Why Build Cloud Native Applications?

Why Build Cloud Native Applications?

- speed
- safety
- scale
- mobile

Speed Wins in the Marketplace

How fast are you delivering applications?

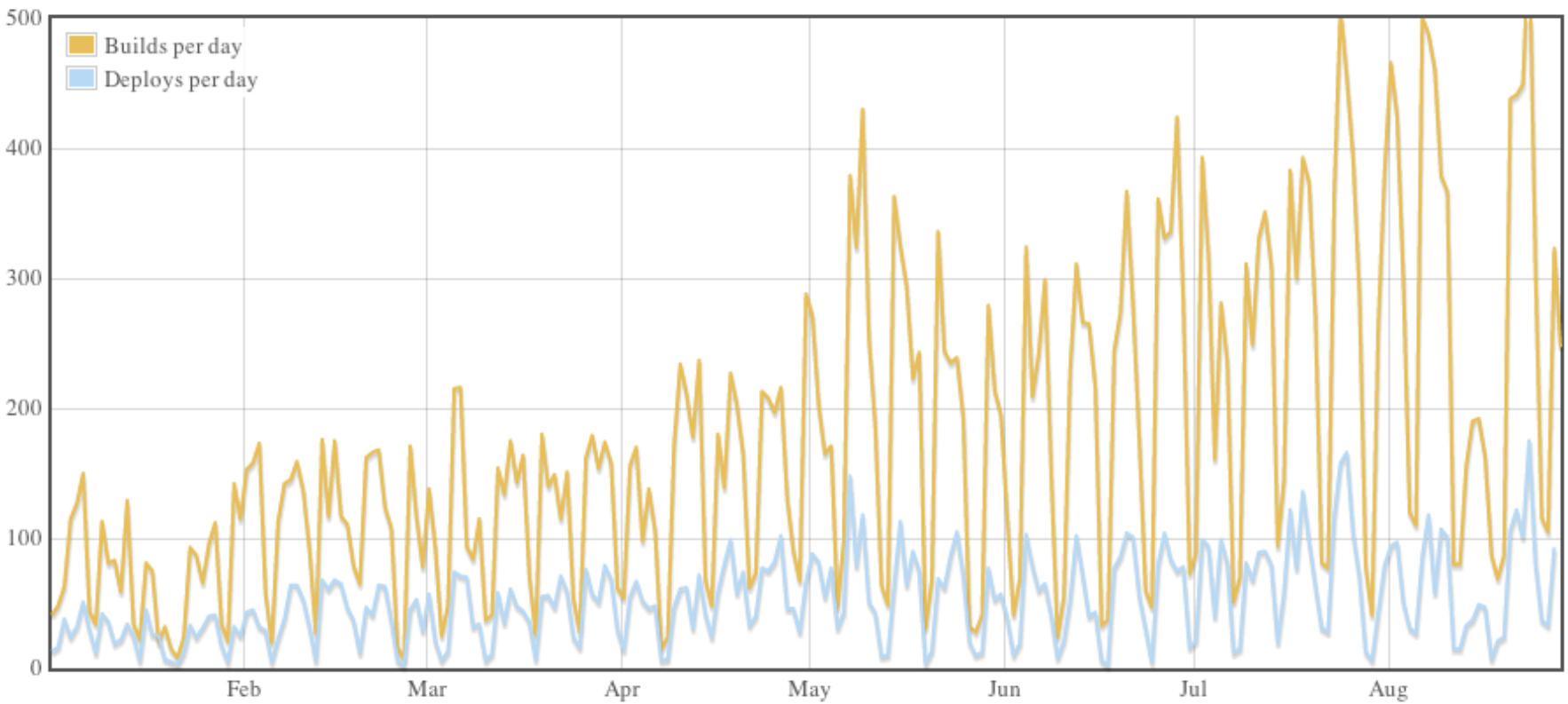
"How long would it take your organization to deploy a change that involves just one single line of code?"

Mary Poppendieck, Lean Expert

Always be shipping

Github Deployment Stats (company wide - 2012):

- 41,679 builds
- 12,602 deploys



Why Build Cloud Native Applications?

- speed
- safety
- scale
- mobile

How can we move fast, but do so safely?

Visibility

Measure everything, establish profiles for what is normal.

Use modern monitoring and operational tools.

Fault Isolation

Limit the scope of components that can be impacted by a failure.

Monoliths often bring down the entire app if one component is failing.

Does Amazon stop you from checking out if recommendations are down (think μservices)?

