Release Notes

Product	VoltDB
Version	V10.2.29
Release Date	January 30, 2026

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@voltdb.com. Thank you.

Important

Starting with the next feature release, version 11.0, VoltDB will switch from using Python 2 to Python 3. This means Python 3 will be required by all VoltDB command line utilities and the VoltDB Python API.

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 **voltdb start**).

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.

For customers upgrading from V8.x or earlier releases of VoltDB, please see the *V8.0 Upgrade Notes*.

For customers upgrading from V7.x or earlier releases of VoltDB, please see the *V7.0 Upgrade Notes*.

For customers upgrading from V6.x or earlier releases of VoltDB, please see the *V6.0 Upgrade Notes*.

Changes Since the Last Release

Users of previous versions of VoltDB should take note of the following changes that might impact their existing applications.

1. Release V10.2.29 (January 30, 2026)

1.1. Recent Changes

- This is the final update to VoltDB V10, as the software has reached the end of its long-term support period.
- The Volt Enterprise and Prometheus Agent containers have been updated to Alpine 3.23 and JRE 11.0.30.

1.2. Security Updates

The following high and critical CVEs were resolved:

CVE-2025-8677
CVE-2025-12183
CVE-2025-13878
CVE-2025-15467
CVE-2025-40778
CVE-2025-40780
CVE-2025-50181
CVE-2025-50182
CVE-2025-66418
CVE-2025-66471
CVE-2025-66566
CVE-2025-69419
CVE-2026-21441
CVE-2026-23490
CVE-2026-24049

2. Release V10.2.28 (October 09, 2025, updated January 14, 2026)

2.1. Security Updates

The following high and critical CVEs were resolved:

CVE-2024-13009
CVE-2024-24806
CVE-2025-26519
CVE-2025-58056

2.2. Additional Improvements

The following limitations in previous versions have been resolved:

- Sometimes, if the DR_OVERFLOW device ran out of space, all I/O processing was affected, blocking stored procedure execution and in some cases possibly generating multi-partition deadlocks. This issue has been resolved.

- When using XDCR between heterogeneous clusters (that is, clusters with a differing number of unique partitions), it was possible for the joining cluster to run out of heap memory while processing the initial synchronizing snapshot. This issue has been resolved.

2.3. Correction

The following changes, originally announced as part of the V10.2.28 release, were accidentally left out of the final build. They are currently scheduled for inclusion in the next release:

- Previously, if a DELETE statement did not contain a WHERE clause, it was treated like a TRUNCATE statement, deleting all data in the table on the current cluster but not creating the binary logs needed to synchronize with other XDCR clusters. This behavior has been corrected so that DELETE always creates the necessary logs to maintain consistency across XDCR clusters.

- In certain rare instances, particularly involving frequent multi-partition procedures, XDCR flow control could stall, resulting in the affected cluster no longer processing incoming XDCR data. This issue has been resolved.

3. Release V10.2.27 (February 20, 2025)

3.1. Security Updates

The following high and critical CVEs were resolved:

CVE-2025-24970

3.2. Additional Improvements

The following limitations in previous versions have been resolved:

- There was a rare race condition in XDCR clusters where, if someone performs a schema change while the initial synchronization snapshot is being sent to one of the clusters, it was possible for XDCR communication to fail and replication stop. This issue has been resolved.

4. Release V10.2.26 (December 2, 2024)

4.1. Security Updates

The following high and critical CVEs were resolved:

CVE-2024-47561

4.2. Additional Improvements

The following limitations in previous versions have been resolved:

- There was a degradation in performance starting in VoltDB V9, specifically for SQL queries using an index where the requested data did not exist within the database. This issue has been resolved.

5. Release V10.2.25 (September 24, 2024)

5.1. Important Note for Kubernetes Users

Previous releases of Volt used control groups V1 to calculate the available memory on Kubernetes. However, recent updates to Kubernetes have switched to control groups V2, causing the V1 calculations to return incorrect and inflated values. This could result in the Volt server process exhausting memory and generating an out of memory error or triggering a resource limit and pausing.

With the release of 10.2.25, Volt now supports both control groups V1 and V2. *All Kubernetes users are strongly encouraged to upgrade at their earliest convenience* to avoid issues now or in the future.

5.2. Security updates

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

CVE-2024-6232
CVE-2024-6345
CVE-2024-7254
CVE-2024-45490

5.3. Additional improvements

The following limitations in previous versions have been resolved:

- There was a situation where, if the procedure call timeout was set to a fraction of a second on the client call, the timeout response could be delayed if it coincided with a topology change in the cluster (that is, a node dropping out of rejoining. This was caused by the timeout and topology tracking running on the same thread. The problem was also specific to the version 1 Java client only. This issue has been resolved.

- Under certain conditions, XDCR clusters could generate a large volume of log messages warning of export gaps being recovered. This was caused by two issues: an unusually large number of XDCR conflicts, and a bug in the handling of the conflict log when a cluster node drops and rejoins. In rare cases a race condition could exacerbate the situation and cause nodes to crash.

  The internal handling of the conflict log and resulting race condition have been corrected. However, it is important for the integrity of your data to recognize (and correct) the application logic if it is causing frequent and persistent transaction conflicts in an XDCR environment.

- In Kubernetes, Volt Active Data now supports control group V2 resource management when determining how much memory is available to the pods. Previously, Volt used control group V1, which could return incorrect values on recent releases of Kubernetes that use V2 by default.

6. Release V10.2.24 (August 2, 2024)

6.1. Recent improvements

The following limitations in previous versions have been resolved:

- Recent releases of V10.2 (starting with 10.2.16) did not use the native implementation of OpenSSL. As a result, enabling TLS/SSL encryption could negatively impact performance. This issue has been resolved.

- There was an issue where one specific Kafka client could not publish records to a Volt opaque topic. The cause of the problem was that the topic was reporting an incorrect offset back to the client. This issue has been resolved.

