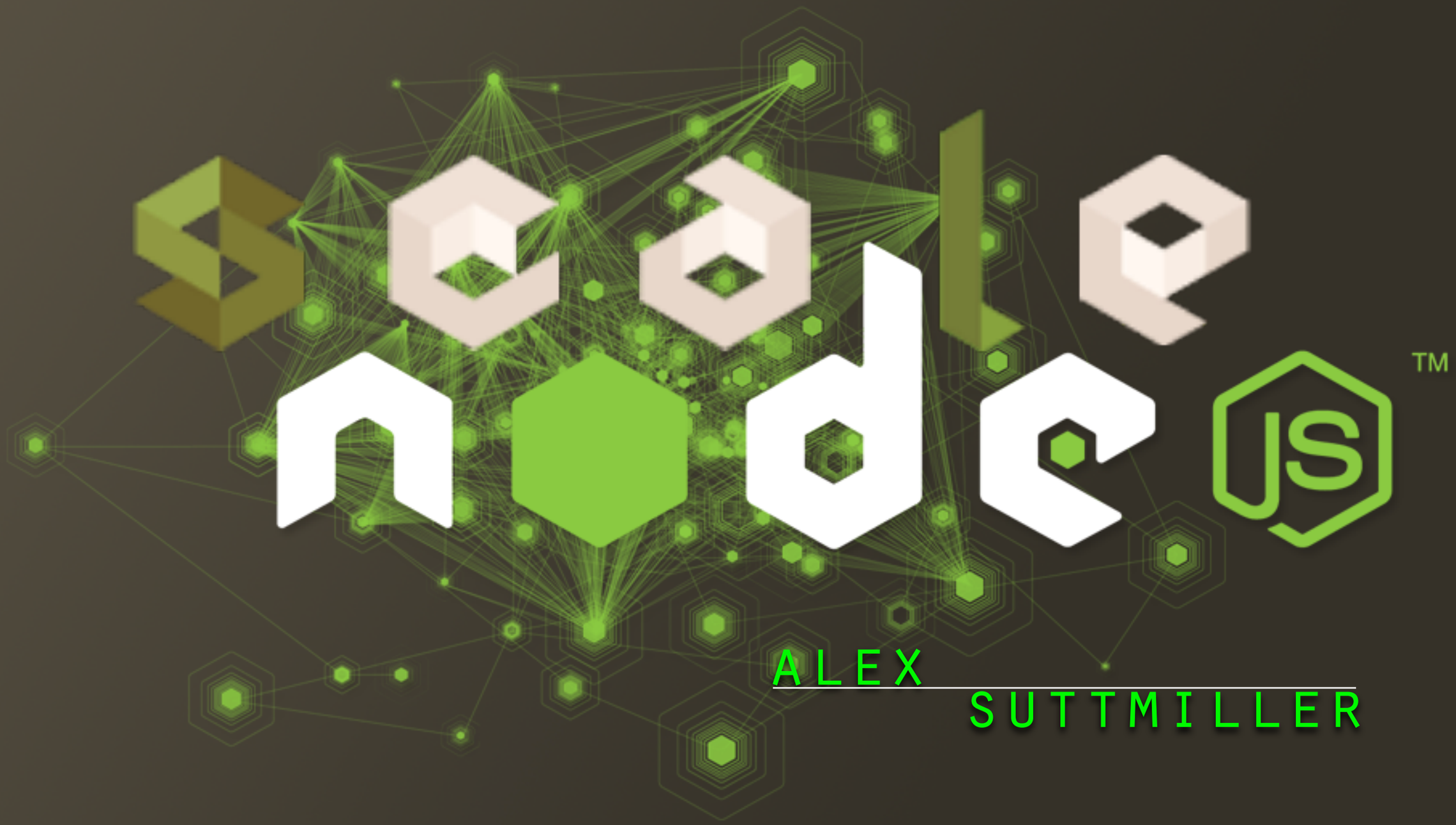


ALEX SUTTMILLER

About me

- Contributor to the MEANJS Google group
- Spoken previously at COCCUG on Azure HDInsight
- Speaking at CloudDevelop, October 17th
- Follow me on twitter @asuttmiller
- Very interested in building up the tech community in Columbus

