Glusto-tests Documentation

Release 0.1

Gluster Community

Feb 04, 2018

Contents

Glusto-tests is a repository for gluster-related test libraries using Glusto.

CHAPTER 1

User Guide

1.1 Glusto Tests

The **glusto-tests** repo contains automated testcases for testing gluster software. It provides the Libraries/Modules necessary for automating tests for gluster under the Glusto framework.

The Libraries/Modules/Tests in glusto-tests are written using the 'glusto' framework. TestCases in glusto-tests can we written/run using standard PyUnit, PyTest or Nose methodologies as supported by 'glusto' framework.

Refer to 'http://glusto.readthedocs.io/en/latest/' for info on 'glusto' framework.

1.1.1 Prereqs

To automate/run glusto-tests we need to install the following packages:

- glusto
- glustolibs-gluster
- glustolibs-io

1.1.2 Installing Glusto

- pip install

```
# pip install --upgrade git+git://github.com/loadtheaccumulator/glusto.git
```

. . . or . . .

- git clone

```
# git clone https://github.com/loadtheaccumulator/glusto.git
# cd glusto
# python setup.py
```

Refer to: http://glusto.readthedocs.io/en/latest/userguide/install.html

1.1.3 Installing Glustolibs

To install the glustolibs-gluster and glustolibs-io libraries...

```
# git clone http://review.gluster.org/glusto-tests
# cd glusto-tests/glustolibs-gluster
# python setup.py install
# cd glusto-tests/glustolibs-io
# python setup.py install
```

1.2 How to run the test case

- Create config file containing info about the servers, clients, volumes, mounts. Please refer to example config file under tests directory in glusto-tests repo. The example config file is in yaml format and defines sections which provides info about the gluster cluster. We can use any 'glusto' framework supported formats for writing the config files. Refer : http://glusto.readthedocs.io/en/latest/userguide/configurable.html
- glusto-tests are run using the 'glusto' command available after installing the glusto framework. The various options to run tests as provided by glusto framework:

To run PyUnit tests

```
# glusto -c 'config.yml' -d 'tests'
# glusto -c 'config.yml unittest_list.yml' -u
```

To run PyTest tests:

```
# glusto -c 'config.yml' --pytest='-v -x tests -m bvt'
```

To run Nose tests:

```
    # glusto -c 'config.yml' --nosetests='-v -w tests'

Refer: http://glusto.readthedocs.io/en/latest/userguide/glusto.html#options-for-
↪running-unit-tests
```

1.3 Writing tests in glusto-tests:

The *tests* directory in glusto-tests contains testcases. One might want to create a dir with feature name as the name of test directory under *tests* to add new testcases.

TestCases in glusto-tests can we written using standard PyUnit, PyTest or Nose methodologies as supported by 'glusto' framework.

For more information on PyUnit tests, see http://glusto.readthedocs.io/en/latest/userguide/unittest.html

For more information on PyTest tests, see http://glusto.readthedocs.io/en/latest/userguide/pytest.html

For more informaiton on Nose tests, see http://glusto.readthedocs.io/en/latest/userguide/nosetests.html

1.4 Logging

The Log file name and Log level can be passed as argument to glusto command while running the glusto-tests. For example:

```
# glusto -c 'config.yml' -l /tmp/glustotests_bvt.log --log-level DEBUG --pytest='-v -
↪x tests -m bvt'
```

One can configure log files, log levels in the testcases as well. For details on how to use glusto framework for configuring logs in tests Refer to: http://glusto.readthedocs.io/en/latest/userguide/loggable.html

Default log location is '/tmp/glustomain.log'.

Note: When using 'glusto' via the Python Interactive Interpreter, the default log location is '/tmp/glusto.log'.

CHAPTER 2

API

2.1 glustolibs package

2.1.1 Subpackages

glustolibs.gluster package

Submodules

glustolibs.gluster.bitrot_ops module

Description: Library for gluster bitrot operations.

glustolibs.gluster.bitrot_ops.**enable_bitrot**(*mnode*, *volname*)
Enables bitrot for given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

enable_bitrot("abc.com", testvol)

glustolibs.gluster.bitrot_ops.**disable_bitrot**(*mnode*, *volname*)
Disables bitrot for given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

disable_bitrot("abc.com", testvol)

glustolibs.gluster.bitrot_ops.**is_bitrot_enabled**(*mnode*, *volname*)
Checks if bitrot is enabled on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns True on success, False otherwise

Example

is_bitrot_enabled("abc.com", testvol)

glustolibs.gluster.bitrot_ops.**is_file_signed**(*mnode*, *filename*, *volname*, *expected_file_version=None*)
Verifies if the given file is signed

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **filename** (*str*) – relative path of filename to be verified
- **volname** (*str*) – volume name

Kwargs:

expected_file_version (str): file version to check with getfattr output If this option is set, this function will check file versioning as part of signing verification. If this option is set to None, function will not check for file versioning. Defaults to None.

Returns True on success, False otherwise

Example

is_file_signed("abc.com", 'file1', "testvol", expected_file_version='2')

glustolibs.gluster.bitrot_ops.**is_file_bad**(*mnode*, *filename*)
Verifies if scrubber identifies bad file :param filename: absolute path of the file in mnode :type filename: str :param mnode: Node on which cmd has to be executed. :type mnode: str

Returns True on success, False otherwise

Example

is_file_bad("abc.xyz.com", "/bricks/file1")

glustolibs.gluster.bitrot_ops.**bring_down_bitd**(*mnode*)
Brings down bitd process :param mnode: Node on which cmd has to be executed. :type mnode: str

Returns True on success, False otherwise

Example

bring_down_bitd("abc.xyz.com")

glustolibs.gluster.bitrot_ops.**bring_down_scrub_process**(*mnode*)
Brings down scrub process :param mnode: Node on which cmd has to be executed. :type mnode: str

Returns True on success, False otherwise

Example

bring_down_scrub_process("abc.xyz.com")

glustolibs.gluster.bitrot_ops.**set_scrub_throttle**(*mnode*, *volname*, *throttle_type='lazy'*)
Sets scrub throttle

Parameters

- **volname** (*str*) – volume name
- **mnode** (*str*) – Node on which cmd has to be executed.

Kwargs:

throttle_type (str): throttling type (lazy|normal|aggressive) Defaults to 'lazy'

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

set_scrub_throttle(mnode, testvol)

glustolibs.gluster.bitrot_ops.**set_scrub_frequency**(*mnode*, *volname*, *frequency_type='biweekly'*)

Sets scrub frequency

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

frequency_type (str): frequency type (hourly|daily|weekly|biweekly| monthly). Defaults to 'biweekly'

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

set_scrub_frequency("abc.com", testvol)

glustolibs.gluster.bitrot_ops.**pause_scrub**(*mnode*, *volname*)

Pauses scrub

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

pause_scrub("abc.com", testvol)

glustolibs.gluster.bitrot_ops.**resume_scrub**(*mnode*, *volname*)

Resumes scrub

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

resume_scrub("abc.com", testvol)

glustolibs.gluster.bitrot_ops.**get_bitd_pid**(*mnode*)
Gets bitd process id for the given node :param mnode: Node on which cmd has to be executed. :type mnode: str

Returns pid of the bitd process on success NoneType: None if command execution fails, errors.

Return type str

Example

get_bitd_pid("abc.com")

glustolibs.gluster.bitrot_ops.**get_scrub_process_pid**(*mnode*)
Gets scrub process id for the given node :param mnode: Node on which cmd has to be executed. :type mnode: str

Returns pid of the scrub process on success NoneType: None if command execution fails, errors.

Return type str

Example

get_scrub_process_pid("abc.com")

glustolibs.gluster.bitrot_ops.**is_bitd_running**(*mnode*, *volname*)
Checks if bitd is running on the given node

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns True on success, False otherwise

Example

is_bitd_running("abc.com", "testvol")

glustolibs.gluster.bitrot_ops.**is_scrub_process_running**(*mnode*, *volname*)
Checks if scrub process is running on the given node

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns True on success, False otherwise

Example

is_scrub_process_running("abc.com", "testvol")

glustolibs.gluster.bitrot_ops.**scrub_status**(*mnode*, *volname*)
Executes gluster bitrot scrub status command

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

scrub_status("abc.com", testvol)

glustolibs.gluster.bitrot_ops.**get_scrub_status**(*mnode*, *volname*)
Parse the output of gluster bitrot scrub status command

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns scrub status in dict format NoneType: None if command execution fails, errors.

Return type dict

Example

>>>get_scrub_status("abc.com", testvol) {'State of scrub': 'Active', 'Bitrot error log location': '/var/log/glusterfs/bitd.log', 'Scrub impact': 'aggressive', 'Scrub frequency': 'hourly', 'status_info': {'localhost': {'Duration of last scrub (D:M:H:M:S)': '0:0:0:0', 'corrupted_gfid': ['475ca13f-577f-460c-a5d7-ea18bb0e7779'], 'Error count': '1', 'Last completed scrub time': '2016-06-21 12:46:19', 'Number of Skipped files': '0', 'Number of Scrubbed files': '0'}, '10.70.47.118': {'Duration of last scrub (D:M:H:M:S)': '0:0:0:1', 'corrupted_gfid': ['19e62b26-5942-4867-a2f6-e354cd166da9', 'fab55c36-0580-4d11-9ac0-d8e4e51f39a0'], 'Error count': '2', 'Last completed scrub time': '2016-06-21 12:46:03', 'Number of Skipped files': '0', 'Number of Scrubbed files': '2'}}, 'Volume name': 'testvol', 'Scrubber error log location': '/var/log/glusterfs/scrub.log'}

glustolibs.gluster.brick_libs module

Description: Module for gluster brick related helper functions.

glustolibs.gluster.brick_libs.**get_all_bricks**(*mnode*, *volname*)

Get list of all the bricks of the specified volume. If the volume is 'Tier' volume, the list will contian both 'hot tier' and 'cold tier' bricks.

Parameters

- **mnode** (*str*) – Node on which command has to be executed
- **volname** (*str*) – Name of the volume

Returns List of all the bricks of the volume on Success. NoneType: None on failure.

Return type list

glustolibs.gluster.brick_libs.**get_hot_tier_bricks**(*mnode*, *volname*)

Get list of hot-tier bricks of the specified volume

Parameters

- **mnode** (*str*) – Node on which command has to be executed
- **volname** (*str*) – Name of the volume

Returns List of hot-tier bricks of the volume on Success. NoneType: None on failure.

Return type list

glustolibs.gluster.brick_libs.**get_cold_tier_bricks**(*mnode*, *volname*)

Get list of cold-tier bricks of the specified volume

Parameters

- **mnode** (*str*) – Node on which command has to be executed
- **volname** (*str*) – Name of the volume

Returns List of cold-tier bricks of the volume on Success. NoneType: None on failure.

Return type list

glustolibs.gluster.brick_libs.**bring_bricks_offline**(*volname*, *bricks_list*, *bring_bricks_offline_methods=None*)

Bring the bricks specified in the bricks_list offline.

Parameters

- **volname** (*str*) – Name of the volume
- **bricks_list** (*list*) – List of bricks to bring them offline.

Kwargs:

bring_bricks_offline_methods (list): List of methods using which bricks will be brought offline. The method to bring a brick offline is randomly selected from the bring_bricks_offline_methods list. By default all bricks will be brought offline with 'service_kill' method.

Returns

True on successfully bringing all bricks offline. False otherwise

Return type bool

glustolibs.gluster.brick_libs.**bring_bricks_online**(*mnode*, *volname*, *bricks_list*, *bring_bricks_online_methods=None*)

Bring the bricks specified in the bricks_list online.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.
- **bricks_list** (*list*) – List of bricks to bring them online.

Kwargs:

bring_bricks_online_methods (list): List of methods using which bricks will be brought online. The method to bring a brick online is randomly selected from the bring_bricks_online_methods list. By default all bricks will be brought online with ['glusterd_restart', 'volume_start_force'] methods. If 'volume_start_force' command is randomly selected then all the bricks would be started with the command execution. Hence we break from bringing bricks online individually

Returns

True on successfully bringing all bricks online. False otherwise

Return type bool

glustolibs.gluster.brick_libs.**are_bricks_offline**(*mnode*, *volname*, *bricks_list*)

Verify all the specified list of bricks are offline.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.
- **bricks_list** (*list*) – List of bricks to verify offline status.

Returns True if all bricks offline. False otherwise. NoneType: None on failure in getting volume status

Return type bool

glustolibs.gluster.brick_libs.**are_bricks_online**(*mnode*, *volname*, *bricks_list*)

Verify all the specified list of bricks are online.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.
- **bricks_list** (*list*) – List of bricks to verify online status.

Returns True if all bricks online. False otherwise. NoneType: None on failure in getting volume status

Return type bool

glustolibs.gluster.brick_libs.**get_offline_bricks_list**(*mnode*, *volname*)

Get list of bricks which are offline.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Returns List of bricks in the volume which are offline. NoneType: None on failure in getting volume status

Return type list

glustolibs.gluster.brick_libs.**get_online_bricks_list**(*mnode*, *volname*)
Get list of bricks which are online.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Returns List of bricks in the volume which are online. NoneType: None on failure in getting volume status

Return type list

glustolibs.gluster.brick_libs.**delete_bricks**(*bricks_list*)
Deletes list of bricks specified from the brick nodes.

Parameters **bricks_list** (*list*) – List of bricks to be deleted.

Returns True if all the bricks are deleted. False otherwise.

