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 D2.4.3
Cloud Application Template Catalogue

Workpackage: WP2

Version: 1.0

Date: 31/01/2016

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), Marco Battarra (Hewlett Packard Italiana S.r.l)

Reviewed by: Alkiviadis Giannakoulias (European Dynamics)

Version Control

Modified by	Date	Version	Comments
Marco Consonni Marco Battara	15/01/2016	Draft	First draft
Marco Consonni	14/01/2016	0.1	Ready for Internal Review
Marco Consonni Alkiviadis Giannakoulias	18/01/2016	0.2	Internal Review done
Marco Consonni	18/01/2016	1.0	First Version

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 [1].

Work Package 2 (WP2) of the 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 platform, can be used for facilitating the deployment of cloud-based applications.

This document is the third issue of the iterative deliverable "Cloud-Application Template Catalogue" that describes the current status of the tool library; the following table summarizes the evolution of the tools throughout the project milestones:

MILESTONE	TITLE	NOTES
M9	Prefab Images	A list of prefabricated Virtual Machine (VM) images obtained by manually installing the software packages. The images are used as the "starting point" for manually deploying the application services
M15	Basic Automation	Tools for automating the creation of the prefab VM images
M24	Fully Automated Application Deployment	A set of tools for automatically deploying the applications

Table 0-1 Cloud-Application Template Catalogue

The table reads as follows:

- **MILESTONE**: delivery date;
- **TITLE**: brief title summarizing the implementation stage
- **NOTES**: more detailed description of the implementation status.

This document reports the current state of art of the tool library and describes the implemented tools.

Table of Contents

Abbreviations

Acronym	Description
CA	Certification Authority
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
FQDN	Fully Qualified Domain Name
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure
IaaS	Infrastructure as a Service
IT	Information Technology
N/A	Not Available or Not Applicable
PA	Public Authority
PGP	Pretty Good Privacy
SCP	STORM Cloud Platform
TBD	To Be Defined
TBW	To Be Written
URL	Uniform Resource Locator
VIP	Virtual IP (address)
VRRP	Virtual Router Redundancy Protocol
VM	Virtual Machine
WP	Work Package
YAML	YAML Ain't Markup Language

1 Introduction

STORM CLOUDS Platform (SCP), the cloud platform hosting the application services of the STORM CLOUDS project, implements automation functions that allows cloud users to automatically deploy applications. Heat [2], the OpenStack orchestration engine, is the software module that implements such functionality.

In order to use Heat, a cloud user writes *Heat Templates* that declare the list of objects to create. These objects are Infrastructure-as-a-Service (IaaS) elements like virtual machines, virtual volumes, virtual network trunks, virtual routers, etc. When the user wants to deploy an application in the cloud, s/he submits a Heat Template to Heat that, in turn, creates all the objects listed into it. Heat Templates are written in HOT a YAML-based language.

Virtual Machines (or Servers, for using HOT terminology) host the application software and they need to be instructed for installing and configuring the software components of the application. This can't be obtained solely using HOT language but requires some language that is interpreted by the operating system (or some other software) running on the virtual machine(s).

For this purpose, Heat Templates contain references to external scripts; the following figure shows the structure of a typical Heat Template for deploying an application.

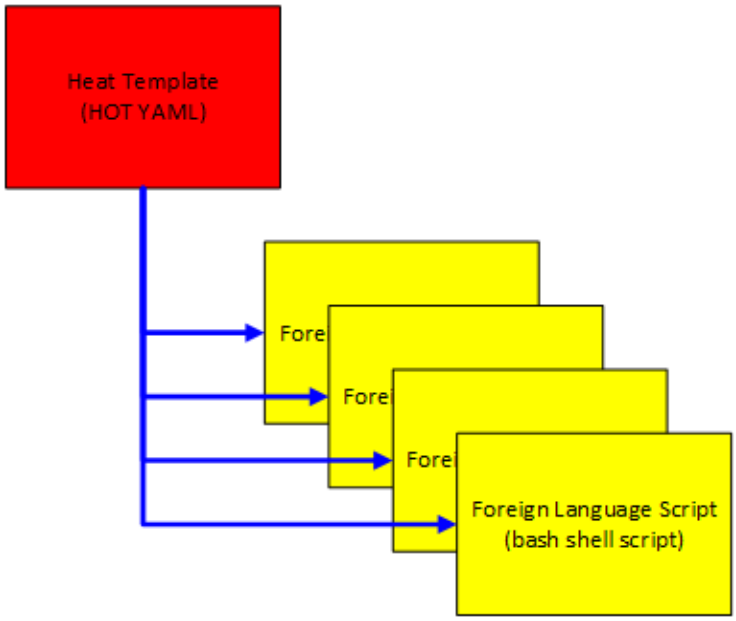

Figure 1-1 – Heat Template for Deploying Applications

When the user executes a HOT YAML Template, Heat creates the virtual machine(s) listed in the template and upon VM boot sends the shell scripts to it (them) for being interpreted.

