Gerapy

Release 0.9.2

Feb 12, 2020

Contents

1 Introduction **1**
- 1.1 Scrapy . . . 1
- 1.2 Scrapyd . . . 1
- 1.3 Scrapyd-Client . . . 2
- 1.4 Scrapyd-API . . . 2
- 1.5 Gerapy . . . 3

2 Installation **5**

3 Usage **7**
- 3.1 Initialization . . . 7
- 3.2 Database Configuration . . . 7
- 3.3 New User . . . 8
- 3.4 Startup service . . . 8
- 3.5 Host Management . . . 12
- 3.6 Project management . . . 14
- 3.7 Timing tasks . . . 20

4 Docker **23**
- 4.1 Run . . . 23
- 4.2 Docker Image . . . 23

5 Contribution **25**
- 5.1 Preparation . . . 25
- 5.2 Cloning Project . . . 25
- 5.3 Dependencies . . . 25
- 5.4 Run . . . 26
  - 5.4.1 Backend . . . 26
  - 5.4.2 Frontend . . . 27
- 5.5 Description . . . 28
- 5.6 Code Release . . . 28

6 Matintainers **29**

7 License **31**

CHAPTER 1

Introduction

Someone who has worked as a crawler with Python may use Scrapy. Scrapy is indeed a very powerful crawler framework. It has high crawling efficiency and good scalability. It is basically a necessary tool for developing crawlers using Python.

1.1 Scrapy

If you use Scrapy as a crawler, then of course we can use our own host to crawl when crawling, but when the crawl is very large, we can't run the crawler on our own machine, a good one. The method is to deploy Scrapy to a remote server for execution.

1.2 Scrapyd

At this time, you might use Scrapyd. With it, we only need to install Scrapyd on the remote server and start the service. We can deploy the Scrapy project we wrote. Go to the remote host. In addition, Scrapyd provides a variety of operations API, which gives you free control over the operation of the Scrapy project. For example, we installed Scrapyd on IP 88.88. On the .88.88 server, then deploy the Scrapy project. At this time, we can control the operation of the Scrapy project by requesting the API. The command is as follows:

```
curl http://88.88.88.88:6800/schedule.json -d project=myproject -d spider=myspider
```

This is equivalent to launching the myspider crawler for the myproject project, instead of using the command line to launch the crawler, and Scrapyd also provides a set of APIs for viewing crawler status, canceling crawler tasks, adding crawler versions, removing crawler versions, and more. So, with Scrapyd, we can control the crawler's operation through the API and get rid of the command line dependencies.

1.3 Scrapyd-Client

The crawler deployment is still a hassle because we need to upload the crawler code to the remote server. This process involves two processes of packaging and uploading. In Scrapyd, the API for this deployment is called, which is called addversion, but the content it receives is Egg package file, so to use this interface, we have to package our Scrapy project into an egg file, and then use the file upload method to request the addversion interface to complete the upload, this process is more cumbersome, so it has appeared A tool called Scrapyd-Client, with its scrapyd-deploy command, we can complete two functions of packaging and uploading, which is a convenient step.

1.4 Scrapyd-API

So we have solved the deployment problem. In the end, what if we want to see the running status of Scrapy on the server in real time? As I said earlier, of course, I am requesting Scrapyd's API. If we want to use Python programs to control it? We also use the requests library to request these APIs again and again? This is too much trouble, so in order to solve this need, Scrapyd-API has appeared again, with it we can use only simple Python code It is possible to monitor and run the Scrapy project:

```
From scrapyd_api import ScrapydAPI
Scrapyd = ScrapydAPI('http://88.888.88.88:6800')
Scrapyd.list_jobs('project_name')
```

The result of this return is the operation of each Scrapy project. E.g:

```
{
    'pending': [
    ],
    'running': [
        {
            'id': u'14a65...b27ce',
            'spider': u'spider_name',
            'start_time': u'2018-01-17 22:45:31.975358'
        },
    ],
    'finished': [
        {
            'id': '34c23...b21ba',
            'spider': 'spider_name',
            'start_time': '2018-01-11 22:45:31.975358',
            'end_time': '2018-01-17 14:01:18.209680'
        }
    ]
}
```

This way we can see the running status of the Scrapy crawler.

So, with them, what we can accomplish is:

- Complete the deployment of Scrapy projects with Scrapyd
- Control the startup and status monitoring of Scrapy projects through the API provided by Scrapyd
- Simplify deployment of Scrapy projects with Scrapyd-Client
- Control Scrapy project via Python via Scrapyd-API

Is it more convenient?

1.5 Gerapy

but? Is it really convenient to achieve it? Certainly not! If all of this, from Scrapy's deployment, startup to monitoring, log viewing, we only need a few clicks of the mouse and keyboard to complete, isn't it beautiful? In addition, we can visually configure various timing tasks and monitoring functions to conveniently schedule Scrapy crawler projects. Or, even Scrapy code can automatically generate it for you, isn't that cool?

There is motivation for demand, yes, Gerapy was born.

Gerapy is a distributed crawler management framework that supports Python 3, based on Scrapy, Scrapyd, Scrapyd-Client, Scrapy-Redis, Scrapyd-API, Scrapy-Splash, Django, Vue.js. Gerapy can help us:

- More convenient control of crawler runs
- View reptile status more intuitively
- View crawl results in more real time
- Easier timing tasks
- Easier project deployment
- More unified host management
- Write crawler code more easily

With it, the management of the Scrapy distributed crawler project is no longer difficult.

CHAPTER 2

Installation

Gerapy supports Python 3.x and does not support Python 2.

Installation command:

```
pip3 install -U gerapy
```

The gerapy command can be called directly after the installation is complete:

```
gerapy
```

If it prints like this, the installation is successful:

```
Usage: gerapy [-v] [-h] ...

Gerapy 0.9.1 - Distributed Crawler Management Framework

Optional arguments:
  -v, --version Get version of Gerapy
  -h, --help Show this help message and exit

Available commands:
    Init Init workspace, default to gerapy
    Initadmin Create default super user admin
    Runserver Start Gerapy server
    Migrate Migrate database
    Createsuperuser Create a custom superuser
    Makemigrations Generate migrations for database
    Generate Generate Scrapy code for configurable project
    Parse parse project for debugging
    Loaddata Load data from configs
    Dumpdata Dump data to configs
```

If an error occurs, please go to Gerapy Issues to search for a solution or issue a question, thank you for your support.

CHAPTER 3

Usage

This article focuses on the basic use of Gerapy and hopes to provide some help for you who are using Gerapy.

