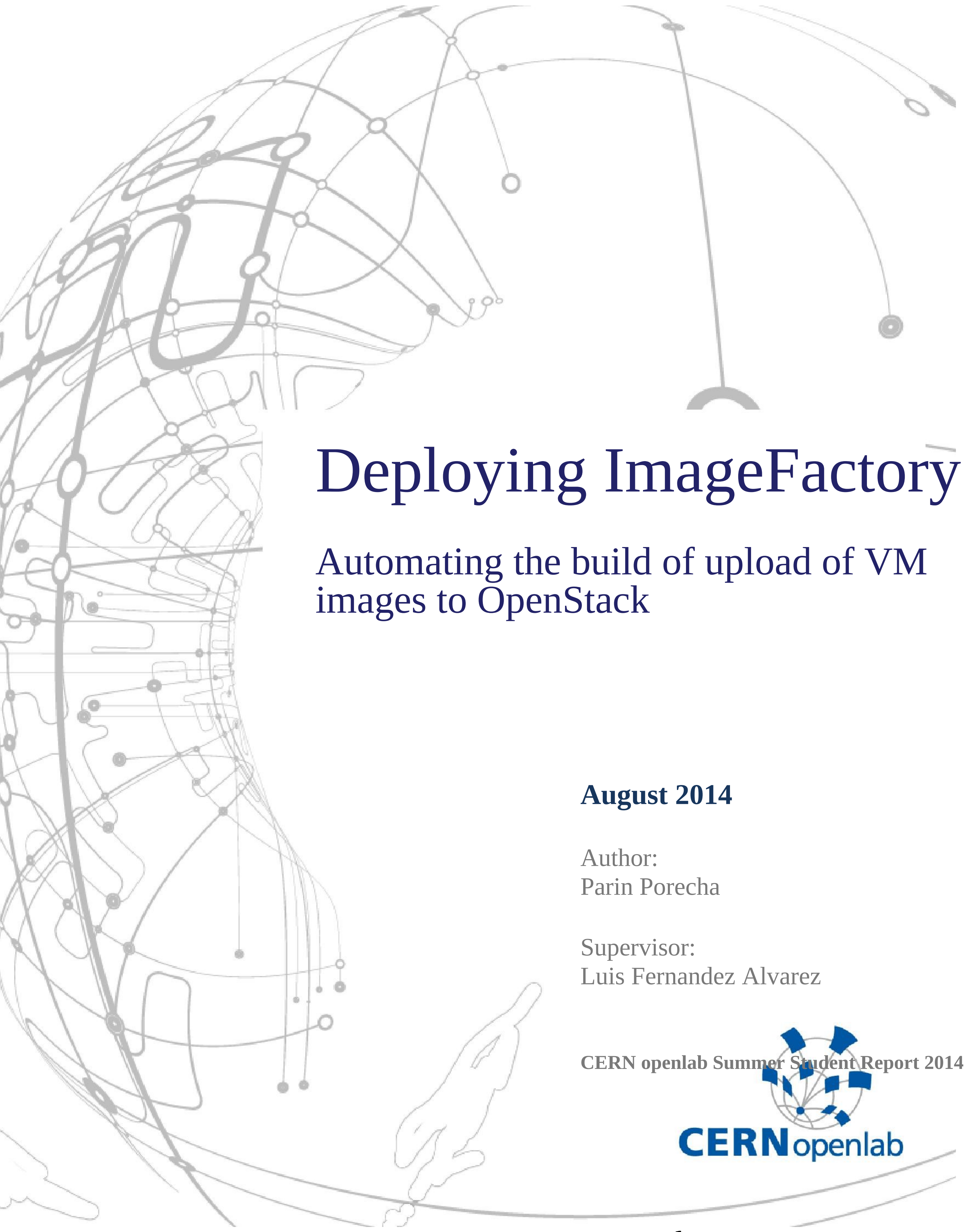


Deploying ImageFactory

Automating the build of upload of VM images to OpenStack

August 2014

Author:
Parin Porecha

Supervisor:
Luis Fernandez Alvarez

CERN openlab Summer Student Report 2014

Abstract

The common practice between OpenStack users is to manually install a base operating system, boot it up, install packages, add necessary configuration and then snapshot it for later use. Much of this can be automated using kickstart files, Puppet, etc. but it’s still a tedious process.

