Chie Shu
Software Engineer
chie@yelp.com

Dorothy Jung
Software Engineer
dorothy@yelp.com

Wenting Wang
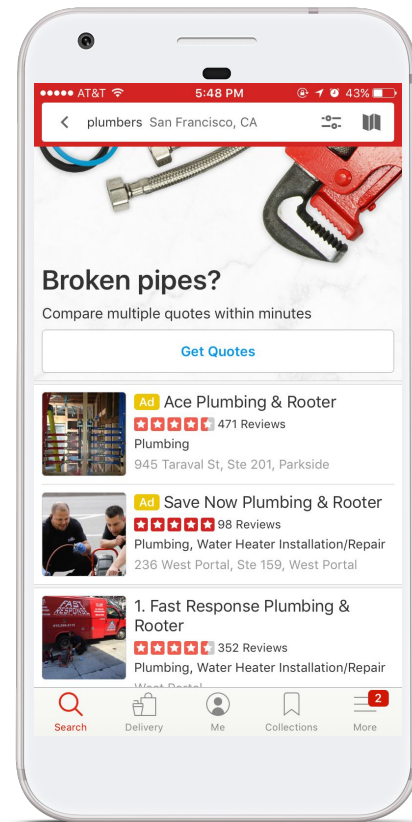Software Engineer
wwang@yelp.com

Yelp Local Ads

🏪 Connect people with great local businesses

💲 Advertiser billing and analytics

Our team's challenges

1. Financially critical systems

~90% of company revenue is from ads

2. Wears many hats 🎩

On-call + Feature + Infra

3. Owns systems with many different tech stacks

Makes being on-call more challenging

4. Majority of the team is new grad hires

Makes onboarding even more important

Our story

- 🏠 **Joined the team as new grad hires**

- ⚒️ **Learned how to be on-call the hard way...**

- ✳️ **Now mentoring other engineers**

Story time: Newbie on-call struggles

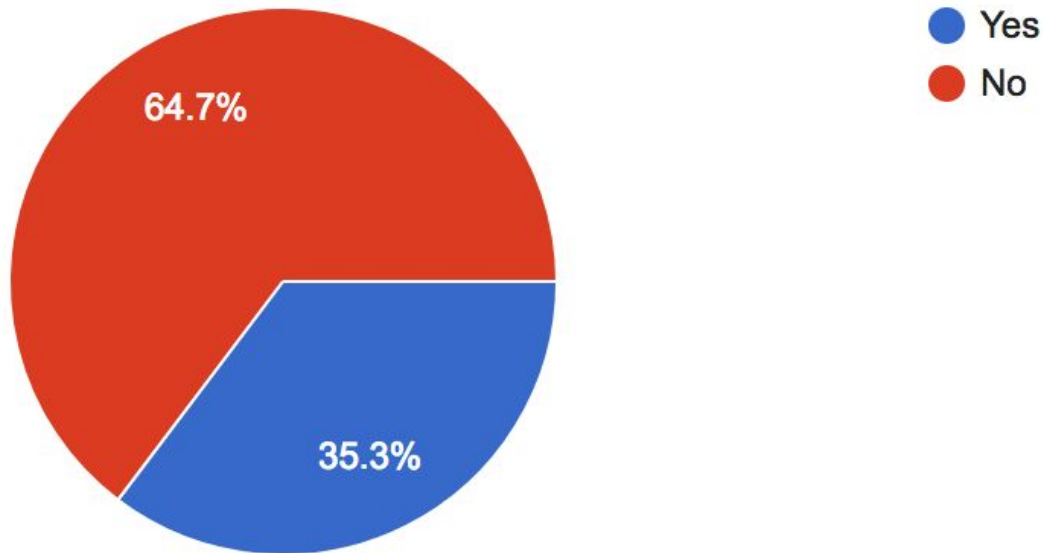"I didn't feel prepared"

"I wish there was better documentation"

"I wish I was more experienced"

Did you feel ready before going on-call for the first time?

Yes

No

64.7%

35.3%

Survey within Yelp Engineering (2018)

Agenda

1. **Onboarding strategies**

2. **Documentation for incident response**

3. **Learning from the past**

"I didn't feel prepared"

4 Common On-call Myths
4 Tips for Effective Onboarding

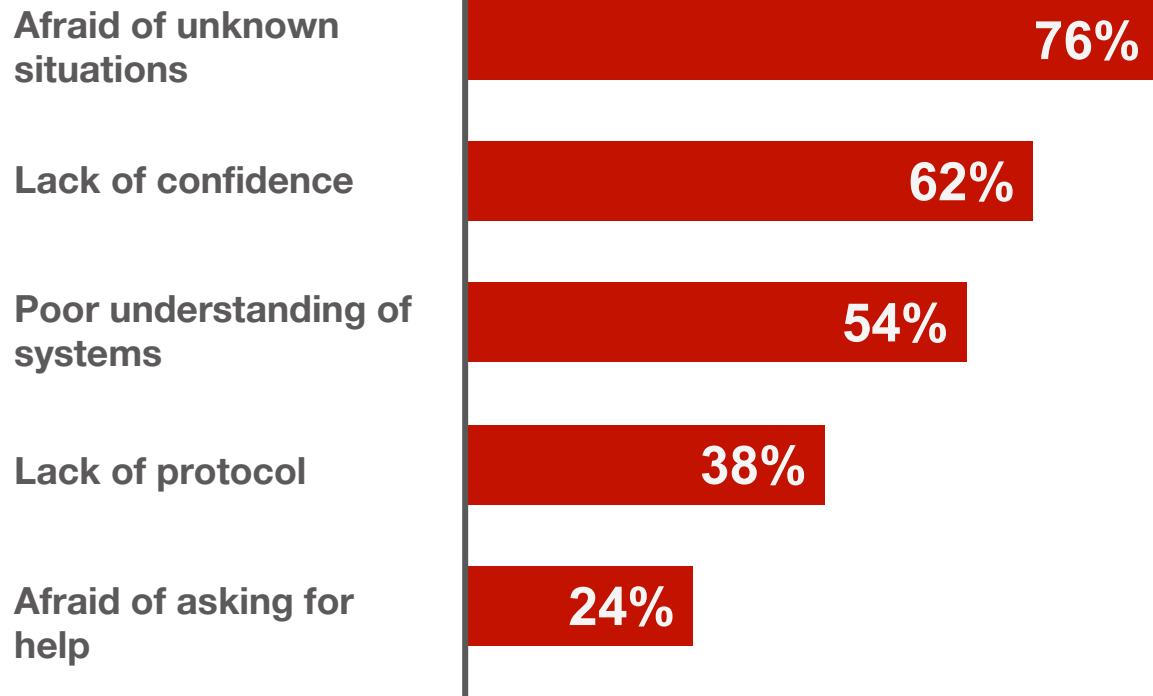# Why care about good onboarding?

