A Tool to Support Deployment of Scientific Software as a Service

Maximilien P. M. De Bayser, Leonardo G. Azevedo, Leonardo P. Tizzei, Renato Cerqueira

[1]IBM Research – Brazil
Av. Pasteur 146 & 138, Botafogo, Rio de Janeiro, RJ, Brasil – 22290-240

{mbayser, lga, ltizzei, rcerq}@br.ibm.com

Abstract. *Most of existing e-Science infrastructure is based on shared, interconnected grids. This approach lowers the bar for access to large-scale computational resources and makes possible the collaboration of geographically dispersed teams. However, it requires a big up-front investment and the acquired resources must be continuously maintained and upgraded. Cloud infrastructures are an alternative since resources can be allocated on demand and recently have become more suitable for HPC (High-performance computing). This paper describes our work on the installation and configuration automation of software packages of transient grids based on Infrastructure as Code (IaC).*

1. Introduction

Due to the large investment and maintenance costs of HPC resources, research organizations all over the world have tried to share and pool these resources in large computational grids. Because they are based on voluntary collaborations between multiple tenants, access to them is subject to administrative and political issues. Grids are often the result of public funding, so their utilization must be high to justify the expenditure. Being a shared resource, job submission mechanisms often enforce a fair use policy where everyone gets the same priority. Giving different priorities is technically possible but often politically complicated. As the computational resources are finite, it is only possible to obtain best effort QoS guarantees [Singh et al. 2006].

The advent of commercial cloud computing has brought a different model of large scale computing in which users pay only for resources they actually use. Another important feature of clouds is their elasticity, which means that resources can be scaled up or down according to current demands. It has been proposed to use a commercial cloud to complement grids in a hybrid model, where the cloud is used when grid resources are depleted [Kim et al. 2009]. But most importantly, clouds support entirely virtual environments that can be set up and modified with great flexibility and without affecting other users. This flexibility was noted as early as 2006 when Childs et al. realized that they could implement *replica grids* or *grid testbed* simulating physical machines with an entirely virtual environment [Childs et al. 2006]. More recently several entirely cloud based grids were proposed focusing on reduction of costs, more flexibility, ease of use for researchers without a computer science background and isolation between virtual grids [Strijkers et al. 2010] [Rehr et al. 2012] [Singh et al. 2006]. These initiatives are what is called a platform as a service (PaaS). In the beginning commercial cloud offerings were not geared towards HPC requirements, but recently more powerful nodes are available.

Be it physical or virtual grids, the management of configuration is far from trivial. Different research groups use different software packages and operating systems and often different versions of a same package are required generating installation conflicts. In virtual grids this problem is reduced because there can be total isolation between virtual grids, but it still is a considerable amount of work to assign each node a different role in the system and manage the software stack.

Software tools such as SmartFrog[1] provide orchestration capabilities to distributed applications, but do not support the installation of these applications. Other software tools such as Chef[2] [Marschall 2013] and Puppet[3] (see section 2) support software installation, but do not provide the abstractions for non-expert users.

This paper describes our work on Installation Service, a tool that aims to provide uniform interface for several IaC tools making it easier to build virtual environments. We already have an intuitive user interface that can be used by a non-expert user and a RESTful service interface that can be used to automate the provisioning of nodes that are automatically added to scale the grid. We see the Installation Service as an important component in a HPC PaaS architecture.

The tool has been used to deploy infrastructure and application softwares for a computing and data-intensive scientific application. This application as a service in the cloud. It is reuquires to install its dependencies (e.g. libraries, software packages). So, Installation Service tool is used to install the softwares and manage dependencies and also enable the customization of the cloud environment.

2. Background

Complex configuration management is not a problem only for scientific environments. Software development companies also struggle with different environments and how they trigger different bugs. Traditionally when the development team delivers a new version to the operations team, a lot of problems are discovered because the production environment is different. Often developers cannot reproduce the problems reported by the operations team. This scenario led to the development of a methodology called DevOps, a portmanteau of development and operations, that eliminates the division between the two teams to create a pipeline of continuous delivery. DevOps extends the Agile philosophy to operations. A cornerstone of this methodology is the concept of Infrastructure as Code (IaC) [Hüttermann 2012].

The idea of IaC is to write all the software requirements of a computational infrastructure in an executable form. On the node level, this means not only that there is a list of the software packages that should be installed but also a set of scripts detailing how these packages should be configured and customized. Packages such as the Apache web server and OpenMPI usually require a configuration phase before they can be used. On the grid level IaC means that there is a script that details how many nodes exist, which operating system should run and which set of packages and configurations should be applied to each individual node according to their role.

[1] http://www.smartfrog.org/
[2] http://www.getchef.com/
[3] http://puppetlabs.com/

Currently there are several IaC tools such as Vagrant[4], Chef and Puppet. The improvement of these tools over simple shell scripts is that they provide a uniform and concise notation, that is mostly declarative. They also assure a uniform interpretation of the scripts even on different operating systems.

A big advantage of IaC scripts is that they can be kept under version control systems (e.g., Git). Hence, for each particular grid setup there is a corresponding version of these scripts in the repository. When a problem occurs, it is easy for a developer to recreate the environment where the problem appeared. This feature is also interesting for open science. Many research institutes already open their source code making it easier for other researchers to inspect, reproduce and validate their work or even to start a new line of research based on an existing one. However the infrastructure setup is not always trivial. If virtual lab setup scripts were published along with source code, it would be much easier for other scientist to get up to speed.

