opentext™

Product:	OpenText™ Exceed TurboX
Version:	12.0
Task/Topic	Deployment
Audience:	Administrators, Decision Makers
Platform:	All
Document ID:	600004
Updated:	May 29, 2019

Solution Paper

Exceed TurboX 12 High Availability Configuration Guide

Contents

Revision History

Revision Number	Date	Comments
1.0	May 24, 2019	Draft Version
1.1	May 28, 2019	Draft Version
1.2	May 31, 2019	First Public Version

Introduction

OpenText™ Exceed TurboX (ETX) version 12 introduces a new type of High Availability (HA) support based on the *Active-Active Server* model. This model allows administrators to install up to four ETX Servers in a highly available "cluster" which has the following attributes:

- All servers in the cluster can serve requests (for example: web sign-ins, REST API calls) at the same time
- All servers in the cluster maintain an identical[1] copy of all ETX Site data in a locally-managed datastore
- All servers in the cluster synchronize changes made through local user sessions to other servers in the cluster via direct TCP connections, creating a mesh-style “network” of replicated servers
- Servers that fail or disconnect from the cluster will re-synchronize with all servers in the cluster once the failed server is brought back online

---

[1] Except for changes that are not yet replicated, and differences in node and session information due to different communication intervals between servers and nodes.

Server Design

This section provides fundamental information about ETX Server and about how data replication works in a High Availability cluster. It is recommended that you review this material before deploying your own HA server cluster.

Server architecture

ETX Server consists of several core components: the server core (Java main app), the Jetty web server (which serves the web user interface), the embedded datastore, and the bi-directional replication module.

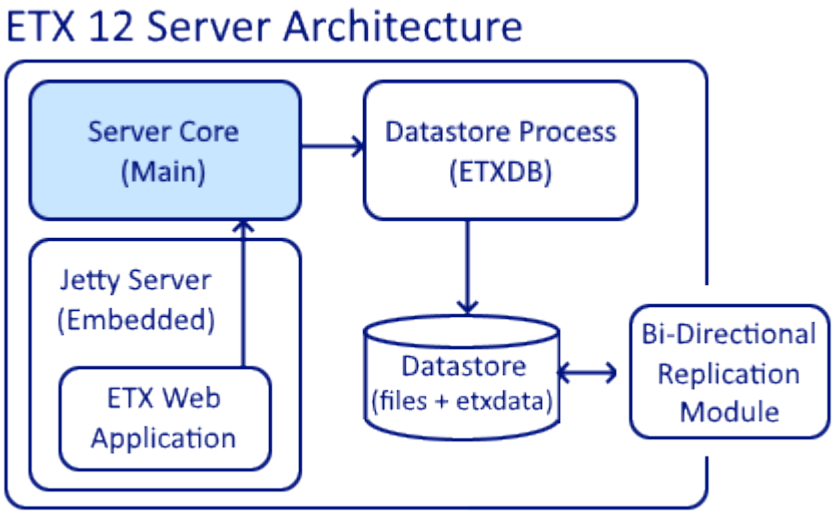


Figure 1 - ETX Server internal architecture

Data replication architecture

Each ETX Server contains a bi-directional data replication (BDR) module that performs the following functions:

- Broadcasts local datastore changes (files and data) to other servers in the cluster
- Applies updates from other servers in the cluster to the local datastore
- Maintains connections to other servers in the cluster and reconnects after a disconnect
- When another server in the cluster becomes disconnected, a *journal* of local changes is created to “play back” on the remote servers, when the remote servers are reconnected

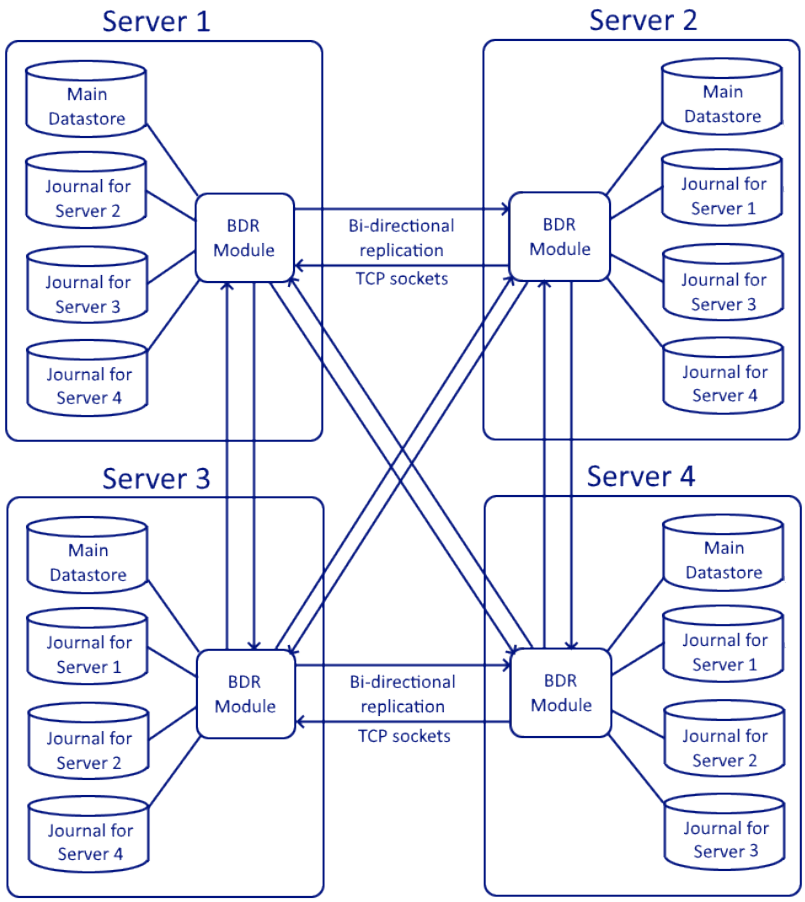


Figure 2 - Data replication architecture

To ensure data integrity, write operations to each server’s local (main) datastore are **atomic**, meaning that they can occur in any order, without data corruption.

To ensure data consistency between servers, write operations to the datastore are time-stamped and executed in sequence, so that each server in the cluster performs changes in the same order. This ensures that each server maintains the same data state. Also, since timestamps are based on local server time and are stored internally

in UTC, servers can be deployed in different time zones. However, for proper cluster functionality, all servers in a cluster need to have their clocks synchronized.

What data gets replicated

Not all data on the ETX server is replicated to other servers in the cluster. The following table describes which data is synchronized in the ETX Server Cluster.

Synchronized	Not Synchronized
• Nodes and node groups • Users and user groups • Profiles and sessions • Published applications • Site settings ('Site Settings' tab in Server Manager) • Custom profile icon files • License keys and license utilization • Binary files* (/data/custom, /data/runtimes) • Server activity logs	• Server configuration files (/data/conf) • Server SSL certificates (/data/ssl) • Executable & library files (/bin, /lib) • Temporary files (/run) • Server logs (/logs) • Files from the installation package (/data/clientlaunchers, /data/filedepot, /data/nodepackages) • Session information, including screenshots (/data/screenshots), activity logs from connection nodes, and node resource usage (each Server in the cluster gets this independently from the connection nodes) • Session and license reports (each Server in the cluster generates reports independently for the whole cluster)

* Binary files may not get synchronized immediately, due to the time delay when replicating files across the network.

ETX Server Cluster Deployment Scenarios

This section presents common deployment scenarios for high availability and discusses their advantages and disadvantages. The recommended scenario for ETX 12 Server Clusters is the HTTP(S) load balancer.

