Upgrading to MySQL 8.4 at Booking.com

Simon J Mudd

Senior Site Reliability Engineer

2nd February 2025, FOSDEM, Brussels

Booking.com

Usual Disclaimer

Opinions are my own and not necessarily of my employer

B.

Agenda

- About us
- First MySQL Usage at Booking.com
- A few words about running MySQL
- Upgrading to MySQL 8.4
  - Why?
  - How?
  - Issues Seen
  - Progress
- Conclusion

Booking .com

About Us

B.

Simon Mudd

- Senior Site Reliability Engineer
- Working at Booking.com with MySQL since 2007
- 17 years working with MySQL
- 30 years of experience in System Administration and Site Reliability Engineering.
- Previous work experience in wholesale financial markets in London, Madrid and Amsterdam
- Other database experience managing a real-time trading system managing a replicated Sybase cluster
- Been an Oracle ACE / MySQL Rockstar

ACE Pro

B.

MySQL Teams at Booking.com

- 20+ Engineers
- Located in
  - Amsterdam
  - Cambridge
  - Madrid
  - Manchester

A few team members in a previous conference

Booking .com

The company

- One of the largest travel e-commerce sites worldwide
- Founded in 1996 in Amsterdam
- It is part of Booking Holdings Inc, BKNG on Nasdaq
- 28 million accommodation listings in over 200 countries
- Our website, available in 43 languages, covers:
    - accommodation
    - rental cars
    - flights
    - attractions

but first...

MySQL is 30

Congratulations to Oracle on investing in and improving MySQL following the acquisition of Sun Microsystems in 2009!

---

without MySQL booking.com might not be here (in its current form)

B.

Booking First MySQL Usage

- When Booking.com was founded in 1996 it started using Postgres but as the company grew it had trouble scaling up
- In 2003-2004 work was made to use MySQL (while still running Postgres)
  - 32-bit Linux, 4GB of RAM
  - MyISAM tables
  - MySQL 4
- Since then we have continued to run MySQL to this day
  - the original cluster 'bp' is still running holding a lot of common central data
  - we used replication to allow for upgrades to be seamless avoiding downtime
- So we have been using MySQL for ~22 years.

B.

Our MySQL Usage

B.

How do we run *MySQL*?

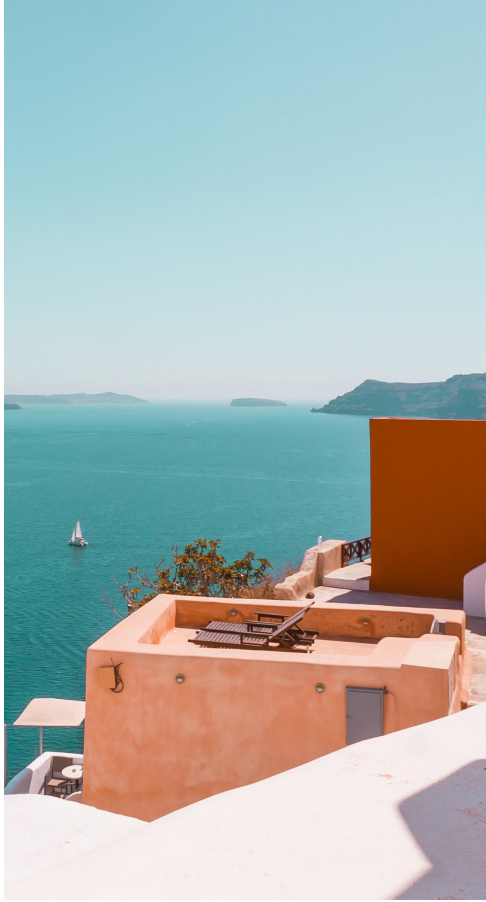
Centralised management (single framework) covering all workflows:

- On-prem / openstack / EC2
- Bare metal or VMs - thousands of production instances, hundreds of clusters
    - our production-like development MySQL environment has similar scale

Centralised management offloads a lot of work from developers.

- upgrades, application access management, compliance, backups, auto-scaling, automated online schema changes, ...

Self-managed cloud vendor offerings are being investigated.



Upgrading to MySQL 8.4 at Booking

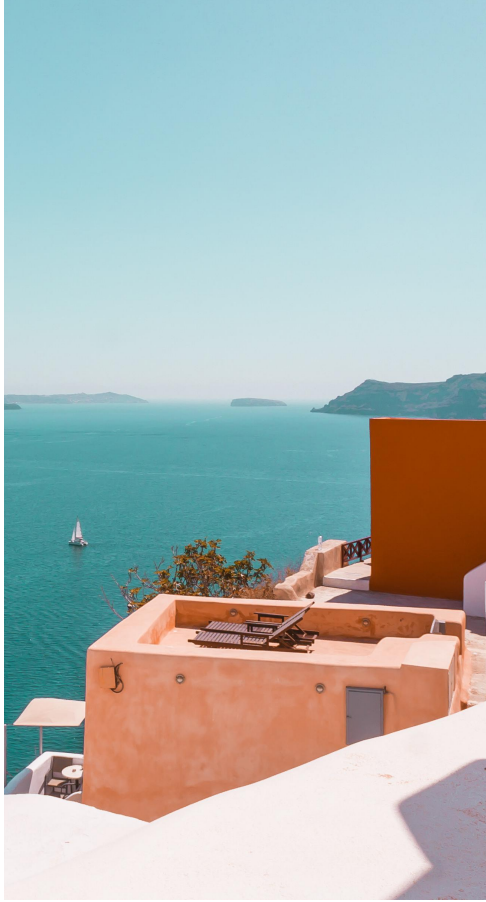
Why?

B.

Upgrading to 8.4: Why?

The **usual** reasons make sense:

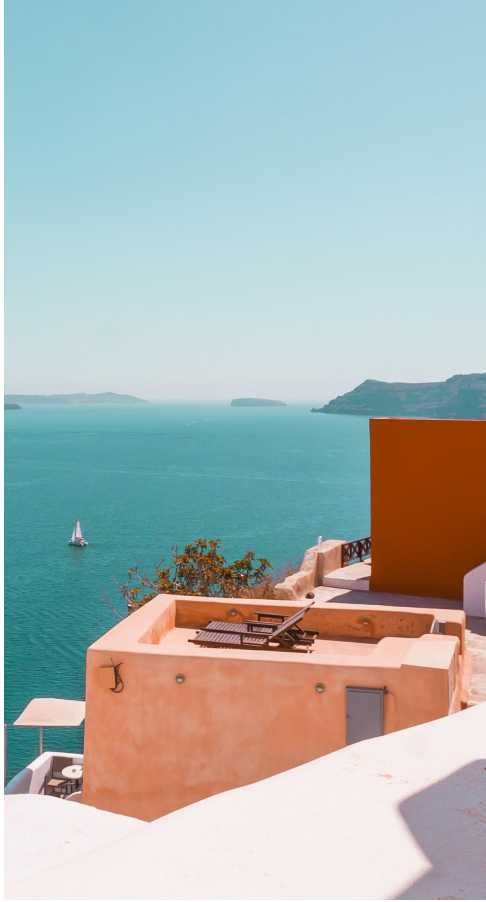
- MySQL 8.0: EOL April 2026 → ~1 year → **Get off 8.0 !!!**
- MySQL 8.4: latest stable version
- Take advantage of "New features"
- Take advantage of "Improved performance"
- Stay (more) secure



Upgrading to MySQL 8.4: Why?

