MySQL Shell 8.0 Release Notes

Abstract

This document contains release notes for the changes in each release of MySQL Shell 8.0.

For additional MySQL Shell documentation, see http://dev.mysql.com/.

Updates to these notes occur as new product features are added, so that everybody can follow the development process. If a recent version is listed here that you cannot find on the download page (https://dev.mysql.com/downloads/), the version has not yet been released.

The documentation included in source and binary distributions may not be fully up to date with respect to release note entries because integration of the documentation occurs at release build time. For the most up-to-date release notes, please refer to the online documentation instead.

For legal information, see the Legal Notices.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

Document generated on: 2019-03-20 (revision: 17442)

Table of Contents

Preface and Legal Notices

This document contains release notes for the changes in each release of MySQL Shell 8.0.

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Changes in MySQL Shell 8.0.16 (Not yet released)

Version 8.0.16 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL Shell 8.0.15 (2019-02-01, General Availability)

This release contains no functional changes and is published to align version number with the MySQL Server 8.0.15 release.

Changes in MySQL Shell 8.0.14 (2019-01-21, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- When started from the command line, MySQL Shell prints information about the product, information about the session (such as the default schema and connection ID), warning messages, and any errors that are returned during startup and connection. You can now suppress printing of information that you do not need by using the `--quiet-start[=1|2]` `mysqlsh` command-line option. With a value of 1 (the default when the option is specified), information about the MySQL Shell product is not printed, but session information, warnings, and errors are printed. With a value of 2, only errors are printed.

  As part of this work, the printed information was tidied up so that the information about the MySQL Shell product is printed before the information about the session. Also, the handling of error printing was normalized to send diagnostic data to `stderr`, and errors to `stdout`. (Bug #28833718, Bug #28855291)

- MySQL Shell connections using classic MySQL protocol now support compression for information sent between the client and the server. You can specify compression when you start MySQL Shell and connect using command line options, or in a URI string or a key-value pair when you create a session using other interfaces. You can also use the MySQL Shell configuration option `defaultCompress` to enable compression for every global session.

  For MySQL Shell connections that use Unix socket files, the `--socket` command line option can now be specified with no argument to connect using the default Unix socket file for the protocol. (Bug #28730149)

- The *`Cluster`*`.status()` operation has been extended to enable you to display information about the underlying Group Replication group used by the cluster. Now you can retrieve information from all members of a cluster without having to connect to each member individually.

  To see information about the `groupName` and `memberId`; and general statistics about the number of transactions checked, proposed, and rejected by members issue:

```
Cluster.status({extended:true})
```

To see information about recovery and regular transaction I/O, applier worker thread statistics and any lags; applier coordinator statistics, if parallel apply is enabled; error, and other information from I/O and applier threads issue:

```
Cluster.status({queryMembers:true})
```

In addition, in previous versions the URI-type string shown for `groupInformationSourceMember` in the output of *`Cluster`*`.status()` could be the cluster's MySQL Router address, rather than the address of the instance which provided the displayed group information. This has been improved to ensure `groupInformationSourceMember` always shows the correct `hostname`, or `report_host`, value and `port`, or `report_port`, value of the instance which provided the group information. (Bug #28636963, Bug #26519466, Bug #27824265, Bug #28366027)

- The MySQL Shell JSON import utility can now process BSON (binary JSON) data types that are represented in JSON documents. The data types used in BSON documents are not all natively supported by JSON, but can be represented using extensions to the JSON format. The import utility can process documents that use JSON extensions to represent BSON data types, convert them to an identical or compatible MySQL representation, and import the data value using that representation. The resulting converted data values can be used in expressions and indexes, and manipulated by SQL statements and X DevAPI functions.

  To convert JSON extensions for BSON types into MySQL types in this way, you must specify the `convertBsonTypes` option when you run the import utility. Additional options are available to control the mapping and conversion for specific BSON data types. If you import documents with JSON extensions for BSON types and do not use this option, the documents are imported in the same way as they are represented in the input file.

- A MySQL Shell configuration option `showColumnTypeInfo` and command line option `--column-type-info` have been added to display metadata for each column in a returned result set, such as the column type and collation. The metadata is printed before the result set, and is only shown in SQL mode.

  In the metadata, the column type is returned as both the type used by MySQL Shell (`Type`), and the type used by the original database (`DBType`). For MySQL Shell connections using classic MySQL protocol, `DBType` is as returned by the protocol, and for X Protocol connections, `DBType` is inferred from the available information. The column length (`Length`) is returned in bytes.

- The upgrade checker utility provided by MySQL Shell, which is the `checkForServerUpgrade()` function of the `util` global object, has several enhancements:

  - The utility can now select and provide advice and instructions for relevant checks that cannot be automated, and must be performed manually. The manual checks are rated as either warning or notice (informational) level, and are listed after the automated checks. In MySQL Shell 8.0.14, the utility provides advice where relevant about the change of default authentication plugin in MySQL 8.0.

  - A check has been added for the removed `log_syslog_*` system variables that previously configured error logging to the system log (the Event Log on Windows, and `syslog` on Unix and Unix-like systems).

  - A check has been added for specific schema inconsistencies that can be caused by the deletion or corruption of a file, including the removal of the directory for a schema and the removal of a `.frm` file for a table.

  You can access the upgrade checker utility from within MySQL Shell or start it from the command line. For instructions and further information, see MySQL Shell Utilities.

- MySQL Shell can print results in table, tabbed, or vertical format, or as pretty or raw JSON output. From MySQL Shell 8.0.14, the new MySQL Shell configuration option `resultFormat` can be used to specify any of these output formats as a persistent default for all sessions, or just for the current session. Changing this option takes effect immediately. Alternatively, the new command line option `--result-format` can be used at startup to specify the output format for a session. The existing command line options `--table`, `--tabbed`, and `--vertical` are now aliases for the `--result-format` option given with the corresponding value.

  The existing command line option `--json` controls JSON wrapping for all MySQL Shell output from a session. Specifying `--json` or `--json=pretty` turns on JSON wrapping and generates pretty-printed JSON. Specifying `--json=raw` turns on JSON wrapping and generates raw JSON. With any of these options, the value of the `resultFormat` MySQL Shell configuration option is ignored. Specifying `--json=off` or not specifying the `--json` option turns off JSON wrapping, and result sets are output as normal in the format specified by the `resultFormat` configuration option.

  The `outputFormat` MySQL Shell configuration option is now deprecated. This option combined the JSON wrapping and result printing functions, which have now been separated. If this option is still specified in your MySQL Shell configuration file or scripts, the behavior is as follows:

  - With the `json` or `json/raw` value, `outputFormat` activates JSON wrapping with pretty or raw JSON respectively.
  - With the `table`, `tabbed`, or `vertical` value, `outputFormat` turns off JSON wrapping and sets the `resultFormat` MySQL Shell configuration option for the session to the appropriate value.
- The V8 library used by MySQL Shell has been updated to version 6.7.288.46.
- AdminAPI no longer relies on the mysqlprovision check command. This work has resulted in the following:

  - The `errors` field in the JSON returned by `dba.checkInstanceConfiguration()` has been removed, because it was only used to hold errors issued by mysqlprovision. Any errors are now reported directly, for example as RuntimeError.
  - The `dba.verbose` value no longer influences the amount of debug information displayed for `dba.checkInstanceConfiguration()`, `dba.configureInstance()`, and `dba.configureLocalInstance()` because it was only used to control the verbosity of the information displayed from mysqlprovision. Instead, the generic verbose value from MySQL Shell is used to control the verbosity level for those functions.
  - In addition, the messages returned have been generally improved to make them more accurate.

  References: See also: Bug #28737777, Bug #27305806, Bug #28768627, Bug #27702439, Bug #28733883.
- When you create a cluster, you can set the timeout before instances are expelled from the cluster, for example when they become unreachable. Pass the new `expelTimeout` option to the `dba.createCluster()` operation, which configures the `group_replication_member_expel_timeout` variable on the seed instance. All instances running MySQL server 8.0.13 and later which are added to the cluster are automatically configured to have the same `group_replication_member_expel_timeout` value as configured on the seed instance.
- You can now configure an InnoDB cluster's mode while the cluster remains online. This enables you to configure the underlying Group Replication group to choose a specific instance as the new primary in single-primary mode, or to change between multi-primary and single-primary modes without taking the

cluster offline. This uses the group coordinator and the UDFs added in WL#10378, see Configuring an Online Group. Use the following operations:

- *Cluster*.setPrimaryInstance(*instance*), which forces the election of *instance* as the new primary by overriding any election process.
- *Cluster*.switchToMultiPrimaryMode(), which switches the cluster to multi-primary mode. All instances become primaries.
- *Cluster*.switchToSinglePrimaryMode([*instance*]), which switches the cluster to single-primary mode. If *instance* is specified, it becomes the primary and all the other instances become secondaries. If *instance* is not specified, the new primary is the instance with the highest member weight (and the lowest UUID in case of a tie on member weight).

- You can now check and modify the settings in place for an InnoDB cluster while the instances are online. To check the current settings of a cluster, use the following operation:

  - *Cluster*.options(), which lists the cluster configuration options for its ReplicaSets and instances. A boolean option all can also be specified to include information about all Group Replication system variables in the output.

  This work also enables you to configure the InnoDB cluster options at a cluster level or instance level, while instances remain online. This avoids the need to remove, reconfigure and then again add the instance to change InnoDB cluster options. Use the following operations:

  - *Cluster*.setOption(option, value) to change the settings of all cluster instances globally
  - *Cluster*.setInstanceOption(instance, option, value) to change the settings of individual cluster instances

  The way which you use InnoDB cluster options with the operations listed depends on whether the option can be changed to be the same on all instances or not. These options are changeable at both the cluster (all instances) and per instance level:

  - exitStateAction
  - memberWeight

  This option is changeable at the per instance level only:

  - label

  These options are changeable at the cluster level only:

  - failoverConsistency
  - expelTimeout
  - clusterName

- The *cluster*.rescan() operation has been extended to enable you to detect changes to the cluster's topology, and modfiy the cluster metadata, for example to remove old instance data. Now you can:

  - use the updateTopologyMode option to detect if the Group Replication mode (single-primary or multi-primary mode) registered in the metadata matches the current mode of the cluster, updating that information in the metadata if requested through a new option or by a prompt confirmation. You can use this option to update the metadata

after using the *`Cluster`*`.switchToSinglePrimaryMode([`*`instance`*`])` and *`Cluster`*`.switchToMultiPrimaryMode()` options added in WL#12052.

- use the `addInstances` option to specify a list of new instances to add to the metadata, or the `removeInstances` option to specify a list of obsolete instances to remove from the metadata. Pass the `auto` value to these options to automatically add or remove instances from the metadata, without having to specify an explicit list of instances. This enables the function to update the metadata even in noninteractive mode, making it consistent with the other AdminAPI operations.

- In addition, a new interactive option has been added to the *`cluster`*`.rescan()` operation, to enable or disable interactive mode prompts specifically for the *`cluster`*`.rescan()` command.

References: See also: Bug #28997465, Bug #28529362, Bug #28889563, Bug #25675665, Bug #28542904.

- The `Table` and `Collection` objects now support the `.count()` method, part of the X DevAPI.

- In 8.0.14, Group Replication introduces the the ability to specify the failover guarantees (eventual or "read your writes") if a primary failover happens in single-primary mode (see WL#11123). Configure the failover guarantees of an InnoDB cluster at creation by passing the new `failoverConsistency` option to the `dba.createCluster()` operation, which configures the `group_replication_consistency` system variable on the seed instance. This option defines the behavior of a new fencing mechanism used when a new primary is elected in a single-primary group. The fencing restricts connections from writing and reading from the new primary until it has applied any pending backlog of changes that came from the old primary (sometimes referred to as "read your writes"). While the fencing mechanism is in place, applications effectively do not see time going backward for a short period of time while any backlog is applied. This ensures that applications do not read stale information from the newly elected primary.

