CHAPTER 10

DevOps Finetuning

Additional Considerations, Concepts, and Practices

Congratulations, you've read this far and are ready to floor the DevOps accelerator. Like a Formula 1 driver in pole-position at the start of a race, you're eager to get the green light and hit the gas. Hopefully, you're part of a team with a winning culture, managing to business outcomes, and building the strategies needed for continuous improvement.

But this is no time to be complacent. Like Formula circuits and race day conditions, business changes constantly. Just as the Monaco circuit demands a different race strategy than Monza, DevOps programs must adjust to different circumstances. For a business focused on operational efficiency (doing more of the same, only faster and cheaper), or government agencies with shrinking budgets, Lean aspects of DevOps might take precedence. But if a business is adjusting its operating model (doing the same things, but in radically different ways), agile methods and continuous delivery will come more to the fore. Whatever the case, be prepared for change and never underestimate the importance of strong culture and building and cultivating great teams.

A. Ravichandran et al., *DevOps for Digital Leaders*, DOI 10.1007/978-1-4842-1842-6_10

Cultural Recalibration

Formula driver Daniel Ricciardo drove brilliantly at Monaco in 2016, but when his pit-crew wasn't ready for a tire change, he lost precious seconds and the race. And while ten seconds or so doesn't seem like much, it made the difference between Ricciardo spraying champagne and finishing a miserable second.

Incidents like these reaffirm that people make mistakes all the time—it's what makes us human. Even a finely-tuned DevOps program won't be immune to people-related issues and why culture can't be overstated.

Unlike process, culture can't be automated (not yet anyway), which is why organizations must constantly work on fine-tuning. Yes, this involves building a strong collaborative workforce, but also understanding what conditions exist that cause people to persist or revert back to behaviors counter to DevOps thinking-also known as "falling off the DevOps wagon".

The Normalization of Deviance

How many times have you witnessed a sub-optimal IT practice that everyone else thinks is okay? Then over time have you accepted the behavior and started practicing it too? We all have—it's quite normal.

Regardless of whether you lead a startup or work in an established business, we all have a tendency to accept suspect behaviors. Even if outsiders see them as wrong, our IT teams are so accustomed to using them (without any adverse consequences) that they're quickly established as "normal" and accepted.

Studies into what's commonly referred to as the "normalization of deviance" have been conducted in areas from healthcare to aerospace, with evidence showing that many serious errors and disasters occur because established standards have been bypassed and bad practices "normalized".

While examining this phenomena is critical in the context of safety, it's equally applicable in how we develop, secure, and operate software applications. With the boundaries blurred between the digital and physical world, any adverse behavior leading to security and reliability issues could have dire consequences. And when software becomes infused into long-lasting products (from light bulbs to limousines), it's not so easy to discretely exit markets.

As businesses increasingly rely on software innovation for market expansion, faster time-to-market and a high-quality customer experience become essential differentiators. Unfortunately, both can be compromised if rigid change controls or "speed at all cost" mandates lead to a bad behaviors.

Interestingly, guidance from other fields can help IT identify and eliminate poor practices. In the healthcare field, for example, studies have identified seven factors that lead to a normalization of deviance.[1]

All of these are extremely relatable to IT.

The rules are stupid, dumb and inefficient—In healthcare, accidents can occur when medical practitioners disable equipment warning systems because alarms are seen as distracting. This happens in all the time in IT with dire consequence. Like when IT operations staff miss major problems because they filter out noise and alerts on monitoring consoles they regard as irrelevant. Or when testing is deliberately skipped because of lengthy manual processing and provisioning delays.

Knowledge is imperfect and uneven—Employees might not know that a rule exists, or they might be taught a practice not realizing it's sub-optimal. In IT this persists because many new employees feel uncomfortable asking for help, or the application of new technologies distorts logical thinking.

The work itself, along with new technology, disrupts work behaviors—To support goals of more continuous software delivery, organizations are introducing many new technologies and methods, such as microservices and containers. Along with the new tech, new work practices and learning demands may lead staff to poorly implement the technology or use it to perform a function it was never designed for. For example, containerizing a legacy system just because it's "technically feasible".

