



# What I Wish I Knew Before Going On-call

SRECon 2019

Survey

<http://bit.ly/survey-srecon19>



## WHO WE ARE



**Chie Shu**  
Software Engineer  
chie@yelp.com



**Wenting Wang**  
Software Engineer  
wwang@yelp.com



## WHO WE ARE

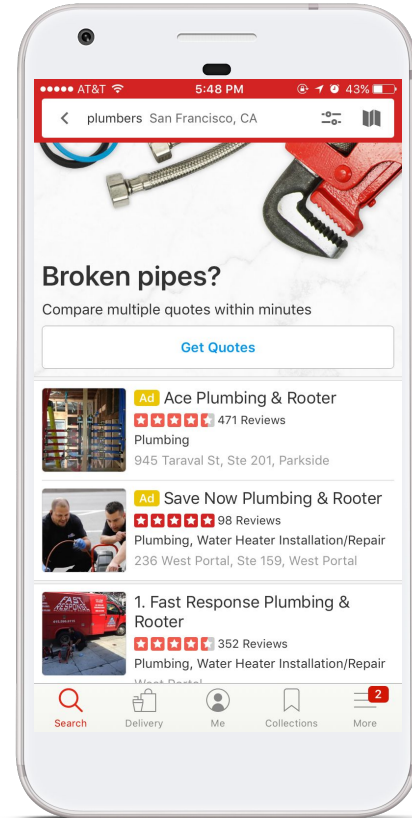
# Yelp Local Ads



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**



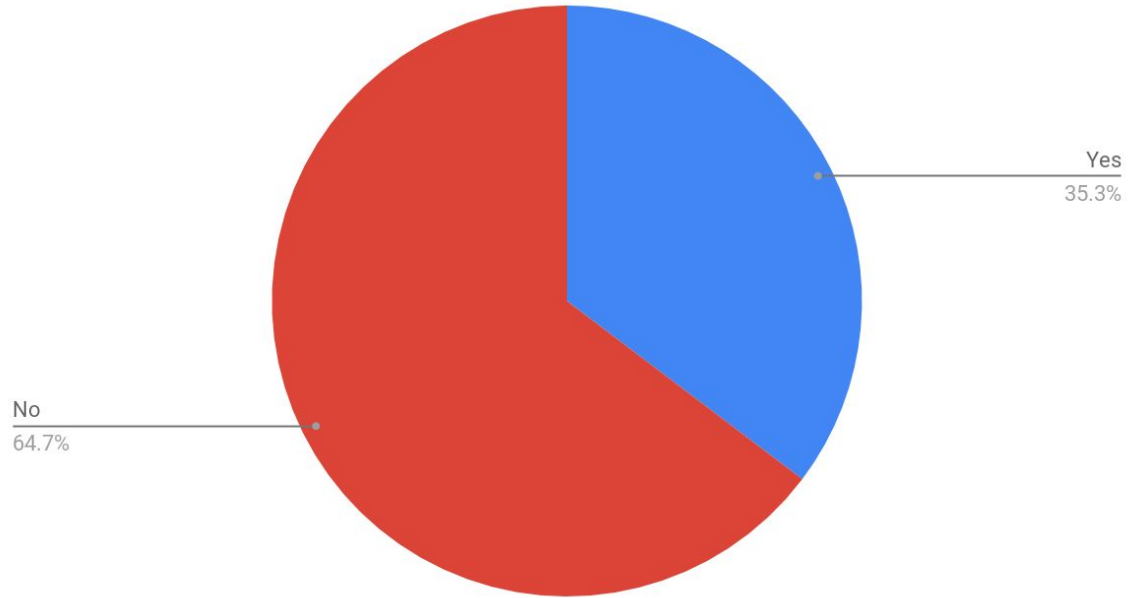
## Newbie on-call struggles



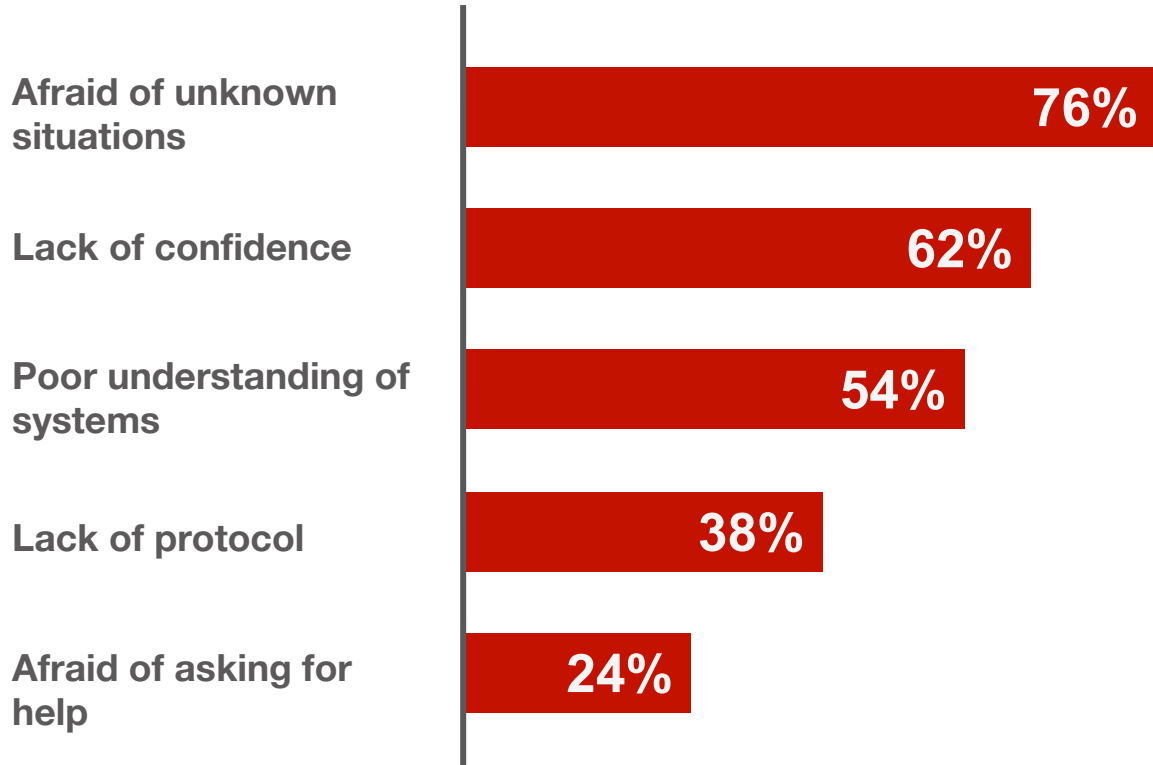
- **No established training process**
- **Decentralized + Outdated documentations**
- **So much financial impact/pressure**

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

Survey within Yelp Engineering (2018)



## Why didn't you feel ready?





## 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



# Workshop Goal

**Build an efficient oncall onboarding system  
for your organization**



# Agenda

1. **Common Myths about On-Call**
2. **How to Create Training Program**
3. **Runbook for Effective Incident Response**



# **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”**

**Root cause finding is a non-goal**



## **Myth #4**

**“I need to make the best/long-term fix”**

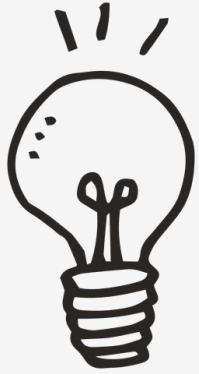
**You are supposed to mitigate the issue**





