RUNDECK

7 Pitfalls of Incident Management

An illustrated guide to shorter incidents and fewer escalations.

Featuring solutions leveraging runbook automation

DAMON EDWARDS

Failure is inevitable. Systems break. Mistakes are made. We all do our best to avoid incidents, but incidents are going to happen.

How quickly and effectively we can diagnose the issue and repair the service is the measure of our operational capabilities.

Let's explore seven painful anti-patterns that can get in the way of you and your colleagues' response to incidents.

1. "I Could Fix It, If I Could Get To It"
2. "The Dogpile"
3. "I'm An Expert, I Don't Check the Wiki."
4. “Do it. Do it again. Then do it again.”
5. "Known Workaround. Bug closed."
6. “Once We Replace Our Current Automation...”
7. “Incident! Page everyone.”

Our goal: Shorter incidents. Fewer escalations

This book is a collection of lessons gathered by our fellow operations community. Feel free to reach out to us at anytime at hello@rundeck.com.

01

I Could Fix It, If I Could Get To It

⚠️The Pain⚠️

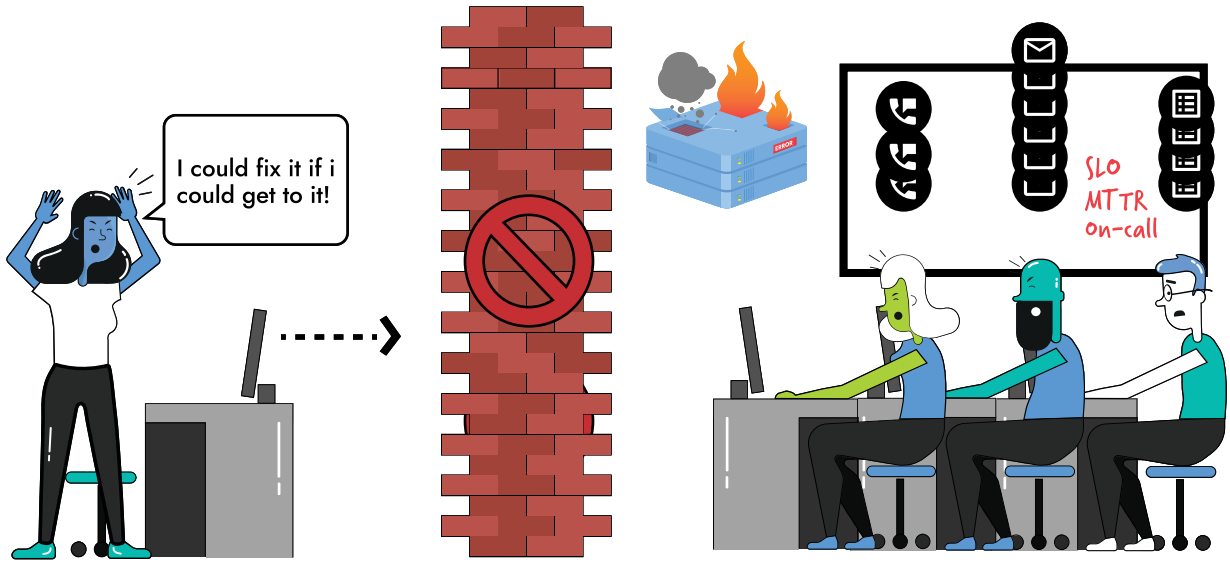
Business: Longer incidents. Limited capacity for key experts (schedule slippage)

Team: More waiting. More interruptions. Bottlenecks.

Engineers: Frustration.

Disjointed access is a problem endemic to enterprise operations. Policy or politics block people responding to an incident from taking action in production environments — even though if they have the necessary technical knowledge and experience.

Sometimes, security or compliance concerns are what get in the way. Other times the lack of access is just a byproduct of a siloed organization or political turf concerns.



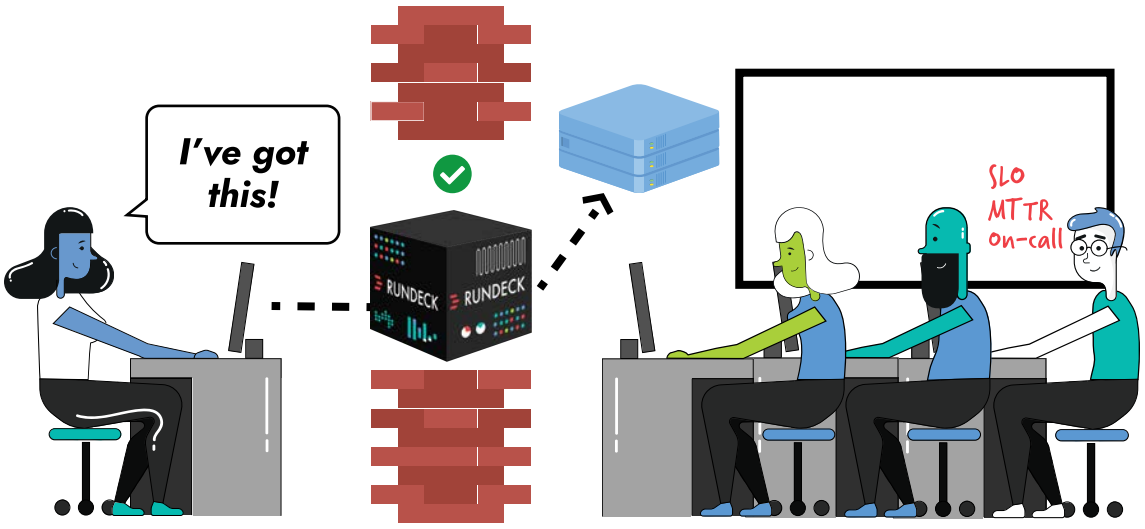
Access issues can undermine response efforts

In either case, the problem is the same: those who have the immediate context of the problem that needs to be solved aren't able to take action.

They end up opening tickets or try to find by IM or phone a colleague who can help them. The result? More delays and escalations. The incident drags on. There are more handoffs and opportunities for errors. Disruptive escalations become the routine process rather than the exception.

Also, you are unable to distribute work evenly in your organization. Instead, you have bottlenecks and queues around your overworked subject matter experts

Instead, take a self-service approach using automated runbooks. Safely give anyone responding to incidents the access they need to take action quickly and effectively.



Safe and compliant self-service access to any environment

Using Rundeck to overcome disconnected access safely

1. Identify the most common procedures where improved access would speed up incident resolution or decrease escalations (restarts, diagnostics, fetching logs, rollbacks, failover, etc.)

2. Have subject matter experts collaborate on defining the runbook for those procedures.

3. Use Rundeck to configure the automated workflow across your existing tools, scripts, API calls, and system commands.

4. Use Rundeck's access control policies to hand out fine-grain access to run the right procedures in the right environments.

