Clustercron Documentation

Release 0.5.4

Maarten

Oct 21, 2021

Contents:

Contents:

CHAPTER 1

Clustercron

Clustercron is cronjob wrapper that tries to ensure that a script gets run only once, on one host from a pool of nodes of a specified loadbalancer. **Clustercron** select a *master* from all nodes and will run the cronjob only on that node.

- Free software: ISC license
- Documentation: https://clustercron.readthedocs.org/en/latest/

1.1 Features

Supported load balancers (till now):

- AWS Elastic Load Balancing (ELB)
- AWS Elastic Load Balancing v2 (ALB)

CHAPTER 2

Clustercron AWS LB

2.1 Clustercron ELB

Clustercron ELB will retrieve the *instance ID* of the instance (node in a load balanced cluster) it is running on.

After retrieving the *health states* of all the nodes in the *load balancer group*, clustercron checks if the node's *instance ID* is the first in the (alphabetic) list of instances that have the state *InService*. If so it will consider the node *master* and will run the given *command*.

If no *command* is given clustercron will exit 0 if the node is considered *master* else clustercron will return 1.

2.2 Clustercron ALB

Clustercron ALB will retrieve the *instance ID* of the instance (node in a target group) it is running on.

After retrieving the *health states* of all the nodes in the *target group*, **Clustercron** checks if the node's *instance ID* is the first in the (alphabetic) list of instances that have the state *healthy*. If so it will consider the node *master* and will run the given *command*.

If no *command* is given clustercron will exit 0 if the node is considered *master* else clustercron will return 1.

2.3 Cron job times and clustercron timeouts

Cron jobs wrapped with **Clustercron** should be started on the same time on every node in the cluster. For syncronized time a **NTP** (or **Chrony**) client is strongly advised.

Clusteron's timeout and retries settings should be minimized to syncronize clustercron runs as much as possible. A time out of 2 seconds and a maximum of 1 retry is advised.

2.4 Boto 2

Clustercron ELB and **Clustercron ALB** both uses boto2 for retreiving the node's *instance ID* and optionally the node's *region* (if no *region* is configured).

These actions don't need any boto2 configuration or access management.

2.5 Boto 3

Clustercron ELB and **Clustercron ALB** both uses boto3 for retreiving the *health states* of all the nodes in the *load balancer group* (**ELB**) or the *target group* (**ALB**).

AWS policies are needed for retreiving this information. The *AWS policies* can be directly attached to *Users/Groups* or to *Roles* attached to the *AWS EC2 instances*.

When a *AWS Policy* is directly attached to a user, credentials need to be configured.

2.6 AWS Policy ELB

The user (or role) for checking the AWS **Load Balancer Groups** has to have the following read rights:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeInstanceHealth",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeLoadBalancerPolicyTypes",
                "elasticloadbalancing:DescribeLoadBalancerPolicies",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeTags"
            ],
            "Resource": "*"
        }
    ]
}
```

2.7 AWS Policy ALB

The user (or role) for checking the AWS **Target Groups** has to have the following read rights:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth"
```

(continues on next page)

(continued from previous page)

```
            ],
            "Resource": "*"
        }
    ]
}
```

2.8 AWS credentials

When *AWS policies* are directly attached to a user, credentials be configured in the *~/.aws/credentials* file.

When *AWS policies* are attached to *AWS Roles* there is no need for this configuration.

2.9 AWS config

When no *region* is configured **Clustercfron ELB** and **Clustercron ALB** will get the **AWS region** of the instance it is running on (using *instance_metadata*). If a *region* is configured, it will use the configured *region*.

Example configuration *~/.aws/config*:

```
[default]
region = eu-west-1
```

CHAPTER 3

Installation

3.1 Virtualenv

- Make virtual environment:

```
$ virtualenv /<path>/<to>/<virtualenv_name>
```

- Activate virtual environment:

```
$ source /<path>/<to>/<virtualenv_name>/bin/activate
```

- Install *clustercron*:

```
(<virtualenv_name>)$ pip install clustercron
```

CHAPTER 4

Configuration

Clustercron's operations are mostly handled by command line options.