## Setting the right **expectations**

1. **Reduce (unnecessary) fear**
2. **More productive + efficient on-call**



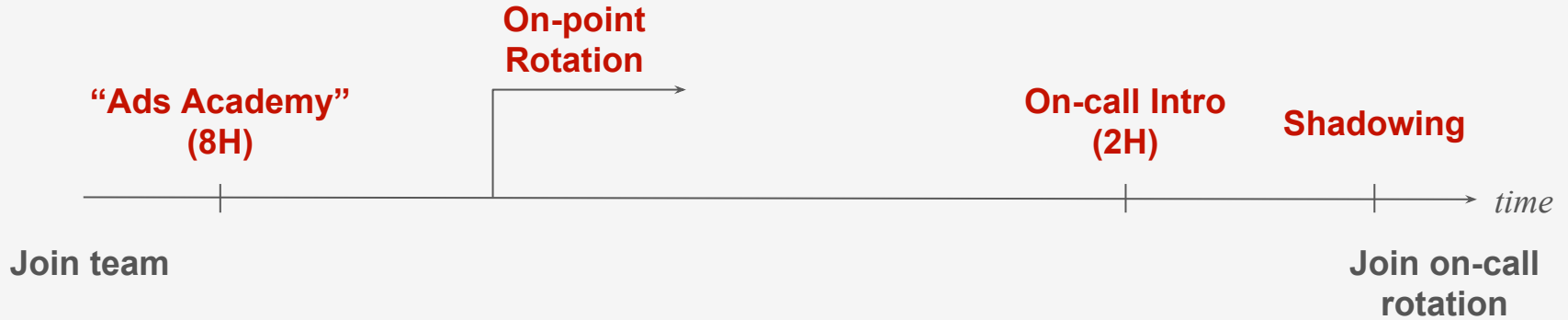
Set the right **expectations**  
during training!



**Now onto the training program...**



# My On-call “Training”



## 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



# The Goal of Training Program

## Goal 1.

Be able to draw a mental **picture of your system**

## Goal 2.

Understand **failure modes/alerts** for the system

## Goal 3.

Know the **tools** for investigation



## **Exercise**

**Let's make an oncology training program!**





# Exercise Agenda

Let's make an oncall training program!

1. Make a Curriculum
2. Create Introduction
3. Cover Failure Modes
4. List Necessary Tools

What you need:

Text editor of your choice



# Exercise Agenda

Let's make an oncall training program!

1. Make a Curriculum
2. Create Introduction
3. Cover Failure Modes
4. List Necessary Tools



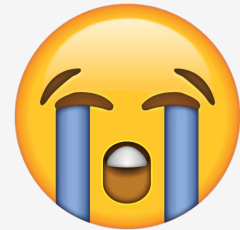
## **Exercise #1**

**Let's make a curriculum!**



# Anti-example

Lesson	Topic
1	Everything you need to know about ads on-call (2 hours)





**Tip:** Avoid information overload



Lesson	Topic
1	Oncall Expectation + Overview of Ad systems
2	Billing (Critical)
3	Ad Delivery (Critical)
4	Ad Internal Reports/Metrics (Less Critical)
5	Targeting (Less Critical)

**Ask yourself a question:  
Is there information-overload happening?**

Lesson	Topic
1	Oncall Expectation + Overview of Ad systems ← <b>Should be super high level</b>
2	Billing (Critical)
3	Ad Delivery (Critical)
4	Ad Internal Reports/Metrics (Less Critical)
5	Targeting (Less Critical)

**Ask yourself a question:  
Is there information-overload happening?**

Lesson	Topic
1	Oncall Expectation + Overview of Ad systems
2	Billing (Critical) ← <b>What if this is a complicated data pipeline with many alerts?</b>
3	Ad Delivery (Critical)
4	Ad Internal Reports/Metrics (Less Critical)
5	Targeting (Less Critical)

**Ask yourself a question:  
Is there information-overload happening?**



Lesson	Topic
1	Oncall Expectation + Overview of Ad systems
2	<b>Ad Analytics Pipeline (Critical)</b> <b>Split it into a reasonable unit!</b>
3	<b>Billing Pipeline(Critical)</b>
4	Ad Delivery (Critical)
5	Ad Internal Reports/Metrics (Less Critical)
6	Targeting (Less Critical)

Lesson	Topic
1	Oncall Expectation + Overview of Ad systems
2	Ad Usage Pipeline (Critical)
3	Billing Pipeline(Critical)
4	Ad Delivery (Critical)
5	Ad Internal Reports/Metrics (Less Critical)
6	Targeting (Less Critical)

**Ask yourself a question:  
Does the order of the topics make sense?**

Lesson	Topic
1	Oncall Expectation + Overview of Ad systems
2	Ad Usage Pipeline (Critical)
3	Billing Pipeline(Critical)
4	<b>Ad Delivery (Critical)</b>
5	Ad Internal Reports/Metrics (Less Critical)
6	Targeting (Less Critical)

**Ask yourself a question:  
Does the order of the topics make sense?**

Lesson	Topic
1	Oncall Expectation + Overview of Ad systems
2	Ad Analytics Pipeline
3	Billing Pipeline(Critical)
4	<b>Ad Delivery (Critical)</b> ← This is an upstream of #2 and #3
5	Ad Internal Reports/Metrics (Less Critical)
6	Targeting (Less Critical)

**Ask yourself a question:  
Does the order of the topics make sense?**

Lesson	Topic
1	Oncall Expectation + Overview of Ad systems
<b>2</b>	<b>Ad Delivery (Critical)</b>
3	Ad Analytics Pipeline (Critical)
4	Billing Pipeline (Critical)
5	Ad Internal Reports/Metrics (Less Critical)
6	Targeting (Less Critical)

**Ask yourself a question:  
Does the order of the topics make sense?**

## Exercise #1

Let's make an oncology training curriculum!

- Come up with a list of topics
- Chunk it into a “reasonable” size
- Sort them



**3 mins**



# Exercise Agenda

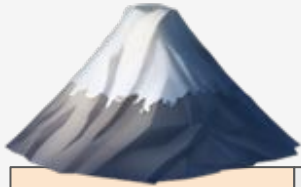
Let's make an oncall training program!

1. Make a Curriculum
2. Create Introduction
3. Cover Failure Modes
4. List Necessary Tools



## Exercise #2

Let's write a 10000 ft overview of the system!