However:

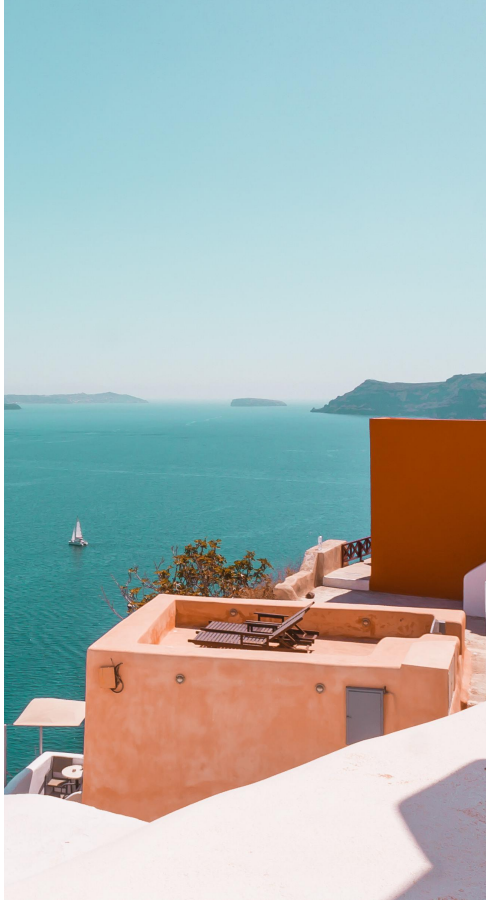
- MySQL 8.4 has few new features and appears to be mainly a partial cleanup of old code in MySQL 8.0 and earlier - good but not exciting
- New hypergraph optimiser in code base but not built into code (disabled) to allow for experimentation - make it easier for us to test new code
- open telemetry plugin is enterprise only
- the MySQL 8.4 upgrade feels like a "because I have to" upgrade



Upgrading to 8.4: Why?

More importantly the main benefit of the MySQL 8.4 upgrade is:

- test out new features in 9.X prior to it going LTS in 9.7 in April 2026 (~1 year)
    - provide feedback to Oracle on changes we like or need
    - not much time to do that
    - few new features to test up to 9.2.0
- looking forward to surprises over the next five 9.X innovation releases



Upgrading to MySQL 8.4 at Booking

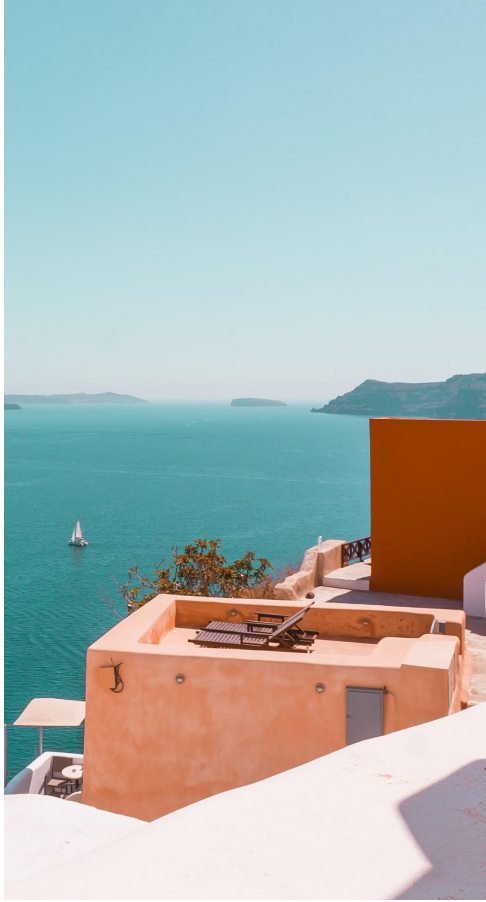
How?

B.

Upgrading to 8.4: How?

Considerations

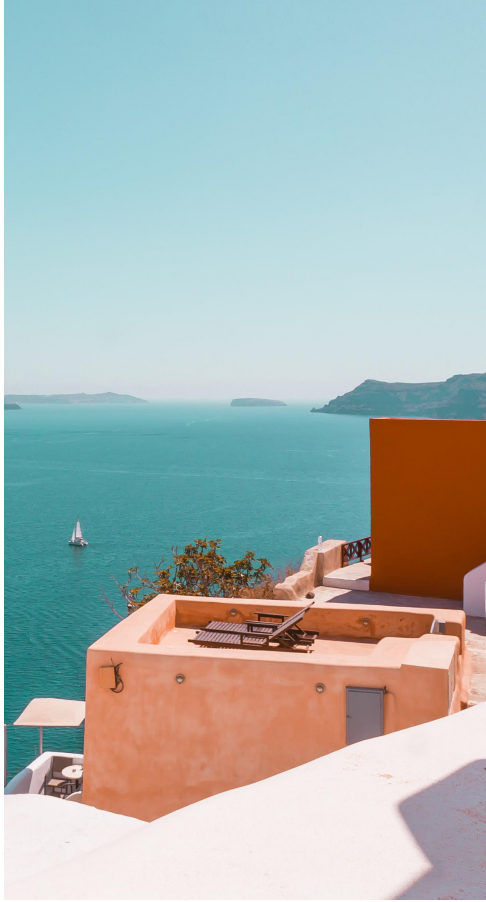
- We have quite a large MySQL fleet so our environment is complex
    - thousands of servers
    - hundreds of clusters
    - MySQL clusters span 3 primary regions
    - 3-tier replication topology using GR or semi-sync depending on workloads
    - no downtime (within usual SLAs) allowed



Upgrading to 8.4: How?

From a simple perspective all that is required is to check/adjust:

- SQL consumers continue to work - few changes here
    - removal of `mysql_native_password` (can be reinstated) forces client credential rotation, use of TLS and may be problematic
- configuration changes: cleanup of many settings in 8.0 and master/slave terminology removal
    - also affects column names in outputs from some tables/commands
    - applications / users also monitor replication status
- binlog consumers continue to work
    - many binlog consumer applications are community lead and fail to keep up with changes in MySQL. binlog format changes such as GTID format allowing tags is an example. May be problematic for downstream pipelines
    - may block upgrades if they can not adapt quickly

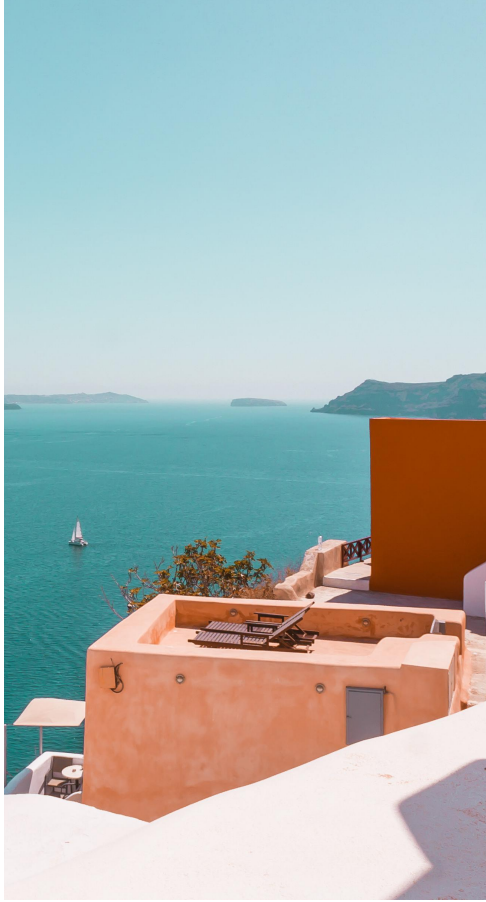


Upgrading to 8.4: How?

