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 3.1.4

Deployment of the services in the Cloud Infrastructure for Cloudification customization and testing

Work Package: WP3

Version: 1.0

Date: 31/01/2017

Status: WP leader accepted

Nature: Other

Dissemination Level: PUBLIC

Editor: Alkiviadis Giannakoulias (European Dynamics SA)

Authors: Alkiviadis Giannakoulias (European Dynamics SA)

Reviewed by: Agustin González-Quel (RTDI)

Version Control

Modified by	Date	Version	Comments
Alkiviadis Giannakoulias	17/08/2016	0.1	1st version.
Alkiviadis Giannakoulias	10/01/2017	0.2	Updated contents for SCP V3.0
Alkiviadis Giannakoulias	25/01/2017	0.3	Updated introduction, summary and added security tests and cost calculations. Ready for Review

Executive Summary

Work Package 3 (WP3) of the STORM CLOUDS project aims to adapt the selected services and to integrate them to the STORM CLOUDS Platform (SCP) infrastructure created in WP2, while at the same time create the tools and procedures to facilitate services migration to the cloud. Once applications are successfully tested in the private cloud infrastructure they will be deployed in the pilot cities public cloud infrastructure.

In this fourth iterative and final deliverable release, we address:

- Deployment of services in the 4th innovation/cloudification cycle.
- A number of critical security issues that were identified during vulnerability analysis.

Regarding data protection the procedure and tools presented in D3.1.2 were used resulting in the implementation of a backup scheme similar to the one already described and implemented in D3.1.2.

Technical details regarding the adaptation of the services and integration to the cloud infrastructure are provided in the Annexes.

Although in the previous version of the deliverable we have described that the High Availability options, through clustering, would be investigated this option was dropped due to increased cost of ownership.

The document is organized into the following sections:

Section 2 – Security: it describes the necessary modifications to the services in order to address a number of security vulnerabilities that were detected in the applications prior to cloudification, such as transport layer vulnerabilities, web applications authentication vulnerabilities and application specific security issues.

Section 3 – Summary and conclusions

At the time of writing, adaptation of new services and integration into the private cloud infrastructure (hosted on physical infrastructure housed at Hewlett Packard's premises used by project participants for development and testing purposes) used the same adaptation procedure and tools described in [2] section 2. Similarly the same data protection and automation procedure and tools described in [3] section 3 and 4 respectively were used.

Table of Contents

List of Figures

List of Tables

Abbreviations

Acronym	Description
CSC	Cloud Service Consumer
CSP	Cloud Service Provider
ESAPI	OWASP Enterprise Security API
HA	High Availability
HOT	Heat Orchestration Template
HTML	HyperText Markup Language
HTTPS	Hyper Text Transfer Protocol Secure
LUKS	Linux Unified Key Setup-on-disk-format
OWASP	Open Web Application Security Project
POODLE	Padding Oracle On Downgraded Legacy Encryption
RC4	Rivest Cipher 4
RTO	Recovery Time Objective
SCP	STORM CLOUDS Platform
SFTP	SFTP File Transfer Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
XSS	Cross Site Scripting

1 Introduction

During the 4th innovation cycle we have successfully cloudified the:

- "Improve My City" application from URENIO that enables citizens to report non-emergency local problems such as potholes, illegal trash dumping, faulty street lights, broken tiles on sidewalks and illegal advertising boards. Moreover, it allows citizens to suggest solutions for improving the environment of their neighbourhood.
- "Location Plans" application from Agueda that prepare and print location plans. These prints are often requested by public authorities for legal purposes.
- "OPENDATA" application from Miskolc that sources local events databases which store data in an open platform. The service provides XML, PDF, SQL and JSON format data streams for internal and external developers free of charge.

Meanwhile previsouly cloudified applications were launched in other municipalities:

1. "Cloud Funding" from Thessaloniki was launched for Veria (http://smartcity.veria.gr/crowd-funding/);
2. "Virtual City Market" from Thessaloniki was launched for Veria (http://smartcity.veria.gr/virtual-city-market/el/index.html);
3. "City Branding" from Thessaloniki was launched for Veria (http://smartcity.veria.gr/citybranding/en/);
4. "Improve My City" from URENIO was launched for Athens (http://178.239.182.56/improvemycity/);
5. "Live the City" from Valladolid was launched for Agueda (http://vive.sig.cm-agueda.pt/);
6. "Have Your Say" from Agueda was launched for Thessaloniki (https://smartcity.thessaloniki.gr/haveyoursay/);
7. "Cloud Funding" from Thessaloniki was launched for Valladolid (http://80.247.66.86/crowd-funding/);
8. "Have Your Say" from Agueda was launched for Guimaraes (http://178.239.183.156/);
9. "Virtual City Market" from Thessaloniki was launched for Valladolid.

However, in order to better support the exploitation of the project we performed a financial excercise on the deployment model/architecture (SCP v2.0). A tool was created (see section 3), that allowed us to calculate the cost of ownership, showing that the current SCP 'overhead' is around 70% with 10 hosted applications and drops to 50% with 20 hosted applications. This immediately indicated that we have to simplify the platform, by removing High Availability (Database and File Clustering) from our design.

As a result all applications have:

- Updated installation/configuration scripts to include the necessary security modifications;
- Updated configuration scripts to include the necessary software modules for supporting the backup strategy;
- New deployment scripts that take advantage of the latest simplified SCP architecture.

2 Security

The OWASP web application penetration testing methodology and the penetration testing tools already presented in [4] were used in order to identify critical security issues, resulting in services modifications in order to address them.

2.1 URENIO

2.1.1 Compliance at a Glance (Improve My City)

This section of the report is a summary and lists the number of alerts found according to individual compliance/risk categories:

OWASP Security Risks Categories	Total number of alerts
A1-Injection	4 (SQL Injection)
A2-Broken Authentication and Session Management	33 (Cookie set without HttpOnly flag) 20 (Password Autocomplete in browser)
A3-Cross-Site Scripting (XSS)	2 (Cross Site Scripting (Reflected)) 48 (Web Browser XSS Protection Not Enabled) 20 (Cross-Domain JavaScript Source File Inclusion)
A4-Insecure Direct Object References	✓ **No alerts in this category**
A5-Security Misconfiguration	48 (X-Content-Type-Options Header Missing)
A6-Sensitive Data Exposure	26 (Private IP Disclosure)
A7-Missing Function Level Access Control	✓ **No alerts in this category**
A8-Cross-Site Request Forgery (CSRF)	✓ **No alerts in this category**
A9-Using Components with Known Vulnerabilities	✓ **No alerts in this category**
A10-Unvalidated Redirects and Forward	48 (X-Frame-Options Header Not Set)

Table 1: URENIO (Improve My City) Compliance at a Glance

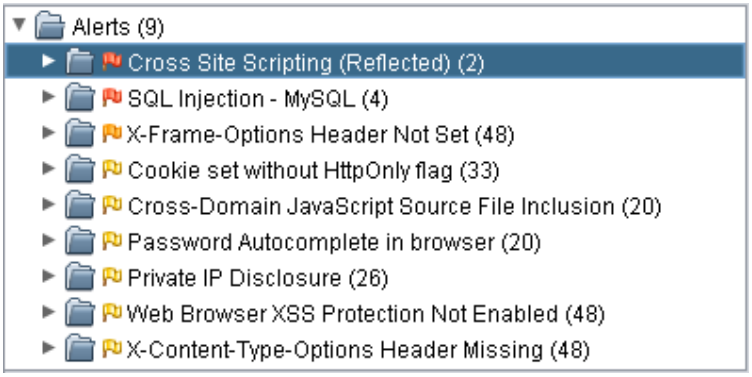

Figure 2-1: OWASP ZAP Compliance at a Glance – URENIO (Improve My City)

Figure 2-2: Vega Compliance at a Glance – URENIO (Improve My City)

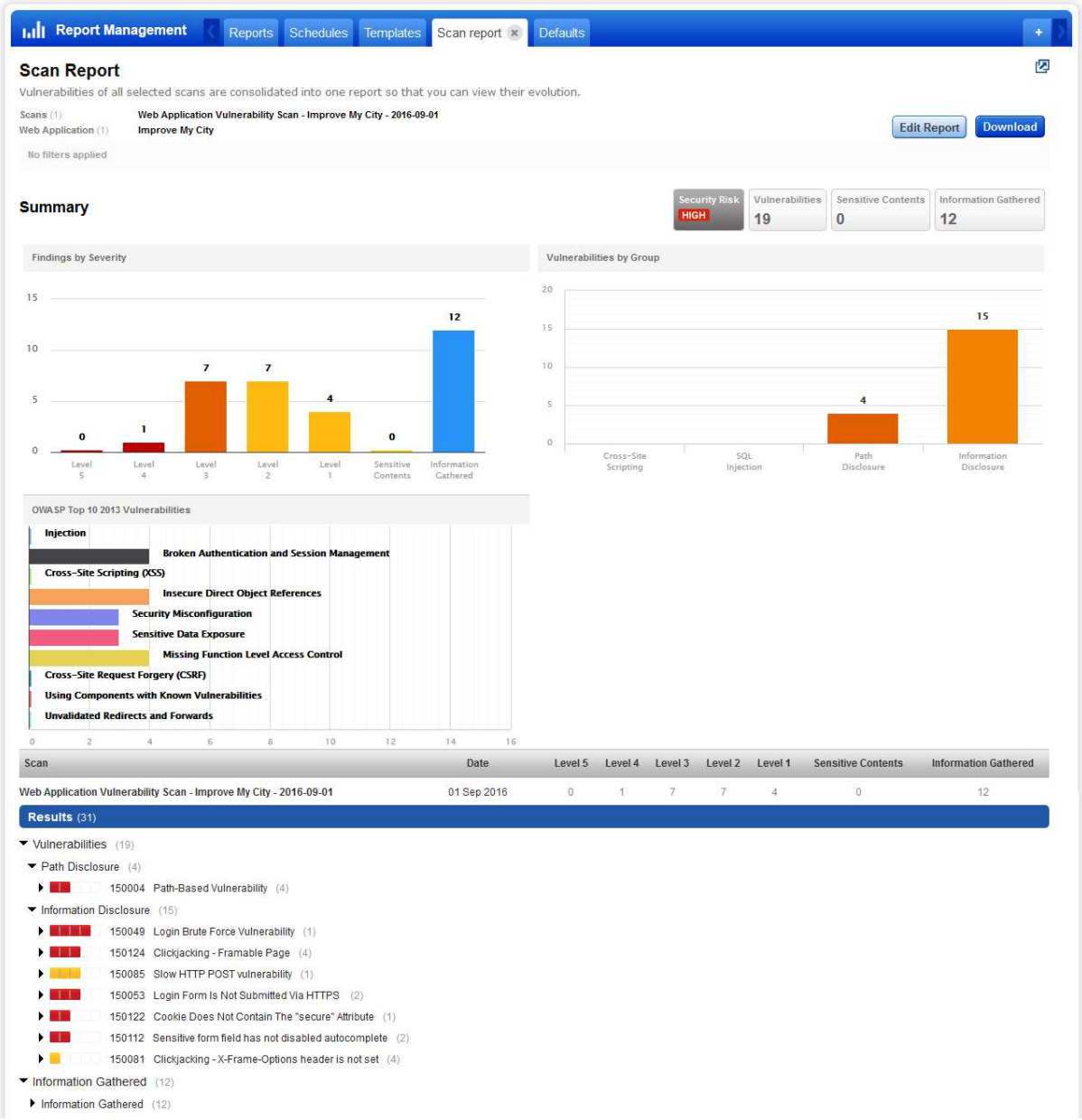

Figure 2-3: Qualys Web Application Scanning – URENIO (Improve My City)

2.2 Municipio de Águeda

2.2.1 Compliance at a Glance (Location Plans)

This section of the report is a summary and lists the number of alerts found according to individual compliance/risk categories:

OWASP Security Risks Categories	Total number of alerts
A1-Injection	✓ No alerts in this category
A2-Broken Authentication and Session Management	✓ No alerts in this category
A3-Cross-Site Scripting (XSS)	3 (Web Browser XSS Protection Not Enabled)
A4-Insecure Direct Object References	✓ No alerts in this category
A5-Security Misconfiguration	48 (X-Content-Type-Options Header Missing)

A6-Sensitive Data Exposure	✓ **No alerts in this category**
A7-Missing Function Level Access Control	✓ **No alerts in this category**
A8-Cross-Site Request Forgery (CSRF)	✓ **No alerts in this category**
A9-Using Components with Known Vulnerabilities	✓ **No alerts in this category**
A10-Unvalidated Redirects and Forward	3 (X-Frame-Options Header Not Set)

Table 2-2: Municipio de Águeda (Location Plans) Compliance at a Glance

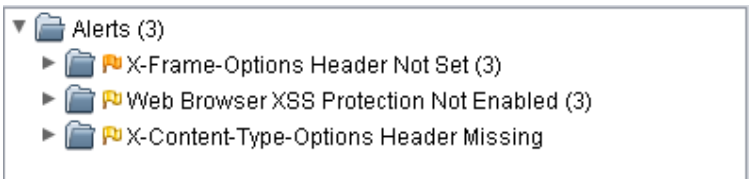

Figure 2-4: OWASP ZAP Compliance at a Glance – Municipio de Águeda (Location Plans)

2.3 City of Miskolc

2.3.1 Compliance at a Glance (OPENDATA)

This section of the report is a summary and lists the number of alerts found according to individual compliance/risk categories:

OWASP Security Risks Categories	**Total number of alerts**
A1-Injection	✓ **No alerts in this category**
A2-Broken Authentication and Session Management	✓ **No alerts in this category**
A3-Cross-Site Scripting (XSS)	360 (Web Browser XSS Protection Not Enabled)
A4-Insecure Direct Object References	1 (Directory listing)
A5-Security Misconfiguration	✓ **No alerts in this category**
A6-Sensitive Data Exposure	261 (Private IP Disclosure)
A7-Missing Function Level Access Control	✓ **No alerts in this category**
A8-Cross-Site Request Forgery (CSRF)	✓ **No alerts in this category**
A9-Using Components with Known Vulnerabilities	✓ **No alerts in this category**
A10-Unvalidated Redirects and Forward	35 (X-Frame-Options Header Not Set)

Table 2-3: Miskolc (OPENDATA) Compliance at a Glance

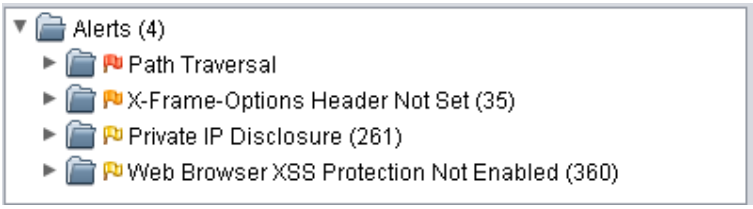

Figure 2-5: OWASP ZAP Compliance at a Glance – Miskolc (OPENDATA)

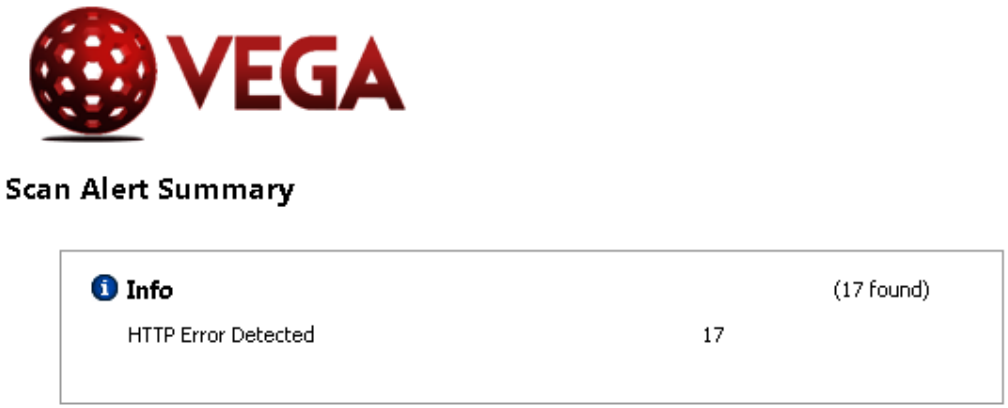

Figure 2-6: Vega Compliance at a Glance – Miskolc (OPENDATA)

3 SCP Cost Calculations

The following "tool" was used to calculate the cost of owning the "High Availability" clustering design in the SCP

SCP										
Tier	**Component**	**Element**	**Type**	**Flavor**	**Unit prc.**	**Qty**	**Incl.**	**Monthly**	**Yearly**	**Cost of owning**
Data	Database	DBNode1	Instance	e1standard.x4	€ 63.36	1	1	€ 63.36	€ 760.32	**16%**
Data	Database	DBNode2	Instance	e1standard.x4	€ 63.36	1	1	€ 63.36	€ 760.32	**16%**
Data	Database	DBNodeQ	Instance	e1standard.x2	€ 22.32	1	1	€ 22.32	€ 267.84	**6%**
Data	File System	GlusterFSNode1	Instance	e1standard.x4	€ 63.36	1	1	€ 63.36	€ 760.32	**16%**
Data	File System	GlusterFSNode2	Instance	e1standard.x4	€ 63.36	1	1	€ 63.36	€ 760.32	**16%**
Management	Monitoring	ZabbixNode1	Instance	e1standard.x2	€ 22.32	1	1	€ 22.32	€ 267.84	
Management	Monitoring	ZabbixNode2	Instance	e1standard.x2	€ 22.32	1	1	€ 22.32	€ 267.84	
Management	Monitoring	ZabbixNodeQ	Instance	e1standard.x2	€ 22.32	1	1	€ 22.32	€ 267.84	
Management	Console	Console	Instance	e1standard.x2	€ 22.32	1	1	€ 22.32	€ 267.84	
Data	Database	DBVolume1	Volume	Standard	€ 0.08	50	1	€ 4.00	€ 48.00	
Data	Database	DBVolume2	Volume	Standard	€ 0.08	50	1	€ 4.00	€ 48.00	
Data	Database	DBVolumeQ	Volume	Standard	€ 0.08	1	1	€ 0.08	€ 0.96	
Data	File System	GlusterFSVolume1	Volume	Standard	€ 0.08	100	1	€ 8.00	€ 96.00	
Data	File System	GlusterFSVolume2	Volume	Standard	€ 0.08	100	1	€ 8.00	€ 96.00	
Total								**€ 389.12**	**€ 4,669.44**	

Catalogue						
Image	**Size**	**Versions**	**Unit prc.**	**Qty**	**Monthly**	**Yearly**
trusty-server-cloudimg-amd64-heat-hook	2	3	0.045	1	0.27	3.24
Generic Application	3	3	0.045	8	3.24	38.88
Total					**3.51**	**42.12**

Platform Cost	Cost	Cost per VM
	€	€
Monthly	392.63	39.26

	€	€
Yearly	4,711.56	471.16
N. of App	10	
Data Tier	1	<-- yes=1, no=0

4 Summary and Conclusions

This document has described the deployment of new services to the cloud infrastructure for customization and testing as well as the necessary modifications to the services in order to address a number of security vulnerabilities detected in the applications prior to cloudification. In addition, we describe the actions performed to safeguard against vulnerabilities coming from virtualization technologies.

The procedure and the tools, presented in D3.1.3, for operational data backup have been used enabling municipalities to safeguard important information from corruption and/or loss. The methodology presented in D3.1.2 on how to automatically deploy services, has been applied allowing municipalities to re-deploy their services in another CSP if needed.

As identified during the financial excercise on the cost of owning the SCP platform, a simplified platform architecture was produced removing High Availability (Database and File Clustering) from the design and thus resulting in updated deployment scripts for all applications.

Meanwhile previously cloudified applications were launched in other municipalities including municipalities that participated in the 2nd "Call for Cities"

The document is the last of a four issues series aimed describing the steps needed to adapt services to the cloud paradigm and the steps needed for replicating these services in the cloud infrastructure of the pilot cities. What we have presented in this issue is procedures and supporting tools to enable the creation of a generic cloud-based services portfolio.

