DiSTAF Gluster Libs Documentation

Release 0.1

Red Hat, Inc.

June 27, 2016

Contents

CHAPTER 1

Contents

CHAPTER 2

API

2.1 distaflibs.gluster package

2.1.1 Submodules

distaflibs.gluster.brick_ops module

distaflibs.gluster.brick_ops.**add_brick**(*volname*, *nbricks*, *replica=1*, *stripe=1*, *peers=''*, *mnode=''*)

Does the gluster add-brick. If peer is '', peers from the config is taken. And replica/stripe will not be used by default. Returns the output of add-brick command, which would be a tuple of (retcode, stdout, sstderr) from gluster add-brick command.

distaflibs.gluster.brick_ops.**bring_down_brick**(*volname*, *bindex*, *node=''*)

Kills the glusterfsd process of the particular brick Returns True on success and False on failure

distaflibs.gluster.class_setup_nfs_ganesha_vol module

class distaflibs.gluster.class_setup_nfs_ganesha_vol.**SetupNfsGaneshaVol**(*config_data*, *nfs_options='vers=3'*)

Bases: *distaflibs.gluster.gluster_base_class.GlusterBaseClass*

This is the base class for the ganesha-gluster tests It is a subclass of GlusterBaseClass. All ganesha-gluster tests can subclass this and then write test cases

Initialise the class with the config values Kwargs:

nfs_options (str): This argument takes the nfs options, say vers=3 or vers=4. Default value is vers=3.

setup()

Function to setup ganesha and create volume for testing.

teardown(*teardown_ganesha_setup=False*)

The function to cleanup the test setup Kwargs:

teardown_ganesha_setup (bool): If True teardowns ganesha setup, else leaves the ganesha setup as it is. Default value is False

cleanup(*Self*, *delete_vol=False*)

The function to cleanup the volume Kwargs:

delete_vol (bool): If True deletes the volume. else leaves the volume as it is. Defualt value is False

distaflibs.gluster.ganesha module

Description: Library for gluster NFS-Ganesha operations.

distaflibs.gluster.ganesha.**vol_set_nfs_disable**(*volname*, *option=True*, *mnode=None*)
Enables/Disables nfs for the volume. :param volname: Volume name. :type volname: str

Kwargs:

option (Optional[bool]): If True it disables nfs for that volume else enables nfs for that volume. Default value is True.

mnode (Optional[str]): Node on which the command has to be executed. Default value is tc.servers[0].

Returns True if successful, False otherwise.

Return type bool

distaflibs.gluster.ganesha.**vol_set_ganesha**(*volname*, *option=True*, *mnode=None*)
Enables/Disables ganesha for the volume. :param volname: Volume name. :type volname: str

Kwargs:

option (Optional[bool]): If True it enables ganesha for that volume else disables ganesha for that volume. Default value is True.

mnode (Optional[str]): Node on which the command has to be executed. Default value is tc.servers[0].

Returns True if successful, False otherwise.

Return type bool

distaflibs.gluster.ganesha.**validate_ganesha_ha_status**(*mnode=None*)
Validates Ganesha HA Status. Kwargs:

mnode (Optional[str]): Node on which the command has to be executed. Default value is tc.servers[0].

Returns True if successful(HA status is correct), False otherwise.

Return type bool

distaflibs.gluster.ganesha.**set_nfs_ganesha**(*option=True*, *mnode=None*)
Enables/Disables NFS-Ganesha Cluster Kwargs:

option (Optional[bool]): If True it enables the nfs-ganesha HA Cluster, else disables the nfs-ganesha HA Cluster. Default value is True.

mnode (Optional[str]): Node on which the command has to be executed. Default value is tc.servers[0].

Returns True if successful, False otherwise.

Return type bool

distaflibs.gluster.ganesha.**get_host_by_name**(*servers=None*)
Get hostname of the specified servers. Kwargs:

servers (Optional[str]): Get hostnames of the specified servers.

Returns dict with 'hostname or ip_address" of the server as key and 'hostname' of the server as value.

Return type dict

distaflibs.gluster.ganesha.**create_nfs_passwordless_ssh**(*snodes=[]*, *guser=None*, *mnode=None*)
Sets up the passwordless ssh between mnode and all other snodes. :param snodes: List of nodes for which we require passwordless

ssh from mnode.

