From a Single Server to the Cloud

Modernizing a Large Fan Website

@codelemur
www.robpeck.com

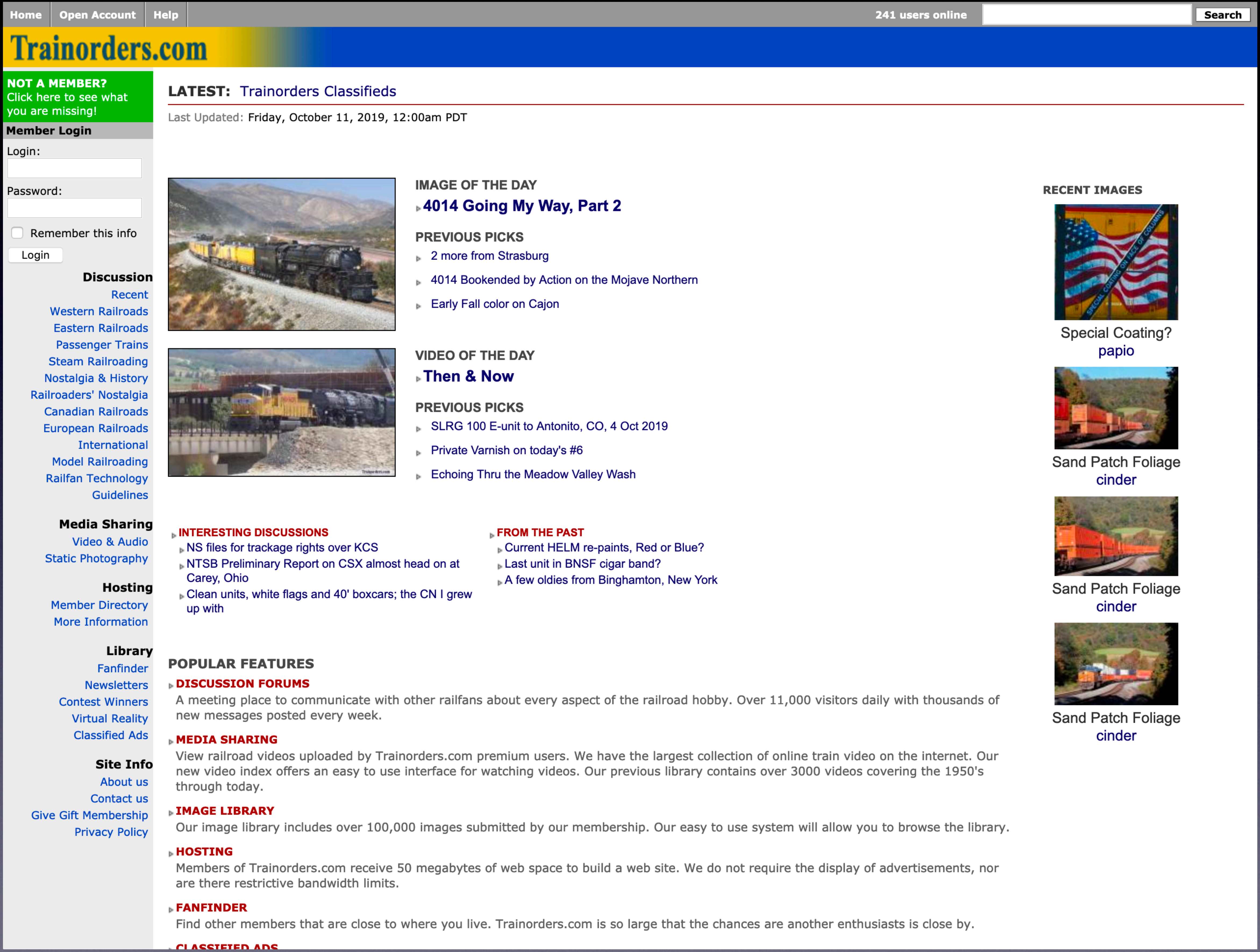
Devspace 2019
Huntsville, Ala.

About Me

- I’m a technical lead at DealNews, and I occasionally take side gigs.
    - DealNews is hiring! dealnews.com/jobs
- I’ve been doing software development somewhat professionally for 20 years.

