Believe in Erlang in Games!!!

Noah Gift
Mario Izquierdo
James Mayfield

High Level Overview of Erlang in Game Backends

Overview of Our Talk: 3 for 1 Special

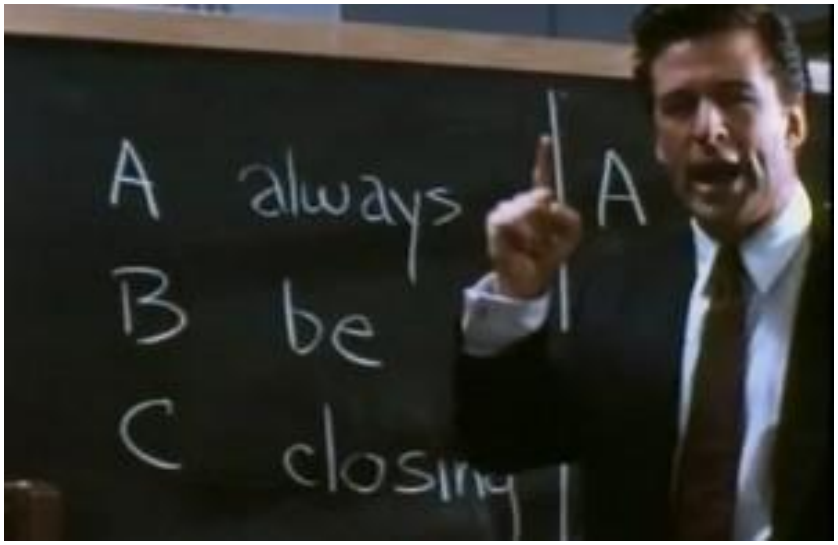
- Advocating Erlang in A Game Company: Noah Gift
- RabbitMQ in DIO: Mario Izquierdo
- Versu an AI Game Evolution: James Mayfield

You Can Get Fired For Using Erlang

- Erlang is "functional" and this is "complicated" for "regular" programmers
- It has been around for too long
- It is not Java, or Python or Ruby or C++ or "X".
- Your boss is an idiot who can't code like this guy???

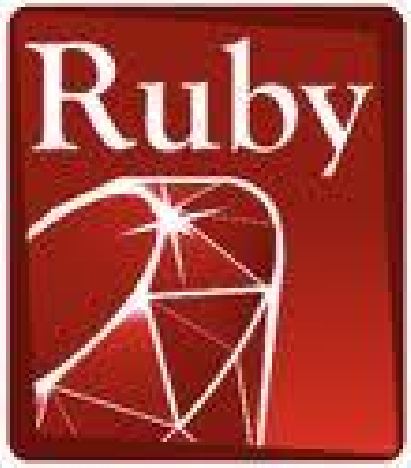
Always Be Delivering

- On day 1 at work don't suggest rewriting the billing systems in Haskell
- **Show up early, stay late, underpromise and overdeliver**. Rinse and Repeat * N
- Then suggest Erlang on a small project...



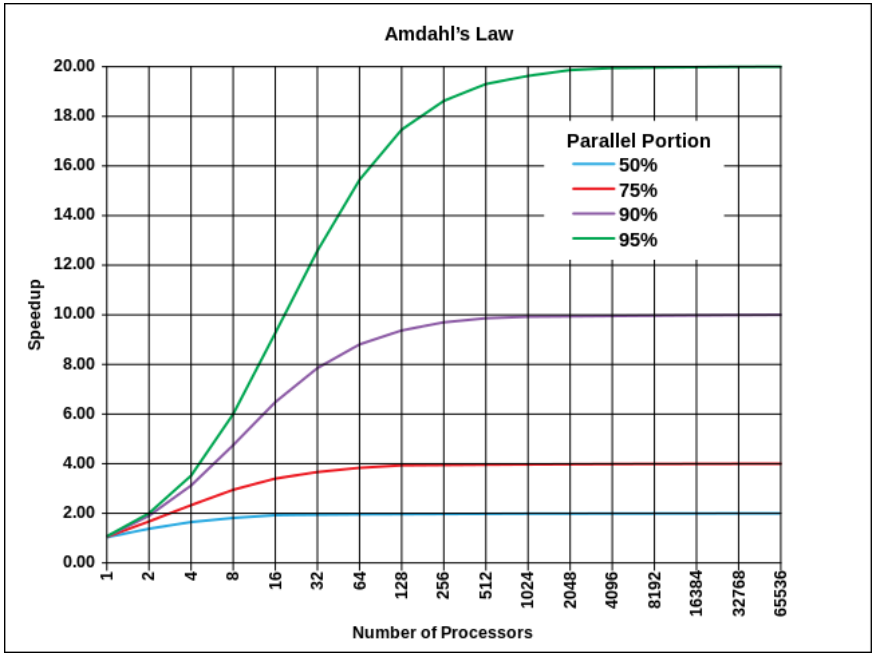
Ubiquitous Multiprocessing in Erlang: Not TBD...

- PyCon 2013 Keynote: Async Multiprocessing...TBD
- Event Loops in Ruby, Node and Python can be simple at first, then exponentially difficult
- 



More on Event Loops in Scripting Languages

- What do you do when your one processor gets hot and starts to consume a lot of CPU?
- Start to write Erlang in a scripting language on a tight deadline?
- Amdah's Law (sucks)
- Energy Efficiency



More on Event Loops in Scripting Languages

- Maybe you simply use RabbitMQ to dance around it...notice how the "X" language's largest website always has a **secret ingredient**...(hint not their language)
- Erlang (RabbitMQ): ***script language crack***.

RabbitMQ in dio

Mario Izquierdo

Erlang can help you even if you don't know erlang!

What is RabbitMQ?

- Robust **messaging** for applications
- **Easy to use**
- Runs on all major **operating systems**
- Supports a huge number of **developer platforms**
- **Open source** and **commercially supported**

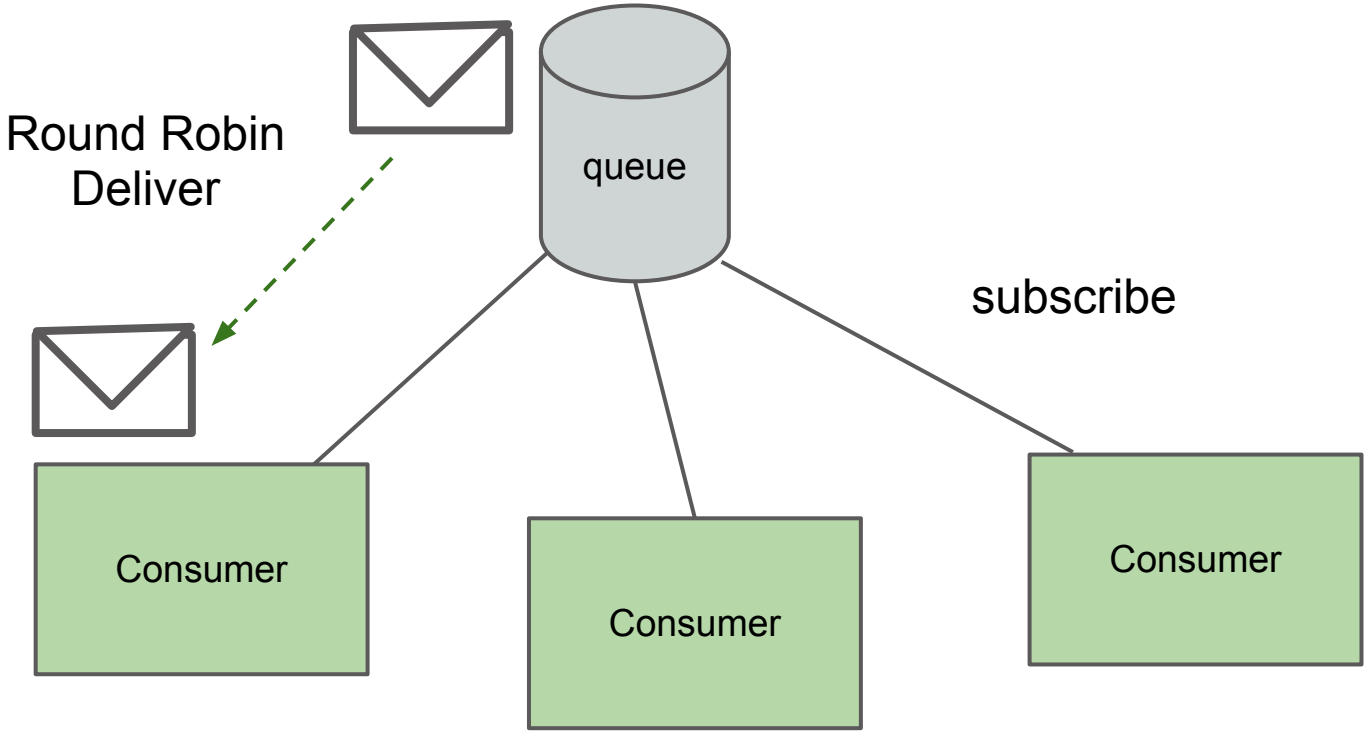
What is RabbitMQ?

- A service for messaging that implements the **AMQP protocol**
- **Erlang** nodes running Erlang applications (so it must be good)

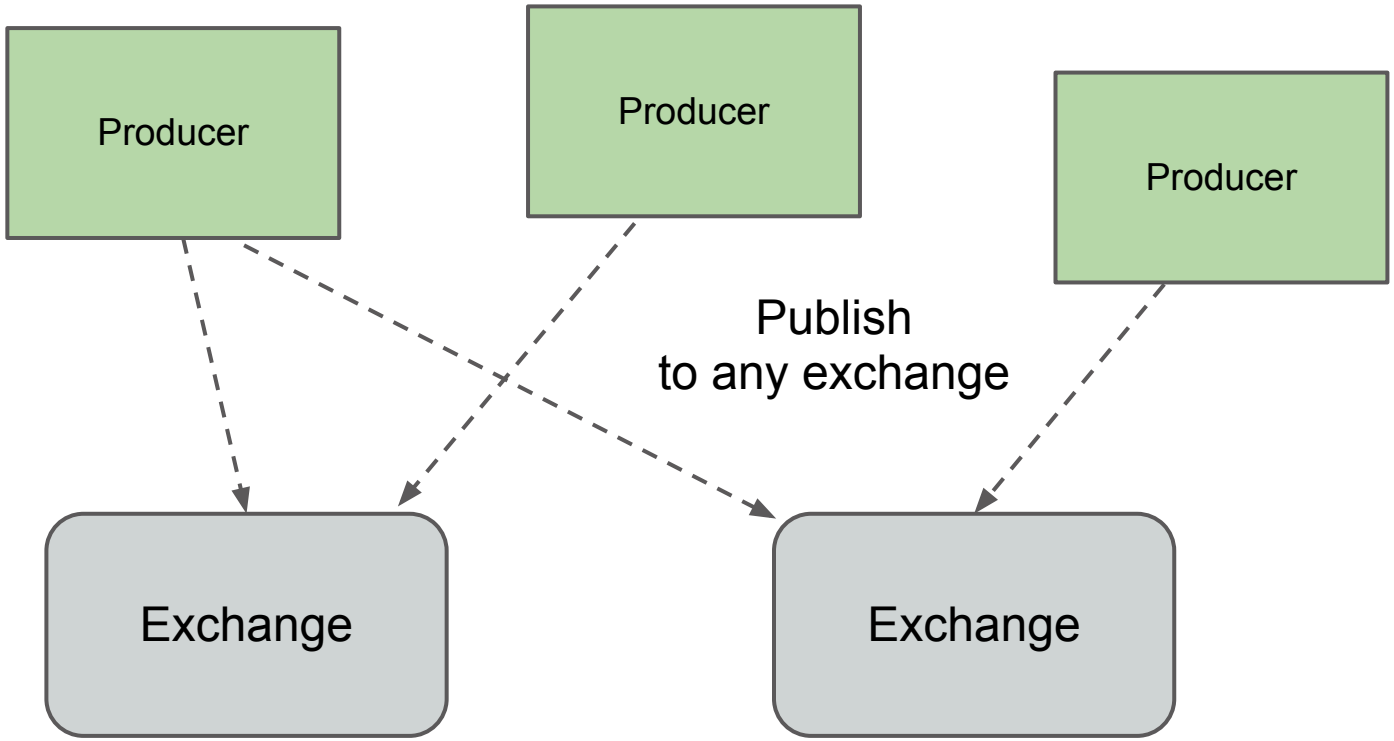
Think of it as a **service** (like MySQL), where you can:

- Open a connection
- Define **Exchanges**, **Queues** and **Bindings** (instead of tables and indexes)
- Send and Receive **Messages** (instead of reading/adding records)

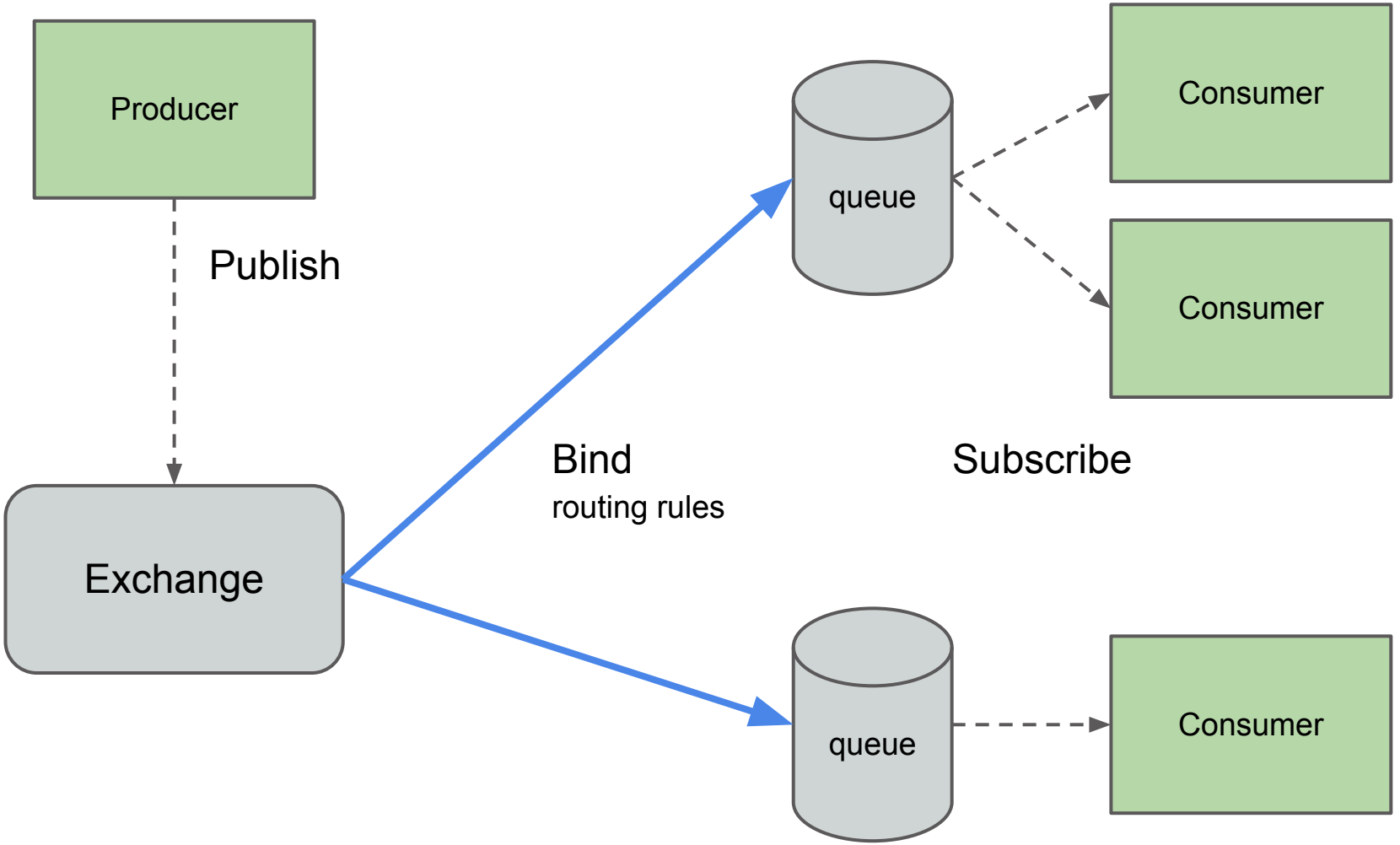
Queues and Consumers



Exchanges and Producers



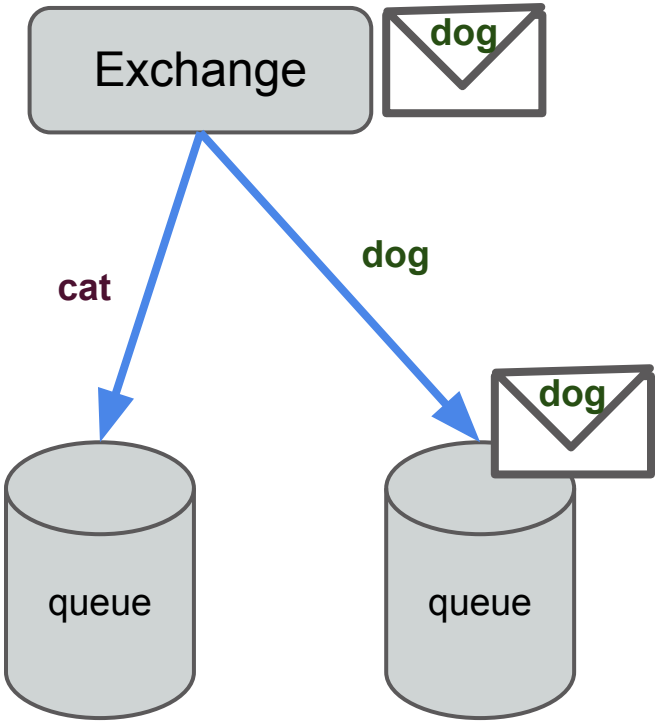
Bindings and Routing



Types of Exchanges and bindings

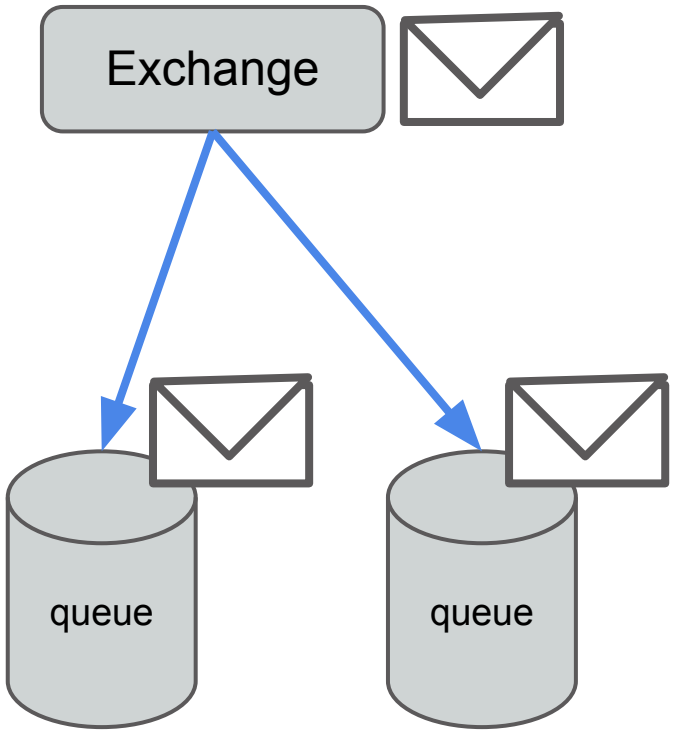
Direct

uses routing_key



Fanout

broadcasts to all



Typical usage of RabbitMQ

- Connect with RabbitMQ
- Define an exchange
  - Start publishing messages
- Define a queue and its bindings
  - Subscribe to the queue and listen for incoming messages
- Add more consumers (workers) to the same queue for load balancing
- Connect applications written in different languages
- "Send and forget", i.e. logging

Real usage example: dio.com

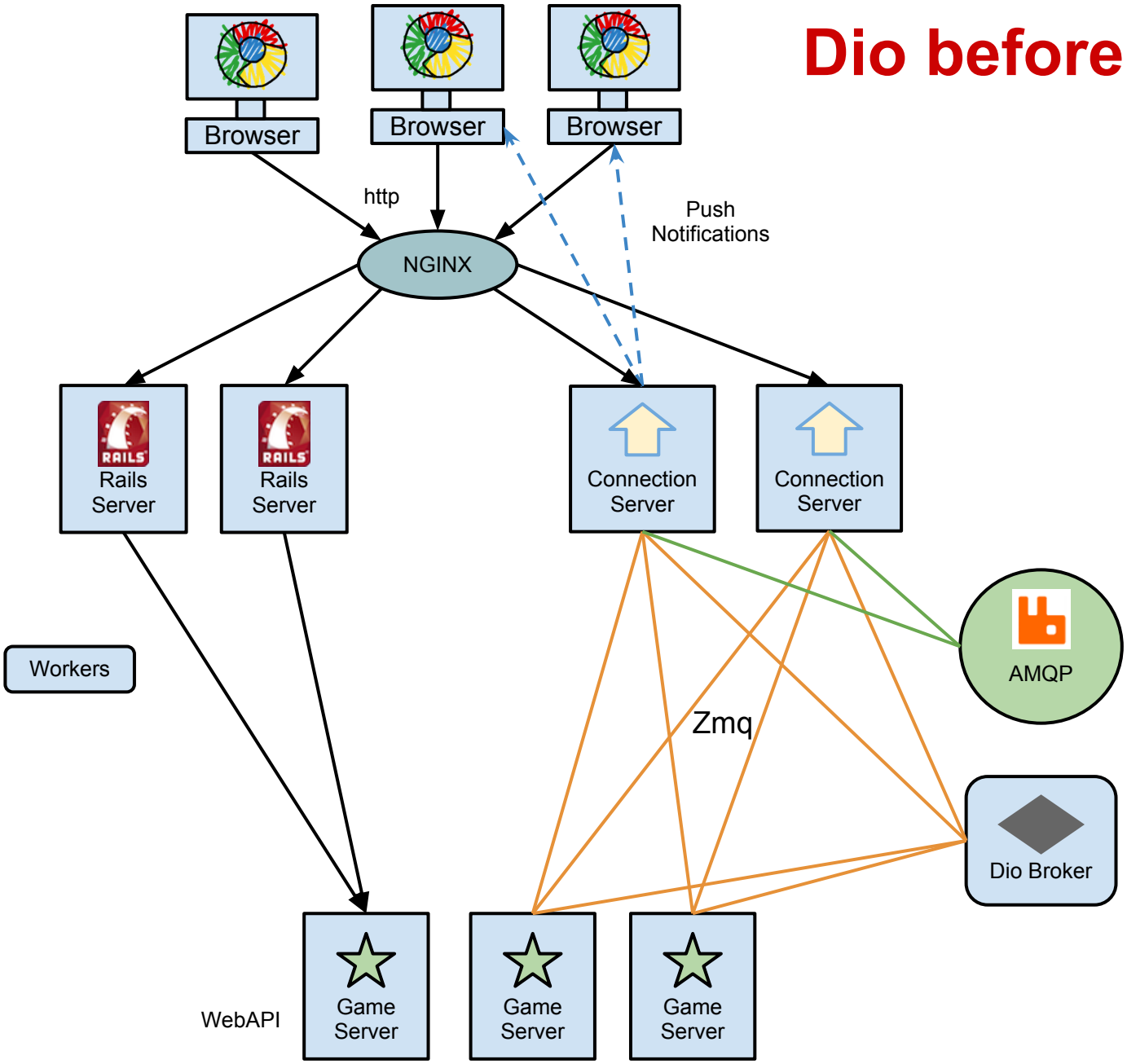
dio is a web app built on ruby, optimized to support multiplayer experiences in real time.

A place (world) instance runs in memory on one of the available GameServers.

