



Manual de referencia para el diseño y desarrollo del sistema operativo Canaima GNU/Linux y derivados

VE2-0002-1002-340-09-04-0001.05

Caracas, Junio 2009

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Ficha catalográfica

ONUVA, Integración de Sistemas

Manual de referencia para el diseño y desarrollo del sistema operativo Canaima GNU/Linux y derivados . / ONUVA, Integración de Sistemas; Centro Nacional de Tecnologías de Información; José Miguel Parrella. – Caracas: ONUVA, 2009

Manual

1. Sistema operativo - Debian. 2. Sistema operativo - Canaima. I. Parrella, José. II. Manual de referencia para el diseño y desarrollo del sistema operativo Canaima GNU/Linux y derivados. III. Centro Nacional de Tecnologías de Información.

Control de versiones		
<i>Fecha</i>	<i>Responsable</i>	<i>Descripción del cambio</i>
29 de diciembre de 2008	José Miguel Parrella Romero ONUVA Integración de Sistemas	Versión inicial
04 de enero de 2009	José Miguel Parrella Romero ONUVA Integración de Sistemas	Uso de qemu en lugar de kvm en los ejemplos Ajuste en la introducción del tema de destrezas comunes
04 de enero de 2009	Ailé Carelén Filippi Sánchez ONUVA Integración de Sistemas	Ficha catalográfica Versión final
12 de enero de 2009	José Miguel Parrella Romero ONUVA Integración de Sistemas	Ajustes a la ficha y a la portada Segunda versión
04 de junio de 2009	José Miguel Parrella Romero ONUVA Integración de Sistemas	Ajustes técnicos

Contenido

Créditos y licencia.....	4
Convenciones tipográficas.....	4
Introducción a Canaima.....	5
Componentes nativos de Canaima.....	7
Listas de software.....	7
Listas funcionales.....	8
Árbol de dependencias.....	12
Repositorios.....	15
Instalador.....	20
Módulo del instalador.....	23
Paquete de integración.....	24
Medios vivos.....	28
Estrategias comunes de desarrollo de Canaima.....	29
Construir una jaula de la rama de pruebas de Debian.....	30
Preparar una carpeta de trabajo del instalador.....	31
Construir un nuevo instalador a partir de la carpeta de trabajo.....	32
Modificar un disco volátil inicial.....	33
Modificar el contenido de un paquete binario.....	34
Construir paquetes binarios a partir de un paquete fuente.....	35
Casos específicos de desarrollo de Canaima.....	36
Agregar paquetes binarios de software en el instalador.....	37
Remover paquetes binarios de software del instalador.....	38
Agregar, remover o modificar la preconfiguración de Debconf en el instalador.....	39
Agregar, remover o modificar la postconfiguración en el instalador.....	40
Cambiar la preconfiguración del manejador de ventanas GNOME.....	41
Cambiar el estilo visual.....	42
Cambiar el perfil predeterminado de nuevos usuarios.....	43
Crear un repositorio parcial para uso institucional.....	44
Agregar o cambiar llaves PGP para el sistema de paquetes.....	46
Referencias.....	47

Créditos y licencia



© 2008-2009 Centro Nacional de Tecnologías de Información

© 2008-2009 ONUVA Integración de Sistemas

Este documento se distribuye al público como *documentación y conocimiento libre* bajo los términos de la Licencia Pública General GNU, que puede obtener en la dirección Web:

<http://www.gnu.org/copyleft/gpl.html>

Convenciones tipográficas

Texto enfatizado, *anglicismos*, **texto resaltado**, comandos, salidas, paquetes o contenido de archivos.



Indica información muy importante con respecto al contenido



Indica información importante para la puesta en práctica



Indica comandos, salidas en pantalla o contenido de archivos



Indica otros recursos donde puede conseguir información adicional



Indica información complementaria referente al capítulo



Indica los pasos de un procedimiento

Introducción a Canaima

En términos generales, Canaima GNU/Linux es una distribución de software libre y estándares abiertos basada en el sistema de paquetes APT dirigida a usuarios finales venezolanos y desarrollada en concordancia con el marco legal vigente en Venezuela¹.

Su base de software es pequeña y sustentable, sobre todo cuando se le compara con otros proyectos como Debian o Ubuntu, contando con menos de mil quinientos (1500) paquetes binarios de software. A la fecha se mantiene como premisa la compatibilidad binaria con, al menos, Debian y Ubuntu.

Canaima es mantenida para las arquitecturas x86 (i386) y x86-64 (amd64) que se encuentra en procesadores de los fabricantes Intel, AMD y VIA de 32 y 64 bits².



Es importante resaltar que el mayor esfuerzo en desarrollo y soporte de Canaima GNU/Linux se invierte en la arquitectura i386, por ser esta la que cuenta con mayor cantidad de usuarios a nivel nacional.