We're breaking rules for the good of the business—Staff may bypass rules and good practice when they're incentivized on faster delivery times or delivering new functional software enhancements. For example, procuring additional (but unnecessary) hardware capacity to rush through an update, rather than addressing the root cause of persistent performance problems.

The rules don't apply to us…just trust us—Empowered agile teams are fantastic, but with great power comes great responsibility. Teams might slam through more code, but poorly governed open source access or bypassing compliance policies when generating test data can derail a program or lead to massive security breaches. Unfortunately in today's fast-paced digital business, talented professionals often feel completely justified in playing the "trust us, we know what we're doing" card.

Employees are afraid to speak up—Violations become normal when employees stay silent for fear of admonishment. How many times have bad software code practices been tolerated because junior staff is afraid to speak up? Even in IT organizations with a strong blameless culture, people can remain silent for fear of appearing "mean" to their colleagues.

[1] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2821100/

Leaders withhold or dilute findings on application problems— Whether you work in healthcare or IT, no one wants to look bad to managers. Rather than present ugly and unpleasant realities, many will distort the truth; presenting diluted or misleading information up the command chain. In IT, this behavior is easily normalized, especially if teams get away with reporting technical vanity metrics over business outcome-based performance indicators.

Tip Staff will never admit to anything they don't see as wrong. Get out of the office and "gemba," or walk your own software production line. Look for all the warning indicators that might be causing people to do the wrong things.

Combatting Sub-Optimal Behaviors

No sudden cultural reawakening across the IT organization will eliminate ingrained bad practices, but DevOps and Lean thinking can help identify the warning signals and can become the catalyst to drive behavioral improvement. This can involve:

- *Fostering open discussion*—It's easy to stand back and do nothing if a sub-optimal practice has yet to cause problems. But over time, poor behaviors become more difficult to remove, so catching early is critical. Ongoing education and open discussion can help with both development and operations participating. Some of this might be uncomfortable and peer-level critiques are never easy, but it's better than conducting blame games, post outage or major incident.
- *Managing to outcomes*—People are reluctant to change behaviors if they perceive the probability of a major incident or outage to be low. Part of this is normal human behavior, but can be exacerbated when the staff downgrades the likelihood of problems eventuating because diagnostics point only to minor technology issues. Therefore, it's important for staff to clearly understand the business impact (worst case scenarios) of poor practices.
- *Finding visual clues*—This includes visualizing the flow of value delivered by software applications and pinpointing all the bottlenecks and constraints. Analogous to unsafe stepping stones across a stream, these are the value interrupts. When examined, they reveal all the process and

technology issues causing staff to do the wrong things. Immediate candidates are software release and testing processes, but analysis shouldn't be restricted to development. Everything from enterprise architecture, stakeholder engagement, and information security, to vendor management and customer support can be a choke-point.

DevOps and Talent Management

While DevOps promotes the collaboration of software development and other IT groups, there is nothing stopping other departments across the organization from becoming involved in the program. Even the HR team can be included, especially with regards to talent acquisition and development.

In the application economy, demand for technology professionals in areas such as IT architecture, big data/analytics, and cloud computing is growing fast. Having DevOps skills is also becoming financially attractive. As was indicated in a 2015 survey from the cybersecurity firm Incapsula, which revealed that high school graduates in entry-level jobs requiring DevOps skills are seeing a median starting salary of $106,000.[2] But in advanced economies, demand is outstripping supply. In the UK, for example, the *Tech Nation* report, from Tech City and innovation charity Nesta, indicated that 40 percent of digital entrepreneurs say they face challenges finding skilled digital workers.[3]

The source of skills is also changing. The *Tech Nation* report also revealed informal or in-house training as the most common source of skills, with traditional universities ranking relatively low in IT activities. This is in no small part due to easier access and cheaper technology. For example, and as of July 2016, GitHub (the open source development platform) has 15 million people collaborating across 38 million repositories, while the credit card-sized raspberry PI processor has reached 5 million sales.