7. Release V10.2.23 (April 10, 2024)

7.1. Running 10.2.23 on Kubernetes

All Volt V10.2 releases since 10.2.21 are supported by the latest Volt Operator for Kubernetes. This means you can start a database using Volt V10.2.23 simply by specifying the software version in the `global.voltdbVersion` property, as described in the *Volt Operator for Kubernetes* guide:

```
$ helm install mydb voltdb/voltdb \
  --set global.voltdbVersion=10.2.23
```

If you prefer to use the previous, V10-specific operator, you can do that by specifying the operator version using the `--version` qualifier and the software image in the `cluster.clusterSpec.image.tag` property:

```
$ helm install mydb voltdb/voltdb \
  --version=1.3 \
  --set cluster.clusterSpec.image.tag=10.2.23
```

7.2. Recent improvements

The following limitations in previous versions have been resolved:

- In certain rare situations, if a schema change coincides with a node's attempt to rejoin the cluster, a race condition could result in the rejoin request being rejected. This issue has been resolved.

- In previous releases, if an Active(N) conflict occurs due to a row containing an invalid timestamp, replication would be broken and the server reported an SQL error related to the CAST() function. Now the conflict is recorded correctly in the conflict log and replication continues.

- There were situations where Volt failed to allocate memory, but rather than reporting a meaningful error, the server process failed with a segmentation violation (SIGSEGV). The original error is now caught and an appropriate error that sufficient memory could not be allocated is reported.

- There was an issue related to elastically expanding a cluster where export was enabled or XDCR replication was enabled, even if XDCR was not currently connected to another cluster. If, after the expansion, the cluster performed a stop node action followed by a rejoin, there was a potential conflict over which nodes were controlling which partitions. In the case of export, multiple nodes might attempt to export the same data. In the case of XDCR, attempting to perform an in-service upgrade would fail when the second node being updated would not shutdown. This issue has been resolved.

8. Release V10.2.22 (February 16, 2024)

8.1. Recent improvements

The following limitations in previous versions have been resolved:

- On rare occasions, if a node dropped out of the cluster, then during the initial rejoin it failed again or was interrupted, subsequent attempts to rejoin the cluster would fail claiming there was a snapshot in progress. Rejoining a new node (or the same node re-initialized) would clear the condition. This issue has been resolved.

- Under certain circumstances, it was possible to trigger a small but persistent memory leak in the DR port if the port was configured for TLS/SSL encryption. Specifically, when programs such as port scanners or load balancers repeatedly test the DR port, creating new connections each time, memory usage would grow. This issue has been resolved.

- There was an issue with Active(N) cross datacenter replication (XDCR), where if a producer node stops and restarts, it could take up to a minute for the consumer to reconnect after the producer comes back up. This issue has been resolved.

9. Release V10.2.21 (January 12, 2024)

9.1. Security updates

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

CVE-2023-5363
CVE-2023-44487

9.2. Additional improvements

The following limitations in previous versions have been resolved:

- A number of edge cases related to elastically resizing the database cluster have been detected and corrected, significantly improving the reliability and robustness of elastic operations.

- Extra snapshot files manually copied into the snapshot folders in the database root directory could result in nodes failing to rejoin the cluster or the cluster failing to start after a crash or shutdown. This issue has been resolved. However, manually copying additional files into the root directory structure is strongly discouraged and can cause unpredictable behavior, including failures.

- A race condition in command logging could cause the cluster to crash on startup. This condition could only be triggered if the command logs contained a schema update. This issue has been resolved.
- If the JDBC export connector fails to write a row, or batch of rows, to the target, it reports an error in the error log. Previously, this error message included the entire contents of the failed row(s), which filled the logs with unnecessary information. The error message has been rewritten to report only the most pertinent information, that is, the name of the table in question.
- Under certain circumstances, when resizing a database cluster to reduce the number of nodes, a flurry of informational messages reporting that an invocation request was rejected could fill up the log files. This issue has been resolved.

10. Release V10.2.20 (October 20, 2023)

10.1. Security updates

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

CVE-2023-3341
CVE-2023-4236
CVE-2023-38039
CVE-2023-39410
CVE-2023-43642

10.2. Additional improvements

The following limitations in previous versions have been resolved:

- It is possible, after pausing the database, to alter a DR table on all of the participating clusters. Unfortunately, after altering the table and resuming the clusters, it was possible in certain cases for subsequent tuple updates (UPDATE or DELETE) to generate unnecessary and potentially misleading conflicts in the DR conflict log. This issue has been resolved.
- There was a minor memory leak associated with closing client connections with SSL/TLS enabled. Although trivial under normal conditions, for applications that connect and disconnect repeatedly in rapid succession, the cumulative effect can ultimately use up all available memory. This issue has been resolved.

11. Release V10.2.19 (September 19, 2023)

11.1. Recent improvements

The following limitations in previous versions have been resolved:

- Recent releases of Volt Active Data V10 did not include a required support library, resulting in errors at runtime, such as when using the @QueryStats system procedure or `querystats` directive in **sqlcmd**. This issue has been resolved.
- In certain cases, if a node rejoining the cluster unexpectedly fails, the rejoin operation fails, but subsequent attempts to rejoin the node also fail with no clear explanation. This issue has been resolved.
- There were issues related to the DR RESET command that could, in certain edge cases, result in unexpected behavior. In some cases, clusters could not rejoin an XDCR mesh without first having the DR ID changed. Or if the rejoin operation failed, any further attempt to rejoin the cluster to the mesh would also fail. Or the cluster might inadvertently report that the rejoin was complete before the operation actually finished. These issues have been resolved.

12. Release V10.2.18 (July 25, 2023)

12.1. Support for Kubernetes version 1.25.

This release of the VoltDB Operator and Helm chart adds support for Kubernetes version 1.25. It can be used only on Kubernetes version 1.21 and later.

Kubernetes removed support for the deprecated PodSecurityPolicy in version 1.25. To this end, the default chart setting for `global.rbac.pspEnabled` has been changed from "true" to "false".

12.2. Improved memory management

This release provides additional information and control when managing memory on VoltDB servers and in particular, the undo pool. The *undo pool* is used to store temporary information needed to "undo" a transaction in case it must be rolled back. The pool grows as needed, based on how much data is needed to undo the current transactions. If your workload includes certain infrequent but memory-intensive transactions (such as deleting large volumes of data on a weekly basis) the undo pool can grow quite large, artificially inflating the resident set size (RSS).

A column has been added to the results of the @Statistics system procedure MEMORY selector. The column, UNDO_POOL_SIZE, measures the amount of memory, in kilobytes, allocated for the undo pool.

