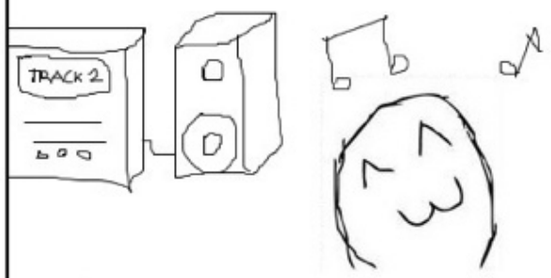
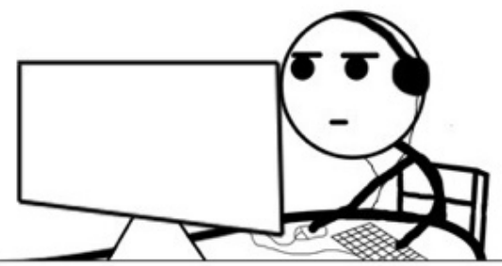
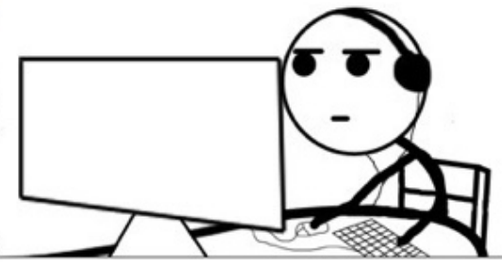
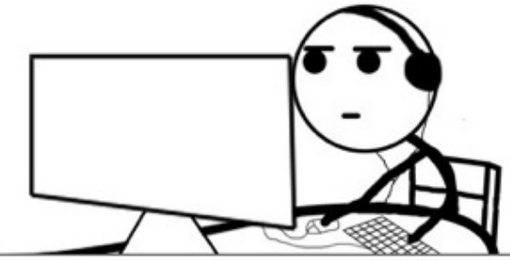
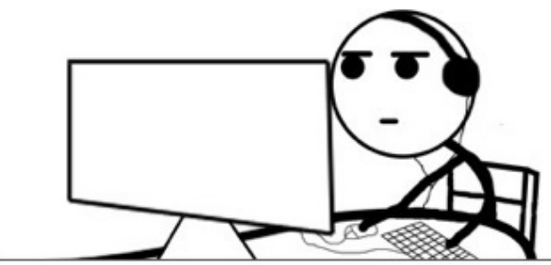
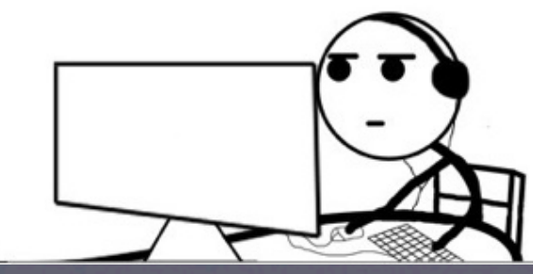
Jeremy Edberg

	15 Years ago	Today
Listening to music		
Watching a movie		
Contacting people		
Reading the news		
Making Music		

Why am I here?

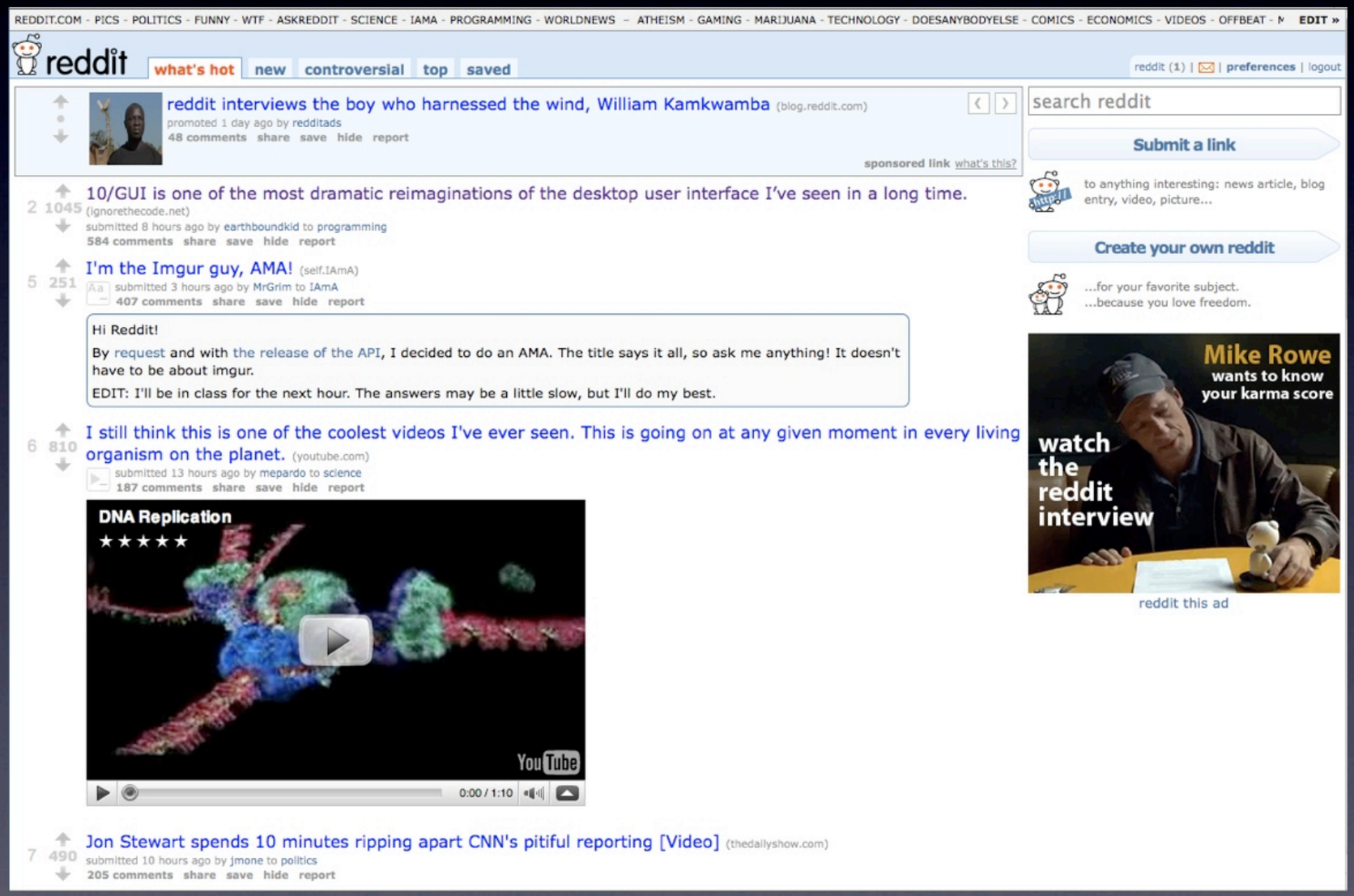
Why should we learn from **other people's** mistakes?


2

Mistakes we've made

What is reddit?

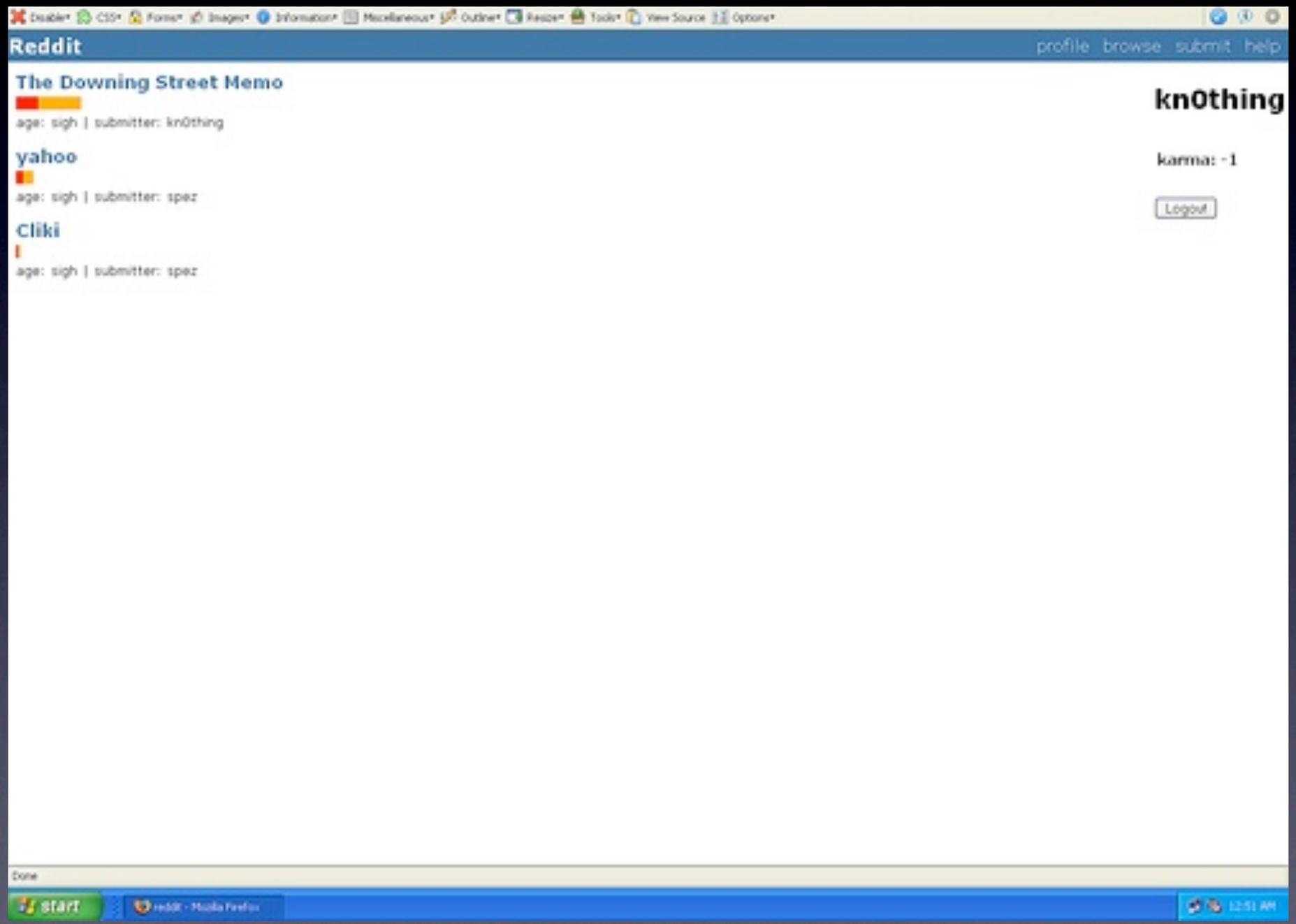
reddit is an online community



Way back in 2005...

- Two UVA students applied for this thing called YCombinator
- They were rejected
- They were called back