Dio Architecture

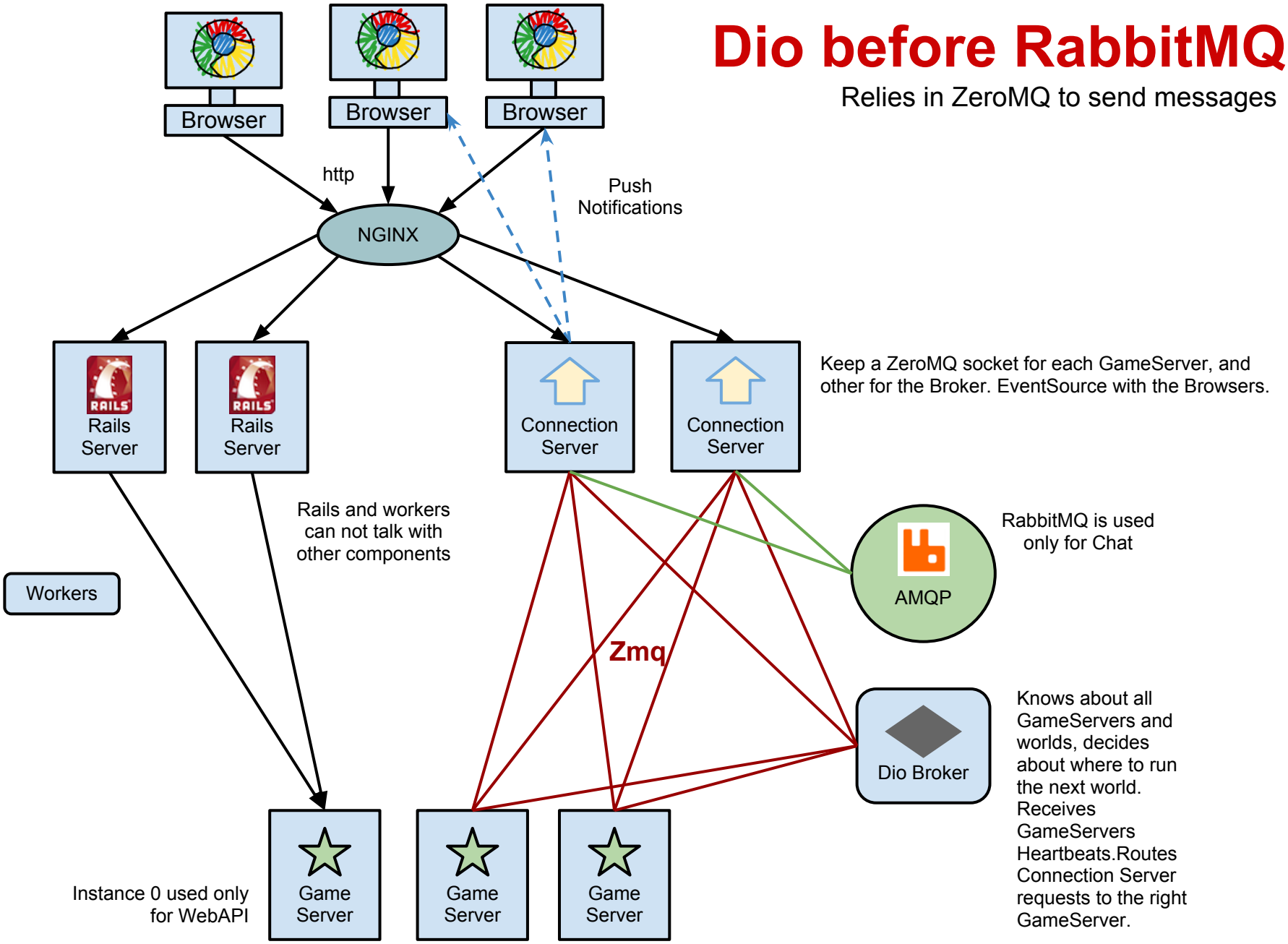
- Browsers send HTTP requests to **Rails** and Connection Servers
- **Connection Servers** use HTML5 EventSource for push notifications
- Connection Servers send in-game messages to world instances that are running in Game Servers
- Connection and **Game servers** need to talk to each other constantly, and they need to scale

Dio before RabbitMQ



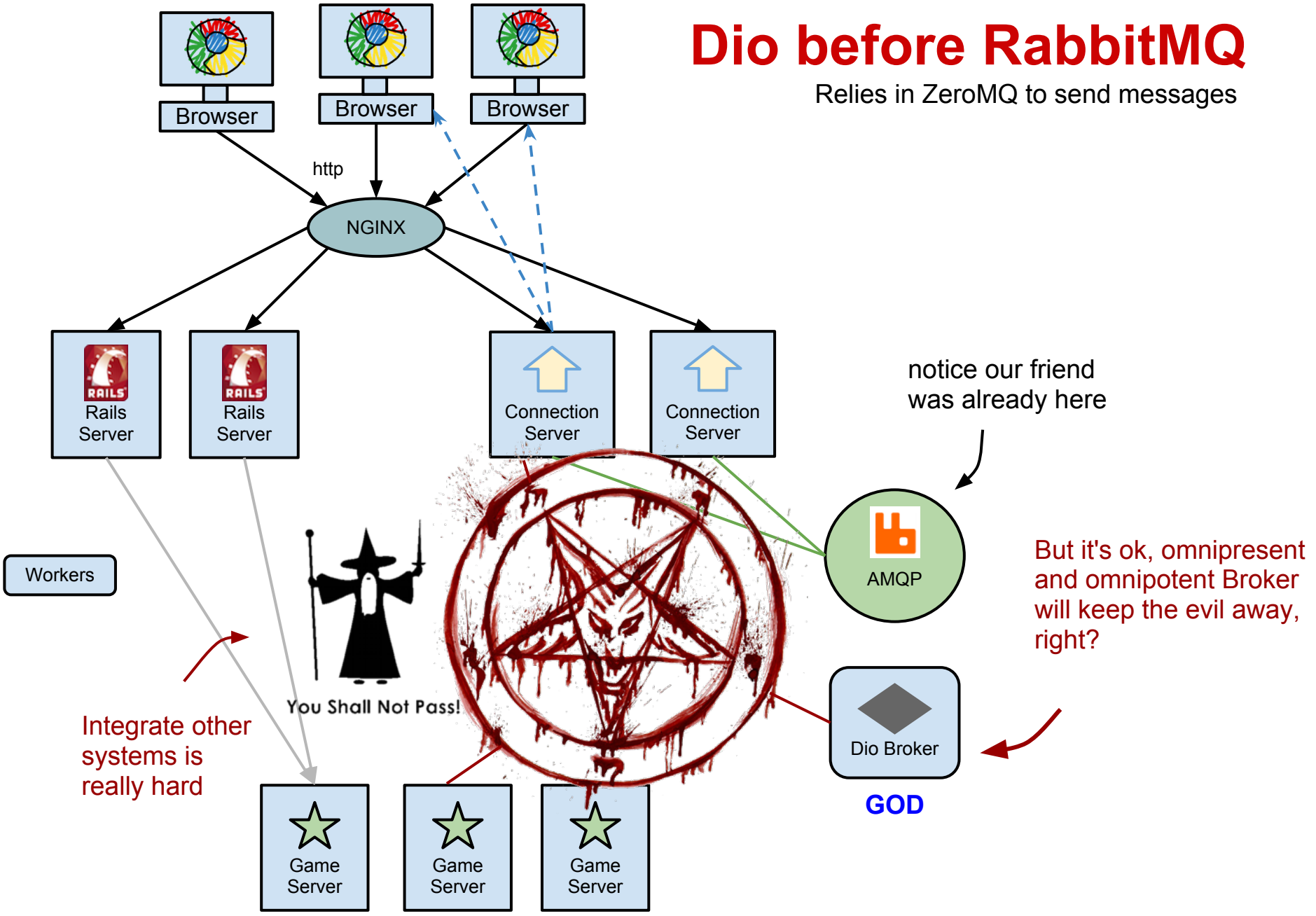
Dio before RabbitMQ

Relies in ZeroMQ to send messages



Dio before RabbitMQ

Relies in ZeroMQ to send messages



Pentagram Antipattern

Build a **complex system** with a lot of interdependencies and you will **cast the devil**

- Dependence on black magic
- Dependence on priests
- Multiple global-aware Gods

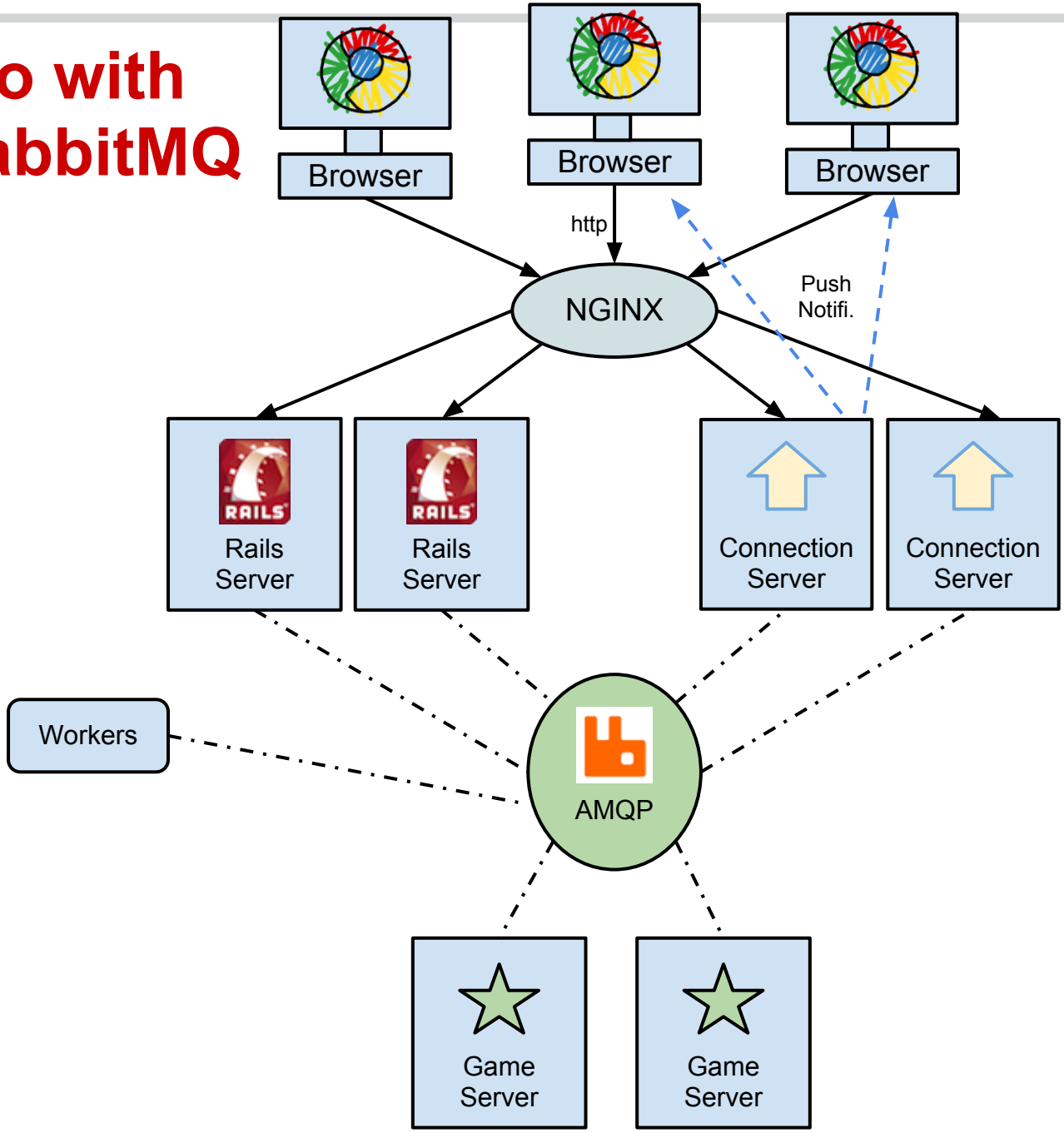
Fix it with SCIENCE!