That’s where Image Factory comes into play. It allows you to describe your virtual image (the operating system, architecture, installed packages, etc.) and have it uploaded to various cloud providers (such as EC2, oVirt or OpenStack). This could be useful to the OpenStack users even without all the cross-cloud features.[1]

Table of Contents

1 Introduction

The main technologies used in this project were – OpenStack, Puppet, ImageFactory, Horizon.

Here is a brief description of each one -

1.1 OpenStack

OpenStack is a free and open-source software cloud computing platform. It is primarily deployed as an infrastructure as a service (IaaS) solution. The technology consists of a series of interrelated projects that control pools of processing, storage, and networking resources throughout a data center, able to be managed or provisioned through a web-based dashboard, command-line tools, or a RESTful API.[2]

The OpenStack team at CERN uses IceHouse release as of August 2014, and Puppet for configuration management.

1.2 Puppet

Puppet is a tool designed to manage the configuration of Unix-like and Microsoft Windows systems declaratively. The user describes system resources and their state, either using Puppet's declarative language or a Ruby DSL (domain-specific language). This information is stored in files called "Puppet manifests". Puppet discovers the system information via a utility called Facter, and compiles the Puppet manifests into a system-specific catalog containing resources and resource dependency, which are applied against the target systems. Any actions taken by Puppet are then reported.[3]