References

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

[2] D3.1.1 "Deployment of the services in the Cloud Infrastructure for Cloudification customization and testing", Alkiviadis Giannakoulias, June 2015

[3] D3.1.2 "Deployment of the services in the Cloud Infrastructure for Cloudification customization and testing", Alkiviadis Giannakoulias, October 2015

[4] D4.3 "Privacy and security measures", Alkiviadis Giannakoulias, August 2015

[5] D2.2.2 "Storm Clouds Platform Architectural Design", M. Consonni, A. Milani, February 2015

[6] https://www.ctl.io/blog/post/load-balancing-high-availability-and-disaster-recovery-what-they-are/

Annex A Services Deployment

A.1 URENIO

A.1.1 "Improve My City" Application

A.1.1.1 Installation and configuration commands list

After installing the VM and configuring the network access (see D3.1.1) we used Putty and issued the following list of commands to install and configure the application on the cloud infrastructure:

$ sudo su

$ vi /etc/hostname
and replace with the name of the application, i.e. ImproveMyCity

$vi /etc/hosts
Here update line 1 with 127.0.0.1 ImproveMyCity

$ vi /etc/resolv.conf
Here update to allow traffic to pass through DNS server with nameserver 10.15.5.22
$ reboot

$ apt-get update

Install all necessary packages:
$ apt-get install –y apache2
$ apt-get -y install debconf-utils
$ apt-get -y install mysql-client-core-5.6 mysql-client-5.6
$ debconf-set-selections <<< 'mysql-server-5.6 mysql-server/root_password password '$DBPASS
$ debconf-set-selections <<< 'mysql-server-5.6 mysql-server/root_password_again password '$DBPASS
$ apt-get -y install mysql-server-5.6

$ apt-get -y install unzip

Install composer
$ curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin --filename=composer
$ export COMPOSER_HOME="/usr/local/bin/composer"

Install automation script using Composer:
$ sudo composer global require joomlatools/joomla-console

Tell our system where to find the executable of joomla-console by adding the composer directory to your PATH.
export PATH="$PATH:~/.composer/vendor/bin"

Verify the installation
$ joomla –version

Create the new improve my city (imc) site with the latest available Joomla version
$ joomla site:create imc --www=/var/www/html --mysql-login=root:root --mysql-host=localhost --mysql-database=imcdb
Options
--www=<value>
Web server root
Default: /var/www

--mysql-login=<value>

MySQL credentials in the form of user:password *Default: root:root* *--mysql-host=<value>* *MySQL host* *Default: localhost* *--mysql-database=<value>* *MySQL database name.*
Install extra languages (greek) *$ wget -O /tmp/greek.zip http://joomlacode.org/gf/download/frsrelease/18750/162390/el-GR_joomla_lang_full_3.4.2v1.zip && joomla extension:installfile --www=/var/www/html imc /tmp/greek.zip*
Get and install the latest improve my city (imc) extension *$ wget -O /tmp/com_imc.zip https://github.com/itsam/imc/archive/master.zip && joomla extension:installfile --www=/var/www/html imc /tmp/com_imc.zip*
Get and install the latest imc theme template *$ wget -O /tmp/tpl_imc.zip https://github.com/icos-urenio/ImproveMyCity-Template/archive/master.zip && joomla extension:installfile --www=/var/www/html imc /tmp/tpl_imc.zip*
Get and install T3 plugin by Joomlart *$ wget -O /tmp/plg_T3.zip http://improve-my-city.com/download/plg_system_t3.v2.6.1.zip && joomla extension:installfile --www=/var/www imc /tmp/plg_T3.zip*
Get and install sample database *$ wget -O /tmp/imc_sample_data.sql https://raw.githubusercontent.com/icos-urenio/sampledata_imc/master/imc_sample_data.sql && joomla database:install --www=/var/www/html --mysql-login=root:root --mysql-host=localhost --mysql-database=imcdb --sql-dumps /tmp/cb_sample_data.sql -e imc*
Get the sample media (images and panorama) *$ #wget -O /tmp/imc_sample_images.zip https://github.com/icos-urenio/sampledata_imc/raw/master/imc_images.zip && sudo unzip /tmp/imc_sample_images.zip -d /var/www/html/imc/images*
Update .htaccess *$ echo "<IfModule mod_env.c>* *SetEnv HTTPS on* *</IfModule>" >> /var/www/html/imc/configuration.php*
Disable the deafult site, standard site is fine *$ a2dissite 1-__Appl_Name__.conf* *$ service apache2 restart*
Change the ssh configuration to enable password authentication *$ sed -i 's/.*PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config* **Restart ssh service** *$ restart ssh* *$ sed -i 's/.*error_reporting.*/error_reporting = E_ALL \& ~E_WARNING \& ~E_DEPRECATED | E_STRICT/g' /etc/php5/apache2/php.ini*
Give appropriate permissions in the directory *$ chown -R www-data.www-data /var/www/html/imc* *$ chmod -R 755 /var/www/html/imc*

Table A-1: Improve My City Application Deployment

A.1.1.2 Backup Setup

A.1.1.2.1 MySQL Database Backup

Using mysqldump and duplicity we can perform a full back using the following list of commands (backup_full.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

if [ ! -d /home/ubuntu/backup ]; then
   mkdir -p /home/ubuntu/backup
fi

if [ ! -d /home/ubuntu/backup/db ]; then
   mkdir -p /home/ubuntu/backup/db
fi

if [ ! -d /home/ubuntu/backup/db/sql ]; then
   mkdir -p /home/ubuntu/backup/db/sql
fi
cd /home/ubuntu/backup

$(which mysqldump) -uroot -proot imcdb --add-drop-database --lock-tables > /home/ubuntu/backup/db/sql/imcsql
$(which duplicity) full --encrypt-sign-key=DE6877E3 /home/ubuntu/backup/db/sql/ swift://imc_db
$(which duplicity) remove-all-but-n-full 3 --force swift://imc_db
```

To perform the incremental backup we issue the following list of commands (backup_incremental.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

if [ ! -d /home/ubuntu/backup ]; then
   mkdir -p /home/ubuntu/backup
fi

if [ ! -d /home/ubuntu/backup/db ]; then
   mkdir -p /home/ubuntu/backup/db
fi

if [ ! -d /home/ubuntu/backup/db/sql ]; then
   mkdir -p /home/ubuntu/backup/db/sql
fi
cd /home/ubuntu/backup

$(which mysqldump) -uroot -pmysql imcdb --add-drop-database --lock-tables > /home/ubuntu/backup/db/sql/imc.sql
$(which duplicity) --encrypt-sign-key= DE6877E3 /home/ubuntu/backup/db/sql/ swift://imc_db
```

To program these two jobs so that we have a full back every Sunday at 04:00 am and incremental ones daily at 04:00am we edit crontab:

$ crontab –e

And on the editor add:

00 04 * * 0 /home/ubuntu/backup/backup_full.sh >> /home/ubuntu/backup/backup.log

00 04 * * 1-6 /home/ubuntu/backup/backup_incremental.sh >> /home/ubuntu/backup/backup.log

To verify that cron jobs were updated we issue the same list of commands as the one presented in A.1.1.2.2 of D3.1.2

A.1.1.2.2 Application Folders Backup

The following the list of commands for backing up application specific folders using duplicity.

```
$ sudo mkdir /home/ubuntu/backup
$ cd /home/ubuntu/backup
$ source /home/ubuntu/backup/SCP_duplicity.rc
$(which duplicity) --encrypt-sign-key=DE6877E3 /var/www/html/imc/images / swift://imc_files
```

A.2 Municipio de Águeda

A.2.1 "Location Plans" Application

A.2.1.1 Installation and configuration commands list

After installing the VM and configuring the network access (see D3.1.1) we used Putty and issued the following list of commands to install and configure the application on the cloud infrastructure:

$ vi /etc/hostname
and replace with the name of the application, i.e. LocationPlans

$vi /etc/hosts
Here update line 1 with 127.0.0.1 LocationPlans

$ vi /etc/resolv.conf
Here update to allow traffic to pass through DNS server with nameserver 10.15.5.22
$ reboot

$ sudo apt-get update
$ sudo apt-get -y upgrade

Install all necessary packages:
$ sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install language-pack-en
$ sudo DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install redis-server build-essential subversion graphicsmagick imagemagick htop nmap unzip openssh-server
$ sudo apt-get -y install libgeos-dev gdal-bin
$ sudo apt-get -y install git

Install PostgreSQL database server
Before installing PostgreSQL we should install at least one language package. PostgreSQL will not create the initial database cluster without any language installed. Install this package or others for different languages.
$ sudo locale-gen "en_US.UTF-8"
$ sudo apt-get install language-pack-en

$ echo "deb http://apt.postgresql.org/pub/repos/apt/ trusty-pgdg main" | sudo tee /etc/apt/sources.list.d/pgdg.list
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install postgresql-9.5 postgresql-9.5-postgis-2.2 postgresql-contrib postgresql-client-9.5

Create new database

$ sudo su postgres

$ psql postgres -c "CREATE ROLE geobox LOGIN PASSWORD 'geobox' SUPERUSER INHERIT CREATEDB CREATEROLE REPLICATION;"

$ createdb -O geobox geopublic

$ psql geopublic -c "CREATE EXTENSION adminpack;"

$ psql geopublic -c "CREATE EXTENSION postgis;"

$ psql geopublic -c "CREATE EXTENSION pgcrypto;"

$ exit

"**geobox**" is the password that will be used latter so make a note on it.

Install Node.js

$ curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash –

$ sudo DEBIAN_FRONTEND=noninteractive apt-get -y install nodejs

$ sudo npm install -g forever

$ sudo npm install -g forever-service

Install JAVA, Tomcat and MapFish:

$ sudo apt-get install -y python-software-properties debconf-utils

$ sudo add-apt-repository -y ppa:webupd8team/java

$ sudo apt-get update

$ echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 select true" | sudo debconf-set-selections

$ sudo apt-get install -y oracle-java8-installer

Install JAVA JAI

$ cd /tmp

$ wget http://data.opengeo.org/suite/jai/jai-1_1_3-lib-linux-amd64-jdk.bin

$ wget http://data.opengeo.org/suite/jai/jai_imageio-1_1-lib-linux-amd64-jdk.bin

$ sed s/+215/-n+215/ jai_imageio-1_1-lib-linux-amd64-jdk.bin > jai_imageio-1_1-lib-linux-amd64-jdk_fixed.bin

$ sudo cp jai-1_1_3-lib-linux-amd64-jdk.bin /usr/lib/jvm/java-8-oracle

$ sudo cp jai_imageio-1_1-lib-linux-amd64-jdk_fixed.bin /usr/lib/jvm/java-8-oracle

$ cd /usr/lib/jvm/java-8-oracle

$ sudo yes | sh jai-1_1_3-lib-linux-amd64-jdk.bin

$ sudo yes | sh jai_imageio-1_1-lib-linux-amd64-jdk_fixed.bin

Install Tomcat

$ sudo apt-get -y install tomcat7 tomcat7-admin tomcat7-common

$ sudo sed -i '/#JAVA_HOME=/c\JAVA_HOME="/usr/lib/jvm/java-8-oracle"' /etc/default/tomcat7

$ sudo sed -i '/JAVA_OPTS="-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC"=/c\JAVA_OPTS="-Djava.awt.headless=true -Xmx1536m -XX:+UseConcMarkSweepGC"' /etc/default/tomcat7

Set user/password:

$ sudo sed -i '/<\/tomcat-users>/c\ <role rolename="tomcat"\/>\n <user username="tomcat" password="locationplans2k16" roles="manager,manager-gui,manager-script"\/>\n<\/tomcat-users>' /etc/tomcat7/tomcat-users.xml

$ sudo service tomcat7 restart

-- **Wait until tomcat7 is ready,** with:

-- tail -f /var/log/tomcat7/catalina.out
-- Wait until the server reports:
-- INFO: Server startup in xxxxx ms

Install Geoserver

$ mkdir -p ~/geoserver/data
$ sudo chown tomcat7:tomcat7 ~/geoserver/data
$ cd /tmp
$ wget http://ncu.dl.sourceforge.net/project/geoserver/GeoServer/2.9.0/geoserver-2.9.0-war.zip
$ unzip geoserver-2.9.0-war.zip
$ sudo cp geoserver.war /var/lib/tomcat7/webapps/
-- **Wait until geoserver app is deployed,** with:
-- tail -f /var/log/tomcat7/catalina.out
-- Wait until the server reports:
-- INFO: Server startup in xxxxx ms

$ sudo service tomcat7 stop
$ sudo cp -a /var/lib/tomcat7/webapps/geoserver/data ~/geoserver
$ sudo perl -i -p0e 's#<!--
<context-param>
<param-name>GEOSERVER_DATA_DIR.?-->#*
<context-param>
<param-name>GEOSERVER_DATA_DIR</param-name>
<param-value>/home/ubuntu/geoserver/data</param-value>
</context-param>#s' /var/lib/tomcat7/webapps/geoserver/WEB-INF/web.xml

$ sudo service tomcat7 start
-- **Wait until tomcat7 is ready,** with:
-- tail -f /var/log/tomcat7/catalina.out
-- Wait until the server reports:
-- INFO: Server startup in xxxxx ms

Install MapFish

$ cd /tmp
$ wget http://repo1.maven.org/maven2/org/mapfish/print/print-servlet/3.5.0/print-servlet-3.5.0.war
$ sudo cp print-servlet-3.5.0.war /var/lib/tomcat7/webapps/print.war

-- Wait until mapfish app is deployed, with:
-- tail -f /var/log/tomcat7/catalina.out
-- Wait until the server reports:
-- INFO: Server startup in xxxxx ms

Deploy the application from GitHub

$ cd
$ git clone https://github.com/jgrocha/LocationPlans.git
$ cd LocationPlans

$ sudo su postgres
$ cd /home/ubuntu/LocationPlans
 # create database tables


```
$psql dashboard -f dashboard.sql

# populate supporting tables
$ psql dashboard -f dashboard-data.sql

# initial user;
$ psql -c "insert into users.utilizador (idgrupo, email, password, nome, emailconfirmacao) values (1, 'alkiviadis.giannakoulias@eurodyn.com', encode(digest('pa55word', 'sha1'), 'hex'), 'Administrator', true);"

$ mkdir -p ~/public_html/public
$ cd ~/LocationPlans
$ cp -rf server/* ~/public_html
$ cp -rf build/production/Admin/* ~/public_html/public

$ sudo cp -rf print-apps/plantas /var/lib/tomcat7/webapps/print/print-apps
$ sudo chown -R tomcat7:tomcat7 /var/lib/tomcat7/webapps/print/print-apps/plantas

$ cd ~/public_html
$ npm update

$ cp server-config-TEMPLATE.json server-config.json$

# port where the server run. Port 80 maybe taken by Apache
$ sed -i 's/"port": [0-9]\+/"port": 80/' server-config.json
# full address
$ sed -i 's/localhost/web.sig.cm-agueda.pt/' server-config.json
# production DB
$ sed -i 's/"dbproduction": "[^"]\+"/"dbproduction": "postgres:\/\/geobox:geobox@localhost\/dashboard"/' server-config.json

$ sudo forever-service install -e "NODE_ENV=production" dashboard --script server.js

$ sudo start dashboard
```

Table A-2: Location Plans Application Deployment

A.2.1.2 Backup Setup

A.2.1.2.1 PostgreSQL Database Backup

Using *pg_dump* and duplicity we can perform a full back using the following list of commands (backup_full.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

# Database
BACKUP_DIRECTORY=/home/ubuntu/backup
log=$BACKUP_DIRECTORY/$data.log


if [ ! -d BACKUP_DIRECTORY ]; then
   mkdir -p BACKUP_DIRECTORY
fi
```

```
cd $BACKUP_DIRECTORY
rm -rf $BACKUP_DIRECTORY/*

bds="geopublic"

if [ ! -a /home/ubuntu/.pgpass ]; then
   echo 'localhost:5432:*:geobox:geobox' >> /home/ubuntu/.pgpass
   chmod 600 /home/ubuntu/.pgpass
fi

# db passwords should be in /backup/.pgpass
for bd in $bds
do
   backupfile=$BACKUP_DIRECTORY/$bd.dmp
   echo Processing postgres backup. File to generate : $backupfile
   echo Initializing export.

   pg_dump -h localhost -p 5432 -U geobox --no-password -F c -O -v -f $backupfile $bd >> $log 2>&1
done

$(which duplicity) full --encrypt-sign-key=10FAFE7E $BACKUP_DIRECTORY swift://LocationPlans_db
$(which duplicity) remove-all-but-n-full 3 --force swift://LocationPlans_db

echo Finished!
```

To perform the incremental backup we issue the following list of commands (backup_incremental.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

# Database
BACKUP_DIRECTORY=/home/ubuntu/backup
log=$BACKUP_DIRECTORY/$data.log


if [ ! -d BACKUP_DIRECTORY ]; then
   mkdir -p BACKUP_DIRECTORY
fi

cd $BACKUP_DIRECTORY
rm -rf $BACKUP_DIRECTORY/*

bds="geopublic"

if [ ! -a /home/ubuntu/.pgpass ]; then
   echo 'localhost:5432:*:geobox:geobox' >> /home/ubuntu/.pgpass
   chmod 600 /home/ubuntu/.pgpass
fi

# db passwords should be in /backup/.pgpass
for bd in $bds
do
   backupfile=$BACKUP_DIRECTORY/$bd.dmp
   echo Processing postgres backup. File to generate : $backupfile
   echo Initializing export.

   pg_dump -h localhost -p 5432 -U geobox --no-password -F c -O -v -f $backupfile $bd >> $log 2>&1
done
```



```
$(which duplicity) --encrypt-sign-key=10FAFE7E $BACKUP_DIRECTORY swift://LocationPlans_db
```

To program these two jobs so that we have a full back every Sunday at 04:00 am and incremental ones daily at 04:00am we edit crontab:

$ crontab –e

And on the editor add:

00 04 * * 0 /home/ubuntu/backup/backup_full.sh >> /home/ubuntu/backup/backup.log

00 04 * * 1-6 /home/ubuntu/backup/backup_incremental.sh >> /home/ubuntu/backup/backup.log

A.2.1.2.2 Application Folders Backup

The following the list of commands for backing up application specific folders using duplicity.

```
$ sudo mkdir /home/ubuntu/backup
$ cd /home/ubuntu/backup
$ source /home/ubuntu/backup/SCP_duplicity.rc
$(which duplicity) --encrypt-sign-key=DE6877E3 /home/ubuntu/public_html/public/ swift://LocationPlans_files
```

A.3 City Of Miskolc

A.3.1 "OPENDATA" Application

A.3.1.1 Installation and configuration commands list

After installing the VM and configuring the network access (see D3.1.1) we used Putty and issued the following list of commands to install and configure the application on the cloud infrastructure:

$ sudo su

$ vi /etc/hostname
and replace with the name of the application, i.e. ImproveMyCity

$vi /etc/hosts
Here update line 1 with 127.0.0.1 ImproveMyCity

$ vi /etc/resolv.conf
Here update to allow traffic to pass through DNS server with nameserver 10.15.5.22
$ reboot

$ apt-get update

Install all necessary packages:
$ apt-get install –y apache2
$ apt-get -y install debconf-utils
$ apt-get -y install mysql-client-core-5.6 mysql-client-5.6
$ debconf-set-selections <<< 'mysql-server-5.6 mysql-server/root_password password '$DBPASS
$ debconf-set-selections <<< 'mysql-server-5.6 mysql-server/root_password_again password '$DBPASS
$ apt-get -y install mysql-server-5.6

$ apt-get -y install unzip

Install PostgreSQL database server
Before installing PostgreSQL we should install at least one language package. PostgreSQL will not create the initial database cluster without any language installed. Install this package or others for different

languages. *$ sudo locale-gen "en_US.UTF-8"* *$ sudo apt-get install language-pack-en* *$ echo "deb http://apt.postgresql.org/pub/repos/apt/ trusty-pgdg main" | sudo tee /etc/apt/sources.list.d/pgdg.list* *$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -* *$ sudo apt-get update* *$ sudo apt-get install postgresql-9.5 postgresql-9.5-postgis-2.2 postgresql-contrib postgresql-client-9.5*
Update .htaccess *$ echo "<IfModule mod_env.c>* *SetEnv HTTPS on* *</IfModule>" >> /var/www/html/imc/configuration.php*
Disable the deafult site, standard site is fine *$ a2dissite 1-__Appl_Name__.conf* *$ service apache2 restart*
Change the ssh configuration to enable password authentication *$ sed -i 's/.*PasswordAuthentication.*/PasswordAuthentication yes/g' /etc/ssh/sshd_config* **Restart ssh service** *$ restart ssh* *$ sed -i 's/.*error_reporting.*/error_reporting = E_ALL \& ~E_WARNING \& ~E_DEPRECATED | E_STRICT/g' /etc/php5/apache2/php.ini*
Give appropriate permissions in the directory *$ chown -R www-data.www-data /var/www/html/imc* *$ chmod -R 755 /var/www/html/imc*

Table A-3: Improve My City Application Deployment

A.3.1.2 Backup Setup

A.3.1.2.1 MySQL Database Backup

Using mysqldump and duplicity we can perform a full back using the following list of commands (backup_full.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

if [ ! -d /home/ubuntu/backup ]; then
   mkdir -p /home/ubuntu/backup
fi

if [ ! -d /home/ubuntu/backup/db ]; then
   mkdir -p /home/ubuntu/backup/db
fi

if [ ! -d /home/ubuntu/backup/db/sql ]; then
   mkdir -p /home/ubuntu/backup/db/sql
fi
cd /home/ubuntu/backup

$(which mysqldump) -uroot -proot imcdb --add-drop-database --lock-tables > /home/ubuntu/backup/db/sql/OpenData_MySQL_db
```



```
$(which duplicity) full --encrypt-sign-key=DE6877E3 /home/ubuntu/backup/db/sql/ swift://OpenData_MySQL_db
$(which duplicity) remove-all-but-n-full 3 --force swift://OpenData_MySQL_db
```

To perform the incremental backup we issue the following list of commands (backup_incremental.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

if [ ! -d /home/ubuntu/backup ]; then
   mkdir -p /home/ubuntu/backup
fi

if [ ! -d /home/ubuntu/backup/db ]; then
   mkdir -p /home/ubuntu/backup/db
fi

if [ ! -d /home/ubuntu/backup/db/sql ]; then
   mkdir -p /home/ubuntu/backup/db/sql
fi
cd /home/ubuntu/backup

$(which mysqldump) -uroot -pmysql opendatadb --add-drop-database --lock-tables > /home/ubuntu/backup/db/sql/OpenData_MySQL_db.sql
$(which duplicity) --encrypt-sign-key= DE6877E3 /home/ubuntu/backup/db/sql/ swift://OpenData_MySQL_db
```

To program these two jobs so that we have a full back every Sunday at 04:00 am and incremental ones daily at 04:00am we edit crontab:

$ crontab –e

And on the editor add:

00 04 * * 0 /home/ubuntu/backup/backup_full.sh >> /home/ubuntu/backup/backup.log

00 04 * * 1-6 /home/ubuntu/backup/backup_incremental.sh >> /home/ubuntu/backup/backup.log

To verify that cron jobs were updated we issue the same list of commands as the one presented in A.1.1.2.2 of D3.1.2

A.3.1.2.2 PostgreSQL Database Backup

Using *pg_dump* and duplicity we can perform a full back using the following list of commands (backup_full.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

# Database
BACKUP_DIRECTORY=/home/ubuntu/backup
log=$BACKUP_DIRECTORY/$data.log


if [ ! -d BACKUP_DIRECTORY ]; then
   mkdir -p BACKUP_DIRECTORY
fi

cd $BACKUP_DIRECTORY
rm -rf $BACKUP_DIRECTORY/*

bds="geopublic"
```

```
if [ ! -a /home/ubuntu/.pgpass ]; then
   echo 'localhost:5432:*:geobox:geobox' >> /home/ubuntu/.pgpass
   chmod 600 /home/ubuntu/.pgpass
fi

# db passwords should be in /backup/.pgpass
for bd in $bds
do
   backupfile=$BACKUP_DIRECTORY/$bd.dmp
   echo Processing postgres backup. File to generate : $backupfile
   echo Initializing export.

   pg_dump -h localhost -p 5432 -U geobox --no-password -F c -O -v -f $backupfile $bd >> $log 2>&1
done

$(which duplicity) full --encrypt-sign-key=10FAFE7E $BACKUP_DIRECTORY
swift://OpenData_PostgreSQL_db
$(which duplicity) remove-all-but-n-full 3 --force swift://OpenData_PostgreSQL_db

echo Finished!
```

To perform the incremental backup we issue the following list of commands (backup_incremental.sh).