The Client

- Trainorders.com is one of the largest sites on the Internet for train fans (“railfans”).
- Was founded in 1997 by Todd Clark from California, initially as a home for a motion activated train webcam.
- Was purchased by Mark Cuban in 1998, then by Yahoo! later that year. Todd bought it back in 2000 after the crash and has owned it ever since.



Home | Open Account | Help

241 users online | Search

Trainorders.com

NOT A MEMBER?
Click here to see what you are missing!

Member Login

Login:

Password:

Remember this info

Login

Discussion
- Recent
- Western Railroads
- Eastern Railroads
- Passenger Trains
- Steam Railroading
- Nostalgia & History
- Railroaders' Nostalgia
- Canadian Railroads
- European Railroads
- International
- Model Railroading
- Railfan Technology
- Guidelines

Media Sharing
- Video & Audio
- Static Photography

Hosting
- Member Directory
- More Information

Library
- Fanfinder
- Newsletters
- Contest Winners
- Virtual Reality
- Classified Ads

Site Info
- About us
- Contact us
- Give Gift Membership
- Privacy Policy

LATEST: Trainorders Classifieds

Last Updated: Friday, October 11, 2019, 12:00am PDT

IMAGE OF THE DAY

4014 Going My Way, Part 2

PREVIOUS PICKS
- 2 more from Strasburg
- 4014 Bookended by Action on the Mojave Northern
- Early Fall color on Cajon

VIDEO OF THE DAY

Then & Now

PREVIOUS PICKS
- SLRG 100 E-unit to Antonito, CO, 4 Oct 2019
- Private Varnish on today's #6
- Echoing Thru the Meadow Valley Wash

RECENT IMAGES

Special Coating?
papio

Sand Patch Foliage
cinder

Sand Patch Foliage
cinder

Sand Patch Foliage
cinder

INTERESTING DISCUSSIONS
- NS files for trackage rights over KCS
- NTSB Preliminary Report on CSX almost head on at Carey, Ohio
- Clean units, white flags and 40' boxcars; the CN I grew up with

FROM THE PAST
- Current HELM re-paints, Red or Blue?
- Last unit in BNSF cigar band?
- A few oldies from Binghamton, New York

POPULAR FEATURES

DISCUSSION FORUMS
A meeting place to communicate with other railfans about every aspect of the railroad hobby. Over 11,000 visitors daily with thousands of new messages posted every week.

MEDIA SHARING
View railroad videos uploaded by Trainorders.com premium users. We have the largest collection of online train video on the internet. Our new video index offers an easy to use interface for watching videos. Our previous library contains over 3000 videos covering the 1950's through today.

IMAGE LIBRARY
Our image library includes over 100,000 images submitted by our membership. Our easy to use system will allow you to browse the library.

HOSTING
Members of Trainorders.com receive 50 megabytes of web space to build a web site. We do not require the display of advertisements, nor are there restrictive bandwidth limits.

FANFINDER
Find other members that are close to where you live. Trainorders.com is so large that the chances are another enthusiasts is close by.

CLASSIFIED ADS

Statistics

- In an average 24 hour period, trainorders has about:
  - 66GB of transfer traffic
  - 2,500 unique paid user accounts
  - 500 new uploads (images and videos) averaging 1.3 gigabytes a day.
  - 10 requests/sec (peaks at about 15).
- In the event of a rail-related news story, these numbers may double or triple!

My Involvement

- My involvement began in 2013 when Todd hired me for a side video project he was working on.
- We later shifted into preparing a new server for trainorders to run on.

Starting Point

- The state of the codebase in 2014 was complex.
- Standard LAMP with Apache/MySQL on a single server.
- Colo in a rack in a datacenter in Los Angeles.
- Much of the code was not in version control.
- Still running on PHP 5.2.

Starting Point

- Lots of custom stuff, odd Apache modules, Lua log parsing, etc.
- Servers were running Gentoo Linux.
- No monitoring other than looking at switch graphs or grepping logs.
- No analytics beyond Google.

Ending Point

- Everything is running on DigitalOcean.
- Multiple independent nodes with (some) redundancy.
- Easy to scale up to meet additional load.
- Observable.
- Modern PHP code in version control.