5. Show your security team and auditors Rundeck's access control and logging features. Demonstrate how, with Rundeck, you can put operational capabilities into the hands of a broader team while improving your organization's security and compliance posture. Everybody wins.

Rundeck's access control capabilities are helpful in cases where the experts who are defining the response procedures — and taking action when problems arise — are outside of the traditional operations team (like developers in a "you build it, you run it" org model).

02

The Dogpile

⚠️The Pain⚠️

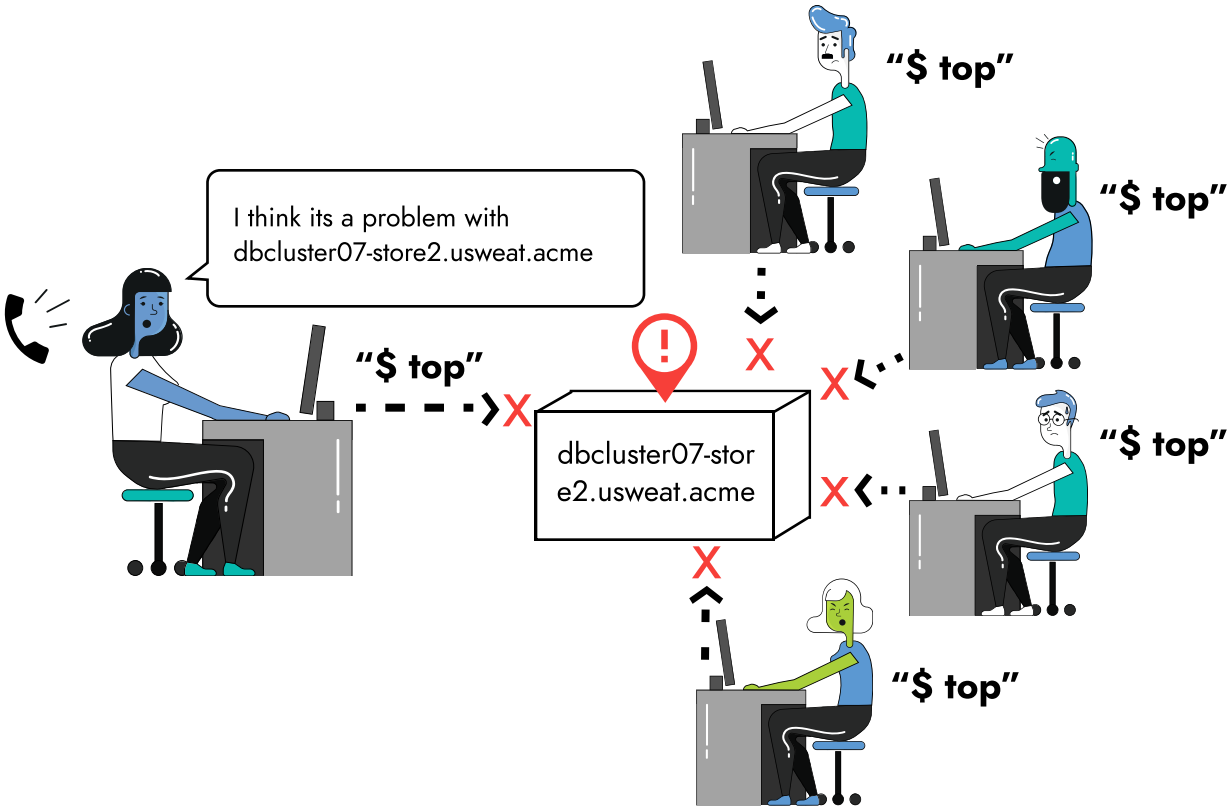
Business: Longer incidents.

Team: Slower incident resolution. Inefficient use of resources.

Engineers: Frustration. “Spinning your wheels”.

Multi-party conference bridges or chat rooms are a routine part of major (and often minor) incidents. This well-intentioned idea hopes that casting the broadest possible net will ensure the correct knowledge and skills will be available to resolve an issue quickly.

Unfortunately, multiple eager people all interacting with the same systems, at the same time, often results in unintended collisions, interference, and miscommunication.



Collisions often occur during collaborative response to incidents.

This phenomenon is amusingly referred to as ‘the dogpile’ — a close cousin to the "too many cooks in the kitchen" concept.

A familiar example of the dogpile:

1. Someone says they think it might be a problem with server A.
2. You log into server A, run the 'top' command
3. The first ten entries are your other colleagues also running 'top.'

Other, more destructive, dogpile examples:

- Errors due to multiple people thinking they are talking about the same thing when in reality they are each looking at something different

- Extended or worsening outages due to various people making conflicting changes

- Key information missed due to people getting incomplete or partial second-hand details ("What are you seeing? Try this.")

One obvious way to counteract the dogpile effect is to put strict rules into place (e.g., only one person interacts with the system at a time).

However, the success of such rules depends on the understanding, agreement, and discipline of all involved — no small feat under the pressure of an outage.

Smartly applied runbook automation can help improve the efficiency, effectiveness, and safety of groups responding to incidents.

Using Rundeck to collaborate while responding to incidents safely

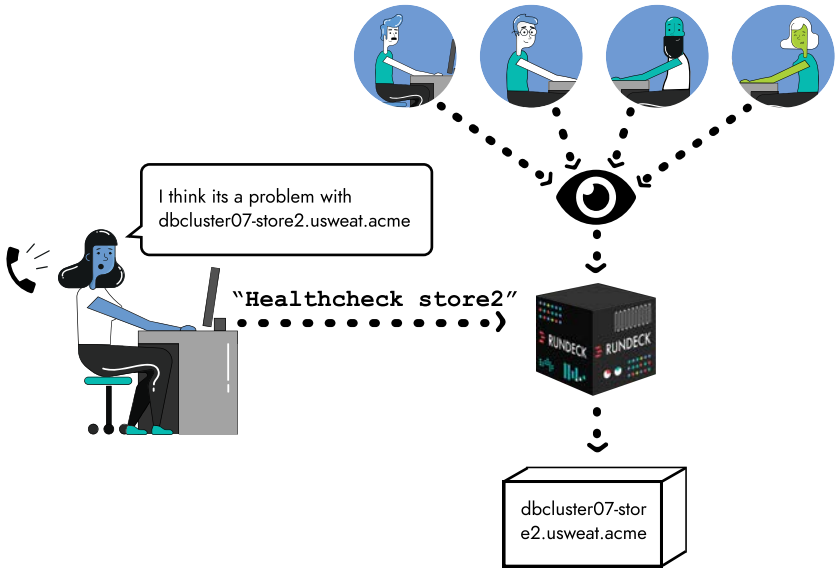
1. Use Rundeck's access control to make it so anyone who would be involved in an incident can see the output of any diagnostic or repair actions taken

2. Create Rundeck jobs for performance checklists and diagnostic procedures. Collaborate with subject matter experts to have diagnostics and performance checklists for each service and each layer your stack.

