BUILDING HIGHLY SCALABLE APPS

MARK PILLER
BACKENDLESS CORP

ABOUT ME

- Mark Piller
- Dallas, TX
- Founder and CEO of Backendless Corp
- Serial entrepreneur
- Technologist/Architect
- Why I am qualified to talk about scalability

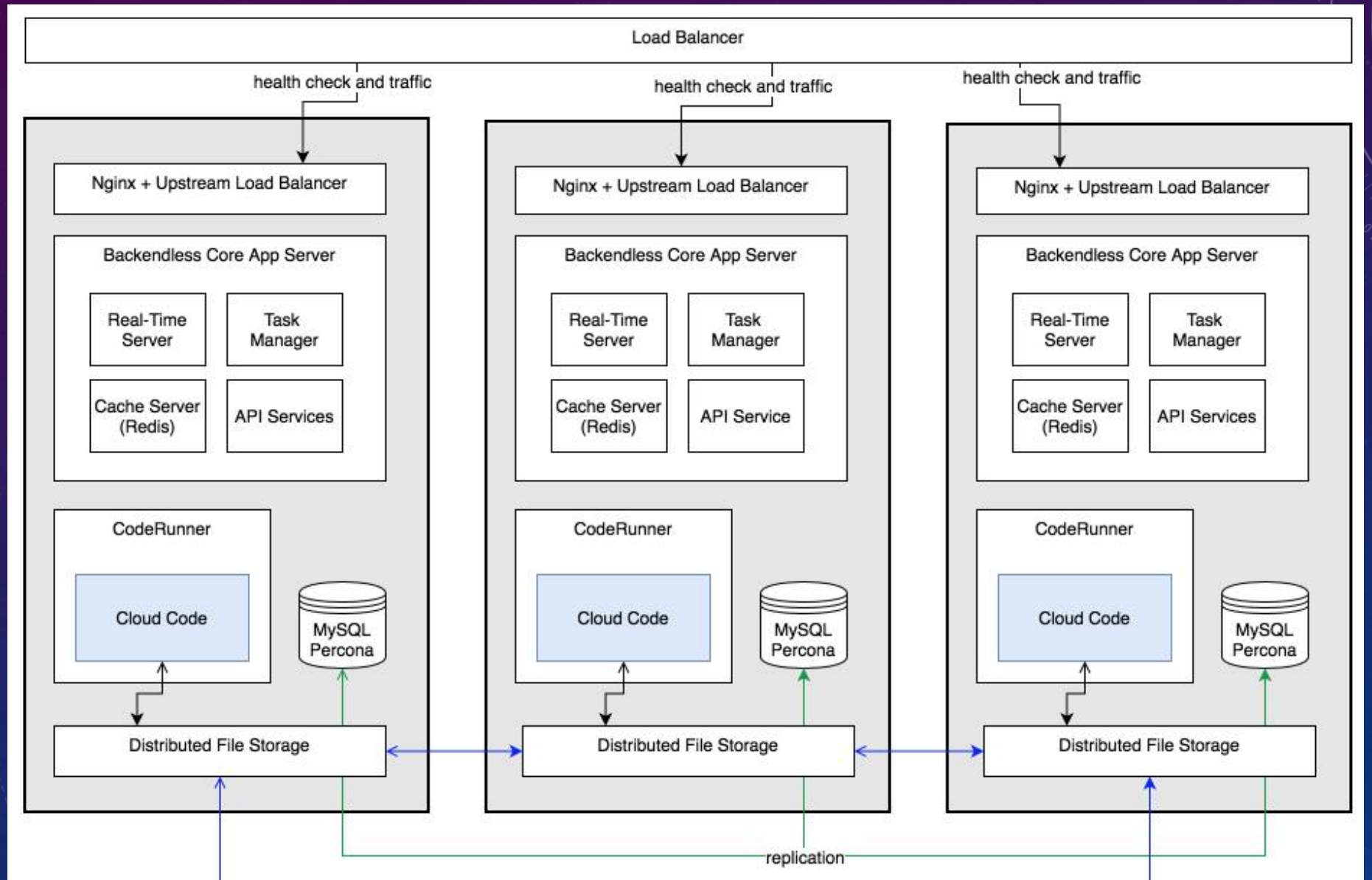
SCALABILITY

- Scalability is hard.
- “Let’s throw more servers at it” << Recipe for a failure
- Must not be an after-thought
- Scalability must be “baked” into the architecture
- Spans over every element of your solution: client-server integration, database, caching, file system, etc.

BASIC RULES

- Avoid blocking execution
- Avoid centralization
- Foster easy replication
- Stateless programming model
- Pagination
- Test and validate before committing to any component of your technology stack

REFERENCE ARCHITECTURE



TECHNOLOGY STACK

- Web server
- App server
- Database
- Cache
- Redis
- Async task processing
- File system

DATABASE SETUP

- Percona – lets us have master/master replication
- All nodes are masters
- We can instantly switch from one node to another
- ProxySQL – automatically switches to a live node
- ProxySQL – lets us split write from read IO between the nodes
- Database sharding – built into the app design

SCALING DATABASE

- Connection pooling
- Master/master replication
- DDL (schema change) operations lock tables
- Backups can lock tables. Use `--skip-lock-tables` and `--single-transaction` options
- ZFS snapshots are preferred to backups
- Do backups on the same machine as the database >> avoid network transfer
- Record counting – locks tables, avoid at any cost
- Data pagination – never retrieve all data

REDIS

- Caching
- Job/Task Queues
- Atomic counters
- Synchronization (Distributed locking)

SCALING REDIS

- Connection pools
- Slow query log
- Avoid synchronous dumps
- Check how frequently dumps are made
- Use replication
- Control master assignment with Sentinel

FILE SYSTEM

- GlusterFS is the preferred choice (for us)
- Tried CEPH. Configuration is too complex, requires more server-side resources.
- Amazon EFS – too slow with sync for large number of files

CACHING

- Ehcache is the preferred system.
- Ehcache dropped replication support in v3 to push Terracotta
- Alternatives are Infinispan and Hazelcast

ASYNC JOB PROCESSING

- Import/Export, Mass emails, Push notifications, etc
- Centralized queue in Redis
- Workers pull jobs from the queue
- Easy to scale out – simply add workers
- Why Redis?

CODE BEST PRACTICES

- Using `synchronized` is a bad idea for scalability
- Using Redis for distributed locking (google “locking with setnx”)
- Use `Future` (async computation)
- Use non-blocking operations

KUBERNETES

- Let’s you scale fast (single click or through automation)
- Scaling is done by adding pods to workers
- A single pod can contain multiple docker images, but we assign only one
- Use Rancher for orchestration.

MONITORING

- Monitoring is essential
- Expect failure
- Monitor everything (cpu, memory, file storage, queues, connection/thread pools, bandwidth, response time, etc)
- Tools to consider: M/Monit, Zabbix

THANK YOU

- Contact info:
  mark@backendless.com
  twitter: @midnightcoder, @backendless