La base de software de Canaima ha sido tomada de una captura de la rama de pruebas de Debian GNU/Linux³ para Mayo 2008. Ya que la rama de pruebas de Debian GNU/Linux no ha sido congelada para ser liberada oficialmente, las versiones de algunos paquetes de software difieren entre Canaima y Debian y, de hecho, podrían presentar incompatibilidades al momento de su instalación.

1 Esto incluye no sólo el Decreto Presidencial 3390, sino también las políticas de Estado en materias de tecnologías de información y comunicaciones, las Normas Técnicas del CNTI y varias Leyes, Decretos Leyes y Reglamentos vinculados.

2 Exceptuando procesadores de la familia Intel Itanium.

3 El nombre código que se le dará a esta rama cuando se libere al público es *lenny*.

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Canaima utiliza el sistema de paquetes APT⁴, posiblemente el mecanismo de distribución gestionada de software de mayor difusión a nivel internacional. Es utilizado por distribuciones de impacto global como Debian y Ubuntu, y es la base de centenares de distribuciones incluyendo algunas utilizadas por OEMs en equipos portátiles y de escritorio.

En ese sentido, una de las partes más importantes de Canaima es su repositorio, que cuenta con tres (3) ramas con niveles de servicio diferenciados y la posibilidad de sincronizar sus paquetes de software con los repositorios de la rama de pruebas de Debian. Así mismo, es posible incluir nuevos paquetes de software en los repositorios en un momento dado. Los repositorios son autocontenidos.

Canaima se distribuye en distintos medios, que incluyen el repositorio, el instalador en formato DVD para arquitecturas i386 y amd64, el LiveDVD para múltiples arquitecturas con su instalador integrado y el instalador para dispositivos USB.

Como se explicará con mayor detalle en este manual, Canaima incorpora muchas mejoras con respecto a otras distribuciones de software libre y sistemas operativos propietarios, incluyendo una lista de software para usuarios finales, un estilo visual de alta calidad, perfiles para nuevos usuarios e instalación sencilla.



Para mayor información sobre el proyecto Canaima GNU/Linux visite la página Web **canaima.softwarelibre.gob.ve**.

⁴ *Herramienta avanzada de empaquetado*, por sus siglas en inglés.

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Componentes nativos de Canaima

Luego de realizar nuestra introducción a la distribución Canaima, pasaremos a describir en detalle cada uno de los componentes nativos del proyecto, que requieren el esfuerzo intelectual para la arquitectura, diseño y desarrollo que hacen de Canaima un producto inédito en el mercado regional.

Listas de *software*

Canaima GNU/Linux es un sistema operativo. El objetivo de un sistema operativo es darle sentido útil a un computador, por lo que debe contener programas y aplicaciones que permitan al usuario explotar el sistema informático.

En cada versión de Canaima el *Equipo de Desarrollo* determina que software se quiere incluir en la distribución. Este software debe ser distribuido como software libre basado en estándares abiertos de acuerdo al marco legal vigente, con las excepciones que el Centro Nacional de Tecnologías de Información autorice.

Usualmente, las aplicaciones que se quieren incluir en Canaima ya han sido preparadas por los desarrolladores de distribuciones como Debian para su distribución bajo el sistema APT. En ese caso, se hace uso del Sistema de Rastreo de Paquetes de Debian⁵ para ubicar el software deseado y encontrar el nombre del paquete o paquetes correspondientes.

En caso contrario, uno o más desarrolladores de Canaima preparan el software para

⁵ [Http://packages.qa.debian.org/](http://packages.qa.debian.org/)

su distribución bajo el sistema APT de acuerdo a las mejores prácticas del Proyecto Debian, reflejadas en documentos fundacionales como *Debian Policy Manual* (Jackson et al., 1996) y *Guía del nuevo desarrollador de Debian* (Rodin et al., 1998)



El desarrollo de paquetes fuentes y binarios del sistema APT está fuera del alcance de este manual. Puede consultar el material en castellano elaborado por Parrella, J. para varios talleres a nivel internacional en **distribuciones.com.ve**.

En todo caso se contempla la posibilidad de que el software empaquetado para el sistema APT o las mejoras realizadas a paquetes ya preparados por Debian puedan ser enviadas de vuelta a Debian y otros proyectos con la finalidad de colaborar en su desarrollo, lo que forma parte esencial de la visión del Proyecto Canaima.

Listas funcionales

Una vez que se determinan los nombres de los paquetes que se desean incluir en una versión de Canaima, se genera una **lista funcional** en texto plano que contiene los nombres de los paquetes, uno por línea, separados por caracteres de nueva línea.



Tenga en cuenta que los nombres de los paquetes pueden cambiar en las arquitecturas i386 y amd64, por lo que se generan dos (2) listas funcionales; la primera para i386 y la segunda para amd64.