3. Installation Service tool

We propose Installation Service, a tool whose main goals are: (i) Facilitate software installation (software applications and infrastructures) by inexperienced users; (ii) Enable environment customization, e.g., selection of providers (e.g., Softlayer or OpenStack for Virtual Machines provisioning), automation infrastructures (*e.g.*, Chef or Puppet), and operating systems (e.g., Linux or Windows); (iii) Provide an API that can be consumed by other applications to automate software configuration following a SaaS (Software as a Service) approach or it can also be deployed as part of a PaaS.

The current version of Installation Service allows to: (i) Bootstrap a node: it makes a physical, virtual, or cloud location able to download and apply configuration scripts; (ii) Upload/download of installation scripts to/from a server repository; (iii) Set the installation scripts that should be executed on a node; (iv) Synchronize a node: update the node configuration with its current configuration scripts list.

Installation Service is composed by two modules (Figure 1): `Installation Service` **`API`** and `Installation Service` **`UI`**. The `UI` provides a user-friendly web interface to manage software installation and configuration scripts. It consumes the services provided by the `API`. The `API` provides RESTful web services that encapsulate IT automation platforms (currently Chef). Hence the `API` consumers have a standard interface that works with different IT platforms.

The current version of the `API` encapsulates Chef. Chef is an *Infrastructure as Code* tool used to implement the automatic deployment and continuous integration of the *DevOps* methodology. Chef is a systems and cloud infrastructure automation framework to deploy servers and applications into physical, virtual, or cloud locations, named as nodes [Marschall 2013]. The Chef encapsulation is done in three different ways: (i) Bootstrap a node calling Knife - a command-line tool; (ii) Update the node using ssh to connect to the node and to run chef-client - a program installed in the node during bootstrap process whose responsibilities are to connect to Chef Server, download current configuration scripts, and apply them to the node; (ii) All other operations are performed invoking Chef Server REST API services.

[4]`http://www.vagrantup.com/`

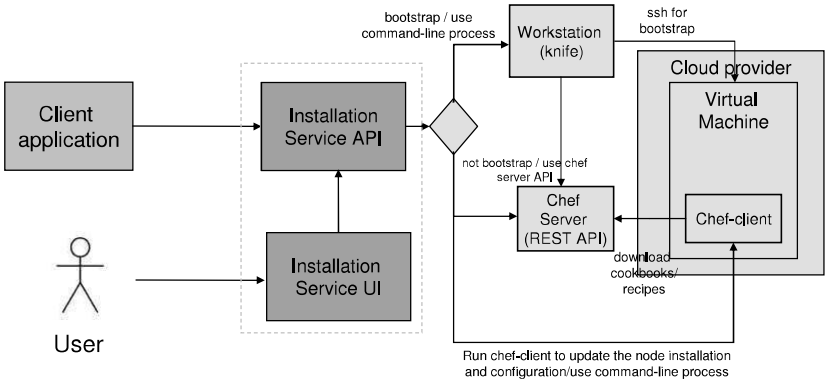

Figure 1. High-level structure of the Installation Service

4. Conclusion

This work reasons on the transition from grid to Cloud Computing in e-Science. We demonstrated that IT automation tools for installation and configuration are required since they improve the reproducibility of environment configurations. Our proposal of Installation Service evolves current tools allowing to use more than one tool simultaneously to handle environment specific features. Our tool provides a friendly user interface and supports multi-tenancy. It also exposes a service API that can be used as an infrastructure component in a PaaS or as a stand-alone SaaS. The current implemented use cases are presented in Section 3.

As future work, we are going to evolve Installation Service to provide new automation features, to work with other tools than Chef, to handle package life-cycle, and consider semantic patterns to integrate with API consumer applications. We will also evaluate the tool in real scenarios.

References

Childs, S., Coghlan, B., and McCandless, J. (2006). Gridbuilder: A tool for creating virtual grid testbeds. In *Proc. Intl. Conf. on e-Science and Grid Computing*, pages 77–.

Hüttermann, M. (2012). *DevOps for Developers*, volume 1. Springer.

Kim, H., el Khamra, Y., Jha, S., and Parashar, M. (2009). An autonomic approach to integrated hpc grid and cloud usage. In *Proc. Intl. Conf. on e-Science*, pages 366–373.

Marschall, M. (2013). *Chef Infrastructure Automation Cookbook*. Packt Publishing Ltd.

Rehr, J. J., Jorissen, K., Vila, F. D., and Johnson, W. (2012). High-performance computing without commitment: Sc2it: A cloud computing interface that makes computational science available to non-specialists. In *Proc. Intl. Conf. on E-Science*, pages 1–6.

Singh, G., Kesselman, C., and Deelman, E. (2006). Application-level resource provisioning on the grid. In *Proc. Intl. Conf. on e-Science and Grid Computing*, pages 83–.

Strijkers, R., Toorop, W., Hoof, A. v., Grosso, P., Belloum, A., Vasuining, D., Laat, C. d., and Meijer, R. (2010). AMOS: Using the cloud for on-demand execution of e-science applications. In *Proc. Intl. Conf. on e-Science*, pages 331–338.