HTTP(S) load balancer

In this scenario, a load balancer distributes web sessions to the ETX Server cluster. The ETX Servers may be installed in the same physical location or in different physical locations.

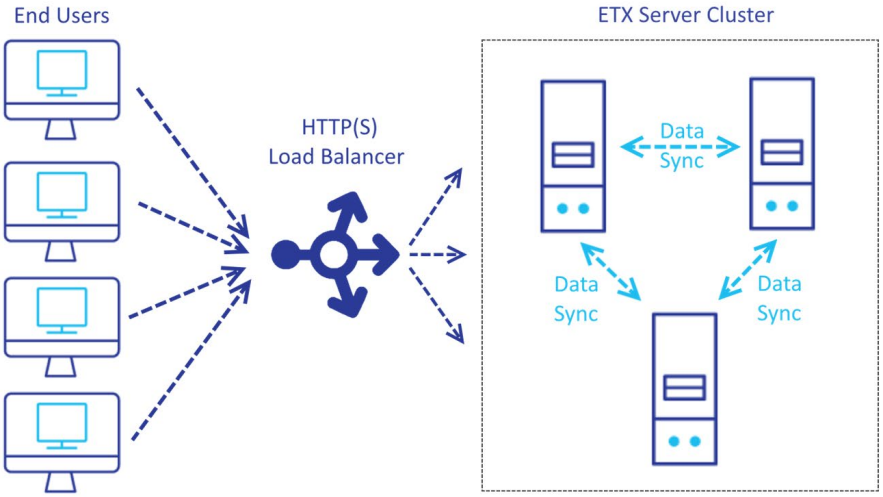


Figure 3 - Simple HTTP(S) load balancer

For an example load balancer configuration, see *Appendix A: Load Balancing with HAProxy*. Note that this HAProxy example is provided for education and testing; for production environments it is recommended that you use your enterprise-standard load balancer.

Load balancers provide a number of benefits, including:

- **Naming abstraction**: Users only need to remember the load balancer URL, instead of multiple back-end server URLs.
- **Fault tolerance:** Load balancers check the health of each ETX Server in the cluster and only forward users to healthy servers.
- **Performance:** Load balancers can assign users to servers in the same geographical region via IP-defined assignment rules and can be configured to distribute sessions to servers with the least load.
- **Reporting:** Most load balancers provide a reporting tool which gives a snapshot of server health, uptime, number of sessions, and other server statistics. This helps administrators keep the environment running smoothly.

When configuring a load balancer, it is important to enable persistent (a.k.a. "sticky") sessions so that users are assigned to the same server for the duration of the session. An example sticky session configuration is presented in Appendix A.

NOTE: Most enterprise load balancers support VRRP protocol (see: *Hot failover (VRRP)*), which allows a second load balancer to take over load balancing duties when the first load balancer fails. The backup load balancer can be located in a different datacenter and have its own local ETX server cluster, providing a completely separate environment for disaster recovery purposes.

Round-robin DNS

In Round-robin DNS, you configure your DNS server with multiple A records for resolving your hostname (in this case etx-site.com is mapped to four ETX servers from 10.0.0.1 to 10.0.0.4). DNS will automatically cycle through each A record, returning the next IP address to each end user who connects.

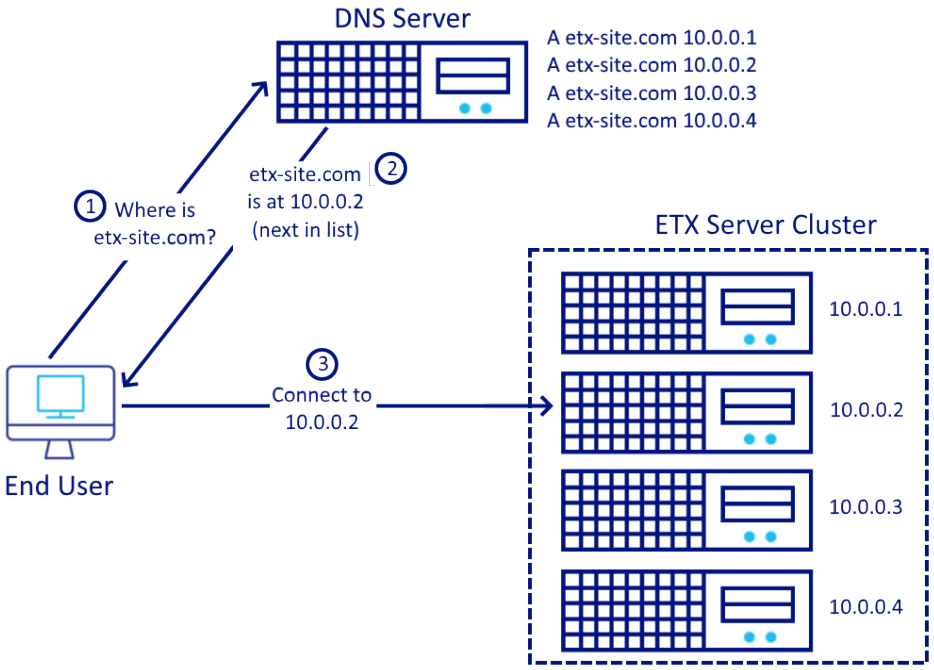


Figure 4 - Round-robin DNS load balancing

The advantage of this approach is that it does not require the configuration of a load balancer. However, it cannot detect if an ETX Server in the cluster is not responding; for example, if one out of four ETX Servers has failed, then ¼ of users will be sent to the failed server.

DNS Delegation

In DNS Delegation load balancing, the DNS authority for the ETX Site URL (for example: *etx-site.com*) has a Name Server (NS) record for each back-end server. When a client contacts the DNS to resolve the IP for *etx-site.com*, the DNS forwards the request to the Name Servers for the domain simultaneously. Each ETX Server is also a DNS server with an A record set to its local address; the first to respond to the NS request for etx-site.com returns its own IP address. This means that the new session will be assigned to that server.

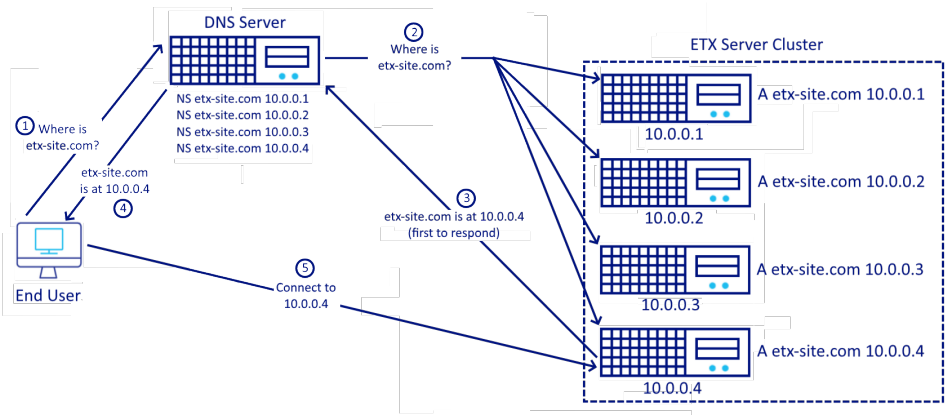


Figure 5 - DNS Delegation load balancing

This approach improves on DNS round-robin by adding load balancing and fault tolerance, because failed servers will not respond to the DNS request, and the fastest (presumably least loaded) server responds first to the DNS request and gets the new session assigned to its own IP address. However, this approach requires that you install a DNS service on each back-end server.