Return type bool

glustolibs.gluster.brick_libs.**select_bricks_to_bring_offline**(*mnode*, *volname*)
Randomly selects bricks to bring offline without affecting the cluster

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Returns

On success returns dict. Value of each key is list of bricks to bring offline. If volume doesn't exist returns dict with value of each item being empty list. Example:

brick_to_bring_offline = { 'is_tier': False, 'hot_tier_bricks': [], 'cold_tier_bricks': [], 'volume_bricks': [] }

Return type dict

glustolibs.gluster.brick_libs.**select_volume_bricks_to_bring_offline**(*mnode*, *volname*)
Randomly selects bricks to bring offline without affecting the cluster from a non-tiered volume.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Returns

On success returns list of bricks that can be brough offline. If volume doesn't exist or is a tiered volume returns empty list

Return type list

glustolibs.gluster.brick_libs.**select_tier_volume_bricks_to_bring_offline**(*mnode*, *volname*)

Randomly selects bricks to bring offline without affecting the cluster from a tiered volume.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Returns

On success returns dict. Value of each key is list of bricks to bring offline. If volume doesn't exist or is not a tiered volume returns dict with value of each item being empty list. Example:

brick_to_bring_offline = { 'hot_tier_bricks': [], 'cold_tier_bricks': [], }

Return type dict

glustolibs.gluster.brick_libs.**select_hot_tier_bricks_to_bring_offline**(*mnode*, *volname*)

Randomly selects bricks to bring offline without affecting the cluster from a hot tier.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Returns

On success returns list of bricks that can be brough offline from hot tier. If volume doesn't exist or is a non tiered volume returns empty list.

Return type list

glustolibs.gluster.brick_libs.**select_cold_tier_bricks_to_bring_offline**(*mnode*, *volname*)

Randomly selects bricks to bring offline without affecting the cluster from a cold tier.

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Returns

On success returns list of bricks that can be brough offline from cold tier. If volume doesn't exist or is a non tiered volume returns empty list.

Return type list

glustolibs.gluster.brick_libs.**get_bricks_to_bring_offline_from_replicated_volume**(*subvols_list*, *replica_count*, *quorum_info*)

Randomly selects bricks to bring offline without affecting the cluster for a replicated volume.

Parameters

- **subvols_list** – list of subvols. It can be volume_subvols, hot_tier_subvols or cold_tier_subvols. For example:

  subvols = volume_libs.get_subvols(mnode, volname) volume_subvols = subvols_dict['volume_subvols'] hot_tier_subvols = subvols_dict['hot_tier_subvols'] cold_tier_subvols = subvols_dict['cold_tier_subvols']

- **replica_count** – Replica count of a Replicate or Distributed-Replicate volume.

- **quorum_info** – dict containing quorum info of the volume. The dict should have the following info:

  – is_quorum_applicable, quorum_type, quorum_count

  For example: quorum_dict = get_client_quorum_info(mnode, volname) volume_quorum_info = quorum_info['volume_quorum_info'] hot_tier_quorum_info = quorum_info['hot_tier_quorum_info'] cold_tier_quorum_info = quorum_info['cold_tier_quorum_info']

Returns

List of bricks that can be brought offline without affecting the cluster. On any failure return empty list.

Return type list

glustolibs.gluster.brick_libs.**get_bricks_to_bring_offline_from_disperse_volume**(*subvols_list*, *redundancy_count*)

Randomly selects bricks to bring offline without affecting the cluster for a disperse volume.

Parameters

- **subvols_list** – list of subvols. It can be volume_subvols, hot_tier_subvols or cold_tier_subvols. For example:

  subvols = volume_libs.get_subvols(mnode, volname) volume_subvols = subvols_dict['volume_subvols'] hot_tier_subvols = subvols_dict['hot_tier_subvols'] cold_tier_subvols = subvols_dict['cold_tier_subvols']

- **redundancy_count** – Redundancy count of a Disperse or Distributed-Disperse volume.

Returns

List of bricks that can be brought offline without affecting the cluster.On any failure return empty list.

Return type list

glustolibs.gluster.brick_libs.**wait_for_bricks_to_be_online**(*mnode*, *volname*, *timeout=300*)

Waits for the bricks to be online until timeout

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Kwargs: timeout (int): timeout value in seconds to wait for bricks to be online

Returns True if all bricks are online within timeout, False otherwise

glustolibs.gluster.brick_ops module

Description: Module for gluster brick operations

glustolibs.gluster.brick_ops.**add_brick**(*mnode*, *volname*, *bricks_list*, *force=False*, ***kwargs*)

Add Bricks specified in the bricks_list to the volume.

Parameters

- **mnode** (*str*) – None on which the commands are executed.
- **volname** (*str*) – Name of the volume
- **bricks_list** (*list*) – List of bricks to be added

Kwargs:

force (bool): If this option is set to True, then add brick command will get executed with force option. If it is set to False, then add brick command will get executed without force option

****kwargs**

The keys, values in kwargs are:

- replica_count : (int)|None
- arbiter_count : (int)|None

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.brick_ops.**remove_brick**(*mnode*, *volname*, *bricks_list*, *option*, *xml=False*, ***kwargs*)

Remove bricks specified in the bricks_list from the volume.

Parameters

- **mnode** (*str*) – None on which the commands are executed.
- **volname** (*str*) – Name of the volume
- **bricks_list** (*list*) – List of bricks to be removed
- **option** (*str*) – Remove brick options: <start|stop|status|commit|force>

Kwargs: xml (bool): if xml is True, get xml output of command execution. ******kwargs

The keys, values in kwargs are:

- replica_count : (int)|None

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.brick_ops.**replace_brick**(*mnode*, *volname*, *src_brick*, *dst_brick*)
Replace src brick with dst brick from the volume.

Parameters

- **mnode** (*str*) – None on which the commands are executed.
- **volname** (*str*) – Name of the volume
- **src_brick** (*str*) – Source brick name
- **dst_brick** (*str*) – Destination brick name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.gluster_base_class module

Description: Module containing GlusterBaseClass which defines all the variables necessary for tests.

class glustolibs.gluster.gluster_base_class.**runs_on**(*value*)
Bases: glusto.carteplex.CarteTestClass

Decorator providing runs_on capability for standard unittest script

class glustolibs.gluster.gluster_base_class.**GlusterBaseClass**(*methodName='runTest'*)
Bases: unittest.case.TestCase

GlusterBaseClass to be subclassed by Gluster Tests. This class reads the config for variable values that will be used in gluster tests. If variable values are not specified in the config file, the variable are defaulted to specific values.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

volume_type = None

mount_type = None

classmethod inject_msg_in_gluster_logs(*msg*)
Inject all the gluster logs on servers, clients with msg

Parameters **msg** (*str*) – Message string to be injected

Returns

True if injecting msg on the log files/dirs is successful. False Otherwise.

Return type bool

classmethod get_ip_from_hostname(*nodes*)
Returns list of IP's for the list of nodes in order.

Parameters **nodes** (*list*) – List of nodes hostnames

Returns List of IP's corresponding to the hostnames of nodes.

Return type list

classmethod validate_peers_are_connected()
Validate whether each server in the cluster is connected to all other servers in cluster.

Returns (bool): True if all peers are in connected with other peers. False otherwise.

classmethod setup_volume(*volume_create_force=False*)

Setup the volume:

- Create the volume, Start volume, Set volume

options, enable snapshot/quota/tier if specified in the config file. - Wait for volume processes to be online - Export volume as NFS/SMB share if mount_type is NFS or SMB - Log volume info and status

Parameters **volume_create_force** (*bool*) – True if create_volume should be executed with 'force' option.

Returns (bool): True if all the steps mentioned in the descriptions passes. False otherwise.

classmethod mount_volume(*mounts*)
Mount volume

Parameters **mounts** (*list*) – List of mount_objs

Returns (bool): True if mounting the volume for a mount obj is successful. False otherwise

classmethod setup_volume_and_mount_volume(*mounts*, *volume_create_force=False*)
Setup the volume and mount the volume

Parameters

- **mounts** (*list*) – List of mount_objs
- **volume_create_force** (*bool*) – True if create_volume should be executed with 'force' option.

Returns (bool): True if setting up volume and mounting the volume for a mount obj is successful. False otherwise

classmethod unmount_volume(*mounts*)
Unmount all mounts for the volume

Parameters **mounts** (*list*) – List of mount_objs

Returns (bool): True if unmounting the volume for a mount obj is successful. False otherwise

classmethod cleanup_volume()
Cleanup the volume

Returns (bool): True if cleanup volume is successful. False otherwise.

classmethod unmount_volume_and_cleanup_volume(*mounts*)
Unmount the volume and cleanup volume

Parameters **mounts** (*list*) – List of mount_objs

Returns (bool): True if unmounting the volume for the mounts and cleaning up volume is successful. False otherwise

classmethod setUpClass()
Initialize all the variables necessary for testing Gluster

setUp()

tearDown()

classmethod tearDownClass()

glustolibs.gluster.gluster_init module

Description: This file contains the methods for starting/stopping glusterd and other initial gluster environment setup helpers.

glustolibs.gluster.gluster_init.**start_glusterd**(*servers*)
Starts glusterd on specified servers if they are not running.

Parameters **servers** (*str|list*) – A server|List of server hosts on which glusterd has to be started.

Returns

True if starting glusterd is successful on all servers. False otherwise.

Return type bool

glustolibs.gluster.gluster_init.**stop_glusterd**(*servers*)
Stops the glusterd on specified servers.

Parameters **servers** (*str|list*) – A server|List of server hosts on which glusterd has to be stopped.

Returns

True if stopping glusterd is successful on all servers. False otherwise.

Return type bool

glustolibs.gluster.gluster_init.**restart_glusterd**(*servers*)
Restart the glusterd on specified servers.

Parameters **servers** (*str|list*) – A server|List of server hosts on which glusterd has to be restarted.

Returns

True if restarting glusterd is successful on all servers. False otherwise.

Return type bool

glustolibs.gluster.gluster_init.**is_glusterd_running**(*servers*)
Checks the glusterd status on specified servers.

Parameters **servers** (*str|list*) – A server|List of server hosts on which glusterd status has to be checked.

Returns if glusterd running 1 : if glusterd not running

Return type

0

-1 : if glusterd not running and PID is alive

glustolibs.gluster.gluster_init.**env_setup_servers**(*servers*)
Set up environment on all the specified servers.

Parameters **servers** (*str|list*) – A server|List of server hosts on which environment has to be setup.

Returns

True if setting up environment is successful on all servers. False otherwise.

Return type bool

glustolibs.gluster.heal_libs module

Description: Module for gluster heal related helper functions.

glustolibs.gluster.heal_libs.**is_heal_enabled**(*mnode*, *volname*)
Check if heal is enabled for a volume.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns True if heal is enabled on volume. False otherwise. NoneType: None if unable to get the volume status.

Return type bool

glustolibs.gluster.heal_libs.**is_heal_disabled**(*mnode*, *volname*)
Check if heal is disabled for a volume.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns True if heal is diabled on volume. False otherwise. NoneType: None if unable to get the volume status shd or parse error.

Return type bool

glustolibs.gluster.heal_libs.**are_all_self_heal_daemons_are_online**(*mnode*, *volname*)

Verifies whether all the self-heal-daemons are online for the specified volume.

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

True if all the self-heal-daemons are online for the volume. False otherwise.

NoneType: None if unable to get the volume status

Return type bool

glustolibs.gluster.heal_libs.**monitor_heal_completion**(*mnode*, *volname*, *time-out_period=1200*)

Monitors heal completion by looking into .glusterfs/indices/xattrop directory of every brick for certain time. When there are no entries in all the brick directories then heal is successful. Otherwise heal is pending on the volume.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume
- **heal_monitor_timeout** – time until which the heal monitoring to be done. Default: 1200 i.e 20 minutes.

Returns True if heal is complete within timeout_period. False otherwise

Return type bool

glustolibs.gluster.heal_libs.**is_heal_complete**(*mnode*, *volname*)

Verifies there are no pending heals on the volume. The 'number of entries' in the output of heal info for all the bricks should be 0 for heal to be completed.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns True if heal is complete. False otherwise

Return type bool

glustolibs.gluster.heal_libs.**is_volume_in_split_brain**(*mnode*, *volname*)

Verifies there are no split-brain on the volume. The 'number of entries' in the output of heal info split-brain for all the bricks should be 0 for volume not to be in split-brain.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns True if volume is not in split-brain. False otherwise

Return type bool

glustolibs.gluster.heal_libs.**get_unhealed_entries_info**(*volname*, *mnode=''*)

Get the information of all gfid's on which heal is pending. The information includes - stat of gfid, getfattr output for all the dirs/ files for a given gfid

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

True if getting unhealed entries info is successful. False otherwise

Return type bool

glustolibs.gluster.heal_libs.**wait_for_self_heal_daemons_to_be_online**(*mnode*, *volname*, *timeout=300*)

Waits for the volume self-heal-daemons to be online until timeout

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Kwargs: timeout (int): timeout value in seconds to wait for self-heal-daemons to be online.

Returns True if all self-heal-daemons are online within timeout, False otherwise

glustolibs.gluster.heal_ops module

Description: Module for gluster heal operations.

glustolibs.gluster.heal_ops.**trigger_heal**(*mnode*, *volname*)

Triggers heal on the volume.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns True if heal is triggered successfully. False otherwise.

Return type bool

glustolibs.gluster.heal_ops.**trigger_heal_full**(*mnode*, *volname*)

Triggers heal 'full' on the volume.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns True if heal is triggered successfully. False otherwise.

Return type bool

glustolibs.gluster.heal_ops.**enable_heal**(*mnode*, *volname*)

Enable heal by executing 'gluster volume heal enable' for the specified volume.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

True if heal is enabled on the volume. False otherwise.

Return type bool

glustolibs.gluster.heal_ops.**disable_heal**(*mnode*, *volname*)

Disable heal by executing 'gluster volume heal disable' for the specified volume.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

True if heal is disabled on the volume. False otherwise.

Return type bool

glustolibs.gluster.heal_ops.**enable_self_heal_daemon**(*mnode*, *volname*)

Enables self-heal-daemon on a volume by setting volume option 'self-heal-daemon' to value 'on'

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

True if setting self_heal_daemon option to 'on' is successful. False otherwise.