Win 1: Makes your team scalable!

Win 2: Improve incident response

Win 3: Teaching is the best way to learn

Win 4: Confident new hires

Why didn't you feel ready?

Afraid of unknown situations — 76%

Lack of confidence — 62%

Poor understanding of systems — 54%

Lack of protocol — 38%

Afraid of asking for help — 24%

Survey within Yelp Engineering (2018)

4 Common Myths About On-calls

Myth #1
"I need to know everything"

You are not supposed to know everything

Myth #2
"I need to solve everything by myself"

You are supposed to ask for help

Myth #3
"I need to find the root cause"

You do not need to find a root cause as part of IR

Myth #4
"I need to make the best/long-term fix"

You are supposed to mitigate the issue

Setting the right expectations

If we are not supposed to know everything, what are we supposed to know?

The Basics

Step 1.
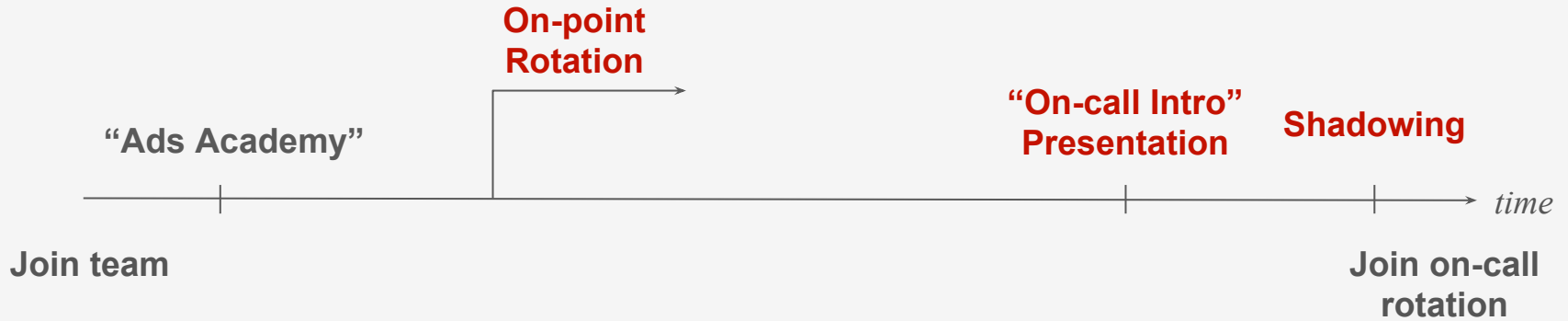**Be able to draw a mental picture of your system**

Step 2.
Have a high-level understanding of company infrastructure

Step 3.
Know the tools for investigation

My On-call "Training"

On-point Rotation

"Ads Academy"

"On-call Intro" Presentation

Shadowing

time

Join team

Join on-call rotation

What was good about my training?

✽ **It existed**

✽ **On-point rotation**

✽ **Shadowing**

What was difficult about my training?

* **Information dump**

* **No emphasis on connections between systems**

* **No emphasis on investigation/debugging tools**

4 Tips for Effective Onboarding

Tip #1
Avoid information overload

Tip #2
Make it relevant / Provide context

Tip #3
Focus on tools

Tip #4
Keep learning

Some things we do

- "Infratalks"
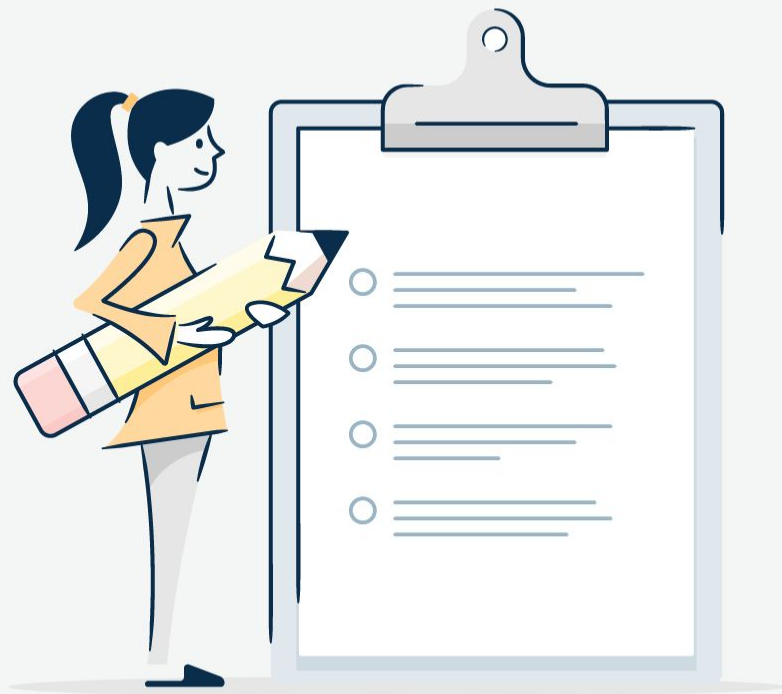- Brown bag sessions
- Wargames
- Little day-to-day things

What can you take home?
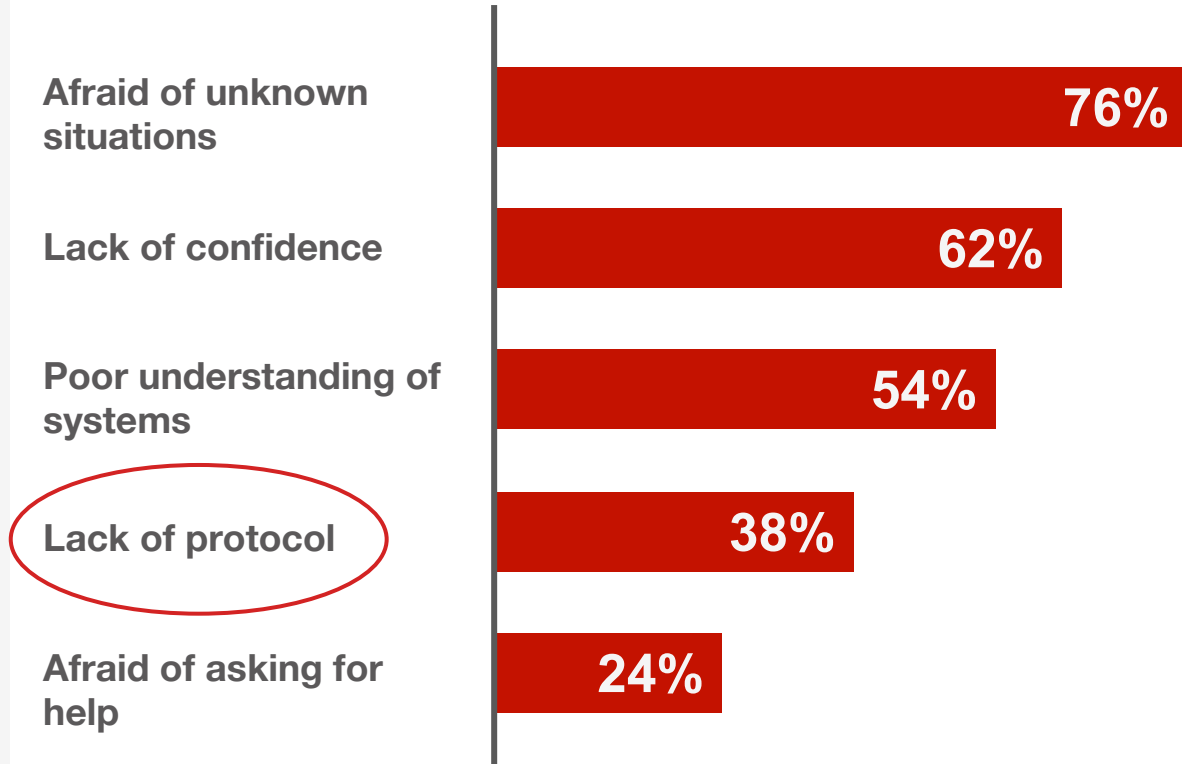
✻ **Start a new program (yay!)**