**10000 ft  
overview**

**Actual oncall training**





## Exercise #2

# Why give an overview in on-call training?

- Make sure students are on the same page
- Make failure points clearer



**10000 ft  
overview**

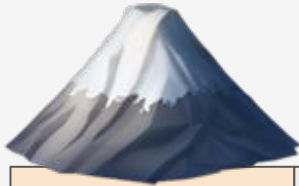
**Actual oncall training**



## Exercise #2

# What should a 10000 ft overview include?

- Simple Diagram
- Summary of the system (What it does, what depends on it etc)



**10000 ft  
overview**

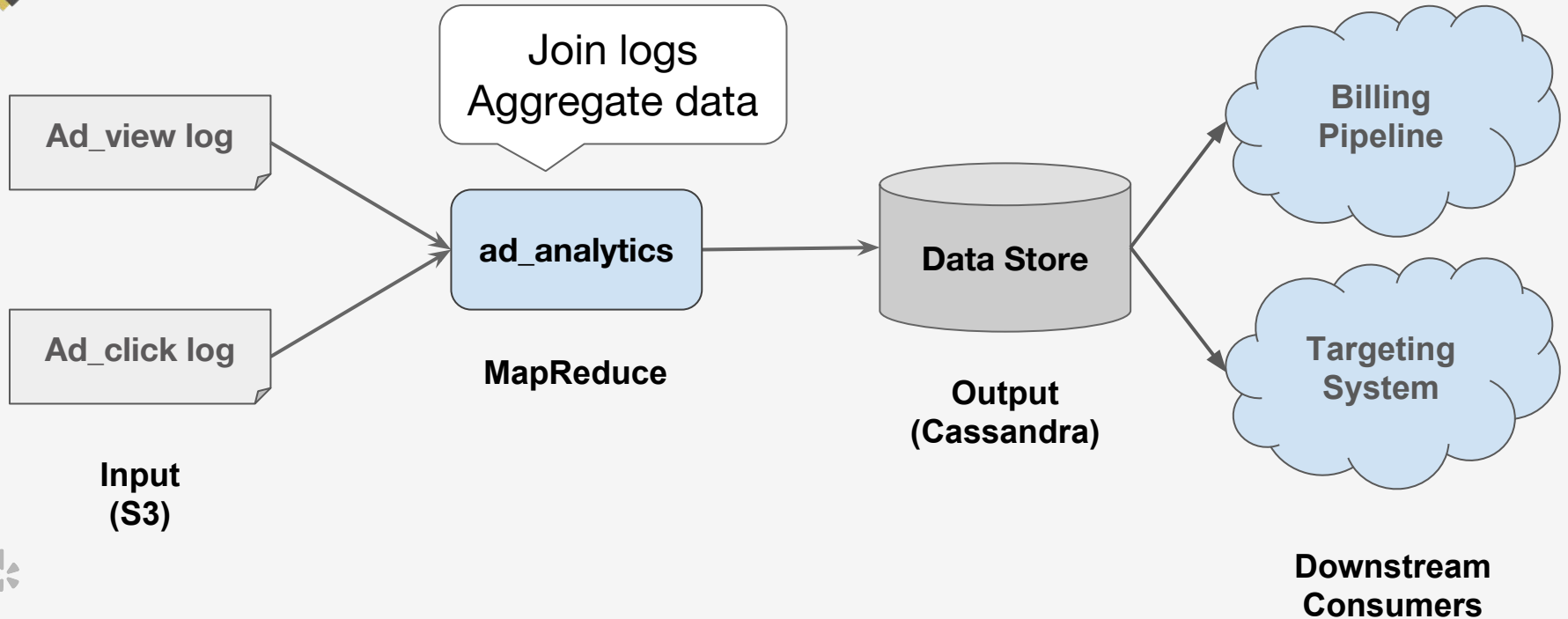
**Actual oncall training**



Lesson	Topic
1	What is oncall? + Overview of Ad systems
2	Ad Delivery (Critical)
<b>3</b>	<b>Ad Analytics Pipeline (Critical)</b>
4	Billing Pipeline (Critical)
5	Ad Internal Reports/Metrics (Less Critical)
6	Targeting (Less Critical)