June 23rd, 2005



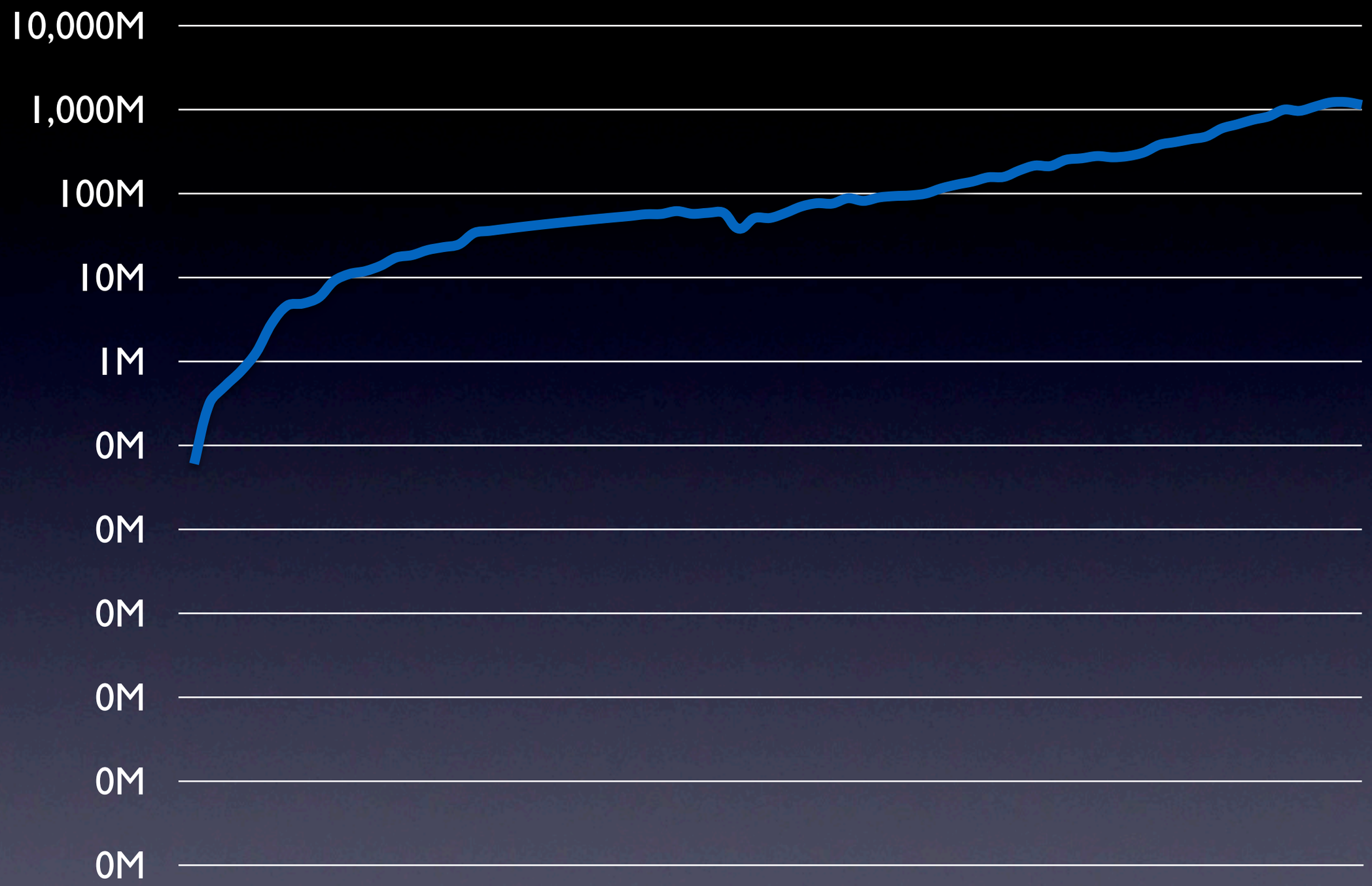
1,500M

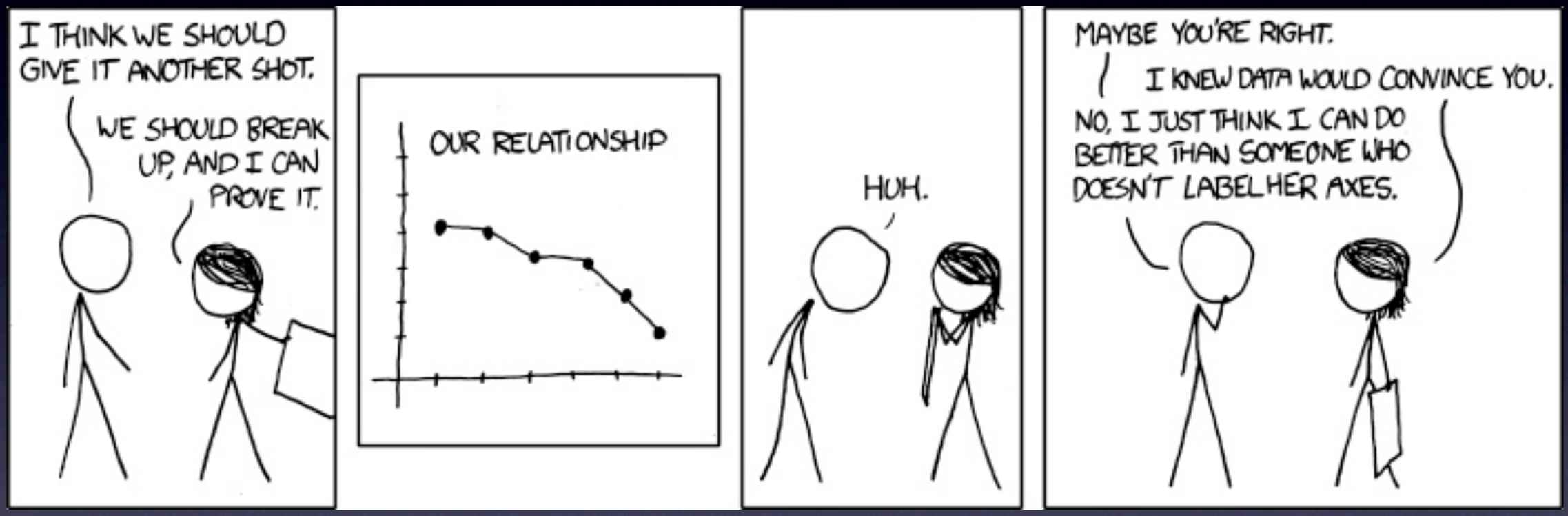
1,125M

750M

375M

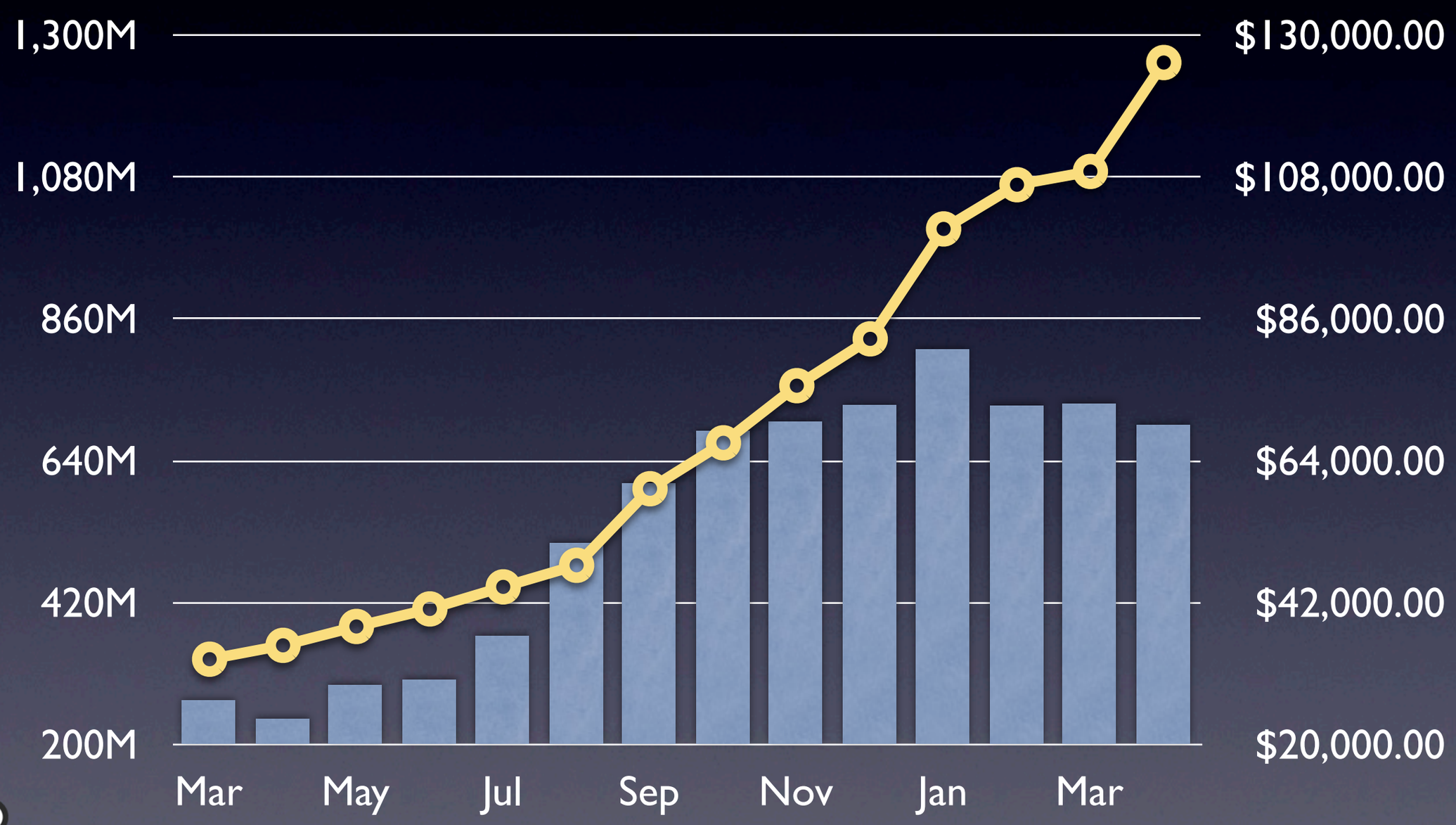
0M


10,000M
1,000M
100M
10M
1M
0M
0M
0M
0M
0M
0M



http://xkcd.com/833/

Monthly Page Views and Costs



Monthly Page Views and Costs

If it won’t scale, it'll fail.

-- paradrox

The key to scaling is finding the bottlenecks before your users do

-- jedberg

Infrastructure

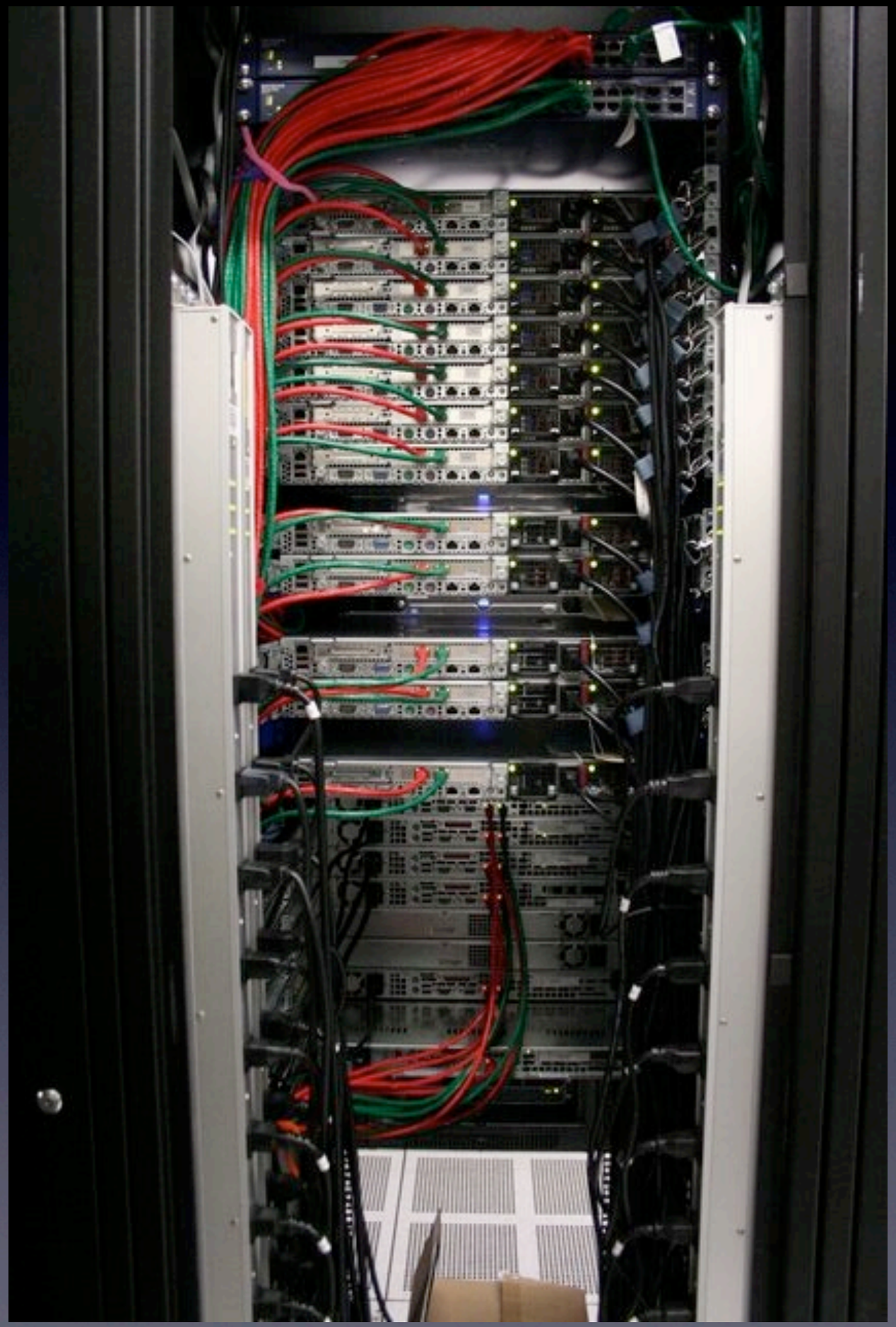
reddit Timeline

April 2006 -- S3 for logos

September 2007 -- S3 for thumbnails

November 2008 -- EC2 for batch processing

May 2009 -- EC2 for entire site



reddit moved from self hosting to EC2

EC2 for Overflow

Used openvpn to create a secure link to our datacenter for batch processing

Moving to EC2

- Started by migrating all data
- Got a complete stack running on EC2
- Long Friday night finishing the migration