DNS Delegation also does not solve the fundamental problem with DNS Round Robin, which is that there is no application-level health check to determine if the ETX Server application is running and ready to accept connections. DNS Delegation can only tell if the server is responding to DNS requests. For this reason, neither DNS approach is the recommended method for load balancing ETX Server clusters.

Hot failover (VRRP)

Another common high availability deployment model uses VRRP (Virtual Router Redundancy Protocol) to map a set of physical servers to a shared virtual IP address. In the “hot failover” model, two to four ETX servers can be mapped to a shared Virtual IP. These servers may operate as stand-alone ETX Servers, or as part of an ETX Server Cluster with a replicated datastore.

To configure this environment, you need to install a VRRP service (such as *keepalived*) on all ETX Servers. The currently active server (the one to which the virtual IP is mapped) has a *keepalived* process running which periodically executes a health check script on the local ETX Server to see if it is responding. If the check succeeds, the active server will advertise (via multicast) to the other VRRP (keepalived) services that the server is running and healthy. If the active server fails to issue such an update for three successive health check periods, one of the other servers (as defined by the *VRRP priority*) will take over the responsibility of responding to the virtual IP address. This means that one of the backup ETX Server systems will take over the active duty of serving web sessions to users, who are accessing ETX Server via HTTP(S) on the virtual shared IP address.

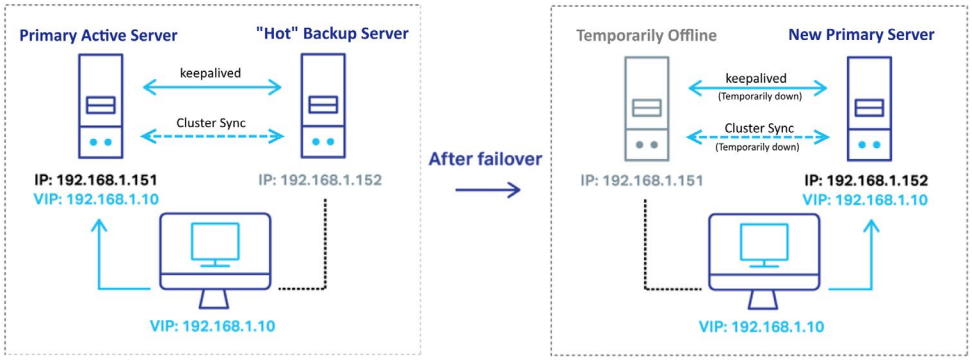


Figure 6 - Hot failover using VRRP protocol

This type of "hot failover" scenario was the recommended failover model in version 11.5 and earlier because *keepalived* provides script-based health checks of ETX Server, which is a reliable method of detecting that the ETX Server is functioning properly. *keepalived* also has a feature to send an email notification when a failover occurs.

However, the failover model is not ideal for version 12 because it fails to take advantage of the active-active nature of ETX 12 high availability, where multiple servers can be active at the same time and distribute web session load; there is also a significant delay during VRRP failover before a backup server takes over the virtual IP. During such a failure, no server will respond to HTTP(S) requests arriving at the virtual IP, and users will get 404 or 503 errors. A load balancer like HAProxy will detect a failure more quickly.

For these reasons, a VRRP ("hot failover“) scenario is not ideal for ETX 12 Server clusters and will not be discussed at length in this whitepaper.

ETX Server Cluster Setup

Creation and management of the ETX Server cluster is designed to be simple and automatable via shell scripts. Cluster setup consists of the following basic tasks:

Step 1: Allocate sufficient disk space to each server

Before proceeding with cluster setup, you must verify that each Server in the cluster has sufficient free disk space to store binary files (for example: runtimes, screenshots, patches), server and activity logs, and the *replication journal* tables, which record information about local datastore changes when one or more Servers in the cluster become disconnected. **Failing to allocate sufficient disk space to each ETX Server for the replication journals can result in corruption of the shared datastore.**

It is recommended that you:

- Have at least 50GB of free disk space on each server in the cluster for a light to moderately loaded environment (up to 300 active web sessions per server)
- Have at least 200GB of free disk space on each server in the cluster for a heavily loaded environment (up to 2000 active web sessions per server)
- Configure messages in ETX Server Manager (Site Settings -> Messages tab) to notify administrators when disk space on the server is running low; set the threshold to 10GB for a lightly loaded server or 50GB for a heavily loaded server

NOTE: DO NOT install an ETX Connection Node on an ETX Server which is a member of an HA cluster, unless disk quotas are enabled for all users (except for the ETX Server user). Otherwise, if a user starts a session trace and forgets to stop it, the trace can consume all free disk space on the server, resulting in potential cluster corruption.

Step 2: Extract the ETX Server .tar onto two to four servers

The Server Cluster can support up to four ETX Servers. A single ETX Server with sufficient hardware can support thousands of web sessions; therefore, with the exception of very large deployments, the number of servers you deploy in the cluster has more to do with the desired level of redundancy (and perhaps the performance of web sessions in different regions) than it does with load distribution.

When copying files to all servers, you must ensure that the same version is installed (including all patches and hotfixes) on each server in the cluster. Different versions of ETX Server may not be compatible in a cluster scenario. This is also why patches must be applied simultaneously to all Servers in a cluster; for instructions on patching a server cluster, see: *Applying Patches to the Server Cluster*.

Step 3: Disable the firewall on replication ports

The bi-directional replication module installed on each ETX Server must maintain one inbound and one outbound TCP connection to each other server in the cluster for the

purposes of data replication. For example, in a four-server cluster, each server must maintain three outbound and three inbound TCP connections.

For example, if you have four servers at 10.0.0.1 to 10.0.0.4 with the replication port set to 5610 for all servers, and you only want to allow connections from other servers in the cluster, then an example firewall configuration on server 10.0.0.1 might be:

For a system running *iptables* (for example: CentOS / RHEL 6)

```
iptables -I INPUT -p tcp -s 10.0.0.2 --dport 5610 -j ACCEPT
iptables -I INPUT -p tcp -s 10.0.0.3 --dport 5610 -j ACCEPT
iptables -I INPUT -p tcp -s 10.0.0.4 --dport 5610 -j ACCEPT
service iptables save
```

For a system running *firewalld* (for example: CentOS / RHEL 7)

```
firewall-cmd --permanent --zone=public --add-source=10.0.0.2
firewall-cmd --permanent --zone=public --add-source=10.0.0.3
firewall-cmd --permanent --zone=public --add-source=10.0.0.4
firewall-cmd --permanent --zone=public --add-port=5610/tcp
firewall-cmd --reload
```

NOTE: Each server in the cluster must have a unique IP and data replication port combination. For example, if you try to install two ETX Servers on the same host and use an identical data replication port for both server instances, the `bin/etxsvr cluster join` command will fail with an error. You must specify a different replication port for each server.

NOTE: To avoid opening ports in your public firewall, it is recommended that you install your ETX Servers on the same physical or virtual private network (VPN).

In addition to the system firewall, you may need to allow HTTP / HTTPS ports for accessing the Dashboard and Server Manager, as well as the License Server port if you choose to connect your HA Site to an ETX License Server. Please consult the *Installation and Configuration Guide* for full installation instructions.

