YB O/S Maintenance script

Date Created	1/25/2024
Last Updated	1/25/2024
Document Version	1.0
OS Maint version	1.26
Created by	Mike LaSpina

Summary

Yugabyte database nodes often need to be shut down for maintenance purposes. One of the more common scenarios is the upgrading of the underlying Operating System. In order to dramatically increase the likelihood of this activity going smoothly, shutting down the node should only be done if the cluster is healthy. In addition, other ancillary items should be performed to give the maximum chance of success. This document will mostly focus on what is done in the process as well as what exactly defines a 'healthy' cluster. A brief overview of the code will also be discussed so that this script can be modified to suit the needs of a given organization.

General Information and Standards

The script was developed using Python 2.7. It is compatible with all known versions of Python 3 up to 3.9. The script is maintained on gitHub under the Yugabyte/yb-tools[1] repository.

This script was written with an eye toward readability rather than efficiency and error handling. While duplicating blocks of code was avoided as much as possible, there are a few places where that may be the case. Error handling and logging is generally not up to the standards of a production level system developed by a professional development team. The logging portion of the code is all done using a 'log' function, which in this form simply calls a 'print' statement. This allows logging to be easily modified as needed.

There are several places where API calls to Prometheus as well as masters and tservers will try both HTTPS and HTTP protocols. In general, the check on HTTPS is done first and then HTTP is attempted. This approach eliminates the need to specify which protocol these components are using.

The script makes heavy Use of the Yugabyte Anywhere API. Anyone who wishes to modify this for their organization should be familiar with the following in Python:

- REST API calls (specifically using the 'requests' library)
- JSON processing - all data from the YBA API is returned in JSON format
- YBA access - ensure that the YBA Host, API Key and Customer UUID are known

Items needed to access Yugabyte anywhere (Host, Customer UUID, API Token) should be stored in a hidden rc file with the name specified in the global variable `ENV_FILE_PATTERN`. As delivered, the pattern is `'.yba*.rc'`, which allows for multiple environments such as .yba_dev.rc, .yba_qa.rc, etc. The script will search the folder specified in the global variable `ENV_FILE_PATH` for this file. Once found, it will parse out the values for the items needed. The file should set the environment variables so that they can be used outside of the script as well. The following is an example of the rc file:

```
export YBA_HOST='http:yba-dev.mycompany.com'
export API_TOKEN='11111111-2222-3333-4444-555555555555'
export CUST_UUID='aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee'
```

Functions within the script often accept the parameters 'universe' and 'host' or 'node'. These values are JSON objects that are returned from the API. Specifically, the given universe from the universes[2] endpoint and the given node from within the given universe's JSON.

[1] https://github.com/yugabyte/yb-tools/tree/main/yb_os_maint
[2] https://api-docs.yugabyte.com/docs/yugabyte-platform/branches/2.18/66e50c174046d-list-universes

Actions

The script can perform the following actions:

Stop

If run from a DB node, stops the tserver and master process on that node. The following actions are taken:

- A health check is performed. The Universe MUST pass this check in order to continue with shut down.
- Create a maintenance window for 20 minutes in order to suppress any alerts. The length of this window can be changed by modifying the variable `MAINTENANCE_WINDOW_DURATION_MINUTES`
  This step is skipped in either of the following cases:
  - The flag '--skip_maint_window' is specified
  - More than 1 node is to be shut down by using the –region and --availability_zone flags.
- Pause xCluster replication
  This step is skipped in either of the following cases:
  - The flag '--skip_xcluster' is specified
  - More than 1 node is to be shut down by using the –region and --availability_zone flags.
- Step down node to be stopped if it is a master leader. Note - this currently uses the 'yb-admin' command, but is the only touchpoint to that command.
- Shut down tserver and master processes

When stopping multiple nodes, each node is shut down sequentially rather than in parallel. This will prevent the cluster from entering a state where not enough nodes to form a quorum are up.

If run from a YBA server, shuts down the server by executing 'systemctl stop' on all services in the list stored in `YBA_PROCESS_STOP_LIST`

Resume

If running from a DB node, resume the tserver/master processes on that node. The following actions are taken:

- Start tserver and master processes on the given node
- Resume xCluster replication
  This step is skipped in either of the following cases:
  - The flag '--skip_xcluster' is specified
  - More than 1 node is to be shut down by using the –region and --availability_zone flags.
- Remove the maintenance window created in the 'stop' function to allow alerting to resume
  This step is skipped in either of the following cases:
  - The flag '--skip_maint_window' is specified
  - More than 1 node is to be shut down by using the –region and --availability_zone flags.

If run from a YBA server, starts all services by executing 'systemctl start' on all services in the list stored in `YBA_PROCESS_STOP_LIST` in reverse order.

Verify

Ensure that the master and tserver are in the state that YB Anywhere thinks they are in. Communication to the master and tserver http endpoints on the given node is initiated. These must communicate (or not) based on the state returned by YBA.

Fix

This function takes an item to fix. Current functionality only accepts 'placement' as an option. This is designed to work around a specific version issue and should not be needed in most operational use cases. Contact Yugabyte support or professional services if you are having issues with an errant `placementModificationTaskUuid` in your universe's JSON payload.