Return type bool

glustolibs.gluster.heal_ops.**disable_self_heal_daemon**(*mnode*, *volname*)

Disables self-heal-daemon on a volume by setting volume option 'self-heal-daemon' to value 'off'

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

True if setting self_heal_daemon option to 'off' is successful. False otherwise.

Return type bool

glustolibs.gluster.heal_ops.**heal_info**(*mnode*, *volname*)

Get heal info for the volume by executing: 'gluster volume heal <volname> info'

Parameters

- **mnode** – Node on which commands are executed.
- **volname** – Name of the volume

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.heal_ops.**heal_info_summary**(*mnode*, *volname*)

Get heal info summary i.e Bricks and it's corresponding number of entries, status.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.heal_ops.**heal_info_healed**(*mnode*, *volname*)

Get healed entries information for the volume by executing: 'gluster volume heal <volname> info healed'

Parameters

- **mnode** – Node on which commands are executed.
- **volname** – Name of the volume

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.heal_ops.**heal_info_heal_failed**(*mnode*, *volname*)

Get entries on which heal failed for the volume by executing: 'gluster volume heal <volname> info heal-failed'

Parameters

- **mnode** – Node on which commands are executed.
- **volname** – Name of the volume

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.heal_ops.**heal_info_split_brain**(*mnode*, *volname*)

Get entries that are in split-brain state for the volume by executing: 'gluster volume heal <volname> info split-brain'

Parameters

- **mnode** – Node on which commands are executed.
- **volname** – Name of the volume

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.heal_ops.**get_heal_info**(*mnode*, *volname*)

From the xml output of heal info command get the heal info data.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

None if parse errors. list: list of dictionaries. Each element in the list is the

heal_info data per brick.

Return type NoneType

glustolibs.gluster.heal_ops.**get_heal_info_summary**(*mnode*, *volname*)

From the xml output of heal info command get heal info summary i.e Bricks and it's corresponding number of entries, status.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

None if parse errors. dict: dict of dictionaries. brick names are the keys of the dict with

each key having brick's status, numberOfEntries info as dict. Example:

heal_info_summary_data = {

'ijk.lab.eng.xyz.com': { 'status': 'Connected' 'numberOfEntries': '11' },

'def.lab.eng.xyz.com': { 'status': 'Transport endpoint is not connected', 'numberOfEntries': '-' }

}

Return type NoneType

glustolibs.gluster.heal_ops.**get_heal_info_split_brain**(*mnode*, *volname*)

From the xml output of heal info split-brain command get the heal info split-brain data.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

None if parse errors. list: list of dictionaries. Each element in the list is the

heal_info_split_brain data per brick.

Return type NoneType

glustolibs.gluster.heal_ops.**get_heal_info_split_brain_summary**(*mnode*, *volname*)

Get heal info split_brain summary i.e Bricks and it's corresponding number of split-brain entries, status.

Parameters

- **mnode** – Node on which commands are executed
- **volname** – Name of the volume

Returns

None if parse errors. dict: dict of dictionaries. brick names are the keys of the dict with

each key having brick's status, numberOfEntries info as dict. Example:

heal_info_split_brain_summary_data = {

'ijk.lab.eng.xyz.com': { 'status': 'Connected' 'numberOfEntries': '11' },

'def.lab.eng.xyz.com': { 'status': 'Connected', 'numberOfEntries': '11' }

}

Return type NoneType

glustolibs.gluster.layout module

Module for library DHT layout class and related functions

class glustolibs.gluster.layout.**Layout**(*pathinfo*)

Bases: object

Default layout class for equal-sized bricks. Other layouts should inherit from this class and override/add where needed.

Init the layout class

Parameters **brickdirs** –

_get_layout()
Discover brickdir data and cache in instance for further use

brickdirs
list – a list of brickdirs associated with this layout

is_complete
Layout starts at zero, ends at 32-bits high, and has no holes or overlaps

has_zero_hashranges
Check brickdirs for zero hashrange

zero_hashrange_brickdirs
list – the list of zero_hashrange_brickdirs

is_balanced
Checks for balanced distribution in equal-sized bricks

glustolibs.gluster.lib_utils module

Description: Helper library for gluster modules.

glustolibs.gluster.lib_utils.**append_string_to_file**(*mnode*, *filename*, *str_to_add_in_file*, *user='root'*)

Appends the given string in the file.

Example

append_string_to_file("abc.def.com", "/var/log/messages", "test_1_string")

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **filename** (*str*) – absolute file path to append the string
- **str_to_add_in_file** (*str*) – string to be added in the file, which is used as a start and stop string for parsing the file in search_pattern_in_file().

Kwargs: user (str): username. Defaults to 'root' user.

Returns True, on success, False otherwise

glustolibs.gluster.lib_utils.**search_pattern_in_file**(*mnode*, *search_pattern*, *filename*, *start_str_to_parse*, *end_str_to_parse*)

checks if the given search pattern exists in the file in between 'start_str_to_parse' and 'end_str_to_parse' string.

Example

search_pattern = r'.*scrub.*' search_log("abc.def.com", search_pattern, "/var/log/messages",

"start_pattern", "end_pattern")

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **search_pattern** (*str*) – regex string to be matched in the file
- **filename** (*str*) – absolute file path to search given string
- **start_str_to_parse** (*str*) – this will be as start string in the file from which this method will check if the given search string is present.
- **end_str_to_parse** (*str*) – this will be as end string in the file whithin which this method will check if the given search string is present.

Returns True, if search_pattern is present in the file False, otherwise

glustolibs.gluster.lib_utils.**calculate_checksum**(*mnode*, *file_list*, *chksum_type='sha256sum'*)

This module calculates given checksum for the given file list

Example

calculate_checksum("abc.com", [file1, file2])

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **file_list** (*list*) – absolute file names for which checksum to be calculated

Kwargs:

chksum_type (str): type of the checksum algorithm. Defaults to sha256sum

Returns None if command execution fails, parse errors. dict: checksum value for each file in the given file list

Return type NoneType

glustolibs.gluster.lib_utils.**get_extended_attributes_info**(*mnode*, *file_list*, *encoding='hex'*, *attr_name=''*)

This module gets extended attribute info for the given file list

Example

get_extended_attributes_info("abc.com", [file1, file2])

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **file_list** (*list*) – absolute file names for which extended attributes to be fetched

Kwargs: encoding (str): encoding format attr_name (str): extended attribute name

Returns None if command execution fails, parse errors. dict: extended attribute for each file in the given file list

Return type NoneType

glustolibs.gluster.lib_utils.**get_pathinfo**(*mnode*, *filename*, *volname*)
This module gets filepath of the given file in gluster server.

Example

get_pathinfo(mnode, "file1", "testvol")

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **filename** (*str*) – relative path of file
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. list: file path for the given file in gluster server

Return type NoneType

glustolibs.gluster.lib_utils.**list_files**(*mnode*, *dir_path*, *parse_str=''*, *user='root'*)
This module list files from the given file path

Example

list_files("/root/dir1/")

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **dir_path** (*str*) – directory path name

Kwargs: parse_str (str): sub string of the filename to be fetched user (str): username. Defaults to 'root' user.

Returns None if command execution fails, parse errors. list: files with absolute name

Return type NoneType

glustolibs.gluster.lib_utils.**get_servers_bricks_dict**(*servers*, *servers_info*)
This module returns servers_bricks dictionary. :param servers: A server|List of servers for which we

need the list of bricks available on it.

Parameters **servers_info** (*dict*) – dict of server info of each servers

Returns

key - server value - list of bricks

Return type OrderedDict

Example

get_servers_bricks_dict(g.config['servers'], g.config['servers_info'])

glustolibs.gluster.lib_utils.**get_servers_used_bricks_dict**(*mnode*, *servers*)

This module returns servers_used_bricks dictionary. This information is fetched from gluster volume info command.

Parameters

- **servers** (*str|list*) – A server|List of servers for which we need the list of unused bricks on it.
- **mnode** (*str*) – The node on which gluster volume info command has to be executed.

Returns

key - server

value - list of used bricks or empty list(if all bricks are free)

Return type OrderedDict

Example

get_servers_used_bricks_dict(g.config['servers'][0]['host'], g.config['servers'])

glustolibs.gluster.lib_utils.**get_servers_unused_bricks_dict**(*mnode*, *servers*, *servers_info*)

This module returns servers_unused_bricks dictionary. Gets a list of unused bricks for each server by using functions, get_servers_bricks_dict() and get_servers_used_bricks_dict() :param mnode: The node on which gluster volume info command has

to be executed.

Parameters

- **servers** (*str|list*) – A server|List of servers for which we need the list of unused bricks available on it.
- **servers_info** – dict of server info of each servers

Example

get_servers_unused_bricks_dict(g.config['servers'][0]['host'], g.config['servers'], g.config['servers_info'])

glustolibs.gluster.lib_utils.**form_bricks_list**(*mnode*, *volname*, *number_of_bricks*, *servers*, *servers_info*)

Forms bricks list for create-volume/add-brick given the num_of_bricks servers and servers_info.

Parameters

- **mnode** (*str*) – The node on which the command has to be run.
- **volname** (*str*) – Volume name for which we require brick-list.

- **number_of_bricks** (*int*) – The number of bricks for which brick list has to be created.
- **servers** (*str|list*) – A server|List of servers from which the bricks needs to be selected for creating the brick list.
- **servers_info** (*dict*) – dict of server info of each servers.

Returns list - List of bricks to use with volume-create/add-brick None - if number_of_bricks is greater than unused bricks.

Example

form_bricks_path(g.config['servers'](0), "testvol", 6, g.config['servers'], g.config['servers_info'])

glustolibs.gluster.lib_utils.**is_rhel6**(*servers*)
Function to get whether the server is RHEL-6

Args: servers (str|list): A server|List of servers hosts to know the RHEL Version

Returns: bool:Returns True, if its RHEL-6 else returns false

glustolibs.gluster.lib_utils.**is_rhel7**(*servers*)
Function to get whether the server is RHEL-7

Args: servers (str|list): A server|List of servers hosts to know the RHEL Version

Returns: bool:Returns True, if its RHEL-7 else returns false

glustolibs.gluster.lib_utils.**get_disk_usage**(*mnode*, *path*, *user='root'*)
This module gets disk usage of the given path

Parameters

- **path** (*str*) – path for which disk usage to be calculated
- **conn** (*obj*) – connection object of the remote node

Kwargs: user (str): username

Returns disk usage in dict format on success None Type, on failure

Return type dict

Example

get_disk_usage("abc.com", "/mnt/glusterfs")

glustolibs.gluster.lib_utils.**get_disk_used_percent**(*mnode*, *dirname*)
Module to get disk used percent

Parameters

- **mnode** (*str*) – node on which cmd has to be executed
- **dirname** (*str*) – absolute path of directory

Returns used percent for given directory None Type, on failure

Return type str

Example

get_disk_used_percent("abc.com", "/mnt/glusterfs")

glustolibs.gluster.lib_utils.**check_if_dir_is_filled**(*mnode*, *dirname*, *percent_to_fill*, *timeout=3600*)

Module to check if the directory is filled with given percentage.

Parameters

- **mnode** (*str*) – node to check if directory is filled
- **dirname** (*str*) – absolute path of directory
- **percent_to_fill** (*int*) – percentage to fill the volume

Kwargs:

timeout (int): overall timeout value for wait till the dir fills with given percentage

Returns True, if volume is filled with given percent, False otherwise

Return type bool

Example

check_if_dir_is_filled("abc.com", "/mnt/glusterfs", 10)

glustolibs.gluster.lib_utils.**install_epel**(*servers*)

Module to install epel in rhel/centos/fedora systems.

Parameters **servers** (*str|list*) – A server|List of servers in which epel to be installed.

Returns True, if epel is installed successfully, False otherwise

Return type bool

Example

install_epel(["abc.com", "def.com"])

glustolibs.gluster.lib_utils.**inject_msg_in_logs**(*nodes*, *log_msg*, *list_of_dirs=None*, *list_of_files=None*)

Injects the message to all log files under all dirs specified on nodes.

Parameters

- **nodes** (*str|list*) – A server|List of nodes on which message has to be injected to logs
- **log_msg** (*str*) – Message to be injected
- **list_of_dirs** (*list*) – List of dirs to inject message on log files.
- **list_of_files** (*list*) – List of files to inject message.

Returns True if successfully injected msg on all log files.

Return type bool

glustolibs.gluster.mount_ops module

Description: Module for Mount operations.

class glustolibs.gluster.mount_ops.**GlusterMount**(*mount*)
Gluster Mount class

Parameters

- **mount** (*dict*) – Mount dict with mount_protocol, mountpoint, server, client, volname, options, smbuser, smbpasswd, platform, super_user as keys

  Note: smbuser, smbpasswd are applicable for windows mounts.

  client is a dict with host, super_user, platform as keys.

  platform should be specified in case of windows. By default it is assumed to be linux.

  super_user is 'root' by default. In case of windows the super_user can be the user who has all the admin previliges.

- **Example** –

  mount =

  {'mount_protocol': 'glusterfs', 'mountpoint': '/mnt/g1', 'server': 'abc.lab.eng.xyz.com', 'client': {'host': 'def.lab.eng.xyz.com'}, 'volname': 'testvoi', 'options': '' }

  mount =

  {'mount_protocol': 'nfs', 'mountpoint': '/mnt/n1', 'server': 'abc.lab.eng.xyz.com', 'client': {'host': 'def.lab.eng.xyz.com'}, 'volname': 'testvoi', 'options': ''}

  mount =

  {'mount_protocol': 'smb', 'mountpoint': '', 'server': 'abc.lab.eng.xyz.com', 'client': {

  'host': 'def.lab.eng.xyz.com', 'super_user': 'Admin' },

  'volname': 'testvoi', 'options': '', 'smbuser': 'abc', 'smbpasswd': 'def'}

Returns Instance of GlusterMount class

mount()
Mounts the volume

Parameters **instance args passed at init**(*uses*) –

Returns True on success and False on failure.

