Define Your DevOps Action Plan

A WHITEPAPER FOR IT LEADERS

ITinvolve

People Powered IT™

www.ITinvolve.com

A WHITEPAPER FOR IT LEADERS

Define Your DevOps Action Plan

DevOps is being defined, promoted, touted and discussed as the next game changer in IT. This whitepaper will provide guidance on how you can determine what DevOps will mean for your organization and help you build an action plan to get started.

A common goal cited for DevOps is to enable faster release and deployment cycles by taking advantage of agile development methodologies; improved collaboration between business stakeholders, application development and operations teams; and automation tools that enable the continuous integration of developed code and continuous deployment into production of new capabilities. Beyond these elements, DevOps also requires a cultural acceptance of the need to focus on the flow of work across teams vs. optimizing individual teams and their specific units of work.

There is no "one-size-fits-all" DevOps, no matter what pundits, consultants, and vendors might tell you. It should be grounded in the realities of your specific organization. Start by actually taking a step back. Ensure you are clearly taking into account your industry, your applications, your culture, and your people when developing your DevOps strategy; then apply DevOps principles against that foundation. Because every organization's purpose will be different, your way of doing DevOps will not be the same as anyone else, even among your industry peers.

Begin With Your Applications

Defining your DevOps action plan starts with an understanding of your industry and your applications. One suggested method is to analyze your business goals and application portfolio using the pace-layering method, which focuses on three categories of applications – systems of record, systems of differentiation, and systems of innovation.

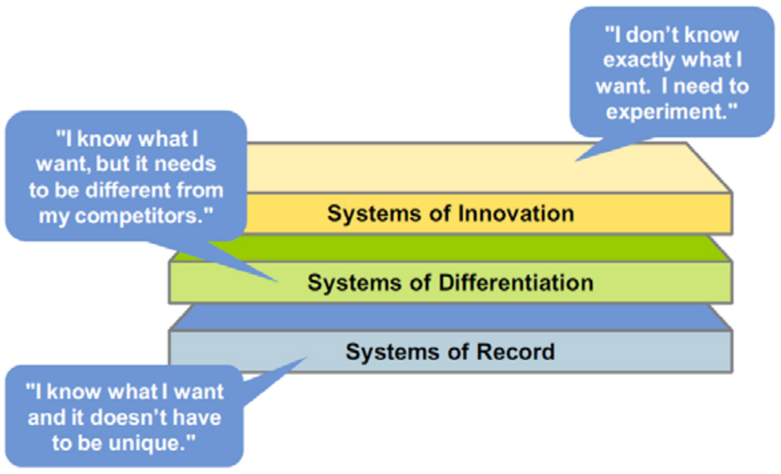
"I don't know exactly what I want. I need to experiment."
"I know what I want, but it needs to be different from my competitors."
Systems of Innovation
Systems of Differentiation
Systems of Record
"I know what I want and it doesn't have to be unique."

Figure 1
Gartner's Pace-Layered Application Strategy: Governance and Change Management"
15 April 2011, Bill Swanton

Systems of record are usually stable with infrequent updates. This is often due to regulatory requirements and internal policies, and they tend to have a high degree of consistency within an industry and even across industries. Systems of record are often good candidates for waterfall development methodologies and longer release timelines. Your general ledger or payroll systems are good examples of systems of record.

Systems of differentiation are those applications that are likely common to your industry but where you have an opportunity to differentiate your company from the competition based on functionality and the pace with which you update them. For example, if you are a financial services firm that provides retirement investment programs, you might make changes to your customer portal and how clients select new investment options on a quarterly or monthly basis in order to enhance customer experience and remain competitive and differentiate. If you are an airline, this might be your crew scheduling application, which ensures you have the right flight crews available when needed and can improve your on-time departure percentages to differentiate from the competition. In healthcare, your systems of differentiation might be the applications that health professionals use to store patient information and correlate test results to aid in diagnosis and treatment. In manufacturing it may be your supply chain management, process control, or shop floor applications.

