Be Your Company's IT Hero: Be Excellent in Crisis

Treating failure as inevitable provides a basis for engineering robust, high-performance solutions

Failure is normal. Experience of failure makes us stronger – gives us 'grit' and resilience and makes us, ultimately, more prone to succeed.

This is, at least, what they tell the privileged tech bros, burning through angel rounds. Am I right? The rest of us are counseled that failure (as in service outages) is disastrously costly: inflicting damage to revenue and business reputation, so a must to avoid.

And yet, we IT Heroes, too, can profit from the understanding that failure is normal. In fact, failure is happening all the time in complex systems, and is often no big deal. We engineer deployments and build applications with this understanding: that our jobs are very much about delivering uninterrupted services, despite the fact that things are failing all the time in lower levels of our application stacks and infrastructure.

Knowing this, the question on the table is: how can we further leverage and extend our understanding of systems failure, to better and more efficiently provide uninterrupted service, while also making ourselves look smart and earning raises and promotions?

Your Circus, Your Monkeys

First, some homely folk wisdom. There's an old and deeply-cynical Polish expression – Nie mój cyrk, nie moje małpy (literally: "Not my circus, not my monkeys") – that lets one disclaim responsibility for an evolving, chaotic scenario with obvious incipient and catastrophic failure modes. The perfect opportunity to use this saying is when you've given the circus-owner some really good advice, which has been ignored – and you're now standing on the sidelines, watching the predicted disaster unfold. You can earn extra international amity points by using this expression in combination with the classic Gallic Shrug, which conveys roughly the same idea.

The thing is, if you're an IT Hero, it actually is your circus, and those are your monkeys. Luckily, just like Matt Damon in the film The Martian, you can "science the heck out of this."

Monkey Minding for IT Heroes

Before we get to the science, be advised that you can also common-sense the heck out of this, with the help of your team.

Recall from our solution brief on Proactive IT that we recommended creating and fully resourcing a team to do proactive maintenance on systems and infrastructure. Extending this idea, here are some more:

Review on-call frequently. Aspiring IT Heroes may already be assigned rotations in a well-thought-out and properly-incentivized on-call structure. It makes sense to review these frequently at every level, and make sure managers and leaders are provided with information about current events, where peoples' heads are at, conflicting responsibilities, incipient burnouts, etc., so they can reallocate resources in healthier, more effective ways.

Limit on-call time and night shifts. This is serious stuff. The most effective, responsive on-call teams in the world strictly

> Experience of failure makes us stronger – gives us grit and resilience and makes us, ultimately, more prone to succeed.

limit the time individual employees spend on call and under stress, because they have very demanding SLOs and need on-call staffers to be fully (and also non-self-destructively) engaged. (Those are, after all, your monkeys.) They are extremely careful in assigning night shift work, because they know that working night shifts is bad for health and service quality – much less effective than "follow the sun" rotations that put people on call during normal waking/working hours.

Train well and gently. Google and other top-performing organizations also insist on humane training of their SREs, usually by hands-off shadowing of more experienced engineers, followed by highly-supervised hands-on experience – the idea being to help newcomers feel more confident and less stressed out while learning. These organizations are also insistent about sharing out the on-call burden very widely – in particular, requiring that application developers take on-call responsibility for the things they create (avoids "throwing junk over the wall and letting Ops lose sleep over it") and making sure that managers, up to a certain level, are included in on-call rotations.

Don't let on-call folks become complacent. They also recognize that insufficient stress leads to detachment and erosion of knowledge. Google works very hard to balance stress with what they call "On-Call Underload," where engineers don't spend enough time on-call to polish fast-reaction forensic skills, or are on-call to manage only "quiet" systems that rarely experience issues.

Rehearse! On-call folks at the largest organizations often report that automated systems frequently slot them into rotations – effectively making them a part of crisis response teams – without managers taking the time to actually build out teams of folks scheduled to work together, to replace one another on shifts, or to make various on-call tiers mutually aware of each other before people are obliged to work together under pressure. Don't be that manager. Team members need to know one another, understand each others' skills, and practice before being thrown into the deep end of the pool.

The science part comes in when you start exploring the notion of using controlled chaos to better understand the failure modes of large systems. The touchstone for this is an intermittently maintained open source manifesto document called Principles of Chaos, which outlines a methodology for measuring steady-state system performance, and then applying empirical principles to determine how much confidence you should have in a system's resilience. Some of the most technologically-advanced organizations in the world apply these principles in an (apparently mostly successful) effort to force engineers to build systems that deal better with failure. There are even sophisticated tools, like Netflix Chaos Monkey, that help automate the random failure of resources in production. Adopt this kind of monkey at your own risk.

How Does This Connect with Monitoring?

Obviously, monitoring connects to all of this: particularly to the details of how mature monitoring products let on-call operators visualize system and component states, drill down into deep IT stacks, triage and determine root causes of issues. What's also clearly central, however, is how mature enterprise monitoring platforms let you structure notifications and integrate with systems that accelerate effective on-call response.

Sophisticated notification capabilities should be built in (or easy to bolt on). Fully mature monitoring platforms should have notification capabilities adequate for most needs, and the ability to easily integrate with purpose-built notification platforms to gain support for extra features, conditional escalation, etc. Designated users should be able to prevent escalation of non-critical events and keep from being pestered by repeated alerts when repairs are underway.

Integration with preferred or corporate standard ITOM, ticketing, and other backbone workflow support systems should be simple and powerful. Your monitoring solution should be able to connect with back-end systems, SaaS services, and other tools to help on-call folks see what's wrong, collaborate to fix it, and resolve the issue, all from within familiar environments and using familiar tools.