Return type bool

is_mounted()
Tests for mount on client

Parameters **instance args passed at init**(*uses*) –

Returns True on success and False on failure.

Return type bool

unmount()
Unmounts the volume

Parameters **instance args passed at init** (*uses*) –

Returns True on success and False on failure.

Return type bool

glustolibs.gluster.mount_ops.**is_mounted**(*volname*, *mpoint*, *mserver*, *mclient*, *mtype*, *user='root'*)

Check if mount exist.

Parameters

- **volname** (*str*) – Name of the volume
- **mpoint** (*str*) – Mountpoint dir
- **mserver** (*str*) – Server to which it is mounted to
- **mclient** (*str*) – Client from which it is mounted.
- **mtype** (*str*) – Mount type (glusterfs|nfs|smb|cifs)

Kwargs: user (str): Super user of the node mclient

Returns True if mounted and False otherwise.

Return type bool

glustolibs.gluster.mount_ops.**mount_volume**(*volname*, *mtype*, *mpoint*, *mserver*, *mclient*, *options=''*, *smbuser=None*, *smbpasswd=None*, *user='root'*)

Mount the gluster volume with specified options.

Parameters

- **volname** (*str*) – Name of the volume to mount.
- **mtype** (*str*) – Protocol to be used to mount.
- **mpoint** (*str*) – Mountpoint dir.
- **mserver** (*str*) – Server to mount.
- **mclient** (*str*) – Client from which it has to be mounted.

Kwargs: option (str): Options for the mount command. smbuser (str): SMB USERNAME. Used with mtype = 'cifs' smbpasswd (str): SMB PASSWD. Used with mtype = 'cifs' user (str): Super user of the node mclient

Returns

Tuple containing three elements (ret, out, err). (0, '', '') if already mounted. (1, '', '') if setup_samba_service fails in case of smb. (ret, out, err) of mount commnd execution otherwise.

Return type tuple

glustolibs.gluster.mount_ops.**umount_volume**(*mclient*, *mpoint*, *mtype=''*, *user='root'*)

Unmounts the mountpoint.

Parameters

- **mclient** (*str*) – Client from which it has to be mounted.
- **mpoint** (*str*) – Mountpoint dir.

Kwargs: mtype (str): Mounttype. Defaults to ‘’. user (str): Super user of the node mclient. Defaults to ‘root’

Returns

Tuple containing three elements (ret, out, err) as returned by umount command execution.

Return type tuple

glustolibs.gluster.mount_ops.**create_mount_objs**(*mounts*)

Creates GlusterMount class objects for the given list of mounts

Parameters

- **mounts** (*list*) – list of mounts with each element being dict having the specifics of each mount
- **Example** –

  mounts: [

  {‘protocol’: ‘glusterfs’, ’mountpoint’: ‘/mnt/g1’, ‘server’: ‘abc.lab.eng.xyz.com’, ‘client’: {‘host’: ‘def.lab.eng.xyz.com’}, ‘volname’: ‘testvoi’, ‘options’: ‘’, ‘num_of_mounts’: 2},

  {‘protocol’: ‘nfs’, ’mountpoint’: ‘/mnt/n1’, ‘server’: ‘abc.lab.eng.xyz.com’, ‘client’: {‘host’: ‘def.lab.eng.xyz.com’}, ‘volname’: ‘testvoi’, ‘options’: ‘’}

  {‘protocol’: ‘smb’, ’mountpoint’: ‘’, ‘server’: ‘abc.lab.eng.xyz.com’, ‘client’: {

  ’host’: ‘def.lab.eng.xyz.com’, ‘super_user’: ‘Admin’ },

  ’volname’: ‘testvoi’, ‘options’: ‘’, ‘smbuser’: ‘abc’, ‘smbpasswd’: ‘def’, ‘num_of_mounts’: 2}

  ]

Returns List of GlusterMount class objects.

Return type list

Example

mount_objs = create_mount_objs(mounts)

glustolibs.gluster.mount_ops.**create_mounts**(*mount_objs*)

Creates Mounts using the details as specified in the each mount obj

Parameters **mount_objs** (*list*) – list of mounts objects with each element being the GlusterMount class object

Returns

True if creating the mount for all mount_objs is successful. False otherwise.

Return type bool

Example

ret = create_mounts(create_mount_objs(mounts))

glustolibs.gluster.mount_ops.**unmount_mounts**(*mount_objs*)
Creates Mounts using the details as specified in the each mount obj

Parameters **mount_objs** (*list*) – list of mounts objects with each element being the GlusterMount class object

Returns

True if unmounting the mount for all mount_objs is successful. False otherwise.

Return type bool

Example

ret = unmount_mounts(create_mount_objs(mounts))

glustolibs.gluster.peer_ops module

Description: Library for gluster peer operations.

glustolibs.gluster.peer_ops.**peer_probe**(*mnode*, *server*)
Probe the specified server.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **server** (*str*) – Server to be peer probed.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.peer_ops.**peer_detach**(*mnode*, *server*, *force=False*)
Detach the specified server.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **server** (*str*) – Server to be peer detached.

Kwargs: force (bool): option to detach peer. Defaults to False.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.peer_ops.**peer_status**(*mnode*)
Runs 'gluster peer status' on specified node.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.peer_ops.**pool_list**(*mnode*)
Runs 'gluster pool list' command on the specified node.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.peer_ops.**peer_probe_servers**(*mnode*, *servers*, *validate=True*, *time_delay=10*)
Probe specified servers and validate whether probed servers are in cluster and connected state if validate is set to True.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **servers** (*str|list*) – A server|List of servers to be peer probed.

Kwargs:

validate (bool): True to validate if probed peer is in cluster and connected state. False otherwise. Defaults to True.

time_delay (int): time delay before validating peer status. Defaults to 10 seconds.

Returns True on success and False on failure.

Return type bool

glustolibs.gluster.peer_ops.**peer_detach_servers**(*mnode*, *servers*, *force=False*, *validate=True*, *time_delay=10*)
Detach peers and validate status of peer if validate is set to True.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **servers** (*str|list*) – A server|List of servers to be peer probed.

Kwargs:

force (bool): option to detach peer. Defaults to False.

validate (bool): True if status of the peer needs to be validated, False otherwise. Defaults to True.

time_delay (int): time delay before executing validating peer. status. Defaults to 10 seconds.

Returns True on success and False on failure.

Return type bool

glustolibs.gluster.peer_ops.**nodes_from_pool_list**(*mnode*)

Return list of nodes from the 'gluster pool list'.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Returns None if command execution fails. list: List of nodes in pool on Success, Empty list on failure.

Return type NoneType

glustolibs.gluster.peer_ops.**get_peer_status**(*mnode*)

Parse the output of command 'gluster peer status'.

Aargs: mnode (str): Node on which command has to be executed.

Returns None if command execution fails or parse errors. list: list of dicts on success.

Return type NoneType

Examples

```
>>> get_peer_status(mnode = 'abc.lab.eng.xyz.com')
[{'uuid': '77dc299a-32f7-43d8-9977-7345a344c398',
'hostname': 'ijk.lab.eng.xyz.com',
'state': '3',
'hostnames' : ['ijk.lab.eng.xyz.com'],
'connected': '1',
'stateStr': 'Peer in Cluster'},
```

{'uuid': 'b15b8337-9f8e-4ec3-8bdb-200d6a67ae12', 'hostname': 'def.lab.eng.xyz.com', 'state': '3', 'hostnames': ['def.lab.eng.xyz.com'], 'connected': '1', 'stateStr': 'Peer in Cluster'}]

glustolibs.gluster.peer_ops.**get_pool_list**(*mnode*)

Parse the output of 'gluster pool list' command.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Returns None if command execution fails, parse errors. list: list of dicts on success.

Return type NoneType

Examples

```
>>> get_pool_list(mnode = 'abc.lab.eng.xyz.com')
[{'uuid': 'a2b88b10-eba2-4f97-add2-8dc37df08b27',
'hostname': 'abc.lab.eng.xyz.com',
'state': '3',
'connected': '1',
'stateStr': 'Peer in Cluster'},
```

{'uuid': 'b15b8337-9f8e-4ec3-8bdb-200d6a67ae12', 'hostname': 'def.lab.eng.xyz.com', 'state': '3', 'hostnames': ['def.lab.eng.xyz.com'], 'connected': '1', 'stateStr': 'Peer in Cluster'}]

glustolibs.gluster.peer_ops.**is_peer_connected**(*mnode*, *servers*)
Checks whether specified peers are in cluster and 'Connected' state.

Parameters

- **mnode** (*str*) – Node from which peer probe has to be executed.
- **servers** (*str|list*) – A server|List of servers to be validated.

Returns

bool [True on success (peer in cluster and connected), False on] failure.

glustolibs.gluster.quota_ops module

Description: Library for gluster quota operations.

glustolibs.gluster.quota_ops.**enable_quota**(*mnode*, *volname*)
Enables quota on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

enable_quota("abc.xyz.com", testvol)

glustolibs.gluster.quota_ops.**disable_quota**(*mnode*, *volname*)
Disables quota on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

disable_quota("abc.xyz.com", testvol)

glustolibs.gluster.quota_ops.**is_quota_enabled**(*mnode*, *volname*)
Checks if quota is enabled on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

True, if quota is enabled False, if quota is disabled

Return type bool

Example

is_quota_enabled(mnode, testvol)

glustolibs.gluster.quota_ops.**quota_list**(*mnode*, *volname*, *path=None*)
Executes quota list command for given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs: path (str): Quota path

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

quota_list(mnode, testvol)

glustolibs.gluster.quota_ops.**set_quota_limit_usage**(*mnode*, *volname*, *path='/'*, *limit='100GB'*, *soft_limit=''*)

Sets limit-usage on the path of the specified volume to specified limit

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

path (str): path to which quota limit usage is set. Defaults to /.

limit (str): quota limit usage. defaults to 100GB soft_limit (str): quota soft limit to be set

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> set_quota_limit_usage("abc.com", "testvol")
```

glustolibs.gluster.quota_ops.**get_quota_list**(*mnode*, *volname*, *path=None*)
Parse the output of 'gluster quota list' command.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Kwargs: path (str): Quota path

Returns None if command execution fails, parse errors. dict: dict on success.

Return type NoneType

Examples

```
>>> get_quota_list('abc.lab.eng.xyz.com', "testvol")
{'/': {'used_space': '0', 'hl_exceeded': 'No', 'soft_limit_percent':
'60%', 'avail_space': '2147483648', 'soft_limit_value': '1288490188',
'sl_exceeded': 'No', 'hard_limit': '2147483648'}}
```

glustolibs.gluster.quota_ops.**set_quota_limit_objects**(*mnode*, *volname*, *path='/'*, *limit='10'*, *soft_limit=''*)

Sets limit-objects on the path of the specified volume to specified limit

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Kwargs:

path (str): path to which quota limit usage is set. Defaults to /.

limit (str): quota limit objects. defaults to 10. soft_limit (str): quota soft limit to be set

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> set_quota_limit_objects("abc.com", "testvol")
```

glustolibs.gluster.quota_ops.**quota_list_objects**(*mnode*, *volname*, *path=None*)
Executes quota list command for given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs: path (str): Quota path

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

quota_list_objects("abc.com", testvol)

glustolibs.gluster.quota_ops.**get_quota_list_objects**(*mnode*, *volname*, *path=None*)
Parse the output of 'gluster quota list-objects' command.

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs: path (str): Quota path

Returns None if command execution fails, parse errors. dict: dict of dict on success.

Return type NoneType

Examples

```
>>> get_quota_list_objects('abc.lab.eng.xyz.com', "testvol")
{'/': {'available': '7', 'hl_exceeded': 'No', 'soft_limit_percent':
'80%', 'soft_limit_value': '8', 'dir_count': '3', 'sl_exceeded':
'No', 'file_count': '0', 'hard_limit': '10'}}
```

glustolibs.gluster.quota_ops.**set_quota_alert_time**(*mnode*, *volname*, *time*)
Sets quota alert time

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name
- **time** (*str*) – quota alert time in seconds

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> set_quota_alert_time("abc.com", "testvol", <alert time>)
```

glustolibs.gluster.quota_ops.**set_quota_soft_timeout**(*mnode*, *volname*, *timeout*)
Sets quota soft timeout

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **timeout** (*str*) – quota soft limit timeout value

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> set_quota_soft_timeout("abc.com", "testvol", <timeout-value>)
```

glustolibs.gluster.quota_ops.**set_quota_hard_timeout**(*mnode*, *volname*, *timeout*)
Sets quota hard timeout

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **timeout** (*str*) – quota hard limit timeout value

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> set_quota_hard_timeout("abc.com", "testvol", <timeout-value>)
```

glustolibs.gluster.quota_ops.**set_quota_default_soft_limit**(*mnode*, *volname*, *timeout*)

Sets quota default soft limit

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **timeout** (*str*) – quota soft limit timeout value

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> set_quota_default_soft_limit("abc.com", "testvol",
                                 <timeout-value>)
```

glustolibs.gluster.quota_ops.**remove_quota**(*mnode*, *volname*, *path*)

Removes quota for the given path

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **path** (*str*) – path to which quota limit usage is set. Defaults to /.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> remove_quota("abc.com", "testvol", <path>)
```

glustolibs.gluster.quota_ops.**remove_quota_objects**(*mnode*, *volname*, *path*)

Removes quota objects for the given path

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **path** (*str*) – path to which quota limit usage is set. Defaults to /.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Examples

