vm5k Documentation

Release 0.7-dev

Laurent Pouilloux

Mar 23, 2018

Contents

A Python module designed to perform reproducible cloud experiments. It will help you to manage virtual machines on the *Grid'5000 <https://www.grid5000.fr/>* plaform.

It is composed of three main tools:

- a command line tool that deploy virtual machines (vm5k)
- an experimental engine that conduct user defined workflow (vm5k_engine)
- a lib to setup Debian hosts with libvirt and manage virtual machines

Developped by the Hemera initiative (2010-2014).

CHAPTER 1

Readme for the vm5k package

A python module to ease the experimentations of virtual Machines on the Grid‘5000 platform. It is composed of:

- a script that deploy virtual machines (vm5k)
- an experimental engine that conduct user defined workflow for a set of parameters (vm5k_engine)
- a lib to setup Debian hosts with libvirt and manage virtual machines

Developed by the Inria Hemera initiative 2010-2014 https://www.grid5000.fr/mediawiki/index.php/Hemera

See documentation on http://vm5k.readthedocs.org

1.1 Requirements

The module requires:

- execo 2.4, <http://execo.gforge.inria.fr/>

1.2 Installation

Connect on a Grid‘5000 frontend and type the following commands:

```
export http_proxy="http://proxy:3128"
export https_proxy="https://proxy:3128"
easy_install --user execo
easy_install --user vm5k
```

Add .local/bin to your path and run vm5k !

1.3 People

1.3.1 Contributors

- Laurent Pouilloux
- Daniel Balouek-Thomert
- Jonathan Rouzaud-Cornabas
- Flavien Quesnel
- Jonathan Pastor
- Takahiro Hirofuchi
- Adrien Lèbre

1.3.2 Grid'5000 technical support

- Matthieu Imbert
- Simon Delamare

CHAPTER 2

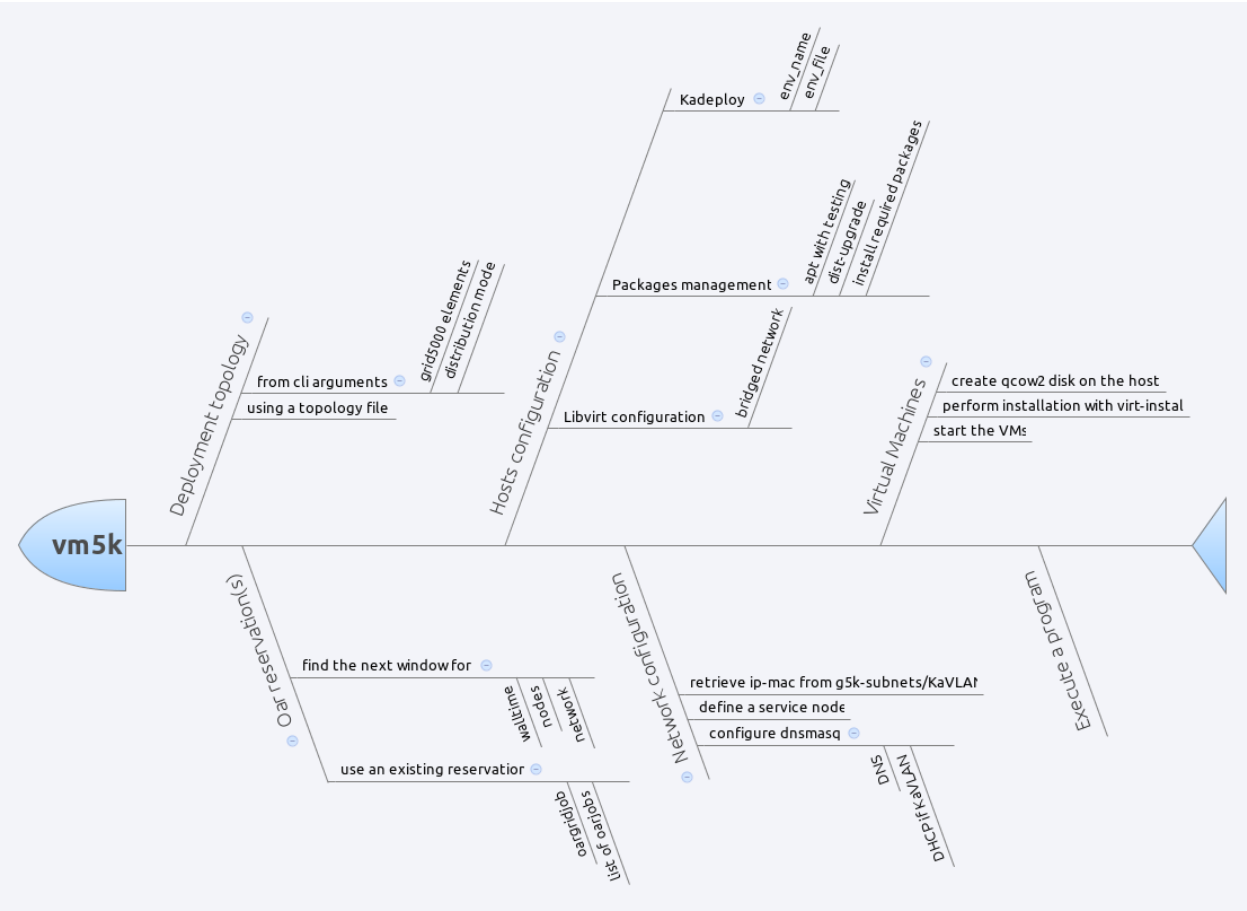
vm5k: automatic virtual machines deployment

