

Virtual EasyVoxBox

A Virtual Hosted IP PBX
Server Implementation using
the Asterisk Toolkit.

Mark Barry, Astricon (Sept. 2007)

EasyVoxBox.org

EasyVoxBox

Introduction

- Exabar Phone Systems, Austin Texas
- PBXProducts.com
- Exabar.com
- NameBUB.com
- EasyVoxBox.org
- AllSipDial.com
- The latest version of this presentation and references can be found at:

<http://easyvoxbox.org>

About EasyVoxBox

- A full featured IP PBX phone system. EasyVoxBox (EVB) is based on Asterisk and FreePBX and installs a complete system quick and easy, from a CD ISO. It gives full control and provides additional system utilities via the web based user interface.
- Baseline installation
- Multi level access control
- Builds and compiles from source code.
- Asterisk 1.4.11
- FreePBX 2.3
- CentOS 5

Overview

- The base EasyVoxBox system is built using Asterisk and FreePBX. The auto install provides a consistent baseline for each instance.
- OpenVZ is used for the virtual Linux operating system.
- EasyVoxBox is installed on the virtual instance.
- Multiple templates can be built for quick virtual installations.
- Supports the Zaptel kernel device drivers for IAX2 channel, Meetme conferences and shared hardware.

Hosted IP PBX Characteristics

- PSTN service
- Sip Trunking service.
- Endpoints, handsets and provisioning.
- Remote offices
- Customer connection to PBX, private vs public internet.
- Billing

Virtual Hosted PBX

- ITSP/ASP Service Model
- Open source solution with commercial support available.
- Provides cost effective high quality and manageable system.
- Provides a baseline to add more services.