```
>>> remove_quota_objects("abc.com", "testvol", <path>)
```

glustolibs.gluster.rebalance_ops module

Description: Library for gluster rebalance operations.

glustolibs.gluster.rebalance_ops.**rebalance_start**(*mnode*, *volname*, *fix_layout=False*, *force=False*)

Starts rebalance on the given volume.

Example

rebalance_start("abc.com", testvol)

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

fix_layout (bool) [If this option is set to True, then rebalance] start will get execute with fix-layout option. If set to False, then rebalance start will get executed without fix-layout option

force (bool): If this option is set to True, then rebalance start will get execute with force option. If it is set to False, then rebalance start will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.rebalance_ops.**rebalance_stop**(*mnode*, *volname*)
Stops rebalance on the given volume.

Example

rebalance_stop("abc.com", testvol)

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.rebalance_ops.**rebalance_status**(*mnode*, *volname*)
Executes rebalance status on the given volume.

Example

rebalance_status("abc.com", testvol)

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.rebalance_ops.**get_rebalance_status**(*mnode*, *volname*)

Parse the output of 'gluster vol rebalance status' command for the given volume

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns

None if command execution fails, parse errors. dict: dict on success. rebalance status will be in dict format

Return type NoneType

Examples

```
>>> get_rebalance_status('abc.lab.eng.xyz.com', testvol)
{'node': [{'files': '0', 'status': '3', 'lookups': '0', 'skipped': '0',
'nodeName': 'localhost', 'failures': '0', 'runtime': '0.00', 'id':
'11336017-9561-4e88-9ac3-a94d4b403340', 'statusStr': 'completed',
'size': '0'}, {'files': '0', 'status': '1', 'lookups': '0', 'skipped':
'0', 'nodeName': '10.70.47.16', 'failures': '0', 'runtime': '0.00',
'id': 'a2b88b10-eba2-4f97-add2-8dc37df08b27', 'statusStr':
'in progress', 'size': '0'}, {'files': '0', 'status': '3',
'lookups': '0', 'skipped': '0', 'nodeName': '10.70.47.152',
'failures': '0', 'runtime': '0.00', 'id':
'b15b8337-9f8e-4ec3-8bdb-200d6a67ae12', 'statusStr': 'completed',
'size': '0'}, {'files': '0', 'status': '3', 'lookups': '0', 'skipped':
'0', 'nodeName': '10.70.46.52', 'failures': '0', 'runtime': '0.00',
'id': '77dc299a-32f7-43d8-9977-7345a344c398', 'statusStr': 'completed',
'size': '0'}], 'task-id': 'a16f99d1-e165-40e7-9960-30508506529b',
'aggregate': {'files': '0', 'status': '1', 'lookups': '0', 'skipped':
'0', 'failures': '0', 'runtime': '0.00', 'statusStr': 'in progress',
'size': '0'}, 'nodeCount': '4', 'op': '3'}
```

glustolibs.gluster.rebalance_ops.**rebalance_stop_and_get_status**(*mnode*, *volname*)

Parse the output of 'gluster vol rebalance stop' command for the given volume

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns

None if command execution fails, parse errors. dict: dict on success. rebalance status will be in dict format

Return type NoneType

Examples

```
>>> rebalance_stop_and_get_status('abc.xyz.com', testvol)
{'node': [{'files': '0', 'status': '3', 'lookups': '0', 'skipped': '0',
'nodeName': 'localhost', 'failures': '0', 'runtime': '0.00', 'id':
'11336017-9561-4e88-9ac3-a94d4b403340', 'statusStr': 'completed',
'size': '0'}, {'files': '0', 'status': '1', 'lookups': '0', 'skipped':
'0', 'nodeName': '10.70.47.16', 'failures': '0', 'runtime': '0.00',
'id': 'a2b88b10-eba2-4f97-add2-8dc37df08b27', 'statusStr':
'in progress', 'size': '0'}, {'files': '0', 'status': '3',
'lookups': '0', 'skipped': '0', 'nodeName': '10.70.47.152',
'failures': '0', 'runtime': '0.00', 'id':
'b15b8337-9f8e-4ec3-8bdb-200d6a67ae12', 'statusStr': 'completed',
'size': '0'}, {'files': '0', 'status': '3', 'lookups': '0', 'skipped':
'0', 'nodeName': '10.70.46.52', 'failures': '0', 'runtime': '0.00',
'id': '77dc299a-32f7-43d8-9977-7345a344c398', 'statusStr': 'completed',
'size': '0'}], 'task-id': 'a16f99d1-e165-40e7-9960-30508506529b',
'aggregate': {'files': '0', 'status': '1', 'lookups': '0', 'skipped':
'0', 'failures': '0', 'runtime': '0.00', 'statusStr': 'in progress',
'size': '0'}, 'nodeCount': '4', 'op': '3'}
```

glustolibs.gluster.rebalance_ops.**wait_for_rebalance_to_complete**(*mnode*, *volname*, *timeout=300*)

Waits for the rebalance to complete

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Kwargs:

timeout (int): timeout value in seconds to wait for rebalance to complete

Returns True on success, False otherwise

Examples

```
>>> wait_for_rebalance_to_complete("abc.com", "testvol")
```

glustolibs.gluster.samba_ops module

glustolibs.gluster.snap_ops module

Description: Library for gluster snapshot operations.

glustolibs.gluster.snap_ops.**snap_create**(*mnode*, *volname*, *snapname*, *timestamp=False*, *description=''*, *force=False*)

Creates snapshot for the given volume.

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.

- **volname** (*str*) – volume name
- **snapname** (*str*) – snapshot name

Kwargs:

timestamp (bool): If this option is set to True, then timestamps will get appended to the snapname. If this option is set to False, then timestamps will not be appended to snapname.

description (str): description for snapshot creation force (bool): If this option is set to True, then snap

create will get execute with force option. If it is set to False, then snap create will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_create("abc.com", testvol, testsnap)

glustolibs.gluster.snap_ops.**snap_clone**(*mnode*, *snapname*, *clonename*)
Clones the given snapshot

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **snapname** (*str*) – snapshot name to be cloned
- **clonename** (*str*) – clone name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_clone("abc.com", testsnap, clone1)

glustolibs.gluster.snap_ops.**snap_restore**(*mnode*, *snapname*)
Executes snap restore cli for the given snapshot

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **snapname** (*str*) – snapshot name to be cloned

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_restore(mnode, testsnap)

glustolibs.gluster.snap_ops.**snap_restore_complete**(*mnode*, *volname*, *snapname*)
stops the volume restore the snapshot and starts the volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name
- **snapname** (*str*) – snapshot name

Returns True on success, False on failure

Return type bool

Example

snap_restore_complete(mnode, testvol, testsnap)

glustolibs.gluster.snap_ops.**snap_status**(*mnode*, *snapname=''*, *volname=''*)
Runs 'gluster snapshot status' on specific node

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Kwargs: snapname (str): snapshot name volname (str): volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_status("abc.com")

glustolibs.gluster.snap_ops.**get_snap_status**(*mnode*)
Parse the output of 'gluster snapshot status' command.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Returns

None if command execution fails, parse errors. list: list of dict on success. Each snap status will be

in dict format

Return type NoneType

Examples

```
>>> get_snap_status('abc.lab.eng.xyz.com')
[{'volCount': '1', 'volume': {'brick': [{'path': '10.70.47.11:
testvol_brick0', 'pid': '26747', 'lvUsage': '3.52', 'volumeGroup':
'RHS_vg0', 'lvSize': '9.95g'}, {'path': '10.70.47.16:/testvol_brick1',
'pid': '25497', 'lvUsage': '3.52', 'volumeGroup': 'RHS_vg0',
'lvSize': '9.95g'}], 'brickCount': '2'}, 'name': 'snap2', 'uuid':
'56a39a92-c339-47cc-a8b2-9e54bb2a6324'}, {'volCount': '1', 'volume':
{'brick': [{'path': '10.70.47.11:testvol_next_brick0', 'pid': '26719',
'lvUsage': '4.93', 'volumeGroup': 'RHS_vg1', 'lvSize': '9.95g'}],
'brickCount': '1'}, 'name': 'next_snap1',
'uuid': 'dcf0cd31-c0db-47ad-92ec-f72af2d7b385'}]
```

glustolibs.gluster.snap_ops.**get_snap_status_by_snapname**(*mnode*, *snapname*)

Parse the output of 'gluster snapshot status' command for the given snapshot.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **snapname** (*str*) – snapshot name

Returns None if command execution fails, parse errors. dict: on success.

Return type NoneType

Examples

```
>>> get_snap_status_by_snapname('abc.lab.eng.xyz.com',
                                'snap1')
{'volCount': '1', 'volume': {'brick': [{'path': '10.70.47.11:
testvol_brick0', 'pid': '26747', 'lvUsage': '3.52', 'volumeGroup':
'RHS_vg0', 'lvSize': '9.95g'}, {'path': '10.70.47.16:/testvol_brick1',
'pid': '25497', 'lvUsage': '3.52', 'volumeGroup': 'RHS_vg0',
'lvSize': '9.95g'}], 'brickCount': '2'}, 'name': 'snap2', 'uuid':
'56a39a92-c339-47cc-a8b2-9e54bb2a6324'}
```

glustolibs.gluster.snap_ops.**get_snap_status_by_volname**(*mnode*, *volname*)

Parse the output of 'gluster snapshot status' command for the given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – snapshot name

Returns None if command execution fails, parse errors. list: list of dicts on success.

Return type NoneType

Examples

```
>>> get_snap_status_by_volname('abc.lab.eng.xyz.com',
                               'testvol')
[{'volCount': '1', 'volume': {'brick': [{'path': '10.70.47.11:
testvol_brick0', 'pid': '26747', 'lvUsage': '3.52', 'volumeGroup':
'RHS_vg0', 'lvSize': '9.95g'}, {'path': '10.70.47.16:/testvol_brick1',
'pid': '25497', 'lvUsage': '3.52', 'volumeGroup': 'RHS_vg0',
'lvSize': '9.95g'}], 'brickCount': '2'}, 'name': 'snap2', 'uuid':
'56a39a92-c339-47cc-a8b2-9e54bb2a6324'}, {'volCount': '1', 'volume':
{'brick': [{'path': '10.70.47.11:testvol_next_brick0', 'pid': '26719',
'lvUsage': '4.93', 'volumeGroup': 'RHS_vg1', 'lvSize': '9.95g'}],
'brickCount': '1'}, 'name': 'next_snap1',
'uuid': 'dcf0cd31-c0db-47ad-92ec-f72af2d7b385'}]
```

glustolibs.gluster.snap_ops.**snap_info**(*mnode*, *snapname=''*, *volname=''*)

Runs 'gluster snapshot info' on specific node

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Kwargs: snapname (str): snapshot name volname (str): volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_info("abc.com")

glustolibs.gluster.snap_ops.**get_snap_info**(*mnode*)

Parse the output of 'gluster snapshot info' command.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Returns None if command execution fails, parse errors. list: list of dicts on success.

Return type NoneType

Examples

```
>>> get_snap_info('abc.lab.eng.xyz.com')
[{'description': 'This is snap2', 'uuid':
'56a39a92-c339-47cc-a8b2-9e54bb2a6324', 'volCount': '1',
'snapVolume': {'status': 'Stopped', 'name':
'df1882d3f86d48738e69f298096f3810'}, 'createTime':
'2016-04-07 12:01:21', 'name': 'snap2'}, {'description': None,
```

```
'uuid': 'a322d93a-2732-447d-ab88-b943fa402fd2', 'volCount': '1',
'snapVolume': {'status': 'Stopped', 'name':
'2c790e6132e447e79168d9708d4abfe7'}, 'createTime':
'2016-04-07 13:59:43', 'name': 'snap1'}]
```

glustolibs.gluster.snap_ops.**get_snap_info_by_snapname**(*mnode*, *snapname*)

Parse the output of 'gluster snapshot info' command for the given snapshot.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **snapname** (*str*) – snapshot name

Returns None if command execution fails, parse errors. dict: on success.

Return type NoneType

Examples

```
>>> get_snap_info_by_snapname('abc.lab.eng.xyz.com', 'snap1')
{'description': 'This is snap2', 'uuid':
'56a39a92-c339-47cc-a8b2-9e54bb2a6324', 'volCount': '1',
'snapVolume': {'status': 'Stopped', 'name':
'df1882d3f86d48738e69f298096f3810'}
```

glustolibs.gluster.snap_ops.**get_snap_info_by_volname**(*mnode*, *volname*)

Parse the output of 'gluster snapshot info' command for the given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – snapshot name

Returns None if command execution fails, parse errors. list: list of dicts on success.

Return type NoneType

Examples

```
>>> get_snap_info_by_volname('abc.lab.eng.xyz.com',
                             'testvol')
{'originVolume': {'snapCount': '1', 'name': 'testvol',
'snapRemaining': '255'}, 'count': '1', 'snapshots':
[{'description': 'This is next snap1', 'uuid':
'dcf0cd31-c0db-47ad-92ec-f72af2d7b385', 'volCount': '1',
'snapVolume': {'status': 'Stopped', 'name':
'49c290d6e8b74205adb3cce1206b5bc5'}, 'createTime':
'2016-04-07 12:03:11', 'name': 'next_snap1'}]}
```

glustolibs.gluster.snap_ops.**snap_list**(*mnode*)

Lists the snapshots

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_list("abc.com")

glustolibs.gluster.snap_ops.**get_snap_list**(*mnode*)
Parse the output of 'gluster snapshot list' command.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Returns None if command execution fails, parse errors. list: list of snapshots on success.

Return type NoneType

Examples

```
>>> get_snap_list('abc.lab.eng.xyz.com')
['snap1', 'snap2']
```

glustolibs.gluster.snap_ops.**snap_config**(*mnode*, *volname=None*)
Runs 'gluster snapshot config' on specific node

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Kwargs: volname (str): volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_config(mnode)

glustolibs.gluster.snap_ops.**get_snap_config**(*mnode*, *volname=None*)
Parse the output of 'gluster snapshot config' command.

Parameters **mnode** (*str*) – Node on which command has to be executed.

Kwargs: volname (str): volume name

Returns None if command execution fails, parse errors. dict: on success.

Return type NoneType

Examples

