Why I Prefer Good Testing over Excellent Testing

Tina Fletcher - @fletchertinam

Oracle NetSuite HCM Summit - 08/02/18

Outline:

- Definitions and why I care about this
- Story time (5):
  - Defects I've missed lately
  - Times I've been saved by failure planning
- So what?

~~with enough time, we'll find every bug!~~

But...

Do testing, and also prepare for failure

Excellent Testing:

- Think of as many test cases as you can
- Try to run them all
- Keep testing until you run out of time

Good Testing:

- Think of as many test cases as you can
- Run the ones that will tell you the most about how well your product is working
- Spend remaining time planning failure handling

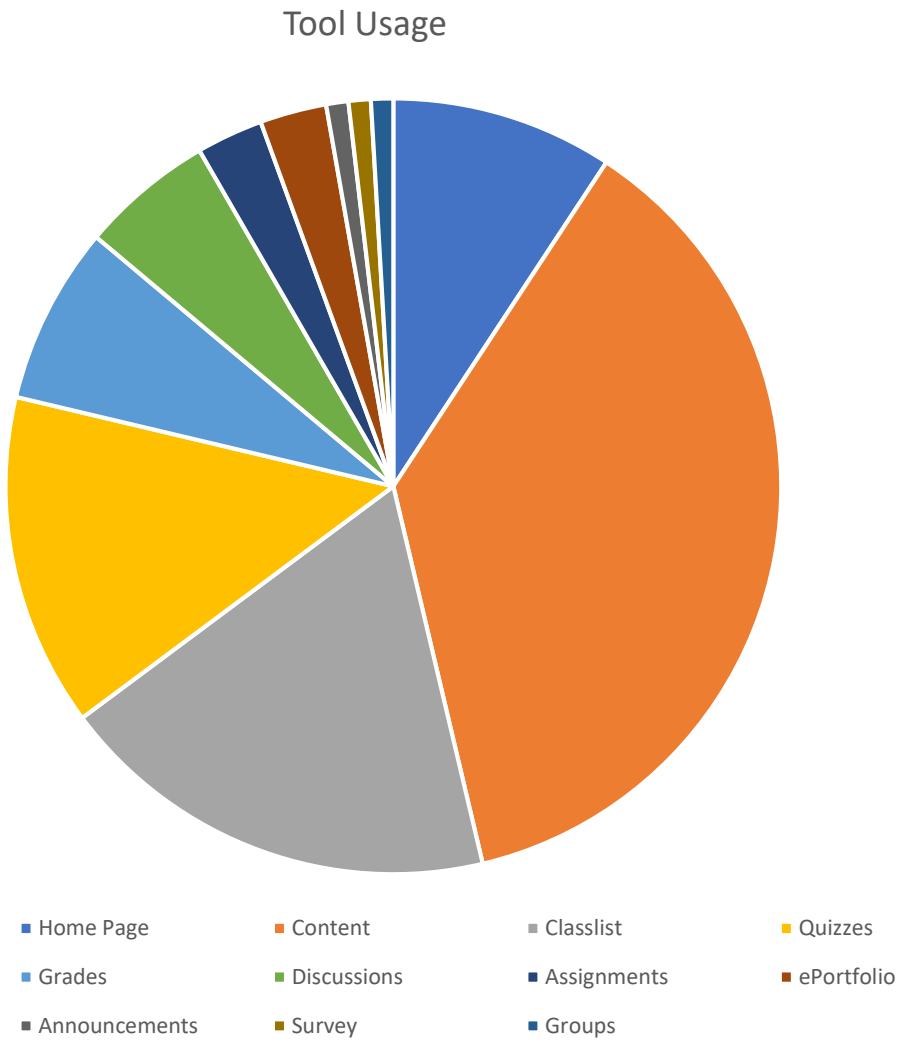
Defects I've Missed Lately

(and wouldn't have found with all the time in the world)

Story 1:

Monitoring user REALLY loves the home page

Story 1: Monitoring user REALLY loves the home page



Story 1: Monitoring user REALLY loves the home page

we did a lot of testing
for this report.

Story 1: Monitoring user REALLY loves the home page

What if there are a really large or small number of tools available?

What if tool names contain special characters, or appear in other languages?

Can the graph be understood by someone using a screen reader?

we tested the appearance of the graph.

Does the graph look ok on phones and tablets?

Does the graph look ok in all the browsers we support?

What if there are really long or short tool names?

Story 1: Monitoring user REALLY loves the home page

How long does it take from the time an event is sent to the time it is reflected in the report?

Are all of the appropriate events fired even when the system is under heavy load?

What if the same user visits the same tool many times in quick succession?

we tested the event flow through our system.

Is an event fired if a tool is accessed via API rather than through the UI?

Will events get re-queued if the event processing system is down or not responding?

What kind of event is sent when using the learning platform's impersonation functionality?

What happens if an event is malformed in some way?

Story 1: Monitoring user REALLY loves the home page

What is displayed if no data is available, or data cannot be retrieved?

What happens if the retrieved data arrives in an unexpected format?

Is retrieval of the data appropriately restricted to users that have permissions to view this report?

we tested data retrieval functionality.

Does data retrieval time increase for very large data sets?

