OTEEMO

10 Key Elements to Ensure DevOps Success

The biggest challenge IT organizations face today is figuring out an optimal way to enable their businesses to be agile, and build and deliver digital services to their customers in a fast and efficient manner without compromising quality— all while their IT budgets are shrinking rapidly. IT has generally been under intense budgetary pressure for quite a while now, but the current time is very different because of the massive adoption of cloud and cloud native platforms, along with movement towards machine learning and artificial intelligence. Customers are becoming impatient with longer wait times for the services they need. If businesses don't catch up and accelerate their digital innovation to meet customer expectations they run the risk of being disrupted.

> *The COVID-19 crisis continues to test the risk tolerance of organizations of all sizes, and business continuity has become a primary focus as we collectively stare into the maelstrom of a rapidly evolving business environment. Agility in the face of crisis is now a requirement — what used to work just fine doesn't work as well any more. DevOps provides a comprehensive and proven toolset to evolve businesses to be more agile, but it needs more than just adopting a set of practices.*

Welcome to DevOps!

Practically every CIO and IT leader has been thinking about implementing DevOps in their organization. Some have already started DevOps initiatives while others are figuring out where to start and how to make sense of DevOps within their complex environments. Regardless of whether you started a DevOps initiative or are in the process of figuring out how best to start and make progress, here is a checklist of things that are critical to ensure successful adoption of DevOps within your organization.

1. Define the "WHY and WHAT" of DevOps: IT leaders shouldn't just say "Let's do DevOps" and delegate down. A clear understanding of what DevOps is specifically going to address in their business is critical for success. Goals and objectives need to be clearly captured, and metrics to measure and track progress relevant to your business should be defined early on. A baseline needs to be established that shows how your organization currently measures against these KPIs or metrics. Is it the cycle time for deploying a feature into production, or the release management efficiency, or the time taken to provision an entire application environment/ platform — or all of the above? Without establishing this clarity first, the best laid DevOps initiative can spiral out of control very quickly, because not everyone may be on the same page.

2. Share the goals, metrics and progress with all teams involved: People involved in DevOps initiatives are typically developers, testers and operations teams. Most of the time the business value that DevOps initiatives provide are not shared with the engineer or practitioner levels. Engineers love technology and will want to play with the latest technology and tools, but it's extremely important for everyone involved to understand what impact this game of DevOps is going to have on the business. Everyone needs to understand that IT is there to enable the business. Fill this out for your organization.

"DevOps is enabling my organization by ____"

Now read what you just wrote. Are the technology initiatives mapping to your business goals? How are they adding value to your business? How would you adjust for changes to your business environment? How would you measure that change at a business level?

3. Every organization is unique, and so are its DevOps goals: We have talked to many IT leaders — CIOs, VPs and IT managers — across various industries and sectors about DevOps. One very common statement we have heard from them (especially from those that work in highly regulatory sectors such as banking) is "I don't need to do 10 or 50 or 100 deployments a day so I cannot do DevOps, or I cannot give any developer access to production environment, so DevOps probably doesn't apply for me". Wrong. DevOps is a way of building, delivering and maintaining quality software. Do what works for you but embrace automation. Continuous Delivery tends to measure the number of deployments as a common metric but that number can vary from organization to organization. Automation that can reduce the provisioning times for application environments, proactive monitoring and management of processes and systems, notifications and alerts in production to developers early-on are also goals that can improve overall application development and deployment process thereby reducing the cycle time. The bottom line is to pick the metrics that are realistic for your organization. But definitely be a little aggressive and push the envelope.

4. Automation is the cornerstone of DevOps: Automate as much as possible. But ***do not automate before you standardize***. Standardize your technologies and processes first, and have a good configuration management and change management process. We have seen people automate things that frequently change or those that are not standardized. Doing so creates a maintenance nightmare to keep track of all the non-standardized automated programs and scripts. So, it's very important that you **Standardize before you Automate**.

Infrastructure, platform and environment provisioning should be automated. Cloud and Cloud Native frameworks are absolute accelerators for Infrastructure provisioning. Implement continuous integration, automate your testing processes as much as possible, and implement push button deployments for promoting code into various environments.