- Study the problem
- Burn Churches
- Use the right tool for the right problem

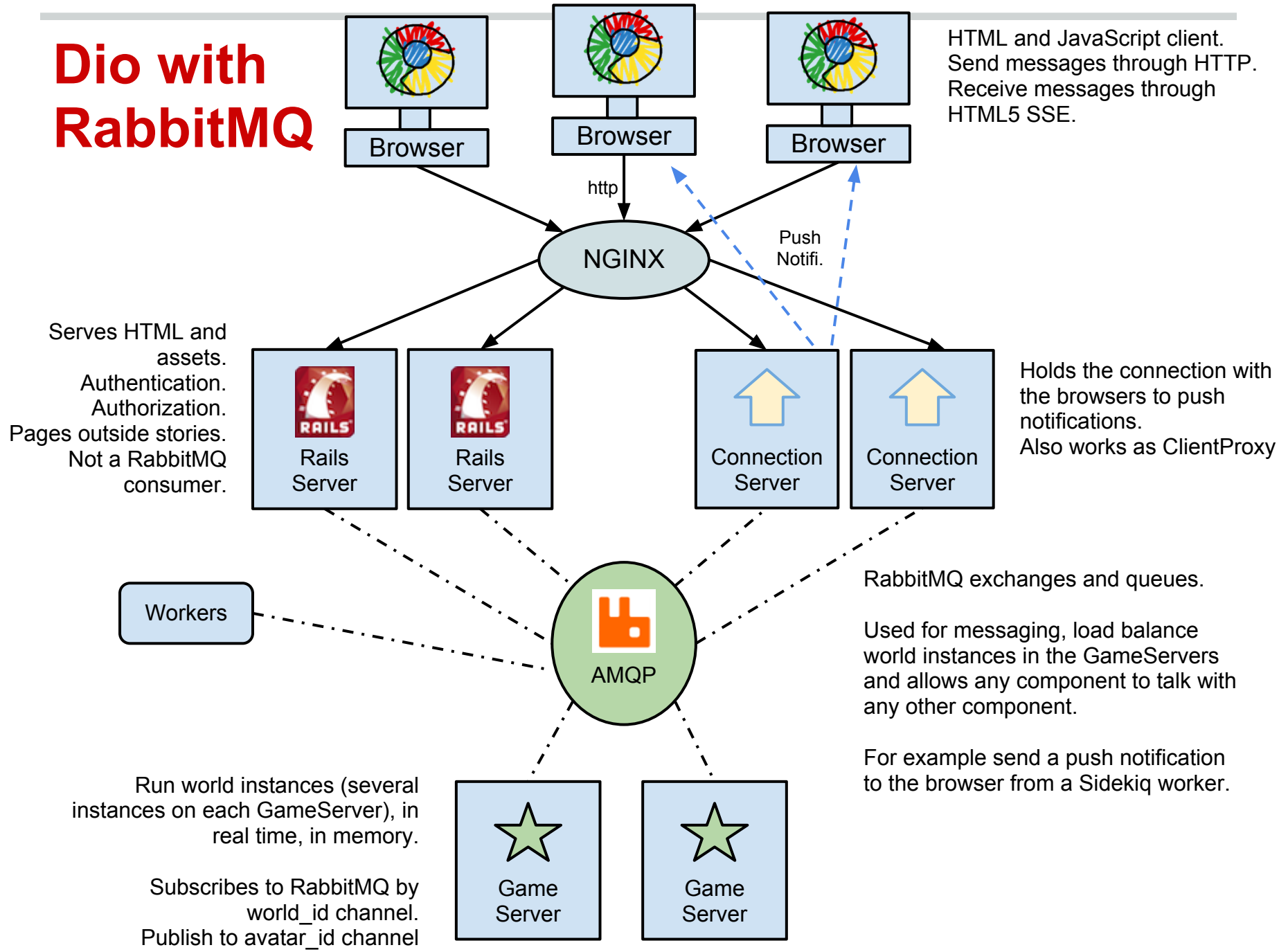
Dio before RabbitMQ: Problems

- **ZeroMQ has bad support** in ruby, the libraries are outdated.
- **ZeroMQ sockets are very low level**, actually they are a very weird mix of a low level library with black magic, they are never disconnected.
- **Too many sockets**: it needs an all-to-all socket connections between all game servers, connection servers and broker (**Pentagram**).
- **Hard to scale**
- **Limited communications**: for example if we want to send a message from Rails to a Game Server, or from a Worker to a ConnectionServer. Too hard that seems impossible, so we end up doing hacks like short polling mongodb or adding HTTP methods in the ConnectionServers (**Gandalf**).
- **Broker is too complex**. It is needed for routing and also to decide where to run new world instances. Needs to know about every GameServer and needs to be in sync, which makes it very error prone and unstable (**GOD**)

Dio with RabbitMQ



Dio with RabbitMQ



HTML and JavaScript client.
Send messages through HTTP.
Receive messages through HTML5 SSE.

Serves HTML and assets.
Authentication.
Authorization.
Pages outside stories.
Not a RabbitMQ consumer.

Holds the connection with the browsers to push notifications.
Also works as ClientProxy

RabbitMQ exchanges and queues.

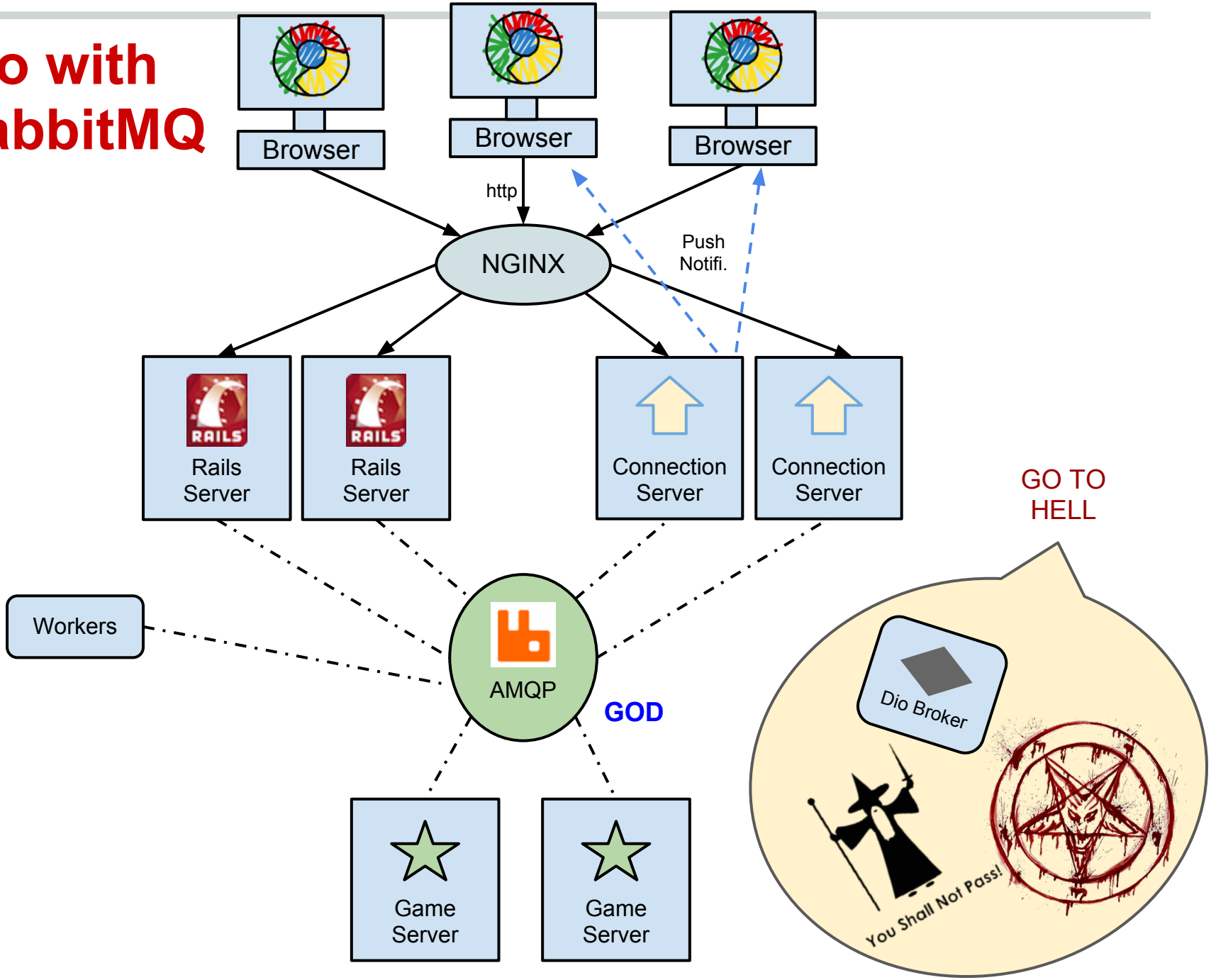
Used for messaging, load balance world instances in the GameServers and allows any component to talk with any other component.

For example send a push notification to the browser from a Sidekiq worker.

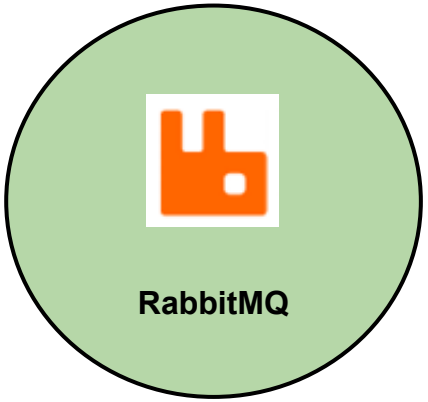
Run world instances (several instances on each GameServer), in real time, in memory.

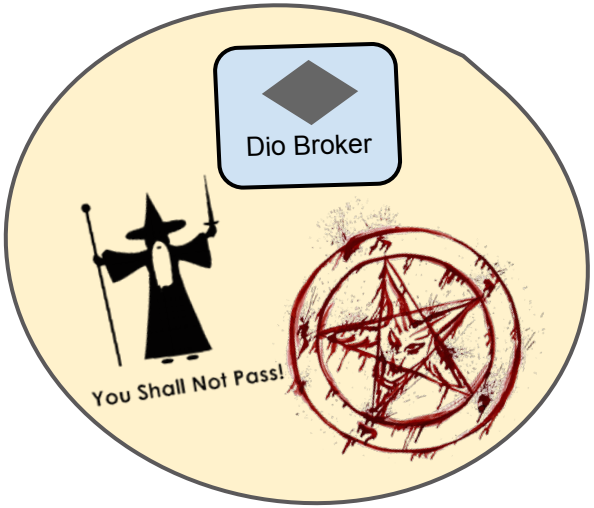
Subscribes to RabbitMQ by world_id channel.
Publish to avatar_id channel

Dio with RabbitMQ



Who would you prefer to worship?





