ceibacli Documentation

Release 1.0.0

Felipe Zapata

Mar 22, 2021

CONTENTS:

CHAPTER

ONE

CEIBA-CLI

command line interface to interact with the insilico web server. See the documentation and blog post.

1.1 Installation

To install ceiba-cli, do:

```
pip install git+https://github.com/nlesc-nano/ceiba-cli.git@main
```

1.1.1 Contributing

If you want to contribute to the development of ceiba-cli, have a look at the contribution guidelines.

1.1.2 License

1.1.3 Credits

This package was created with Cookiecutter and the NLeSC/python-template.

CHAPTER

TWO

AUTHENTICATION

2.1 Generate an OAuth token

You need to generate an **OAuth token from GitHub** in order to login into the application! For doing so, you should:

1. Go to github tokens and click the `Generate new token` button.
2. Provide your GitHub password when prompted
3. Fill in a description for the token, for example, *ceiba access token*.
4. **Do not enable any scope** therefore the token will grant read-only access to the app
5. Click `Generate` at the bottom. Make sure to copy its value because you will need it to login!

CHAPTER

THREE

USAGE

The **ceibacli** command line interface offers four actions to interact with the Ceiba web service. You can check them by trying the following command in your terminal:

```
user>  ceibacli --help
```

You should see something similar to:

```
usage: ceibacli [-h] [--version] {login,add,compute,report,query,manage} ...

positional arguments:
  {login,add,compute,report,query,manage}
                        Interact with the properties web service
    login               Log in to the Insilico web service
    add                 Add new jobs to the database
    compute             Compute available jobs
    report              Report the results back to the server
    query               Query some properties from the database
    manage              Change jobs status

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
```

After running one of the previous commands a log file named `ceibacli_output.log` is generated.

CHAPTER

FOUR

LOGIN

We certainly want to restrict who can access and modify the data. Therefore users are required to login with the web service. For doing so, you should have a GitHub account, then you need to request a **read-only** token from GitHub personal access token service.

Once you have a read-only GitHub token, you can login into the web service like:

```
ceibacli login -w http://YourCeibaInstance:8080/graphql -t Your_token
```

CHAPTER

FIVE

HOW DOES IT WORK?

The *Ceiba server* will contact GitHub and will check if you are a known user there.

CHAPTER

SIX

ADD (FOR ADMINISTRATORS)

The `add` command is an adminstrative action to add new jobs into the database.

To add jobs you need to run the following command in the terminal:

```
ceibacli add -w http://yourCeibaInstance:8080/grapqhl -c collection_name  -j Path/to/
↪jobs.json
```

Where the *-w* option is the web service URL. the *collection_name* is the collection where the data is going to be stored. Finally, the *-j* options is the path to the *JSON* file containing the jobs as an array of JSON objects. See the next *Jobs File* section for further information.

6.1 Jobs File

The job file is a list of json objects, like:

```
[
    {
        "type": "awesome_simulation_1",
        "parameters": {
            "value": 3.14
        }
    },
    {
        "type": "awesome_simulation_2",
        "parameters": {
            "value": 2.187
        }
    }
]
```

Each job is a JSON object with the parameters to perform the simulation.

6.2 How does it work?

The *add* command will read each job in the JSON jobs file. For each job it will generate a unique identifiers. Then, the jobs and their identifier will be stored a collection named *job_your_collection_name*.

CHAPTER

SEVEN

COMPUTE

The `compute` command ask the *web service* for available jobs that need to be run. To run some jobs you need to type in the terminal:

```
ceibacli compute -i input_compute.yml
```

Where the *input_compute.yml* is an file in YAML format containing the *Compute Input File* metadata.

The compute command takes the user's input, request some available job and *Job Scheduling* those jobs using the information provided by the user.

7.1 Compute Input File

The input file contains the following mandatory keywords:

```
# Web service URL
web: "http://YourCeibaInstance:8080/graphql"

# Name of the collection to compute
collection_name: "simulation_name"

# Command use to run the workflow
command: compute_properties
```

Other optional keywords are:

```
# Configuration of the job scheduler
scheduler:
   "none"


# Path to the directory where the calculations are going to run (default: workdir_
↪ceibacli)
workdir:
   /path/to/workdir

# Number of jobs to request and run (default: 10)
max_jobs:
   5
```

7.2 Job Scheduling

Most of the scientific simulation are usually perform in supercomputers that use a job scheduler. *ceiba-cli* supports two of the most popular ones: SLURM. If you choose a *scheduler* different from `none`, *ceiba-cli* will automatically contact the job scheduler with the options that you have provided. Below you can find a description of the available options:

```
# Job scheduler. Of of "none" or "slurm" (default: none)
scheduler:
   slurm

# Number of computing nodes to request (default: 1)
nodes:
   1

# Number of CPUs per task (default: None)
cpus_per_task:
   48

# Total time to request ind "days:hours:minutes" format (default: 1day)
walltime:
  "01:00:00"

# Partion name (queue's name) where the job is going to run (default: None)
partion_name:
  "short"
```

You can alternatively provide a string with all the options for the queue system like,

```
scheduler:
  slurm

# String with user's Configuration
free_format: "#!/bin/bash
#SBATCH -N 1
#SBATCH -t 00:15:00
....
"
```

7.3 Job State

The user's requested jobs are initially marked as `RESERVERED`, in the web service to avoid conflicts with other users. Then, if the jobs are sucessfully scheduled they are marked as *RUNNING*. If there is a problem during the scheduling or subsequent running step the job would be marked as *FAILED*.

