企业级Rails架构

from startup to enterprise

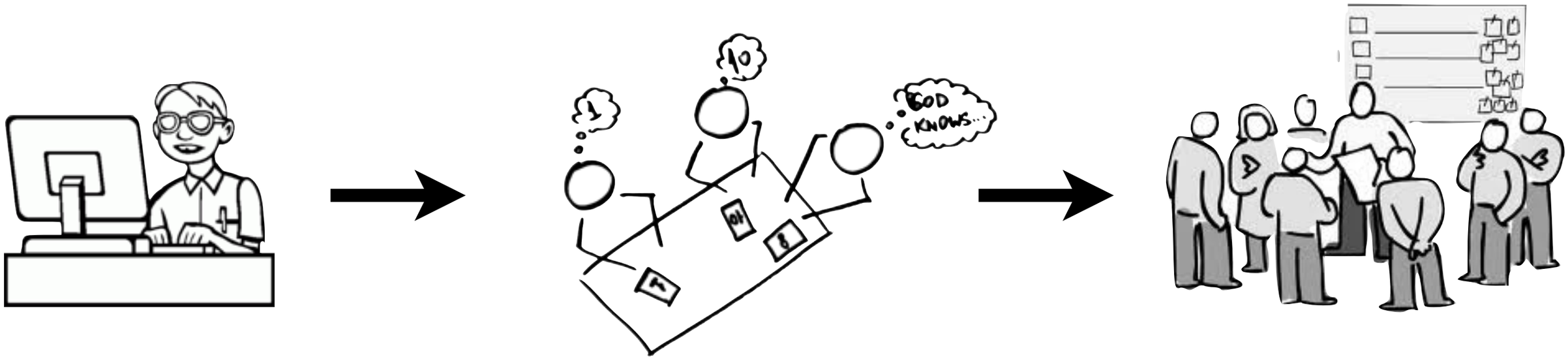
About Me

- 陈金洲 **Thought**Works咨询师
- 金数据创始人 jinshuju.net
- 从2001年开始编写商用软件
- Java, .NET背景。2007年开始使用Rails
- @mechiland

企业级?

- 多个人，多种技术，多种产品或服务
- ... is an ongoing business function that helps an 'enterprise' figure out how to best execute the strategies that drive its development [wikipedia]

从个人开始……



……到团队

- 有规模的企业刚开始的时候也很小
- 小团队使用Rails因为其犀利的生产力
- 技术栈随着团队和产品一起成长

“15分钟创建Blog”

“24小时创建一个网站”