Kwargs: guser (Optional[str]): Username . Default value is root. mnode (Optional[str]): Node from which we require passwordless

ssh to snodes. Default value is tc.servers[0].

Returns True if successfull, False otherwise

Return type bool

distaflibs.gluster.ganesha.**validate_ganesha_ha_failover**(*mnode=None*, *snodes=None*)
Validates HA failover status Kwargs:

mnode (Optional[str]): Node on which the ha status command has to be executed. Default value is tc.servers[0].

snodes (Optional[str]): Node/Nodes on which ganesha process is Killed/stopped or Node shutdown

Returns True if successfull, False otherwise.

Return type bool

distaflibs.gluster.ganesha.**get_ganesha_ha_failover_nodes**(*mnode=None*, *snodes=None*)
Returns HA status and dictionary of Kwargs:

mnode (Optional[str]): Node on which the ha status command has to be executed. Default value is tc.servers[0].

snodes (Optional[str]): Node/Nodes on which ganesha process is Killed/stopped or Node shutdown

Returns If successfull True,dict False otherwise

Return type bool,dict

distaflibs.gluster.ganesha.**update_ganesha_ha_conf**(*no_of_servers=None*)
Updates the ganesha-ha.conf file, with VIPs and hostnames. Kwargs:

no_of_servers (Optional[int]): The number of nodes on which we have to modify the ganesha-ha.conf file. Default it takes the number of servers from the pool list.

Returns True if successfull, False otherwise.

Return type bool

distaflibs.gluster.ganesha.**cluster_auth_setup**(*no_of_servers=None*)

Sets the hacluster password, starts pcsd service and runs pcs cluster auth command.

Kwargs:

no_of_servers (Optional[int]): The number of nodes on which we have to setup the HA cluster. Default it takes the number of servers from the pool list.

Returns True if successfull, False otherwise.

Return type bool

distaflibs.gluster.ganesha.**setup_nfs_ganesha**(*no_of_servers=None*)

Setup NFS-Ganesha HA cluster. Kwargs:

no_of_servers (Optional[int]): The number of nodes on which we have to setup the HA cluster. Default it takes the number of servers from the pool list.

Returns True if successfull, False otherwise.

Return type bool

distaflibs.gluster.ganesha.**teardown_nfs_ganesha_setup**(*mnode=None*)

Teardowns the NFS-Ganesha HA setup. Kwargs:

mnode (Optional[str]): Node on which the command has to be executed. Default value is tc.servers[0].

Returns True if successful, False otherwise.

Return type bool

distaflibs.gluster.gluster_base_class module

class distaflibs.gluster.gluster_base_class.**GlusterBaseClass**(*config_data*)

This is the base class for the distaf tests

All tests can subclass this and then write test cases

Initialise the class with the config values

_create_volume()

Create the volume with proper configurations

setup()

Function to setup the volume for testing.

teardown()

The function to cleanup the test setup

cleanup()

The function to cleanup the volume

distaflibs.gluster.gluster_init module

distaflibs.gluster.gluster_init.**start_glusterd**(*servers=''*)
Starts glusterd in all servers if they are not running

Returns True if glusterd started in all servers Returns False if glusterd failed to start in any server

(Will be enhanced to support systemd in future)

distaflibs.gluster.gluster_init.**stop_glusterd**(*servers=''*)
Stops the glusterd in specified machine(s)

Returns True if glusterd is stopped in all nodes Returns False on failure

distaflibs.gluster.gluster_init.**env_setup_servers**(*snap=True*, *servers=''*)
Sets up the env for all the tests Install all the gluster bits and it's dependencies Installs the xfs bits and then formats the backend fs for gluster use

Returns 0 on success and non-zero upon failing

distaflibs.gluster.mount_ops module

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

Parameters **mount** (*dict*) – Mount dict with 'mount_protocol', 'mountpoint', 'server', 'client', 'volname', 'options' as keys

Returns Instance of GlusterMount class

client_register = 0

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

distaflibs.gluster.mount_ops.**is_mounted**(*volname*, *mpoint*, *mserver*, *mclient*)
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.