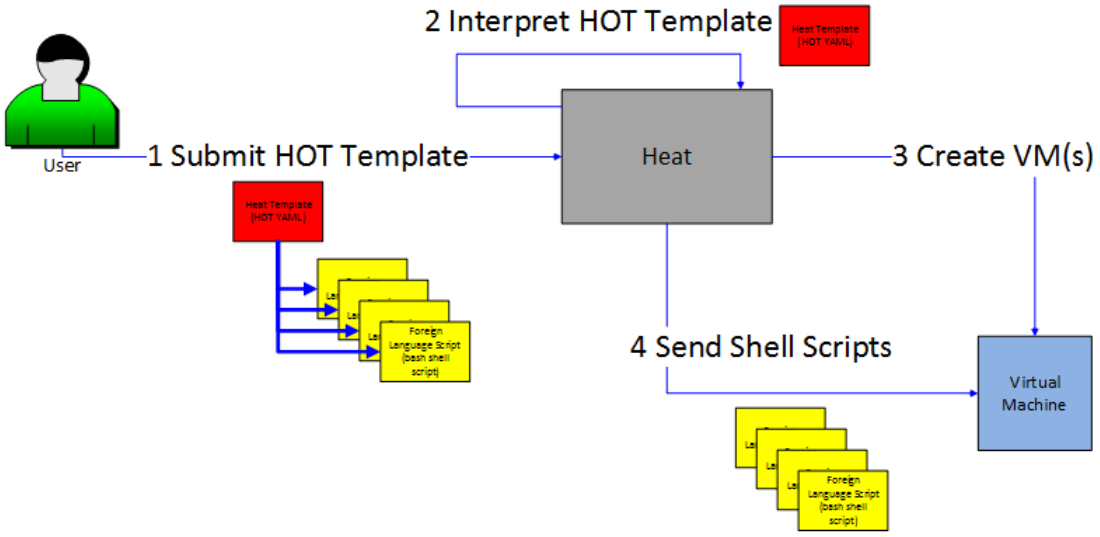

Figure 1-2 – Orchestrating Virtual Machine Creation

This document describes the HOT templates and the shell scripts implemented in the project for automating the deployment of the STORM CLOUDS application catalogue.

They are classified into the following categories:

- **General Purpose Tools**: tools used by all applications;
- **Application Specific Tools**: tools specifically designed for deploying single applications in the catalogue.

For each tool the following information is provided:

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

2 General Purpose Tools

The tools described in this section are used for implementing functions common to all applications running on SCP and promote standard practices in the deployment of the applications as well as on the use of the resources available in the cloud platform (e.g. all the applications use the Object Store for saving backup-data, the name of the backup data set follow the same naming conventions, all the virtual machines hosting applications are named according the same naming conventions, etc.).

These tools fall in the following categories:

1. Bash scripts used to configure the VM(s) hosting the application software: these are collectively called **Bash Script Tools**.
2. Tool for implementing a HTTP proxy server that can be used for enabling complex deployment scenarios like, high-availability/load-balancing configurations of single applications, availability of several applications at a single Fully Qualified Domain Name, or combinations of the two scenarios. This tool is called **HA Proxy Tool**.

2.1 Bash Script Tools

The following tables describe the generic bash script tools in details.

