MANAGING YOUR HEROES

The People Aspect of Monitoring
(a.k.a. Dealing with Outages and Failures)

Alex Solomon
alex@pagerduty.com

Nagios® World Conference North America

WHO AM I?

Alex Solomon

- Founder / CEO of PagerDuty
- Intersect Inc.
- Amazon.com

PAGERDUTY amazon.com

DEFINITIONS

Service **L**evel **A**greement (SLA)

Mean **T**ime **T**o **R**esolution (MTTR)

Mean Time To Response

Mean **T**ime **B**etween **F**ailures (MTBF)

OUTAGES

Can we prevent them?

"I'll have an ounce of prevention."

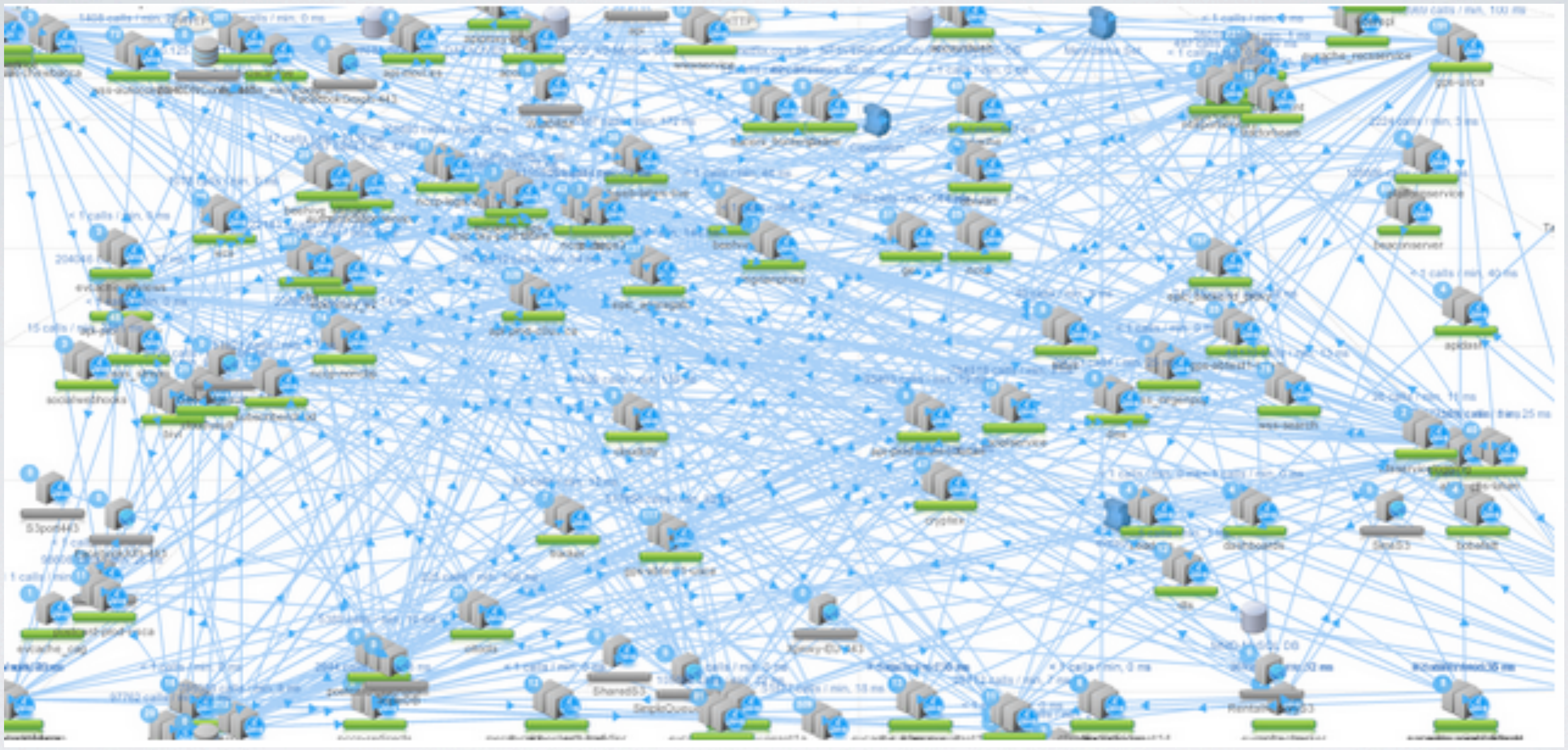
PREVENTING OUTAGES

✗ Single Points of Failure (SPOFs)

