Qube 7.5-0 Release Notes

##
#
Qube Release Notes
#
##

##
@RELEASE: 7.5-0

==== CL 22602 ====
@NEW: add one more file for initialization of the central preferences database, qubedb_prep_0054.sql, which creates the "prefs" DB user.

JIRA: QUBE-3795

==== CL 22597 ====
@CHANGE: implemented code to make changes to postgresql.conf and pg_hba.conf files on installation, needed to support the new central preferences feature.

JIRA: QUBE-3795

==== CL 22596 ====
@CHANGE: "pfx" account's default password changed to a longer one (for new installations, except on Linux)

JIRA: QUBE-3667

==== CL 22593 ====
@NEW: add SQL to initialize the central preferences database.

JIRA: QUBE-3795

==== CL 22592 ====
@FIX: init_supe_db.py: make all calls to the "psql" command using the DB owner

==== CL 22458 ====
@CHANGE: point PYTHONPATH to $QBDIR/lib/python3.8 before supervisor service is started, for its embedded python interpreter (Linux, macOS)

==== CL 22288 ====
@NEW: add "disable_central_prefs" flag to supervisor_flags

JIRA: QUBE-3778

==== CL 22230 ====
@FIX: a bunch more fixes to make python-based backends to work properly with python3 while maintaining python2 compatibility.

Now if a python-based jobtype's job.conf specifies "execute_binding = Python" or "execute_binding = Python3", python3 will be used. If "execute_binding = Python2", python2 is used.

JIRA: QUBE-3747

==== CL 22190 ====
@CHANGE: Switch supervisor's embedded Python interpreter to python3.8.

JIRA: QUBE-2762, QUBE-3749

==== CL 22179 ====
@CHANGE: made Linux installation (RPM and DEB for CentOS/RHEL and Ubuntu, repectively) require "python3"

JIRA: QUBE-3767

==== CL 22177 ====
@CHANGE: convert python-based jobtypes (appFinder, pyCmdline, pyCmdrange) qube/types/ from python2 to python3

JIRA: QUBE-3747

==== CL 22176 ====
@CHANGE: convert example python scripts in qube/examples/python from python2 to python3

JIRA: QUBE-3746

==== CL 22175 ====
@CHANGE: convert python scripts in qube/scripts from python2 to python3

JIRA: QUBE-3745

==== CL 22174 ====
@CHANGE: convert python scripts in qube/utils from python2 to python3

JIRA: QUBE-3745

==== CL 22081 ====
@FIX: "pfx" user to be created w/o a home directory now, in the install_supervisor script, which is used to do some initialization on DEB-based Linux platforms (i.e. Ubuntu).

Was previously set up to create a home dir, causing the DEB qube-supe
installation to exit prematurely when the root user doesn't have write
permissions to create "/home/pfx" (e.g. NFS-mounted /home).

Now the "useradd" command points to "/var/tmp" as the pfx user's homedir.

==== CL 22080 ====
@NEW: add perl 5.30 support (for Ubuntu 20.04 LTS)

JIRA: QUBE-3721

==== CL 22077 ====
@FIX: "pfx" user to be created w/o a home directory now, in the install_supervisor script, which is used to do some initialization on RPM-based Linux platforms.

Was previously set up to create a home dir, causing the RPM qube-supe
installation to exit prematurely when the root user doesn't have write
permissions to create "/home/pfx" (e.g. NFS-mounted /home).

Now the "useradd" command points to "/usr/tmp" as the pfx user's homedir.

==== CL 22057 ====
@NEW: Ubuntu 18.04 and 20.04 support

JIRA: QUBE-3720, QUBE-3721

==== CL 22039 ====
@NEW: Add Python 3.7 (standard) and 3.8 (homebrew) API support on macOS

==== CL 22035 ====
@NEW: Add Python 3.6, 3.7, and 3.8 API support for Windows.

JIRA: QUBE-2762

==== CL 22030 ====
@NEW: Add python 3.6, 3.7, and 3.8 Qube API support on Linux.

==== CL 21802 ====
@CHANGE: macOS to build Qube core with Qt 5.14.2

JIRA: QUBE-3688

==== CL 21801 ====
@CHANGE: remove python 2.6 support from all platforms

JIRA: QUBE-3691

==== CL 21800 ====
@CHANGE: Linux to build with Qt5.14.2

JIRA: QUBE-3688

==== CL 21769 ====
@NEW: add Python 3.8 compatibility to the main Qube Python API , including its supporting .py scripts.

JIRA: QUBE-2762

==== CL 21731 ====
@FIX: Fix crash when no options are given. Now usage message is printed when no args are present.
@CHANGE: Made the "checks" arguments instead of options.
@INTERNAL: refactored a bunch of stuff.

