scrapydd Documentation

Release 0.7.4

kevenli

Sep 16, 2020

Contents

Scrapydd (Scrapy Distributed Deamon) is a distributed scrapy spiders scheduling system. On the scrapydd system, you can hold and control versions of spider project eggs, schedule spiders, watch job history and logs. It can be also scale out easily.

Contents:

CHAPTER 1

Installation

1.1 Requirements

- **tornado** For async programming and the web server.
- **apscheduler** Internal scheduling engine.
- **sqlalchemy** Data accessing
- **sqlalchemy-migrate** Database migrations.

1.2 Installing Scrapydd

By pip:

```
pip install scrapydd
```

You can also install scrapydd manually:

1. Download compressed package from github releases.
2. Decompress the package
3. Run `python setup.py install`

CHAPTER 2

Node Registration

Scrapydd is a distributed system, mainly consist of two two roles: server and worker (also known as agent).

System administrator can add/remove node with the WebUI and some commands on the node machine.

2.1 Concepts

There are two types of workers:

Temporary Node: This type of worker runs without the need of pre-registration, and once it is closed for any reason, a new node will be created when it goes online again.

Permanent Node: This type of worker needs to be registered before it can run. Each time its process go online back, it relates to the same Node.

Temporary node is intended to be use in small deployment to provide easier way to use, along with the *enable_authentication* settings set to *false*. It is *STRONGLY RECOMMENDED* to enable authentication for any business use.

2.2 Registration Process

To register a Permanent Node, the Sys Admin need to operate the follow processes.

1. Go to the *Admin Area* - *Nodes* Page.
2. Copy *New Node Key* and *SecretKey*. Each pair of keys will last few minutes.
3. Go to the worker machine, first check the *server*, *tags* settings under the *agent* section are set to appropriate value.
4. Run *scrapydd agent -g*, input *node_key* and *secretkey* following the prompt.

CHAPTER 3

Webhook

Webhook help to support system integrations. When a spider job is completed, the server will start to send crawled data to a customized url.

The webhook post data to `payload_url`, each key/value field is urlencoded before post, unicode data will be treated as UTF8 encoding, and if the value is dict/tuple/list, it will be json enconded. One request for each crawled item.

The frequency of posting data would be no more than 1 request/second.

You can modify spider's webhook settings list this:

```
curl -XPOST http://localhost:6800/projects/{projectname}/spiders/{spidername}/webhook␣
↪-d payload_url = {address}
```

Or to delete an existing webhook:

```
curl -XDELETE http://localhost:6800/projects/{projectname}/spiders/{spidername}/
↪webhook
```

CHAPTER 4

Configuration

Both server and agent use the `scrapydd.conf` file for system configuration. The file will be looked up in the following locations:

- /etc/scrapydd/scrapydd.conf
- /etc/scrapyd/conf.d/*
- ./scrapydd.conf
- ~/.scrapydd.conf

Config can be also overriden by environment variables, environment variables should have a "**SCRAPYDD_**" prefix and then the config name with upper case. For example to override a server address on agent, an environment variable should be "SCRAPYDD_SERVER=xxx".

4.1 Server

Server configurations should appears under the `[server]` section.

4.1.1 bind_address

The ipaddress which web server bind on. Default: `0.0.0.0`

4.1.2 bind_port

The port web server running on. Default: `6800`

4.1.3 client_validation

Whether validate client's certificate on SSL, Default: `false`

4.1.4 database_url

Database connection url. This will be passed to the inside sqlalchemy *create_engine* method. Default: `sqlite:///database.db`

4.1.5 debug

Whether run server on debug mode. Debug mode will set logging level to DEBUG. Default: `false`.

4.1.6 enable_authentication

Whether enable authentication, once this option is on, user need to login to make operation. Default: `true`

4.1.7 https_port

HTTPS port to listen on, specify this key will enable SSL mode.

Default: None

4.1.8 runner_type

Project package runner, Default: venv.

Available options: venv(run sub-command on VirtuanEnv), docker (run sub-command on Docker container)

4.1.9 runner_docker_image

Runner container image name, Default: kevenli/scrpaydd

This effects when runner_type is docker.

4.1.10 server_name

Server's hostname. When SSL enabled, the public certificate will be loaded as filename *server_name*.crt and private certificate will be loaded as filename *server_name*.key in the `keys` folder. Default: `localhost`

4.2 Agent

Agent configurations should appears under the `[agent]` section.

4.2.1 debug

Whether run agent on debug mode. Debug mode will set logging level to DEBUG. Default: `false`

4.2.2 server

The IP address or hostname of the server which this agent connect to. Default: `localhost`

4.2.3 server_port

The port of server. Default: `6800`

4.2.4 slots

How many concurrent jobs the agent would run. Default: `1`

4.2.5 request_timeout

Request timeout in seconds when communicating to server. Default: `60`

4.3 Example

Server configuration:

```
[server]
bind_address = 0.0.0.0
bind_port = 6800
debug = false
```

Agent configuration:

```
[agent]
server = localhost
server_port = 6800
debug = false
slots = 1
```

CHAPTER 5

Tags

For some reason (e.g. specify an only ip for one spider in a large cluster), we need an agent and spider matching mechanism. The tags system is here for that.

Spider may has up to one tag. Agent may has many tags. A spider must be run on the agent which has its tag.

Matching rules see:

Spider Tag, Agent Tags	None	a	b	a, b
None	True	False	False	False
a	False	True	False	True
b	False	False	True	True

Tips: An agent with some tag will never match any none tag spider.

CHAPTER 6

Runner

Runner is the unit which controls executing *scrapy* command on specify scrapy project package in system, i.e. extract spider list or executing crawling job.

There two types of built-in runners in the system. a *VenvRunner* and a *DockerRunner*

6.1 VenvRunner

To isolate environment from each spider project execution. System will create a temporary environment for each command running, with isolated executables, libraries, and files.

VenvRunner will create a virtualenv environment, run command in a sub process and clean it up when command finished.

It is the default runner in system, which do not need any additional operation to let it to be enabled.

6.2 DockerRunner

Docker runner provides a more start-fast and more secure mechanism to run a job, it start the sub-process in a container, in which the host will not be threatened by any third-party spider project.

And it need additional requisition to be enabled.

- A docker daemon.
- Set *runner_type* to *docker* in config file.
- **Optional: set *runner_docker_image* to whatever image name you what** that the runner can call pancli command on. It is *pansihub/pancli* which is built by the author by default.
- **Pull the image before you use. The system will not pull any docker image** at the runtime. *docker pull pansihub/pancli*

If you want run server/agent in docker, it is possible use the DockerRunner. To let a server/agent process access the docker daemon outside the container, You can map the host's docker sock file into container.

In docker-compose it can be setted:

volumes:

- "/var/run/docker.sock:/var/run/docker.sock"

To run in docker command it can be

docker run -v "/var/run/docker.sock:/var/run/docker.sock" ...