RabbitMQ GOD

supported by a big community,
widely used,
known practices,
and written in erlang

ARCANE GODS

defined by some dudes
in a reactive way
to remove the fears
of the people of their time

Good choice

RabbitMQ ROCKS

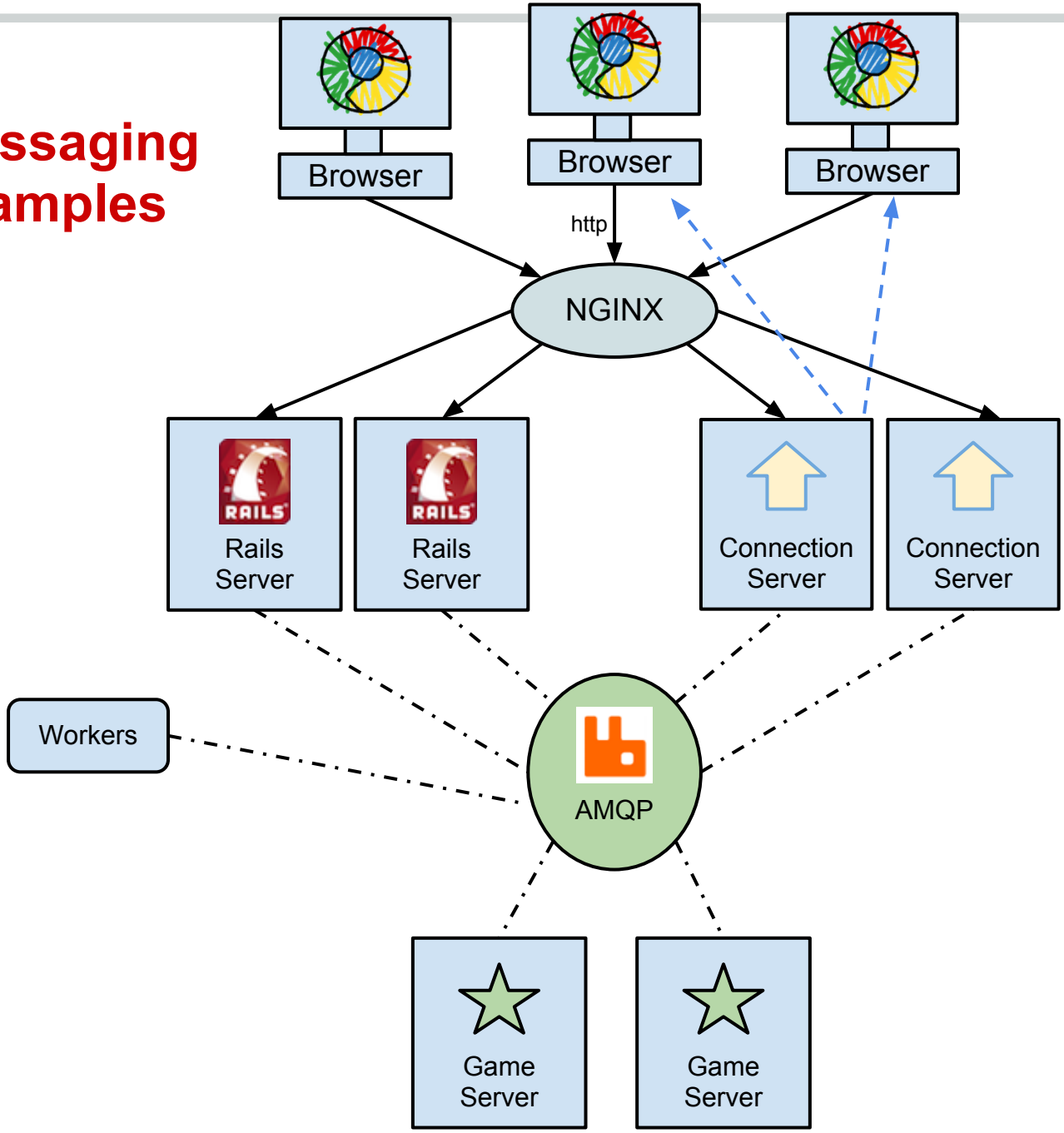
ME TOO

AGREE

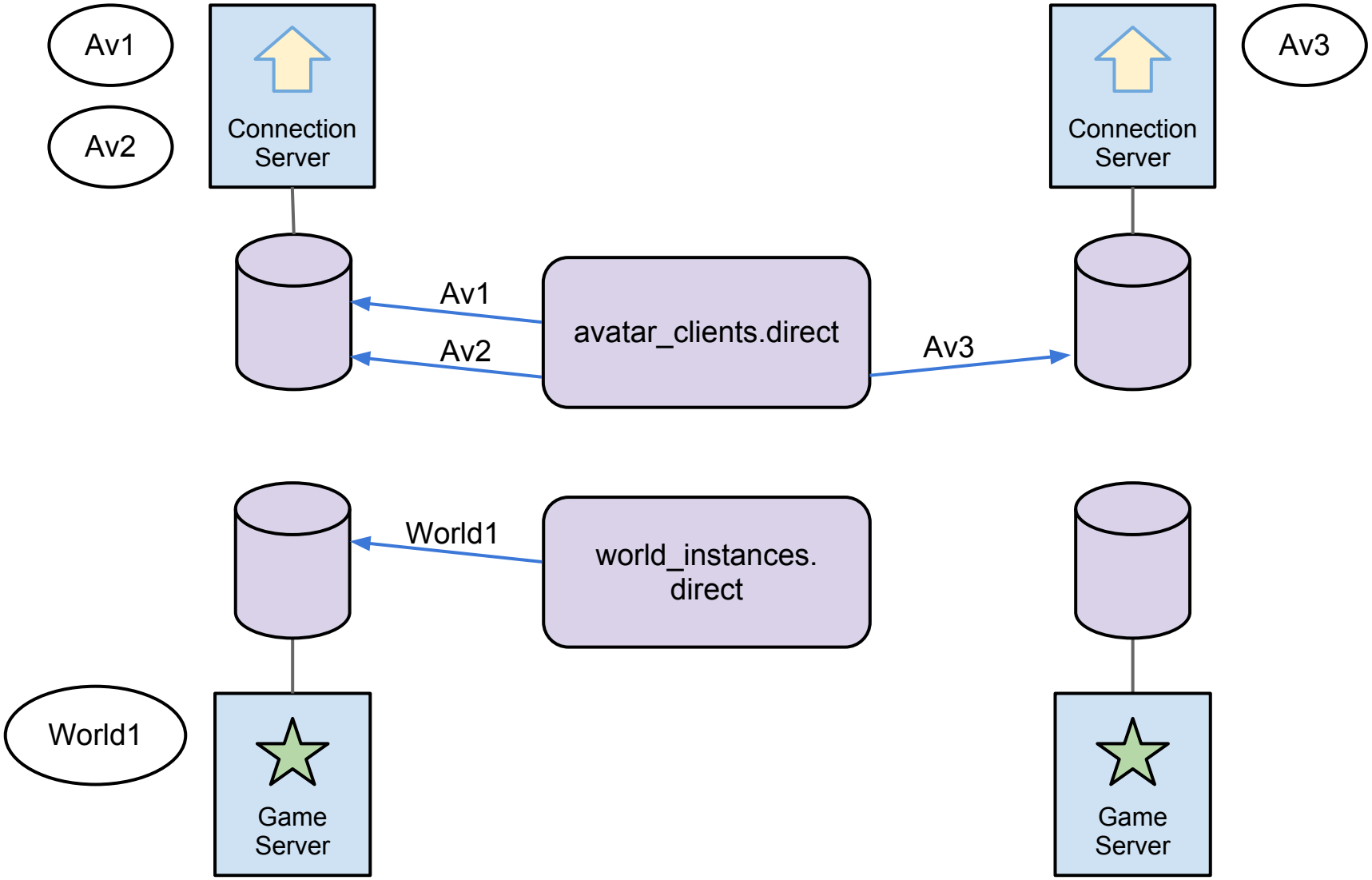
Dio with RabbitMQ: Advantages

- **Consistency**: AMQP is going to be the protocol for messaging.
- **Flexibility**: is easy to add more components when needed, and they instantly are connected with any other component.
- **Visibility**: RabbitMQ is more easy to monitor than a lot bunch of sockets.
- **Less restrictions**: Any system can be a producer and send messages to RabbitMQ, so if we need to send a message to a world from Rails is easy, and if we need to send a push notification to the user from rails or from a worker is also easy.
- **Support**: RabbitMQ has much better support in ruby than ZeroMQ.
- **More Features**: RabbitMQ can accomplish a lot more. ZeroMQ is like a simple abstraction over sockets whether RabbitMQ is a full messaging system (similar to compare the file system with a database).
- **Routing out of the box**: for example a Game Server can subscribe for messages to the words that are running on it, and the Connection Server just publishes to that world, no need to know what Game Server is running that world.
- **No complex Broker and automatic load balancing**: We can use RabbitMQ queues to round-robin new worlds, and the Game Servers can subscribe/unsubscribe to that queue when they are available or overloaded. So they don't need to know about other Game Servers.