```
#!/bin/bash
source /home/ubuntu/backup/SCP_duplicity.rc

# Database
BACKUP_DIRECTORY=/home/ubuntu/backup
log=$BACKUP_DIRECTORY/$data.log


if [ ! -d BACKUP_DIRECTORY ]; then
   mkdir -p BACKUP_DIRECTORY
fi

cd $BACKUP_DIRECTORY
rm -rf $BACKUP_DIRECTORY/*

bds="geopublic"

if [ ! -a /home/ubuntu/.pgpass ]; then
   echo 'localhost:5432:*:geobox:geobox' >> /home/ubuntu/.pgpass
   chmod 600 /home/ubuntu/.pgpass
fi

# db passwords should be in /backup/.pgpass
for bd in $bds
do
   backupfile=$BACKUP_DIRECTORY/$bd.dmp
   echo Processing postgres backup. File to generate : $backupfile
   echo Initializing export.

   pg_dump -h localhost -p 5432 -U geobox --no-password -F c -O -v -f $backupfile $bd >> $log 2>&1
done

$(which duplicity) --encrypt-sign-key=10FAFE7E $BACKUP_DIRECTORY swift://OpenData_PostgreSQL_db
```


To program these two jobs so that we have a full back every Sunday at 04:00 am and incremental ones daily at 04:00am we edit crontab:

$ crontab –e

And on the editor add:

00 04 * * 0 /home/ubuntu/backup/backup_full.sh >> /home/ubuntu/backup/backup.log

00 04 * * 1-6 /home/ubuntu/backup/backup_incremental.sh >> /home/ubuntu/backup/backup.log

Annex B Services Validation

B.1 URENIO – "Improve My City" Application

Accessing the main page is possible through the private IP at http://smartcity.veria.gr/improvemycity/

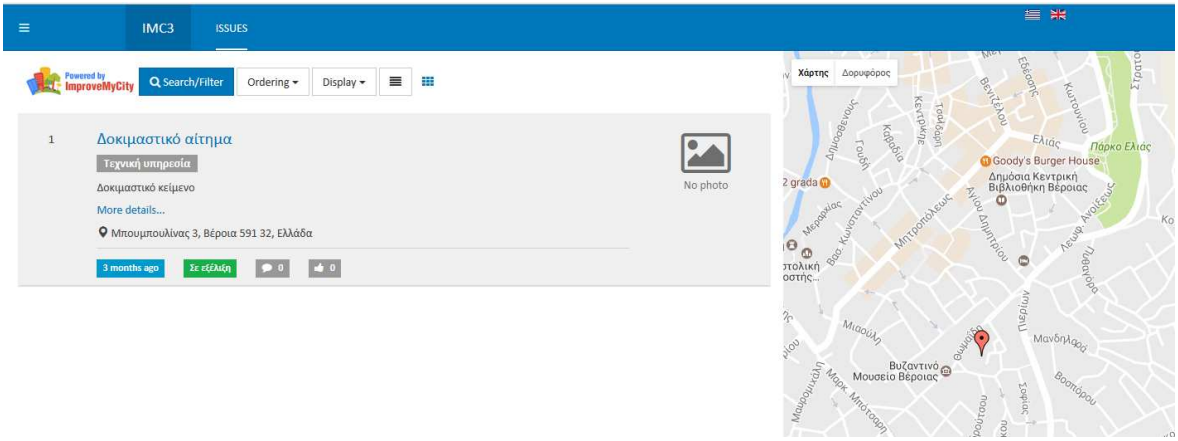

Figure B-1: URENIO – Improve My City

B.2 Municipio de Águeda – "Location Plans" Application

Accessing the main page is possible through the private IP at http://178.239.184.243/#dashboard

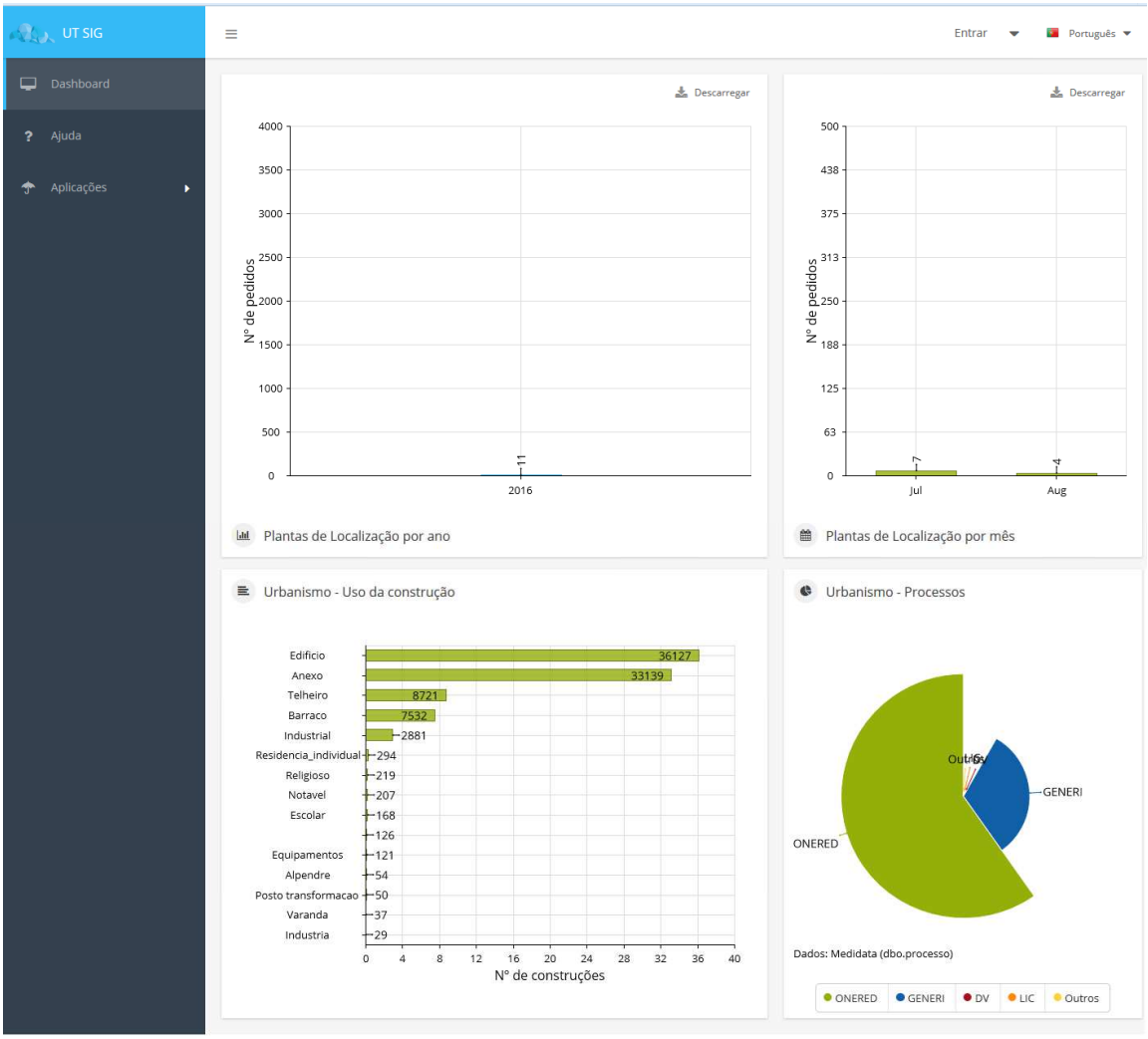

Figure B-2: Municipio de Águeda – Location Plans (Dashboard)

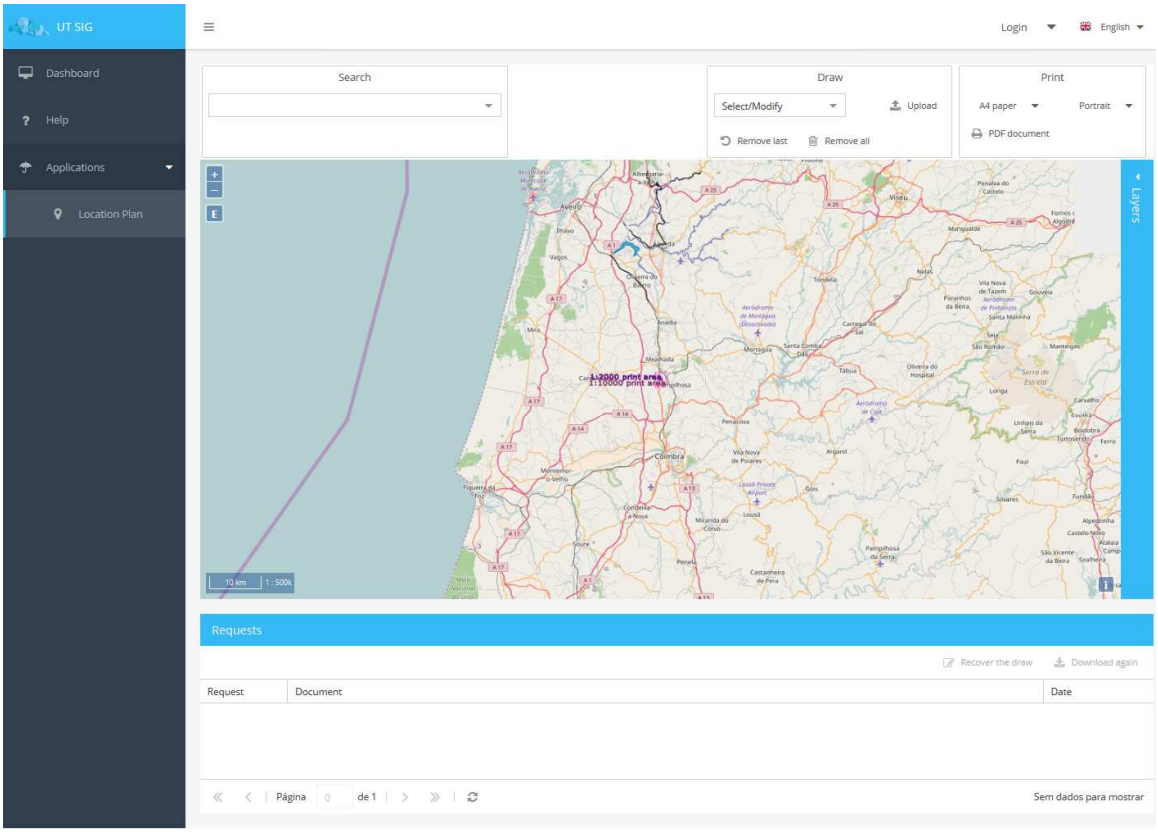

Figure B-3: Municipio de Águeda – Location Plans

B.3 City of Miskolc – "OpenData" Application

Accessing the main page is possible through the private IP at http://80.247.66.119/

Figure B-4: City of Miskolc – OpenData

B.4 Municipality of Veria

Accessing the main page is possible through the private IP at http://smartcity.veria.gr/

B.4.1 "City Branding" Applications

Accessing the main page is possible through the private IP at http://smartcity.veria.gr/citybranding/en/

B.4.2 "Cloud Funding" Application

Accessing the main page is possible through the private IP at http://smartcity.veria.gr/crowd-funding

B.4.3 "Improve My City" Application

Accessing the main page is possible through the private IP at http://smartcity.veria.gr/improvemycity/

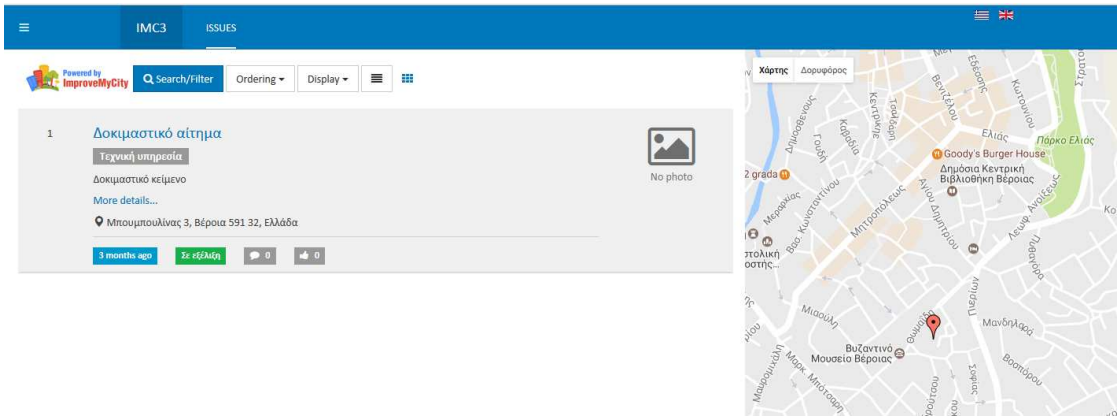

Figure B-5: Municipality of Veria – Improve My City

B.4.4 "Virtual City Market" Application

Accessing the main page is possible through the private IP at http://smartcity.veria.gr/virtual-city-market

Annex C Services Detailed Security Report

C.1 URENIO

C.1.1 Compliance According to Categories: A Detailed Report (Improve My City)

C.1.1.1 (A1) Injection

Total number of alerts in this category: 4

Alerts in this category

SQL Injection	
Risk:	High
Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization. **Reference:** https://www.owasp.org/index.php/SQL_Injection https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet https://www.owasp.org/index.php/Top_10_2013-A1-Injection	

C.1.1.2 (A2) Broken Authentication and Session Management

Total number of alerts in this category: 53

Alerts in this category

Cookie set without HttpOnly flag	
Risk:	Low
A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. **Solution:** Ensure that the HttpOnly flag is set for all cookies. **Reference:** https://www.owasp.org/index.php/HttpOnly	
Password Autocomplete in browser	
Risk:	Low
AUTOCOMPLETE attribute is not disabled in HTML FORM/INPUT element containing password type input. Passwords may be stored in browsers and retrieved. **Solution:** Turn off AUTOCOMPLETE attribute in form or individual input elements containing password by using AUTOCOMPLETE='OFF'. **Reference:** https://www.owasp.org/index.php/Testing_for_Vulnerable_Remember_Password_(OTG-AUTHN-005)	

C.1.1.3 (A3) Cross-Site Scripting (XSS)

Total number of alerts in this category: 70

Alerts in this category

Cross Site Scripting (Reflected)	
Risk:	High
Cross site scripting (also referred to as XSS) is a vulnerability that allows an attacker to send malicious code (usually in the form of JavaScript) to another user. Because a browser cannot know if the script should be trusted or not, it will execute the script in the user context allowing the attacker to access any cookies or session tokens retained by the browser. **Solution:** Use a vetted library or framework, such as Microsoft's Anti-XSS library, the **OWASP ESAPI Encoding module**, and Apache Wicket, that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For every page generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. **Reference:** https://www.owasp.org/index.php/Testing_for_Reflected_Cross_site_scripting_(OTG-INPVAL-001) https://www.owasp.org/index.php/Cross-site_Scripting_(XSS) https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet **NOTE:** Municipality has confirmed that appropriate source code modifications were successfully performed and the risk was removed.	
Web Browser XSS Protection Not Enabled	
Risk:	Low
Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server. **Solution:** Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. **Reference:** https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet	
Cross-Domain JavaScript Source File Inclusion	
Risk:	Low
Pages include one or more script files from a third-party domain. **Solution:** Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.	

C.1.1.4 (A4) Insecure Direct Object References

No alerts in this category.

C.1.1.5 (A5) Security Misconfiguration

Total number of alerts in this category: 48

Alerts in this category

X-Content-Type-Options Header Missing	
Risk:	Low
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. **Solution:** Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. **Reference:** https://www.owasp.org/index.php/List_of_useful_HTTP_headers https://msdn.microsoft.com/library/gg622941(v=vs.85).aspx	

C.1.1.6 (A6) Sensitive Data Exposure

Total number of alerts in this category: 26

Alerts in this category

Private IP Disclosure	
Risk:	Low
Private IP 10.15.5.226 has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems. **Solution:** Remove the private IP address from the HTTP response body. For comments, use JSP/ASP comment instead of HTML/JavaScript comment which can be seen by client browsers.	

C.1.1.7 (A7) Missing Function Level Access Control

No alerts in this category.

C.1.1.8 (A8) Cross-Site Request Forgery (CSRF)

No alerts in this category.

C.1.1.9 (A9) Using Components with Known Vulnerabilities

No alerts in this category.

C.1.1.10 (A10) Unvalidated Redirects and Forward

Total number of alerts in this category: 48

Alerts in this category

X-Frame-Options Header Not Set	
Risk:	Medium
X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. **Solution:** Ensure that the X-Frame-Options HTTP header is set on all web pages returned by the application. Since we don't expect the page to be framed by other servers, we should use DENY. **Reference:** https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet	

https://www.owasp.org/index.php/Clickjacking
https://www.owasp.org/index.php/List_of_useful_HTTP_headers
https://developer.mozilla.org/en-US/docs/Web/HTTP/X-Frame-Options

NOTE: Municipality has confirmed that site configuration was successfully updated by adding: *Header always append X-Frame-Options SAMEORIGIN* and the risk was removed.

C.2 Municipio de Águeda

C.2.1 Compliance According to Categories: A Detailed Report (Location Plans)

C.2.1.1 (A1) Injection

No alerts in this category.

C.2.1.2 (A2) Broken Authentication and Session Management

No alerts in this category.

C.2.1.3 (A3) Cross-Site Scripting (XSS)

Total number of alerts in this category: 3

Web Browser XSS Protection Not Enabled	
Risk:	Low
Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server. **Solution:** Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'. **Reference:** https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet	

C.2.1.4 (A4) Insecure Direct Object References

No alerts in this category.

C.2.1.5 (A5) Security Misconfiguration

Total number of alerts in this category: 1

Alerts in this category

X-Content-Type-Options Header Missing	
Risk:	Low
The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. **Solution:** Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. **Reference:** https://www.owasp.org/index.php/List_of_useful_HTTP_headers https://msdn.microsoft.com/library/gg622941(v=vs.85).aspx	

C.2.1.6 (A6) Sensitive Data Exposure

No alerts in this category.

C.2.1.7 (A7) Missing Function Level Access Control

No alerts in this category.

C.2.1.8 (A8) Cross-Site Request Forgery (CSRF)

No alerts in this category.

C.2.1.9 (A9) Using Components with Known Vulnerabilities

No alerts in this category.

C.2.1.10 (A10) Unvalidated Redirects and Forward

Total number of alerts in this category: 3

Alerts in this category

X-Frame-Options Header Not Set	
Risk:	Medium
X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. **Solution:** Ensure that the X-Frame-Options HTTP header is set on all web pages returned by the application. Since we don't expect the page to be framed by other servers, we should use DENY. **Reference:** https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet https://www.owasp.org/index.php/Clickjacking https://www.owasp.org/index.php/List_of_useful_HTTP_headers https://developer.mozilla.org/en-US/docs/Web/HTTP/X-Frame-Options **NOTE:** Municipality has confirmed that site configuration was successfully updated by adding: *Header always append X-Frame-Options SAMEORIGIN* and the risk was removed.	

C.3 City of Miskolc

C.3.1 Compliance According to Categories: A Detailed Report (OPENDATA)

C.3.1.1 (A1) Injection

No alerts in this category.

C.3.1.2 (A2) Broken Authentication and Session Management

No alerts in this category.

C.3.1.3 (A3) Cross-Site Scripting (XSS)

Total number of alerts in this category: 360

Alerts in this category

Web Browser XSS Protection Not Enabled	
Risk:	Low
Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server. **Solution:** Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP	

response header to '1'.

Reference: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

C.3.1.4 (A4) Insecure Direct Object References

Total number of alerts in this category: 1

Alerts in this category

Directory listing	
Risk:	Medium
It is possible to view the directory listing. Directory listing may reveal hidden scripts, include files, backup source files etc. which be accessed to read sensitive information. **Solution:** Disable directory browsing. If this is required, make sure the listed files does not induce risks. **Reference:** https://www.owasp.org/index.php/Top_10_2013-A5-Security_Misconfiguration http://www.thesitewizard.com/apache/prevent-directory-listing-htaccess.shtml http://httpd.apache.org/docs/current/mod/core.html#options https://wiki.apache.org/httpd/DirectoryListings http://www.linuxscrew.com/2008/06/03/faq-how-to-disable-directory-browsing-in-apachehttpd/ **NOTE:** Municipality has confirmed that relevant modification to the Apache configuration was successfully performed and the risk was removed.	

C.3.1.5 (A5) Security Misconfiguration

No alerts in this category.

C.3.1.6 (A6) Sensitive Data Exposure

Total number of alerts in this category: 261

Alerts in this category

Private IP Disclosure	
Risk:	Low
Private IP 10.15.5.226 has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems. **Solution:** Remove the private IP address from the HTTP response body. For comments, use JSP/ASP comment instead of HTML/JavaScript comment which can be seen by client browsers.	

C.3.1.7 (A7) Missing Function Level Access Control

No alerts in this category.

C.3.1.8 (A8) Cross-Site Request Forgery (CSRF)

No alerts in this category.

C.3.1.9 (A9) Using Components with Known Vulnerabilities

No alerts in this category.

C.3.1.10 (A10) Unvalidated Redirects and Forward

Total number of alerts in this category: 35

Alerts in this category

X-Frame-Options Header Not Set	
Risk:	Medium
X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. **Solution:** Ensure that the X-Frame-Options HTTP header is set on all web pages returned by the application. Since we don't expect the page to be framed by other servers, we should use DENY. **Reference:** https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet https://www.owasp.org/index.php/Clickjacking https://www.owasp.org/index.php/List_of_useful_HTTP_headers https://developer.mozilla.org/en-US/docs/Web/HTTP/X-Frame-Options **NOTE:** Municipality has confirmed that site configuration was successfully updated by adding: *Header always append X-Frame-Options SAMEORIGIN* and the risk was removed.	

Annex D Services Automation (OpenStack Heat Templates SCP v3.0)

The following scripts take advantage of the new simplified SCP architecture shown bellow:

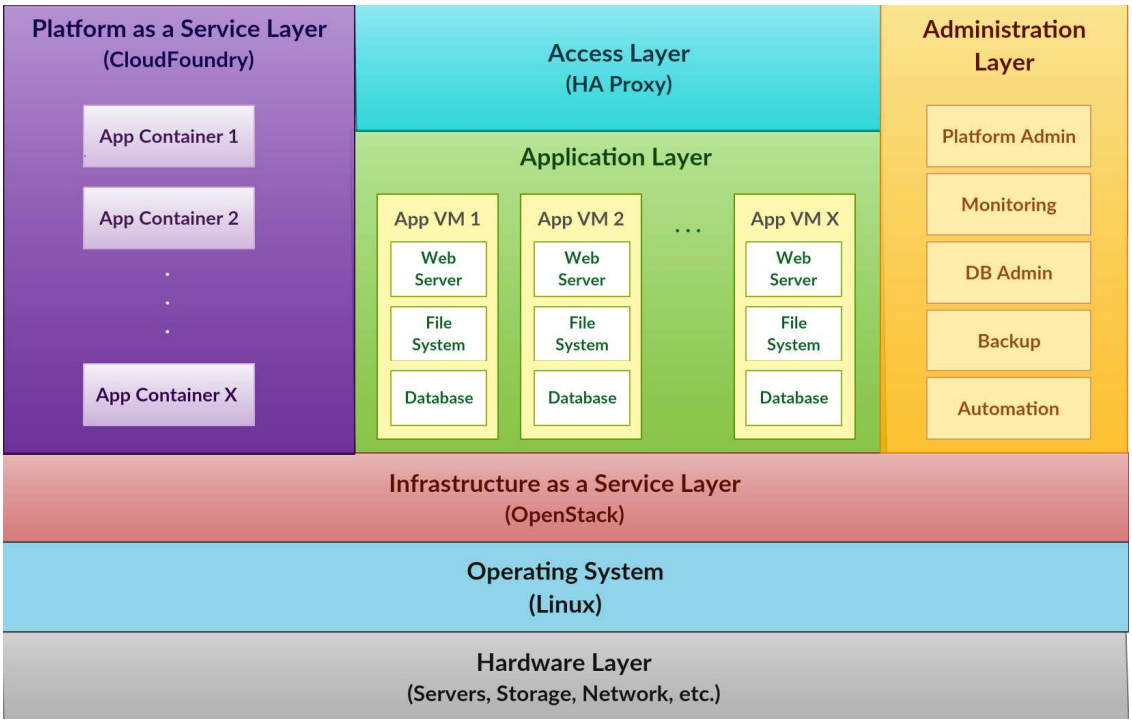

Figure C-6: SCP Simplified Architecture

The difference to the scripts presented in the previous deliverables is that both the application database and the file system are hosted on the same virtual machine where applications are installed. The financial exercise on the previous deployment model/architecture (SCP v2.0) showed that the cost of the 'shared' database and file system services were coming at a hogh cost (around 70% with 10 hosted applications and 50% with 20 hosted applications)

As a result a new approach on how applications are deployed was implemented as shown bellow:

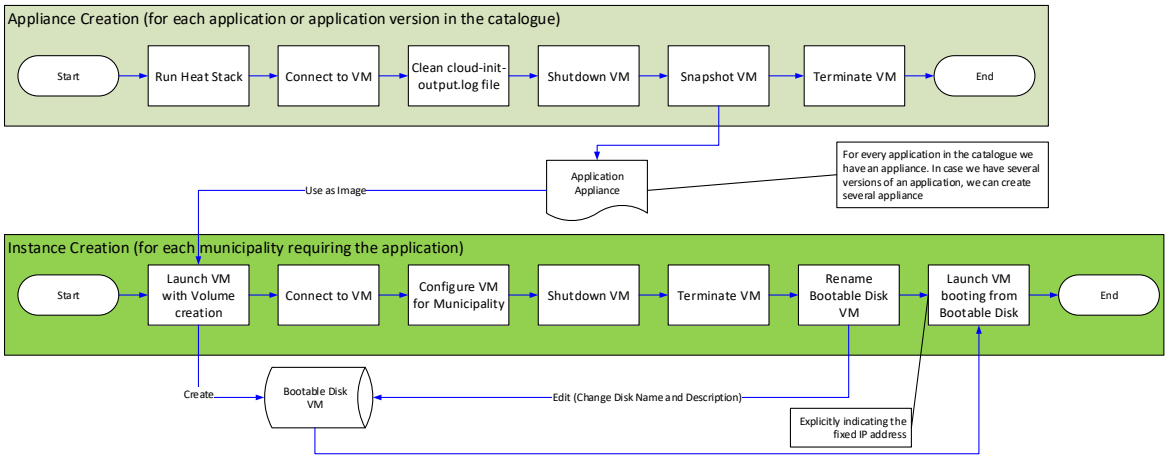

Figure C-7: SCP Simplified Architecture, Creating and Using an Appliance

Moreover, in order to re-use software as much as possible we have used a series of bash script files that are re-used across different applications. The following files are involved:

1. create-appliance.sh bash shell script, that is responsible for creating the virtual appliance (VM);

2. create-image.sh bash shell script, that is reposnible to create an image based on the above appliance;
3. create-volume.sh bash shell script, that is reponsible for creating a new bootable volume out of the above image;
4. lauch-service.sh, that is the main bash shell script for launching a service for a specific municipality;
5. apache_install.sh, that is responsible for installing Apache HTTP Server version 2.x;
6. console_apache-configure.sh, that is responsible for creating the /etc/apache2/sites-available/000-000-default-ssl.conf file and enabling it, installing the SSL certificates and configuring the SSL engine in order for the application to be accessible via secure HTTPS links
7. mysql-singlenode_setup.sh, that is responsible for setting up a MySQL server for the applications that require one;
8. postgresql-singlenode_setup.sh, that is responsible for setting up a PostgreSQL server for the applications that require one;
9. phpmyadmin_setup.sh, that is responsible for setting up a phpMyAdmin instance for managing a MySQL server and its contents;
10. phppgadmin_setup.sh, that is responsible for setting up a phpPgAdmin instance for managing a PostgreSQL server and its contents;
11. console_schedule-backup.sh, that is responsible for setting up the scheduled cron jobs for executing the backup strategy;
12. postfix_install.sh and postfix_setup.sh, that are responsible for installing and configuring postfix (Mail Transfer Agent (MTA)) for applications that require an email server and
13. haproxy.yaml and haproxy.sh, that are responsible for setting up a HAProxy for municipalities that require one.