Para ilustrar el contenido de estas listas funcionales, se transcribe la versión 2.0 de la lista funcional para la arquitectura i386:



alien	openoffice.org-gnome
console-common	planner
console-data	rhythmbox
dosfstools	synaptic
ethtool	totem-mozilla
manpages-es	tsclient
manpages-es-extra	update-notifier
module-assistant	ttf-bitstream-vera
openssh-client	ttf-freefont
openssh-server	ttf-opensymbol
sudo	sun-java6-jre
xfspgrog	sun-java6-plugin
alsa-base	ffmpeg
alsa-utils	gspca-modules-2.6-686
anacron	gstreamer0.10-alsa
avahi-daemon	gstreamer0.10-esd
cpufrequtils	gstreamer0.10-ffmpeg
cupsys	gstreamer0.10-gnomevfs
cupsys-bsd	gstreamer0.10-plugins-base
cupsys-client	gstreamer0.10-plugins-good
cupsys-driver-gutenprint	gstreamer0.10-plugins-ugly
desktop-base	gstreamer0.10-x
dia-gnome	lame
discover1	mesa-utils
eject	w32codecs
foomatic-db	vlc
foomatic-db-engine	vorbis-tools
foomatic-db-gutenprint	openoffice.org-gtk
foomatic-db-hpijs	openoffice.org-style-tango
foomatic-filters	less
foomatic-filters-ppds	xdg-user-dirs
foo2zjs	xdg-user-dirs-gtk
gdm-themes	scribus

gimp	xscreensaver
glabels	xscreensaver-gl
hibernate	aspell-es
hotkey-setup	thunderbird
hpijs	thunderbird-gnome-support
hpijs-ppds	thunderbird-locale-es-ar
hplip	firefox
libnss-mdns	firefox-gnome-support
menu	firefox-l10n-es-ar
openoffice.org	freemind
printconf	splashy
rar	gnome-games
unrar	smbclient
twm	smbfs
uswsusp	samba
vbetool	python-gst0.10
xdebconfigurator	bash-completion
xdg-utils	openclipart-png
xorg	openclipart-openoffice.org
x-window-system	xnest
xresprobe	obex-data-server
xterm	gnome-bluetooth
acpi	openproj
acpid	brasero
acpi-support	gtkpod
apmd	os-prober
avahi-autoipd	pidgin
bluez-utils	build-essential
bluetooth	linux-headers-2.6-686
hibernate	ntfs-3g
pcmciautils	mc
radeontool	screen
tpconfig	xsane

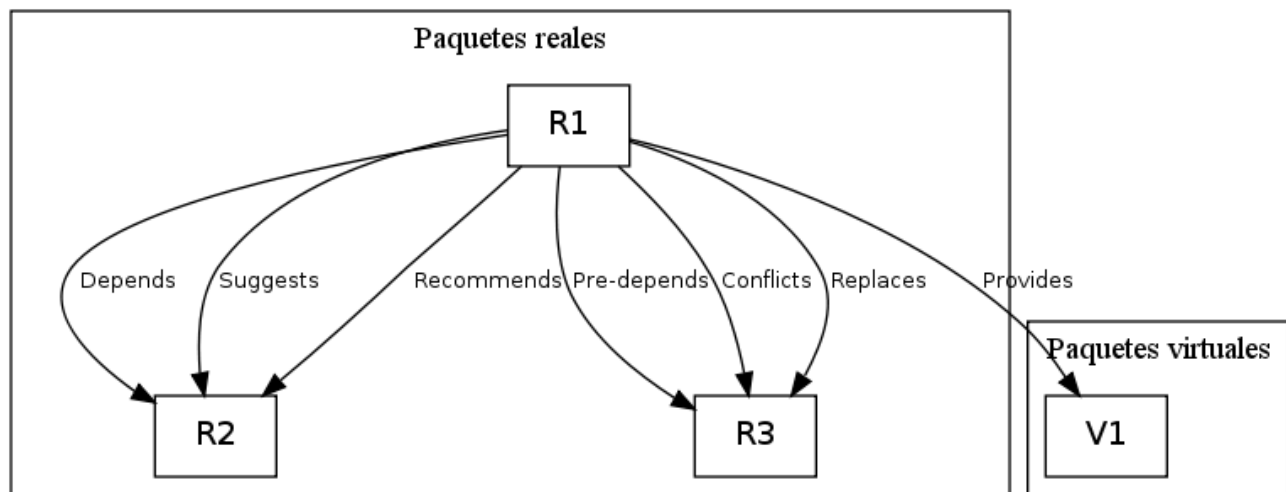
wireless-tools	convmv
wpasupplicant	telnet
myspell-es	subversion
openoffice.org-help-es	htop
openoffice.org-l10n-es	tcptrack
openoffice.org-thesaurus-es	esound-clients
alacarte	evolution-plugins
amsn	perl-doc
bluez-gnome	wine
deskbar-applet	libui-dialog-perl
ekiga	ntfs-config
gdebi	firmware-iwlwifi
gksu	system-config-printer
gnome-btdownload	gnome-audio
gnome-desktop-environment	debconf-utils
gnome-power-manager	gstreamer0.10-lame
gnome-screensaver	ntp
gnome-themes-extras	language-env
gparted	canaima-acerca
gstreamer0.10-ffmpeg	canaima-llaves
gthumb	canaima-base
i2e	canaima-estilo-visual
inkscape	
libgnome2-perl	
liferea	
menu-xdg	
nautilus-sendto	
nautilus-open-terminal	
network-manager-gnome	