Step 4: Start one server and complete initial setup

In order to create the ETX Server cluster, one of the ETX Servers in the cluster must be started and configured up to the point where the *etxadmin* account can successfully sign in to Server Manager. You can also create the cluster from a production server that has been operating as a stand-alone server for some time.

If this is a new environment and no servers have been started, you must perform the following steps on ONE of the servers:

Update the server configuration

```
bin/etxsvr config
```

This command prints to console all the configuration settings for the server. If you want to change any settings before starting the server, you may do so now. For example, to set the HTTP port for Dashboard and Server Manager to Port 80 (instead of 8080), run the following commands:

```
bin/etxsvr config dashboardHttpPort=80
bin/etxsvr config adminHttpPort=80
```

Start the server

```
bin/etxsvr start
```

If `bin/etxsvr start` completes successfully, it prints the URL of the Server Manager to console. For example:

```
[root@qals-centos66-01 etxsvr]# bin/etxsvr start
Starting Exceed TurboX Server...
Waiting Exceed TurboX Server with PID 28662 to become ready.......
ETX Server Manager URL: http://etx1.mycompany.com:8080/etx/admin
```

Sign in to Server Manager

Copy the Server Manager URL (highlighted above) into your browser to complete setup. You will be asked to accept the EULA and set the password for the *etxadmin* user.

NOTE: When the ETX Server starts for the first time, a datastore is created and initialized automatically. If you connect to Server Manager immediately after starting the ETX Server, you may see an initialization screen before the screen switches to the EULA screen.

Completing server setup from shell

For a script-driven setup, you may wish to avoid opening a browser to complete server setup. In this case, you must execute the following commands from the console, prior to starting ETX Server for the first time:

- Initialize the datastore: `bin/etxsvr datastore init`
- Accept the EULA: `bin/etxsvr config eulaAccepted=1`
- Set the *etxadmin* password: `bin/etxsvr etxadmin setpasswd`

Step 5: Create the cluster

Once you have successfully signed in to ETX Server Manager as the *etxadmin* user, you can proceed with creating the cluster. You must stop the server before creating the cluster.

```
bin/etxsvr stop
bin/etxsvr cluster create
bin/etxsvr start
```

Once these steps have completed successfully, you can join the other servers to the cluster.

Step 6: Join servers to the cluster

Before you join a new server to the cluster, you must ensure that:

- All servers in the cluster are currently running.
- If there is more than one server already in the cluster, sign in to each server and verify that the **Replication Status** says **Ready** in the Site Settings -> Cluster tab (or check it via console using the `bin/etxsvr cluster status` command).
- The joining server has never been started (via `bin/etxsvr server start`). Joining a server to the cluster requires that the joining server has a blank datastore, as the datastore is initialized as part of the `join` operation. If the server was started at some point in the past, stopping the server and deleting the `/data/etxdata` directory will allow you to join it to the cluster. It is recommended to start the join from a freshly untarred server.
- The HTTP and HTTPS ports used by all servers in the cluster are set correctly. To change HTTP or HTTPS ports for a server in a cluster, you will have to remove and re-add the server to the cluster (see: *Changing port numbers for servers in an HA cluster*).

To join the new server to the cluster, execute the `cluster join` command and start the server:

```
bin/etxsvr cluster join
bin/etxsvr start
```

The join script will ask you to specify a data replication port and a unique name for the server.

NOTE: The default server name uses the format <hostname>_<replication_port>. This is because each server in the cluster must use a unique name. A name derived from the hostname and replication port number is unique for each server in the cluster.

Step 7: Verify the cluster

Once you have finished joining servers to the cluster, you should verify that all servers in the cluster are able to communicate, by signing in to Server Manager **on each server in the cluster*** and checking the Site Settings -> Cluster tab. The **Replication status** column should say **Ready** for each other server in the cluster.

* Cluster status must be checked **from all servers** because of the mesh topology. For example, if you sign in to s1, you will see the connection status for (s1-s2), (s1-s3) and (s1-s4). You will not see the status of (s2-s3), (s2-s4), or (s3-s4). Checking one server only gives a partial picture of cluster health.

You can also check cluster status from console, by running the following command **on each server** in the cluster:

```
bin/etxsvr cluster status
```

Converting a server in the cluster to a stand-alone server

NOTE: Converting an ETX Server Cluster into multiple stand-alone servers is not recommended, as each stand-alone server will continue connecting to the old server cluster and Connection Nodes. Connection Nodes are not designed to operate as part of multiple ETX Sites.

To delete the ETX Server Cluster and revert your Site to a stand-alone configuration:

1. Make sure that all servers in the cluster are running.
2. On the server which will be converted to a stand-alone server, call `bin/etxsvr remove <name>` and then `bin/etxsvr delete <name>` for all servers *except* for the local server.
3. On the server which will be converted to a stand-alone server, call `bin/etxsvr cluster remove <local_name> --force` to convert the server to a stand-alone server.
4. On the servers that were removed from the cluster, call `bin/etxsvr stop` to stop the server, then `bin/etxsvr bootstart disable` to remove the startup script, and finally delete the server root folder using `rm -rf <server_root_path>`.

Cluster licensing

An ETX Server Cluster can be licensed the same way as a stand-alone ETX Server:

- By installing a local license key on one of the servers in the cluster
- By connecting one of the servers in the cluster to a license server

The ETX Server Cluster operates in the same way that a single ETX Server does, for the purposes of license consumption. Specifically:

- Installing a license key on one of the servers in a high availability cluster will share the key among all servers in the cluster. License consumption is based on the total number of active or suspended sessions that are reported by the Connection Nodes; since this data is shared among all servers, the licenses consumed will be identical across all servers.
- To connect an ETX cluster to a license server, you need to register only one of the cluster servers with the license server. The Connected ETX Sites grid on the license server will only show the name of the registered server, but the licenses will be allocated to the whole cluster. If you remove the original server from the cluster, the name in the Connected Sites grid will update to a different server name in the cluster.

Changing port numbers for a server in the HA cluster

The ETX Server Cluster is not designed to handle live changes to the data replication port or the HTTP(S) ports used by individual servers in the cluster. HTTP(S) port information is a server setting, which is local and not replicated to the shared datastore except when joining the server to a cluster. Changing data replication ports will interrupt communication with other cluster servers and should only be done during initial cluster configuration.

To change the data replication port or HTTP(S) ports for a server in the cluster, you must remove it from the cluster and re-join it with the following procedure:

1. Verify that all servers in the cluster (other than the server being removed) are started and the **Replication status** is shown as **Ready** for all other servers. The Replication Status can be checked on the Site Settings -> Cluster tab in Server Manager or by checking the output of `bin/etxsvr cluster status`. The replication status **must be verified on all servers in the cluster**; it is not enough to check the status from only one server.
2. Call `bin/etxsvr cluster remove <server_name>` on any server which is remaining in the cluster (NOT on a server which is being removed). This removal will propagate to other servers which are still part of the cluster.
3. Call `bin/etxsvr cluster delete <server_name>` to remove the record of the removed server from the datastore. If you skip this step you will not be able to re-join the server back to the cluster with the same name. **Note:** The deletion will not be propagated to the removed server, but this is fine because the removed server's datastore will be deleted and rebuilt during this procedure.

4. Verify that the removed server is no longer listed as a cluster member by other ETX Servers in the cluster (check the Cluster tab or run `bin/etxsvr cluster status` on each other server).
5. On the removed server, stop the ETX Server process with `bin/etxsvr stop` then delete the `/data/etxdata` directory.
6. (Optional) To modify HTTP or HTTPS ports, call `bin/etxsvr config` on the removed server to see a list of server properties, then call `bin/etxsvr config <propertyname> = value` to set the new HTTP or HTTPS ports.
7. Re-join the server to the cluster by calling `bin/etxsvr cluster join` on the removed server and follow the prompts to set a new data replication port, if desired.
8. Start the newly joined server with `bin/etxsvr start`.