✻ **Tweak an existing process**

✻ **Change how you organize a presentation**

✻ **Change the way you mentor**

"I wish there was better documentation"

Why didn't you feel ready?

Afraid of unknown situations — 76%

Lack of confidence — 62%

Poor understanding of systems — 54%

Lack of protocol — 38%

Afraid of asking for help — 24%

Survey within Yelp Engineering (2018)

How would you improve the on-call training process?

70% Reviewed the team's runbooks before going on-call

"Update and improve documentation and runbooks"

"More documentation"

"Better documentation"

"Clear protocol of pages we can get and how to handle them"

"Runbooks should be obvious to find and execute. At 3 AM you need dummy-proof instructions."

Survey within Yelp Engineering (2018)

What is a **runbook**?

❋ **Quick guide in an emergency**

❋ **Step-by-step protocol for incident response**

❋ **Updated regularly in tandem with code changes**

❋ **Good for common cases**

Know "standard procedures" before being able to improvise

Why it's **useful**

❊ **Increase efficiency**

❊ **Reduce nervousness**

❊ **Stand-in for a mentor or backup**

❊ **Continuous learning**

2 Types of Runbooks

❄ **Technical**

Recover a failing system or process

❄ **Non-technical/Human process**

Communicating with stakeholders, general protocol or guidelines

Technical
Runbooks

✢ **Pager playbook**
   Maps every alert to a step-by-step guide for "what do to when…"

✢ **Disaster recovery plans**
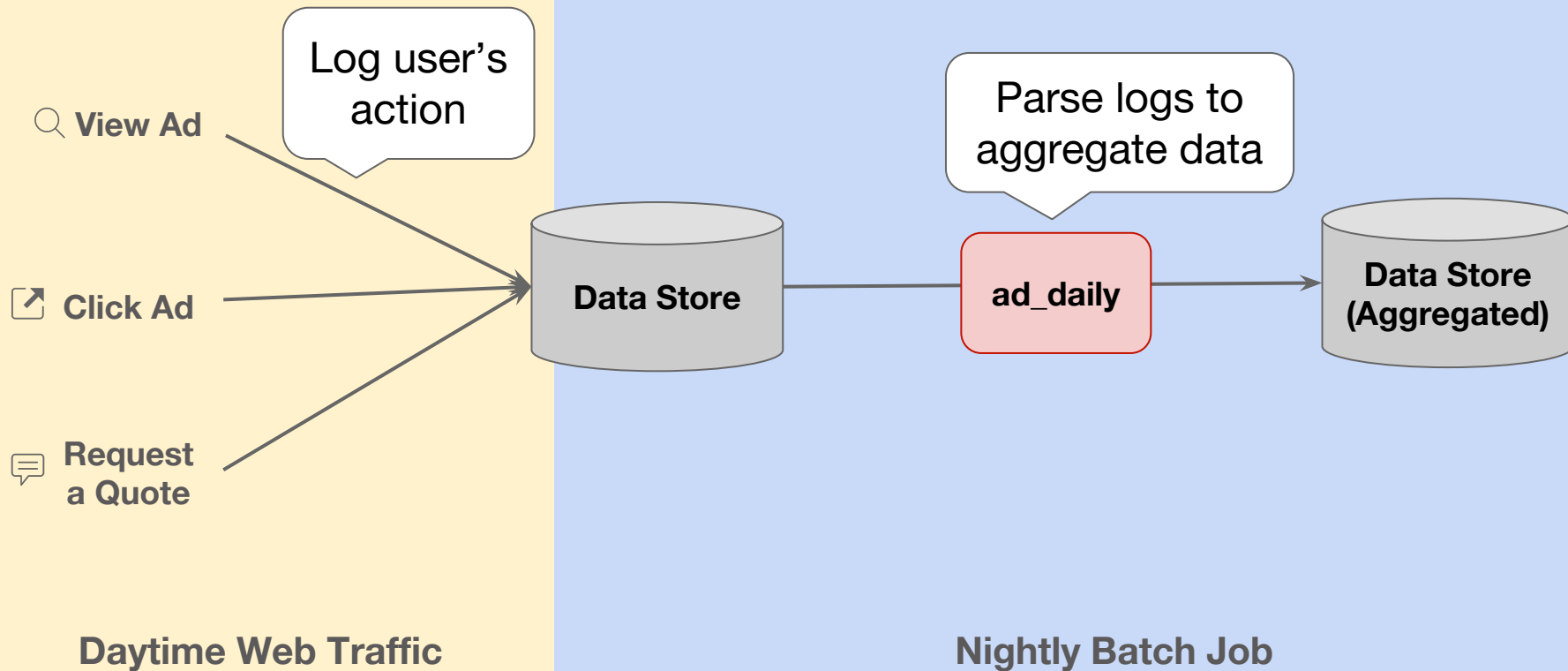
✢ **Impact assessment**
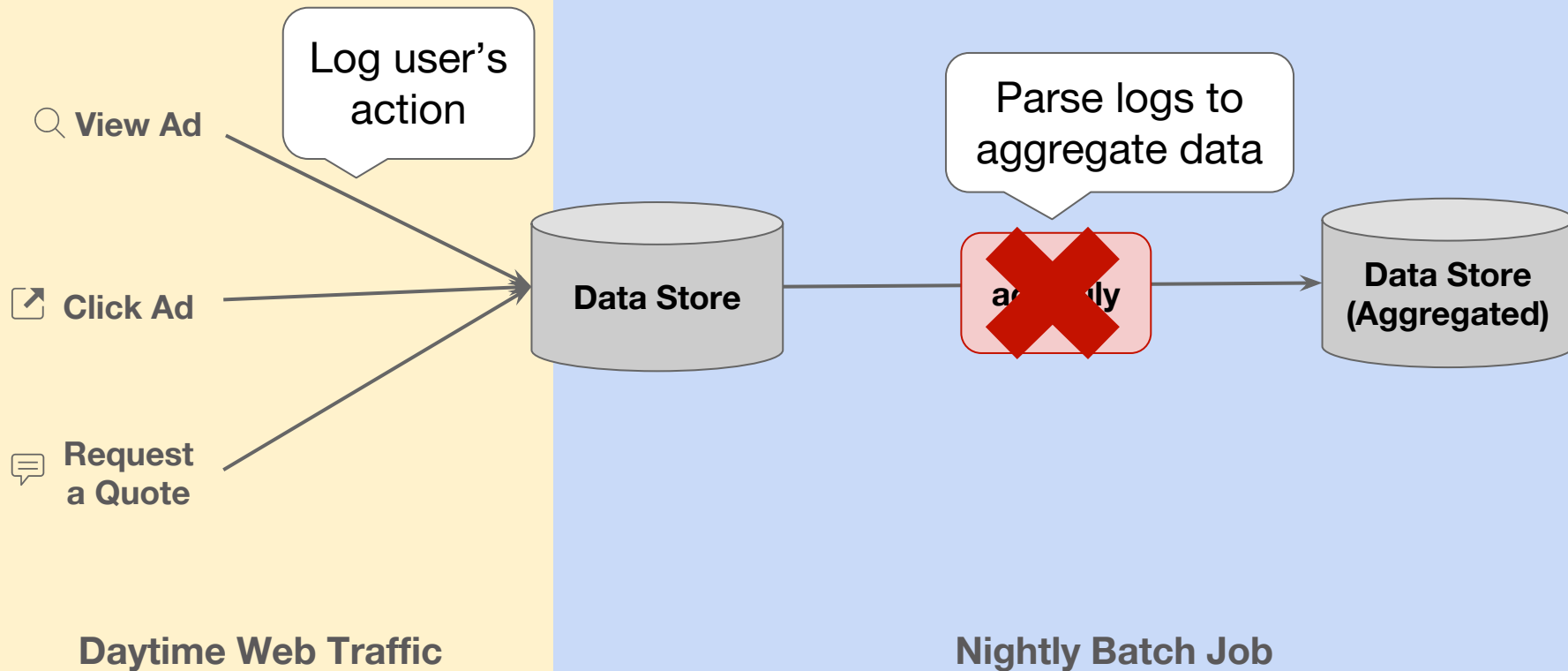
✢ **How to rollback or revert**

