Release Notes

Product	Volt Active Data
Version	13.1.3
	VoltDB Operator 2.3.3 VoltDB Helm Chart 2.3.3
Release Date	February 7, 2024

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@voltactivedata.com. Thank you.

Upgrading From Older Versions

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

1. Shutdown the database, creating a final snapshot (using **voltadmin shutdown --save**).

2. Upgrade the VoltDB software.

3. Restart the database (using **voltdb start**).

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

Finally, for all customers upgrading from earlier versions of VoltDB, please be sure to read the *Volt Upgrade Guide*, paying special attention to the notes for your current and subsequent releases, including *V8*, *V10*, *V11*, and *V12*.

Changes Since the Last Release

In addition to the major enhancements listed above, users of previous versions of VoltDB should take note of the following changes that might impact their existing applications. See the *VoltDB Operator Release Notes* for changes specific to the use of VoltDB on the Kubernetes platform.

1. Release V13.1.3 (February 7, 2024)

1.1. Additional improvements

The following limitations in previous versions have 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 effect was amplified during an in-service upgrade of an XDCR cluster, because producer nodes are stopped and restarted in series, in some cases causing XDCR replication to stall for the duration of the upgrade. This issue has been resolved.

- In some circumstances, if a pod failed to start with the new version during an in-service upgrade, subsequent attempts to rollback to the previous version would also fail. This issue has been resolved.

2. Release V13.1.2 (January 31, 2024)

2.1. Improvements to In-Service Upgrade

This release provides significant improvements to the in-service upgrade process, eliminating most interruptions and errors returned by in-flight transactions due to servers stopping and restarting. In the initial release of in-service upgrade, stopping the individual nodes could result in transactions returning an error if the queue stopped before the transaction completed. This issue was exacerbated in Kubernetes environments by the Operator prematurely stopping the pod before the server process completed its shutdown procedure. Both of these situations have been significantly improved through better coordination of the server, the client, and the operator.

Please note, that these improvements depend on changes to all three components and are only available if you upgrade all three. That is, using the latest Volt Operator, Volt server software, and linking your client applications against the latest Java client library (2.3.2 or later of the operator and V13.1.2 or later of the server software and client library).

2.2. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2023-7104

2.3. Additional improvement

The following limitation in previous versions 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 were situations where a topology-aware client might fail to acknowledge all change announcements, leading to the client not connecting to all available servers. This was most likely to occur with large clusters after the cluster restarts or where the client application starts before the cluster starts.

  This issue has been resolved. However, it is important to note that the fix is in the Java client, *not* the server software. So to resolve this problem your client applications must be run against the latest version (V13.1.2) of the Volt Client Library JAR.
- In some circumstances, if a pod failed to start with the new version during an in-service upgrade, subsequent attempts to rollback to the previous version would also fail. This issue has been resolved.

3. Release V13.1.1 (January 11, 2024)

3.1. Additional improvements

The following limitations in previous versions have been resolved:

- In Kubernetes, a bug in the Go language processor was causing the Volt Operator to mistakenly update pod settings repeatedly. Although normally this would go unnoticed, in the case of services using the Border

Gateway Protocol (BGP) -- such as certain load balancers used for negotiating XDCR connections -- the repeated updates could result in network issues, excessive latency, and, ultimately, XDCR connection failures. This issue has been resolved.

- As a result of extensive internal testing, a few rare edge cases that could cause node rejoins to fail have been identified and corrected.

4. Release V13.1.0 (December 21, 2023)

4.1. ARM 64

Volt Active Data now supports the Arm CPU architecture — and specifically Aarch64 — as a hardware platform for production use of VoltDB. Talk to your Volt Active Data sales representative for more information.

4.2. In-Service Upgrade

Volt Active Data now supports *in-service upgrades*: the ability to upgrade the VoltDB server software for a single cluster without requiring any downtime (uninterrupted upgrades using multiple Active(N) clusters have been available for some time now). In-service upgrade is a new, licensable feature where you remove, upgrade, and rejoin individual nodes from the cluster one at a time. On Kubernetes, the Volt Operator automates this process through a new property, `cluster.clusterSpec.enableInServiceUpgrade`. See the sections on in-service upgrades in the *Volt Administrator's Guide* and *Volt Kubernetes Administrator's Guide* for more information.

