The Hard Way to Virtual Machine Administration: towards DevOps

A Bridge between Developers and IT Operators

Christian Rodríguez, Lia Molinari, Francisco Javier Díaz
LINTI (Laboratory of Investigation in New Information Technologies)
National University of La Plata, UNLP
{car, lmolinari, javierd}@info.unlp.edu.ar

Abstract—The coexistence of multiple platforms and the implementation of different virtualization models makes server administration more complex every day. The undeniable benefits both methodologies offer in terms of performance optimization and energy saving can be overshadowed if clear guidelines are not established for configuration and maintenance in accordance with the needs of increasingly agile development models that demand quick responses. DevOps is a possible solution to this situation. However, it demands a new perspective in traditional roles within technology areas.

Keywords—DevOps tools; IT Governance

I. Introduction

Technological areas in organizations have had to adapt to a service model in general, and a development model in particular, that calls not only for effectiveness, but also efficiency. This new way of working demands strong interaction between development areas and the so-called support areas. With the advent of new programming paradigms such as agile methods, these development areas have begun to train in obtaining results within a short period of time. Support areas, with progressively more demands from development areas generally function without a proper structure suited for current trends. The LINTI development area of the Computer Science School identified this problem at an early stage, and searching for solutions, decided to analyze and implement DevOps through multiple tools (Chef[1], Puppet[2]).

The formerly markedly separated areas of support and development began to experience the demands of technological evolution in terms of virtualization and agile methods.

Developers are releasing versions in very short lapses of time, sometimes more than once a day, and these applications must be tested and released to the public.

Moreover, IT operations must deal with different environments and their instances, which imply configuration changes. Making these changes individually can be tedious and impractical, and may introduce a higher probability of errors. The response must also be fast.

DevOps is a new trend oriented towards obtaining a collaborative work relationship between development and operations areas in Information Technology (hereinafter IT). The goal of this collaboration is to improve response times and complete project phases according to plan without disregarding reliability, stability, resilience and security in the operations environment [1]. Important firms such as Etsy and Facebook, among others, have chosen DevOps.

Reference [2] offers the following definition of DevOps: "an IT service delivery approach rooted in agile philosophy, with an emphasis on business outcomes, not process orthodoxy. The DevOps philosophy (if not the term itself) was born primarily from the activities of cloud service providers and Web 2.0 adopters as they worked to address scale out problems due to increasing online service adoption. DevOps is bottom up based, with roots in the Agile Manifesto and its guiding principles. Because it doesn't have a concrete set of mandates or standards, or a known framework (e.g., ITIL, CMMI), it is subject to a more liberal interpretation".

DevOps is the result of the convergence of several movements, such as Velocity Conference, Infrastructure as Code, Agile Infrastructure, and Agile System Administration, among others. It constitutes a methodology that promotes communication, collaboration and integration between software

1 http://www.opscode.com/chef/
2 https://puppetlabs.com/solutions/devops/

developers and IT professionals. Its goal is to release products and software services more quickly.

According to Mike Loukides [3], modern applications must be resilient and fault tolerant; they must be monitored and able to solve errors quickly. In Cloud Computing, in Platform as a Service environments (PaaS), the operations area and the development area are integrated, blurring the traditional separation of functions. The evolutionary trend is towards the code, and those responsible for the infrastructure, i.e. systems administrators and IT corporate groups, must understand and accept this evolution. It is time to stop segregated work. They must cooperate and collaborate with developers. Loukides informally calls this DevOps. One of the DevOps mottos is “developers that think like operators and operators that think like developers”.

Saanjev Shrama, in an article published on the web[3], introduces two concepts in this framework:

1. Cycle time

Cycle time is defined as the average time taken from the time a new requirement is approved, a change request is requested or a bug that needs to be fixed via a patch is identified, to the time it is delivered to production. Agile organizations want this time to meet the bare bones minimum.

2. Versioning environments

Operators must deal with multiple configurations and update installations. These changes imply the creation of a new “version” of the environment. The only way this can be done in time, with documentation and minimum errors is by applying all changes through scripts. These demands result in the creation of Infrastructure as a Code.