✓ Redundant systems

✗ Complex, monolithic systems

✓ Service-oriented architecture

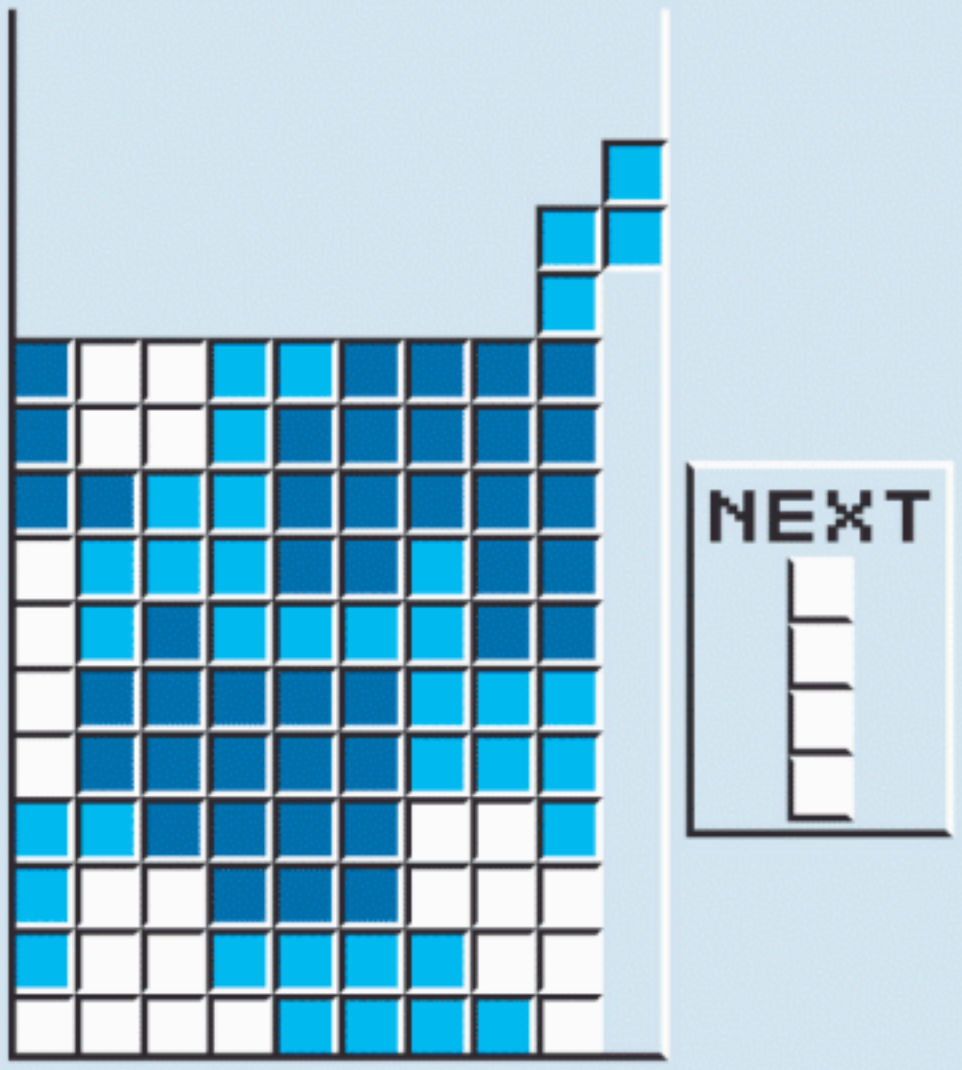


Netflix distributed SOA system

PREVENTING OUTAGES

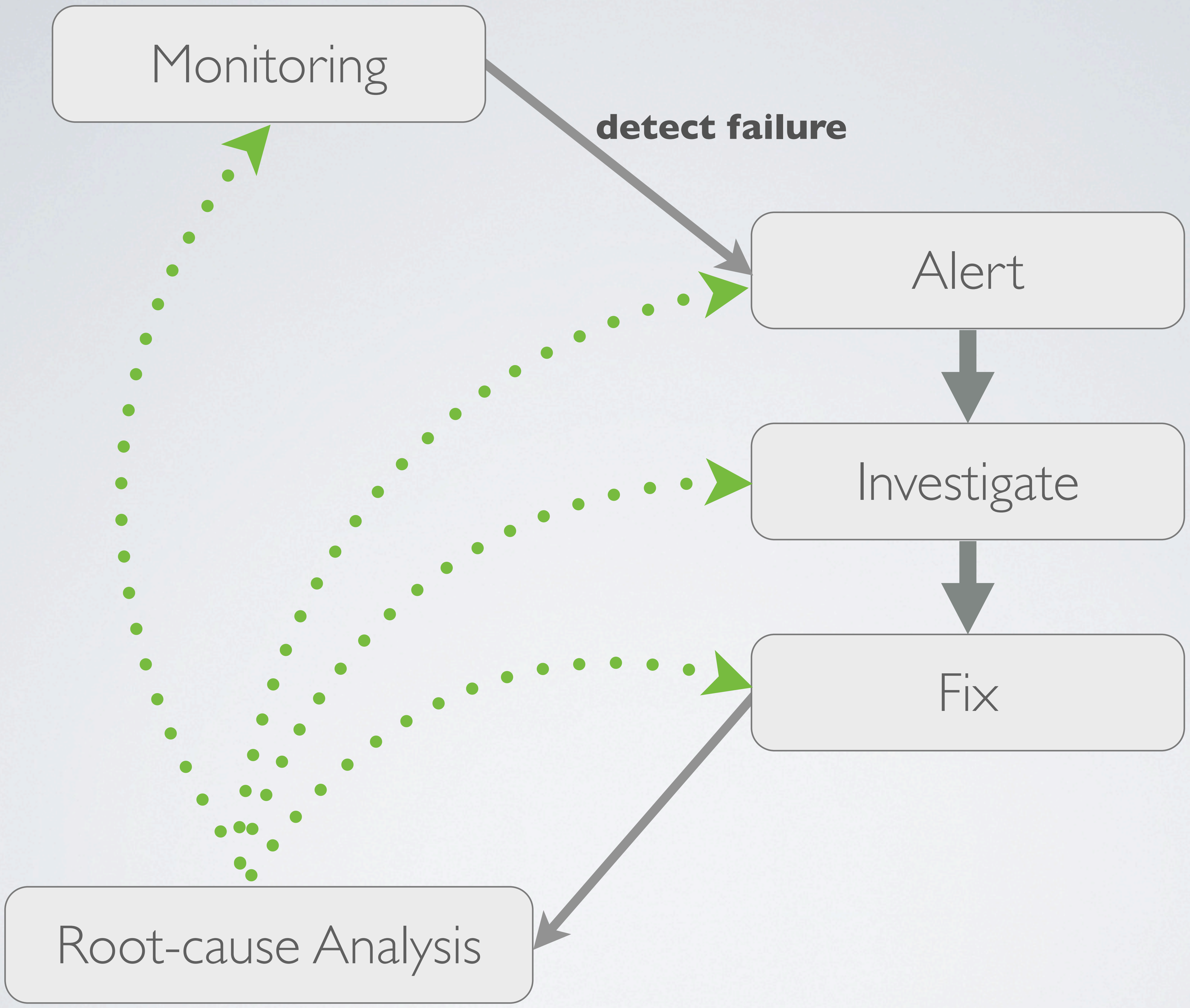
✗ Change

(not much you can do about this one)

