

VOLT

ACTIVE DATA

Release Notes

Product	Volt Active Data
Version	11.4.3
	VoltDB Operator 1.8.3
	VoltDB Helm Chart 1.8.3
Release Date	November 3, 2022

This document provides information about known issues and limitations to the current release of VoltDB. If you encounter any problems not listed below, please be sure to report them to support@voltageactive.com. Thank you.

Upgrading From Older Versions

The process for upgrading from the recent versions of VoltDB is as follows:

1. Shutdown the database, creating a final snapshot (using **voltadmin shutdown --save**).
2. Upgrade the VoltDB software.
3. Restart the database (using **voltadmin start**).

For Kubernetes, see the section on "Upgrading the VoltDB Software and Helm Charts" in the *VoltDB Kubernetes Administrator's Guide*. For DR clusters, see the section on "Upgrading VoltDB Software" in the *VoltDB Administrator's Guide* for special considerations related to DR upgrades. If you are upgrading from versions before V6.8, see the section on "Upgrading Older Versions of VoltDB Manually" in the same manual.

Finally, for all customers upgrading from earlier versions of VoltDB, please be sure to read the upgrade notes for your current and subsequent releases, including V6, V7, V8, and V10.

Changes Since the Last Release

Users of previous versions of VoltDB should take note of the following changes that might impact their existing applications. See the *VoltDB Operator Release Notes* for changes specific to the use of VoltDB on the Kubernetes platform.

1. Release V11.4.3 (November 3, 2022)

1.1. Security updates

Various packages within Volt Active Data have been updated to eliminate known security vulnerabilities, including:

- CVE-2020-26160
- CVE-2021-38561
- CVE-2021-38561
- CVE-2022-1996
- CVE-2022-21698
- CVE-2022-23307

CVE-2022-32149

CVE-2022-42003

1.2. Additional improvements

The following limitations in previous versions have been resolved:

- The DCR round trip time statistics (that is, the columns starting *DR_ROUNDTRIPTIME_* in the `@Statistics DRPRODUCER` selector results) intermittently reported excessively high and incorrect latency values. This issue has been resolved.
- The HTTP export connector has been improved to cancel all pending export messages if the connection to the export target times out. This allows the connection to be reset and the blocked requests to be resubmitted.
- Previously, if the cluster encountered corrupted command log files during restart it could result in the nodes repeatedly reporting remote hangups and a missing partition list. This issue has been resolved and the server now correctly reports a failure due to corrupted command logs.

2. Release V11.4.2 (September 8, 2022)

2.1. Help for the Prometheus agent

The Volt Prometheus agent provides metrics concerning the database performance and status in a format appropriate for the Prometheus monitoring system through the URL `http://server:port/metrics`. The agent now provides another endpoint, `http://server:port/help`, for displaying information about which metrics the agent supports. By default, the agent displays formatted information about the metric names, description and available labels for filtering. You can use the `/help?format=text` query string to receive a simple alphabetical list of metric names only.

2.2. Additional improvement

The following limitation in previous versions has been resolved:

- Previously, requests for DR statistics (through the `@Statistics` system procedure) would wait for current data before returning to the caller. This could result in variable latency on the `@Statistics` call, which could cause problems, in particular for the Volt Operator for Kubernetes. To avoid this, the DR statistics are now collected asynchronously and calls to `@Statistics` for DR information returns immediately with the most recent data collected.

3. Release V11.4.1 (August 11, 2022)

3.1. General release of the Helm voltadmin plugin

The Helm plugin **voltadmin** is now ready for production use. The plugin simplifies the management of VoltDB clusters in Kubernetes by replicating commands, such as **dr reset**, **pause**, **resume**, and **shutdown**, found in **voltadmin** on non-Kubernetes systems.

The Helm plugin is a separate installation from the standard VoltDB Helm charts. It requires the same versions of Python, Helm, and Kubernetes as the Volt software; plus an additional Python module, *click* (version 8.0.1 or later). To install the plugin:

1. Install the Python *click* module. The easiest way to do this is using **pip**:

```
$ python3 -m pip install click
```

If you do not already have **pip** installed, you may need to install it first. And if you do not want to install *click* for all users, you can use a virtual environment with **virtualenv**.

2. Do a directory of the repository for Volt Active Data charts and plugins (<https://storage.googleapis.com/voltdb-kubernetes-charts/>) and search for "voltadmin". Select the most recent version and install it using the **helm plugin install** command, specifying the URL of the plugin you want. For example, the following command installs the current version, 1.4.2:

```
$ helm plugin install \
https://storage.googleapis.com/voltdb-kubernetes-charts/voltadmin-1.4.2.tgz
```

3. Once installed, you can use the voltadmin command directly on the helm command line. Use the `--help` flag to get a list of the commands and arguments you can use. For example:

```
$ helm voltadmin --help
```

3.2. Production support for Java 17

Java 17 is now fully tested and supported for production use with Volt Active Data.

3.3. Additional statistics for tracking communication between XDCR clusters

Several additional columns have been added to the first results table for the `@Statistics DRPRODUCER` selector (and the corresponding Prometheus agent metrics) to help evaluate the time between when binary logs are ready for transmission and when acknowledgement is received from the consumer. The new columns are the following and are reported in milliseconds:

- `DR_ROUNDTRIPTIME_1MINUTE_MAX`: The maximum time it took to receive acknowledgement from the consumer, over the past minute.
- `DR_ROUNDTRIPTIME_1MINUTE_AVG`: The average time it took to receive acknowledgement from the consumer, over the past minute.
- `DR_ROUNDTRIPTIME_5MINUTE_MAX`: The maximum time it took to receive acknowledgement from the consumer, over the past five minutes.
- `DR_ROUNDTRIPTIME_5MINUTE_AVG`: The average time it took to receive acknowledgement from the consumer, over the past five minutes.

The corresponding metrics in the Prometheus agent are:

- `replication_roundtrip_time_1m_max`
- `replication_roundtrip_time_1m_avg`
- `replication_roundtrip_time_5m_max`
- `replication_roundtrip_time_5m_avg`

3.4. Sample dashboards for Grafana

Grafana is a popular solution for graphing and displaying information collected by Prometheus. A major benefit of using Prometheus and Grafana together is that you can integrate metrics from multiple applications to create a custom dashboard for your business. To get you started, Volt provides a set of dashboards offering different views of the database cluster's performance, status, and features. The dashboards are available in Grafana JSON format from a public Github repository:

```
https://github.com/VoltDB/volt-monitoring/
```

You can copy and install the individual dashboards manually, include them via the Grafana REST API, or use any of the other methods for installing dashboards. See the Grafana documentation and the README file for the Volt dashboards for more information.

3.5. Security update

The reload4j library has been updated to eliminate a known security vulnerability (CVE).

3.6. Additional improvements

The following limitations in previous versions have been resolved:

- The `voltadmin` command has a feature where if you specify a username but no password, it prompts for the password. However, starting with V11.0, this feature was broken and instead you received an error stating that the "object is not callable." This issue has been resolved.
- When certain errors interrupt communication between XDCR clusters, a **`voltadmin dr reset`** command could hang and never complete. A timeout has been added to allow the DR RESET operation to complete.
- There was an issue where, if a stored procedure queued more than 200 SQL statements before calling `volatileExecuteSQL()` and at least one of the statements was a SELECT statement that returned data, the result buffer could become corrupted causing one or more nodes to crash. This issue has been resolved.
- The timeout period associated with export block operations has been extended to avoid erroneously timing out operations for slower export targets, such as JDBC.
- The statistics and warning messages related to "missing" export data (that is, rows that have not been exported but are not currently available in the export buffers) have been significantly improved to provide a more accurate view of the actual state of export. Previously, under certain conditions, the statistics on missing rows could be misleading due to overcounting.
- There was an issue where if an export stream was dropped and recreated, then the database was immediately shutdown and restored, the newly created export stream would have an inaccurate pointer (associated with its previous incarnation). The consequence of this problem was that any records subsequently inserted into the export source were never written to the associated target. This issue has been resolved.
- When enabling security in VoltDB on Kubernetes, the Helm chart automatically added the administrator account specified for the operator `auth.username`. However, in V11.4, if the account was also specified in the cluster configuration, the cluster would fail to start due to duplicate definitions. This issue has been resolved and only one definition is included, whether specified explicitly or automatically.