A new system procedure, @PurgeUndo, lets you reset the undo pool size to zero. If you find your RSS growing incrementally and you suspect the undo pool, you can use the UNDO_POOL_SIZE column in the @Statistics MEMORY procedure results to verify the amount of space being consumed by the undo pool. If the pool is unnecessarily large, you can use the @PurgeUndo procedure to reset it.

12.3. Security updates

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

CVE-2023-2976
CVE-2023-34455

12.4. Additional improvements

The following limitations in previous versions have been resolved:

- Previously, assigning an export target to a named thread pool in the configuration file but not defining the thread pool itself, did not cause an error. However, as soon as a table or stream is declared exporting to that target, the database would stop with a fatal error. Now, the database will not start if the configuration does not have a thread pool defined for any named in the export declarations.
- An issue was found regarding export. If a server crashed unexpectedly (due to failure or a forced shutdown) transactions being processed at the time may be interrupted and rolled back, leaving the database unchanged. However, if any of those transactions included both writing to an export (or topic) target and to database tables, it was possible in rare cases, due to a race condition in export handling, for an export or topic record to slip through, resulting in it being queued and sent to the target, creating an atomicity error.

  This issue has been resolved. Now, if the transaction fails, the export or topic publishing is also canceled. Note that one consequence of this correction is that the maximum time before export records are queued to the export target, which previously was solely controlled by the flush interval, now includes the time required for the transaction to be submitted, processed, and returned by the appropriate partitions and sites.
- The export tools that can be found in the `tools/exporttools/` folder after installing Volt Active Data have been updated to be compatible with the latest version of the export overflow files.

13. Release V10.2.17 (June 6, 2023)

13.1. Security updates

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

CVE-2021-3712
CVE-2021-23840
CVE-2022-0778
CVE-2022-1292
CVE-2022-1304
CVE-2022-2068
CVE-2022-41723
CVE-2023-29491

13.2. Additional improvements

The following limitations in previous versions have been resolved:

- Previously, if a client JAR file contained additional unexpected entries, the **sqlcmd** utility would stall attempting to load information from the JAR. The utility now ignores unexpected entries, resolving this issue.
- There was an edge case where a **voltadmin dr reset** command could result in a deadlock, causing the database to hang. The issue has been resolved.

14. Release V10.2.16 (February 18, 2023)

14.1. Security updates

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

CVE-2022-41721
CVE-2022-41881

14.2. Additional improvements

The following limitations in previous versions have been resolved:

- 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.
- There was a race condition where a problem pausing export connections during a schema or configuration change could result in a deadlock. This issue has been resolved.
- Under normal conditions, after elastically shrinking the cluster (that is, removing nodes) the cluster saves a snapshot as a final step. If the snapshot accidentally starts before the nodes are completely removed, later attempts to shrink the cluster could fail, reporting that an elastic operation is already in progress. This issue has been resolved.
- In certain cases when attempting to shutdown a cluster in Kubernetes, if the nodes take too long to stop, the shutdown could fail. This issue has been resolved..

15. Release V10.2.15 (November 15, 2022, updated August 9, 2023)

15.1. New Prometheus metrics added

Information related to the configuration and status of the cluster, also available from the @SystemInformation system procedure, is now available as metrics shared through the Prometheus agent. See the sections on integrating with Prometheus in the *Volt Administrator's Guide* and *Volt Kubernetes Administrator's Guide* for more information about using Volt Active Data with Prometheus.

15.2. Log4J replaced by reload4J

VoltDB does not use any of the components implicated in the published CVEs related to Log4J. 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.

15.3. Security updates

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

CVE-2020-26160
CVE-2021-28165
CVE-2021-38561
CVE-2022-1996
CVE-2022-21698
CVE-2022-3171
CVE-2022-42003

15.4. Additional improvements

The following limitations in previous versions have 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.
- There was a minor memory leak associated with statistics triggered by ad hoc queries. Although normally not sufficient to even be noticed, constant and very frequent ad hoc queries (for example thousands an hour for days) each creating a separate connection could eventually cause excessive memory usage, slowing down the database and, in extreme cases, ultimately blocking further transactions.This issue has been resolved.

16. Release V10.2.14 (September 1, 2022)

16.1. Security Notice

The following package updates have been added to the Kubernetes release of Volt Active Data to address known security vulnerabilities:

- AdoptOpenJDK 11.0.16_8
- Alpine 3.16.2

16.2. Additional improvements

The following limitations in previous versions have been resolved:

- Previously, when configuring VoltDB on Kubernetes with security enabled, if you specified a username and password for the Operator, but did not define any other users, installing the Helm release would fail. This issue has been resolved and the Operator now automatically adds the specified user definition to the database configuration.

- There was an edge case when using XDCR where, if a cluster stops and rejoins the XDCR environment, then stops again before any XDCR data is exchanged, replication is broken and the cluster must be reinitialized and join the XDCR environment from scratch to reestablish communication. This issue has been resolved.

17. Release V10.2.13 (July 20, 2022)

17.1. 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_roundtriptime_1m_max
- replication_roundtriptime_1m_avg
- replication_roundtriptime_5m_max
- replication_roundtriptime_5m_avg

17.2. Additional improvements

The following limitations in previous versions have been resolved:

- In the situation where a cluster failed or was forcibly shutdown while a node was being added or removed, attempting to restart the cluster could result in an error claiming there were "incomplete command logs", even if command logging was not enabled. This was caused by an incomplete snapshot left by the interrupted cluster expansion. The issue has been resolved.
- Previously, the **voltadmin release** command did not always release export on all partitions within the cluster. This issue has been resolved.
- 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.
- 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 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.

- The timeout period associated with export block operations has been extended to avoid erroneously timing out operations for slower export targets, such as JDBC.

- 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.

- 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.

- There was an issue where, if a stored procedure queued more than 200 SQL statements before calling `voltExecuteSQL()` 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.

- 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.

18. Release V10.2.12 (May 6, 2022)

18.1. Additional improvements

The following limitations in previous versions have 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.

- The binaries of AdoptOpenJDK and Alpine in the Volt Docker image for Kubernetes have been updated to versions 11.0.14 and 3.15.4, respectively, to eliminate potential security vulnerabilities.

- 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.

19. Release V10.2.11 (April 26, 2022)

19.1. 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.

- 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.