4.3. Elastically reducing the cluster size in Kubernetes

It is now possible to both elastically grow and shrink clusters in the Kubernetes environment. Reducing the cluster size is just a matter of setting the replica count to the appropriate value. (You can only reduce the cluster by K+1 nodes at a time.) Once initialized, the Volt Operator automates the process and provides periodic status information through Kubernetes events and object properties. See the sections on elastically resizing the cluster in the *Volt Kubernetes Administrator's Guide* for more information.

4.4. Diagnostics tools for Kubernetes

Volt now provides maintenance and management tools optimized for Kubernetes as a separately installable pod. The diagnostics pod provides access to **sqlcmd** as well as the **collect** command for gathering logs and other diagnostics data from the cluster. See the appendix on diagnostics tools in the *Volt Kubernetes Administrator's Guide* for more information.

4.5. FORMAT() Function

There is a new SQL function, FORMAT(), that formats a text string, using the standard replacement placeholders found in functions such as printf to insert values from table columns and other SQL expressions.

4.6. New license file format

VoltDB uses a new license file format with a more flexible structure to allow for new features in the future. With the introduction of the new file format, this release of Volt Active Data no longer accepts older, outdated license formats. This means that before upgrading existing databases, *please make sure you are using a recent license issued in the past two years*.

To verify that you have a recent license, connect to your database server and issue the command **voltadmin inspect**. If the command reports a permit version of 3 or 4, you are all set. If the command reports a permit version of 2 or less, or returns an error that no such command exists, you will need to replace your license before upgrading. Talk to your Volt Active Data customer support representative to receive a replacement license.

In addition to a new license format, several minor changes related to license management have been made:

- The @SystemInformatin OVERVIEW system procedure no longer reports on individual licensed features, but conitnues to report on the license type and expiration date.
- The @SystemInformation LICENSE system procedure now reports on all licensed features, not just command logging and database replication.

4.7. Java 8 use deprecated for VoltDB servers

Use of the Java 8 runtime to run the VoltDB server software is now deprecated. The recommended Java versions are Java 11 and 17. Note that Java 8 is still supported for running applications using the VoltDB client API.

Although deprecated, use of Java 8 to run the VoltDB server continues to work at this time. However, support for such usage will be removed in a future release.

4.8. Passive Database Replication deprecated

The original passive database replication (DR) functionality involving one master and one replica cluster, is now deprecated. The ability to set up passive replication using Active(N) — also known as XDCR — easily and more flexibly (with multiple masters, any mix of active masters and passive replicas) makes a separate and less functional alternative impractical. By focusing on a single, integrated set of capabilities for both active and passive DR, the product can move forward with new capabilities more rapidly.

The separate passive DR implementation continues to function in V13. However, it will be removed from the product in some future major release.

4.9. Volt support for Helm 3.11 and later

To match the currently supported versions of Helm, Volt Active Data now requires Helm 3.11 or later when running in Kubernetes.

4.10. Additional improvements

The following limitations in previous versions have been resolved:

- In recent releases (starting with V12.3), events in the VMC service were not being logged properly under Kubernetes. This issue has been resolved
- Volt no longer checks if TCP offload or generic receive offload are disabled. Previously, these settings had to be disabled for VoltDB to start. By eliminating the check, it is possible to run VoltDB in environments, most notably Kubernetes, where these OS features may be enabled.
- 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.
- Rigorous testing of failure scenarios uncovered a rare condition associated with tables declared with the EXPORT TO TARGET or EXPORT TO TOPIC clause and actively exporting during a sequence of node failures and rejoins. A race condition while nodes are dropping and rejoining could potentially result in the table not queueing all appropriate rows to the associated target. This issue has been resolved.

- There was a race condition associated with the TTL (time to live) feature during a rejoin operation. It was possible, under certain limited circumstances, for a TTL operation during a rejoin to cause data to silently diverge. This issue has been resolved.

- Another race condition, in command logging this time, could cause the cluster to crash on startup. This condition could only be triggered if the command logs contained a schema update in the binary logs. This issue has been resolved.

