SITE RELIABILITY ENGINEERING, OPERATIONS, AND PRODUCTION ENGINEERING

CSC491, UNIVERSITY OF TORONTO

WE WILL TALK ABOUT:

LECTURE 1

- What is Operations? SRE? Production Engineering?
- What is the difference?

LECTURE 2

- Infrastructure and hosting
- Teams and focuses
- On-call schedules

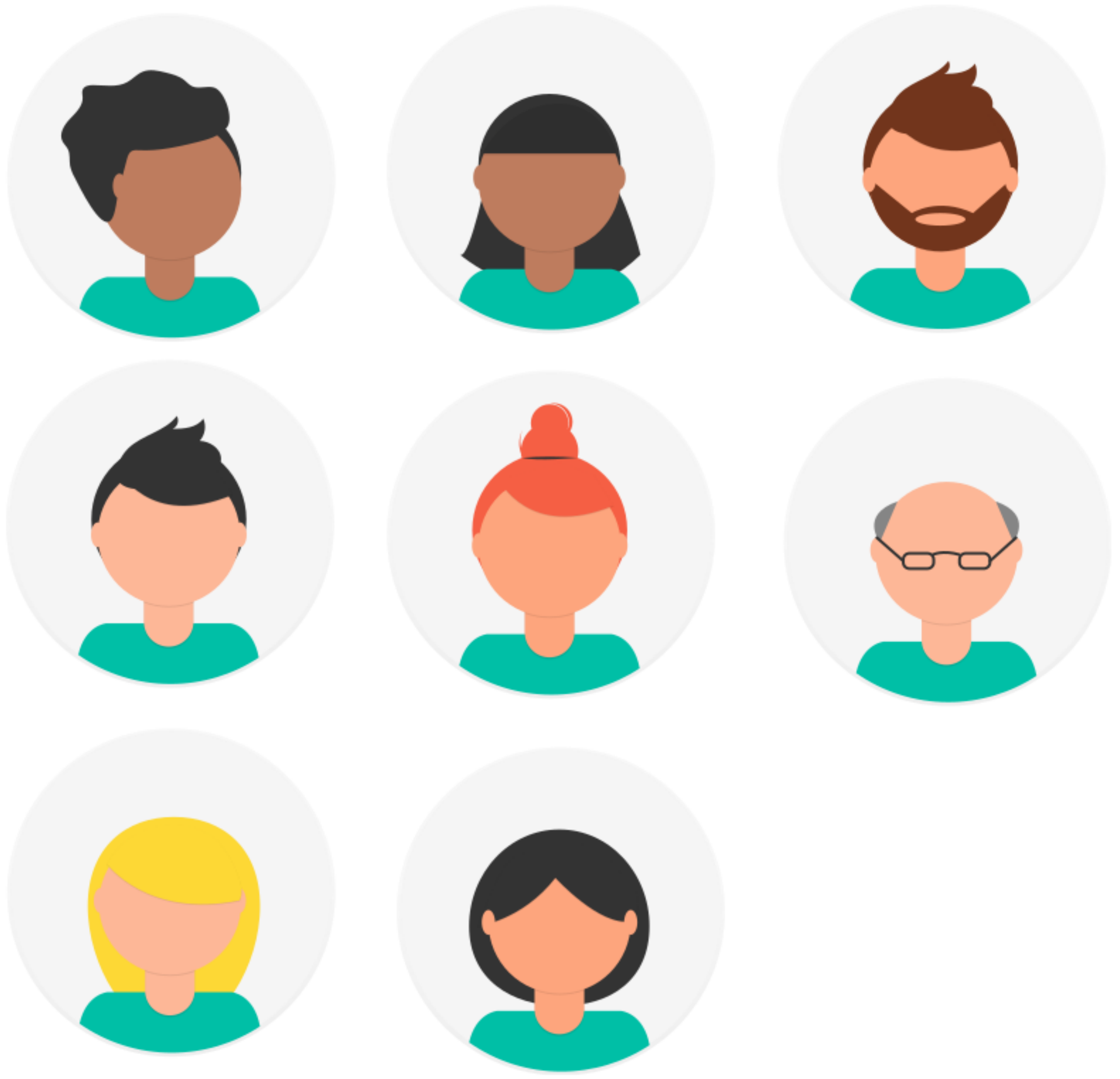
OPERATIONS

WHAT IS “OPERATIONS”?