4. Release V11.4 (June 27, 2022)

4.1. Compound procedures

Volt V11.4 introduces a new type of procedure: the *compound procedure*. Compound procedures let you call multiple transactional procedures, process results, define error handling and include other business processing within a single procedure that can be invoked with a single call. This is particularly useful as a target for topics, where each record invokes a single procedure. Compound procedures expand the capabilities of topic processing providing almost limitless flexibility.

At the same time, it is important to recognize that compound procedures are *not* transactional. Although each regular stored procedure invoked by the compound procedure will complete or rollback as a unit, the compound procedure itself does not and must provide its own error handling and recovery. For more information, see the chapter on Using Compound Procedures in the *VoltDB Performance and Customization Guide*.

4.2. Topics now support tables as well as streams

This release extends Volt topics to include support for input from tables, integrating with many existing table features — including migration, time to live (TTL) and change data capture. Specifically, the latest version includes the following features:

- **EXPORT TO TOPIC in CREATE TABLE** — You can now export data from tables to topics the same way you can direct output to an export target. The EXPORT TO TOPIC clause supports the same features as EXPORT TO TARGET, including the use of the ON clause for change data capture.
- **MIGRATE TO TOPIC and USING TTL in CREATE TABLE** — You can migrate data via topics when executing time-to-live (TTL) or explicitly migrating data with the MIGRATE statement.

These features are described in more detail in the CREATE TABLE reference pages in the *Using VoltDB* guide.

4.3. Stream views now support time-to-live (TTL) and migration

Stream views, which are computed based on ephemeral data passing through streams rather than persistent data in tables, can be managed by manually updating and deleting records from the view. You can now automate the management process with the USING TTL clause for scheduling the deletion of records based on a column in the view. You can also use the MIGRATE TO TARGET and MIGRATE TO TOPIC clauses for archiving view records before they are deleted. See the description of the CREATE VIEW statement in the *Using VoltDB* guide for more details.

4.4. New ALTER VIEW statement

Now that you can automate the management of stream views with the USING TTL clause, an ALTER VIEW ALTER USING TTL... statement has been added so you can modify certain aspects of the view without deleting it. The ALTER VIEW statement lets you modify the TTL value, the batch size, and the frequency of TTL processing. See the description of ALTER VIEW in the *Using VoltDB* guide for details.

4.5. Additional statistics

Several different categories of statistics have been added to the @Statistics and @SystemInformation system procedures. The new selectors for the @Statistics system procedure are the following:

- **@Statistics CLOCKSKEW** returns information about the time difference between the system clocks of servers in the cluster. The clock skew statistics are gathered periodically and the system procedure returns the most recent results. You can use the <clockskew> element in the configuration file to control how frequently clock skew is checked.
- **@Statistics COMPOUNDPROC and COMPOUNDPROC SUMMARY** return information on compound procedures similar to what PROCEDURE and PROCEDUREPROFILE return for regular stored procedures.
- **@Statistics COMPOUNDPROCCALLS** returns information about what stored procedures each compound procedure calls and how frequently.
- **@Statistics DRCONFLICTS** returns information about what data replication (DR) conflicts occurred and how many of each type.
- **@Statistics LIMITS** returns information about connection and file descriptor limits and the current number of active connections for each server.

The new data provided by the @SystemInformation includes the following:

- **@SystemInformation DEPLOYMENT: priorititesenabled** — specifies whether queue prioritization is enabled or not.
- **@SystemInformation DEPLOYMENT: drpriority** — if queue prioritization is enabled, the priority level for DR processing.

- **@SystemInformation DEPLOYMENT: drrole** — specifies the current DR role of the cluster (master, replica, xdcr, or none).
- **@SystemInformation OVERVIEW: HOSTSTARTTIME** — the time when the current server process started (as opposed to STARTTIME, which is when the current cluster instance started).

See the descriptions of @Statistics and @SystemInformation in the *Using VoltDB* guide for more information.

4.6. Improved Prometheus support

The data available from the VoltDB Prometheus agent has been significantly extended and rationalized. Specifically:

- Many of the values added to the @Statistics selectors mentioned above are now available through Prometheus.
- Similarly, several of the data points concerning the status of the cluster and database provided by @SystemInformation (such as cluster size and license expiration date) are also accessible through the Prometheus agent.
- The values returned by the system procedures have been adjusted for consistency wherever possible. For example, use of the procedure name rather than the class path when referring to stored procedures has been standardized.

Additional information about the Prometheus agent and what data is available through it will be provided in upcoming releases.

4.7. Support for Rocky Linux

Rocky Linux version 8.0 has been added as a supported base platform for VoltDB. Rocky Linux is a replacement for CentOS, which has reached end of life and is no longer supported by Red Hat, its sponsor.

4.8. Support for Kubernetes 1.23.0

The VoltDB Operator and Helm charts now support Kubernetes 1.23.0.

4.9. Ability to choose which Java Garbage Collector to use

VoltDB provides a new environment variable that you can use to select which Java garbage collector the server process uses. By defining VOLTDB_GC_OPTS before starting the server, you can specify the Java arguments necessary for selecting a garbage collector and setting any related properties it needs. For example, the following selects the G1 garbage collector and sets string deduplication:

```
$ export VOLTDB_GC_OPTS="-XX+useG1GC -XX+UseStringDeduplication"  
$ voltdb start --dir=~ /mydb -H serverA
```

Note that the garbage collector has a significant impact on memory-intensive applications like VoltDB. Also, VoltDB does not validate the correctness of the arguments you specify. So this option should be used with great caution. See the appendix on server configuration options in the *VoltDB Administrator's Guide* for more details.

4.10. New function TIME_WINDOW()

There is a new function, TIME_WINDOW(), that defines time slices (such as every 15 minutes) that you can use for windowing applications. TIME_WINDOW() can be particularly useful when used in conjunction with stream views. For example, the following example tracks the number of stock events (buys and sells) in five minute "windows" so activity can be assessed without having to keep all of the underlying transaction data.

```
CREATE VIEW volatility (total, event_type, window)
  AS SELECT count(*), buy_or_sell, TIME_WINDOW(MINUTE, 5, trx_time)
     FROM transactions
     GROUP BY buy_or_sell, TIME_WINDOW(MINUTE, 5, trx_time);
```

4.11. New function DATEDIFF()

The new DATEDIFF() function provides the difference between two timestamps in the requested units. The result of the function is an integer value. For example, the following function returns the number of whole minutes between the login and logout timestamps:

```
INSERT INTO billing (account_id, minutes)
  SELECT user, DATEDIFF(MINUTE, login, logout)
     FROM session;
```

4.12. Change to CREATE PROCEDURE statement for DIRECTED and COMPOUND procedures

With the addition of compound procedures, the syntax for the CREATE PROCEDURE statement has been modified. Previously, the syntax for creating a directed procedure was CREATE PROCEDURE DIRECTED. The syntax has been updated to be grammatically correct, putting the adjective first. The new statements are CREATE COMPOUND PROCEDURE and CREATE DIRECTED PROCEDURE.

4.13. @SwapTables no longer allowed on active XDCR clusters

The @SwapTables system procedure acts like a schema change, which on cross-datacenter replication (XD-CR) clusters requires that the cluster be paused before changing the schema. However, previously the system procedure did not enforce this requirement. Now @SwapTables returns an error when the system procedure is invoked if the XDCR cluster is not paused.