- There are reports that VoltDB does not operate correctly when the networking environment is configured to use *jumbo frames*. The symptoms are that TCP packets with MTU>9000 are dropped. However, this is not a Volt-specific issue. When configuring the network to use jumbo frames, it is crucial to thoroughly test the network to guarantee that TCP packets are not fragmented or have their segments reordered during reassembly. If not, there is a danger of lost packets in the network layer, unrelated to the specific application involved.

5. Release V13.0.3 (November 27, 2023)

5.1. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2023-4586
CVE-2023-5363

5.2. Additional improvements

The following limitations in previous versions have been resolved:

- Several errors logged by the new metrics subsystem were discovered and corrected.

- 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.

6. Release V13.0.2 (November 6, 2023)

6.1. Beta release of ARM architecture support.

We are happy to announce that VoltDB 13.0.2 is available in both the traditional x86 kit and in a new ARM64 architecture build. The ARM build is a beta release for testing only, it is not intended for production use.

The only known limitation to the beta release is that the separate Volt Management Center (VMC) pod for Kubernetes does not work yet. The workaround is to set the `vmc.enabled` flag to *false* when starting the cluster:

```
$ helm install mydb voltdb/voltdb   \
   --values myconfig.yaml           \
   --set vmc.enabled=false
```

Please report any further issues and feedback to betasupport@voltactivedata.com. Thank you.

6.2. New diagnostics pod available for Volt Active Data on Kubernetes.

For Kubernetes users, there is a new tool, the diagnostics pod, that automates the collection of logs and other diagnostic information that is needed when debugging issues you report to the Volt support team. The diagnostics pod is a standalone product that is installed separately, so can be added to existing VoltDB installations without affecting ongoing operations. The diagnostics pod is available from the same repository as the Volt Active Data server software and can be installed using Helm:

```
$ helm install volttools voltdb/volt-diagnostics
```

Instructions are available in the readme included in the kit:

```
$ helm show readme voltdb/volt-diagnostics
```

6.3. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2023-34455
CVE-20 3-44487

6.4. Additional improvements

The following limitations in previous versions have been resolved:

- In releases of Volt Active Data since V12.0, when restoring a snapshot from a larger cluster to a smaller cluster (for example, moving a snapshot from a five node to a three node cluster), it was possible for the restore operation to run out of heap memory, even though the new cluster had sufficient memory for the saved data. This issue has been resolved.

- Previously, TLS/SSL encryption was enabled for the new metrics port whenever SSL was enabled for any other VoltDB port on the server, whether internal or external. TLS/SSL should only be enabled for the metrics port when it is enabled for external ports. This issue has been resolved.

7. Release V13.0 (September 30, 2023)

7.1. New Prometheus metrics system

Volt V13 announces the general production release of a new metrics system, where every server in the cluster reports its own data through a Prometheus-compliant endpoint. You enable Prometheus metrics in the configuration file when initializing the database by adding the <metrics> element to the Volt configuration file:

```
<deployment>
 <cluster kfactor="1"/>
 <metrics enabled="true"/>
</deployment>
```

Once enabled, each Volt server reports server-specific information through the metrics port, which defaults to 11781. The new metrics system replaces the standalone Prometheus agent, which has now been deprecated. See the sections on integrating with Prometheus in the *Volt Administrator's Guide* and *Volt Kubernetes Administrator's Guide*.

7.2. New Grafana dashboards

To match the new metrics system, Volt provides sample Grafana dashboards to help visualize your database's performance and status. The dashboards are available from github:

- https://github.com/VoltDB/volt-monitoring/tree/main/dashboards/Volt-V13.x/new-metrics

7.3. Support for alternate character sets

Volt provides full support for international characters through its use of UTF-8 for storing and displaying textual data. However, there are other character encodings that may be in use by customers, including both older, established encodings such as Shift_JIS and newer encodings like the Chinese GB18030-2022 standard. For customers using alternate character encodings as the default in their client environment, Volt now provides automatic conversion of character encodings both interactively and for file input. The **sqlcmd** utility automatically converts to and from the terminal session's localized character set for input and display. When reading and writing files, you can use the `--charset` qualifier to specify the character encoding of the file, both with the **sqlcmd** and **csvloader** utilities.