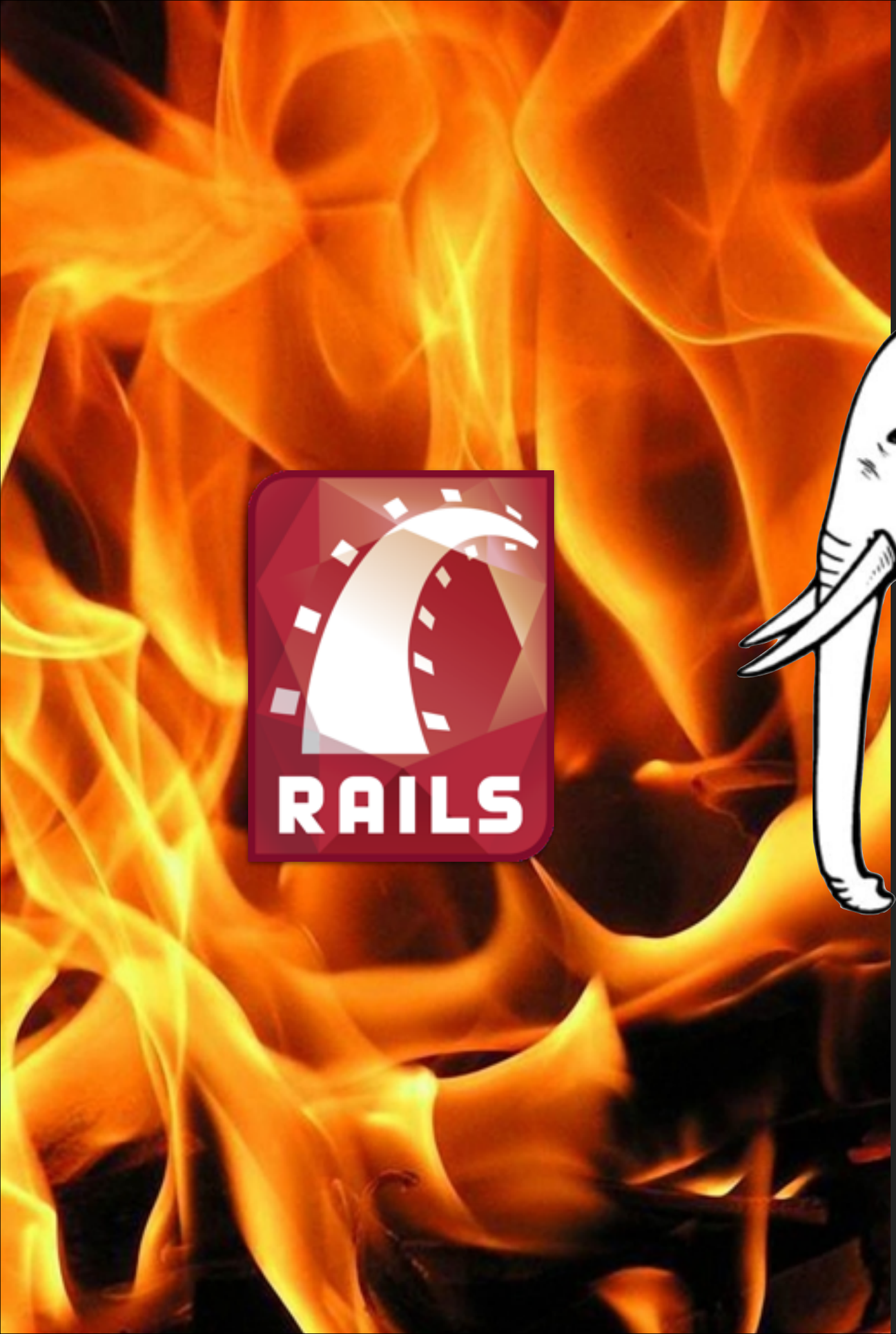
企业

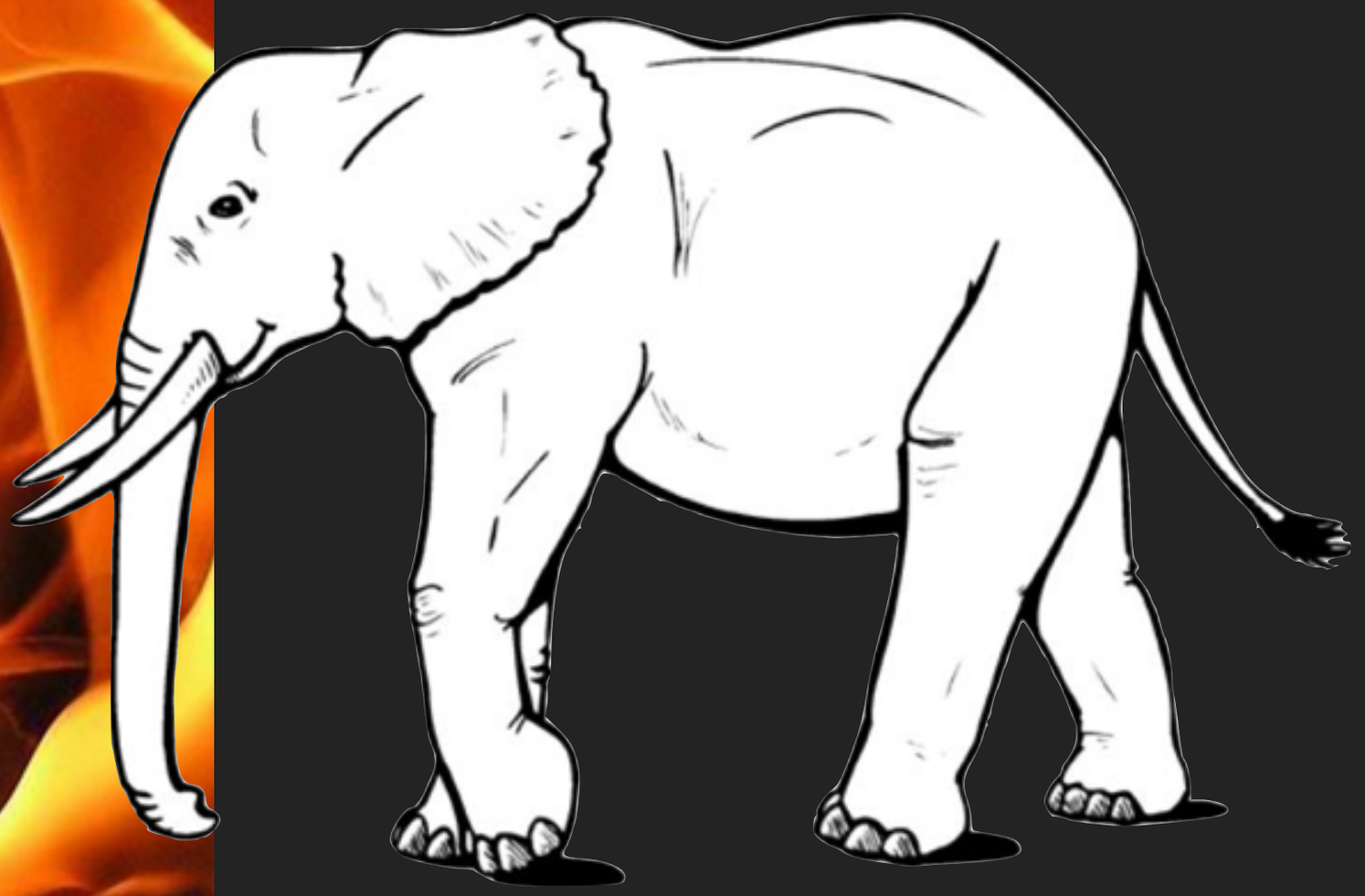
服务
众多用户

新功能
开发慢

客户
响应慢

技术
陈旧


RAILS



如何在架构层面利用Rails的高生产力，保持技术上的竞争力?