and “forklifting” the last bits of data

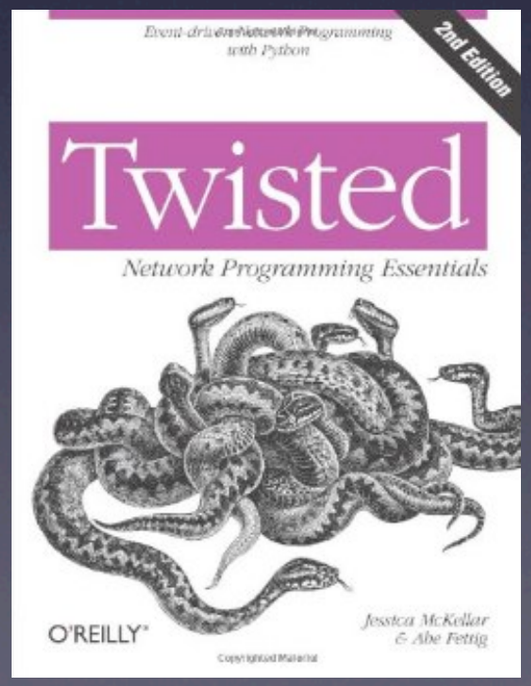
Takeaways

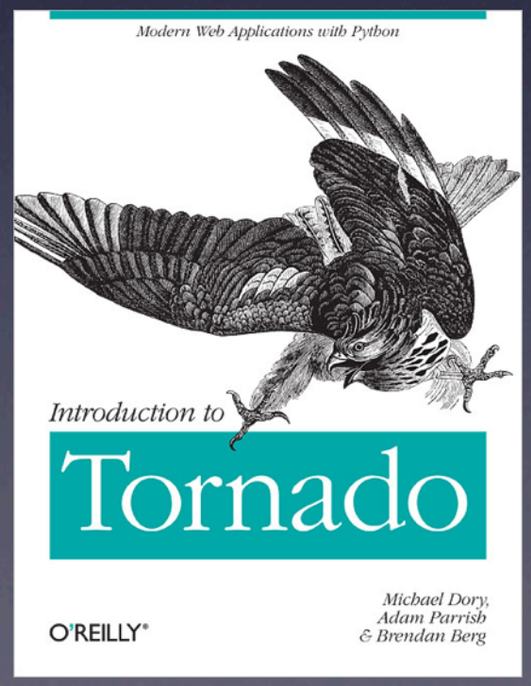
- EC2 makes things easier, but isn't a magic bullet.
- The higher network latency and noisy neighbors will be problematic -- expect to work around it.
- Scaling on EC2 is a lot like anywhere else, but you need to be more disciplined.

Webserver or Proxy?

What about event driven and non-blocking web servers?

- Good for long connections
- More complicated to start, but scales better





Mistake

- Not accounting for increased latency in a virtualized environment
- Workaround: Fewer network calls, ask for more data at a time.

Pain Points

Instances (or machines) go away sometimes or become so slow that you want to make them go away.

Workaround: Avoid single points of failure and make sure your servers have automated configuration.

Protip

Security was not the first thought when a lot of the cloud systems were designed

Make it your first thought though. A little planning goes a long way. Use security groups judiciously and keep those keys safe!

Protip

Keep track of those limits!

To prevent someone from consuming too much, all resources have per account limits. Keep track of them and get them raised ahead of when you need them. Make sure to catch the exceptions too.

Mistake

- Relying on a single cloud product and expecting it to work as advertised

Bleeding edge in production

- Cassandra wasn't always perfect
- No data loss, but it was a pain sometimes

Automate all the things!

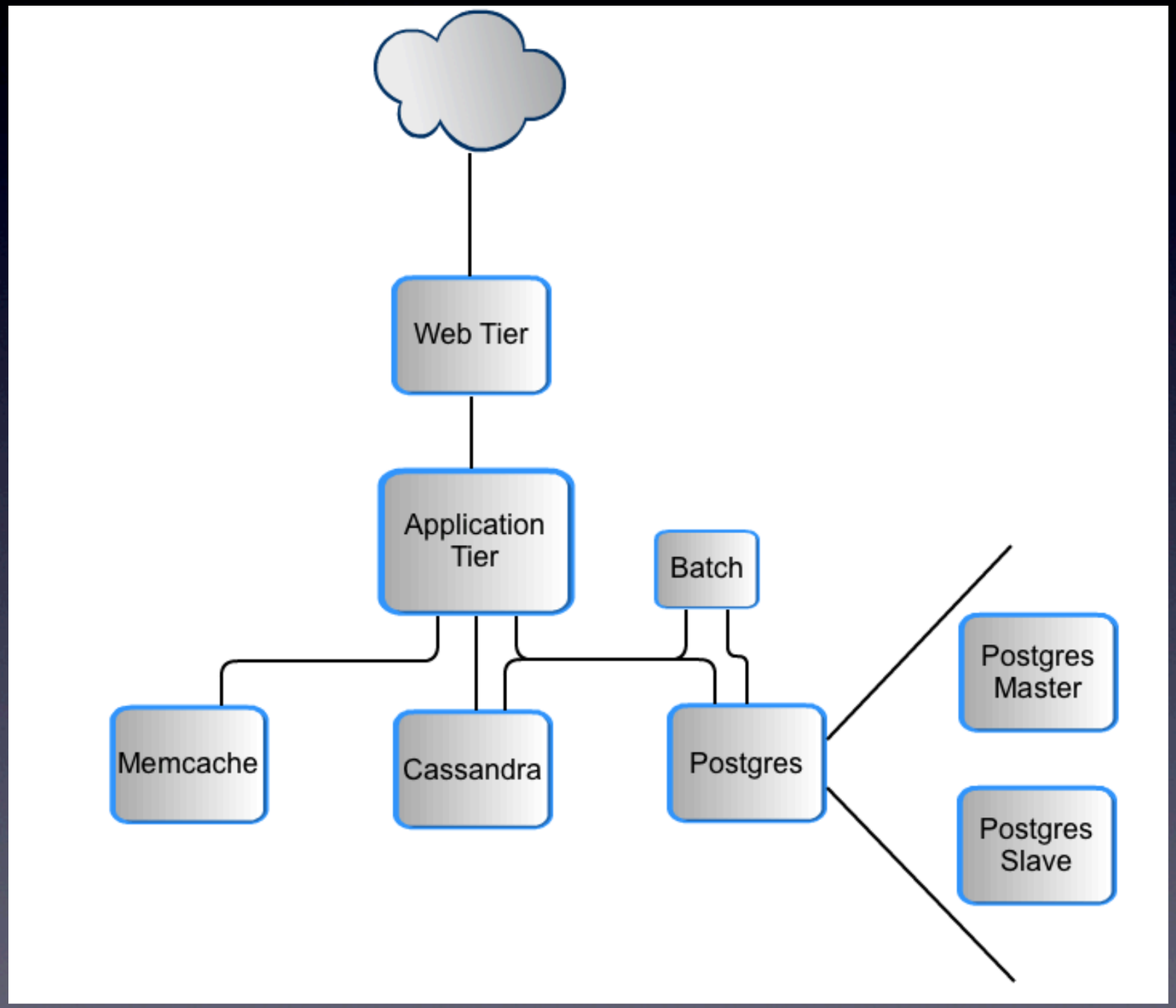
(Including Your Infrastructure)

http://hyperboleandahalf.blogspot.com/2010/06/this-is-why-ill-never-be-adult.html

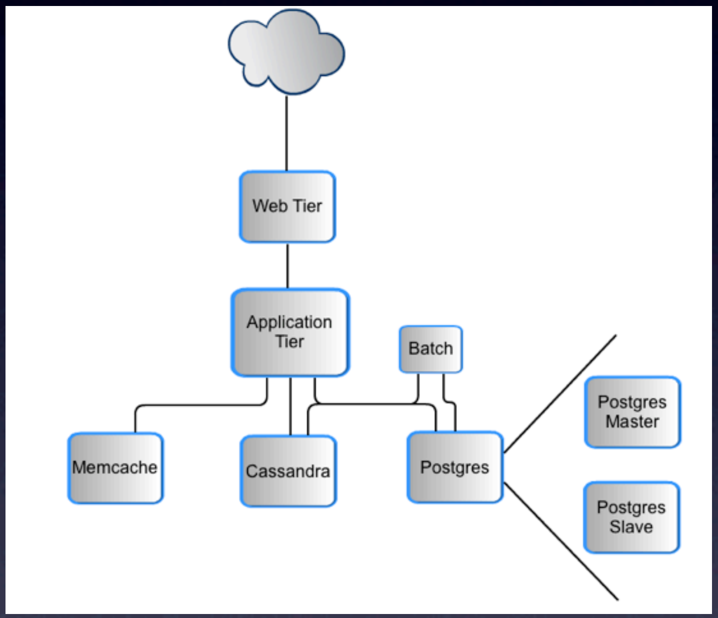
Architecture

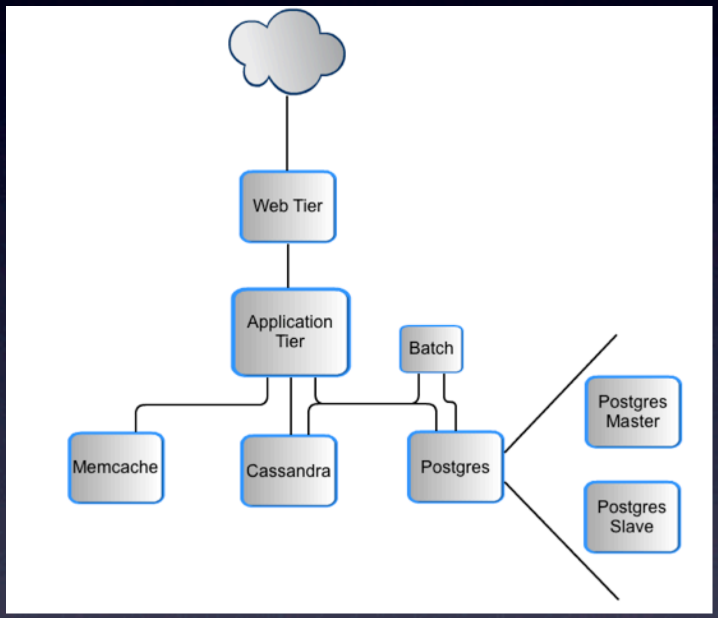
Is it necessary to build a scalable architecture from the beginning?