Tabla 1: Lista funcional para i386 (2.0)

Árbol de dependencias

En el sistema de paquetes APT, un paquete real **A** tiene varias interacciones posibles con un paquete real **B**. Por ejemplo, **A** puede depender de **B**, **B** puede recomendar a **A**, o **A** puede tener conflictos con **B**. Si las dependencias no están presentes, el instalador de Canaima no finalizará la instalación correctamente.

Es por ello que, una vez definida la lista funcional, es necesario resolver el árbol de dependencias de todos los paquetes involucrados.



Interrelaciones granulares entre paquetes en el sistema APT

Ilustración 1: Algunas relaciones entre paquetes binarios del sistema APT

El portafolio de herramientas libres disponibles hoy en día en el mercado no ofrece una opción recomendada para resolver este problema. El *Equipo de Desarrollo de Canaima* estudió algunas de las alternativas disponibles⁶, de las cuales se presenta una tabla

⁶ Obviamente, existen varias alternativas disponibles, algunas de ellas desarrolladas en el marco del Proyecto Ubuntu, pero por razones de tiempo no fue posible investigarlas todas

comparativa a continuación:

Alternativa	Resultado del análisis
CDD	<i>Custom Debian Distributions</i> presenta un enfoque de alto nivel a herramientas como <i>debootstrap/cdebootstrap</i> , <i>debian-cd</i> y <i>reprepro</i> , permitiendo construir discos de instalación de Debian a partir de listas funcionales. Sin embargo, la experiencia de Canaima 1.0 y 1.1 con CDD no fue satisfactoria, y herramientas como <i>simple-cdd</i> presentan problemas en la actualidad para integrar paquetes de software de distintas ramas y con distintos niveles de compatibilidad binaria.
<i>debpartial-mirror</i>	<i>debpartial-mirror</i> es una herramienta diseñada para armar repositorios resolviendo dependencias a partir de listas funcionales. Para ello utiliza la librería <i>python-cdd</i> (entre otras herramientas) – las pruebas realizadas no fueron satisfactorias ya que el repositorio generado no era autocontenido.
<i>python-cdd</i>	<i>python-cdd</i> es la librería de Python utilizada por <i>debpartial-mirror</i> , entre otras herramientas. Posee un algoritmo de resolución de dependencias que puede ser aprovechado en un nuevo desarrollo basado en Python para obtener el árbol de dependencias. Sin embargo, por restricciones de tiempo no fue posible ahondar en esta alternativa.
<i>debootstrap</i> <i>cdebootstrap</i>	Esta alternativa utiliza las librerías del sistema APT para resolver el árbol de dependencias, tal y como lo haría una herramienta de alto nivel como <i>apt-get</i> o <i>aptitude</i> . Es la alternativa utilizada por el Instalador de Debian, que toma una lista de paquetes funcionales y los instala dentro de una jaula una vez que se ha instalado un sistema base funcional. La jaula también es un subproducto importante para otros productos, lo cual se mostrará más adelante.

Tabla 2: Alternativas para calcular el árbol de dependencias de una lista funcional de paquetes de software

Una jaula es un directorio donde se replica la raíz de un sistema operativo basado en

GNU/Linux. Herramientas como `debootstrap` y `cdebootstrap` hacen una instalación de Debian en una jaula compuesta por paquetes binarios de prioridad `essential` e `important`. No se incluyen en la jaula los paquetes correspondientes al *kernel* Linux ni al gestor de arranque GRUB, ya que la jaula no se inicia como un sistema normal.

Luego de realizar esta instalación básica es posible ejecutar comandos dentro de la jaula para actualizar el sistema a la rama de pruebas de Debian, utilizando la imagen realizada por Canaima en Mayo 2008, y utilizar `aptitude` para instalar la lista de paquetes funcionales dentro de la jaula.

