

500px

What 500px Engineering Learned from Upgrading Rails 3.2 -> 4.2.5

#RubyLightningTalksTO

Kevin McLoughlin
Jr. Web Developer

Home to everyone's best photos

500px is a photo community and marketplace for discovering, sharing, buying, and selling inspiring photography powered by creative people around the world



Me

Junior Web Developer

Graduated Bitmaker Aug. 2015

Hired by 500px Oct. 2015

Github: @trustyknave




An overview of our journey.



The process we engaged in as an organization.

The technical challenges we overcame along the way.





**Encourage you to push for upgrading
your own applications, by dispelling
fears associated with scale & unknowns.**

Goal



Process

Retrospective

Resources



Project Team

Sr. Software Eng.



Kevin

QA Lead



Artem

PMO



Aneta

Software Dev



Devon

Project Lead



Chris

Web Lead



Zimu

Jr. Web Dev



Kevin

Web Dev



Michael

Sr. Web Dev



Ryan

Our Process

Phase 1: *Investigation*

- Goal: All tests pass

Phase 2: *Implementation*

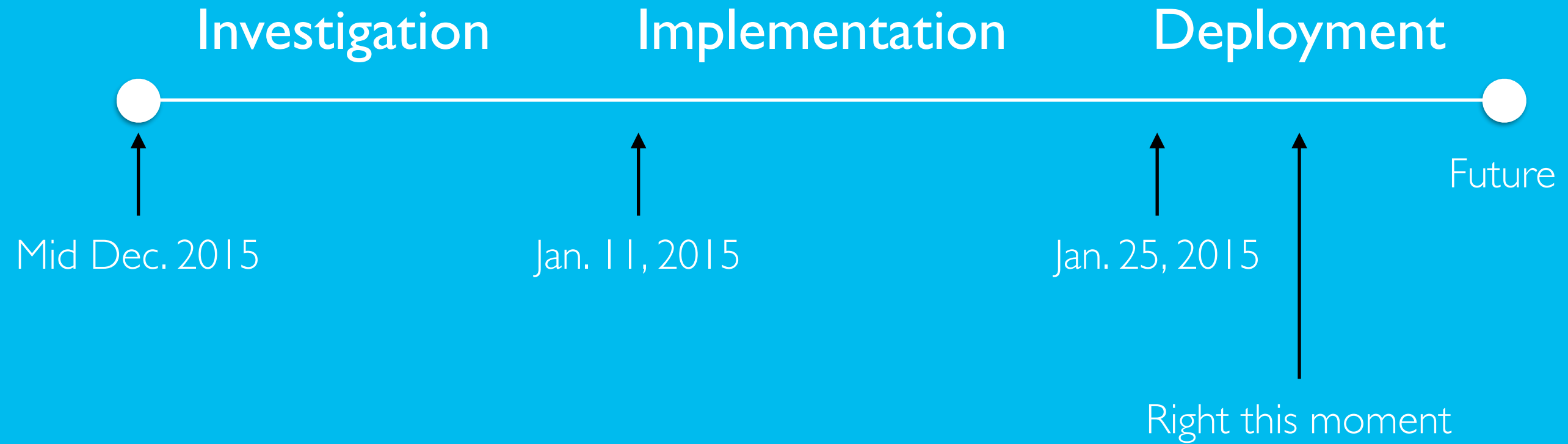
- Goal: No known bugs remaining

Phase 3: *Deployment*

- Goal: Equivalent or better performance



Timeline



Phase 1: Investigation

Goal: All tests pass





Know the state of your system



**A comprehensive test suite is
necessary and *invaluable*.**



Have a plan



It aligns everyone on the value of the project, and the steps that exist between your current and future states.





Rails 3.2 to 4.2.5 Upgrade Plan

Certain specifics have been removed
to maintain confidentiality.

Phase 1: *Investigation*


Goal: All tests pass

Step 1: Get your environment running

- Update your Gemfile to use Rails 4.2.5 and compatible versions of gems.
- Make database creation, migration, and seeding work.
- Correct all code violating Rails syntax/security checks so we can start a console.
- Get sidekiq workers to start (okay if errors are reported).
- Get test environment to run (okay if specs fail).