# Example: Ad Analytics Pipeline



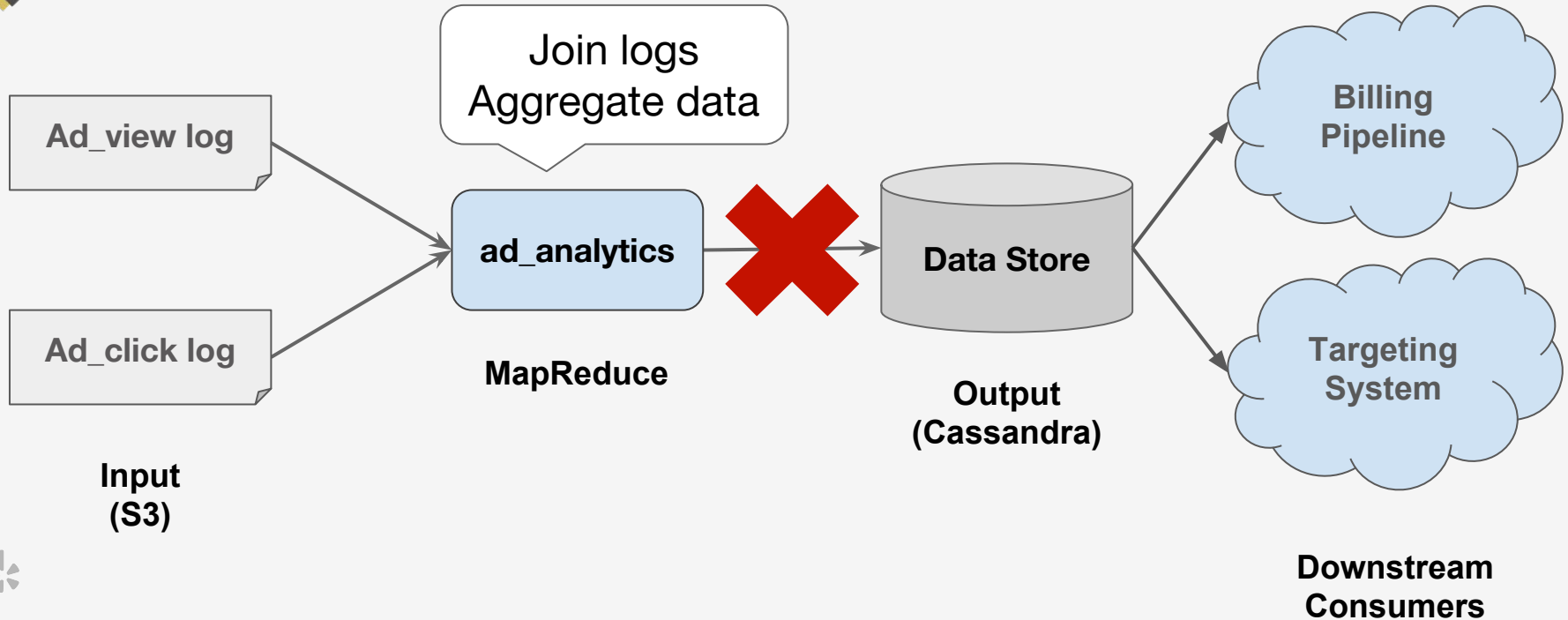


**Tip:** Use visual aid you can reuse



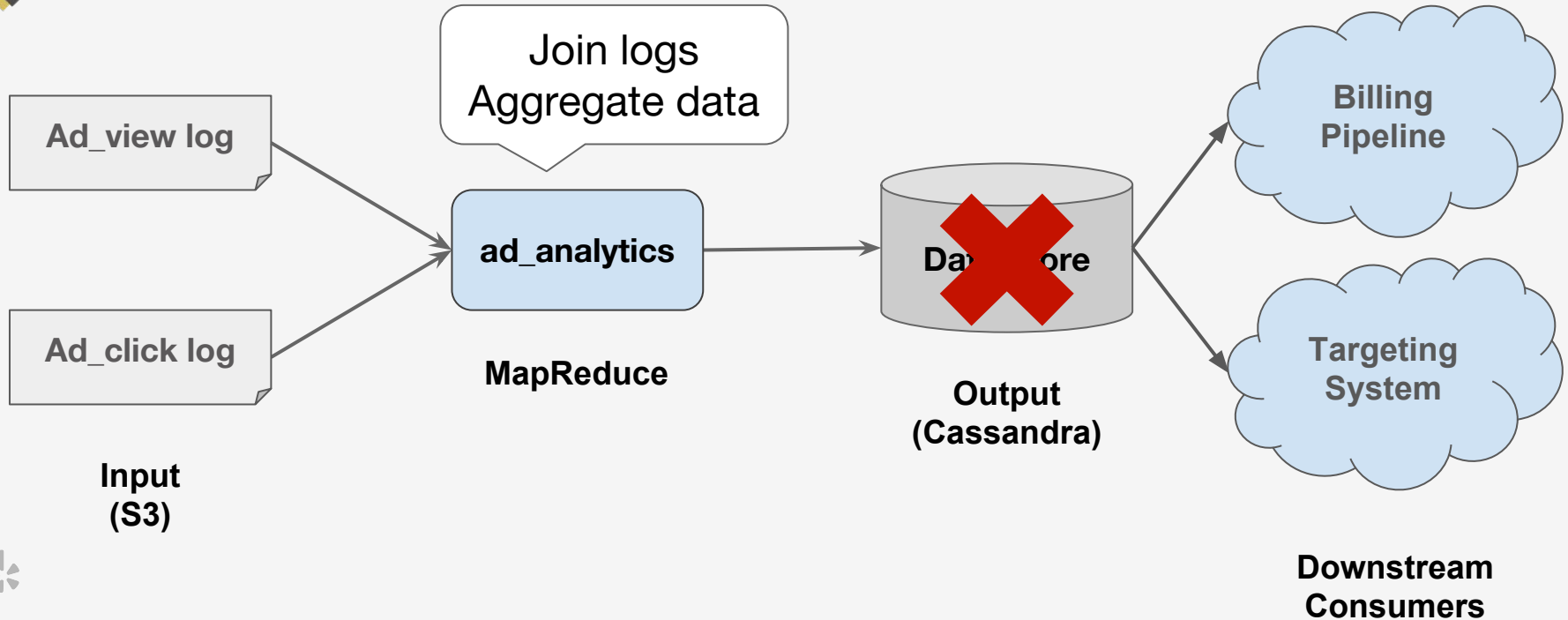


# Example: Ad Analytics Pipeline





# Example: Ad Analytics Pipeline



## Exercise #2

Let's write a 10000 ft overview of the system!

1. Pick one topic from the curriculum
2. Summarize the system, techstack, and failure points
3. Add a diagram



**3 mins**





# Exercise Agenda

Let's make an oncall training program!

1. Make a Curriculum
2. Create Introduction
3. Cover Failure Modes
4. List Necessary Tools



## Exercise #3

Let's write the “actual on-call training”



10000 ft  
overview

**Actual oncall training**



## Exercise #3

Let's write the “actual on-call training”



10000 ft  
overview

**Actual oncall training**



Usually talks about failure modes/alerts  
and how to respond to them





**Tip**

**Use Past Incidents**



## **Exercise #3**

### **Why use past incidents?**

- **Examples are the best teachers!**
- **Opportunity to make it interactive**



# Example: Ad Analytics Pipeline

**Alert:**

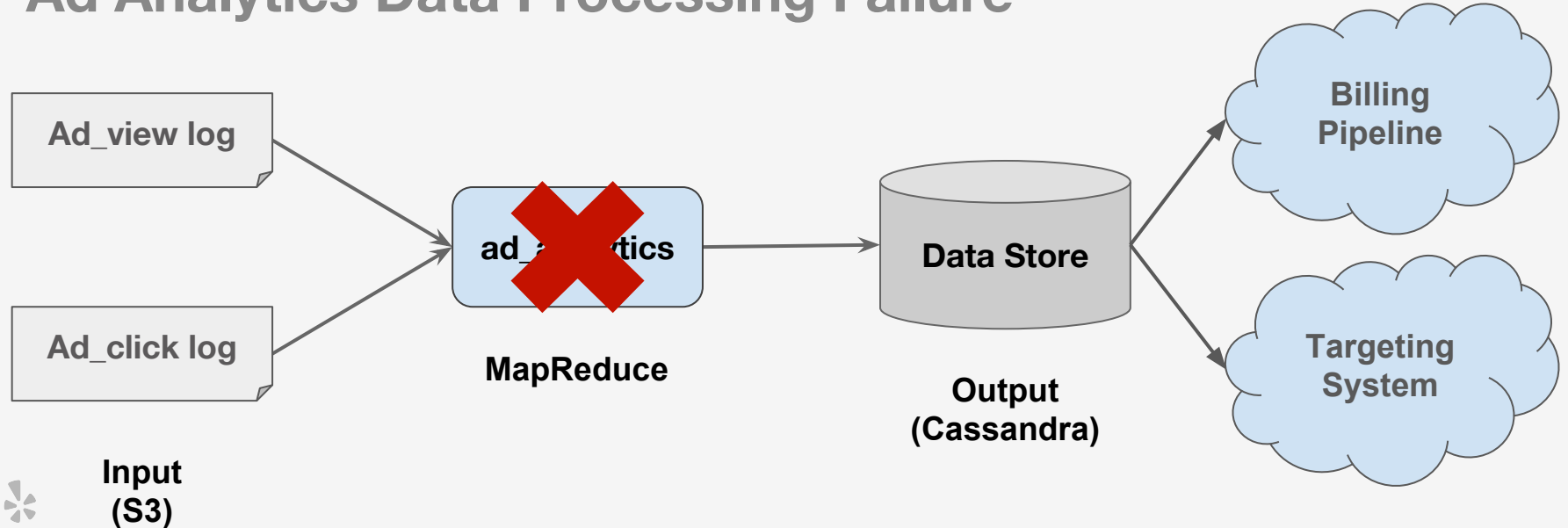
**Ad Analytics Data Processing Failure**



# Example: Ad Analytics Pipeline

**Alert:**

**Ad Analytics Data Processing Failure**



# Example: Ad Analytics Pipeline

## Alert:

Ad Analytics Data Processing Failure

## Past Incidents:

- Backward-incompatible input schema change
- MapReduce task timeouts





## Exercise #3

Let's write the “actual on-call training”

- List alerts/failure modes
- Map it in your 10000 ft diagram
- Find at least one past incident for each alert



**3 mins**



# Exercise Agenda

Let's make an oncall training program!