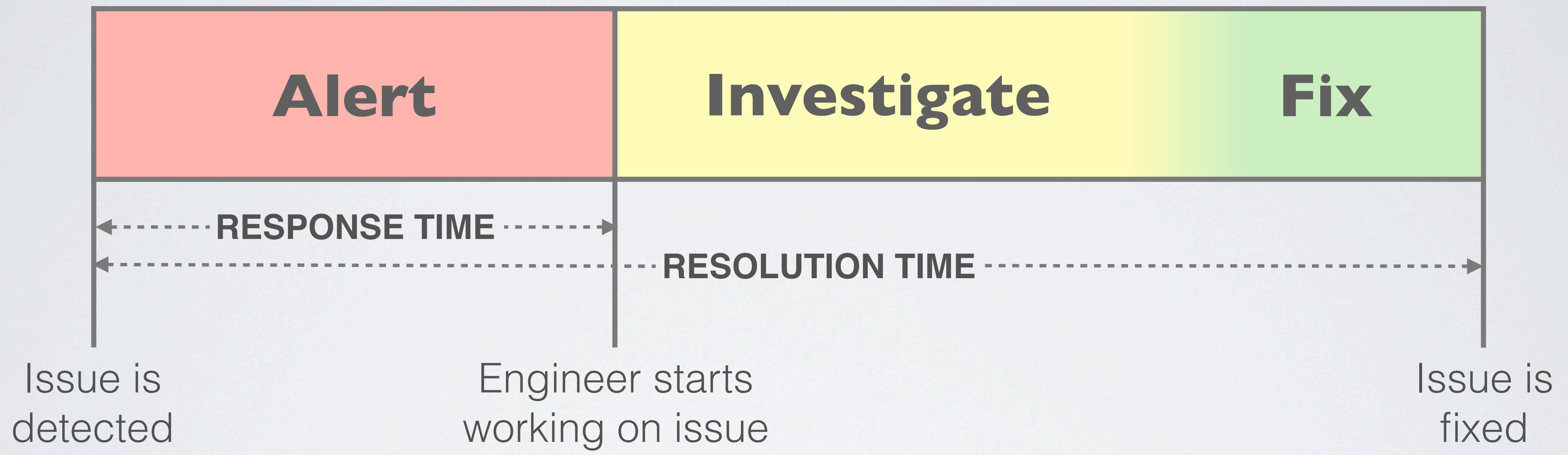


OUTAGES ~~SHIT~~ HAPPEN~~S~~

FAILURE LIFECYCLE


Monitoring
detect failure
Alert
Investigate
Fix
Root-cause Analysis

Critical Incident Timeline



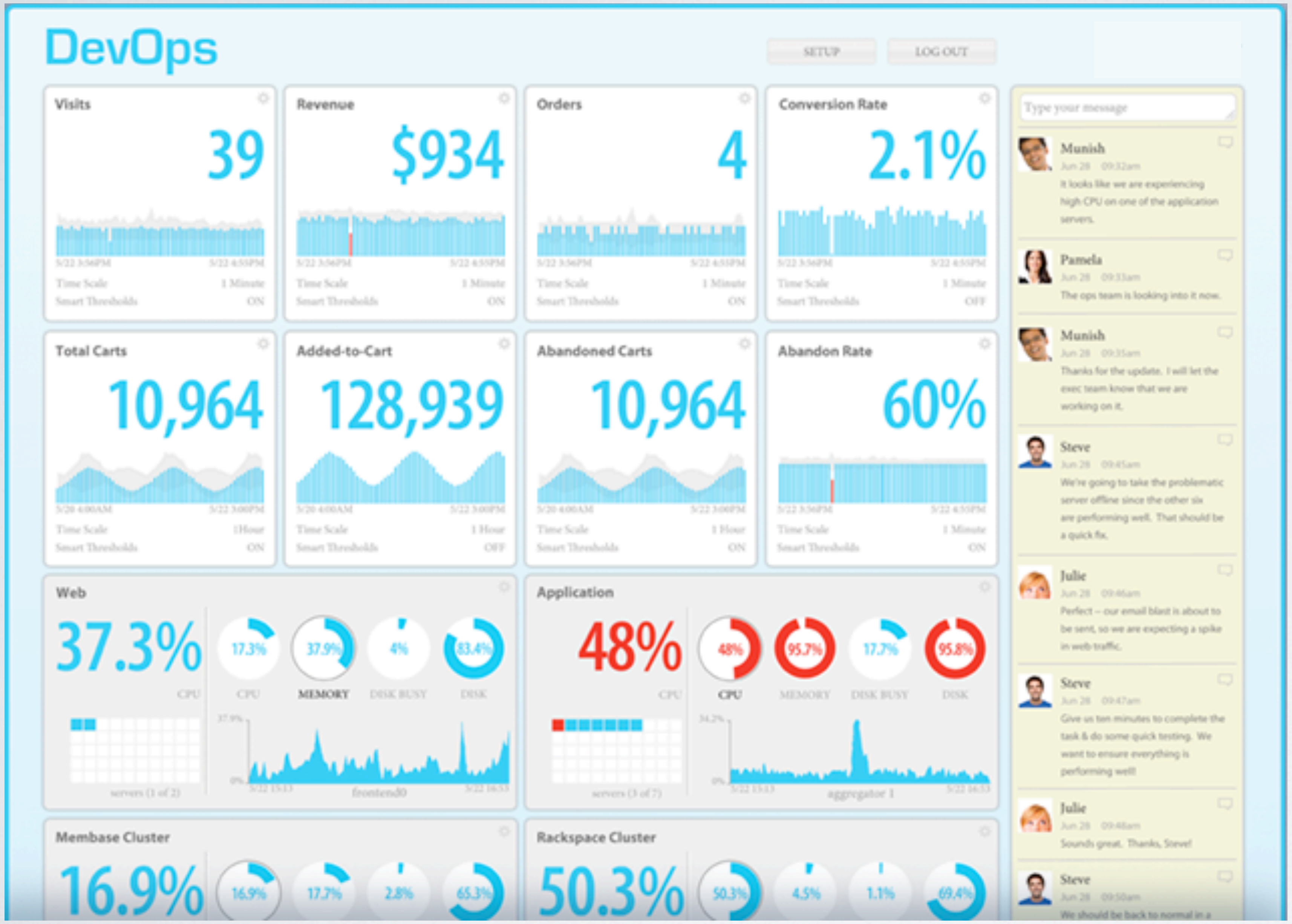
MONITOR

MONITOR EVERYTHING!

All levels of the stack

- Data center
- Network
- Servers
- Database
- Application
- Website
- Business Metrics

WHY MONITOR EVERYTHING?