So You Think You Can DevOps?

With major supply challenges, HR can work collaboratively with IT to enrich the organization with much-needed DevOps skills. This will involve sourcing by traditional mechanisms (e.g., online job boards), but since DevOps requires no formal qualifications or certification, careful attention during candidate screening and selection is needed.

[2]DevOps Salary Survey, 2015: http://lp.incapsula.com/rs/incapsulainc/images/Report%20-%20DevOps%20Salary%20Survey%202015.pdf
[3]Tech Nation, 2016: http://www.nesta.org.uk/sites/default/files/tech_nation_2016_report.pdf

Ideally, DevOps practitioner should exhibit the following skills:

- *People*—DevOps practitioners need to be good communicators, which is a skill that's not normally associated with introverted technologists. Look for people who can exhibit evidence of written and verbal communication—within, across, and external to organizations. For example, has the candidate presented at conferences or tech meet-ups?

Note HR can help IT develop good communication skills through the introduction of formal courseware and "lunch and learn" sessions.

  Careful consideration should be given to candidates who have experience working with business stakeholders and can clearly articulate strategy, priorities, and goals. DevOps practitioners should be able to describe situations where they collaborated and shared information to achieve better business outcomes.

- *Process*—While it's possible for DevOps to work with Waterfall development, success is more likely in environments where agile methods have been applied. Strong candidates will demonstrate understanding and experience in one or more agile approaches (e.g., Kanban, Scrum, or SAFe).

  Value should also be placed on candidates who can demonstrate knowledge in Lean manufacturing concepts such as Just-In-Time and Theory of Constraints. Extra marks if they can illustrate examples of where and how they applied these principles in an IT context.

Note Working with IT managers, HR could introduce an online library of DevOps, agile, and Lean related books. Some great examples include *The Phoenix Project* and *The Goal* and *Lean Enterprise.*[4] Don't limit access to IT; make the resources available across the organization.

---

[4]*The Phoenix Project,* Gene Kim, Kevin Behr, George Spafford, ISBN 978-0988262591
The Goal: A Process of Ongoing Improvement, Eliyahu M. Goldratt, ISBN 978-0-88427-178-9
Lean Enterprise, Jez Humble, Joanne Molesky, Barry O'Reilly, ISBN 9781449368425

- *Technology*—DevOps practitioners will have acquired a range of skills across the software development lifecycle, including, but not limited to, agile project management, build management, and release automation. Strongly consider candidates who are experienced integrating these to build a fully automated DevOps toolchain.

■ **Note** Look for individuals who have worked in cross-functional teams and have broadened their skills by learning new technologies. Individuals who not only use technology but have contributed to its development are also good candidates.

Rather than present resumes, strong technologists will emphasize their achievements and expertise via actual contributions within the agile and DevOps community. This can include technology blogs, and open source contributions (code, scripts, and commentary). Ideally, sysadmin candidates will have coding skills across a variety of programming languages, while developers will be familiar with scripting methods.

Various Talent Management Hacks

Age shouldn't be a factor when selecting DevOps talent. Many of the best practitioners have years of experience adopting new technologies and practices. In the enterprise, this is especially valuable since many legacy systems will need to be integrated with cloud applications, which requires a comprehensive understanding of the inner workings of many technologies and all their limitations. That said, however, it'll be necessary to replenish DevOps programs with fresh talent. And since supply is problematic, HR can assist technology teams with innovative acquisition and development strategies. This could include the following.

IT Apprenticeship and Graduate Intake Programs

Many organizations have graduate intake programs, some of which don't require applicants to have formal IT qualifications. They encourage anyone to apply who has a passion for technology. One example is the ICT Apprenticeship program run by the Department of Finance in Australia.[5] After a series of interviews, successful applications are employed by a sponsoring government agency. Upon assignment, they combine four days of work with one day of formal IT study, leading to certification.