JIRA: QUBE-3206

==== CL 21644 ====
@FIX: not all child processes of job instances sometimes not dying properly when parent thread dies

ZD: 20225

==== CL 21641 ====
@FIX: not all child processes of job instances sometimes not dying properly when parent thread dies

ZD: 20225

==== CL 21556 ====
@FIX: fixed problem where a job can get stuck in "dying" state due to a timing-related issue.

This was causing, among other things, global resources to not be released properly.

ZD: 20307

==== CL 21432 ====
@CHANGE: install_supervisor script: install Data W/H DB by calling $QBDIR/datawh/install_datawarehouse_db.sh

==== CL 21385 ====
@TWEAK: Don't give up on the first error in enableRequiredPrivileges(), but try enabling all privileges. Also print number of errors.

==== CL 21360 ====
@FIX: Agressively preempted frames can get missed and left in "pending" while instances all finish

ZD: 20177

==== CL 21351 ====
@CHANGE: add SE_DEBUG_NAME to list of privileges to be enabled; also add more info to print to workerlog

* add SE_DEBUG_NAME to the list of privileges to be enabled
* print WARNING when OpenProcess() fails in cleanup(), and the reason
* add instance name to print in more output lines when available/applicable

==== CL 21242 ====
@FIX: On host reboot, supervisor needs to start after postgresql is started and ready

* added code to check the DB connection at supervisor's boot time, and retry after 10 seconds, up to 6 attempts (1 minute),
effectively delaying the supervisor boot until after the DB is ready.

JIRA: QUBE-3637

==== CL 21239 ====
@NEW: add a way to tell qbjobinfo() API routine to only query for and pull selective job data (aka "columns" or "fields").

Developers using the Qube C++ and/or Python API can now tell qbjobinfo() routine (qb.jobinfo() for Python) to only query
for and pull selective job data (aka "columns" or "fields"), for leaner, meaner, more economical queries.

* Add support for explicitly specifying needed fields in C++ API's qbjobinfo().
* Add support for explicitly specifying needed fields in Python API's qb.jobinfo(), a la 6.10's direct query API.
* Also add "-fields" option to qbjobs
* qbjobs now makes leaner queries by default (unless an option to display details is specified, like "-long" or "-notes")

[Examples]

C++:
```
QbString query_fields_str = "id,username,status";
QbStringList query_fields;
QbExpression::split(query_fields_str, query_fields);
QbQuery query;
for(int i = 0; i < query_fields.length(); i++) {
QbField *f = new QbField(*query_fields.get(i));
query.fields().push(f);
}
QbJobList jobs;
qbjobinfo(query, jobs)
```

Python:
```
jobs = qb.jobinfo(fields = ['id','username','status'])
```

JIRA: QUBE-3623
ZD: 19955

==== CL 21234 ====
@FIX: Add timeout for agenda-based jobs stuck in "running" status, in a "waiting" loop.

TL;DR:
Sometimes, agenda-based job instances can get stuck in the "running" state, in a "wating" loop. A timeout, currently hardcoded to 60 seconds,
has been added to force those jobs to break out of the loop.

Details:

Sometimes, agenda-based job instances can get stuck in a "wating" loop, with messages like the following repeating indefinitely in the job's stdout:

[Dec 20, 2019 18:23:46] HOSTNAME[47572]: requesting work for: 424805.0
[Dec 20, 2019 18:23:46] HOSTNAME[47572]: got work: -1: - waiting
[Dec 20, 2019 18:23:46] HOSTNAME[47572]: INFO: informing worker[127.0.0.1]
INFO: told to wait & retry from supe-- sleeping for [7] seconds

A job instance stuck in this state can tie up a worker's job slot(s) until it is manally intervened with (killed, migrated, etc), or until it hits its "subjob timeout" (assuming the job was setup with it).

This issue, newly introduced in 7.x, has been found to happen due to race conditions.

It is particularly likely to occur when the following conditions are met:
* jobs have the migrate_on_frame_retry job flag set AND they use retrywork/retrysubjob
* job instances fail quickly (i.e. job process/renderer crashes and exits quickly)
* there are idle workers

(There are other scenarios that this can also happen, such as when aggressive preemption is done rapidly, but there's normally not many idle workers when preemptions do happen, so it's less likely.)

In a nutshell:
* instance fails on a worker
* supe detect the failure, migrates and starts the instance on a new worker
* the new worker reports the instance now "running"
* the first worker finishes cleaning up and reports that the instance is now "pending"
* instance gets stuck in a "wating" loop on the new worker.

A timeout, currently hardcoded to 60 seconds, has been added to force those jobs to break out of the infinite loop.

ZD: 19977, 20094, 19967
JIRA: QUBE-3638

==== CL 21217 ====
@FIX: Add timeout for agenda-based jobs stuck in "running" status, in a "waiting" loop.

TL;DR:
Sometimes, agenda-based job instances can get stuck in the "running" state, in a "wating" loop. A timeout, currently hardcoded to 60 seconds, has been added to force those jobs to break out of the loop.

