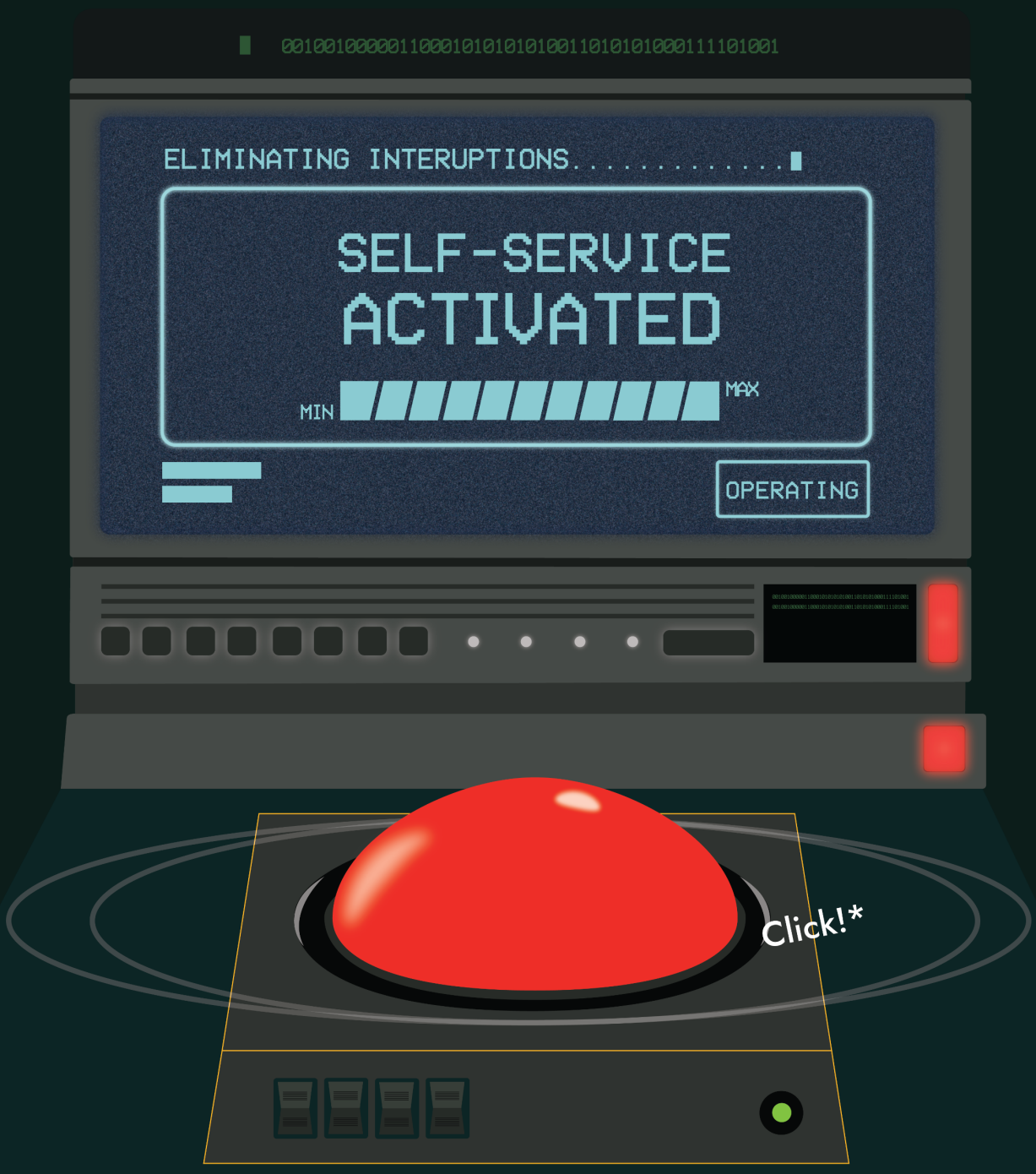
RUNDECK

SELF-SERVICE OPERATIONS

The Guide to Fewer Interruptions, Less Waiting, and Getting More Done



Do you want to eliminate the interruptions, waiting, and frustration that undermines Operations work?

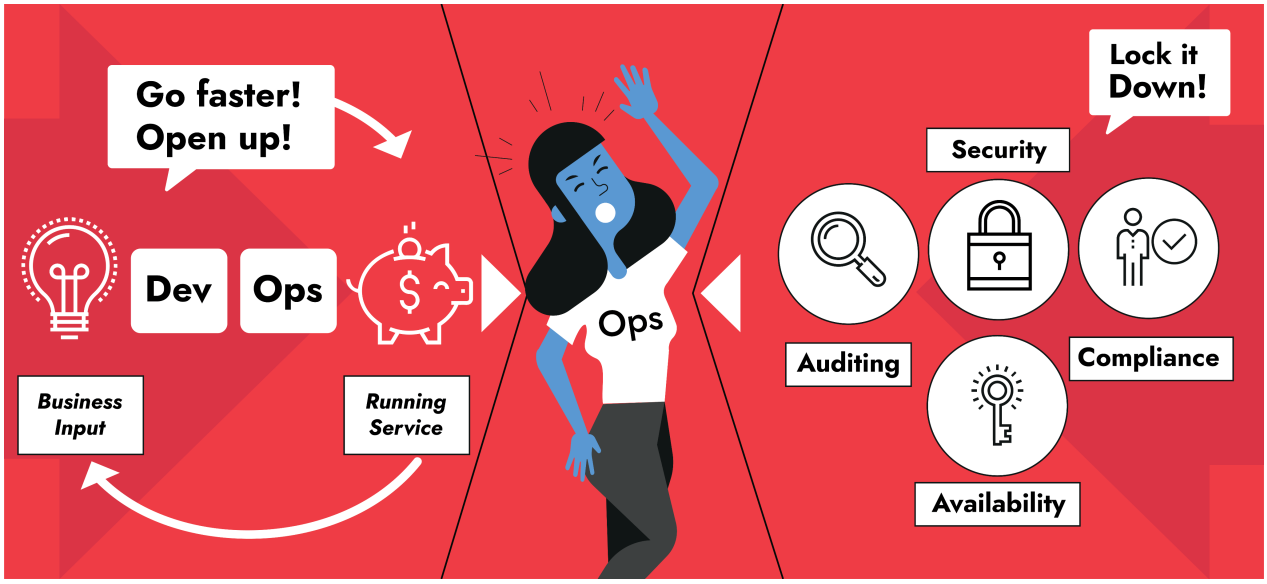
This guide is for you. Let's dive into the why, what, and how of Self-Service Operations.

Table of Contents

Introduction

The Operations Squeeze

DevOps and Digital Transformations are driving an unprecedented increase in the pace and volume of daily change. While this might be great news to Development and Product teams, their counterparts in Operations are alarmed at the problems and challenges that comes with this increase in pace and volume of work.



Operations organizations are finding themselves squeezed between two unrelenting forces. On one side there are the business-driven demands of DevOps and Digital Transformation

(“Go faster! Open things up!). On the other side there are the business-driven demands to maximize security and stability ("Don't be the next hack! Don't be the next outage! Lock things down!"). And there, in the middle, is an already over-burdened Operations organization doing their best to avoid being squeezed beyond the breaking point.

Operations is reaching an inflection point. To deliver what the business demands, Operations has been challenged to find a way to provide unprecedented levels of organizational responsiveness and throughput — all while “locking things down” to sufficiently meet today’s risk profiles.

A lot is riding on how Operations responds to this challenge. A failure here is not just a localized IT failure. A failure will undermine a business’s ability to operate. Failing to solve this IT challenge will turn into a competitive disadvantage for the business.

On the flip side, this challenge also presents a great opportunity. Operations can take this business mandate and use it to reimagine how both planned and unplanned work is handled. This is a chance to improve how Operations both serves the broader business and improves the day-to-day lives of Operations professionals.

Self-Service Operations

Self-Service Operations is a key design pattern that allows an Operations organization to move faster, be more flexible, and lock things down. Self-Service Operations is also critical technique for breaking down the organizational barriers that prevent enterprises from achieving DevOps and Digital transformations

On the surface, Self-Service Operations is a straightforward concept — turn your operations tasks into automated services that can be consumed on-demand (via GUI, command line, or API) by anyone who needs those operations task performed.

However, when you look deeper, you'll see that a lot goes into making this straightforward vision a reality. This book introduces the why, the what, and the how of Self-Service Operations.

Part 1: Why Self-Service Operations?

Interruptions Keep Getting in the Way

Operations is one of the few areas in all of business where both planned and unplanned work co-exists by design.

On the one hand, there is engineering and project work to be done — like delivering new platforms, spinning up new environments, working on scaling and performance, and analyzing new technologies.

On the other hand, there are the constant interruptions — responding to incidents, scaling events, security events, customer requests, and inquiries from colleagues.

Most of the interruptions coming into Operations are time sensitive. Obviously, outages are urgent, and everyone depends on you to respond immediately. However, even the requests that someone should have seen coming — like business and customer requests or a colleague who ran into an operational issue delivering a project — all get marked “urgent” and presented to you at the last minute, interrupting your work.

Making matters worse, you often can't give the engineering work to one team, and the interruptions to another team. Knowledge, skills, and capability are distributed unevenly throughout your organization. The various bits of planned and unplanned work require unpredictable expertise in different domains and familiarity with previous work. This is how you end up with everyone juggling all types of work at one point or another.

From an organizational perspective, these interruptions are expensive. People's productivity ends up suffering from death by a thousand cuts. From a manager's perspective, this leads to the dreaded, but all too common, "everyone is busy, but nothing is getting done" syndrome.

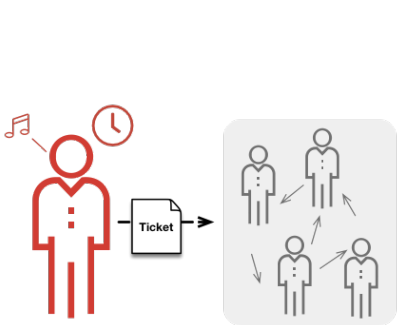
From an individual perspective, interruptions are a source of frustration and stress. Interrupts are frustrating because they prevent you from getting to the important work you know you need to do. The feeling that you are behind or that your control over your time is slipping away is stressful, demoralizing, and can lead to burnout.

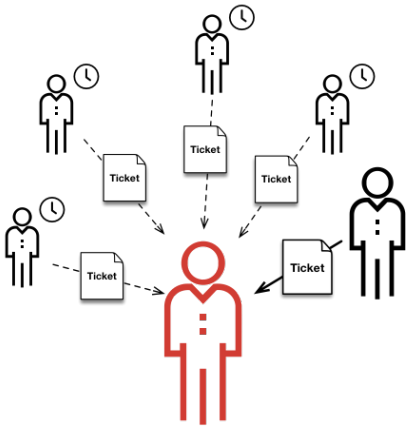
Self-Service Operations helps you avoid interruptions. Easily setup self-service interfaces for any number of operational tasks and cut out all of the repetitive tasks that chew up people's time.

Too Much Time Is Spent Waiting

Much like interruptions undermine productivity in Operations, waiting is also a significant source of waste.

You lose time waiting for someone to do something for you. Others lose time waiting for you to do something for them. Waiting is all too often a repetitive fact of life in Operations.





If you simply add up the time spent directly waiting, you can get to a big number. However, the cost to the organization is even more substantial if you take into account the compounding nature of delays in a complex system like a technology organization in an enterprise.

Of course, we can't forget what the the Lean, Agile, and DevOps movements taught us about delays. Delays slow down feedback, and slow feedback leads to lower quality. Delays also increase the likelihood that the context of the work has changed, which can cause further issues, like doing the wrong thing. Delays also have a business cost (e.g., slower time to market). Small delays can compound into big problems.