3.1 Initialization

First you can create a new project with the gerapy command. The command is as follows:

```
gerapy init
```

This will generate a gerapy folder in the current directory. This gerapy folder is the working directory of Gerapy. When you enter the gerapy folder, you will find two folders:

- projects , which are used to store Scrapy crawler projects.
- logs, used to store the Gerapy run log.

If you want to change the name of the working directory, you can add the name of the working directory after the command. If you want to create a working directory named GerapySpace, you can create it with the following command:

```
gerapy init GerapySpace
```

Its internal structure is the same.

3.2 Database Configuration

Gerapy uses the database to store various project configurations, timing tasks, etc., so the second step is to initialize the database.

First enter the working directory, for example, the working directory name is gerapy, execute the following command:

```
cd gerapy
```

At this point, first initialize the database, execute the following command:

```
gerapy migrate
```

This will generate a SQLite database, which will be used to save each host configuration information, deployment version, timing tasks, and so on.

At this time, you can find another folder in the working directory:

- dbs, which is used to store the database required by the Gerapy runtime.

3.3 New User

Gerapy has login authentication turned on by default, so you need to set up an admin user before starting the service.

For convenience, you can quickly create an admin administrator by directly using the command of the initial administrator. The password is also admin. The command is as follows:

```
gerapy initadmin
```

If you do not want to create the admin user directly, you can also manually create an administrator user with the following command:

```
gerapy createsuperuser
```

At this point Gerapy will prompt us to enter the username, email, password, etc., and then log in to Gerapy with this user.

3.4 Startup service

Next start the Gerapy service, the command is as follows:

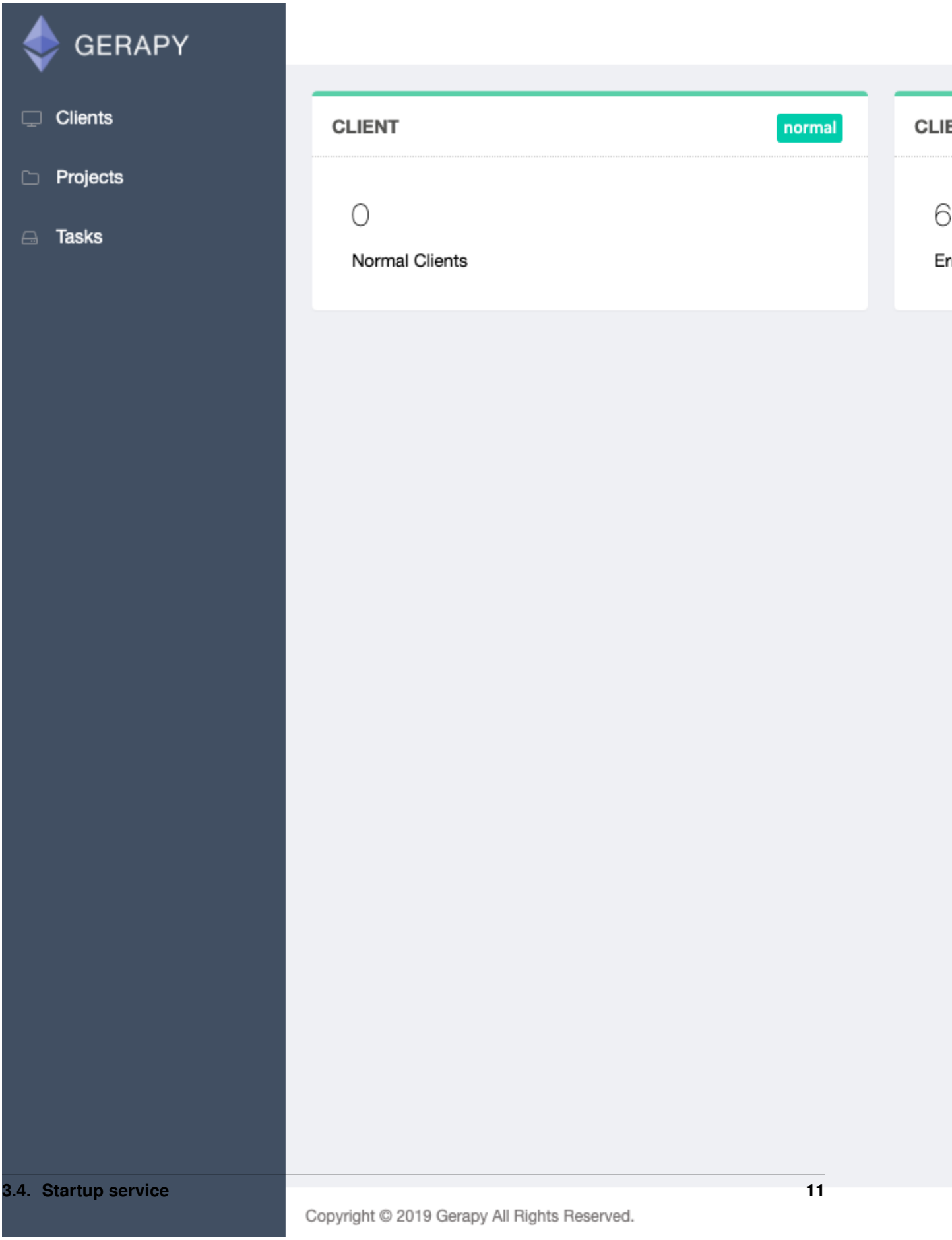
```
gerapy runserver
```

This will open the Gerapy service on the default 8000 port.

At this time, open http://localhost:8000 in the browser to enter Gerapy.

First Gerapy will prompt the user to log in, the page is as follows:

Enter Gerapy's home page by entering the username and password you created in the previous step:


GERAPY
Clients
Projects
Tasks
CLIENT
normal
0
Normal Clients
Copyright © 2019 Gerapy All Rights Reserved.

If you want Gerapy run in public, you can specify host and port like this:

```
gerapy runserver 0.0.0.0:8000
```

Then Gerapy can be accessed through 8000 port from public.