3. During an incident, send out links to Rundeck job execution (either live output or historical) so that others can follow along (what is running and what the output is)

4. Have key subject matter experts perform any ad-hoc commands through Rundeck's commands capability so that actions are logged and visible

The dogpile may be human nature, but with the right policies and runbook automation in place, you can keep it at bay.



Collaboration during an incident is easier and safer with Rundeck.

03

I'm An Expert, I Don't Check the Wiki.

⚠️The Pain⚠️

Business: Longer incidents.

Team: Self-inflicted incidents or prolonged incidents.

Engineers: Frustration. Uncertainty.

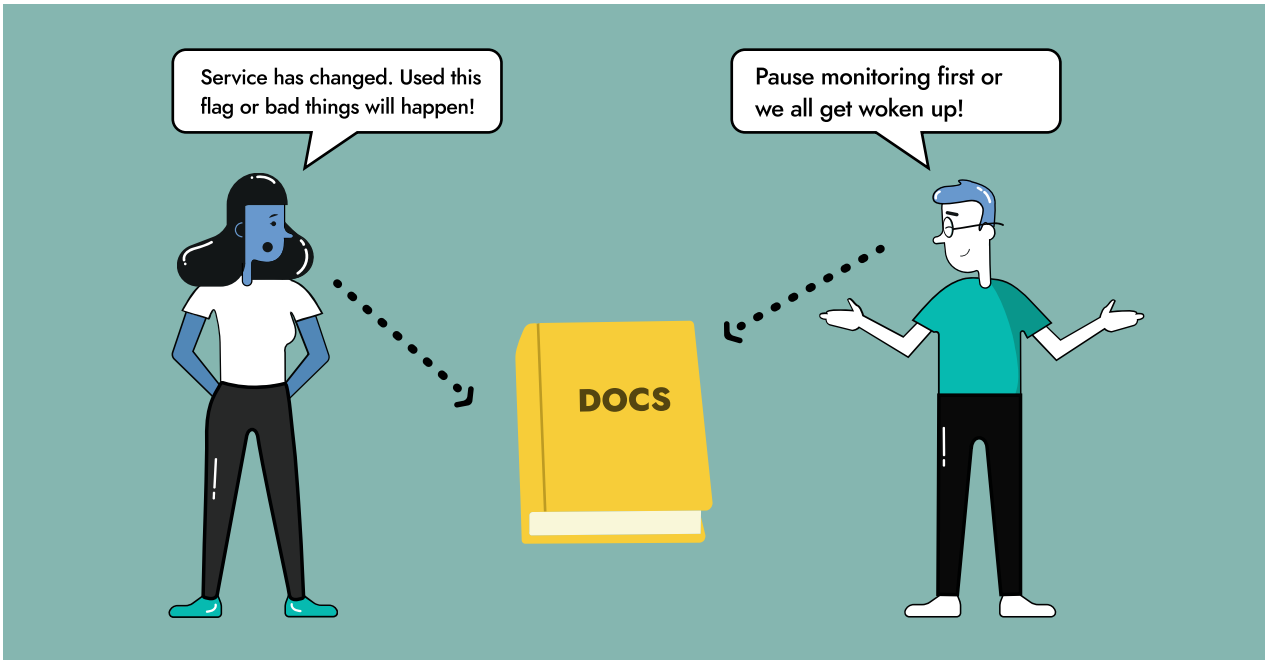
What is the most challenging aspect of written documentation? Getting people to read it!

Change is a constant in enterprise operations. Services, configuration, and underlying infrastructure are continually changing. Procedures are frequently updated. How do you adequately convey those changes, so few mistakes occur?

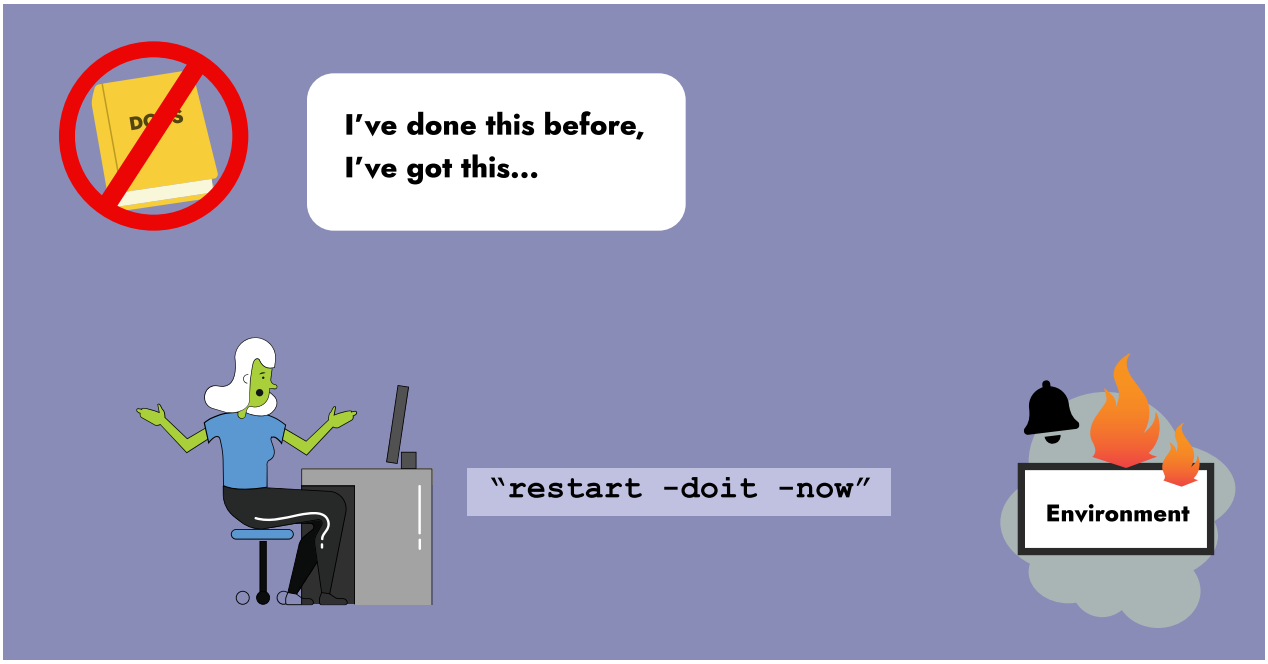
Traditionally, our industry's focus has gone into how to get people to write documentation. The problem of getting people to read that documentation receives far less attention.

How do you get people to stop and see if the procedure has changed or the environment is not what they expected?

Getting people to stop and read is difficult enough during project work, but becomes even less likely during emergencies.



Later...