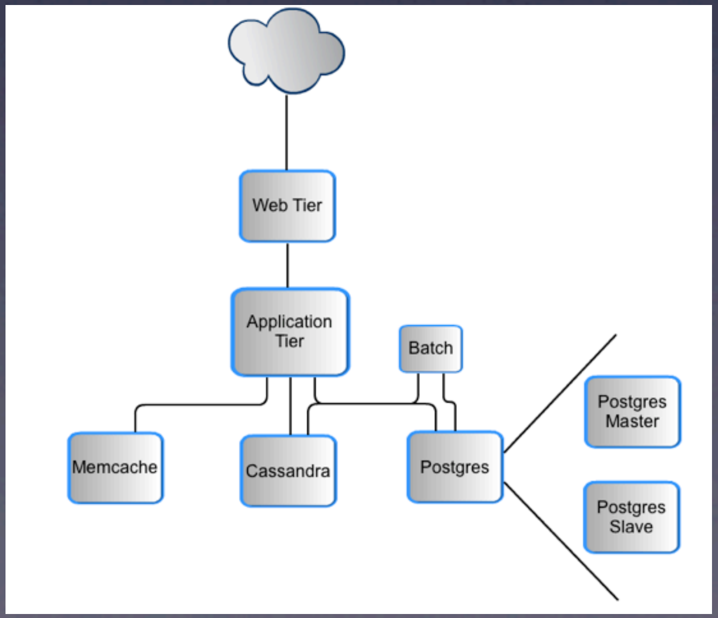
Example 1 -- reddit

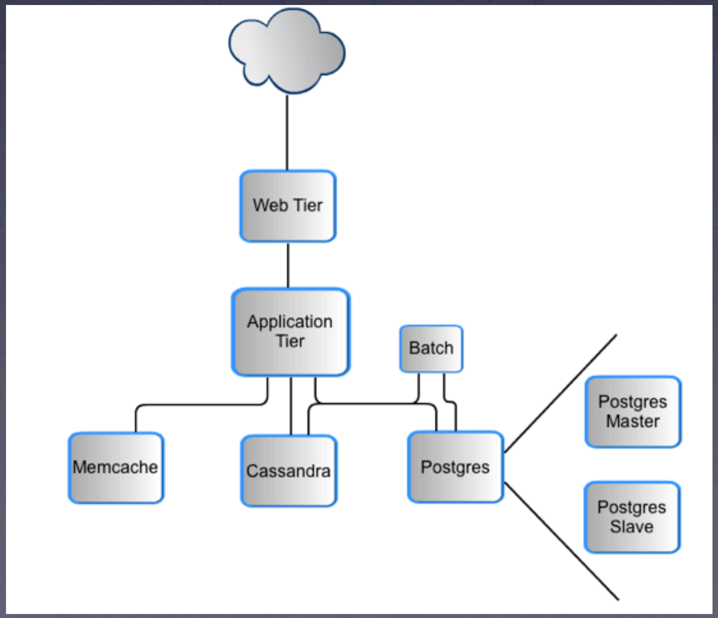


Example 2 -- Netflix









2B requests per day into the Netflix API

API

12B outbound requests per day to API dependencies

Personaliz a-tion Engine

User Info

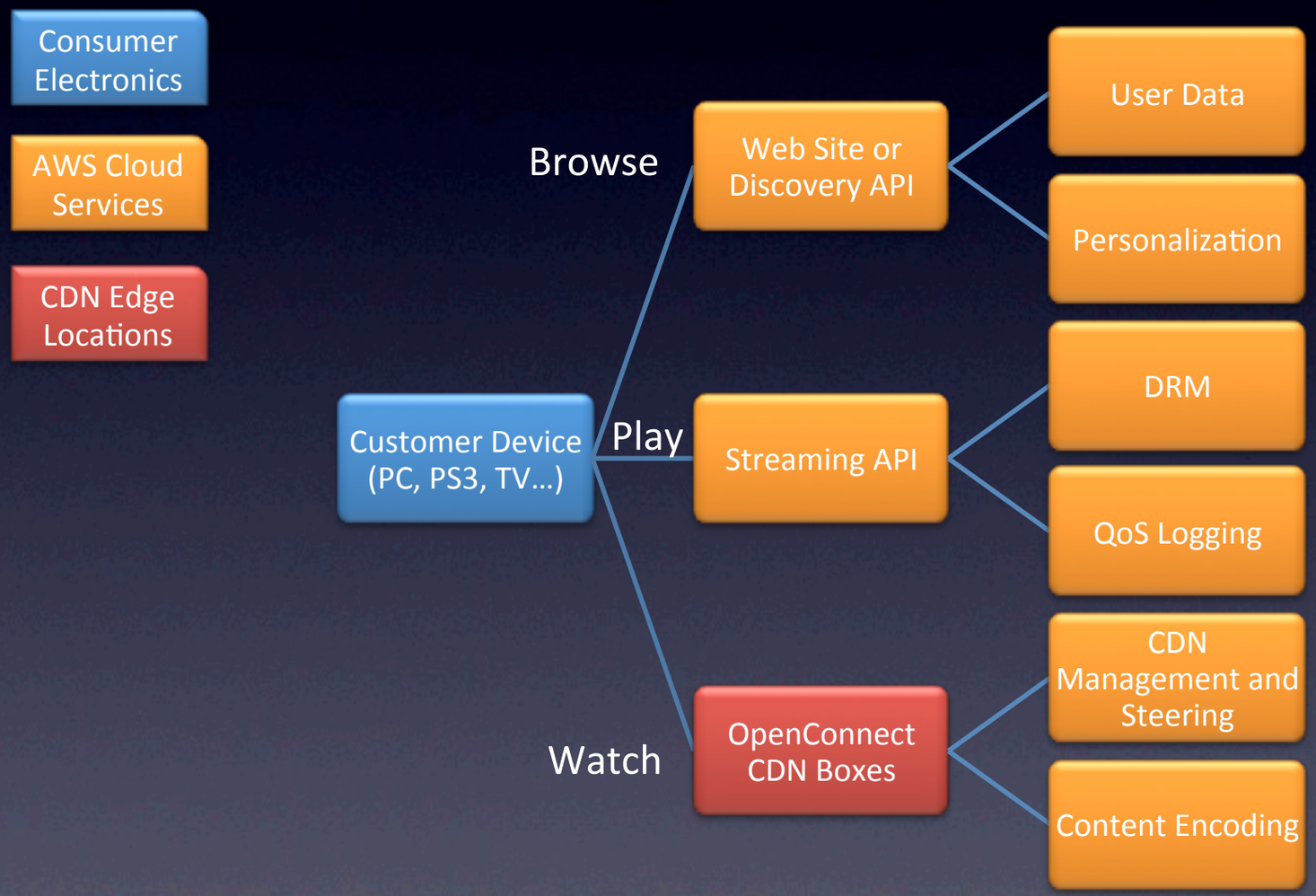
Movie Metadat

Movie Ratings

Similar Movies

Reviews

A/B Test Engine


Consumer Electronics
AWS Cloud Services
CDN Edge Locations
Browse
Web Site or Discovery API
User Data
Personalization
Customer Device (PC, PS3, TV…)
Play
Streaming API
DRM
QoS Logging
Watch
OpenConnect CDN Boxes
CDN Management and Steering
Content Encoding

Advantages to a Service Oriented Architecture

- Easier auto-scaling
- Easier capacity planning
- Identify problematic code-paths more easily
- Narrow in the effects of a change
- More efficient local caching

Disadvantages to a Service Oriented Architecture

- Need multiple dev teams, or need people to work on multiple services.
- Need to come up with a common platform, otherwise work will be duplicated.
- Too much overhead for a small team just starting out.

Netflix built a global PaaS

- Service Oriented Architecture
- HTTP/Rest interfaces between services

Netflix PaaS features

- Supports all regions and zones
- Multiple accounts
- Cross region/account replication
- Internationalized, localized and GeoIP routed
- Advanced key management
- Autoscaling with 1000s of instances
- Monitoring and alerting on millions of metrics

What AWS Provides

- Instances
- Machine Images
- Elastic IPs
- Load Balancers
- Security groups / Autoscaling groups
- Availability zones and regions

What Netflix provides

- Applications
- Clusters
- Discovery services
- Application routing
- Monitoring

Instance Architecture

Linux Base AMI (CentOS or Ubuntu)

Optional Apache

Monitoring

Log Rotation to S3

Appdynamics Machine Agent

Java (JDK 6 or 7)

Appdynamics App Agent

monitoring

GC and thread dump logging

Tomcat

Application war file, base servlet, platform, interface jars for dependent services

Healthcheck, status servelets, JMX interface, Servo autoscale

Instance Architecture

Linux Base AMI (CentOS or Ubuntu)

Optional Apache

Monitoring

Log Rotation to S3

Appdynamics Machine Agent

Python 2.7

monitoring

Exception logging

Django

Application file, base server, platform, interface libs for dependent services

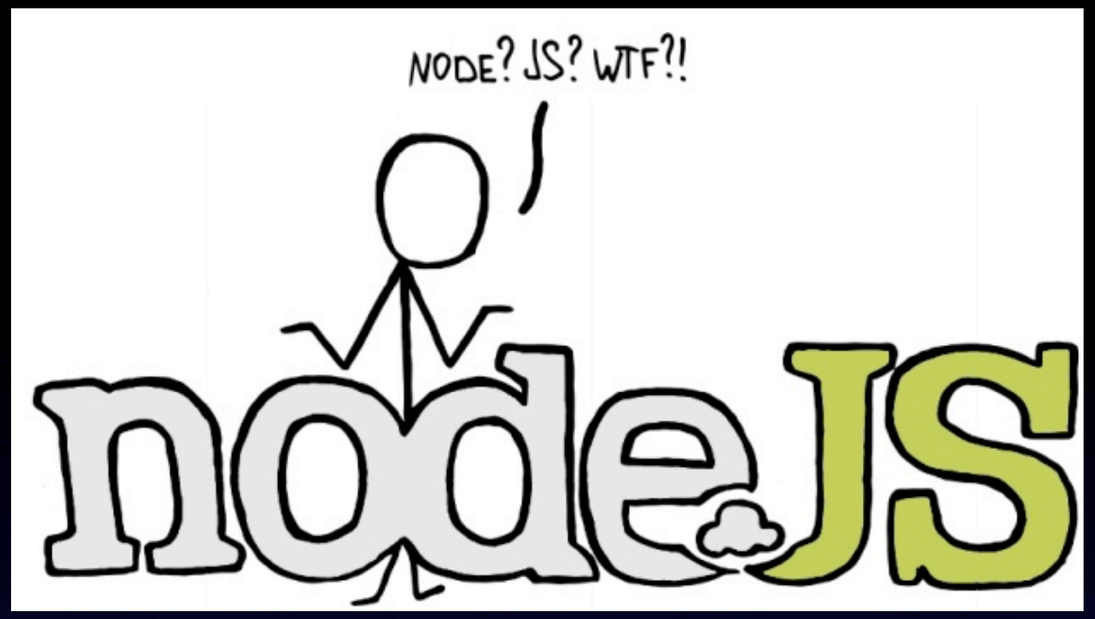
Freedom and Responsibility

- Developers deploy when they want to
- They also manage their own capacity and autoscaling
- And fix anything that breaks at 4am!

“Build for three”

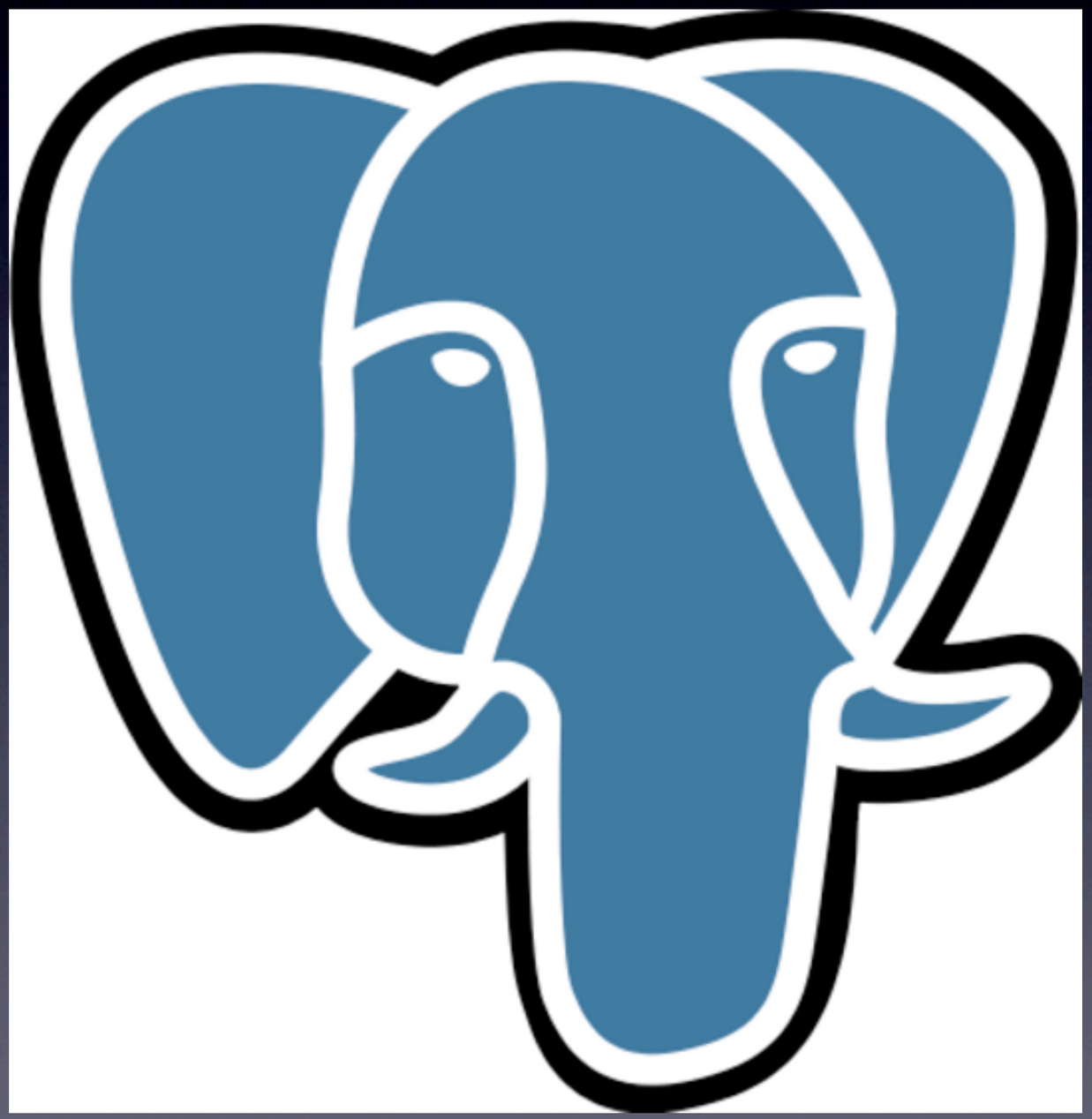
We hold a boot camp every 5-8 weeks for new engineers to teach them how to build for a highly distributed environment.

All systems choices assume some part will fail at some point.



Don’t follow fads

Postgres is still a good database



Offload to the client with Javascript

What else do you need to worry about?

- Queues
- Locking service (can you avoid the locks?)
- Email (outsource it: deliverability is a pain)
- ???

Limits everywhere!

