OpenVZ

Tipos de Virtualización

- **Emulación o simulación**: la máquina virtual simula un hardware completo, admitiendo un sistema operativo "**guest**" sin modificar para una CPU completamente diferente. Este enfoque fue muy utilizado para permitir la creación de software para nuevos procesadores antes de que estuvieran físicamente disponibles. Por ejemplo **Bochs**, **PearPC**, **Qemu** sin aceleración, y el emulador Hercules. La emulación es puesta en práctica utilizando una variedad de técnicas, desde state machines hasta el uso de la recopilación dinámica en una completa plataforma virtual.

- **Virtualización nativa** y **virtualización completa**: la máquina virtual simula un hardware suficiente para permitir un sistema operativo "**guest**" sin modificar (uno diseñado para la misma CPU) para correr de forma aislada. Típicamente, muchas instancias pueden correr al mismo tiempo. Este enfoque fue el pionero en 1966 con CP-40 y CP[-67]/CMS, predecesores de la familia de máquinas virtuales de IBM. Algunos ejemplos: **VMware Workstation**, **VMware Server**, **Parallels Desktop**, **Adeos**, **Mac-on-Linux**, **Win4BSD**, **Win4Lin Pro** y **z/VM**.

- **Virtualización parcial** (y aquí incluimos el llamado "address space virtualization"): la máquina virtual simula múltiples instancias de mucho (pero no de todo) del entorno subyacente del hardware, particularmente address spaces. Este entorno admite compartir recursos y aislar procesos, pero no permite instancias separadas de sistemas operativos "guest". Aunque no es vista como dentro de la categoría de máquina virtual, históricamente éste fue un importante acercamiento, y fue usado en sistemas como CTSS, el experimental IBM M44/44X, y podría decirse que en sistemas como OS/VS1, OS/VS2 y MVS.

- **Paravirtualización**: la máquina virtual no necesariamente simula un hardware, en cambio ofrece una API especial que solo puede usarse mediante la modificación del sistema operativo "guest". La llamada del sistema al hypervisor tiene el nombre de "hypercall" en **Xen** y **Parallels Workstation**; está implementada vía el hardware instruction DIAG ("diagnose") en el CMS de VM en el caso de IBM (este fue el origen del término hypervisor). Ejemplo: **VMware ESX Server**, **Win4Lin 9x** y **z/VM**.

- **Virtualización a nivel del sistema operativo**: virtualizar un servidor físico a nivel del sistema operativo permitiendo múltiples servidores virtuales aislados y seguros correr en un solo servidor físico. El entorno del sistema operativo "guest" comparte el mismo sistema operativo que el del sistema "host" (el mismo kernel del sistema operativo es usado para implementar el entorno del "guest"). Las aplicaciones que corren en un entorno "guest" dado lo ven como un sistema autónomo. Ejemplos: **Linux-VServer**, **Virtuozzo**, **OpenVZ**, **Solaris Containers** y **FreeBSD Jails**.

- **Virtualización de aplicaciones**: consiste en el hecho de correr una desktop o una aplicación de server localmente, usando los recursos locales, en una máquina virtual apropiada. Esto contrasta con correr la aplicación como un software local convencional (software que fueron "instalados" en el sistema). Semejantes aplicaciones virtuales corren en un pequeño entorno virtual que contienen los componentes necesarios para

ejecutar, como entradas de registros, archivos, entornos variables, elementos de uso de interfaces y objetos globales. Este entorno virtual actúa como una capa entre la aplicación y el sistema operativo, y elimina los conflictos entre aplicaciones y entre las aplicaciones y el sistema operativo. Los ejemplos incluyen el Java Virtual Machine de Sun, Softricity, Thinstall, Altiris y Trigence (esta metodología de virtualización el claramente diferente a las anteriores; solo una pequeña línea divisoria los separa de entornos de máquinas virtuales como Smalltalk, FORTH, Tel, P-code).

¿ Qué es OpenVZ ?

OpenVZ es una tecnología de virtualización en el nivel de sistema operativo para GNU/Linux. OpenVZ permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (SPV) o Entornos Virtuales (EV).

OpenVZ consiste del núcleo y de herramientas en el nivel de usuario.

Desventajas

* Ofrece menor flexibilidad en la elección del sistema operativo: tanto los huéspedes como los anfitriones deben ser GNU/Linux.

Ventajas

- La virtualización a nivel de sistema operativo de OpenVZ proporciona mejor rendimiento.
- Escalabilidad.
- Administración de recursos dinámicos.
- Densidad : es decir con poca memoria RAM podemos ejecutar muchas EV.
- Facilidad de administración que las alternativas.

Escenarios de uso

- **Seguridad** : Se pone cada servicio de red como (apache, servidor de correo, dns, etc). Siendo un entorno virtual separado.