4.14. Deprecating obsolete features

As Volt Active Data evolves, certain features are replaced or become obsolete. When this happens, they are deprecated, which means their use is no longer recommended and they are removed from the product in a future release. The following features are now deprecated.

- CentOS is still supported as a base platform. However, it has reached end-of-life and will not be supported in the future by VoltDB. Rocky Linux is the recommended replacement for CentOS in production environments.
- The process for upgrading VoltDB online using limited hardware is being deprecated, along with the **voltadmin plan upgrade** command. The process is error prone because it reduces the K-safety of the cluster, which is not recommended for production systems. The recommended upgrade process is either an online upgrade using two XDCR clusters or using a snapshot to shutdown, upgrade, and restore.
- The REPLICATIONROLE return value from the @SystemInformation OVERVIEW call is now deprecated. REPLICATIONROLE applies to passive DR only and duplicates information available from the more complete @Statistics DRROLE procedure call and the DRROLE return value of the @SystemInformation DEPLOYMENT procedure call.

4.15. Security update

VoltDB does not use any of the Log4J components implicated in the published CVEs. However, to avoid any confusion, VoltDB has replaced the Log4J library with reload4J, a drop-in replacement that replicates the log4J namespace and functionality, but eliminates all known security vulnerabilities.

4.16. Additional improvements

The following limitations in previous versions have been resolved:

- VoltDB uses a special prefix, `VOLTDB_AUTOGEN`, for indexes that are not explicitly named in the `CREATE TABLE` statement. Previously, if a user defined an index explicitly using the `VOLTDB_AUTOGEN` prefix in an index name, the `CREATE TABLE` statement would succeed. However, any subsequent attempts to modify the schema in any way would fail. This issue has been resolved.
- Previously, if Kubernetes pods were started with IPv6 disabled, the VoltDB Operator did not detect it and the database failed to start when it tried using IPv6. The operator now recognizes this situation and acts accordingly. The issue no longer exists.
- There was a race condition discovered where, if a node was starting to rejoin the cluster just as a schema change was initiated, that the cluster could report a hash mismatch because a transaction was planned against two different versions of the schema. This issue has been resolved.
- It was possible for a call to the `@Statistics` system procedure with the `PROCEDUREPROFILE` selector to return invalid values for minimum and maximum if the call coincided with the first instances of a procedure being executed. This race condition has been resolved.
- There was an issue when using Kubernetes with TLS/SSL enabled and custom credentials for the metrics pod, where the pod would not start and report the error "CreateContainerConfigError". The VoltDB Operator now ensures the configuration is complete to ensure TLS/SSL starts properly for the metrics pod.
- In previous releases, frequent client connection attempts could result in excessive messages in the log file, although the messages were meant to be limited to one every 60 seconds. This issue has been resolved and the rapidly repeated messages are now muted.
- The `@Statistics` system procedure takes two arguments: a selector and a delta flag. If the delta flag is non-zero, for certain selectors the procedure returns results since the last call to `@Statistics` for that selector class. However, the list of affected selectors in the documentation was wrong; the `QUEUE` and `TTL` selectors always return total, not incremental results. The documentation has been corrected.
- Recently, issues have surfaced related to the use of replicated tables in database replication (DR) where certain conditions can cause DR processing to stop consuming data. When this happens the console and log report that "no new DR transactions have been processed." In one case, if a replicated table is defined with the `MIGRATE TO TARGET` clause, migrating rows can cause an error in the multi-partition initiator, which subsequently stalls DR traffic. In another case, a race condition while processing multiple multi-partition procedures in a row followed by a partitioned procedure could also trigger a failure in DR. These issues have been resolved.
- In Operator and Chart versions 1.6.x, the Helm properties for specifying the public IP address and port numbers for database replication (DR) were changed from two properties to a single property. However, the new property did not set the public IP address when using nodeports. As a result, XDCR connections could not be established in certain configurations. This issue has been resolved.
- The Helm properties for configuring XDCR in Kubernetes have been enhanced to allow Kubernetes to automatically assign port numbers for the external replication ports when using nodeports. By setting the `cluster.serviceSpec.perpod.dr.StartReplicationNodePort` property to `-1` the public replication ports are assigned automatically using available ports.
- Previously, the **voltadmin release** command did not always release export on all partitions within the cluster. This issue has been resolved.
- In previous releases, restarting a database with lots of export connectors could take a significant amount of time. And the delay was particularly noticeable if the connectors had fallen behind, leaving large numbers of files in the export overflow directory. The startup process (as well as the contents of the `export_overflow`

directory) have been restructured to dramatically reduce the time required to validate these files and thereby speed up the database startup itself. Also, the log messages related to export startup have been streamlined and rewritten to be less intrusive and more informational.

- There was a problem where, if a properties file in the database root was corrupted, the database would issue a fatal error with no explanation. The error now identifies the corrupted file and the names of the missing properties.
- Previously, the database would periodically report an error indicating that a VoltPort had "died". As drastic as it sounds, the message did not indicate a serious problem (just that a connection had been closed) and was usually followed shortly by the client reconnecting. Therefore, the message has been downgraded to a warning and rewritten to more accurately reflect that a connection closed unexpectedly.

5. Release V11.3.2 (June 15, 2022)

5.1. Recent improvements

The following limitations in previous versions have been resolved:

- Previously, when stopping a node on a K-safe cluster in Kubernetes (using the `cluster.clusterSpec.stoppedNodes` property), if the stop node operation failed for any reason, the VoltDB Operator could get stuck in an endless loop retrying. This issue has been resolved.
- VoltDB Operator and Helm chart improvements:
 - V1.7.4 — The Docker image for Kubernetes has been updated with new versions of Alpine (3.16.0) and Adopt OpenJDK (11.0.15+10) to correct security vulnerabilities.
 - V1.7.5 — In recent releases, specifying the public IP address and port numbers for database replication (DR) was changed from two properties to a single property. However, the new property did not set the public IP address when using nodeports. This issue has been resolved.
 - V1.7.5 — The Helm `cluster.serviceSpec.perpod.dr.StartReplicationNodePort` property has been extended to allow Kubernetes to automatically assign port numbers for the external replication ports when using nodeports by setting the property to -1.

6. Release V11.3.1 (April 21, 2022)

6.1. Welcome to Volt Active Data

VoltDB, the company, is now Volt Active Data. The in-memory database VoltDB continues to be the foundation of our product offering. This software release updates messages and other references to provide the most accurate and up-to-date access to product resources.

6.2. Volt Active Data monitoring in DataDog

Our partner, DataDog, offers monitoring services that let you combine metrics from multiple products into a single view of your business systems. Their support for Volt Active Data was recently improved and expanded. For more information, see their original blog post announcing Volt support or visit their website.

6.3. VoltDB Operator and Helm Charts updates

The VoltDB Operator and Helm charts for this release of Volt Active Data have been updated twice:

- V1.7.2 (May 12, 2022) — When starting XDCR using `nodePorts` in earlier V1.7.x releases, the external IP addresses were not always being passed correctly and, as a result, the `nodePort` was not properly initiated. This issue has been resolved.

- V1.7.3 (May 19, 2022) — After dropping a cluster from the XDCR mesh using the `cluster.clusterSpec.dr.excludeClusters` property and resetting the property, any subsequent attempt to drop a cluster would fail. This issue has been resolved.

6.4. Additional improvements

The following limitations in previous versions have been resolved:

- Previously, changing the property `cluster.config.deployment.dr.connection.enabled` from true to false would cause the cluster to restart unnecessarily. This issue has been resolved.
- There was a problem in previous releases where restarting a cluster with large volumes of unprocessed export and topic data could fail with I/O errors from too many open files. This only occurred in extreme cases — hundreds of export connectors or topics with literally thousands of overflow files due to their targets being down prior to the database stopping. This issue has been resolved.

