Project Acronym: STORM CLOUDS

Grant Agreement number: 621089

Project Title: STORM CLOUDS – Surfing Towards the Opportunity of Real Migration to CLOUD-based public Services

Deliverable D2.4.2
Cloud Application Template Catalogue

Workpackage: WP2

Version: 1.0

Date: 29/07/2015

Status: WP leader accepted

Dissemination Level: PUBLIC

Nature: REPORT

Editor: Marco Consonni (Hewlett Packard Italiana S.r.l.)

Authors: Marco Consonni (Hewlett Packard Italiana S.r.l)

Reviewed by: Alkiviadis Giannakoulias (European Dynamics)

Version Control

Modified by	Date	Version	Comments
Marco Consonni	15/7/2015	Draft	First draft
Marco Consonni Alkiviadis Giannakoulias	29/7/2015	1.0	Ready for Review

Executive Summary

Surfing Towards the Opportunity of Real Migration to Cloud-based public Services (STORM CLOUDS) is a project partially funded by the European Commission within the 7th Framework Program in the context of the Capital Improvement Plan (CIP) project (Grant Agreement No. 621089). The project has the objective of exploring the shift to a cloud-based paradigm for deploying services that Public Authorities (PAs) currently provide using more traditional Information Technology (IT) deployment models. In this context, the term "services" refers to applications, usually made available through Internet, that citizens and/or public servants use for accomplishing some valuable task. The project aims to define useful guidelines on how to implement the process of moving application to cloud computing and is based on direct experimentation with pilot projects conducted in, at least, the cities participating to the consortium [1].

Work Package 2 (WP2) of the Storm Clouds project is aimed at designing and implementing a reference architecture for the Storm Clouds Platform (SCP), the cloud platform infrastructure for hosting application services selected for being ported to cloud. In the WP scope there is also the preparation of a library of tools (prefab VM Images, cloud-application templates and other artefacts) that, taking advantage of the automation functions implemented in IaaS, can be used for facilitating the deployment of cloud-based applications [1].

This document is the second issue of the iterative deliverable “Cloud-Application Template Catalogue” that describes the current status of the tool library.

The first issue of the document [2] describes a list of prefabricated Virtual Machine (VM) images: some of them simply provide the operating system, others implement basic functions like, for example, a database engine (e.g. MySQL or PostgreSQL). They were obtained by manually installing the software packages and used as the “starting point” for deploying the application services. This approach, albeit correct, has some drawbacks in particular in a cloud environment where services need to be activated and de-activated in very short time. In fact, the steps for deploying the software must be well documented and without automation the cloud user is required to perform them manually. In addition to being tedious, the process is also error-prone because it is usually hard to document all the single steps required for installing the software.

As mentioned in [3], automation can significantly improve the situation and SCP implements some automation functions - like OpenStack Heat - that permits the cloud user to describe all the IaaS objects she needs for an application in a script called **stack**. Using Heat stacks it is possible to "*control the entire lifecycle of infrastructure and applications within OpenStack clouds*" [4]. In this perspective, recent activities of WP2 were focussed on the implementation scripts (e.g. heat scripts) for automating the deployment of Virtual Machines and/or other cloud computing objects (e.g. virtual volumes, virtual routers, etc.).

This document reports the current state of art of the tool library mostly describing the implemented scripts.

Table of Contents

Abbreviations

Acronym	Description
CLI	Command Line Interface
CIP	Capital Improvement Plan
CIP-PSP	*See CIP and PSP*
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DOW	Description of Work
FP7	Framework Program 7
IaaS	Infrastructure as a Service
IT	Information Technology
N/A	Not Available or Not Applicable
PA	Public Authority
PDF	Portable Document Format
PSP	Policy Support Program
SCP	STORM Cloud Platform
SME	Small and Medium Enterprise
TBD	To Be Defined
TBW	To Be Written
URL	Uniform Resource Locator
WP	Work Package
YAML	YAML Ain't Markup Language

1 Tools Description

This chapter describes the implemented tools.

They are classified into the following categories:

- **Tenant Preparation Tools**: tools for configuring an OpenStack tenant;
- **Deployment Tools**: tools for deploying stacks of IaaS objects (VMs, Virtual Volumes, Virtual Networks, etc.).