  The `failoverConsistency` option is only supported if the target MySQL server version is 8.0.14 or later, and instances added to a cluster which has been configured with the `failoverConsistency` option are automatically configured to have `group_replication_consistency` the same on all cluster members that have support for the option. The variable default value is controlled by Group Replication and is `EVENTUAL`, change the `failoverConsistency` option to `BEFORE_ON_PRIMARY_FAILOVER` to enable the fencing mechanism. Alternatively use `failoverConsistency=0` for `EVENTUAL` and `failoverConsistency=1` for `BEFORE_ON_PRIMARY_FAILOVER`.

  

  Note

  Using the `failoverConsistency` option on a multi-primary InnoDB cluster has no effect but is allowed because the cluster can later be changed into single-primary mode with the *`Cluster`*`.switchToSinglePrimaryMode()` operation.

Bugs Fixed

- The TAR build of MySQL Shell comes with Python 2.7. When attempting to include the site package, an error was emitted because of missing build files needed by the include. (Bug #28973138)

- Handling procedures for user-supplied data in MySQL Shell were refactored to ensure correct cleanup after use. (Bug #28915716)

- The exception type and error messages returned by MySQL Shell functions for parameter errors have been standardized across the different functions. (Bug #28838958)

- MySQL Shell stopped unexpectedly if the `shell.setCurrentSchema()` method was called to set the default schema before an active session had been established. MySQL Shell now validates that there is an active session when the operation takes place. (Bug #28814112)

- The MySQL Shell JSON import utility no longer requires an empty dictionary to be supplied if there are no import options. (Bug #28768585)

- In SQL mode, MySQL Shell does not add statements to the history if they include the strings `IDENTIFIED` or `PASSWORD`, or other strings that you configure using the `--histignore` command option or `shell.options["history.sql.ignorePattern"]`. However, this previously meant that filtered-out statements were not available to be corrected immediately after entry, and had to be re-typed in case of any errors. MySQL Shell now always makes the last executed statement available to be recalled by pressing the Up arrow, regardless of the filters set in the history ignore list. If filtering applies to the last executed statement, it is removed from the history as soon as another statement is entered, or if you exit MySQL Shell immediately after executing the statement. (Bug #28749037)

- The result printing logic in MySQL Shell has been refactored to use back-end rather than high-level result data, delivering performance improvements for all types of result data and more accurate representation for JSON data. (Bug #28710831)

- A memory leak was fixed that occurred when the new MySQL Shell command-line syntax was used. (Bug #28705373)

- The check for partitioned tables in shared tablespaces in the upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) did not return correct results for the 8.0.11 and 8.0.12 target versions. The check now uses alternative Information Schema tables that are populated with the required information in these versions. (Bug #28701423)

- The default value for `group_replication_exit_state_action` is `ABORT_SERVER`, but AdminAPI now overrides this and sets the default on instances to `READ_ONLY`. This ensures that instances which leave the group unexpectedly continue running and can be rejoined to the cluster. (Bug #28701263)

- The MySQL Shell command `\option` ignored additional arguments separated by spaces that were specified for an option after the initial value. (Bug #28658632)

- MySQL Shell permitted newline characters (line feed and carriage return) in passwords to be passed to a Secret Store Helper using the `shell.storeCredential` method, resulting in an error in the Secret Store Helper. MySQL Shell now returns an exception if newline characters are used in supplied passwords for the `shell.storeCredential` method, and does not pass them to the Secret Store Helper. (Bug #28597766)

- On the Windows platform, UTF-8 encoded strings were printed to the console using the `cout` object, which transfers a byte at a time. This resulted in multi-byte Unicode characters, such as a single quotation mark, being displayed and handled incorrectly. MySQL Shell now uses alternative functions for printing, and verifies that multi-byte UTF-8 characters are emitted as a complete unit. (Bug #28596692)

- When executing an SQL script in MySQL Shell, an inaccurate line number was reported for the location of syntax errors in the script. The number referenced the current SQL block rather than the line number in the script. The error message now uses the global line number. (Bug #28545982)

- The SQL statement splitting logic in MySQL Shell has been refactored to fix a number of issues and to match behaviors of the MySQL command-line tool `mysql`:

  - The backslash character (`\`) is no longer accepted in the delimiter string.

  - The use of the word “delimiter” in contexts other than as a command is now handled correctly.

  - In scripts, comments are not discarded, and groups of comments and statements are now split in the same way as `mysql` would split them.
  - Large scripts can now be successfully split into incremental chunks even when some tokens span across more than one chunk.
  - Scripts can now be parsed in the `ANSI_QUOTES` SQL mode.
  - Multi-line strings and comments that contain quotes are now parsed correctly.
  - Inline commands are handled in the same way as by `mysql`, as follows:
    - A `\` character appearing at the beginning of a statement is interpreted as the start of a multi-letter MySQL Shell command.
    - A `\` character appearing within a statement is interpreted as the start of a single-letter command. The command is executed immediately, then stripped out of the input statement.
    - A `\` character appearing after the end of a statement is interpreted as the start of a single-letter command.

  (Bug #27959016, Bug #25689071)

- When a cluster was created on a server that did not have the X Plugin enabled, a silent assumption was being made about the X Protocol port value. Now the value of an X Protocol port is only stored for instances on which X Plugin is enabled. (Bug #27677227)
- The handling of Windows named pipe connections by MySQL Shell has been improved and systematized. Now, if you specify the host name as a period (.) on Windows, MySQL Shell connects using a named pipe.
  - If you are connecting using a URI type string, specify `user@.`
  - If you are connecting using a data dictionary, specify `{"host": "."}`
  - If you are connecting using individual parameters, specify `--host=.` or `-h .`

  By default, the pipe name `MySQL` is used. You can specify an alternative named pipe using the `--socket` option or as part of the URI type string. If a URI type string is used, the named pipe must be prepended with the characters `\\.\` as well as being either encoded using percent encoding or surrounded with parentheses, as shown in the following examples:

```
(\\.\named:pipe)
\\.\named%3Apipe
```

  (Bug #27381738)

- The `dba.checkInstanceConfiguration()` operation was not checking if the Performance Schema was enabled on the target instance. This could result in a situation where you could create a cluster but could not run several management operations on it, for example the *`Cluster`*`.status()` operation. Now, `dba.checkInstanceConfiguration()` checks that the Performance Schema is enabled on instances. (Bug #25867733)
- When JSON format output was enabled for MySQL Shell, the properties of the Shell API Options class (`shell.options`) and AdminAPI Cluster class (`dba.getCluster`) were not printed, only the class name. (Bug #25027181)

- When *`Cluster`*`.checkInstanceState()` was executed on an instance which was already a member of the current cluster, the output indicated that the instance was fully recoverable. This was misleading and was caused by a missing validation to ensure the instance does not belong to a cluster. (Bug #24942875)
- The `dba.checkInstanceConfiguration()` operation did not recognize privileges when they were associated to a user through a role (available in MySQL server 8.0 and higher). In such a case, a missing privileges error was being incorrectly issued despite the user possessing all the required privileges. Now users with their privileges assigned by roles are recognized by AdminAPI operations correctly. (Bug #91394, Bug #28236922)

Changes in MySQL Shell 8.0.13 (2018-10-22, General Availability)

Functionality Added or Changed

- The upgrade checker utility provided by MySQL Shell, which is the `checkForServerUpgrade()` function of the `util` global object, has several enhancements:
  - You can now use the upgrade checker utility to check servers at earlier MySQL 8.0.x releases, as well as MySQL 5.7 servers, for compatibility errors and issues for upgrading.
  - You can now specify a target MySQL Server version to which you plan to upgrade. In MySQL Shell 8.0.13, you can specify release 8.0.11 (the MySQL Server 8.0 GA release), 8.0.12, or 8.0.13. The upgrade checker utility carries out the checks that are relevant for the specified target release. If you specify the short form version number 8.0, or omit the `targetVersion` option, the utility checks for upgrade to the MySQL Server release number that matches the current MySQL Shell release number, currently 8.0.13.
  - A check has been added for the removed syntax `GROUP BY ASC/DESC`, returning an error message if this syntax is found in a trigger, event, view, stored procedure, or function.
  - A check has been added for columns defined as `ENUM` or `SET` that contain elements longer than 255 characters, returning an error message if any such columns are found.
  - The upgrade checker utility no longer returns a value, making its output easier to parse and process when automation is used.

  You can access the upgrade checker utility from within MySQL Shell or start it from the command line. For instructions and further information, see MySQL Shell Utilities.
- The behavior of *`Cluster`*`.dissolve()` has been updated to make it more consistent with other AdminAPI commands. Now you do not have to pass in the `force` option to start the command, and there is an interactive prompt available. When all instances belonging to the cluster are online, if MySQL Shell is running in interactive mode then you are prompted to confirm the operation of dissolving the cluster. If MySQL Shell is running in noninteractive mode, when all instances are reachable, or online, then the command removes the instances from the cluster. In the case that instances are not reachable an error is thrown. Pass in the `force` option to remove instances which are not reachable.

  References: See also: Bug #27833605, Bug #27837231.
- MySQL Shell onscreen output can now be displayed using a pager such as `less` or `more`. You can configure the pager you want to use with the `shell.options[`*`pager`*`]` option, the `\pager` command,

or the `--pager` command option. This improves how you work with longer text output in MySQL Shell, specifically the online help and the results of SQL operations. See Using a Pager.

- The integration of MySQL Shell into command-line environments has been improved. Use the `mysqlsh [options] -- shell_object object_method [method_arguments]` syntax to pass operations directly to MySQL Shell global objects, bypassing the REPL interface. For example:

```
mysqlsh -- util check-for-server-upgrade user@example --output-format=JSON
```

  which executes the equivalent `util.checkForServerUpgrade(user@example, {"outputFormat": "JSON"})` with MySQL Shell and returns the output in JSON format. This makes it easy to integrate MySQL Shell into your automation scripts. To get help for this interface, use the MySQL Shell command `\help cmdline`. See API Command Line Interface.

- A new optional parameter `exitStateAction` can be used with the `dba.createCluster()` and `cluster.addInstance()` commands, which enables you to configure the `group_replication_exit_state_action` variable of an InnoDB cluster member. The `group_replication_exit_state_action` variable enables you to specify what action is taken if a member involuntarily leaves the group. When `group_replication_exit_state_action` is set to `ABORT_SERVER` (the default value), the instance shuts itself down, and when `group_replication_exit_state_action` is set to `READ_ONLY` the instance switches itself to super read only mode instead and goes into the Group Replication `ERROR` state.

- The new optional `memberWeight` option can be used with the `dba.createCluster()` and `Cluster.addInstance()` functions to enable you to set the `group_replication_member_weight` system variable of an InnoDB cluster server instances in a single-primary cluster. The default value is 50, in other words the system variable default. Set the `memberWeight` option to an integer between 0 and 100 to configure a member's weight in the failover election process. The value determines the chance of the instance being elected as the primary in the event of a failover. See Single-Primary Mode for more information.

