DECEMBER 13, 2016

Day 13 - Injecting Modern Concepts into Legacy Processes

Written by: Michael Jenkins (@ManagedKaos)
Edited by: Ivana Ivanovic (@VoiceofIvana)

It's great to read a blog post or listen to a podcast about the latest application or technology, then download it and have it up and running in just a few minutes.

At the same time, it's a little depressing to think that the very same software you so easily got up-and-running on a workstation may not see a deployment on one of your production servers for months or even years — or ever.

So what does it take to bring new technology into a production environment? How do we get our legacy processes up to date?

If you start walking down the path to modernizing your technologies and processes, you may be surprised to find that getting from legacy to modern doesn't have much to do with the tech and more to do with the processes and people involved in the transition.

Here are a few tips to get started.

FOCUS ON THE PAIN POINTS; BUILD A BUSINESS CASE

It's easy to say you want a new tool because it's, well…new. The specs look good. Maybe it's open source and freely available. There's probably a super popular conference that everyone is rushing to so they can learn more about it. And there's even a company pushing the tool, swearing that you'll see an improvement in your workflow orders of magnitude better than whatever you're currently doing.

But don't get lost in the newness of an application or a technique just for newness sake. Ask yourself "What problem really needs to be solved?"

One way to get past the allure of new tools — and find the right one—is to think about the pain points. What's slowing you down as a developer or system administrator? What's breaking over and over again to the point that fixing it becomes a painful routine? Will this new tool or technique solve those problems? If so, how?

In addition to focusing on the rough patches, gather metrics to show how things can improve. For example, if a manual deployment is taking hours but it can be shortened to a few minutes with an automation tool, that makes for a compelling case. Metrics also arm you with facts to back up statements and answer questions.

In the end, you'll want to make a business case for why things need to change and how productivity will improve.

FIND ALLIES, SHARE, AND COLLABORATE

Once potential improvements have been identified, it's time to start spreading the word. Share your findings with your team and managers; propose potential solutions. This can be a tough step since you may be proposing something that changes the status quo or may cause your company to incur a cost in training or software licensing.

Whatever the perceived roadblocks may be, it's good to recruit allies that agree with your message. An easy way to pick up allies is to find others outside of your immediate team that may benefit from the change or the new technique you're proposing.

A good example is an application that benefits both development and operations teams, like Application Performance Monitoring (APM).

Adding APM could bring more insight into the way an application runs and the way the server running the application performs. If a problem happens and causes downtime, both devs and ops can use the APM data to figure out what went wrong and then take steps to

prevent the same thing from happening again.

With the prospect of identifying and resolving problems, developers and operators can share a common, collaborative interest.

START SMALL AND BUILD FROM THE BOTTOM UP

Once there's some traction to make a change, it's best to start with an easy win. Starting with something small is a good way to build confidence in the change while working out issues.

Let's go back to the deployment example. Perhaps it's the case that the manual deployment is done by a different administrator each time without a runbook. Maybe the team feels they know the deployment so well they can wing it without following a script or checklist, but occasionally a step gets skipped, causing rollbacks or rework that lengthens deployments and makes them unpredictable.

An easy, technology-independent way to improve this example process is to clearly document the deployment and share the doc with everyone. Once the deployment steps are documented, the next step would be to automate each step using a scripting language or specialized tool. Having the script be an executable description of the deployment means it should run consistently, independent of the admin that runs it.

Once the deployment script is being executed consistently, it is much easier for the team to think about adding an automation tool—like Jenkins or Rundeck—to manage the deployment. Suddenly, introducing a new technology seems not only painless but necessary. And once the deployment is completely automated, the next step might be to add continuous integration so that deployments happen frequently and without much human interaction.

IN CLOSING...

Reading this post's title, you probably imagined it'd be all about containers, microservices, and the promise of new technologies that are changing the way we work as developers and system administrators. But this post is really about people: convincing them (with a strong business case built on metrics), finding common ground and interest, and easing team members into the changes that come with new technology.

The best part about this approach? Before you even start advocating a new tool, you will have checked your own motivation to make sure you are not falling for the allure of something shiny and new. Instead, you'll be prepared to bring real value to your organization.