Whitepaper

Let's Re-Run the Runbook!

The resurgence of an essential IT automation practice for the modern DevOps SRE

Jason English

Principal Analyst, Intellyx

August 2020

Why replay this story now?

If you are a grizzled veteran of the pre-Internet-age IT world, when you think of runbooks, you might picture binders full of instructions for mainframe reboots and system workarounds sitting in the server room, right next to the night Ops manager's TV dinner and 13" TV playing a re-run of *The Carol Burnett Show.*

Fast-forward to today, where complex n-tier applications are defined by developers with Infrastructure-as-Code (IaC) automation scripts such as Ansible, Chef and Terraform, then delivered in hybrid IT environments that can span on-premises servers, managed infrastructure and cloud-based services with ephemeral containers and microservices.

A lot has changed since then, especially our ability to deliver more application releases to production, faster. The DevOps movement led to ever-higher levels of business agility and automation, including the advent of the SRE (Site Reliability Engineer, or Service Reliability Engineer) role.

After dutifully driving automation into every aspect of testing, deployment, monitoring and IT service management, SREs and Ops teams are still getting their pagers blown up every day.

What happened to the business continuity we expected from a modern infrastructure, and why are we still getting hammered with urgent tickets? The drive for always-on applications has never been higher.

We need to equip both our expert SREs and new "you build it, you run it" DevOps teams with purposeful control over any our applications or platforms at a moment's notice. In today's post COVID-19 world, we are all too aware that needed

subject matter experts may be unavailable. Therefore, we have to equip others with the power to fill in during an incident or other critical activity.

Runbook automation is a channel to IT value worth watching right now. We can reinvent the valuable knowledge and processes locked in runbooks, with renewed intelligence to help manage all the IT work required for successful operations.

Hell, HBO Max paid $425 Million to stream re-runs of the 90s sitcom *Friends*. What's old shall be born anew.

Let's re-run the runbook!

Challenges: How did we get into this spinoff?

"There's 57 channels and nothin' on." - Bruce Springsteen

Was there a point in time when companies lost interest in runbooks, and flipped the channel? Not exactly. There has always been a need for skilled Ops teams to develop and execute standard operating procedures.

The runbook never went away for traditional enterprise ops teams, but the discussion of CIOs and IT thought leaders moved elsewhere as Digital Transformations heated up and we built an obsession with 'deploy, deploy, deploy.'

Developer-led Agile methodology, then the increased use of CI/CD pipelines and IaC definitions, and then the DevOps movement, shifted the conversation toward the automated, self-service provisioning of development environments, and moving to ever-faster releases and frictionless deployments.

Vendors sold enterprises 'DevOps platforms' -- tool chains for master architects to work in conjunction with software to specify and govern 'golden state' builds and dev/test environments for the company, storing them in a blessed CMDB master file or application delivery system. The advent of cloud computing only accelerated the rate of automation with elastic scaling.

If the next release falls short of customer demand, developers react, readjust, redeploy in the cloud -- then take feedback, fail faster, and release faster -- in smaller increments that contribute to the desired end state. There's a lot of

innovative goodness in this approach, until the moment the DevOps team forgets that a whole lot of 'Ops' happens after deployment.

It was only a matter of time until some vendors started touting the concept of 'NoOps' and an infrastructure approach that could heal itself, without the need for IT Ops requests and trouble tickets. After all, developers wanted to invent new things, not muck with infrastructure.

Everyone held out Google and Netflix as the paradigm-shifting leaders for investing in such a developer-led, DevOps transition ahead of their competitors, and releasing to production a thousand times faster than their peers.

Major companies attempted to follow in their footsteps, while migrating their critical apps to cloud and Hybrid IT infrastructures. Production issues arose in higher frequency, and took too long to resolve. Some companies turned back to their old, slower ways.

The failed early projects failed to account for how category leaders like Google and Netflix invested just as heavily in operational resiliency, arming SRE and Ops teams with more comprehensive observability, issue reporting and feedback mechanisms, and, of course, **runbook automation.**

What's programmed for Day 2?

> *"There is nothing wrong with your television set. Do not attempt to adjust the picture... We will control the horizontal. We will control the vertical. We can roll the image, make it flutter...." - The Outer Limits intro (TV series, 1963-1965)*

No matter how much automation you apply, speed and complexity always create operational challenges. We can now embed millions of automated tests for code and data integrity in the software lifecycle, certify each build and integration, run compliance and security checks, and monitor the deployment for security and performance reasons.

Alerts are set to roll out of deployment and CM tools, from functional and performance monitoring tools, from commercial observability and event logging platforms, which roll into issues in wikis, support, SIEM and ITSM ticketing tools.

What happens on Day 2 -- and every succeeding day after release? Operations teams, still charged with keeping everything running, are overwhelmed by incidents from so many sources.