Getting people to read documentation may be more difficult than getting people to write it.

Further undermining documentation efforts is the classic "relax, I've done this before" syndrome.

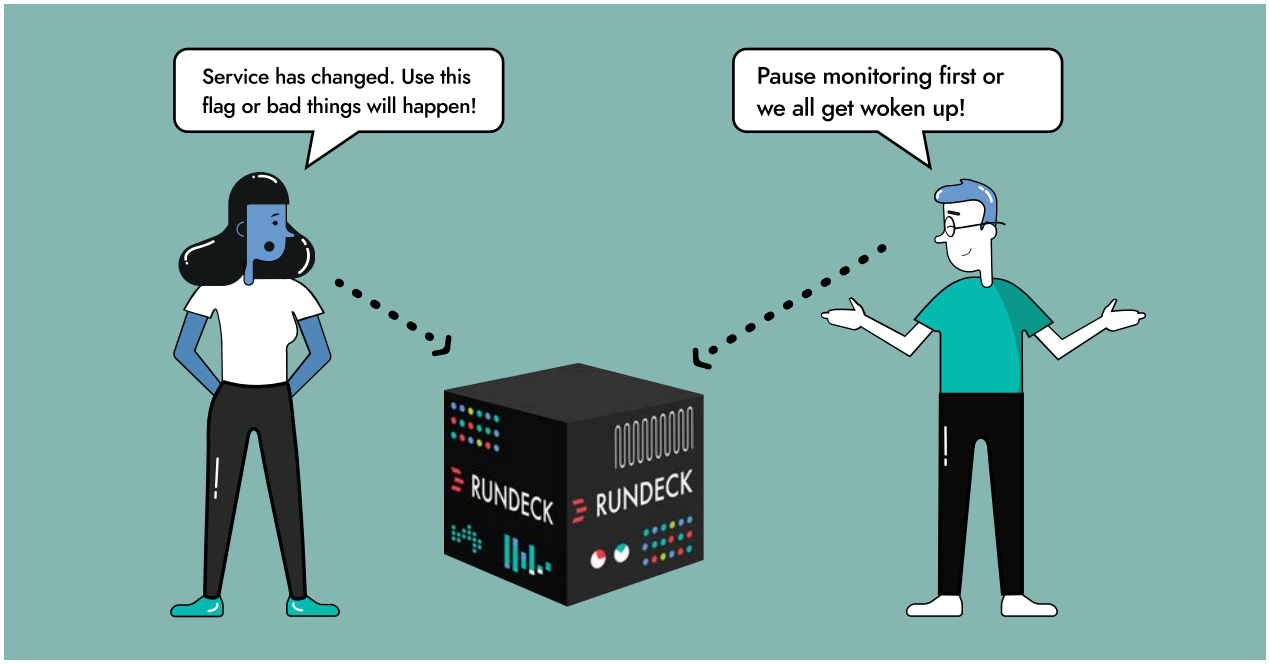
The more times that someone performs a task, the more they believe that they understand what to do and expect underlying conditions and results to be the same. The more routine and seemingly mundane the task, the more likely it is that the person performing the task will approach it with confidence and not feel the need to look for instructions.

It is difficult to expect people to stop, especially in an emergency, and look at the documentation to see if anything has changed (be it a wiki, static site, man pages, or other tools). Of course, this common trait of human nature has led to many self-inflicted and prolonged outages.

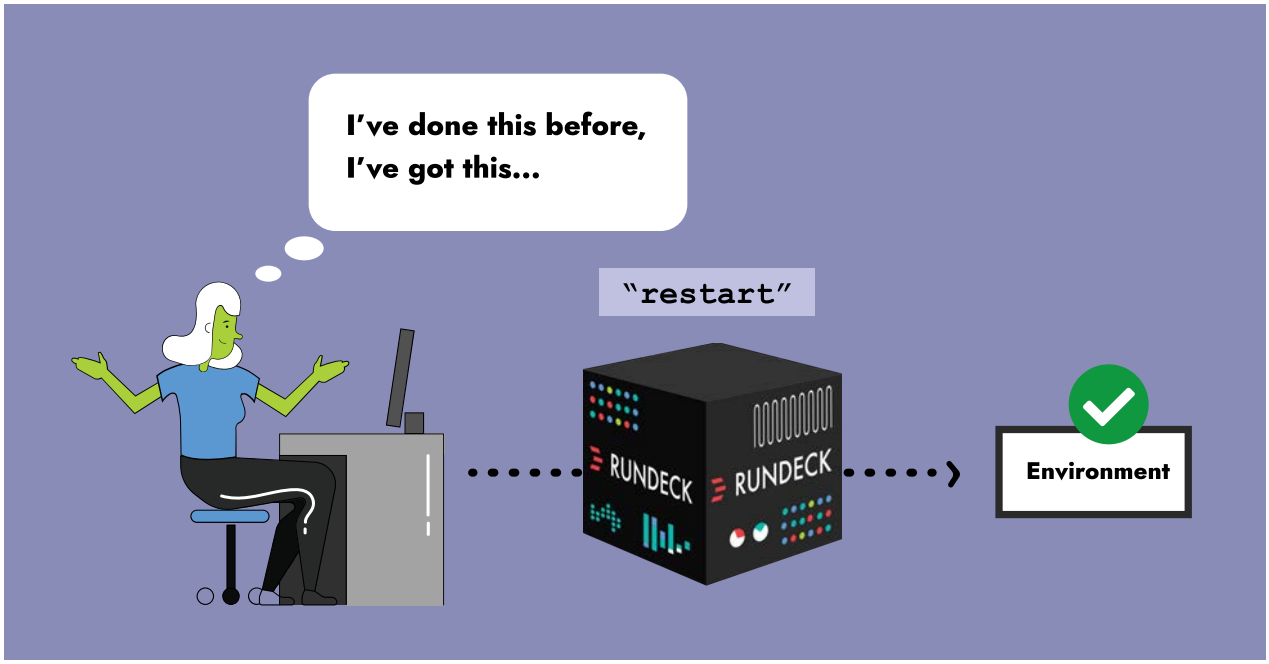
Perhaps, relying on documentation isn't the best way to convey information — especially changes to operations procedures. Email notices or Slack blasts aren't always helpful either, as these messages get lost in the day-to-day communication noise.

Communicating through code and automated procedures is much more effective. Runbook automation provides the tooling necessary to quickly enable subject matter experts to define and update automated procedures in the natural flow of their work.

Now when it is 3 am, and you roll out of your bed to respond to that alert, you won't be asking yourself: "Are these still the correct commands? Am I passing the correct options to these scripts? Has anything changed?"



Later...



With Rundeck your procedures are easy to share and keep up to date

Documentation is essential and absolutely has its place. But, when it comes to communicating operations procedures, do it through code.