Name	apache_app-configure.sh	
Languages	Bash shell	
Description	The script configures and enables the Apache virtual host files, dictating how the Apache web server will respond to various domain requests. If either a Fully Qualified Domain Name (FQDN) or an IP address is provided as parameter, the configuration redirects all the HTTP requests from port 80 to port 433 (HTTPS port). This happens because the VM is supposed to be directly exposed to Internet without any HTTP proxy intermediation. In such a case, the script autonomously creates a key-pair along with a self-signed digital certificate: they are used for securing the HTTPS connections. Municipalities are free to substitute these certificates with 'real' ones, released by a Certification Authority (CA). On the other hand, if the application is deployed behind a HTTP proxy, the script configures Apache to receive requests only on port 80, the default HTTP traffic port. In such a case, it's up to the HTTP proxy to secure the communication channel between the clients and the server.	
Input	__FQDN_or_IP__	Fully Qualified Domain Name or Internet IP address of the application. If set to empty string (i.e. ""), the assumption is that the application is not directly exposed to Internet
	__ADMIN_EMAIL__	e-mail address of the system administrator
	__SERVER_NAME__	name of the HTTP server
	__APP_DIRECTORY__	directory of the application
	__MUNICIPALITY__	name of the municipality
Output	None	

Name	duplicity_setup.sh	
Languages	Bash shell	
Description	The script configures Duplicity [3], the software tool - indicated in SCP architecture [4] – for executing backups of application data. Duplicity requires that the user provides a PGP private key that is used for encrypting data. This script automatically provides a 'convenience' key; in case the user requires more security, s/he can substitute the automatically generated key with another –maybe more trusted – key.	
Input	__Encrypt_Sign_Key__	name of the encryption key
	__KeyPairPassphrase__	a password for protecting the encryption key
Output	None	

Name	backup_setup.sh	
Languages	Bash shell	
Description	This script configures the VM for executing the application data backup. It creates on the VM two scripts for the backup: **backup_full.sh** and **backup_incremental.sh** respectvely performing a full and an incrementlal backup. These scripts are scheduled as crontab jobs: full backup every week on Sunday at 04:00 AM, incremental backup every day at 04:00 AM. As the actual data to backup depends on the application (e.g. different application have different databases and/or files to backup), these two scripts just take all the files in a 'well known' directory, create a data set out of them and send the data set to the OpenStack object store. The data to backup are expected to be prepared before. For this reason, backup_setup.sh receives two parameters (__Full_Backup_Script__ and __Incremental_Backup_Script__) that are supposed to prepare the backup data in the well-known directory. The name of the object saved in the OpenStack object-store is created by concatenating the name of the municipality (__MUNICIPALITY__) and the name of the application (__Appl_Name__).	
Input	__MUNICIPALITY__	name of the municipality
	__Appl_Name__	name of the application
	__Encrypt_Sign_Key__	name of the encryption key
	__Full_Backup_Script__	name of the script for creating a full backup dataset
	__Incremental_Backup_Script__	name of the script for creating an incremental backup dataset
Output	None	

Name	mysqldump_prepare-script.sh	
Languages	Bash shell	
Description	This tool creates a script for preparing the data stored in a MySQL database for backup. The created script calls the **mysqldump** utility.	
Input	__DB_Host__	name or IP address of the server running MySQL
	__DB_Database__	name f the database to backup
	__DB_User__	user of the database
	__DB_Password__	password
	__DB_Dump_File__	name of the DB dump file
	__DB_Dump_Script__	name of the script to create
Output	None	

Name	postgresqldump_prepare-script.sh	
Languages	Bash shell	
Description	This tool creates a script for preparing data stored in a PostgreSQL database for backup. The created script calls the **pg_dump** utility.	
Input	__DB_Host__	name or IP address of the server running PostgreSQL
	__DB_Database__	name f the database to backup
	__DB_User__	user of the database
	__DB_Password__	Password
	__DB_Dump_File__	name of the DB dump file
	__DB_Dump_Script__	name of the script to create
Output	None	

2.2 HA Proxy Tool

The HA Proxy Tool is the implementation of the High Availability HTTP Proxy mentioned in the SCP architecture (see [4], Appendix A – Active-Passive Cluster on Linux Platform). This tool implements a HA-Proxy [5] cluster allowing complex deployment scenarios where:

- a single application is deployed on several virtual machines in order to implement load-balancing and high-availability features;
- a municipality runs several applications that are accessed through a single IP address or a single FQDN.

Combinations of the two above mentioned scenarios are also supported as illustrated in the following figure.