- This is a more traditional model of running hardware and servers.
- In this model, you have a person or a team who are in charge of running servers on which to host software. Often they will be the ones handling deploys as well.

WHAT IS "OPERATIONS"?

- This generally scales poorly as you must employ an increasing number of operations people to manage an increasing load, instead of automating it with software



MY OPINION OF OPERATIONS

I'm not a fan of operations-style management of production systems.

- It doesn't scale very well
- It requires people to do things that servers are better, and less error-prone, at - click and type configuration

MY OPINION OF OPERATIONS

This may be easier at the beginning though

- Hiring a contractor or part-time person whose job it is to keep the servers running is much easier than hiring a team
- When you have few servers this is a very isolatable role from the rest of your product team and lets you focus on your product.

SITE RELIABILITY ENGINEERING (SRE)

“SRE IS WHAT YOU GET WHEN YOU TREAT OPERATIONS AS IF IT’S A SOFTWARE PROBLEM.”

THIS IS A PHRASE THAT WAS COINED BY GOOGLE: HTTPS://LANDING.GOOGLE.COM/SRE/

“SITE RELIABILITY ENGINEERING SEEKS TO BALANCE THE RISK OF UNAVAILABILITY WITH THE GOALS OF RAPID INNOVATION AND EFFICIENT SERVICE OPERATIONS, SO THAT USERS’ OVERALL HAPPINESS—WITH FEATURES, SERVICE, AND PERFORMANCE—IS OPTIMIZED.”

MARC ALVIDREZ, SITE RELIABILITY ENGINEER, MOUNTAIN VIEW

"HERE'S WHAT YOU DO WHEN SOMEONE BREAKS SOMETHING OR FINDS SOMETHING VERY DIFFICULT TO DEBUG: YOU SAY THANK YOU [...] FINDING THIS EDGE CASE, HIGHLIGHTING THIS OVERCOMPLICATED PART OF OUR SYSTEM, POINTING OUT THIS GAP IN OUR DOCS. AND THEN YOU GO MAKE IT SO NOBODY CAN BREAK IT THE SAME WAY AGAIN."

TANYA REILLY, GOOGLE SRE 2005 - 2018

WHAT IS “SRE”?

- Fundamentally, SRE is a practice in which the running of your <u>systems is managed via configuration software</u>
- SRE outlines some core tenets as well, such as:
  - a blameless culture
  - psychological safety to move and work around in production

PRODUCTION ENGINEERING

"PRODUCTION ENGINEERING IS A COMBINATION OF MANUFACTURING TECHNOLOGY, ENGINEERING SCIENCES WITH MANAGEMENT SCIENCE. A PRODUCTION ENGINEER TYPICALLY HAS A WIDE KNOWLEDGE OF ENGINEERING PRACTICES AND IS AWARE OF THE MANAGEMENT CHALLENGES RELATED TO PRODUCTION."

HTTPS://EN.WIKIVERSITY.ORG/WIKI/PRODUCTION_ENGINEERING

WHAT IS “PRODUCTION ENGINEERING”?

- Originally a classic engineering term (mechanical, civil, etc) with regards to factories
- The goal is to accomplish the production process in the smoothest, most-judicious and most-economic way

WHAT IS "PRODUCTION ENGINEERING"?

- <u>Scale and integrate resources.</u>
- Usually required to consider physical, human and financial resources at high efficiency and low cost, yet considering the possibility of continuous further improvement
- <u>Make proper use of math and statistics</u>
  - Used to model production systems during decision making process
- <u>Design, implement and refine products, services, processes and systems</u>
  - Takes into consideration the constraints and particularities of the related communities

WHAT IS "PRODUCTION ENGINEERING"?