---

[5] http://www.australia.gov.au/information-and-services/jobs-and-workplace/australian-government-jobs/ict-apprenticeship-programme

Such programs are a great way to infuse talent within an organization, but there are challenges that HR teams must address. Often in these types of initiatives, apprentices are rotated through different areas of IT—the aim being to familiarize new staff with all aspects of the IT business. This is fine in theory, but in a DevOps context (where cultural improvement is so important), it can actually be counterproductive. If, for example, the organization is functionally siloed, divisive, and averse to change, apprentices could become isolated, and rotating passionate and eager technologists through highly process-centric areas can be the fastest way to dampen their spirit and enthusiasm.

Tip Consider organizing graduates or IT apprentices in self-managing teams or "pods" with full program oversight by experienced leaders and mentors. Always assign work that is meaningful, valuable, and measurable. Stagger the arrival of new intakes and place them in the same workplace as existing apprentices.

DevOps Hackathons and Coding Days

Hackathons are short events where all kinds of specialists come together to work intensively on small technology projects. Hackathons can be run as a contest (e.g., prize for best mobile app innovation), hosted internally or externally, or purely social. In some cases, they even have the lofty goal of creating commercially viable software solutions.

With DevOps, hackathons can be valuable for instilling the value of collaboration and teamwork within the IT organization. HR can take the lead in helping to organize and arrange these events, but should work with IT leadership to ensure maximum DevOps benefit. For example, conducting the hackathon away from the workplace where participants could be distracted. Other factors to consider include:

- *Balancing teams*—Including developers, sysadmins and project managers, enterprise architects, and security specialists.
- *Providing broader deliverables*—Don't restrict hackathons to new apps. Look for other opportunities to improve quality. For example, give prizes for the fastest way to find the root cause of a complex application performance problem; the most innovative application of security to increase resilience; and the best API to securely expose enterprise data.

- *Managing expectations*—Don't expect clean and robust code from a short hackathon; you'll get alpha at best. Consider longer online developer challenges if the goal is to fuel business with viable business prototypes (e.g., integrating valuable enterprise data with third-party systems).

Involving partner developers in a hackathon is a great opportunity to nurture innovation but requires additional oversight and support. Providing badly formatted open data without metadata, or poorly documented and unmanaged APIs, should be avoided. When involving third-party developers, consider methods to simplify API discovery, registration, support, and engagement, including providing an API catalog, sample code, and mobile device SDKs.

DevOps and the Internet of Things

At the 1990 Interop computer conference, two guys connected a kitchen toaster to the Internet. This early attempt at appliance networking paved the way for what's now called the *Internet of Things* (IoT).[6]

As of 2016, it's estimated there could be more than 10 billion Internet-connected elements, ranging from watches and cars to just about anything in the "thing" category, including light bulbs, fitness wearables, fridges, and, yes, even pets and diapers.

But beyond the big numbers, marketing spin, and hype, the business potential of IoT is being increasingly exploited by early movers. Perhaps not surprisingly, this includes some major businesses and brands.

Bosch, for example, the German multinational engineering and electronics company, announced an IoT cloud. This is the latest initiative as the company transitions from traditional industrial company to a blended manufacturer and technology provider.[7]

While connected home and car use cases grab our attention, many industries are exploiting standards, unified architecture (IT and industrial), and innovative software solutions to connect device and sensors to IT systems.

Take for example manufacturing. When plant production systems can be integrated with customer demand analytics, maintenance windows can be optimized. Similarly in energy and utilities, integrating smart meters with back-end applications and analytics could become a system to predict energy demand across the grid.

---

[6] http://www.livinginternet.com/i/ia_myths_toast.htm
[7] http://www.wsj.com/articles/robert-bosch-launches-own-cloud-for-internet-of-things-1457528014

This convergence will require new approaches to developing and operating applications, and DevOps with its focus on shortening delivery cycles while improving software quality is absolutely essential. However, IoT project success depends on DevOps practitioners addressing some new challenges:

- *Bridging another cultural divide*—The development and operations divide is nothing compared to the cultural chasm spanning traditional IT and industrial operations. While IT speaks of Java, C++, NoSQL, and the IP stack, plant and industrial engineers talk of programmable logic controllers and supervisory control and data acquisition (SCADA).