Details:
Sometimes, agenda-based job instances can get stuck in a "wating" loop, with messages like the following repeating indefinitely in the job's stdout:

[Dec 20, 2019 18:23:46] HOSTNAME[47572]: requesting work for: 424805.0
[Dec 20, 2019 18:23:46] HOSTNAME[47572]: got work: -1: - waiting
[Dec 20, 2019 18:23:46] HOSTNAME[47572]: INFO: informing worker[127.0.0.1]
INFO: told to wait & retry from supe-- sleeping for [7] seconds

A job instance stuck in this state can tie up a worker's job slot(s) until it is manally intervened with (killed, migrated, etc), or until it hits its "subjob timeout" (assuming the job was setup with it).

This issue, newly introduced in 7.x, has been found to happen due to race conditions.

It is particularly likely to occur when the following conditions are met:
* jobs have the migrate_on_frame_retry job flag set AND they use retrywork/retrysubjob
* job instances fail quickly (i.e. job process/renderer crashes and exits quickly)
* there are idle workers

(There are other scenarios that this can also happen, such as when aggressive preemption is done rapidly, but there's normally not many idle workers when preemptions do happen, so it's less likely.)

In a nutshell:
* instance fails on a worker
* supe detect the failure, migrates and starts the instance on a new worker
* the new worker reports the instance now "running"
* the first worker finishes cleaning up and reports that the instance is now "pending"
* instance gets stuck in a "wating" loop on the new worker.

A timeout, currently hardcoded to 60 seconds, has been added to force those jobs to break out of the infinite loop.

ZD: 19977, 20094, 19967
JIRA: QUBE-3638

==== CL 21060 ====
@FIX: bug where jobs passively preempted while working on the final agenda item don't complete properly but go "pending" with the agenda

items 100% done

JIRA: QUBE-3626
ZD: 19967

Also:
@TWEAK: made IP address to also print, in addition to the hostname, when qbwrk.conf for a host is loaded (QbSupervisor::loadWrkConfig()). Now prints something like:
loaded config for host: hostname (aaa.bbb.ccc.ddd)

==== CL 21059 ====
@INTERNAL CHANGE/FIX: Removed the special, undocumented "feature" where the "str" passed to "QbField::value(str)" may optionally be prefixed with a special character to specify an operator (or "sign") to be applied for the field.

The special char was one of [~,=,>,<,%,*], and was setting "sign" to a
string representing the ASCII value of the op character ("itoa" value, for
example '%' to "37").

Instead of just fixing that, we're ditching this special feature, as it is
unnecessary and confusing. The operator can always be specified explicitly
by calling QbField::sign(str). A scan of the qube code base didn't reveal
any use of this feature, but it's possible that peripheral code (namely
python apps, like WV or AV) may be using it (but I highly doubt it).

==== CL 20817 ====
@FIX: fix_mysql_column_orders.sql: add back AUTO_INCREMENT to columns ('id' or 'uq') of the following MySQL tables: globalcallback, globalevent, jobid, lostevent

The ALTER TABLE done to modify column orders on these tables were wiping
the AUTO_INCREMENT of these tables' specified columns. This was in turn
causing issues (job submissions failing, for example) when downgrading 7.0
to 6.10.

ZD: 19833

==== CL 20683 ====
@FIX: qubeproxy on Linux does not use the same standard default password as it does on macOS/Windows

JIRA: QUBE-613

==== CL 20681 ====
@NEW: Add ability to lock/unlock workers by MAC address

JIRA: QUBE-243

==== CL 20658 ====
@CHANGE: Use of FK (foreign keys) in Postgres for job removal.

* Optimized job removal.
* Added utils/pgsql/qubedb_0054.sql which will "ALTER TABLE" (via init_supe_db.py) the relevant job-related tables.

JIRA: QUBE-3319

==== CL 20650 ====
@FIX: Setting a *_flags param to an empty string should mean "no flags", not "default flags"

Specifically, the supervisor_job_flags would not behave as expected, and
would take on the default values when specified as:

supervisor_job_flags = ""

or

supervisor_job_flags =

JIRA: QUBE-620

==== CL 22713 ====
@CHANGE: Disabling "query" in supervisor_verbosity by default, to avoid overcrowding the supelog with the frequent (2 or more per second) "job/host query received" messages generated by the new supervisorProxy queries

##
#
The Qube! Supervisor Proxy Release Notes
#
##

The Qube! Supervisor Proxy:

Watches for changes to jobs and workers on the Qube! supervisor and pushes those changes out to the Qube! UI clients running on the network. This reduces load on the supervisor in large farms and the UIs get interative updates for the jobs and workers panels.

##
@RELEASE: 7.5-0
##

What's new:

The Qube! Supervisor Proxy is now available on all Qube! supported platforms.