```
>>> get_snap_config('abc.com')
{'volumeConfig': [{'softLimit': '230', 'effectiveHardLimit': '256',
'name': 'testvol', 'hardLimit': '256'}, {'softLimit': '230',
'effectiveHardLimit': '256', 'name': 'testvol_next',
'hardLimit': '256'}], 'systemConfig': {'softLimit': '90%',
'activateOnCreate': 'disable', 'hardLimit': '256',
'autoDelete': 'disable'}}
```

glustolibs.gluster.snap_ops.**set_snap_config**(*mnode*, *option*, *volname=None*)
Sets given snap config on the given node

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **option** (*dict*) – dict of single snap config option

Kwargs: volname (str): volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

>>>option={'snap-max-hard-limit':'200'} set_snap_config(mnode, option)

glustolibs.gluster.snap_ops.**snap_delete**(*mnode*, *snapname*)
Deletes the given snapshot

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **snapname** (*str*) – snapshot name to be deleted

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_delete(mnode, testsnap)

glustolibs.gluster.snap_ops.**snap_delete_by_volumename**(*mnode*, *volname*)
Deletes the given snapshot

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_delete_by_volumename(mnode, testvol)

glustolibs.gluster.snap_ops.**snap_delete_all**(*mnode*)
Deletes all the snapshot in the cluster

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_delete_all("abc.com")

glustolibs.gluster.snap_ops.**snap_activate**(*mnode*, *snapname*, *force=False*)
Activates the given snapshot

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **snapname** (*str*) – snapshot name to be cloned

Kwargs:

force (bool): If this option is set to True, then snap activate will get execute with force option. If it is set to False, then snap activate will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_activate("abc.com", testsnap)

glustolibs.gluster.snap_ops.**snap_deactivate**(*mnode*, *snapname*)

Deactivates the given snapshot

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **snapname** (*str*) – snapshot name to be cloned

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

snap_deactivate("abc.com", testsnap)

glustolibs.gluster.tiering_ops module

Description: Library for gluster tiering operations.

glustolibs.gluster.tiering_ops.**add_extra_servers_to_cluster**(*mnode*, *extra_servers*)

Adds the given extra servers to cluster

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **extra_servers** (*str|list*) – A serverllist of extra servers to be attached to cluster

Returns

True, if extra servers are attached to cluster False, otherwise

Return type bool

Example

add_extra_servers_to_cluster("abc.com", ['peer_node1','peer_node2'])

glustolibs.gluster.tiering_ops.**tier_attach**(*mnode*, *volname*, *num_bricks_to_add*, *extra_servers*, *extra_servers_info*, *replica=1*, *force=False*)

Attaches tier to the volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name
- **num_bricks_to_add** (*str*) – number of bricks to be added as hot tier
- **extra_servers** (*str|list*) – from this server|these servers, hot tier will be added to volume
- **extra_servers_info** (*dict*) – dict of server info of each extra servers

Kwargs: replica (str): replica count of the hot tier force (bool): If this option is set to True, then attach tier will get executed with force option. If it is set to False, then attach tier will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_attach("abc.com", testvol, '2', ['extra_server1','extra_server2'], extra_server_info)

glustolibs.gluster.tiering_ops.**tier_start**(*mnode*, *volname*, *force=False*)

Starts the tier volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

force (bool): If this option is set to True, then attach tier will get executed with force option. If it is set to False, then attach tier will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_start("abc.com", testvol)

glustolibs.gluster.tiering_ops.**tier_status**(*mnode*, *volname*)
executes tier status command

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_status("abc.com", testvol)

glustolibs.gluster.tiering_ops.**get_tier_status**(*mnode*, *volname*)
Parse the output of 'gluster tier status' command.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. dict: dict on success.

Return type NoneType

Examples

```
>>> get_tier_status('abc.lab.eng.xyz.com', 'testvol')
{'node': [{'promotedFiles': '0', 'demotedFiles': '0', 'nodeName':
'localhost', 'statusStr': 'in progress'}, {'promotedFiles': '0',
'demotedFiles': '0', 'nodeName': '10.70.47.16', 'statusStr':
'in progress'}], 'task-id': '2ed28cbd-4246-493a-87b8-1fdcce313b34',
'nodeCount': '4', 'op': '7'}
```

glustolibs.gluster.tiering_ops.**tier_detach_start**(*mnode*, *volname*)
starts detaching tier on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.

- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_detach_start("abc.com", testvol)

glustolibs.gluster.tiering_ops.**tier_detach_status**(*mnode*, *volname*)
executes detach tier status on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_detach_status("abc.com", testvol)

glustolibs.gluster.tiering_ops.**tier_detach_stop**(*mnode*, *volname*)
stops detaching tier on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_detach_stop("abc.com", testvol)

glustolibs.gluster.tiering_ops.**tier_detach_commit**(*mnode*, *volname*)
commits detach tier on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_detach_commit("abc.com", testvol)

glustolibs.gluster.tiering_ops.**tier_detach_force**(*mnode*, *volname*)
detaches tier forcefully on given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

tier_detach_force("abc.com", testvol)

glustolibs.gluster.tiering_ops.**get_detach_tier_status**(*mnode*, *volname*)
Parse the output of 'gluster volume tier detach status' command.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. dict: dict on success.

Return type NoneType

Examples

```
>>> get_detach_tier_status('abc.lab.eng.xyz.com', "testvol")
{'node': [{'files': '0', 'status': '3', 'lookups': '1', 'skipped': '0',
'nodeName': 'localhost', 'failures': '0', 'runtime': '0.00', 'id':
'11336017-9561-4e88-9ac3-a94d4b403340', 'statusStr': 'completed',
'size': '0'}, {'files': '0', 'status': '3', 'lookups': '0', 'skipped':
'0', 'nodeName': '10.70.47.16', 'failures': '0', 'runtime': '0.00',
'id': 'a2b88b10-eba2-4f97-add2-8dc37df08b27', 'statusStr': 'completed',
'size': '0'}], 'nodeCount': '4', 'aggregate': {'files': '0', 'status':
'3', 'lookups': '1', 'skipped': '0', 'failures': '0', 'runtime': '0.0',
'statusStr': 'completed', 'size': '0'}}
```

glustolibs.gluster.tiering_ops.**tier_detach_start_and_get_taskid**(*mnode*, *volname*)

Parse the output of 'gluster volume tier detach start' command.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. dict: dict on success.

Return type NoneType

Examples

```
>>> tier_detach_start_and_get_taskid('abc.lab.eng.xyz.com',
                                     "testvol")
{'task-id': '8020835c-ff0d-4ea1-9f07-62dd067e92d4'}
```

glustolibs.gluster.tiering_ops.**tier_detach_stop_and_get_status**(*mnode*, *volname*)

Parse the output of 'gluster volume tier detach stop' command.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. dict: dict on success.

Return type NoneType

Examples

```
>>> tier_detach_stop_and_get_status('abc.lab.eng.xyz.com',
                                    "testvol")
{'node': [{'files': '0', 'status': '3', 'lookups': '1', 'skipped': '0',
'nodeName': 'localhost', 'failures': '0', 'runtime': '0.00', 'id':
'11336017-9561-4e88-9ac3-a94d4b403340', 'statusStr': 'completed',
'size': '0'}, {'files': '0', 'status': '3', 'lookups': '0', 'skipped':
'0', 'nodeName': '10.70.47.16', 'failures': '0', 'runtime': '0.00',
'id': 'a2b88b12-eba2-4f97-add2-8dc37df08b27', 'statusStr': 'completed',
'size': '0'}], 'nodeCount': '4', 'aggregate': {'files': '0', 'status':
```

```
'3', 'lookups': '1', 'skipped': '0', 'failures': '0', 'runtime': '0.0',
'statusStr': 'completed', 'size': '0'}}
```

glustolibs.gluster.tiering_ops.**wait_for_detach_tier_to_complete**(*mnode*, *volname*, *timeout=300*)

Waits for the detach tier to complete

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Kwargs: timeout (int): timeout value to wait for detach tier to complete

Returns True on success, False otherwise

Examples

