loon Documentation

Release 0.4.2

ShixiangWang

May 19, 2020

Contents

This is the documentation of **loon**.

CHAPTER 1

Contents

1.1 loon - A Python toolkit for operating remote host

GitHub repo size PyPI

1.1.1 Description

loon is a Python toolkit for operating remote host based on SSH. Idea for developing **loon** comes from **sync-deploy**, which is limited by its pure bash code. Therefore, I use Python to implement it and more features will be added to it in the future.

1.1.2 Installation

Install from pypi:

```
pip install loon
```

Install from GitHub:

```
pip install git+https://github.com/ShixiangWang/loon
```

1.1.3 Usage

Configuration

To access remote host without typing password, you have to generate ssh key with `ssh-keygen` command if it is not available.

```
$ ssh-keygen
```

Follow the commands, for simplicity, just type `ENTER` to proceed.

Copy your key to remote server, replace `user` with your `username` and `host` with your remote host ip address.

```
$ ssh-copy-id -i ~/.ssh/id_rsa user@host
```

Host management

- Add a remote host

```
$ loon add -U wsx -H 127.0.0.1 -P 22
=> Added successfully!
# Default port is 22, so don't need to specify it
# And we can create a host alias, otherwise
# it is same as username of remote host
$ loon add -U wsx -H 127.0.0.2 -N host2
=> Added successfully!
```

- List all remote hosts

```
$ loon list
+-----+--------+----------+----+
|Alias|Username|IP address|Port|
+-----+--------+----------+----+
|<wsx>|wsx     |127.0.0.1 |22  |
+-----+--------+----------+----+
|host2|wsx     |127.0.0.2 |22  |
+-----+--------+----------+----+
<active host>
```

- Rename alias

```
$ loon rename wsx host1
$ loon list
+-------+--------+----------+----+
|Alias  |Username|IP address|Port|
+-------+--------+----------+----+
|<host1>|wsx     |127.0.0.1 |22  |
+-------+--------+----------+----+
|host2  |wsx     |127.0.0.2 |22  |
+-------+--------+----------+----+
<active host>
```

- Switch active remote host

```
$ loon switch -N host2
=> Activated.
$ loon list
+-------+--------+----------+----+
|Alias  |Username|IP address|Port|
+-------+--------+----------+----+
|host1  |wsx     |127.0.0.1 |22  |
+-------+--------+----------+----+
|<host2>|wsx     |127.0.0.2 |22  |
+-------+--------+----------+----+
<active host>
```

- Delete a host

```
$ loon delete -N host2
=> Removing host from available list...
=> Removing active host...
=> Changing active host to host1
$ loon list
+-------+--------+----------+----+
|Alias  |Username|IP address|Port|
+-------+--------+----------+----+
|<host1>|wsx     |127.0.0.1 |22  |
+-------+--------+----------+----+
<active host>
```

Common tasks

- Run commands

```
$ loon run 'ls -l ~'
total 168
drwxr-xr-x     2 wsx liulab     25 Apr  7 23:26 bin
drwxr-xr-x.    2 wsx liulab      6 Apr  4 10:36 Desktop
drwxr-xr-x.    2 wsx liulab      6 Apr  4 10:36 Documents
drwxr-xr-x.    3 wsx liulab     69 Jun 10 16:57 Downloads
drwxr-xr-x     2 wsx liulab      6 Sep 30 10:23 facet
drwxr-xr-x    11 wsx liulab   4096 Sep 22 20:13 metawho
drwxr-xr-x.    2 wsx liulab      6 Apr  4 10:36 Music
drwxr-xr-x     3 wsx liulab     60 Apr 30 17:50 Notebooks
drwxr-xr-x.    2 wsx liulab      6 Apr  4 10:36 Pictures
drwxr-xr-x     6 wsx liulab    114 Sep 27 17:33 projects
drwxr-xr-x     6 wsx liulab     96 Jun 27 16:50 projects_bk
drwxr-xr-x.    2 wsx liulab      6 Apr  4 10:36 Public
drwxr-xr-x.    2 wsx liulab      6 Apr  4 10:36 Templates
drwxr-xr-x     5 wsx liulab   4096 Oct  3 12:24 test
drwxr-xr-x  3480 wsx liulab 114688 Oct  3 13:44 tmp
drwxr-xr-x     3 wsx liulab     32 Aug 22 17:13 tools
drwxr-xr-x.    2 wsx liulab      6 Apr  4 10:36 Videos
```

- Run local scripts

This will upload scripts to remote host firstly, then run them.

```
$ loon run -f ../../tests/scripts/t*.py
=> Starting upload...

t1.py                                         100%   50    49.0KB/s   00:00
t2.py                                         100%   50    77.6KB/s   00:00

=> Finished uploading in 1s
=> Getting results:
This is t1 script.
This is t2 script.
```

- If input contains both files and directories, all files in directory will not be executed. This is a way to include child scripts which does not need to be executed.
- If input is only a directory, all scripts (not including scripts in subdirectories) under it will be executed. This is the way to maintain an independent project.