Scripts generate new virtual environments that must be versioned like traditional code, managing the configuration in compliance with best practices.

In order to implement this methodology, it is necessary to agree on the following:

- Infrastructure automation
- Control sharing
- One-step development and deployment
- Metric defining

[3] Understanding DevOps – Part 5: Infrastructure as Code. http://sdarchitect.wordpress.com/2012/12/13/infrastructure-as-code/

II. Analysis of the Problem

The difficulty of managing the infrastructure in a coordinated manner with the LINTI software developments, where the main demand is the constant modification of the programs already in production, led to the analysis of available alternatives for deployment automation (code release) and configuration management. This analysis led to researching the emerging technologies known as Virtualization, DevOps and Infrastructure as Code, among others.

The best way to describe the road taken is to do it chronologically, showing the evolution of knowledge gained in accordance with information and experiences obtained in each stage. The initial methodology, when there were few applications, was to install each one of them in a separate server. When the number of applications began to increase, the first alternative was to install them in the same server with several virtual hosts. The goal of this action was to simplify the administration on the part of the support area and, indirectly, to lower the costs, as having one server per application was unnecessarily costly. Although this approach simplified the administration, an incident in which one of the applications was targeted, which led to the rest of the applications in the server being exposed, compromised the security of the systems. At the time of the aforementioned security incident, two important decisions were made:

- Independently from the security incident, the application development area decided to adopt the fundamental philosophy of reuse, distribution and integration of applications, based on the use of APIs (Application Programming Interfaces), WEB Services and SSO (Single Sign-On).
- In light of the incident, alternatives were analyzed and a decision was made to separation the areas of production and final user testing, adding the development and testing areas to converge with new development trends.

With these two crucial points in mind, two separate paths were followed – environment generation and application development with this new architecture. In this instance, the need to build an environment for new applications integrated through APIs was made more complex by the amount of relations and dependencies mandatory for its correct operation. This had so much impact that the application developers' roles and responsibilities were blurred, resulting in junior developers without experience in configuring services provided by IT areas having to install extremely complex environments, losing sight of their real function – developing.

Moreover, from the production perspective, many applications require replication environments, load balancing,caches and performance improving entities. This makes it necessary to define and update procedures and instructions that specify the steps to be followed for installing each one of the involved parts. This situation triggered a new stage: researching trends in technological infrastructure management. Multiple alternatives have been tested with the permanent goal of simplifying the deployment of new services, backups and hot service migration to servers with more resources. Virtualization was the chosen option.

Once virtualization had been adopted, the proliferation of virtual machines slowly turned the initial enthusiasm to awareness of a difficult issue: their management was out of control. Although the creation process of virtual machines is fairly simple, their proliferation and maintenance, together with the number of servers up for administration, updates and configuration made it a daunting task. A clear example of the increasing complexity of this environment was the update of the SSL certificates before their expiry date.

Thus, a research into alternative server management and automation methods began.

Infrastructure as Code seemed to satisfy this demand. After an exhaustive analysis, we opted for Opscode Chef, a product that allows “infrastructure programming” and testing using development concepts like TDD (Test Driven Development). The procedure was to program recipes that allowed us to describe the creation of web servers that hosted applications. It was also necessary to release the applications using deployment automation tools such as Capistrano, a task reserved to the most experienced developers only. This was the first convergence point between administrators and developers. They analyzed the tools in a collaborative manner, and agreed on the convenience of management simplicity as an investment in the future for both parties.

III. THE SITUATION TODAY

Using tools like Chef allows recipes installed in our servers to be versioned with SCM (Source Code Management) like GIT and tested in virtual machines like VirtualBox by administrators wishing to test changes before applying them in production. We used the following tools for the recipe development phase:

- Chef-solo
- Vagrant[4]
- Berkshelf[5]
- Chef recipes available at http://community.opscode.com

Once a recipe works, it is uploaded to the Chef server. This server holds all the tools used by the servers whose administration has been automated. Installing a new virtual server became considerably simpler with the use of templates provided by the virtualization tool itself, added to integration with Chef. The automated servers are the ones that have a direct impact in the development area. It is projected that this procedure will be applied to other services from different areas. For the development scenario, Vagrant machines were used that offer the less experienced developers the opportunity to work and focus their efforts on development and not in environment building. As regards environment separation, the replication of the production environment was simplified into a preproduction and testing environment.