```
#!/bin/bash -e

echoerr() { echo "$@" 1>&2; }

if [ "$#" -ne 1 ]
then
  echoerr "Usage: $(basename $0) <application>"
  exit 1
fi

export APPNAME=$1
export APPLIANCE=$APPNAME-appliance
export NETWORK_ID=`neutron net-list | grep SCP-Network | awk '{ print $2 }'`
export SUBNET_ID=`neutron subnet-list | grep SCP-Subnet | awk '{ print $2 }'`

if [ -z "$NETWORK_ID" ]
then
  echoerr "Error: network 'SCP-Network' not defined"
  exit 1
fi

if [ -z "$SUBNET_ID" ]
then
  echoerr "Error: subnet 'SCP-Subnet' not defined"
  exit 1
fi

heat stack-create -f heat/$APPNAME.yaml \
```

```
                -P Network_Id=$NETWORK_ID          \
                -P Subnet_Id=$SUBNET_ID            \
                -P OS_USERNAME=$OS_USERNAME        \
                -P OS_PASSWORD=$OS_PASSWORD        \
                -P OS_TENANT_NAME=$OS_TENANT_NAME  \
                -P OS_TENANT_ID="                  \
                -P OS_REGION_NAME=$OS_REGION_NAME  \
                -P OS_AUTH_URL=$OS_AUTH_URL        \
                -P Console_Fixed_IP='10.20.0.98'   \
                -P Server=$APPLIANCE               \
                $APPLIANCE
exit $?
```

Table C-4: create-appliance.sh

```
#!/bin/bash -e

echoerr() { echo "$@" 1>&2; }

if [ "$#" -ne 1 ]
then
  echoerr "Usage: $(basename $0) <appliance>"
  exit 1
fi

export APPNAME=$1
export APPLIANCE=$APPNAME-appliance

nova image-create --show --poll $APPLIANCE $APPNAME

exit $?
```

Table C-5: create-image.sh

```
#!/bin/bash -e

echoerr() { echo "$@" 1>&2; }

if [ "$#" -ne 2 ]
then
  echoerr "Usage: $(basename $0) <municipality> <image>"
  exit 1
fi

export MUNICIPALITY=$1
export IMAGENAME=$2
export IMAGEID=`nova image-list | grep $IMAGENAME | awk '{ print $2 }'`
if [ -z "$IMAGEID" ]
then
  echoerr "Image $IMAGEID not found"
  exit 1
fi
export VOLNAME=$MUNICIPALITY-$IMAGENAME

cinder create --image-id     $IMAGEID \
              --display-name $VOLNAME \
              25
```



```
exit $?
```

Table C-6: create-volume.sh

```
#!/bin/bash -e

echoerr() { echo "$@" 1>&2; }

if [[ "$#" -lt 3 ||  "$#" -gt 4 ]]
then
  echoerr "Usage: $(basename $0) <municipality> <application> <IP address> {flavor}"
  exit 1
fi

export MUNICIPALITY=$1
export APPLICATION=$2
export IPADDRESS=$3
export FLAVOR=${4:-e2standard.x2}
export VOLNAME=$MUNICIPALITY-$APPLICATION

export VOLID=`cinder list | grep $VOLNAME | awk '{ print $2 }'`
if [ -z "$VOLID" ]
then
  echoerr "Boot volume $VOLNAME not found"
  exit 1
fi

if [ -z "$VOLID" ]
then
  echoerr "Boot volume $VOLNAME not found"
  exit 1
fi

export NETWORK_ID=`neutron net-list | grep SCP-Network | awk '{ print $2 }'`
export SUBNET_ID=`neutron subnet-list | grep SCP-Subnet | awk '{ print $2 }'`

if [ -z "$NETWORK_ID" ]
then
  echoerr "Error: network 'SCP-Network' not defined"
  exit 1
fi

if [ -z "$SUBNET_ID" ]
then
  echoerr "Error: subnet 'SCP-Subnet' not defined"
  exit 1
fi

nova boot --flavor      $FLAVOR  \
        --boot-volume $VOLID   \
        --key-name    $MUNICIPALITY \
        --nic net-id=$NETWORK_ID,v4-fixed-ip=$IPADDRESS \
        --security-groups SSH_Security_Group,Zabbix_Security_Group,WebFEE_Security_Group \
        $VOLNAME

exit $?
```

Table C-7: lauch-service.sh

```
#!/bin/bash -ex
sudo su

# Install Apache HTTP Server version 2.x
apt-get -y install apache2
apt-get -y install php5 libapache2-mod-php5 php5-mcrypt php5-common php5-mysql php5-cli php5-gd
a2enmod headers
a2enmod ssl
a2enmod rewrite
```

Table C-8: apache_install.sh

```
#!/bin/bash -ex
sudo su
#create the html for the index page
cd /var/www/html
cat << EOF > index.html
<html>
        <head>
                <link rel="stylesheet" type="text/css"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/css/bootstrap.min.css" />
        </head>
        <body class="text-center">
                <h1>STORM CLOUDS Platform Console</h1>
                <div class="main">
                        <ul class="list-unstyled">
                                <li><a href="zabbix">Monitoring</a></li>
                                <li><a href="phpmyadmin">MySQL Administration</a></li>
                                <li><a href="phppgadmin">PostgreSQL Administration</a></li>
                                <li><a href="__OpenStack_URL__">OpenStack Administration</a></li>
                        </ul>
                </div>
        </body>
</html>
EOF

#generate key
mkdir /etc/apache2/ssl
openssl req -x509 -nodes -newkey rsa:2048 -out /etc/apache2/ssl/server.crt -keyout
/etc/apache2/ssl/server.key \
   -subj "/C=/ST=/L=/O=/OU=/CN=__Floating_IP__/emailAddress=/"

#install SSL support on Apache server
a2enmod ssl

ln -s /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-enabled/000-default-ssl.conf

#import key into Apache
sed -i 's/SSLCertificateFile.*/SSLCertificateFile \/etc\/apache2\/ssl\/server.crt/g' /etc/apache2/sites-
enabled/000-default-ssl.conf
sed -i 's/SSLCertificateKeyFile.*/SSLCertificateKeyFile \/etc\/apache2\/ssl\/server.key/g'
/etc/apache2/sites-enabled/000-default-ssl.conf

#Force https redirect
sed -i '/DocumentRoot/ a Redirect \/ https://__Floating_IP__\/' /etc/apache2/sites-enabled/000-
```



```
default.conf
sed -i '/DocumentRoot/ a Redirect \/ https://__Floating_IP__\/' /etc/apache2/sites-available/000-default.conf

service apache2 restart
```

Table C-9: console_apache-configure.sh

```
#!/bin/bash -ex
sudo su
apt-get -y update
debconf-set-selections <<< 'mysql-server-5.6 mysql-server/root_password password __DB_Admin_Pass__'
debconf-set-selections <<< 'mysql-server-5.6 mysql-server/root_password_again password __DB_Admin_Pass__'
apt-get -y install mysql-server
sed -i -E '/bind-address/s/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/0.0.0.0/' /etc/mysql/my.cnf
mysql -uroot -p__DB_Admin_Pass__ << EOF
GRANT ALL PRIVILEGES ON *.* TO '__DB_Admin_User__'@'%' IDENTIFIED BY '__DB_Admin_Pass__';
GRANT GRANT OPTION ON *.* TO '__DB_Admin_User__'@'%';
FLUSH PRIVILEGES;
COMMIT;
EXIT
EOF
service mysql restart
```

Table C-10: mysql-singlenode_setup.sh

```
#!/bin/bash -ex
sudo su
apt-get install -y postgresql-9.3
apt-get install -y postgresql-9.3-postgis-2.1
apt-get install -y postgresql-contrib-9.3
sudo -u postgres psql -c "CREATE USER __DB_Admin_User__ WITH PASSWORD '__DB_Admin_Pass__' SUPERUSER;"
sed -i -E 's/(#listen_addresses.*)/listen_addresses = \x27*\x27\n\1/' /etc/postgresql/9.3/main/postgresql.conf
echo "host all all * md5" >> /etc/postgresql/9.3/main/pg_hba.conf
echo "local all all md5" >> /etc/postgresql/9.3/main/pg_hba.conf
service postgresql restart
```

Table C-11: postgresql-singlenode_setup.sh

```
#!/bin/bash -ex
sudo su
debconf-set-selections <<< "phpmyadmin phpmyadmin/internal/skip-preseed boolean true"
debconf-set-selections <<< "phpmyadmin phpmyadmin/reconfigure-webserver multiselect apache2"
debconf-set-selections <<< "phpmyadmin phpmyadmin/dbconfig-install boolean false"
apt-get -y install phpmyadmin
while ! mysql -u__DB_Admin_User__ -p__DB_Admin_Pass__ -h __Virtual_IP__ -e "quit" ;
do
  echo "Waiting for remote DB __Virtual_IP__..."
  sleep 3
done
mysql -u__DB_Admin_User__ -p__DB_Admin_Pass__ -h __Virtual_IP__ << EOF
DROP DATABASE IF EXISTS phpmyadmin;
CREATE DATABASE phpmyadmin;
USE phpmyadmin;
\. /usr/share/dbconfig-common/data/phpmyadmin/install/mysql
```

```
COMMIT;
quit
EOF
cp /etc/phpmyadmin/config.inc.php /etc/phpmyadmin/config.inc.php.save
cp /usr/share/doc/phpmyadmin/examples/config.sample.inc.php /etc/phpmyadmin/config.inc.php
sed -i 's/localhost/__Virtual_IP__/' /etc/phpmyadmin/config.inc.php
```

Table C-12: phpmyadmin_setup.sh

```
#!/bin/bash -ex
sudo su
apt-get -y install phppgadmin
cp /etc/apache2/conf.d/phppgadmin /etc/apache2/conf-available/phppgadmin.conf
sed -i -E 's/(\s*deny .*)/#COMMENT OUT\n#\1/' /etc/apache2/conf-available/phppgadmin.conf
sed -i -E 's/(\s*allow .*)/#COMMENT OUT\n#\1\nallow from all/' /etc/apache2/conf-available/phppgadmin.conf
sed -i -E 's/localhost/__Virtual_IP__/' /etc/phppgadmin/config.inc.php
a2enconf phppgadmin
service apache2 reload
```

Table C-13: phppgadmin_setup.sh

```
#!/bin/bash -ex
sudo su
echo "00 03 * * 0 /root/SCP/SCP/tools/admin/SCP-backup_full.sh >> /root/SCP/SCP/tools/admin/backup.log" >> /var/spool/cron/crontabs/root
echo "00 03 * * 1-6 /root/SCP/SCP/tools/admin/SCP-backup_incremental.sh >> /root/SCP/SCP/tools/admin/backup.log" >> /var/spool/cron/crontabs/root
chown root:crontab /var/spool/cron/crontabs/root
```

Table C-14: console_schedule-backup.sh

```
#!/bin/bash -ex
sudo su
debconf-set-selections <<< "postfix postfix/mailname string __Mail_Name__"
debconf-set-selections <<< "postfix postfix/main_mailer_type string 'Internet Site'"
apt-get install -y postfix
apt-get install -y mailutils
```

Table C-15: postfix_install.sh

```
#!/bin/bash -ex
sudo su
debconf-set-selections <<< "postfix postfix/mailname string __Mail_Name__"
debconf-set-selections <<< "postfix postfix/main_mailer_type string 'Internet Site'"
apt-get install -y postfix
apt-get install -y mailutils
mv /etc/postfix/main.cf /etc/postfix/main.cf.bck
cat << EOF >> /etc/postfix/main.cf
relayhost=__Mail_Relay_Host__
smtp_sasl_auth_enable=yes
smtp_sasl_password_maps=hash:/etc/postfix/sasl_password
smtp_sasl_security_options=noanonymous
smtp_tls_CAfile=/etc/ssl/certs/ca-certificates.crt
smtp_use_tls=yes
EOF
cat << EOF >> /etc/postfix/sasl_password
__Mail_Relay_Host__ __Mail_Relay_Account__:__Mail_Relay_Password__
```



```
EOF
postmap hash:/etc/postfix/sasl_password
chown root:root /etc/postfix/sasl_password
chmod 600 /etc/postfix/sasl_password
cat /etc/ssl/certs/Thawte_Premium_Server_CA.pem | tee -a /etc/postfix/cacert.pem
/etc/init.d/postfix reload
HOSTNAME=`hostname`
HOSTIP=`ifconfig eth0 | sed -n '/inet addr/s/.*addr.\([^ ]*\) .*/\1/p'`
echo "Posfix installed on $HOSTNAME IP address: $HOSTIP" | mail -s "STORM CLOUDS Platform: Postifix Installed" __Mail_Relay_Account__
postfix flush
```

Table C-16: postfix_setup.sh

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy HA Proxy on an Ubuntu instance using Heat's
  software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.

  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
  Image:
    type: string
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'
  Mail_Name:
```

```
    type: string
    default: ''

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  postfix_install:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/postfix_install.sh }
          params:
            __Mail_Name__: {get_param: Mail_Name}

  haproxy:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/haproxy-1.0.sh }
          params:
            __NOPARAM__: "NOPARAM"

  config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: postfix_install}
      - config: {get_resource: haproxy}

  node_port:
    type: OS::Neutron::Port
    properties:
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
      security_groups: [
        SSH_Security_Group,
        Zabbix_Security_Group,
```



```
        WebFEE_Security_Group,
        HAProxy_Security_Group
      ]

  instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [instance, first_address] }
```