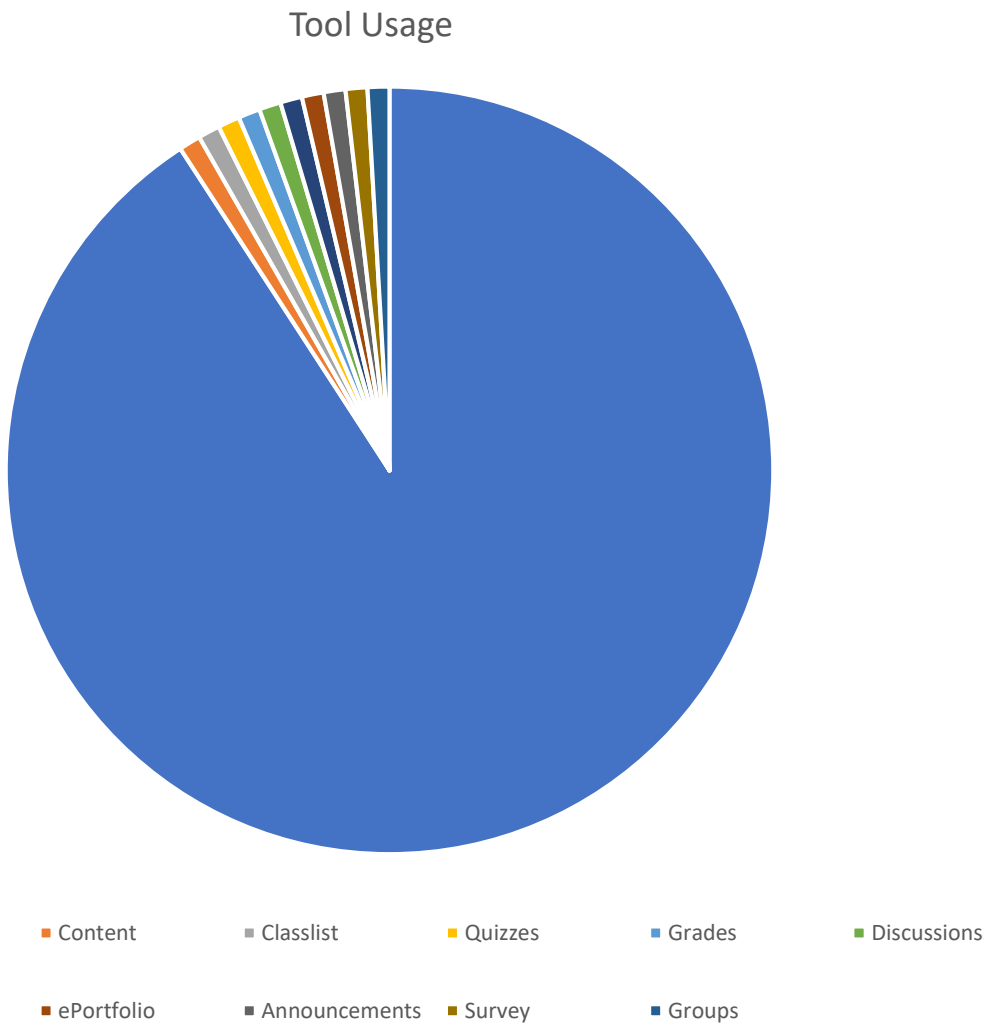
Does data retrieval time increase if a large number of users are accessing it at the same time?

How is the graph impacted if a user retrieves data while in a different time zone?

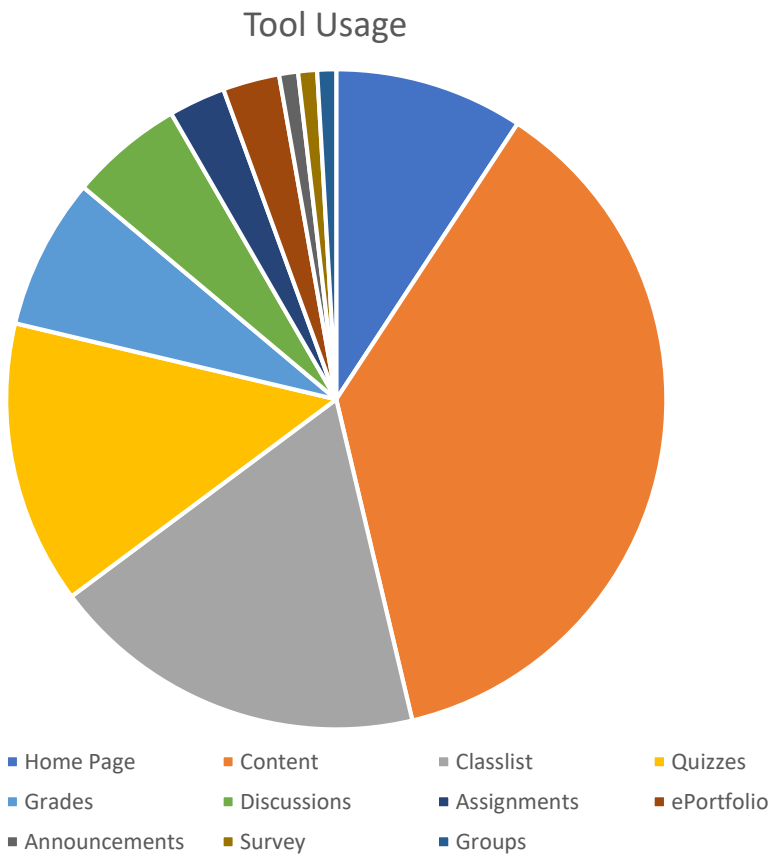
Story 1: Monitoring user REALLY loves the home page

we did a lot of testing
for this report.

