ceph-medic Documentation

Release 0.0.1

Andrew Schoen, Alfredo Deza

Sep 18, 2020

Contents

CHAPTER 1

Introduction

ceph-medic is a very simple tool that runs against a Ceph cluster to detect common issues that might prevent correct functionality. It requires non-interactive SSH access to accounts that can sudo without a password prompt.

1.1 Usage

The basic usage of ceph-medic is to perform checks against a ceph cluster to identify potential issues with its installation or configuration. To do this, run the following command:

```
ceph-medic --inventory /path/to/hosts --ssh-config /path/to/ssh_config check
```

1.1.1 Inventory

ceph-medic needs to know the nodes that exist in your ceph cluster before it can perform checks. The inventory (or hosts file) is a typical Ansible inventory file and will be used to inform ceph-medic of the nodes in your cluster and their respective roles. The following standard host groups are supported by ceph-medic: mons, osds, rgws, mdss, mgrs and clients. An example hosts file would look like:

```
[mons]
mon0
mon1

[osds]
osd0

[mgrs]
mgr0
```

The location of the hosts file can be passed into ceph-medic by using the --inventory cli option (e.g ceph-medic --inventory /path/to/hosts).

If the `--inventory` option is not defined, `ceph-medic` will first look in the current working directory for a file named `hosts`. If the file does not exist, it will look for `/etc/ansible/hosts` to be used as the inventory.

Note: Defining the inventory location is also possible via the config file under the `[global]` section.

1.1.2 Inventory for Containers

Containerized deployments are also supported, via `docker` and `podman`. As with `baremetal` deployments, an inventory file is required. If the cluster was deployed with `ceph-ansible`, you may use that existing inventory.

To configure ceph-medic to connect to a containerized cluster, the glocal section of the configuration needs to define `deployment_type` to either `docker` or `podman`. For example:

```
[global]

deployment_type = podman
```

1.1.3 Inventory for Container Platforms

Both `kubernetes` and `openshift` platforms can host containers remotely, but do allow to connect and retrieve information from a central location. To configure ceph-medic to connect to a platform, the glocal section of the configuration needs to define `deployment_type` to either `kubernetes`, which uses the `kubectl` command, or `openshift`, which uses the `oc` command. For example:

```
[global]

deployment_type = openshift
```

When using `openshift` or `kubernetes` as a deployment type, there is no requirement to define a `hosts` file. The hosts are generated dynamically by calling out to the platform and retrieving the pods. When the pods are identified, they are grouped by deamon type (osd, mgr, rgw, mon, etc...).

1.1.4 SSH Config

All nodes in your `hosts` file must be configured to provide non-interactive SSH access to accounts that can `sudo` without a password prompt.

Note: This is the same ssh config required by ansible. If you've used `ceph-ansible` to deploy your cluster then your nodes are most likely already configured for this type of ssh access. If that is the case, using the same user that performed the initial deployment would be easiest.

To provide your ssh config you must use the `--ssh-config` flag and give it a path to a file that defines your ssh configuration. For example, a file like this is used to connect with a cluster comprised of vagrant vms:

```
Host mon0
  HostName 127.0.0.1
  User vagrant
  Port 2200
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
```

(continues on next page)

(continued from previous page)

```
  PasswordAuthentication no
  IdentityFile /Users/andrewschoen/.vagrant.d/insecure_private_key
  IdentitiesOnly yes
  LogLevel FATAL

Host osd0
  HostName 127.0.0.1
  User vagrant
  Port 2201
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile /Users/andrewschoen/.vagrant.d/insecure_private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

Note: SSH configuration is not needed when using `kubernetes` or `openshift`

1.1.5 Logging

By default `ceph-medic` sends complete logs to the current working directory. This log file is more verbose than the output displayed on the terminal. To change where these logs are created, modify the default value for `--log-path` in `~/.cephmedic.conf`.

1.1.6 Running checks

To perform checks against your cluster use the `check` subcommand. This will perform a series of general checks, as well as checks specific to each daemon. Sample output from this command will look like:

```
ceph-medic --ssh-config vagrant_ssh_config check
Host: mgr0                  connection: [connected  ]
Host: mon0                  connection: [connected  ]
Host: osd0                  connection: [connected  ]
Collection completed!