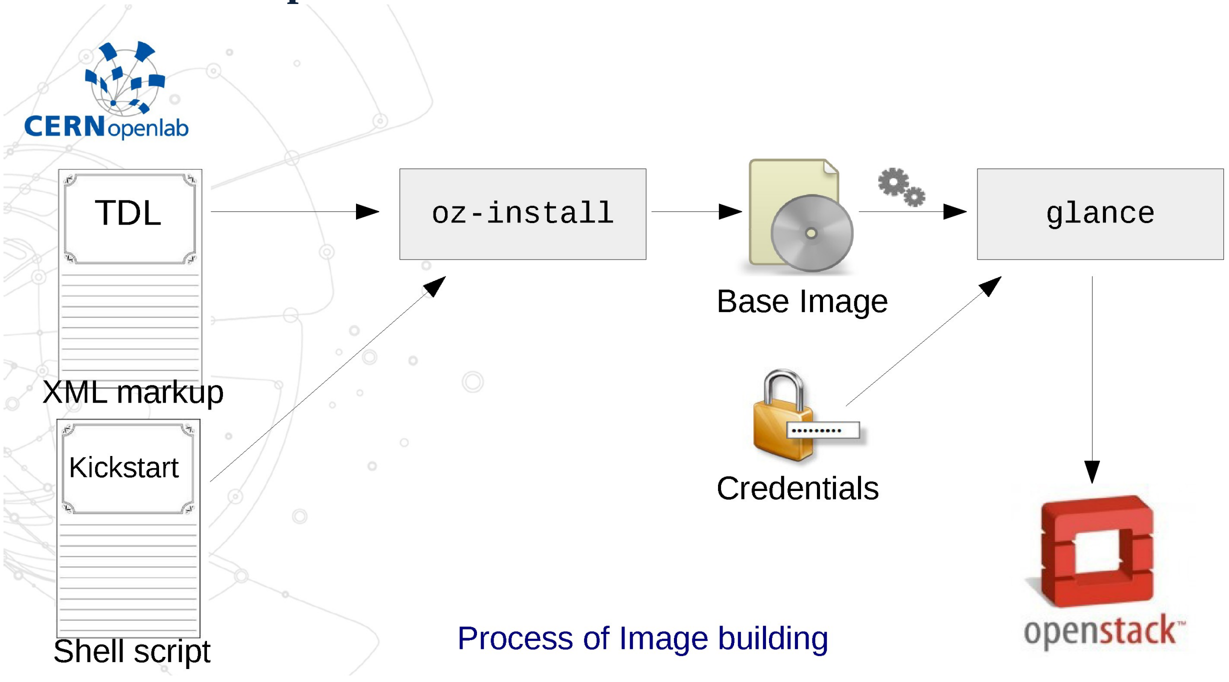
1.3 ImageFactory

ImageFactory is a open-source tool by RedHat which is used to automate the process of building images from a markup file, customizing them according to the target clouds and finally uploading them. Apart from a linux executable, ImageFactory also provides a REST API which can be accessed with appropriate credentials.

1.4 Horizon

Horizon is the canonical implementation of OpenStack's Dashboard, which provides a web based user interface to OpenStack services including Nova, Swift, Keystone, etc. Horizon ships with three central dashboards, a “User Dashboard”, a “System Dashboard”, and a “Settings” dashboard. Between these three they cover the core OpenStack applications and deliver on Core Support.[4]

2. Current Setup



The above flow diagram shows the process of building an image to uploading it.

2.1 Working

- A specification file written in XML markup called a TDL is the main input for oz-install. An optional Kickstart file can also be given containing shell commands for installing base packages, post-install hooks etc.
- oz-install takes the markup and creates a base image. This base image is then de-contextualized (i.e. - all the specific information related to that image is removed (example – hostname), the image is made as generic as possible so that clones can be made from it.
- The next step involves compressing and converting the image to QCOW2 (a format which OpenStack uses) or other formats depending on the cloud it has to be uploaded to.
- The QCOW2 image is then given to glance which uploads it to OpenStack.

2.2 Drawbacks

- The process is tedious. It requires human intervention to run the commands necessary
- Automation is possible but that requires shell scripting. Shell scripts are hard to maintain, and whenever an image needs to be updated, the relevant portion in the script needs to be changed.

- The only way to build an image is by logging in to the server through SSH, and calling the commands/script. Not everyone is comfortable using the command-line, so image building is not possible for them.

