remote$_multicommandDocumentation$

Release 0.1.3

Jonatan Dellagostin

January 04, 2017

Contents

CHAPTER 1

remote_multicommand

remote_multicommand provides execution of multiple commands in multiple servers in parallel (multiple processes)

1.1 Executing a list of commands in multiple servers in parallel

```
>>> from remote_multicommand import RemoteMultiCommand
>>> cmds_list = ['hostname','whoami']
>>> num_of_process = 4
>>> rm_cmd = RemoteMultiCommand('/tmp/sshkey')
Log: Changing log level to ERROR | Log level:ERROR | Date:01/11/2016 16:40:10
>>> rm_cmd.set_log_level('DEBUG')
Log: Changing log level to DEBUG | Log level:DEBUG | Date:01/11/2016 16:40:12
>>> servers_list = ['serverOne', 'serverTwo', 'serverThree', 'serverFour']
>>> rm_cmd.launch_list_of_commands(cmds_list, num_of_process, servers_list, ssh_log_level='DEBUG')
Log: Executing 2 commands in the list of servers: | Log level:INFO | Date:01/11/2016 16:40:27
Log: Processing in the 4 servers will be done in 1 iterations. | Log level:INFO |
  # Date:01/11/2016 16:40:27
Log: Processing 4 servers in this iteration. | Log level:DEBUG | Date:01/11/2016 16:40:27
Log: Servers: ['serverOne', 'serverTwo', 'serverThree', 'serverFour'] | Log level:DEBUG |
  # Date:01/11/2016 16:40:27
Log: It took 2.338 seconds to execute command 'hostname' in all 4 servers. | Log level:INFO
  # | Date:01/11/2016 16:40:30
Log: Processing in the 4 servers will be done in 1 iterations. | Log level:INFO |
  # Date:01/11/2016 16:40:30
Log: Processing 4 servers in this iteration. | Log level:DEBUG | Date:01/11/2016 16:40:30
Log: Servers: ['serverOne', 'serverTwo', 'serverThree', 'serverFour'] | Log level:DEBUG |
  # Date:01/11/2016 16:40:30
Log: It took 2.396 seconds to execute command 'whoami' in all 4 servers. | Log level:INFO |
  # Date:01/11/2016 16:40:32
Log: Server serverTwo:
 - All 2 commands were issued: Yes
 - Number of commands issued: 2
 - Number of commands bypassed: 0 | Log level:INFO | Date:01/11/2016 16:40:32
Log: Server serverOne:
 - All 2 commands were issued: Yes
- Number of commands issued: 2
 - Number of commands bypassed: 0 | Log level:INFO | Date:01/11/2016 16:40:32
Log: Server serverThree:
 - All 2 commands were issued: Yes
  - Number of commands issued: 2
 - Number of commands bypassed: 0 | Log level:INFO | Date:01/11/2016 16:40:32
```

```
Log: Server serverFour:
 - All 2 commands were issued: Yes
 - Number of commands issued: 2
 - Number of commands bypassed: 0 | Log level:INFO | Date:01/11/2016 16:40:32
Log: It took 4.735 seconds to execute the list of commands in all 4 servers. | Log level:INFO
| Date:01/11/2016 16:40:32

{'serverTwo': [OrderedDict([('command', 'hostname'), ('access', True),
('result', True), ('output', 'serverTwo\n')]),
OrderedDict([('command', 'whoami'), ('access', True), ('result', True), ('output', 'root\n')])],
'serverOne': [OrderedDict([('command', 'hostname'), ('access', True), ('result', True),
('output', 'serverOne\n')]),
OrderedDict([('command', 'whoami'), ('access', True), ('result', True), ('output', 'root\n')])],
'serverThree': [OrderedDict([('command', 'hostname'), ('access', True), ('result', True),
('output', 'serverThree\n')]),
OrderedDict([('command', 'whoami'), ('access', True), ('result', True), ('output', 'root\n')])],
'serverFour': [OrderedDict([('command', 'hostname'), ('access', True), ('result', True),
('output', 'serverFour\n')]),
OrderedDict([('command', 'whoami'), ('access', True), ('result', True), ('output', 'root\n')])]}
```

1.2 Installation

To install remote_multicommand, simply run:

```
$ pip install remote_multicommand
```

remote_multicommand is compatible with Python 2.6+

1.3 Documentation

https://remote_multicommand.readthedocs.io

1.4 Source Code

Feel free to fork, evaluate and contribute to this project.

Source: https://github.com/jonDel/remote_multicommand

1.5 License

GPLv3 licensed.

1.6 OBS

Due to bug https://github.com/paramiko/paramiko/issues/753, we must use paramiko versions under or equal 1.17.2

CHAPTER 2

remote_multicommand package contents:

2.1 remote_multicommand package

2.1.1 Submodules

2.1.2 remote_multicommand.remote_multicommand module

class `remote_multicommand.remote_multicommand.`**`RemoteMultiCommand`**(*ssh_key*, ***kwargs*)

Bases: `loggers.loggers.Loggers`

Execute commands in parallel in remote servers

Provides a layer of abstraction for executing multiple commands in multiple servers with multiple processes in parallel

Parameters

- **`key_ssh`** (`str`) – path of the ssh private key to connect (must be None if using user and pasword to connect)
- **`log_folder`** (`str`, **optional** , *default* =None) – folder where the log files of this class will be generated
- **`username`** (`str`, *optional* , *default* =root) – username using the connection
- **`password`** (`str`,optional, *default* =None) – password for connection if using user and password instead of key
- **`ssh_port`** (`str`, optional, *default* =22) – ssh tcp port
- **`server_has_dns`** (`bool`, optional, *default* =True) – if the server is not registered in a DNS domain and/or has not its DNS name equals to its hostname, this flag must de set to False, otherwise this condition will be checked to certify we are trully connected to the right server.

`execute_command`(*server*)

Execute a command in a remote server

Issues a command in the server and updates the dictionary self.servers_cmd_dict, which maintains the state of all commands executed in this object

Parameters **`server`** (`str`) – server where the command will be executed

Returns dictionary containing the server, the command executed, the result of the connection attempt and the result of the command issued

launch_list_of_commands(*script_cmds*, *num_of_process*, *servers_list*, *ssh_log_level='CRITICAL'*)
Launch a list of parallel commands

Launches several processes that execute a sequence of commands in a list of servers For each server, the next commands will only be executed if the preceding command was successfull.

Parameters

- **script_cmds** (`str` or `list`) – list or string containing the commands (interprets ";", new line character and comments)
- **num_of_process** – (`int`) number of separated process launched in each iteration
- **servers_list** (`list`) – servers list
- **ssh_log_level** (`str`, *default* = 'CRITICAL') – log level of the ssh connection. Could be 'DEBUG', 'INFO', 'ERROR' or 'CRITICAL'

Returns

dictionary containing the servers and the result of the command

Return type servers_cmd_dict (`dict`)

launch_multicommand(*cmd*, *num_of_process*, *servers_list*, *ssh_log_level='CRITICAL'*)
Launches several processes that execute the command in a list of servers

Parameters

- **cmd** (`str`) – command to be executed in each server of the list
- **num_of_process** (`int`) – number of separated process launched in each iteration
- **servers_list** (`list`) – servers list
- **ssh_log_level** (`str`, *default* = 'CRITICAL') – log level of the ssh connection. Could be 'DEBUG', 'INFO', 'ERROR' or 'CRITICAL'

Returns

dictionary containing the servers and the result of the command

Return type servers_cmd_dict (`dict`)

2.1.3 Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index