biobb_remote Documentation

Release 1.2.2

Bioexcel Project

Feb 22, 2022

CONTENTS

CHAPTER

ONE

CONTENTS

1.1 biobb_remote

1.1.1 Introduction

Biobb_remote is a package to allow biobb's to be executed on remote sites through ssh

The latest documentation of this package can be found in our readthedocs site: latest API documentation.

1.1.2 Version

v1.2.3 November 2021

1.1.3 Installation

Using PIP:

> **Important:** PIP only installs the package. All the dependencies must be installed separately. To perform a complete installation, please use ANACONDA.

- Installation:

```
pip install "biobb_remote>=1.2.2"
```

- Usage: Python API documentation

Using ANACONDA:

- Installation:

```
conda install -c bioconda "biobb_remote>=1.2.2"
```

- Usage: With conda installation BioBBs can be used with the Python API documentation and the Command Line documentation

1.1.4 Copyright & Licensing

This software has been developed in the MMB group at the BSC & IRB for the European BioExcel, funded by the European Commission (EU H2020 675728).

- (c) 2015-2021 Barcelona Supercomputing Center
- (c) 2015-2021 Institute for Research in Biomedicine

Licensed under the GNU Lesser General Public License v2.1.

1.2 biobb_remote

1.2.1 biobb_remote package

Submodules

biobb_remote.ssh_credentials module

Module to generate and manage SSL credentials

class biobb_remote.ssh_credentials.**SSHCredentials**(*host=''*, *userid=''*, *generate_key=False*, *look_for_keys=True*)

Bases: object

biobb_remote SSHCredentials
Class to generate and manage SSL key-pairs for remote execution.

Parameters

- **host** (*str*) *(Optional)* – Target host name.
- **userid** (*str*) *(Optional)* – Target user id.
- **generate_key** (*bool*) *(Optional)* – (False) Generate a pub/private key pair.
- **look_for_keys** (*bool*) *(Optional)* – (True) Look for keys in user's .ssh directory if no key provided.

check_host_auth()
Checks for public_key in remote .ssh/authorized_keys file. Requires users' SSH access to host.

generate_key(*nbits=2048*)
Generates RSA keys pair

Parameters **nbits** (*int*) – (2048) Number of bits of the generated key.

get_private_key(*passwd=None*)
Returns a readable private key.

Parameters **passwd** (*str*) – (None) Use passwd to encrypt key.

get_public_key(*suffix='@biobb'*)
Returns a readable public key suitable to add to authorized keys.

Parameters **suffix** (*str*) – ('@biobb') Suffix added to the key for identify it.

install_host_auth(*file_bck='bck'*)
Installs public_key on remote .ssh/authorized_keys file. Requires users' SSH access to host.

Parameters **file_bck** (*str*) – ('bck') Extension to add to backed-up authorized_keys file.

load_from_file(*credentials_path*, *passwd=None*)
Recovers SSHCredentials object from disk file.

Parameters

- **credentials_path** (*str*) – Path to packed credentials file.
- **passwd** (*str*) – (None) Use to decrypt private key.

load_from_private_key_file(*private_path*, *passwd=None*)
Loads private key from an standard file.

Parameters

- **private_path** (*str*) – Path to private key file.
- **passwd** (*str*) – (None) Password to decrypt private key.

remove_host_auth(*file_bck='biobb'*)
Removes public_key from remote .ssh/authorized_keys file. Requires users' SSH access to host.

Parameters **file_bck** (*str*) – ('biobb') Extension to add to backed-up authorized_keys.

save(*output_path*, *public_key_path=None*, *private_key_path=None*, *passwd=None*)
Save packed credentials on external file for re-usage.

Parameters

- **output_path** (*str*) – Path to file
- **public_key_path** (*str*) – (None) Path to a standard public key file.
- **private_key_path** (*str*) – (None) Path to a standard private key file.
- **passwd** (*str*) – (None) Password to be saved.

biobb_remote.ssh_session module

Module to manage SSH sessions

class biobb_remote.ssh_session.**SSHSession**(*ssh_data=None*, *credentials_path=None*, *private_path=None*, *passwd=None*, *debug=False*)

Bases: object

biobb_remote ssh_session.SSHSession
Class wrapping ssh operations

Parameters

- **ssh_data** (SSHCredentials) *(Optional)* – (None) SSHCredentials object.
- **credentials_path** (*str*) *(Optional)* – (None) Path to packed credentials file to use.
- **private_path** (*str*) *(Optional)* – (None) Path to private key file.
- **passwd** (*str*) *(Optional)* – (None) Password to decrypt credentials.
- **debug** (*bool*) *(Optional)* – (False) Prints (very) verbose debug information on ssh transactions.