Visits 39
5/22 3:56PM 5/22 4:55PM
Time Scale 1 Minute
Smart Thresholds ON

Revenue $934
5/22 3:56PM 5/22 4:55PM
Time Scale 1 Minute
Smart Thresholds ON

Orders 4
5/22 3:56PM 5/22 4:55PM
Time Scale 1 Minute
Smart Thresholds ON

Conversion Rate 2.1%
5/22 3:56PM 5/22 4:55PM
Time Scale 1 Minute
Smart Thresholds OFF

Total Carts 10,964
5/20 4:00AM 5/22 3:00PM
Time Scale 1Hour
Smart Thresholds ON

Added-to-Cart 128,939
5/20 4:00AM 5/22 3:00PM
Time Scale 1 Hour
Smart Thresholds OFF

Abandoned Carts 10,964
5/20 4:00AM 5/22 3:00PM
Time Scale 1 Hour
Smart Thresholds ON

Abandon Rate 60%
5/22 3:56PM 5/22 4:55PM
Time Scale 1 Minute
Smart Thresholds ON

Web 37.3% CPU
17.3% CPU, 37.9% MEMORY, 4% DISK BUSY, 83.4% DISK
servers (1 of 2) frontend0

Application 48% CPU
48% CPU, 95.7% MEMORY, 17.7% DISK BUSY, 95.8% DISK
servers (3 of 7) aggregator 1

Membase Cluster 16.9%
16.9%, 17.7%, 2.8%, 65.3%

Rackspace Cluster 50.3%
50.3%, 4.5%, 1.1%, 69.4%

Type your message

Munish
Jun 28 09:32am
It looks like we are experiencing high CPU on one of the application servers.

Pamela
Jun 28 09:33am
The ops team is looking into it now.

Munish
Jun 28 09:35am
Thanks for the update. I will let the exec team know that we are working on it.

Steve
Jun 28 09:45am
We're going to take the problematic server offline since the other six are performing well. That should be a quick fix.

Julie
Jun 28 09:46am
Perfect -- our email blast is about to be sent, so we are expecting a spike in web traffic.

Steve
Jun 28 09:47am
Give us ten minutes to complete the task & do some quick testing. We want to ensure everything is performing well!

Julie
Jun 28 09:48am
Sounds great. Thanks, Steve!

Steve
Jun 28 09:50am
We should be back to normal in a

Metrics!

Metrics!

Metrics!

TOOLS

- Internal monitoring (behind the firewall):
  - Nagios®
  - splunk>™
- External monitoring (SaaS-based):
  - New Relic.
  - pingdom
- Metrics:
  - Graphite or DATADOG

ALERT

Best Practice: Categorize alerts by severity.

SEVERITIES

Define severities based on business impact:

- **sev1** - large scale business loss
- **sev2** - small to medium business loss

2 critical severities

- **sev3** - no immediate business loss, customers may be impacted
- **sev4** - no business loss, no customers impacted

2 non-critical severities

Each severity level should have its own standard operating procedure (SOP):

- ***Who***
- ***How***
- ***Response time***

- **Sev1**: Major outage, all hands on deck
  - Notify the entire team via phone and SMS
  - Response time: 5 min
- **Sev2**: Critical issue
  - Notify the on-call person via phone and SMS
  - Response time: 15 min
- **Sev3**: Non-critical issue
  - Notify the on-call person via email
  - Response time: next day during business hours

- **Sev1** incidents
  - Rare
  - Rarely auto-generated
  - Frequently start as sev2 which are upgraded to sev1

- **Sev2** incidents
  - More common
  - Mostly auto-generated

- **Sev3** incidents
  - Non-critical incidents
  - Can be auto-generated
  - Can also be manually generated

- Severities can be ***downgraded*** or ***upgraded***
    - ex. **sev2 → sev1** (problem got worse)
    - ex. **sev1 → sev2** (problem was partially fixed)
    - ex. **sev2 → sev3** (critical problem was fixed but we still need to investigate root cause)

One more best-practice:

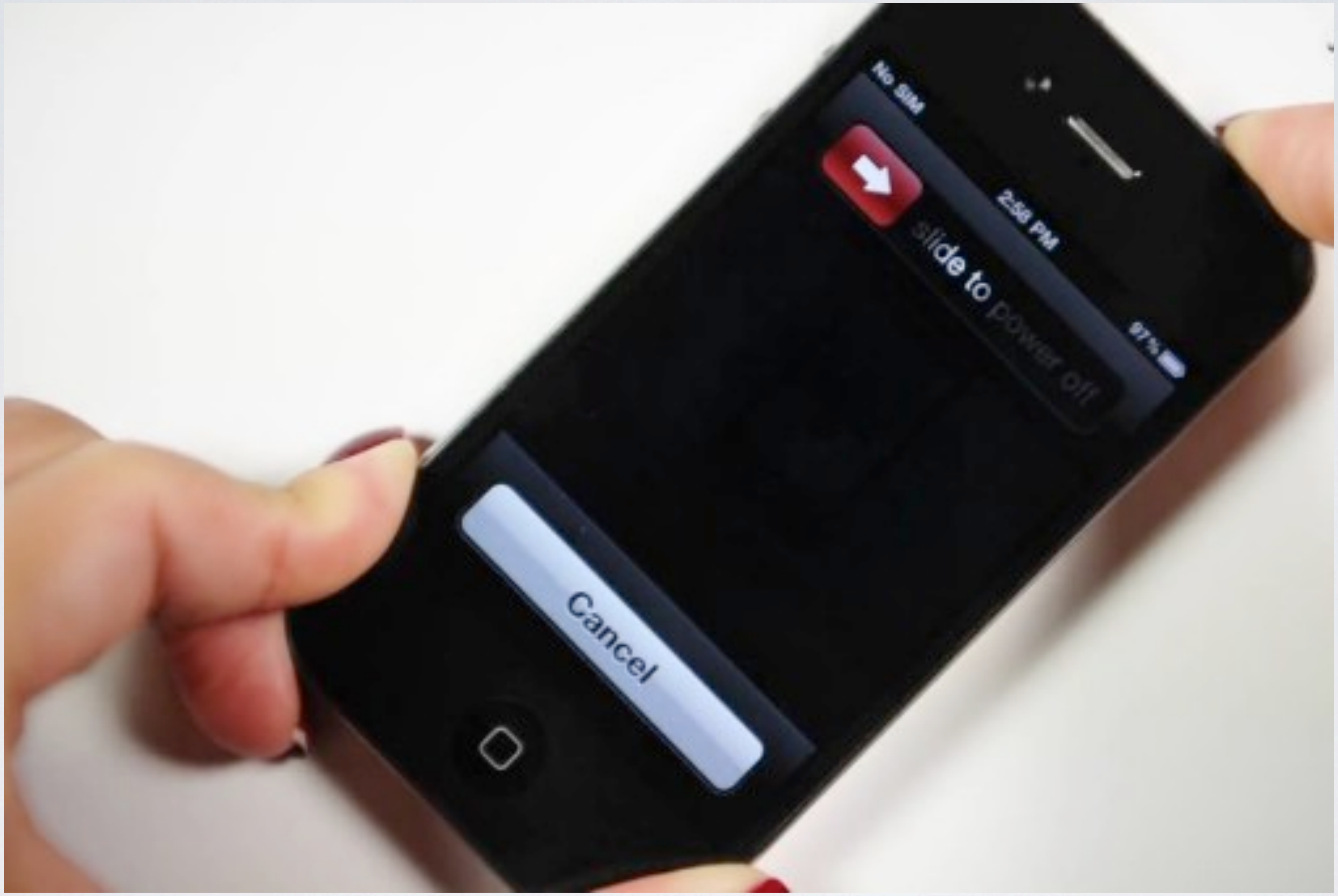
Alert ***before*** your systems fail completely