7. Release V11.3 (February 8, 2022)

7.1. Beta Support for Java 17

VoltDB is now beta testing the use of Java 17 for running VoltDB. This means you can use Java 8, 11, or 17 to run both the VoltDB server product and client API. Additional testing and validation is ongoing and Java 17 is expected to be ready for general release (and production use) in V11.4.

7.2. The new Java Client2 API is ready for production

A new Java client interface, Client2, was introduced as a beta feature in VoltDB V11.2. Testing is complete and this API is now generally available for production use. See the Client2 chapter in the *VoltDB Performance and Customization Guide* for an introduction to the interface and the Javadoc for details.

7.3. Prioritizing transactions

One feature of the new Client2 interface is the ability to specify a priority for individual procedure calls, which is now supported by the server as well. Priorities increase the frequency of high priority transactions running versus low priority transactions. You can specify a priority when calling a stored procedure, or you can specify a priority for classes of transactions, such as snapshot and database replication (DR) in the configuration file when you enable prioritization. See the chapter on Prioritizing Transactions in the *VoltDB Performance and Customization Guide* for information on enabling and specifying priorities.

7.4. Protection against mutable parameters in K-safe clusters

It is possible to pass mutable parameters most notably arrays to stored procedures. If you modify those parameters before passing them to a SQL statement, there is no guarantee that the results will be deterministic on all copies of the partition. Consequences can be unpredictable results, node crashes, or even data corruption.

Warning

To avoid non-deterministic behavior, VoltDB now makes a copy of the mutable parameters before executing the statement. However, depending on the size of the arrays and the frequency of the calls, copying the parameters can consume large amounts of memory and could have *a significant impact on performance*.

If you are sure you do not modify such arrays in your stored procedures, you can tell VoltDB to not copy mutable parameters in the configuration file. See the appendix on Server Configuration Options in the *VoltDB Administrator's Guide* for details.

7.5. Support for Kubernetes 22.0

The VoltDB Operator and Helm charts now support Kubernetes V1.19.x through V22.x.

7.6. VoltDB Management Center improvements

A number of functional improvements have been made to the VoltDB Management Center (VMC), including:

- Ability to enable and disable security in VMC
- Improved user management: adding and modifying users, assigning multiple roles, and support for user-defined roles
- Execution of stored procedures in the SQL Query tab

7.7. Additional improvements

The following limitations in previous versions have been resolved:

- There was an issue where an attempt to modify specific export characteristics of a table with `ALTER TABLE... ALTER EXPORT... ON UPDATE_NEW` would result in a bad table definition in the schema that could no longer be modified. This issue has been resolved.
- In V11.2, if the Client2 API had no connections available for a call, it reported the error as part of the ClientResponse. It now returns this information as a Java exception to make error handling more consistent.
- Previously, if an appropriate index did not exist when creating a table with `CREATE TABLE ... MIGRATE TO TARGET`, migration would not be established, but no error was reported. A warning is now issued to alert the user to the situation.
- There was an issue where attempting to call the `@Note` system procedure using the HTTP interface would result in a null pointer exception. This issue has been resolved.
- There was an unusual edge case where if a database with a large number of tables was left idle for an extended period of time, memory allocation would slowly increase until a node could crash. This condition required hundreds or thousands of tables with no activity at all. Any transaction or update would reset the memory usage. This issue has now been resolved.

8. Release V11.2 (October 28, 2021)

This release focuses on improving the management of resources associated with running VoltDB in production. In particular, several new features allow you to set a retention policy for logs, snapshots, and other files created by the database system. Several of these features have new default values to reduce the proliferation of unnecessary files, as well as options to adjust the policy to your business needs.

8.1. Retention policy for database log files

VoltDB writes informational, warning, and error messages to log files in a subfolder of the database root directory. By default, the log files "roll", creating a new file and renaming the old ones, on a daily basis. Previously, old log files were never deleted. Starting with this release, the default Log4J configuration file has been revised and now retains up to 30 log files. Older files are deleted. You can customize the configuration to change either or both the number of files retained or how frequently a new file is created. See the section on "Managing VoltDB Log Files" in the *VoltDB Administrator's Guide* for details.

Note that if you use a customized Log4J configuration, your customization will take precedence and override the new behavior.

8.2. Retention policy for snapshots during initiation

By default, snapshots created manually or through periodic snapshots are stored in the *snapshots* folder in the database root directory. When you reinitialize the root directory, using the `--force` flag to delete existing data, all files are deleted *except* the snapshot directory, which is renamed *snapshots.1*, *snapshots.2*, etc.

Snapshots can be quite large, since they save the entire contents of the database. And previously, VoltDB saved and renamed all of the extant snapshot folders. To avoid consuming unnecessary disk space, the **voltadmin init --force** command now only saves the two most recent snapshot folders by default. You can specify a different retention value with the `--retain` flag on the **voltadmin init** command. See the description of the **voltadmin** command in the *Using VoltDB* guide for details.

8.3. Retention policy for final snapshots

When you shutdown the database with the **voltadmin shutdown --save** command, a final snapshot is created saving the database contents. Final snapshots are useful, particularly for saving and restoring the database when upgrading the VoltDB software. VoltDB does not delete the final snapshot after it is restored as a safety precaution, in case you decide the restored database has a problem or is not configured correctly. Saving the final snapshot gives you an opportunity to manually restore the database to another location in case of an emergency.

However, once the final snapshot is restored and operational, the snapshot is no longer needed. Consequently, VoltDB now deletes all but the last two final snapshots when you create a new snapshot with the **voltadmin shutdown --save** command. This is a fixed limit.

8.4. Retention policy for XDCR conflict logs

Cross datacenter replication (XDCR) records any conflicts that occur during active-active replication in the XDCR conflict logs. These logs give you the opportunity to review, analyze, and potentially correct any issues that arise from conflicting transactions. However, over time, subsequent transactions will make correcting old conflicts impractical.

How long you need to save the conflict logs is a business decision. Therefore, by default, VoltDB maintains the old behavior of keeping all conflict logs. (Except in the case of Kubernetes, where a default of 30 days is applied.) You can specify a retention policy in terms of how long to keep log files by adding the `conflictretention` attribute to the `<dr>` element in the configuration file:

```
<dr id="1" role="xdcr" conflictretention="14d" >
```

The retention period can be specified in seconds, minutes, hours, or days (using the signifier *s*, *m*, *h*, or *d*). Alternately, you can configure the `VOLTDDB_XDCR_CONFLICTS` export connector, which overrides the `conflictretention` attribute.

8.5. Retention policy for file export

Similarly to XDCR conflict logs, a retention policy can be applied to the output files of any file export connector. By default, all output files are retained. However, you can use the `retention` property of the connector `<configuration>` to specify a time limit, after which old files are deleted. For example, the following configuration retains export files for 14 days:

```
<export>
  <configuration enabled="true" target="log" type="file">
    <property name="retention">14d</property>
    <property name="type">csv</property>
    <property name="nonce">MyLogs</property>
  </configuration>
</export>
```

See the section on "The File Export Connector" in the *Using VoltDB* guide for details.

8.6. VoltDB now supports Kubernetes on OpenShift

The VoltDB Operator has added support for running VoltDB databases in Kubernetes on the OpenShift platform from Red Hat. This expands VoltDB's Kubernetes compatibility to three of the major cloud platforms: AWS, Google Cloud, and OpenShift. For those interested in using OpenShift, please be sure to read the operational note later in this document for specific instructions on handling Transparent Huge Pages (THP) on the OpenShift platform.

8.7. New Java client method for connecting to a list of potential hosts

The `setTopologyChangeAware` property in the VoltDB Java client takes care of maintaining connections to all nodes of the cluster, even if servers drop out or new servers are added. However, traditionally, it was the application's responsibility to explicitly create the initial connections. To simplify the process, the client API adds a new method, `createAnyConnection`, that lets you provide a list of servers and creates one connection to the first available server on the list, and then lets the topology awareness property establish the remaining connections based on the actual current topology of the cluster. See the Javadoc for more information.