In practice what do we do?

Many things to consider technically:

- interaction with other internal systems
- use of other external software
- internal self-built MySQL plugins: upgrade for new versions
- validate all cluster workflows work with the new version
    - provisioning
    - upgrades
    - backups / restores
    - pool management
    - topology management
    - user credential management
    - binlog stream consumers
    - failover and switchover handling
    - monitoring
- ensure clients are updated to latest version



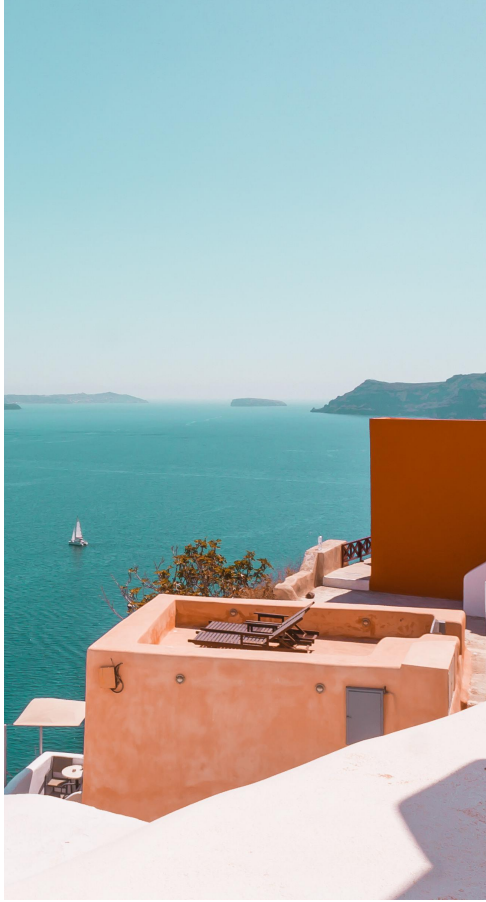
Upgrading to 8.4: How?

At booking.com

- infrastructure management uses old master/slave terminology for replication management and monitoring.
- A large change is required in code base which must support both versions during the transition from 8.0 to 8.4
- This could have been done earlier with less impact but getting time to fix something that is not broken may not be prioritised

Conclusion

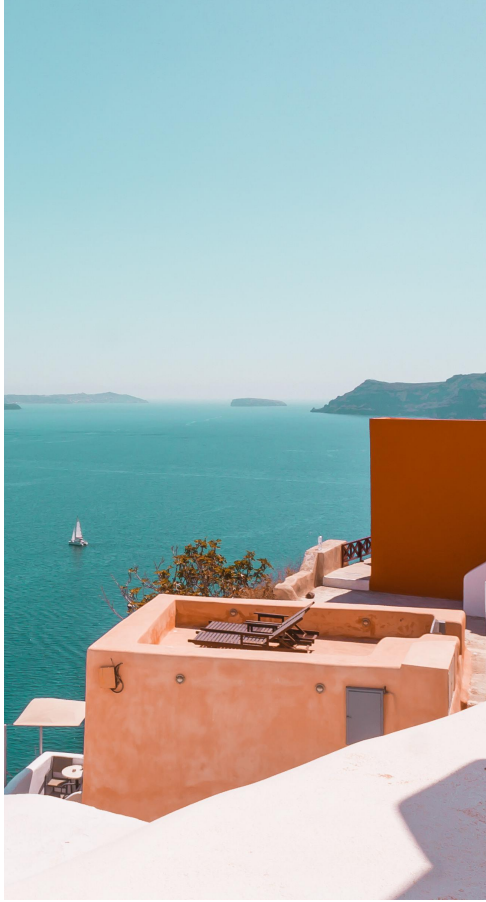
- keeping up to date with MySQL changes is really important and should not be overlooked or delayed
- It would be helpful if Oracle where to highlight this more
- Clearer deprecation timeframes would be useful - helps prioritise work



Upgrading to 8.4: How?

Recommendation

- do not upgrade in-place until you have tested the upgrade procedure
- read the manual
  - most likely the 8.0 configuration will not be understood and the server will refuse to start
  - if you fix the configuration the server may startup, upgrade and lock you out if you were using `mysql_native_password` for your "administration user(s)"
  - upgrade all users using `mysql_native_password` to use `caching_sha2_password` or temporarily use `mysql_native_password=ON` in `/etc/my.cnf` as a temporary workaround
- use replication to test read workloads first on real data
  - this allows final transition from 8.0 to 8.4 in a few seconds as you change the endpoint of the master



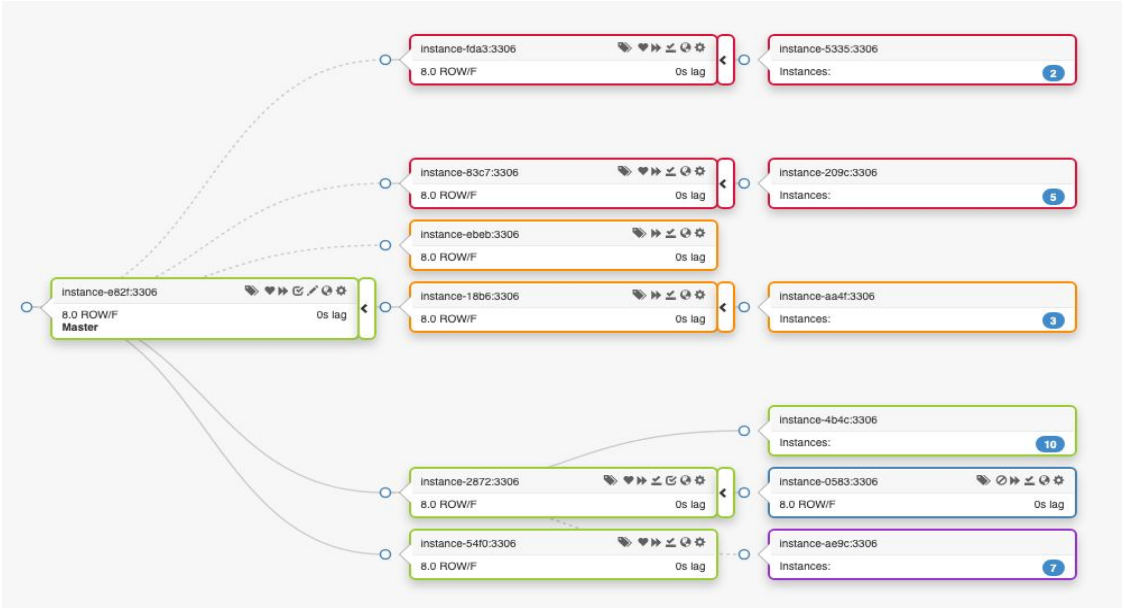
Upgrading to 8.4: How?