Main benefit of severities

Only page on ***critical issues*** (sev1 or 2)

Preserve sanity

Avoid "Peter and the wolf" scenarios



ON-CALL BEST PRACTICES

Person
Level

Team
Level

ON-CALL AT THE PERSON LEVEL

Cellphone

~~Cellphone~~

Smart phone

OR

AND

4G / 3G internet

4G hotspot

4G USB modem

3G/4G tethering

(don't forget your laptop)

Page multiple times until you respond

- **Time zero**: email and SMS
- **1 min later**: phone-call on cell
- **5 min later**: phone-call on cell
- **5 min later**: phone-call on landline
- **5 min later**: phone-call to girlfriend

Bonus: vibrating bluetooth bracelet

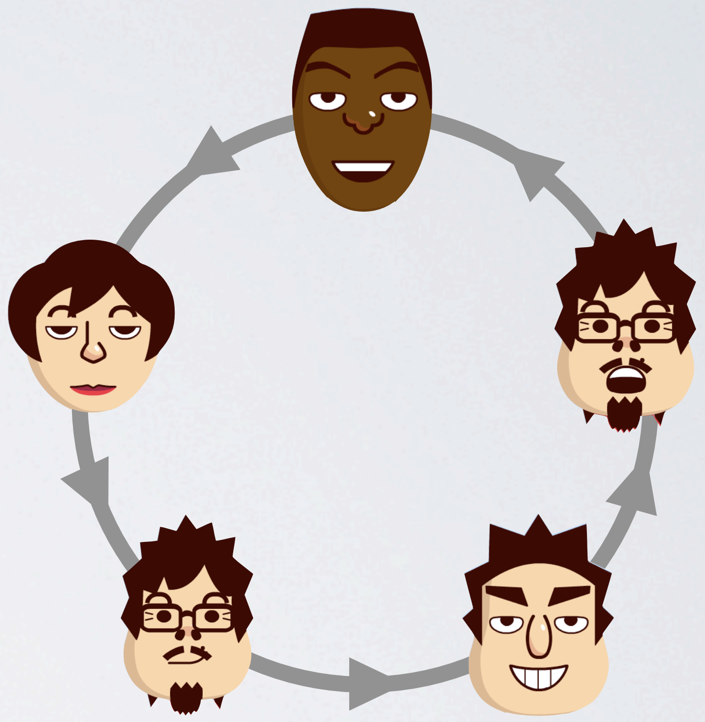
ON-CALL AT THE TEAM LEVEL

Rarely
~~***Do not***~~ send alerts to the entire team

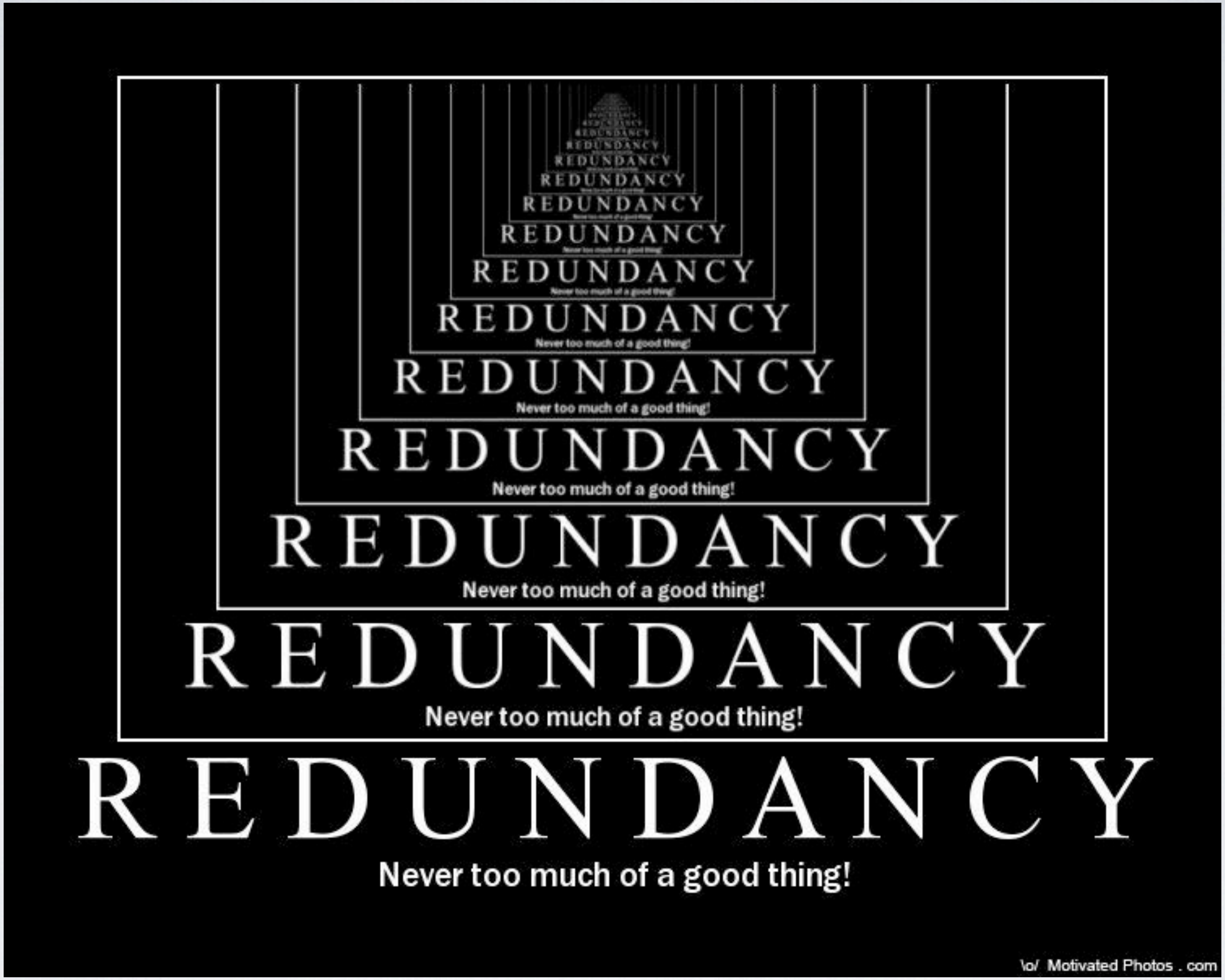
sev1 ***OK***
sev2 ***NO***

On-call schedules:

- *Simple* ***rotation-based*** *schedule*
    - ex. weekly - everyone is on-call for a week at a time
- Set up a ***follow-the-sun*** schedule
    - people in multiple timezones
    - no night-shifts


simple rotation

What happens if the on-call person doesn’t respond at all?