How did we get there?

2014

- Started the migration to dual beefy high-end servers, named Lark and Chief (after famous trains).

- Still running on a single server, but now with redundancy. Could fail completely over to another, or shift load from one to the other.

- Was still a very manual process. No replication, so machines would need to be resynced. We only did this once because it was bad.

2015

- Started upgrading the codebase to support PHP 5.5 and preparing for PHP 5.6.
- Large train crash in May pushed the architecture to an early limit.
  - Spent much of the day tweaking Apache to keep the site online and somewhat performant.

2016

- We also had a Denial of Service attack in 2016 that was “useful” in testing the limits of the single machine architecture.
- It was largely mitigated at the provider level, but we had to temporarily shift the database to the other machine and allow one to just be a web worker.
- This is obviously not optimal.

2017

Amtrak Cascades Derail

2017

- In December of 2017, the Amtrak *Cascades* derailed on the inaugural run over a new section of track.
- Three passengers were killed and 57 passengers and crew were injured.

2017

- In terms of the site, this pushed the old architecture to its absolute limit.
    - At one point we were approaching 40 requests per second, which was about 6x the average traffic at the time.
    - Had to turn off lots of things to keep the site up.

2017

- Had 500 Apache clients on the main machine (using prefork).
  - Repurposed an old machine as a load balancer using mod_proxy_balancer.
  - Brought backup machine online as a web worker with an additional 500 clients.
  - Started hitting connection limits on MySQL.

Conclusions So Far

- Traffic to a site like Trainorders is a lot like traffic to other news/social sites:
  - There is a base level of normal traffic that varies little day-to-day, but slowly grows over time.
  - Major events can result in major traffic spikes that the architecture needs to be able to handle.

A New Approach is Needed

The Decision

- A decision was needed:
  - We could either buy new hardware, which had been the approach in the past. Or…
  - We could migrate to a cloud provider, which would give us a lot more flexibility to scale individual areas when needed.

New Hardware

- Even to just upgrade the two main machines and duplicate the existing architecture on newer, faster machines with more storage was going to be about $14,000.
- Monthly colocation and bandwidth costs of about $600.

DigitalOcean

- No new hardware costs.
- Estimated monthly costs for:
  - 1 network services node
  - 1 MySQL node
  - 2 web nodes
  - 3TB block storage
  - Load balancer
  - Backups

$522.

Hard Decision There.

Location

- We selected the San Francisco data center, primarily because Todd outside Los Angeles and a large portion of the membership is on the west coast.
- Currently looking at an east coast location to better serve east coast members

Architecture

- The biggest initial problem was a lack of data.
  - How much bandwidth do we use over a month?
  - Logs were removed after 30 days
  - No observability in the app
  - Which parts needed to scale (how many nodes do we need?)

Architecture

- We did one month of detailed looking over logging.
- Wrote some scripts to pull things like load, web clients, DB connections, etc. on a minutely basis.
- Compiled into a CSV and used Excel to deduce some data.

Architecture

- Web nodes:
  - At least 2 needed at all times, with the ability to spin up more as demand required.
  - Went with 8GB RAM instances, more memory bound than disk needs.

Architecture

- Web nodes are behind a load balancer
- SSL termination is done at the load balancer
- This dramatically increased the throughput on the web nodes
  - SSL is expensive, let the load balancer do that, it does it quickly.

Architecture

- Database node:
  - Will also run sphinx for search
  - Single 16/320 instance
  - Not redundant, but backed up
  - Looking at managed database to replace this.

Architecture

- We planned a node dedicated to logging.
  - Observability is a first-class citizen.
  - Collecting and understanding logs in a multiple server world requires a different solution.
  - Went with ElasticSearch / Logstash / Kibana (the ELK stack)

Architecture

- A node for out-of-band work
  - Things like making thumbnails, sending email, cronjobs, etc.
  - Video processing moved to Zencoder, so we only need to generate thumbnails there too.
  - 1 gig node works fine here.
  - Frees up web nodes to serve pages.

The Asset Problem

- Trainorders had, at the time, nearly three TERABYTES of images and videos.
- How do we store and serve these?
- The application code assumes that the asset files are physically on the same server as the code, not designed for a multiple-server world.