- Put a limit on everything.
- Make it really really high.
- Lower it or raise it as needed

1 > 2 > 3

Going from two to three is hard

1 > 2 > 3

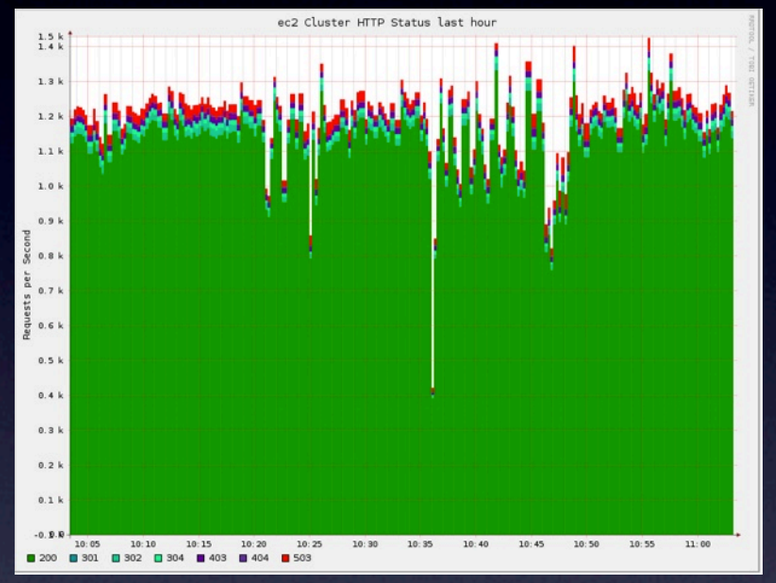
Going from one to two is harder

1 > 2 > 3

If possible, plan for 3 or more from the beginning.

Monitoring

- We used Ganglia at reddit
- Backed by RRD
- Makes good rollup graphs



- Gives a great way to visually detect errors
- Wasn't friendly to rapidly changing infrastructure.

Mistake

- Not having enough monitoring and using a system that isn't "virtualization friendly".

Reliability and $$

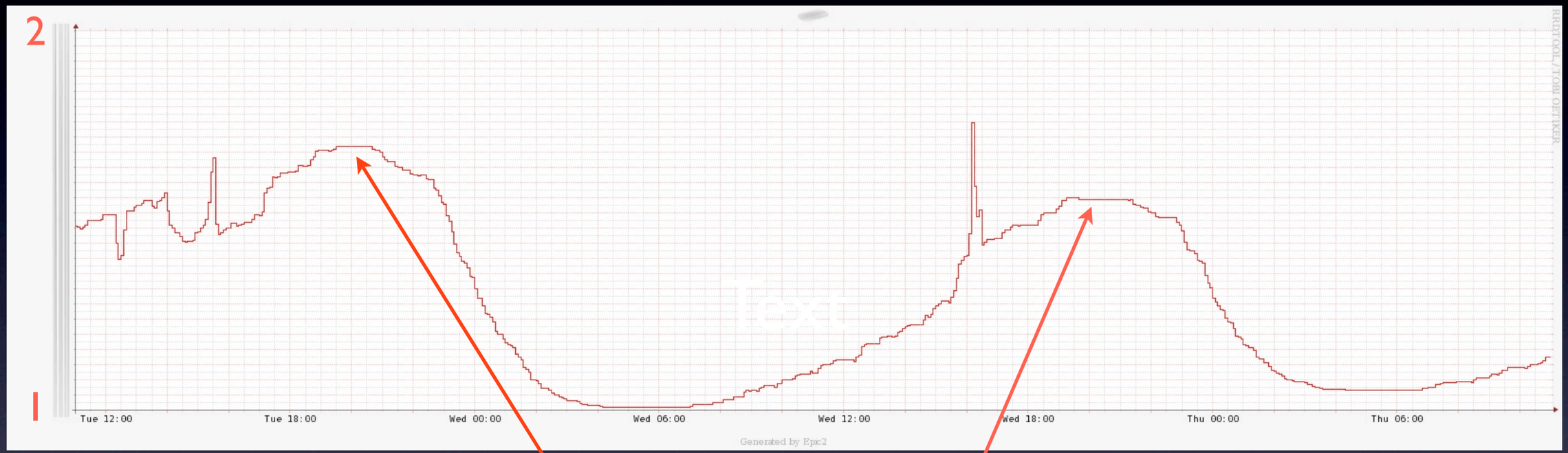
The Monkey Theory

- Simulate things that go wrong
- Find things that are different

The simian army

- **Chaos -- Kills random instances**
- **Latency -- Slows the network down**
- Conformity -- Looks for outliers
- Doctor -- Looks for passing health checks
- Janitor -- Cleans up unused resources
- Howler -- Yells about bad things

Netflix autoscaling



Traffic Peak

Automate all the things!

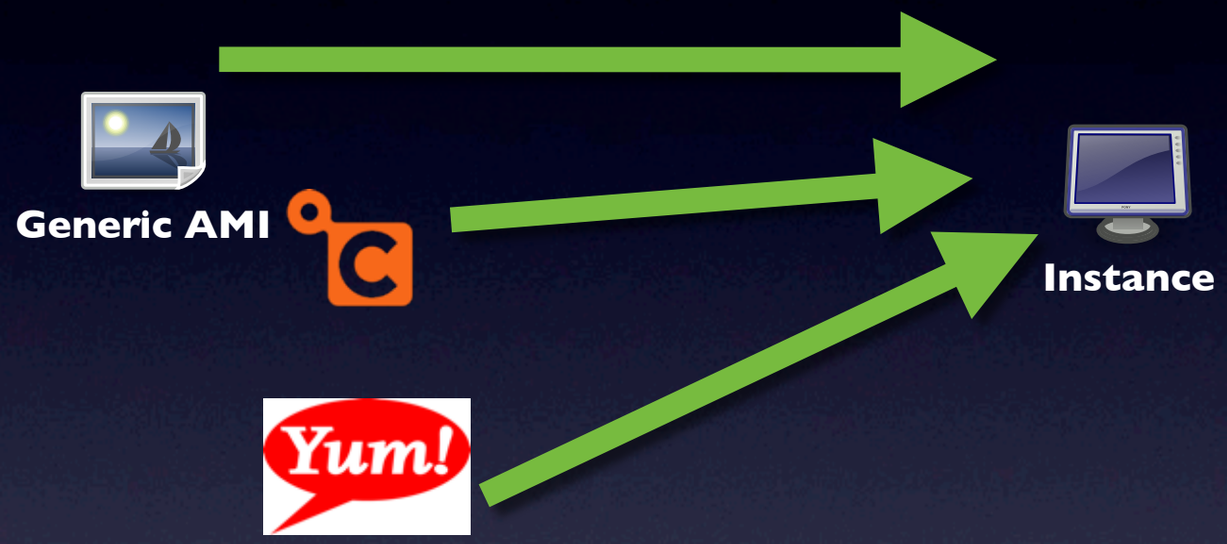
Automate all the things!

- Application startup
- Configuration
- Code deployment
- **System deployment**

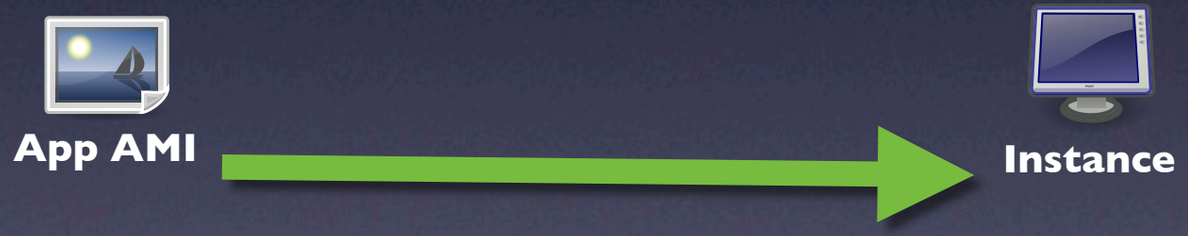
Netflix has moved the granularity from the instance to the cluster

Why Bake?

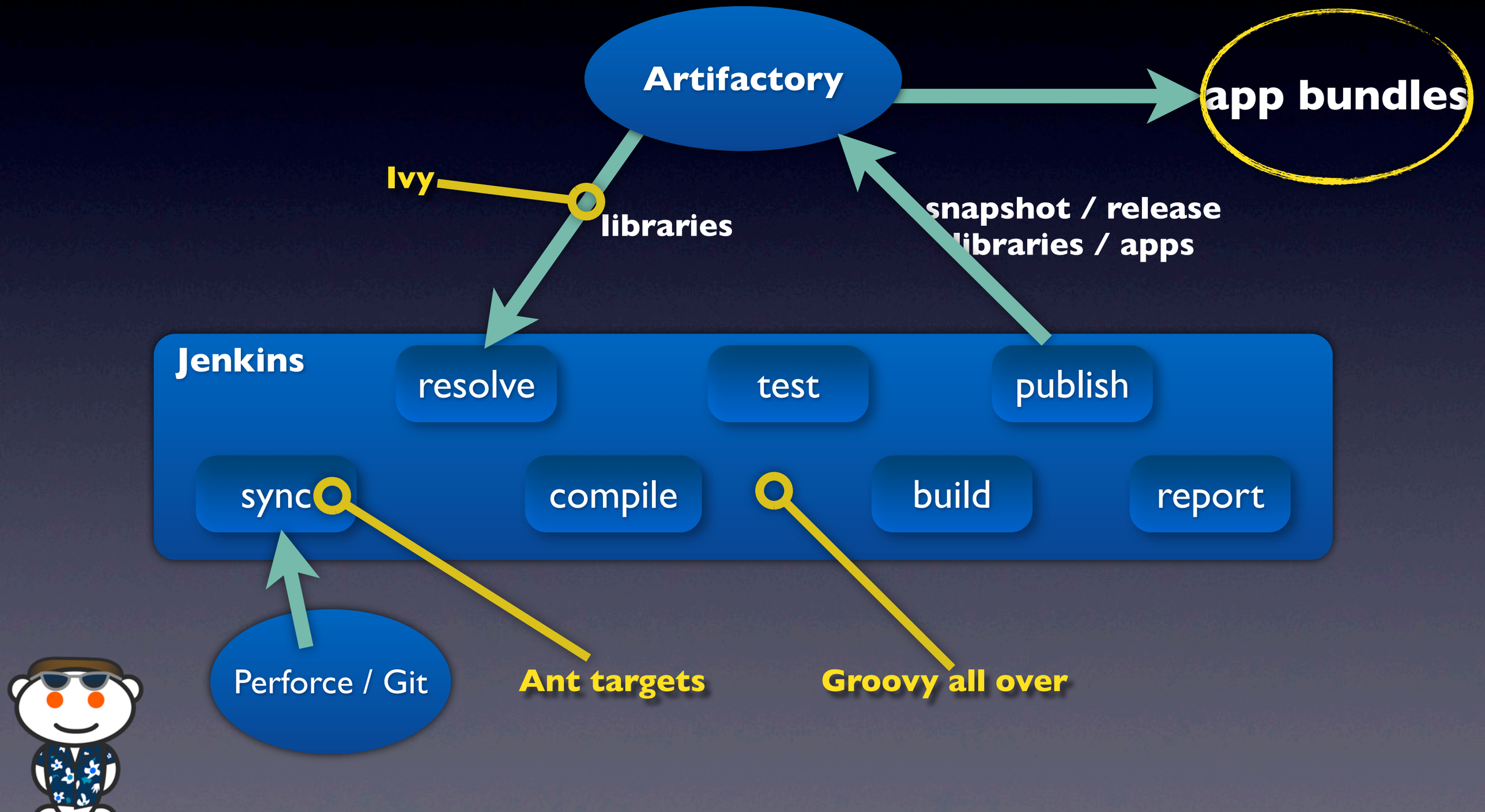
Traditional:
•launch OS
•install packages
•install app