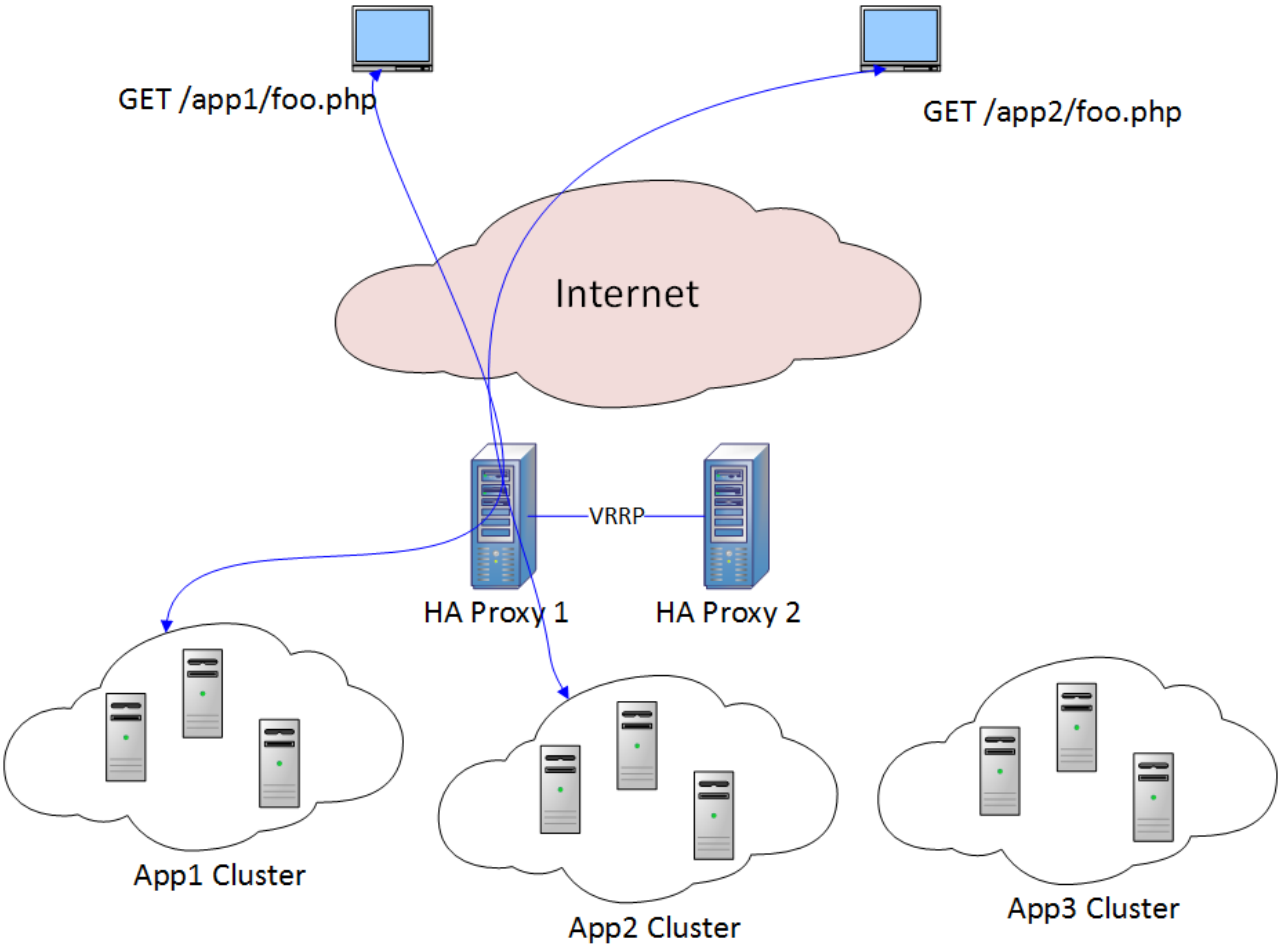

Figure 2-1 – HA Proxy Tool

The tool implements the solution described in [6].

Name	SCP-HAProxy.yaml	
Languages	HOT YAML	
Description	This script creates a two node HA Proxy cluster: one node is set as Master, the other as Backup. The nodes, in addition to having a dedicated IP address, share: • a unique Virtual IP address (VIP) that is managed using Virtual Router Redundancy Protocol (VRRP), and • a unique Floating IP Address – associated to the VIP – for exposing the cluster on Internet.	
Input	PrivateNetwork_Id	identifier of the OpenStack private network to connect the nodes to
	PrivateSubnet_Id	identifier of the OpenStack subnet to connect the nodes to
	HAFloatingIP_Id	identifier of the OpenStack Floating IP address
	HAProxyNode1IP	private IP address of node 1
	HAProxyNode2IP	private IP address of node 2
	HAProxySharedIP	private VIP address
	Municipality	name of the municipality the cluster is activated for
Output	None	