Table C-17: haproxy.yaml

```
#!/bin/bash -ex
sudo su

# Install Apache
apt-get install -y apache2
a2enmod ssl
a2ensite default-ssl
a2enmod proxy_http
a2enmod rewrite

# Install HA Proxy
add-apt-repository -y ppa:vbernat/haproxy-1.6
apt-get update     -y
apt-get install    -y haproxy

export SERVER='haproxy'
export IPADDR=`ifconfig eth0 | grep "inet " | awk -F'[:]' '{ print $2 }'  | awk '{ print $1}'`
export CRTDIR='/etc/apache2/ssl'
export CRTFILE="$CRTDIR/storm.crt"
export KEYFILE="$CRTDIR/storm.key"
export PEMFILE="$CRTDIR/storm.pem"

mkdir $CRTDIR
openssl req -x509 -nodes -newkey rsa:2048 -out $CRTFILE -keyout $KEYFILE \
   -subj "/C=/ST=/L=/O=/OU=/CN=$SERVER/emailAddress=/"

cat $CRTFILE $KEYFILE > $PEMFILE

cat << EOF > /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
```

```
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        RewriteEngine on

        # redirect everything to https, the (.*) has a leading /
        RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R,L]

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
EOF


cat << EOF > /etc/apache2/sites-available/default-ssl.conf
# virtual host for SSL
<IfModule mod_ssl.c>

#NameVirtualHost *:443
<VirtualHost *:443>
        ServerAdmin stormjanitor@gmail.com
        ServerName  $SERVER
        # add ServerAlias as needed

        SSLEngine on
        SSLProxyEngine on
        RewriteEngine On

        # SSL cert files
        # Configure the following 3 lines with your own credentials
        SSLCertificateFile    $CRTFILE
        SSLCertificateKeyFile $KEYFILE
        ### SSLCertificateChainFile /etc/ssl/certs/<YourOwnBundle>.ca-bundle

        # Redirect requests not ending in slash eg. /app1 to /app1/
        ### RewriteRule ^/([^/]+)\$ https://%{HTTP_HOST}/\$1 [R,L]
        # Uncomment this (end enable mod_disk_cache) to enable caching
        # CacheEnable disk /
```



```
        # The above RewriteRule is equivalent to multiple ProxyPass rules, eg
        #  ProxyPass /app1/ http://127.0.0.1:8080/app1/
        # etc.

        # THIS NEEDS A LINE FOR EACH APPLICATION
        ProxyPass /testproxy/       http://127.0.0.1:8080/testproxy/
        ProxyPass /testproxy        http://127.0.0.1:8080/testproxy/

         # add other apps here...

        <Proxy http://127.0.0.1:8080/*>
          Allow from all
        </Proxy>

        # HAProxy Admin access configuration
        <Proxy http://127.0.0.1:8000/*>
          Allow from all
        </Proxy>
        ProxyPassReverse /admin/ http://127.0.0.1:8000/

        ErrorLog \${APACHE_LOG_DIR}/error_ssl.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" **%T/%D**"
combined
        CustomLog \${APACHE_LOG_DIR}/access_ssl.log combined
</VirtualHost>
</IfModule>
EOF

cat << EOF > /etc/haproxy/haproxy.cfg
global
        ##log 127.0.0.1 local0
        ##log 127.0.0.1 local1 notice
        #log loghost    local0 info
        maxconn 4096
        user haproxy
        group haproxy
        daemon
        node lb1
        spread-checks 5     # 5%
        tune.ssl.default-dh-param 4096
# uncomment this to get debug output
        #debug
        #quiet

# This section is fixed and just sets some default values.
# These values can be overridden by more-specific redefinitions
# later in the config
defaults
        log     global
        mode    http
        # option  httplog
        option  dontlognull
```

```
        retries 3
        option redispatch
        maxconn 2000
        timeout connect    5000
        timeout client     50000
        timeout server     50000

# Enable admin/stats interface
# go to http://lb1.example.com:8000/stats to access it
listen stats
      bind 0.0.0.0:8000 ssl crt $PEMFILE
      mode http
      stats enable
      stats hide-version
      stats realm Haproxy\ Statistics
      stats uri /
      stats auth stormjanitor:stormjanitor

# A single frontend section is needed. This listens on 127.0.0.1:8080, and
# receives the requests from Apache.
frontend web
  bind 127.0.0.1:8080
  mode http

  # This determines which application is being requested
  # These ACL will match if the path in the request contains the relevant application name
  # for example the first ACL (want_app1) will match if the request is for /app1/something/, etc.
  acl want_testproxy path_dir testproxy
  # ... add lines for other applications here...

  # these ACLs match if at least one server
  # for the application is available.
  acl testproxy_avail nbsrv(testproxy) ge 1
  # ... add lines for other applications here...

  # Here is where HAProxy decides which backend to use. Conditions
  # are ANDed.
  # This says: use the backend called "app1" if the request
  # contains /app1/ (want_app1) AND the backend is available (app1_avail), etc.
  use_backend testproxy if want_testproxy testproxy_avail

  # ... etc

  # If we get here, no backend is available for the requested
  # application and users will get an error

########### BACKENDS ####################

backend testproxy
  mode http
  option httpclose
  balance roundrobin
  cookie SRVID insert indirect nocache
  option nolinger
  option httpchk GET / HTTP/1.0\r\nUser-Agent:\ HAProxy

  rspirep ^(Set-Cookie:.*\ path=)([^\ ]+)(.*)\$       \1/testproxy\2\3
  reqirep ^Host:.*\$   Host:\ $SERVER
```



```
  ### Loopback: not working, yet
  reqrep ^([^\ ]*\ /)testproxy[/]?(.*)     \1\2
  ### Loopback: not working, yet
  server testproxy_1 $IPADDR:80 cookie testproxy check inter 10s rise 2 fall 2

# these are the error pages returned by HAProxy when an error occurs
# customize as needed

errorfile       400     /etc/haproxy/errors/400.http
errorfile       403     /etc/haproxy/errors/403.http
errorfile       408     /etc/haproxy/errors/408.http
errorfile       500     /etc/haproxy/errors/500.http
errorfile       502     /etc/haproxy/errors/502.http
errorfile       503     /etc/haproxy/errors/503.http
errorfile       504     /etc/haproxy/errors/504.http

EOF

service apache2 restart
service haproxy start
```

Table C-18: haproxy.sh

D.1 Municipality of Thessaloniki

D.1.1 City Branding

This section describes a heat stack for deploying "City Branding" in SCP v3.0. The following files are involved in the stack creation:

1. citybranding.yaml, that is responsible for creating the stack containing the VM;
2. citybranding.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy City Brabding on an Ubuntu instance using Heat's
  software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.
  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
```

```
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
    description: the name of the server
  Image:
    type: string
    description: the name of the boot image
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    description: Flavor to use for the WordPress server.
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  DB_User:
    type: string
    description: name of the DB User
    default: 'stormjanitor'
    hidden: true
  DB_Password:
    type: string
    description: name of the DB Password
    default: 'password'
    hidden: true

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  mysql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/mysql-singlenode_setup.sh}
```



```
        params:
          __DB_Admin_User__: {get_param: DB_User}
          __DB_Admin_Pass__: {get_param: DB_Password}

apache_install:
  type: OS::Heat::SoftwareConfig
  properties:
    group: script
    config:
      str_replace:
        template: {get_file: fragments/apache_install.sh}
        params:
          __NOPARAM__: "NOPARAM"

citybranding_setup:
  type: OS::Heat::SoftwareConfig
  properties:
    group: script
    config:
      str_replace:
        template: { get_file: fragments/citybranding-1.0.sh }
        params:
          __DB_User__:        {get_param: DB_User}
          __DB_Password__:    {get_param: DB_Password}

CityBranding_config:
  type: OS::Heat::MultipartMime
  properties:
    parts:
    - config: {get_resource: scplight-node_setup}
    - config: {get_resource: mysql-singlenode_setup}
    - config: {get_resource: apache_install}
    - config: {get_resource: citybranding_setup}


CityBranding_node_port:
  type: OS::Neutron::Port
  properties:
    network_id: {get_param: Network_Id}
    fixed_ips:
    - subnet_id: {get_param: Subnet_Id}
    security_groups: [
      SSH_Security_Group,
      Zabbix_Security_Group,
      WebFEE_Security_Group
    ]

CityBranding_instance:
  type: OS::Nova::Server
  properties:
    name:       {get_param: Server}
    image:      {get_param: Image}
    flavor:     {get_param: Flavor}
    key_name:   {get_param: Key}
    networks:
      - port: {get_resource: CityBranding_node_port}
    admin_user: ubuntu
    user_data_format: RAW
```

```
    user_data: {get_resource: CityBranding_config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [CityBranding_instance, first_address] }
```

Table C-19: citybranding.yaml

```
#!/bin/bash -ex
sudo su

export APPNAME='citybranding'
export DBHOST='localhost'
export DBNAME=$APPNAME
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export APPDIR="/var/www/html/$APPDIR"

# install composer
curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin --filename=composer
export COMPOSER_HOME="/usr/local/bin/composer"

# Install automation script using Composer:
sudo composer global require joomlatools/joomla-console

# Tell your system where to find the executable of joomla-console by adding the composer directory to your PATH.
# Add the following line to your shell configuration file called either .profile, .bash_profile, .bash_aliases, or .bashrc.
# This file is located in your home folder.
export PATH="$PATH:~/.composer/vendor/bin"

# Verify the installation
joomla --version

# Create the new citybranding site with the latest available Joomla version
joomla site:create $APPNAME --www=/var/www/html --mysql-login=$DBUser:$DBPASS --mysql-host=$DBHOST --mysql-database=$DBNAME

# install extra languages (greek)
wget -O /tmp/greek.zip http://joomlacode.org/gf/download/frsrelease/18750/162390/el-GR_joomla_lang_full_3.4.2v1.zip && joomla extension:installfile --www=/var/www/html $APPNAME /tmp/greek.zip

# get and install the latest citybranding extension
wget -O /tmp/com_citybranding.zip https://github.com/icos-urenio/citybranding/archive/master.zip && joomla extension:installfile --www=/var/www/html $APPNAME /tmp/com_citybranding.zip

# get and install the latest citybranding theme template
wget -O /tmp/tpl_city.zip https://github.com/icos-urenio/tpl_city/archive/master.zip && joomla extension:installfile --www=/var/www/html $APPNAME /tmp/tpl_city.zip

# get and install sample database
wget -O /tmp/cb_sample_data.sql https://raw.githubusercontent.com/icos-urenio/sampledata_citybranding/master/cb_sample_data.sql && joomla database:install --www=/var/www/html --mysql-login=$DBUSER:$DBPASS --mysql-host=$DBHOST --mysql-database=$DBNAME --sql-dumps /tmp/cb_sample_data.sql -e $APPNAME
```



```
# get the sample media (images and panorama)
# make sure you set the correct path to the images /var/www/html/citybranding/images
wget -O /tmp/cb_sample_images.zip https://github.com/icos-urenio/sampledata_citybranding/raw/master/sample_images.zip && sudo unzip /tmp/cb_sample_images.zip -d /var/www/html/$APPNAME/images

chown -R www-data.www-data /var/www/html/$APPNAME
chmod -R 755 /var/www/html/$APPNAME
sed -i 's/.*error_reporting.*/error_reporting = E_ALL \& ~E_WARNING \& ~E_DEPRECATED | E_STRICT/g' /etc/php5/apache2/php.ini
sed -i 's/public $debug = '\''1'\'';/public $debug = ''0'';/g' /var/www/html/$APPNAME/configuration.php
sed -i 's/public $MetaDesc = '\''Joomla! - the dynamic portal engine and content management system'\'';/public $MetaDesc = '''''''''';/g' /var/www/html/$APPNAME/configuration.php
sed -i 's/public $MetaKeys = '\''joomla, Joomla'\'';/public $MetaKeys = '''''''''';/g' /var/www/html/$APPNAME/configuration.php

export SITECONF=1-$APPNAME.conf
cat << EOF > /etc/apache2/sites-available/$SITECONF
<VirtualHost *:80>
   ServerAdmin webmaster@citybranding.dev
   ServerName citybranding.dev
   ServerAlias www.citybranding.dev

   DocumentRoot /var/www/html
   <Directory /var/www/html/citybranding>
     Options FollowSymLinks
     AllowOverride All
   </Directory>
   #RedirectMatch ^/$ /citybranding

   ErrorLog  /var/log/apache2/citybranding.dev-error_log
   CustomLog /var/log/apache2/citybranding.dev-access_log common
</VirtualHost>
EOF

a2ensite  $SITECONF
a2dissite 000-default.conf
service apache2 restart

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
/usr/bin/mysqldump -u$DBUSER -h$DBHOST -p$DBPASS --databases $DBNAME --add-drop-database --lock-tables > ~/backup/data/dbbackup.sql
tar -zcvf ~/backup/data/$APPNAME.tar.gz $APPDIR/images
EOF
chmod +x ~/backup/scripts/backup.sh

#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################

# Update .htaccess
#echo "<IfModule mod_env.c>
#SetEnv HTTPS on
#</IfModule>" >> /var/www/html/improvemycity/.htaccess

#export APPNAME=citybranding
```

```
#sed -i 's/public $sitename = '\"citybranding'\";/public $sitename = """__SMTP_From_Name__""";/g'
/var/www/html/$APPNAME/configuration.php
#sed -i 's/public $smtphost = '\"localhost'\";/public $smtphost = """__SMTP_Host__""";/g'
/var/www/html/$APPNAME/configuration.php
#sed -i 's/public $smtpuser = '\"'\";/public $smtpuser = """__SMTP_User__""";/g'
/var/www/html/$APPNAME/configuration.php
#sed -i 's/public $smtppass = '\"'\";/public $smtppass = """__SMTP_Password__""";/g'
/var/www/html/$APPNAME/configuration.php
#sed -i 's/public $mailfrom = '\"admin@example.com'\";/public $mailfrom = """__SMTP_From__""";/g'
/var/www/html/$APPNAME/configuration.php
#sed -i 's/public $fromname = '\"citybranding'\";/public $fromname = """__SMTP_From_Name__""";/g'
/var/www/html/$APPNAME/configuration.php

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname,"
/etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

# Finally uncomment RedirectMatch in /etc/apache2/sites-available/1-improvemycity.conf
#sudo service apache2 restart

##############################################
```

Table C-20: citybranding.sh

D.1.2 Cloud Funding

This section describes a heat stack for deploying "Cloud Funding" in SCP v3.0. The following files are involved in the stack creation:

1. cloudfunding.yaml, that is responsible for creating the stack containing the VM;
2. cloudfunding.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy CloudFunding on an Ubuntu instance using Heat's
  software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.

  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
```



```
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
  Image:
    type: string
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  Git_User:
    type: string
    description: name of the Git User for connecting to private account
    hidden: true
    default: ayian2004
  Git_Password:
    type: string
    description: Git Password
    hidden: true
    default: 9Wy%GraFZ9rqfonM

  DB_User:
    type: string
    default: 'stormjanitor'
  DB_Password:
    type: string
    default: 'password'

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  mysql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
```

```
  properties:
    group: script
    config:
      str_replace:
        template: {get_file: fragments/mysql-singlenode_setup.sh}
        params:
          __DB_Admin_User__: {get_param: DB_User}
          __DB_Admin_Pass__: {get_param: DB_Password}

apache_install:
  type: OS::Heat::SoftwareConfig
  properties:
    group: script
    config:
      str_replace:
        template: {get_file: fragments/apache_install.sh}
        params:
          __NOPARAM__: "NOPARAM"

cloudfunding_setup:
  type: OS::Heat::SoftwareConfig
  properties:
    group: script
    config:
      str_replace:
        template: { get_file: fragments/cloudfunding-1.0.sh }
        params:
          __DB_User__:        {get_param: DB_User}
          __DB_Password__:    {get_param: DB_Password}
          __Git_User__:       {get_param: Git_User}
          __Git_Password__:   {get_param: Git_Password}

CloudFunding_config:
  type: OS::Heat::MultipartMime
  properties:
    parts:
    - config: {get_resource: scplight-node_setup}
    - config: {get_resource: mysql-singlenode_setup}
    - config: {get_resource: apache_install}
    - config: {get_resource: cloudfunding_setup}

CloudFunding_node_port:
  type: OS::Neutron::Port
  properties:
    network_id: {get_param: Network_Id}
    fixed_ips:
    - subnet_id: {get_param: Subnet_Id}
    security_groups: [
      SSH_Security_Group,
      Zabbix_Security_Group,
      WebFEE_Security_Group
    ]

CloudFunding_instance:
  type: OS::Nova::Server
  properties:
    name:       {get_param: Server}
    image:      {get_param: Image}
```



```
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: CloudFunding_node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: CloudFunding_config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [CloudFunding_instance, first_address] }
```

Table C-21: cloudfunding.yaml

```
#!/bin/bash -ex
sudo su

export DBHOST='localhost'
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export DBNAME="crowdfunding"
export bckqqt=`echo -e '\x60'`
export DBCOLLATION=$bckqqt'utf8_general_ci'$bckqqt
export APPNAME='crowdfunding'
export APPDIR="/var/www/html/crowd-funding"

# Git clone the application source files
cd /var/www/html
git clone https://__Git_User__:__Git_Password__@github.com/STORM-CLOUDS/Cloud-Funding.git

# rename folder
mv /var/www/html/Cloud-Funding $APPDIR

# Now give appropriate permissions in the directory
chown -R www-data.www-data $APPDIR
chmod -R 755 $APPDIR

# prepare  database
mysql -u$DBUSER -p$DBPASS -h$DBHOST << EOF
DROP DATABASE IF EXISTS $DBNAME;
CREATE DATABASE $DBNAME COLLATE $DBCOLLATION;
COMMIT;
QUIT
EOF
mysql -u$DBUSER -h$DBHOST -p$DBPASS $DBNAME < $APPDIR/db/cloud-funding.sql

# Edit local-settings.php and add your mysql connection and other details
# PATH
sed -i 's/define('\"PATH'\", '\"\/cloud-funding'\");/define('\"PATH'\", '\"\/crowd-funding'\");/g'
$APPDIR/local-settings.php
# Metadata
# Database
sed -i 's/define('\"GOTEO_DB_HOST'\", '\"localhost'\");/define('\"GOTEO_DB_HOST'\", ""'$DBHOST'"");/g'
$APPDIR/local-settings.php
sed -i 's/define('\"GOTEO_DB_SCHEMA'\", '\"cloud-funding'\");/define('\"GOTEO_DB_SCHEMA'\",
""'$DBNAME'"");/g' $APPDIR/local-settings.php
sed -i 's/define('\"GOTEO_DB_USERNAME'\", '\"root'\");/define('\"GOTEO_DB_USERNAME'\",
```

```
""$DBUSER"");/g'      $APPDIR/local-settings.php
sed       -i       's/define(\"GOTEO_DB_PASSWORD'\",       '\""\");/define(\"GOTEO_DB_PASSWORD'\",
""$DBPASS"");/g'          $APPDIR/local-settings.php
sed -i 's/Cloud-Funding/crowd-funding/g' $APPDIR/local-settings.php

# Edit .htaccess
sed -i 's/cloud-funding/crowd-funding/g' $APPDIR/.htaccess

sed -i 's/.*error_reporting.*/error_reporting = E_ALL \& ~E_WARNING \& ~E_DEPRECATED | E_STRICT/g'
/etc/php5/apache2/php.ini

cat << EOF > /etc/apache2/sites-available/cloudfunding.conf
<VirtualHost *:80>
      # The ServerName directive sets the request scheme, hostname and port that
      # the server uses to identify itself. This is used when creating
      # redirection URLs. In the context of virtual hosts, the ServerName
      # specifies what hostname must appear in the request's Host: header to
      # match this virtual host. For the default virtual host (this file) this
      # value is not decisive as it is used as a last resort host regardless.
      # However, you must set it for any further virtual host explicitly.
      #ServerName www.example.com

      ServerAdmin smartcity@storm.eu
      ServerName cloudfunding1.0

      ErrorLog ${APACHE_LOG_DIR}/cloudfunding-error.log
      CustomLog ${APACHE_LOG_DIR}/cloudfunding-access.log combined

      DocumentRoot /var/www/html
      <Directory /var/www/html/crowd-funding>
        Options FollowSymLinks
        AllowOverride All
      </Directory>
      #RedirectMatch ^/$ /crowd-funding
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
EOF
a2dissite 000-default.conf
a2ensite  cloudfunding.conf
service apache2 restart

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
/usr/bin/mysqldump -u$DBUSER -h$DBHOST -p$DBPASS --databases $DBNAME --add-drop-database --
lock-tables > ~/backup/data/dbbackup.sql
tar -zcvf ~/backup/data/$APPNAME.tar.gz $APPDIR/data
EOF
chmod +x ~/backup/scripts/backup.sh


#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################

# Updated local-settings.php
#export APPDIR='/var/www/html/crowd-funding/'
```



```
#sed -i 's/define('\"GOTEO_SITE_NAME'\", '\"CityName'\");/define('\"GOTEO_SITE_NAME'\",
""Municipality of __Municipality__"");/g' $APPDIR/local-settings.php

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname,"
/etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

# Finally uncomment RedirectMatch in /etc/apache2/sites-available/cloudfunding.conf
#sudo service apache2 restart

##############################################
##############################################
```

Table C-22: cloudfunding.sh

D.1.3 Virtual City Market

This section describes a heat stack for deploying "Virtual City Market" in SCP v3.0. The following files are involved in the stack creation:

1. virtualcitymarket.yamlimprovemycity.yaml, that is responsible for creating the stack containing the VM;
2. virtualcitymarket.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy Virtual City Market on an Ubuntu instance using Heat's
  software orchestration feature.

parameters:

  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
  Image:
    type: string
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
```

```
    type: string
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  Git_User:
    type: string
    description: name of the Git User for connecting to private account
    hidden: true
    default: ayian2004
  Git_Password:
    type: string
    description: Git Password
    hidden: true
    default: 9Wy%GraFZ9rqfonM

  DB_User:
    type: string
    default: 'stormjanitor'
  DB_Password:
    type: string
    default: 'password'

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  mysql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/mysql-singlenode_setup.sh}
          params:
            __DB_Admin_User__: {get_param: DB_User}
            __DB_Admin_Pass__: {get_param: DB_Password}

  apache_install:
    type: OS::Heat::SoftwareConfig
    properties:
```



```
      group: script
      config:
        str_replace:
          template: {get_file: fragments/apache_install.sh}
          params:
            __NOPARAM__: "NOPARAM"

  virtualcitymarket_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/virtualcitymarket-1.0.sh}
          params:
            __DB_User__:        {get_param: DB_User}
            __DB_Password__:    {get_param: DB_Password}
            __Git_User__:       {get_param: Git_User}
            __Git_Password__:   {get_param: Git_Password}

  vcm_config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: mysql-singlenode_setup}
      - config: {get_resource: apache_install}
      - config: {get_resource: virtualcitymarket_setup}

  vcm_node_port:
    type: OS::Neutron::Port
    properties:
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
      security_groups: [
        SSH_Security_Group,
        Zabbix_Security_Group,
        WebFEE_Security_Group
      ]

  vcm_instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: vcm_node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: vcm_config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [vcm_instance, first_address] }
```

Table C-23: virtualcitymarket.yaml

```
#!/bin/bash -ex
sudo su


export DBHOST='localhost'
export DBNAME="virtualcitymarket"
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export bckqqt=`echo -e '\x60'`
export DBCOLLATION=$bckqqt'utf8_general_ci'$bckqqt
export APPNAME=virtual-city-market
export APPDIR=/var/www/html/$APPNAME

cd /var/www/html
git clone https://__Git_User__:__Git_Password__@github.com/STORM-CLOUDS/Virtual-City-Market.git
mv Virtual-City-Market $APPDIR

# Change owner of cache directory to www-data
chown -R www-data $APPDIR/cache
chown -R www-data $APPDIR/uploads

mysql -u$DBUSER -p$DBPASS << EOF
  CREATE DATABASE $DBNAME COLLATE $DBCOLLATION;
  USE $DBNAME;
  \. $APPNAME/data/market.sql
EOF

