Zero Downtime Deployments with Database Migrations

Bob Feldbauer
twitter: @bobfeldbauer
email: bob.feldbauer@timgroup.com

Deployments

- Two parts to deployment:
  - Application code
  - Database schema changes (migrations, evolutions)

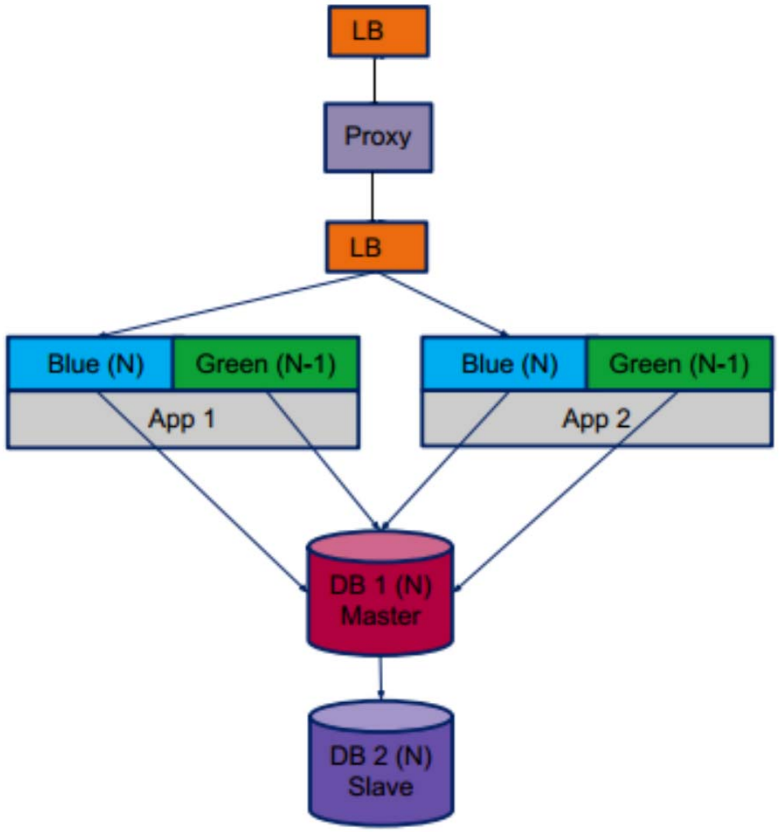
Deployments

- Two parts to deployment:
  - Application code
  - Database schema changes (migrations, evolutions)

- Zero downtime deployments for application code can be solved by load balancing and implementing a Blue-Green deployment pattern