Rundeck is the easiest way to get started as you can plug in all of your existing commands, tools, and scripts. The Rundeck platform does the rest of the heavy lifting for you.

Using Rundeck to ensure procedures are up to date and available for everyone's use

1. Define a "standard vocabulary" for everyday actions for all of your services (e.g., start, stop, restart, deploy, rollback, health check, etc.).

2. Use Rundeck jobs to collaboratively define the automation procedures for each action. Leverage what you already have. Call your existing tools, scripts, API calls, and system commands (whatever you would do at the command line).

3. Assign the ownership of keeping those actions up-to-date to the teams who are developing the services (in collaboration with other teams who can contribute operational knowledge).

04

Do it. Do it again. Then do it again.

⚠️The Pain⚠️

Business: Takes too long. Costs too much. Could get more done.

Team: Frequent interruptions. Repetitive Toil. Overloaded experts.

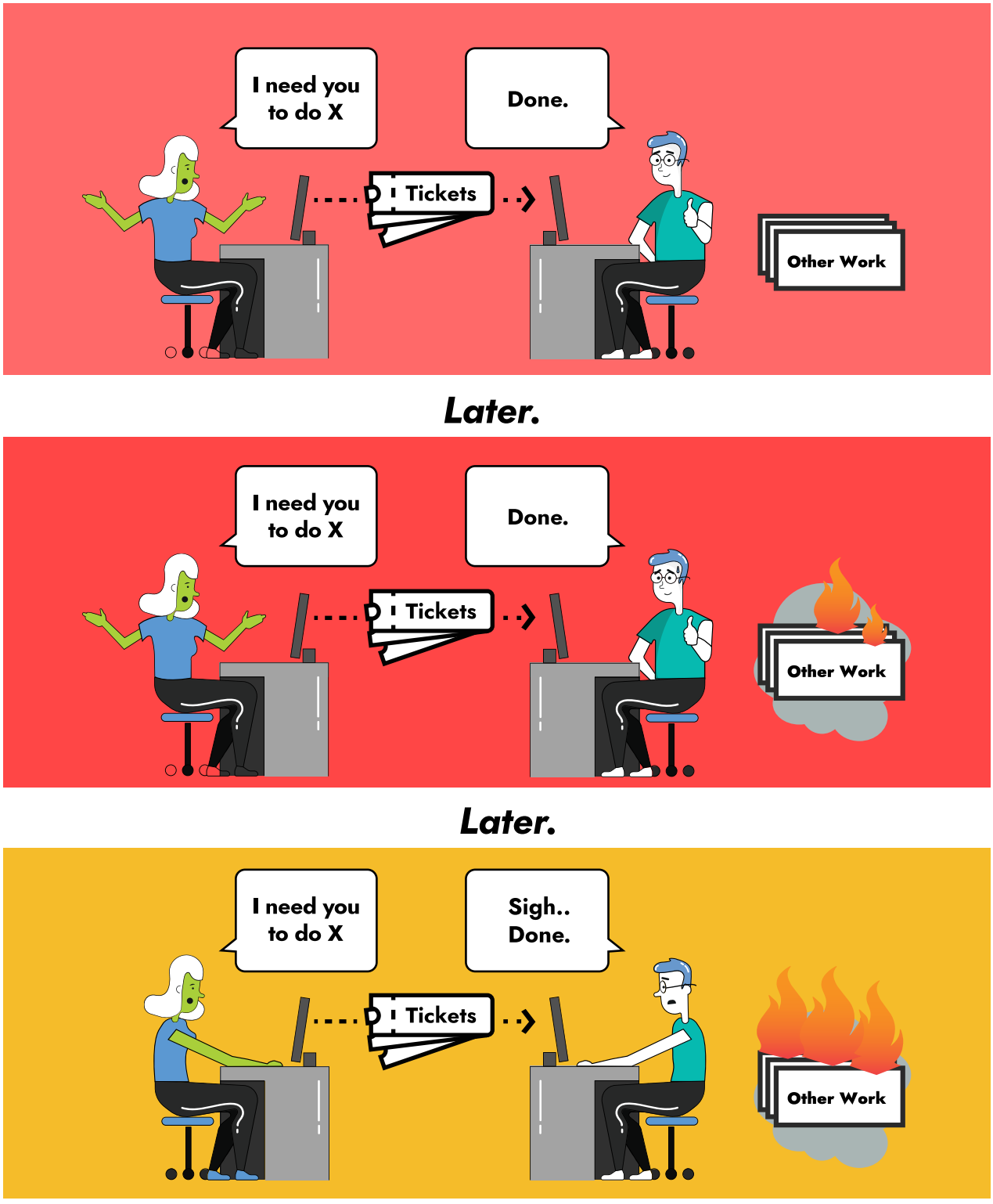
Engineers: Frustration. Time pressure. Less fulfilling work.

How often are you responding to repetitive incidents or short notice requests?

Dealing repeatedly with the same issues is a frustrating fact of life in Operations.

Running diagnostics, restarts, applying known fixes, resetting state, updating configuration, short-notice provisioning — there is never enough budget or perfect planning to mitigate repetitive issues.

Each occurrence adds disruption and interruption that costs your organization time and money. More outage time, more escalations, more time not being spent on other important work. It all adds up



Repetitive requests accumulate to sink productivity and cause delays to other work

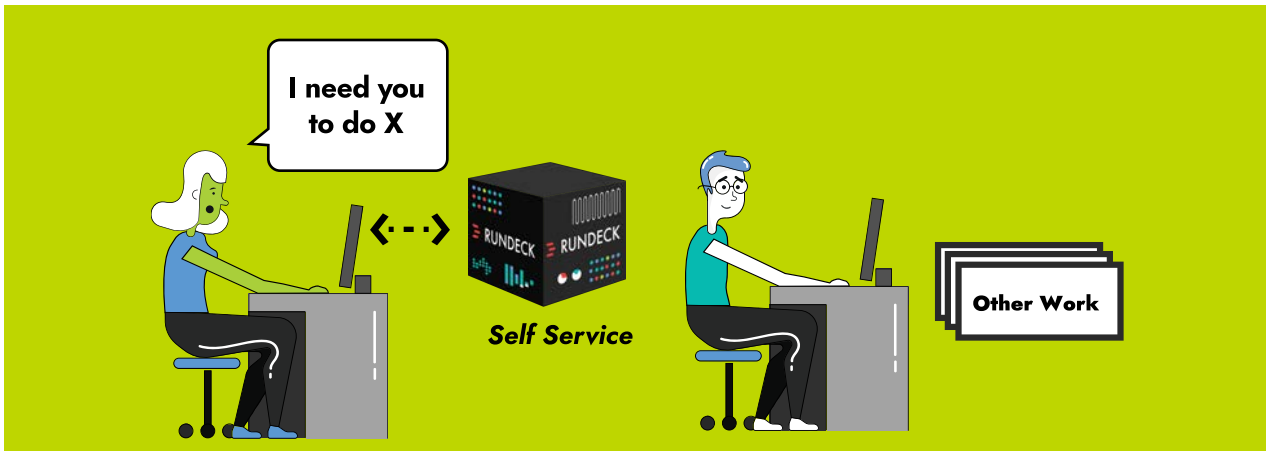
Automating your response to those repeated interruptions provides obvious time savings. Doing so provides standard operating procedures to call upon whenever you need them.