7.4. Improved command line interface for the sqlmd utility

In addition to full character set support, the **sqlcmd** utility has a more complete and consistent set of command line qualifiers. The new `--output-file` qualifier captures the output of SQL statements; it does not echo the commands or informational messages. When used with the `--output-format=csv` qualifier, `--output-file` now generates valid CSV files without extraneous lines. Similarly, the `--file` qualifier lets you process a file containing any valid **sqlcmd** statements or directives. And if the file contains only data definition language (DDL) statements, you can use the `--batch` qualifier to process all of the statement as a single batch significantly reducing the time required to update the schema. See the **sqlcmd** reference page for these and other improvements to the **sqlcmd** utility.

7.5. Beta support for ARM architecture

Beta software kits for running the VoltDB server software natively on ARM64 architecture are now available. If you are interested in participating in the ARM64 beta program, please contact your Volt Customer Success Manager.

7.6. Support for Kubernetes 1.27

Volt Active Data now supports version 1.27 of the Kubernetes platform. See the *Kubernetes Compatibility Chart* for details on what versions of Kubernetes are supported by each version of VoltDB.

7.7. Security updates

All known high and critical CVEs were resolved as of the date of this release, including:

CVE-2022-48174
CVE-2023-3341
CVE-2023-4236

7.8. Additional improvements

The following limitations in previous versions have been resolved:

- Volt V12.3 introduced the ability to alter DR tables without pausing the database. Unfortunately, after altering the table, 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.
- In recent versions of Volt (starting with V12.0), if a schema change coincided with a snapshot save operation, it was possible under certain circumstances for the schema change to hang and never complete. This issue has been resolved.

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. Multi-cluster XDCR environments require command logging.

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

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

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

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

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

4. TTL

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

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

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

5. Export

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

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

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

6. Import

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

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

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

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

7. SQL and Stored Procedures

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

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

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

7.2. Do not use assertions in VoltDB stored procedures.

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

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

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

8. Client Interfaces

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

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

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

9. SNMP

9.1. Enabling SNMP traps can slow down database startup.

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

10. TLS/SSL

10.1. Using Commercial TLS/SSL certificates with the command line utilities.

The command line utilities, such as **sqlcmd**, **voltadmin**, and the data loaders, have an `--ssl` flag to specify the truststore for user-created certificates. However, if you use commercial certificates, there is no truststore. In that case, you need to specify an empty string to the `--ssl` command qualifier to access the database. For example:

```
$ sqlcmd --ssl=""
```

10.2. Bypassing TLS/SSL verification with **voltadmin**.

When TLS/SSL is enabled, the command line utility **voltadmin** assumes you want to verify the authenticity of the database server. If you are using user-created certificates, you must provide the associated truststore on the command line using the `--ssl` flag. However, if you trust the server and do not need to verify the certificate, you can use the syntax `--ssl=nocheck` to bypass the verification. For example:

```
$ voltadmin --ssl=nocheck
```

This is particularly important if you exec into the shell of a Kubernetes pod running Volt with TLS/SSL enabled, since the Volt Docker image includes a slimmed-down set of command line tools not designed for accessing databases outside of the pod.

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. XDCR

2.1. Upgrading existing XDCR clusters for Dynamic Schema Change

Volt Active Data V12.3 introduces a new feature, dynamic schema change for XDCR clusters. To use this feature, clusters must be configured to enable the feature and must be using the latest DR protocol. However, existing clusters that upgrade from earlier versions will not automatically use the new protocol. Instead, you must explicitly upgrade the DR protocol using the **voltadmin dr protocol --update** command.

First, to use dynamic schema change you must enable the feature in the configuration file. This must be done when you initialize the cluster which, for existing XDCR clusters, is easiest to when performing the version upgrade. To upgrade XDCR clusters, you must drop the cluster from the XDCR environment, upgrade the software, then reinitialize and restart the cluster. While reinitializing the cluster, add the `<schemachange enabled="true"/>` element to the configuration file. For example:

```
<deployment>
  <dr id="1" role="xcdr">
    <schemachange enabled="true"/>
    <connection source="paris.mycompany.com,rome.mycompany.com"/>
  </dr>
</deployment>
```

Similarly, for Kubernetes, the YAML would be:

```
cluster:
  config:
    deployment:
      dr:
        id: 1
        role: xdcr
        schemachange:
          enabled: true
        connection:
            source: paris.mycompany.com,rome.mycompany.com
```