8.8. Additional improvements

The following limitations in previous versions have been resolved:

- In certain cases, if a stream associated with a topic in a K-safe cluster was altered to modify or drop a column, the database would generate a hash mismatch forcing the cluster into reduced K-safety mode. This issue has been resolved.
- There was a rare condition where the VoltDB network process could report an index out of bounds error, causing the cluster to hang. This condition is now caught. As a consequence of the error, one of the nodes will stop, but the cluster as a whole will continue and not be deadlocked.

9. Release V11.1.1 (October 11, 2021)

9.1. Additional improvement

The following limitation in previous versions has been resolved:

- You can have multiple schema and class files loaded automatically when starting VoltDB. On the command line, this is done with the `--schema` and `--classes` arguments to the `voltldb init` command. In Kubernetes, by specifying the persistent volumes where the files are located and including the file `.loadorder` in each directory to specify an order for the files. However, in version 11, the processing of the command line interfered with the specified order and could result in the database failing to start. This issue has been resolved.

10. Release V11.1 (September 14, 2021)

10.1. New Java client API, Client2 (BETA)

VoltDB V11.1 includes the Beta release of a new Java client library. *Client2* provides a modern, robust and extensible interface to VoltDB for client applications written in Java. For example, Client2 uses `CompletableFuture` to return information from asynchronous procedure calls, providing a more structured way of handling both successful and unsuccessful calls. The new API will also be the main platform for new client development in the future, starting with the ability to assign a priority for individual requests. Client-side priority queuing is available in the beta release and allows the client to prioritize certain procedure calls over others. See the javadoc for details.

As with all beta releases, the new capabilities are believed to work as described. However, further testing and customer feedback is required before we can recommend its use for production. We encourage you to try the new client interface and provide feedback to support@voltageactive.com. Thank you.

10.2. New Helm plugin, voltadmin (BETA)

Helm and the VoltDB Operator make configuring and running VoltDB in Kubernetes easier. However, there are certain management operations that are awkward to perform using Helm properties alone. VoltDB provides a new Helm plugin, now in Beta release, to simplify these additional VoltDB management tasks.

The Helm **voltadmin** plugin provides commands, similar to regular voltadmin commands, to perform management activities on a running cluster. For its initial release the Helm voltadmin plugin supports the following commands:

- collect
- dr drop
- dr reset
- pause
- resume
- shutdown

Additional commands will be added in future releases. Once installed, you use the plugin on the Helm command line by specifying the plugin name (voltadmin), the name of the Helm release with the `--release` option, the command, and then any additional arguments. You can use `helm voltadmin --help` to get a list of all the commands and options. For example, the following command resets the XDCR connection between the cluster in Helm release *mydb* and the cluster with a DR ID of 1:

```
$ helm voltadmin --release=mydb dr reset --cluster=1
```

You can install the beta release of the voltadmin plugin with the following command:

```
$ helm plugin install https://storage.googleapis.com/voltdb-kubernetes-charts/voltadmin
```

10.3. New --credentials argument added to Prometheus agent

The Prometheus agent for VoltDB has a new argument available when starting the agent from the shell command. The `--credentials` argument lets you specify a text file containing the authentication credentials for accessing the database when security is enabled. The file must define two properties, `username` and `password`. For example:

```
$ cat $HOME/mycreds.txt
username: admin
password: mySpecialPassword
$ ./voltdb/tools/monitoring/prometheus/voltdb-prometheus \
  --credentials $HOME/mycreds.txt
```

Using the `--credentials` argument instead of `--username` and `--password` avoids exposing your credentials on the command line to the `ps` command. Note that the file path must be specified as a full pathname, not a relative path.

10.4. Support for reporting incremental statistics in Prometheus

By default, the VoltDB Prometheus agent reports statistics for all time — since the database started. You can now tell the agent to report statistics in increments, so that each timestamp reports on activity since the last data was reported. For example, if data is collected every 10 seconds, the statistics for transactions per second is the

average TPS over the last 10 seconds, rather than the average since the server started. To report incremental statistics, set the Helm property `metrics.delta=true` in Kubernetes or use the `--delta` option when starting the Prometheus agent:

```
$ ./voltdb/tools/monitoring/prometheus/voltdb-prometheus --delta
```

10.5. Ability to set DSCP priority for DR network traffic

VoltDB now lets you set a DSCP priority level for network packets associated with database replication (DR) by defining the environment variable `DD_SOCKET_TRAFFIC_CLASS` to the DSCP value you wish to use prior to starting the VoltDB process. For example, The following commands start the VoltDB server with the DR network traffic DSCP priority set to 3:

```
$ export VOLTDDB_OPTS="-DDR_SOCKET_TRAFFIC_CLASS=3"
$ voltdb start -D ~/mydb
```

For Kubernetes, the value can be set using the helm property `cluster.clusterSpec.env.VOLTDDB_OPTS`.

10.6. Support for Kubernetes 1.21

The latest VoltDB Operator and Helm charts add support for Kubernetes up to version 1.21. The officially supported versions are Kubernetes 1.16.2 through 1.21.x.

10.7. Additional improvements

The following limitations in previous versions have been resolved:

- VoltDB V9.1 changed how VoltTables are read to improve access by column name. However if only one or two columns are accessed from a large VoltTable, performance actually decreased. The current release adjusted the read access to optimize for all cases where columns are fetched by name.
- The list of servers shown in the VoltDB Management Center Admin tab provides a button for stopping individual nodes in a cluster. However, these buttons did not work consistently and the list itself was sporadically rearranged. Both of these issues have been resolved.
- In previous releases, the `voltdb` command did not always perform command line expansion when using the equals sign to assign a file path to an option. For example `--license ~/mylicense.xml` worked but `--license=~/mylicense.xml` did not. This issue has been resolved and file paths are handled consistently now.

11. Release V11.0 (August 12, 2021)

VoltDB 11 is a major release incorporating features from recent updates plus new capabilities. It includes the following major new features and improvements.

11.1. VoltDB Topics

VoltDB Topics provide the intelligent streaming of VoltDB's existing import and export capabilities, but with the flexibility of Kafka-like streams. Topics allow for both inbound and outbound streaming to multiple client producers and consumers. They also use the existing Kafka interface to simplify integration into existing infrastructure. But most importantly, they allow for intelligent processing and manipulation of the data as it passes through the pipeline.

VoltDB topics, which were released as a beta feature in V10.2, are now ready for production use. See the chapter on Streaming Data in the *Using VoltDB* manual for more information.

11.2. Support for Python 3.6

The VoltDB command line tools have been upgraded to use Python version 3. Python 3 is commonly available on modern operating systems and so simplifies the process of configuring platforms for VoltDB.

11.3. New Kubernetes capabilities

Upgrades to the VoltDB Operator for Kubernetes provide two important new features:

- **Multi-cluster XDCR** — It is now possible to create a cross datacenter replication (XDCR) network with three or more clusters. Additional Helm properties help simplify the management and maintenance of the XDCR clusters. See the chapter on Database Replication in the *VoltDB Kubernetes Administrator's Guide* for details.
- **VoltDB Topic Support** — The Operator now provides the properties necessary to configure and start VoltDB topics in clusters running in Kubernetes.

11.4. VoltDB Java Client improvements

The VoltDB Java client interface has been updated with the following improvements:

- **Topology Awareness** — Previously, there were two options for handling topology changes on the server, `setReconnectOnConnectionLoss()` and `setTopologyChangeAware()`, which were mutually exclusive. This limitation has been removed and `setTopologyChangeAware()` has been enhanced to include reconnection when the last connection is lost, further improving connectivity and resilience.
- **Non-Blocking Asynchronous Calls** — Normally, the asynchronous `callProcedure` method returns an error if the client cannot queue the call because of backpressure. However, it is still possible for the call to block in certain cases. It is now possible to avoid blocking entirely by setting the `setNonblockingAsync()` configuration option on the client. See the Javadoc for details.
- **Connection timeouts** — Handling of connection timeouts by the client has been improved. Now, if the client is able to detect a timeout before it is sent to the server, the client aborts the transaction and returns the procedure status `GRACEFUL_FAILURE`, with a status string of "Procedure call not queued: timed out waiting for host connection."