- The `connect-timeout` connection path parameter (see Connecting Using a URI or Key-Value Pairs) has been added to the X DevAPI, which enables you to specify the number of seconds clients such as MySQL Shell wait until the client stops trying to connect to an unresponsive MySQL server. The value of `connect-timeout` must be a non-negative integer that defines a time frame in milliseconds. The timeout default value is 10000 milliseconds, or 10 seconds. For example:

```
// Decrease the timeout to 2 seconds.
mysqlx.getSession('user@example.com?connect-timeout=2000');

// Increase the timeout to 20 seconds
mysqlx.getSession('user@example.com?connect-timeout=20000');
```

  To disable the timeout set the value to 0, meaning that the client waits until the underlying socket times out, which is platform dependent.

- MySQL Shell has a new JSON import utility that enables you to import JSON documents from a file or standard input to a MySQL Server collection or relational table. The utility parses and validates the supplied JSON documents automatically and inserts them into the target database, removing the need to use multiple INSERT statements or write scripts to achieve this task. The utility can be started in a MySQL Shell session with the `util.importJson()` method in JavaScript or the `util.import_json()` method in Python. From the command line, you can use the `-- util importJson` syntax or the `--import` command to invoke the utility.

Bugs Fixed

- The upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) did not report removed functions if they were used in views or events. (Bug #28642534)
- MySQL Shell incorrectly labeled warning messages as error messages in JSON output. (Bug #28546510)
- When MySQL Shell server connection passwords are persisted using a Secret Store, if a classic MySQL protocol connection was made without specifying a port or socket, the saved password could not be retrieved for a subsequent connection. The password storage and retrieval process now ensures that the server URL used to store the password matches subsequent queries with user-provided connection options, even if defaults were used for the original connection. (Bug #28544628)
- The `dba.deploySandboxInstance()` function in version 8.0.12 deploys the sandbox and includes `log_syslog=OFF` in the instance's configuration file. This variable was deprecated in MySQL 8.0.12 and was removed in MySQL 8.0.13. Now, the variable has been updated to include the `loose_` prefix which makes the server ignore it for a MySQL 8.0.13 sandbox, while maintaining compatibility with earlier version sandboxes. (Bug #28543536)
- A number of improvements have been made to the MySQL Shell prompt, including handling of overlength text, statement splitting, and support for multiple-line prompts. New sample prompt theme files are provided for double-line prompts that use one line for information display and a new line for the input prompt itself, so that additional information can be shown without detracting from the space available for text entry. (Bug #28515394, Bug #92048)
- The `--table` command line option did not produce the appropriate output format in noninteractive mode. (Bug #28511408)
- Extra spaces before or after the parameter used with the `\help` command are now trimmed. Previously, the presence of extra spaces made MySQL Shell unable to find the relevant help topic. (Bug #28508724, Bug #92030)
- Some native MySQL Shell objects were not properly wrapped into JavaScript objects, causing memory leaks. The memory-handling mechanism has been corrected. (Bug #28473341)
- If an empty string was provided as the argument for a MySQL Shell command-line option that expects a non-null argument and has no default defined, MySQL Shell did not return an appropriate error. The error handling for command-line arguments has now been improved so that a suitable error is issued in this situation and execution of the command terminates. (Bug #28378553)
- If a session was explicitly closed by the user without closing MySQL Shell, the prompt continued to display the details of the closed session. (Bug #28314383)
- User credentials stored by MySQL Shell could not be automatically retrieved for hosts identified by IPv6 addresses. (Bug #28261301)
- MySQL Shell now displays the build type (commercial or Community Edition) as part of the product version information displayed at startup and when the `--help` argument is used. (Bug #28242573)
- If a MySQL Shell session was disconnected without closing MySQL Shell (for example, using the `session.close()` method), a subsequent query in SQL mode did not return a "Not connected" error. MySQL Shell now checks not only that the global session object exists, but also that it has a valid connection to the MySQL server instance. (Bug #28240437)
- On the Windows platform, the background color was not reset in the MySQL Shell window when the terminal was cleared using **Ctrl+L**. (Bug #28235701, Bug #91102)

- The commercial MySQL Shell package could not be installed on Debian or Ubuntu if the equivalent Community Edition package had already been installed. (Bug #28223781)
- Occasionally, when adding an instance to an existing cluster the instance got stuck in the distributed recovery phase resulting in an immutable reported status of `RECOVERING`. This issue was related to the automatically generated password for the internal replication users created by InnoDB cluster. (Bug #28219398, Bug #91348)
- MySQL Shell was using some deprecated functions and properties internally, which caused warning messages to appear in the MySQL Shell log file, although the functions were not executed by users. The deprecated functions and properties have now been removed from the internal code. (Bug #28216558)
- If an invalid value was specified for the MySQL Shell option `credentialStore.helper`, the resulting error message at MySQL Shell startup was displayed incorrectly. (Bug #28216485)
- The upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) now correctly handles account names with spaces and other blank characters, and skips the permissions check when the server was started with the `--skip-grants` option. (Bug #28212899, Bug #91326)
- In the default interactive mode, whenever using the function `dba.rebootClusterFromCompleteOutage()` without any parameter, the function failed with an error specifying the cluster name does not exist. Now the default cluster is assumed when the function is issued without a parameter. (Bug #28207565)
- The handling of metadata server changes related to the *`Cluster`*`.addInstance()` has been improved, resulting in the following changes:
  - the correct session is now used for metadata and group operations
  - the *`Cluster`*`.addInstance()` operation aborts and recommends the use of *`Cluster`*`.rescan()` if the instance is in the group but not the InnoDB cluster metadata
  - the unnecessary parameter `super_user_password` has been removed

  (Bug #28200661)
- The Windows scripts generated by `dba.deploySandboxInstance()` incorrectly displayed user output messages in quotes. Additionally, the scripts have been improved and they no longer display executed commands. (Bug #28199954)
- When deleting history entries using the `\history` command in MySQL Shell, you can now specify a number of history entries to be deleted from the tail of the history, using the format `\history delete` *`-number`*. The handling of history entry numbers for deleted entries has been improved so that when the tail of the history is deleted, those history entry numbers are reused for new entries, and there is no gap. If a `\history delete` command empties the history, the numbering of history entries now resets as it does when the `\history clear` command is used. Also, the error message issued if you specify an invalid range of history entries to be deleted (using the format `\history delete` *`firstnumber-lastnumber`*) has been improved. (Bug #28199513)
- The `dba.createCluster()` AdminAPI operation always created replication users, even when the `adoptFromGR` option was used. However, when adopting an already existing Group Replication group no additional users need to be created. (Bug #28054500)
- On the Windows platform, using the **Ctrl+C** key combination in MySQL Shell caused MySQL Shell to close, even if a command was in progress. (Bug #27894642)

- Error messages issued by the AdminAPI relating to an invalid number of arguments for a function did not include the relevant object and method prior to the message text. These messages have now been standardized. (Bug #27832594)

- The MySQL Shell code for identifying whether a given IP address is a loopback address did not account for additional IP addresses (besides the 127.0.0.0/8 address block) that had been added by the user to the loopback interface. All IP addresses assigned to the loopback interface are now checked. (Bug #27703779)

- When running MySQL Shell in batch mode, tab separated format is the default for output. In some situations, table format was used for output instead. This issue has now been fixed. A command line option `--tabbed` has also been added to switch to the tab separated format for output when MySQL Shell is in interactive mode, where the default is table format. (Bug #27546082, Bug #89514)

- Zone IDs in IPv6 addresses are now supported for MySQL Shell connections. A zone identifier is suffixed to the IPv6 address with a percent character (%) as a separator. For example:

  ```
  2001:0db8:3c4d:0015::1a2f:1a2b%14
  ```

  If an IPv6 address with a zone ID is provided as a URI type string, URL encoding (percent-encoding) must be used for the percent character. For example:

  ```
  mysqlsh --uri=user@[2001:0db8:3c4d:0015::1a2f:1a2b%2514]:33060
      shell.connect("user@[2001:0db8:3c4d:0015::1a2f:1a2b%2514]:33060")
  ```

  If an IPv6 address with a zone ID is provided using individual parameters or a data dictionary, URL encoding does not need to be used for the percent character, and the IPv6 address can be supplied as seen. For example:

  ```
  mysqlsh --user=user --host=2001:0db8:3c4d:0015::1a2f:1a2b%14 --port=33060
  ```

  See Connecting Using a URI or Key-Value Pairs. (Bug #27539702)

- When MySQL Shell log entries were output to `stderr` by prepending @ (at sign) to the value of the `--log-level` option, and the JSON output format was selected for MySQL Shell, some log entries were not being output in JSON format. The logger now checks and uses the current value of the MySQL Shell `outputFormat` configuration option as the output format when writing log entries to `stderr`. (Bug #27480887)

- In SQL mode, MySQL Shell erroneously entered multi-line mode if an unknown command was executed, or if multiple consecutive SQL statement delimiters were used. Now, an appropriate error is returned in these situations. (Bug #27411526)

- If you do not specify a protocol with the `\connect` command or when starting MySQL Shell, MySQL Shell automatically attempts to use X Protocol for the session's connection, and falls back to MySQL protocol if X Protocol is unavailable. The connection type option `-ma`, which specified that behavior explicitly, is now deprecated.

  The use of a single dash with the connection type options `-mx` and `-mc`, for an X Protocol and MySQL protocol connection respectively, is also deprecated. These options must now be specified with a double dash (that is, `--mx` and `--mc`) with the `\connect` command or when starting MySQL Shell. They are now defined as aliases of the long form `--mysql` (`--mc`) and `--mysqlx` (`--mx`) connection type options. (Bug #27363459)

- When using `dba.createCluster()` or *`Cluster`*`.addInstance()`, the AdminAPI was setting the values of `auto_increment_offset` and `auto_increment_increment` incorrectly. Now the variables are set according to the following logic:

  - for a cluster running in single-primary mode:
    - `auto_increment_offset=2`
    - `auto_increment_increment=1`
  - for a cluster running in multi-primary mode:
    - `auto_increment_offset=1` and `server_id` % 7
    - `auto_increment_increment=7`

  (Bug #27084767)

- On the Windows platform, when a long command was accessed from the MySQL Shell command history and edited at the right edge of the console window, a cursor positioning error caused the command to move up one line and overwrite the output of previous commands. The issue has now been fixed. (Bug #27068352)
- The mysqlsh command-line options `--dbpassword[=password]` and `--dbuser=user_name` are now deprecated. Use the options `--password` (`-p`) and `--user` (`-u`) instead. (Bug #26049681)
- AdminAPI was using the incorrect terms for Group Replication. Now clusters are described as single-primary and multi-primary, the `multiMaster` option has been deprecated, and the `multiPrimary` option has been added. (Bug #25926603)
- The result printer in MySQL Shell was refactored to improve handling of binary data based on the output format, handling of multi-byte characters, and alignment of table formatting when multi-line characters are present. (Bug #24912154, Bug #24967872)
- The result of calling `dba.get_cluster().status()` when quorum was lost could not be converted to a JSON object, because the string representation of the resultant object contained escape sequences. This issue was not limited to the `Cluster.status()` method, but affected all arrays and dictionaries returned by the Shell API in Python mode. The internal representation of arrays and dictionaries has been fixed. (Bug #91304, Bug #28200499)

Changes in MySQL Shell 8.0.12 (2018-07-27, General Availability)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- **Important Change:** An RPM package for installing ARM 64-bit (aarch64) binaries of MySQL Shell on Oracle Linux 7 is now available in the MySQL Yum Repository and for direct download.

  Known Limitation for this ARM release: You must enable the Oracle Linux 7 Software Collections Repository (ol7_software_collections) to install this package, and must also adjust the libstdc++7 path. See Yum's Platform Specific Notes for additional details.