Returns True if mounted and False otherwise.

Return type bool

distaflibs.gluster.mount_ops.**mount_volume**(*volname*, *mtype='glusterfs'*, *mpoint='/mnt/glusterfs'*, *mserver=''*, *mclient=''*, *options=''*)

Mount the gluster volume with specified options.

Parameters **volname** (*str*) – Name of the volume to mount.

Kwargs: 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. option (str): Options for the mount command.

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

distaflibs.gluster.mount_ops.**umount_volume**(*mclient*, *mpoint*)

Unmounts the mountpoint.

Parameters

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

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

Return type tuple

distaflibs.gluster.peer_ops module

Description: Library for gluster peer operations.

distaflibs.gluster.peer_ops.**peer_probe**(*server*, *mnode=None*)

Probe the specified server.

Parameters **server** (*str*) – Server to be peer probed.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.peer_ops.**peer_detach**(*server*, *force=False*, *mnode=None*)
Detach the specified server.

Parameters **server** (*str*) – Server to be peer detached.

Kwargs:

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

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.peer_ops.**peer_status**(*mnode=None*)
Runs 'gluster peer status' on specified node.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.peer_ops.**pool_list**(*mnode=None*)
Runs 'gluster pool list' command on the specified node.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.peer_ops.**peer_probe_servers**(*servers=None*, *validate=True*, *time_delay=10*, *mnode=None*)
Probe specified servers and validate whether probed servers are in cluster and connected state if validate is set to True.

Kwargs:

servers (list): List of servers to be peer probed. If None, defaults to nodes.

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.

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

Returns True on success and False on failure.

Return type bool

distaflibs.gluster.peer_ops.**peer_detach_servers**(*servers=None*, *force=False*, *validate=True*, *time_delay=10*, *mnode=None*)

Detach peers and validate status of peer if validate is set to True.

Kwargs:

servers (list): List of servers to be peer detached. If None, defaults to nodes.

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.

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

Returns True on success and False on failure.

Return type bool

distaflibs.gluster.peer_ops.**nodes_from_pool_list**(*mnode=None*)

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

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Return type NoneType

distaflibs.gluster.peer_ops.**get_peer_status**(*mnode=None*)

Parse the output of command 'gluster peer status'.

Kwargs:

mnode (str): Node on which command has to be executed. if None, defaults to nodes[0].

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'}]

distaflibs.gluster.peer_ops.**get_pool_list**(*mnode=None*)

Parse the output of 'gluster pool list' command.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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'}]

distaflibs.gluster.peer_ops.**is_peer_connected**(*servers=None*, *mnode=None*)

Checks whether specified peers are in cluster and 'Connected' state.

Kwargs:

servers (list): List of servers to be validated. If None, defaults to nodes.

mnode (str): Node from which peer probe has to be executed. If None, defaults to nodes[0].

Returns

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

distaflibs.gluster.quota_ops module

Description: Library for gluster quota operations.

distaflibs.gluster.quota_ops.**enable_quota**(*volname*, *mnode=None*)

Enables quota on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.quota_ops.**disable_quota**(*volname*, *mnode=None*)

Disables quota on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.quota_ops.**is_quota_enabled**(*volname*, *mnode=None*)

Checks if quota is enabled on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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

Return type bool

Example

is_quota_enabled(testvol)

distaflibs.gluster.quota_ops.**quota_list**(*volname*, *mnode=None*)

Executes quota list command for given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.quota_ops.**set_quota_limit_usage**(*volname*, *path='/'*, *limit='100GB'*, *soft_limit=''*, *mnode=None*)

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

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

If None, defaults to servers[0].