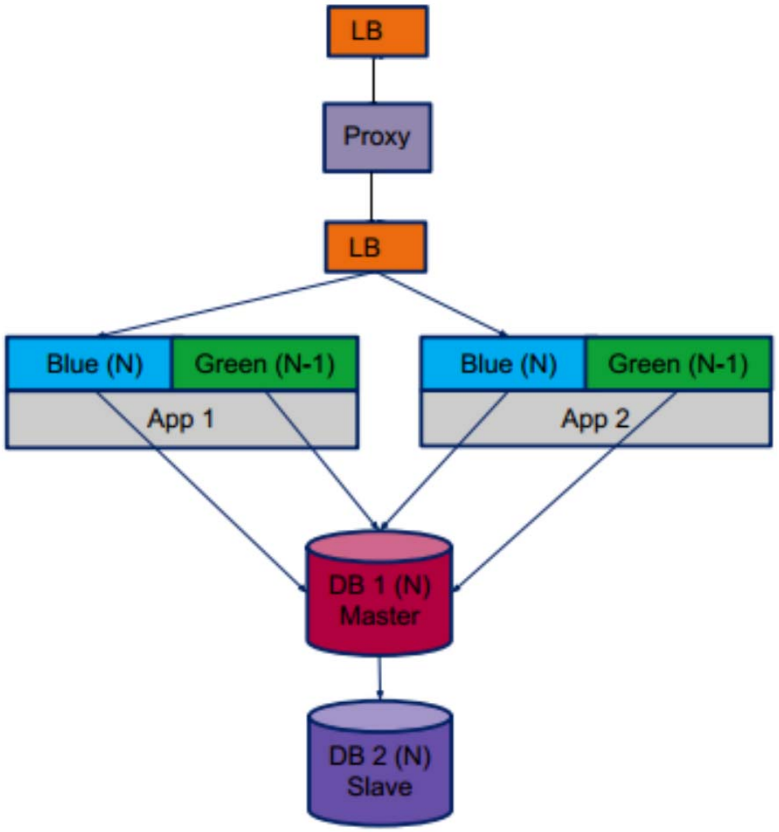
Application Deployment

- Let’s walk through a sample high-availability architecture for an application



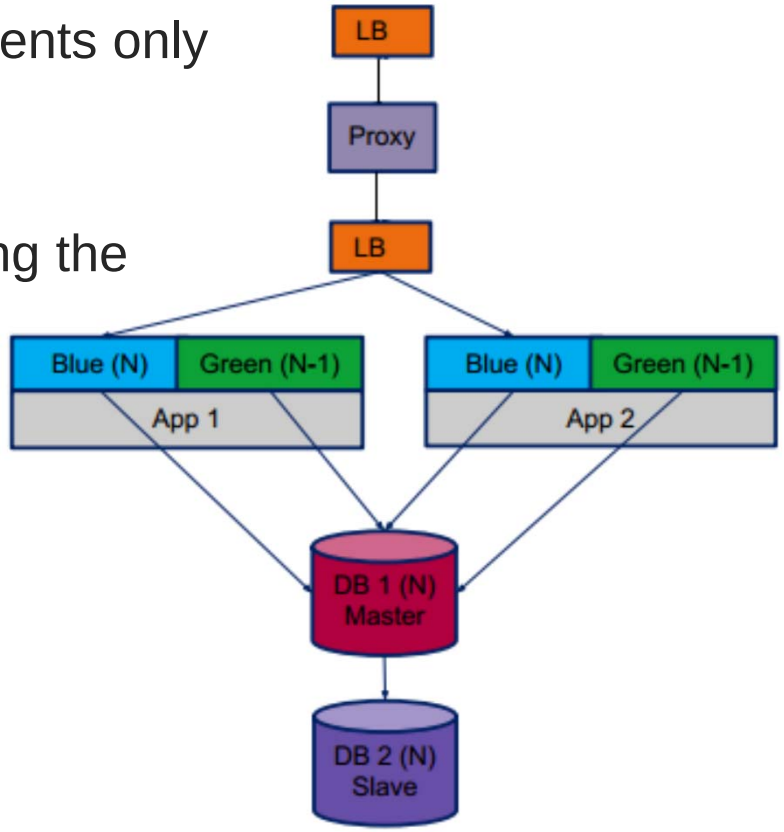
Application Deployment

- Let’s walk through a sample high-availability architecture for an application
  - Incoming traffic hits a pair of HA load balancers
  - Routed to a pair of HA web proxies
  - Proxied to a pair of HA load balancers
  - Routed to a pair of HA app servers in a Blue-Green versions (N, and N-1 respectively)