close()
Closes active SSH session

is_active()
Tests whether the defined session is active

run_command(*command*)
Runs a shell command on remote, produces stdout, stderr tuple

Parameters **command** (*str | list(str)*) – Command or list of commands to execute on remote.

run_sftp(*oper*, *input_file_path*, *output_file_path=''*, *reuse_session=True*)
Opens a SFTP session on remote and execute some file operation

Parameters

- **oper** (*str - Operation to perform*) –
  - **get** - gets a single file from input_file_path (remote) to output_file_path (local).
  - **put** - puts a single file from input_file_path (local) to output_file_path (remote).
  - **create** - creates a file in output_file_path (remote) from input_file_path string.
  - **file** - opens a remote file in input_file_path for read). Returns a file handle.
  - **listdir** - returns a list of files in remote input_file_path.
- **input_file_path** (*str*) – Input file path or input string
- **output_file_path** (*str*) – (‘’) Output file path. Not required in some ops.
- **reuse_session** (*bool*) – (True) Re-use active SFTP session

biobb_remote.task module

Base module to handle remote tasks

class biobb_remote.task.**DataBundle**(*bundle_id*, *remote=False*)
Bases: object

biobb_remote task.DataBundle
Class to pack a files manifest

Parameters

- **bundle_id** (*str*) – Id for the data bundle
- **remote** (*bool) (Optional*) – (False) Marks bundle as remote (no stats are generated)

add_dir(*dir_path*)

DataBundle.add_dir
Adds all files from a directory

Parameters **dir_path** (*str*) – Path to the directory

add_file(*file_path*)

DataBundle.add_file
Adds a single file to the data bundle

Parameters **file_path** (*str*) – Path to the file.

get_file_names()

DataBundle.get_file_names
Provides a list of names of included files

get_full_path(*file_name*)

DataBundle.get_full_path
Gives the full path for a given file

Parameters **file_name** (*str*) – Name of the file.

get_mtime(*file_name*)

DataBundle.get_mtime

Gives the modification time for a given file

Parameters **file_name** (*str*) – Name of the file.

`to_json()`

DataBundle.to_json
Generates a Json dump of the DataBundle

`class biobb_remote.task.Task(host=None, userid=None, look_for_keys=True, debug_ssh=False)`
Bases: `object`

task.Task
Abstract classe to handle task executions.
Not to be used directly, should be extended with queue specific inherited classes.

Parameters

- **host** (*str) (Optional*) – (None) Remote FQD of remote host.
- **userid** (*str) (Optional*) – (None) Remote user id.
- **look_for_keys** (*bool) (Optional*) – (True) Look for keys in user's .ssh directory.
- **debug_ssh** (*bool) (Optional*) – (False) Open SSH session with debug activated.

`cancel(remove_data=False)`

Task.cancel
Cancels running task

Parameters **remove_data** (*bool) (Optional*) – (False) Removes remote working directory

`check_job(update=True, save_file_path=None, poll_time=0)`

Task.check_job
Prints current job status

Parameters

- **update** (*bool) (Optional*) – (True) Update status before printing it.
- **save_file_path** (*str) (Optional*) – (None) Local task log file to update progress.
- **poll_time** (*int) (Optional*) –

(0) Poll until job finished (seconds).

`check_queue()`

Task.check_queue
Check queue status

clean_remote()

Task.clean_remote
Remove job data from remote host

get_logs()

Task.get_logs
Get stdout, and stderr queue logs.

get_output_data(*local_data_path=''*, *files_only=None*, *overwrite=True*, *new_only=True*, *verbose=False*)

Task.get_output_data
Downloads the contents of remote working dir

Parameters

- **local_data_path** (*str) (Optional*) – ('') Path to local working dir
- **files_only** (*list(str) (Optional)*) – (None) Only download files in list, if empty download all files
- **overwrite** (*bool) (Optional*) – (True) Overwrite local files if they exist
- **new_only** (*bool) (Optional*) – (True) Overwrite only with newer files
- **verbose** (*bool) (Optional*) – (False) Show file status

get_queue_info()

Task.get_queue_info
Prints remote queue status.
Extended in inherited classes.

get_remote_comm_line(*command*, *files*, *use_biobb=False*, *properties=''*, *cmd_settings=''*)

Task.get_remote_comm_list
Generates a command line for queue script

Parameters

- **command** (*str*) – Command to execute

- **files** (*dict*) – Input/output files. "–" added if not only parameter name is provided
- **use_biobb** (*bool) (Optional*) – (False) Set to prepend biobb path on host
- **properties** (*dict) (Optional*) – ('') BioBB properties
- **cmd_settings** (*dict) (Optional*) – ('') Settings to add to command line (use -x or –xxx as necessary)

get_remote_file(*file*)