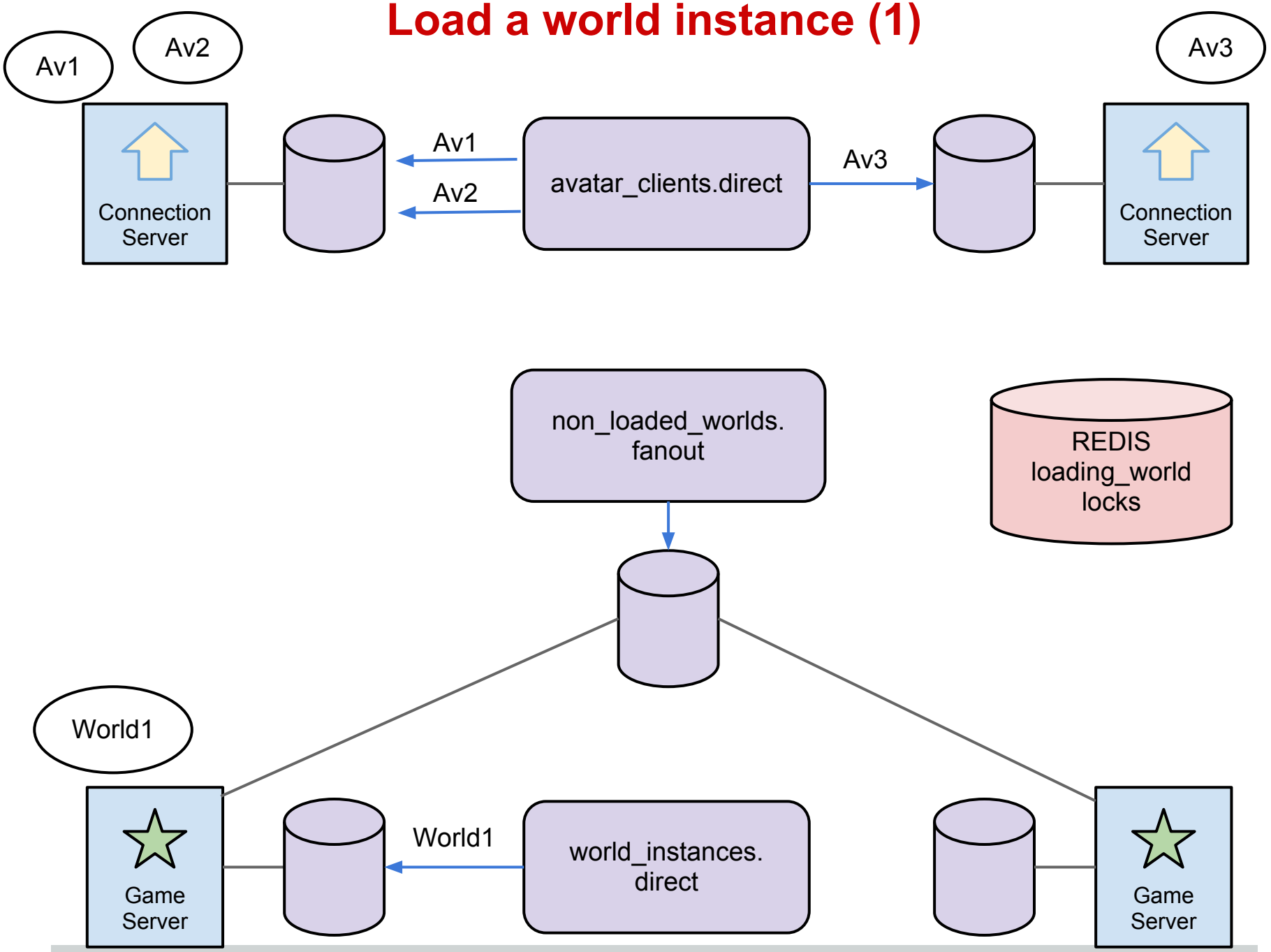
Messaging examples



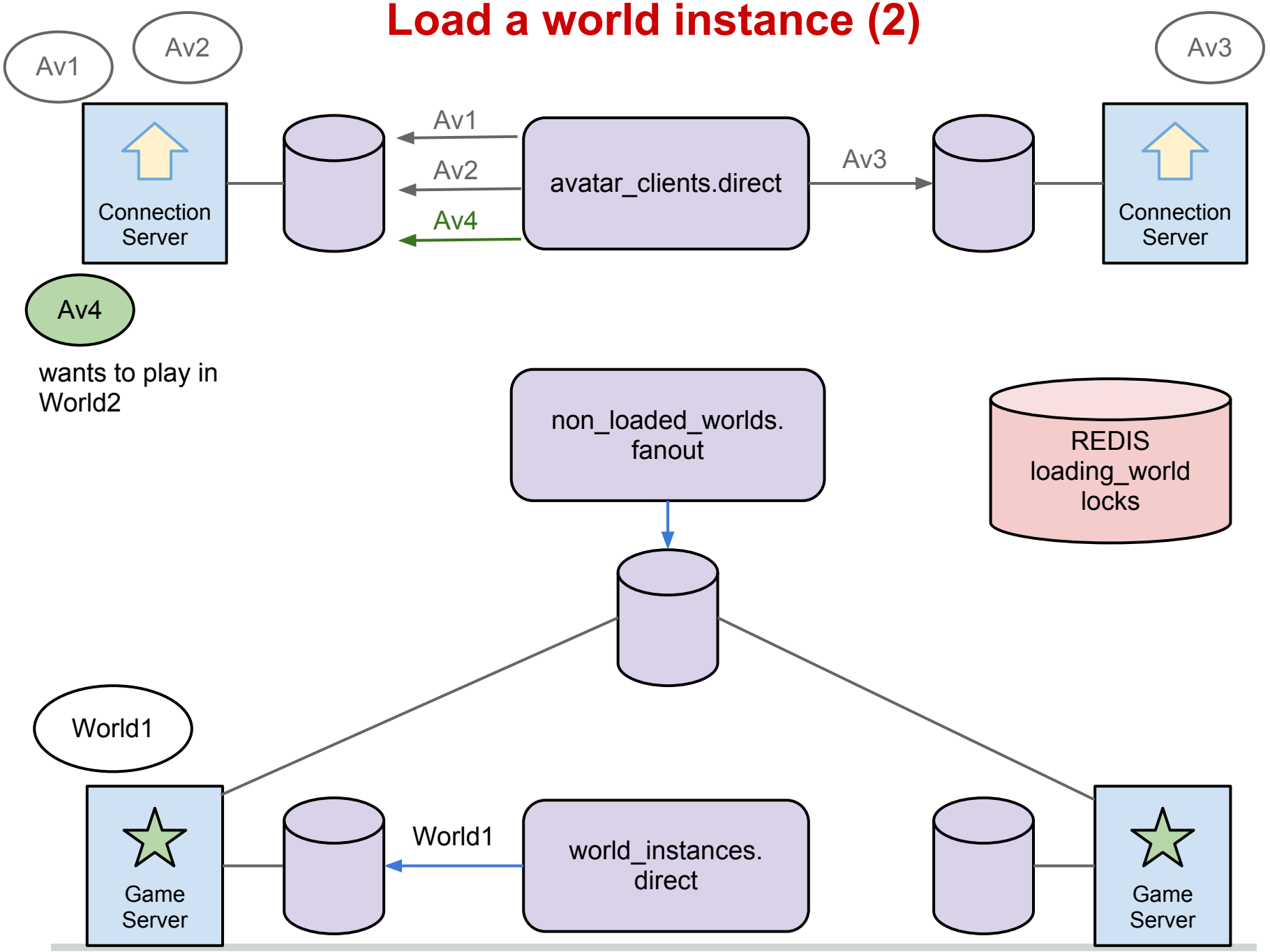
Game Messages



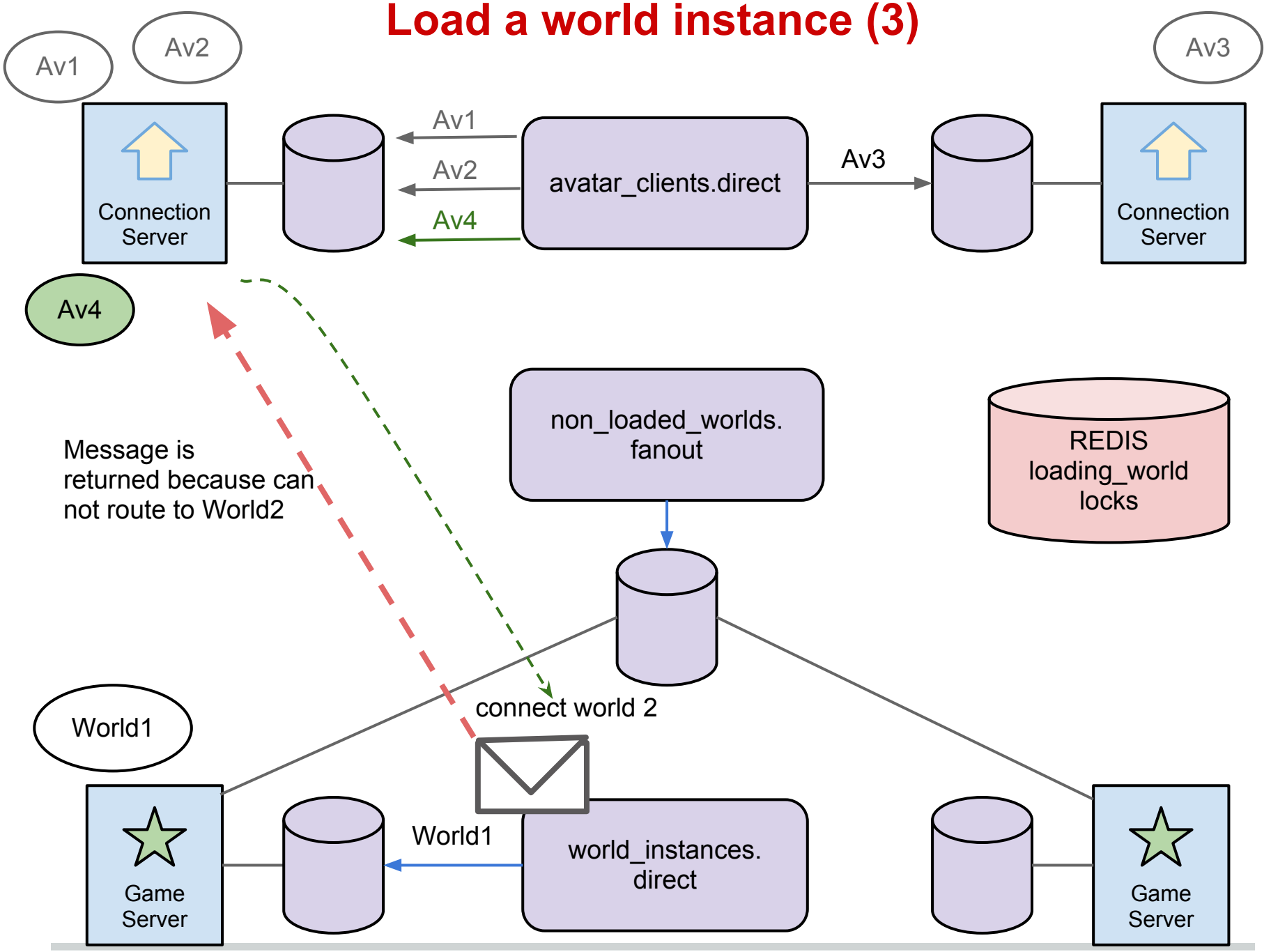
Load a world instance (1)



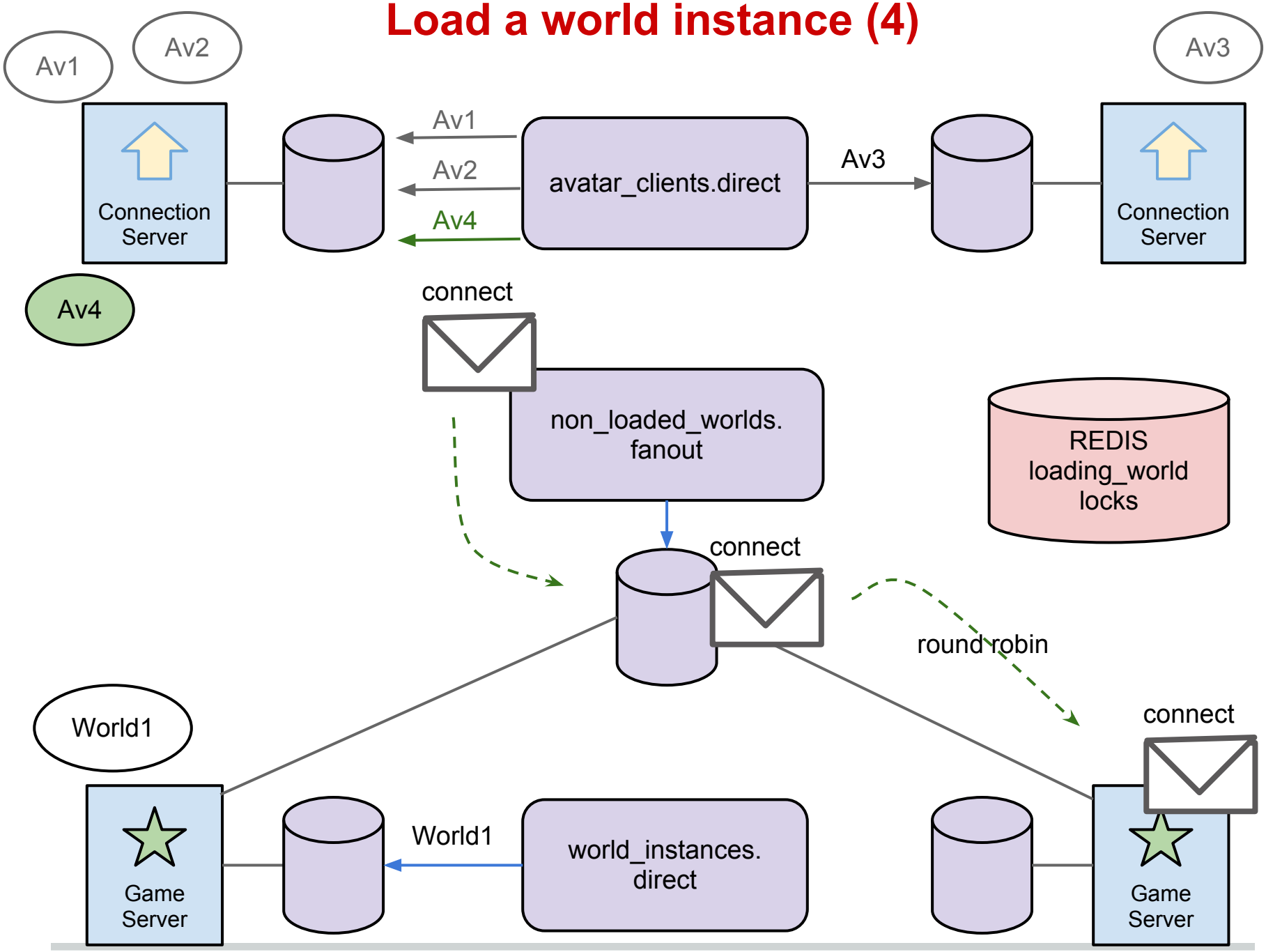
Load a world instance (2)