If you want Gerapy run in daemon, you can just run like this:

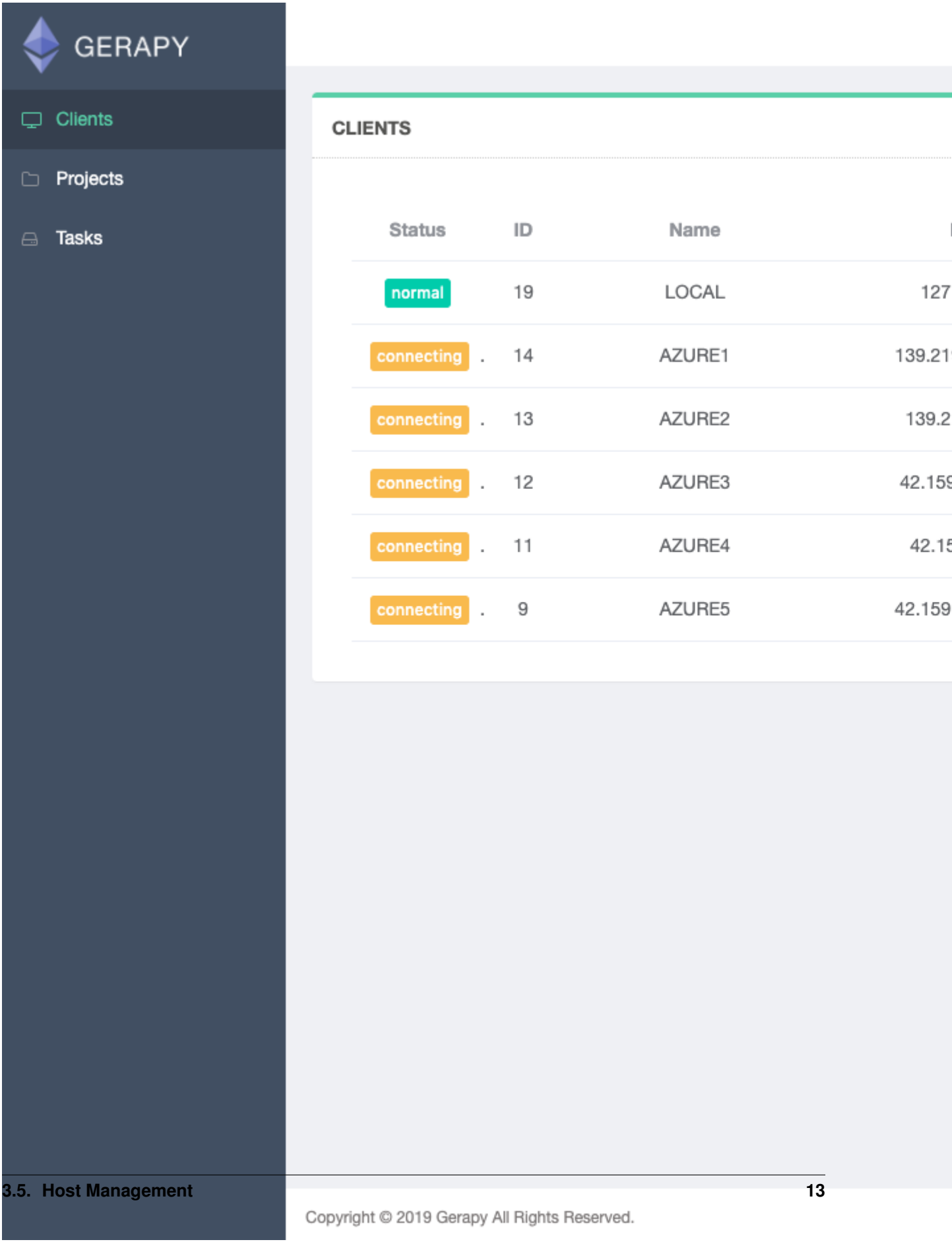
```
gerapy runserver 0.0.0.0:8000 > /dev/null 2>&1 &
```

Then Gerapy will run in daemon and in public.

3.5 Host Management

Here the host is talking about Scrapyd's service host. Scrapyd will run on port 6800 by default. It provides a series of HTTP services such as deployment and operation. For Scrapyd, you can refer to Scrapyd Document to ensure that their services can be accessed externally.

In host management, we can add the Scrapyd running address and port of each host and name it. After adding it, it will appear in the host list. Gerapy will monitor the running status of each host and identify it in different states:


GERAPY
Clients
Projects
Tasks
CLIENTS
Status
ID
Name
normal
19
LOCAL
127
connecting
14
AZURE1
139.21
connecting
13
AZURE2
139.2
connecting
12
AZURE3
42.159
connecting
11
AZURE4
42.15
connecting
9
AZURE5
42.159
Copyright © 2019 Gerapy All Rights Reserved.

Once added, we can easily view and control the crawler tasks that each host is running.

3.6 Project management

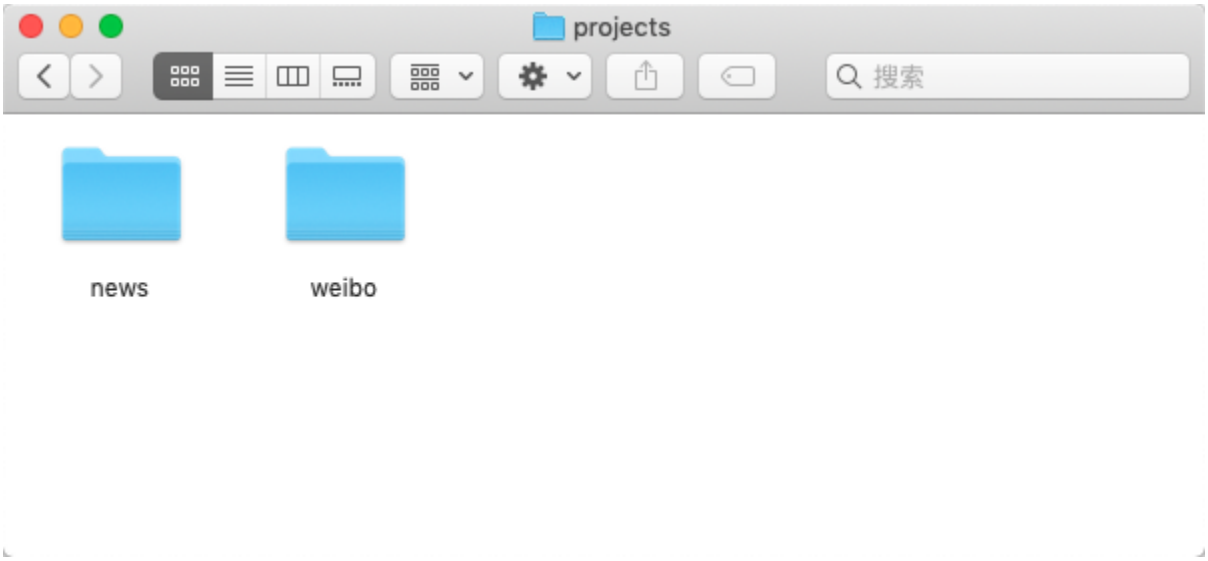
Also mentioned above is an empty projects folder in Gerapy's working directory, which is the folder where the Scrapy directory is stored.

If we want to deploy a Scrapy project, just put the project file in the projects folder.