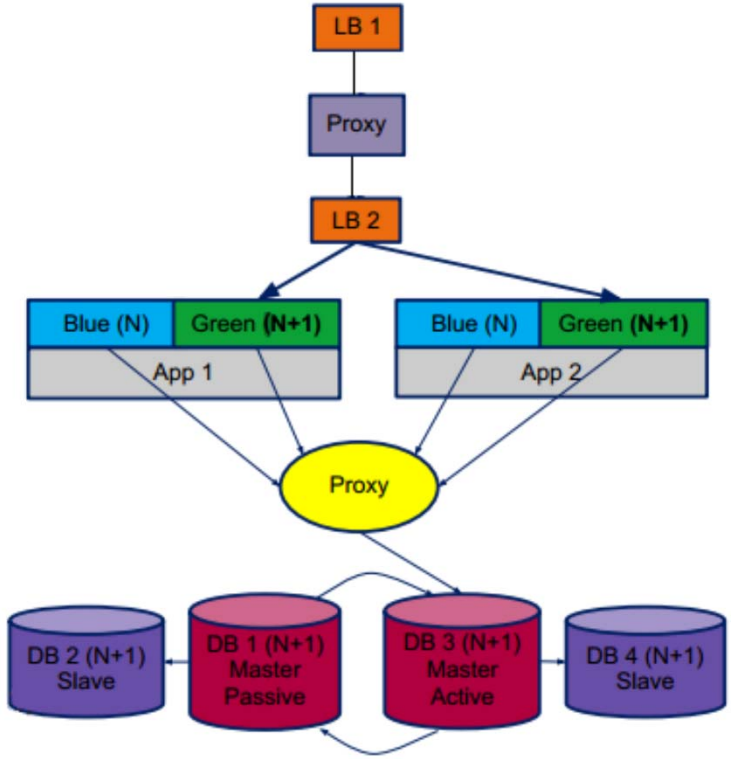
Application Deployment

- There’s no problem here, as long as deployments only consist of application code
- Database schema changes that require locking the database are problematic



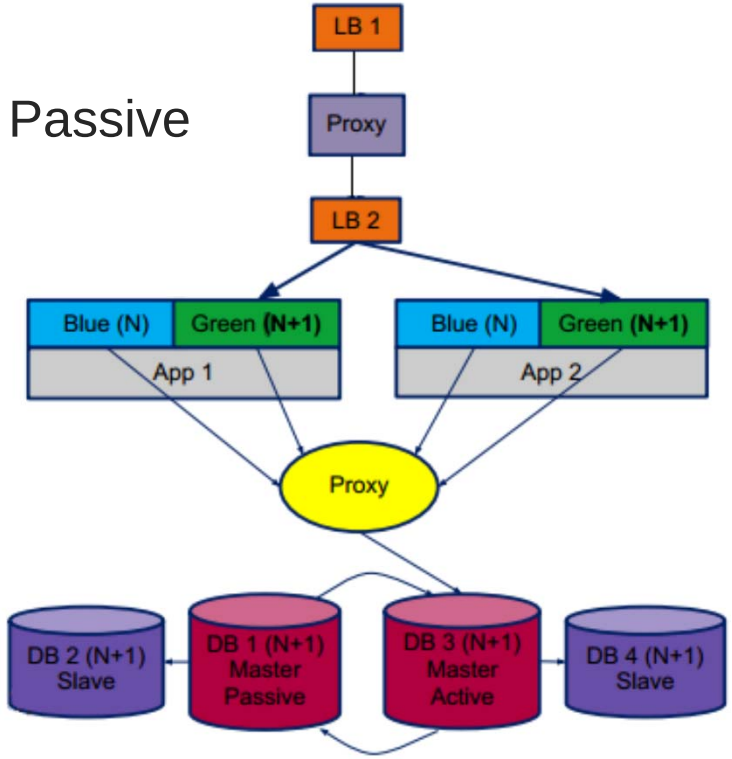
Application Deployment in zero downtime DB-world

- What if we could add a proxy in between the app and DB layer?



