vmcaster

2014-01-21

Virtualisation Working Group Format HEPiX

Owen Synge

vmcaster: 2014-01-21

Virtualisation Working Group Format HEPiX
Owen Synge

Abstract

Virtual Maschine Image list Format as developed by the HVG as an on going document.

Table of Contents

Preface

Table of Contents

`Virtual Machine Image List` are a extendible meta data format used to transfer `Virtual Machine Image` from end user to cloud resource provider. `Virtual Machine Image List` within a working group of HEPiX. HEPiX is a twice yearly conference made up from sites providing compute and storage services for High Energy Physics. HEPiX focusses on the compute, network and storage infrastucture required for High Energy Physics sites.

HEPiX Virtualisation Working Group Observations

The spring 2009 HEPiX conference made the following observations:

- High Energy Physics will want consistent images across sites.
- The Grid is a homogeneous OS environment so requiring High Energy Physics to coordinate OS upgrades across multiple experiments.
- The Grid will need to be shared with non High Energy Physics community.
- High Energy Physics experiments are very coupled to OS.
- Hardware is coupled to OS version.
- Sites believe `Virtual Machine Image`'s are going to be the execution environment for High Energy Physics jobs in the future.
- Change Root, and Xen based `Virtualised Execution Environments` where already present at HEPiX sites.

With these observations the HEPiX Virtualisation Working Group was formed.

A new generation of `Virtualised Execution Environments` such as OpenStack, Open Nebula and Nimbus where already being developed.

HEPiX Virtualisation Working Group Objectives

- Produce a framework to securely run `Virtual Machine Image`'s across multiple sites supporting High Energy Physics.
- Sites need control over `Virtual Machine Image` selection.

With these objectives the HEPiX Virtualisation Working Group was created with Tony Cass as its leader.

Part I. vmcaster

Table of Contents

Initial import from README.md.

Chapter 1. Introduction

Owen Synge

Introduction

Vmcaster is a simple tool for managing and updating your published virtual machines image lists. Following the HEPiX image list format.
The vmcaster software is open source and available at github [https://github.com/hepix-virtualisation/vmcaster]

Users of this application should read security-related policy requirements for the generation and endorsement of trusted virtual machine (VM) images for use on the Grid". [https://edms.cern.ch/document/1080777]

This tool is designed to aid "endorsers" as defined in the security policy, and anyone who wants to publish virtual machine images in a way where tampering will be detected.

vmcaster was designed with the realisation that users typically create new virtual machines images rarely but update them frequently. Vmcaster attempts to be the first of a new generation of image list publishers the minimise the data entry for updating images, with image lists.

The tasks of updating an image and uploading a fresh signed image is now just two quiet short command line operations away. The aim is to make image and image list updates as painless as possible as these are the most common tasks.

Internally the application uses a simple SQL database, SQLite for storing image lists and multiple back ends managing uploading of images / images lists using a facade pattern. This allows transfer protocol to be derived from the meta data of the image list and a configuration file, updates need much less effort.

Chapter 2. Quick Start

Owen Synge

Quick Start

Please note this application has online help.

```
[user] $ vmcaster --help
```

This will always have an up to date list of command line options and state if they take parameters.

vmcaster uses the standard python logging library. To make things simpler the vmcaster command line has "--verbose" and "--quiet" command line options which respectively makes the logging more or less detailed. You can also specify a logging configuration file using "--logcfg". Details of setting up pythons logging library and its associated configuration file are out of scope for this document.

vmcaster requires a configuration file, this file should exist in either "/etc/vmcaster/vmcaster.cfg" or "~/.vmcaster.cfg". An example template is by default installed as "/etc/vmcaster/vmcaster.cfg.template". To get started:

```
[user] $ cp /etc/vmcaster/vmcaster.cfg.template ~/.vmcaster.cfg
```

When it comes to uploading images and image lists you will need to add to this file for your own site specific settings.

Background

An image list contains an array of images and an array of endorsers. Image lists images and endorsers can all have key value pair attributes. Some of these attributes are required in every image list. Similarly, endorsers and images have a set of required meta data. Every image list must have an endorser, but no images are allowed.

