Strumenti per la gestione delle immagini di macchine virtuali

Creazione delle immagini,
Marketplaces, contestualizzazione

R. Brunetti, P. Veronesi

Agenda

- Images generation
- Marketplaces
- Contextualization
- Conclusions

IMAGE GENERATION

Generating VM Images

- Machine image creation might be a barrier to cloud adoption by users
  - Time consuming
  - Usually not (or partially) integrated in the cloud stacks
  - Requires knowledge of external tools
- Not completely disjoint from "contextualization"
- Perhaps it should be considered an "Admin's problem" rather than a "user's problem"
  - Brings some important issue on security (see https://documents.egi.eu/document/771 : "Policy for the endorsement and operation of VM images")
- Getting virtual machine images from the WEB
- Creating your own images
  - Manual receipts
  - Images generator tools
    - Oz
    - BoxGrinder
    - OpenNebulaApps - AppStage

Getting virtual machine images from the WEB

- **Ubuntu images**
  - Canonical maintains an official set of Ubuntu-based images These accounts use ubuntu as the login user.
  - If your deployment uses QEMU or KVM, we recommend using the images in QCOW2 format. The most recent version of the 64-bit QCOW2 image for Ubuntu 12.04 is precise-server-cloudimg-amd64-disk1.img.
- **Fedora images**
  - The Fedora project maintains prebuilt Fedora JEOS (Just Enough OS) images for download at http://berrange.fedorapeople.org/images .
  - A 64-bit QCOW2 image for Fedora 16, f16-x86_64-openstack-sda.qcow2, is available for download.
- **OpenSUSE and SLES 11 images**
  - SUSE Studio is an easy way to build virtual appliances for OpenSUSE and SLES 11 (SUSE Linux Enterprise Server) that are compatible with OpenStack. Free registration is required to download or build images.
  - For example, Christian Berendt used OpenSUSE to create a test OpenSUSE 12.1 (JeOS) image.
- **Rackspace Cloud Builders (multiple distros) images**
  - Rackspace Cloud Builders maintains a list of pre-built images from various distributions (RedHat, CentOS, Fedora, Ubuntu) at rackerjoe/oz-image-build on Github.

Creating manually raw or QCOW2 images

O.S. rpm based

1. Get an iso;
2. Install O.S through virt-manager/virsh command
   - Qcow2/raw image
   - Minimal installation
   - No static network
3. Install and configure all the farming tools needed
   - Or use puppet/cloud-init/rc.local script to do it at boot time
4. Install and configure all the application needed
   - Or use puppet/cloud-init/rc.local script to do it at boot time
5. Edit /etc/sysconfig/network-scripts/ifcfg-eth0 and remove the HWADDR= line.
6. Power off the vm

A receipt: http://docs.openstack.org/trunk/openstack-compute/admin/content/manually-creating-qcow2-images.html

- Windows based
  1. Get an iso;
  2. Install O.S through virt-manager/virsh command
     – **OpenStack (KVM) presents the disk using a VIRTIO interface while launching the instance. Hence the OS needs to have drivers for VIRTIO. So download a virtual floppy drive containing VIRTIO drivers from the following location http://alt.fedoraproject.org/pub/alt/virtio-win/latest/images/bin/ and attach it during the installation**
       - Qcow2/raw image
       - Minimal installation
       - No static network
  1. Install and configure all the farming tools needed
     - Or use a management tool to do it at boot time
  2. Install and configure all the application needed
     - Or use a management tool to do it at boot time
  3. Power off the vm

A receipt: http://docs.openstack.org/trunk/openstack-compute/admin/content/creating-a-windows-image.html

Tools for creating images

There are several open-source third-party tools available that simplify the task of creating new virtual machine images.

- **Oz (KVM)**
  - Oz is a command-line tool that has the ability to create images for common Linux distributions. Rackspace Cloud Builders uses Oz to create virtual machines, see rackerjoe/oz-image-build on Githubfor their Oz templates. For an example from the Fedora Project wiki, see Building an image with Oz.
- **BoxGrinder (KVM, Xen, VMWare)**
  - BoxGrinder is another tool for creating virtual machine images, which it calls appliances. BoxGrinder can create Fedora, Red Hat Enterprise Linux, or CentOS images. BoxGrinder is currently only supported on Fedora.

