bandoleers

Release 3.3

Aug 23, 2022

Contents

This package contains useful command line tools that make deployment and local data bootstrapping less painful.

prep-it This simple utility iterates over data files in a sub-tree and loads them into various backends such as RabbitMQ, Consul, and PostgreSQL.

wait-for This simple utility pings a URL periodically until it responds successfully. It supports HTTP and Postgres URLs out of the box.

CHAPTER 1

Quickstart Development Guide

```
$ python3.4 -mvenv env
$ . ./env/bin/activate
(env) $ pip install -qr requires/development.txt
```

CHAPTER 2

Documentation

2.1 prep-it

The **prep-it** utility scans the *platform* sub-directory and loads data files into various backends. The following sections describe each directory name that is supported and its expected content.

```
prep-it ([-q|--quiet] | [-v|--verbose]) [-d|--dir DIRECTORY]
prep-it (-h | --help | --version)
```

DIRECTORY
Specifies the directory to process. If unspecified, the "platform" directory is processed.

-v, --verbose
Write diagnostics to standard output. Without this flag, informational messages will be displayed.

-q, --quiet
Only show output when an error occurs.

-h, --help
Display a usage synopsis and exit with a failure status.

--version
Display the package version and exit with a failure status.

2.1.1 consul

JSON files containing top-level object definitions (e.g., dictionaries) that are loaded into a Consul key-value store using a `consulate.Consul` instance. The Consul endpoint is configured by setting the *CONSUL_HOST* and *CONSUL_PORT* environment variables.

2.1.2 http

JSON files containing an array of objects, one per request. Each object is simply a list of parameters passed into `requests.request()`. The following properties are usually what you want:

url The targetted resource. This may be modified as described below.

method The HTTP method to request.

params Optional dictionary of query parameters.

headers Optional dictionary of headers to send.

json Optional body that will be JSON encoded before being sent.

The `url` value may contain environment variables in the host and/or port number values. For example `http://$CONSUL_HOST:$CONSUL_PORT/...` is rewritten to `http://127.0.0.1:37832` if the *CONSUL_HOST* environment variable is set to `127.0.0.1` and the *CONSUL_PORT* environment variable is set to `37832`.

The user name and password parameters are removed from the URL and placed into the `auth` keyword parameter if they are present in the URL.

2.1.3 rabbitmq

JSON files that contain RabbitMQ HTTP API commands to execute. Each command is represented by an object with the following properties:

path The resource to send the request to.

method The HTTP method to invoke (e.g., `POST`, `DELETE`)

body The body to send with the request.

The RabbitMQ server is identified by setting the *RABBITMQ* environment variable to the host and port of the HTTP API endpoint.

2.1.4 redis

JSON files each containing a top-level object definition where each property names a redis command. The property value is another object definition where the name is the redis key and the value is a list of values to pass to the command.

For example, the following JSON file would result in calling the `SADD` redis command to add `"abuse"`, `"admin"`, `"postmaster"`, and `"root"` to the `admin_type_address` redis set.

```
{
   "SADD": {
      "admin_type_address": [
         "abuse",
         "admin",
         "postmaster",
         "root"
      ]
   }
}
```

The redis server is configured by setting the *REDIS_URI* environment variable to a redis url.

2.1.5 postgres

SQL files that are executed using queries. The database server is configured by setting the *PGSQL* environment variable. The database name is based on the file name minus the assumed `.sql` suffix. The database will be dropped if it exists and then created anew before running the SQL commands from the file.

The database connection for a specific database can also be specified by setting the `PGSQL_$DBNAME` environment variable where `$DBNAME` is the name of the database in upper-case. If a database specific environment variable exists, **then the database will not be created automatically.**

2.2 wait-for

The **wait-for** utility polls a URL periodically until it gets a successful response.

```
wait-for ([-q|--quiet] | [-v|--verbose]) [-s|--sleep SECONDS]
         [-t|--timeout SECONDS] URL
wait-for (-h | --help | --version)
```

URL
Specifies the URL to poll. The following schemes are supported:

`http`, `https`	Fail for non-200 family status codes
`postgresql`	Fail unless connecting psycopg2 to the URL succeeds.
`tcp`	Fail unless connecting a TCP socket succeeds.

-s <seconds>, **--sleep** <seconds>
Sleep for the specified number of seconds between hitting the URL.

-t <seconds>, **--timeout** <seconds>
Timeout and exit unsuccessfully if a successful response is not received within *timeout* seconds.