If you care about uptime, you need ***redundancy*** in your on-call.

Set up multiple on-call levels with automatic ***escalation*** between them:

Level 1: Primary on-call

↓ ***Escalate after 15 min***

Level 2: Secondary on-call

↓ ***Escalate after 20 min***

Level 3: Team on-call (alert entire team)

Best Practice: Put ***management*** in the on-call chain

Level 1: Primary on-call

Escalate after 15 min

Level 2: Secondary on-call

Escalate after 20 min

Level 3: Team on-call (alert entire team)

Escalate after 20 min

Level 4: Manager / Director

Best Practice: put ***software engineers*** in the on-call chain

- Devops model
- Devs need to own the systems they write
- Getting paged provides a ***strong incentive*** to engineer better systems

Best Practice: measure ***on-call performance***

"You can't improve what you don't measure."

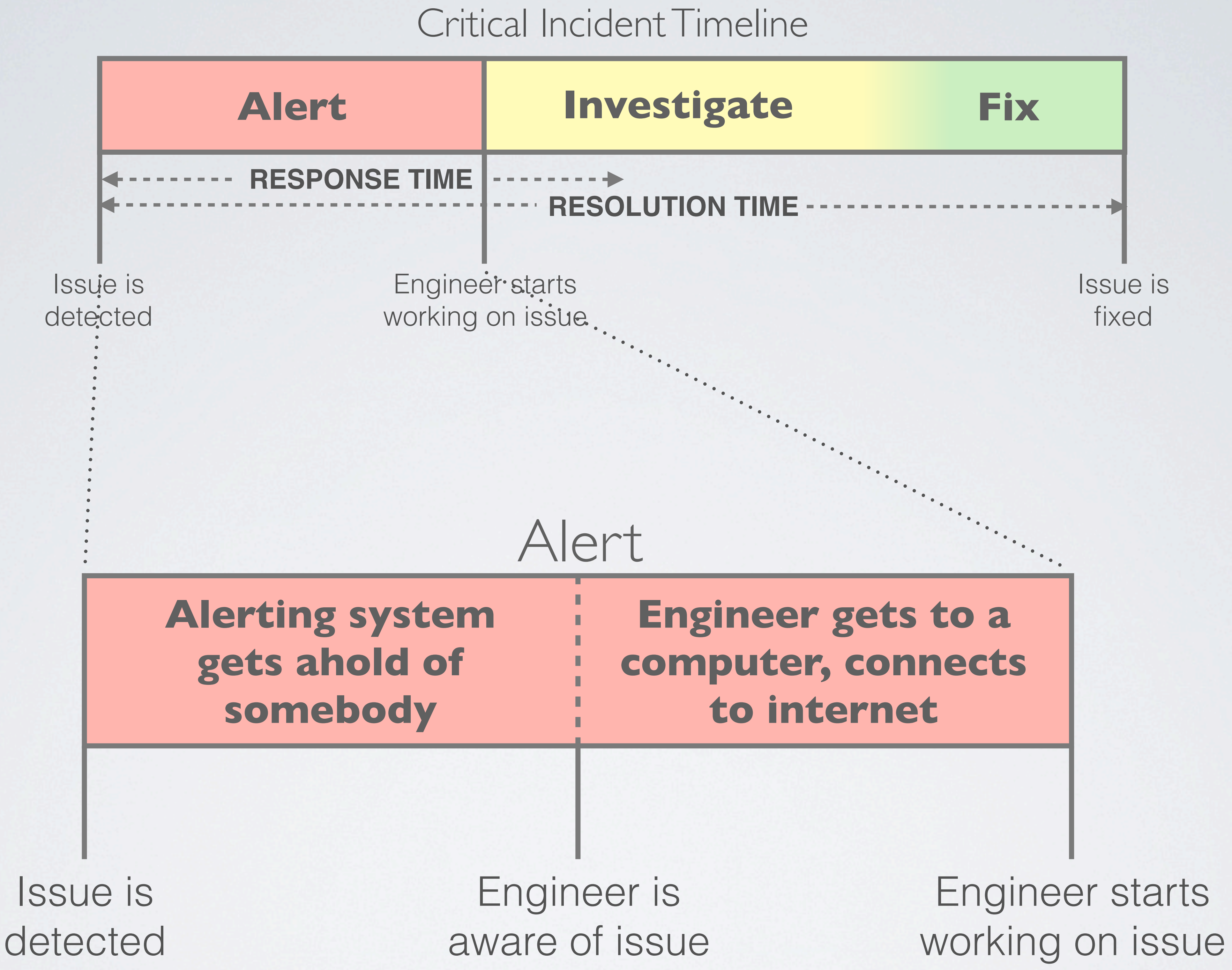
- Measure: mean-time-to-response
- Measure: % of issues that were escalated
- Set up policies to encourage good performance
  - Put managers in on-call chain
  - Pay people extra to do on-call