Fault Tolerance

One failure must not cause a cascading failure across the entire system.

For example, for an application that depends on 30 services where each service has 99.99% uptime, here is what you can expect:

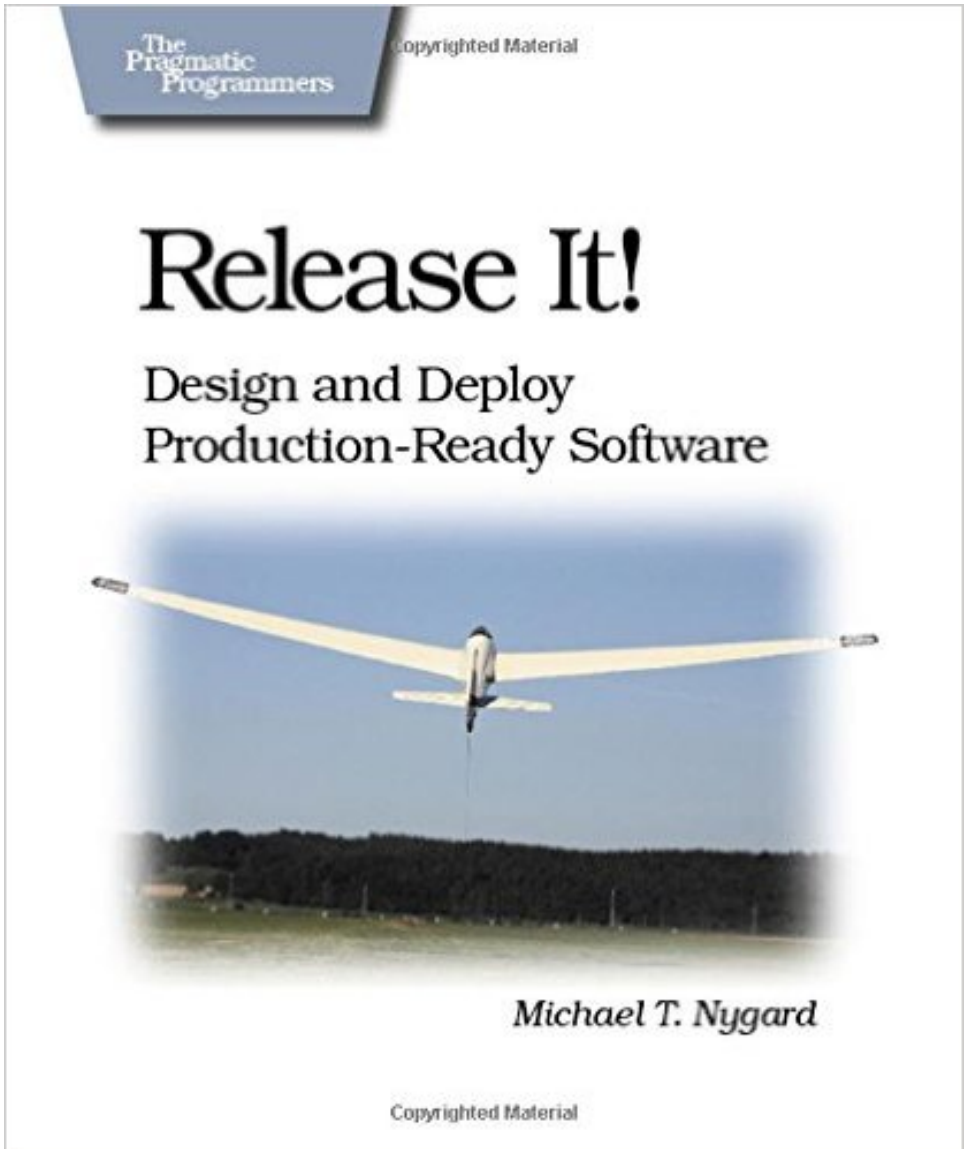
```
99.99^30 = 99.7% uptime

0.3% of 1 billion requests = 3,000,000 failures

2+ hours downtime/month even if all dependencies have excellent uptime.
```

Reality is *generally* worse.

Fault Tolerance

Fault Tolerance Patterns like the *Circuit Breaker* stop cascading failures.

As described by, Micheal T. Nygard in *Release It!*



auto-recovery

Automate recovery of services where possible.