You can include data directory using `--data` flag, specify program like `bash` or `python` using `--prog` flag and set remote directory using `--dir` flag.

- Upload and download files

Use them like `cp` command. At default, use `scp` command to do the job, set `--rsync` to use `rsync` command (`--rsync` is disabled in Windows). Note there are some differences between scp and rsync, especially processing directory.

```
$ loon upload -h
usage: loon upload [-h] [-v] [--rsync] source [source ...] destination

positional arguments:
  source         Source files to upload
  destination    Remote destination directory

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  set loglevel to INFO
  --rsync        Use rsync instead of scp

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  set loglevel to INFO

$ loon download -h
usage: loon download [-h] [-v] [--rsync] source [source ...] destination

positional arguments:
  source         Source files to download
  destination    Local destination directory, note '~' should be quoted in
                 some cases

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  set loglevel to INFO
  --rsync        Use rsync instead of scp
```

- Batch process commands

By providing a structed stdin/file (CSV, TSV etc) and a sample command with placeholders `{index}` refer to column index (0 based) or column name of file, `batch` command can be used to execute a batch of commands. Users can set thread number by `-T` flag and use `--dry` flag to dry run the code.

```
$ loon batch -f src/loon/data/samplefile.csv 'echo hello {0}'
hello TCGA-2A-A8VO-01
hello TCGA-2A-A8VT-01
hello TCGA-2A-A8VV-01
hello TCGA-2A-A8VX-01

$ loon batch -f src/loon/data/samplefile.csv 'echo hello {0}' -T 4
hello TCGA-2A-A8VO-01
hello TCGA-2A-A8VT-01
hello TCGA-2A-A8VV-01
hello TCGA-2A-A8VX-01

$ loon batch -f src/loon/data/samplefile.csv 'echo hello {0}' --dry
=> Running echo hello TCGA-2A-A8VO-01
=> Running echo hello TCGA-2A-A8VT-01
```

(continues on next page)

(continued from previous page)

```
=> Running echo hello TCGA-2A-A8VV-01
=> Running echo hello TCGA-2A-A8VX-01
```

You can also read the input from pipe.

```
$ echo "yes,no" | loon batch 'echo {0} is not {1}'
yes is not no

$ cat src/loon/data/samplefile.csv | loon batch 'echo sample {0} has a longer name {1}
↪'
sample TCGA-2A-A8VO-01 has a longer name TCGA-2A-A8VO-01-01
sample TCGA-2A-A8VT-01 has a longer name TCGA-2A-A8VT-01-01
sample TCGA-2A-A8VV-01 has a longer name TCGA-2A-A8VV-01-01
sample TCGA-2A-A8VX-01 has a longer name TCGA-2A-A8VX-01-01
```

You can also handle header and refer to column names with index or name!

```
$ cat tests/header.txt | loon batch 'echo hello {0}'
hello user
hello wsx
hello zd

$ cat tests/header.txt | loon batch 'echo hello {0}' --header
hello wsx
hello zd
$ cat tests/header.txt | loon batch 'echo hello {0}, your score is {1}' --header
hello wsx, your score is 100
hello zd, your score is 100

$ cat tests/header.txt | loon 'echo hello {user}, your score is {score}' --header
hello wsx, your score is 100
hello zd, your score is 100
$ cat tests/header2.txt | loon 'echo hello {user name}, your score is {score}' --
↪header
hello wsx, your score is 100
hello zd, your score is 100
```

- Generate a batch of (script) files

`gen` command is similar to the `pbsgen` command below, but removes the `.pbs` file extension, so users have to add the file extension in the first column of the SAMPLEFILE if necessary.