■ **Note** Putting all IT "ego" aside, DevOps teams need to build a good understanding of IoT related terms, technologies, and limitations. To assist, try to involve product specialists (e.g., industrial engineers, product managers, and designers) in DevOps/IoT initiatives.

- *Solving complex security problems*—Connecting previously isolated physical devices to networks creates additional threat surfaces. At best this could mean service interruption, at worst physical injury or loss of life. This consequence was disturbingly illustrated in the Jeep security exploit, where hackers demonstrated wireless access to vehicle infotainment and controls.[8] These could be expensive IoT lessons in the automotive industry where fleets of new cars can leave a dealership containing up to 70 specialized computers and more than 20 million lines of code.

■ **Note** Considerable investment will be needed to address IoT security issues. DevOps and security teams must review entire IoT processes (device to cloud or back-end application), thus ensuring that embedded software applications and data can be verified and encrypted.

- *Managing privacy*—Internet privacy issues could be nothing compared to the challenges introduced by IoT. As the physical and digital world becomes blended, smart devices have the potential to track behavioral patterns everywhere—as we sleep, during exercise, when we drive, consume energy—you name it!

---

[8] https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

Tapping into this information means DevOps practitioners must pay careful attention to issues around data ownership, compliance, data access, and sharing arrangements.

■ **Note** As more customer data is obtained from IoT devices and consumed by complex analytical, and big-data systems, careful consideration to accurate and compliant data management (e.g., during testing) becomes more critical.

- *Guidance on IoT interoperability*—Many industrial operations and IoT systems work in isolation and use proprietary networks (e.g., Modbus and BACnet) and protocols. In many cases, what works in one industry vertical won't work in another. While this problem can't easily be addressed, DevOps teams involved in IoT projects must become knowledgeable on emerging IoT standards and interoperability initiatives. Examples include The Open Connectivity Foundation and the Allseen Alliance.[9]
- *Understanding physical device limitations*—Depending on business context, IoT devices characteristics and usage will impose many limitations on software development, testing, and operations. Not the least, constraints on what programming languages can be used for IoT development and strict limitations on runtime size. Since software won't be operating in the confines of a 24/7 data center, there'll also be design issues to factor in (e.g., device battery life, environmental/weather conditions). Consider too, challenges making software updates across sub-optimal networks, the mass migration of sensor data to the cloud for analysis, and determining embedded software security risks (especially in OEM devices and third-party components).

DevOps for the Public Sector

You can be forgiven for thinking that DevOps only works in the private sector. It's simply not true.

Governments across the world continue to search for better ways to deliver citizen services through new technologies and practices. Agile and DevOps are gaining traction, with federal U.S. departments, such as U.S. Citizenship and Immigration Services (USCIS) and the Environmental Protection Agency, becoming adopters.

---

[9] https://allseenalliance.org/https://openconnectivity.org/

But adoption isn't limited to the United States. In Australia, the recently formed Digital Transformation Office (DTO) leverages agile methods and DevOps principles to support its mission of leading the digital transformation of government services.[10]

But as of 2016, the public sector hasn't fully embraced DevOps thinking. This was well illustrated in a Vanson Bourne study, which indicated that public sector organizations are a third less likely to adopt DevOps compared to their private sector counterparts.[11]

Yet those who are starting a DevOps journey are seeing benefits. As indicated in the study, more than one-third of public sector DevOps adopters have experienced application quality improvements, while almost half have seen reductions in IT spend. This last point is especially pertinent for budget-constrained government agencies.

DevOps and Lean to Improve Government IT Outcomes

Government agencies are under increased scrutiny to ensure money is spent wisely. Unlike their private sector counterparts, who can justify mega IT spending as the means to increase revenue and market share, government agencies can only think in terms citizen-centric service improvement, which given the overarching political context and climate, might not be a top priority.

Even when budget is available, the track record in government IT delivery hasn't been great. According to Standish Group, just over 94 percent of all large government IT projects (2003-2012) were over budget, behind schedule, fell short of user expectations, or had to be abandoned completely.[12]

With a 6 percent success record, it's perhaps not surprising that the DevOps mantra of "failing fast" doesn't fully resonate in government departments. Apart from the cost factor, no IT manager would want to front a congressional committee when critical government services have been disrupted.