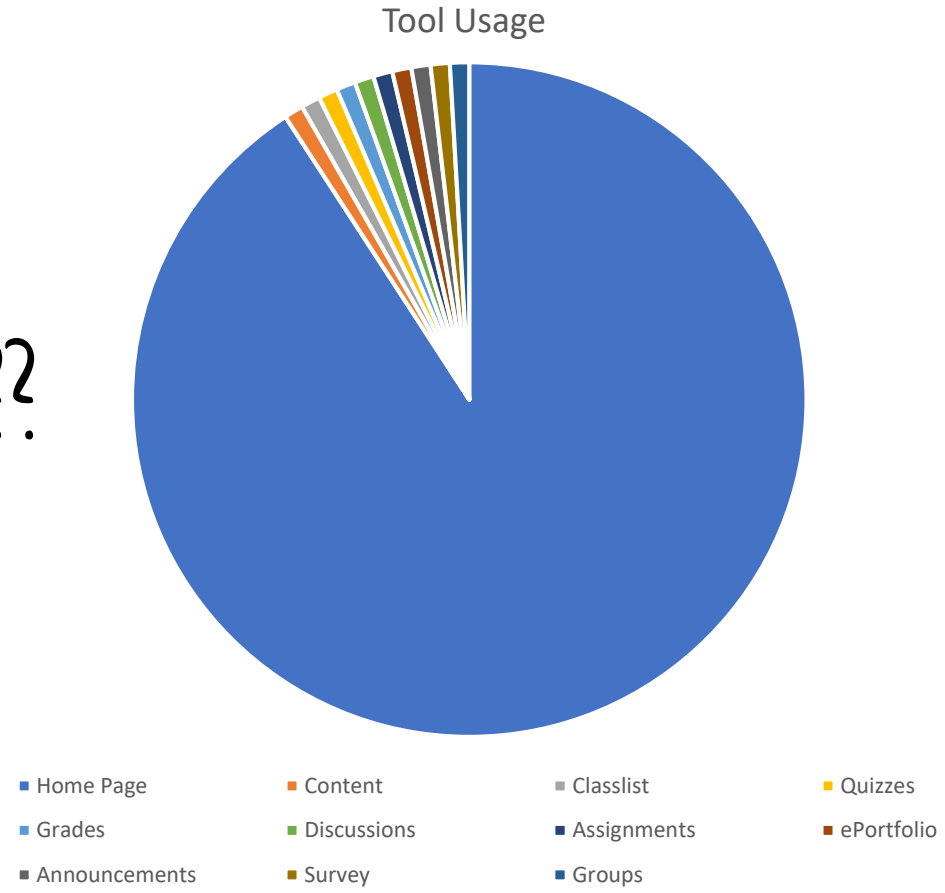
Story 1: Monitoring user REALLY loves the home page



Story 1: Monitoring user REALLY loves the home page



why??



Each school has an automated "monitoring" user that hits the home page tool once <u>every 5 minutes</u> to make sure the site is up.

Story 1: Monitoring user REALLY loves the home page

LESSON:

You don't know what you don't know.

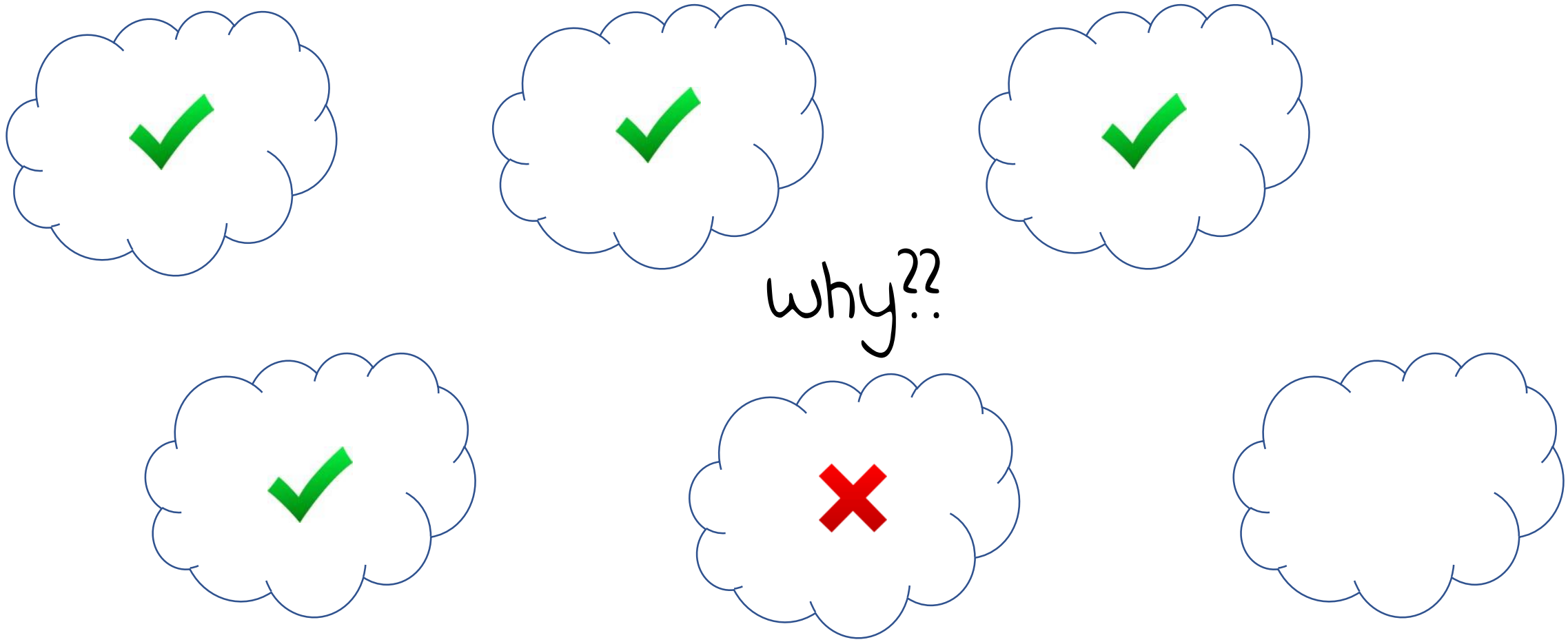
(and you won't be testing for it)

