

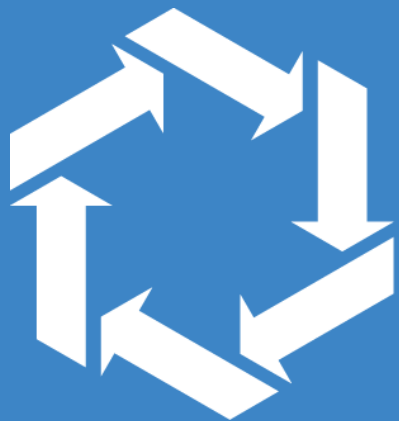
# Faster specs

Speeding up a slow test suite

# Richard Ramsden

@rramsden





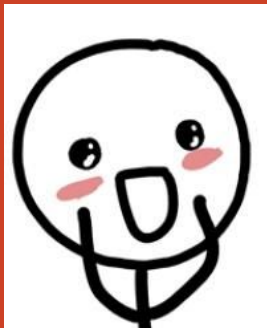
Degica

# A long time ago...

Many pull-requests ago...

# Tests were fast

Developers were happy



```
Run options: include {:focus=>true}
```

```
All examples were filtered out; ignoring {:focus=>true}
```

```
Randomized with seed 49955
```

```
.....
```

```
Finished in 20.94 seconds (files took 0.56839 seconds to load)  
173 examples, 0 failures
```

# 2 years later...

**After a few feature requests...**







**What makes a test suite slow?**

# The Rails Testing Pyramid

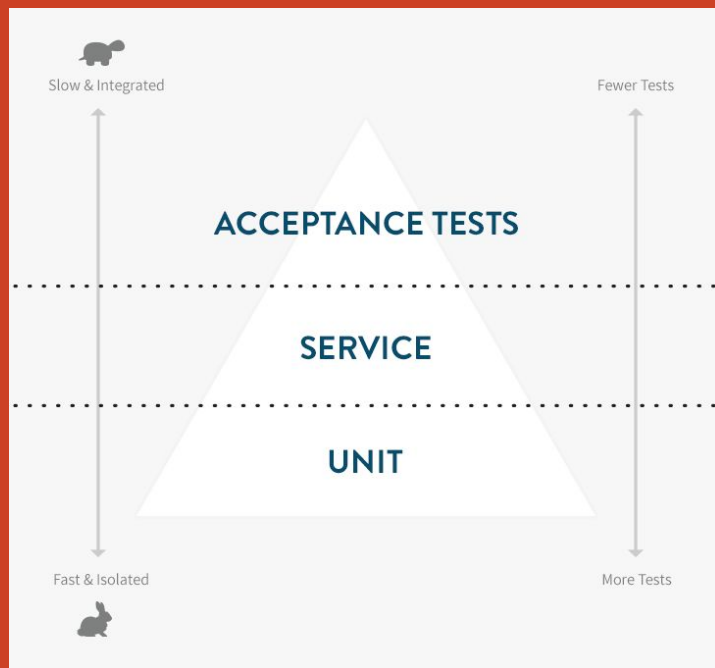


Image Credit: Code Climate Blog

# The Rails Ice Cream Cone

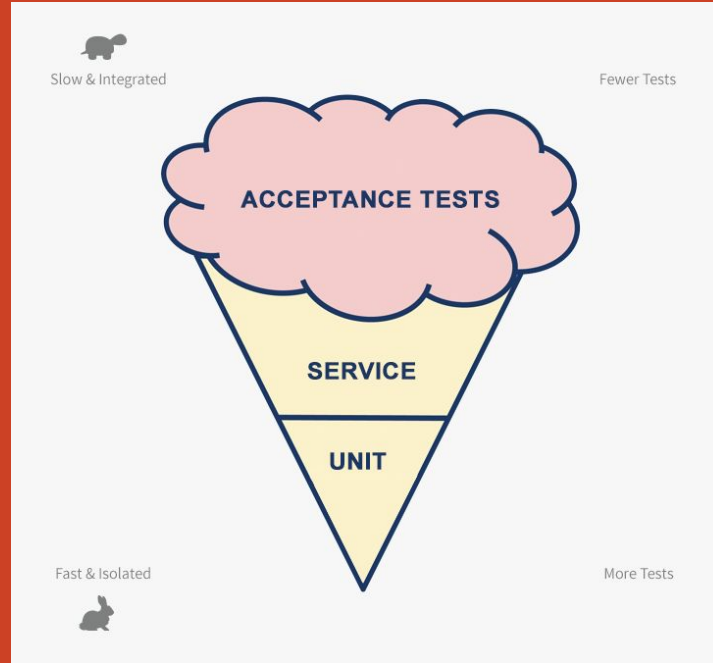


Image Credit: Code Climate Blog

**How do I speed up my test suite?**

# Don't run all your tests “locally”

the fastest tests are no tests (no really?)

# Be selective

- Use an editor that can run tests
- Disable slow tests locally

# Start slow

```
$ rspec --profile
```

```
$ ruby test/minitest_spec.rb --profile
```

# Prune slow tests ✂

(Usually the feature specs)

- Acceptance tests should be small and targeted
- Try to convert a slow tests to unit test
- Don't be afraid to delete tests



**CODE**

# Isolate unit tests

(Extract lots and lots of new classes)

- Rails makes your tests slow
  - Try and reduce Rails dependencies
- Use test doubles to mock out collaborators
- Follow Single Responsibility Principle
  - Extract new classes

**CODE**

# Buy faster tests 💰



# Parallel Builds

should be your last step

Startup	Small Business	Premium
<b>\$129</b> PER MONTH	<b>\$249</b> PER MONTH	<b>\$489</b> PER MONTH
2 Concurrent Jobs	5 Concurrent Jobs	10 Concurrent Jobs
Unlimited Build Minutes	Unlimited Build Minutes	Unlimited Build Minutes
Unlimited Repositories	Unlimited Repositories	Unlimited Repositories
Unlimited Collaborators	Unlimited Collaborators	Unlimited Collaborators
START TRIAL	START TRIAL	START TRIAL

## Containers

All of your containers are shared across your entire organization.

64

## Parallelism

You can run **64** concurrent builds with **64** containers and **1x** parallelism.

1x

4x

8x

12x

16x

### Estimated Cost

Repos	∞
Users	∞
Build Minutes per Month	∞
Max Parallelism	64
Concurrent Builds	64
Total	\$3150

Sign Up Free



# Knapsack

✓	\$ bundle exec rake knapsack:rspec (0)	config	02:13
✓	\$ bundle exec rake knapsack:rspec (1)	config	02:28
✓	\$ bundle exec rake knapsack:rspec (2)	config	02:30
✓	\$ bundle exec rake knapsack:rspec (3)	config	02:30

# Other Tips

- Fail fast in CI
  - Faster feedback for your team
- Don't tune the Ruby garbage collector
  - You're going to have a bad time
- Try to avoid writing to database in unit tests
  - Use an instance instead of persisting a record
  - Too much I/O causes slow tests
- Use Spring

# My progress (~2-3 weeks) 52% faster

Before

```
Finished in 4 minutes 58.2 seconds  
2085 examples, 0 failures
```

After

```
Finished in 2 minutes 42 seconds  
2156 examples, 0 failures
```



# Quick Summary

- Your fastest tests are no tests
  - Don't be afraid to delete slow tests
  - Use selective testing don't run everything
- Reduce class dependencies
  - Extract classes away from Rails
  - Follow Single Responsibility Principle
- Parallel tests can provide extra speed up to a limit
  - Should be your last step

# Questions?

# Thanks!

<https://github.com/r Ramsden/faster-specs>