Once all of the clusters are upgraded to the appropriate software version, you can upgrade the DR protocol. When used by itself, the **voltadmin dr protocol** command displays information about what versions of the DR protocol the XDCR clusters support and which version they are using. When the XDCR relationship starts, the clusters use the highest supported protocol. However, when an existing XDCR group is upgraded, they do not automatically upgrade the originally agreed-upon protocol. In this case, you must go to each cluster and issue the **voltadmin dr protocol --update** command to use the highest protocol that all the clusters can support. Once you issue the **voltadmin dr protocol --update** command on all of the clusters, you are then ready to perform dynamic schema changes on your XDCR environment.

3. Volt Management Center

3.1. 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.

3.2. TLS/SSL for the HTTPD port on Kubernetes

Prior to VoltDB V12.0, encryption for the httpd port which VMC uses was automatically enabled on Kubernetes when you enabled TLS/SSL for the server using the `cluster.config.deployment.ssl.enabled` property. You could then selectively enable SSL for other ports using the `.dr`, `.internal`, and `.external` subproperties of `cluster.config.deployment.ssl`.

Starting with V12.0, VMC on Kubernetes is run as a separate service by default and you can enable or disable TLS/SSL encryption for VMC independently using the `vmc.service.ssl...` properties. However, if you choose not to run VMC as a separate service, by setting the `vmc.enabled` property to "false", VMC encryption is managed the same way as in earlier releases using the `cluster.config.deployment.ssl...` properties as described above.

4. SQL

4.1. 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.

4.2. 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;
```

4.3. 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.

4.4. 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
    . . .
```

5. Runtime

5.1. 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.

5.2. 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.

5.3. 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.

5.4. 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.

5.5. Text Data and Character Sets

Volt Active Data processes and stores text data as UTF-8 encoded strings. When using the Volt Java API, the Java String datatype always stores text in Unicode. So there is no conversion necessary when passing string data to Volt stored procedures. If the application must handle data in alternate character encodings you can use existing Java functionality to convert the data to Unicode on input. For example, using the second argument to specify the character encoding when instantiating a buffered reader:

```
BufferedReader myfile = Files.newBufferedReader(filename,charset);
```

On the other hand, when users enter data interactively using an alternate character encoding, Volt automates the conversion of the character set of the current session to UTF-8 on input and output. In other words, if the user's session is localized to use a specific character set, when entering data at the **sqlcmd** prompt (for example, when executing an INSERT statement) the data is automatically converted to UTF-8 before processing. Similarly, on output (such as displaying the results of a SELECT statement) the UTF-8 data is converted to the user's localized character set.

When processing text files, such as CSV files or **sqlcmd** scripts, the command line utilities provide a `--charset` qualifier that lets you specify the character set of the input file. The `--charset` qualifier affects the FILE directive and `--file` and `--output` qualifiers for **sqlcmd** as well as the input files for the **csvloader**. Note that the user's localized session character set does *not* affect file input. If the `--charset` qualifier is not used the data is assumed to be UTF-8 by default.

Finally, which character sets are supported depends on which Java virtual machine (JVM) release you are using on your servers (for export) or client machines (for **sqlcmd** and **csvloader**). For established character sets, such as Shift_JIS or ISO-8859-1, all supported JVM releases provide support. For newer character sets, you may need a more recent release of the JVM. For example, the recent Simplified Chinese character set GB18030-2022 requires a JVM released in 2023 or later. For OpenJDK this includes the following releases:

- Java 8 — release 8u32-b05
- Java 11 — release 11.0.20+8
- Java 17 — release 17.0.8+7

6. Kubernetes

6.1. Do not change the UID on the Kubernetes account used to run VoltDB

In the security context section of the Helm chart for VoltDB, the user account UID is set to 1001. This value is required. Do not change or override any of the following properties when configuring your database:

cluster.clusterSpec.podSecurityContext.runAsUser
cluster.clusterSpec.podSecurityContext.fsGroup
cluster.clusterSpec.securityContext.runAsUser
cluster.clusterSpec.securityContext.runAsGroup

6.2. OpenShift and Transparent Huge Pages (THP)

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

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

6.3. 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.