- **VMBuilder (KVM, Xen)**
  - VMBuilder can be used to create virtual machine images for different hypervisors.
  - The Ubuntu 12.04 server guide has documentation on how to use VMBuilder.
- **VeeWee (KVM)**
  - VeeWee is often used to build Vagrant boxes, but it can also be used to build KVM images.
  - See the doc/definition.md and doc/template.md VeeWee documentation files for more details.
- **imagefactory**
  - imagefactory is a new tool from the Aeolus project designed to automate the building, converting, and uploading images to different cloud providers. It includes support for OpenStack-based clouds.

Oz: Overview

https://github.com/clalancette/oz/wiki

- Command line tool to create images for the most common Linux and Windows distributions
- Uses KVM and libvirt to create images
- Uses a Template Description Language (XML) to define
  - O.S. (The input can be a ISO or an URL)
  - Repository
  - Which packages have to be installed
  - Which files have to be created during the installation
  - Which commands have to be executed
- TDL specifications at:
  - https://github.com/clalancette/oz/blob/master/docs/tdl.rng
- Is part of AEOLUS http://www.aeolusproject.org

- Supported O.S. can be seen with: *oz-install -h*
- Currently supported architectures are:
  - i386, x86_64
- Currently supported distros are:
  - Fedora: 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
  - Fedora Core: 1, 2, 3, 4, 5, 6
  - RHEL 2.1: GOLD, U2, U3, U4, U5, U6
  - RHEL/CentOS 3: GOLD, U1, U2, U3, U4, U5, U6, U7, U8, U9
  - RHEL/CentOS/Scientific Linux 4: GOLD, U1, U2, U3, U4, U5, U6, U7, U8, U9
  - RHEL/CentOS/Scientific Linux{,CERN} 5: GOLD, U1, U2, U3, U4, U5, U6, U7, U8
  - RHEL/OEL/CentOS/Scientific Linux{,CERN} 6: 0, 1, 2, 3
  - Ubuntu: 6.06[.1,.2], 6.10, 7.04, 7.10, 8.04[.1,.2,.3,.4], 8.10, 9.04, 9.10, 10.04[.1,.2,.3], 10.10, 11.04, 11.10, 12.04
  - Windows: 2000, XP, 2003, 7, 2008
  - RHL: 7.0, 7.1, 7.2, 7.3, 8, 9
  - OpenSUSE: 10.3, 11.0, 11.1, 11.2, 11.3, 11.4
  - Debian: 5, 6
  - Mandrake: 8.2, 9.1, 9.2, 10.0, 10.1
  - Mandriva: 2005, 2006.0, 2007.0, 2008.0

Oz - File .tdl

```
<template>
        <name>centos-6.3</name>   → Univoco
        <description>centos 6.3</description>
        <os>
            <name>CentOS-6</name>
            <version>3</version>   → Esattemente una delle versioni supportate ottenute col comando oz-install -h
            <arch>x86_64</arch>
          <install type='url'>   → Tipi supportati: url (sorgenti) o iso
                <url>http://centos.mirrors.chicagovps.net/6.3/os/x86_64/</url>
          </install>
          <disk>
                <size>50</size>
          </disk>
          <rootpw>password</rootpw>
        </os>
        <files>
          <file name='/root/test1.txt'> file di prova </file>
        </files>
        <commands>
           <command name="copy"> cp /root/test1.txt /root/test2.txt </command>
            <command name='config-monitoring'>
               chkconfig gmond on
               sed -i "s/allowed_hosts=127.0.0.1/allowed_hosts=openstack-01.cnaf.infn.it/g" /etc/nagios/nrpe.cfg
               chkconfig nrpe on
          </command>
        </commands>
        <repositories>
          <repository name='epel-repo'>
               <url>http://download.fedoraproject.org/pub/epel/6/x86_64/</url>
               <signed>False</signed>
               <persisted>True</persisted>
          </repository>
        </repositories>
        <packages>  → I pacchetti vengono installati dai tool nativi dei SO (yum, apt-get, etc)
          <package name='ganglia-gmond'/>
          <package name='nrpe'/>
          <package name='nagios-plugins'/>
        </packages>
</template>
```

- oz-install –d3 <TDL_FILE>
  - genera un file XML libvirt
  - genera una immagine .dsk che contiene solo il sistema operativo
- oz-customize –d3 <TDL_FILE> <LIBVIRT_XML_FILE>
  - comandi di post-installazione, operazioni sui file, installazione di repository e di pacchetti, customizzare l'immagine
  - il file .dsk viene modificato per contenere i comandi di post-installazione
- oz-install –d3 –u <TDL_FILE>
  - Installa e customizza, e' alternativo all'uso dei 2 comandi precedenti