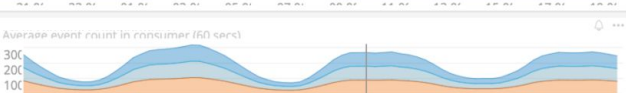
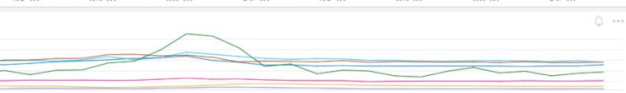
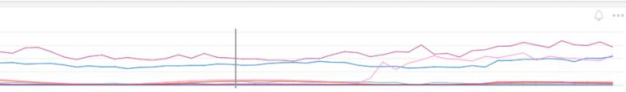
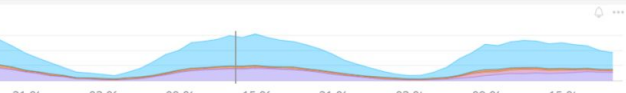
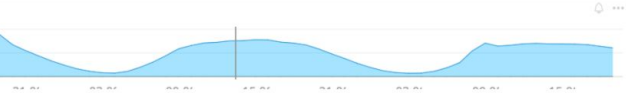
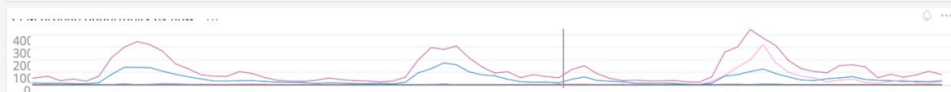
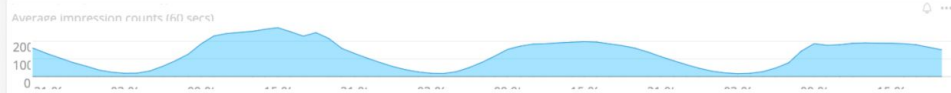
1. Make a Curriculum
2. Create Introduction
3. Cover Failure Modes
4. List Necessary Tools



## **Exercise #4**

**Let's teach necessary tools and know-hows**





# Example

How to read a service SignalFx dashboard



# Example

**How to read a service SignalFx dashboard**

(This should ideally be in runbook)





**Tip:** Let students apply knowledge ASAP



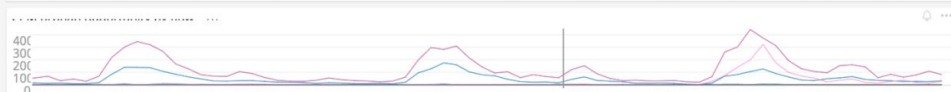
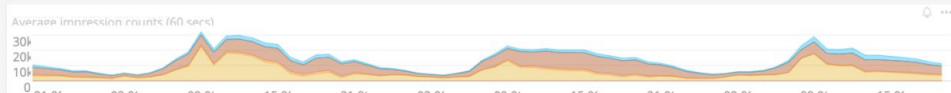
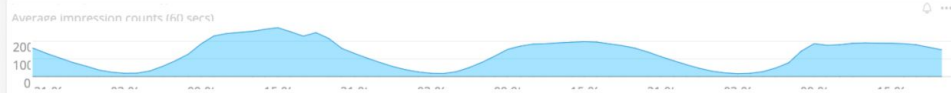
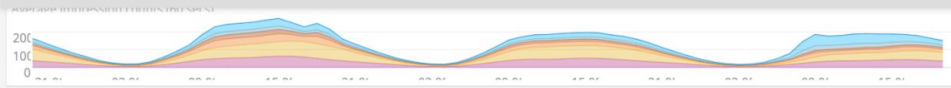
# Example

## How to read a service SignalFx dashboard

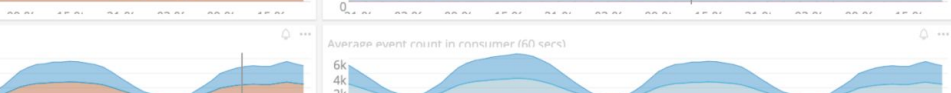
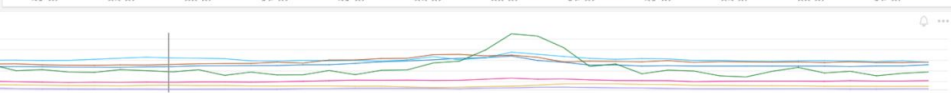
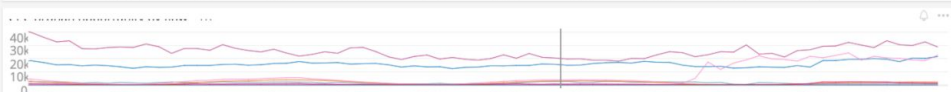
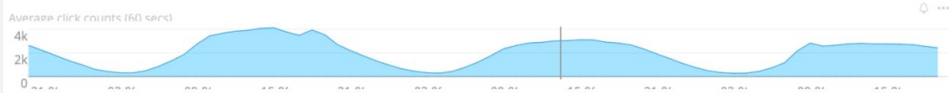
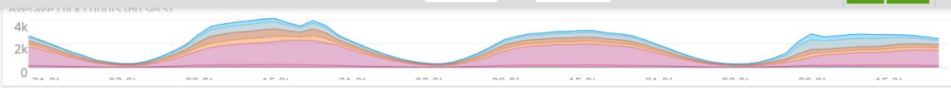
1. Explain
2. Show a dashboard screenshot from a past incident
3. Let students debug + ask questions







Traffic Input Rate (ad ark) 1h

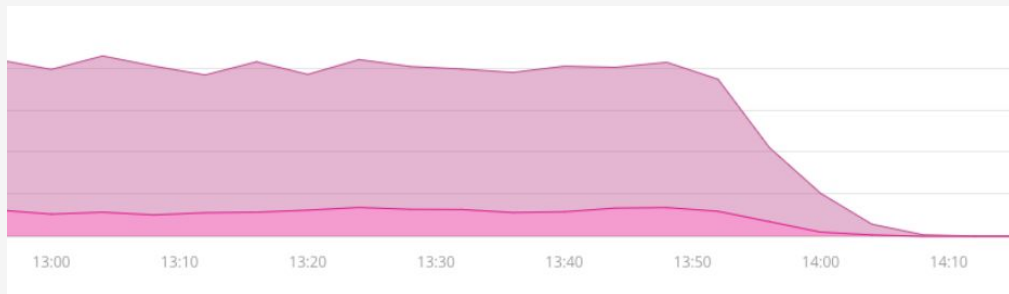


Event Consumers (ad ark) 1h

# Example

## How to read a service SignalFx dashboard

1. Explain
2. Show a dashboard screenshot from a past incident
3. Let students debug + ask questions



## Exercise #4

Let's teach necessary tools and know-hows

1. List tools and know-hows  
(Based on your answers from Exercise #3)
2. Make it interactive



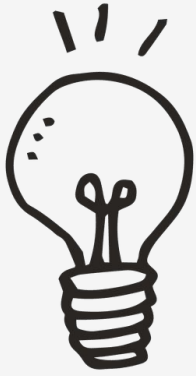
**3 mins**



**Congratulations!**

**You have a (partially complete)  
oncall training program!**





## **Tips (Recap)**

**Avoid information overload**

**Use visual aid you can reuse**

**Use Past Incidents**

**Let students apply knowledge ASAP**



# Beyond Training



## Knowledge sharing



### **Oncall handoff meeting**

Show and tell how recent incidents were resolved

### **Postmortem**

Learning from the past incidents

### **Wargame**

Gain experience in a fast and safe way



## Wargames



- ❖ **Multi-person incident simulation game**

- ❖ **Game master**

  - ❖ Reproduce the incident

  - ❖ Drive conversations

  - ❖ Ask questions and give hints

- ❖ **Oncall Player(s)**

  - ❖ Investigate and mitigate

  - ❖ Apply knowledge and practice using tools





# 3 Steps to start a **wargame**



**Step 1:**  
**Pick a scenario**



## Examples

### \* **Real past incidents**

- Seasonal traffic: Black Friday
- Critical System/Database crashed

### \* **Imaginary Incidents**

- Brainstorm or discuss what could happen and how to handle



**Step 2:**  
**Prepare** a game



## Example

# Wargame template

## Incident Setup

### Interactive

- conduct in safe environment

### Static

- dashboards/screenshots/logs/history of code



## Example

# Wargame template

## Incident Setup

Step-by-step instruction of how to trigger incident

- ❑ Prepare bad code [<link>](#)
- ❑ Prepare dashboard screenshot
- ❑ Set up an isolated env [<config file link>](#)
- ❑ Cmd to run batch in the env
  - ❑ `python ./batch.py --config config.yaml`
- ❑ Wait for batch to fail



## Example

# Wargame template

## Player roles

- Investigator --- <name>
- Communicator --<name>
- Commander -- <name>

## Player checklist

- Get relevant permissions
- Join external wifi/set up VPN
- Use wargame-only communication tools
  - channel #wargame
  - email alias wargame
  - JIRA project WARGAME



## Example

# Wargame template

## Hints

- Did you read runbook?
- Did you check batch log?
- Did you check recent code changes?
- Did you check dashboard: [<screenshot>](#)?