NodeJS in the Press

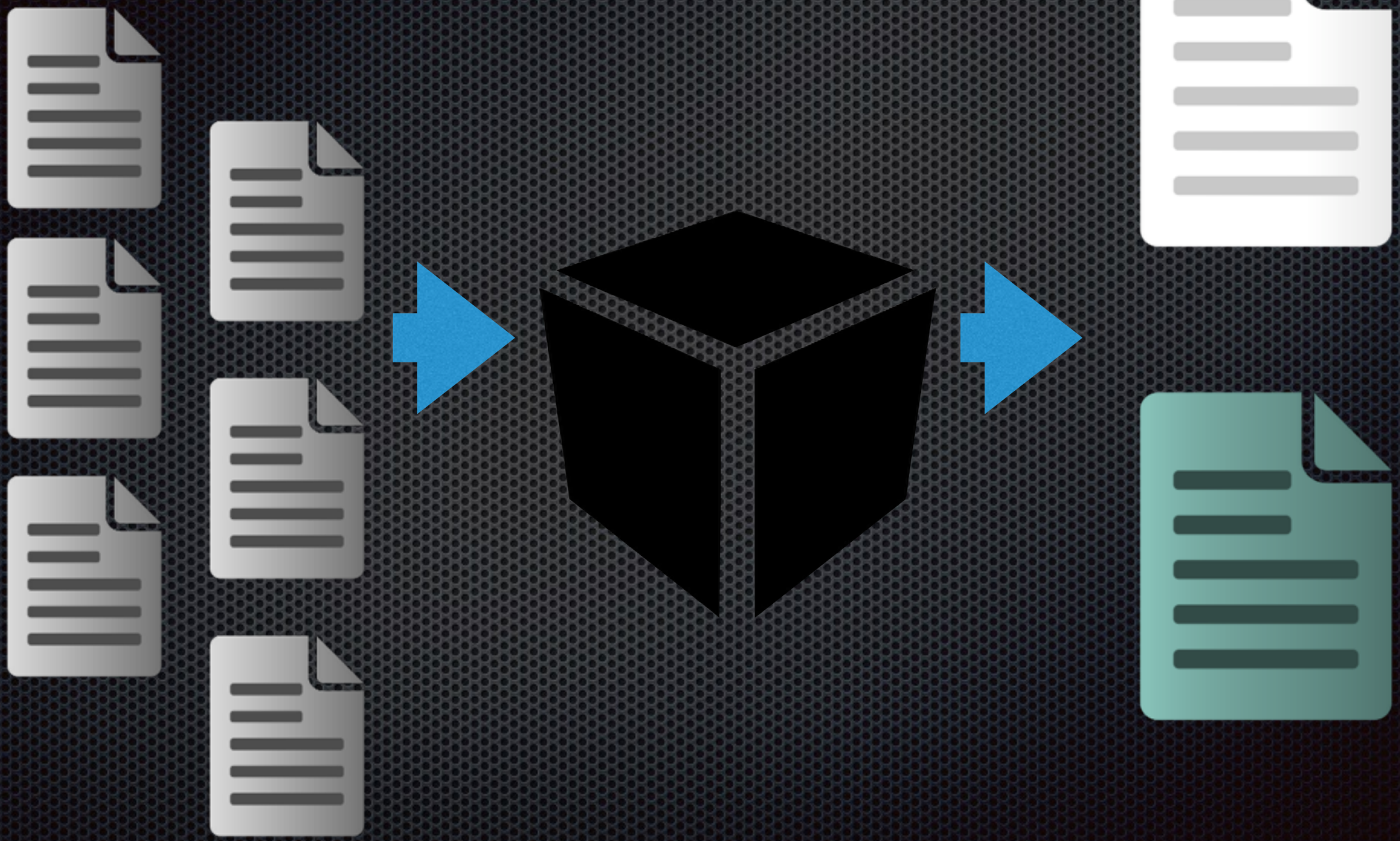
- Increasing demand
  - Job Listings are increasing
- Legitimate adoption at Fortune 500 companies
  - Eran Hammer @ Walmart
- Very rich ecosystem; very active community
  - http://nodejs.org/community/

PayPal success story

- Fast iterations
- Less developers; Less code; Less overhead
- 30% reduction in page render time
- Replaced JAVA UI technology stack
- Kraken.js, SecureExpress.js, Babel.js

This talk brought to you by…

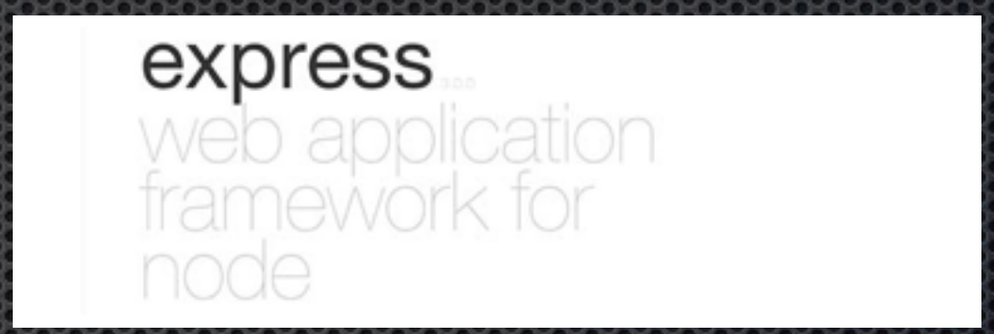
also in part by…



Our Application

Create a web application that scales on demand to meet traffic levels. Use message passing protocols to coordinate tasks across the site.