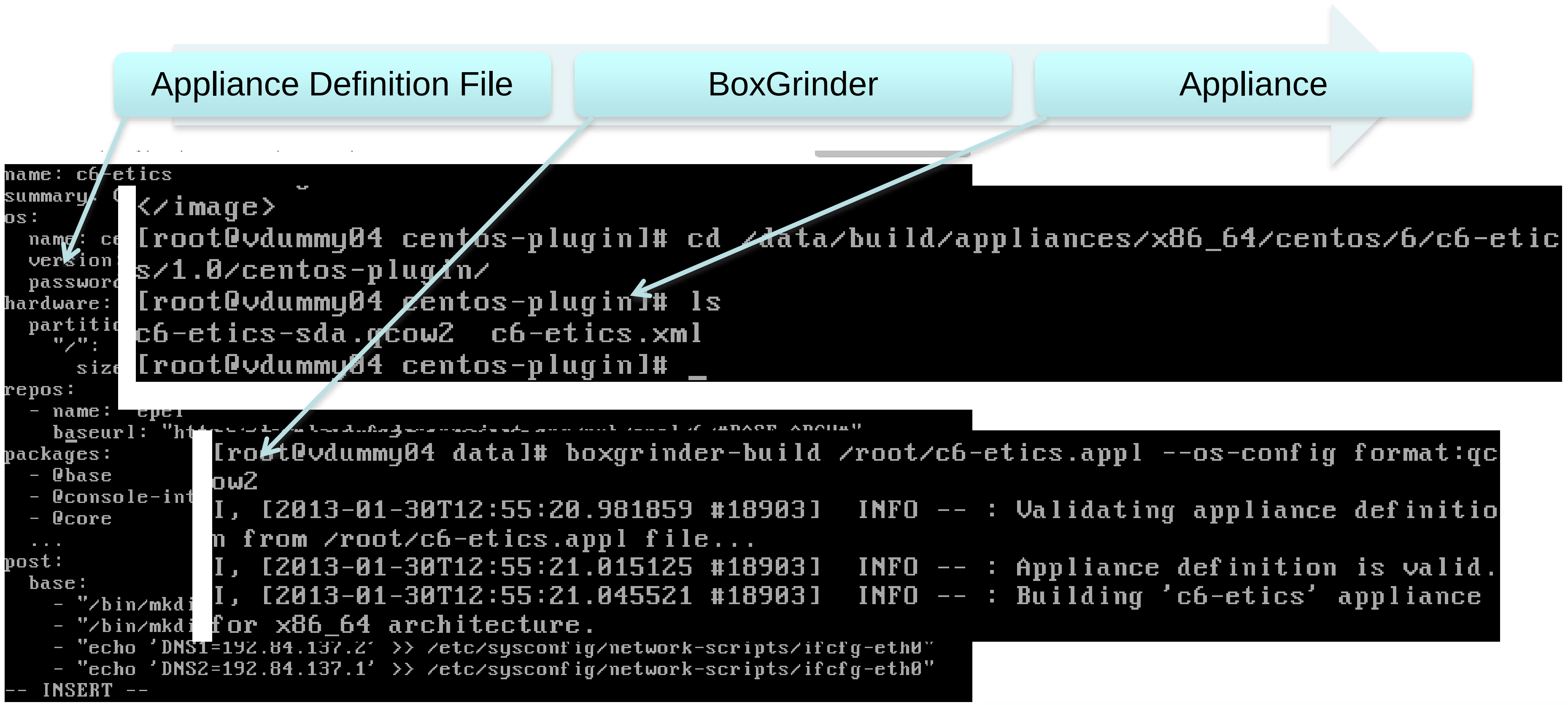
- Centos 6.3
  - Funziona sia da url che da iso
- OpenSUSE 11.4
  - Funziona solo da iso
  - Non sono supportate versioni piu' recenti

BoxGrinder-features

- It creates linux based appliances. Many distributions are already supported using dedicated plugins,
  - Up to now Fedora,RedHat,ScientificLinux and CentOS distributions are supported.
- Allows to specify both remote and local yum repositories,
- Post install instructions can be used to customize appliances, (ie. prepare them for the contextualization phase or to create users.)
- It is however not suitable for creating appliances which are not linux-based.

