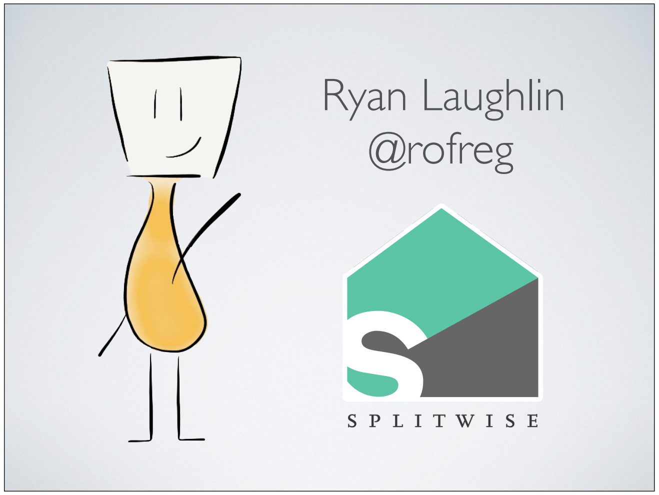
THE DOCTOR IS IN

Using checkups to find bugs in production

Hi everybody!



Ryan Laughlin
@rofreg

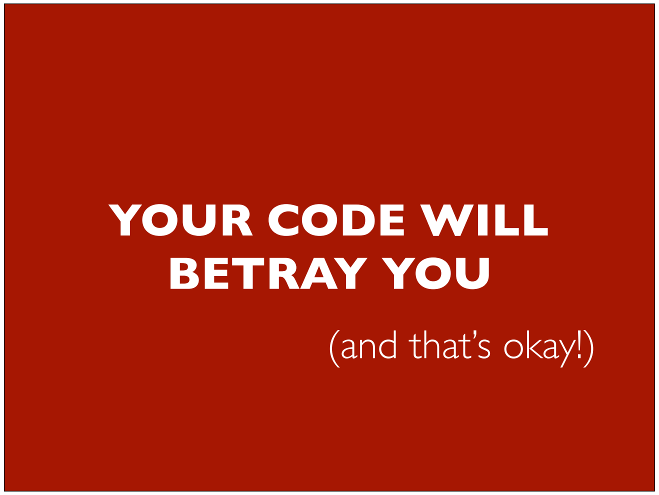
SPLITWISE

I’m Ryan Laughlin, or @rofreg if you know me from the internet.

I’m a cofounder of Splitwise, which is an app for splitting expenses with other people.

I’ve been lucky enough to work at Splitwise for almost 7 years now, as we’ve grown from hundreds of users to millions of users, and that has exposed me to a TON of important lessons about running Ruby in a production environment, especially at a large scale.

One of the most important things that I’ve learned is this:



YOUR CODE WILL BETRAY YOU

(and that's okay!)

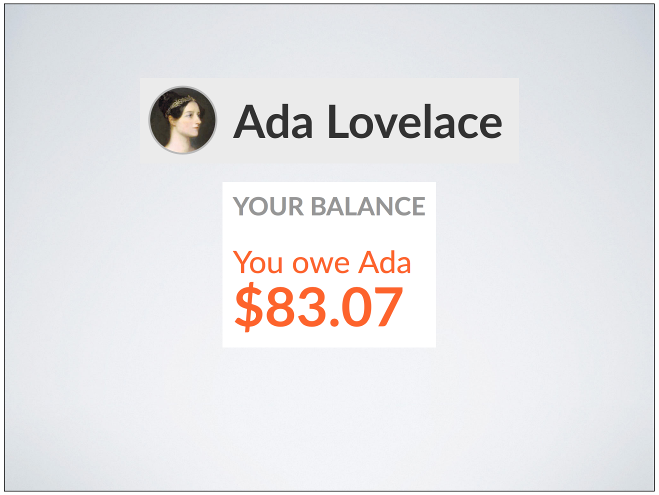
Your code is going to do some weird, unexpected stuff in production.
Like…
Things you could never have anticipated.

And you know what?
That's okay!

Code is imperfect. It has bugs, and it's always going to have bugs.

Your goal is NOT to write code that has zero bugs.

Your goal is to build a system that helps you DISCOVER new bugs as they happen.
Because new bugs are a chance to learn, and to improve your skill as a developer and as an engineer.