Cluster Maintenance

Applying Patches to a Server Cluster

Runtime patches

Runtime patches are provided as .zip files with the word 'runtime' in the name. Runtimes can be uploaded from the Site Settings -> Runtimes tab in Server Manager from any ETX Server in the cluster. Runtimes are automatically replicated to all servers in the cluster, so do not require any special procedures for clustered ETX Servers.

If you try to launch a session that references a newly uploaded runtime before the runtime has replicated to all servers, the session launch may fail with an error. It is best to wait a few minutes (or hours, if servers are deployed across a slow WAN) before launching profiles with new runtimes. You can check the contents of the /data/runtimes folder on a server to see which runtimes have replicated to it.

Server patches

Server patches are provided as .sh shell scripts with ‘etxsvr’ or ‘server’ in the file names. Server patches should be applied to all Servers in the Cluster at the same time. Applying a server patch to only some of the servers in the cluster can result in communication errors between servers and potentially server crashes or datastore corruption.

To apply server patches:

1. Stop all servers in the cluster using `bin/etxsvr stop`
2. Run the server patch script on each server, by following the installation prompts
3. Start all servers in the cluster using `bin/etxsvr start`

Backing up and Restoring the Cluster

This section explains how to take periodic backups of the Server Cluster, and how to restore the cluster from backup files to a fully working state. It is good practice to take nightly or weekly backups of the ETX Server environment and keep those backups for a period of at least one month; in addition, monthly backups should be kept for a period of at least one year.

The most common failure scenario that requires restoring the cluster from backup is permanent corruption of the shared datastore. However, if you detect problems with the datastore, such as synchronization issues between servers, you may be able to resolve these issues by rebuilding the cluster (see *Rebuilding the ETX Server Cluster*). If the rebuild process does not resolve the issues, then the next step is to restore from an earlier backup.

Creating a backup

To create a cluster backup, call `bin/etxsvr datastore backup </path/to/filename>` on any running server in the cluster. Note the following:

- If you specify a file name without a path, the backup file will be created under the `run` directory in the ETX Server installation directory.
- The command is designed to be used against a running ETX Server, to avoid taking the server offline. If the ETX Server is stopped, you may instead tar the server root directory and use this as a backup. **DO NOT** tar a running server as the datastore may be corrupt after a restore.
- The command can be executed manually at any time, or on a schedule (for example using `crontab`).
- It is only necessary to execute the command against a single server in the cluster.

To reduce the size of backup files, ETX Server binaries are not included in the backup. The backup file is designed to be extracted over a fresh installation of ETX Server. It is therefore important to keep track of which patches and hotfixes have been applied to ETX Server before the backup, as the backup must be extracted on top of the same ETX Server version.

The backup file includes:

- The ETX Server datastore (data/etxdata)
- All binary files (/data/filedepot, /data/runtimes, /data/screenshots)
- The server HTTPS certificate (/data/ssl)
- Server properties and customizations (/data/conf and /data/custom)

NOTE: If you performed an offline backup (by stopping the ETX Server and creating a .tar of the server root directory), this .tar will include all necessary files to restore the server from backup.

Excluding files from the backup

It is not always necessary to back up all binary files. For example, the /runtimes/ folder may contain gigabytes of runtimes which do not need to be backed up every night.

To eliminate runtimes or other binary files from the backup, set the `ETX_BACKUP_SKIP_LIST` environment variable prior to executing the `datastore backup` command. For example, to exclude runtimes and screenshots from the backup:

```
ETX_BACKUP_SKIP_LIST=etxsvr-backup/runtimes,etxsvr-backup/screenshots
bin/etxsvr datastore backup /tmp/etxsvr-2019-05-27-partial.backup
```

Restoring from backup

In this example we have four servers in a cluster, named s1 to s4. Nightly backups are configured on s1, and we want to restore the latest nightly backup.

With this scenario in mind, use the following procedure to restore the cluster from the s1 backup:

1. Back up the /data/conf and /data/ssl folders on s2 to s4 using a system command. For example: `tar cf /backup/etx_conf_ssl-s2.tar ./data/conf ./data/ssl`
2. Stop all servers (including s1) using `bin/etxsvr stop`
3. Call `bin/etxsvr bootstart disable` on s1 to s4
4. Delete the server root directory on s1 to s4
5. Extract the server installation package (.tar) followed by the patch .tar files on s1 to
   Note: To restore an offline backup (.tar of s1's root directory), extract it instead and skip step #7.
6. Restore the /data/conf and /data/ssl directories to s2 to s4 which were created in step #1.
   Note: You do not need to back up directories on s1 because they are included in the backup file.
7. Run `bin/etxsvr datastore restore <backup_file>` against s1
8. Start s1 with `bin/etxsvr start`
9. Sign in to Server Manager on s1 to verify that s1 is restored successfully and working as expected

At this point, s1 has been restored from backup and will operate as a stand-alone server, even though it was a member of a cluster when the backup was created. When restoring from a backup file, all cluster-related tables and journals will be discarded, to provide a clean base to rebuild the cluster from.

The procedure to complete the cluster setup is the same as when creating a new cluster:

10. Create the cluster by calling `bin/etxsvr cluster create` on s1
11. Call `bin/etxsvr cluster join` on s2 to s4 to join them to the cluster
12. Verify the cluster by checking the Site Settings -> Cluster tab or calling `bin/etxsvr cluster status` on each server

Taking a server offline for maintenance

If you need to perform maintenance on an ETX Server which is part of a Cluster, if the maintenance is expected to take days or weeks, follow the procedure in *Removing a server from the cluster* to take the server offline, and rejoin it to the cluster when maintenance is complete (see *Join servers to the cluster* for more detail).

If the maintenance is only expected to take a few minutes or hours, follow this procedure:

1. Stop the server using `bin/etxsvr stop`.
2. Call `bin/etxsvr cluster resync` to push any outstanding changes to other servers in the cluster.
3. If the resync command does not complete within a few seconds (or longer, if there is a large replication backlog), verify that all servers in the cluster are online and accessible by the local server (**Note:** The resync command waits indefinitely for all disconnected servers to reconnect to the cluster, before completing).
4. Once the resync command completes, perform maintenance on the ETX Server.
5. Bring the server back online with `bin/etxsvr start`.

Disaster Recovery

ETX Server clusters are designed to recover automatically from temporary outages that occur during normal operations, such as a temporary network interruption or server reboot. However, permanent server failure and other similar failures will require manual intervention to keep the cluster healthy. This section describes the most common failure scenarios and how to recover from these scenarios with minimal downtime and no data loss.

Temporary failure of one or more servers

The Exceed TurboX Server bi-directional replication (BDR) feature keeps the datastore for each server in the cluster synchronized with the datastore of all other servers in the cluster. If the connection between two servers becomes disrupted temporarily, due to a server crash, network disconnect, or short term maintenance, both servers will record local changes to a *replication journal* for each remote (disconnected) server in the cluster. The replication journal contains a time-stamped list of all datastore transactions that have completed locally since the connection was dropped. When the connection is restored, these time-stamped changes can be merged and *played back* in the same order on both servers, to bring the datastore back into parity.