In practice what do we do?

We have shared in many presentations our typical topology:

- 3 regions, colour coded (more shown here)
- writes to master: LHS
- reads to replicas: RHS
- intermediate masters distribute data to replicas and are for failover
- for large clusters they also offload the binlog "load" from the master

Real topologies can be more complex involving deeper levels to leaf replicas as clusters are split up.



Upgrading to 8.4: How?

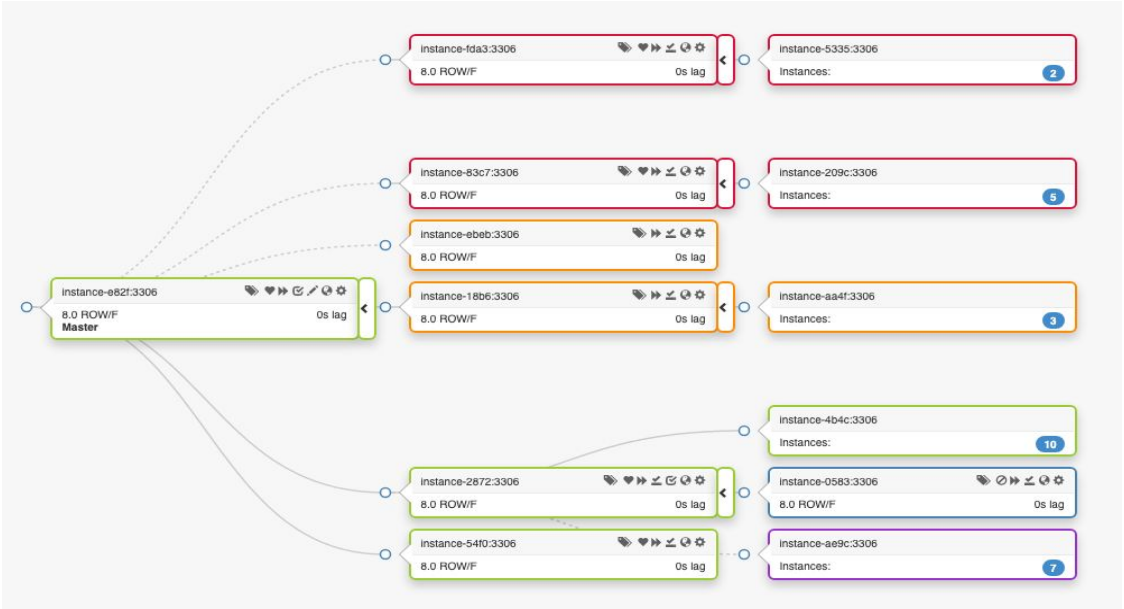
In practice what do we do?

Upgrades to 8.4 move from right to left:

1. leaf replicas
2. intermediate masters
3. masters

GR clusters are almost the same:

1. leaf replicas
2. GR secondaries
3. GR primary by shutting down the primary and triggering a GR failover to one of the new secondaries



Upgrading to 8.4: How?

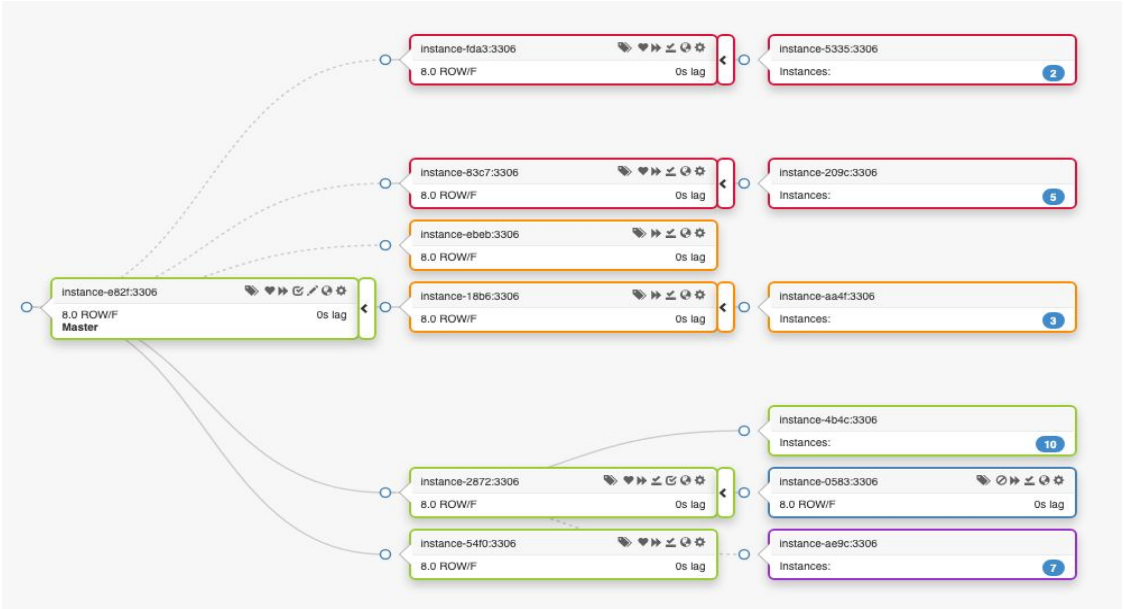
In practice what do we do?

Different optional workflows

- inplace upgrades on inactive server
- provision a new server copying data from an existing 8.0 (or 8.4) server

Incremental process

- initially on canary servers verifying that everything works as expected
- later as a generic upgrade step upgrading the whole cluster
- upgrade cycle within a cluster may run a mix of 8.0/8.4 for several weeks



Upgrading to 8.4: How?

Risks

- We reduce risks by monitoring carefully and are able to backout up to the primary replacement.
- Developers also monitor their workloads are unaffected by the upgrade

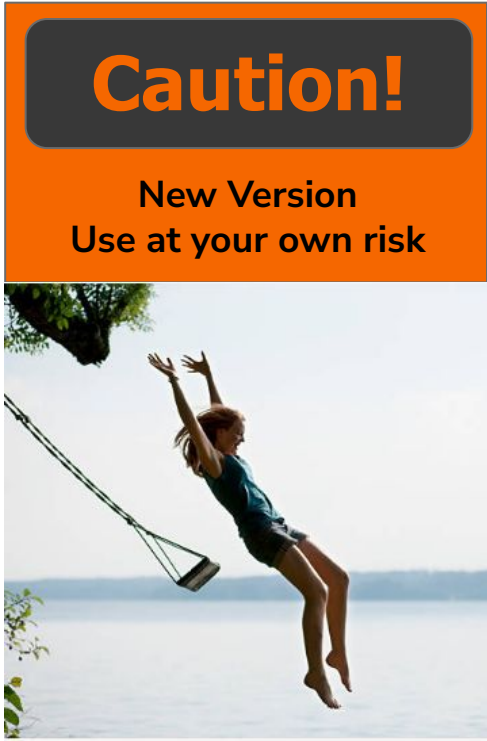
Usual causes:

- internal tooling issues
  - adjust tooling to correct issues
- applications seeing a change in behaviour or a performance degradation
  - report to Oracle to verify and resolve if needed or adjust queries
- MySQL bugs
  - report to Oracle to verify and resolve if needed