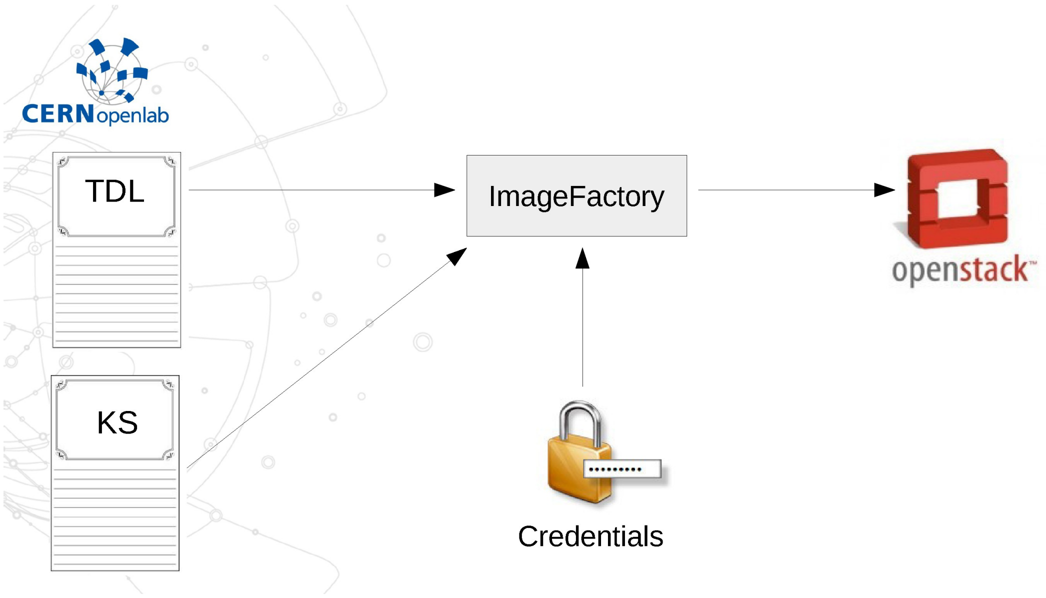
2.3 Proposal

To overcome the drawbacks listed above, it was proposed to replace the whole setup by a single tool which -

- performs almost all of the tasks
- provides multiple ways of building an image
- is easy to maintain.

ImageFactory is a python tool which has solved such problems in the past, so it was decided to deploy it on CERN's image builder machines.

3. Deploying ImageFactory



How ImageFactory replaces all the independent parts with one single layer.

There are 3 inputs along with a provider definition file – a TDL, a Kickstart and credentials.

A TDL file is a XML markup having information regarding the image to be built, such as the URL of the .iso, architecture, version information etc. Here's an example TDL file -

```
<template>
 <name>SLC6 Server - x86_64</name>
 <description>SLC6 Server - x86_64</description>
 <os>
  <name>SLC-6</name>
  <version>5</version>
  <arch>x86_64</arch>
  <install type='iso'>
      <iso>http://linuxsoft.cern.ch/cern/slc65/iso/SLC_6.5_x86_64_dvd.iso
</iso>
  </install>
 </os>
 <packages>
  <package name='virt-what'/>
  <package name='tuned'/>
 </packages>
 <commands>
  <command name='grub'>
exec >>/root/customize.log 2>&amp;1
# Make sure the boot sequence is verbose
[ -f /boot/grub/grub.conf ] &amp;&amp; /usr/bin/perl -ni -e "s/ rhgb//;s/
quiet//;print" /boot/grub/grub.conf || :
```

```
[ -f /boot/grub2/grub.cfg ] &amp;&amp; /usr/bin/perl -ni -e "s/ rhgb//;s/
quiet//;print" /boot/grub2/grub.cfg || :
  </command>
  <command name='tuned'>
exec >>/root/customize.log 2>&amp;1
/usr/bin/tuned-adm profile virtual-guest || :
  </command>
  <command name='de-contextualize'>
exec >>/root/customize.log 2>&amp;1

# cloud-* rpms
/usr/bin/yum install cloud-init -y --enablerepo=slc6-rhcommon

if [ -e /etc/cloud/cloud.cfg ]; then
    /bin/sed -i 's|cloud-user|root|' /etc/cloud/cloud.cfg
    /bin/sed -i 's|^disable_root: 1|disable_root: 0|' /etc/cloud/cloud.cfg

# XXXX cloud.cfg is missing the "growpart" module, which has been
added in
# XXXX the "cloud_init_modules" section in cloud-init-0.7.2-5
# XXXX https://bugzilla.redhat.com/show_bug.cgi?id=966888

fi

/usr/bin/yum install cloud-utils dracut-modules-growroot -y

# clean YUM repo's
/usr/bin/yum clean all --enablerepo=*

# remove and lock root password
/usr/bin/passwd -d root || :
/usr/bin/passwd -l root || :
:
  </command>
 </commands>
</template>
```