Ya que hay paquetes que intentan iniciar servicios y esto no es deseable dentro de la jaula, se modifica el archivo `/usr/sbin/policy-rc.d` para evitar el inicio de los servicios:



```
#!/bin/sh  
exit 101
```

Este es un procedimiento estándar utilizado por distintos proyectos en Debian. De otra manera, es posible determinar el árbol de dependencias pero el sistema anfitrión terminará ejecutando servicios no deseados.



Cualquier operación relacionada con la jaula se debe ejecutar utilizando las herramientas `canaima-jaula` publicadas en el control de versiones Subversion del proyecto `canaima`.

Una vez instalados los paquetes de la lista funcional, `aptitude` habrá resuelto el árbol de dependencias, y éste podrá obtenerse utilizando la herramienta de bajo nivel `dpkg` de la siguiente manera:



```
dpkg --get-selections | grep -v deinstall  
> lista.txt
```

Repositorios

Todos los repositorios de Canaima GNU/Linux, excepto el repositorio universo, utilizan reprepro para su creación y administración. Con reprepro se han cubierto todas las expectativas técnicas del proyecto a la fecha y se estima que el desarrollo de esta herramienta continuará prestando las funcionalidades esperadas por Canaima.

Para configurar un repositorio basado en reprepro es necesario crear una carpeta de trabajo y, dentro de ella, un directorio conf. Este directorio contendrá la configuración del repositorio, dividida en los siguientes archivos:

- `distributions`: archivo principal que define las ramas, componentes y arquitecturas de cada uno de los repositorios
- `updates`: archivo que define repositorios de origen desde donde se pueden descargar paquetes adicionales
- `pulls`: archivo que define reglas para transición de paquetes entre ramas
- `incoming`: se utiliza en el caso que se desee incluir paquetes fuentes y binarios automáticamente desde una ruta predefinida



El único archivo requerido para el funcionamiento básico de reprepro es `conf/distributions`.

Una vez que se escriben los archivos de configuración, es posible utilizar la

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

herramienta reprepro en la línea de comandos para varias tareas, como se cita en la siguiente tabla comparativa:

Acción	Comando
Incluir un paquete binario foo en la rama bar	<code>reprepro includedeb bar foo.deb</code>
Incluir un módulo del instalador foo en la rama bar	<code>reprepro includeudeb bar foo.deb</code>
Remover un paquete foo de la rama bar	<code>reprepro remove bar foo</code>
Actualizar la rama foo con el repositorio declarado en conf/updates	<code>reprepro update foo</code>
Migrar paquetes a la rama foo de acuerdo a conf/pulls	<code>reprepro pull foo</code>
Obtener la versión y arquitectura del paquete foo en la rama bar	<code>reprepro list bar foo</code>

Tabla 3: Comandos comunes de reprepro

Con la finalidad de complementar la información presentada en este apartado, se transcriben y discuten en detalle los archivos de configuración de reprepro del repositorio principal de Canaima que se sirve por HTTP a través de `repositorio.canaima.softwarelibre.gob.ve`:



```
Origin: Canaima
Label: estable
Suite: estable
Codename: estable
Version: 2.0
Architectures: i386 amd64 source
```

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Components: usuarios servidores
Description: Canaima GNU/Linux 2.0
SignWith: repositorios@canaima.softwarelibre.gob.ve

Origin: Canaima
Label: desarrollo
Suite: desarrollo
Update: upstream
Codename: desarrollo
Version: Desarrollo
Architectures: i386 amd64 source
Components: usuarios servidores
Description: Canaima GNU/Linux Desarrollo
SignWith: repositorios@canaima.softwarelibre.gob.ve

Origin: Canaima
Label: pruebas
Suite: pruebas
Codename: pruebas
Pull: desarrollo
Version: Pruebas
Architectures: i386 amd64 source
Components: usuarios servidores
Description: Canaima GNU/Linux Desarrollo
SignWith: repositorios@canaima.softwarelibre.gob.ve

Tabla 4: Contenido del archivo conf/distributions

En este caso se definen tres (3) ramas, estable, desarrollo y pruebas, para las arquitecturas i386 y amd64, incluyendo paquetes fuentes y usando como componentes usuarios y servidores. Todos los repositorios se firman con la llave PGP principal que tenga una identidad repositorios en canaima.softwarelibre.gob.ve.

La rama pruebas toma paquetes directamente del repositorio desarrollo de acuerdo a lo que se especifica en `conf/pulls`. La rama desarrollo se actualiza directamente con el repositorio externo llamado `upstream`, que se define en el archivo `conf/updates` de la siguiente manera:



```
Name: upstream
Method: http://universo.canaima.softwarelibre.gob.ve/
Suite: lenny
Architectures: source i386 amd64
Components: main>usuarios non-free>usuarios contrib>usuarios
UDebComponents: none
FilterList: purge paquetes-canaima
```

Tabla 5: Contenido del archivo `conf/updates`

En este archivo le indicamos a `reprepro` que use `universo.canaima.softwarelibre.gob.ve` como repositorio externo (un repositorio de Debian tradicional), en particular la suite `lenny` y las arquitecturas citadas.