- MySQL Shell now enables you to store user credentials in an operating system specific secret store. You can then enter a MySQL user's password during connection and store it for future connections. Currently the following secret stores are supported:
  - MySQL login-path

  - MacOS keychain
  - Windows API

  (Bug #23304789, Bug #81484)

- The way you access the online Shell help has been standardized. Use the `\help pattern` command to search the help. The scope of the command has been increased to support retrieving help for the following categories:

  - Class and function help for the Admin API, X DevAPI and Shell API. Previously, to retrieve help for API objects, you had to create an instance of the object and use the `object.help()` method.
  - SQL syntax help, provided that a global session object exists.

  Wildcards can now be used to search for help. A number of additional bugs relating to incomplete help information have also been fixed. (Bug #23255291, Bug #81277, Bug #24963435, Bug #25732663, Bug #85481, Bug #25739522, Bug #85511, Bug #25739664, Bug #85514, Bug #26393155, Bug #86950, Bug #24943074, Bug #26429399, Bug #87037, Bug #27870491, Bug #90455, Bug #27870503, Bug #90456, Bug #27875150, Bug #90474, Bug #24948933, Bug #83527)

- The `util.checkForServerUpgrade()` operation has an additional `outputFormat` parameter that you can specify when running the utility. The utility can now generate output in two formats:

  - TEXT format, which is the default. This option provides output suitable for humans, as previously returned by the utility.
  - JSON format. This option provides output suitable for machines, which can be parsed and processed for various further use cases.

- The `cluster.removeInstance()` command has been improved, with the following changes:

  - A new interactive option has been added to enable or disable interactive mode for the command. The output displayed in interactive mode has been improved, displaying more useful information.In interactive mode, you are prompted to continue with the removal of the instance (or not) in case it is not reachable.
  - The operation now ensures that the instance is removed from the metadata of all the cluster members and itself. This only applies to `ONLINE` members.
  - A new global option `dba.gtidWaitTimeout` is available to define the timeout to wait for transactions (GTIDs) to be applied when required by AdminAPI commands. If the timeout value defined by `dba.gtidWaitTimeout` is reached when waiting for the cluster transactions to be applied for `cluster.removeInstance()` and `force: false` (or not defined) then an error is issued and the operation aborted. When `force: true` then the operation continues and does not generate an error.

  References: See also: Bug #27817894.

- When using the `ipWhitelist` to define which servers could access the cluster, the internal user accounts were not matching the whitelist. Now AdminAPI applies the same filtering logic from `ipWhitelist` for the internal administrative accounts.

  References: See also: Bug #26140094, Bug #28165891.

- In order to be compliant with the X DevAPI specification, the following changes have been made:

- `Collection.modify(condition).arrayDelete()` and `Collection.modify(condition).merge()` have been deprecated.
- `Collection.find().limit(x).skip(y)` has been renamed to `Collection.find().limit(x).offset(y)`.
- `Collection.find().limit(x).skip(y)` has been deprecated.
- `Collection.find().limit(x).offset(y)` has been implemented.
- `BaseResult.getAffectedItemsCount()` has been implemented.
- `BaseResult.getWarningCount()` has been deprecated.
- `BaseResult.getWarningsCount()` has been implemented.
- `Result.getAffectedItemCount()` has been deprecated.
- `SqlResult.getAffectedRowCount()` has been deprecated.
- `SqlResult.nextDataSet()` has been renamed to `SqlResult.nextResult()`.
- `SqlResult.nextDataSet()` has been deprecated.
- `SqlResult.nextResult()` has been implemented.

Bugs Fixed

- The sample prompt theme files for MySQL Shell were deployed to an incorrect location on the Windows platform, in the root install folder. The files are now correctly deployed in the `\share\mysqlsh\prompt` sub-folder. (Bug #28188761)
- The upgrade checker utility provided by MySQL Shell (the `util.checkForServerUpgrade()` operation) has been enhanced with a summary count of the errors, warnings, and information level issues found by the tool, and with links to documentation with further information where this is available. (Bug #28171814)
- The `cluster.forceQuorumUsingPartitionOf()` operation sets the `group_replication_force_members` variable on the target instance to force a new group membership and restore the quorum, but it did not reset the value of the variable at the end of the process. Consequently, if Group Replication later needed to be restarted on the target instance it failed because the `group_replication_force_members` variable was still set. Now, the `group_replication_force_members` variable is reset to an empty string at the end of the `cluster.forceQuorumUsingPartitionOf()` operation. (Bug #28064621)
- When upgrading from version 1.0.11 to version 8.0.11 of MySQL Shell on Linux, the upgrade failed if the original package was the community edition and the new package was the commercial edition, or vice versa. Upgrading from one edition to the other edition is now enabled. (Bug #28037407)
- The `util.checkForServerUpgrade()` operation can now use either an X Protocol connection or a classic MySQL protocol connection. (Bug #28027707)
- The `checkForServerUpgrade()` operation to verify upgrade prerequisites included an unnecessary check relating to `ZEROFILL` and display length attributes in columns. The check has now been removed. (Bug #27927641, Bug #90634)

- Some messages displayed by MySQL Shell were showing a MySQL server version that does not exist. (Bug #27924694)
- For sessions using the classic MySQL protocol, if the `session_track_gtids` system variable is set on the server to capture and return GTIDs to the client, MySQL Shell now displays the GTIDs for successfully committed transactions. The returned GTID values are also now recorded in tracing information. (Bug #27871148)
- When the `defaultMode` MySQL Shell configuration option had been set with the `--persist` option, batch code execution from a file was always attempted using the specified default language, even if the file extension indicated a different supported language. Now when a file is loaded for batch processing using the `--file` or `-f` option, files with the extensions `.js`, `.py`, and `.sql` are processed in the appropriate language mode, regardless of the set default language. (Bug #27861407)
- The methods provided in the `shell.options` configuration interface to set and save persistent option values used underscores in JavaScript as well as in Python mode. The methods have now been changed to `shell.options.setPersist()` and `shell.options.unsetPersist()` in JavaScript to follow the appropriate naming convention. (Bug #27861141)
- When executing a SQL script using MySQL Shell, delimiters (such as the default semi-colon character) present in multi-line comments caused execution to fail. Delimiters are now ignored inside multi-line comments. (Bug #27841719)
- MySQL Shell returned an error when querying timestamp values that were zero, because a zero value for the month or day in a date was not accepted. Zero timestamp values can now be used without producing an error. (Bug #27833822, Bug #90355)
- The `shell.getSession()` function returns a reference to the `session` global object representing the already established connection between MySQL Shell and a MySQL server, known as a global session. MySQL Shell now gracefully handles the situation where the function is called when no global session has yet been established. (Bug #27809310)
- It was possible to use AdminAPI operations on server instances running an incompatible version of MySQL. (Bug #27765769)
- The setting of the `bind_address` variable is no longer a requirement. (Bug #27765484)
- When creating a cluster or adding an instance, if the `localAddress` option is not specified, the port used for `group_replication_local_address` is automatically assigned with the value: `port` * 10 + 1. However, if the resulting port determined by the previous rule was already in use then a random port was generated and used. Now MySQL Shell checks that the `group_replication_local_address` port is available, and fails if it is not. (Bug #27758041)
- The MySQL Shell application icon on Microsoft Windows was not being displayed for the MySQL Shell 8.0 GA release, due to an incorrect association introduced for the icon during code refactoring. The icon is now displayed correctly. (Bug #27746532)
- The `dbPassword` option is no longer valid in the options dictionary of all AdminAPI commands. (Bug #27745106)
- The `\status` (`\s`) command in MySQL Shell now displays full information about the version and build of the connected MySQL server. (Bug #27740420)
- The check for reserved keywords carried out by the `util.checkForServerUpgrade()` operation was updated to match the list of reserved keywords for the MySQL 8.0 GA release. (Bug #27724201)
- When handling escape sequences, MySQL Shell now identifies and skips over SQL comments and string literals within quotation marks. (Bug #27665229)

- Python's mapping type has been added to MySQL Shell, so that dictionary syntax can be used to interact with data in Python mode. (Bug #27614110)
- When a file was redirected to standard input for execution in MySQL Shell, on Unix, the first part of the file was taken as being the password. The password prompt now looks for user input first before resorting to standard input. (Bug #27572380)
- It was possible to use the `dba.forceQuorumUsingPartition()` operation on a cluster which had not lost quorum. (Bug #27508698)
- The help message for `dba.rebootClusterFromCompleteOutage()` operation was incorrectly suggesting to use `dba.forceQuorumUsingPartition()`. (Bug #27508627)
- If **Ctrl+C** was entered or an unexpected error occurred at a password prompt in MySQL Shell, the terminal state was not restored correctly afterwards. (Bug #27379834)
- The `dba.rebootClusterFromCompleteOutage()` operation was creating a new user on the target instances, which could lead to the existence of an increasing number of users. The fix ensures that these users are not created by the `dba.rebootClusterFromCompleteOutage()` operation. (Bug #27344040)
- Now when you issue `dba.getCluster()` and retrieve a cluster without quorum a warning is issued in addition to the log message. (Bug #27148943)
- The `memberSslMode` option could be used with *`cluster`*`.addInstance()` and *`cluster`*`.rejoinInstance()` operations but if you specified a different value than the one used at cluster creation an error was thrown. Now set the SSL mode at the cluster level only, in other words when issuing `dba.createCluster()`. The `memberSslMode` option has been removed from *`cluster`*`.addInstance()` and *`cluster`*`.rejoinInstance()`. (Bug #27062122)
- When you issued `dba.configureLocalInsance()` on an instance, it configured the `disabled_storage_engines` variable with the `MyISAM`, `BLACKHOLE`, `FEDERATED`, `CSV`, and `ARCHIVE` storage engines to ensure that the storage engine was set to `InnoDB`, as required by Group Replication. The change to this option was not being reported correctly by AdminAPI, and hence the required restart after changing the `disabled_storage_engines` variable was not clear. This change was deemed a recommendation, rather than a requirement, hence `dba.configureLocalInsance()` no longer configures `disabled_storage_engines`. (Bug #26754410)
- Creating a cluster using an account which was missing the global grant option failed with an ambiguous error message, even though `dba.checkInstanceConfiguration()` did not return any errors. Now when you create a cluster, the account being used to administer the cluster is checked to ensure that it has the global grant option. (Bug #25966235)
- MySQL Shell is able to automatically reconnect global session when running in the interactive mode, but AdminAPI methods lacked this feature. This resulted in you having to reconnect manually. Now, the AdminAPI methods which utilize the global session object have been improved in order to detect an interrupted session and trigger the reconnection mechanism. The Cluster object uses its own internal session instance, which does not support automatic reconnection. If connection to the cluster is lost, you need to manually recreate the Cluster object. (Bug #24702489)
- In the event of a whole cluster stopping unexpectedly, upon reboot the `memberSslMode` was not preserved. In a cluster where SSL had been disabled, upon issuing `dba.rebootClusterFromCompleteOutage()` this could prevent instances from rejoining the cluster. (Bug #90793, Bug #27986413)

Changes in MySQL Shell 8.0.11 (2018-04-19, General Availability)

Functionality Added or Changed

- **Important Change; Microsoft Windows:** Before installing MySQL Shell, make sure you have the Visual C++ Redistributable for Visual Studio 2015 (available at the Microsoft Download Center) installed on your Windows system. This now applies for both Community and Commercial versions of MySQL Shell.