准备

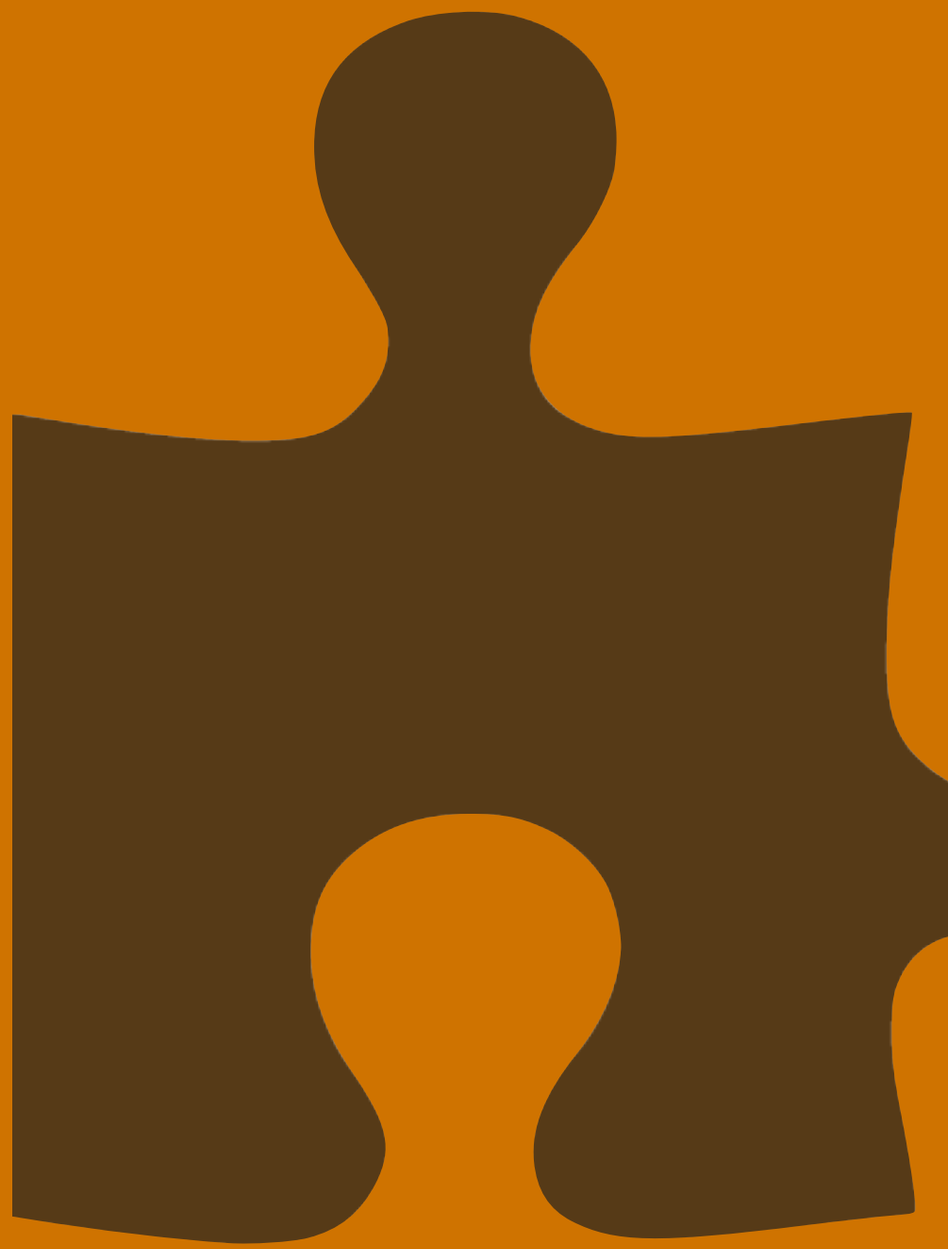
团队

2-6开发人员

产品经理/设计师

前端开发很重要

流程



- 自动化测试，从第一天开始
- 将任务拆解到半天以内
- 晨会, Code Review
- Showcase
- 其他能够增进沟通的手段

工具

Trello

Jenkins

skype™

for Mac

核心工具栈

- RVM/rbenv
- Omniauth
- TestUnit(minitest), RSpec, Jasmini, Capybara
- Bootstrap, purecss
- Slim/haml
- MySQL/PostgreSQL/Mongodb

Rails架构?

Rails项目有架构吗?

