jobtronaut Documentation

Release 0.0.1

Trixter GmbH

May 15, 2023

Contents

Jobtronaut is an extension of the Tractor API for authoring extensive and reusable task hierarchies in a flexible fashion.

Contents:

CHAPTER 1

Prerequisites

Jobtronaut was developed and tested within a Linux environment using Tractor 2.2 1715407 with **Python 2.7**. Besides the tractor python api itself it requires schema as external dependency.

CHAPTER 2

Configuration

To make **Jobtronaut** fit into your environment nicely you can define your own configuation. This is supposed to be a python file and its full filepath need to be made available via the *JOBTRONAUT_PLUGIN_PATH* env var.

Table 1: You can define the following variables within your configuration file to make Jobtronaut fit into your Pipeline.

Variable Name	Type	Description	Default if not defined
COMMANDFLAGS_ARGUMENT_NAME	str	A string that will be used with the given name that can be used to inject additional flags into any command. This is especially useful if you are using an application bootstrap framework.	*additional_command_flags*
LOGGING_NAMESPACE	str	A custom namespace that can be used instead of the default one.	*jobtronaut*
ENABLE_PLUGIN_CACHE	bool	If True it will reuse plugins from its own cache instead of resourcing modules when initializing the Plugins discovery.	*True*
PLUGIN_PATH	list	A list of directories Jobtronaut's Plugins discovery mechanism will use. All .py files in the given directories will be considered.	`[]`
EXECUTABLE_RESOLVER	callable	A callable that can be used to resolve a executable declared in within any of your Command tasks. The first item of *Task.cmd()* will be passed to the callable.	`lambda x: x if os.path.isfile(x) and os.path.isabs(x) and os.access(x, os.X_OK) else (_ for _ in ()).throw(OSError("Command Id `{}` is not an executable.".format(x)))`
ENVIRONMENT_RESOLVER	callable	A callable that can be used to resolve an environment. This will be ignored in case *INHERIT_ENVIRONMENT* is set to False.	`lambda: OrderedDict(sorted(os.environ.items()))`
INHERIT_ENVIRONMENT	bool	If set to True it will use the *ENVIRONMENT_RESOLVER* to add its return value to the envkey attribute of any resulting job.	*True*
ARGUMENTS_SERIALIZED_MAX_LENGTH	int	The maximum number of characters a serialized Arguments object can have within a command. It it exceeds this limit the serialized Arguments will be dumped into a file within the defined *ARGUMENTS_STORAGE_PATH*.	*10000*
ARGUMENTS_STORAGE_PATH	str	A directory path where serialized Arguments objects can be dumped.	
JOB_STORAGE_PATH_TEMPLATE	str	Path in which all job representation files will be dumped whenever a job was submitted.	
KATANA_SCRIPT_WRAPPER	str	Path to a custom python file that acts like a wrapper for Katana to allow consume python code directly.	`<JOBTRONAUT_DIR>/author/scripts/katanascript.py`
MAYA_SCRIPT_WRAPPER	str	Path to a custom python file that acts like a wrapper for Maya to allow consume python code directly.	`<JOBTRONAUT_DIR>/author/scripts/mayascript.mel`
NUKE_SCRIPT_WRAPPER	str	Path to a custom python file that acts like a wrapper for Nuke to allow consume python code directly.	`<JOBTRONAUT_DIR>/author/scripts/nukescript.py`
TRACTOR_ENGINE_CREDENTIALS_RESOLVER	callable	A callable that needs to be set whenever internal processes will use the tractor query api or you will use any of the jobtronaut.query features directly. This callable needs to return a valid Tractor user and password with proper permissions.	`lambda: ("unknown_user", "unknown_password")`

CHAPTER 3

Getting Started!

Contents:

3.1 Build your first Job

3.2 Submit your first Job

3.3 Expand your first Task

3.4 Advanced Examples

Contents:

3.4.1 Chunking Example

CHAPTER 4

Authoring

Contents:

4.1 Jobs

4.2 Tasks

4.3 Processors

4.4 Arguments

CHAPTER 5

Commandline Interface

We provide a commandline interface to easily submit any available task to Tractor or retrieve information about tasks and processors.

5.1 Usage

Available Subcommands:

```
>> jobtronaut -h
usage: -c [-h] {info,visualize,list,submit} ...

positional arguments:
  {info,visualize,list,submit}
    submit              Submit a job to the farm.
    list                Can list all known plugins.
    info                Get more info on a specific plugin.
    visualize           Visualize the resulting task tree.

optional arguments:
  -h, --help            show this help message and exit
```

jobtronaut submit:

```
>> jobtronaut submit -h
usage: -c submit [-h] [--paused] --task TASK [--title TITLE]
                 [--comment COMMENT] [--service SERVICE]
                 [--afterjids JIDS] [--priority PRIORITY]
                 [--tags TAGS [TAGS ...]] [--projects PROJECTS [PROJECTS ...]]
                 [--args ARGNAME:ARGVALUE [ARGNAME:ARGVALUE ...]]
                 [--env ENVVAR:ENVVALUE [ENVVAR:ENVVALUE ...]]

optional arguments:
  -h, --help            show this help message and exit
```

(continues on next page)

(continued from previous page)

```
  --paused              Submit the job in a paused state.
  --local               Submit the job locally. All commands will be enforced
                        to run on the spoolhost.
  --task TASK           Set the root task for the job.
  --title TITLE         Set a custom job title.
  --comment COMMENT     Set a job comment.
  --service SERVICE     Specify a hostmask to limit the blades this job can
                        run on.
  --afterjids JIDS      Only start the job when the jobs with these ids are
                        done.
  --priority PRIORITY   Set the priority of the job.
  --maxactive MAXACTIVE Limit simultaneous active render nodes. Default value
                        is 0 (no limit)
  --tags TAGS [TAGS ...]
                        Speficy custom limit tags on the job.
  --projects PROJECTS [PROJECTS ...]
                        Specify the projects of the job.
  --args ARGNAME:ARGVALUE [ARGNAME:ARGVALUE ...]
                        Job Arguments. Supported value types are: str, int,
                        float, list
  --env ENVVAR:ENVVALUE [ENVVAR:ENVVALUE ...]
                        Custom environment variables that will be set prior to
                        a command's execution on the farm
```

jobtronaut list:

```
>> jobtronaut list -h
usage: -c list [-h] {all,tasks,processors,sitestatusfilters}

positional arguments:
  {all,tasks,processors,sitestatusfilters}
                        Define which plugins you want to list.

optional arguments:
  -h, --help            Show this help message and exit
  --info                Show the detailed information for every plugin.
```

jobtronaut info:

```
>> jobtronaut info -h
usage: -c info [-h] plugin

positional arguments:
  plugin      Specify the plugin name for which you want more information.

optional arguments:
  -h, --help  show this help message and exit
```

jobtronaut arguments:

```
>> jobtronaut arguments -h
usage: -c arguments [-h] [--filter FILTER] search

positional arguments:
  search           A task id to extract the arguments object from.

optional arguments:
```

(continues on next page)

(continued from previous page)

```
  -h, --help       show this help message and exit
  --filter FILTER  A regex pattern to filter argument names. Default: '.*'
```

CHAPTER 6

Utilities

CHAPTER 7

Release Notes