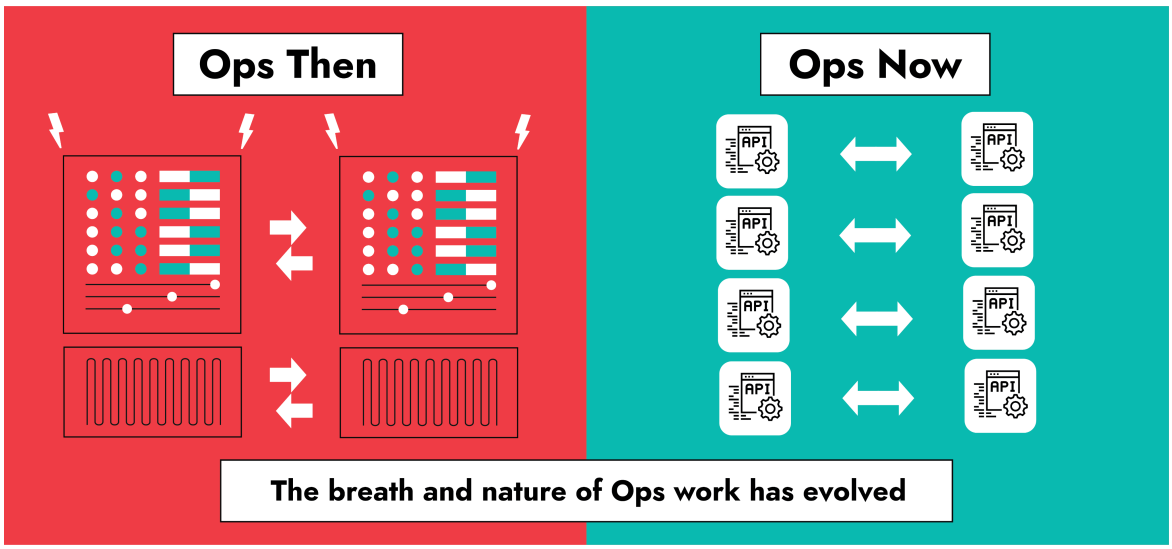
Self-Service Operations helps you eliminate waiting and delays. Easily setup self-service interfaces for any number of repetitive operational tasks and eliminate the amount of time that people spend waiting for you or you spend waiting for others.

You Need to Do More and Do More of It

Operations has always carried the mandate of meeting the business needs through designing, running, and fixing complex systems. However, what that operations work looks like and what those systems are comprised of has changed over time.

In the past few decades, we've seen the focus of typical enterprise operations organizations steadily expand from networking, to server platforms, to service management, to API integration. If we

look at what happens today, operations encompasses the full stack of a very complex, software-defined, API-enabled system running on infrastructure they may or may not own.



Part of this expansion can be attributed to the dramatic evolution of the underlying enterprise compute technologies. From open platforms to virtual machines to cloud to containers to “Cloud Native” — there has been one major shift after another.

Of course, the systems built under each new technology paradigm never fully replace the systems built under the old paradigms. It's not uncommon for an enterprise to have an accumulation of systems built over 10-15 years and have no budget, risk appetite, or even viable way to replace them all.

With each shift, who bares the brunt of the responsibility for making sure the old and the new hang together? Operations, of course. With each new advance, Operations juggles more

complexity and more layers of legacy technologies than ever before.

One belief that has remained consistent throughout most of this evolution is the idea that operations work may only be executed by a distinct and separate Operations team. For decades it was almost considered heresy to suggest dismantling the strong, wall-like division of roles and responsibilities between Development teams and Operations teams.

Today, the DevOps and Cloud Native movements are strongly challenging the old “Ops in a silo” orthodoxy. These movements have been taking an end-to-end look at improving the entire IT

lifecycle. In doing so, they are challenging many old assumptions.

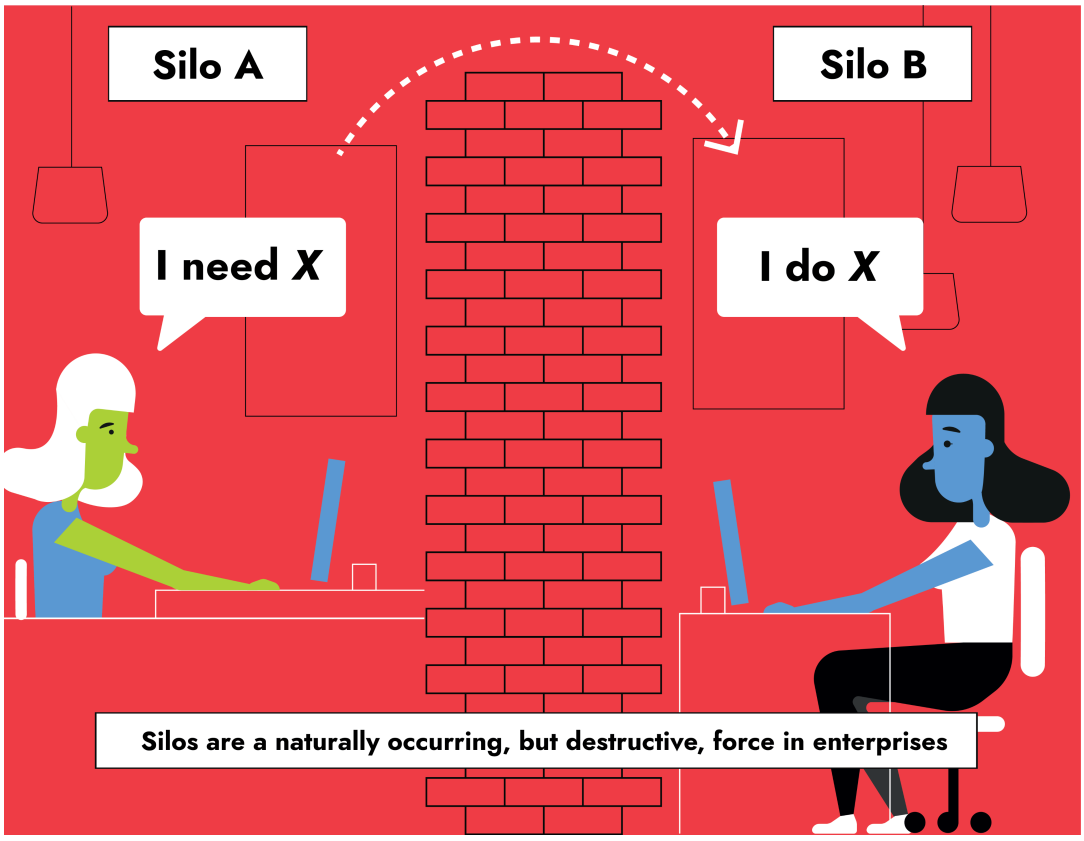
DevOps and Cloud Native share a ruthless focus on improving time-to-market while simultaneously improving quality and lowering costs. These practitioner-led movements have repeatedly shown that the strongly-siloed traditional way of operating is why so much of IT was delivered late, cost too much, and didn't deliver on its promises to the business.

To understand why Self-Service Operations is so important we need understand why the old way of working in silos and passing work through resulting request queues can be so destructive.

Silos and Request Queues Are Highly Destructive

The term "silo" is a bit of DevOps jargon used to describe a specific condition that occurs naturally anywhere a significant number of people gather to do work.

Following human nature, organizations tend to divide up their work and their people by functional specialization. As these divisions occur, human nature further encourages the people within these divisions to focus inward and optimize their work for their specialization. This is where silos start to form and the problems begin.



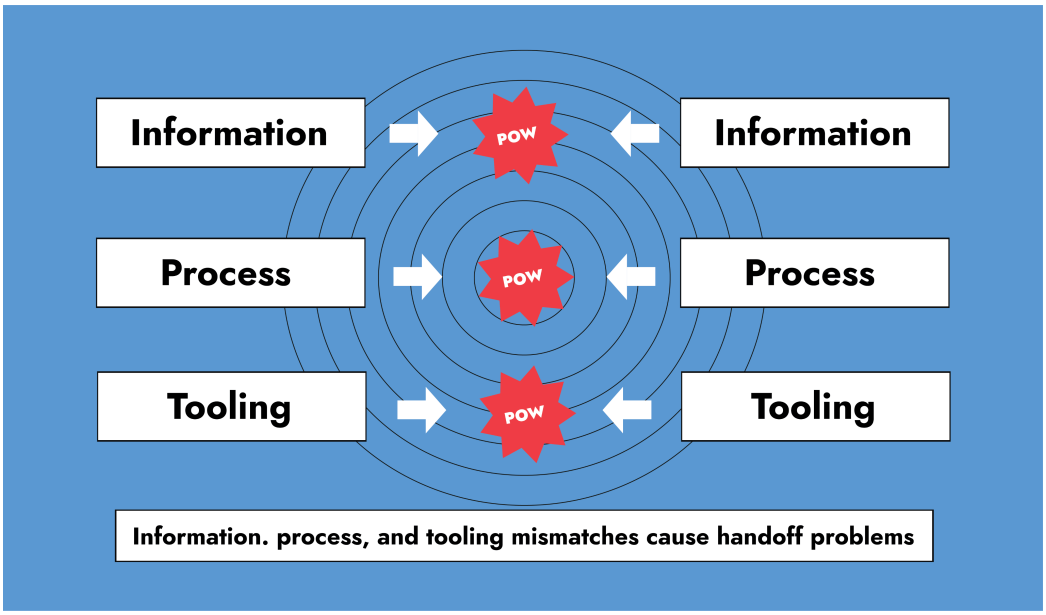
A team is said to be “working in a silo” when that team is working in a different context from other teams, their work comes from a different source (i.e. a different backlog), and they are working under different priorities or incentives. While specialization and focus are inherently a good thing for skills development, we have to be vigilant to avoid the unintended formation of organizational silos. It should be noted that silos are not meant to be in indictment of having organizational structure or division. While structure can contribute to the formation of silos, it is really how people work that makes it a silo. Silos may tend to form along team boundaries, but teams do not necessarily have to turn into silos.

One of the first signs of silo formation is difficult or eror-prone. The most common culprits of these handoff problems are:

- **Information mismatches –** The parties on either side of the handoff are working with different information or are processing the information from different points of view, leading to an increase in errors and rework (i.e. repeat work due to previous errors).

- **Process mismatches –** The parties on either side of the handoff are following either different processes or processes that are nominally the same but take a different approach and produce results not expected by the other party. Timing and cadence mismatches between parts of a process that take place in different silos also lead to an increase in errors and rework.

- **Tooling mismatches –** An increase in errors and rework is seen when different parties on either side of silo boundaries are using different tooling or tooling that isn't setup to connect seamlessly. When the work needs to be translated on the fly by a person moving information and artifacts by hand from one tool to another, delay and variance are bound to be introduced into the process.



Another sign of silo formation is that request queues appear at the boundaries of the silos and grow increasingly longer. The stronger the silo effects, the the longer the requests queues become. As the silo effects take hold, those fulfilling the requests end up working increasingly disconnected from the requestors and, due to the issues listed above, that leads to additional errors and rework.

On the surface, request queues seem like an orderly and efficient way to manage work at organizational divides. However, if you look under the surface you will find that request queues are a major source of economic waste in any business.

Why are requests queues a major source of economic waste? Let us look at the list created by noted author and product development expert, Donald G. Reinertsen.

Queues create:

- **Longer Cycle Time –** Queues increase cycle time as it takes longer to reach the front of a large queue than than a small one. Even small delays can exponentially compound within a complex interdependent system like an enterprise IT organization.

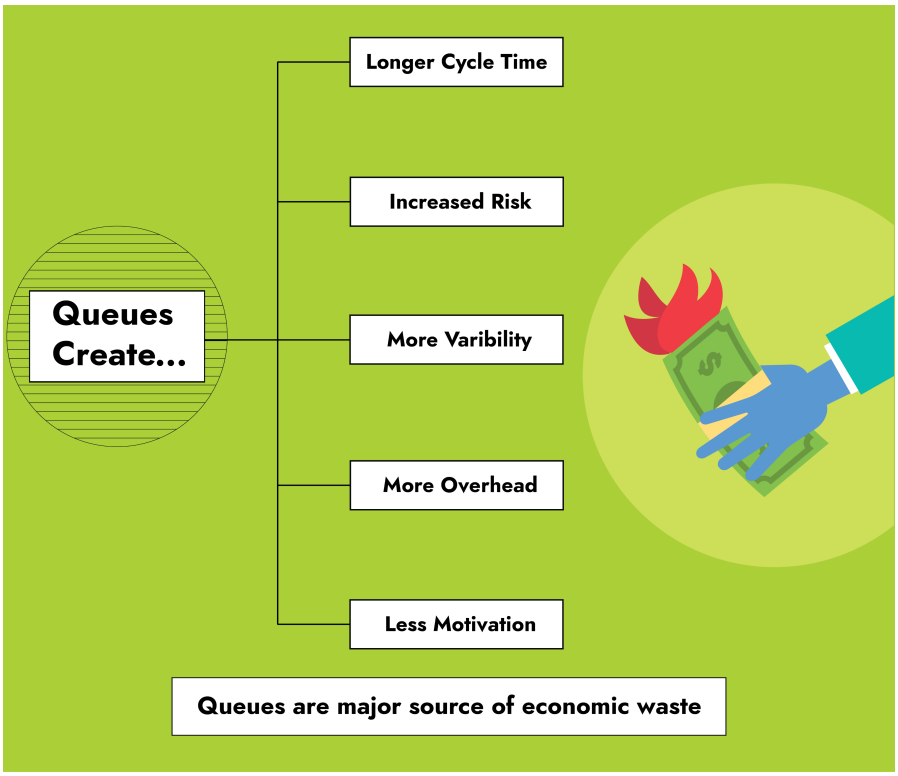
- **Increased Risk –** Queues increase the time between request and fulfillment which in turn increases likelihood of context of the request changing. If a problem does arise, the requester is now in a different mental position (often working on something else) from where they were when they made the request.