Story 2:

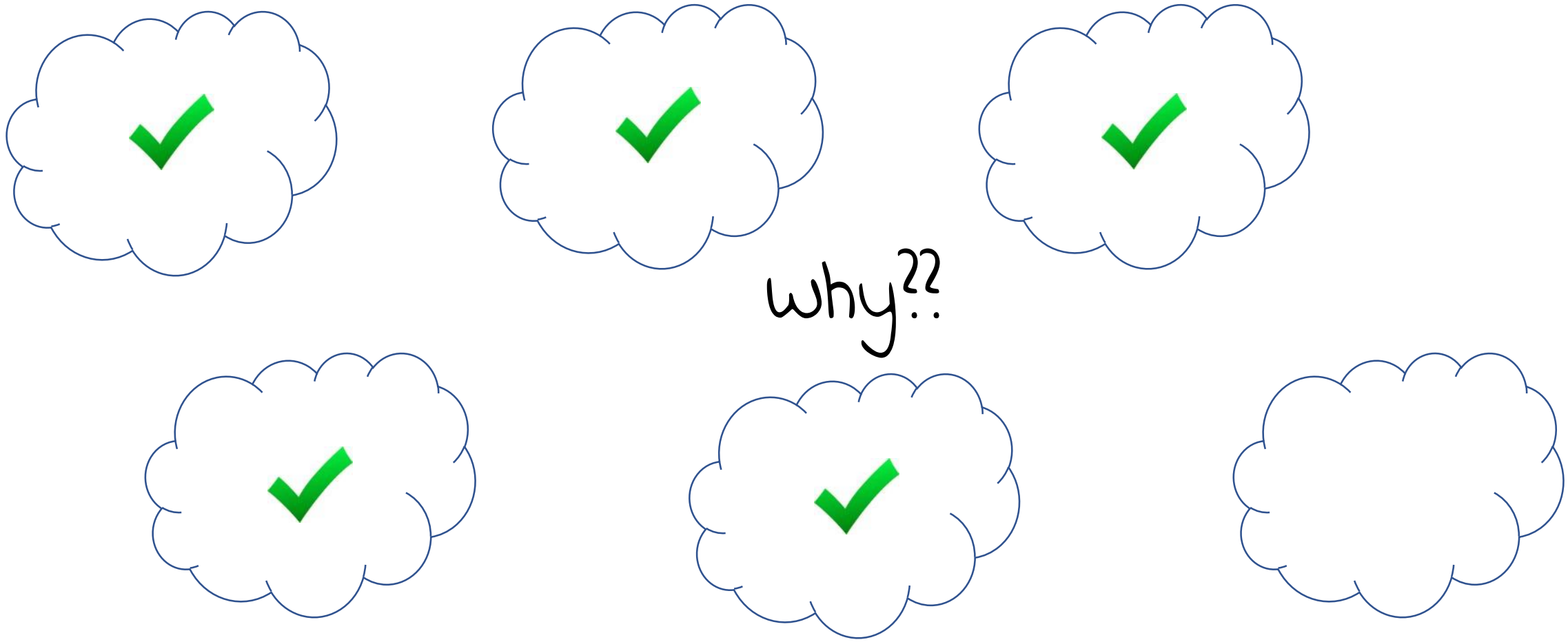
The port that was closed for no reason

Story 2: The port that was closed for no reason

Story 2: The port that was closed for no reason



Story 2: The port that was closed for no reason



Port X was closed. But it was open when we last checked.

Story 2: The port that was closed for no reason

LESSON:

what's true today might not be true tomorrow.

Story 3:

Yeah, but S3 will never go down

Story 3: Yeah, but S3 will never go down

Story 3: Yeah, but S3 will never go down

S3 contingency planning

Security testing

Performance testing

Story 3: Yeah, but S3 will never go down

AWS's S3 outage was so bad Amazon couldn't get into its own dashboard to warn the world

Websites, apps, security cams, IoT gear knackered

By Shaun Nichols in San Francisco 1 Mar 2017 at 03:00 122 SHARE

The day Amazon S3 storage stood still

Posted Mar 1, 2017 by Ron Miller (@ron_miller)

FEB 28, 2017 @ 04:01 PM 29,059

The Little Black Book of Billionaire Secrets

Amazon S3 Outage Has Broken A Large Chunk Of The Internet

How a typo took down S3, the backbone of the internet

Hello, operator

By Casey Newton | @CaseyNewton | Mar 2, 2017, 1:24pm EST

Amazon AWS S3 outage is breaking things for a lot of websites and apps

Posted Feb 28, 2017 by Darrell Etherington (@etherington)

Story 3: Yeah, but S3 will never go down

LESSON:

Things that are out of your control can impact you.