Adicionalmente se hace un mapeo entre los componentes `main`, `contrib` y `non-free` con el componente `usuarios` y se indica a `reprepro` que solo tome los paquetes definidos en el archivo `paquetes-canaima`. Esta es una lista de paquetes de software con el árbol de dependencias resuelto.



```
Name: desarrollo
From: desarrollo
```

Tabla 6: Contenido del archivo `conf/pulls`

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Este archivo simplemente indica que desarrollo es un repositorio del cual se pueden tomar paquetes para actualizar otras ramas.



Name: desarrollo
IncomingDir: /srv/incoming/dir
TempDir: /srv/incoming/tmp
Allow: desarrollo

Tabla 7: Contenido del archivo conf/incoming

En este archivo se indica que la rama desarrollo podrá tomar paquetes fuentes y binarios de la carpeta /srv/incoming/dir, lo cual es útil para enlazar el trabajo de paquetes de software con los repositorios de forma automatizada. En la carpeta se pueden colocar los paquetes por SFTP, FTP o cualquier otro método.

Instalador

El método recomendado para instalar Canaima es el *Instalador de Debian*⁷. Este desarrollo, que cuenta con muchos años de desarrollo y que ha servido como base para instaladores de otros sistemas operativos, es una opción muy robusta para garantizar la calidad en la instalación de una distribución como Canaima.

El instalador de Debian está compuesto por un pequeño sistema GNU/Linux diseñado para arrancar el programa principal que organiza un menú de módulos y realiza algunas tareas básicas como acceder al medio de instalación para obtener más módulos y preconfigurar el entorno de instalación.

Los módulos del instalador se distribuyen en pequeños paquetes binarios del sistema APT de extensión udeb y proporcionan funcionalidad adicional al instalador como configuración de la red, hora, sistema de paquetes, particionado y acceso a sistemas de archivos, instalación inicial, post-configuración, instalación del kernel e instalación del cargador de arranque.

Usualmente el instalador, los módulos básicos y los drivers de Linux necesarios para arrancar una variedad más o menos amplia de sistemas se incluyen en una imagen CPIO comprimida que constituye el initrd o disco volátil inicial. Este disco se descomprime en la memoria RAM del sistema y desde allí se ejecuta todo el proceso de instalación.

Con la finalidad de ilustrar el procedimiento de instalación de Canaima, se describe en detalle el proceso de inicio, carga del instalador y sus módulos y fases funcionales del instalador, cuyo conocimiento resultará de mucha utilidad incluso para diagnosticar

⁷ <http://www.debian.org/devel/debian-installer/>

problemas de arranque de otros sistemas GNU/Linux:



1. Si está configurada para iniciar por el medio seleccionado, la BIOS del computador carga el sector principal de arranque⁸ del medio de instalación de Canaima
2. En el sector de arranque se encuentra GRUB con extensiones para animación gráfica, un cargador de arranque que le presenta al usuario las opciones de inicio de la instalación
3. Una vez que el usuario selecciona la opción de arranque deseada, un kernel Linux (`vmlinux`) se carga en la memoria RAM del sistema y se inicia. Esta tarea la hace el cargador de arranque.
4. El kernel está diseñado para que descomprima el disco volátil inicial (`initrd.gz`) en la memoria RAM y ejecute el comando `/sbin/init` de ese disco inicial.
5. El comando `/sbin/init` invoca al programa principal del instalador, llamado `main-menu`.
6. El instalador le pregunta al usuario el mapa de teclado que desea utilizar durante la instalación, y luego detecta la información del medio de instalación, preconfigura el ambiente de instalación, monta el medio de instalación y carga módulos adicionales del instalador.
7. El instalador configura la red para ser usada durante la instalación, cuyos datos serán usados para el sistema una vez que quede instalado.

⁸ Para facilitar la explicación usamos el término registro principal de arranque aunque el concepto no aplica para medios basados en ISO 9660, donde se utiliza la extensión El Torito.

8. El instalador le presenta al usuario el asistente de particionado para que seleccione donde estará el sistema de archivos raíz del sistema Canaima deseado. Con esta información, el instalador prepara los sistemas de archivos para instalar.
9. El usuario introduce la clave del administrador del sistema, así como los datos de un usuario local sin privilegios para hacer uso del mismo. Este usuario se une a varios grupos del sistema como audio, plugdev y netdev con la finalidad de que pueda hacer uso productivo del sistema.
10. El instalador ejecuta `debootstrap` para instalar los paquetes esenciales e importantes en el sistema de archivos que el usuario seleccionó.
11. El instalador configura el sistema de paquetes APT y carga un módulo del instalador que instala una lista funcional de paquetes de software.
12. Este módulo ejecuta, opcionalmente, un script de postconfiguración en el sistema que está siendo instalado.
13. El instalador instala el kernel Linux correspondiente y el cargador de arranque GRUB.
14. El sistema genera un evento ACPI para reiniciar el equipo, lo que usualmente expulsa el medio de instalación y reinicia usando el disco duro, que ya tiene Canaima instalado.