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("testvol")
```

distaflibs.gluster.quota_ops.**get_quota_list**(*volname*, *mnode=None*)

Parse the output of 'gluster quota list' command.

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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

Return type NoneType

Examples

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

distaflibs.gluster.quota_ops.**set_quota_limit_objects**(*volname*, *path='/'*, *limit='10'*, *soft_limit=''*, *mnode=None*)

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

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

If None, defaults to servers[0].

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("testvol")
```

distaflibs.gluster.quota_ops.**quota_list_objects**(*volname*, *mnode=None*)

Executes quota list command for given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

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

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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

Return type NoneType

Examples

```
>>> get_quota_list_objects("testvol", mnode = 'abc.lab.eng.xyz.com')
{'/': {'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'}}
```

distaflibs.gluster.quota_ops.**set_quota_alert_time**(*volname*, *time*, *mnode=None*)
Sets quota alert time

Parameters **volname** (*str*) – volume name

Kwargs: time (str): quota limit usage. defaults to 100GB mnode (str): Node on which command has to be executed.

If None, defaults to servers[0].

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("testvol", <alert time>)
```

distaflibs.gluster.quota_ops.**set_quota_soft_timeout**(*volname*, *timeout*, *mnode=None*)
Sets quota soft timeout

Parameters **volname** (*str*) – volume name

Kwargs: timeout (str): quota soft limit timeout value mnode (str): Node on which command has to be executed.

If None, defaults to servers[0].

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("testvol", <timeout-value>)
```

distaflibs.gluster.quota_ops.**set_quota_hard_timeout**(*volname*, *timeout*, *mnode=None*)

Sets quota hard timeout

Parameters

- **volname** (*str*) – volume name
- **timeout** (*str*) – quota hard limit timeout value

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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("testvol", <timeout-value>)
```

distaflibs.gluster.quota_ops.**set_quota_default_soft_limit**(*volname*, *timeout*, *mnode=None*)

Sets quota default soft limit

Parameters

- **volname** (*str*) – volume name
- **timeout** (*str*) – quota soft limit timeout value

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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("testvol", <timeout-value>)
```

distaflibs.gluster.quota_ops.**remove_quota**(*volname*, *path*, *mnode=None*)

Removes quota for the given path

Parameters

- **volname** (*str*) – volume name
- **path** (*str*) – path to which quota limit usage is set. Defaults to /.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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("testvol", <path>)
```

distaflibs.gluster.quota_ops.**remove_quota_objects**(*volname*, *path*, *mnode=None*)

Removes quota objects for the given path

Parameters

- **volname** (*str*) – volume name
- **path** (*str*) – path to which quota limit usage is set. Defaults to /.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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("testvol", <path>)
```

distaflibs.gluster.rebalance_ops module

Description: Library for gluster rebalance operations.

distaflibs.gluster.rebalance_ops.**rebalance_start**(*volname*, *mnode=None*, *fix_layout=False*, *force=False*)

Starts rebalance on the given volume.

Example

rebalance_start(testvol)

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.rebalance_ops.**rebalance_stop**(*volname*, *mnode=None*)

Stops rebalance on the given volume.

Example

rebalance_stop(testvol)

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.rebalance_ops.**rebalance_status**(*volname*, *mnode=None*)

Executes rebalance status on the given volume.

Example

rebalance_status(testvol)

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.rebalance_ops.**get_rebalance_status**(*volname*, *mnode=None*)

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

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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(testvol, mnode = 'abc.lab.eng.xyz.com')
{'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'}
```

distaflibs.gluster.rebalance_ops.**rebalance_stop_and_get_status**(*volname*, *mnode=None*)

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

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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(testvol, mnode = 'abc.xyz.com')
    {'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'}
```

distaflibs.gluster.rebalance_ops.**wait_for_rebalance_to_complete**(*volname*, *mnode=None*, *timeout=300*)

Waits for the rebalance to complete

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Returns True on success, False otherwise

Examples

```
>>> wait_for_rebalance_to_complete("testvol")
```

distaflibs.gluster.snap_ops module

Description: Library for gluster snapshot operations.

distaflibs.gluster.snap_ops.**snap_create**(*volname*, *snapname*, *timestamp=False*, *description=''*, *force=False*, *mnode=None*)

Creates snapshot for the given volume.