The example files can be generated using `pbsgen_example`.

```
usage: loon gen [-h] [-v] [-t TEMPLATE] [-s SAMPLEFILE] [-m MAPFILE]
                [-o OUTPUT]

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         set loglevel to INFO
  -t TEMPLATE, --template TEMPLATE
                        A template file containing placeholders
  -s SAMPLEFILE, --samplefile SAMPLEFILE
                        A csv file containing unique filenames (the first
                        column) and replacing labels
  -m MAPFILE, --mapfile MAPFILE
                        A csv file containing placeholders and column index
```

(continues on next page)

(continued from previous page)

```
                        (0-based) indicating replacing labels in samplefile
  -o OUTPUT, --output OUTPUT
                        Output directory
```

PBS management and tasks

- `pbstemp` - Generate a PBS template file
- `pbsgen` - Generate a batch of PBS files
- `pbsgen_example` - Generate example files for pbsgen command
- `pbssub` - Submit PBS tasks
- `pbsdeploy` - Upload a target directory and submit containing PBS files (have `.pbs` extension)
- `pbscheck` - Check status of PBS job on remote host

More details please see `-h` option of the commands above.

Current usage info

```
usage: loon [-h] [-V] [--author]
                    {add,delete,switch,list,rename,run,upload,download,gen,batch,
↪pbstemp,pbsgen,pbsgen_example,pbssub,pbsdeploy,pbscheck}
                    ...

Be an efficient loon.

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --author              show info of program's author

subcommands:
  {add,delete,switch,list,rename,run,upload,download,gen,batch,pbstemp,pbsgen,pbsgen_
↪example,pbssub,pbsdeploy,pbscheck}
                        description
    add                 Add a remote host
    delete              Delete a remote host
    switch              Switch active remote host
    list                List all remote hosts
    rename              Rename host alias
    run                 Run commands or scripts on remote
    upload              Upload files to active remote host
    download            Download files from active remote host
    gen                 Generate a batch of (script) files
    batch               Batch process commands with placeholders
    pbstemp             Generate a PBS template file
    pbsgen              Generate a batch of PBS files (with .pbs extension)
    pbsgen_example      Generate example files for pbsgen command
    pbssub              Submit PBS tasks
    pbsdeploy           Deploy target destination to remote host
    pbscheck            Check status of PBS job on remote host
```

1.1.4 Note

This project has been set up using PyScaffold 3.2.2. For details and usage information on PyScaffold see https://pyscaffold.org/.

1.2 License

1.3 Contributors

- ShixiangWang <w_shixiang@163.com>

1.4 Changelog

1.4.1 Version 0.4.2

- Resolve issue with encoding

1.4.2 Version 0.4.1

- Add version flag *-V*
- Add documentation for functions and methods

1.4.3 Version 0.4.0

- Support column names as placeholders in the *batch* command

1.4.4 Version 0.3.4

- Fix a bug in *batch* command

1.4.5 Version 0.3.2-0.3.3

- Support pipe as input to *batch* command (#20)

1.4.6 Version 0.3.0-0.3.1

- Add *–dry* to all commands

1.4.7 Version 0.2.3

- Fix some bugs about PBS feature

1.4.8 Version 0.2.2

- Add *gen* command, fix #14

1.4.9 Version 0.2.1

- Fixed a bug related to judgement of *–rsync* option
- *–rsync* option is now disabled in Windows
- git bash is removed from documentation due to its bad performance

1.4.10 Version 0.2.0

- Added thread option to *batch* command

1.4.11 Version 0.1.19

- Fixed *header* bug for *batch* command

1.4.12 Version 0.1.18

- Added documentation for *batch* command

1.4.13 Version 0.1.17

- Added *batch* command

1.4.14 Version 0.1.16

- Reformatted code using [**yapf**](https://github.com/google/yapf)

1.4.15 Version 0.1.15

- Added –*rsync* option to *pbsdeploy* command

1.4.16 Version 0.1.14

- Fixed bug about *rsync* attribute checking

1.4.17 Version 0.1.13

- Fixed some bugs
- Added –*rsync* option to *upload* and *download* commands, fixed #10
- Added status code checking to *upload* and *download* commands

1.4.18 Version 0.1.12

- Updated *pbssub* command
- Added *pbsdeploy* command

1.4.19 Version 0.1.11

- Fixed some bugs
- Added *pbsgen* command
- Added *pbsgen_example* command
- Added some data files

1.4.20 Version 0.1.10

- Fixed Unix newline for *pbstemp* command
- Added *pbssub* command
- Tested *pbscheck* command