# Debug off
sed -i 's/define('\"DEBUG'\", true);/define('\"DEBUG'\", false);/g' $APPDIR/config.inc.php
# Database settings
sed  -i  's/define('\"MARKET_DB_USER'\",  '\"root'\");/define('\"MARKET_DB_USER'\",  ""$DBUSER"");/g'
$APPDIR/config.inc.php
sed  -i  's/define('\"MARKET_DB_PASS'\",  '\"'\");/define('\"MARKET_DB_PASS'\",  ""$DBPASS"");/g'
$APPDIR/config.inc.php
sed  -i  's/define('\"MARKET_DB_DATABASE'\",  '\"virtual-city-
market'\");/define('\"MARKET_DB_DATABASE'\", ""$DBNAME"");/g' $APPDIR/config.inc.php

cat << EOF > /etc/apache2/sites-available/$APPNAME.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin storjanitor@storm.eu
        ServerName $APPNAME

        ErrorLog \${APACHE_LOG_DIR}/$APPNAME-error.log
        CustomLog \${APACHE_LOG_DIR}/$APPNAME-access.log combined

        DocumentRoot /var/www/html
```



```
        <Directory /var/www/html/$APPNAME>
          Options FollowSymLinks
          AllowOverride All
        </Directory>
        #RedirectMatch ^/$ /$APPNAME
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
EOF

sed -i 's/.*error_reporting.*/error_reporting = E_ALL \& ~E_WARNING \& ~E_DEPRECATED | E_STRICT/g'
/etc/php5/apache2/php.ini
a2dissite 000-default
a2ensite $APPNAME.conf
service apache2 restart

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
/usr/bin/mysqldump -u$DBUSER -h$DBHOST -p$DBPASS --databases $DBNAME --add-drop-database --
lock-tables > ~/backup/data/dbbackup.sql
tar -zcvf ~/backup/data/$APPNAME.tar.gz $APPDIR/uploads
EOF
chmod +x ~/backup/scripts/backup.sh


#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################

# Updated config-inc.php
# Update reCAPTCHA Keys
#sed       -i       's/define('\"RECAPTCHA_PUBLIC_KEY'\",       '\"'\");/define('\"RECAPTCHA_PUBLIC_KEY'\",
'"""__reCAPTCHA_Public_Key__"""');/g'    $APPDIR/config.inc.php
#sed      -i      's/define('\"RECAPTCHA_PRIVATE_KEY'\",      '\"'\");/define('\"RECAPTCHA_PRIVATE_KEY'\",
'"""__reCAPTCHA_Private_Key__"""');/g' $APPDIR/config.inc.php

# Update Google Maps API Key
#sed              -i              's/define('\"GMAP_API_KEY'\",              '\"'\");/define('\"GMAP_API_KEY'\",
'"""__GMAP_API_Key__"""');/g' $APPDIR/config.inc.php

# Update Google Maps Center longitude and latitude
#sed -i 's/define('\"GMAP_CENTER_LAT'\", '\"40.63018304279187'\");/define('\"GMAP_CENTER_LAT'\",
'"""__GMAP_CENTER_LAT__"""');/g'  $APPDIR/config.inc.php
#sed                                    -i                                    's/define('\"GMAP_CENTER_LNG'\",
'\"22.944927098464987'\");/define('\"GMAP_CENTER_LNG'\",       '"""__GMAP_CENTER_LNG__"""');/g'
$APPDIR/config.inc.php
#sed       -i       's/define('\"GMAP_CENTER_ZOOM'\",       '\"16'\");/define('\"GMAP_CENTER_ZOOM'\",
'"""__GMAP_CENTER_ZOOM__"""');/g'             $APPDIR/config.inc.php

# Update Google analytics tracking code
#sed                    -i                    's#^[                    \t]//define('\"ANALYTICS_TRACKING_CODE'\",
'\"'\");#\tdefine('\"ANALYTICS_TRACKING_CODE'\",                    '"""__GA_Tracking_Code__"""');#g'
$APPDIR/config.inc.php

# Update Google Fusion Table Layer
#sed   -i   's#^[   \t]//define('\"FUSION_TABLE_LAYER'\",   '\"'\");#\tdefine('\"FUSION_TABLE_LAYER'\",
'"""__Fusion_Table_Layer__"""');#g' $APPDIR/config.inc.php
```

```
# Update Mail settings
#sed -i 's/define('\"SUPPORT_EMAIL'\", '\"support@localhost'\");/define('\"SUPPORT_EMAIL'\", '"""__Support_Email__"""');/g' $APPDIR/config.inc.php
#sed -i 's/define('\"MARKET_SMTP_HOST'\", '\"localhost'\");/define('\"MARKET_SMTP_HOST'\", '"""__SMTP_Host__"""');/g' $APPDIR/config.inc.php
#sed -i 's/define('\"MARKET_SMTP_FROM'\", '\"noreply@localhost'\");/define('\"MARKET_SMTP_FROM'\", '"""__SMTP_From__"""');/g' $APPDIR/config.inc.php
#sed -i 's/define('\"MARKET_SMTP_FROM_NAME'\", '\"Virtual city market'\");/define('\"MARKET_SMTP_FROM_NAME'\", '"""__SMTP_From_Name__"""');/g' $APPDIR/config.inc.php
#sed -i 's#^[ \t]//define('\"MARKET_SMTP_USER'\", '\"'\");#\tdefine('\"MARKET_SMTP_USER'\", '"""__SMTP_User__"""');#g' $APPDIR/config.inc.php
#sed -i 's#^[ \t]//define('\"MARKET_SMTP_PASS'\", '\"'\");#\tdefine('\"MARKET_SMTP_PASS'\", '"""__SMTP_Password__"""');#g' $APPDIR/config.inc.php

# Update php/MARKET_Session.class.php
# replace: session_set_cookie_params(0, $this->web_dir . '/' );
# with: session_set_cookie_params(0, '/' );

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname," /etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

# Finally uncomment RedirectMatch in /etc/apache2/sites-available/virtual-city-market.conf
#sudo service apache2 restart

# Also update the /var/www/html/virtual-city-market/lang/en/application.po and ../el/application.po files to match the municipality requirements
# then chown -R www-data /var/www/html/virtual-city-market/lang
# After that access with the web-browser /proxy-ip/virtual-city-market/php/tools/gettext.php in order to update the strings
# And finally chown -R ubuntu /var/www/html/virtual-city-market/lang

##############################################
```

Table C-24: virtualcitymarket.sh

D.2 URENIO

D.2.1 Improve My City

This section describes a heat stack for deploying “Improve My City” in SCP v3.0. The following files are involved in the stack creation:

1. improvemycity.yaml, that is responsible for creating the stack containing the VM;
2. improvemycity.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy Improve My City  on an Ubuntu instance using Heat's
  software orchestration feature.

parameters:
```



```
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.
  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
    description: the name of the server
  Image:
    type: string
    description: the name of the boot image
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    description: Flavor to use for the WordPress server.
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  DB_User:
    type: string
    description: name of the DB User
    default: 'stormjanitor'
    hidden: true
  DB_Password:
    type: string
    description: name of the DB Password
    default: 'password'
    hidden: true

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
```

```
        template: {get_file: fragments/scplight-node_setup.sh}
        params:
          __Console_Node_IP__: {get_param: Console_Fixed_IP}
          __OS_USERNAME__:     {get_param: OS_USERNAME}
          __OS_PASSWORD__:     {get_param: OS_PASSWORD}
          __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
          __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
          __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
          __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  mysql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/mysql-singlenode_setup.sh}
          params:
            __DB_Admin_User__: {get_param: DB_User}
            __DB_Admin_Pass__: {get_param: DB_Password}

  apache_install:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/apache_install.sh}
          params:
            __NOPARAM__: "NOPARAM"

  improvemycity_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/improvemycity-1.0.sh }
          params:
            __DB_User__:        {get_param: DB_User}
            __DB_Password__:    {get_param: DB_Password}

  ImproveMyCity_config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: mysql-singlenode_setup}
      - config: {get_resource: apache_install}
      - config: {get_resource: improvemycity_setup}


  ImproveMyCity_node_port:
    type: OS::Neutron::Port
    properties:
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
```



```
    security_groups: [
      SSH_Security_Group,
      Zabbix_Security_Group,
      WebFEE_Security_Group
    ]

  ImproveMyCity_instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: ImproveMyCity_node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: ImproveMyCity_config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [ImproveMyCity_instance, first_address] }
```

Table C-25: improvemycity.yaml

```
#!/bin/bash -ex
sudo su

export DBHOST='localhost'
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export DBNAME="improvemycity"
export bckqqt=`echo -e '\x60'`
export FULLDBNAME="$bckqqt${DBNAME}$bckqqt"
export DBCOLLATION=$bckqqt'utf8_general_ci'$bckqqt
export APPNAME='improvemycity'
export APPDIR=/var/www/html/$APPNAME

# install composer
curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin --filename=composer
export COMPOSER_HOME="/usr/local/bin/composer"

# Install automation script using Composer:
sudo composer global require joomlatools/console

# Tell your system where to find the executable of joomla-console by adding the composer directory to your
PATH.
# Add the following line to your shell configuration file called either .profile, .bash_profile, .bash_aliases, or
.bashrc.
# This file is located in your home folder.
export PATH="$PATH:~/.composer/vendor/bin"

# Verify the installation
joomla --version

# create the new improve my city (imc) site with the latest available Joomla version
joomla site:create $APPNAME --www=/var/www/html --mysql-login=$DBUSER:$DBPASS --mysql-
```

```
host=$DBHOST --mysql-database=$DBNAME

# install extra languages (greek)
wget -O /tmp/greek.zip http://joomlacode.org/gf/download/frsrelease/18750/162390/el-GR_joomla_lang_full_3.4.2v1.zip && joomla extension:installfile --www=/var/www/html $APPNAME /tmp/greek.zip

# get and install the latest improve my city (imc) extension
wget -O /tmp/com_imc.zip https://github.com/itsam/imc/archive/master.zip && joomla extension:installfile --www=/var/www/html $APPNAME /tmp/com_imc.zip

# get and install the latest imc theme template
wget -O /tmp/tpl_imc.zip https://github.com/icos-urenio/tpl_imc/archive/master.zip && joomla extension:installfile --www=/var/www/html $APPNAME /tmp/tpl_imc.zip

# get and install T3 plugin by Joomlart, run:
wget -O /tmp/plg_T3.zip http://improve-my-city.com/download/plg_system_t3.v2.6.1.zip && joomla extension:installfile --www=/var/www/html $APPNAME /tmp/plg_T3.zip

# get and install the sample database, (script DROPS/CREATE by default imcdb) run:
#wget -O /tmp/imc_sample_data.sql https://raw.githubusercontent.com/icos-urenio/sampledata_imc/master/imc_sample_data_no_db
wget -O /tmp/imc_sample_data.sql https://raw.githubusercontent.com/icos-urenio/sampledata_imc/master/sample_with_categories_el.sql
mysql -u$DBUSER -p$DBPASS -h$DBHOST -Bse "DROP DATABASE IF EXISTS $FULLDBNAME;CREATE DATABASE $FULLDBNAME;USE $FULLDBNAME"
mysql -h$DBHOST -u$DBUSER -p$DBPASS $DBNAME < "/tmp/imc_sample_data.sql"

joomla database:install --www=/var/www/html --mysql-login=$DBUSER:$DBPASS --mysql-host=$DBHOST --mysql-database=$DBNAME --sql-dumps /tmp/imc_sample_data.sql -e $APPNAME

# Now give appropriate permissions in the directory
chown -R www-data.www-data /var/www/html/$APPNAME
chmod -R 755 /var/www/html/$APPNAME

# Remove debug console
sed -i 's/public $debug = '\''1'\'';/public $debug = ''0'';/g' /var/www/html/$APPNAME/configuration.php
sed -i 's/public $MetaDesc = '\''Joomla! - the dynamic portal engine and content management system'\'';/public $MetaDesc = '''';/g' /var/www/html/$APPNAME/configuration.php
sed -i 's/public $MetaKeys = '\''joomla, Joomla'\'';/public $MetaKeys = '''';/g' /var/www/html/$APPNAME/configuration.php

sed -i 's/.*error_reporting.*/error_reporting = E_ALL \& ~E_WARNING \& ~E_DEPRECATED | E_STRICT/g' /etc/php5/apache2/php.ini

cat << EOF > /etc/apache2/sites-available/1-improvemycity.conf
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@improvemycity.dev
    ServerName improvemycity.dev
```



```
    ServerAlias www.improvemycity.dev

    ErrorLog ${APACHE_LOG_DIR}/improvemycity-error.log
    CustomLog ${APACHE_LOG_DIR}/improvemycity-access.log combined

    DocumentRoot /var/www/html
    <Directory /var/www/html/$APPNAME>
      Options FollowSymLinks
      AllowOverride All
    </Directory>
    #RedirectMatch ^/$ /$APPNAME

    ErrorLog  /var/log/apache2/$APPNAME.dev-error_log
    CustomLog /var/log/apache2/$APPNAME.dev-access_log common
</VirtualHost>
EOF

a2dissite 000-default.conf
a2ensite  1-improvemycity.conf
service apache2 restart

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
/usr/bin/mysqldump -u$DBUSER -h$DBHOST -p$DBPASS --databases $DBNAME --add-drop-database --lock-tables > ~/backup/data/dbbackup.sql
tar -zcvf ~/backup/data/$APPNAME.tar.gz $APPDIR/images
EOF
chmod +x ~/backup/scripts/backup.sh

#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################

#Update database
#wget -O /tmp/imc_sample_data.sql https://raw.githubusercontent.com/icos-urenio/sampledata_imc/master/sample_with_categories_el.sql
#mysql -ustormjanitor -ppassword -hlocalhost -Bse "DROP DATABASE IF EXISTS `improvemycity`;CREATE DATABASE `improvemycity`;USE `improvemycity`"
#mysql -hlocalhost -ustormjanitor -ppassword improvemycity < "/tmp/imc_sample_data.sql"


# Update .htaccess
#echo "<IfModule mod_env.c>
#SetEnv HTTPS on
#</IfModule>" >> /var/www/html/improvemycity/.htaccess

#Update configuration.php
#sed -i 's/public $sitename = '\"improvemycity'\";/public $sitename = '"""SMTP_From_Name"""';/g' /var/www/html/improvemycity/configuration.php
#sed -i 's/public $smtphost = '\"localhost'\";/public $smtphost = '"""SMTP_Host"""';/g' /var/www/html/improvemycity/configuration.php
#sed -i 's/public $smtpuser = '\"'\";/public $smtpuser = '"""SMTP_User"""';/g' /var/www/html/improvemycity/configuration.php
#sed -i 's/public $smtppass = '\"'\";/public $smtppass = '"""SMTP_Password"""';/g' /var/www/html/improvemycity/configuration.php

#sed -i 's/public $mailfrom = '\"admin@example.com'\";/public $mailfrom = '"""SMTP_From"""';/g'
```

```
/var/www/html/improvemycity/configuration.php
#sed -i 's/public $fromname = '\"citybranding'\";/public $fromname = """"SMTP_From_Name"""";/g'
/var/www/html/improvemycity/configuration.php


#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname,"
/etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

# Finally uncomment RedirectMatch in /etc/apache2/sites-available/1-improvemycity.conf
#sudo service apache2 restart

##############################################
```

Table C-26: improvemycity.sh

D.3 Municipio de Águeda

D.3.1 Have Your Say

This section describes a heat stack for deploying "Have Your Say" in SCP v3.0. The following files are involved in the stack creation:

1. haveyoursay.yaml, that is responsible for creating the stack containing the VM;
2. haveyoursay.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy Have Your Say on an Ubuntu instance using Heat's software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.

  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string
```



```
  Server:
    type: string
  Image:
    type: string
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  Git_User:
    type: string
    description: name of the Git User for connecting to private account
    hidden: true
    default: ayian2004
  Git_Password:
    type: string
    description: Git Password
    hidden: true
    default: 9Wy%GraFZ9rqfonM

  DB_User:
    type: string
    default: 'stormjanitor'
  DB_Password:
    type: string
    default: 'password'

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  postgresql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/postgresql-singlenode_setup.sh}
          params:
```

```
          __DB_Admin_User__: {get_param: DB_User}
          __DB_Admin_Pass__: {get_param: DB_Password}

  app_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/haveyoursay-1.0.sh }
          params:
            __DB_User__:       {get_param: DB_User}
            __DB_Password__:   {get_param: DB_Password}
            __Git_User__:      {get_param: Git_User}
            __Git_Password__:  {get_param: Git_Password}

  config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: postgresql-singlenode_setup}
      - config: {get_resource: app_setup}

  node_port:
    type: OS::Neutron::Port
    properties:
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
      security_groups: [
        SSH_Security_Group,
        Zabbix_Security_Group,
        WebFEE_Security_Group
      ]

  instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [instance, first_address] }
```

Table C-27: haveyoursay.yaml

```
#!/bin/bash -ex
```



```
cd ~

export APPNAME='haveyoursay'
export DBHOST='localhost'
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export DBNAME="ppgis"

DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install redis-server build-essential subversion graphicsmagick

# install at least one language package
DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install language-pack-pt

# Create pgpass file for accessing psql from command line
cat << EOF > .pgpass
$DBHOST:5432:*:$DBUSER:$DBPASS
EOF
chmod 0600 .pgpass

# Revoke any access to the DB and drop the DB
if psql -h $DBHOST -d postgres -U $DBUSER -w -lqt | cut -d \| -f 1 | grep -w $DBNAME; then
   psql -h $DBHOST -d postgres -U $DBUSER -w << EOF
   REVOKE CONNECT ON DATABASE "$DBNAME" FROM public;
   SELECT pg_terminate_backend(pg_stat_activity.pid)
   FROM pg_stat_activity
   WHERE pg_stat_activity.datname = '$DBNAME';
   DROP DATABASE IF EXISTS "$DBNAME";
   \q
EOF
fi

# Create new database
psql -h $DBHOST -d postgres -U $DBUSER -w << EOF
CREATE DATABASE "$DBNAME" WITH TEMPLATE = template0 ENCODING = 'UTF8' OWNER $DBUSER;
\connect "$DBNAME"
CREATE EXTENSION adminpack;
CREATE EXTENSION postgis;
CREATE EXTENSION hstore;
CREATE EXTENSION pgcrypto;
\q
EOF

su ubuntu -c '
cd /home/ubuntu/

# Install node.js
sudo apt-add-repository ppa:chris-lea/node.js -y
sudo DEBIAN_FRONTEND=noninteractive apt-get -y update
sudo DEBIAN_FRONTEND=noninteractive apt-get -y install nodejs
sudo npm install -g forever
sudo npm install -g forever-service
sudo chown -R $USER:$USER ~/.npm

# Install the PPGIS application
if [ ! -d /home/ubuntu/public_html ]; then
```

```
    mkdir -p /home/ubuntu/public_html
fi
cd /home/ubuntu/public_html/
svn checkout --force https://github.com/jgrocha/geopublic/trunk/server .
svn revert -R .
cp server-config-template.json server-config.json
'

# update server-db.js
cd /home/ubuntu/public_html
sed -i 's#conString = "postgres://geobox:geobox@localhost:5432/geopublic";#conString = "postgres://'$DBUSER':'$DBPASS'@'$DBHOST':5432/'$DBNAME'";#' server-db.js
sed -i 's#conString = "postgres://geobox:geobox@localhost/geopublic";#conString = "postgres://'$DBUSER':'$DBPASS'@'$DBHOST'/'$DBNAME'";#' server-db.js

su ubuntu -c '
cd /home/ubuntu/public_html
sudo apt-get install libpq-dev
npm update
svn checkout --force https://github.com/jgrocha/geopublic/trunk/client/GeoPublic/build/production/GeoPublic public
svn revert -R public

mkdir -p uploads
if [ ! -d /home/ubuntu/public_html/public/participation_data ]; then
    mkdir -p /home/ubuntu/public_html/public/participation_data
fi
if [ ! -d /home/ubuntu/public_html/public/uploaded_images ]; then
    mkdir -p /home/ubuntu/public_html/public/uploaded_images
fi

# Populate supporting tables
wget https://raw.githubusercontent.com/jgrocha/geopublic/master/geopublic-20160115-all.sql
wget https://raw.githubusercontent.com/jgrocha/geopublic/master/geopublic-20160115-data.sql

# Add missing symbolic links for the geopublic/client/GeoPublic/resources/languages/ that are not being copied with the svn checkout command
cd /home/ubuntu/public_html/public/resources/languages

# Removes (-f option) existing destination files, if any, before creating the link
ln -sf pt.js pt-PT.js
ln -sf en.js en-US.js
'

cd /home/ubuntu/public_html

# Populate supporting tables
sed -i 's/geobox/'$DBUSER'/' geopublic-20160115-all.sql
sed -i 's/geobox/'$DBUSER'/' geopublic-20160115-data.sql
psql -h $DBHOST -U $DBUSER $DBNAME -f geopublic-20160115-all.sql
psql -h $DBHOST -U $DBUSER $DBNAME -f geopublic-20160115-data.sql
rm geopublic-20160115-all.sql
rm geopublic-20160115-data.sql

# initial user;
psql -h $DBHOST -U $DBUSER $DBNAME -c "insert into utilizador (idgrupo, email, password, nome, emailconfirmacao) values(1, 'alkiviadis.giannakoulias@eurodyn.com', encode(digest('pa55word', 'sha1'), 'hex'), 'Administrator', true);"
```



```
# This is required to allow the application to run
mkdir -p /home/ubuntu/public_html/$APPNAME
cd  /home/ubuntu/public_html/$APPNAME
ln -s /home/ubuntu/public_html/directppgis .

su ubuntu -c '
# Create the service for the application
cd ~/public_html/
sudo forever-service install -e "NODE_ENV=production" ppgis --script server.js