Task.get_remote_file
Download file from remote working dir

Parameters **file** (*str*) – Name of the remote file to download.

get_remote_file_stats()

Task.get_remote_file_stats
Returns remote files stats

get_remote_py_script(*python_import*, *files*, *command*, *properties=''*)

Task.get_remote_py_script
Generates one-line python command to be inserted in the queue script

Parameters

- **python_import** (*str | list(str)*) – Import(s) required to run the module (; separated).
- **files** (*dict*) – Files required for module execution (parameter:file_name).
- **command** (*str*) – Class name to launch.
- **properties** (*dict | str) (Optional*) – ('') Either a dictionary, a json string, or a file name with properties to pass to the module.

load_data_from_file(*file_path*, *mode='json'*)

Task.load_data_from_file
Loads accumulated task data from local file

Parameters

- **file_path** (*str*) – Path to file
- **mode** (*str) (Optional*) – (json) File format. Accepted: Json | Pickle

load_host_config(*host_config_path*)

Task.load_host_config
Loads a configuration file for the remote host.

Parameters **host_config_path** (*str*) – Path to the configuration file

prep_auto_settings(*total_cores=0*, *nodes=0*, *cpus_per_task=1*, *num_gpus=0*)

Task.prep_auto_settings
Prepare queue settings for balancing MPI/OMP/GPU.

Parameters

- **total_cores** (*int) (Optional*) –
  (0) Aproximated number of cores to use
- **nodes** (*int) (Optional*) –
  (0) Number of complete nodes to use (overrides total_cores)
- **cpus_per_task** (*int) (Optional*) –
  (1) OMP processes per MPI task to allocate
- **num_gpus** (*int) (Optional*) –
  (0) Num of GPUs per node to allocate

prep_remote_workdir(*remote_base_path*)

Task.prep_remote_workdir
Creates a empty remote working dir

Parameters **remote_base_path** (*str*) – Path to remote base directory, task folders created within

save(*save_file_path*, *mode='json'*, *verbose=False*)

Task.save
Saves current task status in a local file.
Can be used to recover session at a later time.

Parameters

- **save_file_path** (*str*) – Path to file
- **mode** (*str) (Optional*) – (json) Format to use json|pickle.
- **verbose** (*bool) (Optional*) – (False) Print additional information on stdout

send_input_data(*remote_base_path*, *create_dir=True*, *overwrite=True*, *new_only=True*)

Task.send_input_data

Uploads data to remote working dir

Parameters

- **remote_base_path** (*str*) – Path to remote base directory, task folders created within
- **create_dir** (*bool) (Optional*) – (True) Creates remote working dir
- **overwrite** (*bool) (Optional*) – (True) Allows overwrite files with the same name if any
- **new_only** (*bool) (Optional*) – (True) Overwrite only with newer files

set_credentials(*credentials*)

Task.set_credentials
Loads ssh credentials from a SSHCredentials object or from a external file

Parameters **credentials** (SSHCredentials | *str*) – SSHCredentials object or a path to a file containing the data.

set_custom_settings(*ref_setting='default'*, *patch=None*, *clean=False*)

Task.set_custom_settings
Adds custom settings to host configuration

Parameters

- **ref_setting** (*str) (Optional*) – (default) Base settings to modify
- **patch** (*dict) (Optional*) – (None) Patch to apply
- **clean** (*bool) (Optional*) – (False) Clean settings

set_local_data_bundle(*local_data_path*, *add_files=True*)

Task.set_local_data_bundle
Builds local data bundle from a local directory

Parameters

- **local_data_path** (*str*) – Path to local data directory
- **add_files** (*bool) (Optional*) – (True) Add all files in the directory

set_private_key(*private_path*, *passwd=None*)

Task.set_private_key
Inserts private key from external file

Parameters

- **private_path** (*str*) – Path to private key file.
- **passwd** (*str) (Optional*) – (None) Password to decrypt private key.

submit(*job_name=None*, *set_debug=False*, *queue_settings='default'*, *modules=None*, *local_run_script=''*, *conda_env=''*, *save_file_path=None*, *poll_time=0*)

Task.submit
Submits task to the queue, return job id, optionally polls until job completion

