WAVES
Web Application for Versatile Enhanced Bioinformatic Services

Marc Chakiachvili \textsuperscript{1,2}, Sylvain Milanesi \textsuperscript{1}, Anne-Muriel Arigon Chifolleau \textsuperscript{1}, Vincent Lefort \textsuperscript{1}*

\textsuperscript{1} LIRMM, Université de Montpellier, CNRS, Montpellier, France
\textsuperscript{2} European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Hinxton, Cambridge, CB101SD, United Kingdom

* Corresponding author: lefort@lirmm.fr

Supplementary Material

- Definitions pp. 2
- WAVES architecture pp. 3
- WAVES main module: WAVES-core
  - General description pp. 4-5
  - Functionalities pp. 6
  - Users interfaces pp. 7-9
- Other WAVES module: WAVES-Galaxy pp. 9
- Licensing pp. 10
- Usefull links pp. 10
DEFINITIONS

**Computing infrastructure**
Comprized of computationally dedicated hardware and the software components required to operate it, including calculation management programs (distributed resource management systems, Galaxy, other APIs for computation resources...).

**Adapter**
A component allowing WAVES to communicate with a computing infrastructure.

**Service**
A bioinformatic tool available online via the http protocol. A service can be accessed from a web form or through REST API calls.

**Usage**
A list of expected inputs and outputs for a service.

**Submission**
The combination of a usage and a computing infrastructure. A service can thus rely on different submissions.

**Job**
Represents a submission run with user-defined values for inputs (files and parameters). A job results in the execution of a command with parameters. It is run on the computing infrastructure defined by the submission. A job generates outputs: exit code, standard output and standard error, and potentially result files.

**User**
A client accessing services. A client can be a physical person using a web browser, or software using the REST API.

**Administrator**
A user with access rights to the WAVES back-office. The administrator manages services, submissions, adapters, and jobs.
The main goal of WAVES is to facilitate bioinformatic tool integration through web interfaces in order to provide the scientific community with online bioinformatic services. The main WAVES component is called WAVES-core. This component enables administrators to create services that are made available automatically for users. WAVES-core creates web interfaces to provide access to services: web pages and web forms, and REST API entry points (Sup. Fig. 1).

WAVES-core integrates predefined adapters for two types of computing infrastructures: one for command line execution and the other for running jobs on DRMAA-compliant resource management systems. These adapters are designed to be extended to other computing infrastructures. We also provide an extension for Galaxy that is implemented within the WAVES-Galaxy adapter (see dedicated section below).

Sup. Fig. 1. WAVES architecture overview. WAVES is built around WAVES-core, which provides the main WAVES functionalities: user interfaces (“WEB PAGES”, “WEB FORMS” and “REST API”), services and user management (“BACK-OFFICE” and “USER MANAGEMENT”) for administrator use only, and WAVES-core adapters for running jobs on local or remote computing infrastructures. WAVES also includes the WAVES-Galaxy adapter, and can be extended easily to other computing infrastructures by creating dedicated adapters.
**WAVES main module: WAVES-core**

**General description**

Most of the time, a bioinformatic tool is defined within a single service that relies on a single submission (Sup. Fig. 2, service B). Each user call to a service creates a job that is submitted to the computing infrastructure with specified inputs. WAVES-core then monitors job status during processing on the computing infrastructure. Once the job is terminated, WAVES-core retrieves the results, stores them, and makes them available online via web pages and API entries.

However, some bioinformatic tools provide several distinct usages. For example, a program can be run using a command-line interface or by providing a configuration file. Thus, the same service can provide several usages for the same tool and can be run on several different computing infrastructures. This is done by configuring a submission for each required combination (Sup. Fig. 2, service A).

A standard use case is to define two submissions for the same tool:

- A default submission with a selection of the most relevant tool parameters with predefined values. This submission is intended to be implemented for inexperienced users.
- An expert submission, which allows fine-tuning of each tool parameter, for advanced users.

**Sup. Fig. 2.** Service submission model within WAVES.
**Service creation**

Once installed on a computing infrastructure, a tool is integrated into WAVES by creating and configuring a dedicated service. This is achieved by filling out the ‘Online Service’ form (Sup. Fig. 3) in the WAVES administration interface to define the tool settings. Services can also be imported into WAVES from Galaxy instances and thus automatically configured (see WAVES-Galaxy section below).

**Sup. Fig. 3.** ‘Online Service’ configuration form from WAVES back-office administration interface.

**Service use**

There are three different ways to interact with WAVES services: web pages, web forms, and a RESTful API. Example code showing how to interact with a WAVES service web form and API are available from the Developer Guide online documentation.
Service access rules

For each service, the WAVES administrator can setup different access levels (Sup. Table 2).