-v, --verbose
Write diagnostics to standard output. Without this flag, the utility is quite silent.

-q, --quiet
Only show output when an error occurs.

-h, --help
Display a usage synopsis and exit with a failure status.

--version
Display the package version and exit with a failure status.

2.3 Environment Variables

CONSUL_HOST
Identifies the IP address or DNS name of the Consul server.

CONSUL_PORT
Identifies the TCP port number that the Consul server is listening on.

RABBITMQ
The network location portion for the RabbitMQ HTTP API. This is inserted directly into a HTTP URL.

REDIS_URI
Identifies the redis server, port, and database to connect to. This value follows the IANA-registered redis url format.

PGSQL
Identifies the PostgreSQL server to connect to using the standard postgresql:// scheme

PGSQL_...
Identifies the PostgreSQL connection to use for the specific database named by the suffix using the standard postgresql:// scheme

2.4 Release History

2.4.1 Next Release

2.4.2 3.3.1 (2022-08-23)

- Loosen redis pin to allow redis 4
- Switch to GitHub Actions for CI

2.4.3 3.3.0 (2021-09-02)

- Replace consulate usage with requests
- Add classifiers for 3.8 & 3.9.

2.4.4 3.2.0 (2019-04-05)

- Use `psycopg2` instead of `queries`

2.4.5 3.1.0 (2019-03-27)

- Expand supported redis package versions

2.4.6 3.0.0 (2018-02-01)

- Upgrade to queries 2.0.x.
- Drop support for Python 2.6.

2.4.7 2.1.0 (2017-10-28)

- Add raw HTTP support to `prep-it`
- Ignore data files that look like temporary files. This specifcally ignores files that start with a period and files that end with a tilde.

2.4.8 2.0.0 (2017-05-22)

- Remove Cassandra support

2.4.9 1.1.1 (2016-08-17)

- Support Python 2.6

2.4.10 1.1.0 (2016-04-13)

- Make the platform directory name an option for `prep-it`
- Add `--sleep` parameter to `wait-for`
- Normalize parameter processing between commands
- Do not create a postgres database if the database-specific environment variable exists.
- Add support for `tcp://` in `wait-for`

2.4.11 1.0.0 (2016-01-27)

- Infect the world!

2.4.12 0.3.5 (2016-01-26)

- Query parameters are now passed from cassandra:// URLs into the cluster instance.

2.4.13 0.3.4 (2016-01-14)

- Add postgres support in `wait-for`

2.4.14 0.3.3 (2015-10-08)

- The default Redis DB must be an integer

2.4.15 0.3.0 (2015-09-15)

- Read and execute the entire SQL file when bootstrapping Postgresql

2.4.16 0.2.1 (2015-09-14)

- Remove `codecs` from `setup.py` for Python 3 compatibility.

2.4.17 0.2.0 (2015-08-05)

- Added the `wait-for` utility

2.4.18 0.1.0 (2015-06-23)

- Initial release of the PrepIt package
- Import @briank's work on prepit.

2.5 How to Contribute

Do you want to contribute fixes or improvements?

> **AWesome!** *Thank you very much, and let's get started.*

2.5.1 Set up a development environment

The first thing that you need is a development environment so that you can run the test suite, update the documentation, and everything else that is involved in contributing. The easiest way to do that is to create a virtual environment for your endevours:

```
$ python3.4 -mvenv env
```

Don't worry about writing code against previous versions of Python unless you you don't have a choice. If you don't have a choice, then install virtualenv to create the environment instead. The next step is to install the development tools that this project uses. These are listed in *requires/development.txt*:

```
$ env/bin/pip install -qr requires/development.txt
```

At this point, you will have everything that you need to develop at your disposal. *setup.py* is the swiss-army knife in your development tool chest. It provides the following commands:

./setup.py build_sphinx Generate the documentation using sphinx.

./setup.py flake8 Run flake8 over the code and report style violations.

If any of the preceding commands give you problems, then you will have to fix them **before** your pull request will be accepted.

2.5.2 Submitting a Pull Request

Once you have made your modifications and added any necessary documentation, it is time to contribute back for posterity. You've probably already cloned this repository and created a new branch. If you haven't, then checkout what you have as a branch and roll back *master* to where you found it. Then push your repository up to github and issue a pull request. Describe your changes in the request and someone will review it, and eventually merge it and release a new version.

Index

Symbols

C

D

E

P

R

U

W