Technology Stack



and others…

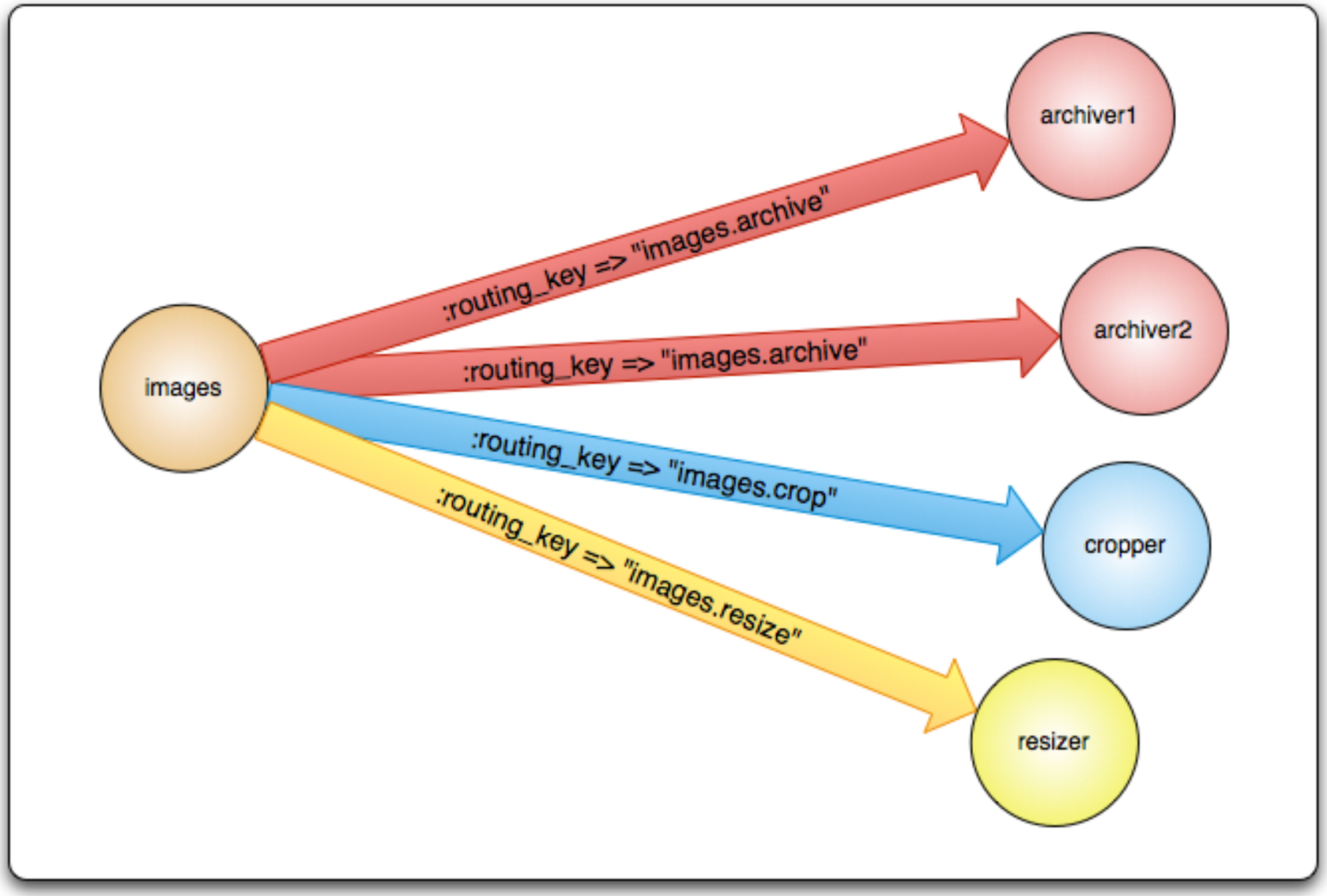
Separation of Concerns

- Platform
- Request Routing
- Persistence
- Scale
- Server Synchronization
- Presentation