```
>>> wait_for_detach_tier_to_complete(mnode, "testvol")
```

glustolibs.gluster.tiering_ops.**get_files_from_hot_tier**(*mnode*, *volname*)

Lists files from hot tier for the given volume

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns if there are no files in hot tier. list: list of files in hot tier on success.

Return type Emptylist

Examples

>>>get_files_from_hot_tier(mnode, "testvol")

glustolibs.gluster.tiering_ops.**get_files_from_cold_tier**(*mnode*, *volname*)

Lists files from cold tier for the given volume

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns if there are no files in cold tier. list: list of files in cold tier on success.

Return type Emptylist

Examples

>>>get_files_from_hot_tier("testvol")

glustolibs.gluster.tiering_ops.**get_tier_promote_frequency**(*mnode*, *volname*)

Gets tier promote frequency value for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: promote frequency value on success.

Return type NoneType

Examples

>>>get_tier_promote_frequency("abc.com", "testvol")

glustolibs.gluster.tiering_ops.**get_tier_demote_frequency**(*mnode*, *volname*)
Gets tier demote frequency value for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: demote frequency value on success.

Return type NoneType

Examples

>>>get_tier_demote_frequency("abc.com", "testvol")

glustolibs.gluster.tiering_ops.**get_tier_mode**(*mnode*, *volname*)
Gets tier mode for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: tier mode on success.

Return type NoneType

Examples

>>>get_tier_mode("testvol")

glustolibs.gluster.tiering_ops.**get_tier_max_mb**(*mnode*, *volname*)
Gets tier max mb for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: tier max mb on success.

Return type NoneType

Examples

>>>get_tier_max_mb("abc.com", "testvol")

glustolibs.gluster.tiering_ops.**get_tier_max_files**(*mnode*, *volname*)
Gets tier max files for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: tier max files on success.

Return type NoneType

Examples

>>>get_tier_max_files("abc.com", "testvol")

glustolibs.gluster.tiering_ops.**get_tier_watermark_high_limit**(*mnode*, *volname*)
Gets tier watermark high limit for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: tier watermark high limit on success.

Return type NoneType

Examples

>>>get_tier_watermark_high_limit(mnode, "testvol")

glustolibs.gluster.tiering_ops.**get_tier_watermark_low_limit**(*mnode*, *volname*)
Gets tier watermark low limit for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: tier watermark low limit on success.

Return type NoneType

Examples

>>>get_tier_watermark_low_limit("abc.com", "testvol")

glustolibs.gluster.tiering_ops.**set_tier_promote_frequency**(*mnode*, *volname*, *value*)
Sets tier promote frequency value for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **value** (*str*) – promote frequency value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_promote_frequency("abc.com", "testvol", '1000')

glustolibs.gluster.tiering_ops.**set_tier_demote_frequency**(*mnode*, *volname*, *value*)
Sets tier demote frequency value for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **value** (*str*) – demote frequency value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_demote_frequency("abc.com", "testvol", "500")

glustolibs.gluster.tiering_ops.**set_tier_mode**(*mnode*, *volname*, *value*)
Sets tier mode for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_mode("abc.com", "testvol", "cache")

glustolibs.gluster.tiering_ops.**set_tier_max_mb**(*mnode*, *volname*, *value*)
Sets tier max mb for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_max_mb("abc.com", "testvol", "50")

glustolibs.gluster.tiering_ops.**set_tier_max_files**(*mnode*, *volname*, *value*)

Sets tier max files for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_max_files("abc.com", "testvol", "10")

glustolibs.gluster.tiering_ops.**set_tier_watermark_high_limit**(*mnode*, *volname*, *value*)

Sets tier watermark high limit for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_watermark_high_limit("abc.com", "testvol", "95")

glustolibs.gluster.tiering_ops.**set_tier_watermark_low_limit**(*mnode*, *volname*, *value*)

Sets tier watermark low limit for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_watermark_low_limit("abc.com", "testvol", "40")

glustolibs.gluster.tiering_ops.**get_tier_pid**(*mnode*, *volname*)
Gets tier pid for given volume.

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns None if command execution fails, parse errors. str: pid of tier process on success.

Return type NoneType

Examples

>>>get_tier_pid("abc.xyz.com", "testvol")

glustolibs.gluster.tiering_ops.**is_tier_process_running**(*mnode*, *volname*)
Checks whether tier process is running

Parameters

- **mnode** (*str*) – Node on which command has to be executed.
- **volname** (*str*) – volume name

Returns True on success, False otherwise

Examples

>>>is_tier_process_running("abc.xyz.com", "testvol")

glustolibs.gluster.uss_ops module

Description: Module for gluster uss operations

glustolibs.gluster.uss_ops.**enable_uss**(*mnode*, *volname*)
Enables uss on the specified volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.uss_ops.**disable_uss**(*mnode*, *volname*)
Disables uss on the specified volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

glustolibs.gluster.uss_ops.**is_uss_enabled**(*mnode*, *volname*)
Check if uss is Enabled on the specified volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns True if successfully enabled uss on the volume. False otherwise.

Return type bool

glustolibs.gluster.uss_ops.**is_uss_disabled**(*mnode*, *volname*)
Check if uss is disabled on the specified volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

True if successfully disabled uss on the volume. False otherwise.

Return type bool

glustolibs.gluster.volume_libs module

Description: Module for gluster volume related helper functions.

glustolibs.gluster.volume_libs.**volume_exists**(*mnode*, *volname*)
Check if volume already exists

Parameters

- **mnode** (*str*) – Node on which commands has to be executed
- **volname** (*str*) – Name of the volume.

Returns If there are errors bool : True if volume exists. False Otherwise

Return type NoneType

glustolibs.gluster.volume_libs.**setup_volume**(*mnode*, *all_servers_info*, *volume_config*, *force=False*)

Setup Volume with the configuration defined in volume_config

Parameters

- **mnode** (*str*) – Node on which commands has to be executed
- **all_servers_info** (*dict*) – Information about all servers.
- **example** –

  all_servers_info = {

  'abc.lab.eng.xyz.com': { 'host': 'abc.lab.eng.xyz.com', 'brick_root': '/bricks', 'devices': ['/dev/vdb', '/dev/vdc', '/dev/vdd', '/dev/vde'] },

  'def.lab.eng.xyz.com':{ 'host': 'def.lab.eng.xyz.com', 'brick_root': '/bricks', 'devices': ['/dev/vdb', '/dev/vdc', '/dev/vdd', '/dev/vde'] }

  }
- **volume_config** (*dict*) – Dict containing volume information
- **example** –

  volume_config = { 'name': 'testvol', 'servers': ['server-vm1', 'server-vm2', 'server-vm3',

  'server-vm4'],

  'voltype': {'type': 'distributed', 'dist_count': 4, 'transport': 'tcp'},

  'extra_servers': ['server-vm9', 'server-vm10', 'server-vm11', 'server-vm12'],

  'quota': {'limit_usage': {'path': '/', 'percent': None,

  'size': '100GB'},

  'enable': False},

  'uss': {'enable': False}, 'tier': {'create_tier': True,

  'tier_type': {'type': 'distributed-replicated', 'replica_count': 2, 'dist_count': 2, 'transport': 'tcp'}},

  'options': {'performance.readdir-ahead': True} }

Returns True on successful setup. False Otherwise

Return type bool

glustolibs.gluster.volume_libs.**cleanup_volume**(*mnode*, *volname*)

deletes snapshots in the volume, stops and deletes the gluster volume if given volume exists in gluster and deletes the directories in the bricks associated with the given volume

Parameters

- **volname** (*str*) – volume name
- **mnode** (*str*) – Node on which cmd has to be executed.

Returns

True, if volume is deleted successfully False, otherwise

Return type bool

Example

cleanup_volume("abc.xyz.com", "testvol")

glustolibs.gluster.volume_libs.**is_volume_exported**(*mnode*, *volname*, *share_type*)
Checks whether the volume is exported as nfs or cifs share

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name
- **share_type** (*str*) – nfs or cifs

Returns If volume is exported returns True. False Otherwise.

Return type bool

glustolibs.gluster.volume_libs.**log_volume_info_and_status**(*mnode*, *volname*)
Logs volume info and status

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Returns True if getting volume info and status is successful. False Otherwise.

Return type bool

glustolibs.gluster.volume_libs.**verify_all_process_of_volume_are_online**(*mnode*, *volname*)
Verifies whether all the processes of volume are online

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

Returns True if all the processes of volume are online. False Otherwise.

Return type bool

glustolibs.gluster.volume_libs.**get_subvols**(*mnode*, *volname*)
Gets the subvolumes in the given volume

Parameters

- **volname** (*str*) – volume name
- **mnode** (*str*) – Node on which cmd has to be executed.

Returns

with empty list values for all keys, if volume doesn't exist dict: Dictionary of subvols, value of each key is list of lists

containing subvols

Return type dict

Example

get_subvols("abc.xyz.com", "testvol")

glustolibs.gluster.volume_libs.**is_tiered_volume**(*mnode*, *volname*)
Check if volume is tiered volume.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns True if the volume is tiered volume. False otherwise NoneType: None if volume doesnot exist.

Return type bool

glustolibs.gluster.volume_libs.**is_distribute_volume**(*mnode*, *volname*)
Check if volume is a plain distributed volume

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns True if the volume is distributed volume. False otherwise NoneType: None if volume doesnot exist.

Return type bool

glustolibs.gluster.volume_libs.**get_volume_type_info**(*mnode*, *volname*)
Returns volume type information for the specified volume.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Retunrs:

dict [Dict containing the keys, values defining the volume type:]

Example:

volume_type_info = { 'is_tier': False, 'hot_tier_type_info': {}, 'cold_tier_type_info': {}, 'volume_type_info': {

'typeStr': 'Disperse', 'replicaCount': '1', 'arbiterCount': '0', 'stripeCount': '1', 'disperseCount': '3', 'redundancyCount': '1' }

}

volume_type_info = { 'is_tier': True, 'hot_tier_type_info': {

'hotBrickType': 'Distribute', 'hotreplicaCount': '1' },

'cold_tier_type_info': { 'coldBrickType': 'Disperse', 'coldreplicaCount': '1', 'coldarbiterCount': '0', 'colddisperseCount': '3', 'numberOfBricks':1 },

'volume_type_info': {}

NoneType: None if volume does not exist or any other key errors.

glustolibs.gluster.volume_libs.**get_cold_tier_type_info**(*mnode*, *volname*)
Returns cold tier type information for the specified volume.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Retunrs:

dict [Dict containing the keys, values defining the cold tier type:]

Example:

cold_tier_type_info = { 'coldBrickType': 'Disperse', 'coldreplicaCount': '1', 'coldarbiterCount': '0', 'colddisperseCount': '3', 'numberOfBricks': '3' }

NoneType: None if volume does not exist or is not a tiered volume or any other key errors.

glustolibs.gluster.volume_libs.**get_hot_tier_type_info**(*mnode*, *volname*)
Returns hot tier type information for the specified volume.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Retunrs:

dict [Dict containing the keys, values defining the hot tier type:]

Example:

hot_tier_type_info = { 'hotBrickType': 'Distribute', 'hotreplicaCount': '1' }

NoneType: None if volume does not exist or is not a tiered volume or any other key errors.

glustolibs.gluster.volume_libs.**get_num_of_bricks_per_subvol**(*mnode*, *volname*)
Returns number of bricks per subvol

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns

Dict containing the keys, values defining number of bricks per subvol Example:

num_of_bricks_per_subvol = { 'is_tier': False, 'hot_tier_num_of_bricks_per_subvol': None, 'cold_tier_num_of_bricks_per_subvol': None, 'volume_num_of_bricks_per_subvol': 2 }

num_of_bricks_per_subvol = { 'is_tier': True, 'hot_tier_num_of_bricks_per_subvol': 3, 'cold_tier_num_of_bricks_per_subvol': 2, 'volume_num_of_bricks_per_subvol': None }

NoneType: None if volume doesnot exist or is a tiered volume.

Return type dict

glustolibs.gluster.volume_libs.**get_cold_tier_num_of_bricks_per_subvol**(*mnode*, *volname*)

Returns number of bricks per subvol in cold tier

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns Number of bricks per subvol on cold tier. NoneType: None if volume doesnot exist or not a tiered volume.

Return type int

glustolibs.gluster.volume_libs.**get_hot_tier_num_of_bricks_per_subvol**(*mnode*, *volname*)

Returns number of bricks per subvol in hot tier

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns Number of bricks per subvol on hot tier. NoneType: None if volume doesnot exist or not a tiered volume.

Return type int

glustolibs.gluster.volume_libs.**get_replica_count**(*mnode*, *volname*)

Get the replica count of the volume

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns

Dict contain keys, values defining Replica count of the volume.

Example:

replica_count_info = { 'is_tier': False, 'hot_tier_replica_count': None, 'cold_tier_replica_count': None, 'volume_replica_count': 3 }

replica_count_info = { 'is_tier': True, 'hot_tier_replica_count': 2, 'cold_tier_replica_count': 3, 'volume_replica_count': None }

NoneType: None if it is parse failure.

Return type dict

glustolibs.gluster.volume_libs.**get_cold_tier_replica_count**(*mnode*, *volname*)

Get the replica count of cold tier.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns Replica count of the cold tier. NoneType: None if volume does not exist or not a tiered volume.

Return type int

glustolibs.gluster.volume_libs.**get_hot_tier_replica_count**(*mnode*, *volname*)
Get the replica count of hot tier.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns Replica count of the hot tier. NoneType: None if volume does not exist or not a tiered volume.

Return type int

glustolibs.gluster.volume_libs.**get_disperse_count**(*mnode*, *volname*)
Get the disperse count of the volume

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns

Dict contain keys, values defining Disperse count of the volume.

Example:

disperse_count_info = { 'is_tier': False, 'cold_tier_disperse_count': None, 'volume_disperse_count': 3 }

disperse_count_info = { 'is_tier': True, 'cold_tier_disperse_count': 3, 'volume_disperse_count': None }

None: If it is non dispersed volume.

Return type dict

glustolibs.gluster.volume_libs.**get_cold_tier_disperse_count**(*mnode*, *volname*)
Get the disperse count of cold tier.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns disperse count of the cold tier. NoneType: None if volume does not exist or not a tiered volume.

Return type int

glustolibs.gluster.volume_libs.**enable_and_validate_volume_options**(*mnode*, *volname*, *volume_options_list*, *time_delay=5*)
Enable the volume option and validate whether the option has be successfully enabled or not

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

- **volume_options_list** (*str|list*) – A volume option|List of volume options to be enabled
- **time_delay** (*int*) – Time delay between 2 volume set operations

Returns

True when enabling and validating all volume options is successful. False otherwise

Return type bool

glustolibs.gluster.volume_libs.**expand_volume**(*mnode*, *volname*, *servers*, *all_servers_info*, *force=False*, *add_to_hot_tier=False*, ***kwargs*)

Forms list of bricks to add and adds those bricks to the volume.

Parameters

- **mnode** (*str*) – Node on which commands has to be executed
- **volname** (*str*) – volume name
- **servers** (*str|list*) – A server|List of servers in the storage pool.
- **all_servers_info** (*dict*) – Information about all servers.
- **example** –

  all_servers_info = {

  'abc.lab.eng.xyz.com': { 'host': 'abc.lab.eng.xyz.com', 'brick_root': '/bricks', 'devices': ['/dev/vdb', '/dev/vdc', '/dev/vdd', '/dev/vde'] },

  'def.lab.eng.xyz.com':{ 'host': 'def.lab.eng.xyz.com', 'brick_root': '/bricks', 'devices': ['/dev/vdb', '/dev/vdc', '/dev/vdd', '/dev/vde'] }

  }

Kwargs:

force (bool): If this option is set to True, then add-brick command will get executed with force option. If it is set to False, then add-brick command will get executed without force option

add_to_hot_tier (bool): True If bricks are to be added to hot_tier. False otherwise. Defaults to False.

****kwargs**

The keys, values in kwargs are:

- replica_count : (int)|None
- arbiter_count : (int)|None
- distribute_count: (int)|None

Returns

True of expanding volumes is successful. False otherwise.

Return type bool

NOTE: adding bricks to hot tier is yet to be added in this function.

glustolibs.gluster.volume_libs.**shrink_volume**(*mnode*, *volname*, *subvol_num=None*, *replica_num=None*, *force=False*, *rebalance_timeout=300*, *delete_bricks=True*, *remove_from_hot_tier=False*, ***kwargs*)