The easiest way to start playing with image list publishing is to copy some one else’s hard work.

```
[user] $ wget https://cernvm.cern.ch/releases/image.list \
    --no-check-certificate --output-document CernVM.list.smime
```

This is a large image list from the CernVm project. I provides a lot of images in a lot of formats so as many customers as possible benefit from the CernVM project. We should now import the image list.

```
[user] $ vmcaster \
    --import-imagelist-smime CernVM.list.smime
```

To check the image has been imported correctly we can use the command:

```
[user] $ vmcaster  --list-imagelist
```

To list the endorsers,

```
[user] $ vmcaster  --list-endorser
```

To list images:

```
[user] $ vmcaster  --list-image
```

To display the imagelist:

```
[user] $ vmcaster \
    --select-imagelist e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
    --show-imagelist
```

Image list Object's "dc:identifier" is a special property of an image list, this is the object identifier and can be used to select items for creation, deletion or modification. Each Object has an Identifier, the Endorser object identifier is the value corresponding with "hv:dn". Image objects identifier is the same as an that of image list "dc:identifier".

UUIDs are used for the image list "dc:identifier" and also images "dc:identifier". These UUIDs should be globally unique and consequently the UUID should be generated using a UUID generator using suitable seeds. With Debian, Redhat and Scientific Linux I use the following UUID generator.

```
[user] $ uuidgen
70d9816a-2f6b-4aea-9412-16716b7539b7
```

To show image meta data:

```
[user] $ vmcaster \
    --select-image b36d8b24-c63c-4fd1-ba13-bda6877207e8 \
    --show-image
```

To show endorser meta data:

```
[user] $ vmcaster \
    --select-endorser "/DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=buncic/CN=379010/CN=Predrag
 Buncic" \
    --show-endorser
```

To create an Endorser:

```
[user] $ vmcaster \
    --select-endorser "/C=DE/O=GermanGrid/OU=DESY/CN=Owen Synge" \
    --add-endorser
```

To create an image list:

```
[user] $ vmcaster \
        --select-imagelist  e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
        --add-imagelist
```

To create an image:

```
[user] $ vmcaster \
        --select-image b36d8b24-c63c-4fd1-ba13-bda6877207e8 \
        --add-image
```

To delete an Endorser:

```
[user] $ vmcaster \
        --select-endorser "/C=DE/O=GermanGrid/OU=DESY/CN=Owen Synge" \
        --del-endorser
```

To delete an image list:

```
[user] $ vmcaster \
        --select-imagelist  e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
        --del-imagelist
```

To delete an image:

```
[user] $ vmcaster \
        --select-image b36d8b24-c63c-4fd1-ba13-bda6877207e8 \
        --del-image
```

To add an image to an image list:

```
[user] $ vmcaster \
        --select-imagelist e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
        --imagelist-add-image \
        --select-image 3a1df02c-121a-461d-b720-521903ef99f0
```

To remove an image from the image list:

```
[user] $ vmcaster \
        --select-imagelist e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
        --imagelist-del-image \
        --select-image 3a1df02c-121a-461d-b720-521903ef99f0
```

To add an endorser to an image list:

```
[user] $ vmcaster \
        --select-imagelist e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
        --imagelist-add-endorser \
        --select-endorser "/C=DE/O=GermanGrid/OU=DESY/CN=Owen Synge"
```

To remove an image from the image list:

```
[user] $ vmcaster \
        --select-imagelist e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
        --imagelist-del-endorser \
        --select-endorser "/C=DE/O=GermanGrid/OU=DESY/CN=Owen Synge"
```

To change or add meta data to an endorser:

```
[user] $ vmcaster \
        --select-endorser  "/C=DE/O=GermanGrid/OU=DESY/CN=Owen Synge" \
        --key-set-endorser "hv:ca" \
        --key-value-endorser "/DC=ch/DC=cern/CN=CERN Trusted Certification Authority"
```

To change or add meta data to an image list:

```
[user] $ vmcaster \
        --select-imagelist e38a3fd2-0ed8-11e2-873a-001cc0beb420 \
        --key-set-imagelist "dc:description" \
```

```
        --key-value-imagelist "DESY Image List SHaring service"
```

To change or add meta data to an image:

```
[user] $ vmcaster \
        --select-image "2934ec2b-7a67-4b96-ba16-6775d66898d0" \
        --key-set-image "hv:uri" \
        --key-value-image "https://cernvm.cern.ch/releases/17/cernvm-desktop-2.6.0-4-1-x86.vpc.gz"
```

To delete meta data from endorser:

```
[user] $ vmcaster \
        --select-endorser  "/C=DE/O=GermanGrid/OU=DESY/CN=Owen Synge" \
        --key-del-endorser "hv:ca"
```

To delete meta data from image list:

```
[user] $ vmcaster \
        --select-imagelist e38a3fd2-0ed8-11e2-873a-001cc0beb420 \
        --key-del-imagelist "dc:description"
```

To delete meta data from image:

```
[user] $ vmcaster \
        --select-image "2934ec2b-7a67-4b96-ba16-6775d66898d0" \
        --key-del-image "hv:uri"
```

Object identifiers cannot be modified with vmcaster, they can only be created and destroyed,. This is intentional to prevent accidental errors.