- **More Variability –** Longer queues lead to high levels of utilization and higher levels of utilization amplify variability. This leads to longer wait times and a higher likelihood of errors.

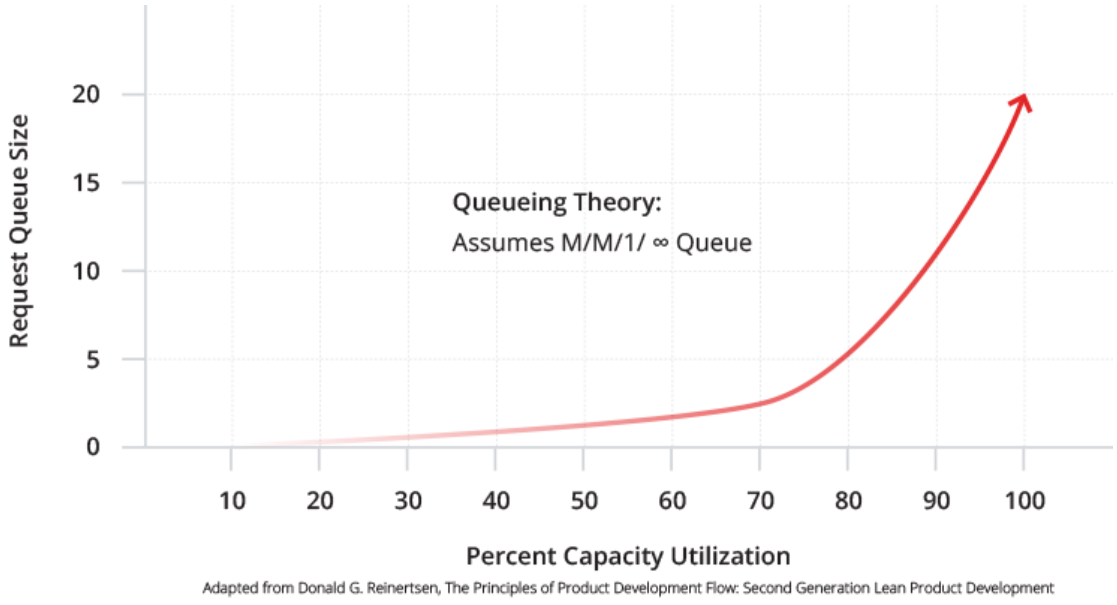
- **More Overhead –** Queues add a management overhead for managing the queue, reporting on status, and handling exceptions. The longer the queue, the more these overhead costs grow in a compounded manner.

- **Lower Quality –** Queues lower quality by delaying feedback to those who are upstream in the process. Delays in feedback causes the cost of fixing problems to be much higher (e.g. bugs are easier to fix when caught sooner) and often means that additional problems of a similar origin have been created before the first negative feedback arrives.

- **Less Motivation –** Queues have a negative psychological effect by undermining motivation and initiative. This is due to queues (especially longer queues) removing the sense of urgency and immediacy of outcomes from the requestor's work. If you don't feel the impact and don't see the outcome, it's human nature to grow negatively disconnected from the work.



Reinertsen also points out that higher the capacity utilization of a team, the exponentially longer the request queues will be (which directly increases all of the economic wastes previously listed). Given that most IT operations teams operate at near full capacity, this effect needs to be understood.



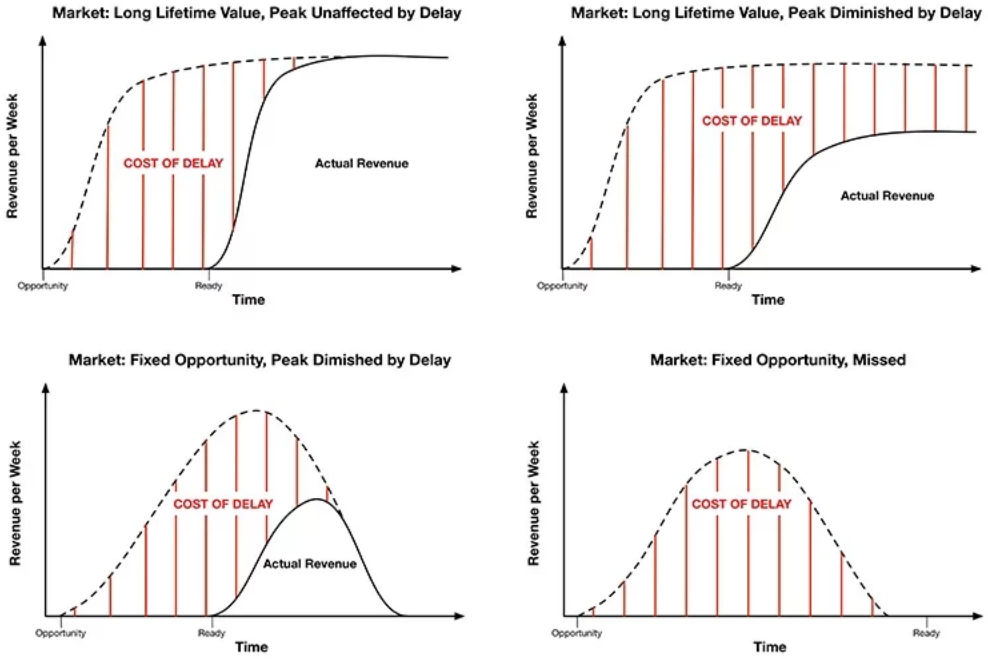
Relationship between queue size and capacity utilization

As a team approach 100% utilization, the request queues increase in size exponentially. As we move from 60% to 80% utilization, the queue doubles. As we move from 80% to 90%, the queue doubles again. As we move from 90 to 95 percent, the queue doubles again. You can see that if a team operates in a high-utilization environments and uses request queues to manage its work, queuing theory dictates that it is nearly impossible to keep request queues small and your organization will continue to suffer from the negative economic effects of large queues.

While the waiting, bottlenecks, errors, and rework that comes with queues can cause operations to be far more expensive than it needs to be, it's the cost of delay that can have a profound effect on a company's fortunes. For every delay introduced into a company's processes, there corresponding effect of reducing how

quickly the business can react to the market and how quickly the business can deliver. While the individual effect of each delay can be almost imperceptible, in an organization full of request queues they add up (and compound) quite quickly.

Delay has a cost and the cost can be quite expensive. As Reinertsen says, “If you only quantify one thing, quantify the cost of delay”.



Understanding your Cost of Delay is key to understanding economic impact of queues

Request queues have a fundamental negative effect on the economics of any business that depends on IT to build or maintain business advantage. Despite this, what is the most common method of managing work inside IT operations organizations? Request queues in the form of ticket systems!

For decades, ticket-driven request queues fulfilled by manual or semi-manual processes have been the most dominant style of working in enterprise IT Operations organizations. The typical operating model was to have lots of specialist teams divided by functional expertise and use ticket systems to govern the flow, approval, and order of work between the different specialist teams. It was also typical for a heavy reliance on project management coordination to push work through this system of silos.

If you examined how any of the teams worked under this operating model, you would likely find the definition of a silo. You would then find most — if not all — of the negative effects of queues and add up the cost to the business (in both increased cost of operations and cost of delay).

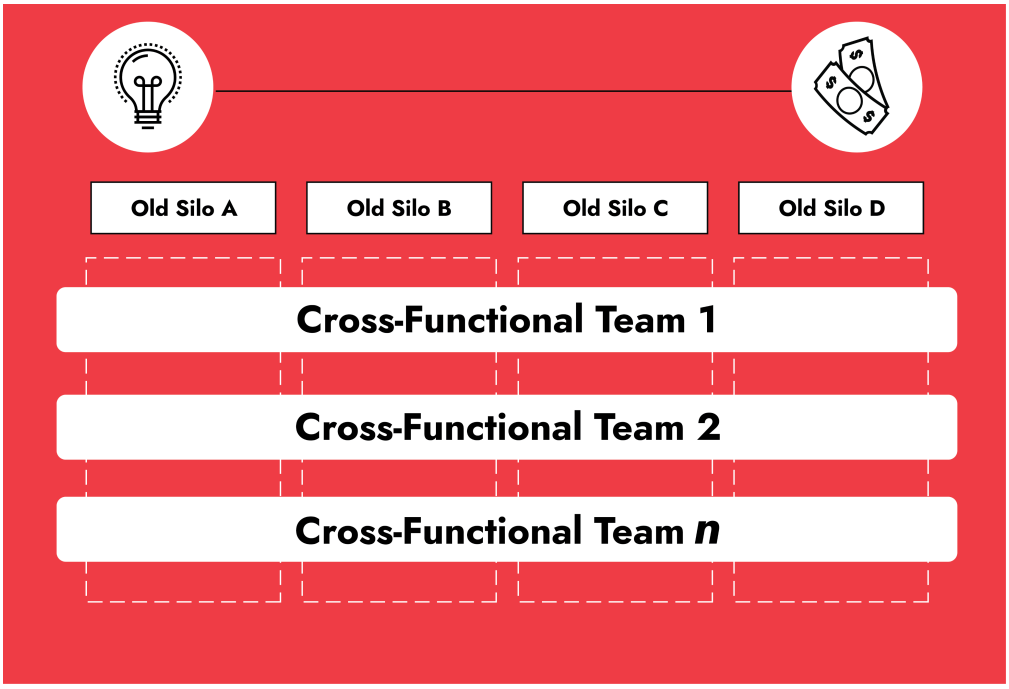
Even as ticket systems have been rebranded as ITSM tools and new features and processes for queue management have been introduced, it still isn't addressing the fundamental problems that come with functional silos reinforced by ticket-driven request queues fulfilled by manual or semi-manual processes.

Design Patterns for Solving Silo Problems

How can enterprise operations organization solve these silo problems? Two key strategies have emerged from the DevOps community.

The first strategy is perhaps the most obvious: get rid of as many silos as you can.

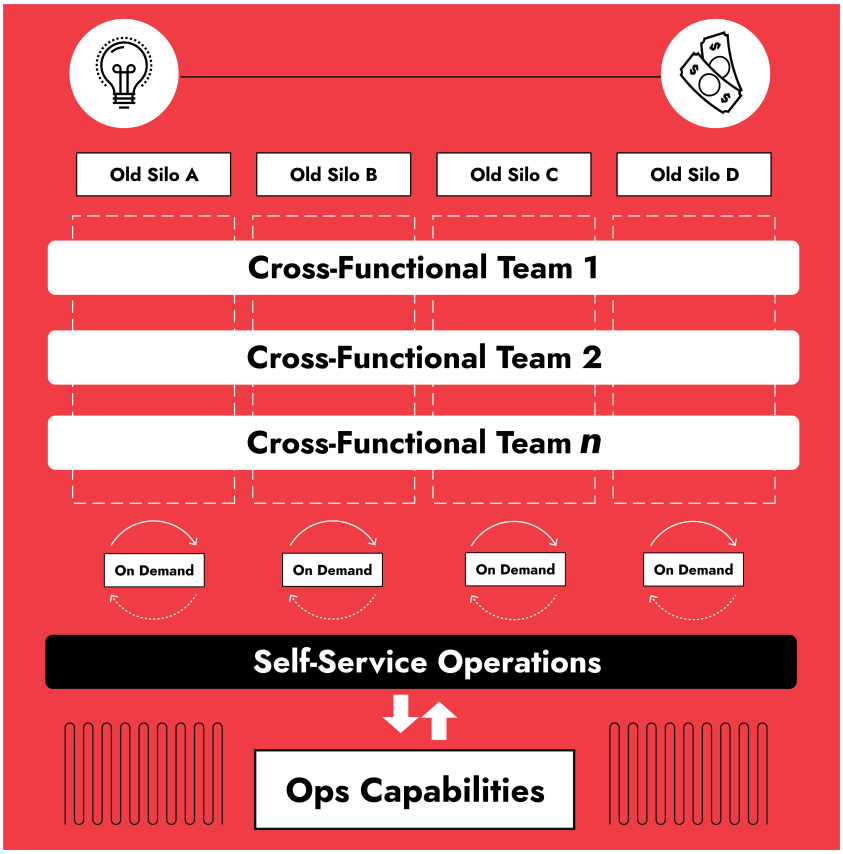
Forward thinking organizations are transforming from a traditional “vertical” structure aligned by function to a “horizontal” structure aligned by value stream.