=======================  Starting remote check session  ========================
Version: 0.0.1    Cluster Name: "test"
Total hosts: [3]
OSDs:    1    MONs:    1     Clients:    0
MDSs:    0    RGWs:    0     MGRs:       1

================================================================================

---------- managers ----------
 mgr0

------------ osds ------------
 osd0

------------ mons ------------
 mon0
```

(continues on next page)

(continued from previous page)

```
17 passed, 0 errors, on 4 hosts
```

The logging can also be configured in the `cephmedic.conf` file in the global section:

```
[global]
--log-path = .
```

To ensure that cluster checks run properly, at least one monitor node should have administrative privileges.

CHAPTER 2

Installation

ceph-medic supports a few different installation methods, including system packages for RPM distros via EPEL. For PyPI, it can be installed with:

```
pip install ceph-medic
```

2.1 Official Upstream Repos

Download official releases of ceph-medic at https://download.ceph.com/ceph-medic

Currently, only RPM repos built for CentOS 7 are supported.

ceph-medic has dependencies on packages found in EPEL, so EPEL will need to be enabled.

Follow these steps to install a CentOS 7 repo from download.ceph.com:

- Install the latest RPM repo from download.ceph.com:

```
wget http://download.ceph.com/ceph-medic/latest/rpm/el7/ceph-medic.repo -O /etc/
↪yum.repos.d/ceph-medic.repo
```

- Install epel-release:

```
yum install epel-release
```

- Install the GPG key for ceph-medic:

```
wget https://download.ceph.com/keys/release.asc
rpm --import release.asc
```

- Install ceph-medic:

```
yum install ceph-medic
```

- Verify your install:

```
ceph-medic --help
```

2.2 Shaman Repos

Every branch pushed to ceph-medic.git gets a RPM repo created and stored at shaman.ceph.com. Currently, only RPM repos built for CentOS 7 are supported.

Browse https://shaman.ceph.com/repos/ceph-medic to find the available repos.

Note: Shaman repos are available for 2 weeks before they are automatically deleted. However, there should always be a repo available for the master branch of `ceph-medic`.

`ceph-medic` has dependencies on packages found in EPEL, so EPEL will need to be enabled.

Follow these steps to install a CentOS 7 repo from shaman.ceph.com:

- Install the latest master shaman repo:

```
wget https://shaman.ceph.com/api/repos/ceph-medic/master/latest/centos/7/repo -O /
↪etc/yum.repos.d/ceph-medic.repo
```

- Install `epel-release`:

```
yum install epel-release
```

- Install `ceph-medic`:

```
yum install ceph-medic
```

- Verify your install:

```
ceph-medic --help
```

2.3 GitHub

You can install directly from the source on GitHub by following these steps:

- Clone the repository:

```
git clone https://github.com/ceph/ceph-medic.git
```

- Change to the `ceph-medic` directory:

```
cd ceph-medic
```

- Create and activate a Python Virtual Environment:

```
virtualenv venv
source venv/bin/activate
```

- Install ceph-medic into the Virtual Environment:

```
python setup.py install
```

`ceph-medic` should now be installed and available in the created virtualenv. Check your installation by running: `ceph-medic --help`

CHAPTER 3

Error Codes

When performing checks, `ceph-medic` will return an error code and message for any that failed. These checks can either be a `warning` or `error`, and will pertain to common issues or daemon specific issues. Any error code starting with `E` is an error, and any starting with `W` is a warning.

Below you'll find a list of checks that are performed with the `check` subcommand.

3.1 Common

The following checks indiciate general issues with the cluster that are not specific to any daemon type.

3.1.1 Warnings

WCOM1

A running OSD and MON daemon were detected in the same node. Colocating OSDs and MONs is highly discouraged.

3.1.2 Errors

ECOM1

A ceph configuration file cannot be found at `/etc/ceph/$cluster-name.conf`.

ECOM2

The `ceph` executable was not found.

ECOM3

The `/var/lib/ceph` directory does not exist or could not be collected.

ECOM4

The `/var/lib/ceph` directory was not owned by the `ceph` user.

ECOM5

The `fsid` defined in the configuration differs from other nodes in the cluster. The `fsid` must be the same for all nodes in the cluster.

ECOM6

The installed version of `ceph` is not the same for all nodes in the cluster. The `ceph` version should be the same for all nodes in the cluster.

ECOM7

The installed version of `ceph` is not the same as the one of a running ceph daemon. The installed `ceph` version should be the same as all running ceph daemons. If they do not match, the daemons most likely have not been restarted correctly after a version change.

ECOM8

The `fsid` field must exist in the configuration for each node.

ECOM9

A cluster should not have running daemons with a cluster `fsid` that is different from the rest of the daemons in a cluster. This potentially means that different cluster identifiers are being used, and that should not be the case.

ECOM10

Only a single monitor daemon shuld be running per host, having more than one monitor running on the same host reduces a cluster's resilience if the node goes down.

3.2 Monitors

The following checks indicate issues with monitor nodes.

3.2.1 Errors

EMON1

The secret key used in the keyring differs from other nodes in the cluster.

3.2.2 Warnings

WMON1

Multiple monitor directories are found on the same host.

WMON2

Collocated OSDs in monitor nodes were found on the same host.

WMON3

The recommended number of Monitor nodes is 3 for a high availability setup.

WMON4

It is recommended to have an odd number of monitors so that failures can be tolerated.

WMON5

Having a single monitor is not recommneded, as a failure would cause data loss. For high availability, at least 3 monitors is recommended.

3.3 OSDs

The following checks indicate issues with OSD nodes.

3.3.1 Warnings

WOSD1

Multiple ceph_fsid values found in /var/lib/ceph/osd.

This might mean you are hosting OSDs for many clusters on this node or that some OSDs are misconfigured to join the clusters you expect.

WOSD2

Setting `osd pool default min size = 1` can lead to data loss because if the minimum is not met, Ceph will not acknowledge the write to the client.

WOSD3

The default value of 3 OSD nodes for a healthy cluster must be met. If `ceph.conf` is configured to a different number, that setting will take precedence. The number of OSD nodes is calculated by adding `osd_pool_default_size` and `osd_pool_default_min_size` + 1. By default, this adds to 3.

WOSD4

If ratios have been modified from its defaults, a warning is raised pointing to any ratio that diverges. The ratios observed with their defaults are:

- `backfillfull_ratio`: 0.9
- `nearfull_ratio`: 0.85
- `full_ratio`: 0.95

3.4 Cluster

Cluster checks run once against the information of a cluster, and are not specific to any deamon.

3.4.1 Errors

ECLS1

No OSD nodes exist as part of the cluster.

ECLS2

The cluster is nearfull.

CHAPTER 4

Cluster node facts

Fact collection happens per node and creates a mapping of hosts and data gathered. Each daemon 'type' is the primary key:

```
...
'osd': {
    'node1': {...},
    'node2': {...},
}
'mon': {
    'node3': {...},
}
```

There are other top-level keys that make it easier to deal with fact metadata, for example a full list of all hosts discovered:

```
'hosts': ['node1', 'node2', 'node3'],
'osds': ['node1', 'node2'],
'mons': ['node3']
```

Each host has distinct metadata that gets collected. If any errors are detected, the `exception` key is set populated with all information pertaining to the error generated when trying to execute the call. For example, a failed call to `stat` on a path might be:

```
'osd': {
    'node1': {
        'paths': {
            '/var/lib/osd': {
                'exception': {
                    'traceback': "Traceback (most recent call last):\n File "remote.py
↪", line 3, in <module>\n os.stat('/var/lib/osd')\n OSError: [Errno 2] No such file␣
↪or directory: '/var/lib/osd'\n",
                    'name': 'OSError',
                    'repr': "[Errno 2] No such file or directory: '/root'"
                    'attributes': {
```

(continues on next page)

(continued from previous page)

```
                        args : "(2, 'No such file or directory')",
                        errno : 2,
                        filename :  '/var/lib/ceph' ,
                        message : '',
                        strerror :  'No such file or directory'
                    }
            }
        }
    }
}
```

Note that objects will not get pickled, so data structures and objects will be sent back as plain text.

Path contents are optionally enabled by the fact engine and will contain the raw representation of the full file contents. Here is an example of what a `ceph.conf` file would be in a monitor node:

```
'mon': {
    'node3': {
        'paths': {
            '/etc/ceph/': {
               'dirs': [],
               'files': {
                   '/etc/ceph/ceph.conf': {
                       'contents': "[global]\nfsid = f05294bd-6e9d-4883-9819-
↪c2800d4d7962\nmon_initial_members = node3\nmon_host = 192.168.111.102\nauth_cluster_
↪required = cephx\nauth_service_required = cephx\nauth_client_required = cephx\n",
                       'owner': 'ceph',
                       'group': 'ceph',
                       'n_fields' : 19 ,
                       'n_sequence_fields' : 10 ,
                       'n_unnamed_fields' : 3 ,
                       'st_atime' : 1490714187.0 ,
                       'st_birthtime' : 1463607160.0 ,
                       'st_blksize' : 4096 ,
                       'st_blocks' : 0 ,
                       'st_ctime' : 1490295294.0 ,
                       'st_dev' : 16777220 ,
                       'st_flags' : 1048576 ,
                       'st_gen' : 0 ,
                       'st_gid' : 0 ,
                       'st_ino' : 62858421 ,
                       'st_mode' : 16877 ,
                       'st_mtime' : 1490295294.0 ,
                       'st_nlink' : 26 ,
                       'st_rdev' : 0 ,
                       'st_size' : 884 ,
                       'st_uid' : 0 ,
                       'exception': {},
                   }
               }
            }
        }
    }
}
```

CHAPTER 5

1.0.8

17-Jun-2020

- Fix issues with podman support

CHAPTER 6

1.0.7

24-Mar-2020

- Fix test bugs that were breaking rpm builds

CHAPTER 7

1.0.6

11-Feb-2020

- Docker, podman container support
- Fix broken SSH config option
- Fix querying the Ceph version via admin socket on newer Ceph versions

CHAPTER 8

1.0.5

27-Jun-2019

- Add check for minimum OSD node count
- Add check for minimum MON node count
- Remove reporting of nodes that can't connect, report them separetely
- Kubernetes, Openshift, container support
- Fix unidentifiable user/group ID issues
- Rook support
- Report on failed nodes
- When there are errors, set a non-zero exit status
- Add separate "cluster wide" checks, which run once
- Be able to retrieve socket configuration
- Fix issue with trying to run `whoami` to test remote connections, use `true` instead
- Add check for missing FSID
- Skip OSD validation when there isn't any ceph.conf
- Skip tmp directories in /var/lib/ceph scanning to prevent blowing up
- Detect collocated daemons
- Allow overriding ignores in the CLI, fallback to the config file
- Break documentation up to have installation away from getting started

CHAPTER 9

1.0.4

20-Aug-2018

- Add checks for parity between installed and socket versions
- Fix issues with loading configuration with whitespace
- Add check for min_pool_size
- Collect versions from running daemons