For each tool the following information is provided:

- **Name**: the name of the tool;
- **Languages**: the programming language(s) used for implementing the tool;
- **Description**: a brief description of the tool;
- **Input**: input parameters;
- **Output**: output values.

The tools are implemented as software scripts written in one or more of the following scripting languages:

- **YAML (Heat)**: used for creating IaaS objects using OpenStack Heat;
- **Bash Shell**: used both for configuring SCP@HP and for configuring the VMs created through YAML (heat) scripts.

It is worth noticing that while YAML (Heat) scripts can be used for creating most of the IaaS objects [5], when it comes to configure VM objects (e.g. installing software packages), the cloud user needs to use different languages like, for example bash scripts, puppet scripts [6], etc. In this perspective, YAML (heat) scripts creating VMs 'host foreign language scripts' that take care of configuring the created VMs.

1.1 Tenant Preparation Tools

During the project, the consortium decided to host the migrated applications on a public cloud operator; Enter S.r.l. (http://www.enterpoint.it) was selected because it provides Infrastructure as a Service (IaaS) functions using technology compliant to the SCP architecture. In addition HP has implemented an in-house SCP instance at its own premises with the main purpose of providing all the partners with a testing and staging environment [7]. The production environment hosted by Enter was named SCP@Enter while the testing environment hosted by HP is SCP@HP.

Because of the full compatibility of the technology used for the implementation, the coexistence of two environments has not raised major issues and the migration of an application from SCP@HP to SCP@Enter (or vice-versa) has resulted in a simple file transfer from one platform to the other. More in detail, once a VM is prepared on the testing environment it can be moved to the production environment by making a snapshot of the VM and exporting the related file from the testing environment to the production environment. This process has just presented some minor issues mainly related to differences in the configuration of the two platforms. For example, the flavors[1] used for launching VMs on SCP@HP differs from the flavors used on SCP@Enter and it was required to "map" the flavors implemented on SCP@HP to a similar flavors defined for SCP@Enter.

The implementation of automatic tools for the deployment of cloud applications requires that even these minor issues are fixed in order to streamline the operations. For this purpose we have implemented some tools for preparing the testing environment and creating operating conditions as closer as possible to the ones we have on the production environment: the **Tenant Preparation Tools** are the tools implemented for that purpose.

In OpenStack, a tenant (or project) is an isolated resource container that form the principal organizational structure within the cloud and, as a common practice, public cloud operators based on OpenStack activate a tenant for each cloud service contract subscription. Having said that, Tenant Preparation Tools create, on the testing environment, a tenant similar to the tenant we have on the production environment.

The following tables provide details on the tools.

Name	CreateTenant.sh
Languages	Bash shell script
Description	It creates on SCP@HP a tenant (**storm-enter**) similar to the one available on SCP@Enter specifying the same resource quotas and flavors. In order to run successfully, it requires the installation of OpenStack CLI tools and the credentials of an OpenStack user with admin role
Input	None
Output	None

Name	GetPublicCloudImages.sh
Languages	Bash shell script
Description	It downloads from Internet the cloud images of the operating systems used for the applications and/or the SCP components, and uploads them to the SCP@HP image repository. It currently downloads the following images: - lucid-server-cloudimg-amd64-disk1.img - trusty-server-cloudimg-amd64-disk1.img In order to run successfully, it requires the installation of OpenStack CLI tools and the credentials of an OpenStack user with admin role
Input	None
Output	None

Name	DeleteTenant.sh
Languages	Bash shell script
Description	It deletes the tenant created by CreateTenant.sh (**storm-enter**). In order to run successfully, it requires the installation of OpenStack CLI tools and the credentials of an OpenStack user with admin role.
Input	None
Output	None

[1] OpenStack flavors are virtual hardware templates defining sizes for RAM, disk, number of cores, and so on.

1.2 Deployment Tools

This section describes the tools implemented for automating the creation of cloud IaaS objects specifically designed for supporting applications running in the cloud. In contrast with the tools described in the previous section, designed for running on SCP@HP, the ones listed here can be run both on SCP@HP and SCP@Enter.