20. Release V10.2.10 (March 8, 2022)

20.1. Additional improvement

The following limitation in previous versions has been resolved:

- There was an issue related to cross datacenter replication (XDCR) with three or more clusters. If a cluster crashed and the remaining clusters were under heavy load when the missing cluster was reinitialized and attempted to rejoin, the rejoin might fail. When this happened, the running clusters reported an "unrecoverable replication error" during the reload. This issue has been resolved.

21. Release V10.2.9 (February 15, 2022)

21.1. 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.

- 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.

22. Release V10.2.8 (January 25, 2022, updated June 1, 2022)

22.1. Database Replication (DR) improvements

A number of improvements to database replication (DR) developed in the follow-on release (V11.x) have been backported to V10.2.8 to increase stability and reliability. These changes include:

- Time allowed for a DR snapshot to initiate a new connection has been increased from 30 seconds to 90 seconds.

- Additional logging of DR and XDCR activity on initiation and teardown to aid in debugging connection issues.

- Previously, if a DR reset command did not complete its cleanup activities, attempting to create a connection from a newly initialized cluster with the same DR ID could result in a Null Pointer Exception on the producer cluster. This problem has been resolved.

22.2. Recent improvements

The following limitations in previous versions have been resolved:

- There was an issue where if a topic was configured specifying the `consumer.key` property but initially there was no stream defined to export to that topic, the cluster would crash on startup with an error indicating that the topic is "not using a stream." This issue has been resolved.

- The VoltDB Management Center lets you use a web browser to perform administrative functions. However, in previous releases, if you attempted to connect to two database instances with security enabled from separate browser tabs, logging on to one database would log you out of the other and vice versa. This problem was erroneously reported as fixed in 10.2.5. However, the appropriate code was accidentally left out until now. This problem is now resolved.

23. Release V10.2.7 (January 4, 2022)

23.1. Security Notice

The jQuery libraries used by the VoltDB Management Center have been updated to the following versions to address security vulnerabilities:

- jQuery V3.5.1
- jQuery UI V1.12.1
- jQuery Slimscroll V1.3.8
- jQuery Validate V1.19.2

23.2. VoltDB Management Center improvements

In addition to the security updates, 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

23.3. Additional improvements

The following limitations in previous versions have 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.
- There was an issue where using the CAST function to convert a VARCHAR column to a BIGINT could generate incorrect values if the number in question had more than 18 digits. This issue has been resolved.
- VoltDB constrains the size of messages sent between cluster nodes and will cancel transactions that exceed the limit. However, in rare situations, the system itself can generate overly large messages and cause a "bad message length" error. This release adds additional hexadecimal information to the logs when this happens, to help identify the root cause of the error.
- 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.
- There was an issue where altering the stream associated with a topic to remove a column could cause a subsequent hash mismatch and crash the cluster. The issue has been resolved.
- Additional information is now logged if the SQL compiler encounters an unexpected error while processing a data definition language (DDL) statement.

24. Release V10.2.6 (September 3, 2021)

24.1. 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.

24.2. General release of VoltDB Topics for production use

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.

24.3. Additional improvements

The following limitations in previous versions have been resolved:

- Previously, it was possible when using IPv6, for the @SystemInformation system procedure to return the string "localhost" as the server's IP address, which also interferes with the server's ability to join a cluster. This problem has been resolved.
- There was an issue with the VoltDB Management Center where, if security was enabled, the user could not log in through the web browser. This problem has been resolved.

25. Release V10.2.5 (June 16, 2021)

25.1. IMPORTANT: Limit partition row feature to be removed in VoltDB V11.0

The LIMIT PARTITION ROWS feature was deprecated in Version 9. It will be removed in Version 11. This is a change to the VoltDB schema syntax that is not forward compatible.

This means that if your database schema still contains the LIMIT PARTITION ROWS syntax, you need to remove the offending clause before upgrading to the upcoming major release. Fortunately, there is a simple process for doing this. You can use the `ALTER TABLE {table-name} DROP LIMIT PARTITION ROWS` statement to correct the table schema while the database is running and with no impact to the database contents.

25.2. Improved Java client handles both topology awareness and reconnections

The VoltDB Java client has two separate features that let you enable topology awareness (`setTopology-ChangeAware`) and reconnection for lost connections (`setReconnectOnConnectionLosss`). Topology awareness uses existing connections to determine if there has been any changes to the servers or ports available and creates connections to all servers in the cluster. Reconnection periodically attempts to reconnect to a specific server and port if a connection is lost.

Previously, these features were mutually exclusive. However, there are times when you might require both. For example, topology awareness fails if there are no remaining connections (such as a cluster reboot). Whereas,

reconnection can only reconnect to addresses it already knows; it cannot detect if a failed server restarts with a new address. To cover this situation, the client has been improved to allow both features to be enabled at the same time.

25.3. The VoltDB Kubernetes Operator now logs all interaction with the individual VoltDB processes

To aid in debugging, the VoltDB Operator now logs all of the commands it issues to the VoltDB processes running on the Kubernetes pods.

25.4. The Java client autotune feature is deprecated

The VoltDB Java client has an autotune feature (with methods in the ClientConfig class) that was originally designed to assist in developing demo applications. This feature is now deprecated and will be removed in a future release.

25.5. The Java client send-reads-to-replicas feature is deprecated

Previously, VoltDB had a feature to enable complete read consistency (to protect against various failure scenarios). The clientConfig method setSendReadsToReplicasByDefault was associated with that feature. However, read consistency is now always enabled, so this method is obsolete and has been deprecated. It will be removed in a future release.

25.6. Additional improvements

The following limitations in previous versions have been resolved:

- Previously, if the snapshot rate limit was set (using the Java property SNAPSHOT_RATELIMIT_MEGABYTES), requesting a CSV formatted snapshot could raise an illegal argument exception stating that "requested permits must be positive" and the resulting snapshot files would be empty. This only affected CSV formatted snapshots. This problem has been resolved.

- In Kubernetes, if you set the property `cluster.clusterSpec.deletePVC` to false then uninstalled and reinstalled a release with the same name, some of the characteristics of the previous instance of the release would be reused, creating problems for the new instance. This problem has been resolved.

- In previous releases, there was an issue when using XDCR in Kubernetes, where repetitive health checks on the DR port could flood the logs with warnings and interfere with regular client connections. A similar condition could occur when enabling SSL on the VoltDB cluster. These problems have been resolved.