The problems detected at first are related to the adoption of this new work scheme. Administrators must now program, which was not normally their task. This generated an initial resistence that grows weaker as the advantages of this type of model surface. A remarkable feature of the Chef server is its node database with information on the whole organization, which simplifies the configuration of backup and monitoring applications such as godrb, SENSU, Nagios, and Bacula, among others.

IV. BEST IT GOVERNANCE PRACTICES

Various practical tools exist to guide and support IT leaders and decision makers in making IT decisions. Some of the most relevant include the following. The Control Objectives for Information and related Technology (COBIT) [4] is a framework that supports IT process management. COBIT describes the central role of ICTs in creating value for business. The Information Technology Infrastructure Library (ITIL) [5] is a set of practices for IT service management. ITIL practices can be used for aligning IT services with organizational needs. ISO standards define guidelines for process quality assurance (ISO 9000) [6] and information security systems management (ISO 27001) [7].

[4] Vagrant is open-source software for creating and configuring virtual development environments. http://www.vagrantup.com/
[5] http://berkshelf.com/

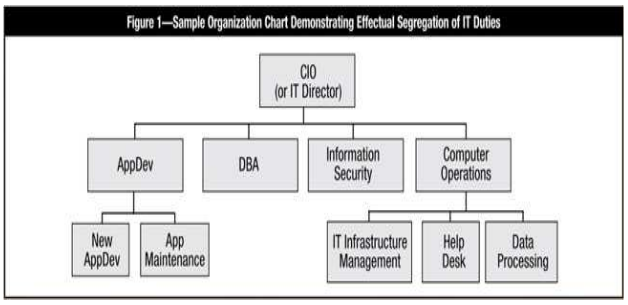

Figure 1

One of the demands in the framework of the functional structure of technological areas is the Segregation of Duties (SoD). SoD plays an important part in designing an IT function. This suggested separation excludes the possibility that a single person can be responsible for multiple critical functions, so errors and wrongful appropriations can be detected early on, in the normal course of business processes. It allows for prevention and deterrence from fraudulent or malicious acts. Reference [8] includes this organizational (and traditional) chart, within the context of SoD.

In Figure 1 and in the aforementioned standards and best practices, the segregation between development and operations functions is unequivocal. AppDev is separated from Computer Operations. IT Infrastructure Management is one of the divisions of Computer Operations.

Note: Figure 1 was extracted from ISACA (www.isaca.org).

DevOps, in order to improve performance, blurs this distinction. One possible variation is that shown in Figure 2.

According to Decision Theory [9], before making a decision, the CIO should ask themselves the following questions: 1) who receives the benefits; 2) what are the associated risks; 3) who bears the risks; and 4) what are the required resources; among others. For answering such questions, decision elements should be identified. Reference [10] includes a reference model for technology innovation-related decision-making processes (Figure 3).

It is clear that the benefit in using DevOps is received by the user by obtaining deployments in the expected time in the framework of agile developments. The risks associated to the implementation deserve to be analyzed, but in a first, light conclusion, we can say that using code to manage the infrastructure merits the definition of controls in light of the possibility of modifications, well- or ill-meaning, which can modify the cycle. These controls would mitigate the risk.

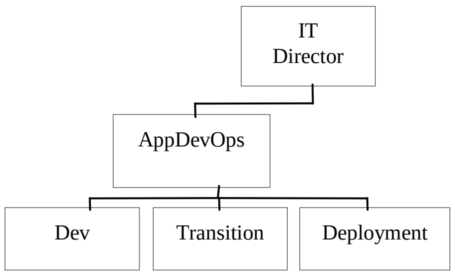

Figure 2

Applying the RACI matrix is an alternative for assigning responsibility levels for process practices to multiple roles and structures. This matrix defines four levels:

– Responsible: Who is performing the task? It refers to roles in charge of the main activity to complete it and produce the expected outcome.

– Accountable: Who is accountable for the success of the task? It assigns responsibility for the completion of the task (where responsibility ends). As a principle, accountability cannot be shared.