<table>
<thead>
<tr>
<th>Service status</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft</td>
<td>Restricted to service creator</td>
</tr>
<tr>
<td>Staff</td>
<td>Access is granted to other WAVES back-office users</td>
</tr>
<tr>
<td>Registered</td>
<td>Access is granted to registered users</td>
</tr>
<tr>
<td>Restricted</td>
<td>Access is restricted to a selection of registered users</td>
</tr>
<tr>
<td>Public</td>
<td>Access is granted to anyone</td>
</tr>
</tbody>
</table>

Sup. Table 1. WAVES services status and corresponding user access rights.

Functionalities

Adapters

By default, WAVES-core comes with predefined adapters to interact with a variety of computing infrastructures (Sup. Table 1). Each adapter parameter is configured by the administrator through the WAVES-core back-office.

<table>
<thead>
<tr>
<th>Adapter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocalShellAdapter</td>
<td>Execute job with standard shell on Linux-based platform</td>
</tr>
<tr>
<td>SshShellAdapter</td>
<td>Connect through SSH protocol with standard credentials user / password on remote server and execute shell command</td>
</tr>
<tr>
<td>SshKeyShellAdapter</td>
<td>Connect through SSH protocol using ssh key pairing credentials on remote server and execute shell command</td>
</tr>
<tr>
<td>LocalClusterAdapter</td>
<td>Run job on locally installed DRMAA(^1) compatible cluster (currently SGE, SLURM, TORQUE, PBS, LSF, CONDOR)</td>
</tr>
<tr>
<td>SshClusterAdapter</td>
<td>Connect through SSH protocol with standard user / password credentials on remote DRMAA compatible cluster and run job</td>
</tr>
<tr>
<td>SSHKeyClusterAdapter</td>
<td>Connect through SSH protocol using ssh key pairing credentials on remote DRMAA compatible cluster and run job</td>
</tr>
</tbody>
</table>

Sup. Table 2. Predefined WAVES-core adapters for computing infrastructures.

---

\(^{1}\) [http://www.drmaa.org/](http://www.drmaa.org/)
**Job spool**

Once submitted, jobs enter a FIFO (First In First Out) queue to be treated and relayed to the relevant adapter at regular time intervals.

After completion, jobs can be deleted either by administrators or by authorized users. WAVES-core is not intended to be a genuine job queue manager. It relies on an external computation scheduler for performing jobs prioritization and authorization configuration.

Each job may be cancelled by its owner, and WAVES-core then attempts to cancel job execution on the associated computing infrastructure.

**User management**

Users are managed with the Django standard authentication / authorization mechanism\(^2\), which may be extended to fit any specific requirements for integration. User management allows fine tuning for user access rights.

**Users interfaces**

**Web pages**

WAVES-core provides already made front-end web pages, designed with the bootstrap 3\(^3\) CSS framework and using Django standard templating engine\(^4\). By default, WAVES-core defines the following URIs for accessing its services (Sup. Table 3).

<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/waves/services/</td>
<td>List all available services</td>
</tr>
<tr>
<td>/waves/services/{service_app_name}/</td>
<td>Display service details</td>
</tr>
<tr>
<td>/waves/services/{service_app_name}/new</td>
<td>Create a job through submission form(s)</td>
</tr>
<tr>
<td>/waves/jobs/{unique_id}/</td>
<td>View job details</td>
</tr>
<tr>
<td>/waves/jobs/inputs/{unique_id}?export=1</td>
<td>View input file online / Download file</td>
</tr>
<tr>
<td>/waves/jobs/outputs/{unique_id}?export=1</td>
<td>View output file online / Download file</td>
</tr>
</tbody>
</table>

**Sup. Table 3.** Web page URIs to access WAVES services.

---

\(^2\) [https://docs.djangoproject.com/en/1.11/topics/auth/](https://docs.djangoproject.com/en/1.11/topics/auth/)

\(^3\) [https://getbootstrap.com/](https://getbootstrap.com/)

\(^4\) [https://docs.djangoproject.com/en/1.11/topics/templates/](https://docs.djangoproject.com/en/1.11/topics/templates/)
RESTful API

WAVES-core services are available through RESTful API endpoints. The API lists the available services, and returns service details and associated usages defined within submissions (Sup. Table 4).

