
VictorOps
PUTTING
DEVS
ON-CALL:
How You Can **Empower Your Team**

"New hires should be kept off **the on-call rotation** until they know the platform **inside-and-out.**"

"It's not my job **to test the code. If there's a mistake,** talk to QA."

Back **in the day**

"I just write the code **and throw it over the wall.** It's up to Ops **to make it work.**"

"I've never had to **carry the pager before...** why should I start now?"

Theory behind **the Why**

Luke Kanies of Puppet Labs puts it this way...

> Ideally, companies work out a system in which whoever makes the mistake pays the price. **The reason Ops is so often scared of Dev deploying is that Dev doesn't really care how secure their apps are, how hard they are to deploy, how hard they are to keep running or how many times you have to restart it, because Ops pays the price for those mistakes, not Dev.**
>
> In most organizations the mandate of a developer is merely to produce a piece of software that worked on a workstation -- if it worked on your workstation and you can't make it work in production, it's Operations' fault if they can't get that to thousands of machines all around the world.
>
> http://readwrite.com/2011/08/23/devops-what-it-is-why-it-exist

According to Ernest Mueller, one of the authors of the Agile Admin:

DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support. DevOps means breaking down the traditional silos that have existed between Ops and Devs.

DevOps Practices-Specific techniques used as part of implementing the above concepts and processes.

Continuous integration and continuous deployment

"Give your developers a pager and put them on-call"

Using Configuration management, metrics and monitoring schemes

Using virtualization and cloud computing to accelerate change

http://theagileadmin.com/what-is-devops/

Atlassian **recommends:**

> Devs can win instant good will by helping to carry one of Ops' biggest burdens: the pager. **Developers tend to bring more craftsmanship into their work knowing that they, or their pals, might suffer the consequences of half-baked code.** Being waist-deep in the immediate resolution and accompanying troubleshooting efforts also provides a valuable perspective on how the application's architecture interacts with the infrastructure it sits on.
>
> Clearly define (and yes: document) escalation paths for emergencies and make sure devs have access to the tools and data they'll need for troubleshooting. Then include the development team in on-call rotation.
>
> https://www.atlassian.com/devops/culture#!shared-responsibilities

At Etsy...

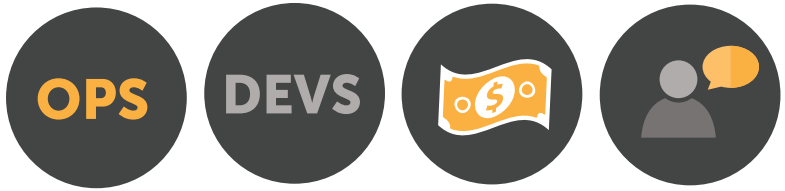
"If you are writing code, you are on-call."

Etsy has devs on-call to address the problem of IT Operations asking,

"Why should I be the only person waking up at 3am?"

They empower developers with responsibility:
let them deploy, have them on-call, no passwords, etc.

On-Call Schedules



Dev On-Call

"Make sure that developers stick around to ensure that their deploy worked."

http://www.slideshare.net/mrtazz/goto-amsterdam2014depolyment1024

How to **get started?**

What if you're not Etsy and you want to get buy-in from your devs... is there a way to make it easier for devs to be included in the on-call process?

Read on to **see how these 4 companies did it...**

Improving the knowledge **base across teams**

Production systems used to be a black box to most people on our team. Only a few developers had any insight whatsoever.

Now that the entire team is on-call (be it in a very flexible way), our front-end devs are really excited when an alert goes off because they want to be the first one to ack it. And it is making our devs better devs because they have a holistic view of our system.

I've built on-call systems myself before, and I'm a firm believer in the developers being on-call. A lot of companies aren't big enough to have dedicated on-call teams of 10 or 20 people, and devs are smart and very capable of doing the work.

Bunchball is an agile development shop, so all teams are self-managed. The trick was convincing everyone that putting devs on-call was the right idea.

Our devs now have an understanding of how the system functions on a day-to-day basis. It feels like everyone is more engaged in the ongoing success of our product.

Nick Goodman,
Director of Platform Engineering, Bunchball

BUNCHBALL
THE LEADER IN GAMIFICATION

Creating a new mindset **for devs on-call**

Before our developers were on-call, I usually had to deal with their deploy that they broke on a Friday night.

Now each of our 13 applications has 2 people in rotation for on-call–and one of those is always a dev. It has created a completely different mindset for our dev team. Now that they know they can be woken up at 4 in the morning, they deploy when they know they can be available to fix things.

There are fewer production issues now that our devs fully understand what it means to push to production.

Michael D'Auria,
Infrastructure Lead at CrowdTap

Making it easier to get devs **into the on-call rotation**

Most of our devs come from writing desktop applications. We wanted to get them into our on-call rotation so they have that shift in mindset to a more holistic view. There was trepidation and a little bit of grumpiness at first because they've never done it before.

Now, we have a paired on-call rotation with our techops and dev team. Each can manage their on-call schedules independently, and each person can manage their notification policies independently.

If the first tier people don't respond to alerts in a certain amount of time, it defaults to a secondary. Then the third tier is me, and the fourth tier is management.

Paul Beltrani,
TechOps at Onshape

Onshape

Empowering devs **while on-call**

We restructured how on-call works. It used to be just Operations team (2-3 people) rotated once per week.

We thought it would make it easier to include the broader team - 25 people in rotation. Alerts would go to the engineer on-call first. If they don't ack within 5 min, it would go on to the next tier of the Operations team.

Some engineers laughed it off. It was new to them and they felt like they didn't know how to do anything.

As a result, we've more than quintupled the number of ops people and engineers in rotation at any given time. This has been especially good for engineers as it gave them a more holistic build-to-deploy view of things. We even built a wiki with common problems and and how to resolve them so everyone felt like they had more power and could take responsibility.

Matt Knox,
SysAdmin at TrackVia

TRACKVIA

http://victorops.com/blog/onshape-case-study/

http://victorops.com/blog/bunchball-case-study/

http://victorops.com/blog/trackvia-case-study-victorops-improved-ttr/

http://www.confreaks.com/videos/2358-mwrc2013-you-should-be-on-call-too

http://www.javacodegeeks.com/2014/01/developers-working-in-production-of-course-maybe-sometimes-what-are-younuts.html

http://www.thoughtworks.com/insights/blog/n%C3%A3o-existe-equipe-de-devops

http://www.paperplanes.de/2013/1/2/on-pager-duty.html

Additional **Resources**