✢ **Links to learn more**
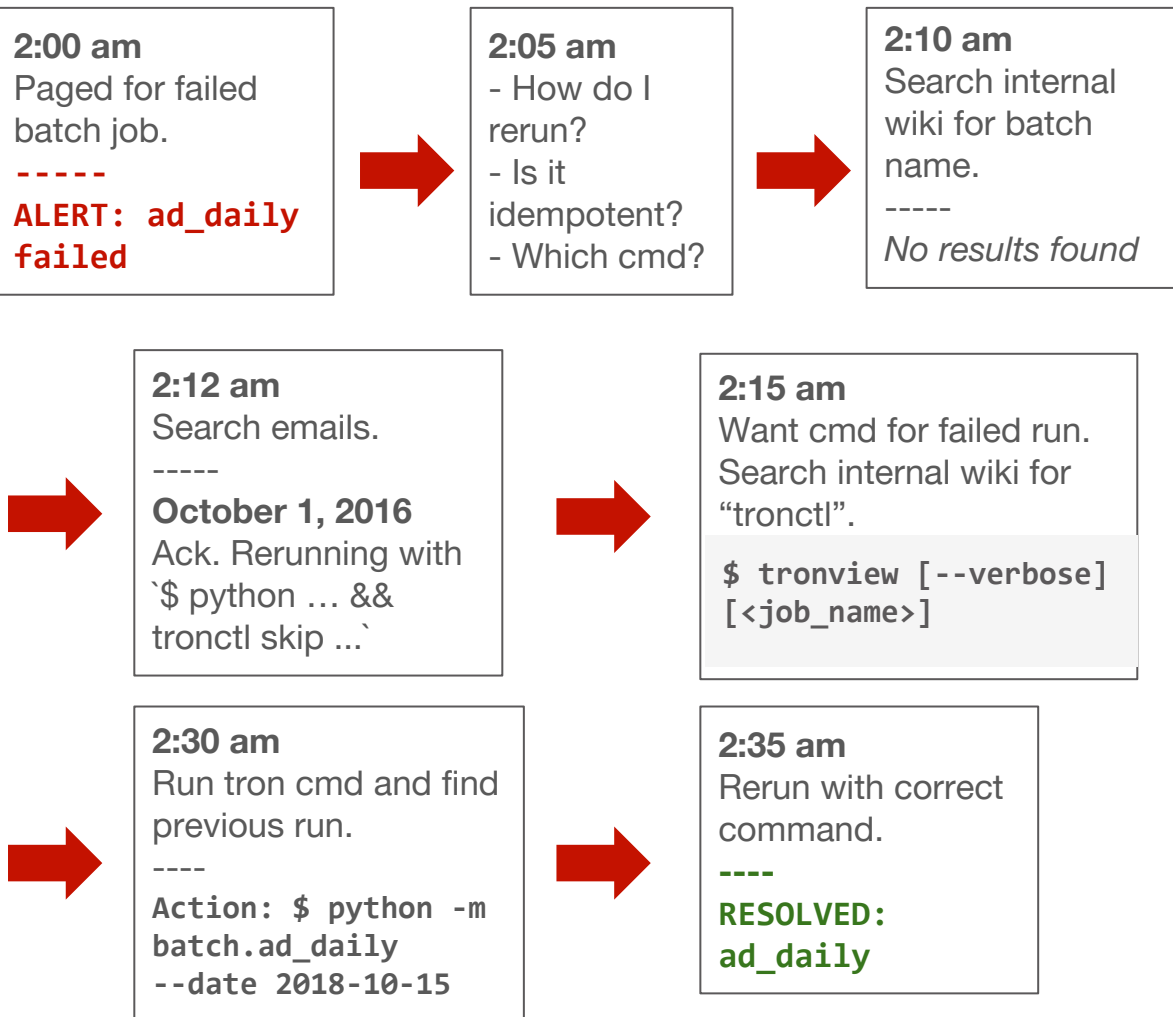
Example:
Technical runbook

Story time: Being on-call without a runbook

TTR: 35 min

2:00 am
Paged for failed batch job.

ALERT: ad_daily failed

➡️

2:05 am
- How do I rerun?
- Is it idempotent?
- Which cmd?

➡️

2:10 am
Search internal wiki for batch name.