However, it is when you put that automation behind a self-service interface, the benefits grow exponentially. Rather than interrupting you, others can take action themselves.

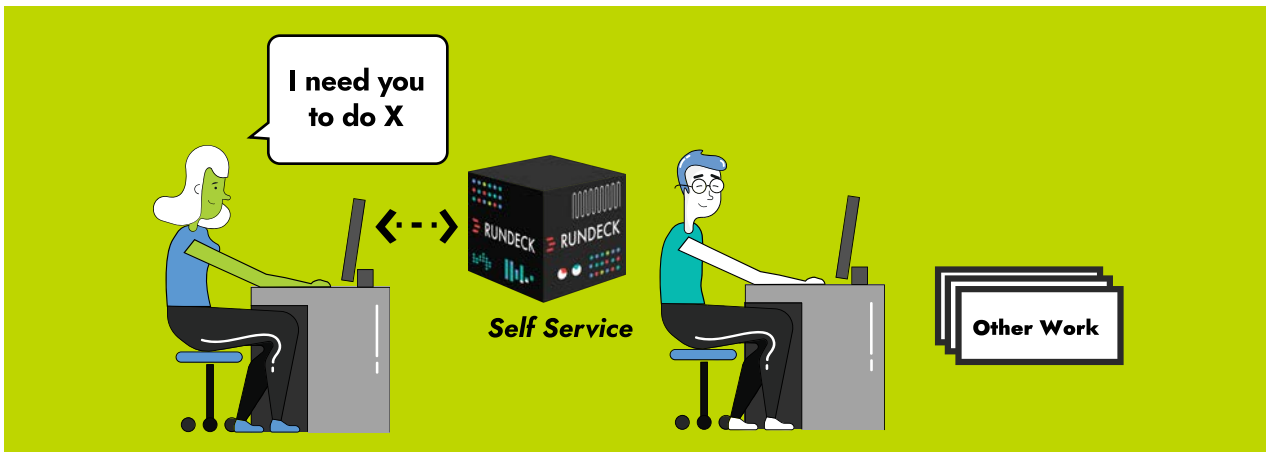
In the case of repetitive incidents, this self-service approach reduces the length of an incident by allowing anyone in your organization to execute diagnostic or repair actions immediately. Shorter incidents. Fewer escalations.

In the case of short-notice requests, self-service spares the subject matter expert — who otherwise would have fielded the request — from interruptions and expensive context switching. Less waiting. Fewer interruptions

Later...



Later...



Help people help themselves and avoid interuptions

Using Rundeck to Reduce the Impact of Repetitive Incidents

1. Get started by analyzing your ticket system to get a list of the most frequent issues/requests. From there, you can evaluate which of those repetitive issues/requests are the most disruptive and which are the easiest to automate.

2. Have the person(s) most familiar with fulfilling those requests collaborate on the runbook — describing the workflow of tools, scripts, APIs, or commands needed to respond.

3. Use Rundeck to automate the workflow and give others access.

One of the primary benefits of using Rundeck for runbook automation is that you can quickly turn your existing scripts, tools, API, and system commands into powerful self-service without having to learn new skills or go through a retooling project. Plug what you've already got into Rundeck and get started today.

05

Known Workaround. Bug closed.

⚠️The Pain⚠️

Business: Diminished capacity. Less gets done.

Team: Interruptions. Toil.

Engineers: Demoralizing. Frustration.

If you've worked in Operations, you know how frustrating it can be to deal with repeated instances of "known issues."

No matter how routine the response is, it still chews up your time with interruptions and toil.

The frustration kicks into high gear when you submit a bug report on the issue — but Dev quickly closes it. The reason? It is usually something budget or priority. And besides, "Ops has a workaround."

Conflict over recurring issues is commonplace

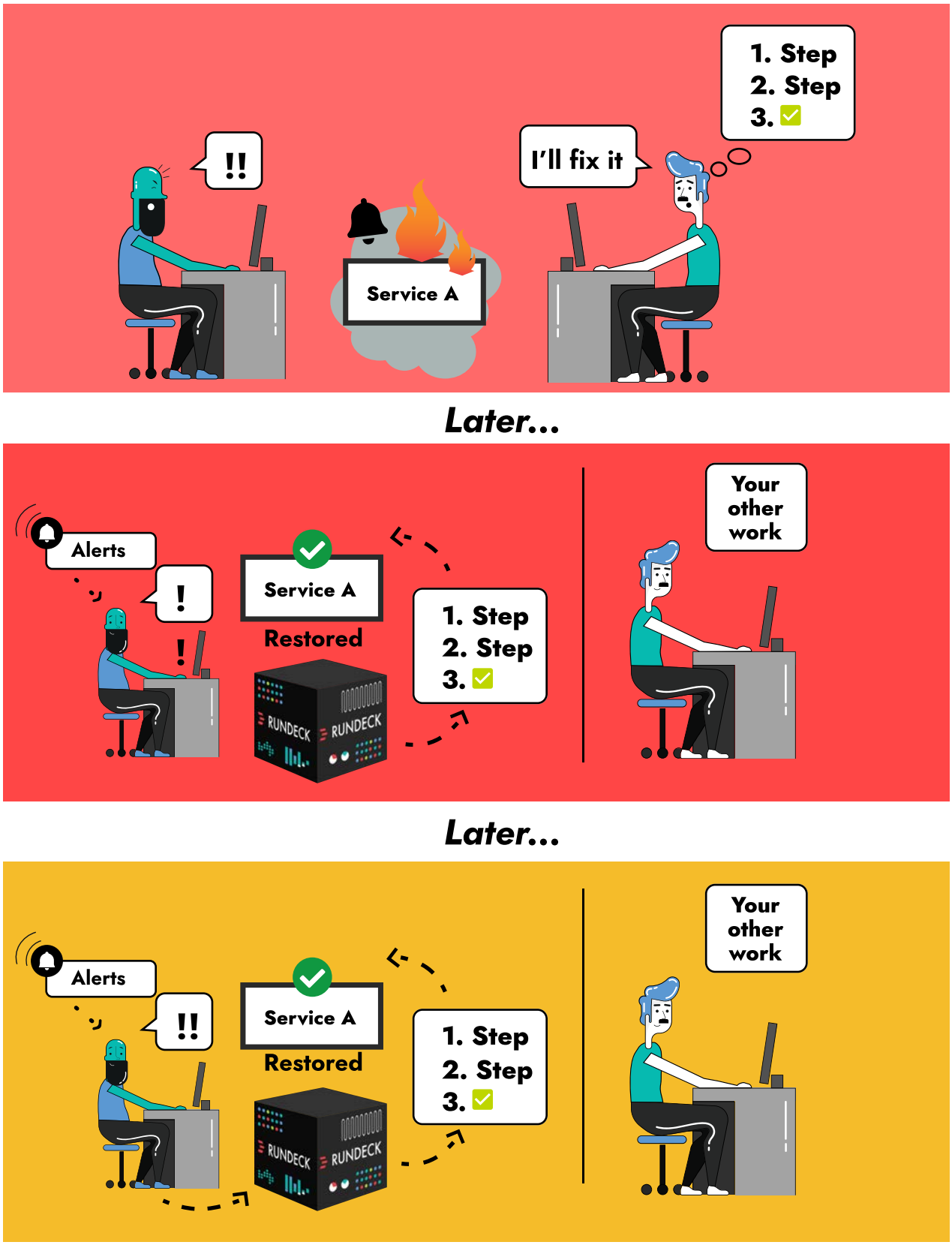
It is just a fact of life that bugs with known operational workarounds will exist (and their permanent fixes will be occasionally deprioritized in favor of other issues or features).