Vm5k is a tool used to deploy a large number of virtual machines on the Grid‘5000 platform. It provides several options to customize your environments and topology.

2.1 Workflow

- define a deployment **topology** on Grid‘5000:
  - distributed virtual machines using a template and a list of clusters/sites
  - or from a given xml file (see example below)
- manage the **reservation**:
  - find the next window available for the deployment
  - or use an existing reservation
- install the **hosts**
  - deployment of a kadeploy environment name/file
  - upgrade the hosts and configure libvirt
  - create the backing file for the virtual machine
- configure the **network**
  - determine the parameters from the oar/oargridjob
  - generate dnsmasq configuration
- deploy the **virtual machines**
  - create the qcow2 disks on the hosts
  - perform installation with virt-install
  - start the virtual machines

- execute a **program** on the frontend



2.2 Basic usage

The basic usage is to create a certain number of virtual machines on Grid'5000. To deploy 100 VMs on *wheezy-x64-base* hosts and with the *wheezy-x64-base.qcow2* KVM image on any Grid5000 cluster with hardware virtualization, for 2 hours:

```
vm5k --n_vm 100 -w 2:00:00
```

This will automatically find free nodes on Grid'5000 that can sustains your virtual machines, perform the reservation and deploy hosts and VMs automatically.

2.2.1 Choose a distribution for the VMs

Default distribution follow a `round-robin` mechanism, i.e. adding vm to host while cycling around them and checking that it can sustain more VM. But you may want to have a the same number of VMs on all hosts. For that use `n_by_hosts`:

```
vm5k -r grid5000:20 -n 100 -d n_by_hosts
```

You can also have a `concentrated` distribution meaning that next host will be used when the previous one cannot sustain more VM, i.e. have enough memory to start it:

```
vm5k -r grid5000:20 -n 100 -d concentrated
```

To control more finely the distribution, you must use the infile option, that is described in Topology file. A generated one can be found in vm5k outdir after deployment or in examples directory of vm5k package.

2.2.2 Select the hosts hardware

If you want to test your application on a specific hardware (CPU, RAM, ...), you can select the Grid'5000 elements you want to use by giving a list of cluster or sites:

```
vm5k --n_vm 100 -r hercule,griffon,graphene  -w 2:00:00
```

or select the number of hosts you want on each element:

```
vm5k --n_vm 100 -r taurus:4,nancy:10 -w 2:00:00
```

See https://www.grid5000.fr/mediawiki/index.php/Special:G5KHardware for more details on the cluster hardware.

2.2.3 Define the VMs template

You can customize the virtual machines components by defining a template:

```
vm5k --n_vm 20 --vm_template '<vm mem="4096" hdd="10" n_cpu="4" cpuset="auto"/>'
```

or using Topology file.

2.2.4 Launch a program after the deployment

If you already have an experimental script that must be run on the deployed hosts and VMs, you can use -p option:

```
vm5k --n_vm 100 -p myscript.sh -o myxp
```

You can access the list of hosts and VMs in myxp directory in simple csv or in XML format. Have a look to the file vm5k/examples/boot_time.py for a simple example in Python.