No results found

➡️

2:12 am
Search emails.

October 1, 2016
Ack. Rerunning with `$ python … && tronctl skip …`

➡️

2:15 am
Want cmd for failed run. Search internal wiki for "tronctl".

```
$ tronview [--verbose]
[<job_name>]
```

➡️

2:30 am
Run tron cmd and find previous run.

Action: $ python -m batch.ad_daily --date 2018-10-15

➡️

2:35 am
Rerun with correct command.

RESOLVED: ad_daily

Story time: Being on-call with a runbook

2:00 am
Paged for failed batch job.

ALERT: ad_daily failed

```
Traceback:
File /yelp/…:
      …
Exception: EMR
Step 1 Failed
```

2:05 am
- How do I rerun?
- Is it idempotent?
- Which cmd?

2:10 am
Search internal wiki for batch name.
-----
1 result found
[Ads] Runbooks - Operations

Story time: Being on-call with a runbook

Runbooks - Operations

- General recovery tips
  - Campaigns not in ad store
  - Errors in ad templates
- Nagios
  - Background
  - Updating alerts
  - Alerts
- ad_daily (tron job)
  - man tronview and man tronctl to understand how to use tron.
  - 1. Identify which run failed
  - 2. Identify which action failed
  - 3. Fix/retry broken actions
  - Specific Batches
    - calculate_ad_analytics
    - calculate_ad_spend
    - business_ad_control
    - calculate_ad_stats
    - apply_scheduled_budget_changes
- Reports
- Rerunning procedures
  - Identify which days need to be rerun
  - Identify which batches need to be rerun
- Gearman
  - View the logging output of the gearman workers
  - View the number of gearman workers and the number of jobs in the queue
  - Adding and removing gearman workers for particular queues
  - Clearing out a queue

**Story time:
Being on-call with
a runbook**

Alerts

When responding to a Nagios alert, make sure you ACKNOWLEDGE the action. ACKNOWLEDGE that an issue is known by looking for the guy with the shovel under "Service Commands" on the right. Alerts that go through pagerduty can be acknowledged directly within pagerduty.

NOTE: This section would benefit a lot from having our actual alerts listed and detailed here.

Story time: Being on-call with a runbook

Runbooks - Operations

- General recovery tips
  - Campaigns not in ad store
  - Errors in ad templates
- Nagios
  - Background
  - Updating alerts
  - Alerts
- ad_daily (tron job)
  - man tronview and man tronctl to understand how to use tron.
  - 1. Identify which run failed
  - 2. Identify which action failed
  - 3. Fix/retry broken actions
  - Specific Batches
    - calculate_ad_analytics
    - calculate_ad_spend
    - business_ad_control
    - calculate_ad_stats
    - apply_scheduled_budget_changes
- Reports
- Rerunning procedures
  - Identify which days need to be rerun
  - Identify which batches need to be rerun
- Gearman
  - View the logging output of the gearman workers
  - View the number of gearman workers and the number of jobs in the queue
  - Adding and removing gearman workers for particular queues
  - Clearing out a queue

ad_daily (tron job)

man tronview and man tronctl to understand how to use tron

**Story time:
Being on-call with
a runbook**

1. Identify which run failed
   `$ tronview ad_daily` (identify the Run ID of today's failed run)

2. Identify which action failed
   `$ tronview ad_daily.XX` (where XX is the Run ID)
Look for which action(s) have a FAIL next to them. These need to be re-run by hand.

3. Fix/retry broken actions

If a batch died due to an EMR, DB, or other intermittent issue, attempt to run the action manually

If a batch died due to a logic error, push a fix and run the action manually

**Story time:
Being on-call with
a runbook**

3. Fix/retry broken actions

If a batch died due to an EMR, DB, or other intermittent issue, attempt to run the action manually

If a batch died due to a logic error, push a fix and run the action manually

To run manually, read the command line printed in this output. It's between the "!Node:" and "Requirements:" lines. You'll have to execute this as batch yourself.
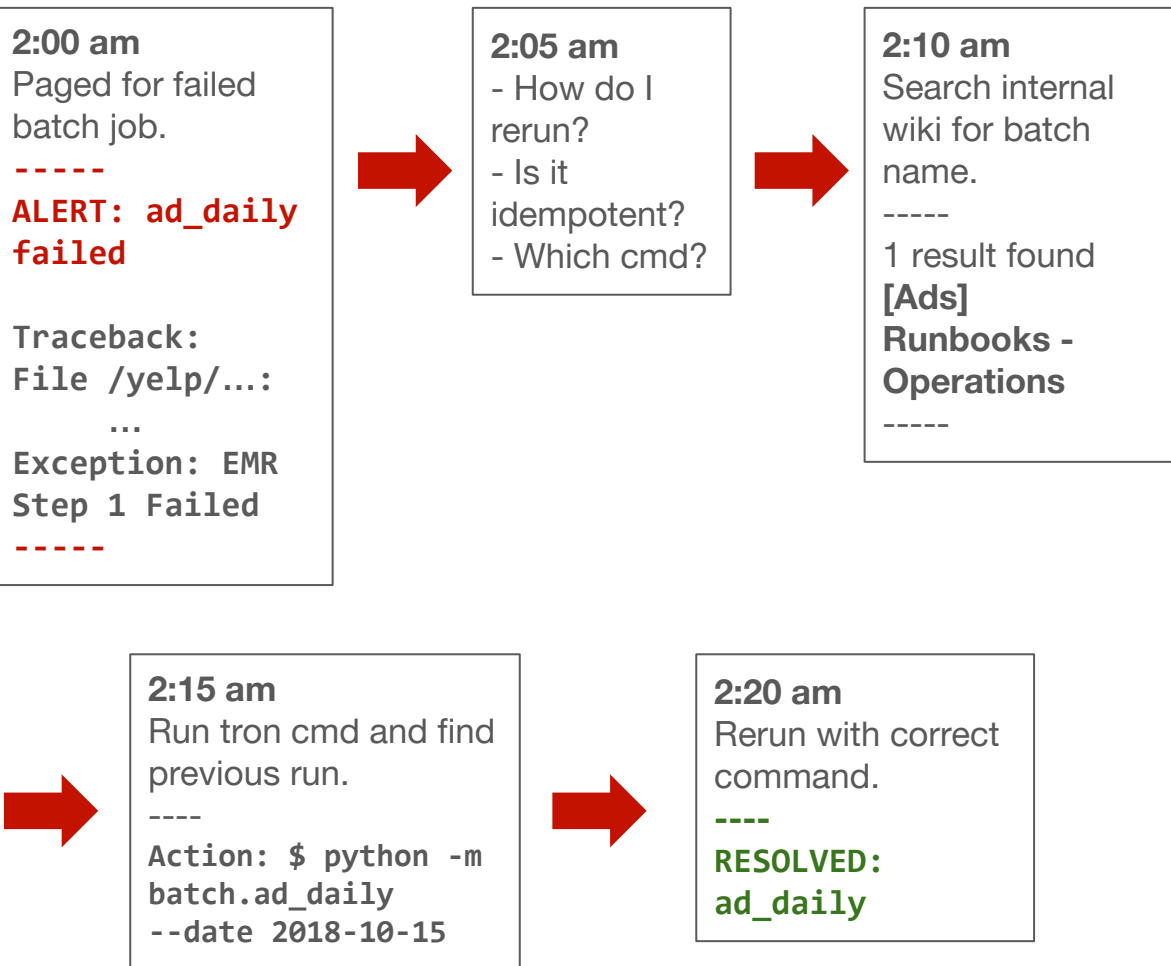
```
$ tronview ad_daily.XX.the_action_name
```

