Kickoff with scale in Mind

Server/Code Solution Architecture to scale easily as you grow

What is (Ready to Scale)

Ability to grow fast without the need to change your architecture

Should I always thing of Scale?

If you want to grow !

Outline

- Server Setup Structure
  - QA-1
- Web/API Application Servers
  - Scaling using NodeJS
  - QA-2
- Data Storage
  - Scaled Storage
  - QA-3
- Final QA

What is Vertical VS Horizontal Scaling Mean?

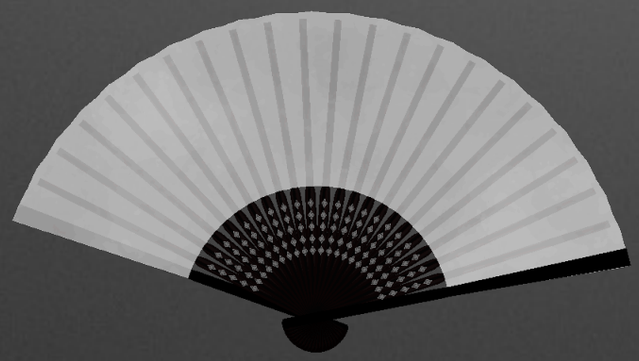
Vertical ex: (Grow Server Resources)

Horizontal ex: (Expand into others servers)