- <u>Predict and analyze the demand</u>
  - Select among scientific and technological appropriate knowledge in order to design, redesign or improve product/service functionality
- <u>Incorporate concepts and quality techniques</u>
  - Deploy organizational standards for control proceedings and auditing
- <u>Stay up-to-date with technological developments</u>
  - Enablee them in enterprises and society

WHAT IS "PRODUCTION ENGINEERING"?

- Understand the relation between production systems and the environment.
  - This relates to the use of scarce resources, production rejects and sustainability
- Manage and optimize flow (information and production flow)

PRODUCTION ENGINEERING
VS
SITE RELIABILITY ENGINEERING

HONESTLY, THE SYSTEMS ARE VERY SIMILAR. THERE ARE SHARED CONCEPTS AND EACH SYSTEM HAS PROS AND CONS.

I'M GOING TO USE AN EXAMPLE FROM A QUORA ANSWER FROM A LEAD RECRUITER AT FACEBOOK. I'LL SUPPLEMENT IT WITH MY EXPERIENCE AS A PRODUCTION ENGINEER AT SHOPIFY.

THE LINK TO THE QUORA ANSWER CAN BE FOUND HERE

THE FUNDAMENTAL DIFFERENCES

- This means that production engineers are embedded on the teams or work alongside the teams to help them run their own services
- Shopify had a little bit of both: Some things were run more like an SRE team (databases, caching, core servers) and some things were more a production engineering focus (developers and SREs worked together) like smaller app servers

THE FUNDAMENTAL DIFFERENCES

- At Google, SRE’s are supporting SWE’s (software engineer) in a “throw it over the fence” model.
- This means that software engineers build the software and someone else runs it
- Production Engineering at Facebook is more of an embedded model

THE FUNDAMENTAL DIFFERENCES

- Both are driven by engineering decisions, not hardware-related ones.
  - A good example of this is a system that Facebook built to automate 300 sysadmins work to find, test, and restart servers in production autonomously.
  - Software is good at detecting repeatable patterns, performing tedious tasks, and completing these 24/7.

TOOLING AND AUTOMATION

- SRE and Production Engineering both require heavy amounts of automatic and autonomously functioning tooling and detection.
- You'll find that both areas tend to invest heavily in orchestration software (Packer, Chef, Kubernetes, etc), monitoring and alerting (observability), and all things tooling/automation.

WHEN YOU CAN SAVE EVERYONE A SECOND A DAY, BUT HAVE 1000 DEVELOPERS… YOU'VE ACTUALLY SAVED 17 MINUTES. OR ABOUT 68 HOURS (1.5 WEEKS!) OF DEVELOPER TIME.

YOU ARE A FORCE MULTIPLIER.

HOSTING, TEAMS, AND ON-CALL SCHEDULES

CSC491, UNIVERSITY OF TORONTO

HOSTING

HOSTING

- A few main ways to host apps:
  - Cloud Provider
  - Own datacentres
  - Full Service providers

CLOUD PROVIDERS

- Examples include AWS, Azure, Google Cloud, Digitalocean
- Provide servers that they manage, sometimes managed services like databases
- Somewhat expensive
- Some work involved

OWN DATACENTRES

- You buy your hardware
- Rent or buy datacentre space
- Install and run it yourself
- Requires operations and “hands on” people
- Less expensive
- Most amount of work

FULL SERVICE PROVIDERS

- Examples include Heroku, and most cloud providers
- Provide the entire stack to host your app
- You choose which add-ons to use (like databases, redis, etc)
- Most expensive
- Least amount of work

TEAMS AND FOCUSES

THE FUNDAMENTAL DIFFERENCES

- **This section will focus on teams and skillsets I've experienced while working at Shopify, consulting, and working with various companies.**
- **There are a number of teams that I find exist at various points in a company's lifespan**

TEAMS AND FOCUSES

- Datastores
  - Databases
  - Caching
  - Queueing
- Networking
  - “The Edge” (where a request leaves and enters your infrastructure)
  - Emails
  - Internal Intranet

TEAMS AND FOCUSES

- Cloud (AWS, Google Cloud, Azure)
- Continuous Integration + Deployment
- Developer Tooling
- Regulated Systems (PCI Compliance, SOX compliance, etc)
- Observability (Logging and Monitoring)
- Language and Platform

DATASTORES

- Databases: Typically handles MySQL, Postgres, etc.
- Caching: Typically handles Redis, Memcached, and other Cache systems.
- Queueing: Typically handles Redis (as a queue), RabbitMQ, and other queue systems.
- For each of these systems, the respective team will manage scaling it, making it easily available where it needs to be, and ensuring that apps don’t abuse limits.