- When using the VoltDB Java client with `setTopologyChangeAware` enabled, the service could generate two calls to the client status listener callback when a connection was created, rather than one. This problem has been resolved.

- Previously, if both `setTopologyChangeAware` and `setReconnectOnConnectionLosss` were enabled, and the last connection was lost long enough to trigger backpressure and the query times out, the procedure callbacks are called repeatedly, causing unnecessary thrashing and CPU consumption. The new, improved client now supports use of these features together and this problem has been resolved.

26. Release V10.2.4 (May 7, 2021)

26.1. New license improvements

This release includes a number of improvements to the licensing and management of VoltDB software. These improvements include:

- A new **voltadmin license** command, which updates the license on a running VoltDB cluster

- A new **voltadmin inspect** command used by VoltDB product support to display summary information about the cluster operating environment, including the current license

The new **voltadmin license** command is the most important of these changes for users, since it allows you to update the license for a cluster without having to restart. Note that the cluster must be complete — with no missing nodes — when you update the license. For example:

```
$ voltadmin license new-license-file.xml
```

26.2. Beta utility voltsql deprecated

There is a beta utility, voltsql, that extends the standard **sqlcmd** utility adding command completion and other interactive aids. The added functionality never fully met its goals and maintaining two separate utilities is both impractical from a product perspective and confusing from a customer perspective. For that reason, **voltsql** is being deprecated and will be removed in the next major release.

26.3. Improved connectivity for XDCR in Kubernetes

In environments such as Kubernetes where IP addresses are transient, XDCR could take an extended period of time to reconnect a server on a remote cluster if the server restarted with a different address. The connection logic has been rewritten to accommodate these environments, eliminating the delay.

26.4. Additional improvements

The following limitations in previous versions have been resolved:

- The **snapshotconverter** utility lets you generate CSV files from VoltDB snapshot files. These files can be used to recover and reload data from individual tables through the **csvloader** utility. However, for certain data — such as XDCR tables, tables defined with MIGRATE, or views with no `COUNT(*)` column — the **snapshotconverter** utility includes hidden columns in its output, which can be confusing. A new command flag has been added, `--filter-hidden`, that lets you exclude these hidden columns from the utility's output.

- The Java method `TaskHelper.getTaskScepe` has been replaced by the method `getTaskScope`. The older method is now deprecated and will be removed in a future release.

- Previously, if a cluster with command logging enabled stopped and restarted multiple times, with the `--missing` argument used during at least one of the restarts, it was possible for the recovery of the command logs to fail with an index out of bounds error. The problem was that the database could not identify the original topology of the cluster. This issue has been resolved. If the same situation occurs now, the cluster assigns a new arrangement to the partitions during recovery.

- There was an issue regarding tasks and directed procedures, where modifying the class (with LOAD CLASSES) for a directed procedure associated with a task that was already running could cause the database to fail with an error stating that active transactions were "moving backwards". This issue has been resolved.

- There was an issue where Prometheus was randomly reporting additional database replication (DR) producer statistics with an invalid timestamp. This problem has been resolved.

- Previously, a problem could occur if a node becomes detached from the cluster (for example, due to network issues) and does not immediately fail but times out. The result was that the remote cluster might stop replication, reporting a "replica ahead of master" error. This issue has also been resolved.

- There was an issue in the export subsystem where, it was possible that releasing an export queue with missing records could result in more records being deleted from the queue than necessary. Normally releasing an export queue with a gap means the export connector "jumps" to the next record after the missing data.

However, if — after the queue pauses at a gap — the database schema was updated before the release command is issued, it was possible for additional records unaffected by the gap to be deleted from the queue. This issue has been resolved.

- There was a potential situation where, if a cluster used for cross datacenter replication (XDCR) suffered one or more node failures, then was shutdown and restarted using command logs to recover, replication might later fail with a "replica ahead of master" error. This underlying issue was related to recovery using the failed node's command logs which did not match the current state of the remote cluster. This problem has been resolved.

- Previously, integer columns (such as INTEGER and BIGINT) were allowed as TTL columns. However, they did not produce the correct results. TTL columns are now constrained to TIMESTAMP columns only.

- In recent releases (since 10.2.2), the command **voltdb get license** failed to run, returning a Java error message instead. This problem has been resolved.

- Recent improvements to VoltDB allow clusters to continue running in a "reduced" K-safety mode after a hash mismatch occurs, rather than shutting down. In reduced mode the extra partition copies are stopped to avoid any data divergence. However, in certain cases when this happened, CPU usage could eventually spike on individual nodes in the cluster. This problem has been resolved.

- There was a race condition where, when using database replication (either passive DR or XDCR), applying multiple schema changes to the consumer cluster could cause the cluster to crash with a SIGSEGV error. This problem has been resolved. However, it is still strongly recommended when applying schema changes on DR clusters to process the DDL statements in batch mode using the **sqlcmd** `file -batch` directive. Batch processing can greatly reduce the possibility of divergence occurring between the clusters.

27. Release V10.2.3 (March 25, 2021)

27.1. Support for Kubernetes 1.19

VoltDB and the VoltDB Operator now support Kubernetes 1.19.

27.2. Recent improvements

The following limitations in previous versions have been resolved:

- There was an issue regarding tasks and directed procedures, where modifying the class (with LOAD CLASSES) for a directed procedure associated with a task that was already running could cause the database to fail with an error stating that active transactions were "moving backwards". This issue has been resolved.

- In certain situations, if an XDCR cluster stopped and recovered using command logs, some partitions on the restarted cluster would not resume consuming data from the other clusters in the XDCR relationship. A possible workaround was to perform a rolling restart of the cluster nodes. However, this issue has now been resolved.

- There was a problem in the Prometheus agent for VoltDB, where database replication (DR) statistics for the DR consumer were not being reported correctly. This issue has been resolved.

28. Release V10.2.2 (March 2, 2021)

28.1. Support for including additional content through Kubernetes persistent volumes

You can now identify additional content — such as schema files, stored procedure classes, and third-party JAR files — to be included when initializing a VoltDB database on Kubernetes by specifying their location in the `additionalVolumes` and `additionalVolumeMounts` properties. Mounting persistent volume

claims to */etc/voltdb/schema*, */etc/voltdb/classes*, and */etc/voltdb/extension* are equivalent to using the `voltdb init --schema` argument, the `--classes` argument, or including JAR files in the `/lib/extension` folder where VoltDB is installed on non-Kubernetes servers.