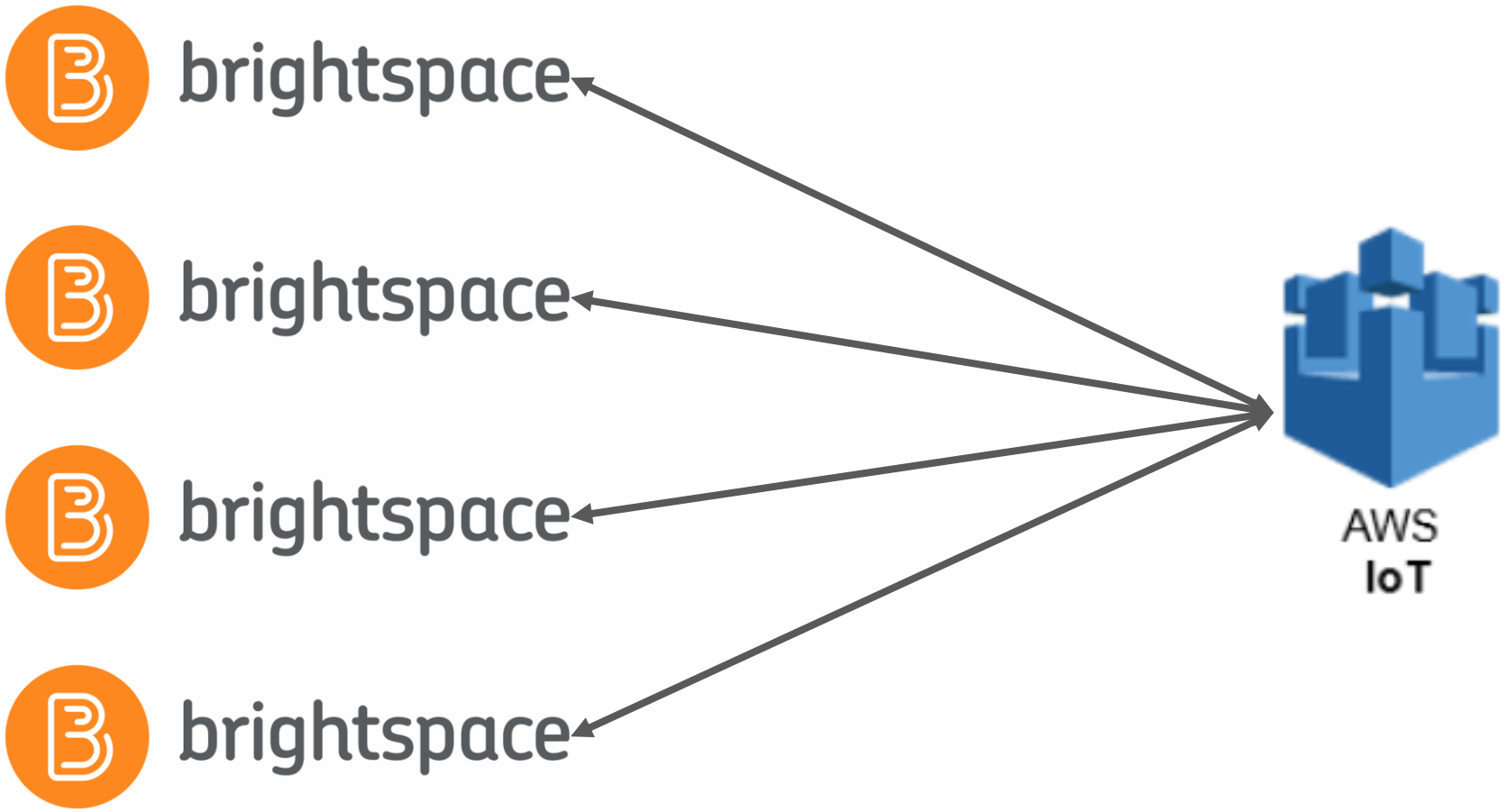
(and nothing is invincible)

Times I Have Been Saved By Solid Monitoring and Roll-Back Strategies

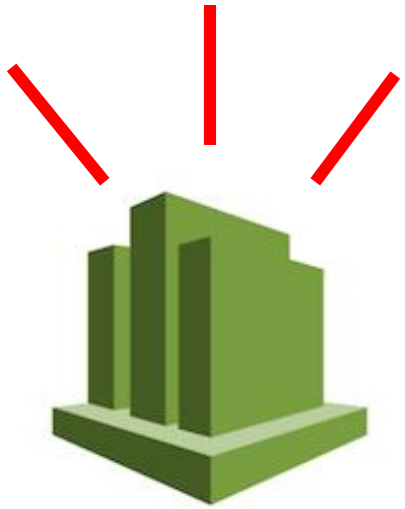
Story 4:

The off by 10,000 error

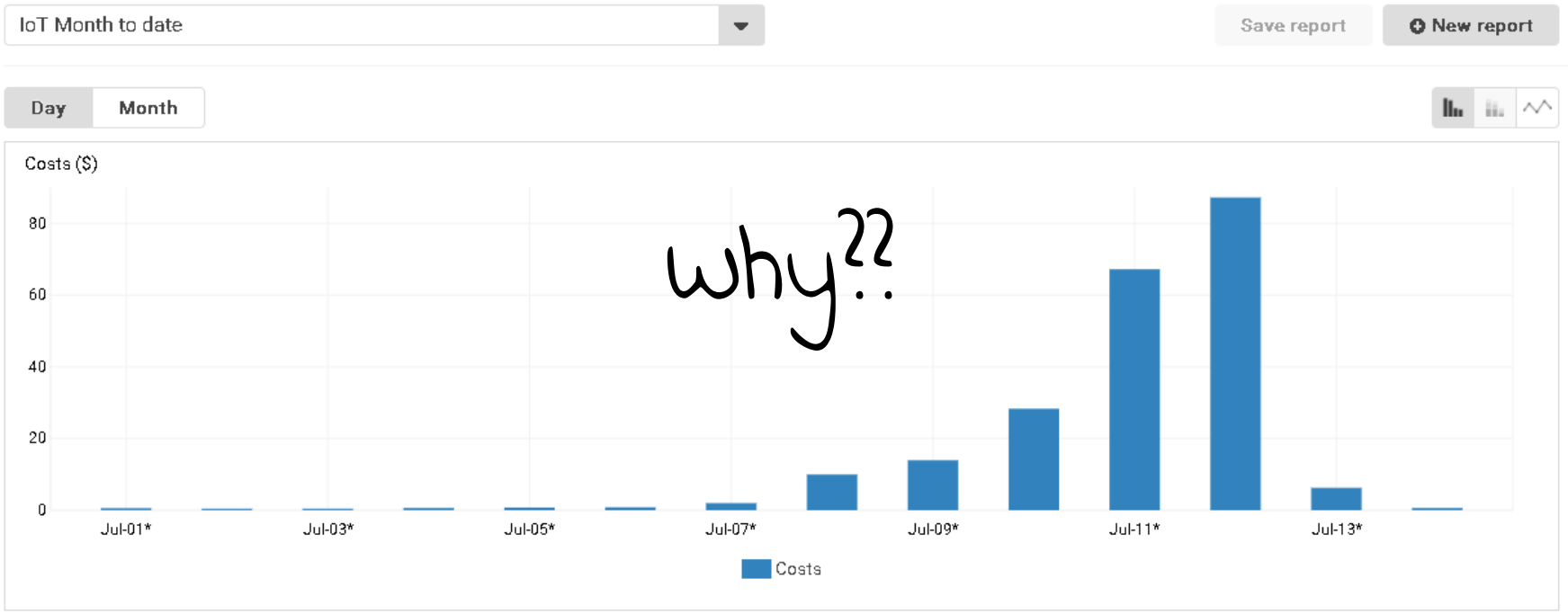
Story 4: The off by 10,000 error

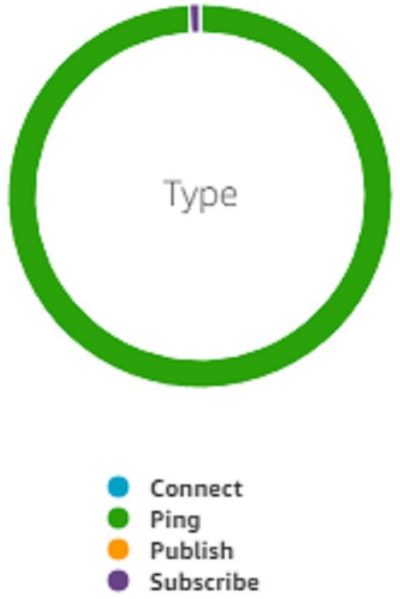


Story 4: The off by 10,000 error



Story 4: The off by 10,000 error





Every instance of the software was pinging the message broker once every one second.

Story 4: The off by 10,000 error

LESSON:

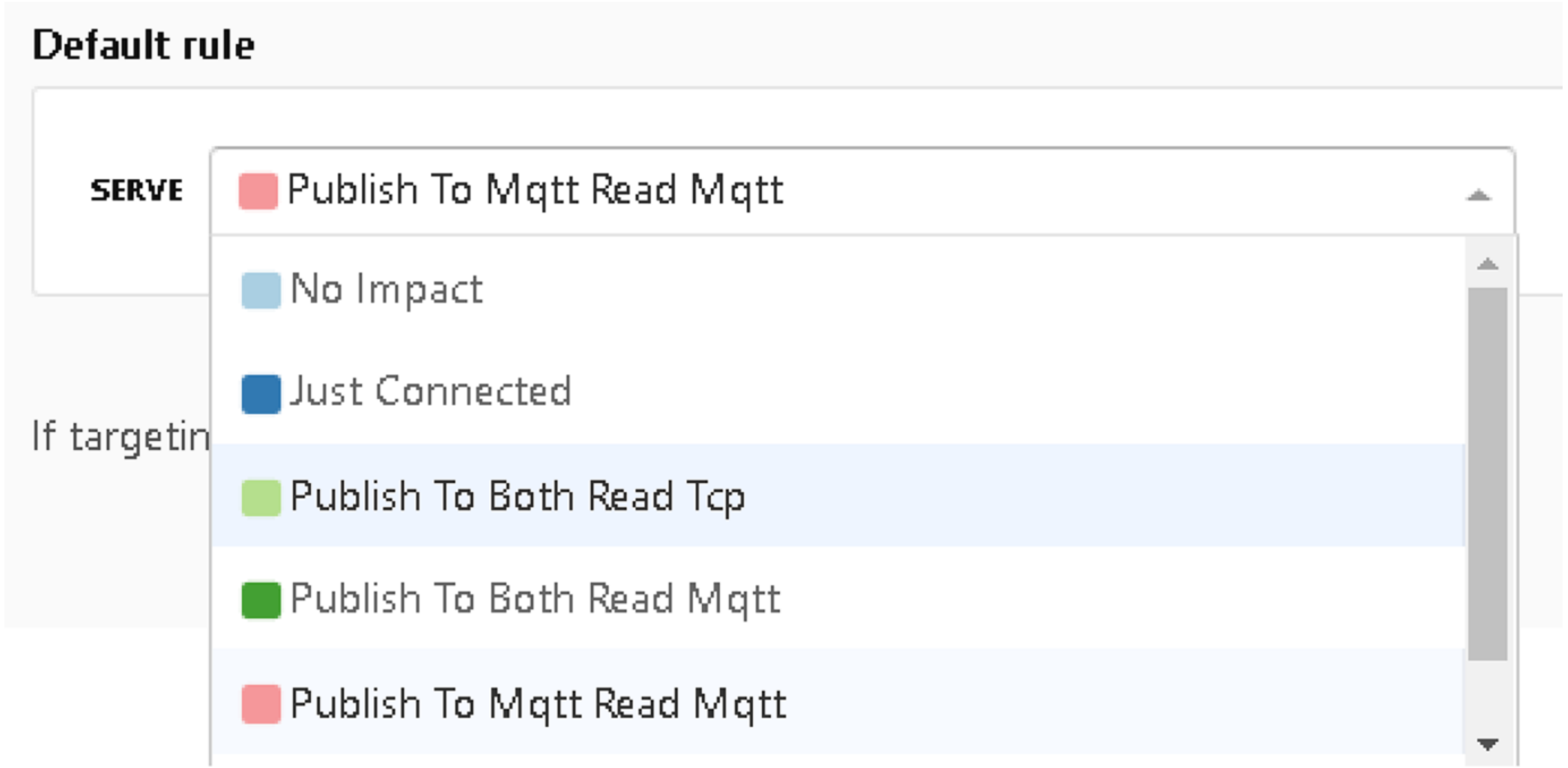
If your thing is working fine, it doesn't mean there are no problems.