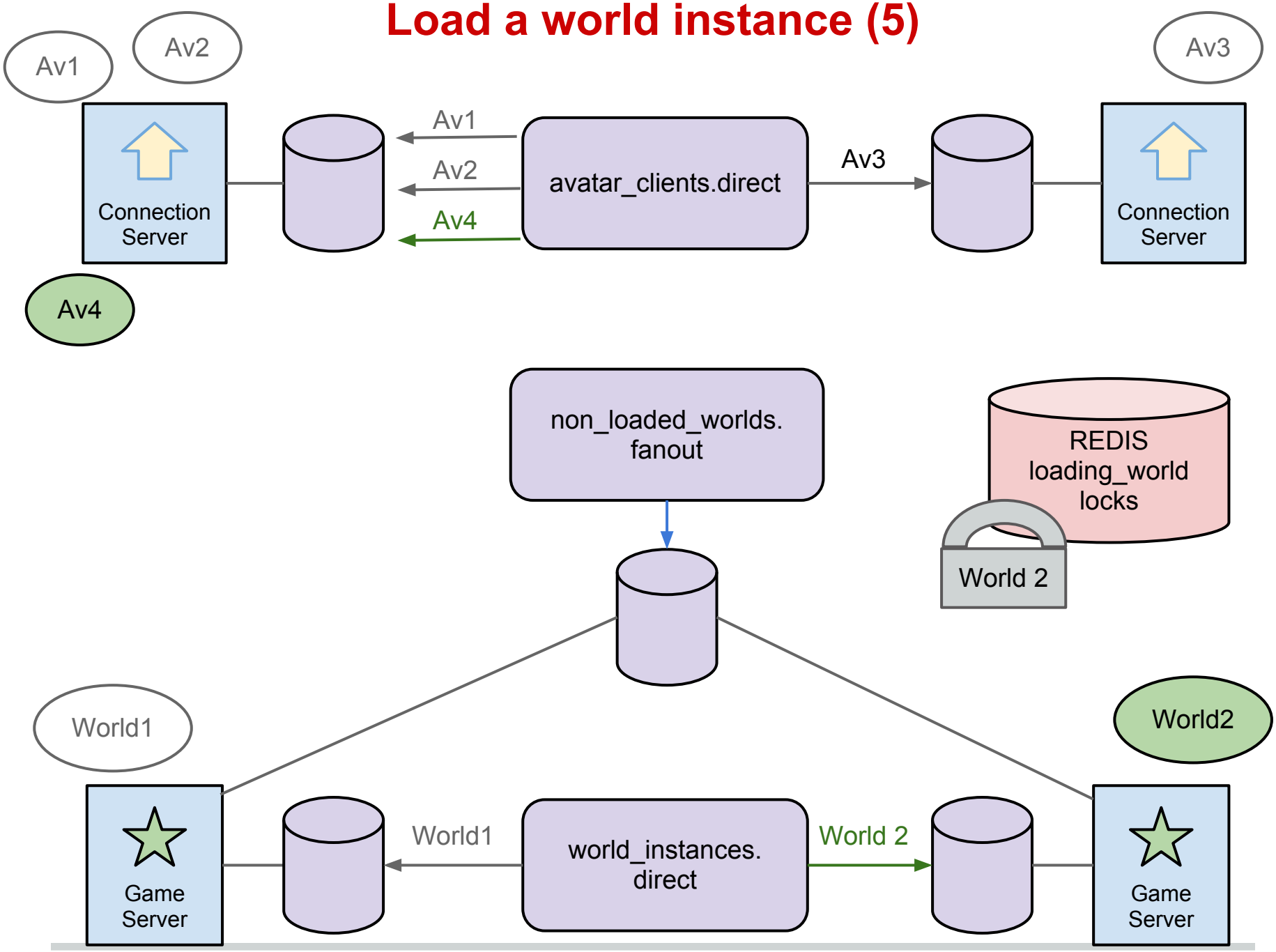
Load a world instance (3)



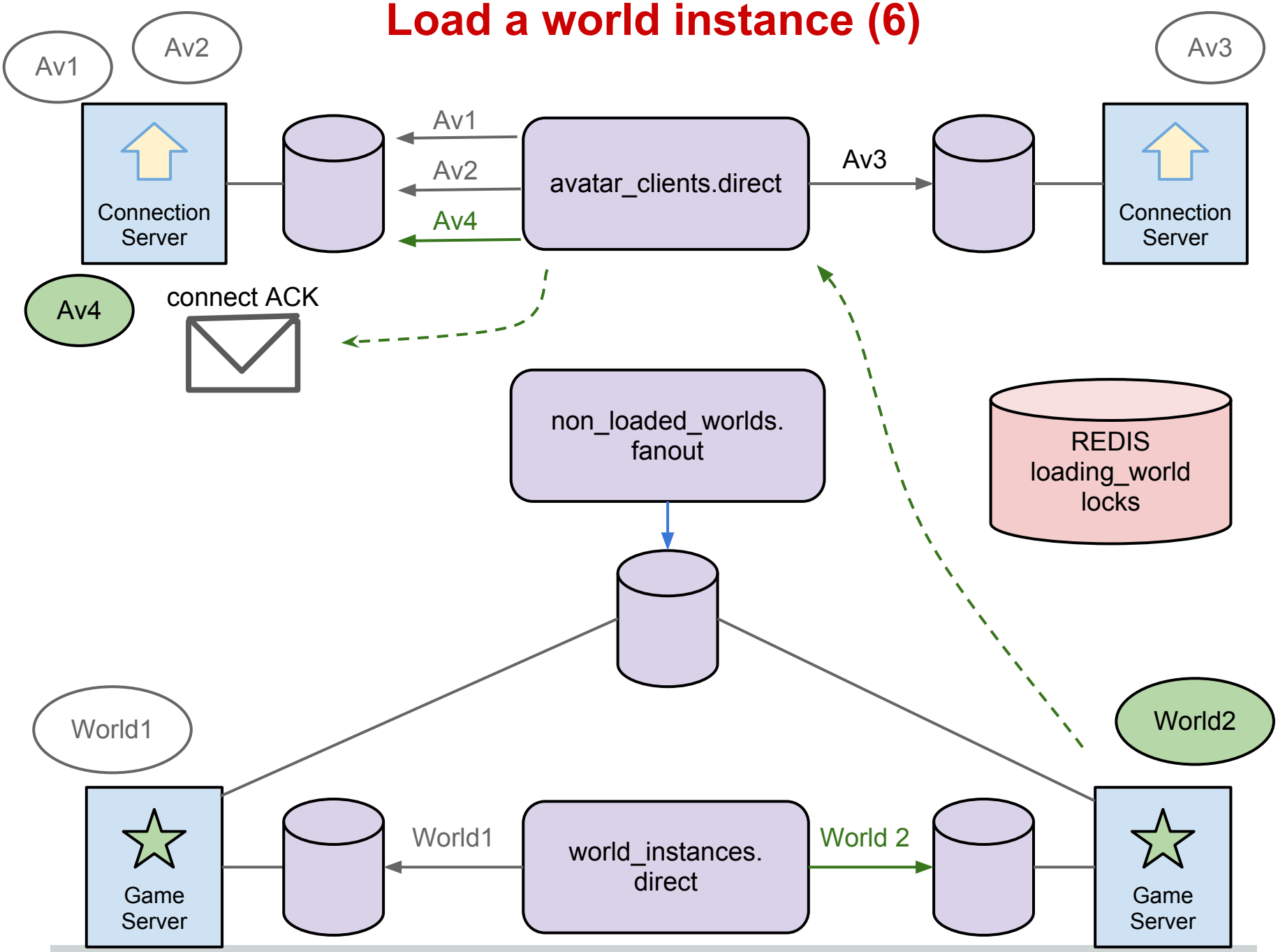
Load a world instance (4)



Load a world instance (5)



Load a world instance (6)



Thanks

Questions for the end

Have fun!

Versu: Evolution of an Erlang Project

James Mayfield

Erlang can help you even if you don't know erlang!

What is Versu?

What is Versu?

"Versu is an interactive storytelling platform that builds experiences around characters and social interaction"

What is Versu?

- Simulator

What is Versu?

- Simulator
  - Written in C

What is Versu?

- Simulator
  - Written in C
  - Robust, but can crash (content errors)

What is Versu?

- "Business Logic"

What is Versu?

- "Business Logic"
  - Talks to C process via ports

What is Versu?

- "Business Logic"
  - Talks to C process via ports
  - Tracks players and games

What is Versu?

- "Business Logic"
  - Talks to C process via ports
  - Tracks players and games
  - Some data transformation of simulator I/O

What is Versu?

- Web App

tm

What is Versu?

- Web App
  - Expose Rest-like interface to simulator functions

tm

What is Versu?

- Web App
  - Expose Rest-like interface to simulator functions
  - UI

tm

What is Versu?

- Data Store

What is Versu?

- Data Store
  - Saved games

What is Versu?

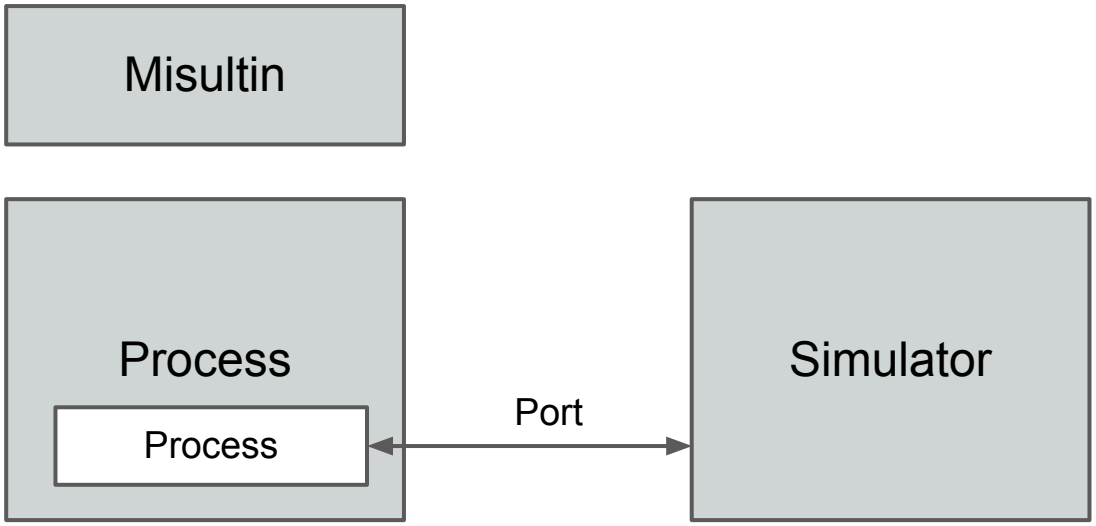
- Data Store
  - Saved games
  - User generated content

The Evolution of Versu

The early days...

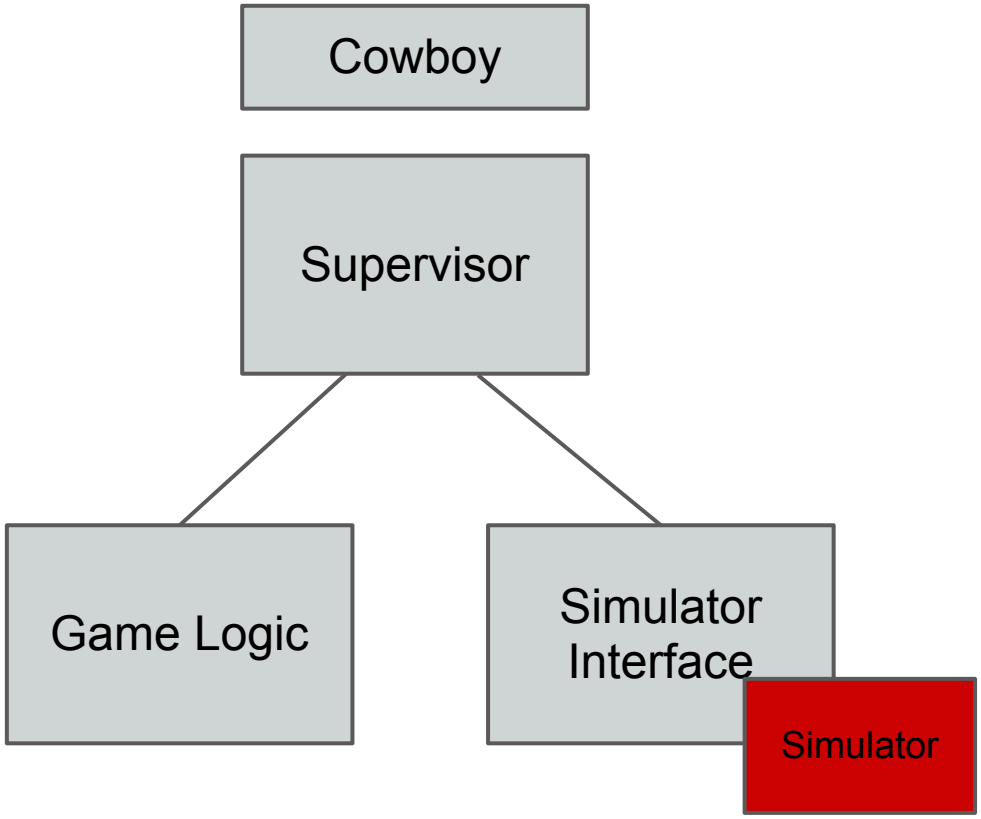
The Evolution of Versu

The early days...