- **Important Change:** The way which document IDs are generated has been changed. Now document IDs are generated by the server, rather than by the client. As a result the `getLastDocumentID()`, `getDocumentId` and `getDocumentIDs()` methods have been removed. To get a list of the document IDs automatically generated by an 8.0.11 and later server, use `Result.getGeneratedIDs()`. The type of column generated for the document ID in a collection has changed from `VARCHAR(32)` to `VARBINARY(32)`. The generated document ID can be overridden manually by including an ID but you must respect the server generated IDs to avoid conflicts. If you are using InnoDB cluster, use the `mysqlx_document_id_unique_prefix` variable to ensure your documents can be moved between replicasets.

  Now, if you are adding documents to a collection on a server running a version of MySQL earlier than 8.0.11, you must manually include a document ID as these versions do not add the ID automatically.

- The indexing of collections has been improved to make large collections of documents more efficient to navigate. Now, you can create an index based on one or more fields found in the documents in the collection, using a JSON document which maps fields from the collection's documents to MySQL types. The majority of MySQL types are supported, in addition to spatial indexes and GeoJSON data.

- Support for the `NOWAIT` and `SKIP LOCKED` `InnoDB` locking modes has been added to the lock operations. Now you can use these locking modes with the `lockShared()` and `lockExclusive()` methods, for example:

  - `Table.select().lockShared([LockContention])`
  - `Table.select().lockExclusive([LockContention])`
  - `Collection.find().lockExclusive([LockContention])`
  - `Collection.find().lockExclusive([LockContention])`

  where `LockContention` can be one of:

  - `DEFAULT` - if the function encounters a row lock it waits until there is no lock
  - `NOWAIT` - if the function encounters a row lock it aborts and generates an `ER_LOCK_NOWAIT` error
  - `SKIP_LOCKED` - if the function encounters a row lock it skips the row and continues

  For more information see Locking Read Concurrency with NOWAIT and SKIP LOCKED.

- MySQL Shell can now connect to a MySQL server with an account that uses the `caching_sha2_password` authentication plugin. Assuming the server is configured for encrypted connections, you can use such accounts over both X Protocol and the classic MySQL protocol. See Using Encrypted Connections.

Important

If you are not using encrypted connections, to connect over X Protocol with an account that uses the `caching_sha2_password` authentication plugin, the user's password must be stored in the cache. Currently there is no way to store the password over X Protocol if not using encrypted connections.

When using the classic MySQL protocol for connections with such an account and unencrypted connections, you can configure MySQL for password exchange using an RSA key pair. MySQL Shell supports such connections and the following command options have been added:

- Use the `--server-public-key-path` option to specify the RSA public key file.
- Use the `--get-server-public-key` option to request the public key from the server.

For more information, see Caching SHA-2 Pluggable Authentication.

- InnoDB cluster instances running MySQL 8.0.11 and higher can now be configured for InnoDB cluster usage remotely, without the need to log in to the instance and run `dba.configureLocalInstance()` locally. Use the new `dba.configureInstance()` operation, which enables remote automatic configuration of compatible instances for InnoDB cluster usage and if necessary also supports remote restart of compatible instances after configuration.

  Similarly, compatible instances support having any changes to their settings automatically persisted by AdminAPI after any cluster topology changes. Now, when working with remote instances if you create a cluster, add an instance to a cluster, or remove an instance from a cluster supporting this functionality, there is no need to log in to the instance and run `dba.configureLocalInstance()` to persist the changes to the instance's option file. This makes working with production clusters, which are built on networked instances, much easier. This also ensures that instances automatically rejoin a cluster in the case of a restart.

  As part of this feature the following improvements were made:

  - Improved display of messages.
  - Improved detection of loopback hostnames on Debian and Ubuntu, and local instances based on the hostname.
  - Improved display of required configuration changes.
  - Replaced automatic configuration of instances after issuing `dba.createCluster` and `dba.addInstance` with the configuration check `dba.checkInstance()`, aborting the command if the check fails.
  - When specifying an account in interactive mode, if the user's hostname is local, for example localhost, you are given the option to recreate the account configured for remote usage by using the hostname `%`, or to create a new InnoDB cluster administrator account.

  References: See also: Bug #27608299, Bug #27112727, Bug #27629803.

- MySQL Shell now has a configuration file which stores configuration changes across sessions. Use the new `\option` MySQL Shell command for querying and changing configuration options. Alternatively use the following methods with the `shell.options` object:

```
shell.options.set_persist(optionName, value)
```

```
shell.options.unset_persist(optionName, value)
```

In addition the new `defaultMode` option has been added, which enables you to configure the programming language which MySQL Shell starts up in. You can override the default mode using the command options.

- The `util.checkForServerUpgrade([uri])` operation has been extended to check for the following incompatible features:
  - obsolete `sql_mode`
  - partitioned tables in shared tablespaces
  - removed functions

Bugs Fixed

- When the `\quit` command was used to exit MySQL Shell, this event could be noted in the error log as an aborted connection. The issue has now been fixed. (Bug #27821045, Bug #90281)
- When the MySQL Shell command-line option `--json=raw` was used, the output was actually provided in pretty-printed format, and an empty string was displayed in place of error messages. These issues have now been corrected. (Bug #27733996, Bug #26737357)
- When MySQL Shell commands were executed from a script, interactive prompting for passwords and confirmations was not available. Now, interactive prompting is enabled by default when the commands are used in a script, as it is when they are used on the command line. The `--no-wizard` command line option disables interactive prompting for MySQL Shell commands used in both ways. (Bug #27702250)
- After creating a cluster administrative user using `dba.createCluster()`, MySQL Shell attempted to reconnect using the new user but still using localhost, which failed causing the whole configuration to fail. Now MySQL Shell no longer switches accounts between administrative account creation and instance configuration. (Bug #27673816)
- The changes introduced by Bug#27545850 mean that now it is not possible to modify Group Replication related options while the plugin is starting or stopping. The AdminAPI operations have been modified to ensure that they respect this change, and attempting to issue a statement which would change a Group Replication operation while the plugin is starting results in an error. There is no check for when the plugin is stopping and so you should ensure you do not attempt to configure an instance if there is a chance the plugin is stopping. (Bug #27545850)
- The `util.checkForServerUpgrade()` operation now prompts the user for a password interactively if the required password is not provided along with the command. If the `--no-wizard` option has been used to disable the connection wizard, missing credentials instead result in an error and the function is not executed. (Bug #27514395)
- The `util.checkForServerUpgrade()` operation was requiring the wrong privileges for the user passed to the function. The user now requires `ALL` privileges, and does not require the `GRANT OPTION` privilege. (Bug #27506702)
- The `util.checkForServerUpgrade()` operation was rejecting host names that included the % (percent) symbol or were specified as a numeric IP address. (Bug #27506079, Bug #27513260)
- When rejoining an instance to a cluster the displayed output has been improved. (Bug #27437389)
- The `dba.createCluster()` function fails if used on an instance with `innodb_page_size=4k`, because the `instance_name` column of the instances table in the InnoDB Cluster metadata

uses a `VARCHAR` of size 256, which is too large for the `innodb_page_size=4k`. However, the size of this column cannot be changed because it is used to hold hostnames of the cluster members which can have a length up to 255 bytes. This limitation is now validated when using `dba.createCluster()`, `dba.checkInstanceConfiguration()`, `dba.configureInstance()`, and `dba.configureLocalInstance()`. (Bug #27329079)

- The AdminAPI commands which manage sandboxes failed with an error when using a sandbox directory with non-ASCII characters in the path. This occurred when the user name had a non-ASCII character because the default sandbox directory is a subdirectory of the `$HOME` or `%userprofile%` path, which is usually based on the current user name. The fix adds Unicode support to the internal provision tool used by AdminAPI. (Bug #27181177)

- MySQL Shell now sets the value of `group_replication_local_address` based on `port` by calculating the result of `((port * 10) + 1)`. The `port` variable defaults to 3306, in which case MySQL Shell sets `group_replication_local_address` to `((3306 * 10) +1)`, resulting in port 33061 instead of the previous default of 13306. (Bug #27146799)

- The online help for all AdminAPI commands which require a connection displayed detailed information on connection data, which cluttered up the details of each command. Now, the help displays an overview of connection details and information about where else to get more help. (Bug #27146290)

- When creating a cluster on a Debian type platform with the default system configuration, the` `cluster.addInstance()` command failed with an error if the instance hostname was used in the connection parameter. This happened because on these platforms the hostname is resolved to the IP address 127.0.1.1 by default, but this IP address is not supported by the Group Replication Group Communication Service (GCS). The fix adds a validation to the `dba.createCluster()` and `cluster.addInstance()` functions to verify if the connection hostname resolves to 127.0.1.1 and issue an error in that case. (Bug #27095984)

- The `Cluster.forceQuorumUsingPartitionOf()` operation was not working with a non-root user, even if that user had all the required privileges. The fix ensures that the specified user is used to connect to all instances, instead of the root user. (Bug #27089930)

- Issuing `dba.createCluster()` in interactive mode as a user which did not have all the required privileges resulted in an unexpected halt. The error message generated when issuing `dba.configureLocalInstance()` and `dba.checkInstanceConfiguration()` if the account being used did not have the required privileges has also been improved. (Bug #27076753, Bug #27324699)

- When using `adoptFromGR` to convert a Group Replication group into an InnoDB cluster, the output recommended adding instances. This message has now been improved to show that the instances were already added. (Bug #27061615)

- By default, MySQL Shell connections are assumed to require a password, which is requested at the login prompt. A new MySQL Shell command-line option `--no-password` is provided to explicitly specify that no password is used, and to disable the password prompt. The `--no-password` option can be used if socket peer-credential authentication is in use (for Unix socket connections), or for any authentication method where the user has a password-less account (though note that this situation is insecure and not recommended).

  The methods previously provided by MySQL Shell for specifying that no password is used for the connection are still valid, and can be used instead of the `--no-password` option. These methods are as follows:

  - If you are connecting using a URI type string, place a `:` after the user name in the URI type string but do not specify a password after it.

  - If you are connecting using individual parameters, specify the `--password=` option with an empty value.

  (Bug #26986360)

- In interactive mode the `cluster.removeInstance()` AdminAPI function was not accepting the second parameter with the options dictionary. This prevented use of the `force` option, which failed with an error. (Bug #26986141)

  References: See also: Bug #27572618.

- The `cluster.addInstance()` function was not checking the validity of the `server_uuid` being added to the cluster. Now, the `server_uuid` of an instance joining the cluster is checked to be unique, and if it is not an error is generated and the instance is prevented from joining the cluster. (Bug #26962715)

- When setting up a cluster a replication user is created to enable distributed recovery. If MySQL Shell logging was enabled, the `CREATE USER` statements were not being correctly added to the log. (Bug #26938488)

- When an error occurred while returning a result set, the fetch operation was interrupted but the associated error was not reported. The error is now reported correctly. (Bug #26906527)

- The `dba.checkInstanceConfiguration()` and `dba.configureLocalInstance()` operations were not correctly reporting any errors related to incorrect configuration of the instance's `server_id`. Now, when an error is reported it is displayed in the list of variables not meeting the InnoDB cluster requirements.