See *Usage* for more information.

There are some options that can be configured in *Clusetercron*'s configuration file. For now only cache options are stated in the configuration.

Clustercron tries to read configuration from */etc/clustercron.ini* and after *.clustercron.ini* from the user's home directory. Options from the first will be overridden by the latter (per option).

Default values for *clustercron.ini*:

```
; clustercron.ini
[cache]
filename = /tmp/clustercron_cache.json
expire_time = 59
max_iter = 20
```

CHAPTER 5

Usage

5.1 clustercron –help

```
Clustercron, cluster cronjob wrapper.

Usage:
    clustercron [options] elb <load_balancer_name> [<cron_command>]
    clustercron [options] alb <target_group_name> [<cron_command>]
    clustercron -h | --help
    clustercron --version

Options:
    -v --verbose  Info logging. Add extra `-v` for debug logging.
    -s --syslog   Log to (local) syslog.
    -c --cache    Cache output from master check.
    -o --output   Output stdout and stderr from <cron_command>.

Clustercron is cronjob wrapper that tries to ensure that a script gets run
only once, on one host from a pool of nodes of a specified loadbalancer.

Without specifying a <cron_command> clustercron will only check if the node
is the `master` in the cluster and will return 0 if so.
```

5.2 Command line examples

Clustercron can be run from command line for debugging.

For **Clustercron ELB** a *Load Balancer Name*. must be specified.

For **Clustercron ALB** a *Target Group Name*. must be specified.

Without a *command* specified clustron will test is the node is *master* and return 0 if that is the case:

```
$ clustercron elb mylbname
$ echo $?
0
```

On a node not determined as 'master':

```
$ clustercron elb mylbname || echo "I'm not master"
I'm not master
```

On a node determined as 'master':

```
$ clustercron elb mylbname && echo "I'm master"
I'm master
```

With options *-v* or *–verbose* **clustercron** will output vebose info to console.

Check if node is determined as master with verbose (info) output:

```
$ clustercron -v elb mylbname
```

Check if node is not determined as master with verbose (info) output:

```
$ clustercron -v elb mylbname
INFO     clustercron.lb : Instance ID: i-05cc16d1d054104f2
INFO     clustercron.elb : Healty instances: i-05cc16d1d054104f2, i-0f13be692f5f35b5e
```

With options *-s* or *–syslog* and *-v* or *–verbose* **clustercron** will only output to syslog. ELB example:

```
$ clustercron -v -s elb mylbname echo test
$ sudo tail /var/log/messages
Jul 31 13:58:24 ip-172-31-7-231 journal: clustercron.lb [31512]: Instance ID: i-
↪05cc16d1d054104f2
Jul 31 13:58:24 ip-172-31-7-231 journal: clustercron.elb [31512]: Healty instances: i-
↪05cc16d1d054104f2, i-0f13be692f5f35b5e
Jul 31 13:58:24 ip-172-31-7-231 journal: clustercron.main [31512]: run command: echo␣
↪test
Jul 31 13:58:24 ip-172-31-7-231 journal: clustercron.main [31512]: stdout: test
Jul 31 13:58:24 ip-172-31-7-231 journal: clustercron.main [31512]: stderr:
Jul 31 13:58:24 ip-172-31-7-231 journal: clustercron.main [31512]: returncode: 0
```

With options *-s* or *–syslog* and *-v* or *–verbose* **clustercron** will only output to syslog ALB example:

```
$ clustercron -v -s alb mytargetgroupname echo test
Jul 31 14:04:53 ip-10-0-2-129 journal: clustercron.lb [31204]: Instance ID: i-
↪05d7670fb9114c58e
Jul 31 14:04:53 ip-10-0-2-129 journal: clustercron.alb [31204]: Healty instances: i-
↪05d7670fb9114c58e, i-0b5be35c81d1b50d4
Jul 31 14:04:53 ip-10-0-2-129 journal: clustercron.main [31204]: run command: echo␣
↪test
Jul 31 14:04:53 ip-10-0-2-129 journal: clustercron.main [31204]: stdout: test
Jul 31 14:04:53 ip-10-0-2-129 journal: clustercron.main [31204]: stderr:
Jul 31 14:04:53 ip-10-0-2-129 journal: clustercron.main [31204]: returncode: 0
```