# Start the application
sudo service ppgis start
'

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
pg_dump -h $DBHOST -U $DBUSER -w -F c -O -v -f /root/backup/data/$DBNAME.sql $DBNAME
tar -zcvf ~/backup/data/$APPNAME.tar.gz /home/ubuntu/public_html/public
EOF
chmod +x /root/backup/scripts/backup.sh

exit 0

#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################
# port where the server run. Port 80 maybe taken by Apache
#sed -i "s/\""port\"": [0-9]\+/\""port\"": 80/" server-config.json
# full address
#sed -i "s/localhost/__Host_URL__/" server-config.json
# smtps host
#sed -i "/\""smtphost\""/c\    \""smtphost\"": \""__SMTP_Host__\"","  server-config.json
#sed -i "/\""smtpport\""/c\    \""smtpport\"": __SMTP_Port__," server-config.json
#sed -i "/\""smtpsecure\""/c\    \""smtpsecure\"": false," server-config.json
#sed -i "/\""smtpfrom\""/c\    \""smtpfrom\"": \""__SMTP_From__\"","  server-config.json
#sed -i "/\""smtpuser\""/c\    \""smtpuser\"": \""__SMTP_User__\"","  server-config.json
#sed -i "/\""smtppass\""/c\    \""smtppass\"": \""__SMTP_Password__\""" server-config.json
#sed -i "/\""url\""/c\    \""url\"": \""http:\/\/__Host_URL__\/__Appl_Name__\"","  server-config.json
#sed -i "/\""urlprefix\""/c\    \""urlprefix\"": \""\/__Appl_Name__\"","  server-config.json
#sed -i "/\""https\""/c\    \""https\"": false," server-config.json

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname,"
/etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

#################################################
```

Table C-28: haveyoursay.sh

D.3.2 Location Plans

This section describes a heat stack for deploying “Have Your Say” in SCP v3.0. The following files are involved in the stack creation:

1. locationplans.yaml, that is responsible for creating the stack containing the VM;
2. locationplans.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy "Location Plan" on an Ubuntu instance using Heat's software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.

  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
  Image:
    type: string
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  Git_User:
    type: string
    description: name of the Git User for connecting to private account
    hidden: true
    default: ayian2004
  Git_Password:
    type: string
    description: Git Password
    hidden: true
    default: 9Wy%GraFZ9rqfonM

  DB_User:
    type: string
```



```
    default: 'stormjanitor'
  DB_Password:
    type: string
    default: 'password'

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  postgresql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/postgresql-singlenode_setup.sh}
          params:
            __DB_Admin_User__: {get_param: DB_User}
            __DB_Admin_Pass__: {get_param: DB_Password}

  app_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/locationplans-1.0.sh }
          params:
            __DB_User__:        {get_param: DB_User}
            __DB_Password__:    {get_param: DB_Password}
            __Git_User__:       {get_param: Git_User}
            __Git_Password__:   {get_param: Git_Password}

  config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: postgresql-singlenode_setup}
      - config: {get_resource: app_setup}

  node_port:
    type: OS::Neutron::Port
    properties:
```

```
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
      security_groups: [
        SSH_Security_Group,
        Zabbix_Security_Group,
        WebFEE_Security_Group
      ]

  instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [instance, first_address] }
```

Table C-29: locationplans.yaml

```
#!/bin/bash -ex

cd ~

export APPNAME='locationplans'
export DBHOST='localhost'
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export DBNAME="locationplans"

# Preparation
locale-gen "en_US.UTF-8"
DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install language-pack-en
DEBIAN_FRONTEND=noninteractive apt-get -y -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install redis-server build-essential subversion graphicsmagick imagemagick htop nmap unzip openssh-server
apt-get -y install libgeos-dev gdal-bin
apt-get -y install git

# Install Node.js
su ubuntu -c '
cd ~

curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash -
sudo DEBIAN_FRONTEND=noninteractive apt-get -y install nodejs
sudo npm install -g forever
sudo npm install -g forever-service
'
```



```
# Install JAVA, Tomcat and MapFish
apt-get install -y python-software-properties debconf-utils
add-apt-repository -y ppa:webupd8team/java
apt-get update
echo "oracle-java8-installer shared/accepted-oracle-license-v1-1 select true" | sudo debconf-set-selections
apt-get install -y oracle-java8-installer

# Install JAVA JAI
cd /tmp
wget http://data.opengeo.org/suite/jai/jai-1_1_3-lib-linux-amd64-jdk.bin
wget http://data.opengeo.org/suite/jai/jai_imageio-1_1-lib-linux-amd64-jdk.bin
sed s/+215/-n+215/ jai_imageio-1_1-lib-linux-amd64-jdk.bin > jai_imageio-1_1-lib-linux-amd64-jdk_fixed.bin

cp jai-1_1_3-lib-linux-amd64-jdk.bin /usr/lib/jvm/java-8-oracle
cp jai_imageio-1_1-lib-linux-amd64-jdk_fixed.bin /usr/lib/jvm/java-8-oracle

cd /usr/lib/jvm/java-8-oracle
yes | sh jai-1_1_3-lib-linux-amd64-jdk.bin
yes | sh jai_imageio-1_1-lib-linux-amd64-jdk_fixed.bin

# Install Tomcat
apt-get -y install tomcat7 tomcat7-admin tomcat7-common
sed -i '/#JAVA_HOME=/c\JAVA_HOME="/usr/lib/jvm/java-8-oracle"' /etc/default/tomcat7
sed -i '/JAVA_OPTS="-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGC"=/c\JAVA_OPTS="-Djava.awt.headless=true -Xmx1536m -XX:+UseConcMarkSweepGC"' /etc/default/tomcat7

sed -i '/<\/tomcat-users>/c\ <role rolename="tomcat"\/>\n <user username="tomcat" password="locationplans2k16" roles="manager,manager-gui,manager-script"\/>\n<\/tomcat-users>' /etc/tomcat7/tomcat-users.xml
service tomcat7 restart

# Install Geoserver
su ubuntu -c '
mkdir -p ~/geoserver/data
'
chown tomcat7:tomcat7 /home/ubuntu/geoserver/data

cd /tmp
wget http://ncu.dl.sourceforge.net/project/geoserver/GeoServer/2.9.0/geoserver-2.9.0-war.zip
unzip geoserver-2.9.0-war.zip
cp geoserver.war /var/lib/tomcat7/webapps/

# Waiting for the webserver to respond
until $(curl --output /dev/null --silent --head --fail http://localhost:8080/geoserver); do
    printf '.'
    sleep 5
done

sleep 2
service tomcat7 stop
cp -a /var/lib/tomcat7/webapps/geoserver/data /home/ubuntu//geoserver

perl -i -p0e 's#<!--
    <context-param>
      <param-name>GEOSERVER_DATA_DIR.*?-->#
```

```
    <context-param>
      <param-name>GEOSERVER_DATA_DIR</param-name>
      <param-value>/home/ubuntu/geoserver/data</param-value>
    </context-param>#s' /var/lib/tomcat7/webapps/geoserver/WEB-INF/web.xml
service tomcat7 start

# Install MapFish
cd /tmp
wget http://repo1.maven.org/maven2/org/mapfish/print/print-servlet/3.5.0/print-servlet-3.5.0.war
cp print-servlet-3.5.0.war /var/lib/tomcat7/webapps/print.war

# Waiting for the webserver to respond
until $(curl --output /dev/null --silent --head --fail http://localhost:8080/geoserver); do
    printf '.'
    sleep 5
done

# Create pgpass file for accessing psql from command line
cd ~
cat << EOF > .pgpass
$DBHOST:5432:*:$DBUSER:$DBPASS
EOF
chmod 0600 .pgpass

# Create new database
psql -h $DBHOST -d postgres -U $DBUSER -w << EOF
CREATE DATABASE "$DBNAME" WITH TEMPLATE = template0 ENCODING = 'UTF8' OWNER $DBUSER;
\connect "$DBNAME"
CREATE EXTENSION adminpack;
CREATE EXTENSION postgis;
CREATE EXTENSION hstore;
CREATE EXTENSION pgcrypto;
\q
EOF

# Deploy the application from GitHub
su ubuntu -c '
cd ~
git clone https://github.com/jgrocha/LocationPlans.git
cd LocationPlans
'

cd /home/ubuntu/LocationPlans
# create database tables
sed -i 's/geobox/'$DBUSER'/' dashboard.sql
psql -h $DBHOST -U $DBUSER $DBNAME -f dashboard.sql
# populate supporting tables
psql -h $DBHOST -U $DBUSER $DBNAME -f dashboard-data.sql
# initial user;
psql -h $DBHOST -U $DBUSER $DBNAME -c "insert into users.utilizador (idgrupo, email, password, nome,
emailconfirmacao) values (1, 'alkiviadis.giannakoulias@eurodyn.com', encode(digest('pa55word', 'sha1'),
'hex'), 'Administrator', true);"

su ubuntu -c '
mkdir -p ~/public_html/public
cd ~/LocationPlans

cp -rf server/* ~/public_html
```



```
cp -rf build/production/Admin/* ~/public_html/public

sudo cp -rf print-apps/plantas /var/lib/tomcat7/webapps/print/print-apps
sudo chown -R tomcat7:tomcat7 /var/lib/tomcat7/webapps/print/print-apps/plantas

cd ~/public_html
sudo apt-get install libpq-dev
npm update
'

forever-service install -e "NODE_ENV=production" dashboard --script server.js
service dashboard start

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
pg_dump -h $DBHOST -U $DBUSER -w -F c -O -v -f /root/backup/data/$DBNAME.sql $DBNAME
tar -zcvf ~/backup/data/$APPNAME.tar.gz /home/ubuntu/public_html/public
EOF
chmod +x /root/backup/scripts/backup.sh

exit 0

#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################
# port where the server run. Port 80 maybe taken by Apache
#sed -i 's/"port": [0-9]\+/"port": 80/' server-config.json
# full address
#sed -i "s/localhost/__Host_URL__/" server-config.json
# production DB
#sed -i '/"dbproduction"/c\    "dbproduction": "postgres://'$DBUSER':'$DBPASS'@'$DBHOST'/'$DBNAME'",'
server-config.json

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname,"
/etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

# Finally uncomment RedirectMatch in /etc/apache2/sites-available/1-improvemycity.conf
#sudo service apache2 restart

#############################################
```

Table C-30: locationplans.sh

D.4 Ayuntamiento de Valladolid

D.4.1 Live the City (Vive)

This section describes a heat stack for deploying “Live the City” in SCP v3.0. The following files are involved in the stack creation:

1. vive.yaml, that is responsible for creating the stack containing the VM;
2. vive.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23
```

```
description: >
  Heat template to deploy Vive on an Ubuntu instance using Heat's
  software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.
  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
    description: the name of the server
  Image:
    type: string
    description: the name of the boot image
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    description: Flavor to use for the WordPress server.
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  Git_User:
    type: string
    description: name of the Git User for connecting to private account
    hidden: true
    default: ayian2004
  Git_Password:
    type: string
    description: Git Password
    hidden: true
    default: 9Wy%GraFZ9rqfonM

  DB_User:
    type: string
    description: name of the DB User
```



```
    default: 'stormjanitor'
    hidden: true
  DB_Password:
    type: string
    description: name of the DB Password
    default: 'password'
    hidden: true

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  mysql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/mysql-singlenode_setup.sh}
          params:
            __DB_Admin_User__: {get_param: DB_User}
            __DB_Admin_Pass__: {get_param: DB_Password}


  apache_install:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/apache_install.sh}
          params:
            __NOPARAM__: "NOPARAM"

  vive_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/vive-1.0.sh }
          params:
            __Git_User__:       {get_param: Git_User}
            __Git_Password__:   {get_param: Git_Password}
```

```
        __DB_User__:       {get_param: DB_User}
        __DB_Password__:   {get_param: DB_Password}

  Vive_config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: mysql-singlenode_setup}
      - config: {get_resource: apache_install}
      - config: {get_resource: vive_setup}


  Vive_node_port:
    type: OS::Neutron::Port
    properties:
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
      security_groups: [
        SSH_Security_Group,
        Zabbix_Security_Group,
        WebFEE_Security_Group
      ]

  Vive_instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: Vive_node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: Vive_config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [Vive_instance, first_address] }
```

Table C-31: vive.yaml

```
#!/bin/bash -ex
sudo su

export DBHOST='localhost'
export DBNAME="vive"
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export bckqqt=`echo -e '\x60'`
export DBCOLLATION=$bckqqt'utf8_general_ci'$bckqqt
export APPNAME='vive'
export APPDIR=/var/www/html/$APPNAME

apt-get install php5-curl
```



```
# Git clone the application source files
cd /var/www/html
git clone https://__Git_User__:__Git_Password__@github.com/STORM-CLOUDS/Vive.git
mv /var/www/html/Vive $APPDIR
chown -R www-data.www-data $APPDIR

# These lines should be un=commented only when deployed in public cloud @ Enter
# Google Analytics support
#echo "<script>
#  (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
#  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
#  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
#  })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

#  ga('create', 'UA-67737651-2', 'auto');
#  ga('send', 'pageview');

#</script>" >> /var/www/html/vive/index.php

#create database
unzip $APPDIR/db_vive_bk.sql.zip -d /tmp
rm $APPDIR/db_vive_bk.sql.zip
mysql -u$DBUSER -p$DBPASS -h$DBHOST << EOF
DROP DATABASE IF EXISTS $DBNAME;
CREATE DATABASE $DBNAME COLLATE $DBCOLLATION;
COMMIT;
QUIT
EOF
mysql -u$DBUSER  -h$DBHOST -p$DBPASS $DBNAME < /tmp/db_vive_bk.sql

sed -i 's/\"host'\" => '\"localhost'\",/'\"host'\" => ""'$DBHOST'"",/g'
$APPDIR/sites/default/settings.php
sed -i 's/'\"database'\" => '\"db_vive_valladolid'\",/'\"database'\" => ""'$DBNAME'"",/g'
$APPDIR/sites/default/settings.php
sed -i 's/'\"username'\" => '\"root'\",/'\"username'\" => ""'$DBUSER'"",/g'
$APPDIR/sites/default/settings.php
sed -i 's/'\"password'\" => '\"mysql@R1adna'\",/'\"password'\" => ""'$DBPASS'"",/g'
$APPDIR/sites/default/settings.php

# Required to fix error messages
mkdir /var/www/html/tmp
chown -R www-data.www-data $APPDIR
chown www-data:www-data /var/www/html/tmp

# Fix for https://github.com/STORM-CLOUDS/Vive/issues/16
if [ ! -f  $APPDIR/sites/all/modules/custom/custom_things_contenttype/custom_things_contenttype.module ];
then
  sed -i 's/\[LANGUAGE_NONE\]//g'
$APPDIR/sites/all/modules/custom/custom_things_contenttype/custom_things_contenttype.module
fi

cat << EOF > /etc/apache2/sites-available/vive.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
```

```
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin smartcity@valladolid.es
        ServerName valladolid-vive

        ErrorLog ${APACHE_LOG_DIR}/vive-error.log
        CustomLog ${APACHE_LOG_DIR}/vive-access.log combined

        DocumentRoot /var/www/html/vive
        <Directory /var/www/html/vive>
          Options FollowSymLinks
          AllowOverride All
        </Directory>
        RedirectMatch ^/$ /vive
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
EOF
a2dissite 000-default.conf
a2ensite  vive.conf
service apache2 restart

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
/usr/bin/mysqldump -u$DBUSER -h$DBHOST -p$DBPASS --databases $DBNAME --add-drop-database --
lock-tables > ~/backup/data/dbbackup.sql
tar -zcvf ~/backup/data/$APPNAME.tar.gz $APPDIR/sites/default/files
EOF
chmod +x ~/backup/scripts/backup.sh

#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################

# If HTTPS is required add
#echo '$_SERVER['\"HTTPS'\"] = '\"on'\";' >> $APPDIR/sites/default/settings.php

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname,"
/etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart


##############################################
```

Table C-32: vive.sh

D.5 City Of Miskolc

D.5.1 TiMi

This section describes a heat stack for deploying "TiMi" in SCP v3.0. The following files are involved in the stack creation:

1. timi.yaml, that is responsible for creating the stack containing the VM;

2. timi.sh, this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy TiMi public place issue Report on an Ubuntu instance using Heat's software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.
  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
    description: the name of the server
  Image:
    type: string
    description: the name of the boot image
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    description: Flavor to use for the WordPress server.
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'

  DB_User:
    type: string
    description: name of the DB User
    default: 'stormjanitor'
    hidden: true
  DB_Password:
    type: string
    description: name of the DB Password
    default: 'password'
```

```
    hidden: true

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  mysql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/mysql-singlenode_setup.sh}
          params:
            __DB_Admin_User__: {get_param: DB_User}
            __DB_Admin_Pass__: {get_param: DB_Password}

  apache_install:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/apache_install.sh}
          params:
            __NOPARAM__: "NOPARAM"

  TiMi_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/timi-1.0.sh }
          params:
            __DB_User__:        {get_param: DB_User}
            __DB_Password__:    {get_param: DB_Password}

  TiMi_config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: mysql-singlenode_setup}
```



```
      - config: {get_resource: apache_install}
      - config: {get_resource: TiMi_setup}


  TiMi_node_port:
    type: OS::Neutron::Port
    properties:
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
      security_groups: [
        SSH_Security_Group,
        Zabbix_Security_Group,
        WebFEE_Security_Group
      ]

  TiMi_instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: TiMi_node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: TiMi_config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [TiMi_instance, first_address] }
```