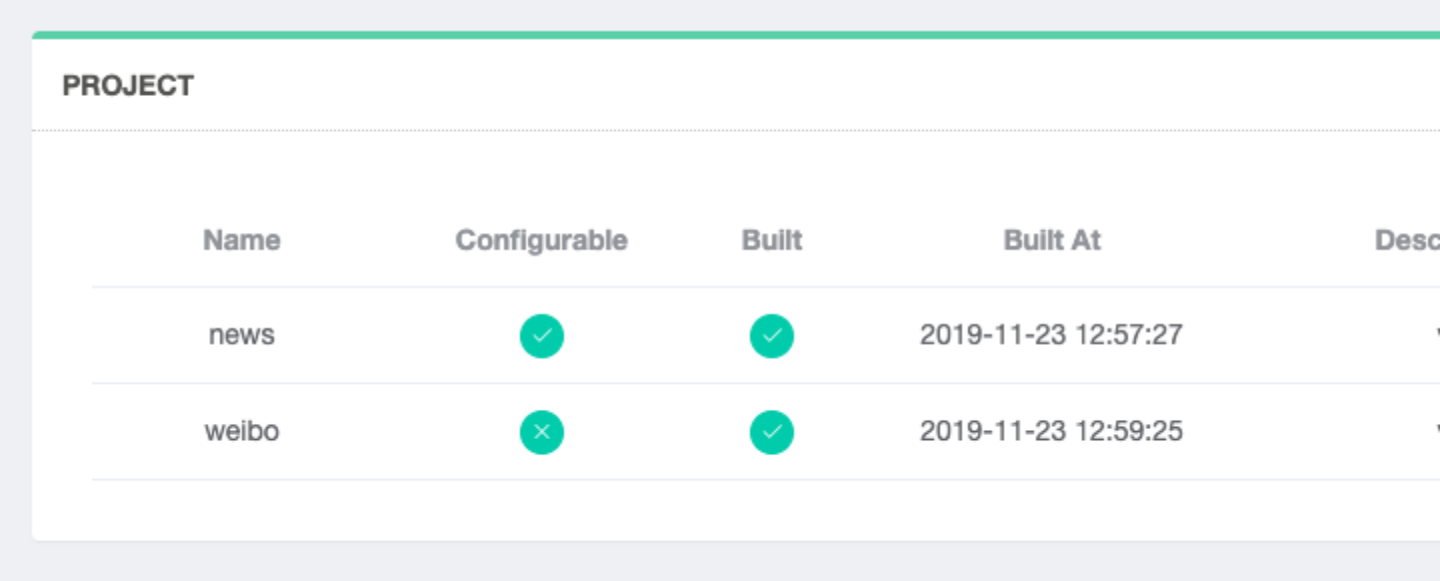
For example, you can put your project into the projects folder as follows:

- Move or copy the local Scrapy project directly to the projects folder.
- Clone or download a remote project, such as Git Clone, and download the project to the projects folder.
- Link the project to the projects folder via a soft connection (using the ln command under Linux, Mac, using the mklink command).

For example, put two Scrapy projects in the projects here:



Then go back to the Gerapy management interface and click on Project Management to see the current project list:



Since the project has a package and deployment record here, it is shown separately here.

In addition, Gerapy provides the project online editing function, we can edit the project visually by clicking Edit:

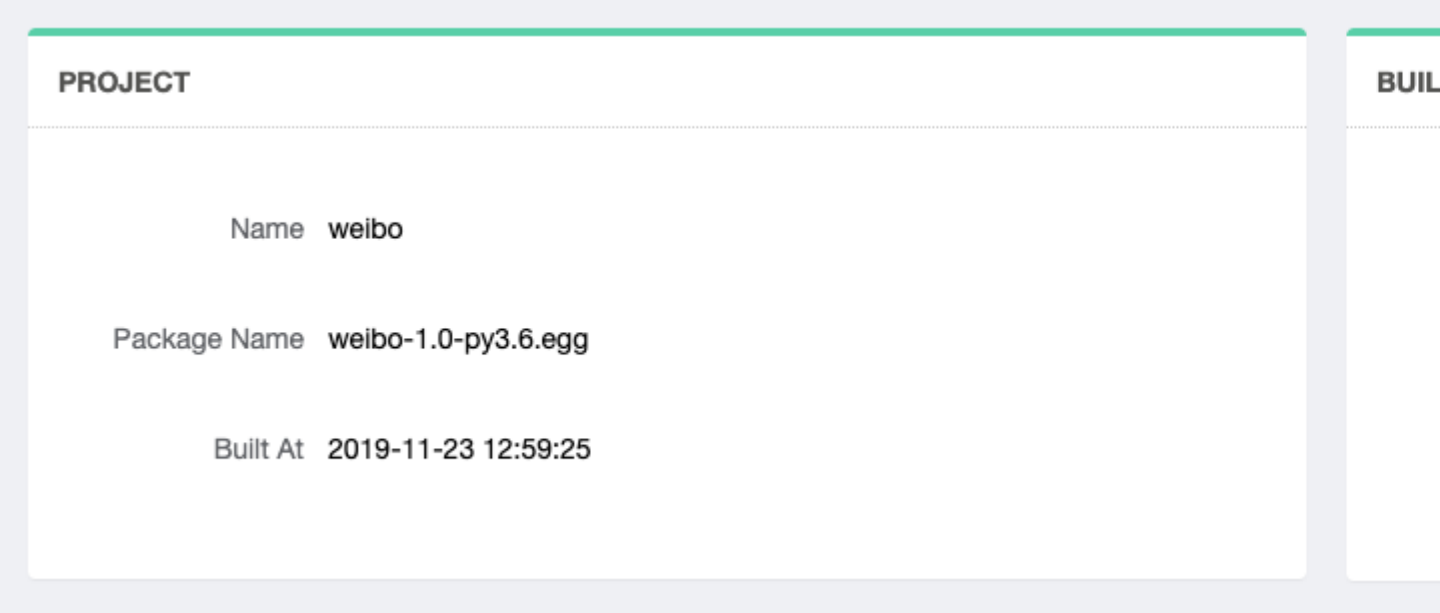
WEIBO

- dbs
- twistd.pid
- README.md
- setup.py
- scrapy.cfg
- weibo
  - spiders
    - __init__.py
    - middlewares.py
    - settings.py
    - items.py
    - pipelines.py