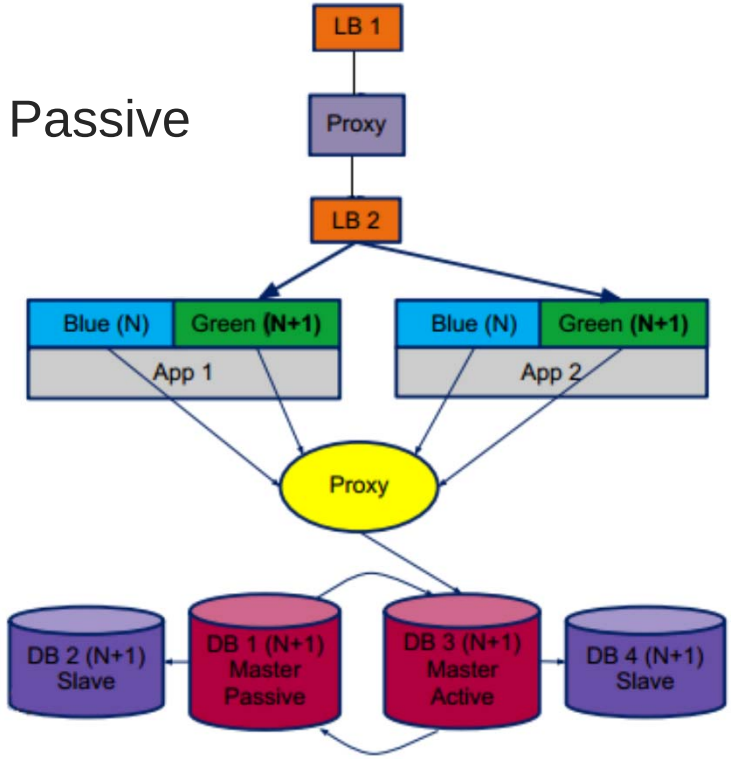
Application Deployment in zero downtime DB-world

- What if we could add a proxy in between the app and DB layer?
- A proxy allows us to switch between Active and Passive database servers



Application Deployment in zero downtime DB-world

- What if we could add a proxy in between the app and DB layer?
- A proxy allows us to switch between Active and Passive database servers
- What are our options?
  - MySQL Proxy
  - HAProxy
  - Your favorite proxy ☺



MySQL Replication basics

- Master records changes to its data in its binary log
- Slave copies the master’s binary log events to its relay log
- Slave replays the events in its relay log and applies them
- Changes get written to the active server’s binary log and flow through replication to the passive server’s relay log.
- The passive server executes the query and writes the event to its own binary log (**log_slave_updates=1).**
- The active server retrieves the same change via replication into its own relay log, but ignores it because the server ID in the event matches its own.

Active/Passive MySQL Server Configuration

- Just make sure to set server-id to unique values for each and this is all you really need:

```
server-id=2
log_bin=/var/lib/mysql/mysql-bin.log
sync_binlog=1
log_slave_updates=1
log_bin_index=/var/lib/mysql/mysql-bin.index
relay_log=/var/lib/mysql/slave-relay.log
relay_log-index=/var/lib/mysql/slave-relay-log.index
binlog_do_db= test
```

Zero downtime database migration process

- STOP SLAVE on both the Active and Passive servers
- Run SQL for database migration on the Passive server
- START SLAVE on the Active server
- Wait for replication lag on the Active server to become small, check with SHOW SLAVE STATUS "Seconds_Behind_Master"
- LOCK TABLES on the Active server for final replication catchup
- Ensure replication lag is zero on the Active server
- Modify your proxy configuration to change the Active/Passive designations
- Unlock the new Passive server
- START SLAVE on new Active server

* basic implementation: 20 lines of Bash

Required schema modification rules

- Backwards compatible with previous schema
  - Add new column with triggers rather than modifying in place
  - New columns cannot be required immediately (for old writes to replicate)
  - No use of server-generated data functions (UUID, NOW, RAND, etc)
- Cannot conflict with pending writes
  - No auto-increment INSERT unless the app doesn't insert to that table
  - No DROP COLUMN nor DELETE rows if used in previous schema
- Enforcement possibilities
  - Test harness
  - Reject at MySQL proxy

Resources

- High Performance MySQL, chapter 8: “Replication”
- http://codeascraft.etsy.com/2012/04/20/two-sides-for-salvation/
- https://devblog.timgroup.com/
- https://www.engineyard.com/blog/2011/zero-downtime-deploys-with-migrations/
- Also, see http://mysql-mmm.org/
  (Multi-Master Replication Manager for MySQL)