Temporary outages are considered part of normal day to day operations, so as long as server outages are dealt with quickly and are brought back online, there is no cause for concern.

Permanent (or medium term) failure of one server

The BDR feature is only designed to synchronize changes for short outages, and the recovery process assumes that all servers in the cluster will eventually come back online to synchronize changes. If a server is offline for longer than several days (or several hours in a busy environment), you should remove it from the cluster using the steps described in *Removing a server from the cluster*.

Removing a failed server from the cluster provides many benefits and little drawbacks:

- It can avoid a serious failure where continuous growth of the *replication journals* causes the working servers to run out of disk space, resulting in corruption of the shared datastore and whole cluster failure.
- It improves performance of all servers in the cluster, by eliminating the need for each server to record all local changes to the replication journal for the failed server.
- It avoids the possibility of a lengthy recovery period when the server reconnects to the cluster, which causes each server to synchronize a long backlog of changes in the journals.
- The removed server can be re-added to the cluster at a later date, by reinstalling ETX Server and re-joining it to the cluster using `etxsvr cluster join`. Note that the process of re-joining a server to the cluster can be much faster than re-synchronizing large journals from multiple servers.

Permanent (or long term) failure of multiple servers

If multiple ETX Servers in the cluster fail at the same time and not all servers are recovered, there is a possibility that the datastore will become out of sync between servers.

For example, imagine the following scenario with three servers (s1, s2, and s3) which form a three-server cluster:

1. s1 goes offline
2. s3 goes offline
3. s1 comes online and synchronizes with s2
4. s1 fails permanently and is removed from the cluster
5. s3 comes back online

At the end of this process, the datastore for s2 and s3 will be out of sync because s1 was only able to synchronize its changes with s2 before being removed from the cluster. s3 does not have the s1 changes and, therefore, s2 and s3 will be out of sync.

Failure scenarios that result in data inconsistencies in the cluster should be rare, as the sequence of events leading to such an event is very unlikely. However, if care is not taken to resolve server failures immediately (within a few hours of receiving a failure notification), then such issues can occur more frequently.

Should such a failure scenario occur, or if you notice data inconsistencies between servers in the same cluster, it is possible that the cluster will need to be re-built. See *Verifying Data Inconsistencies* for a procedure to verify that such a scenario has occurred, before taking mitigation steps.

Verifying Data Inconsistencies

Should the data in your cluster become inconsistent between two or more ETX Servers, you should take steps to verify the synchronization issue, before taking corrective actions.

1. Make a record of any inconsistencies seen between servers (for example: "Profile ID ending with ‘b4fd98’ exists on server s1 but not on server s2").
2. Open the Site Settings -> Cluster tab for each server in the cluster, and verify that:
    a. The **Replication status** says **Ready** for all servers.
    b. **Resync data** and **Replication backlog** are very close to 0 KB for a lightly loaded environment; for heavily loaded servers the values may fluctuate but should stay below 500 KB.
3. If one or more servers have a large Replication backlog, stop the server (`bin/etxsvr stop`), then execute the `bin/etxsvr cluster resync` command to force local changes to synchronize to other members of the cluster. If this command fails with an error, try to address the error then re-run the command until it succeeds. Once the resync command succeeds, start the server (`bin/etxsvr start`).
4. Verify that the inconsistencies identified in step #1 are still present. If the inconsistencies are present, but are easily fixed (for example, by deleting and re-

creating an object), then you may attempt to apply a fix manually through Server Manager. However, for significant (or pernicious) inconsistencies you should proceed with rebuilding the cluster. See *Rebuilding the ETX Server Cluster* for the correct procedure.

Removing a server from the cluster

To remove a server from the cluster, it is important to perform the following steps in order:

1. Verify that all servers in the cluster (other than the server being removed) are started and the **Replication status** is shown as **Ready** for all other servers. The Replication Status can be checked on the Site Settings -> Cluster tab in Server Manager or by checking the output of `bin/etxsvr cluster status`. The replication status **must be verified on all servers in the cluster**; it is not enough to check the status from only one server.
2. Call `bin/etxsvr cluster remove <server_name>` on any server which is remaining in the cluster (NOT on a server being removed). This removal will propagate to other servers which are still part of the cluster.
3. Call `bin/etxsvr cluster delete <server_name>` on any server which is remaining in the cluster (NOT on the server being removed). This will delete the record of the removed server from the datastore. If you skip this step you will not be able to re-join a new server back to the cluster with the same name.
4. Verify that the server is no longer listed as a cluster member by other ETX Servers in the cluster (check the Cluster tab for each server - or run `bin/etxsvr cluster status`).
5. If the removed server is still operational, you must now stop the ETX Server process on that server and delete the ETX Server files, so that the server does not attempt to re-connect to the cluster or to communicate with the connection nodes:

   a. Stop the ETX Server process using `bin/etxsvr stop`
   b. Disable bootstart (if enabled) using `bin/etxsvr bootstart disable`
   c. Delete the ETX Server files using `rm -rf <etx_server_root_dir>`

NOTE: Do not attempt to operate a removed server as a stand-alone ETX Server, as it will try to communicate with servers and connection nodes from the cluster (of which it is no longer a part).

Rebuilding the ETX Server Cluster

In some rare situations, the ETX Server Cluster may need to be rebuilt as a result of a data synchronization issue. One such possible scenario is described in the section *Permanent (or long term) failure of multiple servers.*

If you notice that ETX Servers in the cluster have different data, and you have followed the steps described in *Verifying data inconsistencies* to ensure that the differences are not caused by temporary replication issues, then you can take the following steps to rebuild the ETX Cluster and resolve the data synchronization issue.

1. Select the server which has the most up to date information. This server will be the new "master" server that the other servers will be re-synchronized from. This process will purge the datastore and binary files from other servers in the cluster and restore them from the “master” copy.
2. On the master server, call `bin/etxsvr cluster remove <name>` for each stopped server, to fully remove them from the cluster.
3. On the master server, call `bin/etxsvr cluster delete <name>` for each removed server, to fully delete them from the cluster datastore. This will allow the servers to be re-joined to the cluster, using the same name that they used before the rebuild.
4. On the removed servers, stop the ETX Server process using `bin/etxsvr stop`, then delete the `/data/etxdata` directory.
5. Re-join each stopped server to the cluster using `bin/etxsvr cluster join` and following the prompts. If there are firewalls between servers in the cluster, you may wish to use the same data replication ports as before, to avoid re-configuring the firewalls.
6. Start each of the newly joined servers using `bin/etxsvr start`.

Once the rebuild process is complete, sign in to each Server and verify that the data inconsistencies have been eliminated, and that the **Replication Status** shows as **Ready** for each Server in the Cluster tab.

Appendix A: Load Balancing with HAProxy

The scripting examples in this section are written for CentOS 6.6, but can be easily adapted to other operating systems.

Step 1: Download HAProxy

HAProxy version 1.6 or later is required to support the `external-check` option, which executes a health check script to verify the availability of the ETX Server.

