DevOps at Scale is a Hard Problem: Challenges, Insights and Lessons Learned

Kishore Jalleda

DevOps Enterprise Summit 2017

San Francisco

Yahoo!

1+ Billion users

Web-scale Infrastructure

- **400k+ Servers**
- 6+ Data Centers
- 50+ Edge/POPs.

Let me tell you a story

Bob (not the real name), an engineer at Yahoo, wanted to **build a stock recommender prototype using portfolios data on grid using machine learning.** He starts to do this on his own personal time and gets a prototype working.

When I interviewed him and asked what is stopping him from testing his idea (and several others he had) in a bucket quickly, he responded, **“I wish there were fewer constraints”.**

How can you help people like Bob?

Engineers who come to work everyday wanting to **change the world**

Engineers who want to build something **customers love**!

How can you find more like him at your company?

That is when it struck me

“DevOps” is really about eliminating (most) Technical, Process and Cultural barriers between Idea and Execution -- **using Software**.

Create Intrapreneurs

Democratize Innovation

Velocity (ship fast; fail fast; learn fast).

MVP

Find a co-founder

Found my partner in crime.

Our journey had begun.

“DevOps” to us is about:

Enable a **Culture** of Ownership & Excellence

Engineer Agile & Automated **Processes**

Develop (Re)Usable & Self-Serve **Tools**

to kick ass at…

Delivery	Prevention	Repair

Initial response (to our strategy deck)

“you are preaching to the choir”.

“what is your day job?”

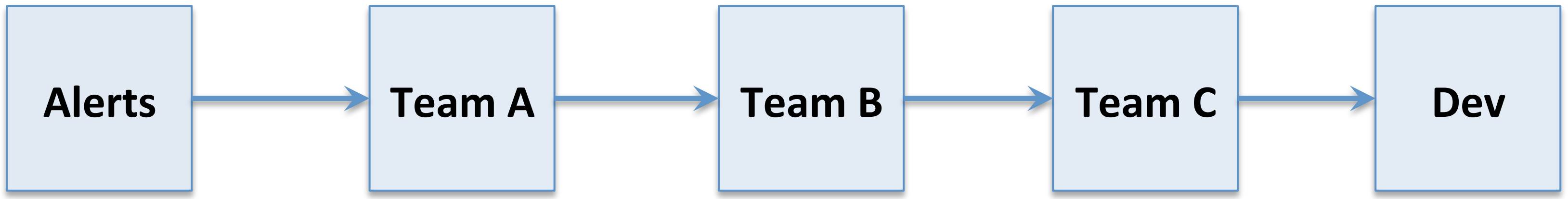
We started executing - soon hit roadblocks.

Initiative #1: Directed Alerting

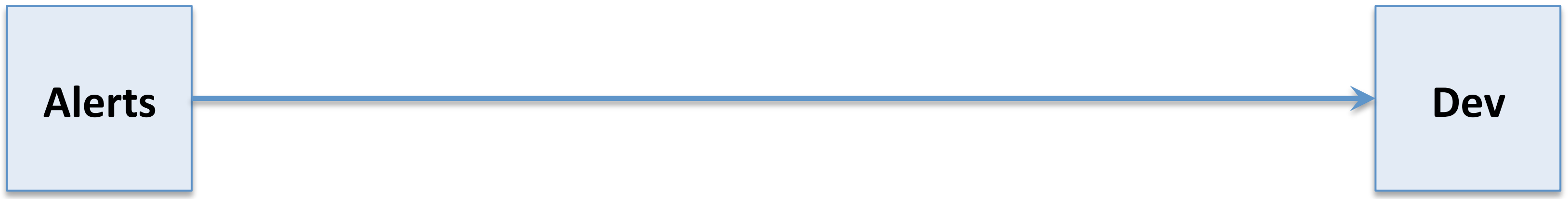
let me ask you a question.

Which one would you pick?

- Option 1:



- Option 2:



Directed Alerting

“You wrote it; you own/run it”

helps getting feedback from prod quickly and directly.

Turns out, it is a hard problem

“Sorry, we don’t need your service” - is a hard thing to say.

“We have always done things this way" - is a hard mindset to change.

"My devs are not going to like being woken up at 3 am"

When you are stuck

Be patient. Breathe. Calm down.

&

Follow these simple rules.

Rule #1: Have a strong Vision; communicate it widely

Page Dev teams directly

Get to <2 alerts /shift (so, you can RC each)

All alerts are actionable; all alerts require human intelligence

Rule #2: Leverage Failures/Outages

Conduct Postmortems

Ask thought-provoking questions; stir emotions; provoke action.

Don’t be a jerk though

Rule #3: Find your Allies

They won't come to you; you must find them.

Helps drive bottom-up change

Not making progress? Infiltrate your enemy territory to find your allies.