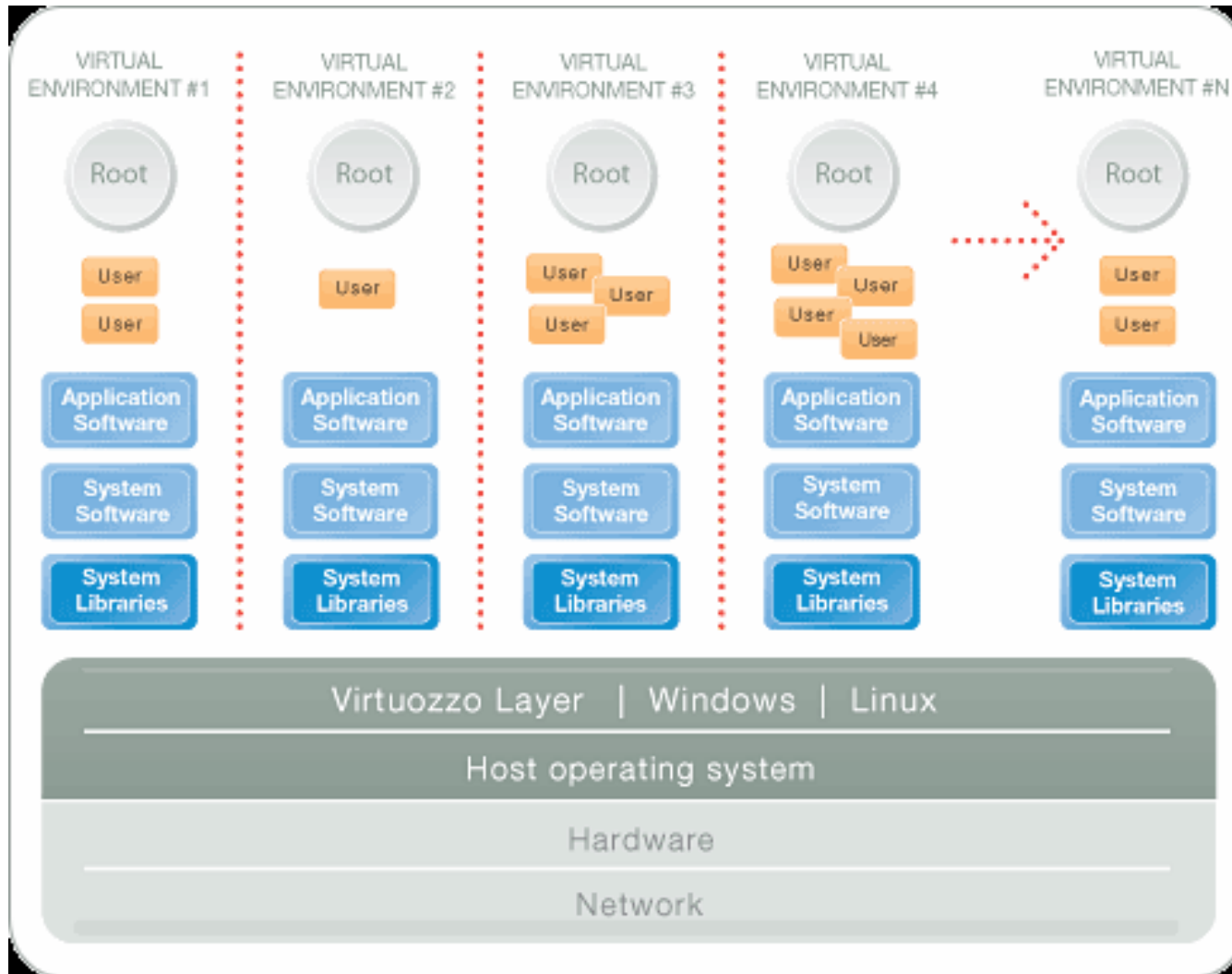
OpenVZ Virtual Linux

OpenVZ is an Operating System-level server virtualization solution, built on Linux.

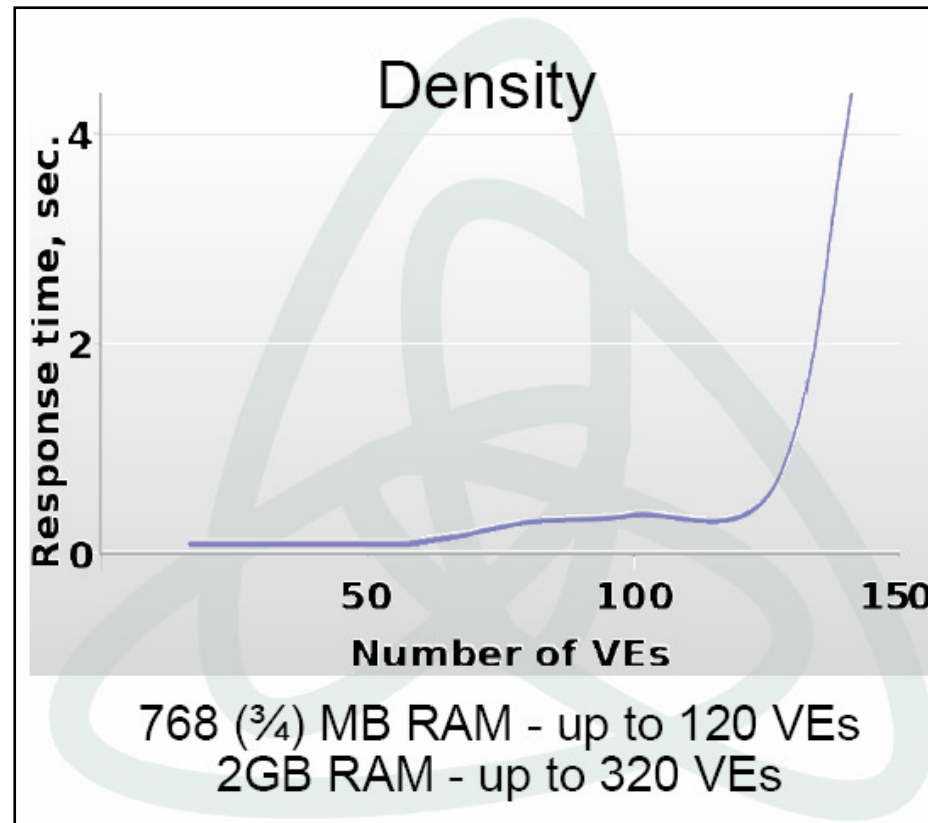
OpenVZ creates isolated, secure virtual environments — VEs (otherwise known as virtual private servers, or VPSs) on a single physical server enabling better server utilization and ensuring that applications do not conflict. The host kernel operating system that the VEs plug into is called the hardware node.

Each VE performs and executes exactly like a stand-alone server; VEs can be rebooted independently and have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files.

Virtual Environment



VPS Density



Exabar has been running 15 EasyVoxBox VEs on a Pentium 4, 3GHZ, and 1.5 MB of RAM. It has performed well and without any problems.