The Asset Problem

- We used NFS to get around this problem in the past, but …
  - NFS is slow. It’s “okay” on the same network, but you are not guaranteed that in a cloud world.
  - Latency issues block web nodes from doing more productive things.

The Asset Problem

- We evaluated object storage. Technically the correct answer, but …
  - At the time it was still in beta.
  - Was not available in SFO2, so would have to be served from NYC!
  - Slow. WAY SLOW. Uploading to the tune of 500k/sec, would have taken weeks to transfer the whole library.

The Asset Problem

- The solution we came up with was to “kinda” use NFS.
  - Assets would live on a block storage volume attached to a single server, shared via NFS to other nodes.
  - Assets would be served directly from this machine.
  - NFS would only be used for writing and “file_exists” type of things.

The Asset Problem

- Used nginx to serve these, because it is fast
- It was terribly hacky solution, but it worked surprisingly well!
- It minimized the code changes that were necessary *at the time*.
- Remember the object storage thing, that will become important in just a little bit.

The Asset Problem

- But there was a problem!
  - The full-sized images and videos are only for paid users.
  - How do we securely serve these assets if they aren’t on the same server?

The Asset Problem

- The first thought was nginx’s “secure URL” functionality, but this is not optimal for a couple of reasons:
  - Not really “secure.” Anyone with the hashed URL can see it (such as if it’s shared).
  - URLs need to be generated per page, defeating application caching.
  - URLs change, defeating browser caching.

The Asset Problem

- What did we do? Cookies!
  - We set cookies on login.
  - The asset server reads them, compares the hashed values to verify the user is current and either serves or declines.
  - Values are hashed using a secret on both servers.
  - Implemented in nginx using Lua.

Users

- Trainorders offers paid users a small space for a personal website.
- These are very low traffic, but we continued to support them on a separate node.
- Storage is block storage attached to the users node, served by nginx.

Building

- Early on I decided I wanted to do all of this with Puppet.
  - Clearly documented, reproducible infrastructure automation.
  - Completely removes my need to “build” a node.
  - Nodes are cattle, not pets.
  - Configs stored in git

Building

- Terraform is also used to build nodes.
  - We can say “give me 4 web nodes” and run Terraform, and it makes 4 nodes.
  - Hands them off to puppet to be provisioned.
  - Within 10 minutes, we can have new nodes serving traffic.

Custom Modules

- Trainorders uses some really esoteric Apache modules.
  - These didn't have Ubuntu/Debian packages and had to be compiled from source.
  - Doing that in Puppet is … not pleasant.
  - It was easier to build our own deb packages and create PPAs. Puppet loves those.

Source Code

- Next step was getting all of the code into version control.
  - Parts were in git, parts in CVS (!), but much of it was untracked.
  - Difficult because the “core” of the site is the forums, which are powered by Phorum.
  - Lots of generated files.
  - Lots of cruft and old things.

Source Code

- Source code management is a prerequisite for reproducible infrastructure
- It took me about a week of evening work (and lots of emails) to get this done.
- We selected Bitbucket because I didn’t want to host a repo tool like Gitlab if I could avoid it and they offered free private repositories.

Source Code

- Upgrading to PHP 5.6 required some fixes.
- Integrating composer for third-party libs.
- Some changes were necessary to support the assets moving off the main server
  - Needed to be able to generate asset URLs anywhere they might be used.

Source Code

- Fixed a lot of other areas of the site (classifieds, image and video galleries, emails, etc) to use an ORM and twig.
- Generally cleaned up a lot of cruft and technical debt.
- Took about a month and a half of evening work to get all this done.

Database

- Migrating the database required taking the site down. :(
    - Total time to dump the database is about 5 minutes!
    - It doesn’t now thanks to xtrabackup :)
- Database changes are now stored in a git repo, and applied using phinx

Testing The Architecture

- We primarily used siege to test performance, and compare it to the existing infrastructure.
- Used the backup physical server for a more apples-to-apples comparison.
- The goal was to be no worse than the physical machine, but ideally to be better. We always were.
    - Overall a 63% increase in capacity over the PEAK of the train crash traffic.