Example

snap_create(testvol, testsnap)

Parameters

- **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

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_clone**(*snapname*, *clonename*, *mnode=None*)

Clones the given snapshot

Example

snap_clone(testsnap, clone1)

Parameters

- **snapname** (*str*) – snapshot name to be cloned
- **clonename** (*str*) – clone name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_restore**(*snapname*, *mnode=None*)

Executes snap restore cli for the given snapshot

Example

snap_restore(testsnap)

Parameters **snapname** (*str*) – snapshot name to be cloned

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_restore_complete**(*volname*, *snapname*, *mnode=None*)

stops the volume restore the snapshot and starts the volume

Example

snap_restore_complete(testvol, testsnap)

Parameters

- **volname** (*str*) – volume name

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

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

Returns True on success, False on failure

Return type bool

distaflibs.gluster.snap_ops.**snap_status**(*snapname=''*, *volname=''*, *mnode=None*)

Runs 'gluster snapshot status' on specific node

Example

snap_status()

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**get_snap_status**(*mnode=None*)

Parse the output of 'gluster snapshot status' command.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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(mnode = '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'}]
```

distaflibs.gluster.snap_ops.**get_snap_status_by_snapname**(*snapname*, *mnode=None*)

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

Parameters **snapname** (*str*) – snapshot name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Return type NoneType

Examples

```
    >>> get_snap_status_by_snapname('snap1',
                                    mnode = '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'}
```

distaflibs.gluster.snap_ops.**get_snap_status_by_volname**(*volname*, *mnode=None*)

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

Parameters **volname** (*str*) – snapshot name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Return type NoneType

Examples

```
    >>> get_snap_status_by_volname('testvol',
                                   mnode = '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'}]
```

distaflibs.gluster.snap_ops.**snap_info**(*snapname=''*, *volname=''*, *mnode=None*)

Runs 'gluster snapshot info' on specific node

Example

snap_info()

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**get_snap_info**(*mnode=None*)

Parse the output of 'gluster snapshot info' command.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Return type NoneType

Examples

```
    >>> get_snap_info(mnode = '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'}]
```

distaflibs.gluster.snap_ops.**get_snap_info_by_snapname**(*snapname*, *mnode=None*)

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

Parameters **snapname** (*str*) – snapshot name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Return type NoneType

Examples

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

distaflibs.gluster.snap_ops.**get_snap_info_by_volname**(*volname*, *mnode=None*)

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

Parameters **volname** (*str*) – snapshot name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Return type NoneType

Examples

```
>>> get_snap_info_by_volname('testvol',
                             mnode = 'abc.lab.eng.xyz.com')
{'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'}]}
```

distaflibs.gluster.snap_ops.**snap_list**(*mnode=None*)

Lists the snapshots

Example

snap_list()

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**get_snap_list**(*mnode=None*)

Parse the output of 'gluster snapshot list' command.

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to nodes[0].

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

Return type NoneType

Examples

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

distaflibs.gluster.snap_ops.**snap_config**(*volname=None*, *mnode=None*)

Runs 'gluster snapshot config' on specific node

Example

snap_config()

Kwargs: volname (str): volume name mnode (str): Node on which cmd has to be executed.

If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**get_snap_config**(*volname=None*, *mnode=None*)

Parse the output of 'gluster snapshot config' command.

Kwargs: volname (str): volume name mnode (str): Node on which command has to be executed.

If None, defaults to nodes[0].

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

Return type NoneType

Examples