Name	Network.yaml
Languages	Yaml (heat script)
Description	It creates the network objects required for hosting the STORM cloud applications. The following picture shows the network objects created by the script:  The scripts creates a private network (named SCP_Network) represented by the orange thick line; this is the network where all the VMs created in the tenant will be connected to. SCP_Network is connected through a router, represented by the small black square, to the external provider network, represented by the blue thick line. The provider network is not created by the script and, being the representation of the network for accessing Internet, must be made available by the cloud operator. SCP_Network supports addresses belonging to subnet 10.0.0.0/24 (IP address range reserved for private network [8] and provides a DHCP service. On the other hand, an IP address subrange (by default addresses from 10.0.0.1 to 10.0.0.127) are reserved for static allocation. The reason for reserving a range of IP address for static allocation is that some SCP services are themselves implemented as VMs (for example the DB layer) and assigning them a static IP address prevents the realization of a local DNS service. Actually the scripts requires that the DNS IP address is passed as a parameter; the actual value could be either an IP address provided by the cloud operator or an IP address of a DNS service publicly available on Internet (e.g. 8.8.8.8 or 8.8.8.4). In order to run successfully, the script requires the installation of OpenStack CLI tools, the credentials of an OpenStack user with member role for the tenant where the network objects are created.
Input	**External_Network**: the name of the provider network **External_DNS_IP**: the IP address of the DNS service
Output	None

Name	HeatImageBuilder.yaml
Languages	YAML (heat script) and bash shell script
Description	This script creates a VM used for preparing virtual machine images that can fully inter-operate with OpenStack heat. As mentioned above, Heat automates the creation and orchestration of IaaS cloud resources but does not provide a language for configuring VMs. This task requires more sophisticated and specific tools that range from shell scripts (e.g. bash shell scripts) to scripts for configuration management solutions like Puppet [6], Ansible [9], etc. As described by [10], full interoperability between VMs running in the cloud, Heat and other configuration management tools can be obtained only if the VMs are equipped with 'heat hooks', small pieces of code that take care of interoperating with Heat when the VM is running. Although shell scripts for configuring VMs can work without heat hooks, in order to use some advanced Heat constructs (e.g. SoftwareConfig), hook installation is mandatory. For this reason, this tool 'prepares' VM images with the hooks installed starting from the VM images with a plain operating system installation that are publicly available. For example, starting from the Ubuntu 14.04.2 LTS (Trusty Tahr) available at [11], the VM creates a VM image named as trusty-server-cloudimg-amd64-heat-hooks that can fully interoperate with Heat. The VM image creation is obtained using **diskimage-builder** [12], a set of tools implementing the "*core functionality for building disk images, file system images and ramdisk images for use with OpenStack*". The obtained images, should be used for implementing applications running on OpenStack instead of their "plain" counterparts freely available on Internet because they allow a higher level of interoperability with OpenStack Heat. It's worth mentioning that those images are used for implementing SCP components like the DB Layer and the Monitoring Layer. The tool is composed of the following scripts: - HeatImageBuilder.yaml: a YAML (Heat) script creating a plain Ubuntu VM Instance - HeatImageBuilder.sh: a bash script installing diskimage-builder software package and running it for creating VM images with Heat hooks. The tool requires: - a plain Ubuntu 14.04.2 LTS (Trusty Tahr) image on the hosting OpenStack cloud - the credentials of an OpenStack user with member role for the tenant where the cloud objects are created By default the tool populates the image repository of the hosting cloud with a VM image called trusty-server-cloudimg-amd64-heat-hooks, implementing an Ubuntu 14.04.2 LTS (Trusty Tahr) installation with Heat hooks. The VM created by the YAML script can be used for creating other fully-interoperable VM images (see [12]) for several operating systems [13] and [14].
Input	**Key**: the name of a keypair used for launching the VM **Flavor**: the flavor used for launching the VM **Private_Network**: the name of the private network to connect the VM to. By default it is set to SCP_Network, created by Network.yaml **__ OS_USERNAME__:** OpenStack account username **__ OS_PASSWORD__:** OpenStack account password **__ OS_TENANT_NAME__**: OpenStack tenant **__OS_AUTH_URL__**: OpenStack authentication URL *Notes:* - *the parameters prefixed and post-fixed with "__" are passed as parameters to the bash script* - *the __OS* parameters are used to upload the created VM image on the cloud image repository*
Output	None