This generally means creating cross-functional teams that can handle as much of the lifecycle as possible without needing to hand work off to other teams. It is very difficult for silos to form if you don’t have handoffs or breaks in context and everyone is working from a single backlog with common priorities.

The second strategy is the primary focus of this paper: Self-Service Operations. Wherever silos cannot be avoided, you must apply techniques and tooling to mitigate the negative impact of those silos. Self-Service Operations does just that by enabling both the definition and execution of operations activity to be delegated

throughout a broader organization and across traditional organizational boundaries. Wait time is eliminated, feedback loops are shortened, breaks in context are avoided, tooling is aligned, and labor capacity is improved.



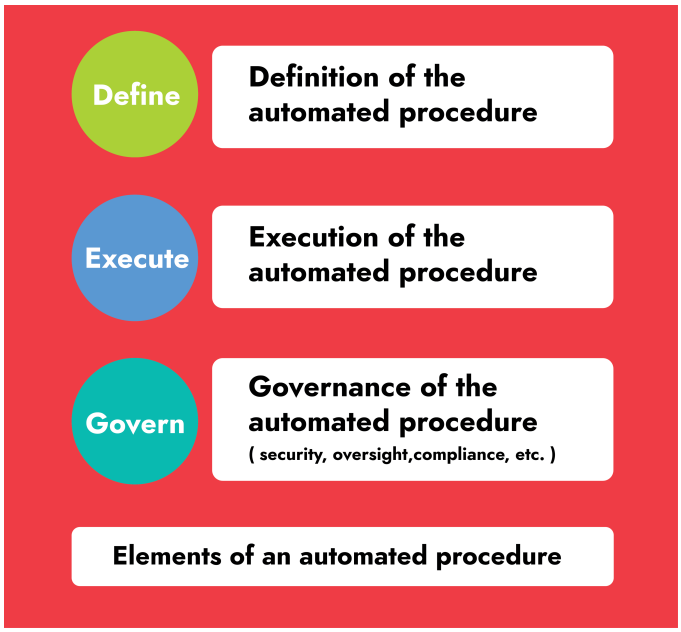
Part 2: What is Self-Service Operations?

Self-Service Operations turns your operations tasks into services that can be consumed on-demand (via GUI, command line, or API) by anyone who needs those operations task performed. Let's look at how this works and how to calculate the ROI.

Change how you think about automated procedures

Self-Service Operations is built on a fundamental shift in thinking about automated operations procedures.

Traditionally, automated procedures have been viewed as monolithic things that are created and live within Operations (and often within the same Operations team). In reality, an automated procedure has three distinct parts: the definition of the automated procedure, the ability to execute that automated procedure, and the security or management policies governing that automated procedure.



Many of the beliefs around what we can and can't do with regard to automating operations procedures stems from this monolithic view. If we can instead break out our thinking about automated procedures into these three distinct parts, the possibilities for changing an organization start to open up.

We can start to think about who should be responsible for each of the elements:

- **Define** – Who creates the definition of the automated procedure and how do they do it?

- **Execute** – Who can execute the automated procedure and how are they provided the ability to do it?

- **Govern** – Who has governance over the procedure and how do they manage security, management oversight, and compliance?

The goal is to be able to move the individual elements of each automated procedure to the part of the organization where the move improves the flow of work and best utilizes your labor.

For example, we can enable a scalable organization where developers collaborate on defining operations procedures (e.g. “This automation smartly restart this application”), the operations group vets and improves on those procedures (e.g. “Is this safe? Will this do what we want? Does it play nice with other systems?”) , and then the security organization can control where the procedure can be run and who can run it (e.g., traditional operations engineers, developers, a dedicated IT Ops support team, etc.).

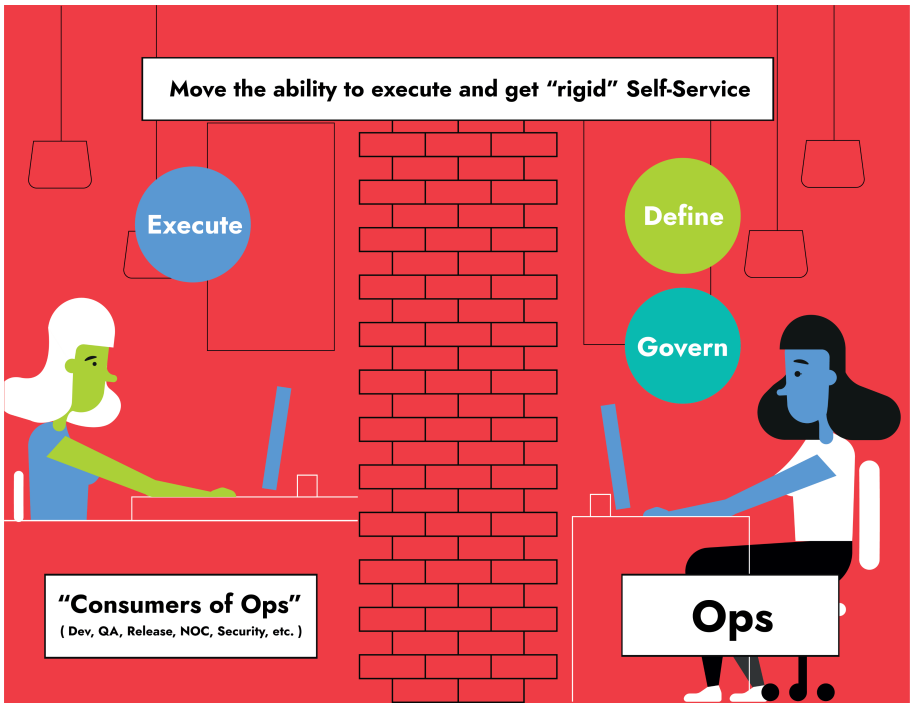
Let’s take a look at what is possible when we move the responsibility for the different elements of an automated operations procedure.

Baseline: Traditional Operations Silo

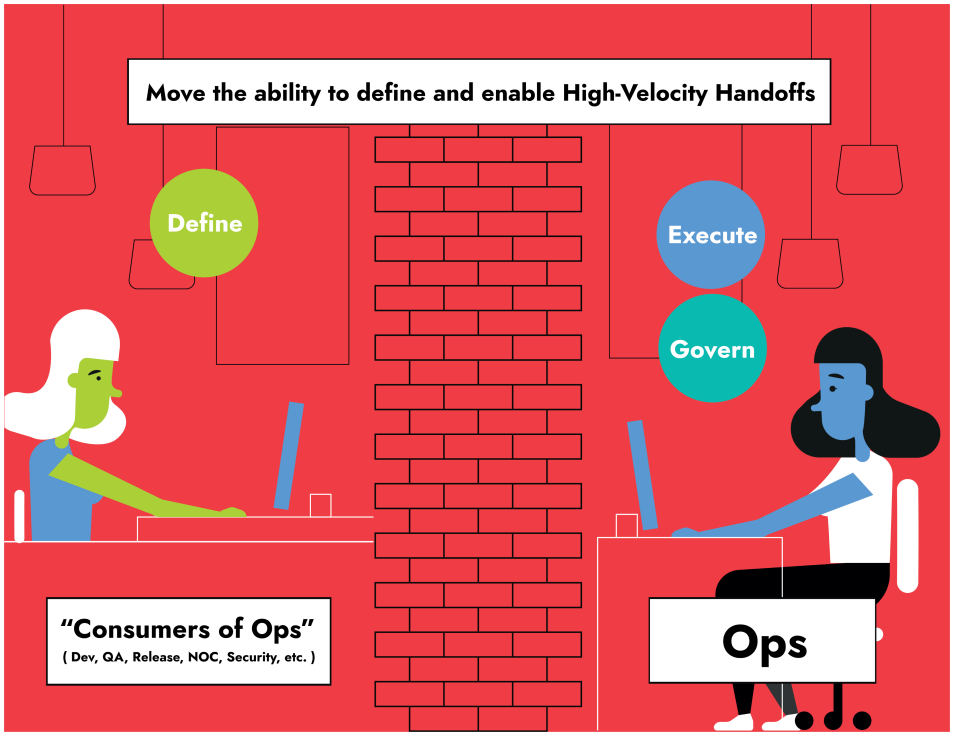
In the traditional siloed way of working, all operations activity is executed by a centralized operations team. All three elements are the sole responsibility of that centralized operations team. Do you need an operations task completed? Submit tickets and wait in request queues. Still waiting? Try to get the ticket escalated.

History has shown us that this model causes the most organizational pain. Operations bottlenecks and labor shortages are common. Handoffs to teams outside of operations (and sometimes even within operations) are long and error-prone. When faced with today's high-velocity software lifecycles and dynamic infrastructures, operations organizations struggle under this model. The recent work of the DevOps community has repeatedly advised against this way of operating.

Rigid Self-Service



High-Velocity Handoffs



In most cases, the systems currently running in your environments were created or assembled by a team outside of operations. Therefore, it is a reasonable assumption that the deepest knowledge of these systems and how to fix them reside outside of Operations. As those teams sprint forward, Operations will always be playing catch-up.

To help Operations get up to speed, the teams outside of Operations often perform a knowledge handoff to Operations in the form of a documentation dump (readme files, "do-this-then-do-that" word documents on SharePoint servers, ticket comments, etc.) and perhaps some "it worked in my environment" scripts. At the handoff, Operations is expected to

quickly come up to speed and build the procedures and the automation necessary for the management of systems in the higher pre-production environments (e.g. UAT, STAGE, etc.) and production environments.

In addition to the error-prone nature of relying on human-to-human knowledge transfer, these handoffs are time consuming and resource intensive, often requiring the full attention of your most skilled engineers. As lifecycles speed up and environments become more dynamic and complex, these handoffs quickly become a major source of bottlenecks and their inevitable errors become a weak link for quality and security.

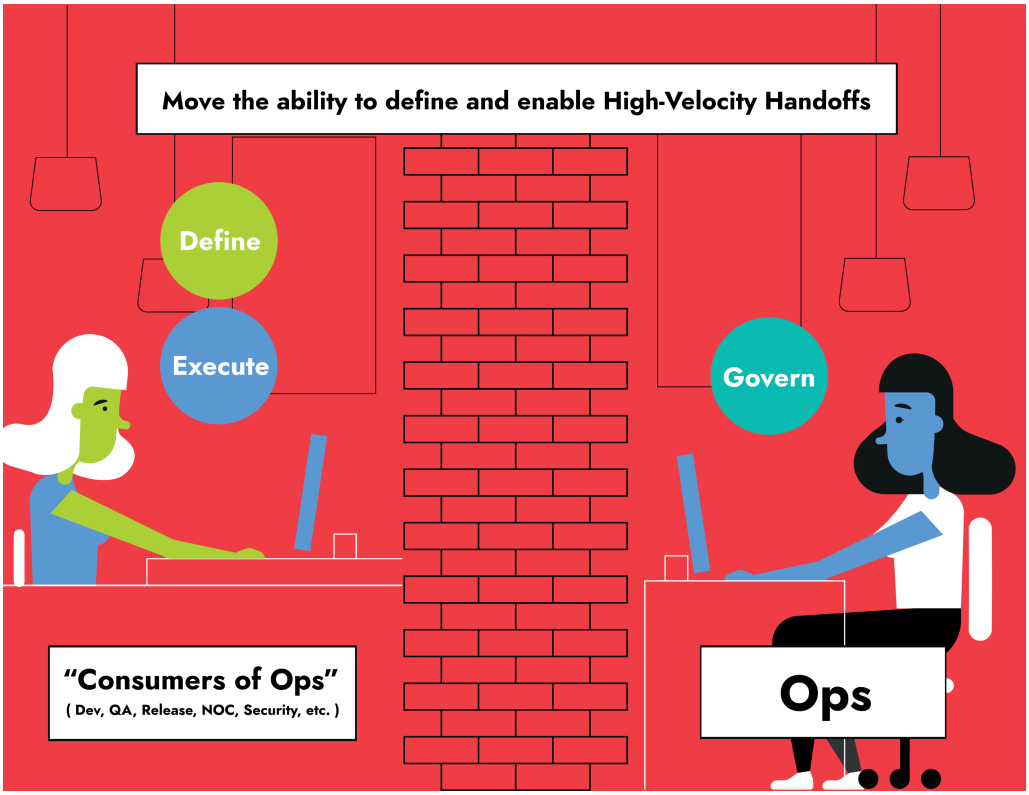
A key step in achieving a self-service style of operations is delegating the responsibility for defining and creating automated operations procedures to the creators of the systems and software running in your environment. A simple example would be a mandate that developers write and maintain all automated operations procedures for all software that they create.

