jasmin-arc-py Documentation

Release

CEDA

Nov 29, 2017

Contents:

`jasmin_arc` is a Python library to facilitate running jobs on LOTUS on JASMIN via the ARC-CE server.

It supports:

- Submitting jobs
- Uploading input files to be processed by jobs
- Retrieving job status
- Downloading job outputs
- Cancelling jobs

To get started, obtain the necessary certificates by following the instructions here.

Hint: See the help pages for more information about using ARC-CE on JASMIN.

CHAPTER 1

Installation

Use `pip` for installation:

```
git clone https://github.com/cedadev/jasmin-arc-py.git
cd jasmin-arc-py
pip install .
```

Alternatively, to install straight from GitHub:

```
pip install git+https://github.com/cedadev/jasmin-arc-py
```

`jasmin_arc` also requires the ARC client Python library to installed. On CentOS 6:

```
# Remove the epel cache if any:
rm -rf /var/cache/yum/x86_64/6/epel
rpm -Uvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
rpm -Uvh  http://download.nordugrid.org/packages/nordugrid-release/releases/15.03/
↪centos/el6/x86_64/nordugrid-release-15.03-1.el6.noarch.rpm
yum update
yum install  nordugrid-arc-client-tools
# Check installation:
arcinfo --version
```

On Ubuntu:

```
wget -q http://download.nordugrid.org/DEB-GPG-KEY-nordugrid.asc -O- | sudo apt-key␣
↪add -
apt-get update
apt-get install nordugrid-arc-python
apt-get install globus-gsi-cert-utils-progs
```

If working in a virtual environment you may also need to add the location of the ARC installation to your `PYTHONPATH`:

```
export PYTHONPATH=/usr/lib64/python2.6/site-packages:$PYTHONPATH
```

Note: The location of your ARC installation may be different from the example given here.

CHAPTER 2

API Usage

2.1 Quickstart

To get started, create a JSON config file that points to your private key and certificate (see *Configuration* for the full list of config options):

```
{
  "CLIENT_KEY": "/path/to/private/key.pem",
  "CLIENT_CERT": "/path/to/cert.pem",
}
```

All actions are performed through the `ArcInterface` class. A basic example is included below:

```
"""
Submit a job, wait till it completes, and print the stdout output
"""
import time
import os.path
from jasmin_arc import ArcInterface, JobStatuses

arc_iface = ArcInterface("/path/to/config.json")
job_id = arc_iface.submit_job("/bin/bash",
                              args=["-c", "echo 'This job is running on `/bin/
↪hostname`'"])

while True:
    status = arc_iface.get_job_status(job_id)
    print("Job status is {}".format(status))

    if status == JobStatuses.COMPLETED:
       print("Job is finished!")
       break

    time.sleep(1)
```

```
out_dir = arc_iface.save_job_outputs(job_id)

if out_dir:
    print("Saved job outputs to {}".format(out_dir))
    print("stdout was:")
    with open(os.path.join(out_dir, "stdout.txt")) as stdout:
       print(stdout.read())
else:
    print("Error saving job outputs")
```

See *Examples* for more examples.

2.2 API

`ArcInterface` contains methods for interacting with ARC on JASMIN. The most useful methods are listed below:

class jasmin_arc.arc_interface.**ArcInterface**(*config_path=None*, *log=<open file '<stdout>', mode 'w'>*, *log_level=<LogLevels.INFO: <class 'sphinx.ext.autodoc.INFO'>>*)

Class to handle interactions with the ARC-CE server

__init__(*config_path=None*, *log=<open file '<stdout>', mode 'w'>*, *log_level=<LogLevels.INFO: <class 'sphinx.ext.autodoc.INFO'>>*)

Create an object to interface with the ARC server.

Parameters

- **config_path** – Path to config JSON file, or `None` to use the default settings
- **log** – File-like object to write log messages to, or `None` to disable logging. Use `sys.stdout` or `sys.stderr` to print messages (default: `sys.stdout`).
- **log_level** – The level of detail logs should show (default: `LogLevels.INFO`). See `LogLevels` for the available levels

Raises **InvalidConfigError** – if config is not valid JSON or is otherwise invalid

cancel_job(*job_id*)

Cancel the given job

Parameters **job_id** – ID of the job as returned by `submit_job`

Raises **JobNotFoundError** – if no job with the given ID could be found

get_job_status(*job_id*)

Return the status of the given job

Parameters **job_id** – ID of the job as returned by `submit_job`

Raises **JobNotFoundError** – if no job with the given ID could be found

Returns The status of the job (see `JobStatuses` for the available values)

save_job_outputs(*job_id*)

Retrieve output files from a job and save them to a temp directory. The file/directory specified in `OUTPUT_FILE` will be downloaded, and `stdout` and `stderr` outputs are saved as `stdout.txt` and `stderr.txt` respectively.