  In addition, the X Plugin load has been removed from the `my.cnf` created by AdminAPI for sandbox instances running MySQL version 8.0.11 and later because X Plugin is installed by default for those versions. (Bug #26836230)

- When using `dba.configureLocalInstance()` with the `clusterAdmin` option to create a user which can administer the cluster, the created account had too many privileges. (Bug #26737608)

- Support for microseconds has been added to the `mysqlx.dateValue()` function and the `Date` object. (Bug #26429497)

- Argument validation and error messages were improved for the `mysqlx.dateValue()` function. (Bug #26429426, Bug #26429377)

- The `db` global object is not available for connections in SQL mode. This reference has been removed from the message returned by the `\connect` command in that situation. (Bug #26428665)

- When you create a cluster and add instances to it internal users are created, which are required by Group Replication for distributed recovery when instances join the cluster. These automatically generated replication users were not being removed from instances after removing them from the cluster or after dissolving the cluster. (Bug #26395608)

- The handling of SQL wildcard characters was corrected for the X DevAPI `Schema.getTable()`, `Schema.getCollection()` and `Session.getSchema()` functions. (Bug #26392984)

- A shortcut to uninstall MySQL Shell is no longer provided in Microsoft Windows menus in the group of installed MySQL programs. Uninstallation of MySQL Shell should be handled through MySQL Installer. (Bug #26317449)

- If an instance contained InnoDB cluster metadata but was stand alone, in other words it did not belong to a cluster, it was not possible to use `dba.dropMetadataSchema()`. (Bug #26315635)

- The runtime timer for MySQL Shell, which reports the time taken for each query execution, has been refactored to provide increased precision of 4 digits for fractional seconds. (Bug #25976636, Bug #86135)
- In the event of a disconnection, MySQL Shell did not reconnect to the schema that was in use before the connection was lost. The last active schema set by the user is now restored during the automatic reconnection process, and during a manually triggered reconnection using the `\reconnect` command. (Bug #25974003, Bug #86115)
- If you issued `dba.configureLocalInstance()` against an instance which was already valid for InnoDB cluster usage and specified the `myCnf` option, but the `my.cnf` file was not readable, the operation would print that there are issues that need to be fixed. Now when you set the `myCnf` option, using either `dba.configureInstance()` and `dba.configureLocalInstance()`, the operations only use it if necessary. In other words, if the target instance is not valid for InnoDB cluster usage and settings must be changed. (Bug #25702994)
- The `cluster.describe()` and `cluster.status()` AdminAPI methods return JSON objects containing the same information but some fields were identified by different tags. To make the tags consistent, the object returned from `cluster.describe()` has changed so that `instances` has been replaced with `topology`, and `host` with `address`. (Bug #25247515)
- MySQL Shell no longer attempts to reconnect automatically when the connection to the server is lost. A new MySQL Shell command `\reconnect` is provided, which makes MySQL Shell try several reconnection attempts for the current global session with the existing connection parameters. If those attempts are unsuccessful, you can make a fresh connection using the `\connect` command and specifying the connection parameters. (Bug #25105307)
- Creating a cluster from an existing Group Replication group by using the `adoptFromGR:true` option on an instance which already belonged to cluster was not being correctly detected. Now such a situation is detected and generates an error. (Bug #25061891, Bug #25664766)
- MySQL Shell is able to create a session to an IPv6 address, but it failed to create an InnoDB cluster using such connections, reporting an URI-related error. This was related to the encoding of the URI containing an IPv6 address, which has been fixed. The core issue, IPv6 support for AdminAPI is a known issue as Group replication requires an IPv4 network, see Group Replication Requirements. The implementation of InnoDB cluster related operations has been modified in order to check if operations are executed using an IPv4 based session and IPv4 connection data. An exception is generated if these conditions are not met. (Bug #25042407)
- A memory leak was fixed that occurred if MySQL Shell was started and a connection was made to the MySQL server, then the user exited MySQL Shell without executing any commands. (Bug #24794589)
- Attempting to add two different instances with the same label to an InnoDB cluster correctly resulted in an error, however the instance with the duplicated label was being left with a partially initialized configuration. (Bug #24761416)

Changes in MySQL Shell 8.0.5 - 8.0.10 (Skipped version numbers)

There are no release notes for these skipped version numbers.

Changes in MySQL Shell 8.0.4 (2018-01-25, Release Candidate)

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- When you create clusters or add instances you can now override the default group name, local addresses, and group seeds. This makes it easier to customize your clusters. The following options were added to the `dba.createCluster()` and `cluster.addInstance()` commands:

  - use `groupName` with `dba.createCluster()` to set the name of the cluster
  - use `localAddress` to set the address which an instance provides to communicate with other instances
  - use `groupSeeds` to set the instances used as seeds when instances join the cluster

  For more information, see Customizing InnoDB clusters. (Bug #26485254, Bug #26838005)

- Connections to an InnoDB cluster have been simplified. Now, when you issue `dba.getCluster()` and the active Shell session is not a connection to the primary, the cluster is queried for the primary information and a connection to the primary is automatically opened. This ensures that configuration changes can be written to the metadata. As part of this improvement you can now configure the type of MySQL Shell connection to an InnoDB cluster with the following options which have been added:

  - `--redirect-primary`
  - `--redirect-secondary`
  - `--cluster`

  Additionally the following changes were made:

  - The `dba.resetSession()` method has been removed.
  - A new `disconnect()` method has been added to the `cluster` object, which closes all internal sessions opened by the cluster. InnoDB cluster operations on a disconnected cluster object result in an error.
  - The output of the `Cluster.status()` method now includes the `groupInformationSourceMember` field, which shows the URI of the internal connection used by the cluster object to obtain information about the cluster.

  References: See also: Bug #25091586.

- MySQL Shell now supports autocompletion of text preceeding the cursor by pressing the Tab key. Autocompletion is available for:

  - Built-in MySQL Shell commands, for example typing **`\con`** followed by the Tab key completes to `\connect`.
  - SQL, JavaScript and Python language keywords depending on the current MySQL Shell mode.
  - Table names, column names, and active schema names in SQL mode, based on the current default schema.

  Autocompletion can be configured using the following command options:

  - `--name-cache`
  - `--no-name-cache`

- A new `patch()` operation has been added to the `modify()` function which enables you to modify a document by merging in a set of changes. For more information see `JSON_MERGE_PATCH()`.
- You can now use Unix sockets for X Protocol connections. Socket file paths in URI type strings should be either percent encoded, such as `root@/home%2Fuser%2Fmysql-sandboxes%2F3310%2Fsandboxdata%2Fmysqlx.sock`, or surrounded by parenthesis such as `root@(/home/user/mysql-sandboxes/3310/sandboxdata/mysqlx.sock)`. The `--socket` option cannot be combined with the `--port` option. See Connecting Using a URI or Key-Value Pairs.
- MySQL Shell performs automatic `_id` generation on collection add operations when no `_id` is specified on the documents being added. The autogenerated _id is created using the UUID() function. Now, the order of the tokens used has changed from `aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee`, where `eeeeeeeeeeee` is the MAC Address, to `eeeeeeeeeeee-dddd-cccc-bbbb-aaaaaaaa`.
- Sessions created by X DevAPI using either `mysql.getClassicSession(connection_data)` or `mysqlx.getSession(connection_data)` now use `ssl-mode=REQUIRED` as the default if no `ssl-mode` is provided, and neither `ssl-ca` nor `ssl-capath` is provided. If no `ssl-mode` is provided and any of `ssl-ca` or `ssl`-capath is provided, all Sessions created by MySQL Shell now default to `ssl-mode=VERIFY_CA`.
- In addition to the existing CRUD commands which work on documents in a collection by matching a filter, the following operations have been added to enable you to work with single documents:
  - `Collection.replaceOne(string id, Document doc)` updates (or replaces) the document identified by `id` with the provided one, if it exists.
  - `Collection.addOrReplaceOne(string id, Document doc)` add the given document. If the `id` or any other field that has a unique index on it already exists in the collection, the operations updates the matching document instead.
  - `Document Collection.getOne(string id)` returns the document with the given `id`. This is a shortcut for `Collection.find("_id = :id").bind("id", id).execute().fetchOne()`.
  - `Result Collection.removeOne(string id)` removes the document with the given `id`. This is a shortcut for `Collection.remove("_id = :id").bind("id", id).execute().`

  Using these operations you can reference documents by ID, making operations on single documents simpler by following a load, modify and save pattern such as:

```
doc = collection.getOne(id);
doc["address"] = "123 Long Street";
collection.replaceOne(id, doc);
```

- You can now use savepoints with MySQL Shell sessions. The following methods have been added to the Session object:
  - Use `setSavepoint()` to generate a savepoint. The server executes the `SAVEPOINT` SQL statement and returns the generated savepoint name.
  - Use `setSavepoint(name)` to specify the name used by the `SAVEPOINT` SQL statement.
  - Use `releaseSavepoint(name)` to execute the `RELEASE` SQL statement.
  - Use `rollbackTo(name)` to execute the `ROLLBACK TO name` SQL statement.

  Any names passed to these functions are checked to make sure that the name is not null or an empty string. Names such as `''`, `""`, ``` `` ```, and so on are not allowed even though they are allowed

by the server. For more information, see SAVEPOINT, ROLLBACK TO SAVEPOINT, and RELEASE SAVEPOINT Syntax.

- X DevAPI now supports row-locking for CRUD operations on tables and collections. Use the `lockShared()` method for write locks and the `lockExclusive()` method for read locks, which have been added to `find()` and `select()`. Either method can be called any number of times, including none, in any combination. But the locking type to be used is always the last one called. Once you call a lock method you can only then execute the operation, calling `execute()` or `bind()`. For more information, see InnoDB Locking.

- The X DevAPI drop operations have been improved. Now, the `drop()` methods are at the same level as the `create()` methods and they all return nothing. There is now no need to use `execute()` after the drop method. If a drop method is called on an object which no longer exists in the database there is now no error.

  This modifies Session objects so that:

  - `dropSchema()` returns Nothing
  - `dropTable(String schema, String name)` is removed
  - `dropCollection(String schema, String name)` is removed
  - `dropView(String schema, String name)` is removed

  This modifies Schema objects so that:

  - Nothing `dropCollection(String name)` is added

  This modifies Collection objects so that:

  - `dropIndex(String name)` is a direct operation, meaning `.execute()` is no longer needed
  - `dropIndex(String name)` returns Nothing

- The `mysql` module, used with classic sessions for SQL connections to instances, has been streamlined. This has resulted in the following changes:

  - The new `getSession()` function has been added to the `mysql` module to create a classic session. This function works in the same way as the existing `mysql.getClassicSession()` function.
  - The `runSql()` method of `ClassicSession` has been extended to take a list of arguments to replace with placeholders in the query. For example, now you can issue:

    ```
    session.runSql("select * from tbl where name = ? and age > ?", ["Joe", "20"])
    ```

    Additionally a `query()` function has been added to `ClassicSession` as an alias to `runSql()`, making it consistent with `Session.query()`.
  - The following functions have been removed from `ClassicSession`:
    - `createSchema()`
    - `getSchema()`
    - `getDefaultSchema()`
    - `getCurrentSchema()`

    - `setCurrentSchema()`
    - `ClassicSchema()`
  - The `ClassicTable` object has been removed.
  - The following `Table` and View DDL functions have been removed from `Schema` objects:
    - `dropTable()`
    - `dropView()`
  - The behavior of the global `db` variable has been modified. When a global connection is created and the connection data includes a default schema, the global `db` variable is set to the corresponding `Schema` object. Now, this is not done for connections through the MySQL protocol, such as those created by `ClassicSession`.
- Use `util.checkForServerUpgrade([uri])` to check if a server can be upgraded from the version it is currently running to the next major series release. For example, you can verify that a server instance running MySQL 5.7 satisfies the upgrade prerequisites for MySQL 8.0, see Preparing Your Installation for Upgrade. To verify a server instance at a URI type string such as `user@example.com:3306` issue:

  ```
  util.checkForServerUpgrade(['user@example.com:3306'])
  ```

  MySQL Shell provides a report on anything in the table space which would cause a problem when upgrading the instance to MySQL 8.0.
- MySQL Shell builds against MySQL server version 8.0.4 and OpenSSL.
- MySQL 8.0.4 uses the `caching_sha2_password` authentication plugin and encrypted connections by default. This means any new accounts created in MySQL use these features. The new authentication method is completely transparent if the connection is made with encrypted connections enabled, which is the default, and the preferred mode of connection. If encrypted connections are explicitly disabled, the following limitations apply:
  - X Protocol sessions require encrypted connections.
  - If the caching_sha2_password plugin reports an error such as `Authentication requires a secure connection`, it is not possible to connect to the account with MySQL Shell, until either an encrypted connection is made or the `mysql` client is used to clear the error.