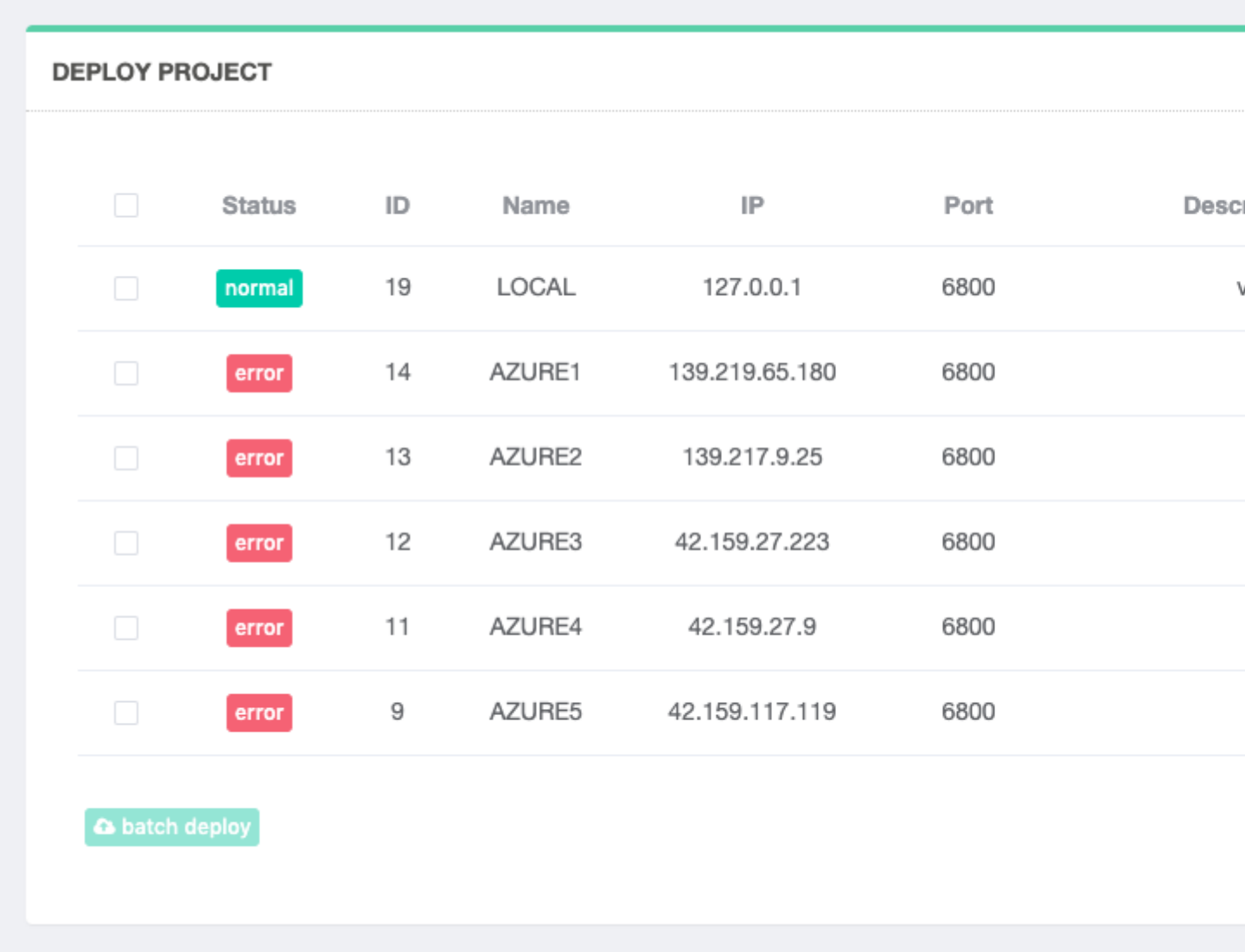
SETTINGS.PY

```
1  # -*- coding: utf-8 -*-
2
3  # Scrapy settings for weibo project
4  #
5  # For simplicity, this file contains only settings consi
6  # commonly used. You can find more settings consu
7  #
8  #     http://doc.scrapy.org/en/latest/topics/settings
9  #     http://scrapy.readthedocs.org/en/latest/topics
10 #     http://scrapy.readthedocs.org/en/latest/topics
11
12 BOT_NAME = 'weibo'
13
14 SPIDER_MODULES = ['weibo.spiders']
15 NEWSPIDER_MODULE = 'weibo.spiders'
16
17 # Crawl responsibly by identifying yourself (and your
18 # USER_AGENT = 'weibo (+http://www.yourdomain.
19
20 # Obey robots.txt rules
21 ROBOTSTXT_OBEY = False
22
23 DEFAULT_REQUEST_HEADERS = {
24     'Accept': 'application/json, text/plain, */*',
25     'Accept-Encoding': 'gzip, deflate, sdch',
26     'Accept-Language': 'zh-CN,zh;q=0.8,en;q=0.6,ja;q
27     'Connection': 'keep-alive',
28     'Host': 'm.weibo.cn',
29     'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac O
30     'X-Requested-With': 'XMLHttpRequest',
31 }
32
```

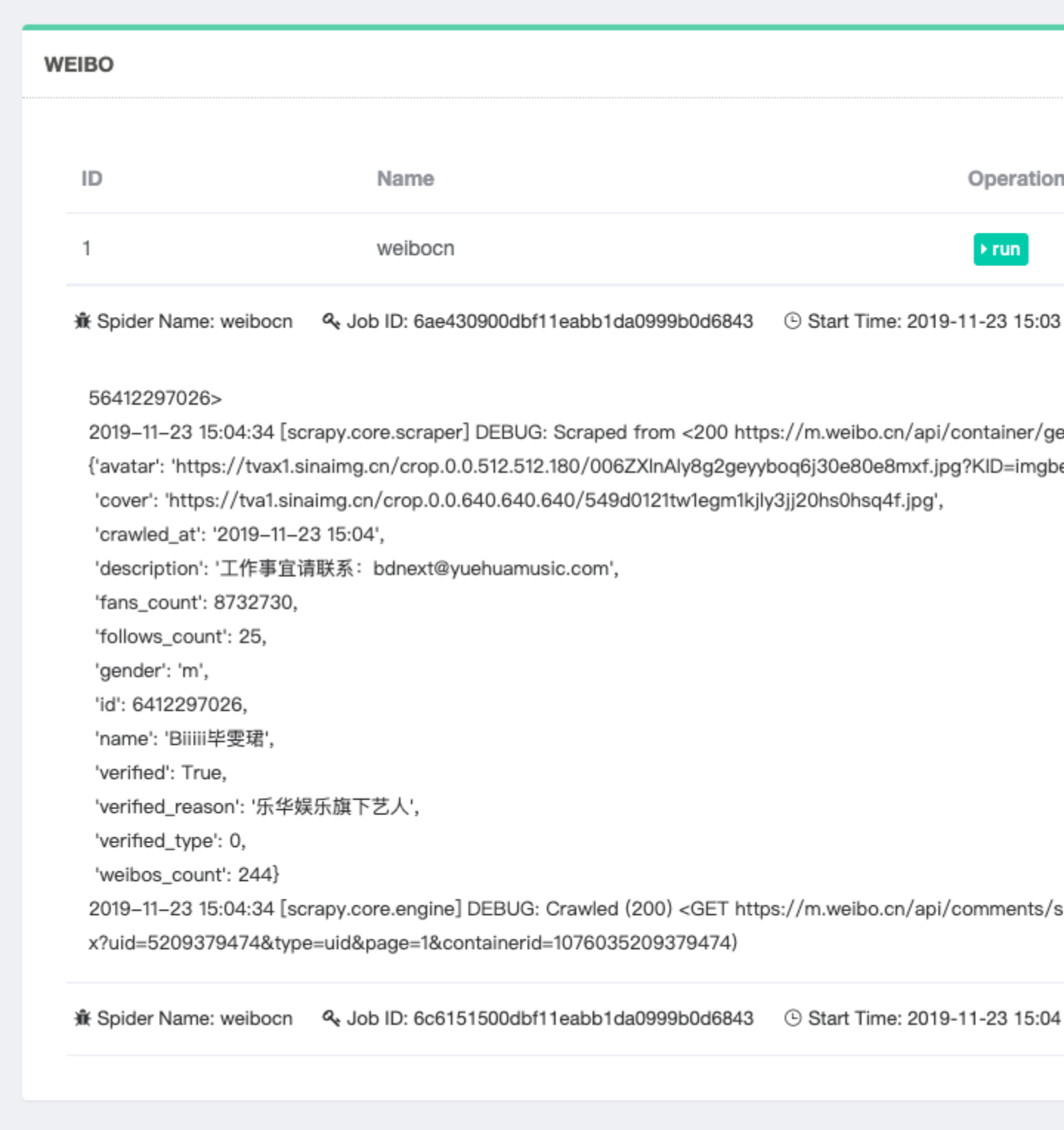
If the project has no problems, you can click on the deployment to package and deploy. You need to package the project before deployment. You can specify the version description when packaging:



After the package is complete, you can click the deployment button to deploy the packaged Scrapy project to the corresponding cloud host, and you can also deploy it in batches.



After the deployment is complete, you can go back to the host management page to schedule the task. Click Schedule to view the task management page. You can view the running status of all tasks on the current host:

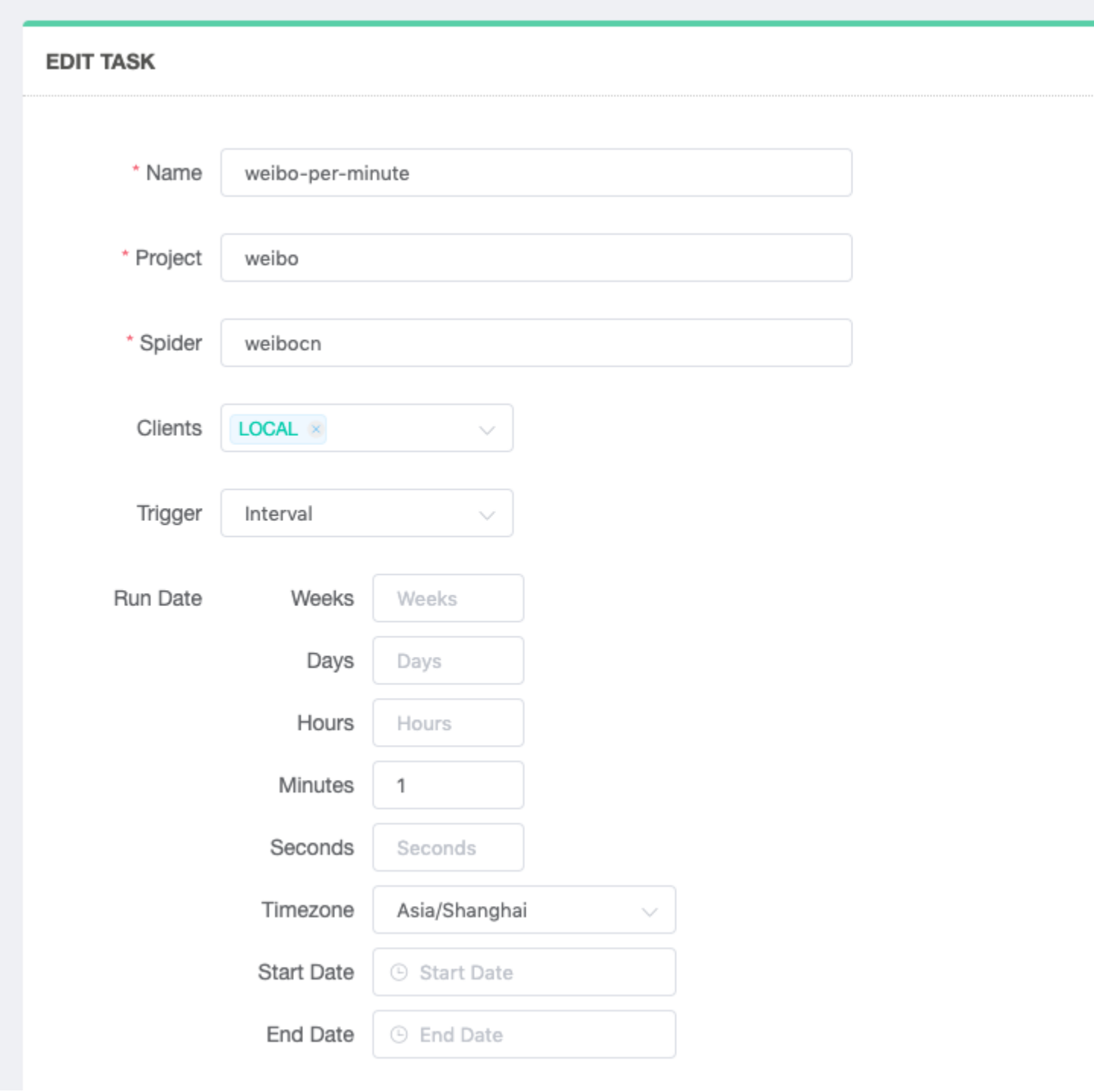


We can start and stop the task by clicking the button such as run, stop, etc., and also can view the log details by expanding the task entry.

This way we can see the status of each task in real time.