With increasing budget constraints, probably the best place to start a DevOps program in a government agency is to focus on the Lean principles that underpin much of the thinking. By identifying all elements of "waste" that incur greater cost and negatively impact delivery, agencies can improve cost structure while speeding delivery. To this end, and before any investment is made in tools,

[10] https://www.dto.gov.au/
[11] https://www.ca.com/us/rewrite/articles/devops/research-report--devops-the-worst-kept-secret-to-winning-in-the-application-economy.register.html
[12] http://www.brookings.edu/blogs/techtank/posts/2015/08/25-yaraghi-government-it-projects

mature government agencies will carefully examine the entire end-to-end software delivery cycle and remove any cultural, process, or technology related inhibitors that add no value. The intention of course is to increase speed and quality, but a very important "by-product" for the budget constrained agency is cost reduction.

Many wasteful practices traditionally associated with manufacturing are applicable in a government software development context.

- *Long waits*—Some government agencies remain fixated on performing lengthy change review processes. But with new technologies now making it possible to deploy fully certified stacks to the cloud, only application changes need to be checked. This significantly reduces release times and administrative costs. Of course, there will be occasions when teams require access to production systems and data for testing, but cycle time can be shortened with technologies that virtualize or emulate dependent systems.
- *Excess inventory*—Over years, agencies have acquired a complex array of technology. These include mainframes, applications, and databases. Each servicing agency requirements in everything from massive transactional processing to data matching and analytics. In order to manage this complexity, agencies have become reliant on contractors who generally shun agile and DevOps concepts, preferring instead contracts with fully defined requirements and longer delivery cycles. To address this, agencies should immediately review the IT culture and skills gaps issues this causes, plus consider engaging nimbler agile and DevOps minded contractors via simpler procurement processes.
- *Slow motion*—As with large commercial enterprises, government agencies can be inhibited by bureaucracy and controls. As governments look to adopt DevOps, it's important to analyze every unproductive process across the software development lifecycle. A large part of the magic in DevOps comes from "doing rather than procrastinating," so agencies should start familiarizing themselves with smaller and iterative approaches to software development combined with fully automated release processes—a state where everything moves with purpose.

Even in situations where rigorous compliance testing is seen as a tough but necessary control point, DevOps and automation can pay dividends. By encapsulating testing within development for example, DevOps practitioners can enact the necessary speed and quality improvements, while demonstrating reduced costs in auditing, risk management, and compliance. This is again very attractive to cash-constrained government departments.

- *Untapped skills*—When things are slow and hard to change, everything atrophies, including staff and their skills. If DevOps is to help slow moving government agencies, it's important to identify all issues that prevent change. Over years many staff will have become accustomed to working in a certain way, with incentives linked to these practices. It's important to maximize the motivational impact of new DevOps programs, while also ensuring staff to stay the course. This may involve changes in organizational structures, new talent acquisition strategies (discussed earlier), and the adoption of shared goals, incentives, and metrics.

Summary: The Seven Point Action Plan

As we've discussed in this book, DevOps aims to break down the traditional barriers between development and other IT groups. By unifying people, technology, and processes across a modern software factory, new applications and updates can be built, tested, and deployed more quickly. Together with continuous feedback, businesses can enhance customer experiences, accelerate time to value, and better leverage IT as a competitive differentiator.

But while interest in DevOps continues to grow, many implementations can fall short of expectations or stall completely. In part, and as we've described, this is because DevOps challenges conventional IT wisdom—placing less emphasis on process rigor and more on removing organizational barriers and continuous adjustment.

As such, DevOps doesn't come pre-packaged with implementation blueprints; however, there are steps any organization (irrespective of DevOps maturity) can follow to optimize a DevOps initiative and we present these as a summary.