...





**Step 3:**  
**Run** the game



# **Tips** for running the game

## **Invite audience**

### **Ask questions**

- Ask what makes them take actions
- Make sure player(s) and audience understand the situation

### **Take notes**

- Runbook/Training/Monitoring/Alerting improvement
- Follow-up learning process



## Wargames

<http://bit.ly/oncall-game>



## Use tools to build your game

Oncall simulation text adventure game using **Twine**

**OnCall of Duty**

You are paged about a large spike in 500 errors on the search page.

**ALERT: HTTP 5xx -- yelp-main: threshold exceeded**

What do you do?

- Notify users >
- Wait for errors to drop >
- Check wiki page >
- Check recent code changes >
- Check dashboards >
- Escalate to a senior engineer >

Time: 0 hrs (+0) | Stress: 0% (+0)

# Break (5 mins)

Oncall twine game:

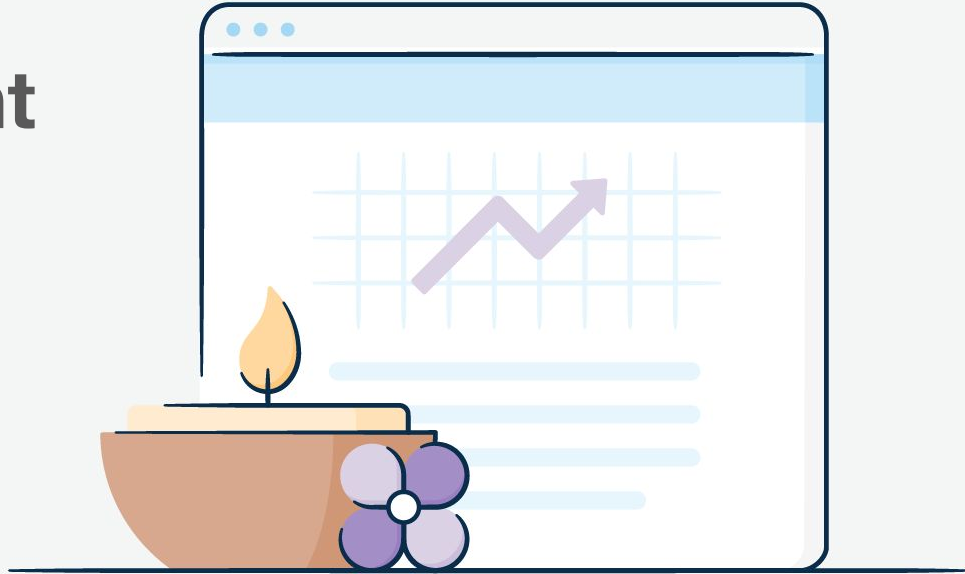
<http://bit.ly/oncall-game>

Optional Materials:

<http://bit.ly/srecon19-oncall>



# Runbooks for **Effective** Incident Response



## Why didn't you feel ready?

**40%** Didn't feel ready due to lack of protocol

**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.”



## Why care about good runbooks?

**Win 1:** Increase efficiency

**Win 2:** Reduce nervousness

**Win 3:** Stand-in for a mentor or back-up



# What is a runbook?

## \* **Technical runbook**

Step-by-step instruction on how to act in an incident

- Impact assessment
- Mitigation
- Disaster recovery

## \* **Non Technical Runbook**

Guidelines for human process

- Human roles
- Communication process
- Escalation policy





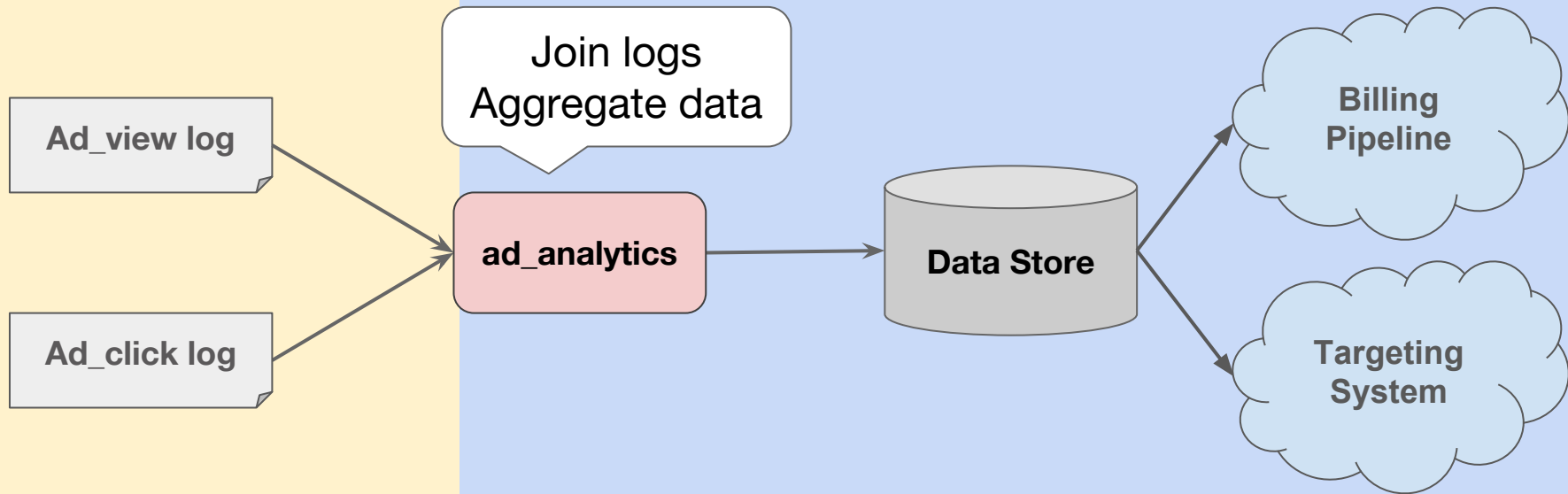
# Example

## Symptoms of **bad** runbook

<http://bit.ly/srecon19-oncall>



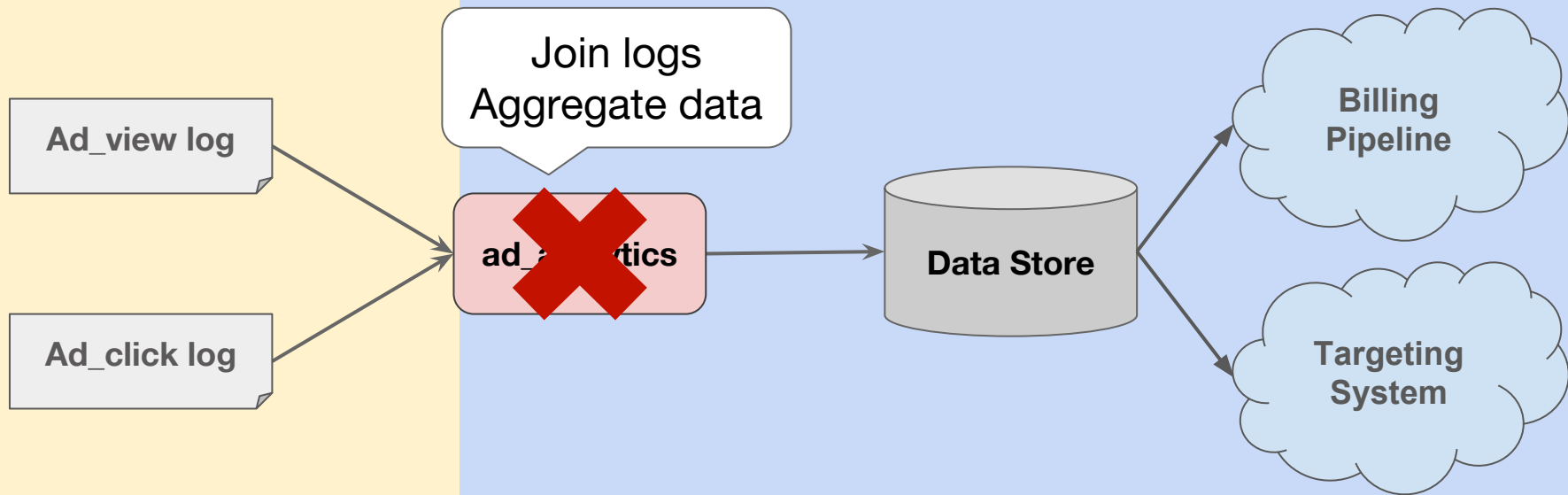
STORY TIME: BATCH  
RECOVERY



Daytime Web Traffic

Nightly Batch Job

STORY TIME: BATCH  
RECOVERY



Daytime Web Traffic

Nightly Batch Job

STORY TIME: BATCH  
RECOVERY

What made this  
runbook **difficult**  
to use?



2:00 am  
Paged for failed  
batch job.

-----  
**ALERT:**  
**ad\_analytics**  
**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**  
-----



## What made this runbook **difficult** to use?



## Runbooks - Operations

- [General recovering tips](#)
  - [Campaigns not in ad\\_store](#)
  - [Errors in ad template](#)
- [Nagios](#)
  - [Background](#)
  - [Updating Alerts](#)
  - [Alerts](#)
- [ad\\_analytics](#)
  - [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](#)
    - [calculated\\_ad\\_analytics](#)
    - [clculate\\_ad\\_spend](#)
    - [Business\\_ad\\_control](#)
- [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 the removing gearman workers for particular queues](#)
  - [Cleaning out a queue](#)



What made this runbook **difficult** to use?

# Alerts

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



What made this  
runbook **difficult**  
to use?

## Runbooks - Operations

- [General recovering tips](#)
  - [Campaigns not in ad\\_store](#)
  - [Errors in ad template](#)
- [Nagios](#)
  - [Background](#)
  - [Updating Alerts](#)
  - [Alerts](#)
- [ad\\_analytics](#)
  - [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](#)
    - [calculated\\_ad\\_analytics](#)
    - [calculate\\_ad\\_spend](#)
    - [Business\\_ad\\_control](#)
- [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 the removing gearman workers for particular queues](#)
  - [Cleaning out a queue](#)



What made this runbook **difficult** to use?

### 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.

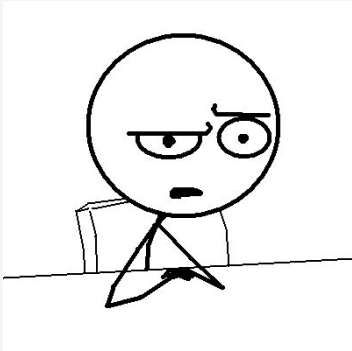
```
$ tronview ad_analytics.XX.the_action_name
```

Once they run successfully manually, resume the rest of the job by skipping the action. `tronctl skip ad_analytics.XX.the_action_name`





What made this runbook **difficult** to use?