– Consulted: Who provides input? These roles are essential. This input must be considered and, if required, measures must be taken to ensure its enforcement, including information on the process owner and the Board of Directors.

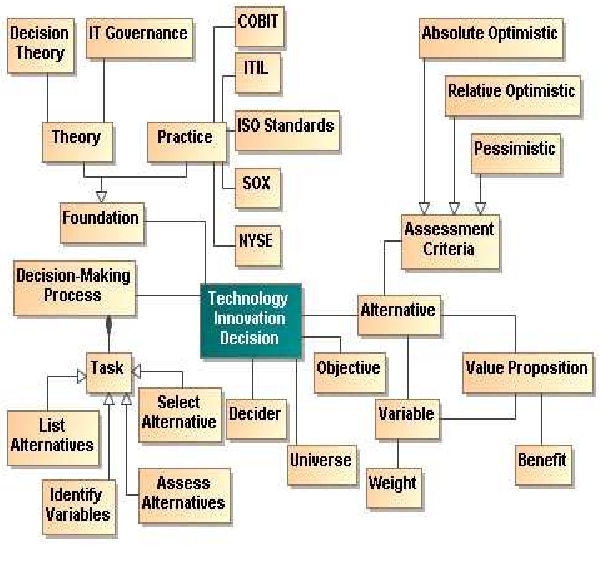

Figure 3

– Informed: Who receives the information? These roles are informed of the achievements and/or deliverable tasks. Of course, the role that is “responsible for making“ must always receive appropriate information for task supervision, as must the roles responsible for the area of interest.

Another aspect to be considered is the competence of the staff that will participate in this development-operation model. DevOps's motto, “developers who think like operators and operators who think like developers”, represents integration from the perspective of business goals. However, how this task is translated to integration in the framework of profile and competence definition is a task that calls for defining the training and background required for these profiles.

When technology areas are audited, the auditor is guided by some of the reference frameworks quoted above, or other best practices that do not contradict them. Even those documents where innovation [11] is taken into account must be analyzed again in light of these trends.

V. CONCLUSIONS

The model adopted has allowed us to reduce release time for applications developed with agile methods. The business goal – to provide quick responses to requirements – goes beyond application development. In order to achieve this goal, all areas must be integrated and synchronized.

However, it is worth clarifying that this type of model demands a revision of the traditional segregation of functions defined in reference frameworks, best practices and standards.

All innovation implies experimentation, learning from mistakes and improving. This, in turn, implies risk. Every organization must analyze just how much risk it is willing to take so that the development of applications through agile methods is a business goal that goes beyond the development area itself.

REFERENCES

[1] “Top 11 things you need to know about DevOps”. Gene Kim. IT Revolution Press. www.itrevolution.com

[2] David Mitchell Smith. Hype Cycle for Cloud Computing, 2011. Gartner Publication Date: 27 July 2011

[3] O’Reilly..What is DevOps?. eBook.

[4] ISACA, COBIT Official Site, http://www.isaca.org/cobit/pages/default.aspx

[5] ITIL Official Site, http://www.itil-officialsite.com/.

[6] ISO, ISO 9000 - Quality management. http://www.iso.org/iso/iso_9000.

[7] ISO, ISO/IEC 27001:2005. Information technology - Security techniques - Information Security Management Systems – Requirements.http://www.iso.org/iso/catalogue_detail? csnumber=42103.

[8] Tommie W. Singleton, Ph.D. What Every IT Auditor Should Know About Proper Segregation of Incompatible IT Activities. ISACA Journal, Volume 6. 2012

[9] P. F.J. Pavesi, La Decisión, Ediciones Cooperativas, 2000. ISBN 987-98315-6-X

[10] Lía Molinari, Elsa Estevez and Francisco Javier Díaz. Technology Innovation – A Reference Model for Decision-Making Processes. CACIC 2012. Universidad del Noroeste de Buenos Aires (UNNOBA). Argentina. In press

[11] Vittorio Chiesa, Paul Coughlan, Chris A. Voss. Development of a Technical Innovation Audit. Journal of Product Innovation Management. Volume 13, Issue 2, pages 105–136, March 1996.