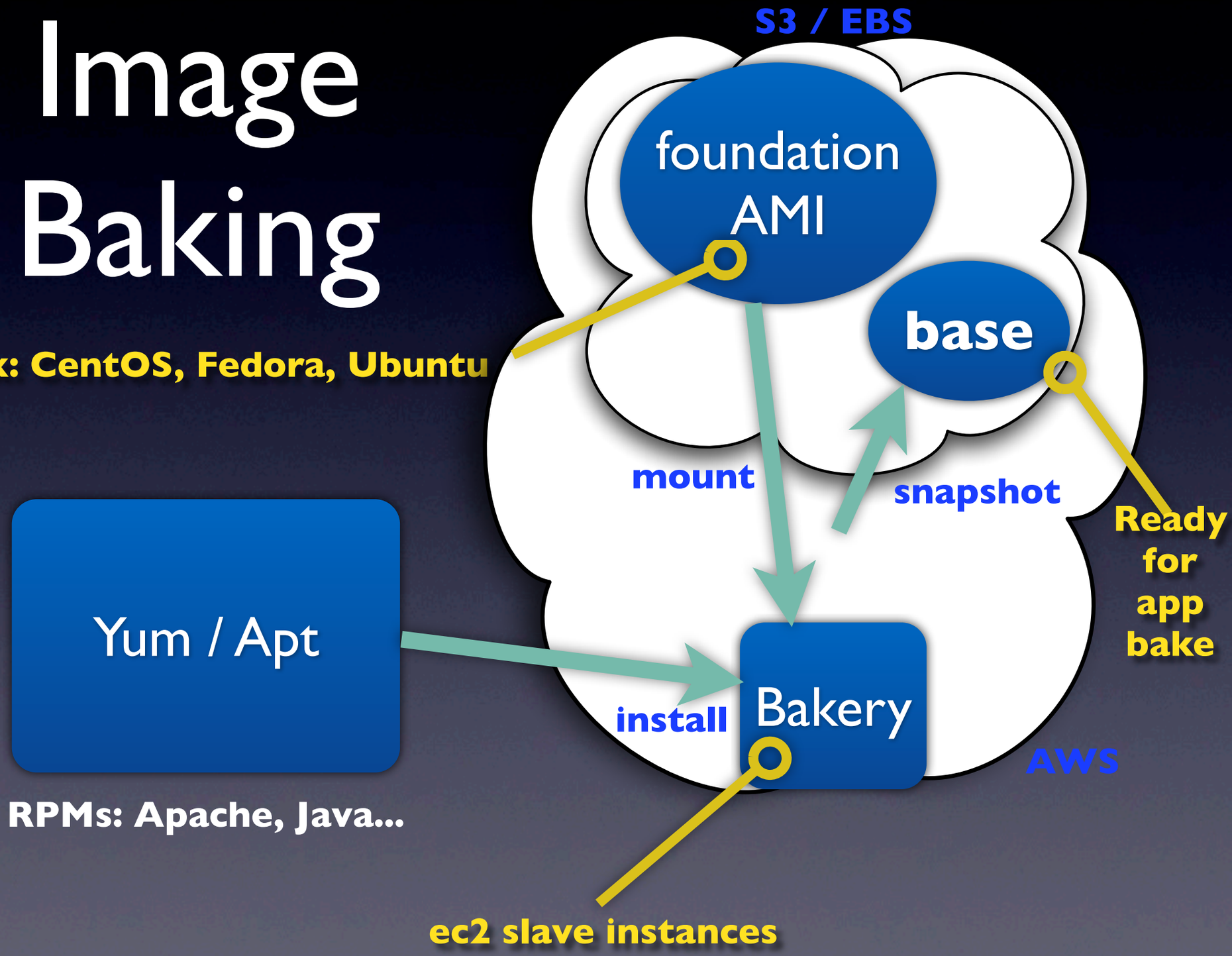
Netflix:
•launch OS+app



Getting Baked



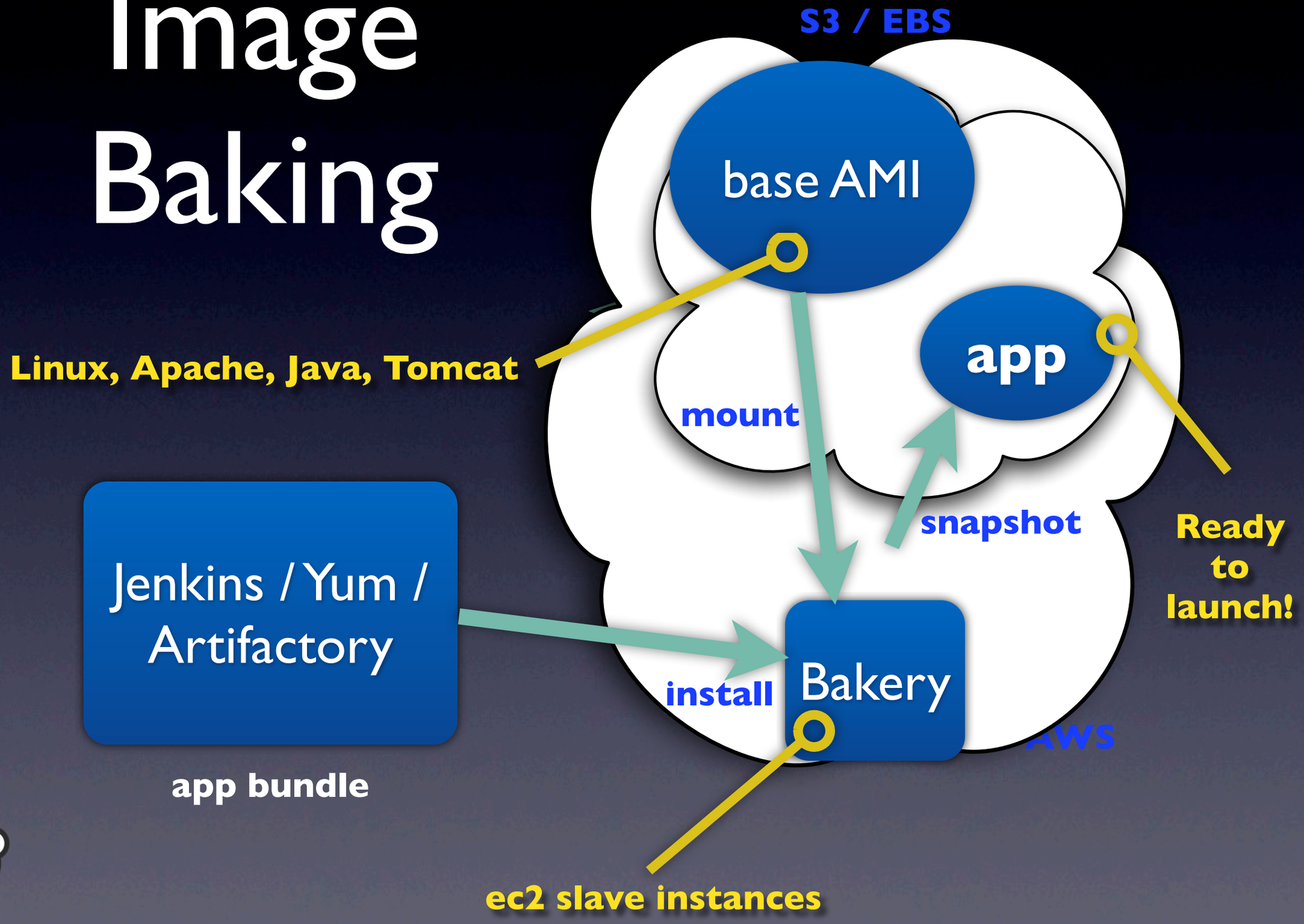
Base Image Baking



Linux: CentOS, Fedora, Ubuntu

RPMs: Apache, Java...

App Image Baking



Code

Picking a framework

You must construct additional Pylons

Scaling Pylons

- pylons scaling == python scaling
- run lots of appservers and make them independent of each other
- We built our own caching
- We built our own database layer

Would I use Pylons again?

Yes (although it's called Pyramid now)

C is faster than Python (sorry)

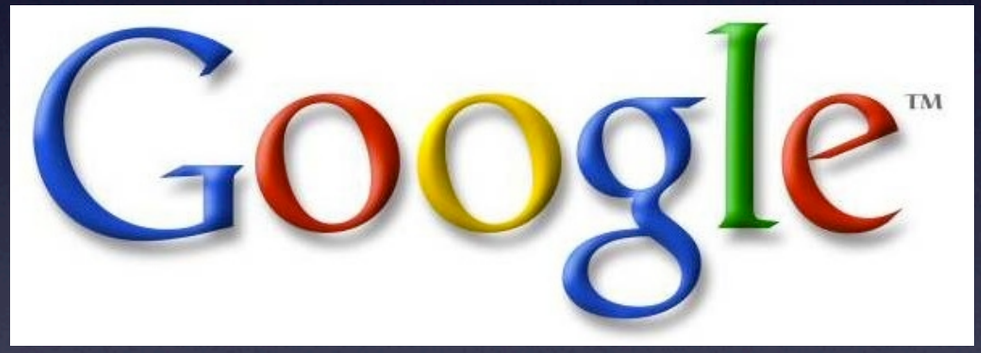
filters
discount (markdown)
memcache

Open Source is Good

Data

Data is the most important asset your business will have.

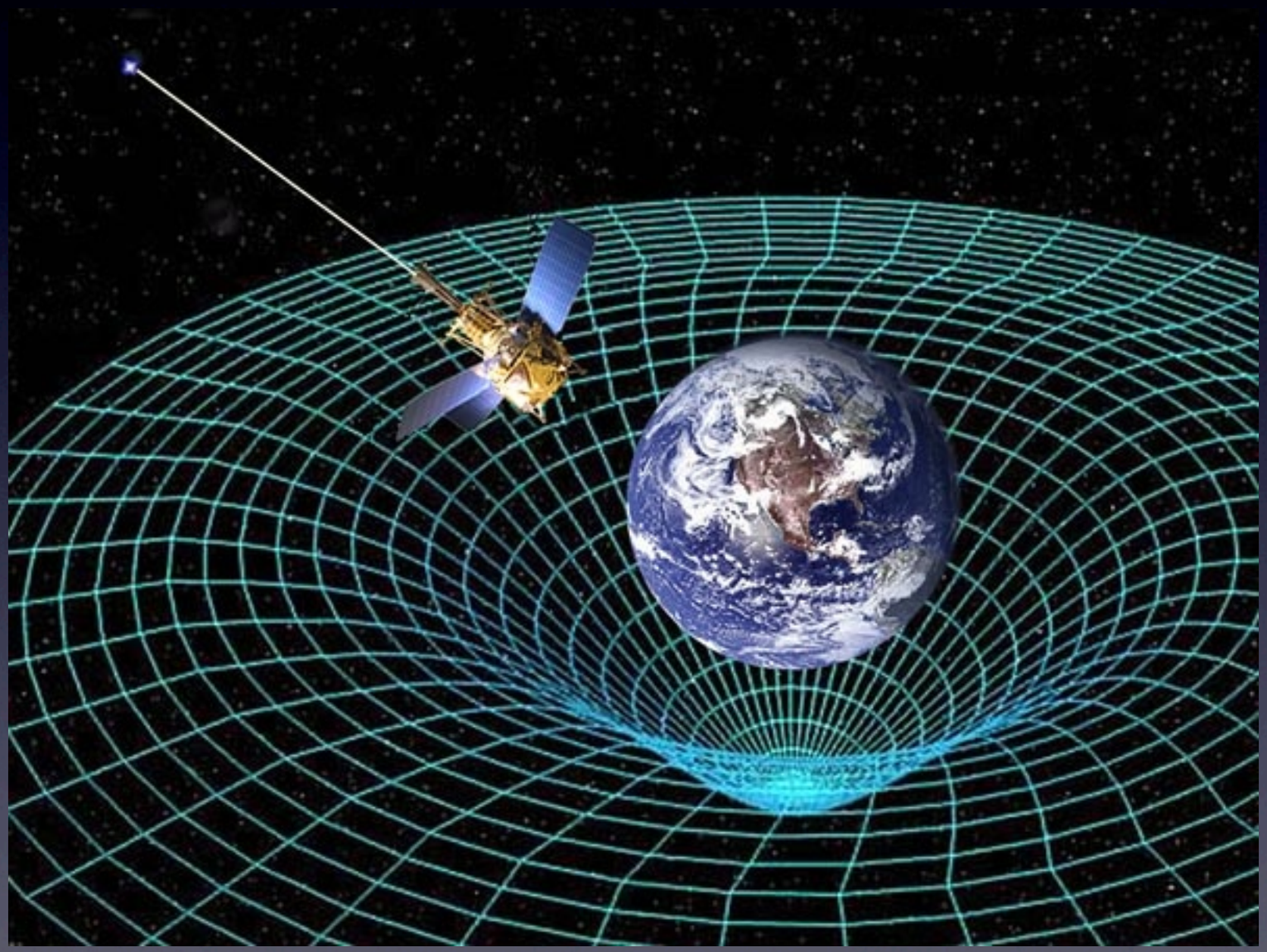

flickr™


Google™

Data Gravity

- Coined by Dave McCrory
- First described here: http://blog.mccrory.me/2010/12/07/data-gravity-in-the-clouds/

What is Data Gravity?



Source: nationalgeographic.com

Data Gravity and you

- The bigger your dataset, the harder it is to move from anywhere to anywhere
- Also, how do you move that data without affecting your running application?

reddit's data gravity problem

- We had a lot of data that was ever-growing
- We were so resource constrained we couldn't move it without hurting our application

Sql or “nosql”?

Relational vs. Non-relational

Mysql, Postgres or something else?