<table>
<thead>
<tr>
<th>METHOD</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/waves/api/services</td>
<td>List all available services</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/services/{service_app_name}</td>
<td>Retrieve service details</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/services/{service_app_name}/form</td>
<td>Retrieve service forms (for all submissions)</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/services/{service_app_name}/jobs</td>
<td>Retrieve service jobs (only for logged-in users)</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/services/{service_app_name}/submissions</td>
<td>List all available submissions for this service</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/services/{service_app_name}/submissions/{submission_app_name}</td>
<td>Get detailed service submission information (inputs, parameters, expected outputs)</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/services/{service_app_name}/submissions/{submission_app_name}/jobs</td>
<td>List all user jobs for this submission</td>
</tr>
<tr>
<td>POST</td>
<td>/waves/api/services/{service_app_name}/submissions/{submission_app_name}/jobs</td>
<td>Create a new job from submitted inputs</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/services/{service_app_name}/submissions/{submission_app_name}/form</td>
<td>Get HTML submission form that submits data to the above &quot;POST&quot; URI</td>
</tr>
</tbody>
</table>

Sup. Table 4. WAVES-core services REST API endpoints. The "POST" method is only accessible through authenticated requests, while "GET" method can be accessed anonymously.
WAVES-core provides REST API endpoints to follow the job execution life cycle and retrieve results once terminated (Sup. Table 5).

<table>
<thead>
<tr>
<th>METHOD</th>
<th>URI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/waves/api/jobs</td>
<td>List all available user job summaries, with related detailed URI</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/jobs/{unique_id}</td>
<td>Detailed job information, allows direct link to submitted inputs, results outputs, status history, related submission and service</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/jobs/{unique_id}/status</td>
<td>Retrieve job status</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/jobs/{unique_id}/inputs</td>
<td>List job inputs</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/jobs/{unique_id}/inputs/{api_name}</td>
<td>Retrieve input content (for file)</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/jobs/{unique_id}/outputs</td>
<td>List job outputs</td>
</tr>
<tr>
<td>GET</td>
<td>/waves/api/jobs/{unique_id}/outputs/{api_name}</td>
<td>Retrieve output file content</td>
</tr>
<tr>
<td>POST</td>
<td>/waves/api/jobs/{unique_id}/cancel</td>
<td>Cancel job and tag it as cancelled</td>
</tr>
<tr>
<td>DELETE</td>
<td>/waves/api/jobs/{unique_id}</td>
<td>Delete job</td>
</tr>
</tbody>
</table>

Sup. Table 5. WAVES-core REST API endpoints for job management. All methods require an authenticated request.

The WAVES-core API is compatible with Core API⁵ specifications. Thanks to Django REST framework⁶, WAVES implements the Core JSON schema which can be browsed from WAVES-demo⁷.

Other WAVES module: WAVES-Galaxy

Using the BioBlend⁸ python library, this adapter enables Galaxy services import into WAVES-core. WAVES-Galaxy⁹ automatically recognizes the tools available in a Galaxy instance. The WAVES administrator can select the tools to integrate within the back-office. WAVES-Galaxy then creates a new service for each selected Galaxy tool automatically. When called from WAVES, each submission of this service is run on the Galaxy instance from which it was imported.

---

⁵ [http://www.coreapi.org/](http://www.coreapi.org/)
⁷ [http://waves.demo.atgc-montpellier.fr/waves/api/schema](http://waves.demo.atgc-montpellier.fr/waves/api/schema)
⁸ [http://bioblend.readthedocs.io/](http://bioblend.readthedocs.io/)
Licensing

Django and its components are open-source software, developed by a large community (more than 10,000 people) and benefiting from a wide range of reusable packages.

WAVES has been made available on GNU Public License version 3, meaning that anyone may use, reuse, and modify the code. We invited people to help us make WAVES even better, so don’t hesitate to participate in the project on GitHub.

Useful links

**WAVES-core**
- GitHub repository: https://github.com/lirmm/waves-core
- Documentation: http://waves-core.readthedocs.io

**WAVES-demo**
- GitHub repository: https://github.com/lirmm/waves-demo
- Documentation: http://waves-demo.readthedocs.io
- Example: http://waves.demo.atgc-montpellier.fr/

**WAVES-galaxy adapter**
- GitHub repository: https://github.com/lirmm/waves-galaxy
- Documentation: http://waves-galaxy-adaptors.readthedocs.io

**SAGA python framework**
- GitHub repository: https://github.com/radical-cybertools/saga-python
- Documentation: http://saga-python.readthedocs.io
- Project home: https://radical-cybertools.github.io

---

10 [https://www.gnu.org/licenses/gpl.html](https://www.gnu.org/licenses/gpl.html)