Table C-33: timi.yaml

```
#!/bin/bash -ex
sudo su

export DBHOST='localhost'
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export DBNAME="timi"
export bckqqt=`echo -e '\x60'`
export DBCOLLATION=$bckqqt'utf8_general_ci'$bckqqt
export APPNAME='timi'
export APPDIR="/var/www/timi"


apt-get -y install unzip
apt-get -y install drush


# 1. Create database
mysql -u$DBUSER -p$DBPASS -h$DBHOST << EOF
DROP DATABASE IF EXISTS $DBNAME;
CREATE DATABASE $DBNAME COLLATE $DBCOLLATION;
COMMIT;
```

```
QUIT
EOF

# 2. Update php.ini file
PHP_MEMORY_LIMIT="$(sed -ne '/^memory_limit/s/[^0-9]//gp' /etc/php5/apache2/php.ini)"
PHP_MAX_EXECUTION_TIME="$(sed -ne '/^max_execution_time/s/[^0-9]//gp'
/etc/php5/apache2/php.ini)"
PHP_POST_MAX_SIZE="$(sed -ne '/^post_max_size/s/[^0-9]//gp' /etc/php5/apache2/php.ini)"
PHP_UPLOAD_MAX_FILESIZE="$(sed -ne '/^upload_max_filesize/s/[^0-9]//gp'
/etc/php5/apache2/php.ini)"
# THESE SETTINGS ARE NECESSARY FOR THE RIGHT FUNCTIONING
# CHANGE PHP_MEMORY_LIMIT IF LESS THAN 256M
if [ "$PHP_MEMORY_LIMIT" -lt 256 ]; then
   sed -i '/memory_limit/c\memory_limit = 256M' /etc/php5/apache2/php.ini
fi
# CHANGE PHP_MAX_EXECUTION_TIME IF LESS THAN 300s
if [ "$PHP_MAX_EXECUTION_TIME" -lt 300 ]; then
   sed -i '/max_execution_time/c\max_execution_time = 300' /etc/php5/apache2/php.ini
fi
# CHANGE PHP_POST_MAX_SIZE IF LESS THAN 16M
if [ "$PHP_POST_MAX_SIZE" -lt 16 ]; then
   sed -i '/post_max_size/c\post_max_size = 16M' /etc/php5/apache2/php.ini
fi
# CHANGE UPLOAD_MAX_FILESIZE IF LESS THAN 16M
if [ "$PHP_UPLOAD_MAX_FILESIZE" -lt 16 ]; then
   sed -i '/upload_max_filesize/c\upload_max_filesize = 16M' /etc/php5/apache2/php.ini
fi

# 3. Update VHost
cat << EOF > /etc/apache2/sites-available/timi.conf
<VirtualHost *:80>
      # The ServerName directive sets the request scheme, hostname and port that
      # the server uses to identify itself. This is used when creating
      # redirection URLs. In the context of virtual hosts, the ServerName
      # specifies what hostname must appear in the request's Host: header to
      # match this virtual host. For the default virtual host (this file) this
      # value is not decisive as it is used as a last resort host regardless.
      # However, you must set it for any further virtual host explicitly.
      #ServerName www.example.com

      ServerAdmin TiMi@storm.eu
      ServerName TiMi1.0

      ErrorLog ${APACHE_LOG_DIR}/TiMi-error.log
      CustomLog ${APACHE_LOG_DIR}/TiMi-access.log combined

      DocumentRoot /var/www/timi/public_html
      <Directory //var/www/timi/public_html>
        Options FollowSymLinks
        AllowOverride All
      </Directory>
      #RedirectMatch ^/$ /timi
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
EOF
a2dissite 000-default.conf
a2ensite  timi.conf
```



```
service apache2 restart

# 4. Create the Directory Structure
# FIRST REMOVE THE PREVIOUS DIRECTORY STRUCTURE IF EXISTS
if [ -d $APPDIR ]; then
   rm -rf $APPDIR
fi

mkdir -p $APPDIR/public_html
chown -R $USER:$USER $APPDIR/public_html
chmod -R 755 $APPDIR

# 5. Git clone the application source files
mkdir -p $APPDIR/github
cd $APPDIR/github
# DOWNLOAD THE APPLICITAION NECCESSERY FILES FROM GITHUB TO THE GITHUB SUBFOLDER(temporary folder)
wget https://github.com/holding-it/timi/archive/master.zip
cd $APPDIR
# DOWNLOAD THE LATEST DRUPAL 7 CORE FILES IN THE DRUPAL SUBFOLDER(temporary folder) WITH DRUSH(Drush is a command-line shell and scripting interface for Drupal)
drush dl drupal-7 --drupal-project-rename=drupal

# COPY ALL FILES FROM DRUPAL SUBFOLDER TO THE APPLICATION FINAL FOLDER(public_html)
cp -avr $APPDIR/drupal/* $APPDIR/public_html
cp -avr $APPDIR/drupal/.htaccess $APPDIR/public_html
cp -avr $APPDIR/drupal/.gitignore $APPDIR/public_html

cd $APPDIR/public_html
# INSTALLING THE NEW DRUPAL 7 AND CREATE THE DATABASE CONNECTION(with the specified parameters) AUTOMATICLY WITH DRUS (Drush is a command-line shell and scripting interface for Drupal https://github.com/drush-ops/drush)
drush site-install standard --db-url='mysql://'$DBUSER':'$DBPASS'@'$DBHOST'/'$DBNAME" --site-name=__Appl_Name__ --yes

# REMOVE THE UNNECESSARY FILES AND FOLDERS IF EXISTS
if [ -d $APPDIR/public_html/sites/default/files ]; then
   rm -rf $APPDIR/public_html/sites/default/files
fi

if [ -d $APPDIR/public_html/sites/all ]; then
   rm -rf $APPDIR/public_html/sites/all
fi

# 6. Populate database
# DROP ALL TABLES FROM THE NEWLY CREATED DRUPAL DATABASE WICH IS CURRENTLY EMPTY ANYWAY(It's not contains data)
# THIS STEP IS NECESSARY TO BE ABLE TO THE APPLICATION DATABASE SHOULD BE IMPORTED SAFELY
drush sql-drop --database=default --yes
# UNZIP AND COPY THE FINAL PLACE THE APPLICATION FILES AND FOLDERS THAN REMOVE THE UNNECESSARY THINGS
cd $APPDIR/github
unzip $APPDIR/github/master.zip
mysql -u$DBUSER -h$DBHOST -p$DBPASS $DBNAME < $APPDIR/github/timi-master/db/timi.sql

# 7. Move contents
cp -avr $APPDIR/github/timi-master/all $APPDIR/public_html/sites
unzip $APPDIR/public_html/sites/all/libraries.zip -d $APPDIR/public_html/sites/all
```

```
rm $APPDIR/public_html/sites/all/libraries.zip
unzip $APPDIR/public_html/sites/all/modules.zip -d $APPDIR/public_html/sites/all
rm $APPDIR/public_html/sites/all/modules.zip
unzip $APPDIR/public_html/sites/all/themes.zip -d $APPDIR/public_html/sites/all
rm $APPDIR/public_html/sites/all/themes.zip

cp -avr $APPDIR/github/timi-master/files $APPDIR/public_html/sites/default
unzip $APPDIR/public_html/sites/default/files/files.zip -d $APPDIR/public_html/sites/default
rm $APPDIR/public_html/sites/default/files/files.zip

cp -avr $APPDIR/github/timi-master/mob $APPDIR/public_html
rm -rf $APPDIR/github
rm -rf $APPDIR/drupal

# 8. SET THE NECESSARY FOLDER AND FILE PERMISSIONS
chmod 644 $APPDIR/public_html/sites/default/settings.php
chown -R :www-data $APPDIR/public_html/sites/default/files
chmod -R 775 $APPDIR/public_html/sites/default/files
# THIS IS THE PHOTOS FOLDER
# Each issue may include images, but not required
chmod -R 775 $APPDIR/public_html/mob/photos

# 9. UPDATE THE APPLICATION DATABASE AND TRUNCATE THE CACHE TABLES WITH DRUSH
cd $APPDIR/public_html
drush updb --yes

# 10. TURN OFF TAG APPLICATION DISPLAY ERRORS WITH DRUSH
drush vset error_level 0 --yes

# 13. TURN OFF THE APPLICATION MAINTENANCE MODE WITH DRUSH
drush vset maintenance_mode 0 --yes

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
/usr/bin/mysqldump -u$DBUSER -h$DBHOST -p$DBPASS --databases $DBNAME --add-drop-database --
lock-tables > ~/backup/data/dbbackup.sql
tar -zcvf ~/backup/data/$APPNAME.tar.gz $APPDIR/public_html/sites/default/files
EOF
chmod +x ~/backup/scripts/backup.sh


#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################

# 11. SET THE CUSTOM VALUES FOR THE FOOTER
#touch update.sql
#chmod +x update.sql
#echo "
#UPDATE locales_target SET translation ="">update.sql
#
#echo '
#UPDATE locales_target SET translation ='
#
#<div class="row">
#<div class="col-lg-3 col-md-3 col-sm-3 col-xs-12">
#<h4>Useful links</h4>
```



```
#
#<div>
#<a target="_blank" href="http://www.miskolc.hu">Miskolc</a><br />
#<a target="_blank" href="http://www.mvkzrt.hu">Mvk</a><br />
#</div>
#</div>
#
#<div class="col-lg-3 col-md-3 col-sm-3 col-xs-12">
#<h4>Contact</h4>
#
#<div>Telephone: +36-70 000 0000<br />
#E-mail: info@tisztamiskolc.hu<br />
#Mailing Address: Miskolc 20 Pf: 1-3</div>
#</div>
#
#<div class="col-lg-3 col-md-3 col-sm-3 col-xs-12">
#<h4>Possibility</h4>
#
#<div>
#<a target="_blank" href="http://#">New announcement</a><br />
#<a target="_blank" href="http://#">Subscribe for newsletter</a><br />
#<a target="_blank" href="http://#">Location search</a><br />
#</div></div>
#
#<div class="col-lg-3 col-md-3 col-sm-3 col-xs-12">
#<h4>Part of the city</h4>
#
#<div>
#<a target="_blank" href="http://www.varosresz1.hu">Varosresz01</a><br />
#<a target="_blank" href="http://www.varosresz2.hu">Varosresz02</a><br />
#
#</div>
#</div>
#</div>
#
#'
#WHERE lid = 13333 AND language = 'en';
#
#UPDATE block_custom SET body =  '<div class="row rtecenter" id=copyright_bar>Digitalis Miskolc 2016 TIMI.</div>'
#WHERE bid =7;">>update.sql
#drush sql-query --file=update.sql
#rm -rf update.sql

# 12. EMPTY THE CACHE FOR THE NEWLY CHANGED VALUES
#drush cc all

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname," /etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

# Finally if application is accessed behind an HAProxy uncomment RedirectMatch in /etc/apache2/sites-available/timi.conf
#sudo service apache2 restart

# reset admin password
# drush uli
```

```
##############################################
```

Table C-34: timi.sh

D.5.2 OPENDATA

This section describes a heat stack for deploying "TiMi" in SCP v3.0. The following files are involved in the stack creation:

1. opendata.yaml, that is responsible for creating the stack containing the VM;
2. opendata., this is the main bash shell script for install and configuring the application

```
heat_template_version: 2013-05-23

description: >
  Heat template to deploy Events OPEN DATA Source on an Ubuntu instance using Heat's software orchestration feature.

parameters:
# The stacks must receive the name of the municipality as a parameter.
# It is used as the activation key name and as a part of the shared volume name.
  Network_Id:
    type: string
  Subnet_Id:
    type: string
  Console_Fixed_IP:
    type: string

  OS_USERNAME:
    type: string
  OS_PASSWORD:
    type: string
  OS_TENANT_NAME:
    type: string
  OS_TENANT_ID:
    type: string
  OS_AUTH_URL:
    type: string
  OS_REGION_NAME:
    type: string

  Server:
    type: string
    description: the name of the server
  Image:
    type: string
    description: the name of the boot image
    default: trusty-server-cloudimg-amd64-heat-hook
  Flavor:
    type: string
    description: Flavor to use for the WordPress server.
    default: e3standard.x2
    constraints:
      - custom_constraint: nova.flavor
  Key:
    type: string
    default: 'stormjanitor'
```



```
  DB_User:
    type: string
    description: name of the DB User
    default: 'stormjanitor'
    hidden: true
  DB_Password:
    type: string
    description: name of the DB Password
    default: 'password'
    hidden: true

resources:

  scplight-node_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/scplight-node_setup.sh}
          params:
            __Console_Node_IP__: {get_param: Console_Fixed_IP}
            __OS_USERNAME__:     {get_param: OS_USERNAME}
            __OS_PASSWORD__:     {get_param: OS_PASSWORD}
            __OS_TENANT_NAME__:  {get_param: OS_TENANT_NAME}
            __OS_TENANT_ID__:    {get_param: OS_TENANT_ID}
            __OS_AUTH_URL__:     {get_param: OS_AUTH_URL}
            __OS_REGION_NAME__:  {get_param: OS_REGION_NAME}

  mysql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/mysql-singlenode_setup.sh}
          params:
            __DB_Admin_User__: {get_param: DB_User}
            __DB_Admin_Pass__: {get_param: DB_Password}

  postgresql-singlenode_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: {get_file: fragments/postgresql-singlenode_setup.sh}
          params:
            __DB_Admin_User__: {get_param: DB_User}
            __DB_Admin_Pass__: {get_param: DB_Password}

  apache_install:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
```

```
        template: {get_file: fragments/apache_install.sh}
        params:
          __NOPARAM__: "NOPARAM"

  OpenData_setup:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template: { get_file: fragments/opendata-1.0.sh }
          params:
            __DB_User__:        {get_param: DB_User}
            __DB_Password__:    {get_param: DB_Password}

  OpenData_config:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: scplight-node_setup}
      - config: {get_resource: mysql-singlenode_setup}
      - config: {get_resource: postgresql-singlenode_setup}
      - config: {get_resource: apache_install}
      - config: {get_resource: OpenData_setup}


  OpenData_node_port:
    type: OS::Neutron::Port
    properties:
      network_id: {get_param: Network_Id}
      fixed_ips:
      - subnet_id: {get_param: Subnet_Id}
      security_groups: [
        SSH_Security_Group,
        Zabbix_Security_Group,
        WebFEE_Security_Group
      ]

  OpenData_instance:
    type: OS::Nova::Server
    properties:
      name:       {get_param: Server}
      image:      {get_param: Image}
      flavor:     {get_param: Flavor}
      key_name:   {get_param: Key}
      networks:
        - port: {get_resource: OpenData_node_port}
      admin_user: ubuntu
      user_data_format: RAW
      user_data: {get_resource: OpenData_config}

outputs:
  instance_ip:
    description: The internal IP address of the deployed instance
    value: { get_attr: [OpenData_instance, first_address] }
```

Table C-35: opendata.yaml


```
#!/bin/bash -ex
sudo su

export DBHOST='localhost'
export DBUSER=__DB_User__
export DBPASS=__DB_Password__
export DBNAME="opendata"
export bckqqt=`echo -e '\x60'`
export DBCOLLATION=$bckqqt'utf8_general_ci'$bckqqt
export APPNAME='opendata'
export APPDIR="/var/www/opendata"


apt-get -y install unzip
apt-get -y install zip
apt-get -y install drush


# 1. Create databases (MySQL and PostgreSQL)
mysql -u$DBUSER -p$DBPASS -h$DBHOST << EOF
DROP DATABASE IF EXISTS $DBNAME;
CREATE DATABASE $DBNAME COLLATE $DBCOLLATION;
COMMIT;
QUIT
EOF

# Create pgpass file for accessing psql from command line
touch /root/.pgpass
chmod 0600 /root/.pgpass
cat << EOF > /root/.pgpass
$DBHOST:5432:*:$DBUSER:$DBPASS
EOF

# Revoke any access to the DB and drop the DB
if psql -h $DBHOST -d postgres -U $DBUSER -w -lqt | cut -d \| -f 1 | grep -w $DBNAME; then
   psql -h $DBHOST -d postgres -U $DBUSER -w << EOF
   REVOKE CONNECT ON DATABASE "$DBNAME" FROM public;
   SELECT pg_terminate_backend(pg_stat_activity.pid)
   FROM pg_stat_activity
   WHERE pg_stat_activity.datname = '$DBNAME';
   DROP DATABASE IF EXISTS "$DBNAME";
   \q
EOF
fi

# Create new database
psql -h $DBHOST -d postgres -U $DBUSER -w << EOF
CREATE DATABASE "$DBNAME" WITH TEMPLATE = template0 ENCODING = 'UTF8' OWNER $DBUSER;
\connect "$DBNAME"
\q
EOF

# 3. Update php.ini file
PHP_MEMORY_LIMIT="$(sed -ne '/^memory_limit/s/[^0-9]//gp' /etc/php5/apache2/php.ini)"
PHP_MAX_EXECUTION_TIME="$(sed -ne '/^max_execution_time/s/[^0-9]//gp'
/etc/php5/apache2/php.ini)"
PHP_POST_MAX_SIZE="$(sed -ne '/^post_max_size/s/[^0-9]//gp' /etc/php5/apache2/php.ini)"
PHP_UPLOAD_MAX_FILESIZE="$(sed -ne '/^upload_max_filesize/s/[^0-9]//gp'
```

```
/etc/php5/apache2/php.ini)"
# THESE SETTINGS ARE NECESSARY FOR THE RIGHT FUNCTIONING
# CHANGE PHP_MEMORY_LIMIT IF LESS THAN 256M
if [ "$PHP_MEMORY_LIMIT" -lt 256 ]; then
   sed -i '/memory_limit/c\memory_limit = 256M' /etc/php5/apache2/php.ini
fi
# CHANGE PHP_MAX_EXECUTION_TIME IF LESS THAN 300s
if [ "$PHP_MAX_EXECUTION_TIME" -lt 300 ]; then
   sed -i '/max_execution_time/c\max_execution_time = 300' /etc/php5/apache2/php.ini
fi
# CHANGE PHP_POST_MAX_SIZE IF LESS THAN 16M
if [ "$PHP_POST_MAX_SIZE" -lt 16 ]; then
   sed -i '/post_max_size/c\post_max_size = 16M' /etc/php5/apache2/php.ini
fi
# CHANGE UPLOAD_MAX_FILESIZE IF LESS THAN 16M
if [ "$PHP_UPLOAD_MAX_FILESIZE" -lt 16 ]; then
   sed -i '/upload_max_filesize/c\upload_max_filesize = 16M' /etc/php5/apache2/php.ini
fi

# 3. Update VHost
cat << EOF > /etc/apache2/sites-available/opendata.conf
<VirtualHost *:80>
       # The ServerName directive sets the request scheme, hostname and port that
       # the server uses to identify itself. This is used when creating
       # redirection URLs. In the context of virtual hosts, the ServerName
       # specifies what hostname must appear in the request's Host: header to
       # match this virtual host. For the default virtual host (this file) this
       # value is not decisive as it is used as a last resort host regardless.
       # However, you must set it for any further virtual host explicitly.
       #ServerName www.example.com

       ServerAdmin OpenData@storm.eu
       ServerName OpenData1.0

       ErrorLog ${APACHE_LOG_DIR}/OpenData-error.log
       CustomLog ${APACHE_LOG_DIR}/OpenData-access.log combined

       DocumentRoot /var/www/opendata/public_html
       <Directory //var/www/opendata/public_html>
         Options FollowSymLinks
         AllowOverride All
       </Directory>
       #RedirectMatch ^/$ /opendata
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
EOF
a2dissite 000-default.conf
a2ensite  opendata.conf
service apache2 restart

# 4. Create the Directory Structure
# FIRST REMOVE THE PREVIOUS DIRECTORY STRUCTURE IF EXISTS
if [ -d $APPDIR ]; then
   rm -rf $APPDIR
fi

if [ ! -d /$APPDIR/public_html ]; then
```



```
    mkdir -p $APPDIR/public_html
fi

chown -R $USER:$USER $APPDIR/public_html
chmod -R 755 $APPDIR

# 5. Git clone the application source files
if [ ! -d $APPDIR/github ]; then
    mkdir -p $APPDIR/github
fi
cd $APPDIR/github
# DOWNLOAD THE APPLICITAION NECCESSERY FILES FROM GITHUB TO THE GITHUB SUBFOLDER(temporary folder)
# Warning! There must be change https://github.com/STORM-CLOUDS/Events-OPEN-DATA-Source
wget https://github.com/holding-it/$APPNAME/archive/master.zip
cd $APPDIR
# DOWNLOAD THE LATEST DRUPAL 7 CORE FILES IN THE DRUPAL SUBFOLDER(temporary folder) WITH DRUSH(Drush is a command-line shell and scripting interface for Drupal)
drush dl drupal-7 --drupal-project-rename=drupal

# COPY ALL FILES FROM DRUPAL SUBFOLDER TO THE APPLICATION FINAL FOLDER(public_html)
cp -avr $APPDIR/drupal/* $APPDIR/public_html
cp -avr $APPDIR/drupal/.htaccess $APPDIR/public_html
cp -avr $APPDIR/drupal/.gitignore $APPDIR/public_html

cd $APPDIR/public_html
# INSTALLING THE NEW DRUPAL 7 AND CREATE THE DATABASE CONNECTION(with the specified parameters) AUTOMATICLY WITH DRUS (Drush is a command-line shell and scripting interface for Drupal https://github.com/drush-ops/drush)
drush site-install standard --db-url='mysql://'$DBUSER':'$DBPASS'@'$DBHOST'/'$DBNAME" --site-name=__Appl_Name__ --yes

# REMOVE THE UNNECESSARY FILES AND FOLDERS IF EXISTS
if [ -d $APPDIR/public_html/sites/default/files ]; then
    rm -rf $APPDIR/public_html/sites/default/files
fi

if [ -d $APPDIR/public_html/sites/all ]; then
    rm -rf $APPDIR/public_html/sites/all
fi

# 6. Populate database
# DROP ALL TABLES FROM THE NEWLY CREATED DRUPAL DATABASE WICH IS CURRENTLY EMPTY ANYWAY(It's not contains data)
# THIS STEP IS NECESSARY TO BE ABLE TO THE APPLICATION DATABASE SHOULD BE IMPORTED SAFELY
drush sql-drop --database=default --yes
# UNZIP AND COPY THE FINAL PLACE THE APPLICATION FILES AND FOLDERS THAN REMOVE THE UNNECESSARY THINGS
cd $APPDIR/github
unzip $APPDIR/github/master.zip
mysql -u$DBUSER -h$DBHOST -p$DBPASS $DBNAME < $APPDIR/github/$APPNAME-master/db/$APPNAME.sql

# 7. Move contents
cp -avr $APPDIR/github/$APPNAME-master/all $APPDIR/public_html/sites
unzip $APPDIR/public_html/sites/all/libraries.zip -d $APPDIR/public_html/sites/all
rm $APPDIR/public_html/sites/all/libraries.zip
unzip $APPDIR/public_html/sites/all/modules.zip -d $APPDIR/public_html/sites/all
```

```
rm $APPDIR/public_html/sites/all/modules.zip
unzip $APPDIR/public_html/sites/all/themes.zip -d $APPDIR/public_html/sites/all
rm $APPDIR/public_html/sites/all/themes.zip

cp -avr $APPDIR/github/$APPNAME-master/files $APPDIR/public_html/sites/default
unzip $APPDIR/public_html/sites/default/files/files.zip -d $APPDIR/public_html/sites/default
rm $APPDIR/public_html/sites/default/files/files.zip

# 8. Copy the demo data
cp -avr $APPDIR/github/$APPNAME-master/contents/programok_xml.zip $APPDIR/public_html/sites/default/files/opendata
cp -avr $APPDIR/github/$APPNAME-master/contents/shops_pdf.zip $APPDIR/public_html/sites/default/files/opendata
cp -avr $APPDIR/github/$APPNAME-master/contents/shops_sql.zip $APPDIR/public_html/sites/default/files/opendata
cp -avr $APPDIR/github/$APPNAME-master/contents/shops_json.zip $APPDIR/public_html/sites/default/files/opendata
cp -avr $APPDIR/github/$APPNAME-master/contents/shops_xml.zip $APPDIR/public_html/sites/default/files/opendata
zip -FF $APPDIR/github/$APPNAME-master/contents/statues_sql.zip --out $APPDIR/github/$APPNAME-master/contents/statues_sql2.zip
cp -avr $APPDIR/github/$APPNAME-master/contents/statues_sql2.zip $APPDIR/public_html/sites/default/files/opendata
cp -avr $APPDIR/github/$APPNAME-master/contents/statues_xml.zip $APPDIR/public_html/sites/default/files/opendata
cp -avr $APPDIR/github/$APPNAME-master/contents/statues_json.zip $APPDIR/public_html/sites/default/files/opendata
cp -avr $APPDIR/github/$APPNAME-master/contents/statues_pdf.zip $APPDIR/public_html/sites/default/files/opendata

rm -rf $APPDIR/github
rm -rf $APPDIR/drupal

# 9. SET THE NECESSARY FOLDER AND FILE PERMISSIONS
chmod 644 $APPDIR/public_html/sites/default/settings.php
chown -R :www-data $APPDIR/public_html/sites/default/files
chmod -R 775 $APPDIR/public_html/sites/default/files

# 10. UPDATE THE APPLICATION DATABASE AND TRUNCATE THE CACHE TABLES WITH DRUSH
cd $APPDIR/public_html
drush updb --yes

# 11. TURN OFF TAG APPLICATION DISPLAY ERRORS WITH DRUSH
drush vset error_level 0 --yes

# 12. EMPTY THE CACHE FOR THE NEWLY CHANGED VALUES
drush cc all

# 13. TURN OFF THE APPLICATION MAINTENANCE MODE WITH DRUSH
drush vset maintenance_mode 0 --yes

# Setup for backup
cat << EOF > ~/backup/scripts/backup.sh
#!/bin/bash
/usr/bin/mysqldump -u$DBUSER -h$DBHOST -p$DBPASS --databases $DBNAME --add-drop-database --lock-tables > ~/backup/data/dbbackup.sql
pg_dump -h $DBHOST -U $DBUSER -w -F c -O -v -f /root/backup/data/$DBNAME.sql $DBNAME
tar -zcvf ~/backup/data/$APPNAME.tar.gz $APPDIR/public_html/sites/default/files
```



```
EOF
chmod +x ~/backup/scripts/backup.sh


#########################################################
## THE FOLLOWING LINES ARE CUSTOM PER MUNICIPALITY ##
#########################################################

#export Hostname=`hostname`
#sed -i -E "s,(^Hostname=.*),#COMMENTED OUT \1\nHostname=$Hostname,"
/etc/zabbix/zabbix_agentd.conf
#sudo service zabbix-agent restart

# Finally if application is accessed behind an HAProxy uncomment RedirectMatch in /etc/apache2/sites-
available/timi.conf
#sudo service apache2 restart

# reset admin password
# drush uli

#################################################
```

Table C-36: opendata.sh