Once they run successfully manually, resume the rest of the job by skipping the action. **tronctl skip ad_daily.XX.the_action_name** - you can attach this to most batch recoveries, ie **sudo -u batch python -m batch.ads.failed_batch --args --more-args && tronctl skip ad_daily.XX.failed_batch_action**

Story time: Being on-call with a runbook

TTR: 20 min

2:00 am
Paged for failed batch job.
`-----`
ALERT: ad_daily failed

```
Traceback:
File /yelp/…:
     …
Exception: EMR
Step 1 Failed
-----
```

2:05 am
- How do I rerun?
- Is it idempotent?
- Which cmd?

2:10 am
Search internal wiki for batch name.
`-----`
1 result found
[Ads] Runbooks - Operations
`-----`

2:15 am
Run tron cmd and find previous run.
`----`
```
Action: $ python -m
batch.ad_daily
--date 2018-10-15
```

2:20 am
Rerun with correct command.
`----`
RESOLVED: ad_daily

What was good about this runbook?

✳ **Runbook exists**

✳ **Able to find runbook**

✳ **Contains actual commands to run**

✳ **Alert contains traceback**

What made this runbook difficult to use?

❊ **Table of contents too long**

❊ **Info about 4-5 different systems on the same page**

❊ **Doesn't map alert to action item**

❊ **Lacking some step-by-step instructions**

What could be better about this runbook?

Don't keep runbooks in a TODO state

Alerts

When responding to a Nagios alert, make sure you ACKNOWLEDGE the action. ACKNOWLEDGE that an issue is known by looking for the guy with the shovel under "Service Commands" on the right. Alerts that go through pagerduty can be acknowledged directly within pagerduty.

NOTE: This section would benefit a lot from having our actual alerts listed and detailed here.

3. Fix/retry broken actions

What could be better about this runbook?

Add guidelines for diagnosis

If a batch died due to an EMR, DB, or other intermittent issue, attempt to run the action manually

If a batch died due to a logic error, push a fix and run the action manually

To run manually, read the command line printed in this output. It's between the "!Node:" and "Requirements:" lines. You'll have to execute this as batch yourself.

```
$ tronview ad_daily.XX.the_action_name
```

Once they run successfully manually, resume the rest of the job by skipping the action. `tronctl skip ad_daily.XX.the_action_name` - you can attach this to most batch recoveries, ie `sudo -u batch python -m batch.ads.failed_batch --args --more-args && tronctl skip ad_daily.XX.failed_batch_action`

3. Fix/retry broken actions

What could be better about this runbook?

Link to docs with step-by-step procedures

If a batch died due to an EMR, DB, or other intermittent issue, attempt to run the action manually

If a batch died due to a logic error, push a fix and run the action manually

To run manually, read the command line printed in this output. It's between the "!Node:" and "Requirements:" lines. You'll have to execute this as batch yourself.

```
$ tronview ad_daily.XX.the_action_name
```

Once they run successfully manually, resume the rest of the job by skipping the action. `tronctl skip ad_daily.XX.the_action_name` - you can attach this to most batch recoveries, ie `sudo -u batch python -m batch.ads.failed_batch --args --more-args && tronctl skip ad_daily.XX.failed_batch_action`

3. Fix/retry broken actions

What could be better about this runbook?

Include example output to guide the user

If a batch died due to an EMR, DB, or other intermittent issue, attempt to run the action manually

If a batch died due to a logic error, push a fix and run the action manually

To run manually, ==read the command line printed in this output. It's between the "!Node:" and "Requirements:" lines.== You'll have to execute this as batch yourself.

```
$ tronview ad_daily.XX.the_action_name
```

Once they run successfully manually, resume the rest of the job by skipping the action. `tronctl skip ad_daily.XX.the_action_name` - you can attach this to most batch recoveries, ie `sudo -u batch python -m batch.ads.failed_batch --args --more-args && tronctl skip ad_daily.XX.failed_batch_action`

3. Fix/retry broken actions

What could be better about this runbook?

If a batch died due to an EMR, DB, or other intermittent issue, attempt to run the action manually

If a batch died due to a logic error, push a fix and run the action manually

To run manually, read the command line printed in this output. It's between the "!Node:" and "Requirements:" lines. ==You'll have to execute this as batch yourself.==

```
$ tronview ad_daily.XX.the_action_name
```

Once they run successfully manually, resume the rest of the job by skipping the action. `tronctl skip ad_daily.XX.the_action_name` - you can attach this to most batch recoveries, ie ==`sudo -u batch python -m batch.ads.failed_batch --args --more-args && tronctl skip ad_daily.XX.failed_batch_action`==

Avoid ambiguous instructions

Tips for writing
good technical runbooks

* Map alert to clear action items

* Inverted pyramid

* Separate critical vs. non-critical

* Examples of actual commands

* Consistent format across runbooks

Story time:
Being on-call with
a better runbook

TTR: 5 min

2:00 am
Paged for failed batch job.

ALERT: ad_daily failed

Runbook: https://y/ad_daily
Dashboard: https://signalfx.yelpcorp.com/...
Tip: Try `ssh production` and `sudo -u batch -i <Actual command>`
Details:
    Last action: ad_daily.29.run_report
    Actual command: python -m batch.ad_daily ...
    Traceback:
    File /.../yelp/batch/ad_daily/report.py:
        ...
    Exception: EMR Step 1 of 1 Failed

2:05 am
Rerun with correct command.

RESOLVED: ad_daily

Story time:
Being on-call with
a better runbook

TTR: 5 min

2:00 am
Paged for failed batch job.

ALERT: ad_daily failed

Runbook: https://y/ad_daily
Dashboard: https://signalfx.yelpcorp.com/...
Tip: Try `ssh production` and `sudo -u batch -i
<Actual command>`
Details:
    Last action: ad_daily.29.run_report
    Actual command: python -m batch.ad_daily ...
    Traceback:
    File /.../yelp/batch/ad_daily/report.py:
        ...
    Exception: EMR Step 1 of 1 Failed

2:05 am
Rerun with correct command.