This means that the automated procedures have to be seamlessly reusable by both the development team and the Operations team. Automation for one component must also work seamlessly with automation for other components. One might think that this implies that a single automation tool would have to be used across all teams. But, that is not actually the case.

Forcing teams to standardize on one language or automation framework just isn't realistic given the heterogeneous nature of modern enterprises. Teams need to be able to use the automation

languages and tools that they want, while allowing for other tools, to orchestrate procedures across those underlying frameworks and languages. If you can achieve this, you get high-velocity handoffs that improve the flow of work, improve quality, and relieve Operations of significant pressure.

Self-Service Operations

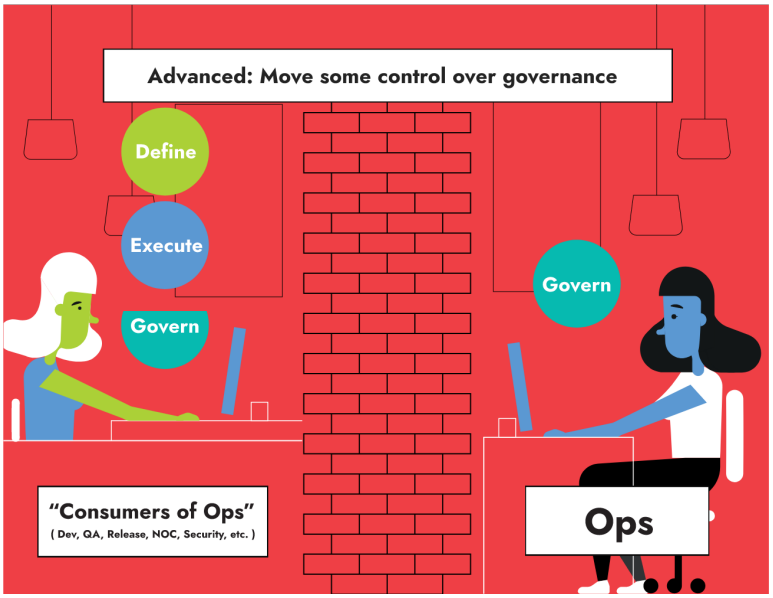


Once you combine self-service capabilities for both defining and executing automated procedures, you have enabled Self-Service Operations. Like any modern, on-demand "_aaS", you are putting as much control as possible into the hands of the requesting party. This lets the requester complete their tasks, when they need to, and keep feedback loops as tight as possible.

ecurity and management controls are significant requirements for Self-Service Operations solution. This way of working can only exist in an enterprise setting if Operations can maintain full security controls, enforce compliance, and have management oversight.

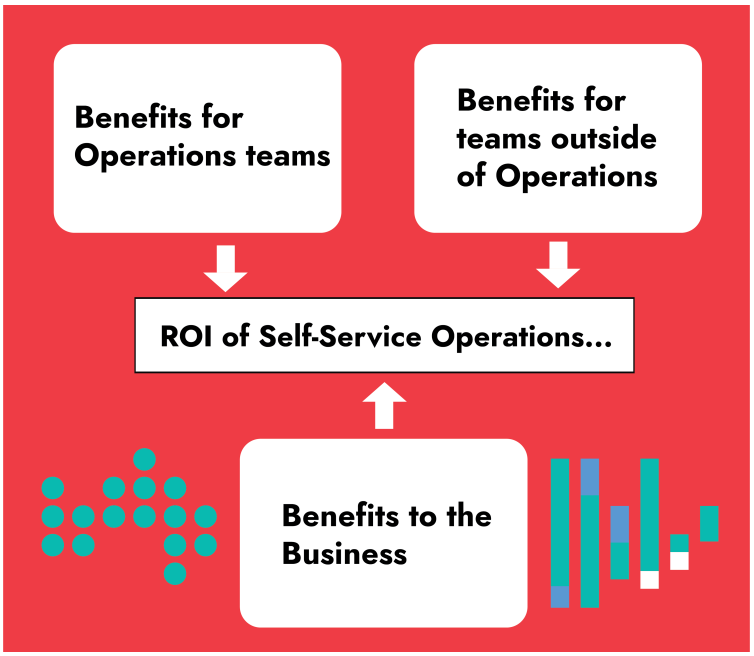
If done correctly in a low-friction manner, everybody wins. Operations gives autonomy to teams who need operations tasks performed while simultaneously locking down critical information to a greater degree than is even possible today.

As an organization advances with the Self-Service Operations, it is a logical next step to push some governance capabilities to other teams. This is popular with organizations that have multiple lines of business with different risk profiles. While Operations must retain ultimate control, there are some access control and compliance decisions that can be pushed closer to the owners of the risk within those lines of business.



ROI of Self-Service Operations

There are multiple business drivers for moving to an Self-Service Operations model. Calculating a return on investment will always depend on a company's unique environment, however here are some ways to build your company specific ROI calculations:



ROI benefits for teams doing operations work:

- Decrease in time to respond to incidents (MTTR / Mean Time to Repair)
- Decrease in errors and rework (Fewer mistakes, greater consistency)
- Increase in total support volume the team can take on (Comfortably do more with same number of people)

- Increase in operational support tasks that can be handled by other teams (Create more capacity by enabling other teams to do operations support tasks

ROI benefits for teams doing operations work:

- Decrease in number of escalations (Fewer interruptions, less context switching)
- Decrease in time spent waiting for completion of tasks by Operations support teams (Safely take action rather than wait for tasks to be done by operations support teams)
- Decrease in issues due to problematic or incorrect handoffs (Less unplanned work)

ROI benefits for teams doing operations work:

- Decrease in total support costs (More cost-effective organization)
- Decrease in time to market (Quicker cycle-time and less schedule slippage)

Increased visibility for audit and compliance

The benefits accrued by each constituent group (Operations and the groups served by Operations) reinforce each other and compound over time. The net effect is that the ROI to the business should grow exponentially as more Self-Service Operations

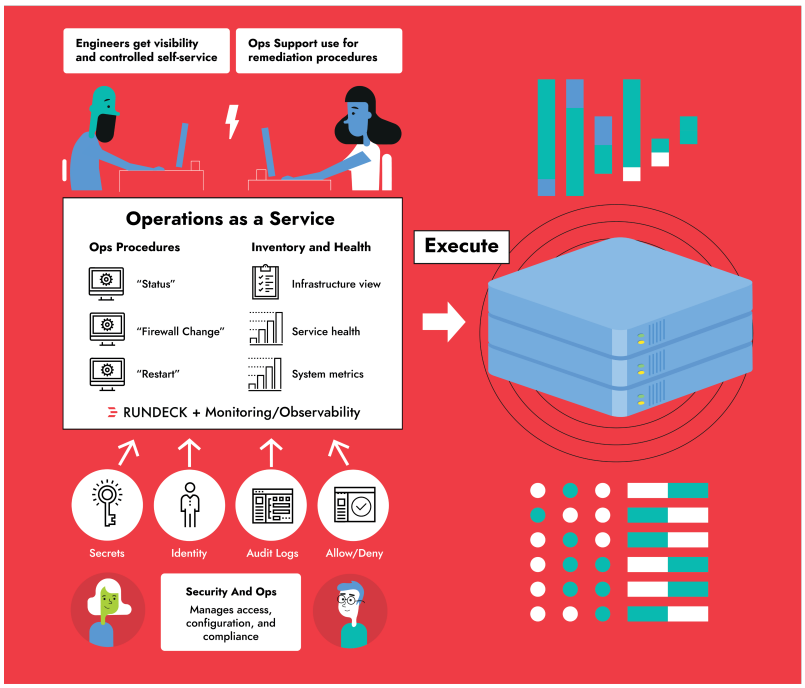
capabilities are rolled out to additional parts of the company. If you are getting started and want the simplest ROI measure that everyone can rally behind, use lead time. There are lead times all over your organization and each one has a cost associated with it.

There are lead times for feature delivery or lead time in resolving an incident or lead time in waiting for an environment change. Calculate how much each of those lead times cost. When talking to engineers you can express lead time in how long they are waiting or how long it takes them to do repetitive tasks for others. When talking to the business you can add up those lead times and quantify the cost of delay. You can then make the financial argument that you should apply the Self-Service Operations pattern to reduce as many of those lead times as possible.

Part 3: How to Build Your Self-Service Operations Capability

Now that we've covered the "Why" and the "What" of Self-Service Operations, let's look at how companies go about building their Self-Service Operations capabilities. We've noticed a general pattern that companies follow. The process can be broken down into the following four steps.

Step #1: Establish your Self-Service Operations Platform



The first step is to focus on creating a central, secure hub that serves three primary functions:

1. Framework for defining and executing automated operations procedures

2. Point of enforcement for access control and governance requirements

3. Views of relevant configuration and health information for the environments and systems on which operations procedures will be run

Heterogeneity is a fact of life in the enterprise. Multiple generations of different platforms and tools will need to co-exist. The idea that an enterprise of significant size can move exclusively to one platform and one automation language/framework just isn't reality. In addition to the logistical, financial, and technical debt barriers to completely retooling, its natural for different teams to want to use reuse their skills and select tools that best fit their specific needs.

Plan for this heterogeneity by focusing on the orchestration and scheduling of those underlying platforms and supporting tooling. Allow teams to create the component automation using the scripting languages or tools that they want to use. The focus of the hub created for Self-Service Operations should be to provide a general orchestration and scheduling capability that can leverage and standardize automation written in any language or built in any legacy tool (as long as you can get API or CLI access to those legacy tools).

Role-based access control is a critical part of the Self-Service Operations hub. All enterprises have the need to enforce strict security and governance requirements. Operations needs the capability to delegate access to other teams who don't traditionally participate in operations activity. The ability to grant read, write, and execute permissions based on roles is essential.

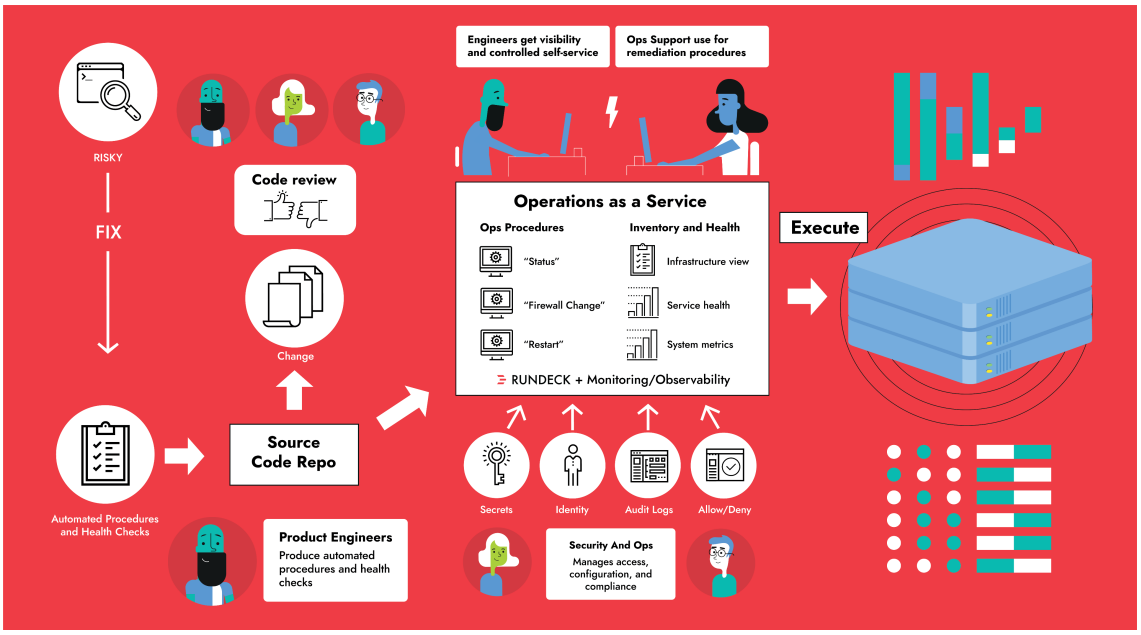
The person executing operations procedures will generally need to understand the context within which they are executing those procedures. The most important parts of that context are generally the current status and current configuration of the system on which they are working. Current status can generally be found in your existing monitoring and metrics tools. Current configuration can be more difficult to determine if a live-updated CMDB doesn't exist. However, following more modern practices there are often configuration templates and configuration management tools that have this information (and newer systems may already expose this data to inspection via API).