http://boxgrinder.org

- BoxGrinder creates virtual appliances from a plain text (YAML) Appliance Definition file.



```
name: c6-etics
summary: C
os:
  name: ce
  version:
  passwor
hardware:
  partitio
    "/":
      size
repos:
  - name: epel
    baseurl: "ht
packages:
  - @base
  - @console-int
  - @core
  ...
post:
  base:
    - "/bin/mkdi
    - "/bin/mkdi
    - "echo 'DNS1=192.84.137.2' >> /etc/sysconfig/network-scripts/ifcfg-eth0"
    - "echo 'DNS2=192.84.137.1' >> /etc/sysconfig/network-scripts/ifcfg-eth0"
-- INSERT --
```

```
</image>
[root@vdummy04 centos-plugin]# cd /data/build/appliances/x86_64/centos/6/c6-etic
s/1.0/centos-plugin/
[root@vdummy04 centos-plugin]# ls
c6-etics-sda.qcow2  c6-etics.xml
[root@vdummy04 centos-plugin]# _
```

```
[root@vdummy04 data]# boxgrinder-build /root/c6-etics.appl --os-config format:qc
ow2
I, [2013-01-30T12:55:20.981859 #18903]  INFO -- : Validating appliance definitio
n from /root/c6-etics.appl file...
I, [2013-01-30T12:55:21.015125 #18903]  INFO -- : Appliance definition is valid.
I, [2013-01-30T12:55:21.045521 #18903]  INFO -- : Building 'c6-etics' appliance
for x86_64 architecture.
```

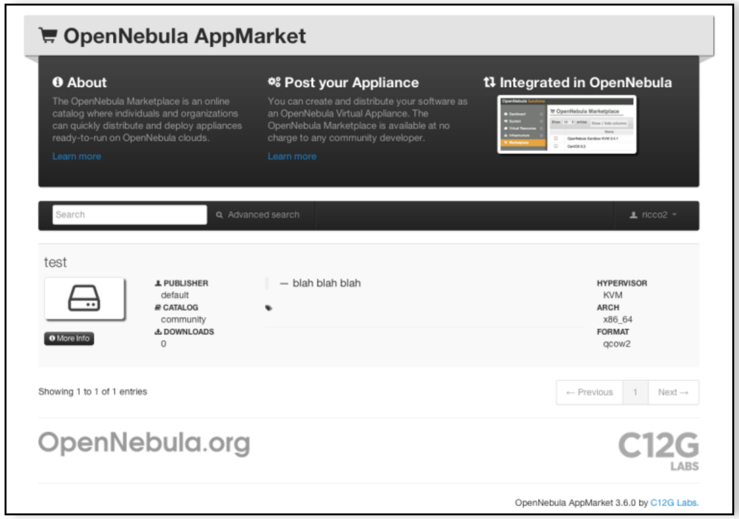
MARKETPLACES

Marketplaces

- Sharing VM images:
  - Increases the usability of our clouds
  - Allows to share knowledge
  - Attracts new users
  - Allows a better control over VM instances
- But…
  - Brings security issues (see JSP doc: https://documents.egi.eu/document/771)
    - Endorsement
    - Integrity
    - Access
    - Control of image “sprawl”
- OpenNebula (AppMarket)
- StratusLab

OpenNebulaApps (AppMarket)

- It is not an image repository but an image registry
- Designed to be a “public” registry.
  - Everyone can download an appliance
  - Only “Developers” (people who have a valid account) can upload.
  - Possibility to define different “catalogues”
  - Images are tagged with MD5/RSA signatures
  - Images may reside on any backend storage which is reachable through an URL
- Can be run standalone (tested) or integrated in OpenNebula
- Interfaces:
  - GUI
  - command line (appmarket)
  - RESTfull



OpenNebula AppMarket

About
The OpenNebula Marketplace is an online catalog where individuals and organizations can quickly distribute and deploy appliances ready-to-run on OpenNebula clouds.
Learn more

Post your Appliance
You can create and distribute your software as an OpenNebula Virtual Appliance. The OpenNebula Marketplace is available at no charge to any community developer.
Learn more