CHAPTER

EIGHT

REPORT

The `report` command send the results of the jobs computed by the user to the web service. You can also send data that is not associated to any job to the server. In the last case, the results don't have all the metadata associated with a job in the server, for example because it has been previously computed or computed in another facility.

To report the results you need to type in the terminal:

```
ceibacli report -w http://yourCeibaInstance:8080/grapqhl
```

Or if you want to have more control over what is reported you can provide an input file like:

```
ceibacli report -i input_report.yml
```

Where the *input_compute.yml* is an file in YAML format containing the *Report results from a job* metadata.

You can also report results without associated jobs, follow the *Report results without associated jobs*.

8.1 Report results from a job

If the results that you want to report where computed with the *ceibacli compute* command, you can optionally provide the following input:

```
# Path to the Folder where the jobs run (default "workdir_ceibacli")
path_results: "workdir_ceibacli"

# Pattern to search for the result files (default "results*csv")
output: "results*csv"

# Pattern to search for the input files (default "inputs*json")
input: "inputs*json"

# If the data is already in server you can either:
# KEEP the old data
# OVERWRITE and discard the old data
# MERGE the new and the old data (Default)
# APPEND new data at the end of the old data array
# Default = KEEP
duplication_policy: "KEEP"
```

Check the *Large objects data storage* for further information on saving large output files.

8.2 Report results without associated jobs

Sometimes you have some results that you have previously computed and you want to share them with your colleagues. You can upload those results into the database very similarly to the previous section, but you need to provide an additional keyword:

```
has_metadata: False
```

You also need to provide the `path_results` and the `output` to look for. The `has_metadata` indicates to *Ceiba-cli* that the results that you want to report don't have metadata about how the results where computed.

8.3 How does it work?

The library enters the `path_results` and search recursively all the files and directories name like `job_*`. In each subfolder, apart from the computed data (specificied with the `pattern` keyword), the `report` command would try to collect the metadata associated with the job in a files named *metadata.yml* containing the following information:

```
job_id: 1271269411
property:
    collection_name: awesome_data
    id: 76950
```

Without the metadata no data is reported back to the server.

8.4 Reporting data without associated jobs

8.5 Large objects data storage

For many simulation it is desirable to store the output plain data and/or the binary checkpoints. Those files can be used to retrieve data that is not available in the database or to restart a calculation to perform further computations.

Those large objects are not suitable for storage in a database but fortunately there are technologies like swift openstack that allows to store these kind of data in an efficient and safely way.

In order to storage large output you need to provide in the yaml file the following keywords:

```
large_objects:
  # URL to the datastorage service
  web: "http://large_scientific_data_storage.pi"
  # The large file(s) to search for
  patterns:  ["output*hdf5"]
```

Note:

- Installing, deploying an mantaining a swift openstack data storage service is a nontrivial task. Therefore it is recommended to request access to this service to a provider. Be aware that **IT COSTS MONEY** to maintain the service running in a server!
- The large files and their corresponding metadata are going to be stored in the swift collection. using the same `collection_name` that has been specified in the *How does it work?*.

CHAPTER

NINE

QUERY

The `query` actions requests some data from the web service and writes the requested data in a csv file.

There are currently two possible query actions:

- request what collections are available
- request a single collection

To request what collections are available you just need to run the following command:

```
ceibacli query -w http://yourCeibaInstance:8080/grapqhl
```

Previous command will ouput something similar to:

```
 Available collections:
name size
simulation1 3
simulation2 42
....
```

In the previous `name` indicates the actual collections' names and `size` how many datasets are stored in that particular collection.

To request all the datasets available in a given collection, you just need to run the following command:

```
ceibacli query -w http://yourCeibaInstance:8080/grapqhl -c simulation2
```

That command will write into your current work directory a file called `simulation2.csv` containing the properties in the requested collection.

CHAPTER

TEN

MANAGE (FOR ADMINISTRATORS)

The `manage` command is an adminstrative action to change the jobs status. For example, jobs that have been marked as `RESERVED` or `RUNNING` for a long period of time can be marked again as `AVAILABLE` if the user doesn't report the results.

To change the jobs status you need to type in the terminal:

```
ceibacli manage -i input_manage.yml
```

Where the *input_manage.yml* is an file in YAML format containing the *Manage Input File* specification.

10.1 Manage Input File

The following snippet represent an input example for the *manage* action:

```
# Web service URL
web: "http://YourCeibaInstance:8080/graphql"

# Target collection to change job status
collection_name: "example_collection"

# Metadata to change jobs status
change_status:
 old_status: RUNNING
 new_status: AVAILABLE
 expiration_time: 24 # one day
```

10.2 How does it work?

ceiba-cli will research in the `collection_name` for all the jobs with `old_status` then it will check if those jobs have been scheduled before the `expiration_time`. If the jobs have expired, ceiba-cli will marked the expired jobs with the `new_status`.

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search