5. Emphasize and support Quality Assurance early on: Based on our experience assessing the state of DevOps for many organizations, we have seen that in general QA (Quality Assurance and Software Testing) gets the least amount of time in the overall SDLC. Doing this leads to mounting technical debt and eventually the quality of the product suffers.

Because building a continuous delivery/deployment pipeline is one of the typical goals in DevOps, it becomes even more critical that QA gets enough time early on during DevOps initiatives to implement automated test suites for regression, functional, non-functional, acceptance, smoke and all other kinds of testing scenarios. Keep in mind that there will be times you have to perform some manual testing to ensure product quality and that is totally acceptable. Just because everything is automated doesn't mean everything will be right.

Doing the wrong things the right way will cause more problems in the future than doing the right things the wrong way. So make sure your QA teams are involved along the way and given more emphasis than before.

6. Everything is code -- Think Infrastructure-as-Code, Configuration-as-Code, Environment-as-Code: Because automation is the cornerstone of DevOps, every software asset definition (infrastructure, platform, environment, application, process) now should be defined by writing code. Hence a good configuration management strategy is extremely important. If something is broken during a process of deployment or provisioning or testing, don't hack and fix it manually for that one time. Stop the process, fix the automation script and re-initiate the process.

6.1. Eliminate Configuration Drift: It is important to have identical configurations in Dev, Test, UAT and Production environments (and DR environments as well). Have processes in place that will check for configuration drifts. Most configuration drifts happen when people tend to hack the system instead of fixing the issue in the source and restarting the process. Having an efficient provisioning and configuration management strategy is key to eliminating configuration drift.

7. Common Frames of References for Dev, QA and Ops: Application developers having access to production environments is a loosely used argument in DevOps and is debatable. Not every organization can/will do that (because of regulations, compliance, policies etc.). But what every organization should be able to do is provide common frames of references for application performance monitoring in production environments. A developer should see the same screen that the Ops person is seeing about how the application that the developer built is performing in a live production environment. Also create a common frame of reference for viewing production logs. This will be especially useful in distributed environments where transactions span multiple nodes and if logs can be consolidated (using various log management tools such as log stash, kibana, elastic search or splunk) and viewed by relevant people it will help speed up and drive analysis of any issues that arise.

Dev and Ops looking at the same thing through a common frame of reference is what we want to achieve in DevOps.

8. Cloud and Containers as Accelerators for DevOps: Yes, everything can be automated but leverage the power of cloud and cloud native platforms to automate infrastructure and platform provisioning as much as possible. As container technologies now are becoming more mainstream, the idea of Dev and Ops teams agreeing upon a common format for software exchange is possible. This common format is packaged as a container.

All the public cloud providers now have made container architectures a key part of their offerings. So our recommendation is to think of moving towards "Containerized DevOps".

9. Start Small and continuously Improve: Don't go big bang with any enterprise initiative and the same applies to DevOps. But starting small doesn't also mean starting trivial. Identify a pilot application that matters, create the required deployment pipeline via automation by bringing together concepts such as rapid environment provisioning, continuous integration, push button deployments and code promotion, automated testing, monitoring and management and application lifecycle management. Go through a few quick iterations to build confidence into the new framework and after the pilot is successful, take it to other applications and teams. Always measure against the defined business metrics and capture lessons learned to continuously improve and push the envelope. DevOps is a way of doing things. Make sure that the people involved initially are influencers and can take the principles back to their respective teams.

10. Capture Response Procedures for Worst Case Scenarios: We have seen organizations mostly only consider best case scenarios when starting off with DevOps initiatives. Starting a green field application and taking the code from Dev through QA through Prod along a fully automated pipeline sounds great. But don't discount the scenarios where something is broken in production during your peak business hours. The question is what does your process look like to get that notification, identify causal factors, call on developers if ops cannot fix it, collaborate real time with everyone involved (what tools and how is collaboration done), provision that application environment in minutes (as developers and testers now probably are working on a newer version), run your end-to-end test suites to ensure the defect is fixed and nothing else is broken along the way and then deploying it back to production.

Have fun building your automation pipelines and software factories. Initiatives such as DevOps require a shift in mindset and cultural change. And that change will cut across people, process and technology. And always keep in mind we are doing this so IT can enable your business to serve your customers efficiently.