3.7 Timing tasks

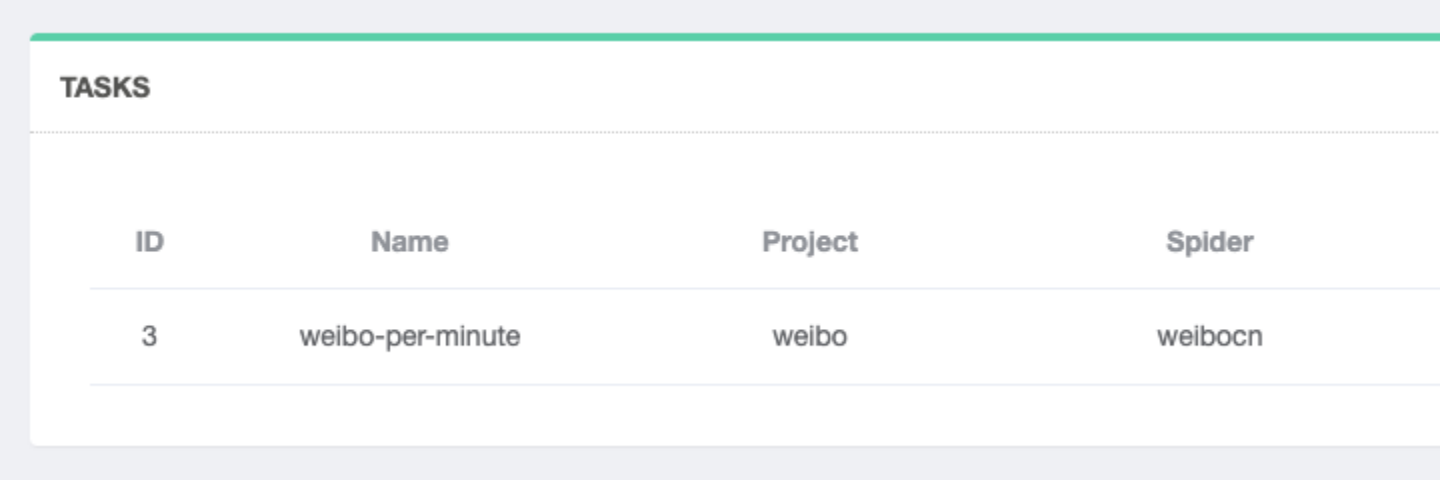
In addition, Gerapy supports setting up scheduled tasks, entering the Task Management page, and creating a new scheduled task, such as creating a new crontab mode, which runs every minute:



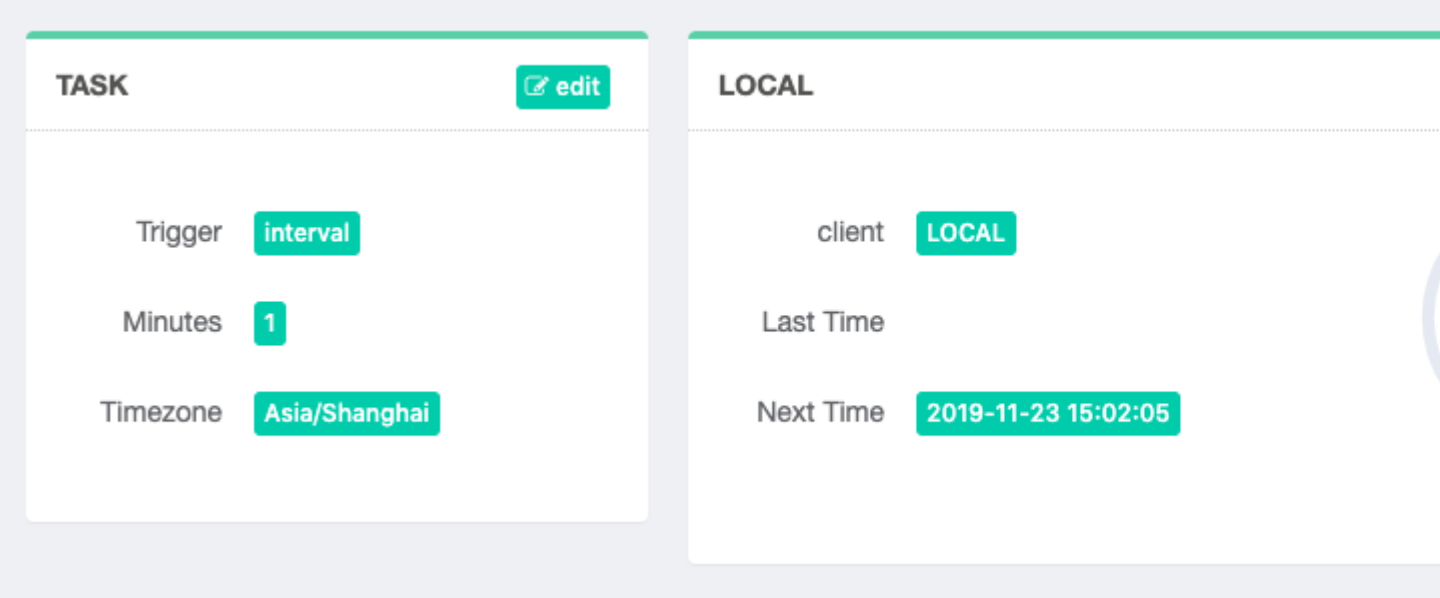
Here, if you set the run every minute, you can set the "minute" to 1, and you can set the start date and end date.

After the creation is complete, return to the Task Management home page to see the list of scheduled tasks that have

been created:



Click “Status” to view the running status of the current task:



You can also manually control scheduled tasks and view the run log by clicking the Schedule button in the upper right corner.

The above is the basic usage of Gerapy.

If you have found an error, or if you have any suggestions for Gerapy, please feel free to post it to Gerapy Issues and thank you for your support. Your feedback and suggestions are invaluable and hope that your participation will help Gerapy do better.

CHAPTER 4

Docker

Gerapy also provides a Docker Image that can be used to quickly launch the Gerapy service.

4.1 Run

First you need to select a directory as the Gerapy working directory, such as ~/gerapy or others. For example, we use the ~/gerapy directory. The Gerapy startup command is as follows:

```
docker run -d --name gerapy -v ~/gerapy:/app/gerapy -p 8000:8000 germey/gerapy
```

After running, go directly to http://localhost:8000 to enter Gerapy.

The default login user name is admin and the password is admin. The username and password can be modified by http://localhost:8000/admin/auth.

The command parameters are as follows:

- -d: running in the background
- –name: specify the container name
- -v: specify the mount directory
- -p: specifies the binding port, the former is the host port and the latter is the container port.

4.2 Docker Image

This image is in Docker Hub at Gerapy, and its version corresponds to Gerapy. For a list of versions, see: Tags.

CHAPTER 5

Contribution

If you are interested in secondary development of Gerapy or contributing to Gerapy, this document details the environment configuration and development process for Gerapy developers.

5.1 Preparation

Development Gerapy requires a Python3 environment and a Node.js environment. The recommended versions are Python 3.6 + and Node.js 10.x +. Please install and make sure to use the python3, pip3, node, npm commands.