Understand the Business Goal

A program like DevOps is going take lots of work. It's great to get carried away with the "DevOps buzz," but DevOps comes with a price tag, which includes new tools and processes to learn, people to cajole (and console), and teams to restructure. As with any business decision, all this cost and work must be carefully weighed against the expected benefits.

Doing DevOps for the sake of DevOps is no way to start. Fall into this trap and customer value and business benefits take a backseat and become less important than protracted technology and process discussions. That's a DevOps disaster before you've even started.

It's important therefore to get business and IT on the same DevOps page. Focusing first on the needs of the business and then aligning DevOps, including metrics, tools, and processes.

By focusing first and foremost on the needs of the business and clearly articulating concrete goals across IT, you'll ensure a consistent approach and avoid misinterpretations or DevOps misfires.

As we like to say—don't fall in love with DevOps, fall in love with the business problem and let DevOps help you solve it!

Cultivate Senior Level Sponsors

If you're going to be changing things like organizational structures, work practices, and incentives, you'll need some support. In large organizations, change takes time, so seek out a sponsor to facilitate decisions and make the tough calls.

Executive level sponsors add substance to the DevOps sauce. They give teams a sense that there's real purpose behind the initiative and that it's not another process "wild goose chase". Remember too that senior managers are great at PR, which is very useful when you want to spread the good word about DevOps, build momentum, and gain more support.

Note There's no such thing as a "free lunch". Executive sponsors expect real returns from DevOps, so try to make sure the goals being set can help them meet theirs.

Select Your Pit-Crew

People will make or break a DevOps initiative, so choosing the right staff is important. There are no hard and fast rules, but seek out employees who can:

- *Work well in teams*—Look for people who have demonstrated a willingness to collaborate across organizational boundaries.

Caution Be careful building a dedicated "DevOps team," as this might become another silo. However, recognize that temporary teaming could be a good way of seeding DevOps goodness across the wider organization.

- *Show resilience and flexibility*—Great DevOps practitioners quickly learn from failure and embrace new ways of thinking to challenge the status quo. With developers for example, this can involve taking full ownership of production rollouts, not just the code they're working on.
- *Empathize with colleagues*—Seek out staff across the organization who consistently place themselves in the "shoes" of their colleagues. If a developer takes time to work with on-call staff to determine where supportability improvements are needed then they're a great candidate.

Note Generalists are preferred over specialists, but strong consideration should be given to staff who have experience with newer practices, including continuous integration and testing, release automation, and toolset integrations.

Revisit Chapter 3 and the section on DevOps culture for more pointers.

Choose an Immediate Deliverable

Select an application that's small enough to remain manageable with new DevOps approaches, but has enough business visibility to stimulate wider adoption. There's nothing to stop applying a DevOps approach to any type of application, but it's generally better suited to mobile and web facing applications, which are architected to take advantage of aspects of DevOps tooling described in this book, including API management, continuous delivery, and "shift-left" monitoring.

Legacy applications and systems-of-record can also benefit hugely from DevOps and shouldn't be ignored. Newer customer-facing mobile apps may need to integrate with back-end applications hosted on mainframes providing essential horsepower. Unfortunately, the practice of managing mainframes in silos can severely impact software development and adversely impact customer experience.

Take, for example, a situation where a mobile development team needs access test data residing on a mainframe or insight into the performance implications of newly released code. These teams must have fast unconstrained access to this data to meet schedules together with end-to-end performance insights—mobile to mainframe. If teams work in silos using point tools, this will be difficult to achieve.

Many advanced tools, together with in agile-style methods, can help align mainframe management with a DevOps initiative. This includes:

- *Extreme performance visibility*—These tools provide deep visibility into complex interactions and transactions across mobile, web, and mainframe applications. They help teams quickly understand and troubleshoot even the oldest and most complex code bases.
- *Constraint removal*—By simulating mainframe infrastructure dependencies and providing synthetic test data, developers and testers can maintain the pace of DevOps delivery without compromising quality or compliance goals.
- *Unified infrastructure management*—Enables mainframe and non-mainframe experts alike to better understand infrastructure interdependencies, thereby detecting and fixing problems faster and without the need for specialist tools.
- *Extensible analytics*—By extending analytics into areas such as *workload automation*, teams can perform real-time forecasting of critical back-end services. Using advanced statistical methods to calculate job duration, administrators can fine-tune workload schedules and processing to cater to changing conditions.
- *Exposing rich data sources*—Allows developers to rapidly create application back-ends for internal applications, mobile apps, standalone microservices, data as a service, and partner integrations. These systems can help decompose large mainframe applications into self-contained units with everything needed for app delivery, including data integration, business logic, and a robust API layer.