Image lists with clashing "dc:identifier" value is considered disruptive at best and hostile at worst, and it will be noticed by image list subscribers. Similarly not having your endorser details correctly is stated in the endorser.

To change "dc:identifier" this we need to dump the image list to file, change the image list UUID, all images we wish to keep.

```
[user] $ vmcaster \
    --select-imagelist e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
    --show-imagelist > output.json
[user] $ OLD_LIST_UUID=e55c1afe-0a62-4d31-a8d7-fb8c825f92a2
[user] $ NEW_LIST_UUID=`uuidgen`
[user] $ sed -e "s/${OLD_LIST_UUID}/${NEW_LIST_UUID}/" \
        output.json  > input.json
[user] $ vmcaster --import-imagelist-json input.json
```

It might be easiest to edit the JSON with a conventional text editor. As you also want to change many other variables. Note that the new UUID is generated using a UUID generator.

It is recommended that only experienced users with good reason share images between image lists (primarily disk space and as part of automation). For this reason while editing the JSON. So it is probably wise to change all image object identifiers ("dc:identifier" for each image.).

All changes to the original image list can be reverted by re - importing an image list previously stored as a JSON file.

$ vmcaster --import-imagelist-json input.json

To Add an image to an image list:

```
[user] $ vmcaster \
    --select-imagelist e55c1afe-0a62-4d31-a8d7-fb8c825f92a2 \
    --imagelist-add-image \
    --select-image 3a1df02c-121a-461d-b720-521903ef99f0
```

The final setting up task is now to set up the configuration file.

Chapter 3. Howto

Owen Synge

Howto

HowToUpdate Image

Now we can select an image to and update it.

```
[user] $ vmcaster \
        --upload-image /var/lib/libvirt/images/hudson-slave-vm06.desy.de.img \
        --select-image 7b1aea46-8776-4447-9450-00e720fc042c
```

TIP: If you are setting up vmcaster for the first time, you may be presented with errors but need more information to debug. You may find adding one or more "--verbose" parameters to the command will give useful extra debugging information.

Shows the image list as it would be made.

```
[user] $ vmcaster \
        --select-imagelist 9b6fad19-d913-4cca-b77d-c4b4fcd9dc36 \
        --show-imagelist
```

which should now have the "hv:uri" set to the correct path to download the image that was just updated, including the "sl:sha512" is now set and the value of "hv:version" has incremented the version number. Once you are happy with the new image list, it is time to publish this.

HowToUpdate Image List

First check the image list is as you expect:

```
[user] $ vmcaster \
        --select-imagelist 9b6fad19-d913-4cca-b77d-c4b4fcd9dc36 \
        --show-imagelist
```

To update and sign image list.

```
[user] $ vmcaster \
        --select-imagelist 9b6fad19-d913-4cca-b77d-c4b4fcd9dc36 \
        --upload-imagelist
```

Clients

Currently only one image list subscriber exists called vmcatcher.

https://github.com/hepix-virtualisation/vmcatcher

Chapter 4. Configuration

Owen Synge

Environment Variables.

- `HOME`

  Used as a prefix for configuration files and certificates.

- `VMCASTER_CFG`

  Path to the configuration file for vmcaster.

- `VMCASTER_RDBMS`

  SQLite based connection string, This defaults to 'sqlite:///vmcaster.db'. This URL refers to the current working directory, to use an absolute path with SQLite, add an extra slash to the URL like syntax.

- `VMCASTER_LOG_CONF`

  vmcaster uses pythons standard logging module, this sets the configuration file to be used for logging. For specifications on how to set this up see the python logging documentation.

- `VMCASTER_X509_DIR`

  Sets the default directory for the users `x.509` certificates. If not set this value is defaulted to `$HOME/.globus`.

- `VMCASTER_X509_KEY`

  Path to the private key for the signing of image lists.

  Sets the default path for the users `x.509` user key. If not set this value is defaulted to `$HOME/.globus/userkey.pem`.

- `VMCASTER_X509_CERT`

  Path to the certificate for the signing of image lists.

  Sets the default path for the users `x.509` user key. If not set this value is defaulted to `$HOME/.globus/usercert.pem`.

Deployment

It is wise to deploy an "image list subscriber" so you can check for errors with vmcaster as your "image list" publisher. vmcatcher (http://github.com/hepix-virtualisation/vmcatcher) is an "image list subscriber" that has almost the same dependencies as vmcatcher.

Security note

It is more secure to use "https" than "http" for serving the image list. While the authenticity of the image list is secured by the image list signature, the signature cannot prevent "man in the middle" presenting an old signed image list.