Due to the popularity of HAProxy, most Linux distributions have a version of HAProxy available in the default repo. You can check if your repo has an already-compiled version of HAProxy 1.6 or later:

```
yum info haproxy | grep Version
```

If your version is 1.6 or later, you can install from repo, for example:

```
yum install haproxy
```

In our case (running on CentOS 6.6), the version of HAProxy in the repo is 1.5.5, so we will need to download and make it from the latest source. You may wish to do this anyway, as there are many bug fixes and new features in the latest version of HAProxy. According to https://www.haproxy.org/#down, the latest (non-development) version is 1.8.20, so we will download and extract it using:

```
cd ~
wget https://www.haproxy.org/download/1.8/src/haproxy-1.8.20.tar.gz
tar -xvf haproxy-1.8.20.tar.gz
```

Step 2: Build and install HAProxy from source (optional)

The README for HAProxy states that Linux kernel 2.4 or later and gcc version 2.95 to 4.8 are required. We can verify our kernel and gcc versions using:

```
uname -r
gcc --version
```

In our case (CentOS 6.6 with default gcc) the kernel version is 2.6.32 and gcc version is 4.4.7, so we will proceed to `make` HAProxy with the `target` set to our kernel version. We will also add the `USE_OPENSSL=1` parameter which will allow HAProxy to work with SSL termination.

```
cd haproxy-1.8.20
make TARGET=linux2628 USE_OPENSSL=1
sudo make install
```

Step 3: Configure HAProxy settings

If you installed HAProxy from repo, some or all of these configuration steps may be unnecessary, as the repo init script should have executed these commands already. However, you may wish to verify that these commands (or similar) have already been run on your system.

First, we will make the directories and stats file for HAProxy:

```
mkdir -p /etc/haproxy
mkdir -p /var/lib/haproxy
touch /var/lib/haproxy/stats
```

Next, we create a symlink to run haproxy commands as a normal user:

```
ln -s /usr/local/sbin/haproxy /user/sbin/haproxy
```

Next, we add `haproxy` as a system service, and configure it to start at boot time:

```
cp /examples/haproxy.init /etc/init.d/haproxy
chkconfig haproxy on
```

Finally, we create a default user for `haproxy` to run under:

```
useradd -r haproxy
```

Step 4: Configure firewall

In our example we will be installing the load balancer on port 80 (http) and 443 (https) and will configure four back-end servers which operate on port 8080 using HTTP.

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -I OUTPUT -p tcp --dport 8080 -j ACCEPT
service iptables save
```

For hosts using *firewalld* (for example: CentOS / RHEL 7):

```
firewall-cmd --add-service=http --zone=public --permanent
firewall-cmd --add-service=https --zone=public --permanent
firewall-cmd --add-port=8080/tcp --zone=public --permanent
firewall-cmd --reload
```

NOTE: Instead of opening all traffic on port 8080 you may instead open the port for specific servers in your cluster, by executing the command multiple times with `-d <hostname_or_ip>` to specify the other servers (or `--add-source=<hostname_or_ip>` for *firewalld*).

Step 5: Create HAProxy configuration file

The HAProxy configuration file is located at `/etc/haproxy/haproxy.cfg`. You will need to create or modify this file (for example, using `vim`) to configure HAProxy. Here is a sample configuration file to get you started:

```
global
   log /dev/log local0
   log /dev/log local1 notice
   stats timeout 30s
   external-check
   user haproxy
   group haproxy
   tune.ssl.default-dh-param 4096
   daemon

defaults
   log global
   mode http
   option httplog
   option dontlognull
   timeout connect 5000
   timeout client 50000
   timeout server 50000
   stats uri /haproxy?stats

frontend http_front
   bind :80
   bind :443 ssl crt /etc/ssl/myhost.pem
   default_backend http_back

backend http_back
   balance roundrobin
   cookie SERVERID maxidle 30m maxlife 12h insert indirect nocache
   option external-check
   external-check command /bin/haproxy/etxstat.sh
   external-check path "/usr/bin:/bin"
   server s1 etx.server1.hostname:8080 check cookie s1
   server s2 etx.server2.hostname:8080 check cookie s2
   server s3 etx.server3.hostname:8080 check cookie s3
   server s4 etx.server4.hostname:8080 check cookie s4
```

Make sure to set the correct server hostnames and port numbers for your ETX Server cluster by modifying the blue configuration entries. The green and yellow entries require additional configuration, as described in the next sections.

Step 5: Create SSL certificate

In the haproxy.cfg file above you will see the following entry:

```
bind :443 ssl crt /etc/ssl/myhost.pem
```

This tells HAProxy to use SSL Termination, which means that traffic between the end user and HAProxy is encrypted (HTTPS) while the connection between HAProxy and the back-end ETX Servers will be left unencrypted (HTTP). SSL Termination provides SSL security over the internet and eliminates encryption load to the back-end ETX web servers. This mode also enables HAProxy to decrypt and analyze traffic, which is required for many layer-7 load balancer features.

Note: Instead of SSL termination, you may instead opt to use SSL pass-through (which leaves all HTTPS traffic between the end user and ETX back-end server encrypted with HTTPS, instead of decrypting on the load balancer. To use SSL termination, replace the 'bind' entry with:

```
bind :443
```

SSL pass-through does not require a certificate to be specified, because HTTPS traffic will be sent along as-is by the load balancer to the back-end ETX Server. In SSL pass-through mode, HAProxy cannot enable many layer-7 load balancer features (such as URL-based forwarding) which require decryption and analysis of the incoming web traffic.

To configure SSL termination, you will need to create the `myhost.pem` file which is referenced in the bind command. In this example we will create a self-signed certificate using the `openssl` utility, which is part of most Linux distributions:

```
openssl req -x509 -nodes -newkey rsa:4096 -keyout myhost.key -out \ myhost.crt
-days 365
```

If the `openssl` command is not recognized, you may need to install it from repo, for example using `yum install openssl`.

This command will prompt you to enter information for the certificate. Be sure to specify the load balancer hostname as the Common Name for the certificate (shown in yellow below):

```
Generating a 4096-bit RSA private key
.............................................++
...................++
writing new private key to 'myhost.key'
-----
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:CA
State or Province Name (full name) []:ON
Locality Name (eg, city) [Default City]: Toronto
Organization Name (eg, company) [Default Company Ltd]: OpenText
Organizational Unit Name (eg, section) []:
```

```
Common Name (e.g. your server's hostname)[]: my.loadbalancer.url.com
Email Address []:
```

Once the key and cert files are created, concatenate them to create a .pem file:

```
cat myhost.crt myhost.key | tee myhost.pem
```

Finally, copy `myhost.pem` to the directory specified in the `haproxy.cfg` file:

```
cp myhost.pem /etc/ssl/
```

Step 6: Server health check script

The health check script is invoked by HAProxy to check the state of the back-end ETX servers that are used to serve web sessions to users. The health check script determines whether to distribute incoming web sessions to the host. HAProxy will periodically invoke the script using the following parameters:

```
command <proxy-ip> <proxy-port> <server-ip> <server-port>
```

The `<server-ip>` and `<server-port>` parameters ($3 and $4 in the script) provide the URL for the back-end server to verify. The health check script must exit with a return code of 0 if the back-end server is available and exit with a non-zero code to indicate that the server is unavailable.

The following configuration in `haproxy.cfg` specify the settings for the health check script:

```
external-check     (in the global section)
external-check command /bin/haproxy/etxstat.sh (in the backend section)
external-check path "/usr/bin:/bin" (in the backend section)
```

`external-check` in the `global` section tells HAProxy to allow external-check commands to be used in the `backend` section. Since script execution can be potentially dangerous, this feature is disabled by default.

`external-check command` specifies the path to the health check command.

`external-check path` specifies the PATH environment variable that is set prior to executing the `external-check command`. Since our command references the `curl` command (located in /usr/bin), we will set the PATH to include /usr/bin.

Below is a simplified health check script that we will use:

```
#!/bin/bash
status=$(curl -s --user username:password http://$3:$4/etx/state)
if [ "$status" = "RUNNING" ]; then
   exit 0
else
```

```
    exit 1
fi
```

ETX Server provides a special REST API (/etx/state) which is not part of the normal REST API set. This API indicates if the server is available for launching new web sessions. If the ETX Server is in MAINTENANCE mode or is not running, this script will return a failure (`exit 1`) and HAProxy will not direct new sessions to the server. This script ensures that sessions are not sent to ETX Servers which are unavailable due to application-level issues, even though the servers are responding to TCP requests.

NOTE: curl must be installed on the load balancer system for this script to work. curl can be installed from most repos (for example: `yum install curl`).

NOTE: You may specify a username and password in the script or switch the command to use API key authentication. Information about REST API authentication methods is provided in the *Exceed TurboX REST API Users Guide*.

After creating the script file, you will need to make it executable:

```
chmod a+x /bin/haproxy/etxstat.sh
```

You should also verify that the `haproxy` user (which HAProxy runs as) can execute the script:

```
sudo -u haproxy /bin/haproxy/etxstat.sh
```

Step 7: Finalize configuration and start haproxy

When you have finished modifying your HAProxy configuration file, creating your health check script, and adding the necessary HTTPS certificate file, you should verify that the haproxy.cfg file is valid using the following command.

```
haproxy -c -V -f /etc/haproxy/haproxy.cfg
```

If everything is properly configured, you should see the message:

```
Configuration file is valid.
```

If you see an error message, take any steps necessary to address the reported issues.

The final step is to start haproxy. In step 3 `haproxy` was configured as a service, so you can start it using:

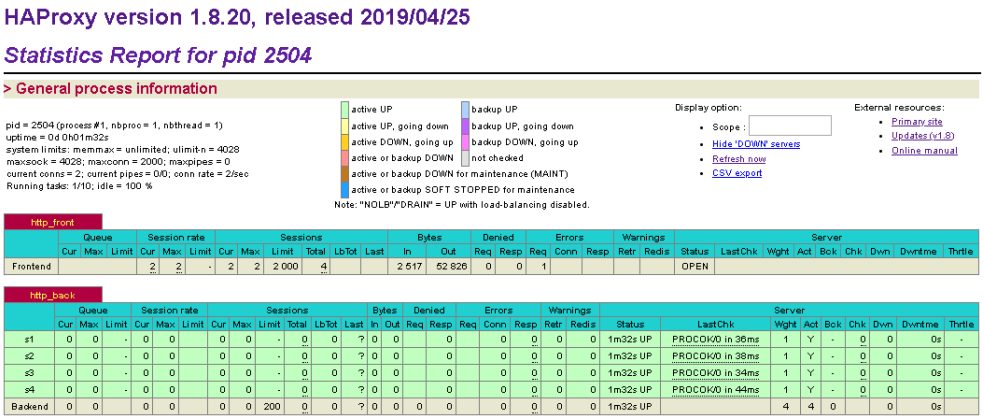
```
service haproxy start
```

Step 8: Verify the environment

To verify that `haproxy` is running correctly, open the stats URI in your browser:

```
http[s]://<load_balancer_hostname_or_ip>/haproxy?stats
```

You should see a statistics page similar to the following:



HAProxy version 1.8.20, released 2019/04/25

Statistics Report for pid 2504

> General process information

pid = 2504 (process #1, nbproc = 1, nbthread = 1)
uptime = 0d 0h01m32s
system limits: memmax = unlimited; ulimit-n = 4028
maxsock = 4028; maxconn = 2000; maxpipes = 0
current conns = 2; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/10; idle = 100 %

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.8)
- Online manual

http_front

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				2	2	-	2	2	2 000	4			2 517	52 826	0	0	1					OPEN								

http_back

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
s1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	1m32s UP	PROCOK/0 in 36ms	1	Y	-	0	0	0s	-
s2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	1m32s UP	PROCOK/0 in 38ms	1	Y	-	0	0	0s	-
s3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	1m32s UP	PROCOK/0 in 34ms	1	Y	-	0	0	0s	-
s4	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	1m32s UP	PROCOK/0 in 44ms	1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	200	0	0	?	0	0	0	0		0	0	0	0	1m32s UP		4	4	0		0	0s	

Figure 7 - HAProxy statistics page

The green rows for s1 to s4 indicate that the back-end ETX servers are up and running. If the rows are red, it means there may be a configuration issue, or the server is not running.

If you are not sure why one or all servers is showing up in red, try disabling certain features by commenting out sections of the haproxy.cfg file, by adding the hash symbol (#) at the start of each line, until you identify the problem.

Finally, to check that the load balancer is working and directing traffic to your back-end ETX servers, type the load balancer URL into your browser:

```
http[s]://<load_balancer_hostname_or_ip>/etx/
```

You should see the ETX sign-in page displayed in your browser:

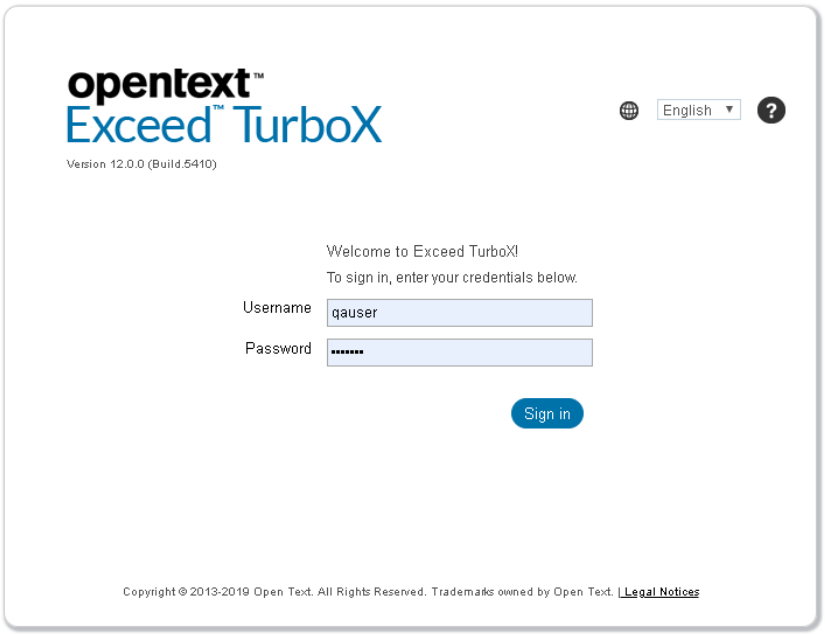


Figure 8 - ETX sign-in page

If you refresh the `/haproxy?stats` URL you should now see that there is a web session assigned to one of your back-end web servers. Try loading more pages in different browsers or devices and watch how web sessions are distributed to back-end servers in a round-robin fashion.

Finally, take a server offline (using `bin/etxsvr stop`) and verify that the HAProxy stats page shows the server as disconnected (in red). Then re-start the server and it should turn green, indicating that it can accept sessions again.

Further documentation for HAProxy can be found online at http://www.haproxy.org/#docs. You may wish to configure other features, such as reporting tools that use syslog to send email notifications of server failures reported by haproxy.

About OpenText

OpenText enables the digital world, creating a better way for organizations to work with information, on premises or in the cloud. For more information about OpenText (NASDAQ: OTEX, TSX: OTC) visit opentext.com.

Connect with us:

OpenText CEO Mark Barrenechea's blog

Twitter | LinkedIn