A Kickstart file contains the pre-install, post-install hooks along with other configuration code such the drivers and packages to install, configuring the network interface etc.
Here's a part of a Kickstart file -

```
install

# installation path, additional repositories
url --url http://linuxsoft.cern.ch/cern/slc65/x86_64/

repo --name="EPEL" --baseurl
http://linuxsoft.cern.ch/epel/6/x86_64
repo --name="SLC6 - updates" --baseurl
http://linuxsoft.cern.ch/cern/slc6X/x86_64/yum/updates/
repo --name="SLC6 - extras" --baseurl
```

```
http://linuxsoft.cern.ch/cern/slc6X/x86_64/yum/extras/
repo --name="SLC6 - cernonly" --baseurl
http://linuxsoft.cern.ch/onlycern/slc6X/x86_64/yum/cernonly/

text
key --skip
keyboard us
lang en_US.UTF-8
skipx
network --bootproto dhcp
rootpw --iscrypted NOT-A-ROOT-PASSWORD

reboot

#############################################
#
# Packages
#
#############################################
%packages
@ Server Platform
-fprintd
%end

#############################################
#
# post installation part of the KickStart configuration file
#
#############################################
%post --log=/root/anaconda-post.log

/usr/bin/logger "Starting anaconda postinstall"

set -x

#
# Update the machine
#
/usr/bin/yum update -y --skip-broken || :
%end
```

Apart from the TDL and Kickstart, the user needs to supply his credentials, and a provider definition file containing information required to upload the image via python-glanceclient module.

For credentials, the user can give the path to his openrc.sh file which can be downloaded from OpenStack's Horizon dashboard. A provider definition file should be in JSON format and can also contain custom properties if any. Here's an example -

```
{
 "glance-host": "openstack",
 "glance-port": 9292,
 "name": "Parin SLC6 ImageFactory",
 "min_ram": 2048,
 "properties": {
   "hypervisor_type": "kvm",
   "os": "LINUX"
  }
}
```

4. Multiple ways of interaction

There are 3 ways by which the user can use ImageFactory to build an image -

1. SSH to the Image building machine and call the executable. Right now, ImageFactory is deployed on lxbst0518@cern.ch . No root permissions are needed, the below command can be run directly -

   ```
   imagefactory provider_image openstack-kvm <<path to provider
   definition file>> <<path to openrc.sh>> --template <<path to TDL
   file>>  --parameter install_script <<Absolute path to Kickstart file>>
   ```

   Here's an example -

   ```
   imagefactory --debug provider_image openstack-kvm
   ./provider_definition ./openrc.sh --template ./slc6-server-x86_64.tdl
   --parameter install_script
   /afs/cern.ch/user/p/pporecha/slc6-server-x86_64.ks
   ```

   Please note - Giving the Kickstart file as a parameter is a little tricky. Since while installing the image, the script is called in a separate sandbox, the absolute path is required.

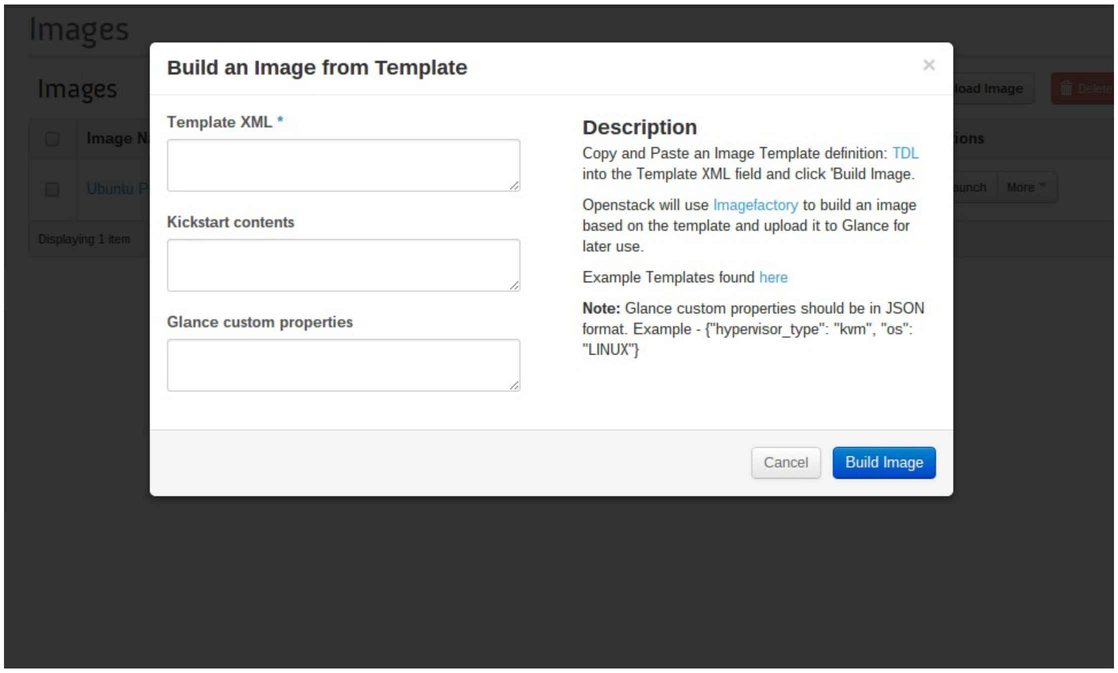
2. To remove the need to SSH to the server, I have written a client script which connects to the library via its REST API. Developers may prefer the SSH way, but this option is recommended for users. Here's how you can use the client -

