pypelined Documentation

Release 0.1.3

Max Fischer

Nov 15, 2017

Documentation Topics Overview:

CHAPTER 1

pypelined Service Templates

The *pypelined* package is ready for use as a daemon, but does not attempt to provide definitions for every system service. Depending on the target system, adjustments for deployment into virtual environments and similar are required.

1.1 systemd

```
# /etc/systemd/system/pypelined@.service
[Unit]
Description=stream and pipeline processing service
Documentation=https://pypi.python.org/pypi/pypelined

[Service]
Type=simple
Restart=on-failure
ExecStart=/usr/bin/python -m pypelined /etc/pypelined/%i*.py
User=daemon
Nice=-19

[Install]
WantedBy=multi-user.target
DefaultInstance=default
```

CHAPTER 2

pypelined Changelog

2.1 Beta Releases

2.2 v0.1.2 2017-07-20

Bugfixes

AliceApMonBackend: added guard against invalid ApMon INSTANCE_ID

AliceApMonBackend: directly forwarding raw reports

2.3 v0.1.1 2017-06-30

Bugfixes

DFSCounter: tweaked acquisition to avoid deadlocks and stale counters

Debug output removed

Missing dependencies added

2.4 v0.1.0 2017-??-??

New Features

Basic pypelined core in working condition

CHAPTER 3

pypelined

3.1 pypelined package

3.1.1 Subpackages

pypelined.conf package

pypelined.conf.**pipelines** = []
pipelines to run

Submodules

pypelined.conf.loader module

pypelined.conf.logger module

pypelined.conf.logger.**configure_logging**(*log_level*, *log_format*, *log_dest*)
Configure logging from CLI options

Parameters

- **log_level** (*int or str*) – logging verbosity
- **log_format** (*str*) – format string for messages
- **log_dest** (*tuple[str]*) – where to send log message to

Each element in *destinations* must be either a stream name (*"stdout"* or *"stderr"*), or it is interpreted as a file name.

pypelined.consumer package

Submodules

pypelined.consumer.alice_apmon module

pypelined.consumer.socket module

pypelined.consumer.telegraf module

pypelined.modifier package

Submodules

pypelined.modifier.dictlets module

pypelined.provider package

Submodules

pypelined.provider.stream module

pypelined.provider.xrootd module

pypelined.utilities package

pypelined.utilities.**safe_eval**(*literal*)
Evaluate a literal value or fall back to string

Safely performs the evaluation of a literal. If the literal is not valid, it is assumed to be a regular string and returned unchanged.

Parameters **literal** (*str*) – literal to evaluate, e.g. *"1.0"* or *"{'foo': 3}"*

Returns evaluated or original literal

Submodules

pypelined.utilities.dfs_counter module

pypelined.utilities.proctools module

pypelined.utilities.proctools.**validate_process**(*pid*, *name=None*)
Check whether there is a process with *pid* and *name*

Parameters

- **pid** – pid of the running process
- **name** (*str or None*) – name of the running process

Returns whether there is a process with the given name and pid

Return type bool

pypelined.utilities.singleton module

class pypelined.utilities.singleton.**Singleton**
Bases: object

Basic implementation of a Singleton

Any instances constructed with the same parameters are actually the same object. Use __singleton_signature__() to specify which parameters define identity.

3.1.2 Submodules

pypelined.driver module

The *pypelined* service and framework lets you build and deploy iterative processing pipelines. Using generator/coroutines with the chainlet library, it is trivial to create pipelines to fetch, process and transform streams of data. Configuration files are written using pure Python, allowing for maximum customization:

```
# this is a pure python configuration file
from chainlet import funclet, filterlet
from pypelined.conf import pipelines

# new pipeline processing element as simple python function
@funclet
def add_time(chunk):
    chunk['tme'] = time.time()
    return chunk

# new pipeline receiving process monitoring reports, modifying them, and sending them␣
↪to another service
process_chain = Socket(10331) >> decode_json() >> filterlet(lambda value: value.get(
↪'rcode') == 0) >> \
    add_time() >> Telegraf(address=('localhost', 10332), name='valid_processes')
# add pipeline for deployment
pipelines.append(process_chain)
```

```
python -m pypelined myconfig.py
```

CHAPTER 4

Contributing and Feedback

The project is hosted on github. If you have issues or suggestion, check the issue tracker: For direct contributions, feel free to fork the development branch and open a pull request.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Documentation built from chainlet 0.1.3 at Nov 15, 2017.

Python Module Index

p

Index