Name	haproxy_setup.sh	
Languages	Bash shell	
Description	This script prepares the HA proxy nodes by installing and configuring the following software packages: - Apache, used as HTTP server front end - Keepalived, implementing VRRP protocol - Haproxy, implementing high-availability and load-balancing HTTTP proxy functions	
Input	__KEEPALIVED_STATE__	MASTER or BACKUP
	__KEEPALIVED_PRIORITY__	keepalived priority of the node (100 = master, 50 = backup)
	__SHARED_IP_MASK__	mask of the Virtual IP address
	__SHARED_IP__	shared Virtual IP address
Output	None	

3 Application Specific Tools

The tools described in this section are designed for the deployment of single applications.

As a general rule and as already described in section 1, the activation of an application requires a HOT YAML script that takes care of orchestrating the whole deployment: it creates the virtual machine(s) for hosting the application and includes other files containing scripts for the installation and configuration of software modules. For this reason, every application in the catalogue is deployed by a specific HOT YAML script that, after creating a VM, invokes bash shell scripts on the VM for installing and configuring the software. Some bash shell scripts are specific to the application others can perform general purpose tasks applicable to all applications deployed on top of SCP.

The following figure presents a typical scenario.

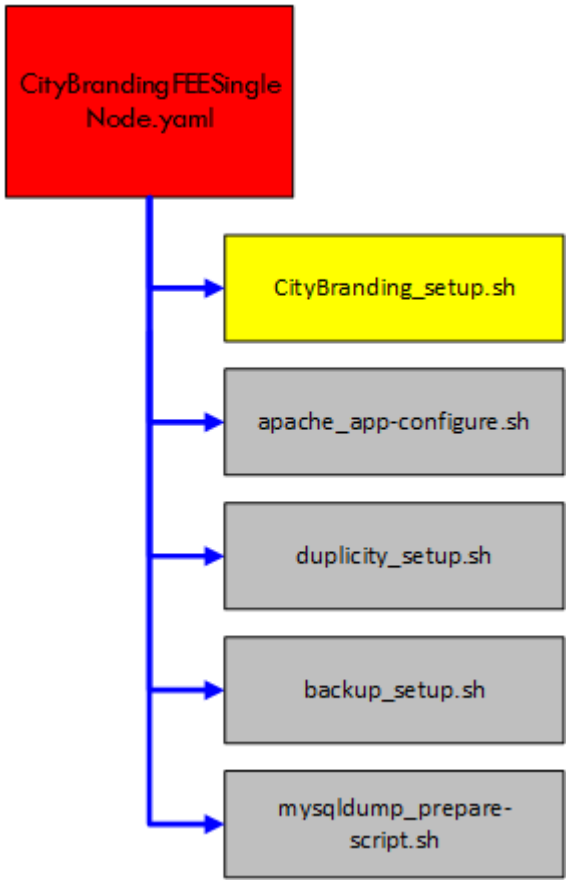

Figure 3-1 – Application Specific Deployment Tool

- One HOT YAML script orchestrates the deployment of the application (in red);
- One or more bash shell scripts install/configure the application (in yellow);
- General purpose bash shell scripts install/configure common software packages or perform common tasks (in grey).

The following sections describe the tools implemented for each application.

3.1 City Branding

This section describes the tools for deploying the "City Branding" application.

Name	CityBrandingFEESingleNode.yaml	
Languages	HOT YAML	
Description	This script creates a single node (i.e. a single VM) hosting the "City Branding" application. It calls CityBranding_setup.sh bash script for installing the application software packages, and apache_app-configure.sh, backup_setup.sh, duplicity_setup.sh mysqldump_prepare-script.sh for more generic installation/configuration tasks.	
Input	Municipality	name of the municipality the application is deployed for
	Server_Name	name of the server (VM)

	Flavor	OpenStack flavour
	Network_Id	identifier of the OpenStack network to connect the VM to
	Subnet_Id	identifier of the OpenStack subnet to connect the VM to
	IP_Address	fixed IP address
	DB_User	user of the MySQL shared database (SCP service)
	DB_Password	database password
	DB_Database	database name
	DB_Dump_File	name of the database dump file (for backup)
	KeyPairPassphrase	password for the convenience PGP key-pair
	KeyPairEmail	e-mail for
	Encrypt_Sign_Key	identifier of the encryption key
	FQDN_or_IP	Fully Qualified Domain Name or Internet IP address of the application
	SMTP_Host	IP address of FQDN of the Simple Mail Transfer Protocol Server (SMTP) for sending mail messages to
	SMTP_From	Name "From" name of mail messages
	SMTP_User	user of the SMTP server
	SMTP_Password	SMTP server password
Output	None	