Server Setup Structures

Linear

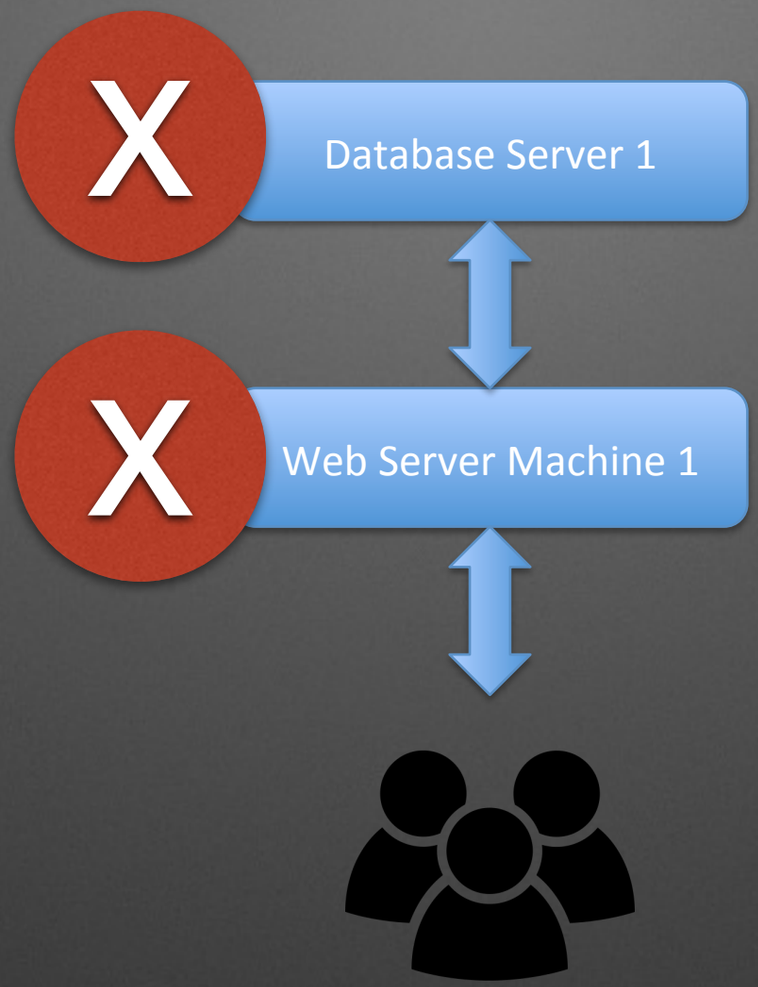
Diamond



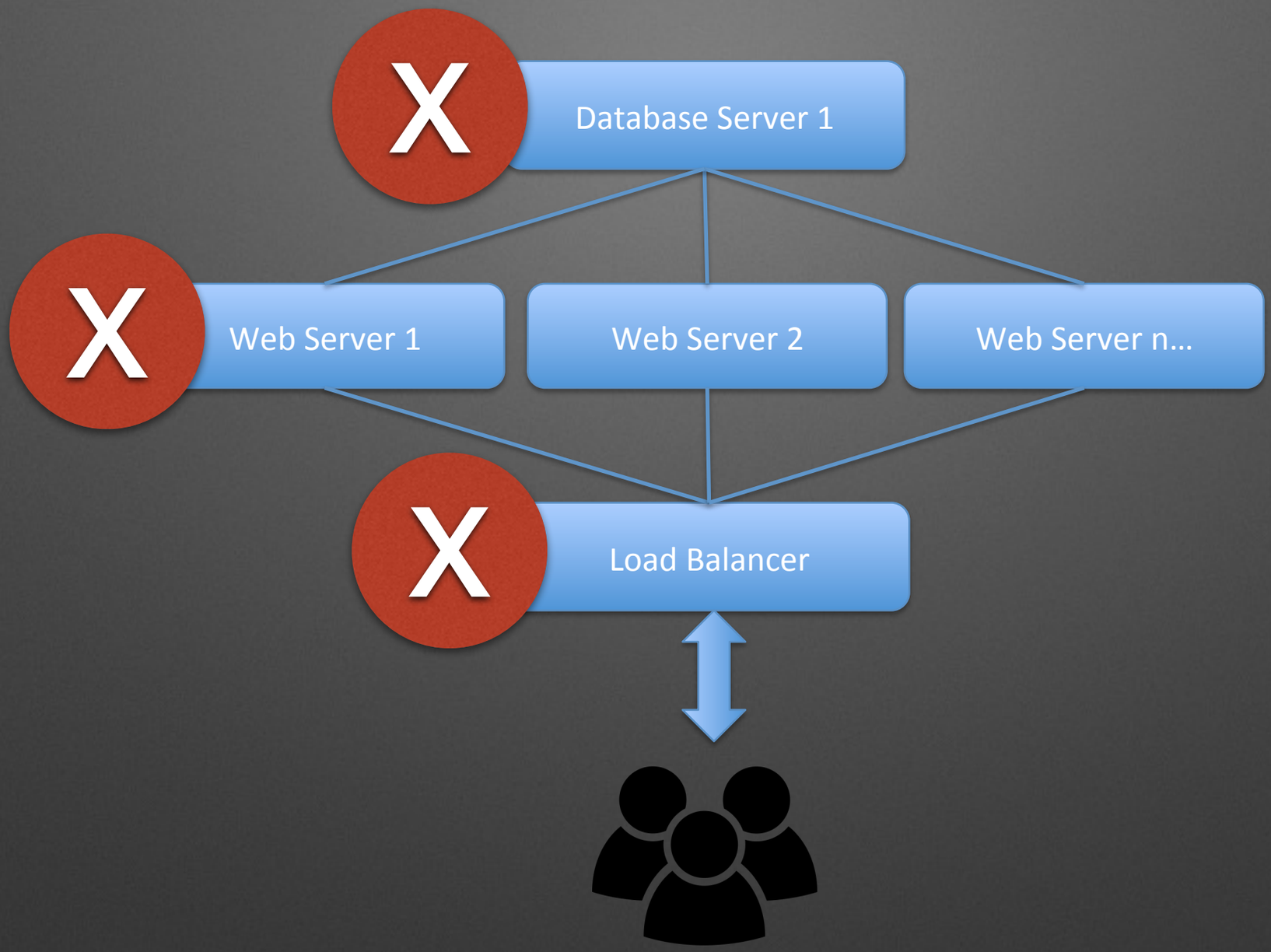
Fan-out

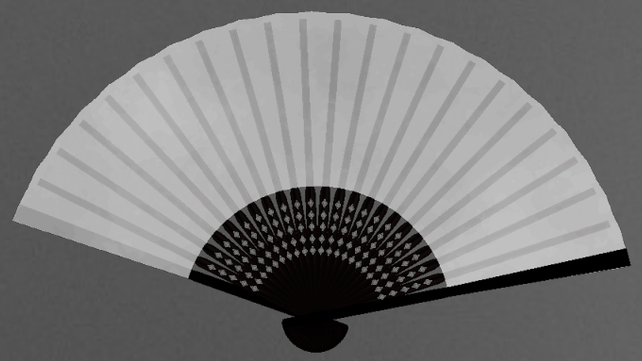
Multi-Fans

Linear