- **Consolidación de Servidores** : Usando OpenVZ los servidores se pueden consolidar migrándolos a Entornos Virtuales. Ahorrando espacio en racks, consumo de electricidad, y esfuerzo de administración.

- **Hosting**.

- **Desarrollo y Pruebas**

- **Educación**

Instalación

Instalación de OpenVZ en Debian para 64 bits:

```
# apt-get install vzdump vzctl vzquota linux-image-2.6.32-5-openvz-amd64
```

Instalación de OpenVZ en Debian para 32 bits:

```
# apt-get install vzdump vzctl vzquota linux-image-2.6.32-5-openvz-686
```

Luego esto se instalara en el GRUB necesitamos reiniciar y seleccionar el nuevo kernel instalado.

Verificamos que la instalación este correcta

Validamos que realmente tengamos el kernel de openvz.

```
$ uname -r
        2.6.32-5-openvz-amd64
```

Y la placa que nos genero

```
$ ifconfig

        …
        venet0 Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
        UP BROADCAST POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
        …
```

¿ Qué son los templates/planillas ?

Son distribuciones mínimas comprimidas como **.tar.gz** que contiene lo mínimo de una distro. Como **Centos, Fedora, Suse, ArchLinux, Debian, Ubuntu, Slackware, etc.**
Estas van de los 150MB a los 300MB, y no hay que descomprimirlos hay que dejarlos tal como los bajamos.

Administrando OpenVZ

- Creando una VE
  - *vzctl create ID --ostemplate Planilla*
    $ vzctl create 100 –ostemplate slackware-13.0-i386-minimal

    La plantilla **slackware-13.37-i386-minimal** ya tiene que existir.

    ```
    minimal
    Creating VE private area (slackware-13.37-i386-minimal)
    Performing postcreate actions
    VE private area was created
    ```

Al crear una VE esto es lo que pasa :

1) Toma los valores del archivo /etc/vz/vz.conf

```
…
LOGFILE=/var/log/vzctl.log
…
TEMPLATE=/var/lib/vz/template
…
VE_ROOT=/var/lib/vz/root/$VEID
VE_PRIVATE=/var/lib/vz/private/$VEID
…
```

Donde en TEMPLATE/cache tendremos nuestros archivo bajados.
VE_ROOT y VE_PRIVATE esta descomprimido nuestra mini distro.

2) Se crea el archivo 100.conf en /etc/vz/conf

$ cat /etc/vz/conf/100.conf

```
ONBOOT="yes"

# UBC parameters (in form of barrier:limit)
KMEMSIZE="11055923:11377049"
LOCKEDPAGES="256:256"
PRIVVMPAGES="65536:69632"
SHMPAGES="21504:21504"

NUMPROC="240:240"
PHYSPAGES="0:2147483647"
VMGUARPAGES="33792:2147483647"
OOMGUARPAGES="26112:2147483647"
NUMTCPSOCK="360:360"
NUMFLOCK="188:206"
NUMPTY="16:16"
NUMSIGINFO="256:256"
TCPSNDBUF="1720320:2703360"
TCPRCVBUF="1720320:2703360"
OTHERSOCKBUF="1126080:2097152"
DGRAMRCVBUF="262144:262144"
NUMOTHERSOCK="360:360"
DCACHESIZE="3409920:3624960"
NUMFILE="9312:9312"
AVNUMPROC="180:180"
NUMIPTENT="128:128"

# Disk quota parameters (in form of softlimit:hardlimit)
DISKSPACE="1048576:1153024"
DISKINODES="200000:220000"
QUOTATIME="0"

# CPU fair sheduler parameter
CPUUNITS="1000"

VE_ROOT="/var/lib/vz/root/$VEID"
VE_PRIVATE="/var/lib/vz/private/$VEID"
OSTEMPLATE="slackware-13.0-i386-minimal"
ORIGIN_SAMPLE="vps.basic"
HOSTNAME="slackware"
```

- Para listar las maquinas creadas

  $ vzlist -a

  ```
  VEID      NPROC STATUS  IP_ADDR         HOSTNAME
  100         - stopped -               -
  ```

100 = es el id que dimos de alta.

Por defecto cuando creamos la maquina virtual esta no levanta.

