Science at work for Canada

National Research Council Canada

PortageLive

Patrick Paul

National Research Council Canada | Conseil national de recherches Canada

Canada

Table of contents

PortageLive

Design Document No : 001.DOC.0007.V02

Updated May 2010 for Portage 1.4

IIT-CNRC

Patrick Paul

Institute for Information Technology (IIT), National Research Council Canada (NRC)

Summary

This document describes how we package the PortageLive environment. This document could also be used to replicate the SOAP interface on a server that has the Portage or PORTAGEshared SMT system installed for easier integration in other applications.

Scope

The scope of this document is limited to describing how to build a PortageLive Virtual Appliance, and assumes you already have the software, the models and the web service rpms on hand. You should consult the PortageLive User/Trainer Manual first, for wider context and instructions for the steps that come before and after building the VA. That user manual is part of the Portage 1.4 User Manual, included on your CD in the doc/ folder.

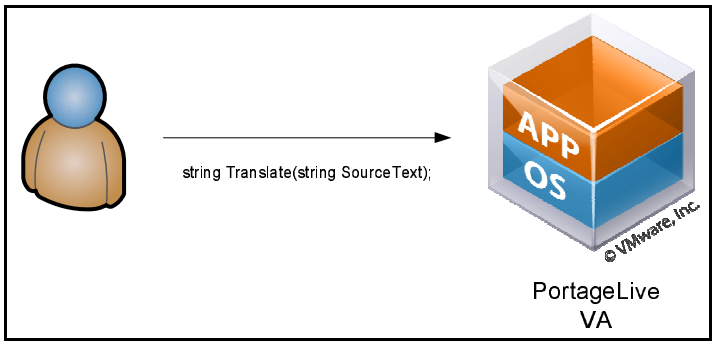
1 Introduction

The present document assumes the reader is familiar with basic Linux administration under CENTOS 5.3 or equivalent (CENTOS is largely inspired by RHEL, as is Scientific Linux). PortageLive is not limited to this single distribution, but the amount of customization could vary significantly between distributions. The current PortageLive system is built using VMWare Studio 2. VMWare Studio 2 offers a number of options to tailor how a virtual appliance (VA) is provisioned; the exact choices when doing a manual installation of the distribution may vary slightly.

The current PortageLive is a virtualized and self contained runtime installation of the Portage Statistical Machine Translation (SMT) system. It offers a rudimentary programmatic API in the form of a web service using SOAP.

Once deployed, any application using a language that supports SOAP can use the translation service offered by an instance of PortageLive (e.g., the .NET variant of all modern languages, php, java, perl, and python all support SOAP).

PLiveVA.doc/pdf 001.DOC.0007.V02 PortageLive 3 / 7



In the very early stages we had expected a much coarser mode of interaction with the virtual machine. Although this mode is still viable it is more cumbersome to put in place and requires more knowledge in order to start using it. The idea was to invoke commands remotely through SSH. With obscure configuration we had identified a way to bind a specific key to be able to run one and only one command on the virtual appliance and through regular Linux pipes, we could actually consider the translation engine as an inline process. This approach had the benefit of being native (with respect to the standard way Portage runs) but required the client application developer to find a way to call SSH through a system call or finding an SSH implementation in the particular language of choice.

In the end the SOAP/.NET approach seems to be by far the most compelling option.

We also have a CGI script to connect to PortageLive via a web page, for cases where integration with the client application is not required.