RESOLVED: ad_daily

**Story time:
Being on-call with
a better runbook**

Runbook - ad_daily

This is the recovery and backfill runbook for on-calls. See Ads Billing Pipeline on why this batch is important and Development - ad_daily for developer workflow and testing.

- Easy Recovery
- P0 Recovery
  - Triage
  - Backfills
  - Multi-day recovery
- Revenue Impact
- Documentation

**Story time:
Being on-call with
a better runbook**

Runbook - ad_daily

Triage

Check logs to get useful traceback information for debugging
```
$ tail batch_ad_daily_log --region prod
```

and look for the line
```
INFO Logs are in s3://yelp-emr-prod/logs/j-xxx/
```

Look through stderr files which can contain errors.
```
$ find . | grep stderr | xargs zcat | less
```

Runbook - ad_daily

**Story time:
Being on-call with
a better runbook**

Triage

Check logs to get useful traceback information for debugging

```
$ tail batch_ad_daily_log --region prod
```

and look for the line
INFO Logs are in s3://yelp-emr-prod/logs/j-xxx/

Look through stderr files which can contain errors.

```
$ find . | grep stderr | xargs zcat | less
```

Includes example output to guide the user

A good technical runbook is **easy to find**

Make alerts rich
Put actual commands and/or runbook link in the alert

Make runbooks searchable

Portals for easy access to important docs

Example:
Non-technical runbook

**Non-technical
runbook**

Incident Response Checklist

This document is for Ads incident first responders.
First assess, escalate until the appropriate team is
established, and take on the appropriate role.

Assess

Escalate

Communicate

Investigate and Fix

Clean Up

**Non-technical
runbook**

Incident Response Checklist

Assess

For example: errors served, % clients impacted, or financial loss to the business.

If it takes more than a few minutes to assess, assume it is very bad and move on to escalation.
- ❏ What is the business-facing impact?
- ❏ What is the consumer-facing impact?

Dashboards to consult:
- ❏ SignalFx - error percentages, latencies
- ❏ Splunk - log lines

Incident Response Checklist

Escalate

Outages run longer and with worse outcomes when tackled alone. It's better to escalate a false alarm than fail to escalate a serious issue.

Page the following as appropriate:

- ❏ Secondary on-call
- ❏ Manager
- ❏ Database Reliability Team (#dba)
- ❏ AWS Support Liaison

**Non-technical
runbook**

Incident Response Checklist

Communicate

Non-technical runbook

- ❏ [Create a ticket](#) in the ADS project with a brief description of the issue.

  - ❏ Add secondary and manager as watchers

  - ❏ Comment on ticket with major updates

- ❏ Consolidate triage communications to #ads-incident.

- ❏ Send email to ads-incident@ to liaise with financial stakeholders and downstream consumers of data: [email templates](#).

Non-technical **runbook**

Incident Response Checklist

Investigate and Fix

- ❏ [Ads Runbooks List](#)

Clean Up

- ❏ Send all-clear email to ads-incident@

- ❏ File follow-up ticket for postmortem and set yourself as the assignee

Incident Response Checklist

This document is for Ads incident first responders. First assess, escalate until the appropriate team is established, and take on the appropriate role.

Assess

For example: errors served, % clients impacted, or financial loss to the business.
If it takes more than a few minutes to assess, assume it is very bad and move on to escalation.
- What is the business-facing impact?
- What is the consumer-facing impact?

Dashboards to consult:
- SignalFx - error percentages, latencies
- Splunk - log lines

Escalate

Outages run longer and with worse outcomes when tackled alone. It's better to escalate a false alarm than fail to escalate a serious issue.
Page the following as appropriate:
- Secondary on-call
- Manager
- Database Reliability Team (#dba)
- AWS Support Liaison

Communicate
- ❏ Create a ticket in the ADS project with a brief description of the issue.
    - ❏ Add secondary and manager as watchers
    - ❏ Comment on ticket with major updates
- ❏ Consolidate triage communications to #ads-incident.
- ❏ Send email to ads-incident@ to liaise with financial stakeholders and downstream consumers of data.

Investigate and Fix
- ❏ Ads Runbooks List

Clean Up
- ❏ Send all-clear email to ads-incident@
- ❏ File follow-up ticket for postmortem and set yourself as the assignee

Make your own runbook

Step 1:
Start with a template

Alert Name	<exact alert name>
Description	<1 sentence description>
Stakeholder impact	<1 sentence impact>
Mitigation steps	1. Try restarting: <command> 2. Monitor dashboards. 3. Inspect logs to diagnose issue: <link or See steps below> If things do not recover, follow Escalation steps.
Escalation steps	Contact <team>. Massive ingestion delays should be communicated to <upstream and downstream teams>.
Related services	<upstream and downstream dependencies>
Dashboards	<links>
Related links	<other docs or related runbooks>

