Modeling should be the first step for #devops when automating a deployment process

When I was a young software developer I had an interview at a large transportation company. This was when object-oriented principles were still the "thing" and Java hadn’t quite yet become the language du jour - but it soon would. Sitting in a rather large conference room with a fairly nice white board I was asked to perform a fairly simple (or so it sounds) task: model a zoo.

Like the much discussed interview puzzle questions of many technology giants today, the exercise was not so much about getting it right (you really can’t model an entire zoo in software during an interview) as about providing the interviewee with insight into whether or not you understand the basic principles of modeling an environment. Are you able to identify the major "objects" and, more importantly, their relationship to other objects in the system? Are you cognizant of the minor objects that interact with the major objects, and what role they play in daily operations? Can you correctly point to not only the attributes of but the role performed by each object?

These are the kinds of questions you answer when you’re actually modeling a system, and it’s not unique to software development. In fact, it’s probably one of the more important aspects of devops that may often be overlooked in favor of focusing on individual tasks.

I had a chance to talk with Dan Gordon at Electric Cloud about “Fail-safe Application Deployments” before the holidays and in reviewing Electric Cloud’s white paper on the topic I was reminded how important modeling is - or should be - to devops.

You might recall Electric Cloud conducted a survey in June 2012 of app developers, 50% of whom said they have missed an application release date because of issues arising in the deployment process. When asked why that was, a majority (69%) pointed to the complexity of the deployment flows combined with the continued practice of manual configuration (62%) in the process as the culprit.

We know automation can help reduce deployment time and ultimately address errors by enabling more testing more often, but automating a poor or incomplete process can be as disastrous as not automating at all. It’s as dangerous to automate a poor or incomplete process as it is to encrypt application data with SSL or TLS and ignore that encrypted malicious code or data is still malicious. What devops needs to do beyond adopting the agile methodologies of development to improve the deployment process is to adopt more of its principles around design and modeling.

Modeling as a Pre-Requisite

One of the five steps to fail-safe application deployments in Electric Cloud’s paper on the topic is automation, of course, but its not just about automation - it’s also about modeling. It suggests that the automation technology chosen to assist devops should offer a number of modeling capabilities:

It should offer extensive process modeling capabilities. There are three essential models to consider:

• Application – the ‘what’
• Environment – the ‘where’
• Workflow execution – the ‘how’

The environment(s) should be modeled as well, with details such as:

• Server configuration
• Associated parameters
• Environment configurations
Of course Electric Cloud’s solutions offer such modeling capabilities. While being able to translate a model into a concrete implementation is always a bonus, it’s more important to go through the modeling exercise than anything else. Whether you’re using a tool capable of modeling the model, as it were, or you’re using scripts or custom developed systems is not nearly as important as actually modeling the deployment process and systems.

Being able to recognize the minutia in a deployment that can often be forgotten is the first step to eliminating missing steps in the deployment process that can cause it to fail. Applications are not islands, they rely on other applications, services, and networking to be deployed successfully, and it is often the case that configurations rely upon IP addresses or other configuration options that must be addressed late in the process - well after the actual application is “deployed” on its platform. Modeling the “objects” in a deployment - as well as their relationships - will help ensure that as the process is automated those relationships and dependent tasks are not overlooked.

Modeling doesn’t have to be a formal exercise. Though many developers use UML tools or other formalized processes to conduct modeling exercises, devops should feel free to discover tools or processes for modeling that best fit their needs.

A rather large conference room and a whiteboard can be a revealing tool, after all.