Be Your Company's IT Hero: Move Fast - Don't Break Things

Speed gives advantage, so being an IT Hero has lots to do with acceleration. Help your company get to market first with a novel solution, and you can capture consumer, media, and funding mindshare. Develop a reputation for delivering new features quickly, and you can stop worrying so much about customer churn. Learn to fix outages in 1/10th the time (or in no time at all), and you can save millions, protect your organization against existential risks, and set yourselves up to *move even faster*.

This series of solution briefs will offer perspectives on how IT Heroes can help their organizations move faster (while also avoiding breaking things and making more right, and fewer wrong moves). Where the technical rubber meets the road, however, much of what we'll talk about comes down to automation: *using software to develop, deploy/scale, monitor, and repair software better.*

Eliminating manual toil using tools like Ansible, Puppet, Chef, or Terraform is one of the best resource investments prospective IT Hero/ines and their teams can make, for two reasons.

First, IT automation efforts almost always pay off in (often impressive) savings and gains. Moreover, these gains tend to be linear: any automation is good, more is almost always better. While snarky notions of "diminishing returns," "automation as distraction," and "the Sorceror's Apprentice" (see note about 'FFFFFF it deleted everything!') perpetually haunt automation efforts, in practice, these problems rarely materialize. The fact that underlying technical dependencies are always changing encourages creation of practical automation solutions that are functional without being overly elaborate (and see below about "immutability"). And the fact that task scales are always increasing means that eventual payoffs for any given automation effort are virtually assured.

In fact, unless your organization is already very disciplined about doing the right stuff, investing in automation may be a surer bet than delivering new features or services. (See metrics cited by Forrester, in Harvard Business Review, and elsewhere, suggesting that only one-third of product features deliver upsides.)

Second, automation helps you consolidate gains made while sprinting, and avoid the pitfalls of moving faster. IT cultures that automate by default end up preserving configurational arcana in centralized deployment codebases ("infrastructure as code"), rather than in separate docs, "golden images," or actual deployments. Having this "single source of actionable truth" makes deployments and other processes repeatable, optimizable, extensible – adding speed, creating de-facto organizational standards, and eliminating production-side manual ad-hockery (hackery?) and attendant risks (i.e., the phenomenon of "trying to fix the engines while the plane is in the air").

Instead, automation-forward IT organizations tend to abandon (read: forbid) hackery completely in favor of principles like "immutability," where manual changes are never made on infrastructure (except occasionally, on sequestered dev systems, as part of research), all tweaks are made in config/automation repos, and components are redeployed via automation when they need updates or malfunction – a highly-functional, testable, brute-force approach to dependency management and whole-system stability. Yes, please.

What's in it for the IT Hero?

For starters, IT automation is fascinating. Whether you come at the work from a 'Dev' or an 'Ops' perspective, working with deployment tools will provide new insights into security, databases, scope and program structure, parameterization, and other details; teach powerful lessons about how to build apps that are more automation-friendly, scalable, and self-maintaining; and give you new superpowers to help align dev, test, and prod.

PUBLIC © 2022 OPSVIEW.

> Learn to fix outages in 1/10th the time (or in no time at all), and you can save millions, protect your organization against external risks, and set yourselves up to move even faster.

Teamwise, working together on automation gives everyone a hugely-improved understanding of how your whole IT stack comes together, and helps everyone climb the value-chain to make more and more strategic contributions. Senior engineering types get to see farther than ever, and have new tools for dealing with architectural and cost/benefit mechanics affecting your whole business' viability (e.g., piloting your global shift towards cloud with greater insight and more degrees of freedom). More junior folks get to work on components of the global plan -- making contributions that are long-term-valuable (i.e., reusable automation elements) while building skills and awareness.

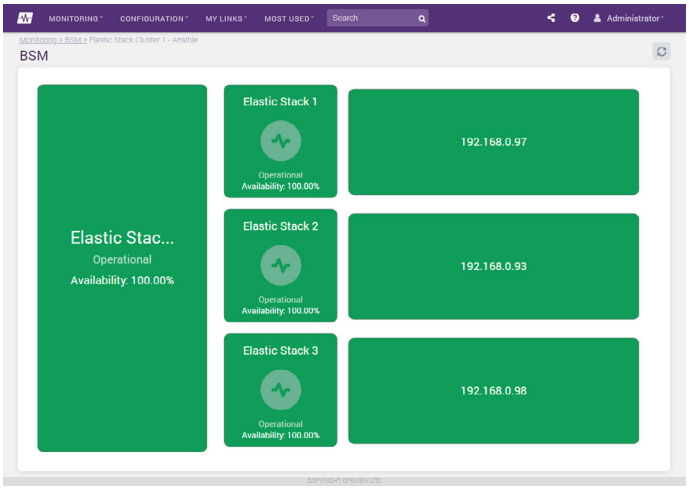
Flexibility is an additional bonus. As we discuss in companion IT Hero briefs, being agile can mean volunteering for more intense collaboration (e.g., pair programming), role exchanges, and other new behaviors – work that's geometrically less difficult if there's a functional single-source-of-truth that documents overall architecture and componentry in the form of what are, basically, human-readable recipes.

Maybe most important, benefits of automation are usually pretty quantifiable and readily-accepted, even by non-technical managers. Manual ops takes huge amounts of (usually) highly-compensated time. Of course, if your management doesn't get it, automation skills are 100% transportable, and highly-valued everywhere.

How does this Connect with Monitoring?

Automation turns monitoring from a cranky, complicated, error-prone post-deployment chore into something that happens magically when solutions are implemented. Tools like Opsview Monitor's integration with Ansible provide a library of functions that dovetail right into virtual infrastructure deployment, letting you add hosts, host groups, define clusters, assemble business service models, etc., then tear it all down again when things (inevitably) change. Handling monitoring this way – ideally all the way from dev to test to production – lets you engineer highly performant, use-case-customized monitoring harnesses that provide the metrics you need to keep solutions happy and fix problems when they arise.

Monitoring can also work to trigger a wide variety of automated mitigations. These can be relatively sophisticated. For example, evidence of a hardware problem on a Kubernetes node might trigger node evacuation, teardown, and redeployment (Kubernetes provides nice features to migrate long-running workloads or simply deploy new instances of stateless workloads in available capacity, so this is often easier than it looks). But even process-level automation (hands off the infrastructure) can pay off enormously. For example, automatically resolving fixed incidents (through bi-directional integration of monitoring with ticketing or other IT Ops Management solutions) might save twenty minutes per incident. Doing so over 50 incidents per day, given average IT staff compensation of $50 USD/hour, could save over $300K in a year.



Ansible was used to deploy, monitor, and create this BSM model of an Elastic Stack cluster, all in one pass.

FIND OUT MORE

USA: +1 866 662 4160

sales@opsview.com

EMEA: +44 1183 242 100