If there is a problem reclone replicas from existing 8.0 servers

Once the final primary/master is upgraded there's really no going back

- issues seen here involve us finding a fix or workaround and moving forward

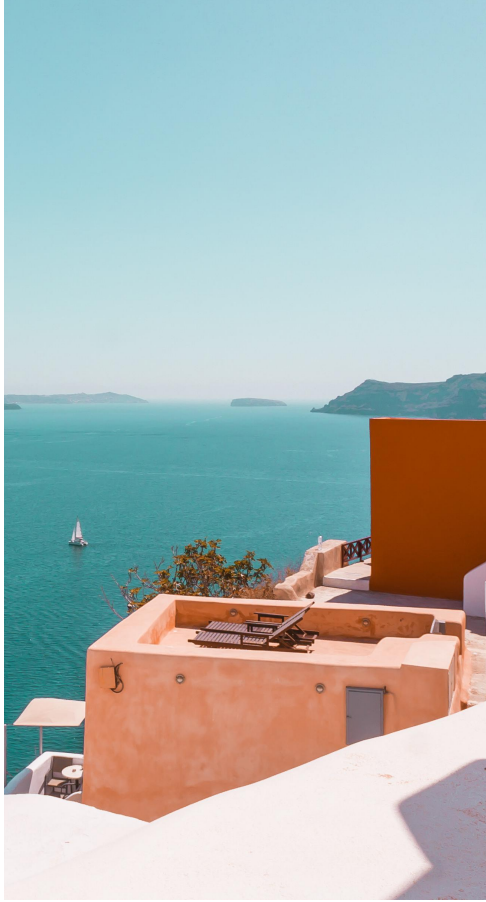


Upgrading to 8.4: How?

Practical Considerations

MySQL clients - I

- Do not forget to notify/remind users to update their clients
- Oracle recommends using latest 9.X version, currently 9.2.0 that they provide
- Many programming languages not supported by Oracle-provided drivers
    - Ensure the drivers you are using are up to date and support the latest 9.X (and earlier) versions
- Client libraries and versions can be detected by querying `performance_schema.session_connect_attrs`
    - Find users on old versions and remind them to upgrade



Upgrading to 8.4: How?

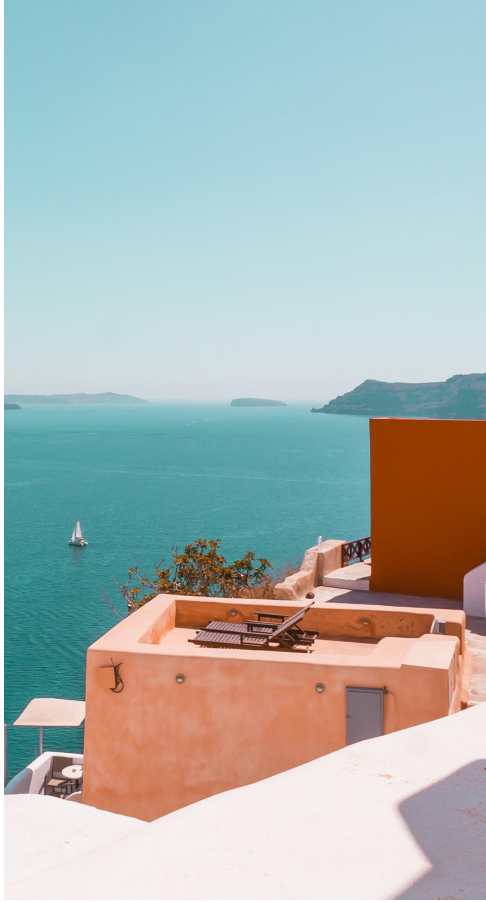
Practical Considerations

MySQL clients - II

Example

os	platform	client_name	client_version	COUNT(*)
Linux	x86_64	libmysql	8.0.34	7804
Linux	x86_64	libmariadb	3.3.1	2796
Linux	x86_64	libmysql	8.0.28	19
Linux	x86_64	libmysql	8.0.36	1
Linux	x86_64	libmariadb	3.3.8	3
Linux	x86_64	libmysql	8.0.40	1
Linux	aarch64	libmysql	8.0.40	4

`libmariadb` was surprising –> turned out to be ProxySQL

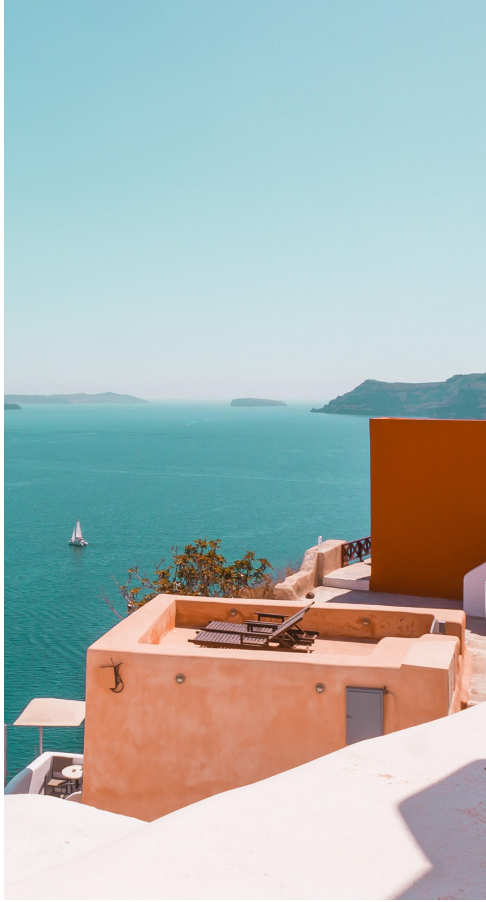


Upgrading to 8.4: How?

Practical Considerations

Grant Management

- this is managed centrally
- `mysql_native_password` to `caching_sha2_password` authentication change required for all users to allow 8.4 to be used
- `mysql-native-password=ON` allows testing before this is complete
- handle % wildcard database grants are used in a few places so grant management changes needed
- we rotate full credentials, user/password, rather than just the password as code was written before MySQL's secondary password was implemented



Upgrading to 8.4: How?

Practical Considerations

Orchestrator

- Like other companies booking.com uses Orchestrator, originally written by Shlomi Noach, but no longer maintained by him
- Provides excellent cluster health visualisation and failover handling
- Upstream Orchestrator is not 8.4 aware so this needed resolving
- Percona helped and we are now running an updated version of Orchestrator which is MySQL 8.4 aware
- Relocation testing can be done immediately
- Failover testing will be done once the master/primary is updated to 8.4 and can be done on temporary non-production test clusters

Upgrading to 8.4: How?