```
>>> get_snap_config()
{'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'}}
```

distaflibs.gluster.snap_ops.**set_snap_config**(*option*, *volname=None*, *mnode=None*)

Sets given snap config on the given node

Example

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

Parameters **option** (*dict*) – dict of single snap config option

Kwargs: volname (str): volume name mnode (str): Node on which cmd has to be executed.

If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_delete**(*snapname*, *mnode=None*)

Deletes the given snapshot

Example

snap_delete(testsnap)

Parameters **snapname** (*str*) – snapshot name to be deleted

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_delete_by_volumename**(*volname*, *mnode=None*)
Deletes the given snapshot

Example

snap_delete_by_volumename(testvol)

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_delete_all**(*mnode=None*)
Deletes all the snapshot in the cluster

Example

snap_delete_all(testsnap)

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_activate**(*snapname*, *force=False*, *mnode=None*)
Activates the given snapshot

Example

snap_activate(testsnap)

Parameters **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

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.snap_ops.**snap_deactivate**(*snapname*, *mnode=None*)
Deactivates the given snapshot

Example

snap_deactivate(testsnap)

Parameters **snapname** (*str*) – snapshot name to be cloned

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to nodes[0].

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

distaflibs.gluster.tiering_ops module

Description: Library for gluster tiering operations.

distaflibs.gluster.tiering_ops.**add_peer_nodes_to_cluster**(*peers*, *mnode=None*)
Adds the given peer nodes to cluster

Parameters **peers** (*list*) – list of peer nodes to be attached to cluster

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

Returns True, if peer nodes are attached to cluster False, otherwise

Return type bool

Example

add_peer_nodes_to_cluster(['peer_node1','peer_node2'])

distaflibs.gluster.tiering_ops.**tier_attach**(*volname*, *num_bricks_to_add*, *peers*, *replica=1*, *force=False*, *mnode=None*)

Attaches tier to the volume

Parameters

- **volname** (*str*) – volume name
- **num_bricks_to_add** (*str*) – number of bricks to be added as hot tier
- **peers** (*list*) – from these servers, hot tier will be added to volume

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

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol, '2', ['peer_node1','peer_node2'])

distaflibs.gluster.tiering_ops.**tier_start**(*volname*, *force=False*, *mnode=None*)

Starts the tier volume

Parameters **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

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.tiering_ops.**tier_status**(*volname*, *mnode=None*)
executes tier status command

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.tiering_ops.**get_tier_status**(*volname*, *mnode=None*)
Parse the output of 'gluster tier status' command.

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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

Return type NoneType

Examples

```
>>> get_tier_status(mnode = 'abc.lab.eng.xyz.com')
{'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'}
```

distaflibs.gluster.tiering_ops.**tier_detach_start**(*volname*, *mnode=None*)
starts detaching tier on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.tiering_ops.**tier_detach_status**(*volname*, *mnode=None*)
executes detach tier status on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.tiering_ops.**tier_detach_stop**(*volname*, *mnode=None*)
stops detaching tier on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.tiering_ops.**tier_detach_commit**(*volname*, *mnode=None*)
commits detach tier on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.tiering_ops.**tier_detach_force**(*volname*, *mnode=None*)
detaches tier forcefully on given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(testvol)

distaflibs.gluster.tiering_ops.**get_detach_tier_status**(*volname*, *mnode=None*)
Parse the output of 'gluster volume tier detach status' command.

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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

Return type NoneType

Examples

```
>>> get_detach_tier_status("testvol", mnode = 'abc.lab.eng.xyz.com')
{'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'}}
```

distaflibs.gluster.tiering_ops.**tier_detach_start_and_get_taskid**(*volname*, *mnode=None*)

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

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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

Return type NoneType

Examples

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

distaflibs.gluster.tiering_ops.**tier_detach_stop_and_get_status**(*volname*, *mnode=None*)

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

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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

Return type NoneType

Examples

```
    >>> tier_detach_stop_and_get_status("testvol",
                                        mnode = 'abc.lab.eng.xyz.com')
    {'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'}}
```

distaflibs.gluster.tiering_ops.**wait_for_detach_tier_to_complete**(*volname*, *mnode=None*, *timeout=300*)

Waits for the detach tier to complete

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which command has to be executed. If None, defaults to servers[0].

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

Returns True on success, False otherwise

Examples