This will take time

208  Gemfile		<button>View</button>	
... @@ -1,111 +1,119 @@			
1	source "http://rubygems.org"	1	source "http://rubygems.org"
2	#ruby '2.1.0'	2	#ruby '2.1.0'
3		3	
4	-gem 'rake', '~>10.1'	4	+gem 'rake', '~> 10.4', '>= 10.4.2'
5	-gem 'rails', '~>3.2.22'	5	+gem 'rails', '~> 4.0', '>= 4.0.13'
6		6	+
7		7	
8	# Rails extensions	8	# Rails extensions
9	-gem "validates_existence", "~> 0.8.0"	9	+gem 'protected_attributes', '~> 1.0', '>= 1.0.3'
10	-gem "strong_parameters", "~> 0.2.1"	10	+gem "validates_existence", "~> 0.8"
11	-gem "activerecord-mysql-index-hint", "~> 0.0.2"	11	+gem 'actionpack', '~> 4.0'
12		12	+# gem "strong_parameters", "~> 0.2"
13		13	
14	# Storage Engines (DB, CDN, CACHE, ETC)	14	# Storage Engines (DB, CDN, CACHE, ETC)
15	-gem 'mysql2', '~> 0.3.20'	15	+gem 'mysql2', '~> 0.3'
16	-gem 'seamless_database_pool', :git =>	16	+gem "activerecord-mysql-index-hint", "~> 0.0.2"
17	'https://github.com/500px/seamless_database_pool.git'	17	+gem 'seamless_database_pool', '~> 1.0'
18	-gem 'dalli', '~> 2.6.2'	18	+
19	-gem 'hiredis'	19	+gem 'dalli', '~> 2.6'
20	-gem 'mongoid', '~> 2.8'	20	+gem 'hiredis', '~> 0.6'
21	-gem 'bson_ext', '~> 1.9'	21	+gem 'mongoid', '~> 4.0', '>= 4.0.2'
22	-gem 'splunk-sdk-ruby', '=1.0.2'	22	+gem 'splunk-sdk-ruby', '~> 1.0'
23	-	23	+
24	-gem 'aws-sdk', '~> 2.0.0'	24	+gem 'aws-sdk', '~> 2.0'
25	-gem 'edgecast_api'	25	+gem 'edgecast_api', '~> 0.0.1'
26	-gem 'fastly'	26	+gem 'fastly', '~> 1.2'

Because we introduced version locking during the project, almost every one of our dependencies updated by at least 1 minor version.

The 20/80

The 20% of issues that took 80% of our time

- Mongoid went from 2.8 -> 4.0.2, and introduced a host of syntax changes and new dependencies for us to investigate and understand.
- New `ActiveRecord` query syntax.
- Scopes now using callable objects like a Proc or Lambda e.g.
 - `scope :active, -> { where active: true }`



Phase 1: *Investigation*

Goal: All tests pass

Step 2: Get all tests to pass

- Model specs: 2982 examples, 96 failures, 910 pending
- Controller specs: 4142 examples, 1145 failures, 15 pending
- Worker specs: 153 examples, 15 failures
- Request specs: 731 examples, 147 failures



This will take time

Thread #3 2 more setup commands +

Launch SSH

```
export THREADNUM=3; ./script/semaphore.sh
```

10:10

Comment

0.0706 seconds average (0.0706 seconds / 1 example) ./spec/models/comment_spec.rb:3

Api::V1::Admin::PhotosController

0.06626 seconds average (0.26504 seconds / 4 examples) ./spec/controllers/api/v1/admin/photos_controller_spec.rb:3

Api::V1::Editors::PhotosController

0.06416 seconds average (0.25666 seconds / 4 examples) ./spec/controllers/api/v1/editors/photos_controller_spec.rb:3

HomeController

0.04612 seconds average (0.04612 seconds / 1 example) ./spec/controllers/home_controller_spec.rb:3

Finished in 5.19 seconds

34 examples, 34 failures

Failed examples:

```
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:422 # Api::V1::FavoritesController#index sorting no sort param provided returns users :
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:434 # Api::V1::FavoritesController#index sorting sort by friends returns friends on top
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:452 # Api::V1::FavoritesController#index when requesting with "include_following" flag
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:458 # Api::V1::FavoritesController#index when requesting with "include_following" flag
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:466 # Api::V1::FavoritesController#index when requesting with "include_following" flag
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:242 # Api::V1::FavoritesController#destroy when logged in when the photo is favorited f
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:232 # Api::V1::FavoritesController#destroy when logged in when the photo is favorited c
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:225 # Api::V1::FavoritesController#destroy when logged in when the photo is favorited f
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:309 # Api::V1::FavoritesController authenticated requests Favorites manipulation GET fa
rspec ./spec/controllers/api/v1/favorites_controller_spec.rb:320 # Api::V1::FavoritesController authenticated requests Favorites mani
```