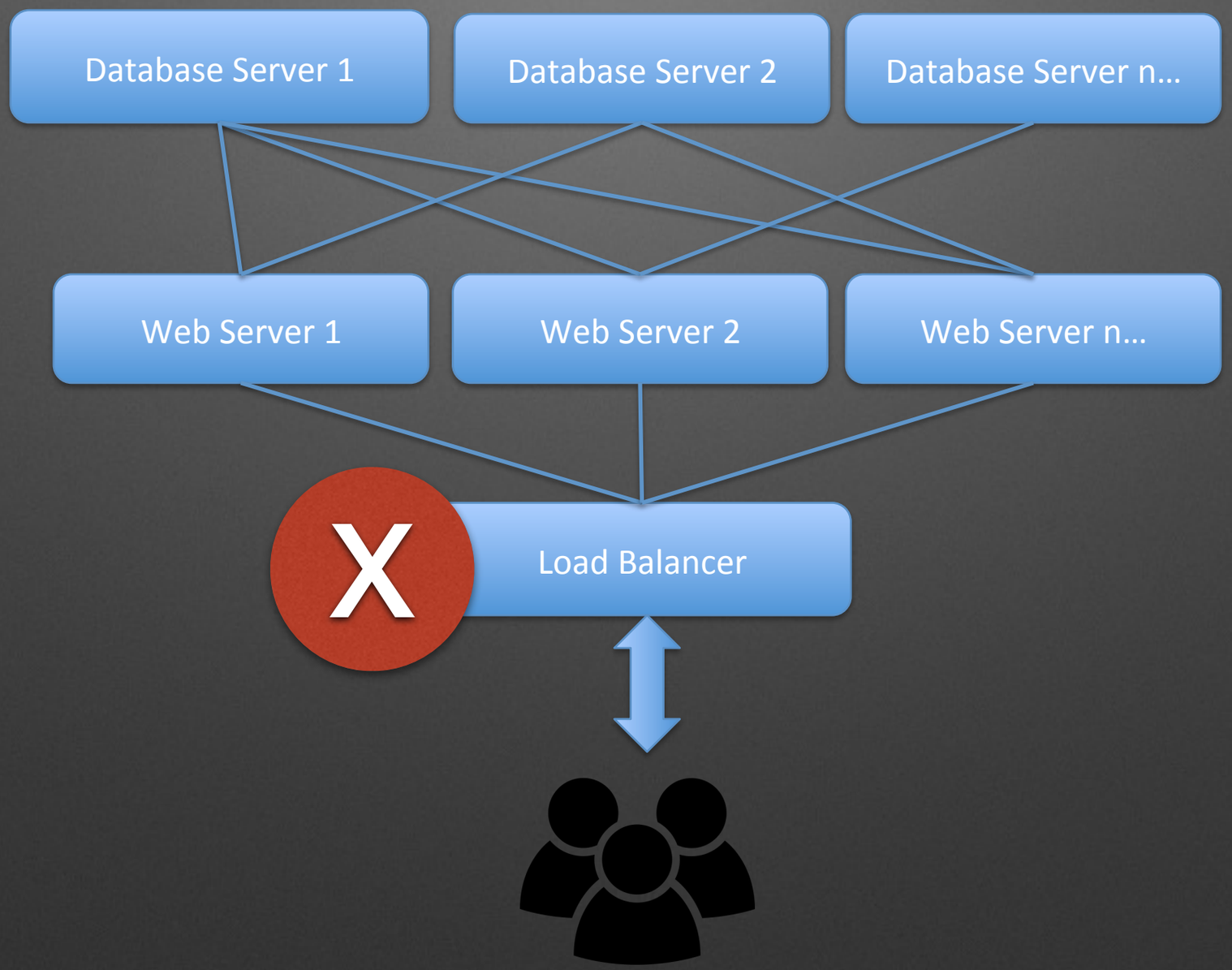


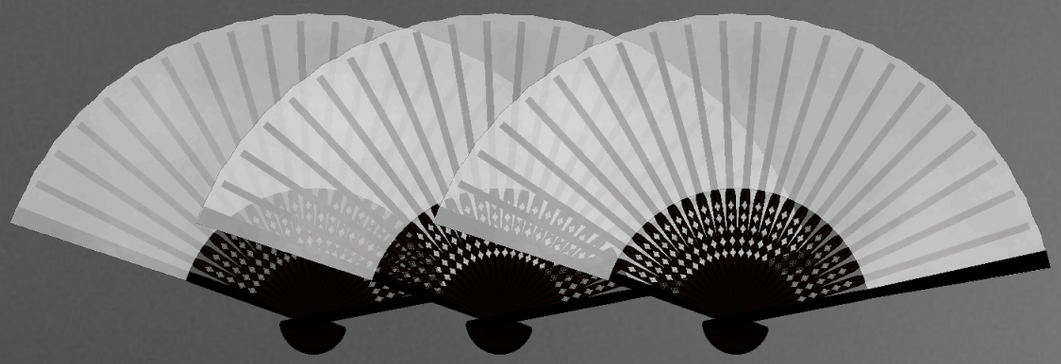
Diamond


X
Database Server 1
X
Web Server 1
Web Server 2
Web Server n…
X
Load Balancer

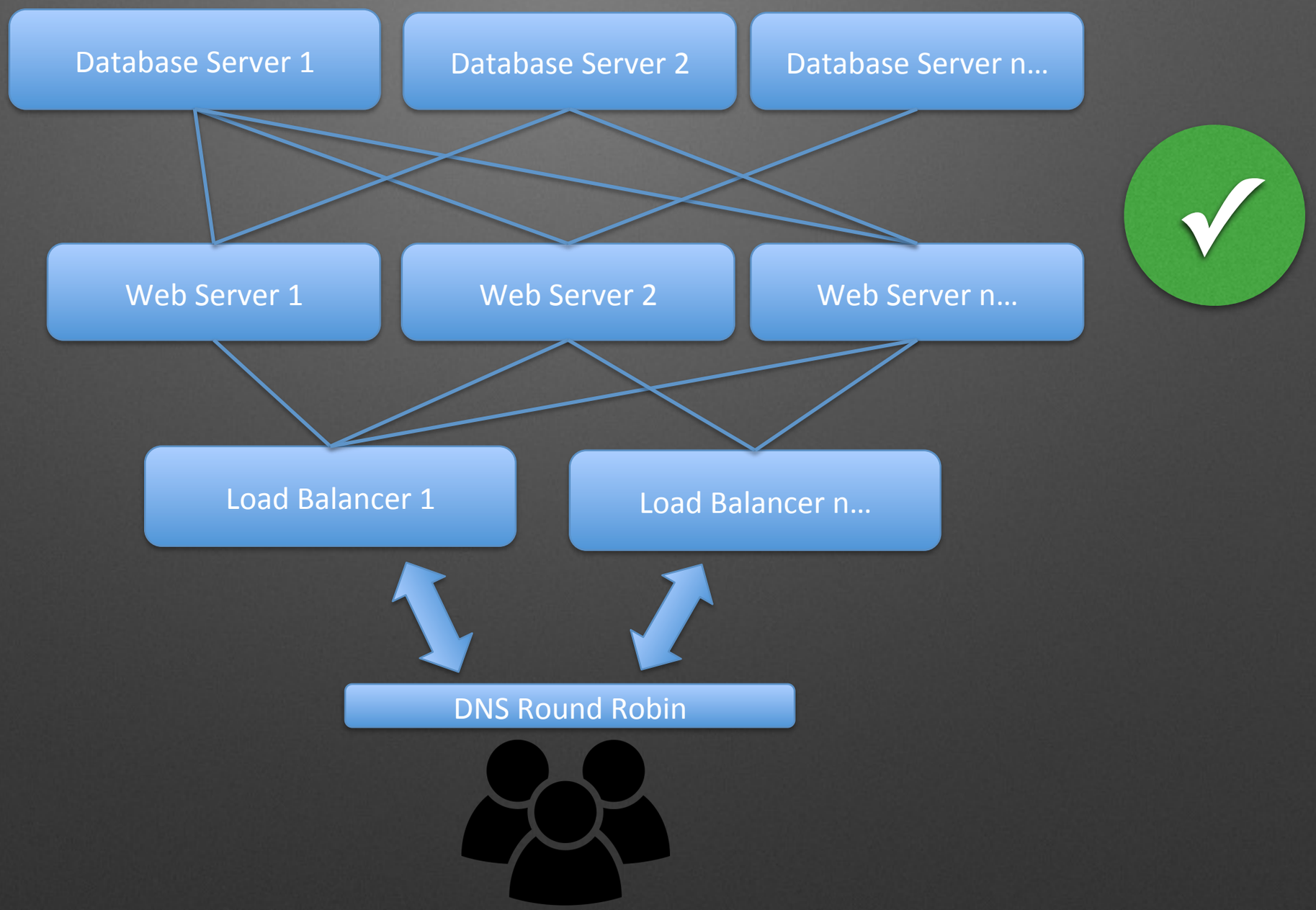


Fan-out