Practical Considerations

Orchestrator

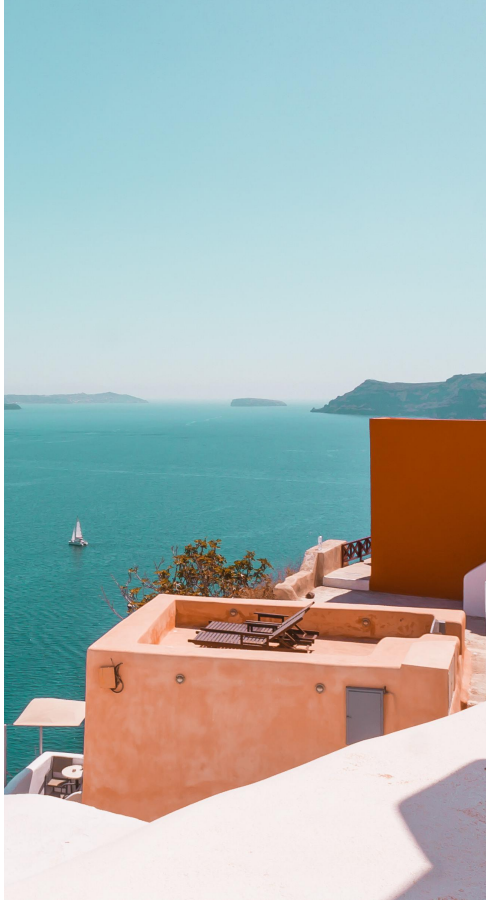
- Orchestrator does not failover GR as unnecessary (handled by the group)
- Oracle has added failover improvements to MySQL allowing:
  - async replicas behind a GR cluster to auto-relocate to another member
  - async replicas to be able to relocate under a different source MySQL server
  - this may reduce our need to use orchestrator for failover handling
- They will not replace Orchestrator's excellent topology visualisation.

Upgrading to 8.4: How?

Practical Considerations

MySQL Plugins I - usage

- booking.com uses internal MySQL plugins for various reasons
  - PAM plugin, code borrowed from Percona Server
  - a max disk usage plugin allowing warnings to be sent to developers if disk usage is too high (from Daniël van Eeden)
  - a couple of other simple, but fast and specialised audit plugins integrated into our infrastructure used for security reporting
  - the audit plugin was not used as only available for the enterprise version, and our plugins are very lightweight

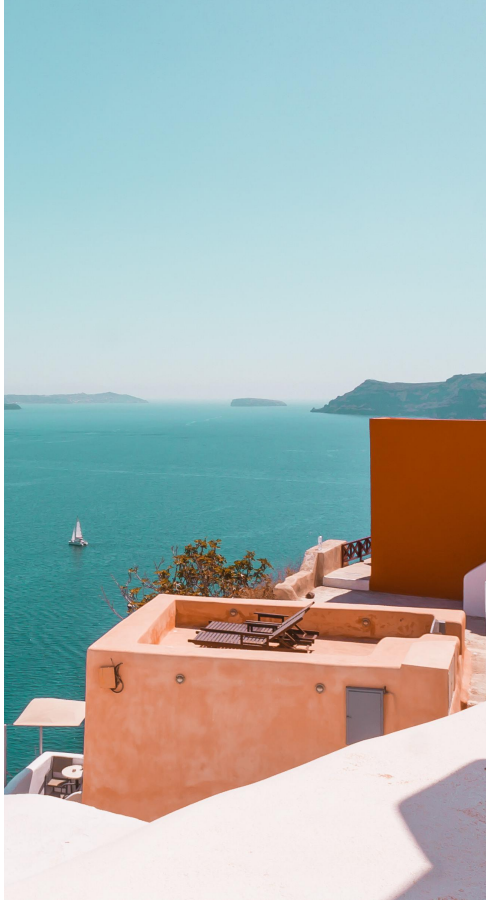


Upgrading to 8.4: How?

Practical Considerations

MySQL Plugins II - maintenance effort

- These plugins have been running since MySQL 5.7. The plugin build infrastructure was mainly manual and plugins updated infrequently
- plugin version/interface stability has been very good only biting us once with a change in the PAM interface requiring us to get an early patch from Percona
- Original plugins were built against CentOS 7 / 8, extended to cover OracleLinux 9 until the recent EOL of the CentOS versions
- With the upgrade to MySQL 8.4 extra CI pipelines have been created ensuring we can build MySQL 8.0, 8.4 and 9.X plugins in a fully automated manner and do this with every quarterly MySQL release
- Percona releases Percona Server a little after MySQL is released so we build end up with 2 builds to get a fully up to date PAM plugin
- gcc-toolset improvements between 9.X versions also required some temporary patching of the PAM plugin due to incompatibilities between 9.0 and 9.1 (now resolved with 9.2 vs 9.1)

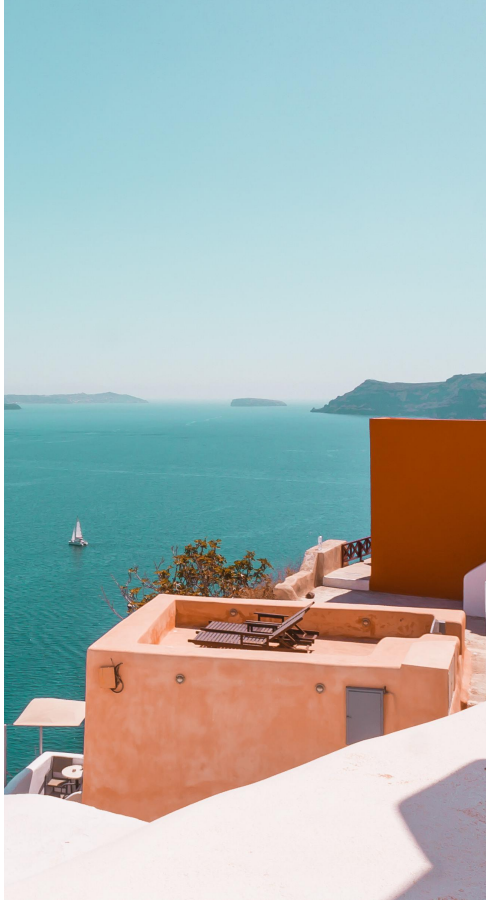


Upgrading to 8.4: How?

Practical Considerations

MySQL Plugins "going away"

- Oracle staff keep telling me that plugins are "going away" to be replaced by "components" but this has not happened
- I hope to see all plugins in 8.4 replaced by components in 9.7 LTS, perhaps with a backwards compatibility layer to give people some more time to migrate
- Fred has provided examples but more blog posts with code examples would be helpful as "plugins" and "components" allow for MySQL to be extended for specific use cases without having to directly patch the server code.
- I would like to see clearer documentation on the internal interfaces the server provides which can be used by components so users of MySQL can know what sort of things they can do with a component.
- https://dev.mysql.com/doc/refman/8.4/en/components.html does not provide this information, but https://dev.mysql.com/doc/dev/mysql-server/latest/ seems to be the place to look.