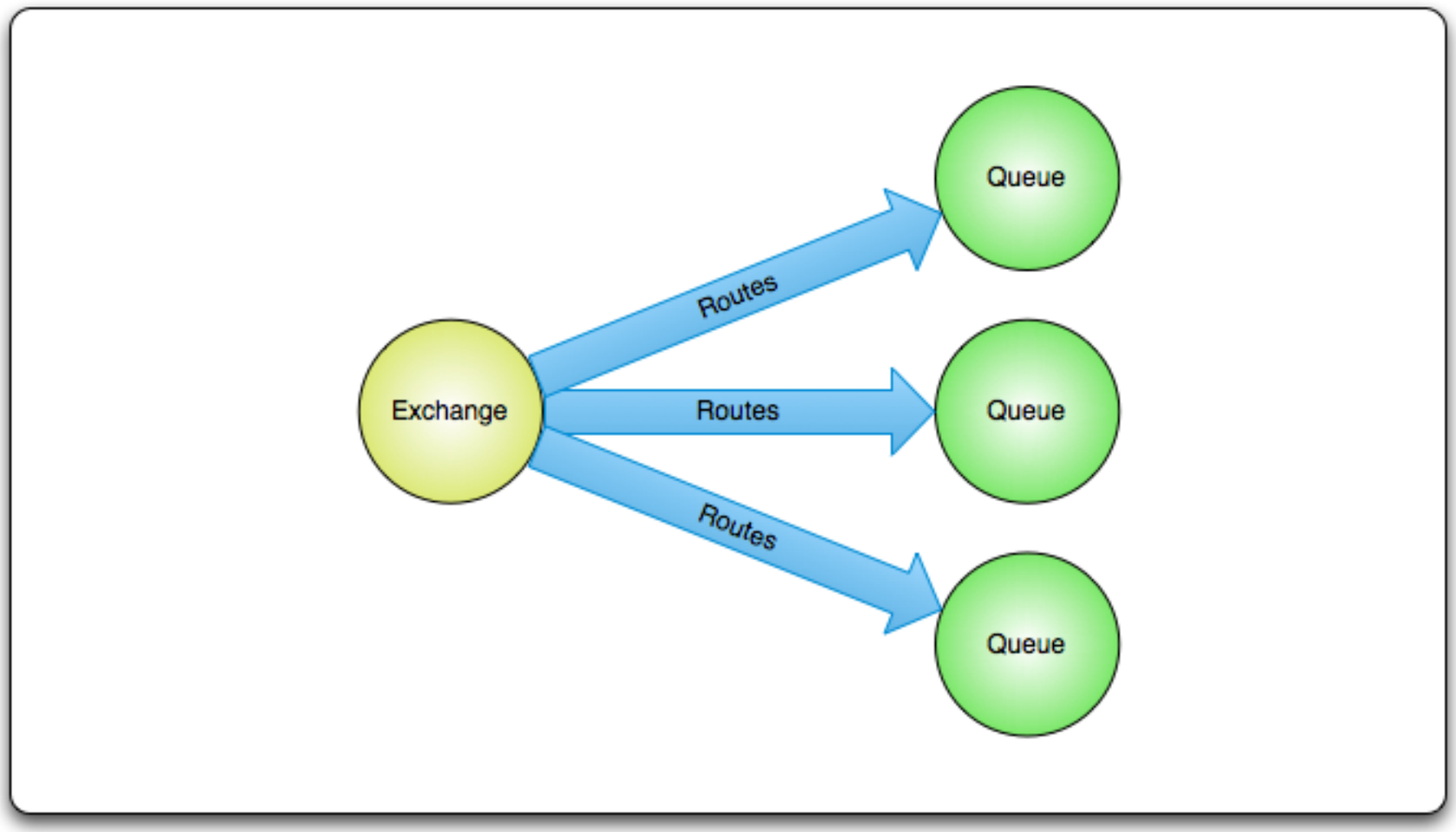
Message Routing (AMQP)

- Publisher(s)
- Subscriber(s)
- Queues
  - Durable
  - Arguments
  - others…
- Exchanges
  - Direct
  - Fanout
  - others…
- Bindings
- Routes & Channels

Direct exchange routing



Fanout exchange routing



Reminder…

- Platform
- Request Routing
- Persistence
- Scale
- Server Synchronization
- Presentation

Legend

Dyno

Message Bus (Rabbit MQ Exchange)

