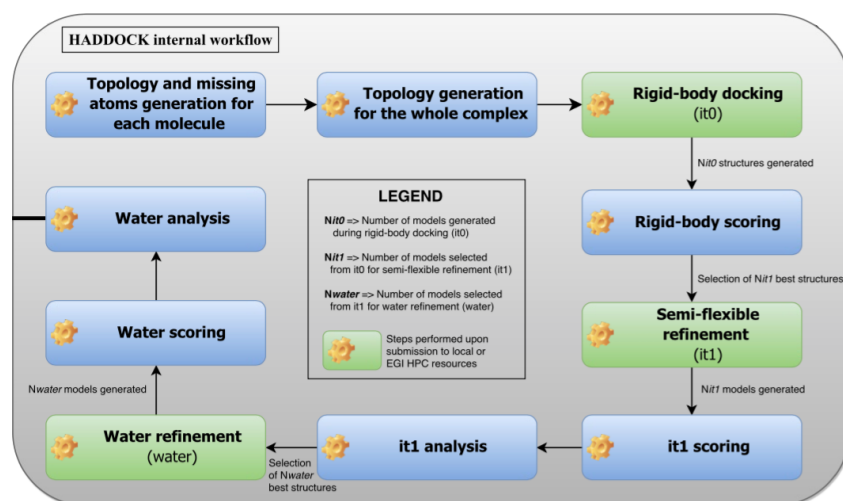
After considering several workflow managers, we decided in first instance to collaborate with the group of Dr. Daan Geerke (molecular toxicology group, VU Amsterdam, NL) who is using [crossbar.io](http://crossbar.io), a Web Application Messenger Protocol (WAMP) that allows to build distributed systems out of application components which are loosely coupled and can communicate in real-time. His group is already building up such a workflow for protein-small ligand docking and binding affinity prediction and they are interested in adding a HADDOCK module to their initial design. We have many good reasons to engage with them for this pilot use-case. Indeed, Daan Geerke's group has joined the BioExcel community through a collaborative partnership that has been settled in December 2016. Moreover, their workflow already contains a MD module (Gromacs) and some post-docking analytics for which we are also interested to contribute.

Their workflow is currently in its finalization stage (expected Q1 2017) and we already organized two face-to-face meetings (Q4 2016) to discuss the technical details of the implementation of HADDOCK as a module into their web-messaging platform. Note that this choice should not prevent us of also exploring the PyCOMPSs-based solutions developed at BSC.

HADDOCK itself is a complex workflow (Fig. 4.10 shows its internal workflow; the complete pipeline is presented in D1.1, Figures 2 and 3). It has been ported to a VM for testing purposes under the INDIGO-Datacloud project, but the specifications of the VM are not scalable (not enough CPU cores) therefore it does not make much sense to use a VM for the local version of HADDOCK. At last, there still is a licensing issue in making HADDOCK freely available in a VM because its computational engine, CNS (<http://cns-online.org>), requires an additional license subscription.

In terms of workflow, it would make more sense to directly connect to the web server (via its xml-rpc interface for example, or with a simple script filling the web forms). The server already has a one-file upload option, which allows a simple upload of all data and settings for the docking within a self-contained text file. We are also working on allowing the user to upload "recipes" in JSON format,

which would simplify the docking set-up.



*Fig. 4.10 – Use Case 4 Workflow current state*

#### 4.2.5 Pilot Use Case 5: Virtual screening

Use case 5 will allow users to run ensemble docking using **Open PHACTS** to obtain pharmacological compounds in combination with the **Gromacs** MD engine to prepare MD ensembles and **HADDOCK** / **Seabed** to run biomolecular docking.

The pipeline behind this workflow has been divided in different steps (see Fig. 4.11), which will be implemented in parallel. The current state of each of these steps is explained in the next points:

##### 1. Recover Protein Structure/s and prepare MD ensemble (Fig. 4.11, green):

This part of the workflow is currently implemented using the PyMDSetup modules presented in section 3.X. Structures are extracted from the PDB IRB (<http://mmb.irbbarcelona.org/api/help/>) or EMBL-EBI (<http://www.ebi.ac.uk/pdbe/pdbe-rest-api>) APIs. A MD trajectory is computed with Gromacs and the PyMDSetup modules, and an ensemble of structures is taken from it (either clusterizing or by other methods yet to be decided).