Database Server 1
Database Server 2
Database Server n…
Web Server 1
Web Server 2
Web Server n…
X
Load Balancer



Multi-Fans

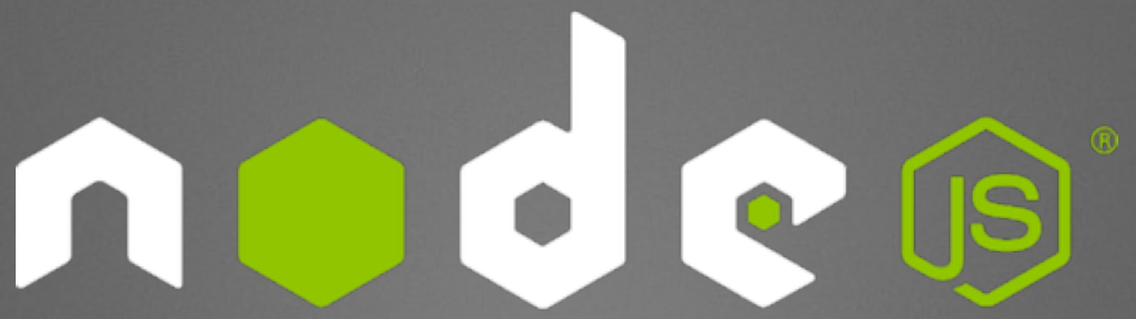

Database Server 1
Database Server 2
Database Server n…
Web Server 1
Web Server 2
Web Server n…
Load Balancer 1
Load Balancer n…
DNS Round Robin

Why is this Applicable Now?

There was no cloud servers, there was no ability to spin off instances fast!