Rule #4: Go all in; take a stand; persist

Half-assed approached don't work

There will be resistance:

- – "Can we still send low priority alerts to them?"
- –"We will have no one else to blame"

Rule #5: Stay calm in tense moments

“This is a big change; don’t f*** things up”.

"You are doing this all wrong; this will cost us"

Rule #6: Top-down support is critical

Make sure your boss is aligned

Boss' boss is aligned

All the way up (set up meetings; onus is on you)

I found a strong Ally

Yahoo Daily Fantasy (new product; high profile; awesome leader)

Great opportunity something as radical as this is actually possible.

Rule #7: Learn to say “no”

Saying “no” is hard.

Also empower your teams to say "no".

Rule #8: Align Incentives

Team B became embeds into Team C.

Team A was shared with other BUs anyway.

(Team C slowly getting out of the way of Devs)

Rule #9: Don't miss the boat

Go/no-go decisions usually come down to a meeting (or two)

Make sure you are prepared.

Drive alignment prior; not during (or after)

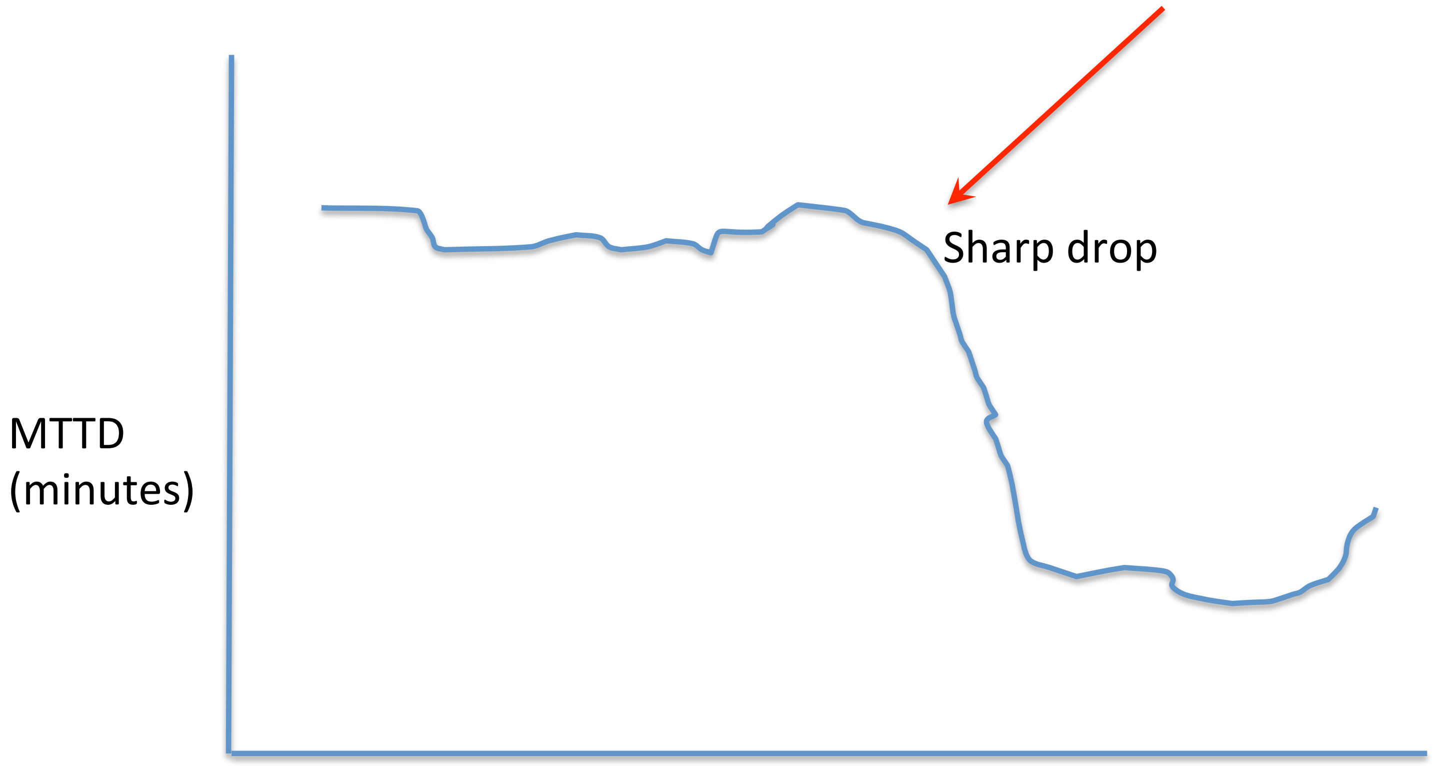
Rule #9: All wins must go Viral; start a movement

Showcase the team and win to the whole company (blogs; all hands; etc).