Parameters **job_id** – ID of the job as returned by `submit_job`

Raises **JobNotFoundError** – if no job with the given ID could be found

Returns Path to the directory the output files were saved in, or `None` if no files were saved

submit_job(*executable*, *args=[]*, *input_files=[]*)
Submit a job and return the job ID

Parameters

- **executable** – The command to run on the LOTUS cluster
- **args** – List of arguments to pass to the executable
- **input_files** – A list of paths to local files to copy to the remote session directory (the directory the job will run from on JASMIN)

Raises

- **InputFileError** – if any of the specified input files do not exist or are directories
- **NoTargetsAvailableError** – if no execution targets can be found on the ARC server
- **JobSubmissionError** – if the job cannot be submitted to any targets

Returns Job ID

2.3 Configuration

`jasmin_arc` uses a JSON file to configure the connection to the ARC CE server. The available options and default values are defined in the `ConnectionConfig` class:

class `jasmin_arc.config.`**ConnectionConfig**(*config_dict*, *logger=None*)
Class to define available config options and their default values

ARCPROXY_PATH = '/usr/bin/arcproxy'
Path to the `arcproxy` binary, which is used to generate a proxy certificate from the private key and certificate

ARC_SERVER = 'jasmin-ce.ceda.ac.uk:60000/arex'
URL to the ARC server

CERTS_DIR = '/etc/grid-security/certificates'
Path to directory containing trusted CA certificates

CLIENT_CERT = '~/.arc/usercert.pem'
Path to grid certificate file

CLIENT_KEY = '~/.arc/userkey-nopass.pem'
Path to the private key file associated with your grid certificate

JOBS_INFO_FILE = '~/.arc/jobs.dat'
Path to job information file used by ARC client tools (arcstat, arcget etc) to load information about submitted jobs

OUTPUT_FILE = 'output'
The name of the file/directory to download when retrieving job outputs.

PROXY_FILE = '/tmp/arcproxy_file'
Path to save the generated proxy certificate to

PROXY_RENEWAL_THRESHOLD = 10
The number of seconds the proxy file can have till expiry before a new proxy is automatically generated

PROXY_VALIDITY_PERIOD = 43200
Number of seconds to set the validity period to when generating a proxy file (default: 12 hours)

For any options not included in the JSON config, the default values shown above will be used. For example, to use the default options except for the path to your private key and the ARC server URL, use the following JSON:

```
{
  "CLIENT_KEY": "/my/private/key",
  "ARC_SERVER": "my-arc-server.ac.uk"
}
```

2.4 Job input/output files

Jobs submitted to LOTUS through ARC-CE are run from a *session directory* unique to each job. The path to this directory will be something like `/work/scratch/arc/grid/RmCMDmqK9brnMQPDjq2vDwNoABFKDmABFKDmvpFKDmWEFKDmnXLGem`, where the long string of numbers at the end is the last component of the job ID.

Note: You do not need to worry about the actual path to the session directory, as the current working directory will be set correctly when your jobs run.

Any input files passed to `ArcInterface.submit_job` are copied into this directory, and can be accessed from your jobs.

Use the `OUTPUT_FILE` config option to specify which file/directory to download with `ArcInterface.save_job_outputs`. This will download the specified file/directory, and also the contents of `stdout` and `stderr` to `stdout.txt` and `stderr.txt` respectively.

Outputs are saved to a temporary directory (in `/tmp` on UNIX platforms), and the path to this directory is returned. You may then move files to a more permanent location as required.

Note: Any other files written to the session directory will be deleted when the job finishes.

2.5 Examples

Setting log output destination and logging level:

```
# Log to a file
with open("/tmp/jasmin_arc.log", "w") as logfile:
    arc_iface = ArcInterface("/path/to/config", log=logfile)

# Change log level to DEBUG (shows a lot of output!)
arc_iface = ArcInterface("/path/to/config", log_level=LogLevels.DEBUG)

# Disable logging
arc_iface = ArcInterface("/path/to/config", log=None)
```

Uploading and processing input files:

```
with open("/tmp/f1.txt", "w") as f1:
    f1.write("This is file 1... ")

with open("/tmp/f2.txt", "w") as f2:
    f2.write("And this is file 2")

arc_iface.submit_job("/bin/bash", args=["-c", "cat *.txt"],
                     input_files=["/tmp/f1.txt", "/tmp/f2.txt"])
```

Downloading output files

```
###################
#   config.json   #
###################
{
    "OUTPUT_FILE": "output.txt",
    ...
}

#################
#   script.py   #
#################
import time
from jasmin_arc import ArcInterface, JobStatuses

arc_iface = ArcInterface("/path/to/config.json")

script = "echo 'This will be deleted' > myfile.txt;" + \
         "echo 'This will be downloaded' > output.txt;" + \
         "ls -l"
job_id = arc_iface.submit_job("/bin/bash", args=["-c", script])

# Wait for job to complete
while arc_iface.get_job_status(job_id) != JobStatuses.COMPLETED:
    time.sleep(1)

out_dir = arc_iface.save_job_outputs(job_id)
print("Outputs saved to {}".format(out_dir))

# The directory `out_dir' now contains 3 files - `output.txt', `stdout.txt' and
# `stderr.txt'. Checking the contents of `stdout.txt' we can see that when the
# job ran there were 4 files, but `myfile.txt' was deleted when the job ended.
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index

Symbols

A

C

G

J

O

P

S