Systems of innovation are those truly groundbreaking applications that help you create new markets and revenue streams. In financial services, that might be a new kind of application that gives day traders a market advantage. For an airline that might be a whole new way of providing inflight entertainment, such as when Wifi and inflight entertainment systems first became available to passengers. In healthcare, it might be a new customer portal that promotes wellness services and reduces office co-pays by uploading fitness tracker data. For a retailer, it might be applications that support a new type of customer loyalty program, such as when Amazon introduced their Amazon Prime service a few years ago.

Both systems of differentiation and systems of innovation are excellent candidates for agile development and DevOps principles. This is because they require high degrees of collaboration between business stakeholders, developers, and operations personnel to ensure the applications are in line with requirements, developed and tested quickly, and then made available in the market to drive competitive differentiation and innovation. They are excellent candidates for the continuous delivery concept in DevOps where small releases are deployed quickly (and can be rolled back just as quickly if there are issues). In this manner, a large number of small deployments moves the competitive needle faster and provides greater value to customers over traditional waterfall development and disconnected development and operations practices.

Foster a DevOps Culture

Today, we find more and more businesses putting pressure on IT to move faster, and, if the IT department struggles to keep up, they are often willing to "go around them" and invest in a cloud or other third-party solution on their own to get what they want, when they want it (aka "shadow IT").

What the business often lacks, however, is a full understanding of the many downstream implications that come with speeding the roll out of new services such as security assessments, integrations to legacy systems, the ongoing cost to administer the solution and take on support for it, etc. It's true that IT Operations teams want to control the reliability and stability of software deployments and that can often slow things down, but they do this because that's what they are paid to do: to deliver stable, reliable, and high-performing services.

This conflict between the business' desire for speed and Operation's charter for security, reliability, and performance, has been brewing for years and this section will discuss a few tips on how you can foster a more engaged and aligned culture as you define your DevOps action plan.

TIP #1: Recognize the cultural challenges between the business, Dev and Ops

Because Development sits closer to the business in the IT value chain, there's usually less conflict between the business' desire to move fast and Development. That isn't to say Development managers aren't sometimes frustrated by how fast the business wants things given the resources available to them, but they usually are motivated and compensated to move in sync with how fast the business wants to move.

Take a step back and benchmark your current culture for both Dev and Ops, because only then will you be able to understand how DevOps principles may impact those cultures. For Dev, ask yourself, in each line of business "Are we still doing waterfall software development with one release every year or eighteen months, or have we adopted agile methodologies that enable us to deliver multiple releases throughout the year?"

Based on your answers, then ask, “Is that in line with what the business wants today? Is that in line with what we think makes the most sense or are there opportunities we could bring to the business if we adopted agile more broadly?” Finally, “what would be the value to the business and cultural impact of adopting DevOps principles for continuous integration of new software development?”

Next, ask yourself, “How focused is our Ops team on stability, reliability and clamping down on change vs. accepting (and even desiring) change? And how does this vary by line of business or application area?” If you have a risk-averse Ops culture that’s supporting a fragile infrastructure or many legacy technologies, then you may need to tackle some of those challenges first in order to help your DevOps culture evolve. You should also look at your Ops culture through a compliance lens. Ask yourself, “Are we in a heavily-regulated industry that has created its own cultural challenges in terms of change approvals and auditing that we need to take into account?” Finally, what would be the value to the business and the cultural impact of adopting DevOps principles for continuous delivery of new releases?”

One of the hallmarks of DevOps is the ability to develop smaller pieces of incremental functionality, deploy them faster (perhaps even a few or many times a day), and quickly roll things back if there’s an issue. This is a big leap for many IT cultures on both the Dev and Ops side, so understanding where your culture is today is a critical first step.

TIP #2: Think about how you view work

Do you view work as tasks done by individual teams or in an end-to-end fashion (e.g. from raw materials to finished goods and customer delivery)? DevOps transformation requires focusing on work in an end-to-end manner. Your goal should be to identify where your IT organization’s bottlenecks are, with the intent of reducing pressure at those bottlenecks to increase end-to-end flow and, therefore, output that benefits the business.