Story 5:

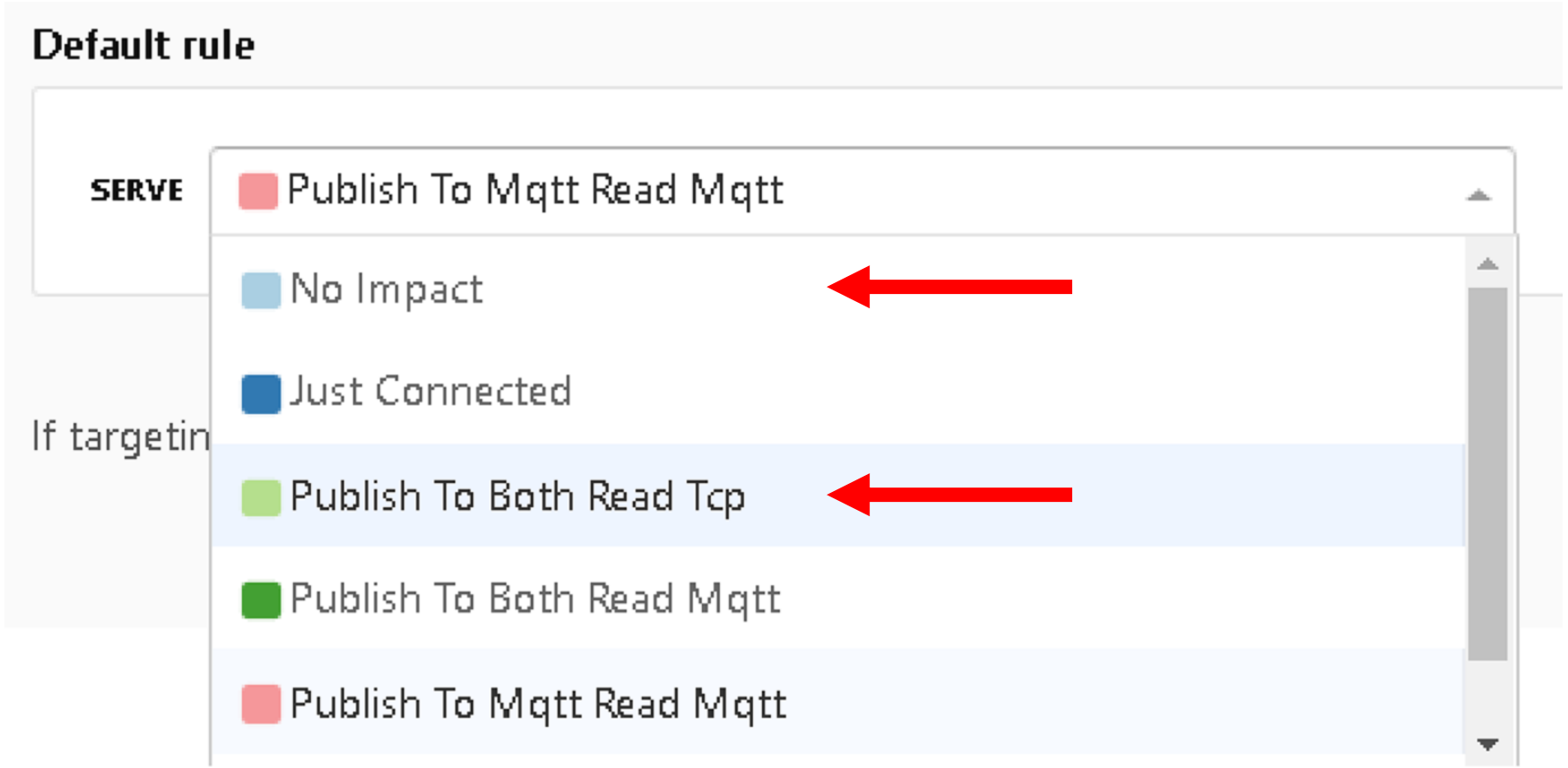
The calmest production failure ever

Story 5: The calmest production failure ever

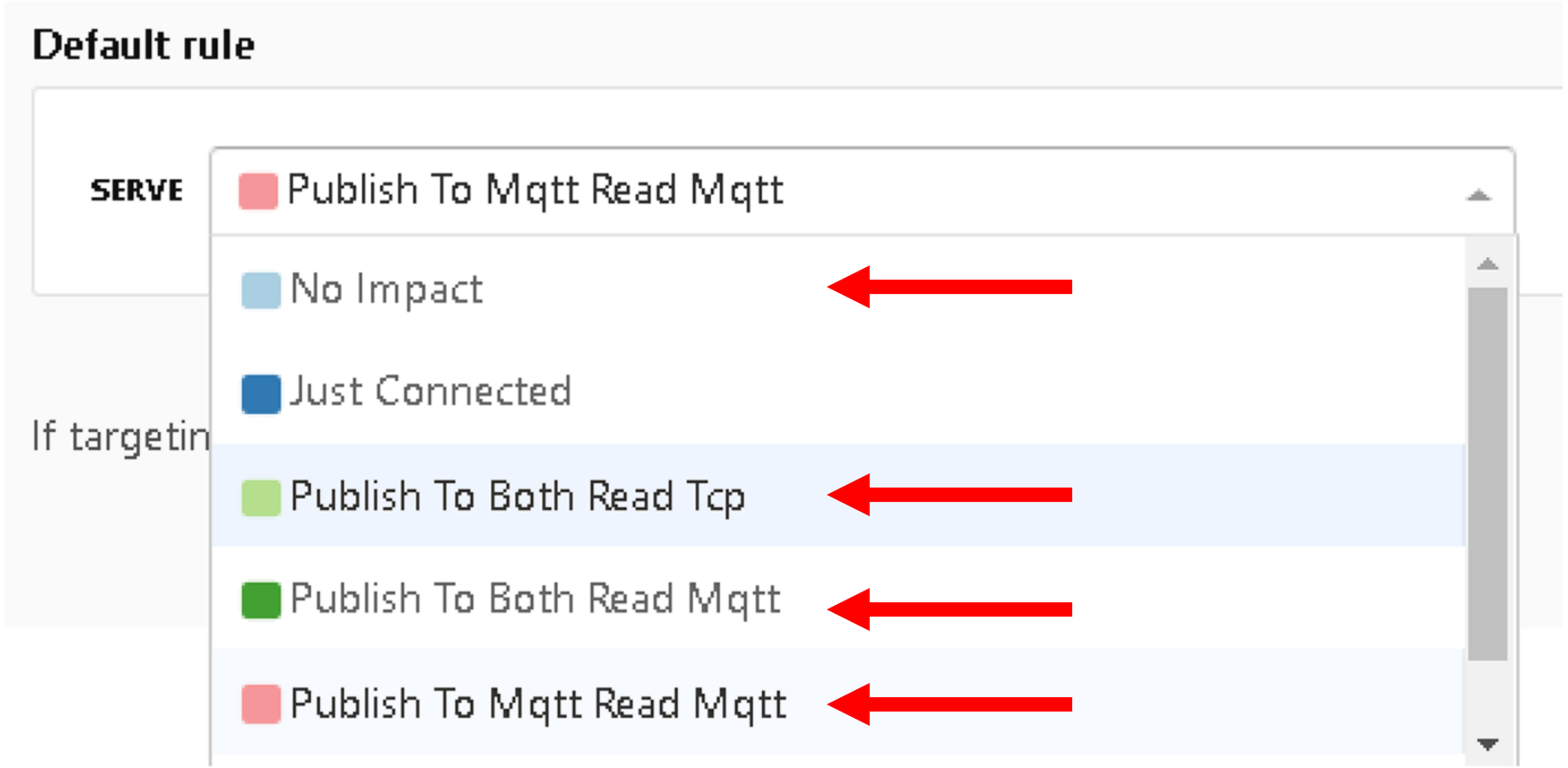
Story 5: The calmest production failure ever



Story 5: The calmest production failure ever



Story 5: The calmest production failure ever



Story 5: The calmest production failure ever

LESSON:

Safe roll-out and roll-back strategies allow you the luxury of thinking clearly during "emergencies".

Story Time Summary:

- You don't know what you don't know
- What's true today could change tomorrow
- Things out of your control can impact you
- Working fine != there are no problems
- Failure handling strategies → clear thinking

Story Time Summary:

Excellent testing would not have helped.

Conclusion:

How do I make "good testing" work for me?

1. Do good testing

- Test until you feel pretty confident
- Maybe focus on cases that are hard to monitor for

1. Do good testing

- Skip tests that are not "real" enough

- Skip tests that cost more than they're worth

- Skip tests where failure would not result in a bug fix

2. Plan for failure - roll out & roll back

- Is it possible to roll out our changes gradually?

- What's the fastest way we could roll back to an earlier version?

2. Plan for failure - detecting issues

- what information is needed to detect that something is going wrong?

- How will we be notified when things are going wrong?

- who is responsible for responding to error notifications?