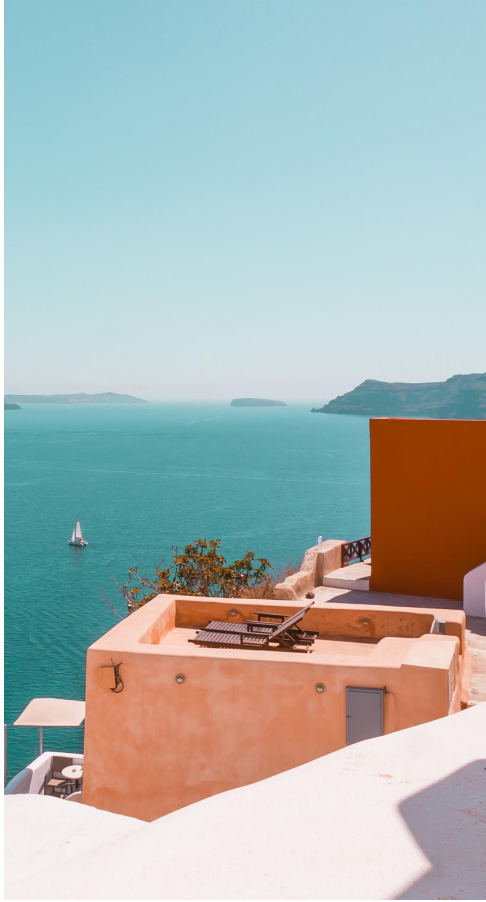


Upgrading to 8.4: How?

Practical Considerations

Group Replication - 2 clusters for testing or while upgrading

- testing a new major version of MySQL with Group Replication is complicated
- code changes from 8.0 to 8.4 may be important
- MySQL 8.4 provides improved instrumentation for GR
- it is easy to test secondary behaviour in a cluster by upgrading a single member and verifying its behaviour
- failover scenarios will now exclude the new higher version from being a candidate for replacement so it reduces the cluster's resilience
- to test the primary behaviour you must upgrade all other members first
    - requires also upgrading all other replicas under the GR cluster
    - this can delay GR upgrade testing

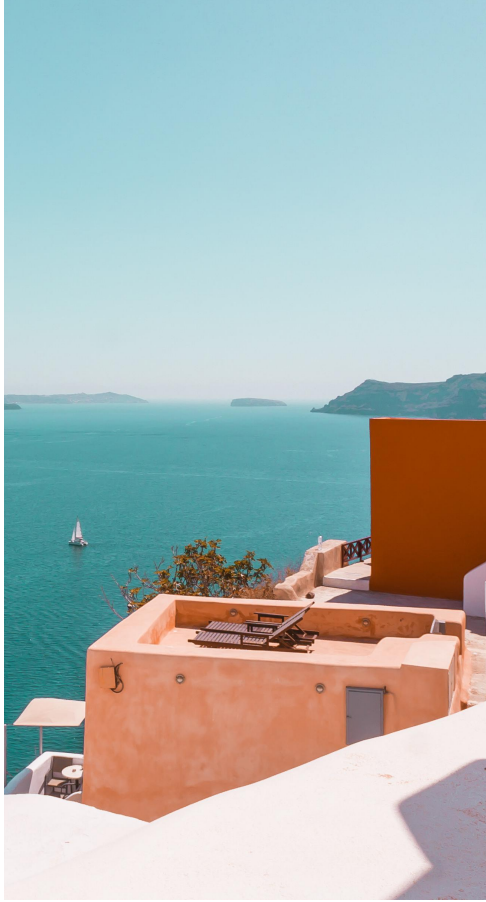


Upgrading to 8.4: How?

Practical Considerations

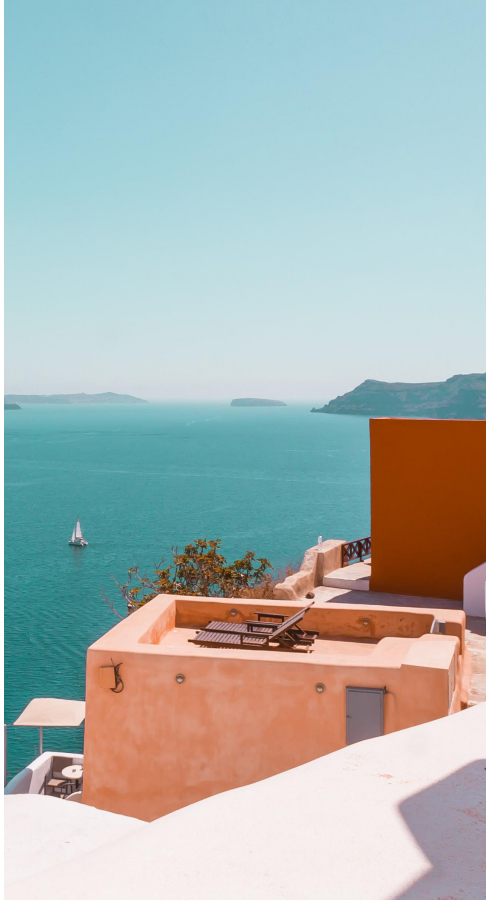
Group Replication - 2 clusters for testing or while upgrading

- a better, but more complex solution, is to build a new GR cluster on 8.4 and make it replicate from the 8.0 GR cluster, adding 8.4 async replicas as required
- this requires more complex tooling, but is safer and allows us to be more confident of the 8.0 to 8.4 GR upgrade process rather than doing this as a "big bang"
- it is not efficient from a resource usage perspective



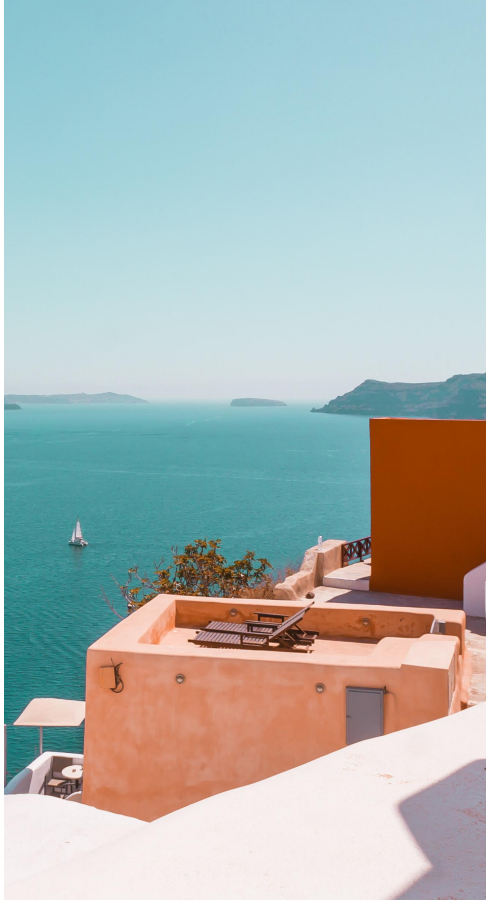
Upgrading to 8.4: Progress

- Projects get delayed due to other business priorities so our 8.4 upgrade is still in progress
- Actual progress is still in early stages though no major issues seen
- Our automation just works so it is largely invisible and we forget how much it is used
    - the work to change terminology used is under way but the impact of spinning up an 8.4 server on our existing infrastructure and seeing the breakage has been quite enlightening
    - changes in code with a global scope over our fleet management need to be done incrementally to reduce risk
- GR cluster upgrades look to be interesting, building a second 8.4 GR cluster and replicating from the 8.0 GR cluster looks like a safer way to verify both our code works as expected but also GR does not give us a surprise
- Once the terminology code changes are completed I expect this to be just another major version upgrade, something we have done many times in the past
- binlog consumers are a concern as users tend to use tooling that is often not fully up to date