   1. Download the script from the github repo - https://github.com/parinporecha/CERN-ImageFactory-Client

   2. The parameters are the same, and the README has all the execution details.

   Please note - since the URL of the image builder has been hardcoded in the script, you'll need to change it if the imagefactory builder changes in future.

3. For users who prefer to use a standard GUI instead of SSH or command-line, I have added the functionality to OpenStack Horizon. Right now, it has been deployed in testing mode. When it goes into production, a one-click solution will be available for building images -



Build an Image from Template

Template XML *

Kickstart contents

Glance custom properties

Description

Copy and Paste an Image Template definition: TDL into the Template XML field and click 'Build Image.

Openstack will use Imagefactory to build an image based on the template and upload it to Glance for later use.

Example Templates found here

Note: Glance custom properties should be in JSON format. Example - {"hypervisor_type": "kvm", "os": "LINUX"}

Cancel | Build Image

This is the most simplest way. A user will just need to copy-paste the contents of the TDL, optional Kickstart file and glance custom properties in JSON format.

5. Conclusion

Through this project we achieved -

- A fully automated procedure which could not have been possible by continuing the reliance on Bash scripts.
- Since ImageFactory is maintained by a RedHat, CERN developers do not need to develop their own tool, and thus there is much less code to maintain.
- Users who are not comfortable with using Linux/command-line can also build images once the Horizon feature goes into production.
- Support for new distributions and releases can be added to ImageFactory by just adding their name to supported distros list in the OpenStack plugin.

6. Future work

This project can be extended in several ways -

- **Use Kerberos –** Right now, ImageFactory requires the user to provide the credentials file if called via the executable or through the client script. This requirement can be removed if Kerberos authentication is used instead. Python-kerberos module exists which can be utilized to fetch the login credentials of the user.

- **Integrate with Jenkins –** Since we are using a customized version of ImageFactory and not the standard RPM, the code should be integrated with Jenkins for continuous integration.

- **Move Image Building to the cloud itself –** ImageFactory builds images on the machine it is deployed and then uploads them to the cloud. There is an open-source tool – Nova Image builder (https://github.com/redhat-imaging/novaimagebuilder) which directly builds images on the cloud and saves them there. This tool can be used in the future when ImageFactory has proved itself to be stable.

- **Add support for Windows images** – Currently, ImageFactory supports Fedora 7-19, RHEL 5.x and 6.x, and other Linux distributions. Support for Windows images can be added in the future. This may require using VB scripts to install packages instead of the Kickstart file.

7. Bibliography

1. http://blog.aeolusproject.org/building-openstack-images-with-image-factory/
2. http://en.wikipedia.org/wiki/OpenStack
3. http://en.wikipedia.org/wiki/Puppet_(software)
4. http://docs.openstack.org/developer/horizon/intro.html