IT Operations groups often have a few key individuals that understand their company’s IT environment better than everyone else and are consumed with solving high-profile outages and deploying new application releases, the result is that these team members are constantly putting out fires and unable to get new software released that’s desperately needed by the business. These valued “heroes” can often be bottleneck to your DevOps evolution.

Ask yourself, “Is my organization optimized around end-to-end flow or are we optimizing just at the individual workstations?” If it’s the latter, then you are probably stacking up work somewhere at one or more bottlenecks and should address those first. For example, a development project isn’t truly done until the code is tested and deployed. So incentivizing developers to finish code without knowing when and how it will get deployed is just going to create more work in process stacking up in your IT factory. It’s just like on an automotive line where you might see cars with shiny paint, brand new tires and cutting edge technical design, but if the cars have no seats they simply won’t be shippable.

TIP #3: Incentivize your people

Don't assume people will change for the "greater good" that DevOps represents – proactively incentivize them to do so.

If you want to change your culture and create one that's more agile – if you want to optimize for flow versus workstations then you should incentivize to initiate the cultural change you seek. Old habits are hard to break even when we want to do things differently. For executives there's the common tendency to appeal to a higher business purpose and assume the folks contributing on the front lines share the same passion for the company's success. Often team members intellectually understand these goals, but executives should seek to make the change relevant to how it will improve their jobs as developers, QA team members, sys admins, and architects not just how it impacts C-level objectives.

For example, Ops folks usually take particular pride in their indispensability, but to be indispensable they have to compromise family time, working late and on weekends to push out a big high-risk deployment. They may seek the attention and recognition for such 'extra effort', but at the same time they are putting more pressure on themselves and more risk on the business.

As mentioned previously, the move toward a more agile IT with DevOps means deploying with far greater frequency, and that can actually help you operate with less stress and risk. Errors and fixes are likely smaller in effort, and you will be able to roll back hick-ups much faster. Sure your folks may need to stay late occasionally, but they can do their work more often during typical working hours, reclaim their weekends and have significantly less stress and pressure than via traditional large-scale rollouts where expectations are high and so is the risk of failure, re-work, and long days. Plus, with DevOps you are actually delivering on business requirements faster – instead of deploying what was needed months ago.

TIP #4: Make it personal—everyone is different

Realizing everyone is inherently different is critical to incentivizing and empowering your IT team to transform. In most IT organizations, the responsibility is on the first line manager to know their people, understand what makes them tick and channel the right talents and motivations. Some team members may see a move to DevOps as the ideal resume enhancement opportunity. Some may feel intimidation because they are innately risk averse. Still others may not be sure what to think especially if you have one team doing DevOps while they remain on a separate team doing things "the old way". What's important for managers is not to stop at how to make DevOps relevant for a role, but to consider the specific individual. Ask yourself, "How will each of my team members react to this cultural change, and how can I tap into their natural motivations to make them enablers of the change not resistors?"

From what I've seen, there also is a generational aspect to DevOps cultural transformation. Younger employees who don't know a world without mobile phones and grew up on social media are likely more comfortable with agile, small release deployments. Those with more experience are also more likely to think about the reasons why DevOps 'can't work' and it will be key for first-line managers to engage them, listen to their concerns, and work to address them in a way that benefits the organization and realizes the wisdom of their experience.

The bottom line here is that if you make it personal you will build trust and that will help you drive transformation in your organization faster. Perhaps even consider establishing a 'Take a Developer or Ops Team Member to Lunch Day'. By working to understand each other's worlds, challenges, pressures and goals you can unleash a swifter cultural change within your organization.