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Módulo del instalador

Al final de la instalación, se ejecuta un módulo que instala los paquetes especificados en la lista funcional y aplica el *shell script* de postconfiguración.

El módulo utilizado para la instalación de los paquetes en la lista funcional y la postconfiguración ha sido desarrollado tomando como base el código del módulo de `simple-cdd-profiles`.

Paquete de integración

Canaima no es sólo una selección de paquetes y un instalador automatizado para facilitar la utilización de software libre y de estándares abiertos; también incluye mejoras con respecto a un sistema operativo libre tradicional, entre las cuales podemos resaltar:

- Estilo visual de calidad internacional desarrollado íntegramente por talento venezolano y con herramientas libres y de estándares abiertos bajo GNU/Linux
- Capacidad de incluir un perfil inicial para nuevos usuarios del sistema
- Archivos esenciales del sistema localizados para Canaima
- Integridad y autenticidad en el sistema de paquetes a través de la utilización de PGP
- Scripts de asistencia al usuario final para tareas comunes

Para lograr la integración de estos elementos de forma elegante dentro del sistema operativo Canaima, se desarrolló un paquete fuente de acuerdo a las mejores prácticas del Proyecto Debian llamado `canaima-integracion`.



Es importante resaltar que uno de los objetivos técnicos de Canaima es evitar incurrir en técnicas poco sustentables de desarrollo de distribuciones, como por ejemplo la utilización de scripts de normalización post-instalación o intervenciones usuario-por-usuario.

El paquete `canaima-integracion` construye cuatro (4) paquetes binarios:

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Paquete	Descripción
canaima-acerca	Menú "Acerca de CANAIMA"
canaima-base	Archivos esenciales del sistema y perfil de nuevos usuarios
canaima-estilo-visual	Estilo visual de Canaima
canaima-llaves	Llaves públicas PGP de los repositorios de Canaima

Tabla 8: Paquetes binarios del paquete fuente canaima-integracion

Previamente, `canaima-integracion` también incluía los paquetes `canaima-actualizador` y `canaima-encuesta`, que han sido descontinuados por solicitud del CNTI.

Como premisa de desarrollo se estima que toda la rama 2.0 de Canaima GNU/Linux necesitará sobrescribir archivos de otros paquetes binarios. Esto no es deseable en sistemas basados en APT ya que si el paquete **A** sobrescribe el archivo `/foo` del paquete **B**, sucesivas actualizaciones del paquete **B** sobrescribirán nuevamente el archivo.

El estado del arte en sistemas de archivos en GNU/Linux y las restricciones de tiempo del Proyecto Canaima no permiten considerar una opción basada en sistemas de archivos como por ejemplo *overlays*, vistas o espacios de nombres para archivos. Por lo tanto, se consideraron las siguientes alternativas de solución:

Alternativa	Observaciones
<i>Modificación de los paquetes binarios</i>	Solución poco elegante, requiere intervención manual con cada actualización, requiere mantener una base de datos de cada cambio.
<i>Modificación de los paquetes fuentes</i>	Solución muy elegante y apegada a mejores prácticas, podría requerir revisión manual de algunos parches, requiere mantener una base de datos de cada cambio, requiere reconstruir los paquetes binarios lo cual puede consumir mucho tiempo.
<i>Sobreescritura de archivos con dpkg-divert</i>	Procedimiento apegado a prácticas del Proyecto Debian, no requiere revisión manual en actualizaciones, no requiere cambiar los paquetes fuentes o binarios.

Tabla 9: Alternativas de responsabilidad compartida sobre archivos en el sistema APT

dpkg-divert es una herramienta que intercepta las escrituras al sistema de archivos en las instalaciones y actualizaciones de paquetes binarios y desvía a los paquetes no autorizados para escribir en una determinada ruta. Así, con nuestro ejemplo anterior, cuando el paquete **B** se actualice, el archivo /foo del paquete **B** iría a otro destino y no sobrescribiría el del paquete **A**.

Para la solución con dpkg-divert es importante considerar que el reporte de error 102144⁹ de Debian afectó el conjunto de pruebas funcionales, ya que estaba contemplada la instalación en un entorno de directorios separados por particiones, y Canaima utiliza

⁹ <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=102144>

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

/var/lib/paquete/diverts para almacenar los archivos desviados, ya que en algunos directorios, como /etc/grub.d, no serviría desviar el archivo dentro de la misma carpeta.