When people are experienced with the services and systems being managed, it is not difficult for them to work from multiple "screens" — looking to different places to get monitoring or configuration data and then going to a different tool to take action. However, when people are less experienced with a particular service or environment (which is bound to happen as you expand self-service), there is benefit to bringing those views together. Meaning, a person would see, all in one place, the monitoring and configuration context as well as the actions they can take.

Either as a prerequisite or a parallel step, work with your peers to define a basic set of standard procedures and a shared

“vocabulary” around actions that can be taken on each environment and system. An example of this would be to say that everything needs a basic set of actions — start, stop, status, configure, update, reset connections, etc. The convention you establish will provide slots that the responsible parties can fill in, and expand upon, with their favorite scripts and tools. This creates a consistent baseline of expectations for both those who created the automated procedures and those who will be executing the automated procedures.

Step #2: Collaborate to define operating procedures



One of the great advances of the DevOps movement is bringing Software Development Lifecyle (SDLC) discipline to operations work. Since everything above the hardware is software, why not treat it as such and use all of the established software

management best practices that we can. These include using versioned source control, having an automated “build” process, using a well-defined promotion process to move code from one environment to the next (hopefully as immutable artifacts), and regular code reviews.

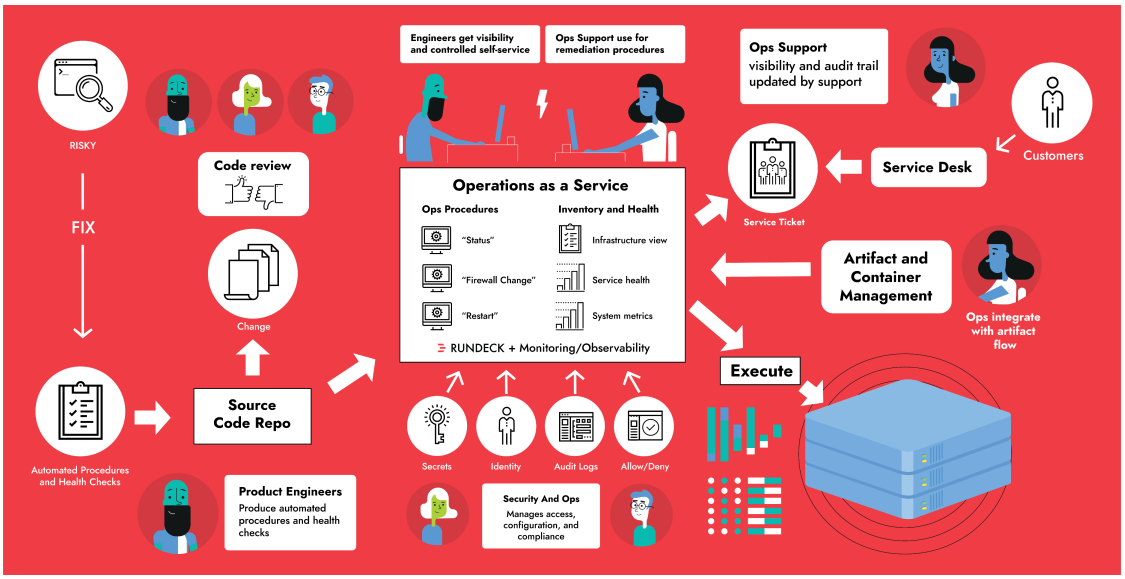
After teams establish their Self-Service Operations platform, the next step is to setup an SDLC that Development and Operations teams can collaborate through to define standard operating procedures. Developers will define the procedures to manage the systems they create and Operations and Security will vet those procedures and perform code reviews. Approved procedures can be tested in lower environments then promoted to production where they can be used by anyone to whom Operations and Security has delegated access.

The primary benefit is that you can leverage the knowledge of Development teams. They are closest to creating the system and are most qualified to determine if it is healthy and how to operate it. By getting them involved early, they can create and test the automated procedures when they are easiest and cheapest to create. This also speeds any handoffs from Development to Operations since Operations can focus on vetting and approving the procedures rather than creating them.

It should also be pointed out that in most enterprises, the total number of developers will be a lot larger than the total number of operations engineers. The only hope of relieving the capacity squeeze on Operations is to leverage all of that labor that can be found outside of Operations.

Operations teams who engage developers with this SDLC-driven approach see a rapid increase in both the usage of their Self-Service Operations hub and the overall delivery throughput of the organization.

Step #3: Integrate with other Enterprise Management tools



Now that the Self-Service Operations hub is up and running and the SDLC around the automated procedures is in use, the next step is to integrate with other enterprise management tools to make your Self-Service Operations capabilities even more intelligent.

For example, ITSM tools have a wealth of ticket and incident information and are key regulators of operations communication

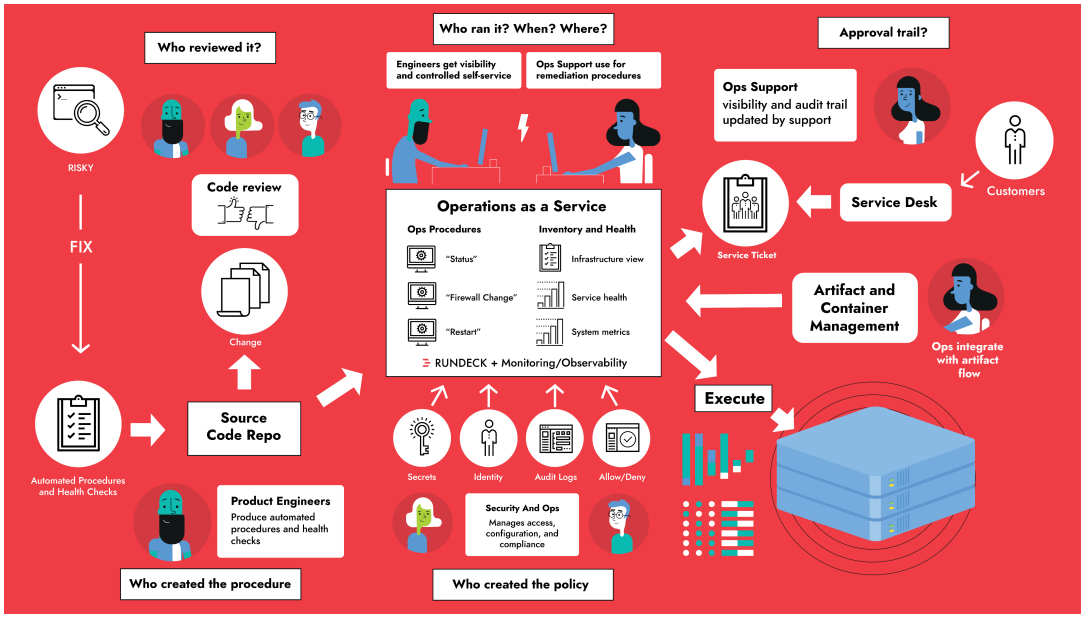
and work permission. You can have your procedures update tickets, check if tickets exist, open tickets on failure, and more.

Software artifact and container repositories will generally know which versions are available and marked approved. Through integration, you can have your automated procedures present users with options based on data from these repositories.

Chat systems are central to the new ChatOps style preferred by engineering and operations teams. Integrate with your ChatOps engines to call your automated procedures and see output within your ChatOps sessions.

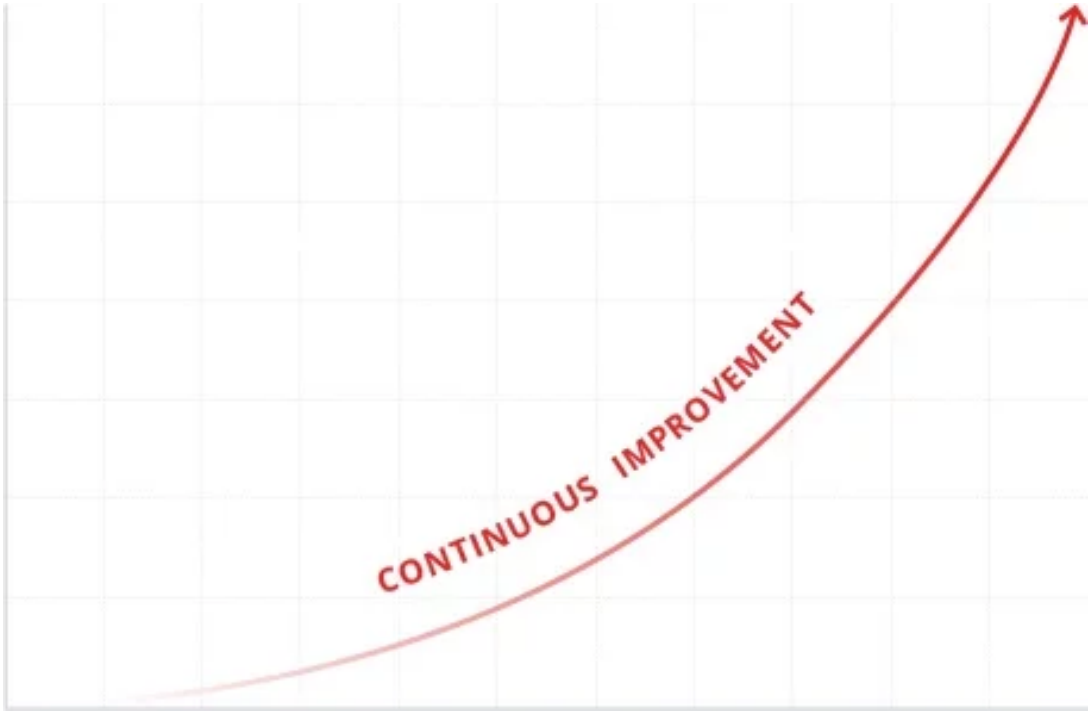
The more integrations you can create, the more context your users will have without leaving the Self-Service Operations hub.

Step #4: Make your auditors happy



While the idea opening up operations activity to a broader team may be initially concerning to traditional auditors, you can prove that your Self-Service Operations platform actually improves your compliance posture. With the platform in place you will have audit evidence automatically generated from start to finish. You can prove: who created the procedure, who reviewed it, what the approval trial was before it was run, who ran it, when it ran, where it ran, and exactly what ran. When you combine that with a platform that logs all events and an audit trail for access control policies -- your Self-Service Operations platform should be your auditor's best friend.

Favor continuous improvement over a "big bang"



A common trait that we see among high-performing companies is that they rollout their Self-Service Operations capabilities using a continuous improvement approach rather than a “big-bang” approach.

These high-performers will start with a limited set of procedures in a particular slice of their business before expanding. During the subsequent expansion, they add more procedures, expand to other parts of the business, and improve their capabilities along the way. They do this on a rolling basis to ensure that they catch any design flaws or user concerns as quickly as possible. The result is that they deliver a long series of quick wins that make everyone happy.

The continuous improvement mindset is also important given the platform nature of Self-Service Operations. A healthy platform is one that all parties can adapt quickly to their need. Recognizing that change is a constant in Operations, we are reminded to make tooling and design choices that favor ease of use, low learning curves, and a rapid delivery lifecycle.

Spot and Fix Anti-Patterns for Quick Wins

One of easiest ways to get started is to look for repeated patterns that lead to interruptions, waiting, or other bad interations between teams. Use self-service operations as the countermeasure for eliminating these “anti-patterns”.