The first (and, yes, the most obvious) way to deal with the known repetitive issues is to understand the impact it has on the org and make the permanent fix.

Fixing a repetitive issue is always the wise first choice.

The next best option is to turn the known workaround into a standard operating procedure that others can use to remediate the issue as it arises. This use of self-service runbook automation doesn't eliminate all of the interruptions and extra ops work. However, through the ability to move the work around the organization, you can mitigate its effects.

Sometimes this means putting more control into the hands of delivery teams. Other times it means giving more control to the NOC or L1 support teams. In different situations, you can set up auto-remediation before alerting.



Use runbook automation to keep the costly interruptions away from your subject matter experts

Using Rundeck to unify the management environments with heterogeneous automation

1. Identify repetitive known error-conditions or failures (prioritize those that occur frequently or where a permanent fix has been deprioritized because there is a workaround)

2. Create two Rundeck jobs: one job to diagnose the known issue, the other to take repair action

3. ***Optional:*** automatically trigger the diagnostic job from your monitoring or incident management tools.

06

Once We Replace Our Current Automation...

⚠️The Pain⚠️

Business: Diminished capacity. Less gets done.

Team: Inefficiency. “Bikeshedding”.

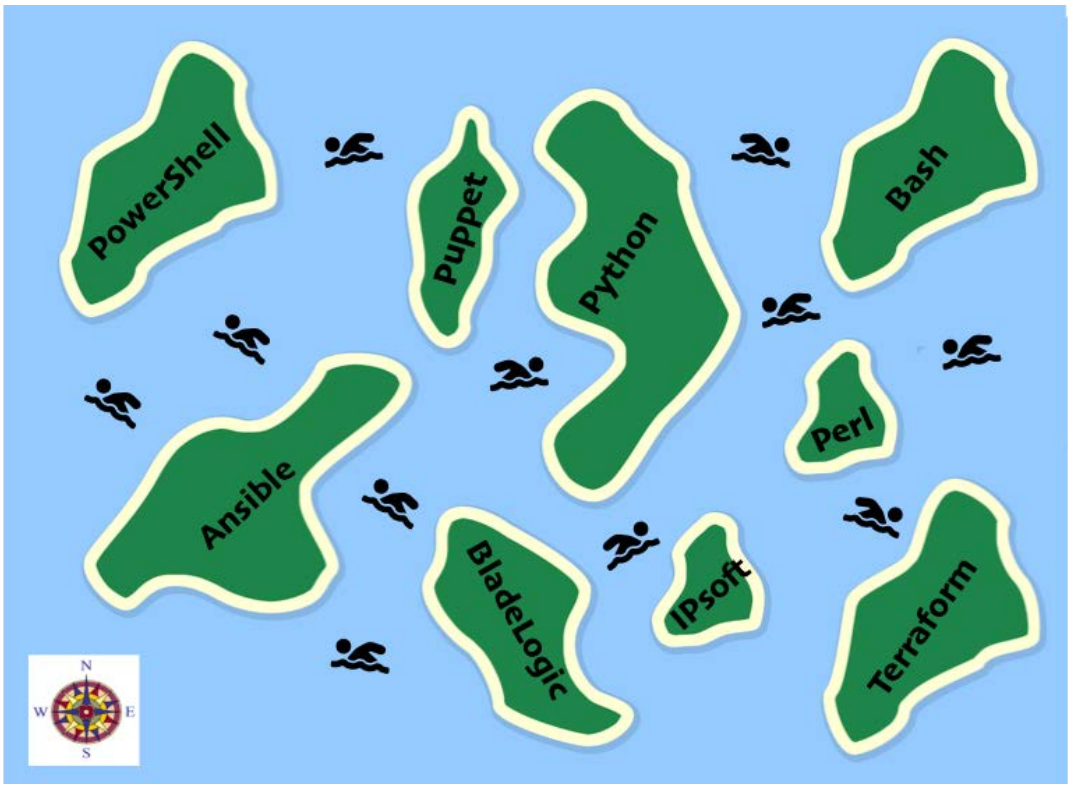
Engineers: Rework. Conflict.

We've all heard something along these lines: "This new automation language is going to be the one that solves our problems. Now, if we can just get everyone to learn it and replace what they already have, we can reap the benefits!"

Let’s put aside the argument that a new language or tool is significantly better than the old. Wholesale retraining and replacement never seem to happen. Instead, we are stuck with some of the new and some of the old.

As this cycle repeats itself, we end up with a bit of everything: Bash, Puppet, Chef, Ansible, Python, Ruby, Powershell, and more.

One can argue that a significant flaw of past automation efforts was that we believed "one language to rule them all" was even possible.



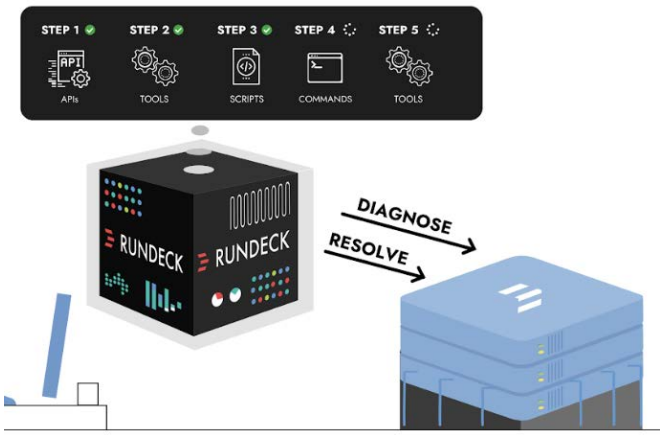
"Islands of Automation" are common in enterprises

Instead, we should accept that — in any sufficiently large enough organization — different teams are going to have different requirements and preferences that lead to different automation choices.