##### 2. Enhance Sampling (Fig. 4.11, blue):

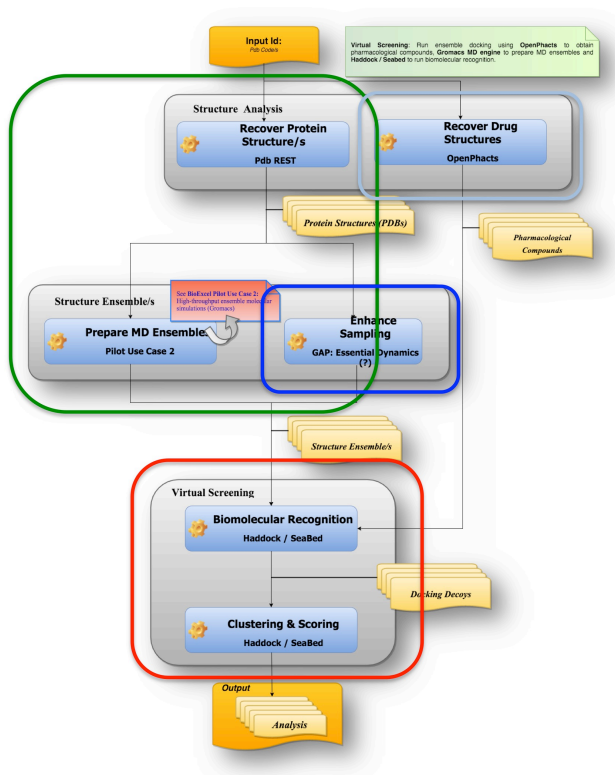
For this part of the workflow, a code is already available in IRB, based in coarse-grained dynamics, but it is still being adapted to the presented Python wrapping model. Once done, it will be integrated in the pipeline.

### 3. Biomolecular recognition (Fig. 4.11, red):

SeaBED[23] server code will be also adapted to our Python wrapping schema in order to easily integrate its functionalities to our pipeline. The possibility of integration of HADDOCK docking package will be also studied.

### 4. Open PHACTS integration (Fig. 4.11, grey):

UNIMAN and BSC partners are currently working together integrating Open PHACTS library in the BSC infrastructure, to be used in the recovering of drug structures in the workflow.



*Fig. 4.11 – Use Case 5 Workflow current state*

## 5 Conclusions & Future Work

In this deliverable, the first release of workflow blocks and portals for the BioExcel center of excellence has been presented. A first workflow prototype (Model Protein Mutants) has been successfully deployed and run in different computational environments. This workflow has been designed and implemented following the workflow development process best practices accepted by the BioExcel community, which has been devised together with Elixir bioinformatics project. The workflow prototype is available through a Virtual Machine (<http://inb.bsc.es/bioexcelova/bioexcel21.ova.zip>) and the source code can be found in BioExcel GitHub repository (<https://github.com/bioexcel/pymdsetup>). The building blocks prepared with the Python wrapper model described in this deliverable will be used in the coming months to build up different workflows and to help in the development of the pilot use cases workflows. These building blocks will be expanded to cover more fields. An update of the 5 pilot use cases described in the project DoA has also been presented. Some of them (either completely or partially) are already available as web servers.

The BioExcel computational infrastructures for development & testing (BSC) and production (EMBL-EBI) are already up and running. A set of preliminary tests has been performed using the transversal workflow prototype. A BioExcel Virtual Organization within the Elixir-Europe one in EGI AppDB has been accepted and will be used to register and deploy our VMs. State-of-the-art technology such as Terraform and Ansible, available in the production infrastructure will start to be tested in the next months.

The first version of the BioExcel Portal, which will present the list of life-science software supported by the project and will allow users to select and access a particular service is prepared and is being tested internally. We plan to have the first public version available in the coming months.