By default **clustercron** will not output *stdout* and *stderr* generated by the 'cron-command':

```
$ clustercron elb mylbname echo test
$
```

With option *-o* or *–output* ‘ **clustercron** will output *stdout* and *stderr* generated by the ‘cron-command’ (when the node is determined as ‘master’):

```
$ clustercron -o elb mylbname echo test
test
```

Output redirection with options *-o* or *–output*:

```
$ clustercron -o elb mylbname echo test > /tmp/output
$ cat /tmp/output
test
```

stdout and *stderr* separated with redirection:

```
$ clustercron -o elb mylbname cat non_existing_file  1>/tmp/output 2>/tmp/error
$ cat /tmp/output
$ cat /tmp/error
cat: non_existing_file: No such file or directory
```

Be aware that redirection takes place on **clustercron** itself, not on the ‘cron-command’. So redirection will also take place when the node is not determined as ‘master’.

When a node is not determind as ‘master’ an empty file will be created when redirecting output from **clustercron**:

```
$ clustercron -o elb clustercrontest echo test  > /tmp/output
$ cat /tmp/output
```

When redirection is only wanted on the ‘cron-command’ when **clustercron** determined a node as ‘master’ a ‘cron-command’ could be wrapped:

```
$ cat wrapped_cron_command.sh
#!/bin/sh
echo test > /tmp/output
```

On a node determined as ‘master:

```
$ clustercron -o elb clustercrontest wrapped_cron_command.sh
$ cat /tmp/output
test
```

On a node not determined as ‘master:

```
$ clustercron -o elb clustercrontest wrapped_cron_command.sh
$ cat /tmp/output
cat: /tmp/output: No such file or directory
```

5.3 Cron entry example

Every day at 5 min to midnight run the command *logger “clustercron run”* on the node that will be picked master . Log with level INFO to syslog:

```
55 23 * * * /<path>/<to>/<virtualenv_name>/bin/clustercron -v -s elb <lb name> logger
↪"clustercron run"
```

5.4 Caching

Clustercron's *master selection* can be cached with the options *-c* or *–cache*:

```
$ clustercron -c elb mylbname echo test
```

By default the cache will stored in */tmp/clustercron_cache.json* and expire after 59 seconds. **Clustercron** will lock the cache file and tries to by default 20 times when the file is locked.

The defaults for caching can only be altered in **Clustercron's** configuration file.

See *Configuration* for more information.

CHAPTER 6

clustercron

6.1 clustercron package

6.1.1 Submodules

6.1.2 clustercron.alb module

clustercron.alb

Modules holds class for AWS ElasticLoadBalancing v2 (ALB)

class clustercron.alb.**Alb**(*name*)
Bases: *clustercron.lb.Lb*

get_healty_instances()

6.1.3 clustercron.cache module

clustercron.cache

class clustercron.cache.**Cache**
Bases: object

expired(*expire_time*)

static iso2datetime_hook(*dct*)

static json_serial(*obj*)
JSON serializer for objects not serializable by default json code

load_json(*fp*)

safe_json(*fp*)

set_now()

clustercron.cache.**check**(*master_check*, *filename*, *expire_time*, *max_iter*)

6.1.4 clustercron.config module

6.1.5 clustercron.elb module

clustercron.elb

Modules holds class for AWS ElasticLoadBalancing (ELB)

class clustercron.elb.**Elb**(*name*)
Bases: *clustercron.lb.Lb*

get_healty_instances()

6.1.6 clustercron.lb module

clustercron.lb

Modules holds base class for AWS ElasticLoadBalancing classes

class clustercron.lb.**Lb**(*name*)
Bases: object

get_healty_instances()

master()

6.1.7 clustercron.main module

clustercron.main

class clustercron.main.**Optarg**(*arg_list*)
Bases: object

Parse arguments from *sys.argv[0]* list. Set usage string. Set properties from arguments.

parse()

clustercron.main.**clustercron**(*lb_type*, *name*, *command*, *output*, *use_cache*)
API clustercron

Parameters

- **lb_type** – Type of loadbalancer
- **name** – Name of the loadbalancer instance
- **command** – Command as a list
- **output** – Boolean

clustercron.main.**command**()
Entry point for the package, as defined in setup.py.

clustercron.main.**setup_logging**(*verbose*, *syslog*)
Sets up logging.