Integrated in OpenNebula

test

PUBLISHER: default
CATALOG: community
DOWNLOADS: 0

— blah blah blah

HYPERVISOR: KVM
ARCH: x86_64
FORMAT: qcow2

Showing 1 to 1 of 1 entries

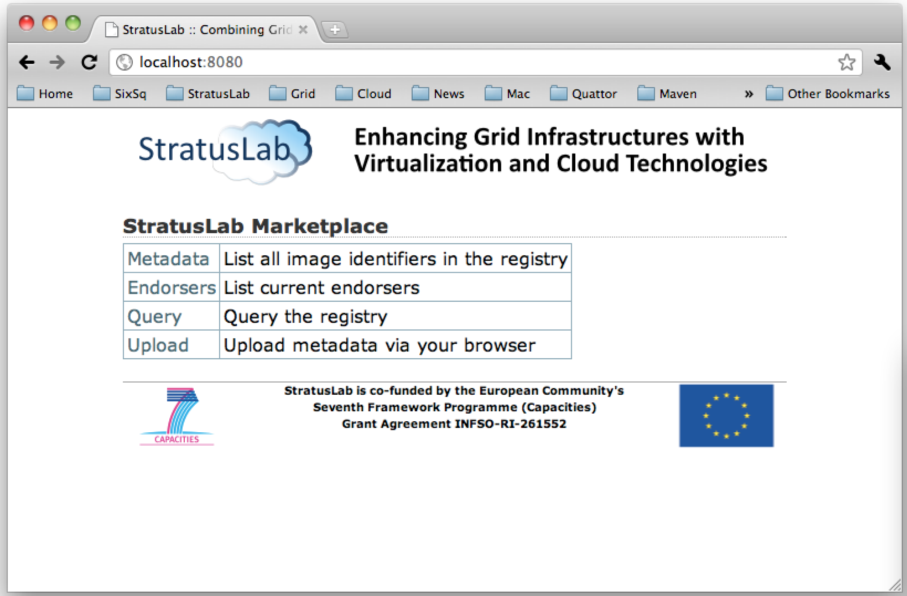
OpenNebula AppMarket 3.6.0 by C12G Labs.

StratusLab Marketplace

- Basically derived from OpenNebula (AppMarket?) with
  some improvements:
  - Image signing with X509 certs
  - Validation
  - Possibility to define local policies
    - Endorsers black/white lists
    - Images black/white lists
  - "Upload the image to cloud, grid, or web storage area" (C. Loomis EGI UF 2011)
    - Not clear what does it mean….

StratusLab

StratusLab Marketplace

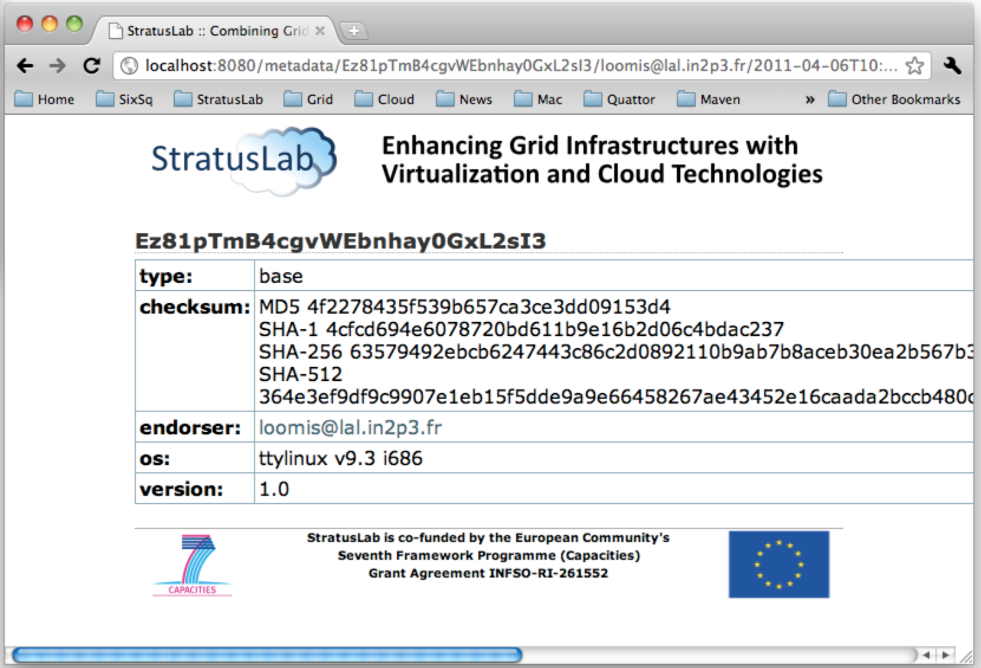


Enhancing Grid Infrastructures with Virtualization and Cloud Technologies

StratusLab Marketplace

Metadata	List all image identifiers in the registry
Endorsers	List current endorsers
Query	Query the registry
Upload	Upload metadata via your browser