After the first stage of our *WP2: Portable environments for computing and data resources*, we are now prepared to jump to the next phase:

- Test workflows and VMs in the development infrastructure.
- Upload and deploy workflows and VMs to the production infrastructure.
- Make these workflows/VMs available through the BioExcel Portal.
- Start collecting feedback from our partners.

## 6 References

1. Peter, A., et al., *Common Workflow Language, v1.0*. 2016, Figshare.
2. Crosswell, L.C. and J.M. Thornton, *ELIXIR: a distributed infrastructure for European biological data*. Trends in Biotechnology. **30**(5): p. 241-242.
3. Ison, J., et al., *EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats*. Bioinformatics, 2013. **29**(10): p. 1325-32.
4. Pettifer, S., et al., *The EMBRACE web service collection*. Nucleic Acids Research, 2010. **38**(suppl 2): p. W683-W688.
5. Kalas, M., et al., *BioXSD: the common data-exchange format for everyday bioinformatics web services*. Bioinformatics, 2010. **26**(18): p. i540-6.
6. Ison, J., et al., *Tools and data services registry: a community effort to document bioinformatics resources*. Nucleic Acids Research, 2016. **44**(D1): p. D38-D47.
7. Ménager, H., et al., *Using registries to integrate bioinformatics tools and services into workbench environments*. International Journal on Software Tools for Technology Transfer, 2016. **18**(6): p. 581-586.
8. Merzky, A., et al., *RADICAL-Pilot: Scalable Execution of Heterogeneous and Dynamic Workloads on Supercomputers*. CoRR, 2015. **abs/1512.08194**.
9. Jha, S. and M. Turilli, *A Building Blocks Approach towards Domain Specific Workflow Systems?* CoRR, 2016. **abs/1609.03484**.
10. Balasubramanian, V., et al., *ExtASY: Scalable and Flexible Coupling of MD Simulations and Advanced Sampling Techniques*. Vol. abs/1606.00093\, 2016.
11. Tejedor, E., et al., *PyCOMPSs: Parallel computational workflows in Python*. International Journal of High Performance Computing Applications, 2015.
12. Afgan, E., et al., *The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update*. Nucleic Acids Research, 2016. **44**(W1): p. W3-W10.
13. Bechhofer, S., et al., *Why linked data is not enough for scientists*. Future Generation Computer Systems, 2013. **29**(2): p. 599-611.
14. Chard, K., et al., *I'll Take That to Go: Big Data Bags and Minimal Identifiers for Exchange of Large, Complex Datasets*.
15. Hospital, A., et al., *MDWeb and MDMoby: an integrated web-based platform for molecular dynamics simulations*. Bioinformatics, 2012. **28**(9): p. 1278-9.
16. Abraham, M.J., et al., *GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers*. SoftwareX, 2015. **1-2**: p. 19-25.
17. Case, D.A., et al., *AMBER 2016*, S.F. University of California, Editor. 2016: University of California, San Francisco.
18. Krivov, G.G., M.V. Shapovalov, and R.L. Dunbrack, *Improved prediction of protein side-chain conformations with SCWRL4*. Proteins, 2009. **77**(4): p. 778-95.
19. Gelpí, J.L., et al., *Classical molecular interaction potentials: improved setup procedure in molecular dynamics simulations of proteins*. Proteins, 2001. **45**(4): p. 428-37.

20. Sousa da Silva, A.W. and W.F. Vranken, *ACPYPE - AnteChamber PYthon Parser interface*. BMC Research Notes, 2012. **5**: p. 367-367.
21. O'Boyle, N.M., et al., *Open Babel: An open chemical toolbox*. Journal of Cheminformatics, 2011. **3**(1): p. 33.
22. van Zundert, G.C.P., et al., *The HADDOCK2.2 Web Server: User-Friendly Integrative Modeling of Biomolecular Complexes*. Journal of Molecular Biology, 2016. **428**(4): p. 720-725.
23. Fenollosa, C., et al., *SEABED: Small molEcule activity scanner weB service baseD*. Bioinformatics, 2015. **31**(5): p. 773-5.