2.3 Advanced usage

2.3.1 Use an existing job

You may use an existing reservation:

```
vm5k --n_vm 100 -j 42895
vm5k --n_vm 10 -j grenoble:1657430
vm5k --n_vm 45 -j grenoble:1657430,toulouse:415866,rennes:673350
```

It will retrieve the hosts that you have, deploy and configure them, and finally distribute the VMs on them.

You can also know how many VMs can be run on a list of hosts (checking RAM availability) using:

```
vm5k_max_vms -j 42895 -t '<vm mem="2048" hdd="10" cpu="4" cpuset="auto"/>'
```

2.3.2 Customize the environments of the hosts and VMs

To perform your experiments, you may want to use specific environments to test the effect of various configurations (distribution version, kernel parameters, vm disk, ...). You can choose hosts operating system with:

```
vm5k --n_vm 50 --walltime 2:00:00 --env_name wheezy-x64-prod
vm5k --n_vm 50 --walltime 2:00:00 --env_name user:env_name
vm5k --n_vm 50 --walltime 2:00:00 --env_file path/to/your/env_file
```

You may also want to use your virtual machines disk:

```
vm5k --n_vm 50 --walltime 2:00:00 --vm_backing_file path_to_my_qcow2_file_on_g5k
```

For more complex situtation, i.e. using different backing_file for the VMs, you need to use the XML topology_file.

2.3.3 Deploy in an isolated vlan

Grid'5000 offers the possibility of using KaVLAN to deploy your nodes in an isolated VLAN, https://www.grid5000.fr/mediawiki/index.php/Network_isolation_on_Grid%275000. You can use it in vm5k with:

```
vm5k --n_vm 100 -r reims -w 2:00:00 -k
vm5k --n_vm 100 -r taurus:4,nancy:10 -w 2:00:00 -k
vm5k --n_vm 600 -r grid5000:100 -w 2:00:00 -k -b reims
```

When using global kavlan, i.e. a isolated VLAN on multiple sites, you must blacklist reims due to bug 4634

2.3.4 Use a topology file

To have the finest control on the deployment topology, you can use an input file that described the topology and VMs characteristics:

```
vm5k -i topology_file.xml -w 6:00:00
```

where *topology_file.xml* is:

```
<vm5k>
  <site id="luxembourg">
    <cluster id="granduc">
      <host id="granduc-2">
        <vm mem="2048" hdd="4" id="vm-11" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-12" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-13" cpu="1"/>
      </host>
      <host id="granduc-3">
        <vm mem="2048" hdd="4" id="vm-14" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-15" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-16" cpu="1"/>
      </host>
      <host id="granduc-5">
        <vm mem="2048" hdd="4" id="vm-17" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-18" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-19" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-20" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-21" cpu="1"/>
      </host>
```

```
      <host id="granduc-9">
        <vm mem="2048" hdd="4" id="vm-22" cpu="1"/>
      </host>
    </cluster>
  </site>
  <site id="lyon">
    <cluster id="hercule">
      <host id="hercule-1">
        <vm mem="2048" hdd="4" id="vm-30" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-31" cpu="1"/>
      </host>
    </cluster>
    <cluster id="orion">
      <host id="orion-1">
        <vm mem="2048" hdd="4" id="vm-32" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-33" cpu="1"/>
      </host>
      <host id="orion-2">
        <vm mem="2048" hdd="4" id="vm-34" cpu="1"/>
        <vm mem="2048" hdd="4" id="vm-35" cpu="1"/>
      </host>
    </cluster>
  </site>
</vm5k>
```

2.4 Options

2.4.1 Execution

Manage how vm5k is executed

-h, --help show this help message and exit

-v, --verbose print debug messages

-q, --quiet print only warning and error messages

-o OUTDIR, --outdir OUTDIR where to store the vm5k log files default=vm5k_20140307_013045_+0100

-p PROGRAM, --program PROGRAM Launch a program at the end of the deployment

--plot draw a topological graph of the deployment

2.4.2 Mode

Define the mode of vm5k

-n N_VM, --n_vm N_VM number of virtual machines

-i INFILE, --infile INFILE XML file describing the placement of VMs on G5K sites and clusters

-j JOB_ID, --job_id JOB_ID use the hosts from a oargrid_job or a oar_job.