OpenVZ Template Terminology

- **OS Template** - A set of packages from some Linux distribution used to populate a VE.
- **OS Template Metadata** – Contains a list of files that form the OS Template, locations of the package repositories, scripts and additional OpenVZ packages.
- **OS Template Cache** - An OS Template installed into a VE and then packed into a gzipped tarball. Using such a cache, a new VE can be created in a matter of minutes.

OpenVZ Tool Commands

- **vzpkgcache**
 - vzpkgcache - create a tarball (cache) of VPS private area
- **vzyum**
 - yum wrapper for use with OpenVZ's VPS
- **vzctl**
 - utility to control a Virtual Environment.
- **vps.conf**
 - configuration file for a Virtual Environment.
- **vz**
 - global OpenVZ configuration file
- **Vzlist**
 - VE listing utility
- **vzmemcheck**
 - utility to display information of node memory parameters.
- **vzrpm**
 - rpm wrapper for use with OpenVZ's VPS

OpenVZ Tool Commands

- **vzsplit**
 - generate a sample VE configuration file with a certain set of system resource control parameters.
- **vzcfgvalidate**
 - validate a VE configuration file
- **arp send**
 - utility to send ARP requests.
- **vzpid**
 - display the VE ID given the process ID (PID).
- **vzcalc**
 - utility to calculate the VE resources usage.
- **vzcpucheck**
 - shows information about the CPU power and utilization.
- **vzdqcheck**
 - counts disk usage
- **vzdqdump**
 - dump user/group quotas
- **vzdqload**
 - load user/group quotas
- **vzquota**
 - manipulate VPS disk quotas
- **vzmigrate**
 - utility for virtual environment migration between hardware nodes.
- **vz.conf**
 - global OpenVZ configuration file

Build Process (Step 1)

1. Install the base Linux operating system.
 - Current version is based on Fedora Core 6.
 - Use the Linux kickstart file from EVB. This will provide the packages needed to run the EVB system.

Build Process (Step 2)

2. Load and setup the OpenVZ system
 - Follow the quick installation instructions on the OpenVZ wiki site.
 - Use the yum method and make sure that you install the ovzkernel-devel package.
 - Follow the template cache preparation section and install all of the utilities.
 - Install the template metadata for FC6
 - Do not install any pre-built templates.

Build Process (Step 3)

3. Download EVB and install Zaptel.
 - Untar the Zaptel distribution and compile.
 - Make; make install; make configure
 - Reboot and make sure that Zaptel starts at boot time.
 - Service zaptel start

Build Process (Step 4)

4. Build the first minimal VE template
 - Run this command to create the template cache:
`vzpkgcache fedora-core-6-i386-minimal`
 - You must have connection to the internet for this operation to gather the packages from the yum repository.
 - Set the configuration in the VE.conf file for network, devices and resource allocations.
 - Once this is complete, create and start a new VE by using the vzctl command.

Build Process (Step 5)

5. Build the EasyVoxBox VE template cache

- At this point, there should be a running VE. Check with `vzlist` and `vzctl enter`.
- Take the original package list from the kickstart file used to build the hardware node and use the `vzyum` command to install the packages in the running VE.
- Once this is complete, enter into the VE and download the EVB distribution.
- The libtonezone libraries must be installed in the VE. This can be copied from the hardware node.
- A modified version of EVB (minus Zaptel and a few other packages) is installed.

Build Process (Step 5 cont')

- Bring up the EVB web interface and configure the system to what you want included in the template.
- Once you have the EVB VE setup, shut down the VE:
`vzctl VE stop`
- You are now ready to create the EVB VE template
cache: `cd /vz/private/VE`
`tar cvzf /vz/template/cache/fedora-core-6-i386-EVB1.tar.gz .`
- Repeat the step 5 process for as many templates as you want to build (EVB2, EVB3, etc ...)

Build Process (Step 6)

6. Create multiple VEs and setup configurations.

- Vzctl create VE –ostemplate fedora-core-6-i386-EVB1
- Set resource limits in VE.conf. This is where you must configure at least the Zaptel pseudo device. Ex. (DEVNODES="tty9:rw zap/1:rw zap/channel:rw zap/ctl:rw zap/pseudo:rw zap/timer:rw")
- Save any template cache for future use.

Wrap up

- Once the EVB template cache is created, it can be reused and steps 4 & 5 can be omitted.
- Can be automated to a high level through shell scripts to provision the system.
- Virtual EasyVoxBox performs well in the production environment.