1.4.21 Version 0.1.9

- Fixed typo and some config

1.4.22 Version 0.1.7-0.1.8

- Updated documentation
- Updated deployment code

1.4.23 Version 0.1.6

- Implemented *run* command
- Implemented *upload* command
- Added *download* command
- Added *pbstemp* command
- Added *pbscheck* command

1.4.24 Version 0.1.5

- Useless badge removed in README
- A pretty table function added to list hosts

1.4.25 Version 0.1.4

- Python version dependency added: Python 3.4 or greater is required for loon
- Badges added to README.md

1.4.26 Version 0.1

- Feature A added
- FIX: nasty bug #1729 fixed
- Feature host management added

1.5 loon

1.5.1 loon package

Submodules

loon.classes module

Classes used in loon package

class loon.classes.**Host**(*hostfile='/home/docs/.config/loon/host.json'*)
Bases: object

Representation of remote host

add(*name*, *username*, *host*, *port=22*, *dry_run=False*)
Add a remote host

Parameters

- **name** – hostname alias, a string
- **username** – hostname, a string
- **host** – host ip address, a string

- **port** – host ip port, an integer
- **dry_run** – if *True*, dry run the code

Returns None

cmd(*commands*, *_logger=None*, *run_file=False*, *data_dir=None*, *remote_file=False*, *dir='/tmp'*, *prog=None*, *dry_run=False*)
Run command(s) in active remote host using channel session Therefore, *open_channel* in *connect* method must be *True* before using it.

Parameters

- **commands** – commands/scripts run on active remote host
- **_logger** – the logging logger
- **run_file** – if *True*, run scripts instead of commands
- **data_dir** – a path representing data directory
- **remote_file** – if *True*, collect input from remote host instead of local machine
- **dir** – Remote directory for storing local scripts
- **prog** – a string representing the program to run the commands
- **dry_run** – if *True*, dry run the code

Returns A string containing result information

connect(*privatekey_file='~/.ssh/id_rsa'*, *passphrase=''*, *open_channel=True*)
Connect active host and open a session

Parameters

- **privatekey_file** – a string representing the path to the private key file
- **passphrase** – a string representing the password
- **open_channel** – if *True*, open the SSH channel

Returns None

delete(*name*, *username*, *host*, *port=22*, *dry_run=False*)
Delete a remote host

Parameters

- **name** – hostname alias, a string
- **username** – hostname, a string
- **host** – host ip address, a string
- **port** – host ip port, an integer
- **dry_run** – if *True*, dry run the code

Returns None

download(*source*, *destination*, *_logger*, *use_rsync=False*, *dry_run=False*)
Download files to local machine from active remote host.

Currently, it is dependent on scp command.

Parameters

- **source** – list of files (directories) in remote host

- **destination** – destination directory in local machine
- **_logger** – the logging logger
- **use_rsync** – if *True*, use rsync instead of scp
- **dry_run** – if *True*, dry run the code

Returns None

get_result(*print_info=True*)
Get result from executed channel

Parameters **print_info** – if *True*, print information

Returns a string containing output from executed commands

host_check(*name*, *username*, *host*, *port=22*)
Check if a host exists

Parameters

- **name** – hostname alias, a string
- **username** – hostname, a string
- **host** – host ip address, a string
- **port** – host ip port, an integer

Returns a list representing the host

list()
List all remote hosts

load_hosts()
Load hosts from file

rename(*old*, *new*, *dry_run=False*)
Rename host name

Parameters

- **old** – a string representing the old host name alias
- **new** – a string representing the new host name alias
- **dry_run** – if *True*, dry run the code

Returns None

save_hosts()
Save hosts to file

switch(*name*, *username*, *host*, *port=22*, *dry_run=False*)
Switch active host

Parameters

- **name** – hostname alias, a string
- **username** – hostname, a string
- **host** – host ip address, a string
- **port** – host ip port, an integer
- **dry_run** – if *True*, dry run the code

Returns None

upload(*source*, *destination*, *_logger*, *use_rsync=False*, *dry_run=False*)
Upload files to active remote host.

Currently, it is dependent on scp command.