11.5. Security updates

The VoltDB Management Center (VMC) web-based console for VoltDB has been updated to the latest versions of the jQuery libraries to address security vulnerabilities. The current library versions for VMC are JQuery v3.5.1 and jQuery-UI v1.12.1.

Known Limitations

The following are known limitations to the current release of VoltDB. Workarounds are suggested where applicable. However, it is important to note that these limitations are considered temporary and are likely to be corrected in future releases of the product.

1. Command Logging

1.1. Do not use the subfolder name "segments" for the command log snapshot directory.

VoltDB reserves the subfolder "segments" under the command log directory for storing the actual command log files. Do not add, remove, or modify any files in this directory. In particular, do not set the command log snapshot directory to a subfolder "segments" of the command log directory, or else the server will hang on startup.

2. Database Replication

2.1. Some DR data may not be delivered if master database nodes fail and rejoin in rapid succession.

Because DR data is buffered on the master database and then delivered asynchronously to the replica, there is always the danger that data does not reach the replica if a master node stops. This situation is mitigated in a K-safe environment by all copies of a partition buffering on the master cluster. Then if a sending node goes down, another node on the master database can take over sending logs to the replica. However, if multiple nodes go down and rejoin in rapid succession, it is possible that some buffered DR data — from transactions when one or more nodes were down — could be lost when another node with the last copy of that buffer also goes down.

If this occurs and the replica recognizes that some binary logs are missing, DR stops and must be restarted.

To avoid this situation, especially when cycling through nodes for maintenance purposes, the key is to ensure that all buffered DR data is transmitted before stopping the next node in the cycle. You can do this using the @Statistics system procedure to make sure the last ACKed timestamp (using @Statistics DR on the master cluster) is later than the timestamp when the previous node completed its rejoin operation.

2.2. Avoid bulk data operations within a single transaction when using database replication

Bulk operations, such as large deletes, inserts, or updates are possible within a single stored procedure. However, if the binary logs generated for DR are larger than 45MB, the operation will fail. To avoid this situation, it is best to break up large bulk operations into multiple, smaller transactions. A general rule of thumb is to multiply the size of the table schema by the number of affected rows. For deletes and inserts, this value should be under 45MB to avoid exceeding the DR binary log size limit. For updates, this number should be under 22.5MB (because the binary log contains both the starting and ending row values for updates).

2.3. Database replication ignores resource limits

There are a number of VoltDB features that help manage the database by constraining memory size and resource utilization. These features are extremely useful in avoiding crashes as a result of unexpected or unconstrained growth. However, these features could interfere with the normal operation of DR when passing data from one cluster to another, especially if the two clusters are different sizes. Therefore, as a general rule of thumb, DR overrides these features in favor of maintaining synchronization between the two clusters.

Specifically, DR ignores any resource monitor limits defined in the deployment file when applying binary logs on the consumer cluster. This means, for example, if the replica database in passive DR has less memory or fewer unique partitions than the master, it is possible that applying binary logs of transactions that succeeded on the master could cause the replica to run out of memory. Note that these resource monitor limits *are* applied on any original transactions local to the cluster (for example, transactions on the master database in passive DR).

2.4. Different cluster sizes can require additional Java heap

Database Replication (DR) now supports replication across clusters of different sizes. However, if the replica cluster is smaller than the master cluster, it may require a significantly larger Java heap setting. Specifically, if the replica has fewer unique partitions than the master, each partition on the replica must manage the incoming binary logs from more partitions on the master, which places additional pressure on the Java heap.

A simple rule of thumb is that the worst case scenario could require an additional $P * R * 20\text{MB}$ space in the Java heap, where P is the number of sites per host on the replica server and R is the ratio of unique partitions on the master to partitions on the replica. For example, if the master cluster is 5 nodes with 10 sites per host and a K factor of 1 (i.e. 25 unique partitions) and the replica cluster is 3 nodes with 8 sites per host and a K factor of 1 (12 unique partitions), the Java heap on the replica cluster may require approximately 320MB of additional space in the heap:

Sites-per-host * master/replace ratio * 20MB

$8 * 25/12 * 20 = \sim 320\text{MB}$

An alternative is to reduce the size of the DR buffers on the master cluster by setting the `DR_MEM_LIMIT` Java property. For example, you can reduce the DR buffer size from the default 10MB to 5MB using the `VOLTDB_OPTS` environment variable before starting the master cluster.

```
$ export VOLTDB_OPTS="-DDR_MEM_LIMIT=5"
```

```
$ voltdb start
```

Changing the DR buffer limit on the master from 10MB to 5MB proportionally reduces the additional heap size needed. So in the previous example, the additional heap on the replica is reduced from 320MB to 160MB.

2.5. The `voltdb status --dr` command does not work if clusters use different client ports

The `voltdb status --dr` command provides real-time status on the state of database replication (DR). Normally, this includes the status of the current cluster as well as other clusters in the DR environment. (For example, both the master and replica in passive DR or all clusters in XDCR.) However, if the clusters are configured to use different port numbers for the client port, VoltDB cannot reach the other clusters and the command hangs until it times out waiting for a response from the other clusters.

3. Cross Datacenter Replication (XDCR)

3.1. Avoid replicating tables without a unique index.

Part of the replication process for XDCR is to verify that the record's starting and ending states match on both clusters, otherwise known as *conflict resolution*. To do that, XDCR must find the record first. Finding uniquely indexed records is efficient; finding non-unique records is not and can impact overall database performance.

To make you aware of possible performance impact, VoltDB issues a warning if you declare a table as a DR table and it does not have a unique index.

3.2. When starting XDCR for the first time, only one database can contain data.

You cannot start XDCR if both databases already have data in the DR tables. Only one of the two participating databases can have preexisting data when DR starts for the first time.

3.3. During the initial synchronization of existing data, the receiving database is paused.

When starting XDCR for the first time, where one database already contains data, a snapshot of that data is sent to the other database. While receiving and processing that snapshot, the receiving database is paused. That is, it is in read-only mode. Once the snapshot is completed and the two database are synchronized, the receiving database is automatically unpaused, resuming normal read/write operations.

3.4. A large number of multi-partition write transactions may interfere with the ability to restart XDCR after a cluster stops and recovers.

Normally, XDCR will automatically restart where it left off after one of the clusters stops and recovers from its command logs (using the `voltdb recover` command). However, if the workload is predominantly multi-partition write transactions, a failed cluster may not be able to restart XDCR after it recovers. In this case, XDCR must be restarted from scratch, using the content from one of the clusters as the source for synchronizing and recreating the other cluster (using the `voltdb create --force` command) without any content in the DR tables.

3.5. Avoid using `TRUNCATE TABLE` in XDCR environments.

`TRUNCATE TABLE` is optimized to delete all data from a table rather than deleting tuples row by row. This means that the binary log does not identify which rows are deleted. As a consequence, a `TRUNCATE TABLE`

statement and a simultaneous write operation to the same table can produce a conflict that the XDCR clusters cannot detect or report in the conflict log.

Therefore, do not use `TRUNCATE TABLE` with XDCR. Instead, explicitly delete all rows with a `DELETE` statement and a filter. For example, `DELETE * FROM table WHERE column=column` ensures all deleted rows are identified in the binary log and any conflicts are accurately reported. Note that `DELETE FROM table` without a `WHERE` clause is *not* sufficient, since its execution plan is optimized to equate to `TRUNCATE TABLE`.

- 3.6.** Use of the `VoltProcedure.getUniqueId` method is unique to a cluster, not across clusters.