For serving images http is as secure as https, as the authenticity of the image is secured by the image list's signature, thus the worst a man in the middle attack can do with images is provide a denial of service when the image does not match the sha512 hash it will be detected and rejected.

It is expected future versions of vmcatcher will issue a warning when subscribing to and retrieving image lists from unauthenticated hosts. Image list subscribers will prefer not to see these warnings, and may decide to only trust image lists hosted on authenticated servers.

Chapter 5. config file

Owen Synge

Configuration file

The configuration file for vmcaster is used to define the hosts and parameters needed to update images and image lists on the servers publishing the image list.

The configuration file is expected system wide at "/etc/vmcaster/vmcaster.cfg" or per user at "~/.vmcaster.cfg". The image list is in ini/cfg format with all values being stored as json. Each section will map to one or more image lists and provides the necessary information to "vmcaster" to update and upload image lists.

The following section is taken from my image list management configuration.

```
[example.org]
server = "www.example.org"
protocol = "scp"
uriMatch = "https://www.example.org/repos/prod/vmcaster"
uriReplace = "user_name@www.example.org:/export/example.org/prod/vmcaster"
```

This states that all image lists to be published on the server "www.example.org", should be updated using the "scp" protocol, and that all writes corresponding with a prefix of "https://www.example.org/repos/prod/vmcaster" should be updated using a prefix of "user_name@www.example.org:/export/example.org/prod/vmcaster". The section name is not significant and is just to group the attributes.

protocols

- scp : This is the standard file transfer tool from the openssh project.
- local : This is when you share a file system with the public server.
- egiappdb : This is the native protocol for the EGI AppDB. This allows you to use vmcaster as a command line tool for updating imagelists.
- gsidcap : For a long time this was the standard POSIX like write protocol for a file storage server called dCache which specialises in storing very large quantities of data at the lowest price possible.

Protocols : Example ’local’

example using the local file system:

```
[foo]
server = "gridvirt.desy.de"
protocol = "local"
uriMatch = "https://gridvirt.desy.de/"
uriReplace = "/tmp/"
```

Note: Publishing an image list without any images is the best way to decommission an image list when no images are expected to be requested ever again.

Protocols : Example ’egiappdb’

Example publishing to the ’egiappdb’

```
[appdb-dev]
server = "vmcaster.appdb-dev.marie.hellasgrid.gr"
protocol = "egiappdb"
uriMatch = ".*"
uriReplace = "egiappdb://example_user@vmcaster.appdb-dev.marie.hellasgrid.gr/vmlist/submit/sso/"

[appdb]
server = "vmcaster.appdb.egi.eu"
protocol = "egiappdb"
uriMatch = ".*"
uriReplace = "egiappdb://_YOUR_EGI_SSO_USERNAME_@vmcaster.appdb.egi.eu/vmlist/submit/sso/"
```

A second example using the local file system:

```
[foo]
server = "gridvirt.desy.de"
protocol = "local"
uriMatch = "https://gridvirt.desy.de/"
uriReplace = "/tmp/"
```

Note: Publishing an image list without any images is the best way to decommission an image list when no images are expected to be requested ever again.

uriMatch and uriReplace

It should be noted that the external download uri is used to generate upload path that is used.

The meta data in the images lists, ’hv:uri’ is used to generate the upload path for the image list. Each image in an imagelist must also has a ’hv:uri’, this is also used to generate the path for uploading images. Processed with the ’regular expression’ matched by ’uriMatch’ and replaced with value of the key in ’uriReplace’. The best uriMatch is the most specific, but due to the flexibility it offers the user even very generic ’uriMatch’ values can be used.

```
uriMatch = ".*/"
```

This is a near universal ’uriMatch’ value as it will always leave some part of the ’filename’ of the External URI in the upload path. Normally I would recommend including at least the hostname and the shared root path for example:

```
uriMatch = "https://www.example.org/repos/imagelist"
```

Only in the case of uploading imagelists over the ’egiappdb’ should you ever use the expression ".*" as this will result in all files being uploaded to the same location.

Chapter 6. Appendix

Owen Synge

Road map

The code should also be publishable into a message Queue service provided by a cloud provider allowing unpublished images to be shared. This work has not been started.

If this application is not suitable for your use case it should not be forgotten that SMIME and json are common standards and can be created in many ways. Below is an example of how to sign a message using SMIME.

```
[user] $ openssl smime -sign -signer .globus/usercred.pem -inkey .globus/userkey.pem \
    -in imagelist.json -out imagelist.json.signed
```

This may prove useful in the long term.