Prep Work Conclusion

- From the time we started evaluating which direction we wanted to go until we were ready to start cutting traffic over was about 5 months.
- This was predominately me working in the evenings and very occasionally a few hours on weekends.

Go Live Day!

Going Live

- Picked a Sunday night (5/20/2018) because that is when the traffic is the lowest.
- About a week before, we started syncing over the bulk of the assets.
- Todd posted a message to the members prepping them for the outage.
- We also dropped the TTL on the DNS records to 1 hour about a week before.

Day Zero

- That evening, we shut the site down and replaced it with an information page explaining the downtime.
- Took a final dump of the database and transferred it over.
- Ran a final rsync of the asset library. Fortunately this was reasonably fast.
- Updated the DNS records to the new site.

Day Zero

- Within minutes, we already had people on the new site.
- What did we forget?
  - EMAIL.
  - Payment gateway needed the new IP addresses.
  - Homepage was updating on UTC and not Pacitic Time.

Day One+

- Over the next few days we fixed a lot of small bugs for things that didn’t get a lot testing.
- Having Kibana was indispensable here because we could see logs live in real time.
- Within a week, almost all of the errors had been fixed and everyone was on the new architecture.

“But wait, there’s more!”

–A great man.

Let’s Talk About Assets

- Again?
- Averaging more than a gig a day of new content.
- It took less than a year for the overhead I built into the block storage volume to get close to full
- This brings us to early 2019.

Object Storage

- As you'll recall from a few slides back, we originally went with block storage because object storage wasn't ready.
- This decision served us well for about a year, in which time Digital Ocean's object storage stabilized and became available in SFO2.

Object Storage

- The big benefit to object storage is that there is no limit on it. We'll never have to worry about running out of space.
- The cost will be drastically cheaper as well, saving us at least $200 a month.
- Downside is you lose some of the cost predictability of block storage.

Object Storage

- So when less than 100 gigabytes were remaining in the block storage volume, we started working to migrate to object storage.
- I calculated that, at the current fill rate, that was a little under 2.5 months.
- So we had a hard deadline to do something.

Migrating to Object Storage

- It can be tempting to try to shortcut this process by using a tool like s3fs-fuse and leaving your application code in place.
- Don’t do this.
- DO NOT DO THIS.
- No serious y’all, do not do this. It WILL end badly.
- You have got to change the application code.

Our Approach

- Wanted it to be transparent to users.
- We decided to do this in two parts:
  - Once we were ready, we would cut new uploads over to object storage
  - Continue serving old assets from block storage until we were ready to backfill.

Using Both?!

- Yes, it can be done!
- We have the block storage mounted on the asset server and using nginx’s `try_files`, we look to see if the file is available locally first, otherwise we proxy to object storage.
- We cache the responses on the asset server, so very few direct requests to either storage medium.

Our Approach

- So we started modifying all the areas that wrote to and read from NFS.
- The most difficult part was the forum mod that handled file uploads.
- Created a storage layer wrapper that checked both and did the right thing.

Incrementally

- Incrementally, we moved parts over as they were ready.
- This helped us flush out any errors we overlooked.
- Classifieds went first.
- Front page storage was next.

Incrementally

- The last to go was images and videos.
- By the time we got here, we were pretty confident that we had found all the edges.
- Rollout of the final parts were anti-climatic. No one noticed. And they shouldn’t. :)

Today

- Site has been humming along nicely.
- 95% of assets are served from object storage.
- Very few incidents in the meantime have tested the architecture.

What’s Next

- Block storage sunset and final transition to object storage.
- Managed database.
- Docker all the things!
- ElasticSearch replacing Sphinx, nginx replacing Apache.
- CDN and/or east-coast location.

Conclusion

- A busy fan website is really not all that different from any other busy website. The same approaches work here too.
- Don’t run your own iron unless you have a good reason … and the personnel, expertise and money to.
- An incremental approach if possible is always the best way to do things.

The Stack

- Ubuntu
- Apache
- nginx
- PHP (mod_php)
- MySQL
- Sphinx
- ElasticSearch
- Logstash
- Kibana
- Grafana
- InfluxDB
- Puppet
- Terraform
- Phinx

The End

Todd removing all the physical servers from the racks.