- Para levantar la maquina:

  $ vzctl start 100

  ```
  Starting VE ...
  VE is mounted
  Setting CPU units: 1000
  ```

```
Configure meminfo: 65536
VE start in progress...
```

```
$ vzlist -a
```

```
VEID      NPROC STATUS  IP_ADDR         HOSTNAME
 100          5 running -               -
```

- Para ejecutar comando en la maquina virtual realizamos lo siguiente:

```
$ vzctl exec 100 ls -l
```

- Para entrar en la maquina virtual:

```
$ vzctl enter 100
```

Para salir escribimos solo **exit**

- Para ver los recursos que utiliza la maquina viertual hacemos lo siguiente:

```
$ cat /proc/bc/100/resources
```

- Para bajar la maquina virtual

```
$ vzctl stop 100
```

```
Stopping VE ...
VE was stopped
VE is unmounted
```

- Para restaurar la maquina virtual

```
$ vzctrl restart 100
```

- Para destruir el ID

```
$ vzctl destroy 100
```

Seteando ciertos parametros de la maquina virtual

* Si queremos cambiar el nombre del host de la maquina virtual :

```
$ vzctl set 100 --hostname slackware --save
```

Cada cambio lo tengo que grabar. Esta información me lo graba en **/etc/vz/conf** y el numero de ID (100) cada vez que creo una maquina virtual me lo graba ahí.

Ej:

```
$ cat /etc/vz/conf/100.conf
```

```
...
HOSTNAME="slackware"
```

* Si queremos asignarle una ip

```
$ vzctl set 100 --ipadd 192.168.1.200 --save
```

* Asignarle al usuario **root** la password

```
$ vzctrl set 100 --userpasswd root:mipasswd --save
```

* Si queremos asignarle 5GB de espacio en disco

```
$ vzctl set 100 --diskspace 5G --save
```

* Si queremos asignarle 4GB de espacio memoria

```
$ vzctl set 100 --privvmpages 4G --save
```

Usando OpenVZ :

Tenemos el archivo de configuracion en **/etc/vz/vz.conf**

```
## Global parameters
VIRTUOZZO=yes
LOCKDIR=/var/lib/vz/lock
DUMPDIR=/var/lib/vz/dump
VE0CPUUNITS=1000

## Logging parameters
LOGGING=yes
LOGFILE=/var/log/vzctl.log
LOG_LEVEL=0
VERBOSE=0

## Disk quota parameters
DISK_QUOTA=yes
VZFASTBOOT=no

# The name of the device whose ip address will be used as source ip for VE.
# By default automatically assigned.
#VE_ROUTE_SRC_DEV="eth0"


# Controls which interfaces to send ARP requests and modify APR tables on.
NEIGHBOUR_DEVS=detect

## Template parameters
TEMPLATE=/var/lib/vz/template

## Defaults for VEs
VE_ROOT=/var/lib/vz/root/$VEID
VE_PRIVATE=/var/lib/vz/private/$VEID
CONFIGFILE="vps.basic"
#DEF_OSTEMPLATE="fedora-core-4"
DEF_OSTEMPLATE="debian"

## Load vzwdog module
VZWDOG="no"

## IPv4 iptables kernel modules
IPTABLES="ipt_REJECT ipt_tos ipt_limit ipt_multiport iptable_filter iptable_mangle ipt_TCPMSS
ipt_tcpmss ipt_ttl ipt_length"

## Enable IPv6
IPV6="no"
```

```
## IPv6 ip6tables kernel modules
IP6TABLES="ip6_tables ip6table_filter ip6table_mangle ip6t_REJECT"
```

DISK_QUOTA=yes si está en **yes** veremos el espacio asignaod de disco de cada una de las virtuales, sino tomara del total del disco.

Donde **TEMPLATE=/var/lib/vz/template** contiene el path donde tenemos nuestras maquinas esto lo podemos modificar.

Ej:

```
$ mkdir -p /virtuales/template/cache

TEMPLATE=/virtuales/template
```

Siempre tenemos que tener el directorio **cache** y ahí adentro nuestras maquinas virtuales.

En **VE_PRIVATE=/var/lib/vz/private/$VEID** es donde crea el directorio por ejemplo 100 (VEID) y donde estará descomprimido el template :

```
$ ls -l /var/lib/vz/private/100
        /bin
        /boot
        /dev
        …
```

Todo el sistema de la distribución.

Montaje de un disco

Primero montamos en nuestro equipo el filesystem para luego compartirlo :

```
$ mkdir /samba
$ mount /dev/sda5 /samba
```

Creamos un archivo llamado **100.mount**

```
$ vi /etc/vz/conf/100.mount

    #!/bin/bash
    source /etc/vz/vz.conf
    source ${VE_CONFFILE}
    mount -n --bind /samba ${VE_ROOT}/samba
```

Realizando copia de nuestro sistema

Es conveniente tener un filesystem de tipo LVM para poner las virtuales asi al realizar un copia y teniendo un LVM nos crea un nuevo filesystem temporal sacando una foto de la virtual y asi no bajar la virtual lo hace en caliente la copia.

```
$ vzdump --size 10000 --compress --dumpdir /dump --snapshot 100
```

Esto realiza una copia de la virtual con **ID 100,** lo guarda en el directorio **/dump** comprimido y crea un **lvm snapshot** de **10GB**.