Tools are Important Too

In IT, we're inundated with tools. Developers have their favorite tools and sys admins do too, so does the project office, the service support group, and the QA team. There are IT tools purchased by organizations years ago that are still being used and others that aren't, tools organizations have recently started using, and others being considered at any given point in time. There are also general-purpose tools the company wants employees to use for things like document sharing, instant messaging, and so on. It's gotten to the point where a lot of IT organizations say, "We have two of everything like Noah's ark", yet they still want more tools.

Part of the reason is that IT practitioners like gadgets and tools. We like the fact that they help us do things and amplify our own abilities, but we also know they can fragment information and can blindside us when someone uses a tool to do something that others weren't aware of.

Do we really need more tools to do DevOps?

DevOps has a close association with the Agile Software Development movement, and one of the core tenets of the Agile Manifesto is to value "Individuals and Interactions over processes and tools". Nevertheless, it's hard to find a DevOps talk or article that doesn't discuss the need for tools like git, docker, jenkins, puppet, chef, and so on. The reason for this is straight-forward: continuous integration and continuous delivery are best done through automation so we can follow a repeatable method and roll things back quickly if there are issues.

At a more basic level, however, we need to acknowledge that there are really two types of tools:

1) Tools that help us amplify our abilities as individuals (e.g. you can drive a nail into a board with a hammer much more effectively than with your hand or a rock because of the strength of the hammer's material and the principle of leverage)
2) Tools that help us coordinate work across many humans, thereby, amplifying our individual abilities (e.g. You can build a house much faster and with better quality by leveraging different experts' skills and using a shared set of blueprints for building construction, plumbing, electrical, heating/cooling, and so on)

The same distinction holds true for software development, deployment, and operations. One individual can theoretically gather requirements, build the software, test the software, deploy the software, and support the software while simultaneously project manage their personal time. Most computer science students have done this at one time or another, but when we are talking about enterprise applications that support core business functions it clearly doesn't make sense because of the amount of effort required at each step and the specialized skills necessary for a high level of competency in each area.

What this means is that we absolutely need tools that amplify our abilities, or that of our individual teams, i.e. the hammers; but we also need to invest in tools that help us coordinate work across teams, i.e. the shared sets of blueprints.

Tools That Amplify Individual and Team Abilities

A lot of the tools attention in the DevOps community has been focused on: source code management systems like git or bitbucket; requirements planning tools like jira or rally; build tools like jenkins; automation tools like puppet, chef, and ansible; and project management tools like MS Project. These tools help us amplify work in each of these respective areas just like hammers, saws, drills, and screwdrivers perform specific functions when building a house and often follow a natural sequence (first you hammer boards in place to create a wall, then hammer on the sheet rock, then saw or drill holes in the sheetrock, screw in electrical outlets and fixtures, and so on.)

Just like building a house has a natural sequence, so does the sequence of tools from code to deployment and it's often referred to as the DevOps "tool chain". This whitepaper doesn't cover each area of the typical DevOps tool chain (bug tracking, automated testing, and other categories are common too), the point is you need to take time to define what the DevOps tool chain should be for your organization and to do so intentionally, taking into account the tool investments your organization already has, the needs and skillsets of your organization, and your own practical budget realities.

Do you have experience in using and self-supporting open source tools or do you have a preference for commercially provided tools? Are you comfortable having multiple requirements tools in different teams or business units, or do you want a single standard? Only you can answer these questions for your organization. There is no prescriptive formula for DevOps tools, although I've mentioned a number of commonly used ones.

Tools that Amplify Work Across Teams

The second area you should focus on are tools to help coordinate work and amplify abilities across teams, i.e. bridging the inherent gaps in the DevOps tool chain, a subject that has received less attention in the DevOps community to-date. This makes sense because it's human nature to think first about our own function and team.

However, this silo thinking is one of the core problems DevOps was developed to address; the long-time focus on tools for specific IT "work stations" has actually entrenched the cultural silos of development, QA, operations, and project office in IT. For example, most IT organizations have two totally separate systems for tracking issues – in operations, it's the service desk application and for code issues in development it's the bug tracking application – and there is little or no relationship between them. In fact, you are lucky if an incident in the service desk can be tied to a bug tracking number.