28.2. Remove requirement for Python 2.7.13 inadvertently added in an earlier release

Improvements associated with SSL/TLS and IPv6 inadvertently added a requirement for Python version 2.7.13 in VoltDB versions 10.2 and 10.1.1. This constraint has been corrected and VoltDB now accepts Python version 2.7.5 and later.

28.3. Additional improvements

The following limitations in previous versions have been resolved:

- Previously, it was possible for a final shutdown snapshot to stall due to "unacknowledged transactions" in export. This could happen if an export stream was declared, but the associated export connector was set to `enabled="false"` in the configuration. If data was then written into the stream and a final shutdown snapshot requested (using the `voltadmin shutdown --save` command), the shutdown could not finish due to the pending data in the queue. This issue has been resolved and pending data in disabled queues is ignored.

- There was a rare condition where, if a node in a K-safe cluster failed while a snapshot was being initiated, the cluster did not properly cleanup the aborted snapshot. As a result, no subsequent snapshots could be started, including the snapshot needed to transfer data to the failed node when it tried to rejoin. This issue has now been resolved.

- There was an issue in the cron scheduler for user-defined tasks (that is, tasks defined using CREATE TASK ON SCHEDULE CRON...). As a consequence of the error, the tasks were always scheduled for immediate execution. This issue has now been resolved.

29. Release V10.2.1 (January 21, 2021)

29.1. Initial Kubernetes release corrected

The initial release of 10.2 on Kubernetes included the wrong Docker image. This issue is resolved by the 10.2.1 point release. Do ***not*** use the initial application and helm chart versions (10.2.0 and 1.3.0). Please be sure to use the latest releases, which are 10.2.1 and 1.3.1 respectively.

This change affects the Kubernetes release of VoltDB only.

30. Release V10.2 (January 19, 2021)

30.1. Configuration updates available in Kubernetes

The VoltDB Operator for Kubernetes now supports changes to cluster and database configuration properties while the database is running. For properties that can be changed dynamically, the change occurs immediately. For other properties, the Operator orchestrates a cluster restart or rolling upgrade, as needed. See the chapter on updates and upgrades in the *VoltDB Kubernetes Administrator's Guide* for details.

30.2. DR initialization snapshots changed to asynchronous processing

At the beginning of database replication (DR), a snapshot of the database is created and sent to the joining cluster. Previously, the initialization snapshot was created as a synchronous snapshot — blocking transactions on the existing database until initialization is complete. However, depending on the size of the database, the snapshot could take a significant amount of time to complete, stalling ongoing database transactions until the snapshot is complete.

This release changes the processing of DR initialization snapshots from synchronous to asynchronous. The asynchronous snapshot eliminates the interruption to ongoing work on the active cluster. The one drawback to this change is, when using cross datacenter replication (XDCR) with more than two clusters, if a node fails on the active cluster during the initialization snapshot, existing XDCR connections to other clusters may be lost and need to be reset.

30.3. DR binary log handling improved for multi-cluster XDCR

Database replication (DR) is managed by passing binary logs between the participating clusters. The DR consumer acknowledges packets after they have been applied. If the consumer falls behind and has no room in its queue, it throws away additional packets and waits to request them again when it is ready. However, for multi-cluster XDCR environments, this means all clusters are constrained by the latency of the slowest cluster.

Starting with VoltDB V10.0, the management of binary logs was enhanced to track the queuing and acknowledgement of packets for each cluster separately. This means that each DR consumer can process packets at an optimal speed. To help understand the impact of this change, extra fields have been added to the return results of the DRCONSUMER and DRPRODUCER selectors for the @Statistics system procedure. See the description of @Statistics in the *Using VoltDB* manual for more information.

30.4. Additional improvements

The following limitations in previous versions have been resolved:

- There was an issue where a stream could stop writing data to its export target after having more than two billion rows inserted into any one partition. The problem surfaced only after the necessary number of records (approximately 2.15 billion) were written to the export connector and the database was saved, shutdown, restarted, and restored. After the snapshot was restored, no further records were written to the target by the export connector.

  This issue has now been resolved. In fact, upgrading to this release using the standard **voltadmin shutdown --save** command, installing 10.2, and then restarting the database will automatically circumvent the issue.

- There was a rare condition where using the CAST function to convert a VARCHAR column to an integer for numeric comparison (for example, `CAST(IQ AS INT) > 140` where *IQ* is a VARCHAR column) could produce an incorrect result. This would only occur if the table containing the column had an index and that index was selected to optimize the query. This issue has been resolved.

- The New Relic latency graph data has been adjusted to improve accuracy.

- The VoltDB Prometheus agent supports monitoring a subset of available statistics, using the `--stats` and `--skipstats` options. However, in earlier VoltDB v10.1 releases, use of these options could cause the agent to hang. This issue was resolved in VoltDB 10.1.2.

- Previously, when running VoltDB in Kubernetes, there were situations when the Helm charts would ignore the `serviceAccountName` if the `global.rbac.create` property was set to false. This issue has been resolved. To use a separately created service account, you must:

  - Set the properties `operator.serviceAccount.name` and `cluster.serviceAccount.name` to match the account in question

  - Set the properties `operator.serviceAccount.create` and `cluster.serviceAccount.create` to false.

- Under certain circumstances, previous versions of the VoltDB Operator for Kubernetes mistakenly used the underlying system, instead of the virtualized container, when calculating available memory. This issue has been resolved.

31. Release V10.1.3 (December 18, 2020)

31.1. Internal improvements to VoltDB Operator

Code improvements to optimize the software upgrade process.

32. Release V10.1.2 (December 15, 2020)

32.1. Adjustments and optimizations for Kubernetes settings

Several settings associated with Kubernetes have been adjusted to provide a better experience when starting and running VoltDB in a Kubernetes environment. Those optimizations include:

- Reducing the timeout for pod liveness and readiness from 3 minutes to 90 seconds.
- Changing the loopback address to a dynamic lookup rather than assuming IPv4 is in use.

32.2. Using load balancers to connect XDCR clusters in different Kubernetes domains

The Helm charts for Kubernetes now allow for alternate methods of establishing a network mesh between clusters for cross datacenter replication (XDCR). In particular, you can now use per-pod load balancers so the clusters can connect to each other through externally available IP addresses. See the *VoltDB Kubernetes Administrator's Guide* for details.