StratusLab is co-funded by the European Community's Seventh Framework Programme (Capacities) Grant Agreement INFSO-RI-261552


StratusLab :: Combining Grid
localhost:8080/metadata/Ez81pTmB4cgvWEbnhay0GxL2sI3/loomis@lal.in2p3.fr/2011-04-06T10:...
Home
SixSq
StratusLab
Grid
Cloud
News
Mac
Quattor
Maven
Other Bookmarks
StratusLab
CAPACITIES

Enhancing Grid Infrastructures with Virtualization and Cloud Technologies

Ez81pTmB4cgvWEbnhay0GxL2sI3

type:	base
checksum:	MD5 4f2278435f539b657ca3ce3dd09153d4 SHA-1 4cfcd694e6078720bd611b9e16b2d06c4bdac237 SHA-256 63579492ebcb6247443c86c2d0892110b9ab7b8aceb30ea2b567b3 SHA-512 364e3ef9df9c9907e1eb15f5dde9a9e66458267ae43452e16caada2bccb480
endorser:	loomis@lal.in2p3.fr
os:	ttylinux v9.3 i686
version:	1.0

StratusLab is co-funded by the European Community's Seventh Framework Programme (Capacities) Grant Agreement INFSO-RI-261552

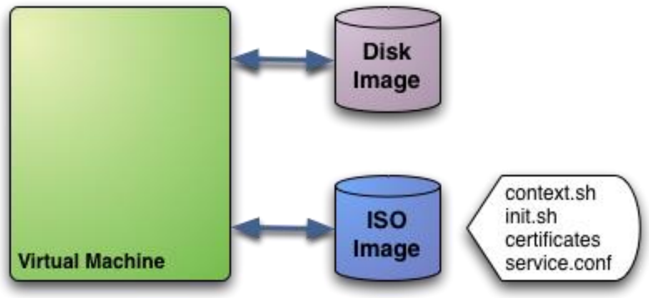
CONTEXTUALIZATION

Contextualization

- Tipically is the end-phase of the VMs provisioning
- Should not depend on the local environment;
  - a VM must rather adapt to the environment.
- Contextualization should be as much as possible “self-consistent”.
- Two possible approaches:
  - Get everything from the user provided template and data.
  - Use an external source of information. The VM “pulls” his configuration from somewhere when it starts

OpenNebula approach

- Very simple (but effective) approach using an ISO image (OVF recommendation)
- The ISO contains all the scripts, config. and data needed to contextualize the VM
- Everything must be referenced inside the “CONTEXT” section on the template definition file
- VM must only mount it and execute the master script if present.



```
CONTEXT = [
  hostname   = "MAINHOST",
  ip_private = "$NIC[IP, NETWORK=\"public net\"]",
  dns        = "$NETWORK[DNS, NETWORK_ID=0]",
  root_pass  = "$IMAGE[ROOT_PASS, IMAGE_ID=3]",
  ip_gen     = "10.0.0.$VMID",
  files      = "/service/init.sh /service/certificates.$UID
/service/service.conf"
]
```

```
CONTEXT=[
  ETH0_IP = "$NIC[IP, NETWORK=\"public\"]",
  ETH0_NETWORK = "$NETWORK[NETWORK_ADDRESS,
NETWORK=\"public\"]",
  ETH0_MASK = "$NETWORK[NETWORK_MASK, NETWORK=\"public\"]",
  ETH0_GATEWAY = "$NETWORK[GATEWAY, NETWORK=\"public\"]",
  ETH0_DNS = "$NETWORK[DNS, NETWORK=\"public\"]",
]
```

Using Cloud-Init

- Python package and a set of associated scripts which can be used to automate the initialization of a virtual machine instance, ie. it can be used to automatically mount the ephemeral disk.
- Available on Ubuntu for some time, but has only recently become available on CentOS 6 / Red Hat Enterprise Linux

