Script to resize cluster nodes

ONTAP Select

David Peterson
November 21, 2019

Table of Contents

Script to resize cluster nodes

You can use the following script to resize the nodes in an ONTAP Select cluster.

```
#!/usr/bin/env python
##------------------------------------------------------------------
#
# File: resize_nodes.py
#
# (C) Copyright 2019 NetApp, Inc.
#
# This sample code is provided AS IS, with no support or warranties of
# any kind, including but not limited for warranties of merchantability
# or fitness of any kind, expressed or implied. Permission to use,
# reproduce, modify and create derivatives of the sample code is granted
# solely for the purpose of researching, designing, developing and
# testing a software application product for use with NetApp products,
# provided that the above copyright notice appears in all copies and
# that the software application product is distributed pursuant to terms
# no less restrictive than those set forth herein.
#
##------------------------------------------------------------------

import argparse
import logging
import sys

from deploy_requests import DeployRequests


def _parse_args():
    """ Parses the arguments provided on the command line when executing this
        script and returns the resulting namespace. If all required arguments
        are not provided, an error message indicating the mismatch is printed and
        the script will exit.
    """

    parser = argparse.ArgumentParser(description=(
        'Uses the ONTAP Select Deploy API to resize the nodes in the cluster.'
        ' For example, you might have a small (4 CPU, 16GB RAM per node) 2 node'
        ' cluster and wish to resize the cluster to medium (8 CPU, 64GB RAM per'
        ' node). This script will take in the cluster details and then perform'
        ' the operation and wait for it to complete.'
    ))
    parser.add_argument('--deploy', required=True, help=(
        'Hostname or IP of the ONTAP Select Deploy VM.'
```

```
    ))
    parser.add_argument('--deploy-password', required=True, help=(
        'The password for the ONTAP Select Deploy admin user.'
    ))
    parser.add_argument('--cluster', required=True, help=(
        'Hostname or IP of the cluster management interface.'
    ))
    parser.add_argument('--instance-type', required=True, help=(
        'The desired instance size of the nodes after the operation is complete.'
    ))
    parser.add_argument('--ontap-password', required=True, help=(
        'The password for the ONTAP administrative user account.'
    ))
    parser.add_argument('--ontap-username', default='admin', help=(
        'The username for the ONTAP administrative user account. Default: admin.'
    ))
    parser.add_argument('--nodes', nargs='+', metavar='NODE_NAME', help=(
        'A space separated list of node names for which the resize operation'
        ' should be performed. The default is to apply the resize to all nodes in'
        ' the cluster. If a list of nodes is provided, it must be provided in HA'
        ' pairs. That is, in a 4 node cluster, nodes 1 and 2 (partners) must be'
        ' resized in the same operation.'
    ))
    return parser.parse_args()


def _get_cluster(deploy, parsed_args):
    """ Locate the cluster using the arguments provided """

    cluster_id = deploy.find_resource('/clusters', 'ip', parsed_args.cluster)
    if not cluster_id:
        return None
    return deploy.get('/clusters/%s?fields=nodes' % cluster_id).json()['record']


def _get_request_body(parsed_args, cluster):
    """ Build the request body """

    changes = {'admin_password': parsed_args.ontap_password}

    # if provided, use the list of nodes given, else use all the nodes in the cluster
    nodes = [node for node in cluster['nodes']]
    if parsed_args.nodes:
        nodes = [node for node in nodes if node['name'] in parsed_args.nodes]

    changes['nodes'] = [
        {'instance_type': parsed_args.instance_type, 'id': node['id']} for node in
nodes]
```

```
90
91      return changes
92
93
94  def main():
95      """ Set up the resize operation by gathering the necessary data and then send
96          the request to the ONTAP Select Deploy server.
97      """
98
99      logging.basicConfig(
100         format='[%(asctime)s] [%(levelname)5s] %(message)s', level=logging.INFO,)
101
102     logging.getLogger('requests.packages.urllib3').setLevel(logging.WARNING)
103
104     parsed_args = _parse_args()
105     deploy = DeployRequests(parsed_args.deploy, parsed_args.deploy_password)
106
107     cluster = _get_cluster(deploy, parsed_args)
108     if not cluster:
109         deploy.logger.error(
110             'Unable to find a cluster with a management IP of %s' % parsed_args
    .cluster)
111         return 1
112
113     changes = _get_request_body(parsed_args, cluster)
114     deploy.patch('/clusters/%s' % cluster['id'], changes, wait_for_job=True)
115
116 if __name__ == '__main__':
117     sys.exit(main())
```