Name	MySQLSingleNode.yaml
Languages	YAML (heat script) and bash shell script
Description	This tool creates an Ubuntu 14.04.2 LTS (Trusty Tahr) VM with an installation of MySQL database engine. It can be used as a starting point for the implementation of YAML (Heat) scripts for deploying single node applications requiring a MySQL database. The script creates a VM and a block storage volume used for storing the data. The scripts also installs Duplicity [15], the backup solution identified in the SCP architecture.
Input	**Key**: the name of a keypair used for launching the VM **Flavor**: the flavor used for launching the VM **Private_Network**: the name of the private network to connect the VM to. By default it is set to SCP_Network, created by Network.yaml **Volume_Size**: the size, in Gigabytes, of the volume for storing the database
Output	**DB_Root_Password**: the password, automatically generated by the script, for the DB administrator

Name	PostgreSQLSingleNode.yaml
Languages	YAML (heat script) and bash shell script
Description	This tool creates a Ubuntu 14.04.2 LTS (Trusty Tahr) VM with an installation of PostgreSQL database engine. It can be used as a starting point for the implementation of YAML (Heat) scripts for deploying single node applications requiring a PostgreSQL database. The script creates a VM and a volume which is used for storing the data. The scripts also installs Duplicity [15], the backup solution identified in the SCP architecture.
Input	**Key**: the name of a keypair used for launching the VM **Flavor**: the flavor used for launching the VM **Private_Network**: the name of the private network to connect the VM to. By default it is set to SCP_Network, created by Network.yaml **Volume_Size**: the size, in Gigabytes, of the volume for storing the database
Output	**DB_Root_Password**: the password, automatically generated by the script, for the DB administrator

2 Summary and Conclusions

This document has described the current state of art of the Cloud Application Template Catalogue that provides tools for facilitating the deployment of STORM CLOUDS applications.

The latest activities of WP2 have been focussed on implementing tools for automating the deployment of applications: the next steps will continue on this direction both for implementing SCP components (e.g. DB Layer) and for revisiting and improving the method currently used for deploying the STORM CLOUDS applications.

References

[1] "Surfing Towards the Opportunity of Real Migration to CLOUD-based public Services," STORM CLOUDS Consortium, November 2013.

[2] "D2.4.1 - Cloud Application Template Catalogue," STORM CLOUDS Consortium, 2014.

[3] "D2.2.2 - Storm Clouds Platform Architectural Design," STORM CLOUDS Consortium, 2015.

[4] "OpenStack Heat - Wiki Page," [Online]. Available: https://wiki.openstack.org/wiki/Heat. [Accessed Jan 2015].

[5] "OpenStack Heat - OpenStack Resource Types," [Online]. Available: http://docs.openstack.org/developer/heat/template_guide/openstack.html. [Accessed June 2015].

[6] "Puppet Open Source," [Online]. Available: http://puppetlabs.com/puppet/puppet-open-source. [Accessed Jan 2015].

[7] "D2.3.2 Storm Clouds Platform Implementation Status Report," STORM CLOUDS Consortium, 2014.

[8] "RFC1918 - Address Allocation for Private Internets," The Internet Engineering Task Force (IETF®), Feb 1996. [Online]. Available: https://www.ietf.org/rfc/rfc1918.txt. [Accessed June 2015].

[9] "Ansible Main Page," [Online]. Available: http://www.ansible.com/home. [Accessed July 2015].

[10] S. Hardy, "Heat SoftwareConfig resources - primer/overview," [Online]. Available: http://hardysteven.blogspot.it/2015/05/heat-softwareconfig-resources.html. [Accessed July 2015].

[11] "Ubuntu 14.04.2 LTS (Trusty Tahr)," [Online]. Available: http://releases.ubuntu.com/14.04/. [Accessed June 2015].

[12] "Openstack diskimage-builder - GitHub Page," [Online]. Available: https://github.com/openstack/diskimage-builder. [Accessed June 2015].

[13] "Openstack diskimage-builder - Documentation," [Online]. Available: http://docs.openstack.org/developer/diskimage-builder/. [Accessed July 2015].

[14] "OpenStack diskimage-builder - Supported Distributions," [Online]. Available: http://docs.openstack.org/developer/diskimage-builder/user_guide/supported_distros.html.

[15] "Duplicity Main Page," [Online]. Available: http://duplicity.nongnu.org/. [Accessed Jan 2015].