If we are adopting DevOps in order to optimize the flow of new software releases to support business agility, then we need to look at how we will coordinate work and avoid information distortion and loss when different teams are using different, disconnected tools. In addition to your tool strategy for optimizing individual DevOps workstations, therefore, you also need to have a strategy for the tools that are going to help you effectively manage flow across those workstations.

Most DevOps transformation efforts start small with a core team of perhaps a dozen or so members, and they're often located in one physical location. In this case, you might be able to make do with regular face-to-face meetings and general-purpose communication tools like email, instant messaging, and a SharePoint site or Wiki with MS Office documents. The scope of your cross-team collaboration tools aligns with the scope of your DevOps effort and you might be able to rely on people interactions to unify the tool chain.

But as you scale your DevOps efforts to larger, distributed teams across time zones and multiple business units and applications, your cross-team tools approach should scale as well. The point-to-point nature of instant messaging, ignored reply-all emails, and SharePoint 'document dumps' aren't effective in coordinating the efforts of dozens or hundreds of developers, testers, admins, and project managers. Information gets lost, information gets overlooked, and the gaps in your tool chain expand. A feature misses a commit and doesn't get into the build and isn't deployed. An operational requirement is missed so the test environment isn't configured properly and the release gets pushed. Three weeks are spent solving a performance issue through code when it could have been addressed faster via hardware. A legacy application dependency in production is overlooked and the new mobile app doesn't work in production as it did in test.

There is an emerging class of purpose-built IT collaboration tools, such as ITinvolve, that enable the creation of cross-team workspaces where information is proactively shared as IT teams and tools get work done. This helps large, distributed cross-functional teams collaborate more effectively with each other in-context to raise questions, provide answers, and incorporate what's happening up and down the DevOps tool chain in their individual work.

For example, the developer will have better visibility into when the next build is going to occur and can ensure the feature is committed in time or can reach out to the build engineer in the workspace to request a delay in order to ensure the feature makes it in. The operations engineer can have earlier and better visibility into operational requirements so he can ensure the test environment is configured properly and avoid unnecessary delays. The legacy application dependency in production can be reproduced in test to ensure the application works properly once moved to production. And so on. By eliminating the handoff gaps and information loss across the DevOps tool chain, you can reduce the risk of communication issues, solve problems more holistically rather than individually, and better achieve the goal of improving business agility.

Defining Your Action Plan

In the first section, this whitepaper focused on the critical step of thinking about DevOps in the context of your organization's applications. In the second section, four tips were provided to help foster a DevOps culture in your organization. In part three, the role of tools was discussed and how they can amplify (or constrain) individual and team abilities as well as work across teams.

In this fourth section, the previous topics of (A)pplications, (C)ulture, and (T)ools will be woven together to illustrate how to define your DevOps action plan. Given the breadth of these three areas, though, it's easy to feel overwhelmed from the start.

Define Your DevOps Action Plan

Here's a tip, whether you are a CEO, VP, or manager, smart leaders set a vision (often at least two or three years out) and then make small decisions every week with the goal of generally moving the organization or team in the direction of the vision over time.

With this in mind, first set a vision for DevOps as it relates to your Applications, your Culture, and your Tools. To set a DevOps vision for your applications, place each of your current applications into one of the three pace layers discussed in the first section – system of record, system of differentiation, or system of innovation. Then, do the same thing for any applications you are planning to add to your portfolio. With that context, specific to your organization, formulate a strategic vision for how DevOps will apply to your systems in each layer. Will you seek to have DevOps principles for continuous integration and continuous delivery in place for all of your systems of innovation, and if so, by when? How about for systems of differentiation? Will DevOps be the norm for each of those applications or just some, and by when? Then ask yourself the same questions about your systems of record.