Health Check

This checks the health of a given universe. If run from a DB Node, a health check will be done on the Universe in which the node resides. This can also be run from anywhere that has access to both the YBA server and the master nodes in the universe. Running from a non-db node requires a Universe name. For example:

```
python yb_os_maint.py -health YB-DEV-1
```

To run a health check on all Universes, specify 'ALL' for the Universe name and a health check will be done for every Universe managed by the YBA instance:

```
python yb_os_maint.py -health ALL
```

A health check is done and must succeed before shutting down any tserver and master process on any nodes. Note that when shutting down multiple nodes, the health check is only executed once at the beginning of the process. Once passed, all the nodes in the region/AZ will be shut down assuming this will not cause a loss of a majority of voting nodes. The following checks are performed regardless of how the health check is run:

- Ensure node to be stopped is in a universe on the given Yugabyte Anywhere instance if calling stop or resume function.
- Ensure that no `placementModificationTaskUuid` exists in the Universe JSON as this will cause issues when shutting down and restarting
- Check for dead nodes, tservers and masters using the …/universe/<universe ID>/status[3] endpoint

[3] https://api-docs.yugabyte.com/docs/yugabyte-platform/branches/2.18/c45fccf078832-get-a-universe-s-status

- Check master and tserver health using the endpoint http(s)://<master node>/api/v1/health-check. A healthy Universe will return the a response similar to this:

```
{
  "dead_nodes": [],
  "most_recent_uptime": 624808,
  "under_replicated_tablets": []
}
```

  - Ensure that all nodes have been up for at least `MIN_MOST_RECENT_UPTIME_SECONDS` - current value is 60
  - Verify that any dead nodes returned are not in the Universe - a decommissioned node would be returned by this healthcheck for a time, so there is a need to verify that it's no longer part of the Universe. If this is the case, the health check will log that a dead node was found but is no longer in the Universe. This scenario will not cause the health check to fail.
  - Verify that there are no under replicated tablets

Check tablet balance and health using the endpoint http(s)://<master node>/api/v1/tablet-servers

- ○ Produce a mini-tablet report with the following format:

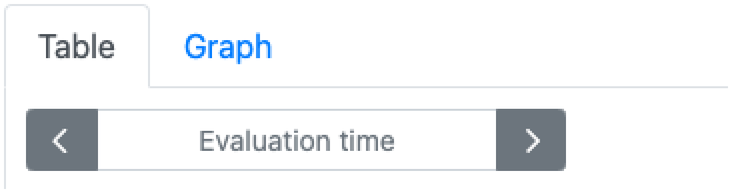
```
TServer 10.202.0.97:9000 Active tablets: 36, User tablets: 24, User tablet
leaders: 0, System Tablets: 12, System tablet leaders: 0

TServer 10.202.0.54:9000 Active tablets: 36, User tablets: 24, User tablet
leaders: 12, System Tablets: 12, System tablet leaders: 6

TServer 10.202.0.55:9000 Active tablets: 36, User tablets: 24, User tablet
leaders: 12, System Tablets: 12, System tablet leaders: 6
```

- ○ Ensure that the tablet-server UI shows as being balanced. This is done by scraping the web page returned from http(s)://<master node>/tablet-servers and searching for the text 'Cluster Load is Balanced' - see global variable `TABLET_BALANCE_TEXT`

- Check tserver tablet lag - this uses Prometheus metrics and ensures lag is under `TSERVER_LAG_THRESHOLD_SECONDS` - current value is 60. Health check will output the Prometheus query which can be pasted directly into the UI and executed if desired. Note if this pasted into prometheus, choose the 'Table' format and then 'Evaluation Time'.

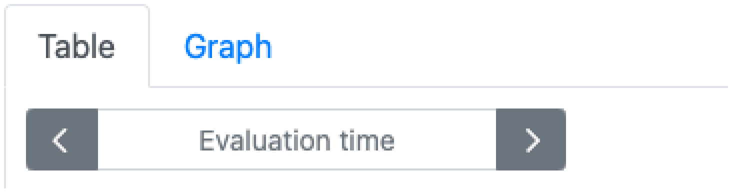


The look back period is `LAG_LOOKBACK_MINUTES` - current value is 10:

```
Executing the following Prometheus query:

max(max_over_time(follower_lag_ms{exported_instance=~"yb-dev-xcluster-src-n1|yb-dev-xcl
uster-src-n3|yb-dev-xcluster-src-n2|",export_type="tserver_export"}[10m]))
```

- Check number of masters is equal to the Universe's replication factor
- Check master tablet lag - this uses Prometheus metrics and ensures lag is under `MASTER_LAG_THRESHOLD_SECONDS` - current value is 60. Health check will output the Prometheus query which can be pasted directly into the UI and executed if desired. Health check will output the Prometheus query which can be pasted directly into the UI and executed if desired. Note if this pasted into prometheus, choose the 'Table' format and then 'Evaluation Time'.

  The look back period is `LAG_LOOKBACK_MINUTES` - current value is 10:
-

```
Executing the following Prometheus query:

max(max_over_time(follower_lag_ms{exported_instance=~"yb-dev-xcluster-src-n1|yb-dev-xcluster-src-n3|yb-dev-xcluster-src-n2|",export_type="master_export"}[10m]))
```

If all the above checks pass, then the Universe is considered healthy. These must pass in order to continue with the 'Stop' command.

Future Modifications

Planned changes include the ability to override some of the global variables with parameters and/or environment variables in order to make the script more portable.