VoltDB provides a way to generate a deterministically unique ID within a stored procedure using the `getUniqueId` method. This method guarantees uniqueness *within the current cluster*. However, the method could generate the same ID on two distinct database clusters. Consequently, when using XDCR, you should combine the return values of `VoltProcedure.getUniqueId` with `VoltProcedure.getClusterId`, which returns the current cluster's unique DR ID, to generate IDs that are unique across all clusters in your environment.

- 3.7.** Multi-cluster XDCR environments require command logging.

In an XDCR environment involving three or more clusters, command logging is used to ensure the durability of the XDCR "conversations" between clusters. If not, when a cluster stops, the remaining clusters can be at different stages of their conversation with the downed cluster, resulting in divergence.

For example, assume there are three clusters (A, B, and C) and cluster B is processing binary logs faster than cluster C. If cluster A stops, cluster B will have more binary logs from A than C has. You can think of B being "ahead" of C. With command logging enabled, when cluster A restarts, it will continue its XDCR conversations and cluster C will catch up with the missing binary logs. However, without command logging, when A stops, it must restart from scratch. There is no mechanism for resolving the difference in binary logs processed by clusters B and C before the failure.

This is why command logging is required to ensure the durability of XDCR conversations in a multi-cluster (that is, three or more) XDCR environment. The alternative, if not using command logging, is to restart all but one of the remaining clusters to ensure they are starting from the same base point.

- 3.8.** Do not use `voltadmin dr reset` to remove an XDCR cluster that is still running

There are two ways to remove a cluster from an XDCR relationship: you can use `voltadmin drop` on a running cluster to remove it from the XDCR network, or you can use `voltadmin dr reset` from the remaining clusters to remove a cluster that is no longer available. But you *should not* use `voltadmin dr reset` to remove a cluster that is still running and connected to the network. Resetting a running cluster will break DR for that cluster, but will result in errors on the remaining clusters and leave their DR queues for the reset cluster in an ambiguous state. Ultimately, this can result in the removed cluster not being able to rejoin the XDCR network at a later time.

4. TTL

- 4.1.** Use of TTL (time to live) with replicated tables and Database Replication (DR) can result in increased DR activity.

TTL, or time to live, is a feature that automatically deletes old records based on a timestamp or integer column. For replicated tables, the process of checking whether records need to be deleted is performed as a write transaction — even if no rows are deleted. As a consequence, any replicated DR table with TTL defined will generate frequent DR log entries, whether there are any changes or not, significantly increasing DR traffic.

Because of the possible performance impact this behavior can have on the database, use of TTL with replicated tables and DR is not recommended at this time.

5. Export

- 5.1.** Synchronous export in Kafka can use up all available file descriptors and crash the database.

A bug in the Apache Kafka client can result in file descriptors being allocated but not released if the `producer.type` attribute is set to "sync" (which is the default). The consequence is that the system eventually runs out of file descriptors and the VoltDB server process will crash.

Until this bug is fixed, use of synchronous Kafka export is not recommended. The workaround is to set the Kafka `producer.type` attribute to "async" using the VoltDB export properties.

6. Import

- 6.1.** Data may be lost if a Kafka broker stops during import.

If, while Kafka import is enabled, the Kafka broker that VoltDB is connected to stops (for example, if the server crashes or is taken down for maintenance), some messages may be lost between Kafka and VoltDB. To ensure no data is lost, we recommend you disable VoltDB import before taking down the associated Kafka broker. You can then re-enable import after the Kafka broker comes back online.

- 6.2.** Kafka import can lose data if multiple nodes stop in succession.

There is an issue with the Kafka importer where, if multiple nodes in the cluster fail and restart, the importer can lose track of some of the data that was being processed when the nodes failed. Normally, these pending imports are replayed properly on restart. But if multiple nodes fail, it is possible for some in-flight imports to get lost. This issue will be addressed in an upcoming release.

7. SQL and Stored Procedures

- 7.1.** Comments containing unmatched single quotes in multi-line statements can produce unexpected results.

When entering a multi-line statement at the `sqlcmd` prompt, if a line ends in a comment (indicated by two hyphens) and the comment contains an unmatched single quote character, the following lines of input are not interpreted correctly. Specifically, the comment is incorrectly interpreted as continuing until the next single quote character or a closing semi-colon is read. This is most likely to happen when reading in a schema file containing comments. This issue is specific to the `sqlcmd` utility.

A fix for this condition is planned for an upcoming point release

- 7.2.** Do not use assertions in VoltDB stored procedures.

VoltDB currently intercepts assertions as part of its handling of stored procedures. Attempts to use assertions in stored procedures for debugging or to find programmatic errors will not work as expected.

- 7.3.** The `UPPER()` and `LOWER()` functions currently convert ASCII characters only.

The `UPPER()` and `LOWER()` functions return a string converted to all uppercase or all lowercase letters, respectively. However, for the initial release, these functions only operate on characters in the ASCII character set. Other case-sensitive UTF-8 characters in the string are returned unchanged. Support for all case-sensitive UTF-8 characters will be included in a future release.

8. Client Interfaces

- 8.1.** Avoid using decimal datatypes with the C++ client interface on 32-bit platforms.

There is a problem with how the math library used to build the C++ client library handles large decimal values on 32-bit operating systems. As a result, the C++ library cannot serialize and pass Decimal datatypes reliably on these systems.

Note that the C++ client interface *can* send and receive Decimal values properly on 64-bit platforms.

9. SNMP

9.1. Enabling SNMP traps can slow down database startup.

Enabling SNMP can take up to 2 minutes to complete. This delay does not always occur and can vary in length. If SNMP is enabled when the database server starts, the delay occurs after the server logs the message "Initializing SNMP" and before it attempts to connect to the cluster. If you enable SNMP while the database is running, the delay can occur when you issue the **voltadmin update** command or modify the setting in the VoltDB Management Center Admin tab. This issue results from a Java constraint related to secure random numbers used by the SNMP library.

10. VoltDB Management Center

10.1. The VoltDB Management Center currently reports on only one DR connection.

With VoltDB V7.0, cross datacenter replication (XDCR) supports multiple clusters in an XDCR network. However, the VoltDB Management Center currently reports on only one such connection per cluster. In the future, the Management Center will provide monitoring and statistics for all connections to the current cluster.

11. Kubernetes

11.1. Shutting down a VoltDB cluster by setting `cluster.clusterSpec.replicas` to zero might not stop the associated pods.

Shutting down a VoltDB cluster by specifying a replica count of zero should shut down the cluster and remove the pods on which it ran. However, on very rare occasions Kubernetes does not delete the pods. As a result, the cluster cannot be restarted. This is an issue with Kubernetes. The workaround is to manually delete the pods before restarting the cluster.

11.2. Specifying invalid or misconfigured volumes in `cluster.clusterSpec.additionalVolumes` interferes with Kubernetes starting the VoltDB cluster.

The property `cluster.clusterSpec.additionalVolumes` lets you specify additional resources to include in the server classpath. However, if you specify an invalid or misconfigured volume, Helm will not be able to start the cluster and the process will stall.

11.3. Using binary data with the Helm `--set-file` argument can cause problems when later upgrading the cluster.

The Helm `--set-file` argument lets you set the value of a property as the contents of a file. However, if the contents of the file are binary, they can become corrupted if you try to resize the cluster with the **helm upgrade** command, using the `--reuse-values` argument. For example, this can happen if you use `--set-file` to assign a JAR file of stored procedure classes to the `cluster.config.classes` property.

This is a known issue for Kubernetes and Helm. The workaround is either to explicitly include the `--set-file` argument again on the **helm upgrade** command. Or you can include the content through a different mechanism. For example, you can include class files by mounting them on a separate volume that you then include with the `cluster.clusterSpec.additionalVolumes` property.

Implementation Notes

The following notes provide details concerning how certain VoltDB features operate. The behavior is not considered incorrect. However, this information can be important when using specific components of the VoltDB product.