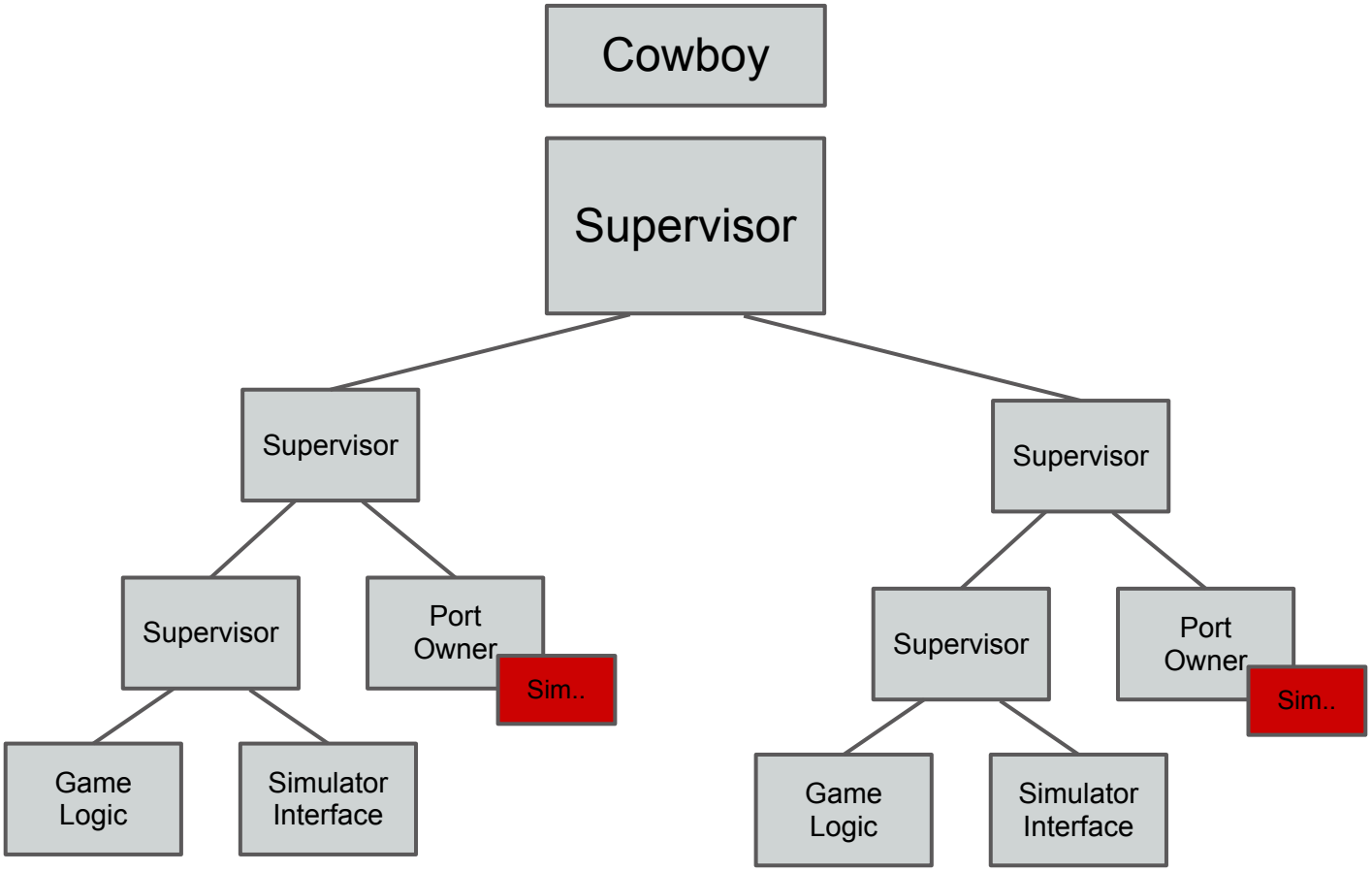
The Evolution of Versu

First Steps...



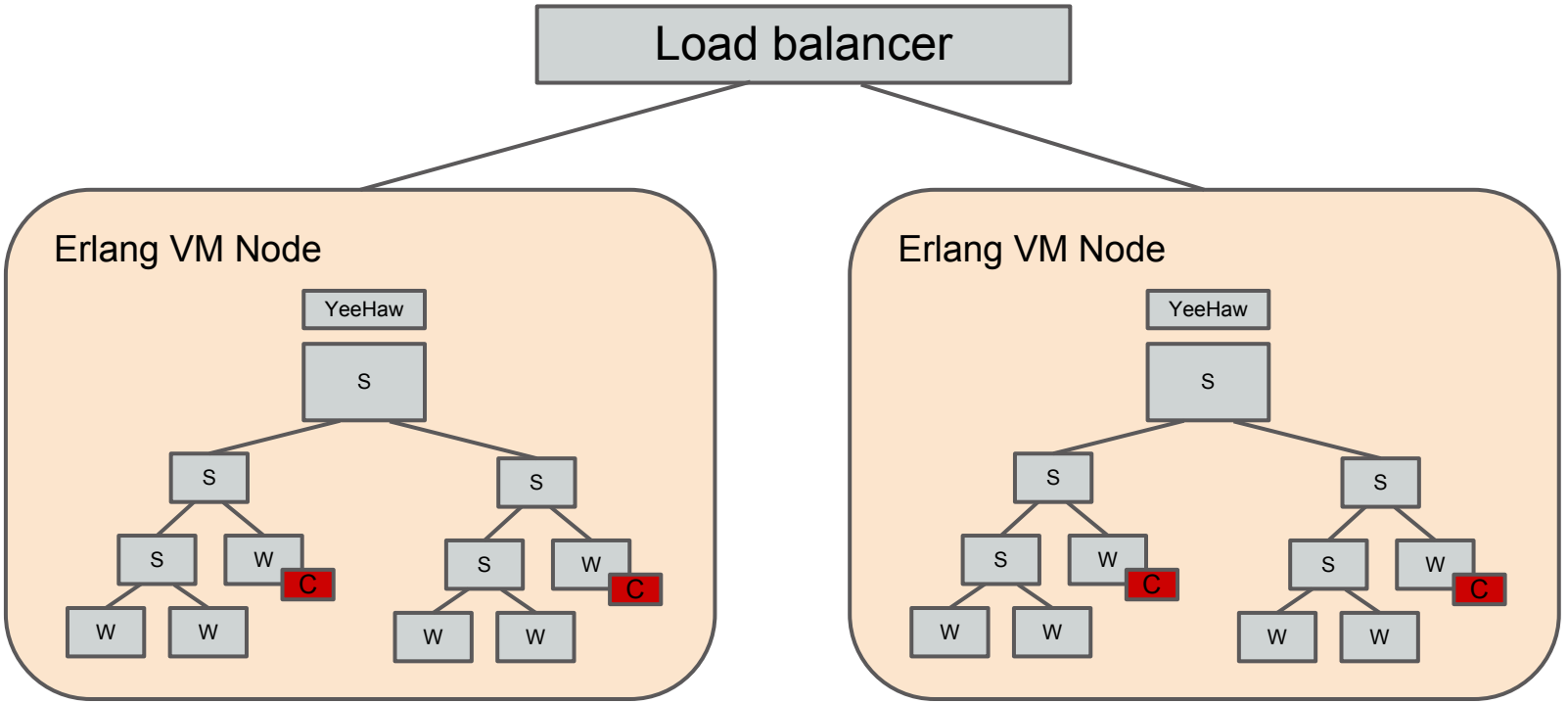
The Evolution of Versu

Many sims per node



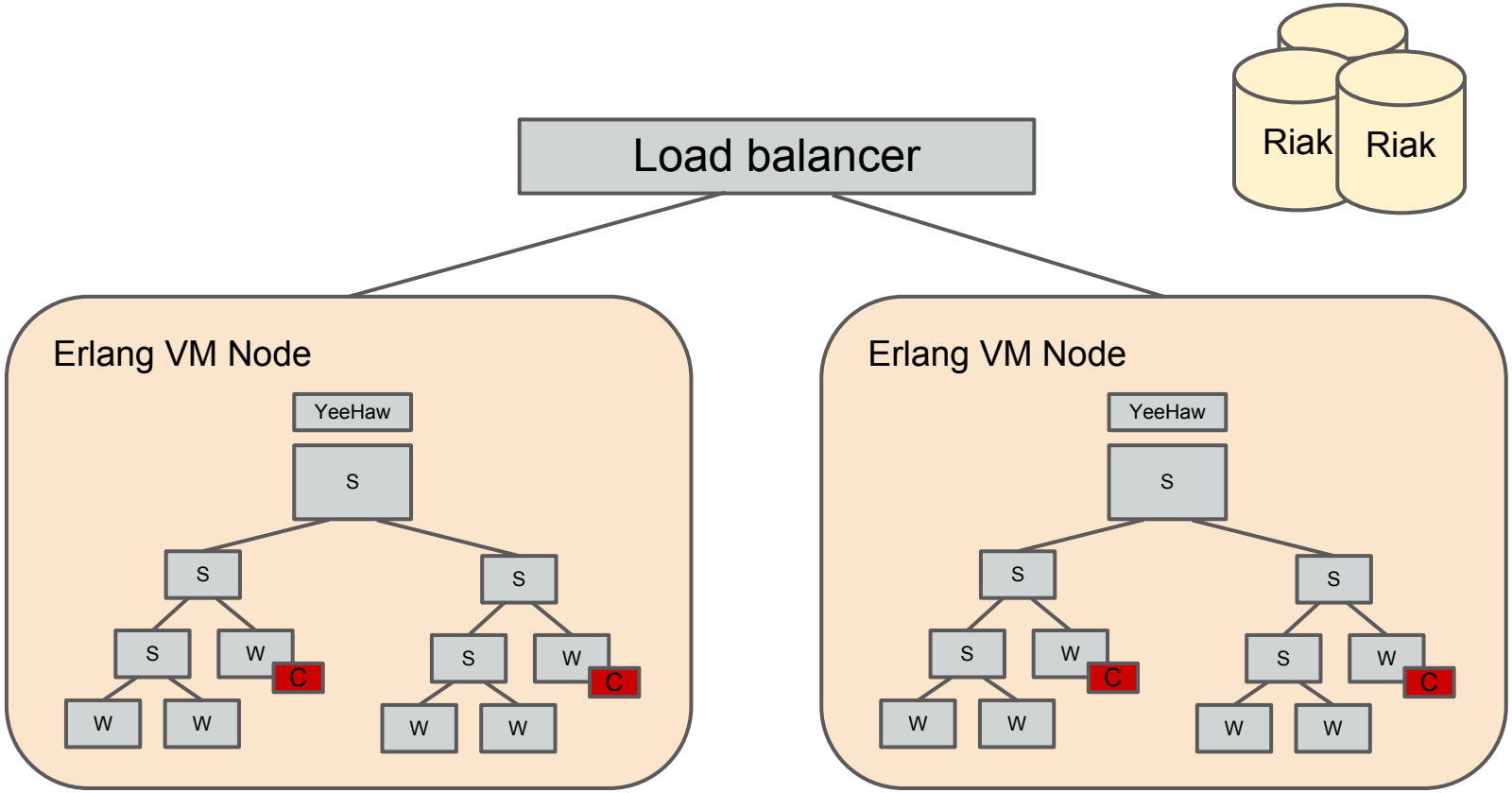
The Evolution of Versu

Increased Awesomeness



The Evolution of Versu

More Increased Awesomeness



Take away

Take away

- Happy devs

Take away

- Happy devs
- Happy ops

Take away

- Happy devs
- Happy ops
- much to learn!

Thank You

References

- http://www.ibm.com/developerworks/cloud/library/cl-optimizepythoncloud2/index.html
-