Data schemas

- Unless you are really really sure of your business model...
- The less schema the better
- reddit's database is literally just keys and values

Expire your data

- It's a lot easier to manage if your data is either gone or in static form
- Users will almost never notice

More Transactions Would Be Good

- Since reddit's data is spread across two tables for each thing, we didn't use sql transactions
- We should probably have made more transactions in Python

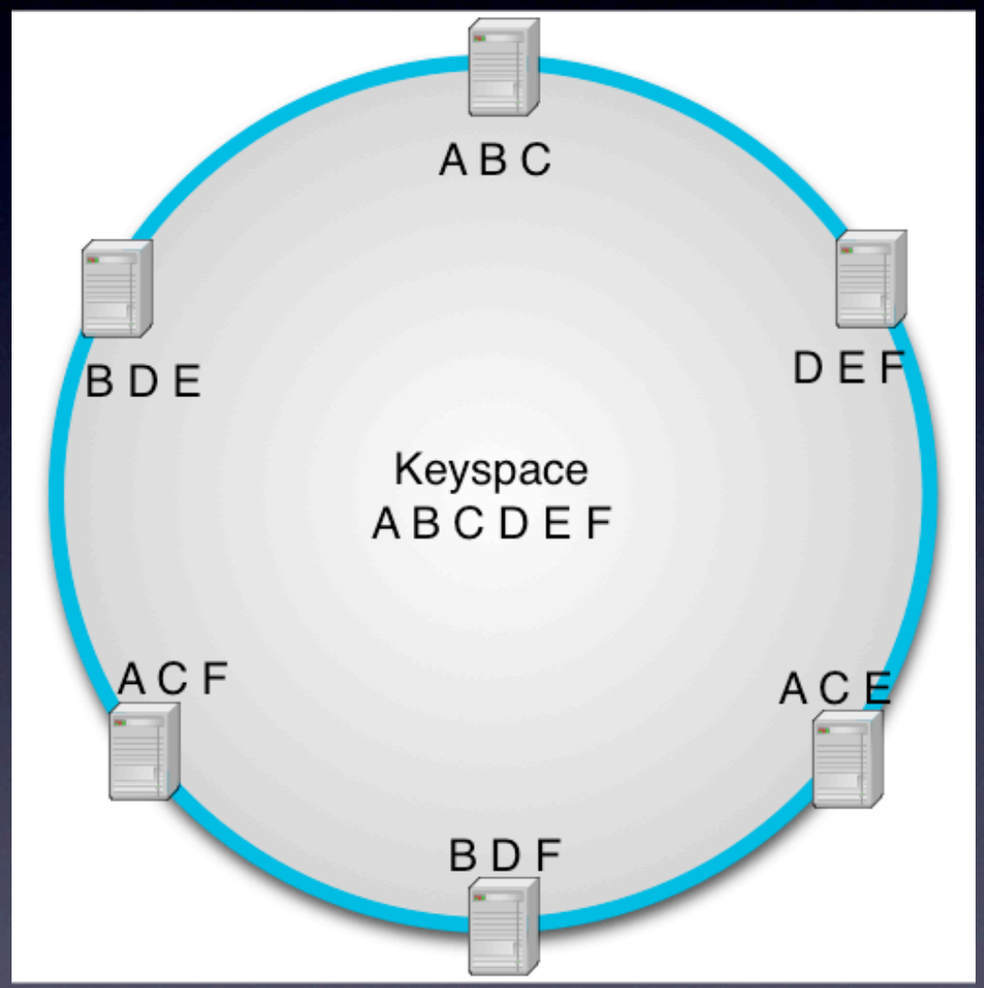
Think of SSDs as cheap RAM, not expensive disk

Database Scaling with Sharding

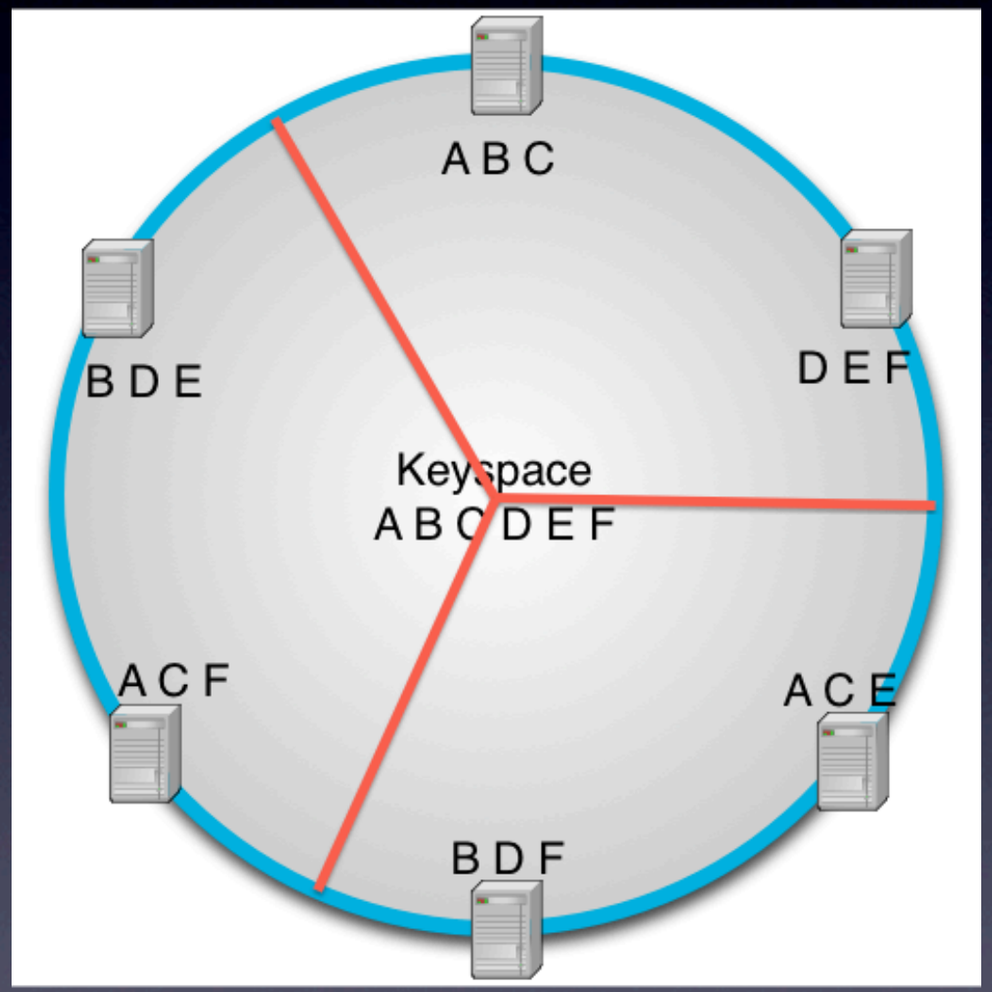
Sharding

- We split our writes across four master databases
- Links/Accounts/Subreddits, Comments, Votes and Misc
- Each has at least one slave
- We avoid reading from the master if possible
- Wrote our own database access layer, called the “thing” layer

Cassandra



Cassandra Architecture



How it works

- Replication factor
- Quorum reads / writes
- Bloom Filter for fast negative lookups
- Immutable files for fast writes
- Seed nodes

Why Cassandra?

- Fast writes
- Fast negative lookups
- Easy incremental scalability
- Distributed -- No SPoF

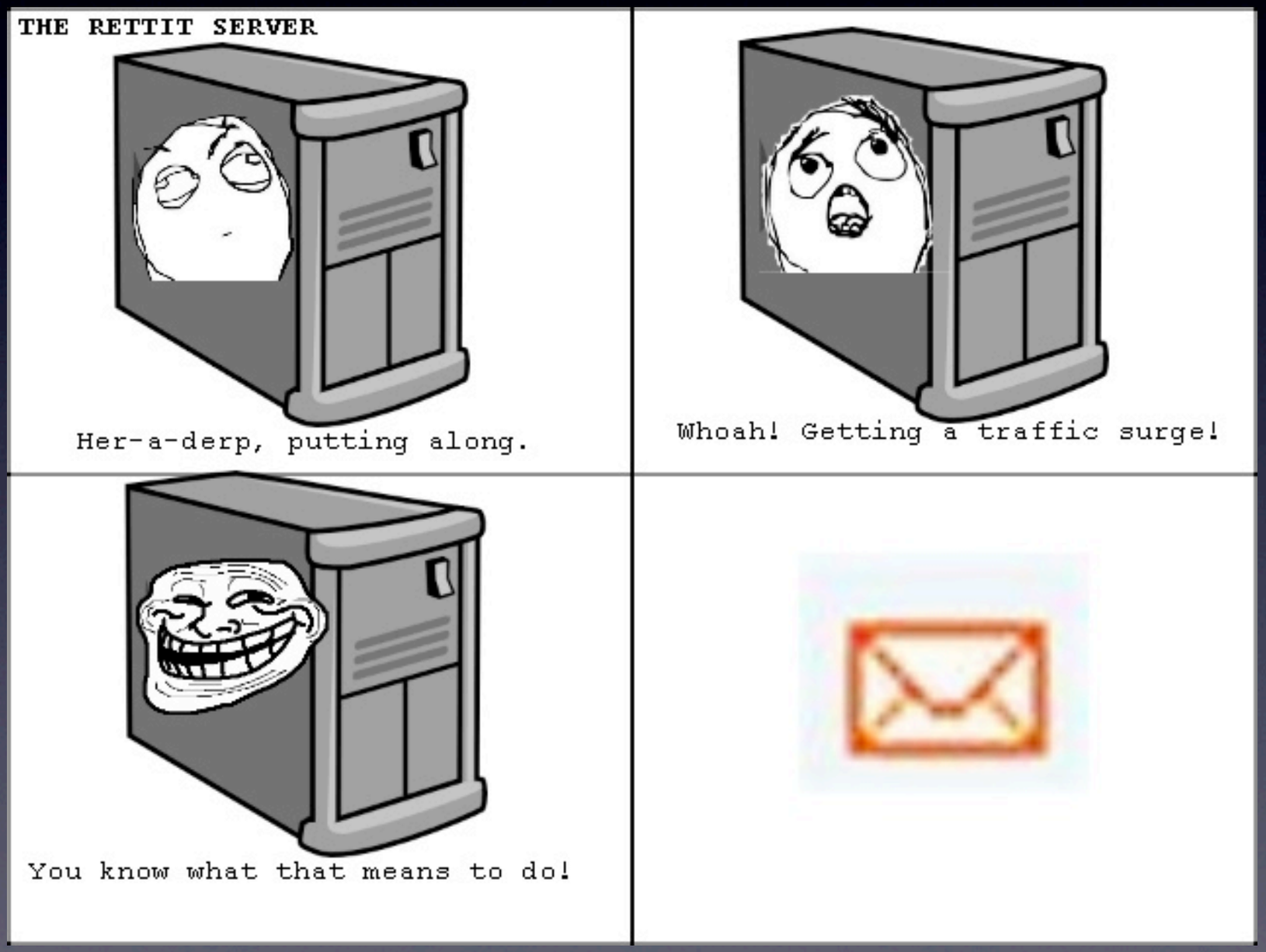
Second class users

- Logged out users **always** get cached content.
- Akamai bears the brunt of reddit's traffic
- Logged out users are about 80% of the traffic

Queues are your friend

- Votes
- Comments
- Thumbnail scraper
- Precomputed queries
- Spam
  - processing
  - corrections

Sometimes users notice your data inconstancy



Mistake

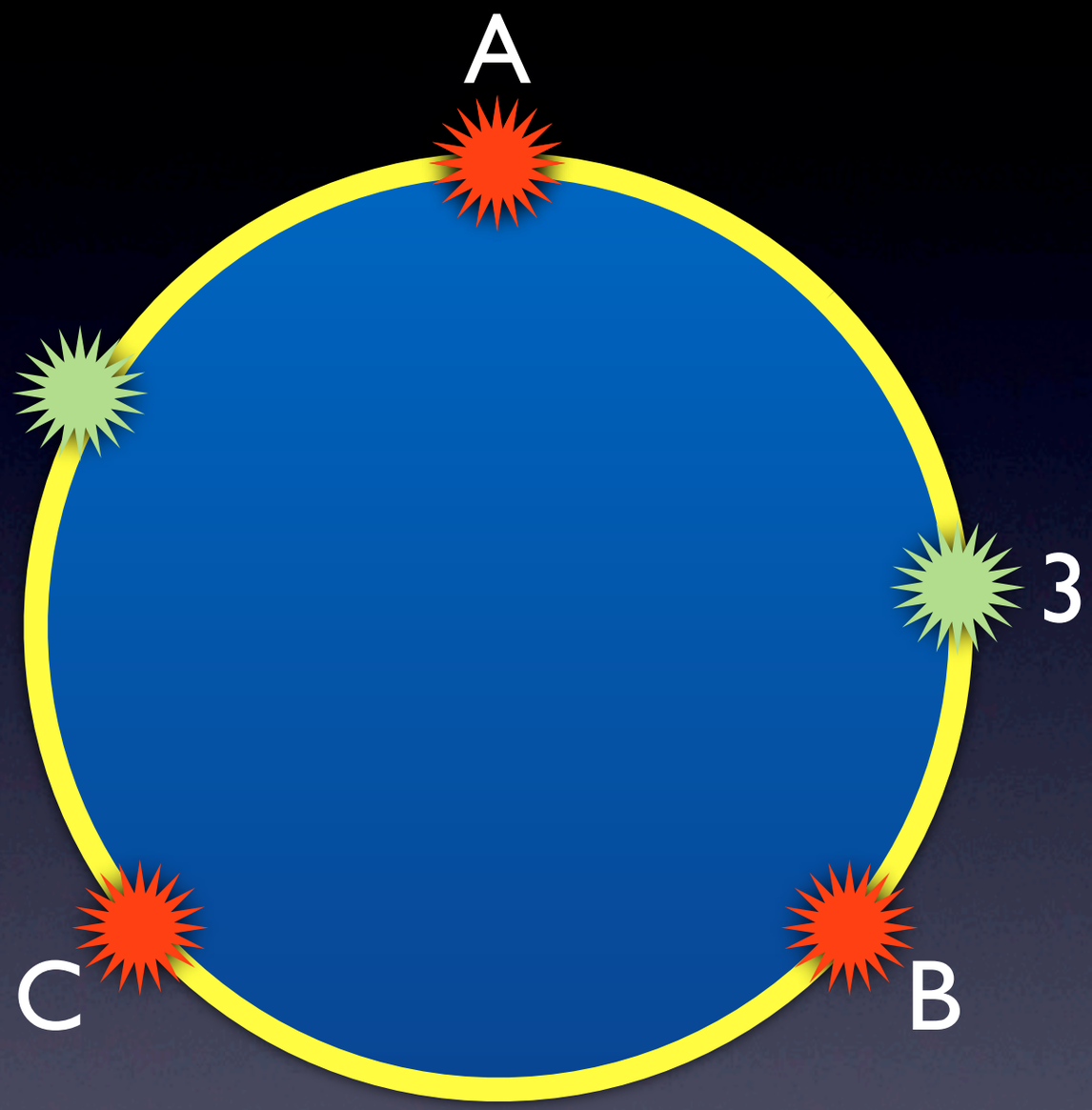
- Not using a consistent key hashing algorithm at first.