-w WALLTIME, --walltime WALLTIME duration of your reservation

-k, --kavlan Deploy the VMs in a KaVLAN

2.4.3 Physical hosts

Tune the physical hosts.

-r RESOURCES, --resources RESOURCES list of Grid'5000 elements

-b BLACKLISTED, --blacklisted BLACKLISTED list of Grid'5000 elements to be blacklisted

-e ENV_NAME, --env_name ENV_NAME Kadeploy environment name

-a ENV_FILE, --env_file ENV_FILE path to the Kadeploy environment file

--forcedeploy force the deployment of the hosts

--nodeploy consider that hosts are already deployed and configured

--host-packages HOST_PACKAGES comma separated list of packages to be installed on the hosts

2.4.4 Virtual machines

Tune the virtual machines.

-t VM_TEMPLATE, --vm_template VM_TEMPLATE XML string describing the virtual machine

-f VM_BACKING_FILE, --vm_backing_file VM_BACKING_FILE backing file for your virtual machines

-l VM_DISK_LOCATION, --vm_disk_location VM_DISK_LOCATION Where to create the qcow2: one (default) or all)

-d VM_DISTRIBUTION, --vm_distribution VM_DISTRIBUTION how to distribute the VMs distributed (default) or concentrated

--vm-clean-disks force to use a fresh copy of the vms backing_file

CHAPTER 3

vm5k_engine: automatizing experiments

An engine to perform automatic cloud experiments on Grid‘5000, based on execo_engine.

More information can be found on https://www.grid5000.fr/mediawiki/index.php/Vm5k_2014_School_tutorial

CHAPTER 4

API documentation

Vm5k can also been used in any other python project. Full documentation for modules can be found here.

4.1 vm5k.deployment

This module provides tools to deploy hosts and virtual machines on the Grid'5000 platform, using a preconfigured version of debian wheezy.

- a wheezy-x64-base environnement
- libvirt-bin and qemu-kvm from debian testing (jessie)
- a bridged networking for virtual machines

It needs an IP address range, either from g5k-subnets or kavlan to configure the VMs.

class vm5k.deployment.**vm5k_deployment**(*infile=None*, *resources=None*, *hosts=None*, *ip_mac=None*, *vlan=None*, *env_name=None*, *env_file=None*, *vms=None*, *distribution=None*, *outdir=None*)

Base class to control a deployment of hosts and virtual machines on Grid'5000. It helps to deploy a wheezy-x64-base environment, to install and configure libvirt from testing repository, and to deploy virtual machines.

The base run() method allows to setup automatically the hosts and virtual machines, using the value of the object.

configure_libvirt(*bridge='br0'*, *libvirt_conf=None*)

Enable a bridge if needed on the remote hosts, configure libvirt with a bridged network for the virtual machines, and restart service.

configure_service_node()

Setup automatically a DNS server to access virtual machines by id and also install a DHCP server if kavlan is used

deploy_vms(*clean_disks=False*, *disk_location='one'*, *apt_cacher=False*)
Destroy the existing VMS, create the virtual disks, install the vms start them and wait until they have rebooted

hosts_deployment(*max_tries=1*, *check_deploy=True*, *conf_ssh=True*)
Deploy the hosts using kadeploy, configure ssh for taktuk execution and launch backing file disk copy

packages_management(*upgrade=True*, *other_packages=None*, *launch_disk_copy=True*, *apt_cacher=False*)
Configure APT to use testing repository, perform upgrade and install required packages. Finally start kvm module

run()
Launch the deployment and configuration of hosts and virtual machines: hosts_deployment, packages_mamangement, configure_service_node configure_libvirt, deploy_vms

4.2 vm5k.actions

A set of functions to manipulate virtual machines on Grid'5000

This module provides tools to interact with the virtual machines.

4.2.1 VM definition and distribution

vm5k.actions.**show_vms**(*vms*)
Print a short resume of vms parameters.