NETWORKING

- “The Edge”: Where a request leaves and enters your infrastructure. This will typically include load balancers, BGP routers and announcements, Points of Presence for TLS termination
- Emails: Handling email equipment, scaling that for the organization and ensuring IP address you use will not be listed as spam
- Internal Intranet: Handles internal network routing between systems. Handles databases connecting to app servers, cache and queues connecting, etc. Often includes manual operations-like work unless you’re in a cloud. Can use BGP for this

CLOUD (AWS, AZURE, GOOGLE CLOUD, ETC)

- Manage integration with services on the cloud, often using orchestration software.

CONTINUOUS INTEGRATION + DEPLOYMENT

- Manages scaling CI systems for testing infrastructure, decreasing time spend on test runs, and manages quick/easy deployment to production

DEVELOPER TOOLING

- Manages creating and developing tooling to get a software project up and running on a computer for development.
  - Includes installing languages, datastores, project dependencies etc locally.
- Often uses some form of orchestration software
- Has to be able to handle a system in any state as developers may change settings at random

REGULATED SYSTEMS (PCI COMPLIANCE, SOX COMPLIANCE, ETC)

- I’ve seen regulated systems (e.g. PCI compliance for Credit Card Processing) be managed by separate teams. This is to reduce the number of people that have access to sensitive systems.

OBSERVABILITY (LOGGING AND MONITORING)

- Logging and Monitoring may not seem like a big problem to solve, but each request to a system will often send out dozens of different telemetry metrics and dozens of logs. This means that those systems need to handle a large load.
- This team ensures that logging and monitoring can keep up with data thrown at it and ensures it can respond quickly to people who need to use that data.

LANGUAGE AND PLATFORM

- Sometimes when your company gets big enough, it starts to drive the full feature development of a framework or language.
- Both GitHub and Shopify employ core contributors to Ruby and Rails teams. This allows them to remove painful parts that reduce productivity of their own engineers, but also give it back to the world through open source.
- A recent example can be seen in Rails 6. GitHub has a multi-database configuration, Rails did not support this. GitHub's engineer, Eileen, worked tirelessly with her team to integrate this functionality in the Rails framework. Now everyone gets to use this feature for free and GitHub doesnt have to maintain something that only they use.