```
    >>> wait_for_detach_tier_to_complete("testvol")
```

distaflibs.gluster.tiering_ops.**get_files_from_hot_tier**(*volname*)

Lists files from hot tier for the given volume

Parameters **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("testvol")

distaflibs.gluster.tiering_ops.**get_files_from_cold_tier**(*volname*)

Lists files from cold tier for the given volume

Parameters **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")

distaflibs.gluster.tiering_ops.**get_tier_promote_frequency**(*volname*)
Gets tier promote frequency value for given volume.

Parameters **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("testvol")

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

Parameters **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("testvol")

distaflibs.gluster.tiering_ops.**get_tier_mode**(*volname*)
Gets tier mode for given volume.

Parameters **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")

distaflibs.gluster.tiering_ops.**get_tier_max_mb**(*volname*)
Gets tier max mb for given volume.

Parameters **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("testvol")

distaflibs.gluster.tiering_ops.**get_tier_max_files**(*volname*)
Gets tier max files for given volume.

Parameters **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("testvol")

distaflibs.gluster.tiering_ops.**get_tier_watermark_high_limit**(*volname*)
Gets tier watermark high limit for given volume.

Parameters **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("testvol")

distaflibs.gluster.tiering_ops.**get_tier_watermark_low_limit**(*volname*)
Gets tier watermark low limit for given volume.

Parameters **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("testvol")

distaflibs.gluster.tiering_ops.**set_tier_promote_frequency**(*volname*, *value*)
Sets tier promote frequency value for given volume.

Parameters

- **volname** (*str*) – volume name
- **value** (*str*) – promote frequency value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_promote_frequency("testvol", '1000')

distaflibs.gluster.tiering_ops.**set_tier_demote_frequency**(*volname*, *value*)
Sets tier demote frequency value for given volume.

Parameters

- **volname** (*str*) – volume name
- **value** (*str*) – demote frequency value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_demote_frequency("testvol", "500")

distaflibs.gluster.tiering_ops.**set_tier_mode**(*volname*, *value*)
Sets tier mode for given volume.

Parameters

- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_mode("testvol", "cache")

distaflibs.gluster.tiering_ops.**set_tier_max_mb**(*volname*, *value*)
Sets tier max mb for given volume.

Parameters

- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_max_mb("testvol", "50")

distaflibs.gluster.tiering_ops.**set_tier_max_files**(*volname*, *value*)
Sets tier max files for given volume.

Parameters

- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_max_files("testvol", "10")

distaflibs.gluster.tiering_ops.**set_tier_watermark_high_limit**(*volname*, *value*)
Sets tier watermark high limit for given volume.

Parameters

- **volname** (*str*) – volume name

- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_watermark_high_limit("testvol", "95")

distaflibs.gluster.tiering_ops.**set_tier_watermark_low_limit**(*volname*, *value*)
Sets tier watermark low limit for given volume.

Parameters

- **volname** (*str*) – volume name
- **value** (*str*) – tier mode value

Returns True on success, False Otherwise

Return type bool

Examples

>>>set_tier_watermark_low_limit("testvol", "40")

distaflibs.gluster.tiering_ops.**get_tier_pid**(*volname*, *mnode*)
Gets tier pid for given volume.

Parameters

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

Returns None if command execution fails, parse errors. str: pid of tier process on success.

Return type NoneType

Examples

>>>get_tier_pid("testvol", "abc.xyz.com")

distaflibs.gluster.tiering_ops.**is_tier_process_running**(*volname*, *mnode*)
Checks whether tier process is running

Parameters

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

Returns True on success, False otherwise

Examples

>>>is_tier_process_running("testvol", "abc.xyz.com")

distaflibs.gluster.volume_ops module

distaflibs.gluster.volume_ops.**create_volume**(*volname*, *mnode=None*, *dist=1*, *rep=1*, *stripe=1*, *trans='tcp'*, *servers=None*, *disp=1*, *dispd=1*, *red=1*)

Create the gluster volume specified configuration volname and distribute count are mandatory argument

Parameters **volname** (*str*) – volume name that has to be created

Kwargs:

mnode(str): server on which command has to be exeeceuted, defaults to tc.servers[0]

dist(int): distribute count, defaults to 1 rep(int): replica count, defaults to 1 stripe(int): stripe count, defaults to 1 trans(str): transport type, defaults to tcp servers(list): servers on which volume has to be created,

defaults to number of servers in pool list, if that is None, then takes tc.servers

disp(int): disperse count, defaults to 1 dispd(int): disperse-data count, defaults to 1 red(int): rdundancy count, defaults to 1

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

create_volume(volname)

distaflibs.gluster.volume_ops.**start_volume**(*volname*, *mnode=None*, *force=False*)

Starts the gluster volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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

start_volume("testvol")

distaflibs.gluster.volume_ops.**stop_volume**(*volname*, *mnode=None*, *force=False*)
