RAX-AutoScaler Documentation

Release 0.2.24

Simon Mirco, Simone Soldateschi, Suraj Thapa, Teddy Schmitz

December 21, 2014

Contents

RAX-AutoScaler use the rackspace APIs to allow for scaling based on aggregate metrics across a cluster. Can be used and installed on the auto-scale group members or on a dedicated management instance.

- GitHub repository: https://github.com/boxidau/rax-autoscaler
- PyPI package: https://pypi.python.org/pypi/rax-autoscaler
- Stories in Ready: https://badge.waffle.io/boxidau/rax-autoscaler.svg?label=ready&title=Ready)](http://waffle.io/boxidau/rax-autoscaler

Contents:

CHAPTER 1

Quick Start

1.1 Installation

```
pip install RAX-AutoScaler
```

1.2 Configuration

Edit config.json adding the following: * API username * API key * Region name * Autoscaling group section should contain:

- AutoScale Group UUID
- Scale Up Policy UUID
- Scale Down Policy UUID
- Check Type (agent.cpu, agent.load_average...)
- Metric Name (depends on the check type)
- Scale Up Threshold
- Scale Down Threshold
- Webhooks Url (Pre & Post commit url(s) for scale up/down)

1.3 Usage

Once configured you can invoke the autoscaler.py script.

–cluster option should be used when this script actually runs on auto-scale group members. Otherwise if it is running on a dedicated management instance you do not require this option.

–as-group option should be used when you have multiple groups listed in the config.json file.

–config-file option should be used if config.json file does not exists in current directory or in '/etc/rax-autoscaler' path.

Once tested you should configure this script to run as a cron job either on a management instance or on all cluster members

1.4 Cloud Init

You can use the cloud-config file to auto-install RAX-Autoscaler on new servers. For example to do so on Rackspace cloud using supernova

> *supernova <region> boot –user-data ./cloud-config –image <image id/name> –flavor <flavor id/name> –config-drive=true <server name>*

To use this with autoscale you would want to set the userdata of your launch configuration to the base64 encoded string of the file:

```
"launchConfiguration": {
      "args": {
          "server": {
              "config_drive" : true,
              "flavorRef": "general1-1",
              "imageRef": "CentOS 6.5 (PVHVM)",
              "key_name" : "MY_SSH_KEY",
              "user_data" : "I2Nsb3VkLWNvbmZpZwoKcGFja2FnZXM6CiAgLSBweXRob24tcGlwCgpydW5jbWQ6CiAgLSBb
              "name": "test-autoscale"
          }
      },
      "type": "launch_server"
  }
```

This has been tested on these images:

- Ubuntu 14.04 LTS (PVHVM)
- CentOS 6.5 (PVHVM)

In the example the value of user_data contains the base64 encoded version of the following script:

```
echo -n "I2Nsb3VkLWNvbmZpZwoKcGFja2FnZXM6CiAgLSBweXRob24tcGlwCgpydW5jbWQ6CiAgLSBbIHBpcCwgaW5zdGFsbCw
```

To base64 encode a script :: cat /path/to/USER_DATA | base64

Size of USER_DATA can be reduced with gzip: :: cat /path/to/USER_DATA | gzip | base64

1.5 Note

> RAX-AutoScaler depends on Rackspace Monitoring Agent to get the data from nodes in scaling group.
>
> If the agent is not installed please read: Install the Cloud Monitoring Agent: http://www.rackspace.com/knowledge_center/article/install-the-cloud-monitoring-agent

1.6 Contributing

- Fork it
- Create your feature branch (git checkout -b my-new-feature)
- Commit your changes (git commit -am 'Add some feature')
- Push to the branch (git push origin my-new-feature)
- Create new Pull Request

CHAPTER 2

What is Sphinx?

Sphinx is a tool that helps to generate python documentation. Features:

- Output formats: HTML (including Windows HTML Help), LaTeX (for printable PDF versions), ePub, Texinfo, manual pages, plain text
- Extensive cross-references: semantic markup and automatic links for functions, classes, citations, glossary terms and similar pieces of information
- Hierarchical structure: easy definition of a document tree, with automatic links to siblings, parents and children
- Automatic indices: general index as well as a language-specific module indices
- Code handling: automatic highlighting using the Pygments highlighter

2.1 Quick start

You can add inline comments about a function or class at the begining for example:

```
def exit_with_error(msg):
    """This function prints error message and exit with error.

    :param msg: error message
    :type name: str
    :returns: 1 (int) -- the return code

    """
```

Execute make command in project directory with doc arg:

```
make doc
```

After successful execution updated .html file sould be available here 'docs/_build/html/'

autoscale.**autoscale**(*group*, *config_data*, *args*)

This function executes scale up or scale down policy

Parameters

- **group** – group name
- **config_data** – json configuration data
- **args** – user provided arguments

autoscale.**exit_with_error**(*msg*)
This function prints error message and exit with error.