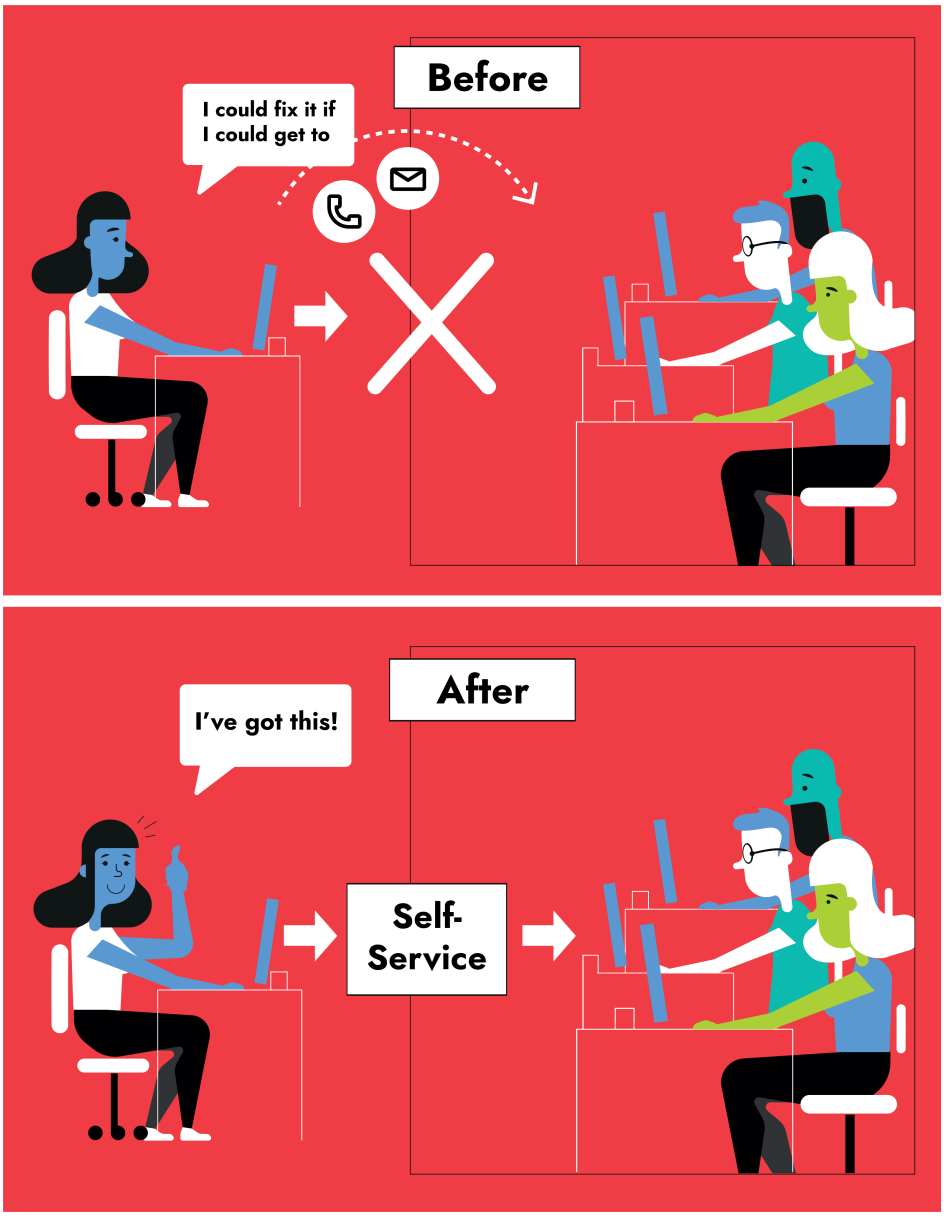
Every organization is different, but her are some common anti-paterns and countermeasures to get you started.

Anti-Pattern: “I could fix it. But I can’t access it.”

Who hasn’t had disjointed or disconnected access policies get in their way? Whether responding to an incident or delivering project work, you’ll find yourself blocked from the environments or infrastructure where you need to take action — even though you have the knowledge and experience to do what needs to be done.

This lack of access could be for any number of valid reasons. Sometimes, security or compliance concerns get in the way. Other times, the lack of access may be a byproduct of a siloed organization or political turf concerns.

In any case, the problem is the same: those who have the full context of the issue that needs to be solved (and usually urgently), aren’t able to take action. These engineers end up opening tickets or trying to find, by IM or phone, a colleague who can help them. This way of working adds additional task-switching, delays, and interruptions.



Instead, we can use a Self-Service Operations Platform to give engineers the access they need to get their job done quickly, effectively, and safely. Those who are experts can define the operational procedures that others can safely (and securely) execute.

Start by looking for repetitive tickets, or person-to-person requests, where the only value the person fielding the request adds is that they have access. Create self-service jobs to replace those requests and use the Self-Service Operations platform to provide fine-grained access to those who need it.

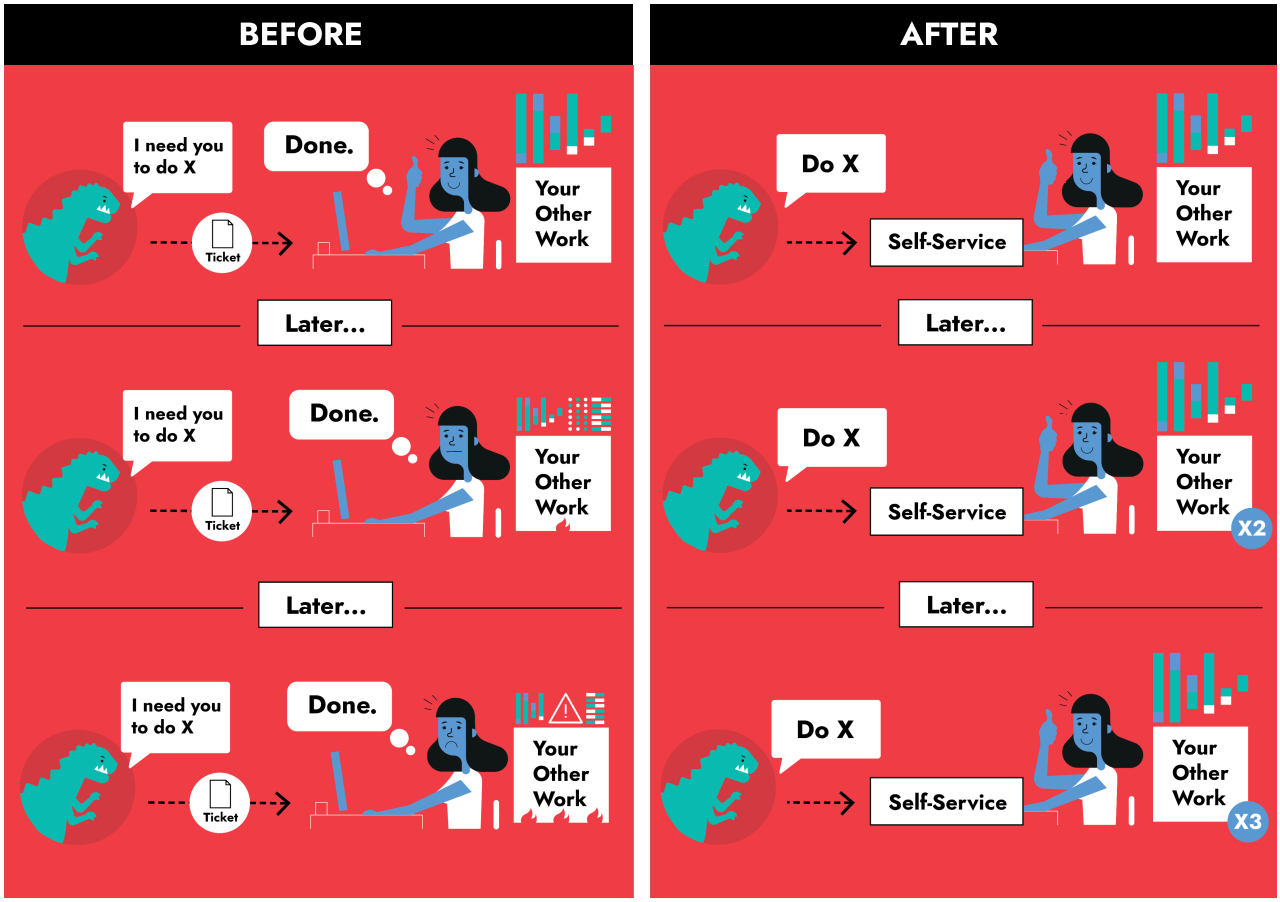
Anti-Pattern: “Do it. Do it again. Then do it again.”

Repetitive requests are one of the easiest types of wastes to spot. Whether you informally poll your colleagues or analyze your ticket system, you can quickly identify requests that are made over and over again.

Automating these requests provides obvious time savings. However, when you put that automation behind a self-service interface, the benefits grow exponentially.

For the person who formerly fielded the requests, they are spared the interruptions and expensive context switching that comes with repetitive requests. Interruptions don’t just eat up time. Interruptions also prevent the person fielding the requests from using their knowledge and experience to address other work.

For the person making the requests, self-service eliminates waiting. Waiting is expensive as it not only delays work, but it delays feedback. As we’ve seen in Lean, Agile, and DevOps, slower feedback leads to lower quality and higher risk.



Start by looking for repetitive tickets. You can analyze your ticket system to get a list of most frequent repetitive requests. From there you can evaluate which of those requests are the most disruptive and which are the easiest to automate.

Next, have the persons who are most familiar with fulfilling those requests collaborate on the scripts to create an automated response. Finally, plug those scripts into your Self-Service Operations platform and give requesters access. If you use the right Self-Service Operations platform, you can quickly turn the simplest of scripts into powerful self-service.

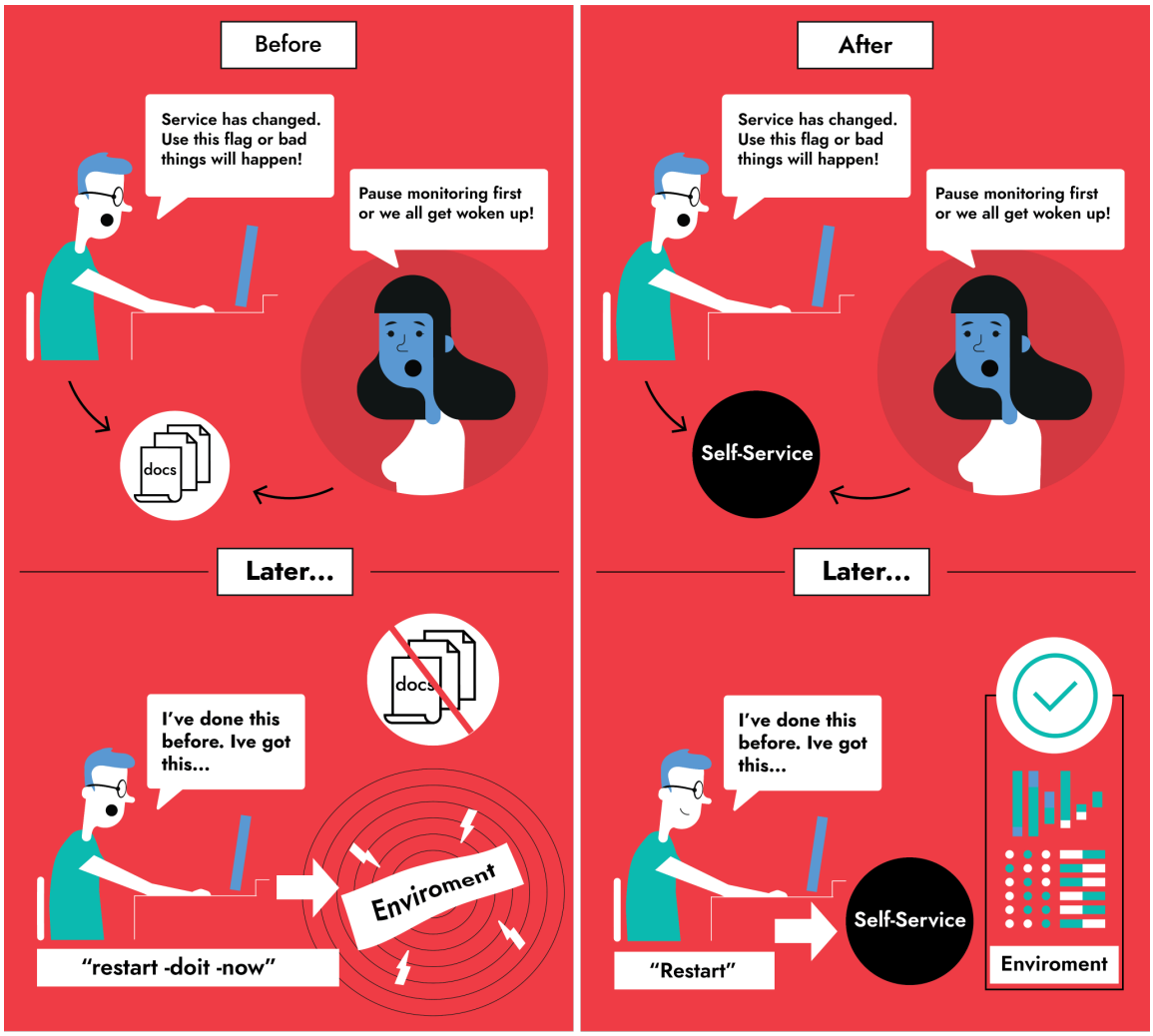
Anti-Pattern: “I’m an expert, I don’t check the wiki.”