Database Connection

Producer Queue

Consumer Queue

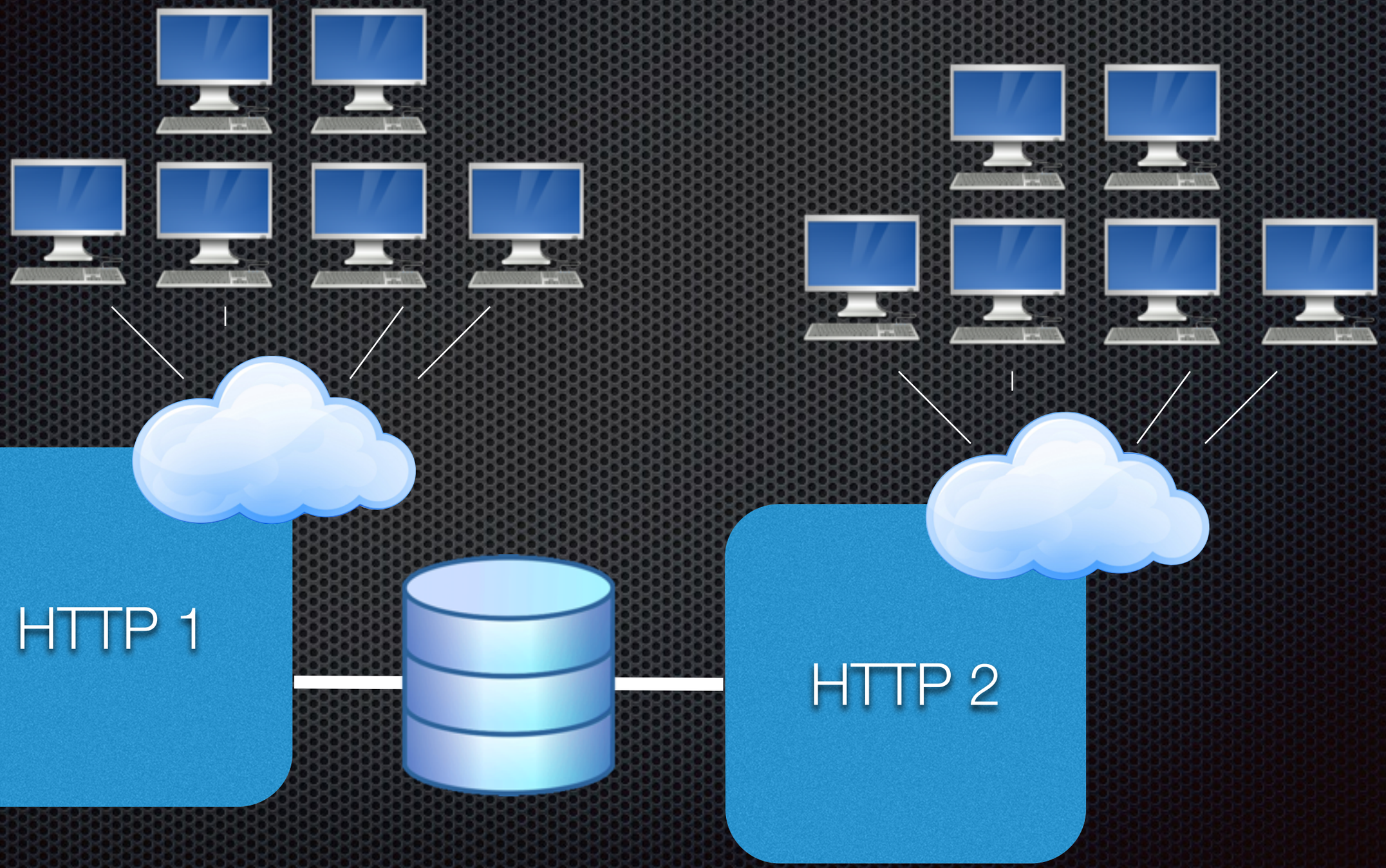
w.r.t. a node instance

Process

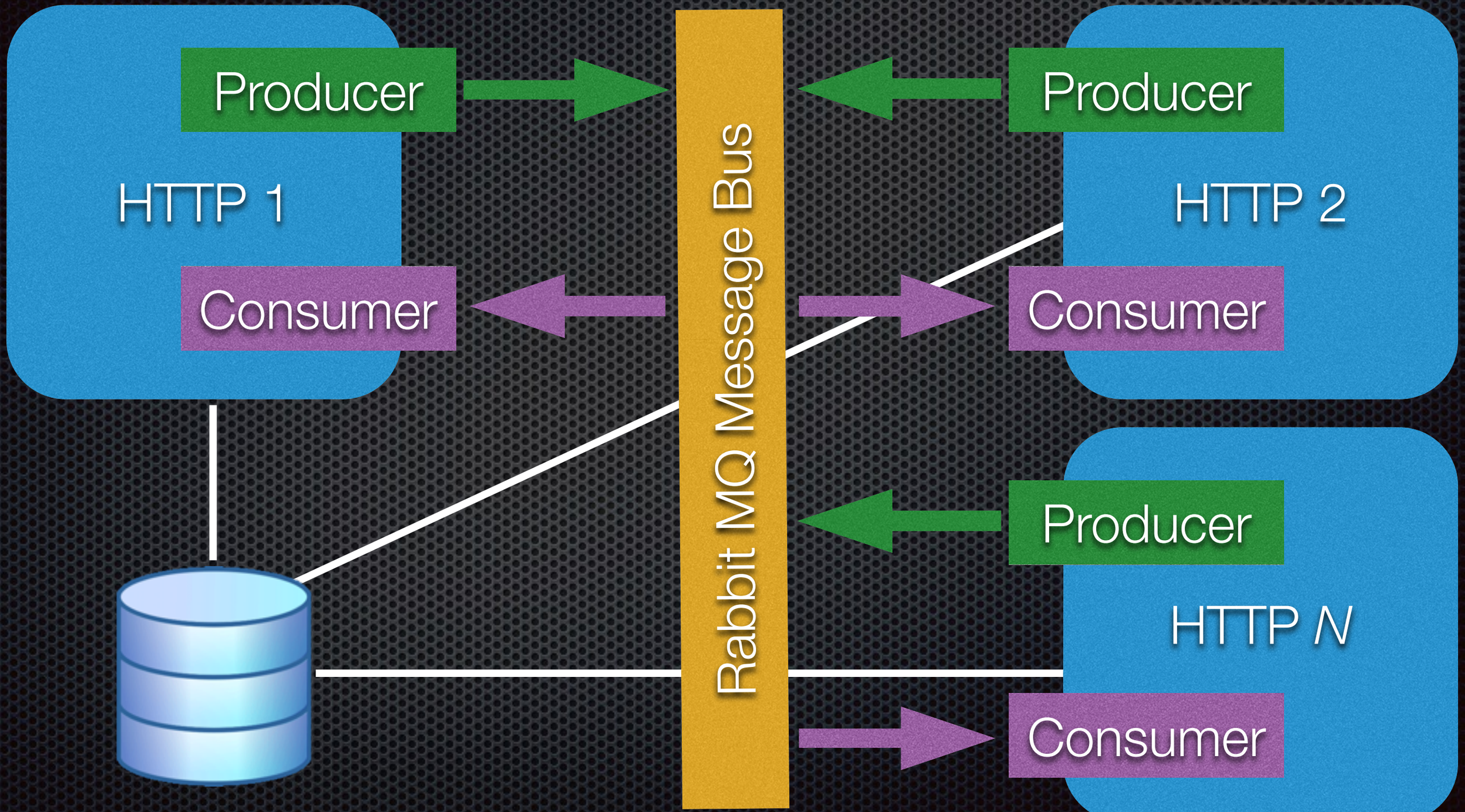
Magic

Twitter, etc…

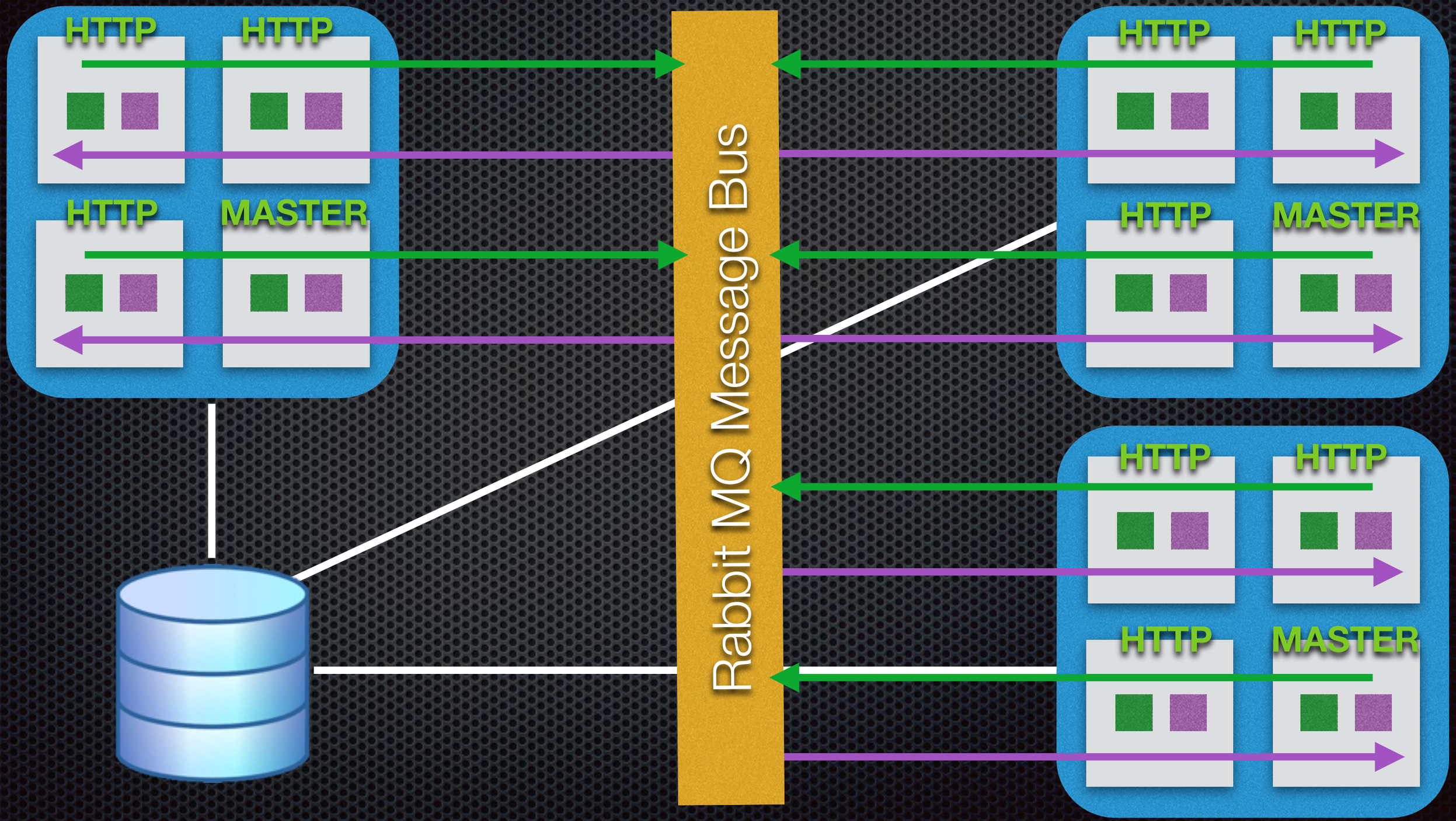
Architecture #0



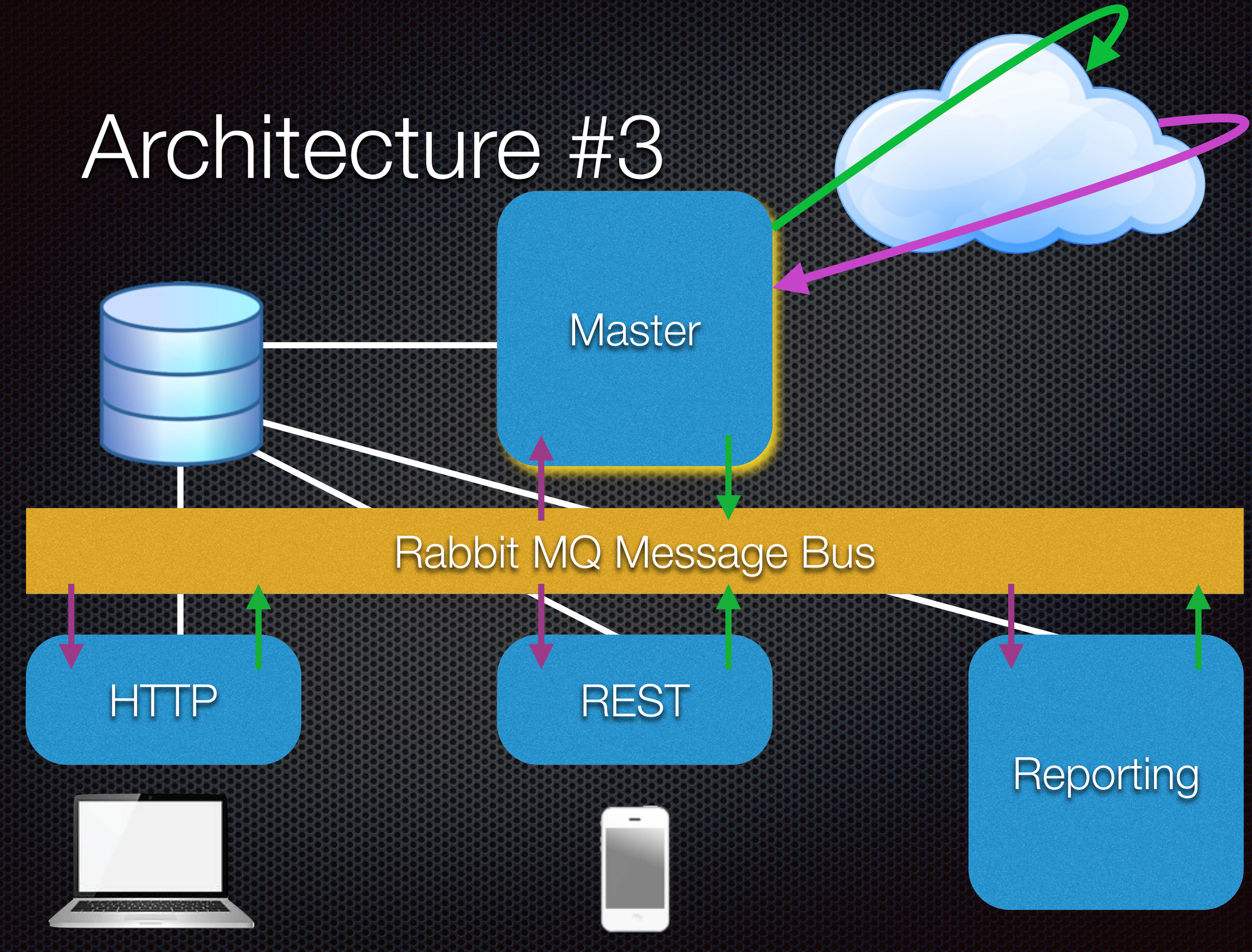
Architecture #1



Architecture #2



Architecture #3



Internet of Things

- Nothing here is NodeJS specific
- Network all your services via message delivery buses
- Leading vendors are working extensively in this arena

Robust Scaling

- Separate your concerns
- Choose the technologies
- Define service levels
- Persistence & Recovery
- Assume everything will fail

http://en.wikipedia.org/wiki/Emperor_tamarin

- Chaos Monkey: https://github.com/Netflix/SimianArmy

Reminder…

- Platform
- Request Routing
- Persistence
- Scale
- Server Synchronization
- Presentation

Robust Scaling

- Separate your concerns
- Choose the technologies
- Define service levels
- Tip: Everything fails

http://en.wikipedia.org/wiki/Emperor_tamarin

- Chaos Monkey: https://github.com/Netflix/SimianArmy

Specific Concerns

- Memory Footprint
- Dead Letter Queues
- Queue Durability
- Database Availability & Database Consistency
- Master Resilience
- Worker Resilience
- Data Security
- Maybe do not use the message bus for IPC (save $)
- Disconnect from bus if no connections