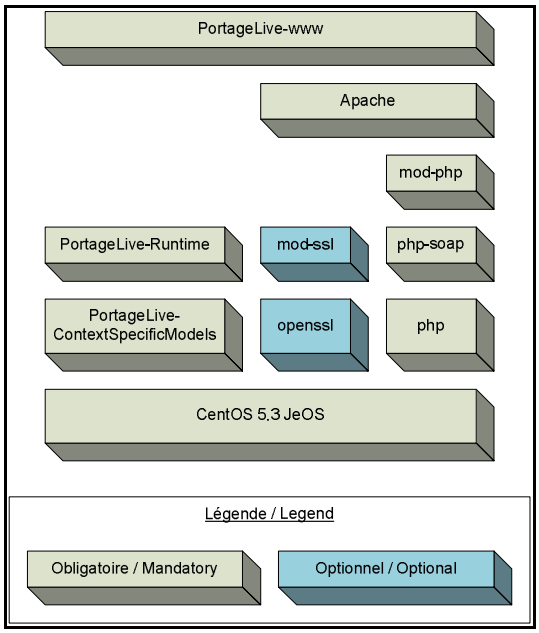
2 Anatomy of the VA

By default VMWare studio 2 selects a minimal set of packages that it considers to be essential. In building virtual appliances we refer to this minimal set of packages as Just Enough OS (JeOS, pronounced juice). A neat feature of this packaging method is the automatic dependency resolution. When one adds extra packages, the list of packages to install is augmented to satisfy all the dependencies. Adding the following packages to the JeOS selection made by VMWare studio 2, we obtain a minimal system with only the bits and pieces we need for PortageLive:

- mod_ssl
- php
- php-soap
- PortageLive-bin (runtime environment)
- PortageLive-models-*context* (models defining the context of the runtime environment)
- PortageLive-www (the components that allows the SOAP xml over http remote procedure call functionality)

The three PortaveLive rpms are created from the Portage software and the models you have trained. When you deploy many VAs, you only need to change the models-*context* rpm, where *context* should be a label specific to each particular context your train. The other two rpms remain the same for all systems. The PortageLive rpms can be built using VMWare studio, among other tools, as described in the PortageLive User Manual.

Here is a block diagram showing the different packages in the proposed implementation:



3 Post-install script

The following commands are run as a post-install script, to perform additional actions not done via rpms. They are meant to be inserted in the Post-install-script box in the VMWare Studio VM creation interface, without the "$" in front of each line.

3.1 Setting the timezone

$ ln -sf /usr/share/zoneinfo/America/Montreal /etc/localtime

3.2 Customize the user's "portage" environment

$ echo 'export PORTAGE=/opt/Portage' >> /home/portage/.bashrc

$ echo 'test -f $PORTAGE/SETUP.bash && source $PORTAGE/SETUP.bash' >> /home/portage/.bashrc

3.3 Pre-populate the user's "portage" with a public key

The following inserts the NRC public key into the virtual machine, matching our own private key. Use your own DSS key instead of the one shown here if you want to enable automated ssh access to the virtual appliance. Note that you don't need this if you're planning to access the translation server via SOAP or CGI.

$ mkdir /home/portage/.ssh

$ chown portage.portage /home/portage/.ssh

$ chmod 700 /home/portage/.ssh

$ echo 'ssh-dss
AAAAB3NzaC1kc3MAAACBAMQEpC/w6FAupe6xyjcPZKSQhFpX+B1otOmN5j6M4HaXbVYO53vWF4oo
qKoh9aJZEpNkaIsyGi3b+qOii1jQsiB1pqiyLImHEu22YILEw6968BZ4WMzJMiePJv+XopHVXmMUzXYwZ
37qMaNOSZqf6i1awrCXBeO1hrKDyqrS68aDAAAAFQDD6cztMWAI5DeKD4DRbWfd2+RsXQAAAIB5z8
Um04nm1R8+bPBmcWyM1nPkv6tggYveTUu9woi8HJCsUFt35RDhvOmsM3oFFRBbZhOMaTa2PQwo3e
bUKXrQJfosBQ6O2ft5q+sLQhmEcufRSutfJezIoNR1O+fLSR5Eog1nRID3yltUmhJnk06hPvWKRTLPHf/wz
taAC258DAAAAIEArmFX1drbEnJV1pyJnDDPx7MNRdloHxAV/vqIBenbX8UfQhi3ObnZuTIq3+6fgqRc0nV
fxhYB4e9T+gAB2SHCmh1IQ+yN6MAZegRxxP1/mYTB2d67Xg4+l4oD0acFoEqZuJYDGosmicRfEkxUJe
FxH8zinpAExgUtIxR4UjmmCRw= PortageLive_key' > /home/portage/.ssh/authorized_keys