Name	CityBranding_setup.sh	
Languages	Bash Shell	
Description	This script installs and configures the software packages of the City Branding application. It also creates the database, for storing application data, on the shared database service implemented by SCP.	
Input	__DB_Database__	name of the DB to create
	__DB_User__	user of the DB
	__DB_Password__	Password
	__Appl_Name__	name of the application (default is "citybranding")
	__SMTP_Host__	IP address of FQDN of the Simple Mail Transfer Protocol Server (SMTP) for sending mail messages to
	__SMTP_From_Name__	"From" name of mail messages
	__SMTP_User__	user of the SMTP server
	__SMTP_Password__	SMTP server password
	__MUNICIPALITY__	name of the municipality the application s deployed for
Output	None	

3.2 Cloud Funding

This section describes the tools for deploying Cloud Funding application.

Name	CloudFundingFEESingleNode.yaml	
Languages	HOT YAML	
Description	This script creates a single node hosting the Cloud Funding application. It calls CloudFunding_setup.sh bash script for installing the Cloud Funding, and apache_app-configure.sh, backup_setup.sh, duplicity_setup.sh mysqldump_prepare-script.sh for more generic installation/configuration tasks.	
input	Server_Name	name of the server (VM)
	Municipality	name of the municipality the application is deployed for
	Flavor:	OpenStack flavour
	Network_Id	identifier of the OpenStack network to connect the VM to
	Subnet_Id	identifier of the OpenStack subnet to connect the VM to
	IP_Address:	fixed IP address
	DB_User:	user of the MySQL shared database (SCP service)

	DB_Password	database password
	DB_Database	database name
	DB_Dump_File	name of the database dump file (for backup)
	KeyPairPassphrase	password for the convenience PGP key-pair
	KeyPairEmail	e-mail for
	Encrypt_Sign_Key	identifier of the encryption key
	FQDN_or_IP	Fully Qualified Domain Name or Internet IP address of the application
Output	None	

Name	CloudFunding_setup.sh	
Languages	Bash Shell	
Description	This script installs and configures the software packages of the Cloud Funding application. It also creates the database, for storing application data, on the shared database service implemented by SCP.	
Input	__DB_Database__	name of the DB to create
	__DB_User__	user of the DB
	__DB_Password__	password
	__Appl_Name__	name of the application (default is “cloudfunding”)
	__MUNICIPALITY__	name of the municipality the application s deployed for
Output	None	

3.3 Have Your Say

This section describes the tools for deploying Have Your Say application.

Name	PPGISFEESingleNode.yaml	
Languages	HOT YAML	
Description	This script creates a single node hosting the Have Your Say application. It calls ppgis_setup.sh bash script for installing Have Your Say application, and apache_app-configure.sh, backup_setup.sh, duplicity_setup.sh postgresqldump_prepare-script.sh for more generic installation/configuration tasks.	
Input	Municipality	name of the municipality the application is deployed for
	Server_Name	name of the server (VM)
	Flavor	OpenStack flavor
	Network_Id	identifier of the OpenStack network to connect the VM to
	Subnet_Id	identifier of the OpenStack subnet to connect the VM to
	IP_Address	fixed IP address
	DB_User	user of the MySQL shared database (SCP service)
	DB_Password	database password
	DB_Database	database name
	DB_Dump_File	name of the database dump file (for backup)
	KeyPairPassphrase	password for the convenience PGP key-pair
	Encrypt_Sign_Key	identifier of the encryption key
	FQDN_or_IP	Fully Qualified Domain Name or Internet IP address of the application
Output	None	

Name	ppgis_setup.sh	
Languages	Bash Shell	
Description	This script installs and configures the software packages of the Cloud Funding application. It also creates the database, for storing application data, on the shared database service implemented by SCP.	
Input	__Appl_Name__	name of the application (default is “citybranding”
	__MUNICIPALITY__	name of the municipality the application s deployed for
Output	None	

Note: due to some application limitations, at the time of writing the script uses a single database; this limitation is going to be removed in the next releases.