5.2 Cloning Project

First clone Gerapy locally:

```
git clone https://github.com/Gerapy/Gerapy.git
```

Once the clone is complete, a Gerapy folder is generated locally.

5.3 Dependencies

Go to the Gerapy folder and install the dependencies you need for development:

```
pip3 install -r requirements.txt
```

Then install Gerapy locally:

```
python3 setup.py install
```

This installs the development version of Gerapy, and if Gerapy was previously installed, this installation will replace the previous version.

The gerapy command can be used after the installation is complete.

Then install the front-end dependencies, the front-end code is in the gerapy/client folder, based on Vue.js development.

Go into the directory and install dependencies.

```
cd gerapy/client
npm install
```

This will generate a node_modules folder under gerapy/client.

5.4 Run

Here are divided into frontend and backend respectively introduced, the two parts need to start at the same time to work properly.

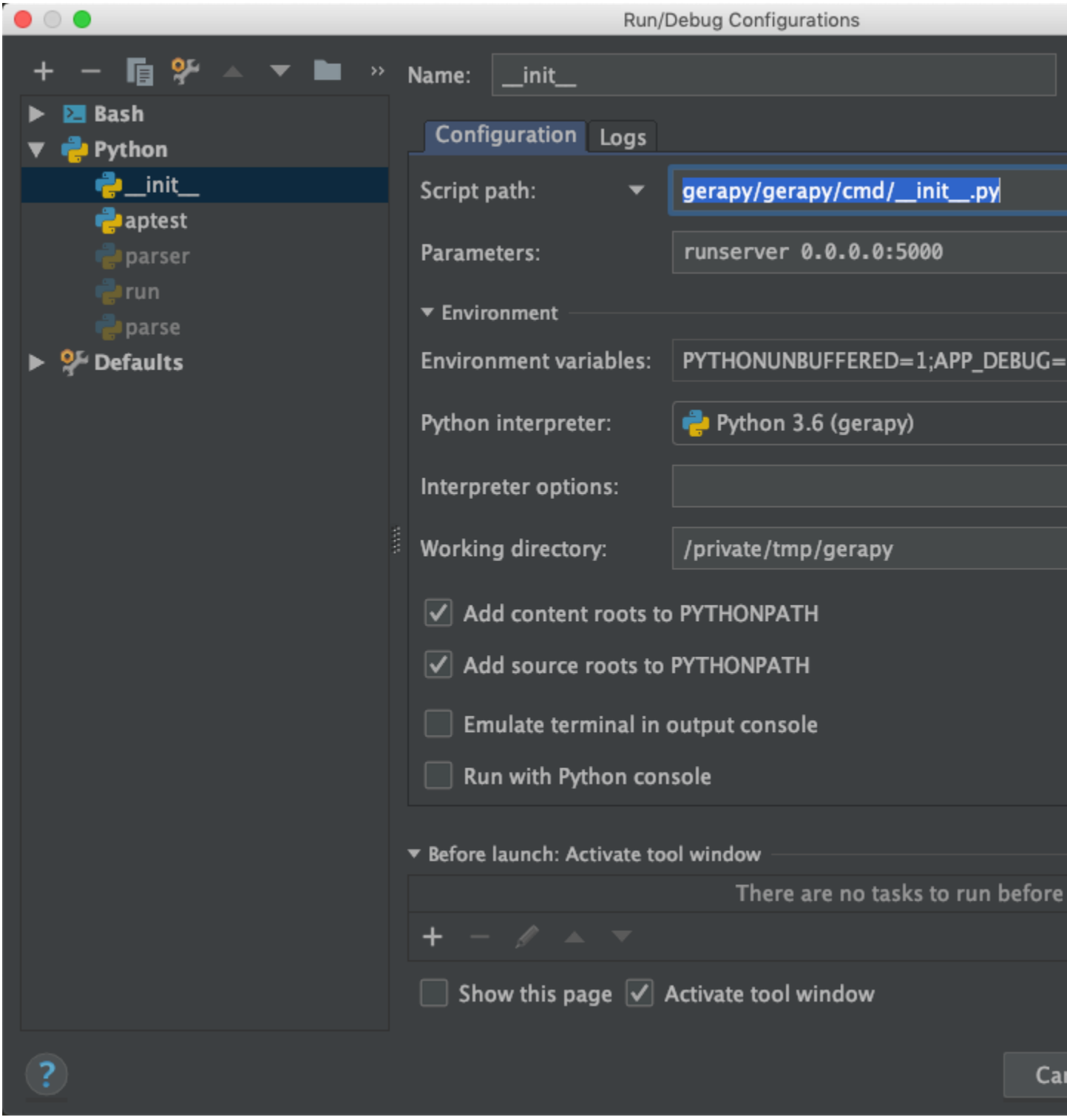
5.4.1 Backend

For the backend, the gerapy command requires the Gerapy installation package to be available. However, it is tedious to manually install each time after the change during development.

It is recommended to set the parameter configuration in PyCharm IDE, and it is also convenient for debugging.

- Script path: `gerapy/gerapy/cmd/__init__.py`, which is the entry file for the command.
- Running parameters: `runserver 0.0.0.0:5000`. Note that it needs to run on port 5000, and the frontend will forward the request to port 5000.
- Environment variables: `PYTHONUNBUFFERED=1;APP_DEBUG=true`, where APP_DEBUG is to set the debug mode, and more debug logs will be printed.
- Working path: The working path generated by the gerapy init command.

as the picture shows:



After starting in this way, Gerapy Server will run on port 5000, and the console will print out debugging information.

5.4.2 Frontend

For the frontend, in the gerapy/client folder, execute:

```
npm run serve
```

It can run on port 8080, and its backend API will be forwarded to port 5000, which is the Gerapy Server just started.

Open http://localhost:8080 to enter Gerapy's frontend page.

5.5 Description

The backend is based on Django development, and the database used is SQLite, whose main logic is in the gerapy/server folder.

The front end is based on Vue.js development and its main logic is in the gerapy/client folder.

5.6 Code Release

After the frontend modification is completed, if you want to officially release it, you can directly execute it:

```
npm run build
```

The result of the build will go to the backend's gerapy/server/core/templates folder.

The release of the backend code can be executed directly:

```
python3 setup.py upload
```

It will be automatically uploaded to PyPi and tagged on GitHub, but you must have PyPi and GitHub permissions.

CHAPTER 6

Matintainers

- Germey
  - Bloghttps://cuiqingcai.com/
  - GitHubhttps://github.com/Germey
- Thsheep
  - Bloghttps://www.thsheep.com/
  - GitHubhttps://github.com/thsheep

CHAPTER 7

License

This project is based on MIT license.