Remove bricks from the volume.

Parameters

- **mnode** (*str*) – Node on which commands has to be executed
- **volname** (*str*) – volume name

Kwargs:

subvol_num (list): List of sub volumes number to remove. For example: If subvol_num = [2, 5], Then we will be removing bricks from 2nd and 5th sub-volume of the given volume. The sub-volume number starts from 0.

replica_num (int): Specify which replica brick to remove. If replica_num = 0, then 1st brick from each subvolume is removed. the replica_num starts from 0.

force (bool): If this option is set to True, then remove-brick command will get executed with force option. If it is set to False, then remove-brick is executed with 'start' and 'commit' when the remove-brick 'status' becomes completed.

rebalance_timeout (int): Wait time for remove-brick to complete in seconds. Default is 5 minutes.

delete_bricks (bool): After remove-brick delete the removed bricks.

remove_from_hot_tier (bool): True If bricks are to be removed from hot_tier. False otherwise. Defaults to False.

****kwargs**

The keys, values in kwargs are:

- **replica_count** [(int)|None. Specify the replica count to] reduce
- **distribute_count: (int)|None. Specify the distribute count to** reduce.

Returns

True if removing bricks from the volume is successful. False otherwise.

Return type bool

NOTE: remove-bricks from hot-tier is yet to be added in this function.

glustolibs.gluster.volume_libs.**replace_brick_from_volume**(*mnode*, *volname*, *servers*, *all_servers_info*, *src_brick=None*, *dst_brick=None*, *delete_brick=True*, *replace_brick_from_hot_tier=False*)

Replace faulty brick from the volume.

Parameters

- **mnode** (*str*) – Node on which commands has to be executed
- **volname** (*str*) – volume name
- **servers** (*str|list*) – A server|List of servers in the storage pool.
- **all_servers_info** (*dict*) – Information about all servers.

- **example** –

  all_servers_info = {

  'abc.lab.eng.xyz.com': { 'host': 'abc.lab.eng.xyz.com', 'brick_root': '/bricks', 'devices': ['/dev/vdb', '/dev/vdc', '/dev/vdd', '/dev/vde'] },

  'def.lab.eng.xyz.com':{ 'host': 'def.lab.eng.xyz.com', 'brick_root': '/bricks', 'devices': ['/dev/vdb', '/dev/vdc', '/dev/vdd', '/dev/vde'] }

  }

Kwargs: src_brick (str): Faulty brick which needs to be replaced

dst_brick (str): New brick to replace the faulty brick

delete_bricks (bool): After remove-brick delete the removed bricks.

replace_brick_from_hot_tier (bool): True If brick are to be replaced from hot_tier. False otherwise. Defaults to False.

Returns

True if replacing brick from the volume is successful. False otherwise.

Return type bool

glustolibs.gluster.volume_libs.**get_client_quorum_info**(*mnode*, *volname*)

Get the client quorum information. i.e the quorum type, quorum count.

Parameters

- **mnode** (*str*) – Node on which commands are executed.
- **volname** (*str*) – Name of the volume.

Returns

client quorum information for the volume.

client_quorum_dict = { 'is_tier': False, 'hot_tier_quorum_info':{

'is_quorum_applicable': False, 'quorum_type': None, 'quorum_count': None },

'cold_tier_quorum_info':{ 'is_quorum_applicable': False, 'quorum_type': None, 'quorum_count': None },

'volume_quorum_info':{ 'is_quorum_applicable': False, 'quorum_type': None, 'quorum_count': None }

} NoneType: None if volume doesnot exist.

Return type dict

glustolibs.gluster.volume_libs.**wait_for_volume_process_to_be_online**(*mnode*, *volname*, *timeout=300*)

Waits for the volume's processes to be online until timeout

Parameters

- **mnode** (*str*) – Node on which commands will be executed.
- **volname** (*str*) – Name of the volume.

Kwargs: timeout (int): timeout value in seconds to wait for all volume processes to be online.

Returns True if the volume's processes are online within timeout, False otherwise

glustolibs.gluster.volume_ops module

glustolibs.gluster.volume_ops.**volume_create**(*mnode*, *volname*, *bricks_list*, *force=False*, ***kwargs*)

Create the gluster volume with specified configuration

Parameters

- **mnode** (*str*) – server on which command has to be executed
- **volname** (*str*) – volume name that has to be created
- **bricks_list** (*list*) – List of bricks to use for creating volume. .. rubric:: Example

  from glustolibs.gluster.lib_utils import form_bricks_list bricks_list = form_bricks_list(mnode, volname, num_of_bricks,

  servers, servers_info)

Kwargs:

force (bool): If this option is set to True, then create volume will get executed with force option. If it is set to False, then create volume will get executed without force option

****kwargs**

The keys, values in kwargs are:

- replica_count : (int)|None
- arbiter_count : (int)|None
- stripe_count : (int)|None
- disperse_count : (int)|None
- disperse_data_count : (int)|None
- redundancy_count : (int)|None
- transport_type : tcp|rdma|tcp,rdma|None
- ...

Returns

Tuple containing three elements (ret, out, err).

The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

(-1, ‘’, ‘’): If not enough bricks are available to create volume. (ret, out, err): As returned by volume create command execution.

Return type tuple

Example

volume_create(mnode, volname, bricks_list)

glustolibs.gluster.volume_ops.**volume_start**(*mnode*, *volname*, *force=False*)
Starts the gluster volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

force (bool): If this option is set to True, then start volume will get executed with force option. If it is set to False, then start volume will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element ‘ret’ is of type ‘int’ and is the return value of command execution.

The second element ‘out’ is of type ‘str’ and is the stdout value of the command execution.

The third element ‘err’ is of type ‘str’ and is the stderr value of the command execution.

Return type tuple

Example

volume_start(“testvol”)

glustolibs.gluster.volume_ops.**volume_stop**(*mnode*, *volname*, *force=False*)
Stops the gluster volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

force (bool): If this option is set to True, then stop volume will get executed with force option. If it is set to False, then stop volume will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element ‘ret’ is of type ‘int’ and is the return value of command execution.

The second element ‘out’ is of type ‘str’ and is the stdout value of the command execution.

The third element ‘err’ is of type ‘str’ and is the stderr value of the command execution.

Return type tuple

Example

volume_stop(mnode, "testvol")

glustolibs.gluster.volume_ops.**volume_delete**(*mnode*, *volname*)

Deletes the gluster volume if given volume exists in gluster and deletes the directories in the bricks associated with the given volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Returns

True, if volume is deleted False, otherwise

Return type bool

Example

volume_delete("abc.xyz.com", "testvol")

glustolibs.gluster.volume_ops.**volume_reset**(*mnode*, *volname*, *force=False*)

Resets the gluster volume

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

force (bool): If this option is set to True, then reset volume will get executed with force option. If it is set to False, then reset volume will get executed without force option

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

volume_reset("abc.xyz.com", "testvol")

glustolibs.gluster.volume_ops.**volume_status**(*mnode*, *volname='all'*, *service=''*, *options=''*)

Executes gluster volume status cli command

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.

- **volname** (*str*) – volume name

Kwargs: volname (str): volume name. Defaults to 'all' service (str): name of the service to get status.

serivce can be, [nfs|shd|<BRICK>|quotad]], If not given, the function returns all the services

options (str): options can be, [detail|clients|mem|inode|fd|callpool|tasks]. If not given, the function returns the output of gluster volume status

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

volume_status("abc.xyz.com")

glustolibs.gluster.volume_ops.**_parse_volume_status_xml**(*root_xml*)
Helper module for get_volume_status. It takes root xml object as input, parses and returns the 'volume' tag xml object.

glustolibs.gluster.volume_ops.**parse_xml**(*tag_obj*)
This helper module takes any xml element object and parses all the child nodes and returns the parsed data in dictionary format

glustolibs.gluster.volume_ops.**get_volume_status**(*mnode*, *volname='all'*, *service=''*, *options=''*)
This module gets the status of all or specified volume(s)/brick

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Kwargs: volname (str): volume name. Defaults to 'all' service (str): name of the service to get status.

serivce can be, [nfs|shd|<BRICK>|quotad]], If not given, the function returns all the services

options (str): options can be, [detail|clients|mem|inode|fd|callpool|tasks]. If not given, the function returns the output of gluster volume status

Returns volume status in dict of dictionary format, on success NoneType: on failure

Return type dict

Example

get_volume_status("10.70.47.89", volname="testvol") >>>{'testvol': {'10.70.47.89': {'/bricks/brick1/a11': {'status': '1', 'pid': '28963', 'bricktype': 'cold', 'port': '49163', 'peerid': '7fc9015e-8134-4753-b837-54cbc6030c98', 'ports': {'rdma': 'N/A', 'tcp': '49163'}}, '/bricks/brick2/a31': {'status': '1', 'pid': '28982', 'bricktype': 'cold', 'port': '49164', 'peerid': '7fc9015e-8134-4753-b837-54cbc6030c98', 'ports': {'rdma': 'N/A', 'tcp': '49164'}}, 'NFS Server': {'status': '1', 'pid': '30525', 'port': '2049', 'peerid': '7fc9015e-8134-4753-b837-54cbc6030c98', 'ports': {'rdma': 'N/A', 'tcp': '2049'}}, '/bricks/brick1/a12': {'status': '1', 'pid': '30505', 'bricktype': 'hot', 'port': '49165', 'peerid': '7fc9015e-8134-4753-b837-54cbc6030c98',

'ports': {'rdma': 'N/A', 'tcp': '49165'}}}, '10.70.47.118': {'/bricks/brick1/a21': {'status': '1', 'pid': '5427', 'bricktype': 'cold', 'port': '49162', 'peerid': '5397d8f5-2986-453a-b0b5-5c40a9bb87ff', 'ports': {'rdma': 'N/A', 'tcp': '49162'}}, '/bricks/brick2/a41': {'status': '1', 'pid': '5446', 'bricktype': 'cold', 'port': '49163', 'peerid': '5397d8f5-2986-453a-b0b5-5c40a9bb87ff', 'ports': {'rdma': 'N/A', 'tcp': '49163'}}, 'NFS Server': {'status': '1', 'pid': '6397', 'port': '2049', 'peerid': '5397d8f5-2986-453a-b0b5-5c40a9bb87ff', 'ports': {'rdma': 'N/A', 'tcp': '2049'}}}}}

glustolibs.gluster.volume_ops.**get_volume_options**(*mnode*, *volname*, *option='all'*)

gets the option values for the given volume.

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name

Kwargs:

option (str): volume option to get status. If not given, the function returns all the options for the given volume

Returns value for the given volume option in dict format, on success NoneType: on failure

Return type dict

Example

get_volume_options(mnode, "testvol")

glustolibs.gluster.volume_ops.**set_volume_options**(*mnode*, *volname*, *options*)

Sets the option values for the given volume.

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **volname** (*str*) – volume name
- **options** (*dict*) – volume options in key value format

Returns

True, if the volume option is set False, on failure

Return type bool

Example

options = {"user.cifs":"enable","user.smb":"enable"} set_volume_option("abc.com", "testvol", options)

glustolibs.gluster.volume_ops.**volume_info**(*mnode*, *volname='all'*)

Executes gluster volume info cli command

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Kwargs: volname (str): volume name. Defaults to 'all'

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

volume_status("abc.com")

glustolibs.gluster.volume_ops.**get_volume_info**(*mnode*, *volname='all'*)

Fetches the volume information as displayed in the volume info. Uses xml output of volume info and parses the into to a dict

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Kwargs: volname (str): volume name. Defaults to 'all'

Returns If there are errors dict: volume info in dict of dicts

Return type NoneType

Example

get_volume_info("abc.com", volname="testvol") >>>{'testvol': {'status': '1', 'xlators': None, 'disperseCount': '0', 'bricks': {'coldBricks': {'colddisperseCount': '0', 'coldarbiterCount': '0', 'coldBrickType': 'Distribute', 'coldbrickCount': '4', 'numberOfBricks': '4', 'brick': [{'isArbiter': '0', 'name': '10.70.47.89:/bricks/brick1/a11', 'hostUuid': '7fc9015e-8134-4753-b837-54cbc6030c98'}, {'isArbiter': '0', 'name': '10.70.47.118:/bricks/brick1/a21', 'hostUuid': '7fc9015e-8134-4753-b837-54cbc6030c98'}, {'isArbiter': '0', 'name': '10.70.47.89:/bricks/brick2/a31', 'hostUuid': '7fc9015e-8134-4753-b837-54cbc6030c98'}, {'isArbiter': '0', 'name': '10.70.47.118:/bricks/brick2/a41', 'hostUuid': '7fc9015e-8134-4753-b837-54cbc6030c98'}], 'coldreplicaCount': '1'}, 'hotBricks': {'hotBrickType': 'Distribute', 'numberOfBricks': '1', 'brick': [{'name': '10.70.47.89:/bricks/brick1/a12', 'hostUuid': '7fc9015e-8134-4753-b837-54cbc6030c98'}], 'hotbrickCount': '1', 'hotreplicaCount': '1'}}, 'type': '5', 'distCount': '1', 'replicaCount': '1', 'brickCount': '5', 'options': {'cluster.tier-mode': 'cache', 'performance.readdir-ahead': 'on', 'features.ctr-enabled': 'on'}, 'redundancyCount': '0', 'transport': '0', 'typeStr': 'Tier', 'stripeCount': '1', 'arbiterCount': '0', 'id': 'ffa8a8d1-546f-4ebf-8e82-fcc96c7e4e05', 'statusStr': 'Started', 'optCount': '3'}}

glustolibs.gluster.volume_ops.**volume_sync**(*mnode*, *hostname*, *volname='all'*)

syncs the volume to the specified host

Parameters

- **mnode** (*str*) – Node on which cmd has to be executed.
- **hostname** (*str*) – host name to which volume has to be sync'ed

Kwargs: volname (str): volume name. Defaults to 'all'.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

volume_sync("abc.xyz.com",volname="testvol")

glustolibs.gluster.volume_ops.**volume_list**(*mnode*)

Executes gluster volume list cli command

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Returns

Tuple containing three elements (ret, out, err). The first element 'ret' is of type 'int' and is the return value of command execution.

The second element 'out' is of type 'str' and is the stdout value of the command execution.

The third element 'err' is of type 'str' and is the stderr value of the command execution.

Return type tuple

Example

volume_list("abc.com")

glustolibs.gluster.volume_ops.**get_volume_list**(*mnode*)

Fetches the volume names in the gluster. Uses xml output of volume list and parses it into to list

Parameters **mnode** (*str*) – Node on which cmd has to be executed.

Returns If there are errors list: List of volume names

Return type NoneType

Example

get_volume_list("abc.com") >>>['testvol1', 'testvol2']

glustolibs.gluster.windows_libs module

Description: Module for windows utility functions

glustolibs.gluster.windows_libs.**powershell**(*command*)

wrap a command in powershell call

Parameters **command** (*str*) – the command to wrap with powershell syntax

Returns string with complete powershell command

glustolibs.gluster.windows_libs.**delete_all_windows_mounts**(*clients_info*)

Deletes all the mounts on the windows clients.

Parameters

- **clients_info** (*list*) – List of windows clients info.
- **any item in the clients info doesn't have the 'platform', it is** (*If*) –
- **that it is not windows client and would be ignored.** (*assumed*) –
- **any item in the clients info doesn't have the 'super_user' key,** (*If*) –
- **default we assume the 'super_user' for windows client to be 'Admin'.** (*by*) –
- **all the windows clients, the 'platform' key should be specified** (*For*) –
- **value 'windows'.** (*with*) –
- **Example** –

clients_info = {

'def.lab.eng.xyz.com': { 'host': 'def.lab.eng.xyz.com', 'super_user': 'Admin', 'platform': 'windows'

},

'ghi.lab.eng.blr.redhat.com': { 'host': 'ghi.lab.eng.xyz.com',

}

}

Returns

True if deleting all the mounts on all clients is successful. False otherwise.

Return type bool

glustolibs.gluster.windows_libs.**list_all_windows_mounts**(*clients_info*)

Lists all the mounts on the windows clients.

Parameters

- **clients_info** (*list*) – List of windows clients info.
- **any item in the clients info doesn't have the 'platform', it is** (*If*) –
- **that it is not windows client and would be ignored.** (*assumed*) –
- **any item in the clients info doesn't have the 'super_user' key,** (*If*) –
- **default we assume the 'super_user' for windows client to be 'Admin'.** (*by*) –
- **all the windows clients, the 'platform' key should be specified** (*For*) –

- **value 'windows'.** (*with*) –
- **Example** –

  clients_info = {

  'def.lab.eng.xyz.com': { 'host': 'def.lab.eng.xyz.com', 'super_user': 'Admin', 'platform': 'windows'

  },

  'ghi.lab.eng.blr.redhat.com': { 'host': 'ghi.lab.eng.xyz.com',

  }

  }

Returns

True if listing all the mounts on all clients is successful. False otherwise.

Return type bool

Module contents

2.1.2 Module contents

CHAPTER 3

Indices and Tables

- genindex
- modindex
- search

Python Module Index

g

Index

Symbols

A

B

C

D

E

F

G

H

I

T

U

V

W

Z