Parameters

- **job_name** (*str) (Optional*) – (None) Job name to display (used to identify queue jobs, and stdout/stderr logs)
- **set_debug** (*bool) (Optional*) – Adjust queue settings to debug QoS (as defined in host configuration)
- **queue_settings** (*str) (Optional*) – (default) Label for set of queue controls (as defined in host configuration). Use 'custom' for patched settings
- **modules** (*str) (Optional*) – (None) Modules to activate (defined in host configuration)
- **conda_env** (*str) (Optional*) – ('') Conda environment to activate
- **local_run_script** (*str) (Optional*) – ('') Path to local bash script to run or a string with the script itself (identified by a leading '#script' tag)
- **save_file_path** (*str) (Optional*) – (None) Path to save task log
- **poll_time** (*int) (Optional*) –

(0) Polling time for job completion (seconds). Set to O to do not wait.

biobb_remote.slurm module

Module to define characteristics of SLURM queue manager

class biobb_remote.slurm.**Slurm**(*host=None*, *userid=None*, *look_for_keys=True*)
Bases: *biobb_remote.task.Task*

biobb_remote slurm.Slurm
Task Class to set specific SLURM settings
Extends biobb_remote.task.Task

Parameters

- **host** (*str) (Optional*) – (None) FQD for remote host.
- **userid** (*str) (Optional*) – (None) Remote user id
- **look_for_keys** (*bool) (Optional*) – (True) Allow using local user's credentials

1.3 Command Line Test Scripts

1.3.1 credentials

Credentials manager. Generates key pairs to be consumed by other utilities

```
credentials [-h] [--user USERID] [--host HOSTNAME]
            [--pubkey_path PUBKEY_PATH] [--nbits NBITS] --keys_path
            KEYS_PATH [--privkey_path PRIVKEY_PATH]
            command
```

Commands:

- **create**: Create key pair
- **get_pubkey**: Print Public key
- **get_private**: Print Private key
- **host_install**: Authorize key in remote host (requires authorized local keys)
- **host_remove**: Revert authorization on remote host (requires authorized local keys)
- **host_check**: Check authorization status on remote host. Operation: create|get_pubkey

optional arguments:

```
-h, --help                show this help message and exit
--user USERID             User id
--host HOSTNAME           Host name
--pubkey_path PUBKEY_PATH Public key file path
--nbits NBITS             Number of key bits
--keys_path KEYS_PATH     Credentials file path
--privkey_path PRIVKEY_PATH Private key file path
-v                        Output extra information
```

1.3.2 scp_service

Simple sftp service

```
scp_service [-h] --keys_path KEYS_PATH [-i INPUT_FILE_PATH]
                   [-o OUTPUT_FILE_PATH]
                   command
```

commands

- **get**: Get remote file
- **put**: Put file to remote
- **create**: Create text file on remote
- **file**: Print remote text file
- **listdir**: List remote directory

optional arguments:

```
-h, --help            - Show this help message and exit
--keys_path KEYS_PATH - Credentials file path
-i INPUT_FILE_PATH    - Input file path | input string
-o OUTPUT_FILE_PATH   - Output file path
```

1.3.3 ssh_command

Simple remote ssh command

```
ssh_command [-h] --keys_path KEYS_PATH [command [command ...]]
```

```
command               - Remote command

-h, --help            - show this help message and exit
--keys_path KEYS_PATH - Credentials file path
```

1.3.4 slurm_test

Complete set of functions to manage slurm submissions remotely

```
slurm_test [-h] --keys_path KEYS_PATH [--script SCRIPT_PATH]
                  [--local_data LOCAL_DATA_PATH] [--remote REMOTE_PATH]
                  [--queue_settings Q_SETTINGS] [--module MODULE]
                  [--task_data TASK_FILE_PATH]
                  command
```

Command

- **submit**: Submit job
- **queue**: Check queue status
- **cancel**: Cancel submitted job
- **status**: Check job status
- **get_data**: Download remote files
- **put_data**: Upload local files to remote
- **log**: Get log files (stdout, stderr)
- **get_file**: Get single remote file

optional arguments:

```
-h, --help                      - show this help message and exit
--keys_path KEYS_PATH           - Credentials file path
--script LOCAL_RUN_SCRIPT       - Path to local script
--local_data LOCAL_DATA_PATH    - Local data bundle
--remote REMOTE_PATH            - Remote working dir
--queue_settings QUEUE_SETTINGS - Predefined queue settings
--modules MODULES               - Software modules to load
--task_data_file TASK_FILE_PATH - Store for task data
--overwrite                     - Overwrite data in output local directory
--task_file_type TASK_FILE_TYPE - Format for task data file (json, pickle). Default:json
--poll POLLING_INT              - Polling interval (seg), 0: No polling (default)
--remote_file REMOTE_FILE       - Remote file name to download (get_file)
```

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

THREE

GITHUB REPOSITORY.

PYTHON MODULE INDEX

b

INDEX

R

S

T