  See Using Encrypted Connections and Caching SHA-2 Pluggable Authentication.

Bugs Fixed

- Attempting a server connection without specifying the port or socket, when using an expired or temporary password, failed with an error requiring a password reset. The default connection data is now set at an appropriate point in connection processing, so the connection succeeds. (Bug #27266648)
- MySQL Shell required the option name `ssl-ciphers` instead of the standard MySQL option name `--ssl-cipher` to specify the list of permitted ciphers for connection encryption. The standard option name `--ssl-cipher` is now required in MySQL Shell. (Bug #27185275)
- The `--show-warnings` option was not working in MySQL Shell. (Bug #27036716)

- The command line options `--classic`, `--node`, `--sqln`, and `--ssl` were stated as deprecated, but actually had been removed. MySQL Shell now handles the options again, but prints a warning message when they are used. The deprecated options are processed as their replacement options, as follows:

  - `--classic` is processed as `--mysql`
  - `--node` is processed as `--mysqlx`
  - `--sqln` is processed as `--sqlx`
  - `--ssl` is processed as `--ssl-mode`

  (Bug #27012385)

- The output of `\status` when using a Unix socket for connections was showing a relative path. Now the absolute path of the socket is shown. (Bug #26983193)

- Account validation did not work correctly unless the session account existed. Now, validation is done using the account that was authenticated by the server. (Bug #26979375)

- The AdminAPI in MySQL Shell for working with InnoDB cluster only supports TCP connections to server instances. The AdminAPI now checks that a TCP connection is in use before starting an operation that requires database access, instead of attempting the operation with another connection type and not succeeding. (Bug #26970629)

- If you issued `STOP GROUP REPLICATION` on an instance that belonged to a cluster, attempting to rejoin the instance to the cluster failed because the wrong Group Replication seeds were being used. Now, `Cluster.rejoinInstance()` correctly sets `group_replication_group_seeds` based on the `group_replication_local_address` of all currently active instances in the cluster. (Bug #26861636)

- Sometimes the `dba.addInstance()` command failed with an error indicating that the server was in `RECOVERING` state despite being `ONLINE`. The fix ensures the correct instance state is returned. (Bug #26834542)

- If the user running MySQL Shell did not have write permissions to the option file configured by AdminAPI, no error was displayed. (Bug #26830224)

- With the addition of WL#10470, the default value of `server_id` has changed to 1. As the server ID has to be unique for each instance, this caused issues with AdminAPI. Now, server instances with `server_id=1` are correctly identified as incorrectly configured for InnoDB cluster use. (Bug #26818744)

- AdminAPI now supports Python 2.6 in addition to Python 2.7, removing the need to manually install on Oracle Linux 6. (Bug #26809748)

- After removing an instance from a cluster using `Cluster.removeInstance()`, the instance silently rejoined the Group Replication group after it restarted. This happened because `group_replication_start_on_boot` was set to `ON` by default. Now, for instances running MySQL version 8.0.4 and later, the fix sets `group_replication_start_on_boot` to `OFF` in the option file. For instances running a MySQL version earlier than 8.0.4, a warning is issued to tell you to manually edit `group_replication_start_on_boot` in the instance's option file to avoid the issue. (Bug #26796118)

- Using AdminAPI commands on Windows that required SSL resulted in an error due to the Python version being used. (Bug #26636911)

- Creating an InnoDB cluster from an existing Group Replication deployment, by using the `adoptFromGR` option with the `dba.createCluster()` command, would fail with an error stating that the instance was

already part of a replication group. The issue was only present in the MySQL Shell default wizard mode. The fix ensures that the interactive layer of the `dba.createCluster()` command allows the use of the `adoptFromGR` option. (Bug #26485316)

- The warnings generated when creating and adding sandbox instances have been improved. (Bug #26393614)
- When working with instances that had `require_secure_transport=ON`, AdminAPI commands that required a connection to the instance failed. (Bug #26248116)
- MySQL Shell now automatically pre-loads the built-in `mysql` and `mysqlx` API modules when it is invoked in batch mode, as well as when it is used in interactive mode. (Bug #26174373)
- The user configuration path for MySQL Shell defaults to `%AppData%\MySQL\mysqlsh\` on Windows, and `~/.mysqlsh/` on Unix. This directory is used for the MySQL Shell history file (`history`), log file (`mysqlsh.log`), and theme file (`prompt.json`). It is also the final location that MySQL Shell searches for startup scripts (`mysqlshrc.js` or `mysqlshrc.py`).

  You can now override the user configuration path by specifying an alternative path using the `MYSQLSH_USER_CONFIG_HOME` environment variable. A directory specified by that environment variable is used by MySQL Shell for the user configuration data in place of `%AppData%\MySQL\mysqlsh\` on Windows, and `~/.mysqlsh/` on Unix. (Bug #26025157)
- The `Cluster.dissolve()` command was trying to stop Group Replication on all of the instances registered in the metadata which lead to connection errors if any of those instances were not contactable, in other words with the state `(MISSING)`. The fix ensures that only instances which can be contacted, in other words with the state `ONLINE`, are stopped. (Bug #26001653)
- When adding instances to an InnoDB Cluster using the appropriate AdminAPI operations, checks are performed to verify the compatibility of any existing tables. If incompatible tables (for example using `MyISAM`) are detected then an error is issued. However the error message was referring to an option not available for the AdminAPI operations: `--allow-non-compatible-tables`. (Bug #25966731)
- For file processing, MySQL Shell now expands a leading tilde in a file path to the appropriate home directory. MySQL Shell identifies the home directory using a relevant environment variable, or looks it up for the logged-in user. (Bug #25676405)
- MySQL Shell now provides a `program_name` connection attribute to the server at connect time, with the value `mysqlsh`. Connection attributes are displayed in the Performance Schema connection attribute tables. (Bug #24735491, Bug #82771)
- Running `help()` in MySQL Shell in Python mode caused an AttributeError. (Bug #24554329, Bug #82767)
- The `cluster.rejoinInstance()` command attempted to rejoin an instance even if was already part of the cluster. Now, only instances in the `MISSING` state are accepted by `cluster.rejoinInstance()`. Attempting to rejoin an instance in any other state fails with an error. (Bug #87873, Bug #26870329)
- On Unix, if Python 3 was installed AdminAPI commands failed. (Bug #87731, Bug #26785584)
- When using the `dba.checkInstanceConfiguration()` and `dba.configurelocalinstance()` commands, the account being used was not being checked if it had enough privileges to actually execute the command. The fix ensures that account has the required privileges before proceeding. This also required a change of the privileges given to `clusterAdmin` users. (Bug #87300, Bug #26609909)
- Arrays and Objects now accept the `IN` operator. For example:

```
collection.find("'Fred' IN username")
```

Changes in MySQL Shell 8.0.3 (2017-09-29, Development Milestone)

MySQL Shell now synchronizes the first digit of its version number with the (highest) MySQL server version it supports. This change makes it easy and intuitive to decide which client version to use for which server version. MySQL Shell now uses the same version number as MySQL Server.

MySQL Shell 8.0.3 is the first release to use the new numbering. It is the successor to MySQL Shell 8.0.0.

- Functionality Added or Changed
- Bugs Fixed

Functionality Added or Changed

- With the addition of and WL#10611 and WL#10960 , it was not possible to add or rejoin instances that belonged to a cluster (or a replication group) because `super_read_only=ON` was being set by Group Replication when stopping. To ensure that AdminAPI supports instances running MySQL 8.0.2 and later, the following functions have been modified:

  - `dba.configureLocalInstance()`
  - `dba.createCluster()`
  - `dba.rebootClusterFromCompleteOutage()`
  - `dba.dropMetadataSchema()`

  Now, if any of these functions is issued against an instance which has `super_read_only=ON`, in interactive mode you are given the choice to set `super_read_only=OFF`. To force the function to set `super_read_only=OFF` in a script, pass the `clearReadOnly` option set to `true`. For example `dba.configureLocalInstance({clearReadOnly: true}).` For more information see Super Read-only and Instances. (Bug #26422638)

- MySQL Shell now handles user interrupts, such as SIGINT, correctly. For example on Linux pressing Control-C when MySQL Shell is not executing anything exits the application. In SQL mode, interruption sends a `KILL QUERY` statement to the active MySQL Shell session from a new temporary session, resulting in the server interrupting the query and returning an error (or in an early return with no error in some cases, like the `sleep()` function). In JavaScript or Python scripting modes, how interruption behaves depends on the specific function being executed. If what is being executed is language code (such as a while loop and other normal script code), an exception is generated in the active language, which causes the code to stop executing. The exception may be caught by the script, but if not, the execution control returns to MySQL Shell. (Bug #24757361)

- MySQL Shell now includes a history function that stores the code which you issue. The history can be saved, searched, and filtered. A new mechanism to customize the MySQL Shell prompt has been added. Information such as the current mode (SQL, JavaScript, or Python), session information (host, uri, port and so on), the current active schema and others can be included in the prompt through variables. The customization information is self-contained in JSON theme files, which can be shared between users. MySQL Shell now supports unicode if the terminal used to run MySQL Shell supports it. Similarly if the terminal supports color, MySQL Shell can be configured to use colors in the theme.

- The connection options passed to MySQL Shell, such as `sslMode` and so on, have been changed to use dashes and no longer be case sensitive. The options are now:

    - `sslMode` is now `ssl-mode`
    - `sslCa` is now `ssl-ca`
    - `sslCaPath` is now `ssl-capath`
    - `sslCert` is now `ssl-cert`
    - `sslKey` is now `ssl-key`
    - `sslCrl` is now `ssl-crl`
    - `sslCrlPath` is now `ssl-crlpath`
    - `sslCiphers` is now `ssl-ciphers`
    - `sslTlsVersion` is now `tls-version`
    - `authMethod` is now `auth-method`
- The types of session available have been simplified. `XSession` and `NodeSession` have been consolidated into `Session`. This has caused the following changes:
    - The following command options have been deprecated: `--node`, `--sqln`, `--classic`
    - The following command options have been introduced to replace the deprecated ones: `-ma`, `--mysqlx` (`-mx`), `--mysql` (`-mc`), `--sqlx`
    - The `\connect` MySQL Shell command no longer supports the arguments `-c`, and `-n`. Now the `\connect` command supports the argument `--mysqlx`(`-mx`) for creating X Protocol connections, and `--mysql`(`-mc`) for creating MySQL protocol connections.
- The interpretation of the `document_path` field in operations such as `modify()` has been changed. Now, when the `document_path` is not set, operations apply to the whole document. All operations always preserve a document's `_id` field.