2. Plan for failure - responding to issues

- what troubleshooting steps will we take when we get an error notification?

- what communications should occur while we are dealing with issues?

- Are there cases where fixes or solutions could be executed automatically?

2. Plan for failure

... plus many more!

Conclusion:

what does all this mean for me and my team?

1. Changes in focus

- Both devs and testers think about observability up front

1. Changes in focus

- Both devs and testers think about observability up front
- Use time not spent on last mile of testing to prep for failure handling

1. Changes in focus

- Both devs and testers think about observability up front
- Use time not spent on last mile of testing to prep for failure handling
- Test your monitoring

2. Changes in expectations

- Maybe you'll release more bugs

2. Changes in expectations

- Maybe you'll release more bugs
- You'll probably NOTICE more bugs

2. Changes in expectations

- Maybe you'll release more bugs
- You'll probably NOTICE more bugs
- You might be on-call (good motivation to invest in reliability and self-healing!)

2. Changes in success criteria

- Instead of "lower escaped defects", lower MTTR

2. Changes in success criteria

- Instead of "lower escaped defects", lower MTTR

- Instead of "how could we have missed this", consider whether it could be automatically resolved next time

Remember:

Be ready to handle failure; you won't find all the bugs anyway

Thanks!

@fletchertinam

tinafletcher.ca