Change is a constant in enterprise operations. Services, configurations, and underlying infrastructure are continually changing. Procedures are frequently updated. How do you effectively convey those changes so that mistakes aren’t made?

Documentation is an often suggested answer. However, written documentation has one substantial and often overlooked shortcoming — it is difficult to get people to read it!

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.

Also working against 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.



Communicating through code and automated procedures is much more effective. Rather than hoping someone reads the documentation and interprets it correctly, you have the certainty that the correct process will be executed. With Self-Service Operations, the infrastructure is there to quickly enable subject matter experts to define and update automated procedures as the need arises.

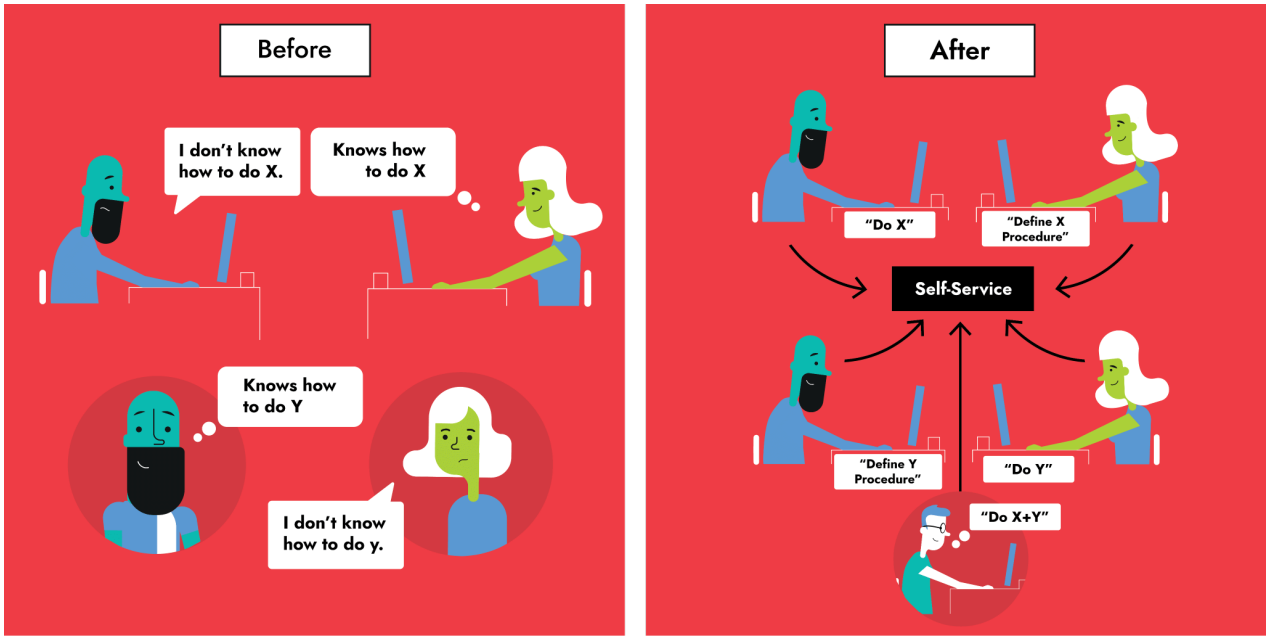
Now when it is 3 am, and you roll out of your bed to respond to that alert, you won't be asking yourself "Are these still the correct commands? Am I passing the right options to these scripts? Has anything changed?"

Anti-Pattern: "Siloed and disconnected knowledge"

Most operations work requires knowledge that spans multiple functional specialties — application, database, operating system, networking, storage, and so on. However, the larger the organization, the more likely it is that much of that knowledge resides with various functional specialists.

These specialists know, for their area of responsibility, the particulars of the current state and the correct procedures to manipulate that state. Distributing that knowledge is a significant challenge. The "everyone knows everything" strategy only works in the very smallest of organizations (and even then, its efficacy is suspect).

A team delivering project work or responding to an incident requires the knowledge of how to manage the full system. Siloed and disconnected knowledge can lead to botj mistakes and the overuse of ticket queues (i.e., "please do this for me" requests).



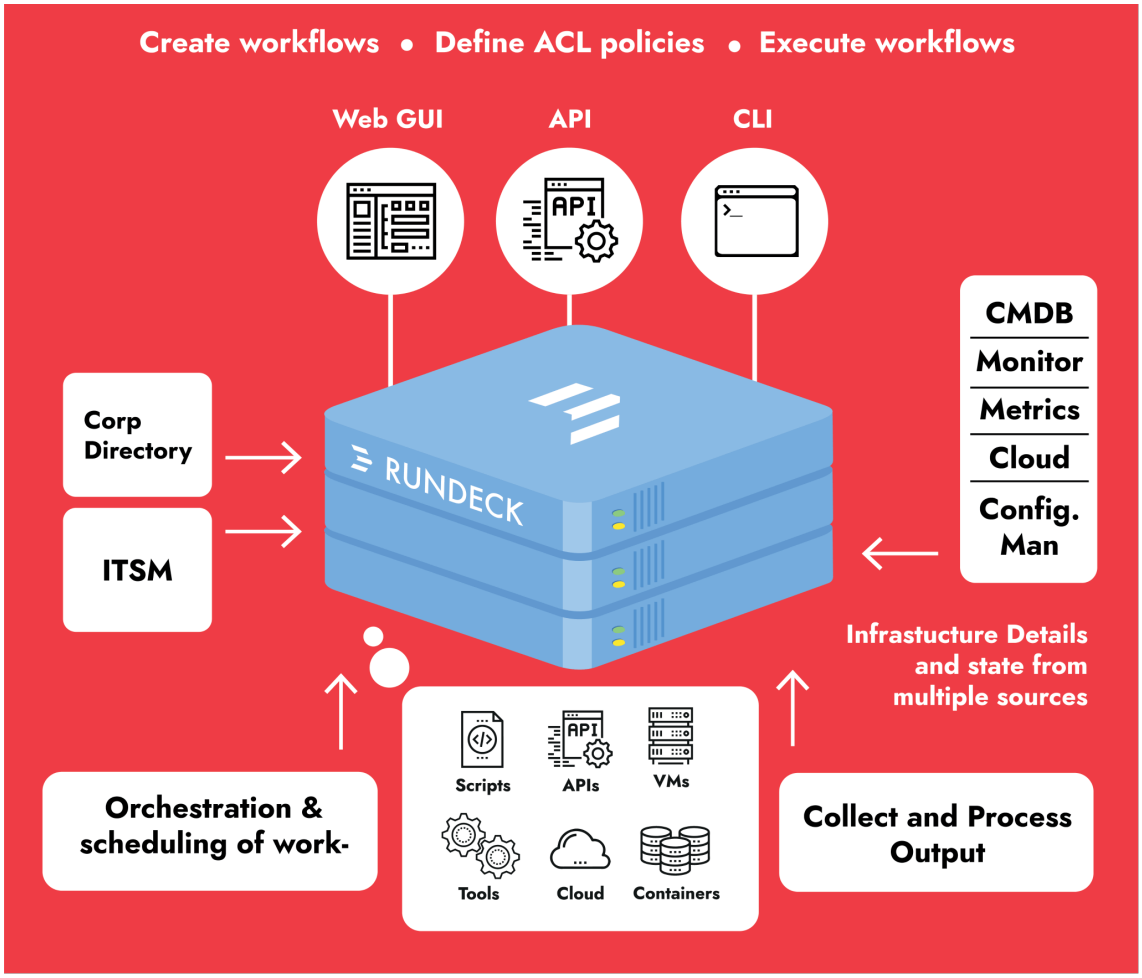
Self-Service Operations enables an organization to connect those who need to do a task with the procedures created and vetted by those who are experts in executing the task. This use of Self-Service Operations allows organizations to move quickly and commit fewer mistakes.

This use of Self-Service Operations also allows organizations to maximize their workforce. If more people can participate in Ops activity, then the workload can be spread more evenly. Self-Service Operations also helps junior team members become productive quicker and protects the organization against knowledge loss.

How Rundeck can Help

Rundeck is ready to be the core of your Self-Service Operations platform. From orchestration and scheduling capabilities to access

control management — Rundeck checks off the boxes of the capabilities you will need to get started.



Rundeck has been built under the motto, “Go fast. Be flexible. Lock things down.” Let’s look more closely at how the Rundeck platform addresses each of these ideals.

Go Fast

Current business and technology trends (e.g. Digital, DevOps) are causing a sharp increase in pace and flexibility demanded of IT.

However, the complexity found within most companies gets in the way of these desires. Rundeck is designed to work equally as well across the “old” and the “new”. Rundeck lets you connect users, tools, scripts, and APIs across any generation of technology. Whether it’s traditional siloed operations of legacy systems or fast and decentralized operations of new Cloud Native systems, Rundeck allows your teams work they need to work. Rundeck lets you move faster as an organization by helping you improve the effectiveness of your operations work and also by enabling self-service operations wherever possible.

Be Flexible

In order to respond quickly as an organization, the ability to safely delegate control to those closest to the need is vital. Whether the need is to respond to a production incident or manage environments during a delivery lifecycle, Rundeck enables companies to be more flexible and empower as many people as needed to do operational tasks. The more operational control that can be delegated, the more flexible and nimble an organization can be.

Lock things down

“Be secure! Don’t be the next data breach! Don’t be the next major outage!” is the strongest business mandate — and one of the greatest challenges — for today’s enterprise IT organizations. If the mandate is to lock things down, how can you meet the other business mandate of going faster? Rundeck approaches this challenge from two different vectors. First, Rundeck enables Ops to implement granular access and compliance controls. Rundeck

provide a control layer across all of your operational activity. Second, as the hub of IT operations, Rundeck greatly increases visibility into all operations activity. All actions are logged and create a permanent record.

Want to know more about how Rundeck helps Self-Service Operations? Get the white paper.

Conclusion

Self-Service Operations is a critical capability that needs to be developed by any enterprise IT operations organization. The speed, flexibility, and security controls dictated by today's business demands can't be met with the old practices that Operations has relied on for decades. Self-Service Operations allows you to distribute and align operations activity so that you can unlock the full potential of your people and move as fast as your business demands. Self-Service Operations is a straightforward, yet powerful, design pattern that should be in every IT leader's playbook.

Ready to get started? Rundeck can give you a big headstart with your Self-Service Operations journey.

About the Authors

Damon Edwards

Damon Edwards has spent the past 15 years working with both the technology and business ends of IT operations. Damon is noted for being a leader in porting cutting-edge DevOps techniques to large enterprise organizations. Damon is a Co-Founder and Chief Product Officer of Rundeck, Inc., the makers of Rundeck, the popular Self-Service Operations platform. Damon Edwards was previously a Managing Partner at DTO Solutions, a DevOps and IT Operations improvement consultancy. Damon is also a frequent conference speaker and writer who focuses on DevOps and operations improvement topics.

Alex Honor

Alex Honor has spent his 25 year career building automation systems in order to improve how enterprises operate. Alex is the founder of the Rundeck open source project and CEO at Rundeck, Inc.. Previously, Alex was a highly sought after consultant designing automation systems for companies such as Zynga, Dunn & Bradstreet, LinkedIn, Intuit, Adobe, and WebEx. Alex was the head of architecture and system engineering at E*TRADE and an engineer at NASA Ames.