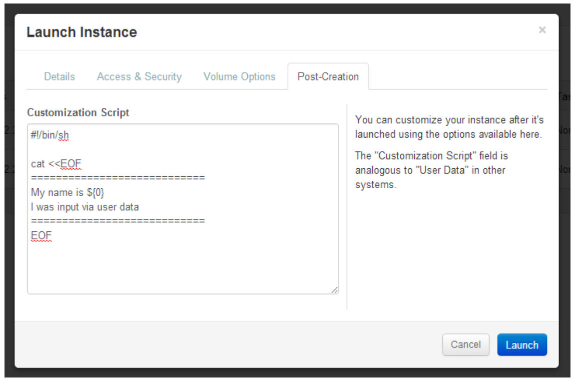
```
cloud-init start running: Fri, 01 Feb 2013 12:57:57 +0000. up 21.27 seconds
found data source: DataSourceEc2

Starting cloud-init-cfg:
Starting system message bus: [  OK  ]^M
Mounting other filesystems:  [  OK  ]^M

Starting cloud-init-cfg: ============================
My name is /var/lib/cloud/instance/scripts/part-001
I was input via user data
============================
ec2:
ec2: ##########################################################
ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
ec2: 1024 f3:06:f9:85:92:78:be:37:7b:65:5a:11:12:32:d4:a3 /etc/ssh/ssh_host_dsa_key.pub (DSA)
ec2: 2048 97:51:4d:a0:11:ec:ef:47:42:38:ec:04:a8:b8:bb:76 /etc/ssh/ssh_host_key.pub (RSA1)
ec2: 2048 2c:37:60:2c:87:a5:b0:13:79:b9:44:ec:dc:77:6b:3e /etc/ssh/ssh_host_rsa_key.pub (RSA)
ec2: -----END SSH HOST KEY FINGERPRINTS-----
ec2: ##########################################################
-----BEGIN SSH HOST KEY KEYS-----
[…]
-----END SSH HOST KEY KEYS-----
cloud-init boot finished at Fri, 01 Feb 2013 12:58:05 +0000. Up 29.02 seconds
```

Automount ephemeral disk as ext3 under /mnt

Just a bash script

Ssh user key

```
# mount
/dev/mapper/VolGroup00-LogVol00 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw,rootcontext="system_u:object_r:tmpfs_t:s0")
/dev/vda1 on /boot type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/dev/vdb on /mnt type ext3 (rw,_netdev)
```



Customization Script

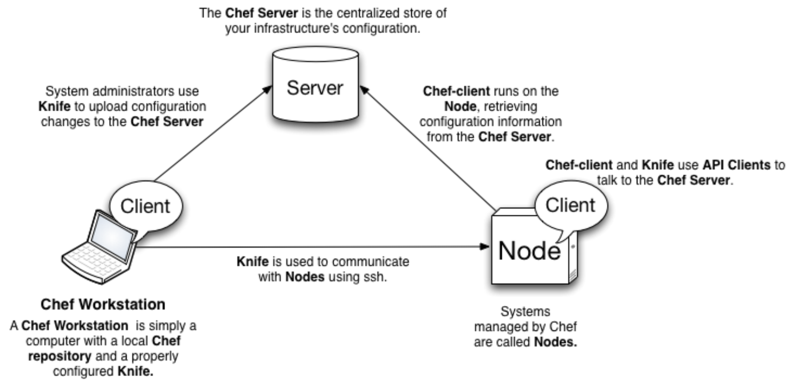
```
#!/bin/sh

cat <<EOF
============================
My name is ${0}
I was input via user data
============================
EOF
```

You can customize your instance after it's launched using the options available here.

The "Customization Script" field is analogous to "User Data" in other systems.

Using an External Tool

- Using an ENC (External Node Classifier) and a configuration management system
- At bootime (and runtime) a client service on the VM asks a server which kind of node the VM is supposed to be.
  - Examples:
    - The Foreman + puppet
    - Chef


The **Chef Server** is the centralized store of your infrastructure's configuration.
Server
System administrators use **Knife** to upload configuration changes to the **Chef Server**
Chef-client runs on the **Node**, retrieving configuration information from the **Chef Server**.
Chef-client and **Knife** use **API Clients** to talk to the **Chef Server**.
Client
Client
Node
Knife is used to communicate with **Nodes** using ssh.
Chef Workstation
A **Chef Workstation** is simply a computer with a local **Chef repository** and a properly configured **Knife.**
Systems managed by Chef are called **Nodes.**

- The server maintains a list of “receipts” to configure the nodes

OpenNebulaApps (AppStage)

- AppStage performs the automatic installation and configuration of the software stack that constitutes an application environment (step forward to PaaS)
- Integrated in the OpenNebula VM workflow
- Uses "*chef-solo*" to perform the installation and configuration of the software stack
  - http://docs.opennebula.pro/features
  - http://wiki.opscode.com/display/chef/Chef+Solo
- Can be "end-user", but requires a significant skill and knowledge (*chef, json* …)
- Can be a powerful tool for the admins to create and distribute complex software platforms (PaaS) that users can instantiate

Ok, where do we go from here?