But it's not the quantity of issues that is the killer here, it's the specialized knowledge required to resolve each issue. Human expertise becomes very siloed in a complex application environment, as each team is incentivized to improve a different metric, whether delivery speed, risk avoidance, cost efficiency or response time.

Even if there were any 'jack-of-all-trades' in IT Ops who could grasp all of the dependencies in production and respond appropriately to a failure, chances are they are already overworked or unavailable. Time to rally all the experts in a call or meeting and dogpile on the problem!

Take for instance a retailer *that is starting to notice an increasing rate of cart abandonment on their site. The abandonment has become serious enough to classify as an S1 issue. For the last two weeks there have been a dozen S2 or S3 issues building up that may have been impacting shoppers, but they just weren't enough of a priority for Ops teams to start digging through scripts for uncertain clues.*

An SRE leader drops into this incident trying to find the remedy for a 'death by a thousand cuts.' She may need to send out a page to everybody who could be involved in the resolution process: get a DBA who is authorized to reboot the database service, request an authorization for two more cloud environments for release and rollback, get a SecOps pro who can run a check of UEBA data to see if there's fishy activity on certain accounts, get an authorization from change control to apply the latest patch to the application server, and so on.

Hopefully none of these essential people are out sick, or busy on something else, as it looks like the resolution will require a very specific sequence of actions. Perhaps they might all need to work simultaneously in their own domains, exacerbating the incident with unanticipated constraints or timing conflicts.

This all-hands incident management approach on Day 2 seems destined to cause employee dissatisfaction and inhibit the company's ability to move forward with future functionality.

That's a shame, because faster incremental releases, self-empowered teams and shared responsibility were the whole reason for trying DevOps in the first place.

Where to start: It's all about the ratings

> *"Ratings have changed, viewer habits have changed and the options for the audience have grown enormously, but I don't think how you tell a story is fundamentally different." – J.J. Abrams*

To get value out of a DevOps transformation, we need to equalize the impact of the SRE both before and after release, rather than treating production support and issue resolution as an afterthought.

We need to ***empower the human operator*** with runbook automation.

The easiest way to start would be deciding on KPIs for improvement through runbook automation, rating your own company's performance, and setting goals. Here's a few suggested measures.

- **MTTR or TTR (Mean time to resolution / time to ticket resolution).** This measure is the granddaddy of all support metrics, as less time spent resolving each issue generally means more effective automation, with fewer complications encountered and lower costs. Organizations can naturally respond faster and take much quicker action with runbook automation, reducing ticket resolution time significantly.

- **Reduced outage time.** Runbooks can also make a big impact on the highly quantifiable cost of system outage time, but like MTTR this metric also has many correlating factors.
- **Reduced escalations/pages.** A trouble ticket can be passed around between teams and escalated to multiple responders. Empowered Ops teams with automation always strive to bring critical pages to overworked staff and firefights under control. Runbook automation can be used to empower lower-level teams and avoid severe escalations.
- **Faster issue acceptance.** Collaboration, metrics, service and support management tools all contribute to this metric, but strong runbook automation can drastically improve the SRE's contact time on an issue by automatically routing tasks to the correct systems or human handlers and managing self-service administrative access and approvals.
- **Transparency**. Like a good supply chain, every IT change or support job should have a documented chain of custody from source to ship built in. Where did the definition for that automated runbook come from, who ran it, with exactly what tools or artifacts, and what actually happened?
- **Risk reduction.** Security and performance platforms may address vulnerabilities, find failure conditions and recommend responses, but in the end, the prescribed course of action often means a complex issue resolution process must be undertaken. Effective runbook automation accelerates remediation time and reduces risk exposure.

You might notice a common thread in all of these ratings. Runbook automation is inexorably intertwined with development automation, collaboration, security, monitoring and support management tools.

Making IT changes is never as simple as moving the bits from System A to System B. Companies invested heavily in automation across all of these platforms, and in the experts to operate them.

Runbook automation maximizes the return on investment for existing automation, rather than replacing it. Skilled teams can 'set and forget' responses for known

issues, then focus their attention on remediating and capturing responses to higher severity issues.

What to re-run: Bring back business continuity

> *"We are going to shorten the incident management lifecycle while delivering features, fixes and updates more frequently, in close alignment with operational objectives."*
> *-- Bhavik Gudka, Director of Software Engineering, Capital One, DOES 2019 event*

Incident management has never been the sexiest part of digital transformation. But if your production infrastructure is running at the scale of a leading global bank, business continuity represents a defining competitive advantage -- it's not the kind of capability you can simply outsource.

Capital One's engineering team focused their own runbook automation design on pinpointing every instance of an escalation, then increasing the speed and quality of automatic runbook responses to each incident to bring that count down.