Upgrading to 8.4: Issues Seen

- Just upgrading to 8.4 without checking anything can lock you out due to `mysql_native_password` removal
  - Oracle: do not upgrade the server to 8.4 if there's no "root privileged" user with a usable authentication mechanism.
  - e.g. mysql-native-password=ON in /etc/my.cnf is your friend as an intermediate workaround while testing real systems
- Semi-sync plugins: there are 4 plugins, master/slave, source/replica and they do not work well together if mixed. Cloning from a server configured with master/slave to one configured with source/replica does bad things and leads to a broken server with all 4 plugins "loaded" at once but neither working
- Hypergraph optimiser code in 8.4+ is not compiled for use in community version
  - this is a new optimiser intended for analytical workloads and used by Heatwave in OCI
  - it would be interesting to use experimentally, allowing feedback to Oracle on non-heatwave servers
  - It is trivial to patch the build process to enable it but that should not be necessary

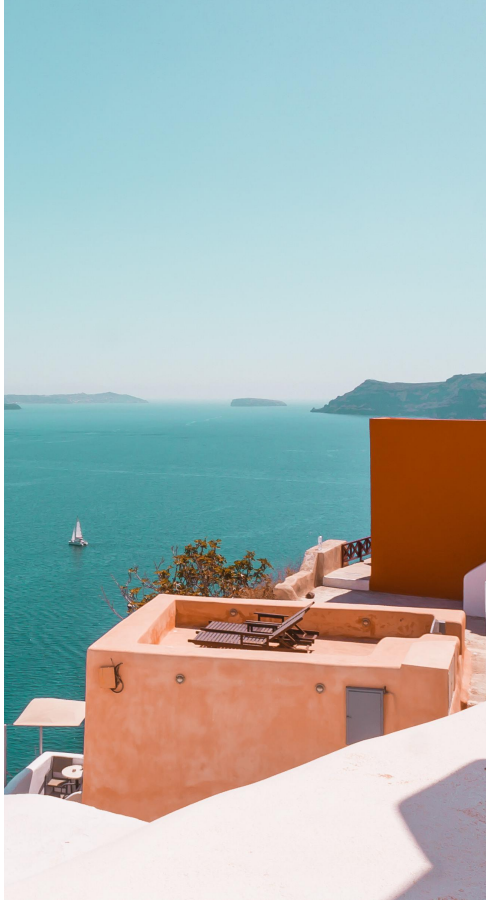


Upgrading: Missing Features

Make our life simpler:

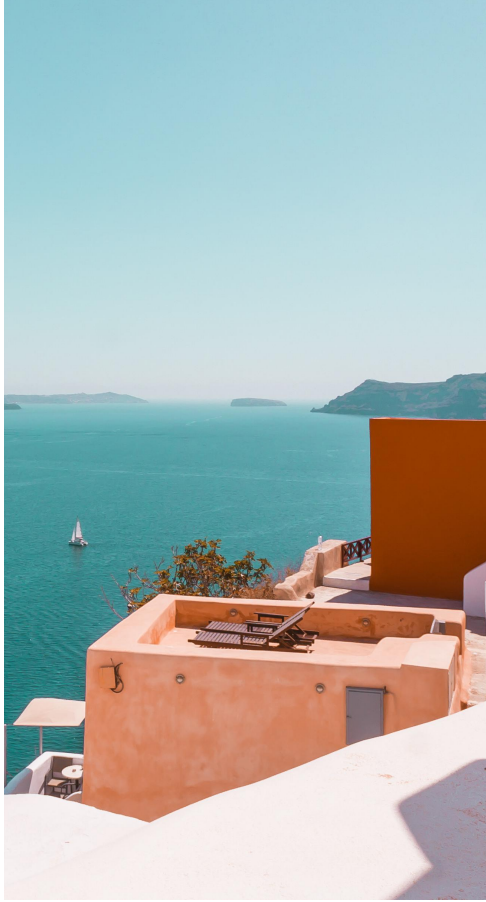
- **native cloning: allow cloning from previous major version**
  - this is the fastest cloning mechanism
  - after cloning, the server restarts and can do an inplace upgrade
  - both *native cloning* and *inplace upgrade* support *already exists*
- **clone a new mysql server directly from the command line**

```
# configuration stored in datadir in persistent variables
mysqld \
  –user=<whatever> \
  –datadir=/path/to/datadir \
  –clone-from=user@host:port \
  <custom parameters to override remote settings>
```



Conclusion

- It is time to upgrade to 8.4 LTS and prepare for 9.7 LTS. **Do not delay**.
  - keeping up to date with MySQL changes should not be overlooked
  - Oracle could highlight to users the need to pay more attention
- Upgrading is not hard, but if not done already the master/slave -> source/replica migration may be painful if you manage your own clusters and the topology is complex
- GR upgrades are more complex if you want to test thoroughly (or have multi-region clusters) as they entail higher risks
- Oracle: move Plugins to Components by 9.7 LTS so we can move with you
- Looking forward to seeing significant new features before 9.7 LTS is released



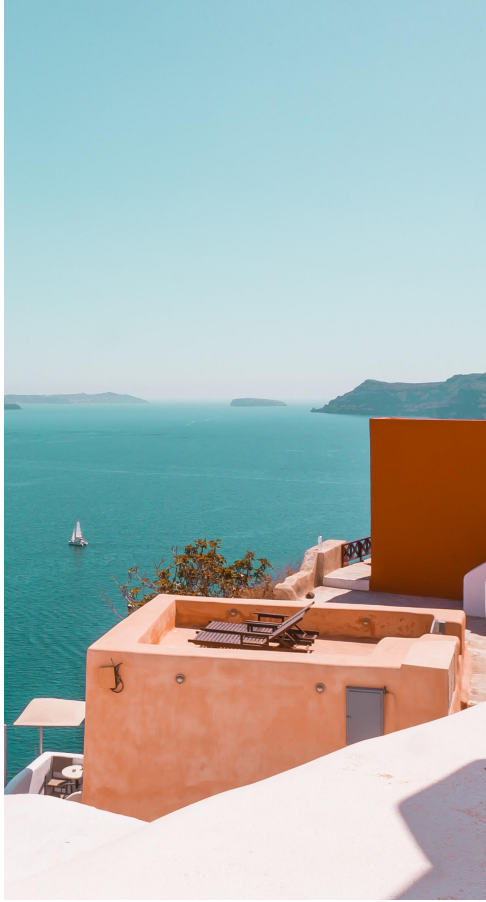
External References

Bugs / Feature Requests

- https://bugs.mysql.com/115051: Improve behaviour of GR primary promotion selection between members
- https://bugs.mysql.com/117291: Improve migration handling of semi-sync plugins from 8.0 to 8.4
- https://bugs.mysql.com/112808: Please build MySQL with the hypergraph optimizer enabled but configured off
- https://bugs.mysql.com/115051: Improve behaviour of GR primary promotion selection between members

External Software

- https://github.com/sjmudd/percona-pam-plugin-for-mysql: Percona PAM plugin rpm rebuilder for MySQL 8.0 / 8.4 / 9.X
- https://github.com/dveeden/mysql_maxdiskusage: MySQL Disk Usage Plugin
- https://github.com/percona/orchestrator: Percona's Orchestrator




Booking.com
Empowering people to experience the world
Join Us: https://jobs.booking.com/careers
Thank you
Simon J Mudd
simon.mudd@booking.com
MySQL