Rails

Model

View

Controller

DB


incoming http
nginx
unicorn
DB

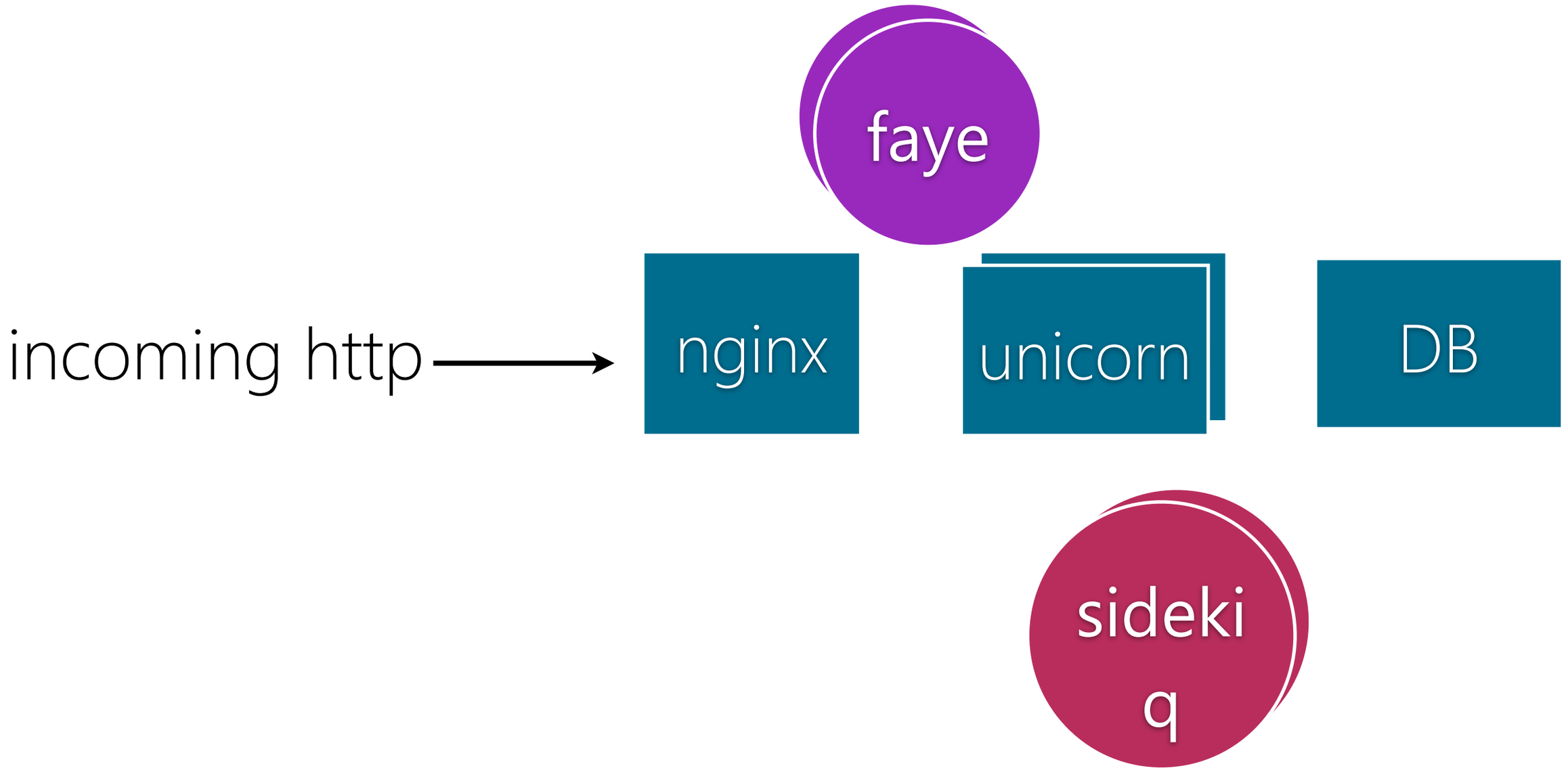
incoming http

nginx

unicorn

DB

sideki
q


faye
incoming http
nginx
unicorn
DB
sideki
q

去年


passenger
Rails deployment that just works


resque


cronjob

Thanks Linux.

Linux/Rails社区已经打好了基础，无状态、分层良好的rails应用在部署架构上几乎没有挑战。

真正的挑战

- 更为复杂的业务
- 不断增多的功能
- 更为先进的（开源）技术

服务
众多用户

新功能
开发慢

客户
响应慢

技术
陈旧

今年

3000

代码行阈值

Name	Lines	LOC	Classes	Methods	M/C	LOC/M
Controllers	2769	2315	36	313	8	5
Helpers	566	481	0	77	0	4
Models	5169	4042	86	665	7	4
Libraries	754	611	9	72	8	6
Integration tests	342	271	4	1	0	269
Functional tests	4760	3917	30	13	0	299
Unit tests	7394	6124	77	21	0	289
Total	21754	17761	242	1162	4	13

Code LOC: 7449 Test LOC: 10312 Code to Test Ratio: 1:1.4

Name	Lines	LOC	Classes	Methods	M/C	LOC/M
Controllers	2769	2315	36	313	8	5
Helpers	566	481	0	77	0	4
Models	5169	4042	86	665	7	4
Libraries	754	611	9	72	8	6
Integration tests	342	271	4	1	0	269
Functional tests	4760	3917	30	13	0	299
Unit tests	7394	6124	77	21	0	289
Total	21754	17761	242	1162	4	13

Code LOC: 7449 Test LOC: 10312 Code to Test Ratio: 1:1.4

一年不到5000行?

今年

核心应用

jinshuju.net

去年

核心应用
jinshuju.net

核心部分
jinshuju.net

数据统计

去年

核心应用
jinshuju.net

核心部分
jinshuju.net

数据统计

核心部分
jinshuju.net

API

数据统计

不必等待变得臃肿

差不多就做应用规划

常见规划策略

- 按业务功能拆分
  - 用户信息，照片，日志
  - 应用1：管理界面，元数据维护，支持
  - 应用2：系统状态，系统监控，部署
  - 应用3：大数据分析，预测，模型

功能重用

- ~~Copy & Paste~~
- iframe, javascript重用
- Gem重用

Gem重用

- 通用功能：发短信，支付
- 业务独立的算法

数据/服务重用

- Cookie
- 共享数据库
- jsonp
- CAS：单点登录
- JSON API

挑战：有了新订单

- 通知分析引擎进行分析
- 财务系统需要更新
- 需要通知发货系统

挑战：有了新订单

- 通知分析引擎进行分析
- 财务系统需要更新
- 需要通知发货系统

谁来更新这些系统?

分布式消息系统

发布/订阅和点对点异步中间件

Amazon SQS

```ruby
EventMachine.run do
  connection = AMQP.connect(:host => '127.0.0.1')

  channel  = AMQP::Channel.new(connection)
  queue    = channel.queue("amqpgem.examples.hello_world", :auto_delete => true)
  exchange = channel.default_exchange

  queue.subscribe do |payload|
    puts "Received a message: #{payload}. Disconnecting..."
    connection.close {
      EventMachine.stop { exit }
    }
  end
  exchange.publish "Hello, world!", :routing_key => queue.name
end
```

RabbitMQ

Amazon SQS

总结

- 小团队，注重交流
- 限制系统的规模（代码行数）
- 在恰当的时候对系统进行规划和拆分
- 通过Gem进行功能重用
- 通过API进行数据重用
- 通过消息服务进行系统间主动通信

谢谢大家！

@mechiland

http://jinshuju.net