
jobtronaut Documentation

Release 0.0.1

Trixter GmbH

Apr 20, 2021

Contents

| | | |
|----------|----------------------------------|-----------|
| 1 | Prerequisites | 3 |
| 2 | Configuration | 5 |
| 3 | Getting Started! | 7 |
| 3.1 | Build your first Job | 7 |
| 3.2 | Submit your first Job | 7 |
| 3.3 | Expand your first Task | 7 |
| 3.4 | Advanced Examples | 7 |
| 4 | Authoring | 9 |
| 4.1 | Jobs | 9 |
| 4.2 | Tasks | 9 |
| 4.3 | Processors | 9 |
| 4.4 | Arguments | 9 |
| 5 | Commandline Interface | 11 |
| 5.1 | Usage | 11 |
| 6 | Utilities | 13 |
| 7 | Release Notes | 15 |

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 configuration. 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 |
|-------------------------------------|----------|---|---|
| COMMAND_FLAGS_ARGUMENT_NAME | string | Even name that can be used to inject additional flags into any command. This is especially useful if you are using an application bootstrap framework. | <i>additional_command_flags</i> |
| LOGGING_NAMESPACE | string | Namespace that can be used instead of the default one. | <i>jobtronaut</i> |
| ENABLE_PLUGIN_CACHE | boolean | Use plugins from its own cache instead of resourcing modules when initializing the Plugins discovery. | <i>True</i> |
| PLUGIN_PATH | list | list of directories Jobtronaut's Plugins discovery mechanism will use. All .py files in the given directories will be considered. | [] |
| EXECUTABLE_RESOLVER | callable | Callable that can be used to resolve a executable declared in within any of your Command tasks. The first item of <i>Task.cmd()</i> will be passed to the callable. | <pre>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)))</pre> |
| ENVIRONMENT_RESOLVER | callable | Callable that can be used to resolve an environment. This will be ignored in case <i>INHERIT_ENVIRONMENT</i> is set to False. | <pre>lambda: OrderedDict(sorted(os.environ.items()))</pre> |
| INHERIT_ENVIRONMENT | boolean | True it will use the <i>ENVIRONMENT_RESOLVER</i> to add its return value to the envkey attribute of any resulting job. | <i>True</i> |
| ARGUMENTS_SERIALIZED_MAX_LENGTH | integer | The maximum number of characters a serialized Arguments object can have within a command. If it exceeds this limit the serialized Arguments will be dumped into a file within the defined <i>ARGUMENTS_STORAGE_PATH</i> . | <i>10000</i> |
| ARGUMENTS_STORAGE_PATH | string | Where serialized Arguments objects can be dumped. | |
| JOB_STORAGE_PATH_TEMPLATE | string | Where job representation files will be dumped whenever a job was submitted. | |
| KATANA_SCRIPT_WRAPPER | string | Python file that acts like a wrapper for Katana to allow consume python code directly. | <JOBTRONAUT_DIR>/author/scripts/katanascript.py |
| MAYA_SCRIPT_WRAPPER | string | Python file that acts like a wrapper for Maya to allow consume python code directly. | <JOBTRONAUT_DIR>/author/scripts/mayascript.mel |
| NUKE_SCRIPT_WRAPPER | string | 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 | 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. | <pre>lambda: ("unknown_user", "unknown_password")</pre> |

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

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.
--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}

positional arguments:
  {all,tasks,processors}
                        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

```


CHAPTER 6

Utilities

CHAPTER 7

Release Notes
