AWS CodeDeploy – Adding some CD Goodness to your EC2 Environments

Configuration Management and Continuous Deployment has become a common tool in the toolkit for many organizations these days. As we see from the content I've been doing with my DevOps series co-author Steven Haines at the Turbonomic blog (http://vmturbo.com/about-virtualization/devs-venus-ops-mars-introduction-continuous-integration-pt1/), we have a lot of people putting their focus on the end-to-end development to deployment experience.

Amazon re:Invent is on right now and there have been announcements abound there. Among them is the newly available CodeDeploy product which is a continuous delivery environment to deploy code into EC2 instances.

Deploy, Scale, Rollback and More

CI/CD FTW should be the new mantra for a modern sysadmin or developer. The DevOps movement is gaining momentum with good reason. This is a proven method to extend the pipeline from just dev to an entire deployment framework. Since we have more people using or investigating AWS as a hosting platform, this is a great opportunity to make the final jump to a fully orchestrated deployment.

Testing is key when we talk about DevOps as a methodology. One of the tenets of successful modern software development is the use of TDD (Test-Driven Development). CodeDeploy doesn't replace or create your TDD frameworks, but it becomes a part of the workflow to go from just CI to full CI/CD goodness.

Being able to deploy across your farm with full awareness of the application state by leveraging autoscale and other tools is a great part of what CodeDeploy can do. The rollback, although our last resort, is a necessary step to have available. This is another great feature of a good CI/CD framework. We all make mistakes sometimes, and while the automated tests worked, and the UAT tests worked, sometimes the customers find interesting ways to put our software to the test that wasn't in the plan.

Works With What You've Got

I'm really pleased to see the great integration with existing CI and version control tools (http://aws.amazon.com/codedeploy/product-integrations/) so if you're worried about interoperability, that should allay any concerns. I would like to see options to take the the same processes and also port them to other workloads as a destination, but obviously there is an end-game by Amazon to build the processes to keep you inside their ecosystem. That's not unexpected.

If you are already tied in with AWS, this is going to be a simpler integration. There are a lot of great docs on the AWS CodeDeploy framework that you can find here: http://aws.amazon.com/codedeploy/developer-resources/ which will be a good place to start. You will also want to be sure that you know the process for adding CodeDeploy into your application lifecycle with existing EC2 instances (http://docs.aws.amazon.com/codedeploy/latest/userguide/how-to-configure-existing-instance.html) to be sure that you don't leave your current systems behind.

The Only Thing Constant is Change

The core of what you'll see me writing about lately, and going forward is that we have to either be doing this stuff or planning for this stuff. Evolution is a funny thing because it happens whether we want it too or not. It's just a matter of whether we make the concerted effort to roll with it.

I'm not saying that the T-Rex would have survived if it had learned Go or Python, but I am saying that a modern sysadmin and developer needs to be acutely aware of what is next. The reason for that is because the CIO is, and at some point the decision to evolve as a business will be higher up than many of us sit on the org chart. Don't let that discourage you from thinking you can be a part of it. The best thing that we can do for ourselves and for our companies is to "be the change" as they often say.

Lean forward a bit on your learning and don't forget to reduce that technical debt in every way possible