Parameters

- **source** – list of files (directories) in local machine
- **destination** – destination directory in remote host
- **_logger** – the logging logger
- **use_rsync** – if *True*, use rsync instead of scp
- **dry_run** – if *True*, dry run the code

Returns None

class loon.classes.**PBS**
Bases: object

Representation of PBS task

check(*host*, *job_id*, *dry_run=False*)
Check PBS task status

Parameters

- **host** – a host object
- **job_id** – a string the job id
- **dry_run** – if *True*, dry run the code

Returns Job status

deploy(*host*, *source*, *destination*, *_logger*, *use_rsync=False*, *dry_run=False*)
Deploy target directory on the active remote host

Upload the target destination and then submit all *.pbs files.

Parameters

- **host** – a host object
- **source** – a string representing the directory (contains .pbs files) to upload
- **destination** – a string representing the path on remote host
- **_logger** – the logging logger
- **use_rsync** – if *True*, use rsync instead of scp
- **dry_run** – if *True*, dry run the code

Returns None

gen_pbs(*template*, *samplefile*, *mapfile*, *outdir*, *_logger*, *pbs_mode=True*, *dry_run=False*)
Generate a batch of (script) files (PBS tasks) based on template and mapping file

Parameters

- **template** – a string representing the path to the template file
- **samplefile** – a string representing the path to the sample file
- **mapfile** – a string representing the path to the mapping file

- **outdir** – a string representing the path to output directory
- **_logger** – the logging logger
- **pbs_mode** – if *True*, use PBS mode
- **dry_run** – if *True*, dry run the code

Returns None

gen_pbs_example(*outdir*, *_logger*, *dry_run=False*)
Generate example files for pbsgen command to specified directory

Parameters

- **outdir** – a string representing the output directory
- **_logger** – the logging logger
- **dry_run** – if *True*, dry run the code

Returns None

gen_template(*input*, *output*, *dry_run=False*)
Generate a PBS template

Parameters

- **input** – a string representing the path to template file
- **output** – a string representing the path to output file
- **dyr_run** – if *True*, dry run the code

Returns None

sub(*host*, *tasks*, *remote*, *workdir*, *_logger*, *dry_run=False*)
Submit pbs tasks

Parameters

- **host** – a host object
- **tasks** – a list of PBS files, glob pattern is supported
- **remote** – if *True*, means that PBS task files are located at the active remote host
- **workdir** – a directory representing the working directory
- **_logger** – the logging logger
- **dry_run** – if *True*, dry run the code

Returns A list of files

loon.skeleton module

The skeleton of loon package. Set command in [options.entry_points] section of setup.cfg file Install with *python setup.py install*

loon.skeleton.**main**(*args*)
Main entry point allowing external calls

Parameters **args** ([*str*]) – command line parameter list

loon.skeleton.**parse_args**(*args*)
Parse command line parameters

Parameters **args** ([*str*]) – command line parameters as list of strings

Returns command line parameters namespace

Return type argparse.Namespace

loon.skeleton.**run**()
Entry point for console_scripts

loon.skeleton.**setup_logging**(*loglevel*)
Setup basic logging

Parameters **loglevel** (*int*) – minimum loglevel for emitting messages

loon.tool module

Tool functions

loon.tool.**batch**(*input*, *cmds*, *sep=', '*, *header=False*, *thread=1*, *dry_run=False*, *_logger=None*)
Batch process commands according to mappings from file

Parameters

- **input** – stdin or a path to input file
- **cmds** – template command (with placeholder) to run
- **sep** – separator, default is ','
- **header** – set *True* if input data contains a header line
- **thread** – number of threads to run in parallel
- **dry_run** – if *True*, dry run the code
- **_logger** – the logging logger

Returns None

loon.tool.**prun**(*x*)

loon.utils module

loon.utils.**create_parentdir**(*path*)
Create parent directory for a file/directory if not exists

Parameters **path** ([*str*]) – a file path

loon.utils.**decomment**(*csvfile*)

loon.utils.**get_filelist**(*dirName*)
Create a list of file and sub directories names in the given directory.

Source code from: https://thispointer.com/python-how-to-get-list-of-files-in-directory-and-sub-directories/

loon.utils.**pretty_table**(*title*, *content*)
Print a pretty table according to the title and content

Parameters

- **title** ([*str*]) – a list containing the title of table, e.g. ['title1', 'title2']
- **content** ([*str*]) – a nested list containing the content of table, e.g. [['a1', 'a2'], ['b1', 'b2']]

loon.utils.**read_csv**(*file_path*, *sep=', '*, *rm_comment=True*)
Read CSV file

Module contents

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index