Ada Lovelace

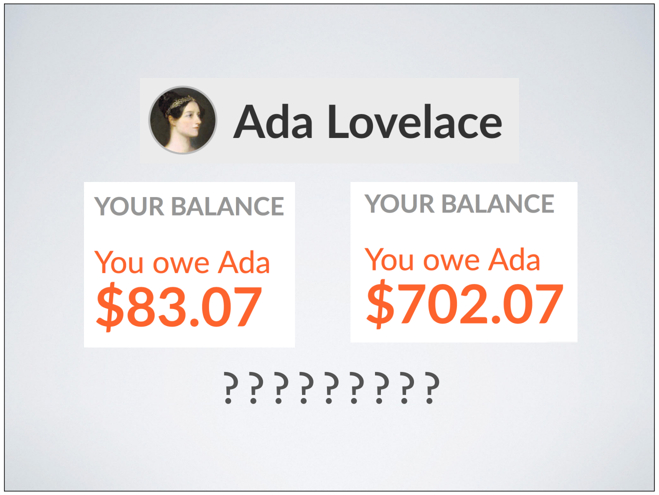
YOUR BALANCE

You owe Ada
$83.07

Let me tell you about one particularly nasty issue that we encountered earlier this year.

One of the most important things that Splitwise does is calculate your total balance with another person.
For example, "You owe Ada $83".

It's really important that we get this calculation right, and we have a bunch of tests to validate that everything adds up correctly.



Ada Lovelace

YOUR BALANCE

You owe Ada

$83.07

YOUR BALANCE

You owe Ada

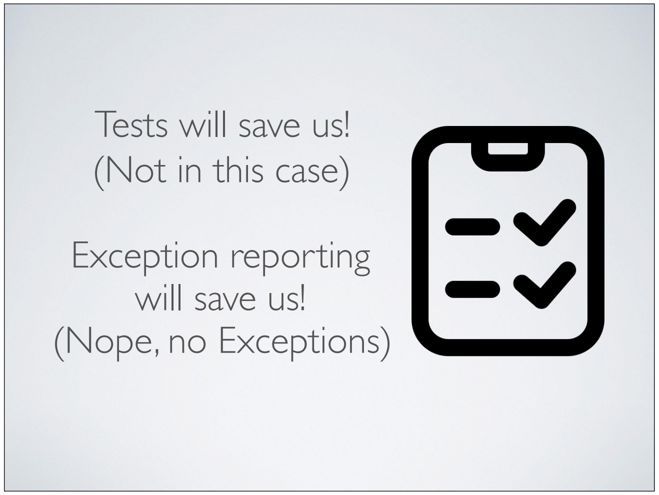
$702.07

? ? ? ? ? ? ? ? ?

But one random Tuesday, we had a massive caching failure in production. And all of a sudden, our code started returning inconsistent results for the *same calculation*.
So when I asked, “How much do I owe Ada?”, our Ruby code might reply: “83 dollars”
But it ALSO might reply: “702 dollars”

This is obviously a huge, user-facing problem

And it would have been really easy to miss (at least, until our users started contacting us)



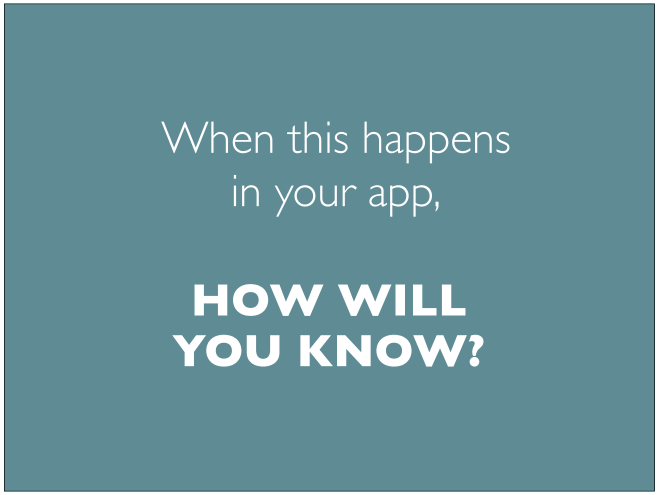
Tests will save us!
(Not in this case)

Exception reporting
will save us!
(Nope, no Exceptions)

So how could you catch this issue?
Well, tests clearly didn’t help – all of our tests were passing just fine, but we had never anticipated this particular failure case

Well, what about exception reporting! Exception reporting is great: it tells us about unexpected exceptions in production!
Well…there weren’t actually any Exceptions to report. The code ran successfully. It just returned the wrong result.

All of the STANDARD techniques for catching bugs and errors had failed us.



When this happens
in your app,

**HOW WILL
YOU KNOW?**

Yesterday, when Chad Fowler was up here, he said “Tests are a design smell”
And we all laughed, but he had a point!

Tests are NOT the same as production.
Sometimes tests can convince you that you have protected yourself from production issues…WHEN YOU ACTUALLY HAVEN’T.

So how CAN you protect yourself in production?

CHECKUPS!

Checkups! That’s how!