¿ Qué es Webvz ?

Es una herramienta por web que me permite administrar **OpenVZ** permitiendo entre otras cosas controlar/configurar nuestras maquinas virtuales.

Instalación de la herramienta Webvz

Bajamos de la siguiente pagina : **http://webvz.sourceforge.net/download.html.** Esta herramienta esta escrita con el **framework** en el lenguaje de programación **Ruby on Rails** basada en **SQLite3** que no necesita otra instalacion de alguna base de datos distinta.

Y usa el servidor **Webrick** el propio servidor HTTP de Ruby.

1) Bajamos Webvz de la siguiente pagina :

   http://webvz.sourceforge.net/download.html

2) Instalamos los paquetes necesarios como root :

   $ apt-get install ruby rubygems libsqlite3-ruby sqlite3 irb1.8 libopenssl-ruby1.8 libreadline-ruby1.8 rdoc1.8

Luego necesitamos la version Rails 2.1.0 para bajarlo realizamos lo siguiente :

   $ gem install -v=2.1.0 rails

3) Descomprimimos el archivo Webvz.

   $ tar xvfz webvz.2.0.tar.gz

4) Creamos los enlaces de los directorios.

   $ ln -sf /var/lib/vz /vz

5) Corremos el servidor.

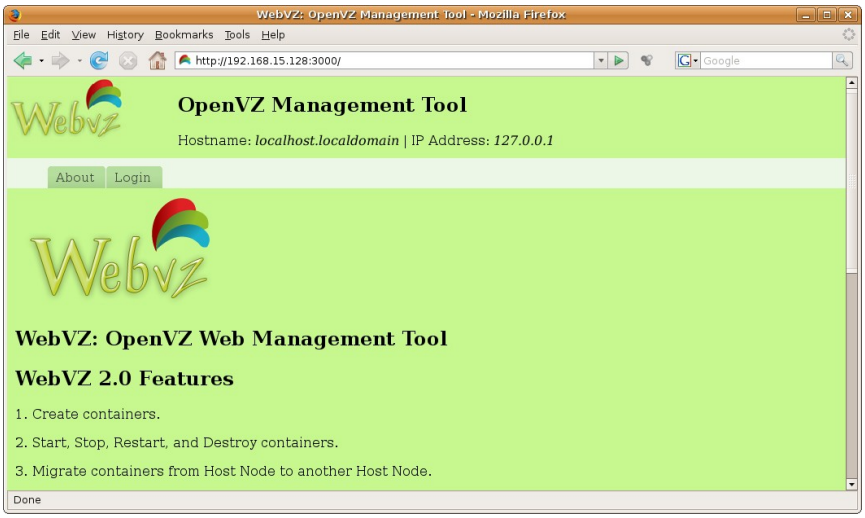
   $ cd webvz.2.0

Esto escuchara en el puerto por defecto 3000.

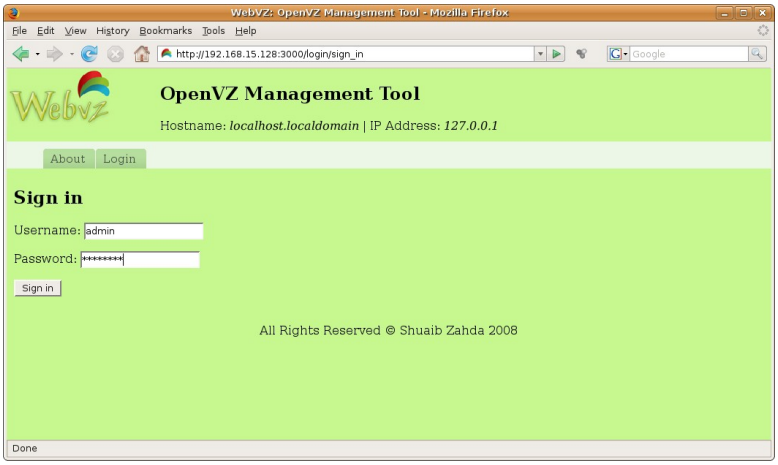
   $ ruby1.8 scripts/server

Si queremos otro puerto.

   $ ruby1.8 scripts/server --port 8100



USUARIO : admin
PASSWORD : admin123



Paginas

Página oficial: http://openvz.org/

Descarga de LiveCDs: http://openvz.org/download/livecd/

Descarga Kernel: http://openvz.org/download/kernel/

Descarga Plantillas/Templates: http://openvz.org/download/template/

Manual Online de comandos: http://openvz.org/documentation/mans/