We can talk days and days but then we need to start to try some solution

- Image generation:
  - Setup a pilot installation of BoxGrinder and/or Oz service and make it available to “everyone”
    - Authn/Authz ??
- Marketplace:
  - Setup an instance of SLM and/or AppMarket
    - SLM seems to be more “featured” but … it’s not mainstream (sustainability?)
  - Start distributing some images which can be “popular” (PBS/LFS servers?, WNs?, LAMP appliances?...)
- Contextualization:
  - Setup an ENC/CMS and make it available to “everyone”
  - Some volunteer for looking at AppStage?

Let’s take some real use case !!
What about preparing a small survey?

MORE SLIDES

OpenNebulaApps (AppStage)

- Can be “end-user”, but requires a significant skill and knowledge (*chef, json* …)
- Can be operated standalone (not tested), but is designed to be integrated in O.N.
- RESTfull API (doc. work in progress)
- Can be a powerful tool for the admins to create and distribute complex software platforms (PaaS) that users can instantiate

Openstack - Image management

- You can use OpenStack Image Services for discovering, registering, and retrieving virtual machine images.
  - The service includes a RESTful API that allows users to query VM image metadata and retrieve the actual image with HTTP requests, or you can use a client class in your Python code to accomplish the same tasks.
- VM images made available through OpenStack Image Service can be stored in a variety of locations from simple file systems to object-storage systems like the OpenStack Object Storage project, or even use S3 storage either on its own or through an OpenStack Object Storage S3 interface.
- The backend stores that OpenStack Image Service can work with are as follows:
  - **OpenStack Object Storage** - OpenStack Object Storage is the highly-available object storage project in OpenStack.
  - **Filesystem** - The default backend that OpenStack Image Service uses to store virtual machine images is the filesystem backend. This simple backend writes image files to the local filesystem.
  - **S3** - This backend allows OpenStack Image Service to store virtual machine images in Amazon’s S3 service.
  - **HTTP** - OpenStack Image Service can read virtual machine images that are available via HTTP somewhere on the Internet. This store is readonly.
- Replicating images across multiple data centers
  - The image service comes with a tool called **glance-replicator** that can be used to populate a new glance server using the images stored in an existing glance server. The images in the replicated glance server preserve the uuids, metadata, and image data from the original.

Ref: http://docs.openstack.org/trunk/openstack-compute/admin/content/ch_image_mgmt.html

About StratusLab

- StratusLab grew from an informal, academic collaboration in 2008 into a formal project co-funded by the European Commission, to develop an open source **IaaS cloud distribution**.
- Now, StratusLab is an **open collaboration** of institutes (CNRS, SixSq, GRNET, and TCD) and individuals that continue to evolve the software and provide support for StratusLab sites and users.
- **Focus on supporting grid services**
- **Based on OpenNebula**

StratusLab Marketplace

Credits: https://indico.egi.eu/indico/materialDisplay.py?contribId=8&sessionId=6&materialId=slides&confId=207

- The Marketplace is at the center of the image handling mechanisms in the StratusLab cloud distribution. It contains metadata about images and serves as a registry for shared images.
- There are two primary use cases for images in the cloud:
  - creating new images and making them available
  - instantiating a referenced image.
- Interfaces
  - REST interface
    - Exposes a simple HTTP-based REST interface
    - Easy to program against in all languages
  - Web interface
    - REST interface also allows browsing via a web browser
    - Signed entries can also be uploaded via the browser

Some details

Image metadata
- Must conform to a defined schema
- Uses the RDF-XML format
- Must be cryptographically signed with a (grid) certificate
- Must contain image ID and checksums to make connection to image
- May contain location elements with image content URL(s)

Typical Marketplace workflow:
- Create image from scratch or based on existing image
- Upload the image to cloud, grid, or web storage area
- Create the metadata for the image
- Sign the metadata with your (grid) certificate
- Upload the signed metadata to the Marketplace

StratusLab cloud will validate image before running it:
- stratus-policy-image: invokes site policy to determine if the referenced image can be used; includes endorser white lists, checksum black lists, etc.
- stratus-download-image: will download a validated image to be used by a VM instance; uses the location URL(s) in the metadata entry