`e.g restart services when down.`

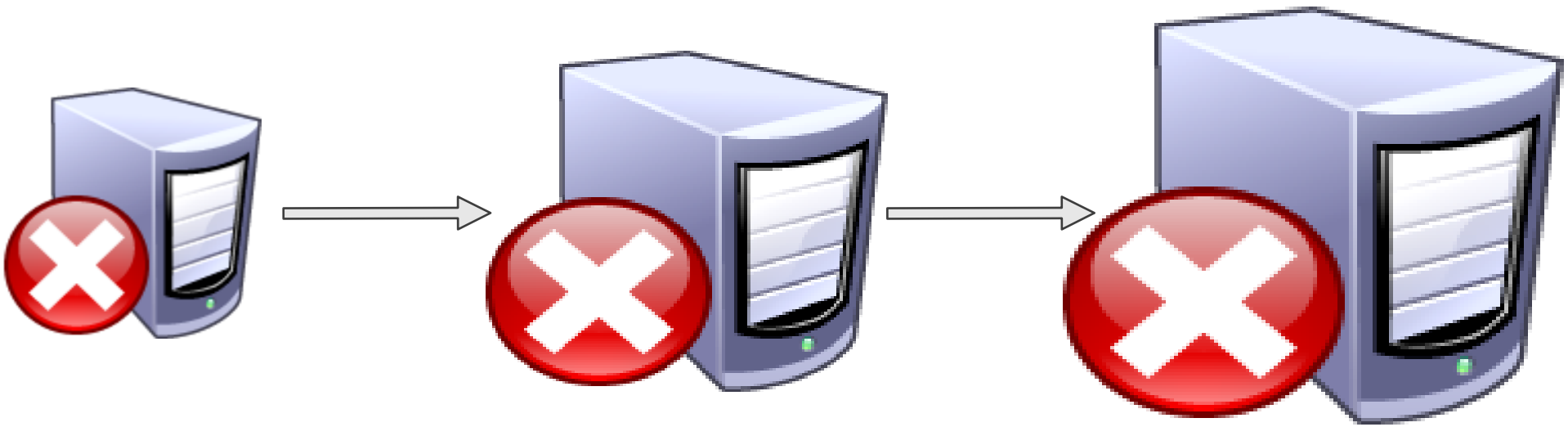
Why Build Cloud Native Applications?

- speed
- safety
- scale
- mobile

How to scale cloud native applications?

