vmgen:
Automatic generation of virtual machines

Alexandru Andrei[1], Mircea Urse[2]
[1] alex.f.andrei@gmail.com
[2] umirc3a@gmail.com

Abstract. This paper presents the development of a tool used for the automatic generation and configuration of virtual machines. The user requests a virtual machine with a specified, configured, operating system installed on it and with a list of applications to be installed. Then the virtual machine is generated and configured, and an archive containing the virtual machine is provided to the user. The user must provide the configuration file, written from scratch, or altered through the vmgCtl utility. In the future there might be a web interface to ease the generation of the configuration file.

Keywords: virtual machines, automatic generation, lxc, OpenVZ, VMware, VirtualBox

1. Introduction

Virtualization is one of the most interesting topics both in industry and academia and virtual machines are used extensively in numerous domains. However, the process of generating and configuring a virtual machine has remained time-consuming.

Vmgen is an application that automatizes this entire process, including the operating system install, hardware configuration, network configuration, user configuration and also applications install. This will reduce considerably the time required to fully generate and configure a virtual machine.

Also, vmgen is intended to be of use to users who are somewhat unfamiliar with how to generate of configure a virtual machine and can also be used to generate complex network topologies relatively easy.

The application will offer support for both operating system-layer virtualization (OpenVZ and lxc) and full virtualization (VMWare and VirtualBox).

2. Aplication architecture

2.1 Commander modules

The application is designed to be modular, as to be easily extended with support for other virtualization solutions. This means that a separate module was built for each of the current virtualization solutions (VMWare,VirtualBox, OpenVZ, lxc). Each of these modules can act as an independent application, but are required to implement a common interface (CommanderBase).

2.2 Installer modules

The same principle is applied to the component responsible for installing the applications. A separate module is built for each software manager (aptitude installer, yum installer, source installer, windows installer).

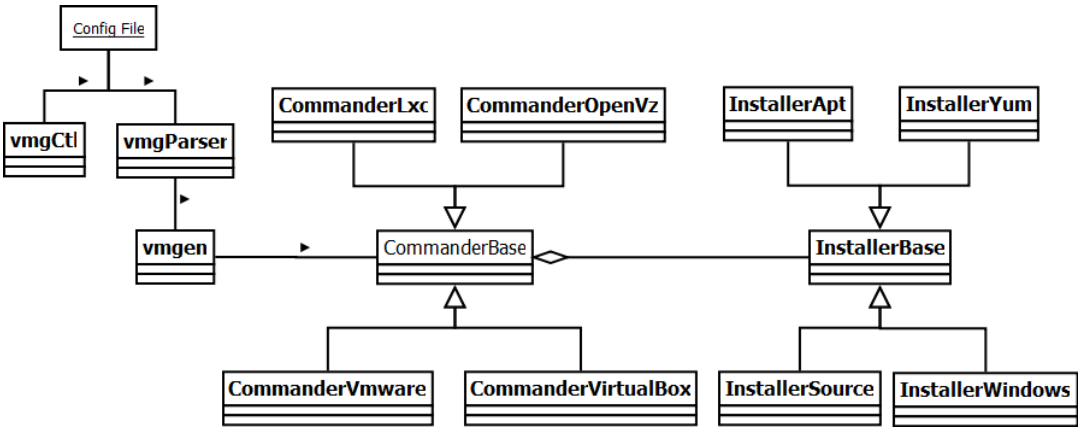

Fig. 1. The three main components of the application: vmgen - central module (*left*), commander modules (*center*) and installer modules (*right*)

2.3 vmgen module

These are all connected by the central module, vmgen, which is responsible for parsing the configuration file. The information from the configuration file is stored in a data structure, which is then serialized and dumped in a new file. This module is also responsible for instantiating the commander module and the installer module. These will use the data in the dump file to restore the data structure and use the information in their tasks. After the commander and the installer finish their tasks, the result is packed by the central module and ready to be sent to the user.

2.4 Configuration file

The configuration file uses the INI format to store the settings of the new virtual machine. The settings are grouped in sections and subsections for easier handling. The file can easily be modified using the vmgCtl tool which can add a new setting to a certain section or modify an existing one.

3. OpenVZ containers

OpenVZ is an operating system-level virtualization that runs over the Linux kernel. It does not have a native support in the kernel, unlike linux containers (lxc). That means additional packages are required in order to create and start an OpenVZ container. Each virtual environment consists of a file system isolated from the rest of the system (using chroot).

Although it does require additional packages, the task of creating a new container is simplified by the existence of pre-created containers (container templates), available for a significant number of Linux distributions.