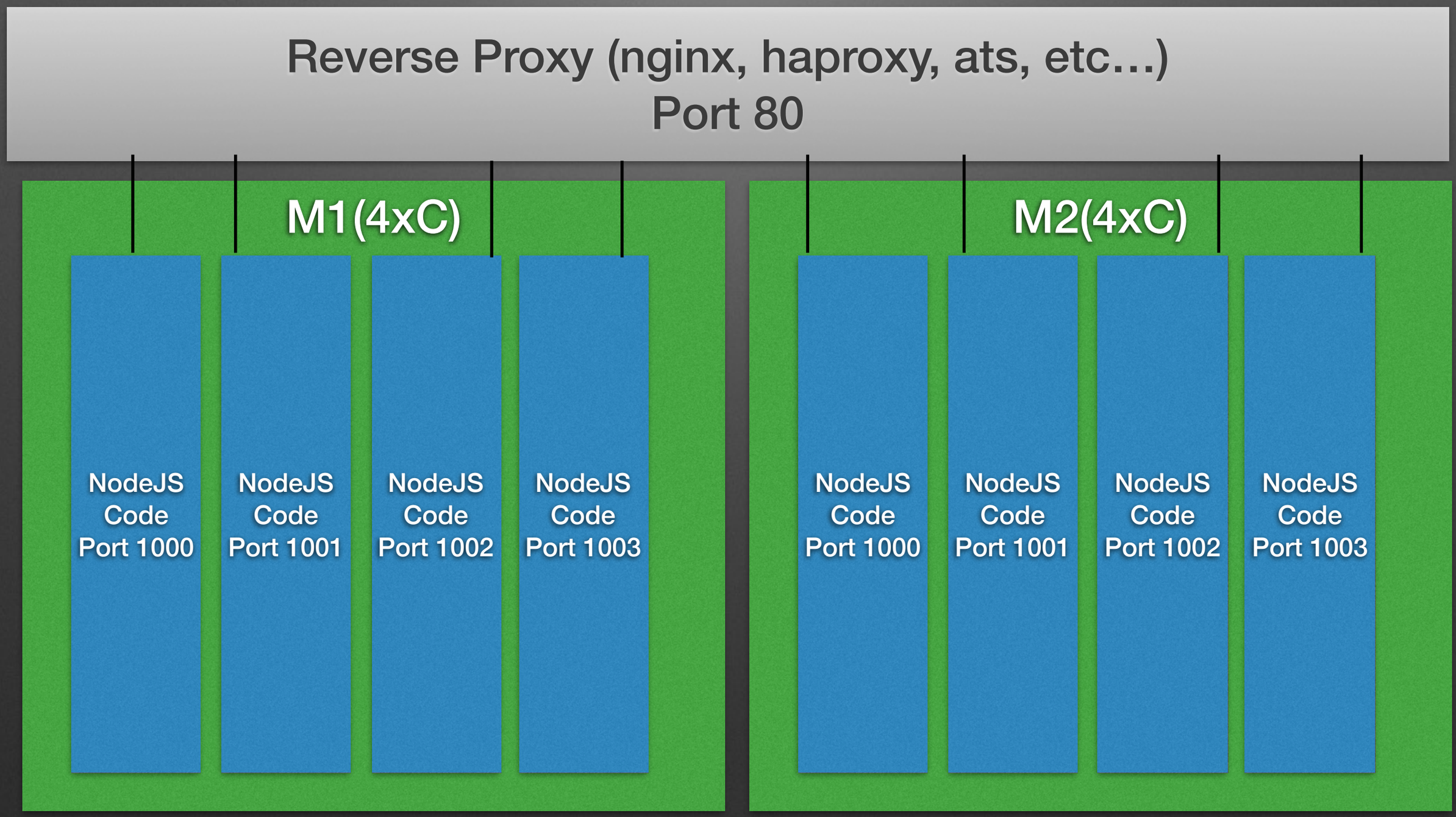
Q/A - 1

WEB/API Application Server



- ASYNC
  - Non-Blocking I/O
  - ASYNC Bindings
- Single Threaded Servers
- Cluster Servers
- FAST !
- Best Used as Server Glue

Scaling Using NodeJS



Q/A - 2

Data Storage (Reads and Writes)

Availability

Always Available for Reads and Writes

A

CP ->QUEUE-> CA

(For High Write Volume and Speed of **Writes**)

CA ->INDEX-> CP

(For Faster **Reads**)

CP ->INDEX-> CP

(For Faster **Reads**)

CA

MYSQL
Postgres
LevelDB

AP

Kasandra
CouchDB
DynamoDB
SimpleDB

C

P

Consistency

All Clients always have Same Data

CP

MongoDB
Redis
SOLR

Partition Tolerance

Works Well over Net. Partitions

Q/A - 3

Overall Q/A