ON-CALL SCHEDULES

ON-CALL SCHEDULES

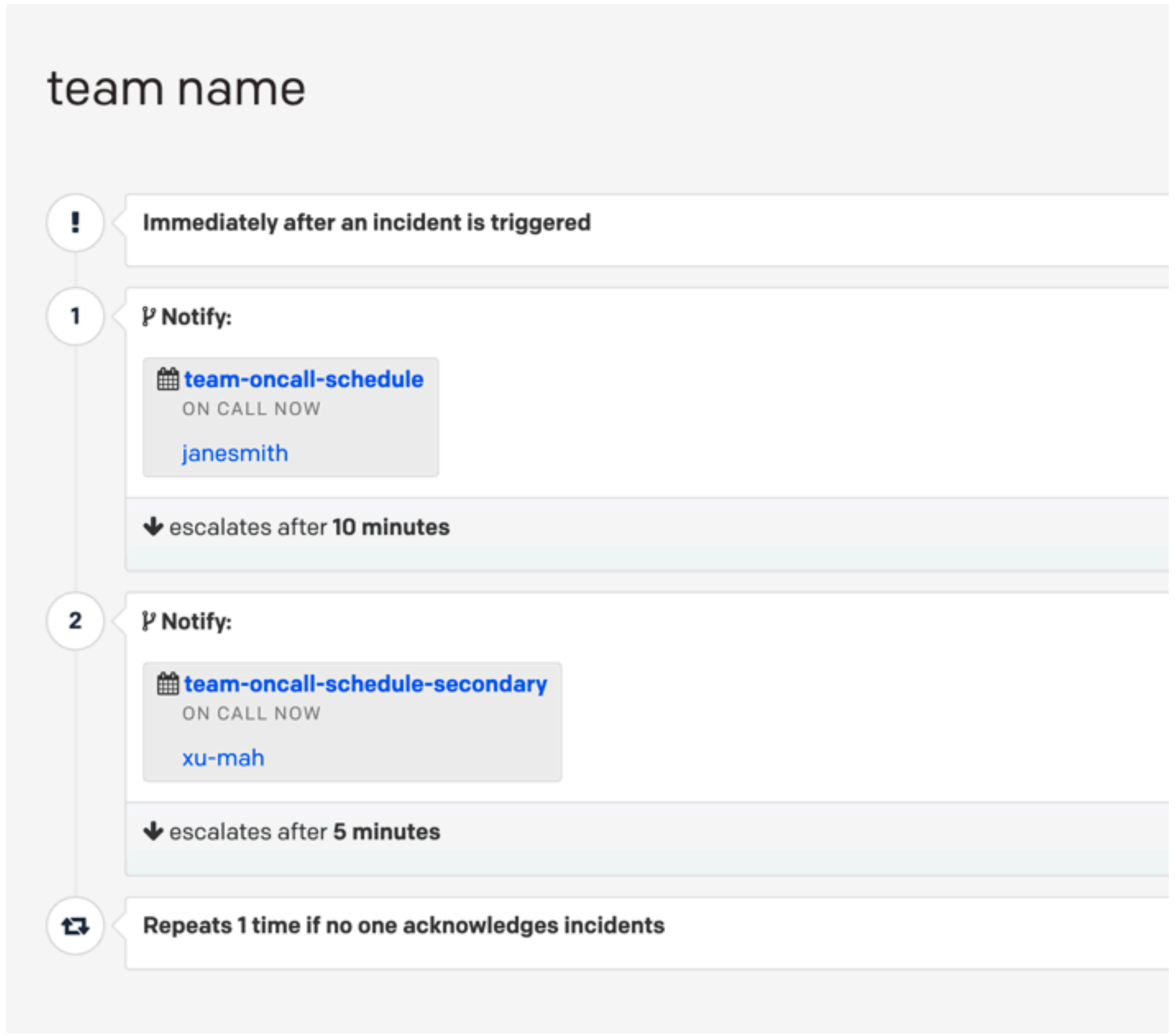
- **An industry standard is to use tools like Pagerduty**
- **A fancy notification service that takes alerts from your systems and alerts/wakes up an engineer to fix a problem**
- **Must have high uptime because they page you when your system is down**

ON-CALL SCHEDULES

- **Examples:**
  - **A database is not responding in time, causing exceptions**
  - **Page an engineer on the databases team**
  - **An app is not responding**
  - **Page an engineer working on that app**

ON-CALL SCHEDULES

- Oncall schedules can be for infrastructure OR for software.



team name

Immediately after an incident is triggered

1 Notify:

team-oncall-schedule
ON CALL NOW
janesmith

escalates after **10 minutes**

2 Notify:

team-oncall-schedule-secondary
ON CALL NOW
xu-mah

escalates after **5 minutes**

Repeats 1 time if no one acknowledges incidents

FULL SERVICE PROVIDERS

- Examples include Heroku, and most cloud providers
- Provide the entire stack to host your app
- You choose which add-ons to use (like databases, redis, etc)
- Most expensive
- Least amount of work

PHOTO CREDITS

- https://unsplash.com/photos/aWslrFhs1w4
- https://unsplash.com/photos/IgUR1iX0mqM
- https://unsplash.com/photos/uOhBxB23Wao
- https://undraw.co

CREDITS

- How does being a production engineer at Facebook compare to being a site reliability engineer at Google? by Sief Khafagi
- How to get started with Site Reliability Engineering by Nikki McDonald
- Production engineering Wikipedia Page