32.3. Security Notice

A number of libraries included in the VoltDB distribution have been updated to eliminate security vulnerabilities, including Guava, Jackson, Jetty, Kafka, Log4J, and Netty

32.4. Additional improvements

The following limitations in previous versions have been resolved:

- There was a problem with the Kinesis importer where the importer could fail with a "no class found" error. This issue has been resolved.
- There was an rare situation where if a schema failed causing a deadlock, subsequent attempts to rejoin nodes to the cluster would fail. This issue has been resolved.
- Two issues associated with the JDBC export connector were identified and fixed. First, when inserting into an Oracle database via the JDBC export connector, it was possible for the export threads to get blocked if the commit failed. Second, it was possible for an insert into MySQL via the JDBC connector to fail if the table definition required duplicate keys. These issues have now been resolved.
- There was an issue in the export subsystem where, it was possible that releasing an export queue with missing records could result in more records being deleted from the queue than necessary. Normally releasing an export queue with a gap means the export connector "jumps" to the next record after the missing data. However, if — after the queue pauses at a gap — the database schema was updated before the release command is issued, it was possible for additional records unaffected by the gap to be deleted from the queue. This issue has been resolved.

33. Release V10.1.1 (November 13, 2020)

33.1. Support for IPv6

VoltDB now supports both IPv4 and IPv6 networking. This includes support for IPv6-only environments. When entering IPv6 network addresses, be sure to enclose the address in square brackets. See the implementation note concerning IPv6 addresses for details.

34. Release V10.1 (October 30, 2020)

34.1. Support for multiple VoltDB databases in the same Kubernetes cluster

With the original release of VoltDB V10.0 and the VoltDB Operator, you could run multiple VoltDB databases in separate Kubernetes clusters or in separate namespaces within a single cluster. You can now run multiple databases within the same Kubernetes cluster and namespace. To do this, you start by running a single copy of the VoltDB Operator, using the following steps:

1. Start the VoltDB Operator by itself (`helm install operator voltdb/voltdb --set cluster.enabled=false`). After the Operator pod is ready...
2. Start the first database without an Operator (`helm install db1 voltdb/voltdb --set operator.enabled=false`)
3. Start the second database without an Operator (`helm install db2 voltdb/voltdb --set operator.enabled=false`)
4. And so on.

When running multiple databases within the same namespace, the only proviso is that you must not stop and delete the Operator until all of the databases it supports are stopped and deleted.

34.2. Support for future upgrades in Kubernetes

Another change to the VoltDB Operator for Kubernetes provides support for future upgrades to VoltDB installations. Although not available for upgrading V10.0 to V10.1, this new functionality will allow scripted upgrades for all future versions of VoltDB in Kubernetes.

For the initial V10.0 release of the VoltDB Operator, the one-time process for upgrading to V10.1 is:

1. Update the VoltDB charts in Helm:

```
$ helm repo update
helm search repo voltdb/voltdb
```

2. Verify that you have the latest charts. The following command should show version 10.1 for VoltDB, and version 1.1.0 or later for the VoltDB Operator and the Helm chart:

```
$ helm search repo voltdb/voltdb
```

3. Shutdown VoltDB taking a snapshot and making sure you do not delete the persistent volume on which the database root directory is stored. This example is shutting down the database call mydb:

```
$ helm upgrade mydb voltdb/voltdb -- version 1.0.2 -reuse-values \
--set cluster.clusterSpec.deletePVC=false \
--set cluster.clusterSpec.takeSnapshotOnShutdown="Always" \
--set cluster.clusterSpec.replicas=0
```

4. Wait for all the cluster pods to be removed from Kubernetes. Then delete the Helm release:

```
$ helm delete mydb
```

5. Wait for the VoltDB Operator pod to be removed from Kubernetes. Then reinstall the Helm release with the latest version:

```
$ helm install mydb voltdb/voltdb --version 1.1.0 \
```

```
[configuration properties...]
```

34.3. Improved SHOW TABLES and DESCRIBE information in sqlcmd

The sqlcmd directives SHOW TABLES and DESCRIBE have been enhanced to provide additional information about the tables in the database schema. The SHOW TABLES directive now sorts the schema objects between regular tables, data replication (DR) tables, streams, and views. Similarly, the DESCRIBE directive now distinguishes between regular tables and DR tables.

34.4. Additional information in the @Statistics and @SystemInformation system procedures

Both the @Statistics and @SystemInformation system procedures have been enhanced to provide additional information. The @Statistics TABLE selector now includes two additional columns indicating whether the table is a DR table or not and, if it is defined as an export table, the name of its export target. The @SystemInformation DEPLOYMENT selector now includes rows for additional paths, such as DR and export overflow and cursors, when appropriate.

34.5. Kafka import and export support Kafka version 2.6.0 and later

The Kafka services within VoltDB (including Kafka import and export) support Kafka version 2.6.0 and later. Support for earlier versions of Kafka is deprecated.

34.6. Additional improvements

The following limitations in previous versions have been resolved:

- The **snapshotconvert** utility has been corrected to interpret null values as end-of-file, rather than reporting an error. At the same time, general error handling has been enhanced and extended to report more detailed information when a failure occurs.
- The VoltDB bulk loader (available in the client API and used in the loader utilities such as csvloader) has been optimized to remove an unnecessary regular expression evaluation of string columns. This change produces a noticeable improvement in load times for large data sets.
- A number of edge cases were discovered that could cause a database deadlock. These situations — some race conditions, some the consequence of unusual failures during a schema change — have now been resolved.
- VoltDB V10.0 introduced a change that caused the New Relic monitoring plugin to fail. This issue has been resolved.
- For its original release, the VoltDB Operator supported Kubernetes versions 1.16.2 through 1.17.x. The Operator now supports Kubernetes versions 1.18.x as well.
- Previously, when attempting to configure XDCR in a Rancher Kubernetes environment, the nodes would not initialize properly. This issue is now resolved.
- The Prometheus agent for VoltDB has been updated to improve the accuracy of the information being reported.
- VoltDB Operator V10.1 changes the location of the VoltDB root directory under Kubernetes from `/pvc/voltdb/{release}-voltdb-cluster/voltdbroot` in V10.0 to `/pvc/voltdb/voltdbroot/` in V10.1. When creating a new database, the Operator creates the root directory in the new location. For existing instances (upgraded using the process described above), the Operator keeps the older existing location.