Show something as radical as this is possible.

People will start to pay attention

Directed Alerting - Results



Initiative #2: Continuous Delivery (CD)

Before we talk about this initiative, let me ask you a question.

Push/Deploy to Production - which do you prefer?

Option 1: “No Humans Allowed”.

Option 2: “Humans allowed”

CD – major push in 2013/2014

- Top-down initiative.
- Buy-in from Marissa.

But, CD at scale is hard

“No humans allowed” - Can be scary.

But, it’s the right thing to do; it’s how modern software must be built.

CD (at scale) – expect failures early on

You will fail more often than you will succeed.

There will be dark, gloomy days

Key is not to give up.

CD – “Warp Drive” to the rescue

A great program at Yahoo.

Helps drive that bottom-up energy so it can align with the top-down.

A highly effective way to bring about a cultural change and achieve technical excellence.

There will still be stragglers

Some will do it for hitting quarterly goals

Will not embrace its spirit

Strategy for Stragglers

Infiltrate team with evangelists/champions

Public shaming / Peer pressure (in moderation)

"Law of Velocities" will eventually kick in

CD - results

Weeks/Days to multiple times a day

PEs no longer do deployments; hundreds of man hours saved.

CD is possible at massive scale.

But, it's 2017 - **CD is table stakes**

(people have been doing CD for more than a decade)

Initiative #3: Automation Culture

Same drill. Before we talk about this initiative, let me ask you a question

A Server/VM/Container is a bad state

Option 1: Wake up a human at 3 am, have him/her take that that resource OOR manually.

Option 2: Automatically take it OOR, run some diagnostics, create a ticket and assign to human.

But wake someone up

When, let’s say, 15% of the cluster is in a bad state.

Challenges

“What about our job security?”

"I don't have the time"

Who said making strategic bets is easy?

All about trade-offs

Ruthless prioritization is critical

My promise to the team

Higher-value-add work: writing software; infra; tooling; etc.

You can’t possibly automate yourselves out of your job.

Unless you say "no" to things that don't matter, you can't say "yes" to things that matter.

Tools built

Auto Remediation

Failure Discovery

Metrics based promotions

Kubernetes

So much more!

Results

100s of auto remediations/hour in prod

Hundreds of man hours saved by reducing “toil”.

Dramatic reduction in (repeat) incidents

Insights and Lessons learned

Tools that only Ops can use are not really tools.

Don't build tools in a vacuum - no one will use them.

Initiative #4: AWS/Hybrid Cloud

One last question before we talk about this initiative.

* - this is not an AWS endorsement. AWS did not pay me for this.

Which one would you pick?

Option 1: Public

Option 2: Private

Option 3: Hybrid.

What problem are you trying to solve?

- Failsafe/Fallback
- Load Testing
- Non-prod / Test frameworks
- Rapid Experimentation
- Launching New, new products (not much dependencies on existing/legacy stacks)

Cultural Challenges

People are afraid to break the rules.

Don't be afraid, just make sure you **break them in broad daylight.**

Make enough noise - People will notice

Getting "consolidated billing" was key

Letting people know that is OK to use it for experimentation was key.

Soon became a corp goal.

Results

Fail-safe stack on AWS

Many new products launched natively on AWS

More to come!

Being selfless

Just because you start something; doesn't mean you are the one ending it.

Blazing the trail is key; someone's got to do it.

Closing thoughts

"DevOps" Anti-patterns

If you find any, reverse them.

A **better model** (call it **“DevOps”** or whatever)

Core Dev Teams own build, test, deploy, monitor, on call, debugging, incident response, capacity, Postmortems, etc.

Ditto for **Non-core Dev & Ops Teams - but they focus on** infrastructure, automation, tooling, network, Databases, Dev productivity, expert services, observability, etc.

Reflect; soul search

As yourself:

"why does my team exist?"

"How is it providing value"

Uptime is overrated

Customers don't care about **five-nines reliability**

customers care about **five-nines customer service**

Velocity is overrated; customer feedback is underrated

Not enough if you ship multiple times a day; focus on customer feedback and quick learning after you ship.

Democratize Ops: every one is Ops; every one is Dev

Ops is moving UP the stack

Dev is moving DOWN the stack

\\\with the democratization of distributed systems and orchestration frameworks

Does "DevOps" matter anymore?

Yes. Of course. But, it is **table stakes.**

Strategic differentiator is **customer obsession.**

Become an engineer your customers love - someone who truly **empathizes with customers.**

You need more Bobs

It's your job to find them; it's your job to groom them; it's your job to remove barriers.

Good luck!

Thank you!

(Questions?)

Twitter: **@KishoreJalleda**
[LinkedIn]
(appreciate/love any feedback)