Params vms a list containing a dict by virtual machine

vm5k.actions.**define_vms**(*vms_id*, *template=None*, *ip_mac=None*, *tap=None*, *state=None*, *host=None*, *n_cpu=None*, *cpusets=None*, *mem=None*, *hdd=None*, *backing_file=None*, *real_file=None*)
Create a list of virtual machines, where VM parameter is a dict similar to {'id': None, 'host': None, 'ip': None, 'mac': None, 'mem': 512, 'n_cpu': 1, 'cpuset': 'auto', 'hdd': 10, 'backing_file': '/tmp/vm-base.img', 'state': 'KO'}

Can be generated from a template or using user defined parameters that can be a single element or a list of element

Parameters

- **vms_id** – a list of string that will be used as vm id
- **template** – an XML element defining the template of the VM
- **ip_mac** – a list of tuple containing ip, mac correspondance
- **state** – the state of the VM
- **host** – the host of the VM (string)
- **n_cpu** – the number of virtual CPU of the VMs
- **real_file** – boolean to use a real file

vm5k.actions.**distribute_vms**(*vms*, *hosts*, *distribution='round-robin'*)
Distribute the virtual machines on the hosts.

Parameters

- **vms** – a list of VMs dicts which host key will be updated

- **hosts** – a list of hosts
- **distribution** – a string defining the distribution type: 'round-robin', 'concentrated', 'n_by_hosts', 'random

vm5k.actions.**list_vm**(*hosts*, *not_running=False*)
Return the list of VMs on hosts using a disk which keys are the hosts and value are list of VM id

4.2.2 VM state

vm5k.actions.**destroy_vms**(*hosts*, *undefine=False*)
Destroy all the VM on the hosts

vm5k.actions.**create_disks**(*vms*)
Return an action to create the disks for the VMs on the hosts

vm5k.actions.**create_disks_all_hosts**(*vms*, *hosts*)
Create a temporary file containing the vms disks creation commands upload it and run it on the hosts

vm5k.actions.**install_vms**(*vms*)
Return an action to install the VM on the hosts

vm5k.actions.**start_vms**(*vms*)
Return an action to start the VMs on the hosts

vm5k.actions.**wait_vms_have_started**(*vms*, *restart=True*)
Scan port 22 on all vms, distributed on hosts

vm5k.actions.**migrate_vm**(*vm*, *host*)
Migrate a VM to an host

vm5k.actions.**rm_qcow2_disks**(*hosts*)
Removing qcow2 disks located in /tmp

4.3 vm5k.config

Define a dict for default VM:

```
default_vm = {'id': None, 'host': None, 'ip': None, 'mac': None,
'mem': 512, 'n_cpu': 1, 'cpuset': 'auto',
'hdd': 10, 'backing_file': '/tmp/vm-base.img',
'state': 'KO'}
```

Create some new color_style.

4.4 vm5k.engine

class vm5k.engine.**vm5k_engine**
The base vm5k engine class, that is build from execo_engine.Engine and can be used to perform virtual machines experiments.

create_paramsweeper()
Generate an iterator over combination parameters

force_options()
Allow to override default options in derived engine

get_resources()
Retrieve the ressources for the vm5k_deployement and define the list of hosts and ip_mac.

make_reservation()
Perform a reservation of the required number of nodes, with 4000 IP.

setup_hosts()
Launch the vm5k_deployment

class vm5k.engine.**vm5k_engine_para**
A engine that use threads to treate combination in parallel

run()
The main experimental workflow, as described in `Using the Execo toolkit to perform ..
.`

CHAPTER 5

Publications

Matthieu Imbert, Laurent Pouilloux, Jonathan Rouzaud-Cornabas, Adrien Lèbre, Takahiro Hirofuchi
Using the EXECO toolbox to perform automatic and reproducible cloud experiments
1st International Workshop on UsiNg and building ClOud Testbeds UNICO, collocated with IEEE CloudCom 2013

Takahiro Hirofuchi, Adrien Lèbre, and Laurent Pouilloux
Adding a Live Migration Model Into SimGrid, One More Step Toward the Simulation of Infrastructure-as-a-Service Concerns
In 5th IEEE International Conference on Cloud Computing Technology and Science (IEEE CloudCom 2013), Bristol, United Kingdom, December 2013

Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec
Adding Virtualization Capabilities to Grid'5000
Cloud Computing and Services Science, vol 367, pp 3-20, Springer International Publishing, 2013

- genindex
- modindex
- search

Python Module Index

v

Index