35. Release V10.0 (August 12, 2020)

35.1. New VoltDB Operator for Kubernetes

VoltDB now offers a complete solution for running VoltDB databases in a Kubernetes cloud environment. VoltDB V10.0 provides managed control of the database startup process, a new VoltDB Operator for coordinating cluster activities, and Helm charts for managing the relationship between Kubernetes, VoltDB and the Operator. The VoltDB Kubernetes solution is available to Enterprise customers and includes support for all VoltDB functionality, including cross data center replication (XDCR). See the *VoltDB Kubernetes Administrator's Guide* for more information.

35.2. New Prometheus agent for VoltDB

For customers who use Prometheus to monitor their systems, VoltDB now provides a Prometheus agent that can collect statistics from a running cluster and make them available to the Prometheus engine. The Prometheus agent is available as a Kubernetes container or as a separate process that can either run on one of the VoltDB servers or remotely and makes itself available through port 1234 by default. See the README file in the `/tools/monitoring/prometheus` folder in the directory where you install VoltDB for details.

35.3. Enhancements to Export

Recent updates to export provide significant improvements to reliability and performance. The key advantages of the new export subsystem are:

- **Better throughput** — Initial performance tests demonstrate significantly better throughput on export queues using the new subsystem over previous versions of VoltDB.
- **Adjustable thread pools** — The new subsystem let's you set the thread pool size for export as a whole or to define thread pools for individual connectors.
- **Fewer duplicate rows** — When cluster nodes fail and rejoin the cluster, the export subsystem resubmits certain rows to ensure they are delivered. The new subsystem keeps better track of the acknowledged rows and does not need to send as many duplicates to maintain the same level of durability.

35.4. Improved license management

Starting with VoltDB V10.0, specifying the product license has moved from the **voltdb start** command to the **voltdb init** command. In other words, you only have to specify the license once, when you initialize the database root directory, rather than every time you start the database. When you do specify the license on the **init** command, it is stored in the root directory the same way the configuration is.

The same rules apply about the default location of the license as before. So if you store your license in your current working directory, your home directory, or the `/voltdb` subfolder where VoltDB is installed, you do not need to include the **--license** argument when initializing the database. Also note that the **--license** argument on the **voltdb start** command is now deprecated but still operational. So if you have scripts to start VoltDB that include **--license** on the **start** command, they will continue to work. However, we recommend you change to the new syntax whenever convenient because support for **voltdb start --license** may be removed in some future major release.

35.5. Support for RHEL and CentOS V8

After internal testing and validation, RHEL and CentOS V8 are now supported platforms for production use of VoltDB.

35.6. RabbitMQ export connector removed

The export connector for RabbitMQ was deprecated in VoltDB version 9 and has now been removed from the product.

35.7. Ubuntu 14.04 no longer supported as production platform

Ubuntu 14, which is no longer supported by Canonical, has been dropped as a production platform for VoltDB.

35.8. Additional improvements

The following limitations in previous versions have been resolved:

- There was a rare edge case where, if a schema change failed due to an internal error and was retried, the cluster could crash with a null pointer exception. This issue has been resolved.
- There was an issue where attempting to insert a row with all null values into a stream with at least one column that allows null values, the server could crash. This issue has been resolved.
- Due to issues in the underlying library used, it was possible for the JSON functions to return results in a different order on different servers, causing a hash mismatch error. This inconsistency, and the resulting issue, have now be resolved.
- Under certain conditions while using the JDBC export connector, altering the stream associated with the connector could cause export to fail. The problem was that the schema change requires an update to the prepared statement used to write to the JDBC target. But if the `createtable` property was set to false or the `ignoregenerations` property set to true, the prepared statement was not updated. This issue has been resolved.
- There was an issue where a query with a complex ORDER BY clause with two separate column expressions, both using the DECODE() function, could return incorrect results. This issue has been resolved.
- Previously, if a user-defined aggregate function threw an exception, the function failed but the specific exception was not passed back to the calling application. Instead, a generic exception was returned. This issue has been resolved and user-defined aggregate functions now return the correct exception.

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 @Statistitcs 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 * 20MB 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 = ~ 320MB

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 **voltadmin status --dr** command does not work if clusters use different client ports

The **voltadmin 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. XDCR in Kubernetes supports two databases only.

You can configure and run an XDCR environment in Kubernetes using the VoltDB Operator. However, the XDCR environment is currently limited to two databases at a time.

3.8. 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.

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 **voltdb** 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 an 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 **voltdb start** command.
- For an individual port, specify the interface and port on the command line. For example **voltdb 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.

4.4. Using VoltDB where the /tmp directory is noexec

On startup, VoltDB extracts certain native libraries into the /tmp directory before loading them. This works in all standard operating environments. However, in custom installations where the /tmp storage is mounted with the "noexec" option, VoltDB cannot extract the libraries and, in the past, refused to start.

For those cases where the /tmp directory is assigned on storage mounted with the ‘noexec’ option, you can assign an alternative storage for VoltDB to use for extracting and executing temporary files. This is now done automatically on Kubernetes and does not require any user intervention.

On non-Kubernetes environments, you can identify an alternative location by assigning it to `volt.tmpdir`, `org.xerial.snappy.tmpdir`, and `jna.tmpdir` in the VOLTDB_OPTS environment variable before starting the server process. The specified location must exist, must be an absolute path, and cannot be on storage mounted with the "noexec" option. For example, the following command assigns an alternate temporary directory called `/volttemp`:

```
export VOLTDB_OPTS="-Dvolt.tmpdir=/volttemp \
                    -Dorg.xerial.snappy.tempdir=/volttemp \
                    -Djna.tmpdir=/volttemp"
```

When using an alternate temporary directory, files can accumulate over time since the directory is not automatically purged on startup. So you should make sure you periodically delete old files from the directory.

5. Platforms

5.1. Kubernetes Compatibility

See the *Volt Kubernetes Compatibility Chart* for information on which versions of the Volt Operator and Helm charts support which version of VoltDB. See the *VoltDB Operator Release Notes* for additional information about individual releases of the VoltDB Operator.