Bugs Fixed

- In MySQL Shell, the `Schema.getCollectionAsTable()` function and the `select()` method could not be used in the same Python statement. (Bug #26552804)
- When using the `clusterAdmin` option, the created account did not have all of the correct privileges. (Bug #26523629)
- MySQL Shell returned some elements of DATE and DATETIME values incorrectly, including month values and fractional seconds. (Bug #26428636)
- The month was incorrectly incremented on insertion of a timestamp in a table using MySQL Shell. (Bug #26423177)
- For columns with the `ZEROFILL` attribute, `NULL` was also returned padded with zeroes. (Bug #26406214)
- The output of the MySQL Shell `\status` command was enhanced with additional information. (Bug #26403909)
- The MySQL Shell help for the `\connect` command indicated that a connection name could be used instead of a URI string, which was incorrect. (Bug #26392676)

- When using the `multiMaster` option with `dba.createCluster()`, the warning displayed in interactive mode was not being logged. (Bug #26385634)
- When making cluster topology or membership changes, AdminAPI was not taking into consideration the value of `group_replication_group_name`, which could lead to incorrect, non-deterministic results in scenarios such as a split brain. Now, the following commands validate the InnoDB cluster Metadata and the corresponding instance's `group_replication_group_name` value:

  - `dba.getCluster()`
  - `Cluster.rejoinInstance()`
  - `Cluster.forceQuorumUsingPartitionOf()`

  If the values of `group_replication_group_name` do not match, the commands abort with an error.

  `dba.rebootClusterFromCompleteOutage()` was also updated to ensure that the `group_replication_group_name` variable has not been changed before rejoining the instance. (Bug #26159339)
- AdminAPI now always uses the active user value for the current `mysqlsh` session, whether the value was explicitly specified by the user or is the result of an implicit default used by `mysqlsh`. (Bug #26132527)
- The checks performed by the AdminAPI upon issuing `dba.rebootClusterFromCompleteOutage()` were more strict than those required by Group Replication. Now, the AdminAPI considers tables with a Primary Key Equivalent (such as a Non Null Unique Key) as compatible, matching the current requirement for Group Replication. (Bug #25974689)
- The MySQL Shell command `\use` did not attempt to reconnect if the connection to the global session was lost. (Bug #25974014, Bug #86118)
- The short form `-?` can now be used as an alias for the `--help` command-line option in MySQL Shell. (Bug #25813228)
- The MySQL Shell command history displayed the commands that were used to automatically import the `mysql` and `mysqlx` API modules when MySQL Shell started. (Bug #25739185)
- The randomly generated passwords used by internal users were not compatible with instances running the Password Validation plugin. (Bug #25714751)
- The MySQL Shell command history displayed the contents of scripts that were run using the `\source` MySQL Shell command. (Bug #25676495)
- It is no longer possible to use the `adoptFromGr` option with the `multiMaster` option. When adopting an existing group to an InnoDB cluster, the group is adopted based on whether it is running as multi-primary or single-primary. Therefore there is no use for the `multiMaster` option when adapting a group. (Bug #25664700)
- Issuing `configureLocalInstance()` when using a URI that contained a user without the correct privileges resulted in an incorrect new user being created. Now, if the user in `configureLocalInstance()` URI does not have enough privileges to grant all the necessary privileges for the new user chosen during the interactive wizard configuration the user is not created. (Bug #25614855)
- The `mysqlx.getNodeSession()` function in MySQL Shell now returns an error if an unrecognized connection option is provided. (Bug #25552033)

- Issuing `Cluster.rescan()` resulted in non-determinstic behavior which could produce incorrect JSON output, showing an instance that was already part of the cluster as belonging to the `newlyDiscoveredInstances[]` list and to the `unavailableInstances`[] list. This also resulted in MySQL Shell prompting to add or remove the instance from the cluster. (Bug #25534693)
- AdminAPI functions now accept the standard connection parameters as used by `shell.connect`. New validations have been added for when `require_secure_transport` is `ON`, now it is not possible to create a cluster with `memberSslMode:DISABLED` or to add an instance with `require_secure_transport=ON` to a cluster where `memberSslMode:DISABLED`. (Bug #25532298)
- The parsing of account names, for example when passing the `clusterAdmin` option to `dba.configureLocalInstance()` has been improved. (Bug #25528695)
- The file permissions of option files created by AdminAPI did not match those of options files created by MySQL install. (Bug #25526248)
- Issuing `configureLocalInstance()` twice could fail. (Bug #25519190)
- When passing the `rejoinInstances[]` option to `dba.rebootClusterFromCompleteOutage()`, if no `rejoinInstances[]` option was specified then members were being incorrectly handled during the rebuild. Now, instances that are eligible to be added to the `rejoinInstances[]` list but that are specified in the `removeInstances[]` list are skipped by the interactive wizard that tries to automatically build a `rejoinInstances[]` list if one was not provided. This fix also ensures that both interactive and noninteractive use of MySQL Shell correctly verify the `rejoinInstances[]` list does not contain a unreachable instance. (Bug #25516390)
- The error messages issued when the SSL mode used by the cluster and the one specified when issuing `addInstance()` command do not match have been improved. (Bug #25495056)
- The `--ssl` option has been deprecated, use the `--ssl-mode` option. Now, if you use the `--ssl` option a deprecation warning is generated and the `--ssl-mode` option is set to either `DISABLED` or `REQUIRED` based on the value used with the `--ssl` option. (Bug #25403945)
- When creating a sandbox instance using the `dba.deploySandboxInstance()` function in MySQL Shell, pressing **Ctrl+C** at the prompt for a MySQL root password for the instance did not cancel the deployment. (Bug #25316811)
- MySQL Shell did not exit gracefully when the user did not have a valid and accessible home directory. (Bug #25298480)
- MySQL Shell created a logger but did not deallocate it on exiting the shell. (Bug #25238576)
- Issuing `removeInstance`() on the last member of a cluster, and particularly the seed member, was resulting in a cluster that could not be dissolved. Now, issuing `removeInstance()` on the last member of a cluster results in an error, and you must use `dissolve()` on that instance to ensure the cluster is correctly dissolved. (Bug #25226130)
- The output of `cluster.status()` now includes the `ssl` parameter, which shows whether secure connections are required by the cluster or disabled. (Bug #25226117)
- MySQL Shell could hang when **Ctrl+C** was used to exit the shell. (Bug #25180850, Bug #84022)
- Attempting to create a multi-primary cluster in interactive mode failed unless you passed in the `{force: true}` option. Now when you confirm that you understand the impact of using multi-primary mode the command correctly creates a multi-primary cluster. (Bug #25034951)
- The `removeInstance()` was not working on stopped instances and it was not possible to remove an unavailable instance from the cluster. The fix adds a new option `force` to the removeInstance()

command to enable you to remove instances from the metadata that are permanently not available, avoiding obsolete data from being kept in the metadata of the cluster. In addition the error message provided when not using the force option has been improved and the online help for the `removeInstance()` was also updated accordingly. (Bug #24916064)

- Unsigned data could be incorrectly read from the database. (Bug #24912358)
- The parsing of Unix sockets provided as part of a URI has been improved. (Bug #24905066)
- The error messages generated by issuing `dba.deployLocalInstance()` against an unsuitable or incompatible instance have been improved. (Bug #24598272)
- The `dba.createCluster()`, `dba.getCluster()`, and `dba.rebootClusterFromCompleteOutage()` functions have been updated to validate the cluster name, using the following rules:

  - Name must start with a letter or the _ character
  - Name can only contain alphanumeric characters and the _ character
  - Cannot be longer than 40 characters
  - Cannot be empty

  The `Cluster.addInstance()` function has been updated to validate the label used on an instance in the cluster, using the following rules:

  - Label can only contain alphanumerics or the _ character
  - Cannot be longer than 256 characters
  - Cannot be empty

  (Bug #24565242)
- MySQL Shell now accepts Unicode characters as input. (Bug #23151666, Bug #81176)

Changes in MySQL Shell 8.0.2 (Skipped)

Version 8.0.2 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL Shell 8.0.1 (Skipped)

Version 8.0.1 has no release notes, or they have not been published because the product version has not been released.

Changes in MySQL Shell 8.0.0 (2017-07-14, Development Milestone)

Functionality Added or Changed

- Calling the `modify()` or `remove()` function without a parameter caused the function to be executed against the whole collection, which could cause unexpected results such as deleting all rows in a table.

To avoid this and make the behavior consistent with `update()` and `delete()`, a client-side exception is now thrown if the `modify()` or `remove()` function is called without a parameter. Now, to execute the `modify()` or `remove()` function against a collection call them with an expression that evaluates to `true`, for example `remove('true')` or `modify('true')`.

Bugs Fixed

- Executing AdminAPI commands on a server with a version of Python lower than 2.7 was failing without the correct error message. (Bug #25975317)
- When using MySQL Shell on Windows any files created or opened, for example those used during `dba.createSandboxInstance()`, could not be deleted. (Bug #25789094)
- The help for `dba.configureLocalInstance(instance[, options])` has been improved to describe the returned JSON object. (Bug #25703028)
- The options in the MySQL Shell options dictionary are now fully documented. (Bug #25701345)
- When using `dba.deploySandboxInstance()` and passing in `sandboxDir`, the specified path must not exceed 89 characters. (Bug #25485035)
- `shell.connect()` did not report an error if an invalid argument was used. An `ArgumentError` is now issued for any invalid argument.

  The following mutually exclusive pairs of options are now checked, and an error is issued if both are specified:

  - `--password` and `--dbPassword`
  - `--user` and `--dbUser`
  - `--port` and `--socket`

  (Bug #25268670)

  References: See also: Bug #24911173.
- `removeInstance()` resulted in unexpected behavior in some cases, for example when an empty password was passed as part of the URI to the instance. (Bug #25111911)
- A number of issues with the output of `shell.help("prompt")` have been corrected. (Bug #25026855, Bug #25242638, Bug #25676343, Bug #25176769)
- MySQL Shell now displays an invalid year as `0000`, matching the behavior of the MySQL prompt, rather than as `0`. (Bug #24912061)
- MySQL Shell did not display fractional seconds for values in DATETIME columns. (Bug #24911885)
- Creating Classic sessions that connect using Unix sockets now uses the correct defaults such as hostname. This resolves the previous limitation of using Unix sockets to connect to InnoDB cluster instances. See MySQL Shell Connections for information on how the defaults are applied to socket connections. (Bug #24848763, Bug #26036466)

  References: See also: Bug #24911068.
- Some issues with the MySQL Shell command line help output were fixed. (Bug #24841749, Bug #24841493, Bug #24910540)
- URIs were incorrectly parsed in MySQL Shell when passwords were hidden. (Bug #24793956)

- `mysqlsh` stopped responding if the `\source` command was given a directory (rather than file) argument. (Bug #23097932, Bug #81060)
- On an instance configured as a multithreaded slave, in other words `slave_parallel_workers` set to greater than 0, and with `slave_parallel_type=DATABASE`, `dba.checkInstanceConfiguration()` was not detecting that the instance was not correctly configured for InnoDB cluster usage.
- If `removeInstance()` failed due to a connection error, an error was reported but the instance was incorrectly removed from the InnoDB cluster metadata, and remained part of the replication group. The fix ensures the metadata is correctly updated according to the result of `removeInstance`().
- In a situation where a new primary instance was elected, adding a new instance to the cluster resulted in an error due to a failed connection to the previous primary instance.
- The functions that modify server variables, such as `dba.createCluster()` and `dba.validateInstance()` now provide more information in interactive mode output and log output about server variables which are changed when executed.
- Deploying instances to paths with directories that contained spaces was failing without error. Use double backslash to specify such paths, for example `D:\\Cluster\\foo bar`.
- The Cluster object obtained from functions such as `dba.createCluster()` or `dba.getCluster()` became unusable once the Shell session in which the object was created is was connected to a different server. The fix modifies the Cluster object so that:
  - The Cluster object holds an internal reference to the Session from which it was created or retrieved.
  - AdminAPI functions that modify the Cluster are made using the session referenced by the object.