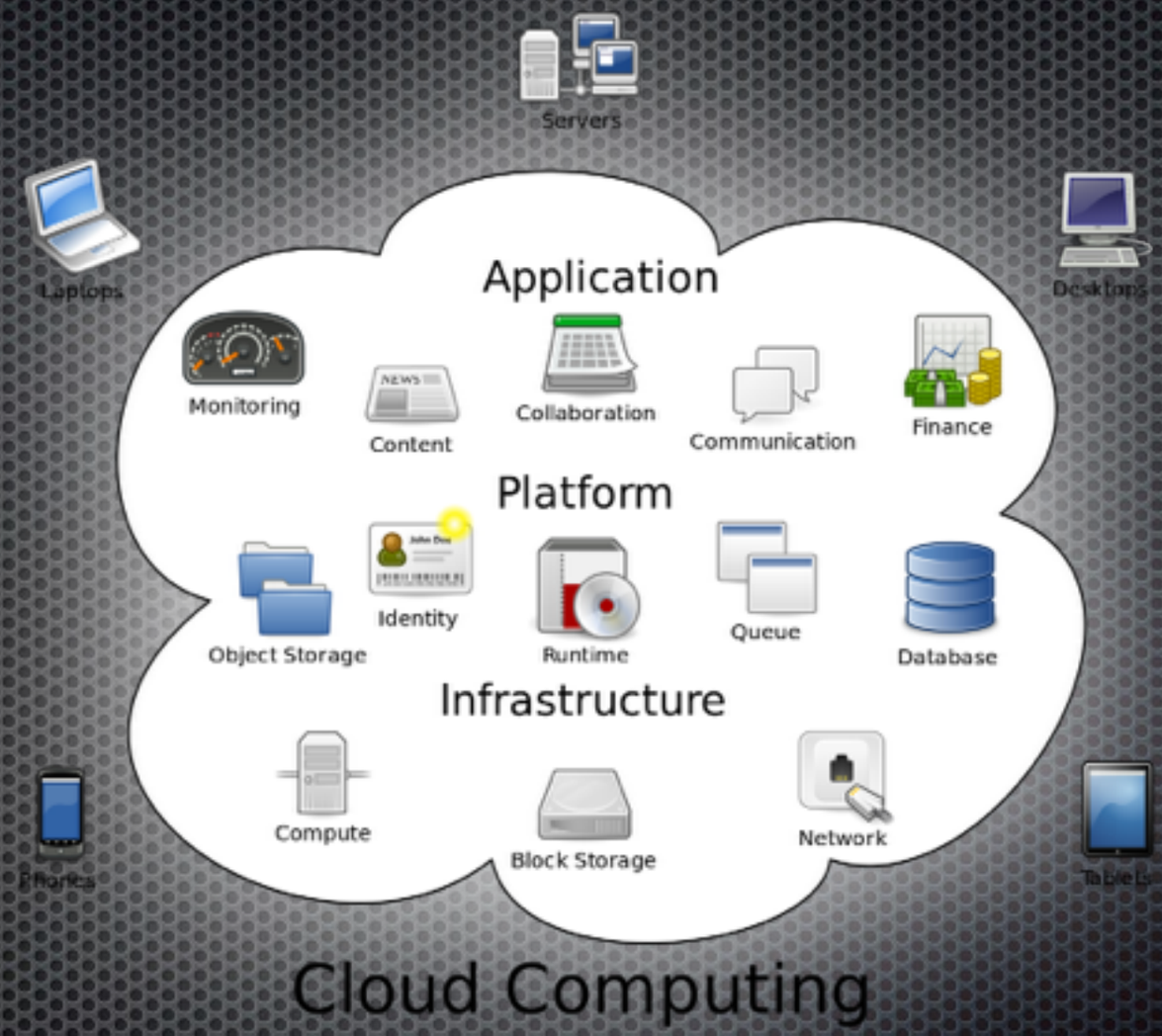
Other Message Buses

Make sure any option you consider meets all requirements, including durability requirements

- Redis has the ability to publish/subscribe message
- socket.io has an adapter that can be configured for use with Redis
- Azure Service Bus

Hosting

- Heroku
- OpenShift
- Nodejitsu
- Microsoft Azure
- Amazon
- AppFog
- Tons more… except Google. Which is strange; totally, completely, inexplicably strange.



Code

- Spike a basic NodeJS MVC application
  A plain old single core NodeJS app server
- Scale to multiple cores
  Use cluster library to spin up more instances of the NodeJS server to use available resources
- Scale to multiple nodes
  Use AMQPlib and RabbitMQ to establish a communication bus
- Show a full stack web server with changes

So is that all?

Now that you have the ability to coordinate messages, you can begin to implement distributed algorithms such as:

- Vector Clocks (check pointing / rollback)
- Quorum Protocols (eventual consistency / mutex)
- Leader Election (durability and recovery)
- Search

Code

- server.js
- ./config/settings.js
- ./config/express.js
- ./management/
- ./app/settings.js
- ./app/base/*

github.com/NeverOddOrEven

“He who asks a question is a fool for five minutes;
he who does not ask a question remains a fool
forever.”

–Chinese proverb