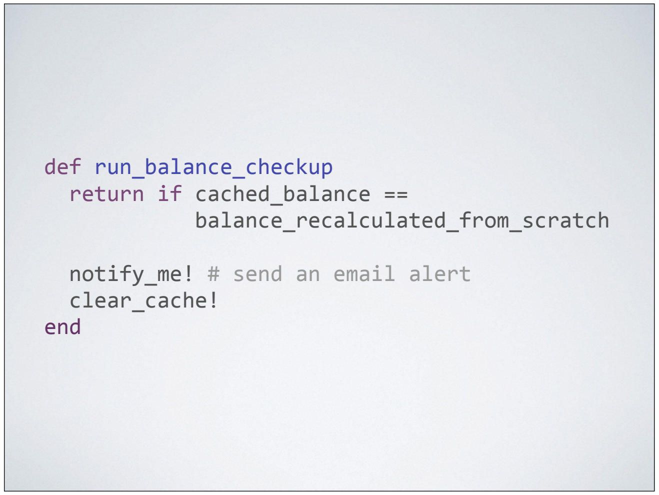
Let me introduce you to this idea, because I think it’s really, REALLY important.

CHECKUPS ARE TESTS FOR PRODUCTION

Checkups are tests for production.

Just like a test, you declare some expectations about how you expect your app to behave in production
Then you run code to evaluate those expectations, ON A REGULAR BASIS, many times per day
If your checkup fails, then you need to be alerted so that you can investigate what happened

Think of it like CONTINUOUS INTEGRATION FOR PRODUCTION.



```
def run_balance_checkup
  return if cached_balance ==
            balance_recalculated_from_scratch

  notify_me! # send an email alert
  clear_cache!
end
```

So let’s go back to this horrible balance bug. A year or two before that, we had built a “checkup” rake task, which was set to run every 10 minutes in production.

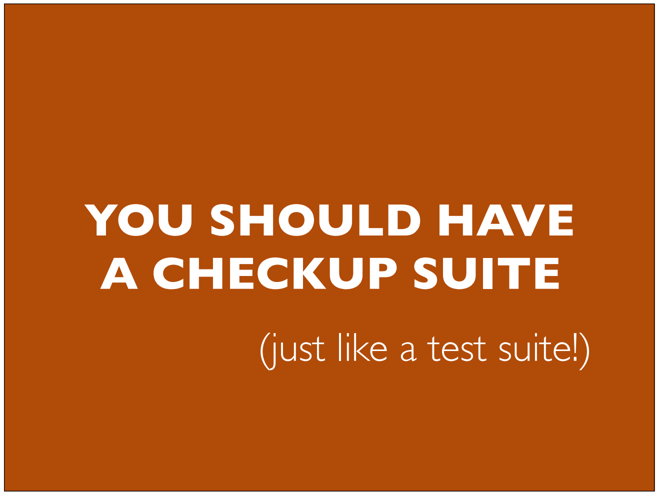
This task double-checked that the cache-dependent version of our “balance” method returned the same result as an alternate implementation that did NOT use the cache.

By comparing these two values for all the accounts updated within the past few minutes, we could CONTINUOUSLY VERIFY that our “balance” method was working as expected.
And if anything went wrong, we could clear the cache

THANK GOD

This saved the day!!!!
I cannot stress enough how useful this little block of code was.

Not only did our checkups ALERT us about a major problem that we would have otherwise missed,
But it actually MITIGATED the problem in realtime until we could figure out a fix and deploy it.

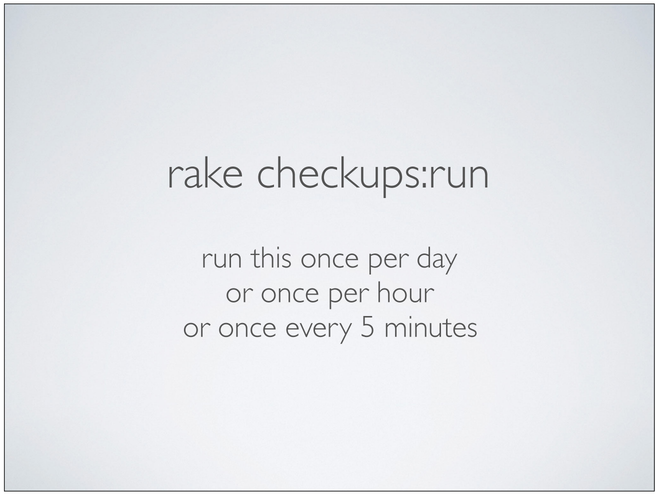


YOU SHOULD HAVE A CHECKUP SUITE

(just like a test suite!)

We have a bunch of checkups like this, and this isn’t the only time that a checkup has saved us and helped us resolve a production issue more quickly.

This isn’t a fully-formed idea yet, but I’m beginning to think that a continuous “checkup” suites may be just as important as test suites.



rake checkups:run

run this once per day
or once per hour
or once every 5 minutes

If you want to add checkups to your own app, it’s really easy.
Just define a rake task, then set it to run regularly.

Talk to me! I’d love to chat further.
@rofreg

This is still an immature idea, so if it piques your interest, please come talk to me!!

I’d love to discuss this further, because I think it’s a really promising idea to improve the way we monitor and debug production apps.