$ chown -R portage.portage /home/portage/.ssh/

$ chmod 600 /home/portage/.ssh/authorized_keys

3.4 Allow php to be interpreted if present in an html document

$ echo 'AddType application/x-httpd-php .html' >> /etc/httpd/conf.d/php.conf

3.5 Make sure apache runs

$ chkconfig httpd on

4 Preparing for the first boot

VMWare Studio also has a box for a script to be run at the first boot of the VA. This is a good place to provide initial information about the VA. It will be executed only once at the initial boot of the virtual appliance. Anything that could depend on the IP of the VA should probably be done here.

4.1 Allowing SSL based web serving

$ sed -i 's/#DocumentRoot "\/var\/www\/html"/DocumentRoot "\/var\/www\/html\/secure"/'
/etc/httpd/conf.d/ssl.conf

4.2 The WSDL file needs to be tailored to have the hostname or IP of this machine

$ sed "s/__REPLACE_THIS_WITH_YOUR_IP__/`ifconfig | grep \"inet addr\" | head -1 | awk {'print $2'} |
cut -d \":\" -f2`/" /opt/Portage/www/PortageLiveAPI.wsdl > /var/www/html/PortageLiveAPI.wsdl

4.3 Two (2) files are required to start serving soap requests

$ ln -s /opt/Portage/www/index.html /var/www/html/index.html

$ ln -s /opt/Portage/www/PortageLiveAPI.php /var/www/html/PortageLiveAPI.php

4.4 Creating the content available through the SSL variant

$ mkdir /var/www/html/secure

$ sed "s/__REPLACE_THIS_WITH_YOUR_IP__/`ifconfig | grep \"inet addr\" | head -1 | awk {'print $2'} |
cut -d \":\" -f2`/" /opt/Portage/www/secure/PortageLiveAPI.wsdl >
/var/www/html/secure/PortageLiveAPI.wsdl

$ ln -s /opt/Portage/www/secure/index.html /var/www/html/secure/index.html

$ ln -s /opt/Portage/www/secure/PortageLiveAPI.php /var/www/html/secure/PortageLiveAPI.php

4.5 Provide the initial password for the “portage” account if login is required

These three echo commands are intended to display login credentials at the console at the first boot, for the convenience of the system administrator. Leave them out if you consider this a security risk. These lines don’t create the account – that’s done elsewhere in the VMWare Studio interface.

$ echo Username: portage

$ echo Password: *enter the portage password here*

$ echo Phonetics: *phonetic encoding*

5 Preparing for subsequent boots

Since the VA could change location on the network some elements of configuration done in the first boot are actually repeated at every boot of the VA. Again, these commands go into the appropriate box in the VMWare Studio VM creation interface.

5.1 Replace a placeholder with the current IP of the VA

$ sed "s/__REPLACE_THIS_WITH_YOUR_IP__/`ifconfig | grep \"inet addr\" | head -1 | awk {'print $2'} | cut -d \":\" -f2`/" /opt/Portage/www/PortageLiveAPI.wsdl > /var/www/html/PortageLiveAPI.wsdl

$ sed "s/__REPLACE_THIS_WITH_YOUR_IP__/`ifconfig | grep \"inet addr\" | head -1 | awk {'print $2'} | cut -d \":\" -f2`/" /opt/Portage/www/secure/PortageLiveAPI.wsdl > /var/www/html/secure/PortageLiveAPI.wsdl

6 Disclaimer

The current state of PortageLive has some limitations. It was develop as proof of concept and most of the coordination work would be done at a higher layer in the software stack either by marshaling the requests or by simply automating some corrective actions.

- No provision for concurrent access
    - Although not a strict limit, it is understood that on large load a certain amount of swapping will inevitably occur limiting the capacity of the VM, which could in some circumstances require manual intervention to correct the situation (e.g., manual reboot of the VM).
- No connection broker
    - The global design would require a brokering mechanism dispatching requests to multiple servers, launching new ones when the load warrants it, and performing general load management. Such facilities don’t exist.