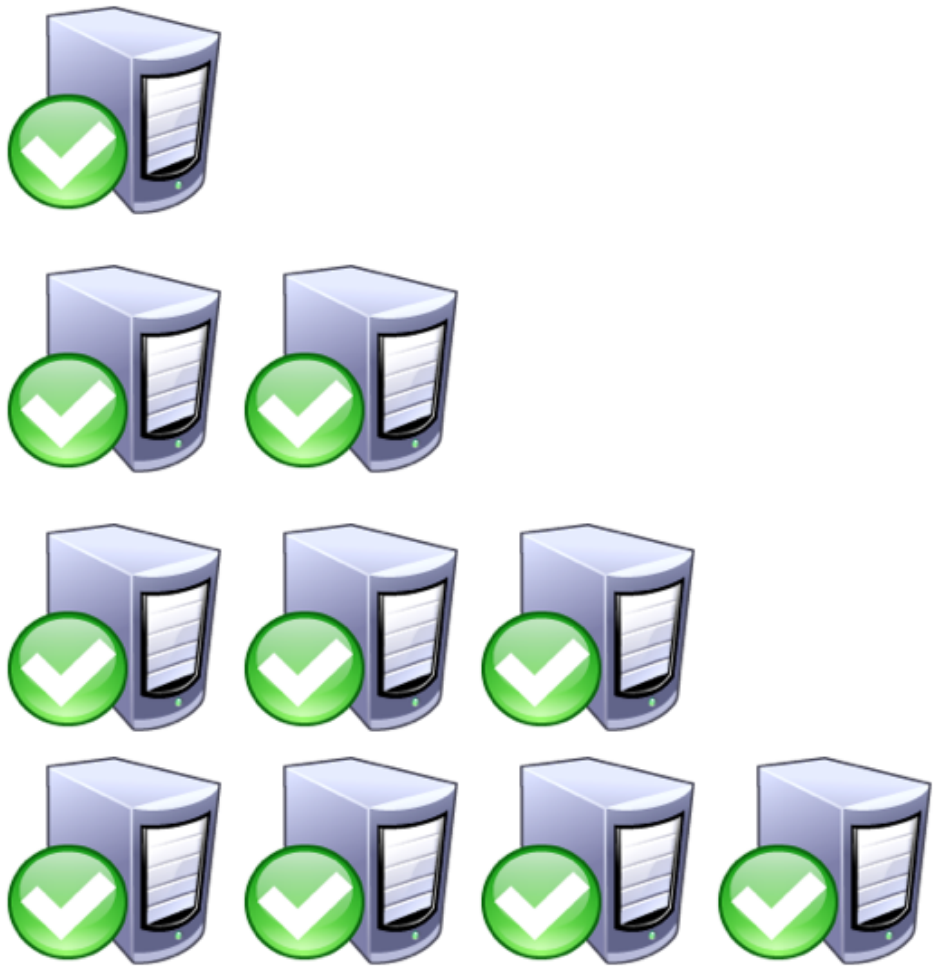
Vertical Scaling Doesn't Work

- Monolithic approach to sizing apps
  - Wrong most of the time
  - Poor utilization
- Long Provisioning Times



Horizontal Scaling Does Work

- Use cheaper, commodity based hardware
- Get better resource utilization through virtualization

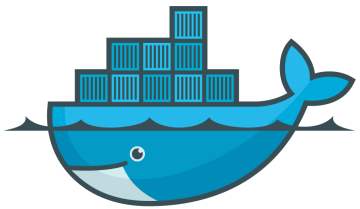


Cloud

- Provision via API
- Accelerate Innovation

Containers Changed the Unit of Deployment Recently

LXC



docker

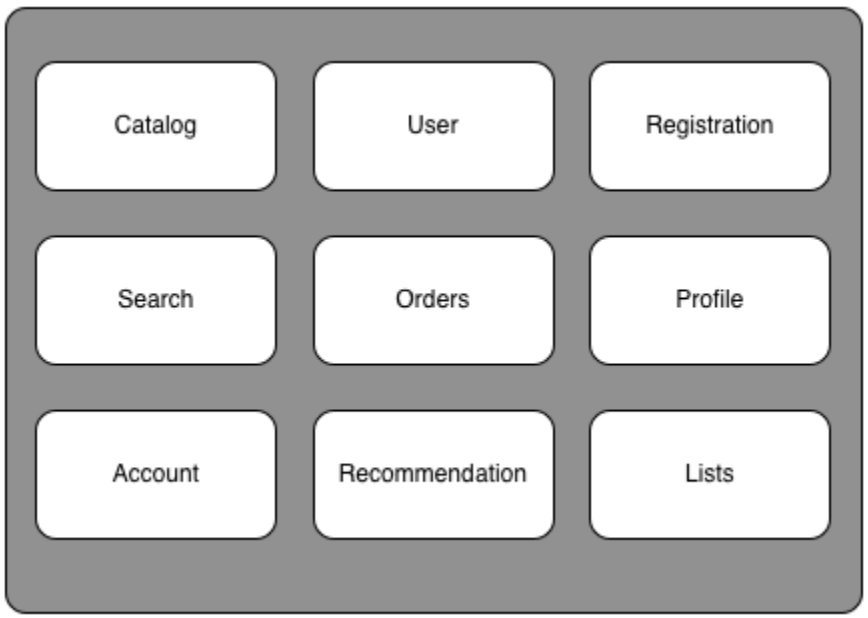
rkt

Downstream effects of cloud on Application Design

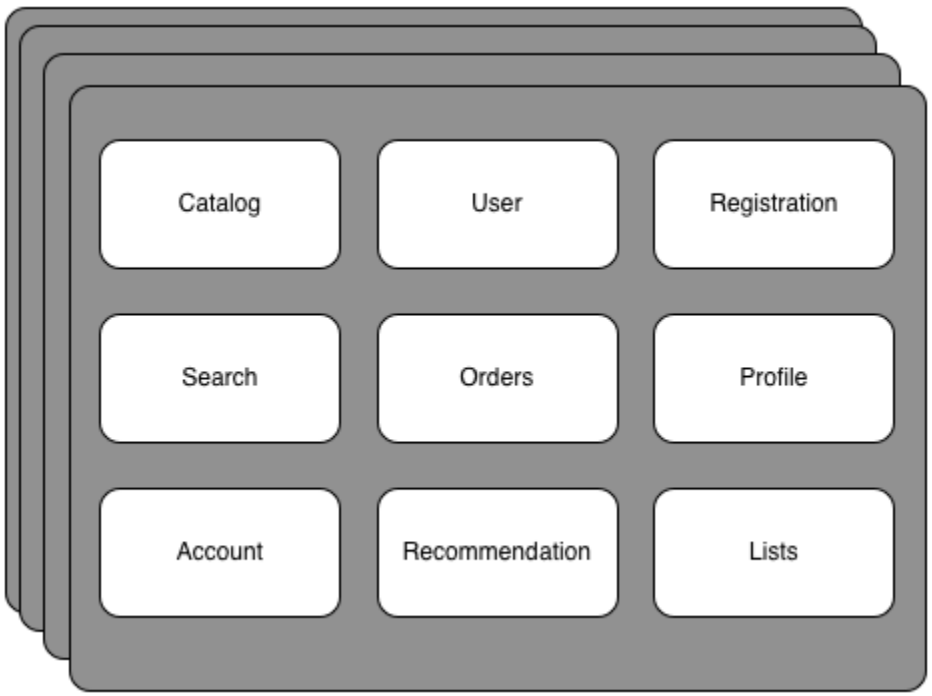
- Resources are ephemeral
- Twelve Factor Patterns
  - e.g. Stateless Applications - Moving state outside of the application (session state, caches, user data)