Step 2:
Customize for your use case

Step 3:
"Commit" it with your code

Step 4:
Update alerts

Keep runbooks **useful**

❋ **Be familiar with it before you need it**

   🔍 Know where to find

   🛠 Use in shadowing/training/wargames

❋ **Update runbooks**

   〰 When creating a new alert

   ▤ Postmortem action item

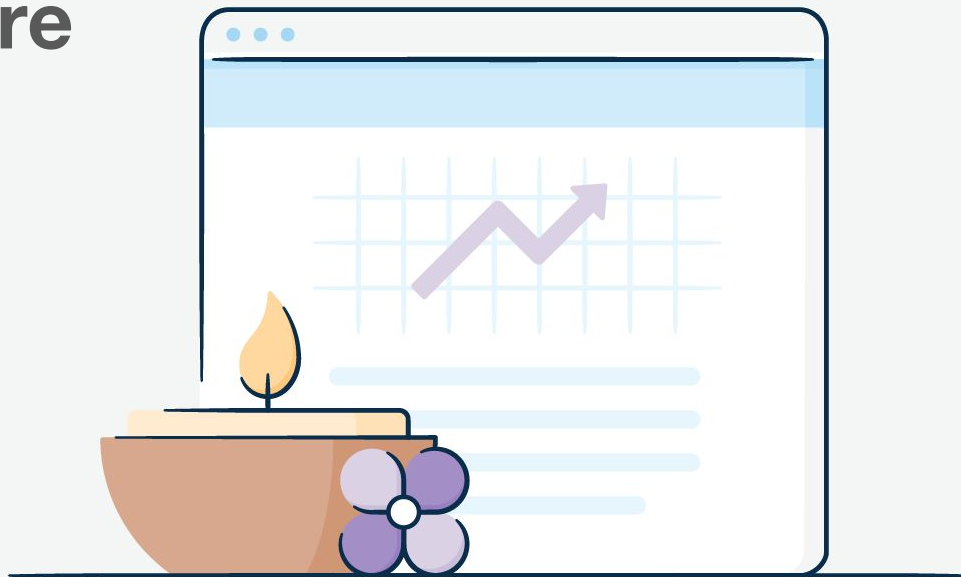   💡 While training; new members can help spot issues
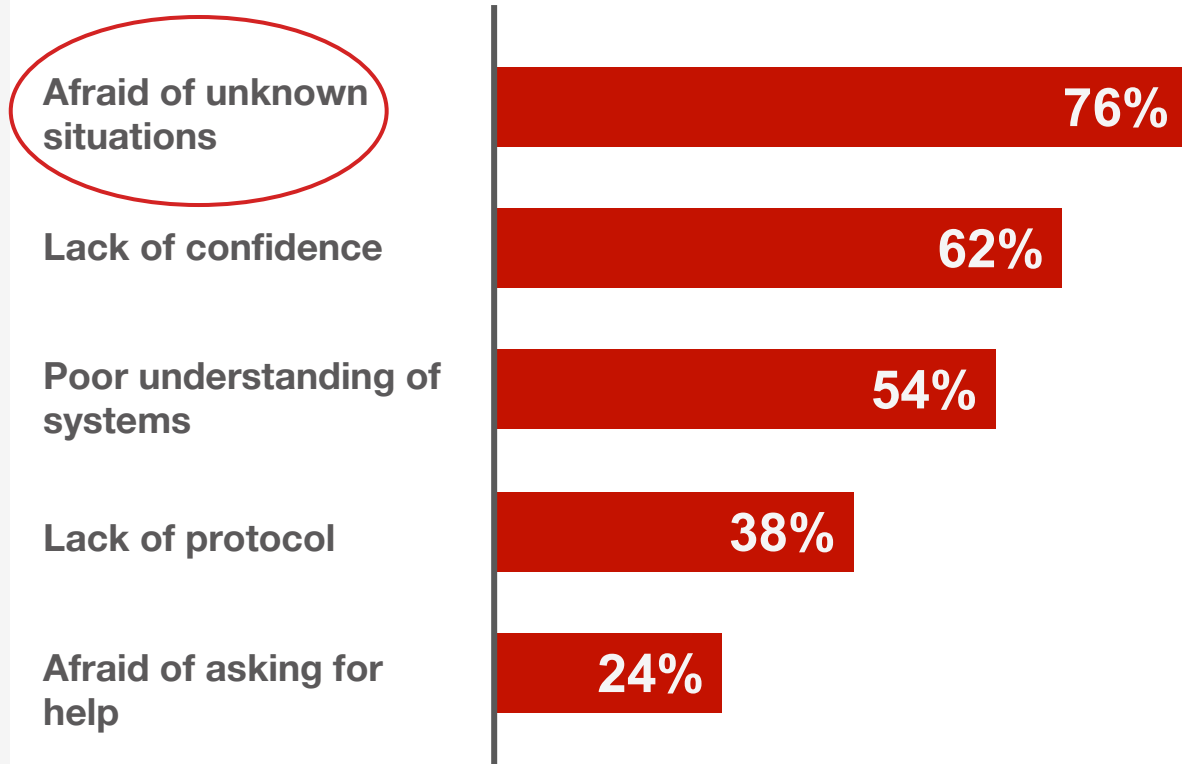
Beyond runbooks

✻ **Good for common cases**

✻ **What about unexpected situations?**

   ✻ Provide tools to help in decision-making

   ✻ Build context around your systems

   ✻ Pattern match with past incidents

✻ **Automate as much as possible**

"I wish I was more experienced"

Why didn't you feel ready?

Afraid of unknown situations — 76%

Lack of confidence — 62%

Poor understanding of systems — 54%

Lack of protocol — 38%

Afraid of asking for help — 24%

Survey within Yelp Engineering (2018)

Postmortems as a **Learning Tool**

Learning from
Postmortems

>80% Read postmortems from their teams

>86% Learned something new

"Nice deep dives"

"Always lessons learned
from reading"

"Make you aware of pain
points broadly"

"Give you a better
view into systems"

Why are **postmortems important** for new engineers?

Knowledge Sharing

Collaboration

Be Blameless

Different Ways to Learn

✳ **Read postmortems**

✳ **Postmortem debriefing meeting**

✳ **Wargames from postmortems**

✳ **Review postmortems**

Read
Postmortems

❉ **Make postmortems accessible**

  ❉ Mailing list

  ❉ Tag with keywords

  ❉ Newsletter

❉ **Encourage reading**

  ❉ Reading club

  ❉ Start with your own team's postmortems

  ❉ Reference in your daily work, e.g. code reviews

Postmortem **Debriefing** Meeting

❊ **Real-time collaboration**

❊ **Goal: Understand the incident from multiple perspectives**

❊ **Facilitator, people closest to the event, and general audience**

Wargames from Postmortems

❉ **Multi-person incident simulation game**

❉ **Create based on past incidents**

    ❉ Uses actual logs/dashboards from the incident

❉ **Make it realistic and safe**

    ❉ Non-prod or other geo environments

    ❉ Simulate communication protocol

Wargame
Roles

❈ **Game master**

  ❈ Reproduce the scene with a postmortem

  ❈ Drive conversations

  ❈ Ask questions to make sure they understand each step

❈ **Primary/secondary "on-call"**

  ❈ Investigate and mitigate

  ❈ Apply knowledge and practice using tools

Review
Postmortems

📋 **Assess postmortem and provide feedback**

💡 **Encourage them to ask questions**

✳ There are no dumb questions

✳ Be blameless

Review
Postmortems

💡 **How people know something is wrong**

"The metric seemed off."	What did you see or hear? When did you notice?

💡 **How things normally work**

"After the on-call restarted it, it did not succeed."	How did you restart it? What usually happens before or after restarting? Have you done/seen this before?

Review
Postmortems

💡 Rationales for choices or decisions

"On-call person decided to…"	What helped you make the decision? Had you done/seen this before?

💡 Alternatives considered

"On-call person decided to…"	What other options did you consider at the time?

💡 Weaknesses in process or protocol

"It was frustrating that…"	What made it difficult to handle this incident? Is this process sustainable?

Postmortems

� **Learning tool to boost experience and build on-call mindset**

� **Blameless culture and knowledge sharing**

NOT READY   **FEELING BETTER**   **READY**

TEAM AND COMPANY CULTURE

BLAMELESS / KNOWLEDGE SHARING / COLLABORATION

Training materials

http://bit.ly/lisa18-oncall

Additional Resources

Training new on-calls

- [Accelerating SREs to On-Call and Beyond](#)
- [From Zero to Hero: Recommended Practices for Training your Ever-Evolving SRE Teams](#)

Runbooks

- [7 Deadly Sins of Documentation](#)
- [Do Docs Better: Practical Tips](#)
- [Yelp Runbook Template](#)

Postmortems

- [Postmortem culture: learning from failure](#)
- [Debriefing facilitation guide](#)

Questions?

Thank you.