6.1.8 Module contents

CHAPTER 7

History

7.1 1.0.0 (2021-10-21)

- dropped python2

7.2 0.6.14 (2021-09-12)

- setuptools_scm (no more bumpversion).

7.3 0.6.13 (2021-09-10)

- More dev env stuff: removed requirements files, all deps in *setup.cfg* now.

7.4 0.6.12 (2021-09-10)

- Fix some shit from migrating master -> main
- Makefile updates

7.5 0.6.11 (2021-09-09)

- More test, deploy and build update , still no code change B)

7.6 0.6.4 (2021-09-05)

- Updated deployment env.

7.7 0.6.3 (2021-09-03)

- Updated requirements

7.8 0.6.2 (2020-04-02)

- Updated requirements

7.9 0.6.1 (2019-09-13)

- Make the region entry in ~/.aws/config optional
- Bug fix Cache file can contain incompatible time format

7.10 0.5.4 (2019-07-31)

- Added boto3 requirements to setup.py
- Docs update

7.11 0.5.2 (2019-07-31)

- Added ElasticLoadBalancingv2 (ALB) support.
- Update requirements

7.12 0.4.10 (2016-10-14)

- Updated dev requirements
- Updated test requirements in setup.py

7.13 0.4.9 (2016-08-28)

- Update requirements
- Removed pinned requirements from setup.py

7.14 0.4.8 (2016-08-20)

- Update requirements: pytest -> 3.0.0

7.15 0.4.7 (2016-08-13)

- Travis/Tox fixes.

7.16 0.4.6 (2016-08-13)

- Added twine to requirements_dev.txt

7.17 0.4.5 (2016-08-13)

- Added pyup.io
- ISC License
- pinned requirements

7.18 0.4.4 (2016-05-27)

- NOQA for false positive in pyflakes

7.19 0.4.1 (2016-05-21)

- Fixed Python3 unicode compatibility issue for json module.

7.20 0.4.0 (2016-05-21)

- Added Caching of *master selection*.

7.21 0.3.7.dev1 (2015-09-12)

- Added option '-o' '–output' for output of wrapped 'cron command'.

7.22 0.3.6 (2015-08-08)

- Add more tests.
- syslog unix_socket path follows symbolic links (fedora)

7.23 0.3.5 (2015-08-07)

- Urllib refactoring with requests.
- Use responses for tests.
- Factored out Mock objects.
- Removed OS X 'open' command from makefile.
- Removed python 2/3 compatibilty module.
- Removed unused exceptions module.

7.24 0.3.4 (2015-07-12)

- Correction in docs/usage.rst

7.25 0.3.3 (2015-07-12)

- Remove :ref: tag from README.rst (for formatting on PyPi)

7.26 0.3.2 (2015-07-12)

- Fix mock requirements in tox.ini (mock 1.1.1 doesn't work with Python 2.6)

7.27 0.3.1 (2015-06-28)

- First release (beta status)

7.28 0.3.0 (2015-06-28)

- First release

7.29 0.3.0.dev2 (2015-06-21)

- First real working version for ELB

7.30 0.3.0.dev1 (2015-06-17)

- First working version for ELB

7.31 0.2.0.dev2 (2015-05-25)

- In Development stage 1
- Removed HAproxy for now.

7.32 0.1.3 (2015-05-22)

- Refactor command line argument parser

7.33 0.1.2 (2015-03-28)

- More test for commandline
- Travis stuff

7.34 0.1.0 (2015-01-23)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index