### 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_analytics.XX.the_action_name
```

Once they run successfully manually, resume the rest of the job by skipping the action. `tronctl skip ad_analytics.XX.the_action_name`



STORY TIME: BATCH  
RECOVERY

## What made this runbook **difficult** to use?

**2:00 am**  
Paged for failed  
batch job.

-----  
**ALERT:**  
**ad\_analytics**  
**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**  
-----

**2:40: am**  
Page secondary  
oncall

**2:50 am**  
Secondary oncall comes online



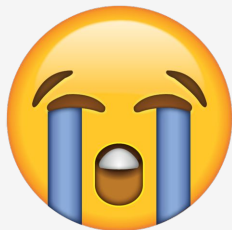
Me: “Where can find the rerun command?”

Secondary: “You can try looking for that in the wiki”

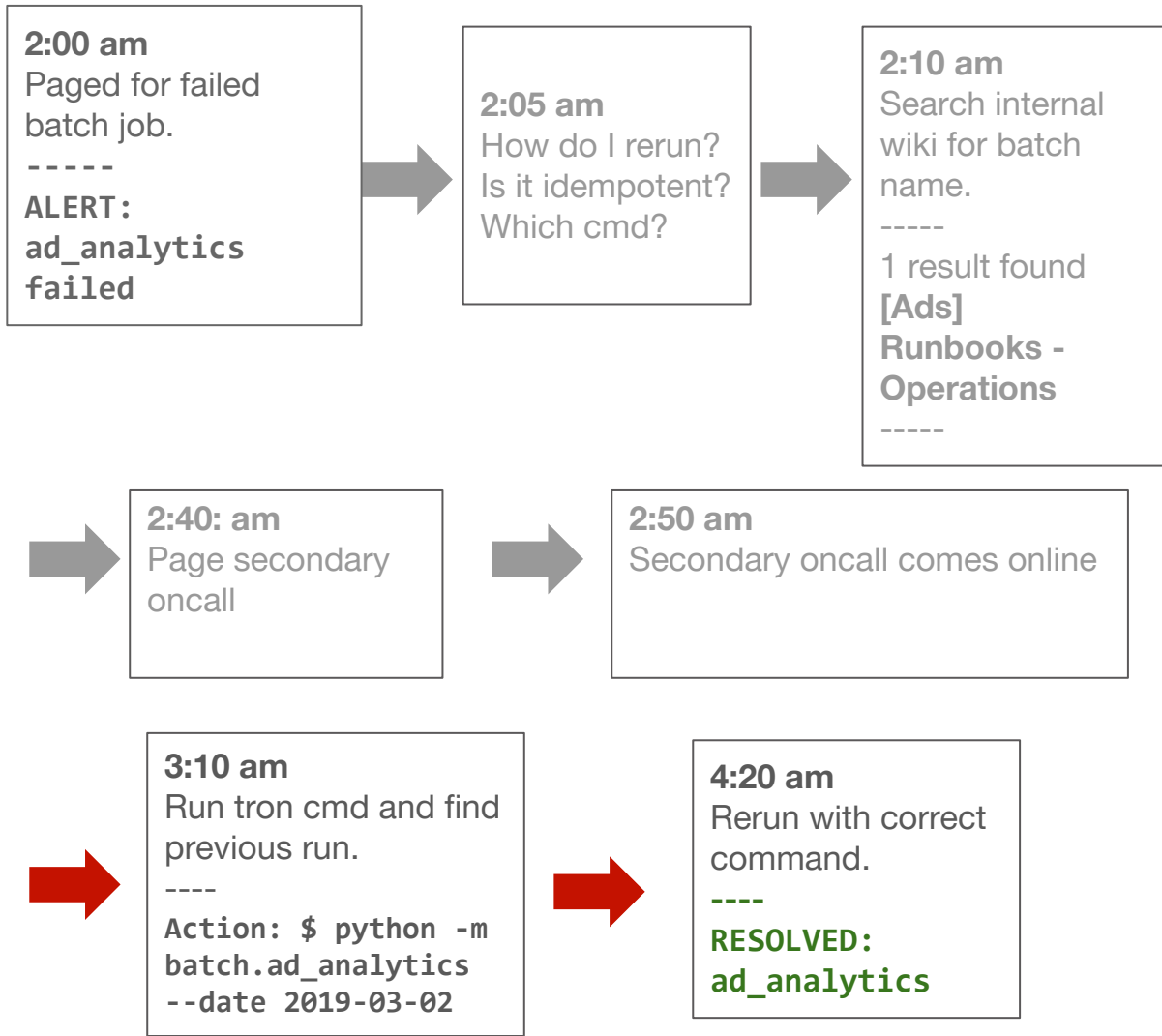
Me: “I just checked, but it’s not very clear.”

Secondary: “Or maybe it’s in the Google Docs repo.  
Oh, and I’ve got some notes in my home directory,  
and I think I saw some emails about that a while  
ago.”

Me:



## What made this runbook **difficult** to use?



## What made this runbook **difficult** to use?

- ✧ Information overload
- ✧ No clear action items
- ✧ Ambiguous wording
- ✧ Out of date
- ✧ Hard to find/search



What makes a **good** Technical  
runbook?



## **Tips for writing good technical runbooks**

- ❖ **Inverted pyramid**
- ❖ **Map alert to clear action items**
- ❖ **Include actual commands/screenshots**
- ❖ **Keep it up-to-date**
- ❖ **Keep format consistent**



<b>Alert Name</b>	<exact alert name>
<b>Description</b>	<1 sentence description>
<b>Stakeholder impact</b>	<1 sentence impact>
<b>Mitigation steps</b>	<ol style="list-style-type: none"><li>1. Try restarting: &lt;command&gt;</li><li>2. Monitor dashboards.</li><li>3. Inspect logs to diagnose issue: &lt;link or See steps below&gt;</li></ol> <p>If things do not recover, follow Escalation steps.</p>
<b>Escalation steps</b>	Contact <team>. Massive ingestion delays should be communicated to <upstream and downstream teams>.
<b>Related services</b>	<upstream and downstream dependencies>
<b>Dashboards</b>	<links>
<b>Related links</b>	<other docs or related runbooks>





# Exercise

**Let's make your own runbook!**

- 1. List all alerts**
- 2. Customize the template**
- 3. Pick a home for runbooks**



**Step 1:**  
**List all alerts**



**2 mins**



# Example: Ad Analytics Pipeline

## Alert:

Ad Analytics Data Processing Failure

## Past Incidents:

- Backward-incompatible input schema change
- MapReduce task timeouts



**Step 2:**  
**Customize** the template



## **Tips for writing good technical runbooks**

- ❖ **Inverted pyramid**
- ❖ **Map alert to clear action items**
- ❖ **Include actual commands/screenshots**
- ❖ **Keep it up-to-date**
- ❖ **Keep format consistent**



## http://bit.ly/srecon19-oncall

<b>Alert Name</b>	<exact alert name>
<b>Description</b>	<1 sentence description>
<b>Stakeholder impact</b>	<1 sentence impact>
<b>Mitigation steps</b>	<ol style="list-style-type: none"><li>1. Try restarting: &lt;command&gt;</li><li>2. Monitor dashboards.</li><li>3. Inspect logs to diagnose issue: &lt;link or See steps below&gt;</li></ol> <p>If things do not recover, follow Escalation steps.</p>
<b>Escalation steps</b>	Contact <team>. Massive ingestion delays should be communicated to <upstream and downstream teams>.
<b>Related services</b>	<upstream and downstream dependencies>
<b>Dashboards</b>	<links>
<b>Related links</b>	<other docs or related runbooks>

## http://bit.ly/srecon19-ocall



**2 mins**

<b>Alert Name</b>	<exact alert name>
<b>Description</b>	<1 sentence description>
<b>Stakeholder impact</b>	<1 sentence impact>
<b>Mitigation steps</b>	<ol style="list-style-type: none"><li>1. Try restarting: &lt;command&gt;</li><li>2. Monitor dashboards.</li><li>3. Inspect logs to diagnose issue: &lt;link or See steps below&gt;</li></ol> <p>If things do not recover, follow Escalation steps.</p>
<b>Escalation steps</b>	Contact <team>. Massive ingestion delays should be communicated to <upstream and downstream teams>.
<b>Related services</b>	<upstream and downstream dependencies>
<b>Dashboards</b>	<links>
<b>Related links</b>	<other docs or related runbooks>

**Step 3:**  
**Pick a **home** for runbooks**

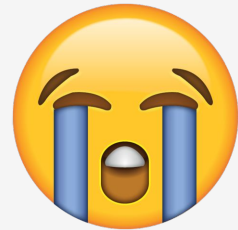


**1 mins**





“You can try looking for that in the wiki, or maybe it’s in the Google Docs repo. Oh, and I’ve got some notes in my home directory, and I think I saw some emails about that a while ago”



## A good runbook is easy to find



### **Make alerts rich**

Put actual commands and/or runbook link in the alert



### **Centralized “home”**



### **Make runbooks searchable**



**Step 3:**  
**Pick a **home** for runbooks**



**1 mins**



# **Example Non-Technical Runbook**



## Non-technical runbook

<http://bit.ly/oncall-srecon19>



# 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

<http://bit.ly/oncall-srecon19>



# Incident Response Checklist

## Communicate

- ❑ [Create a ticket](#) in the ADS project with a brief description of the issue.
  - ❑ Add secondary and manager as watchers
- ❑ Consolidate triage communications to #ads-incident.
- ❑ Send email to ads-incident@ to liaise with financial stakeholders and downstream consumers of data: [email templates](#).



## Productive and Happy On-call

### Oncall Training

- Debunk myths
- Avoid information overload
- Use Visual Aid
- Focus on tools

### Beyond training

- Knowledge share
- Wargames
- Effective Runbooks

 **Continuous Improvement**

**Training materials**

<http://bit.ly/srecon19-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](#)

### Postmortems/wargames

- [Postmortem culture: learning from failure](#)
- [The oncall simulator: Building an interactive game for teaching incident response!](#)





**Thank you!**

Optional Materials:

**<http://bit.ly/srecon19-oncall>**