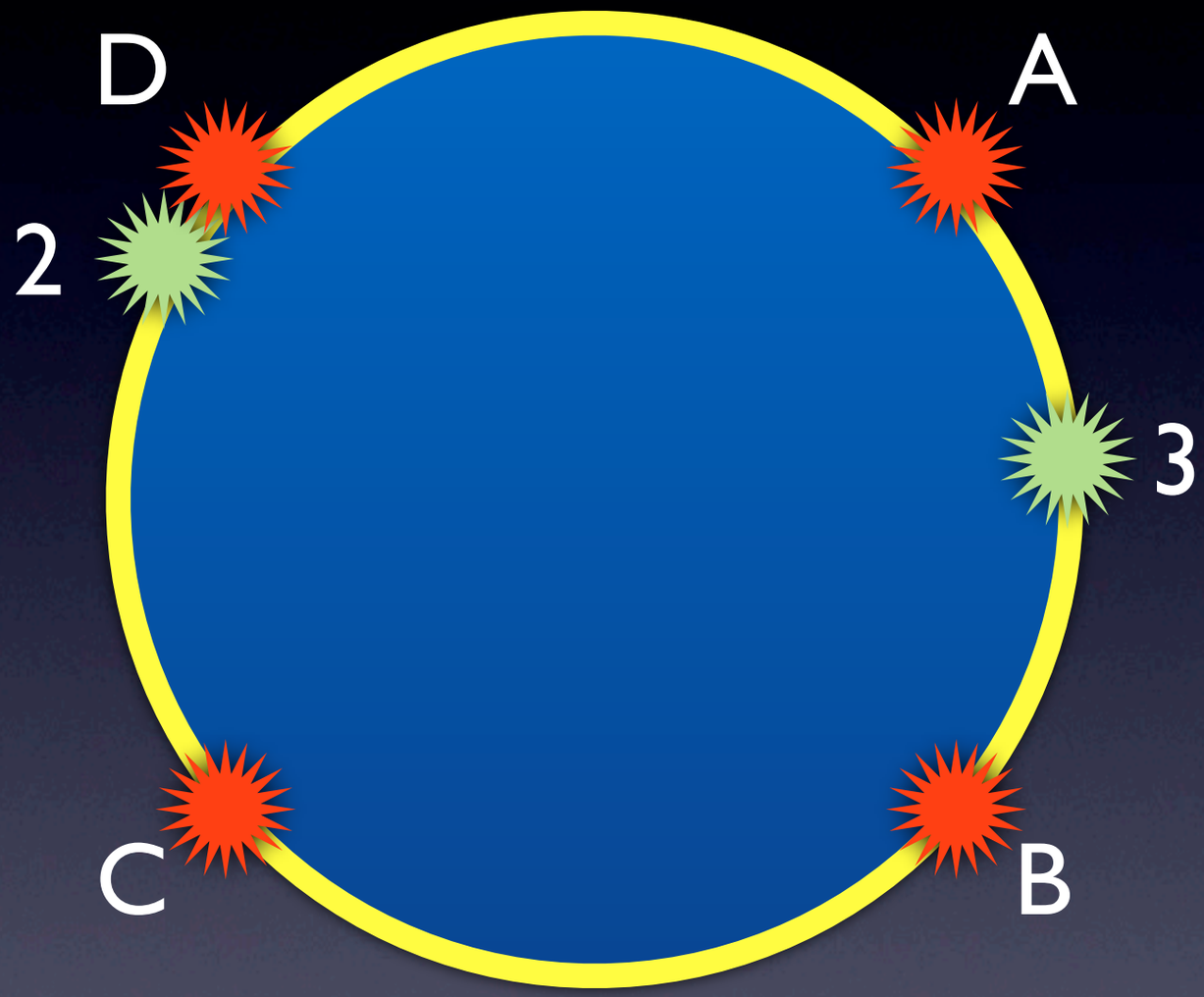
Memcachedb

- Using md5'd keys made it difficult to rebalance.
- It didn't really have a way to rebalance
- Turns out it was pretty slow under high workloads

Solutions

- We moved to using a consistent key hashing for memcache
- We moved to Cassandra, which follows the Dynamo model, which uses a type of consistent hashing


A
2
3
C
B


D
A
2
3
C
B

Protip

The environment in a public cloud is inherently more variant (co-tenants, abusive or heavy users, etc)

Make sure your code is written to handle this -- state should be kept somewhere shared and redundant, not on the instance.

Best Practices

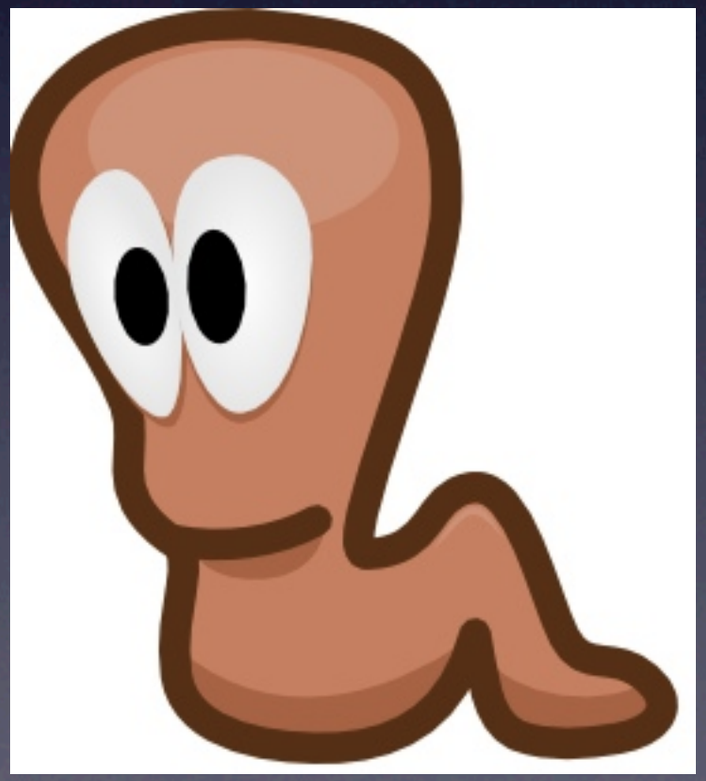
- Keep data in multiple Availability Zones
- Avoid keeping state on a single instance
- Take frequent snapshots of EBS disks
- No secret keys on the instance
- Different functions in different Security Groups

Social aspects of Growth


from
reddit
import
code

The Worm

Or, why you should never have your entire team on one airplane.



Provide an API

The business side of things

- Running a site that requires user input?
- Be one of the most active users
- People like to see the founders participate

Moderation, cheating, spam and fraud

- If you take user input, and get popular, people will cheat and spam.
- If you take money, they will scam people.
- Limits will help a lot, as will pattern detection.
- Hard coded rules only go so far -- you need learning algorithms.
- Let your users do the work for you.

What made reddit successful?

- Empowered users
- Better software
- Community interaction

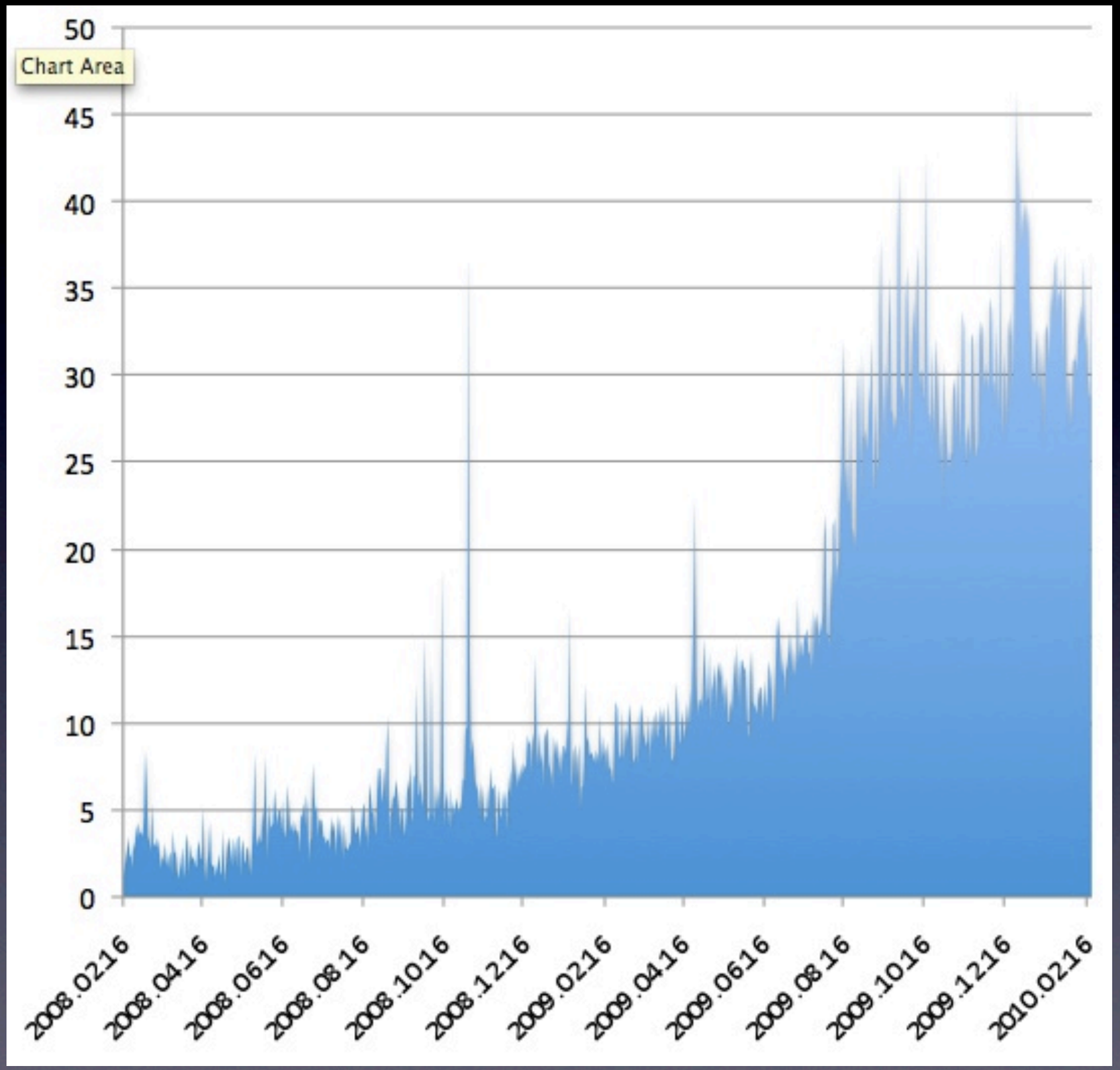


Chart Area
50
45
40
35
30
25
20
15
10
5
0
2008.0216
2008.0416
2008.0616
2008.0816
2008.1016
2008.1216
2009.0216
2009.0416
2009.0616
2009.0816
2009.1016
2009.1216
2010.0216

How does reddit make money?

- Sidebox ads
- Self-serve ads
- Merchandise
- reddit gold
- marketplace

reddit Gold

Ask Me Anything

- Not only did I run technology for reddit but I also was deeply involved in the business.
- Ask me anything about running a profitable social media company.

Getting in touch

Email: jedberg@{gmail,netflix}.com

Twitter: @jedberg

Web: www.jedberg.net

Facebook: facebook.com/jedberg

Linkedin: www.linkedin.com/in/jedberg