Instead of fighting the heterogeneous automation and tooling inevitability, how can we leverage it and make the most of it?

Crossing team and system boundaries are where mixed environments become a problem. Operational procedures (health checks, restarts, updates, etc.) need to work across all of the components of an environment — and the decisions different teams made at different times about those components.

Rundeck is designed to easily create workflows that span different automation tools, system commands, and API calls. Anyone who needs to execute those procedures can do so via Rundeck, regardless of the mix of underlying automation.



Rundeck ties together you existing automation.

Using Rundeck to mitigate the pain of repetitive issues

1. Create Rundeck jobs (workflows) where each step calls your existing scripts, system commands (i.e., what you would type at the command line), or API calls

2. Use Rundeck's fine-grained access control to provide access to anyone who needs to run those jobs

3. Teams who are responsible for individual system components can refactor or replace lower-level automation and update the Rundeck jobs as they see fit (without impacting those who use the jobs)

07

Incident! Page everyone.

⚠️The Pain⚠️

Business: Longer incidents. Incidents are more expensive.

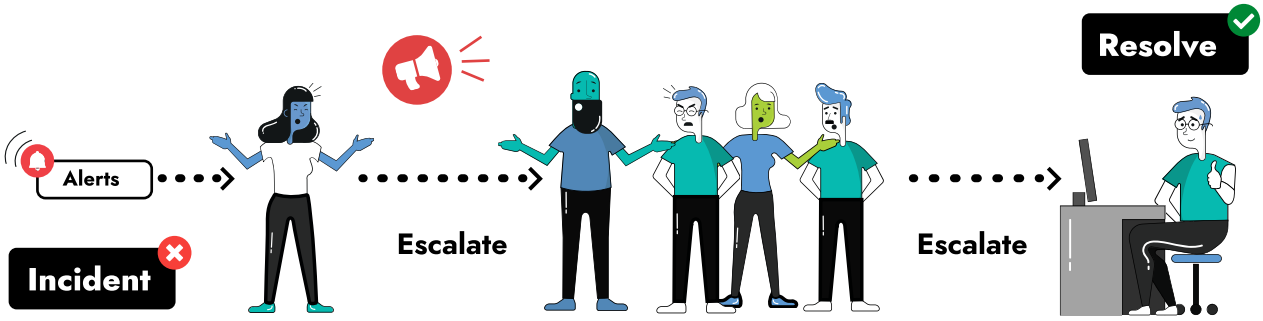
Team: Inefficiency. Work disruption. Slow response metrics.

Engineers: Interruptions. Finger pointing.

Working in Operations means you've undoubtedly been dragged into a war room, incident bridge, or incident chat room filled with your colleagues.

Bringing together everyone who might be involved with the incident has long been standard practice in our industry for major incidents.

Unfortunately, in many organizations, this practice expands beyond major incidents. The "bring everyone" approach is used for lesser severity incidents as well.



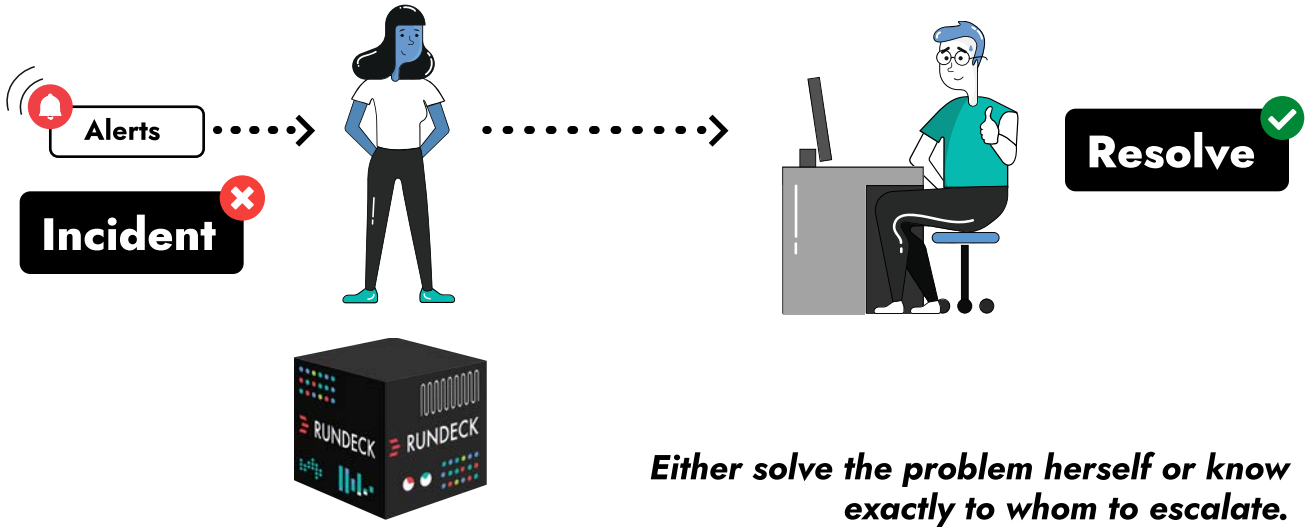
Traditional "war rooms" and escalation patterns are slow and costly.

The theoretical upside of the “bring everyone in” approach is that you have a better chance of getting the right knowledge on the call to diagnose and resolve the incident.

However, the process of alerting and mobilizing a large number of people comes with three downsides:

1. Additional delay - It takes longer to assemble a larger group
2. Higher cost - You are disrupting a larger group of people, injecting exponentially more delay and context switching costs into your organization
3. Adds Coordination cost - There is overhead in managing communication among a larger group and a higher likelihood of miscommunication.

If your goal is to have shorter incidents and fewer escalations, the use of Rundeck can help you achieve your goal.



Either solve the problem herself or know exactly to whom to escalate.

“run health checks”

By using Rundeck to distribute the ability to run diagnostics and health check jobs, your first responders have a better chance of either knowing how to solve the issue right away or know precisely to whom to escalate.

Using Rundeck to mitigate the pain of repetitive issues

1. Collaborate with your subject matter experts to create Rundeck jobs to run health checks for system operating conditions and known error states. This could be as simple as asking, “how would you know if this component is healthy?” and writing a quick script or use Rundeck plugins.

2. Empower anyone in your organization who responds to incidents to run those health check jobs on demand.

3. Optional: empower those same responders to execute repair jobs as well (e.g., restarts, rollback, failover, etc.)

READY FOR SHORTER INCIDENTS AND FEWER ESCALATIONS?

Let's talk.
hello@rundeck.com