1. IPv6

1.1. Support for IPv6 addresses

VoltDB works in IPv4, IPv6, and mixed network environments. Although the examples in the documentation use IPv4 addresses, you can use IPv6 when configuring your database, making connections through applications, or using the VoltDB command line utilities, such as **voltadb** and **voltadmin**. When specifying IPv6 addresses on the command line or in the configuration file, be sure to enclose the address in square brackets. If you are specifying both an IPv6 address and port number, put the colon and port number *after* the square brackets. For example:

```
voltadmin status --host=[2001:db8:85a3::8a2e:370:7334]:21211
```

2. VoltDB Management Center

2.1. Schema updates clear the stored procedure data table in the Management Center Monitor section

Any time the database schema or stored procedures are changed, the data table showing stored procedure statistics at the bottom of the Monitor section of the VoltDB Management Center get reset. As soon as new invocations of the stored procedures occur, the statistics table will show new values based on performance after the schema update. Until invocations occur, the procedure table is blank.

3. SQL

3.1. You cannot partition a table on a column defined as ASSUMEUNIQUE.

The ASSUMEUNIQUE attribute is designed for identifying columns in partitioned tables where the column values are known to be unique but the table is not partitioned on that column, so VoltDB cannot verify complete uniqueness across the database. Using interactive DDL, you can create a table with a column marked as ASSUMEUNIQUE, but if you try to partition the table on the ASSUMEUNIQUE column, you receive an error. The solution is to drop and add the column using the UNIQUE attribute instead of ASSUMEUNIQUE.

3.2. Adding or dropping column constraints (UNIQUE or ASSUMEUNIQUE) is not supported by the ALTER TABLE ALTER COLUMN statement.

You cannot add or remove a column constraint such as UNIQUE or ASSUMEUNIQUE using the ALTER TABLE ALTER COLUMN statement. Instead to add or remove such constraints, you must first drop then add the modified column. For example:

```
ALTER TABLE employee DROP COLUMN empID;  
ALTER TABLE employee ADD COLUMN empID INTEGER UNIQUE;
```

3.3. Do not use UPDATE to change the value of a partitioning column

For partitioned tables, the value of the column used to partition the table determines what partition the row belongs to. If you use UPDATE to change this value and the new value belongs in a different partition, the UPDATE request will fail and the stored procedure will be rolled back.

Updating the partition column value may or may not cause the record to be repartitioned (depending on the old and new values). However, since you cannot determine if the update will succeed or fail, you should not use UPDATE to change the value of partitioning columns.

The workaround, if you must change the value of the partitioning column, is to use both a DELETE and an INSERT statement to explicitly remove and then re-insert the desired rows.

3.4. Ambiguous column references no longer allowed.

Starting with VoltDB 6.0, ambiguous column references are no longer allowed. For example, if both the *Customer* and *Placedorder* tables have a column named *Address*, the reference to *Address* in the following SELECT statement is ambiguous:

```
SELECT OrderNumber, Address FROM Customer, Placedorder
. . .
```

Previously, VoltDB would select the column from the leftmost table (*Customer*, in this case). Ambiguous column references are no longer allowed and you must use table prefixes to disambiguate identical column names. For example, specifying the column in the preceding statement as *Customer.Address*.

A corollary to this change is that a column declared in a *USING* clause can now be referenced using a prefix. For example, the following statement uses the prefix *Customer.Address* to disambiguate the column selection from a possibly similarly named column belonging to the *Supplier* table:

```
SELECT OrderNumber, Vendor, Customer.Address
      FROM Customer, Placedorder Using (Address), Supplier
. . .
```

4. Runtime

4.1. File Descriptor Limits

VoltDB opens a file descriptor for every client connection to the database. In normal operation, this use of file descriptors is transparent to the user. However, if there are an inordinate number of concurrent client connections, or clients open and close many connections in rapid succession, it is possible for VoltDB to exceed the process limit on file descriptors. When this happens, new connections may be rejected or other disk-based activities (such as snapshotting) may be disrupted.

In environments where there are likely to be an extremely large number of connections, you should consider increasing the operating system's per-process limit on file descriptors.

4.2. Use of Resources in JAR Files

There are two ways to access additional resources in a VoltDB database. You can place the resources in the `/lib` folder where VoltDB is installed on each server in the cluster or you can include the resource in a subfolder of a JAR file you add using the sqlcmd **LOAD CLASSES** directive. Adding resources via the `/lib` directory is useful for stable resources (such as third-party software libraries) that do not require updating. Including resources (such as XML files) in the JAR file is useful for resources that may need to be updated, as a single transaction, while the database is running.

LOAD CLASSES is used primarily to load classes associated with stored procedures and user-defined functions. However, it will also load any additional resource files included in subfolders of the JAR file. You can remove classes that are no longer needed using the **REMOVE CLASSES** directive. However, there is no explicit command for removing other resources.

Consequently, if you rename resources or move them to a different location and reload the JAR file, the database will end up having multiple copies. Over time, this could result in more and more unnecessary memory being used by the database. To remove obsolete resources, you must first reinitialize the database root directory, start a fresh database, reload the schema (including the new JAR files with only the needed resources) and then restore the data from a snapshot.

4.3. Servers with Multiple Network Interfaces

If a server has multiple network interfaces (and therefore multiple IP addresses) VoltDB will, by default, open ports on all available interfaces. You can limit the ports to a single interface in two ways:

- Specify which interface to use for internal and external ports, respectively, using the **--internalinterface** and **--externalinterface** arguments when starting the database process with the **voltadb start** command.
- For an individual port, specify the interface and port on the command line. For example **voltadb start --client=32.31.30.29:21212**.

Also, when using an IP address to reference a server with multiple interfaces in command line utilities (such as **voltadmin stop node**), use the @SystemInformation system procedure to determine which IP address VoltDB has selected to identify the server. Otherwise, if you choose the wrong IP address, the command might fail.

5. Platforms

5.1. OpenShift and Transparent Huge Pages (THP)

For production, VoltDB requires that Transparent Huge Pages (THP) are disabled because they interfere with memory-intensive applications. However, THP may be enabled on OpenShift containers and the containers themselves not have permission to disable them. To overcome this situation, you must run the Helm chart for disabling THP from a privileged container:

```
$ helm -n kube-system install thp voltdb/transparent-hugepages \
  --set thp.securityContext.privileged=true
```

5.2. Kubernetes Compatibility

The following table describes the compatibility matrix for versions of the VoltDB software, VoltDB Kubernetes Operator, and associated Helm Charts. The table identifies the versions of the Operator and Helm chart required and supported for running the specified version of the VoltDB server software. See the *VoltDB Operator Release Notes* for additional information about individual releases of the VoltDB Operator.

Table 1. Kubernetes Software Compatibility Chart

VoltDB Software	VoltDB Operator	Helm Chart
11.4.3	1.8.3	1.8.3
11.4.1, 11.4.2	1.8.2	1.8.2
11.4.0	1.8.0, 1.8.1	1.8.0, 1.8.1
11.3.2	1.7.4, 1.7.5	1.7.4, 1.7.5
11.3.1	1.7.3	1.7.3
11.3	1.7.0	1.7.0
11.2	1.5.0, 1.6.2	1.5.0, 1.6.2
11.1, 11.1.1	1.4.0	1.4.0
10.2.6	1.3.5	1.3.5
10.2.5	1.3.5	1.3.5
10.2.4	1.3.4	1.3.4
10.2.3	1.3.3	1.3.3
10.2.2	1.3.2	1.3.2
10.2.1	1.3.0	1.3.1
10.1.3	1.2.1	1.2.1
10.1.2	1.2.0	1.2.0

VoltDB Software	VoltDB Operator	Helm Chart
10.1.0, 10.1.1	1.1.0	1.1.0, 1.1.1
10.0.0	1.0.0	1.0.0 to 1.0.2