Stops the gluster volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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

stop_volume("testvol")

distaflibs.gluster.volume_ops.**delete_volume**(*volname*, *mnode=None*)

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

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

Returns True, if volume is deleted False, otherwise

Return type bool

distaflibs.gluster.volume_ops.**reset_volume**(*volname*, *mnode=None*, *force=False*)
Resets the gluster volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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

reset_volume("testvol")

distaflibs.gluster.volume_ops.**cleanup_volume**(*volname*, *mnode=None*)

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

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

Returns True, if volume is deleted successfully False, otherwise

Return type bool

Example

cleanup_volume("testvol")

distaflibs.gluster.volume_ops.**setup_vol**(*volname*, *mnode=None*, *dist=1*, *rep=1*, *dispd=1*, *red=1*, *stripe=1*, *trans='tcp'*, *servers=None*)

Setup a gluster volume for testing. It first formats the back-end bricks and then creates a trusted storage pool by doing peer probe. And then it creates a volume of specified configuration.

When the volume is created, it sets a global flag to indicate that the volume is created. If another testcase calls this function for the second time with same volume name, the function checks for the flag and if found, will return True.

Parameters **volname** (*str*) – volume name that has to be created

Kwargs:

mnode(str): server on which command has to be exeeceuted, defaults to tc.servers[0]

dist(int): distribute count, defaults to 1 rep(int): replica count, defaults to 1 stripe(int): stripe count, defaults to 1 trans(str): transport type, defaults to tcp servers(list): servers on which volume has to be created,

defaults to number of servers in pool list, if that is None, then takes tc.servers

disp(int): disperse count, defaults to 1 dispd(int): disperse-data count, defaults to 1 red(int): rdundancy count, defaults to 1

Returns True on success and False for failure.

Return type bool

distaflibs.gluster.volume_ops.**volume_status**(*volname='all'*, *service=''*, *options=''*, *mnode=None*)

Executes gluster volume status cli command

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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()

distaflibs.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.

distaflibs.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

distaflibs.gluster.volume_ops.**get_volume_status**(*volname='all'*, *service=''*, *options=''*, *mnode=None*)

This module gets the status of all or specified volume(s)/brick

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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(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'}}}}}

distaflibs.gluster.volume_ops.**get_volume_option**(*volname*, *option='all'*, *mnode=None*)

gets the option values for the given volume.

Parameters **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

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

Returns value for the given volume option in dict format, on success NoneType: on failure

Return type dict

Example

get_volume_option("testvol")

distaflibs.gluster.volume_ops.**set_volume_options**(*volname*, *options*, *mnode=None*)

sets the option values for the given volume.

Parameters

- **volname** (*str*) – volume name
- **options** (*dict*) – volume options in key value format

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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("testvol", options=options)

distaflibs.gluster.volume_ops.**volume_info**(*volname='all'*, *mnode=None*)
Executes gluster volume info cli command

Kwargs: volname (str): volume name. Defaults to 'all' mnode (str): Node on which cmd has to be executed.

If None, defaults to servers[0].

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()

distaflibs.gluster.volume_ops.**get_volume_info**(*volname='all'*, *mnode=None*)

Fetches the volume information as displayed in the volume info. Uses xml output of volume info and parses the into to a dict

Kwargs: volname (str): volume name. Defaults to 'all' mnode (str): Node on which cmd has to be executed.

If None, defaults to servers[0].

Returns If there are errors dict: volume info in dict of dicts

Return type NoneType

Example

get_volume_info(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'}}

distaflibs.gluster.volume_ops.**sync_volume**(*hostname*, *volname='all'*, *mnode=None*)
syncs the volume

Parameters **hostname** (*str*) – host name

Kwargs: volname (str): volume name. Defaults to 'all'. mnode (str): Node on which cmd has to be executed.

If None, defaults to servers[0].

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

sync_volume("abc.xyz.com",volname="testvol")

distaflibs.gluster.volume_ops.**get_subvols**(*volname*, *mnode=None*)

Gets the subvolumes in the given volume

Parameters **volname** (*str*) – volume name

Kwargs:

mnode (str): Node on which cmd has to be executed. If None, defaults to servers[0].

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("testvol")

2.1.2 Module contents

CHAPTER 3

Related Links

DiSTAF Main Documentation

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

d

Symbols

A

B

C

D

E

G

I

M

N

P

Q

R

S

T

U

V

W