Next, turn to your cultural vision. In the second section, the importance of baselining the degree of collaboration you have between the business, dev, and ops as well as how you view work (in workstations or in end-to-end flow terms) were discussed. How you incentivize people to modify their behavior both with respect to their roles and as individuals were also key points. With this as context, define your DevOps cultural vision. What does collaboration between the business and development look like in the future? What does collaboration look like between development and operations? How about between dev, QA, and ops? And don't forget security too.

How will you recognize when your culture is optimized for flow? For example you might document a set of indicators to measure at regular intervals. Here are three good ones. How much work-in-process (WIP) do we have for Application X today vs. how much was there a quarter ago? What is the average time it takes from the definition of a requirement to its deployment in production today vs. a quarter ago? How often did we make updates to Application X in production this quarter vs. last quarter?

Now, consider your tools vision. Take a baseline of the tools you are using today for each step of the typical DevOps tool chain from requirements management to source code control and bug tracking through to build management, automated testing, configuration management and deployment. And don't forget project management tools too. What is the state of your tool investment in each of these areas? Do you have gaps? Do you have multiple tools in the same area, and, if so, is that okay or do you want to standardize on one? Do your tools vary based on the applications they are used with (see pace-layering discussion above)?

Then take an inventory of your collaboration tools across dev and ops. Are you still mostly using email and sharepoint sites? What role does instant messaging play in your dev and ops teams today? What about conference calls and meetings? Once you've baselined where you are today, set a vision for where you want to be in the future. Are you looking to reduce the need for conference calls and meetings, and if so by how much (time or average number of participants)? What issues are you running into around email and IM communications that need to be addressed and also for SharePoint? Have you looked at the emerging class of DevOps collaboration tools and concepts such as ITinvolve and ChatOps? What role can they play in streamlining communications, keeping people informed, and presenting information to staff so they don't have to go hunting around for it in dozens of systems or call more meetings?

With this as background, define your DevOps tools vision by setting priorities for filling identified gaps and rationalizing existing investments, then set a general target time frame for each area.

Now that you've set a vision in each area – (A)pplications, (C)ulture, and (T)ools, it's time to develop your short term action plan. First pick one or two applications to begin your DevOps journey. Ideally, they should be applications visible to the business and where you can lay claim to the impact DevOps can have on business goals like time-to-market and responsiveness to customer needs.

Once those applications are identified, take an inventory of all the people across your organization (including the business) and those responsible for the development and delivery of the selected applications. Develop a specific plan for how you will educate those individuals on DevOps principles (e.g. attend DevOpsDays, read The Phoenix Project, read a sets of articles and watch a set of seminar or demo recordings), and also develop a specific plan for how you will incentivize behavior changes for their roles as well as what is expected of first line managers to take individual personalities, skills, and experiences into account.

> It's also important not to ignore the people who aren't participating in this first DevOps effort. Educate them as well, for example through an all-hands meeting and an email from leadership. Explain why the company is investing in DevOps transformation, why these applications have been chosen, and your long-term vision across Applications, Culture, and Tools to realize the transformation. Help them understand where the organization is going and why, and how it may impact them in the future. Avoid the perception that "the best people were picked for this DevOps effort and you are not among them" or that this group of people is being given license to do things that others can't (e.g. make changes outside of the standard change management process) – otherwise, you risk reducing morale, good people leaving, and even sabotage of your DevOps efforts.

Once you've completed the above, map the relevant tools you have in place today for managing the development and delivery of these applications as well as the tools on your vision roadmap defined above that will be added or rationalized in the next six months.

By following the above recommendations, you will now have a clear action plan for the near term and a long-term vision you can use to guide daily decisions to move you closer to your vision. Lastly, don't be rigid about your short-term plan or long-term vision, recognize that everyone is learning, including you, and that one of the key principles behind DevOps is the feedback loop!

We hope this paper is useful to you on your DevOps journey.

MAIN	877-741-8944
FAX	832-201-8104
WEB	www.itinvolve.com
EMAIL	info@itinvolve.com

11200 Richmond Avenue, Suite 350
Houston, Texas 77082

A WHITEPAPER FOR IT LEADERS

ITinvolve
People Powered IT™