The firm has invested thousands of employee hours building this capability, and other open source DevOps utilities. Their effort pays off not just in the resiliency of their systems, but with better employee job satisfaction, improved recruiting and the increased retention of valuable technical resources.

For the rest of us who can't afford such a development commitment, how can we apply these lessons to evolve our own runbook automation?

- **Leverage existing skillsets and tools.** Don't make teams keep learning new automation languages and tools for their own sake. First, tie all of the existing assets and scripts together with self-service. Then, connect the event stream of runbook jobs with the company's collaboration tools and service management platforms of choice.
- **Relentlessly prioritize self-service.** Make it as easy as possible for SREs and necessary contributors to log in and get permissions to either run jobs or review results. Authorizations may vary by tool, but role-based and user-

specific access policies should be handled across the incident management toolchain to avoid authentication hang-ups.

- **Maintain context across automation stacks.** Most organizations don't suffer from a lack of automation. They usually have lots of Ansible and Puppet scripts, Chef configs or Jenkins builds spinning away somewhere in the stack, alongside multiple commercial tools. The problem is that very few people know how to invoke all of this automation in the right way, in the right sequence, and at the right time. Well-designed runbook automation maintains the context of a job, monitoring results as the jobs cross all of these stacks.
- **Take feedback and document learnings.** Most importantly, we need to consistently record and document what works, and what doesn't work. Observability or faster TTR is only useful for one issue resolution cycle if we don't capture the whole process, any unexpected aspects and the outcomes as better automated runbooks.

No matter how runbook automation is improved, there will always be a need for SREs and human stakeholders to provide oversight and intervention. By automating resolutions to known or expected issues, human operators are empowered to contribute their best attention to higher-risk or unknown issues that are more likely to impact business.

The season finale: A successful run

> *"Now that our runbook is defined, and in version control, and can be audited, we can pass that compliance requirement. It's a chain of custody, you know what the scripts are going to do every time, and what their provenance is.*
>
> *-- Shaun Norris, Head of Cloud Infrastructure Services, Standard Chartered at DevOps Enterprise Summit 2018*

The hallmarks of a successful runbook automation transformation? A continual

improvement in delivery speed, as well as increased system resiliency, security and performance once critical software is in deployment.

The Standard Chartered web team started out using the Rundeck open source tool to automate runbooks for service restarts and failovers, but within a year their runbook automation platform had caught on elsewhere in the enterprise.

The massive financial institution was executing 350,000 runbook jobs across more than 400 applications, reducing their TTR by 25 minutes per incident. For a company that size, that time savings added up to an estimated 28 people-years of work in 2019.

Incident management for the SRE and IT Ops team doesn't need to be a series of interruptions, with multiple people pushing requests, approvals and interdependent tasks at each other.

Instead of relying on manual documentation, wikis and logs, and multiple caches of issue resolution procedures, runbook automation needs to pull from all the sources of truth for configurations and metrics, pushing to all systems of execution and record in the enterprise.

As most enterprises have a heterogeneous mix of scripts, service management tools and hybrid deployment environments in play, providing secure access guardrails and maintaining the context of an automated runbook across all of these systems becomes essential.

Partner companies and technology suppliers can even be brought into the resiliency game with the same fine-grained control over access to automated runbooks, instead of always jumping on a multi-party bridge call when a multi-enterprise response is needed.

The results? Fewer tickets, fewer escalations, and fewer expensive and highly skilled people involved. If people do get interrupted by escalations, it is for much shorter periods of time. That means happier employees, with less overtime and frustration. And better customer experience in production, where it counts most.

Signoff: The Intellyx Take

The way we consume television shows has changed drastically. Forty years ago, viewers waited for re-runs to appear over the airwaves on one of a handful of channels, and now we are binge-watching entire series through on-demand online services.

The way we ensure the resiliency of our applications has transformed even more drastically in today's DevOps and digital-native world, where complex environments can change in moments.

While some application failures will always be inevitable, long and disruptive response times aren't. Disciplined runbook automation usage captures operational knowledge, bringing incident management practices beyond the black-box era not that dissimilar to how we used to see the 'Technical Difficulty, Please Stand By' program interruption screen appear on our television screens.

Let's re-run the runbook as a first-class automation citizen of the DevOps lifecycle -- for self-service reusability and improved resiliency across the IT estate.

About the Author

Jason "JE" English is Principal Analyst and CMO at **Intellyx**, a boutique analyst firm covering digital transformation. His writing is focused on how agile collaboration between customers, partners and employees can accelerate innovation.

He led marketing efforts for the development, testing and virtualization software company ITKO, from its bootstrap startup days, through a successful acquisition by CA in 2011. JE co-authored the book Service Virtualization: Reality is Overrated to capture the then-novel practice of test environment simulation for Agile development, and more than 60 thousand copies are in circulation today.