LINE WRAP OFF

The 20/80

The 20% of issues that took 80% of our time

- New Mongoid query syntax changes.
- No more implicit conversion of `ActiveSupport::TimeWithZone` to a Ruby Time object.
- New Rails caching structure.
- New JSON serialization structure.
- Devise compatibility issues



Phase 1: *Investigation*

Goal: All tests pass

Step 3: Collectively make a go/no-go decision

- List your red flags and check your progress against understanding their potential impacts.
- Make a manual test plan to complement your suite of automated tests.
- Make a deployment plan that includes disaster recovery scenarios.



Phase 2: Implementation

Goal: No known bugs remaining



Phase 2: *Implementation*

Goal: No known bugs remaining

Concurrent execution of manual test plan and debugging

- Get onto an environment that reasonably mimics Production as quickly as possible.
- Do as little as possible. We used the `protected_attributes` and `activerecord-deprecated_finders` gems to reduce the amount of work.
- Most of the significant issues we encountered were very specific to our codebase.



The 20/80

The 20% of issues that took 80% of our time

- `acts_as_taggable`
- Issues arising from the changes to Rails sessions.
- Rebasing against master.
- Deployment preparation.



Phase 3: Deployment

Goal: Equivalent or better performance



Phase 2: *Deployment*

Goal: Equivalent or better performance

Slow rollout of Rails4 branch to the various sections of our infrastructure

- We're about halfway through this right now.
- We've seen a significant reduction in performance on our API servers.
- Current solution is to reduce the number of workers per machine, and order more hardware.
- Some good news. On Rails 4.2.5, 50% of the workers are providing ~75% of the throughput of Rails 3.2.



This may take some time

api10(unicorn:500px API)

Delete this trace

Jan 27, '16 11:20 am

TRACE TIME

2,250 ms

RESP. TIME

1,5

CPU B

Summary

Trace details

Database statements

Total duration	Call count	Total duration	Call count	Database	Statement
53 ms	99	124 ms	100	ActiveRecord	SELECT `licensing`
52 ms	99				
48 ms	99	97 ms	100	ActiveRecord	SELECT `delivery_`
47 ms	99	131 ms	100	ActiveRecord	SELECT `tags`.* FROM `aggings`.`taggabl`
52 ms	99	115 ms	100	ActiveRecord	SELECT `licensing` censing_release_p`
7 ms	1	100 ms	100	ActiveRecord	SELECT `tags`.* FROM `aggings`.`taggabl`
14 ms	1	12 ms	1	MongoDB	QUERY data 01, 136966559, 13



The 20/80

The 20% of issues that took 80% of our time

- Dealing with this performance issue.
- Weird stuff we're seeing:

Rails 3.2

```
irb(main):006:0> Benchmark.measure { 1_000_000.times { (0..5).each { |i| } } }  
=> 0.370000 0.000000 0.370000 ( 0.371422)
```

Rails 4.2.5

```
irb(main):005:0> Benchmark.measure { 1_000_000.times { (0..5).each { |i| } } }  
=> 1.620000 0.030000 1.650000 ( 1.649495)
```



Process

Retrospective

Resources



**Made a way to quickly
measure performance over
the course of the project.**



**Made the data in our
development environments
more similar to that in our
production environment.**



**Done it sooner, when the
state of our dependencies
was understood by a broader
section of the team.**



Process Retrospective Resources



Resources

Information we found useful

- [Official Rails Upgrade Guide](#)
- [Heroku Engineering Post about Upgrading to ActiveRecord 4](#)
- [Remarkable Labs Post about What's new in Active Record 4](#)
- [Official Mongoid 3.0 Upgrade Guide](#)
- [Artsy Post about Upgrading to Mongoid 4](#)



A couple is seen from behind, sitting on a grassy hill. They are looking out over a large, calm lake that is surrounded by steep, forested mountains. The sky is overcast with grey clouds. The overall mood is peaceful and scenic.

We're Hiring.



**Come talk to me about our
company values, and open
positions.**