3.1 Steps for generating a container

The OpenVZ module will download the pre-created container based on the Linux distribution specified and create a new container. The module will then configure the new container based on the user-specified requirements, using the OpenVZ API. Some of the settings which are configurable for an OpenVZ container include hardware configurations (the number of CPUs, the size of the memory available, the hard-disk limit), user configurations (new users/passwords), network configuration (interfaces, IP addresses, name servers). One of the advantages of using the specified API is that it does not require for the container to be running. All these settings will be saved in an OpenVZ configuration file and applied when the container starts.

The output of this module will be a tar archive, representing the container file system and the OpenVZ configuration file. At this stage, the container is functional and can be used immediately after the files are deployed in the corresponding OpenVZ directories.

3.2 Testing OpenVZ containers

OpenVZ containers were generated on a Fedora system and deployed on Fedora and Debian systems.

4. lxc containers generating

The lxc containers may be generated on a Debian distribution or on a Fedora distribution. After a container is created, it can be used on any distribution that has lxc support built-in.

4.1 Steps for generating a container

For generating the container, some bash scripts are used. There are scripts for each supported distribution (Debian or Fedora). To avoid damage to the physical system, and to be able to provide both the platforms simultaneously, two VMware virtual machines are used, one having Debian installed and another having Fedora installed, both with lxc support. The containers are created inside these machines and exported to the physical machine.

An lxc container consists of a configuration file, an optional fstab file, for the mount points, and a directory which contains the file system of the container.

A minimal base file system is downloaded, using debootstrap (for Debian), or febootstrap (for Fedora). Then, the configuration files inside the container are modified according to the user's request.

4.2 Testing the lxc container generating process

lxc containers were generated on both a Fedora system and a Debian system.

5 VMware virtual machines generating

The server stores base virtual disks for each supported operating system. Each virtual disk contains a basic installation of the correspondig operating system, without anything installed on it, besides VMware tools, for easier interaction with the virtual machine later on.

5.1 Steps for generating a virtual machine

The first step in generating a VMware machine is creating the .vmx file, which contains the hardware information about the machine. This file is in text format, and contains key-value associations, according to the user's requests. Then the requested disks are generated using the vmware-vdiskmanager utility provided by VMware, with the specified size and type (IDE, SCSI).

An auxiliary, preconfigured, machine (VMaster) is used to manipulate the newly created disks. The machine runs a minimal installation of Debian, with no graphical interface. After the new disks are created, these and the base disk corresponding to the desired operating system are attached to the VMaster. The VMaster is powered on, and the new disks are partitioned, as the user requested. Then, the system partition on the base disk is copied on the new system disk and the MBR of the new disk is updated, to make it bootable. After the system is copied, the VMaster is turned off and the disks are detached from it.

5.2 Testing the virtual machine generating process

The VMware disk generating was tested for the following systems:

- Debian 5 x86_64
- Ubuntu 10.10 x86_64
- Windows XP
- Windows 7

The newly generated machines worked fine when powered on.

6 Application installing

The user may request which applications to be installed in the newly created virtual machine. The list of applications to be installed is specified in the configuration file.

The list of available applications is predefined:

- Networking tools: traceroute, nmap, ...
- Development tools: emacs, eclipse, vim, valgrind, ...
- Services: httpd, sshd, git, ...
- GUI tools: browser, mail client, ...

6.1 Actual installation of applications

The selected applications will be installed using one of the defined installers, based on the selected operating system:

- apt for Debian, Ubuntu
- yum for Fedora
- source install for Gentoo
- Windows install for Windows XP, Windows 7

7 Current application status

- The parser and the commander interfaces are fully implemented.
- The VMware commander is implemented and tested.

7.1 More things to implement

- Create commanders for OpenVZ and lxc, to integrate the scripts used to generate containers into the main application.
- Implement the installer modules and add support for installing applications.
- Create a commander to add VirtualBox support, similar to the VMware one.
- Create a commander to add KVM virtual machines support.

7.2 Real life testing

- The application needs to be tested by people who request some different machines and give us feedback about how the process went, or what went wrong
- Suggestions of features that should be added to the application will be taken into account.
- The project page is at *http://ixlabs.cs.pub.ro/redmine/projects/vmgen/wiki*

References

1. Practical Virtualization Solutions - Virtualization from the Trenches, Kenneth Hess and Amy Newman
2. Advanced Server Virtualization, David Marshall, Wade Reynolds, Dave McCrory
3. VirtualBox 3.1: Beginner's Guide, Alfonso Romero
4. Operating System Concepts, 7th Edition, Avi Silberschatz, Peter Galvin, Greg Gagne