The Monolithic App

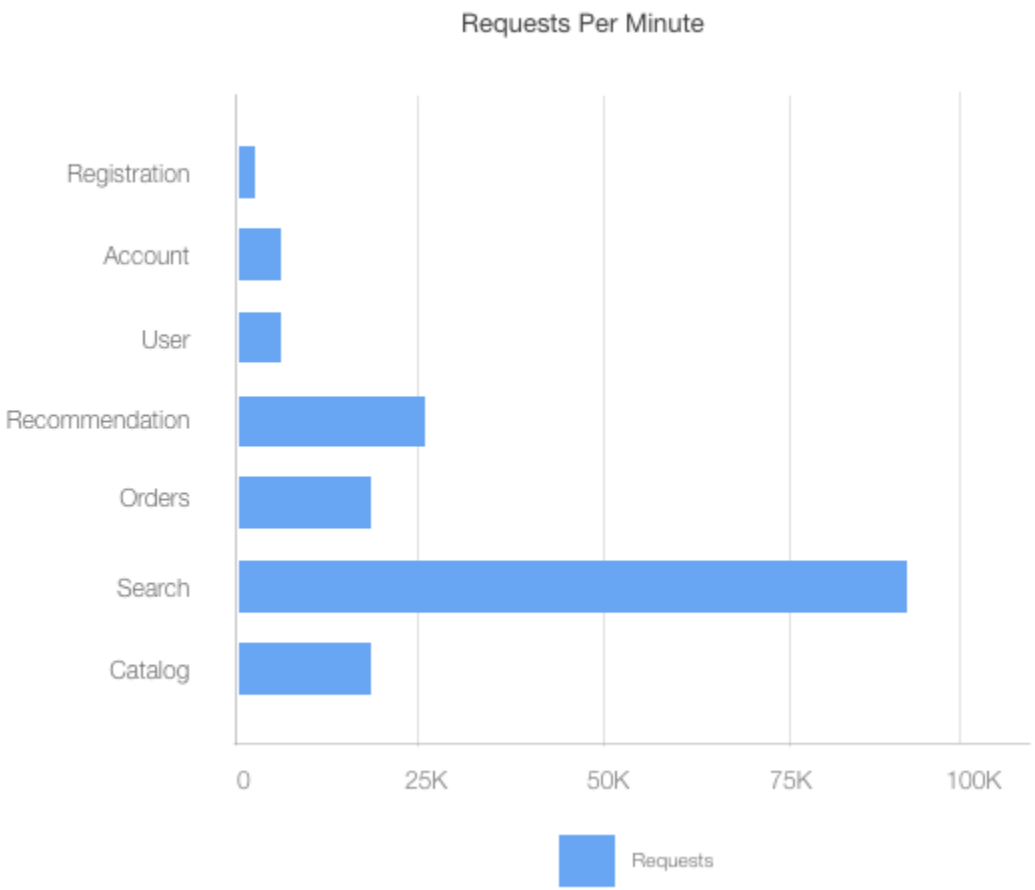
The monolith



Scaling the Monolith



How You Should Scale



Why Build Cloud Native Applications?

- speed
- safety
- scale
- mobile

Mobile Places New Demands on Architectures

Mobile

> *"In January 2014, mobile devices accounted for 55% of Internet usage in the United States. Apps made up 47% of Internet traffic and 8% of traffic came from mobile browsers."*

Source: http://money.cnn.com/2014/02/28/technology/mobile/mobile-apps-internet/

Mobile Effects on Application Design

- Dynamic Workloads
- No downtime
- Integration with Legacy
- API Gateway Pattern