Parameters **msg** – error message

Returns 1 (int) – the return code

autoscale.**get_scaling_group**(*group*, *config_data*)
This function checks and gets active servers in scaling group

Parameters

- **group** – group name
- **config_data** – json configuration data

Returns scalingGroup if server state is active else null

autoscale.**is_node_master**(*scalingGroup*)
This function checks scaling group state and determines if node is a master.

Parameters **scalingGroup** – data about servers in scaling group retrieve from cloudmonitor

Returns 1 : if cluster state is unknown 2 : node is a master None : node is not a master

autoscale.**main**()
This function validates user arguments and data in configuration file. It calls auth class for authentication and autoscale to execute scaling policy

common.**check_file**(*fname*)
This function checks if file exists and is readable.

Parameters **fname** – file name

Returns file name with absolute path

common.**get_config**(*config_file*)
This function read and returns jsons configuration data

Parameters **config_file** – json configuration file name

Returns json data

common.**get_group_value**(*config*, *group*, *key*)

This function returns value in autoscale_groups section associated with provided key.

common.**get_logger**()
This function instantiate the logger.

Returns logger

common.**get_machine_uuid**()
This function uses subprocess to get node uuid and cached it for future use

Returns server uuid None

common.**get_user_value**(*args*, *config*, *key*)

This function returns value associated with the key if its available in user arguments else in json config file.

Parameters

- **args** – user arguments

- **config** – json configuration data
- **key** – key name

Returns value associated with key

common.**get_webhook_value**(*config*, *group*, *key*)

This function returns value in webhooks section of json file which is associated with provided key.

Parameters

- **group** – group name
- **config** – json configuration data
- **key** – key name

Returns value associated with key

common.**webhook_call**(*config_data*, *group*, *policy*, *key*)

This function makes webhook calls.

Parameters

- **config_data** – json configuration data
- **group** – group name
- **policy** – policy type
- **key** – key name

class auth.**Auth**(*username*, *apikey*, *region*, *identity_type='rackspace'*, *token_filename='/home/docs/.rax-autoscaler-token'*)

This class implements Rackspace cloud account authentication

authenticate()

This method loads a token from a file, authenticate with it, and if it fails then tries to authenticate with credentials

Returns True or False (Boolean)

authenticate_credentials()

This method try to authenticate with available credentials

Returns True or False (Boolean)

authenticate_token()

This authenticate with Rackspace cloud using existing token.

Returns True or False (Boolean)

force_unauthenticate()

This unauthenticate and delete token file

load_token()

This loads token from a file

Returns True or False (Boolean)

save_token()

This saves token to a file

Returns True or False (Boolean)

status()
This queries pyrax for values

Returns list – the return value

token_filename
This returns token filename

Returns _token_filename

cloudmonitor.**add_cm_check**(*server_id*, *check_type*, *check_config*)

This function adds Cloud Monitoring cpu check to a server, if it is not already present

Parameters **server_id** – server identity

Returns int – the return code 1 – Success

cloudmonitor.**get_entity**(*agent_id*)
This function get entity for passed agent_id (agent_id := server_uuid)

Parameters **agent_id** – agent id

cloudmonitor.**get_server**(*server_id*)
It gets Cloud server object by server_id

cloudmonitor.**get_server_ipv4**(*server_id*, *_type='public'*)
It gets public IP v4 server address

Parameters **server_id** – server id

cloudmonitor.**get_server_name**(*server_id*)
It returns the name of a server, using cached values

Parameters **server_id** – server id

Returns d_cached_servers_name[server_id] (str)

cloudmonitor.**is_ipv4**(*address*)
It checks if address is valid IP v4

cloudmonitor.**scaling_group_servers**(*sgid*)
list servers' id in scaling group sgid

Parameters **sgid** – scaling group id

class colouredconsolehandler.**ColouredConsoleHandler**(*stream=None*)
console handler with ANSI colours support Usage example:

def white_bold_underlined(self, msg):

return self.decorate(self.BOLD + self.UNDERLINE + self.COLOR['white'], msg)

ref: http://cwoebker.com/posts/ansi-escape-codes

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

CHAPTER 4

License

a

c

A

C

E

F

G

I

L

M

S

T

W