Network **O**perations **C**enter

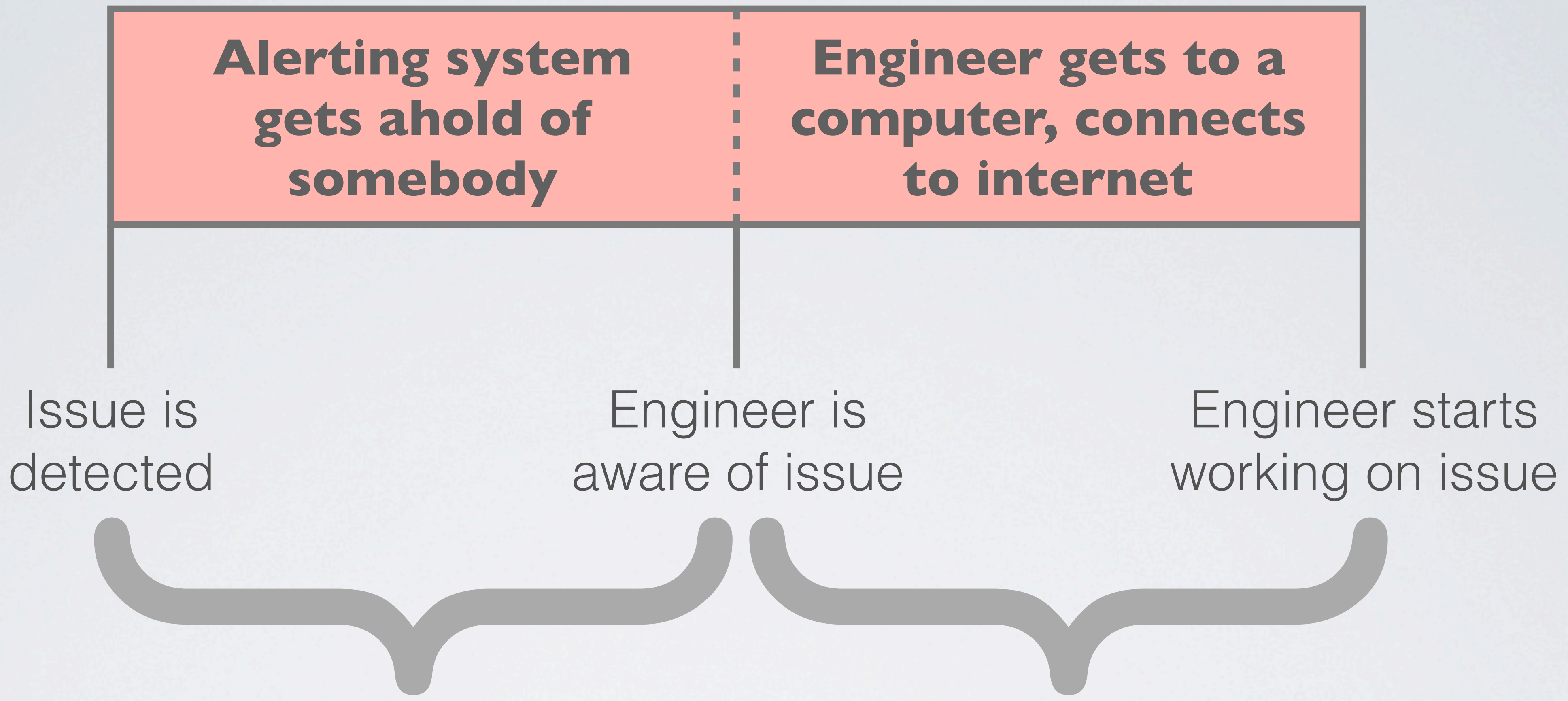
NOC with lots of Nagios goodness

NOCs:

- Reduce the mean-time-to-response drastically
- Expensive (staffed 24x7 with multiple people)
- Train NOC staff to fix a good %age of issues
- As you scale your org, you may want a hybrid on-call approach (where NOC handles some issues, teams handle other issues directly)


Critical Incident Timeline
Alert
Investigate
Fix
RESPONSE TIME
RESOLUTION TIME
Issue is detected
Engineer starts working on issue
Issue is fixed
Alert
Alerting system gets ahold of somebody
Engineer gets to a computer, connects to internet
Issue is detected
Engineer is aware of issue
Engineer starts working on issue

Alert


Alerting system gets ahold of somebody
Engineer gets to a computer, connects to internet
Issue is detected
Engineer is aware of issue
Engineer starts working on issue

How to minimize:

- Alert via phone & SMS
- Alert multiple times via multiple channels
- Failing that, escalate!
- Failing that, escalate to manager!

How to minimize:

- Carry 4G internet device + laptop at all times
- Set loud ringtone at night

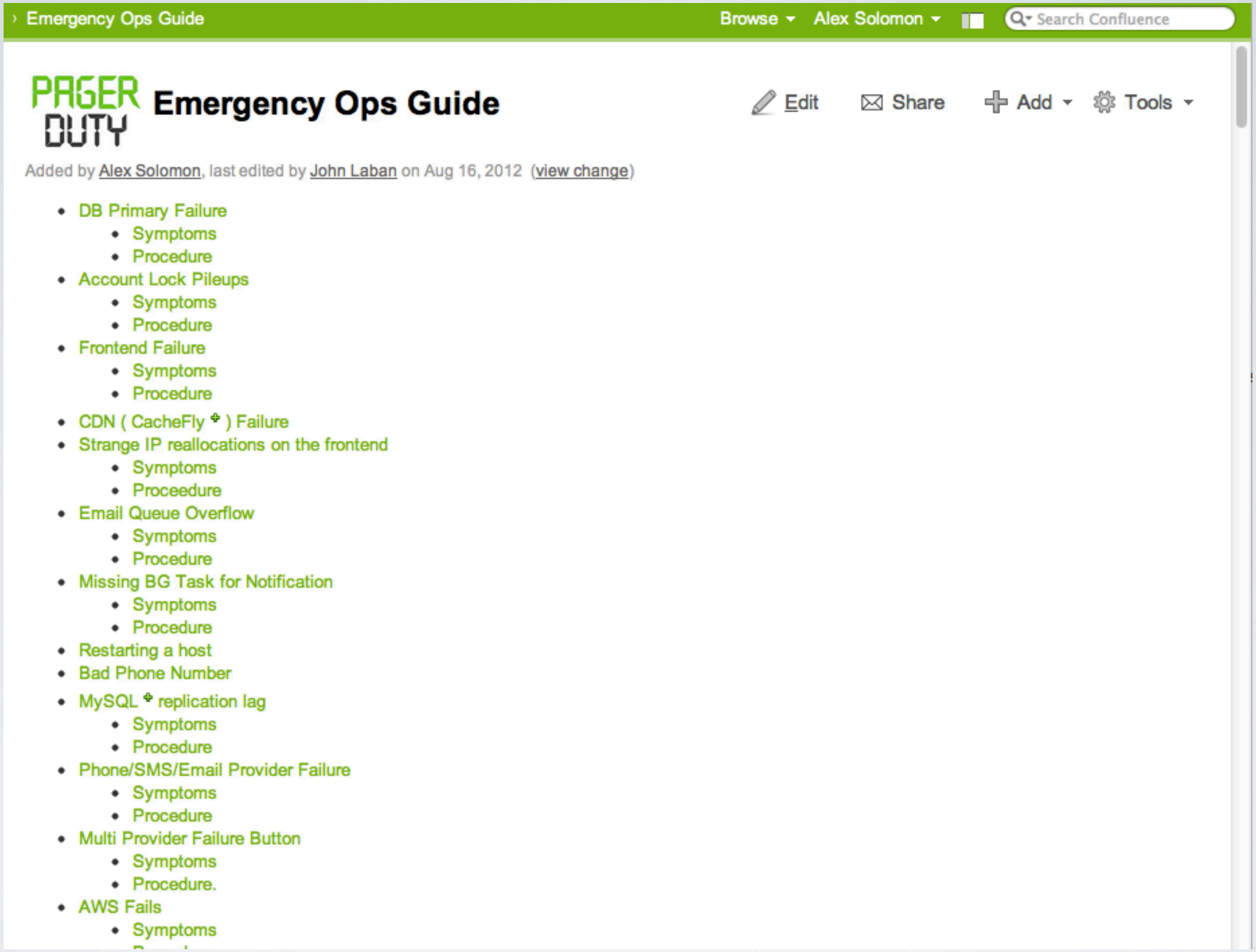
RESEARCH & FIX

How do we reduce the amount of time needed to investigate and fix?

Investigate
Fix

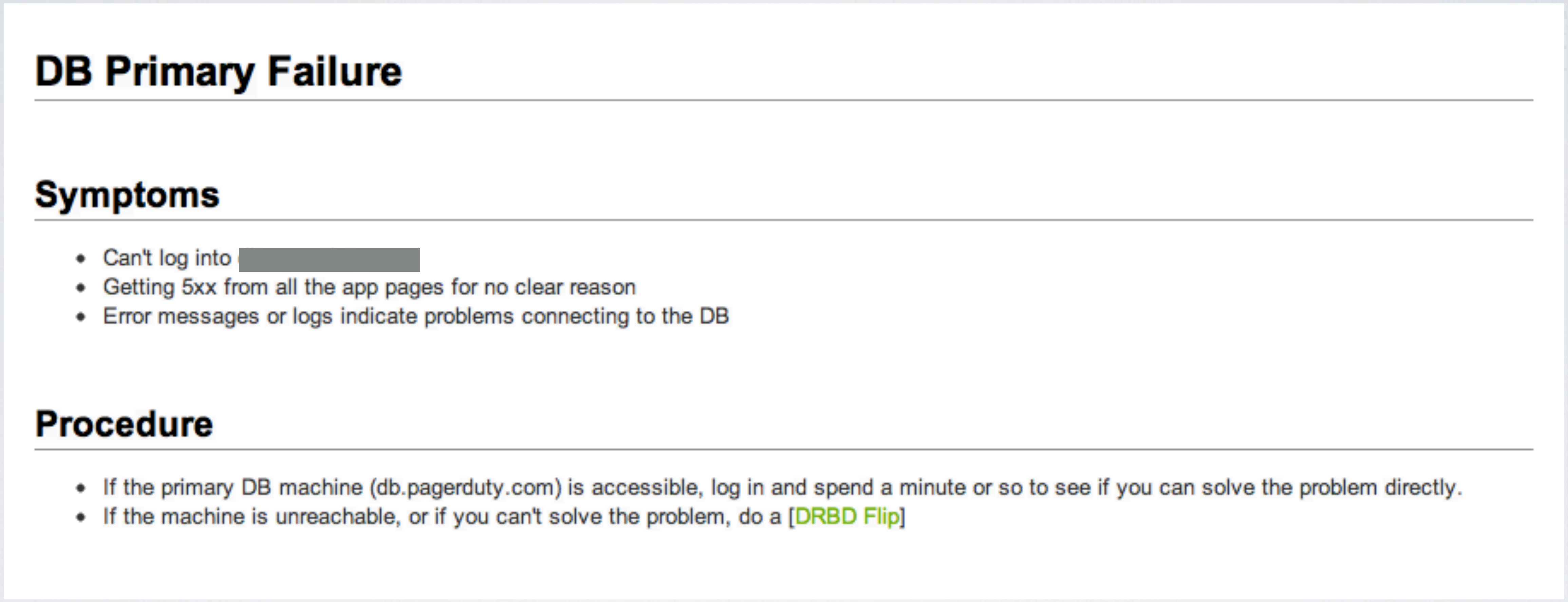
Set up an Emergency Ops Guide:

- When you encounter a new failure, document it in the Guide
- Document ***symptoms***, ***research steps***, ***fixes***
- Use a ***wiki***



Emergency Ops Guide

Added by Alex Solomon, last edited by John Laban on Aug 16, 2012 (view change)



DB Primary Failure

Symptoms

- Can't log into [redacted]
- Getting 5xx from all the app pages for no clear reason
- Error messages or logs indicate problems connecting to the DB

Procedure

- If the primary DB machine (db.pagerduty.com) is accessible, log in and spend a minute or so to see if you can solve the problem directly.
- If the machine is unreachable, or if you can't solve the problem, do a [DRBD Flip]

Automate fixes

or

Add more fault tolerance

You need the right tools:

- Tools to help you diagnose problems faster
  - Comprehensive monitoring, metrics and dashboards
  - Tools that help search for problems in log files quickly (ie. Splunk)
- Tools to help your team communicate efficiently
  - Voice: Conference bridge, Skype, Google Hangout
  - Chat: Hipchat, Campfire

Best Practice: Incident Commander

Incident Commander:

- Essential for dealing with sev1 issues
- In charge of the situation
  - Providers leadership, prevents analysis paralysis
  - He/she directs people to do things
  - Helps save time making decisions

Questions?

Alex Solomon
alex@pagerduty.com