3.4 Virtual City Market

This section describes the tools for deploying Virtual City Market application.

Name	VCMFEESingleNode.yaml	
Languages	HOT YAML	
Description	This script creates a single node hosting Virtual City Market application. It calls vcm_setup.sh bash script for installing the Virtual City Market software, and apache_app-configure.sh, backup_setup.sh, duplicity_setup.sh mysqldump_prepare-script.sh for more generic installation/configuration tasks.	
Input	Municipality	name of the municipality the application is deployed for
	Server_Name	name of the server (VM)
	Flavor	OpenStack flavour
	Network_Id	identifier of the OpenStack network to connect the VM to
	Subnet_Id	identifier of the OpenStack subnet to connect the VM to
	IP_Address	fixed IP address
	DB_User	user of the MySQL shared database (SCP service)
	DB_Password	database password
	DB_Database	database name
	DB_Dump_File	name of the database dump file (for backup)
	KeyPairPassphrase	password for the convenience PGP key-pair
	Encrypt_Sign_Key	identifier of the encryption key
	FQDN_or_IP	fully Qualified Domain Name or Internet IP address of the application
	SMTP_Host	IP address of FQDN of the Simple Mail Transfer Protocol Server (SMTP) for sending mail messages to
	SMTP_From_Name	"From" name of mail messages
	SMTP_User	user of the SMTP server
	SMTP_Password	SMTP server password
Output	None	

Name	vcm_setup.sh	
Languages	Bash Shell	
Description	This script installs and configures the software packages of the Virtual City Market application. It also creates the database, for storing application data, on the shared database service implemented by SCP.	
Input	__DB_Database__	name of the DB to create
	__DB_User__	user of the DB
	__DB_Password__	password
	__Appl_Name__	name of the application (default is "cloudfunding")
	__MUNICIPALITY__	name of the municipality the application s deployed for
Output	None	

3.5 Vive

This section describes the tools for deploying Vive application.

Name	ViveFEESingleNode.yaml	
Languages	HOT YAML	
Description	This script creates a single node hosting the Vive application. It calls vive_setup.sh bash script for installing the Vive software packages, and apache_app-configure.sh, backup_setup.sh, duplicity_setup.sh mysqldump_prepare-script.sh for more generic installation/configuration tasks.	
Input	Municipality	name of the municipality the application is deployed for

	Server_Name	name of the server (VM)
	Flavor	OpenStack flavour
	Network_Id	identifier of the OpenStack network to connect the VM to
	Subnet_Id	identifier of the OpenStack subnet to connect the VM to
	IP_Address	fixed IP address
	DB_User	user of the MySQL shared database (SCP service)
	DB_Password	database password
	DB_Database	database name
	DB_Dump_File	name of the database dump file (for backup)
	KeyPairPassphrase	password for the convenience PGP key-pair
	Encrypt_Sign_Key	identifier of the encryption key
	FQDN_or_IP	fully Qualified Domain Name or Internet IP address of the application
	SMTP_Host	IP address of FQDN of the Simple Mail Transfer Protocol Server (SMTP) for sending mail messages to
	SMTP_From_Name	"From" name of mail messages
	SMTP_User	user of the SMTP server
	SMTP_Password	SMTP server password
Output	None	

Name	vive_setup.sh	
Languages	Bash Shell	
Description	This script installs and configures the software packages of the Vive application. It also creates the database, for storing application data, on the shared database service implemented by SCP.	
Input	__DB_Database__	name of the DB to create
	__DB_User__	user of the DB
	__DB_Password__	password
	__Appl_Name__	name of the application (default is "vive")
	__MUNICIPALITY__	name of the municipality the application s deployed for
Output	None	

4 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. With the tools it is possible to deploy new application instances very quickly because the process is fully automated and the tools are highly parameterized.

The document will be updated in the future for reflecting the actual state of the tool library.

References

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

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

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

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

[5] "HAProxy - Main Page," [Online]. Available: http://www.haproxy.org/. [Accessed Jan 2015].

[6] "Load balancing and HA for multiple applications with Apache, HAProxy and keepalived," [Online]. Available: http://backreference.org/2012/04/25/load-balancing-and-ha-for-multiple-applications-with-apache-haproxy-and-keepalived/. [Accessed 15 2016].

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

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

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

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

[11] "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].

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

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

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

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

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

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