Revisit Chapters 4-7, where these capabilities are described in greater detail. Always consider that tooling shouldn't be limited to managing one particular platform or supporting a single group.

Build a Comprehensive Metrics Program

Traditionally, and as we described in Chapter 3, goals and objectives have been set at a departmental level, with methods for measuring IT capability skewed toward resolving problems and rewarding the "heroes" who fix them. With DevOps, objectives should be set at a "singular team" level and aligned to the desired business end-state—digital transformation via faster release of high-quality software.

With this in mind, work collaboratively to build shared objectives and metrics aligned to business objectives. Avoid vanity metrics that distort the truth or incentives that reward siloed behaviors. Next, set some improvement targets and assess the capability of existing tools in meeting and surpassing your goals.

■ **Tip** Just because a target is constantly being achieved doesn't mean improvements can't be made. Don't get complacent. Just like code, DevOps metrics need constant care and attention!

Integrate DevOps with Existing Practices

As we discussed in Chapter 8, DevOps doesn't preclude leveraging other methodologies and best practices, including ITIL.

To avoid methodology "turf wars," it'll take teams working collaboratively to identify where existing processes need to refining to cater to DevOps-style delivery. Obvious places to start are with existing release and change management, but be prepared for "give and take" on both sides. Never forget that DevOps itself can benefit from well-established processes, such as incident and problem management to strengthen feedback loops and accumulate organizational knowledge.

■ **Tip** Consider establishing a single workflow-based communication process between developers and other groups, including support staff. Integrate these with monitoring, ticketing, and help desk solutions to make problems and solutions visible across the entire organization.

To be fully effective, DevOps practitioners also need to work with many other stakeholders, including enterprise architects and security. Once again, and using the guidance described in Chapter 8, be prepared to adjust existing methods and take other perspectives on board.

Automate the Pipeline; Strengthen the Toolchain

Reexamine the "eight elements of waste" introduced in Chapter 3. It's a useful framework to better identify all the existing process and technology constraints across your own software factory.

While pushing code all the way to production via fully automated releases might be the holy grail of DevOps, it could be better to start by reviewing those areas experiencing the greatest pain or where integrated toolsets can

help fill in the release pipeline "whitespace" (for examples, revisit the release automation integration use cases illustrated in Chapter 6 and the automated testing trifecta presented in Chapter 5).

In other cases, reviewing waste and constraints may indicate that operational functions have the most to benefit from automation in the short term. This could include the adoption of modern configuration methods (infrastructure as code), or establishing performance monitoring and analytics to build critical bidirectional feedback loops between development and IT operations (for an illustrative example, loop back to case study and discussion on "shift-left" monitoring presented in Chapter 7).

Tip As in traditional manufacturing, many wasteful IT practices and constraints inhibit the flow of value/throughput. Identify the greatest painpoint and potential for human error, be it managing test data, accessing infrastructure dependencies, or conducting manual and error-prone handoffs.

In Salute of DevOps

In a world where customers and citizens expect so much more from technology, the race is on to develop a finely tuned high-performance IT function. Hopefully by reading this book you're in a better position to see how DevOps supports this imperative and can develop the strategies needed to reignite your business.

In Formula 1 racing, winning one race doesn't win a championship. It's the same in software development. DevOps teams understand this implicitly. They are smashing silos, collaborating fully, and leveraging advanced automated solutions to eke out business performance improvements everywhere, as they build, test, release, and manage….

…. superior high-quality software solutions, continuously delivered from a modern DevOps-enabled software factory.

….start your engines!