Es por esta razón que el paquete dpkg en Canaima ha sido modificado para incluir el parche recomendado en el reporte 102144, en la versión 1.14.22canaima1.

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Medios vivos

Para aquellos usuarios que sólo deseen probar Canaima sin tener que instalarlo y que dispongan de un sistema con suficientes recursos de hardware, en particular al menos 1024 MB. de memoria RAM, está disponible un medio vivo en formato DVD (LiveDVD) para arquitecturas de 32- y 64-bits.

Para hacer que el usuario experimente un sistema Canaima completo sin tener que instalarlo, se hace uso de la memoria volátil RAM del sistema para almacenar toda la información de estado correspondiente a aplicaciones y datos en ejecución.

Para desarrollar el medio vivo en formato DVD se hace uso de la jaula preparada anteriormente, por lo que se afianza la utilidad de la alternativa de *bootstrapping* para el desarrollo de todos los productos de Canaima. Esta jaula se comprime utilizando SquashFS, con lo que se puede lograr una relación de compresión de 3:1.

Finalmente, se prepara el disco para el arranque utilizando un kernel y un disco volátil inicial especialmente preparado para montar la imagen SquashFS junto con porciones de la memoria RAM haciendo uso de UnionFS. Una vez realizada esta operación, el sistema Live se inicia exactamente como un sistema instalado en un disco duro tradicional, solo que todas las operaciones de escritura se realizarán sobre la memoria volátil.

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Estrategias comunes de desarrollo de Canaima

En este capítulo se describen algunas estrategias comunes de desarrollo de Canaima que son útiles en casi todos los casos específicos de desarrollo que se tratarán más adelante.

Estas estrategias son las mismas que han sido utilizadas por el *Equipo de Desarrollo de Canaima* en el desarrollo de la distribución, y aunque no representan la única forma de trabajar en el marco del desarrollo de distribuciones, son normalizadas y recomendadas como mejores prácticas en este documento.

En los medios de instalación de Canaima y en la plataforma de desarrollo encontrará algunos shell script que facilitan algunos de los procedimientos descritos a continuación, en particular `canaima-initrd`, `canaima-imagen` y `canaima-reconstructor`. La descripción detallada del procedimiento tiene, sin embargo, un gran valor didáctico.



La lista de correo electrónico del Equipo de Desarrollo de Canaima es el medio de comunicación apropiado para discutir cambios en este documento.

Construir una jaula de la rama de pruebas de Debian

Si bien algunos ayudantes de *bootstrap* poseen algoritmos para construir directamente jaulas de la rama de pruebas, este procedimiento es más apropiado para cualquier versión de los ayudantes, a la vez que ayuda a entender el funcionamiento de la jaula:



1. Preparar una carpeta de trabajo, por ejemplo:

```
mkdir -p /srv/canaima/jaulas
```
2. Instalar el paquete `cdebootstrap`:

```
aptitude install cdebootstrap
```
3. Ejecutar `cdebootstrap` para la rama estable de Debian:

```
cdebootstrap stable prueba
```
4. Copiar los archivos `/etc/hosts`, `/etc/resolv.conf` y `/etc/apt/sources.list` dentro de la jaula:

```
cp /etc/hosts prueba/etc  
cp /etc/resolv.conf prueba/etc  
cp /etc/apt/sources.list prueba/etc/apt
```
5. Ingresar a la jaula usando la herramienta `chroot`:

```
chroot prueba/ /bin/bash
```
6. Editar el archivo `/etc/apt/sources.list` (de la jaula) y cambiar `stable` por `testing`
7. Actualizar el sistema de la jaula

```
aptitude update  
aptitude dist-upgrade
```

Preparar una carpeta de trabajo del instalador

La naturaleza de las imágenes ISO 9660 y los medios ópticos indica que son de sólo lectura, por lo que es necesario copiarlos para preparar una carpeta de trabajo del instalador de acuerdo a este procedimiento:



1. Cree una carpeta de trabajo, por ejemplo:

```
mkdir -p /srv/canaima/instalador
```
2. Opcionalmente, instale el paquete rsync (o utilice cp -a) usando aptitude:

```
aptitude install rsync
```
3. Si tiene un DVD de instalación de Canaima, deje que el sistema lo monte automáticamente o móntelo manualmente; en sistemas Debian o derivados superiores a 4.0 basta con ejecutar `mount /cdrom`
 1. Copie el contenido del DVD a la carpeta:

```
rsync -avz /cdrom/ /srv/canaima/instalador
```
4. Si tiene una imagen ISO 9660 del DVD de instalación de Canaima, puede montar la imagen usando:

```
mount -o loop imagen.iso /media/cdrom0
```

 1. Copie el contenido del DVD a la carpeta:

```
rsync -avz /media/cdrom0 /srv/canaima/instalador
```

Construir un nuevo instalador a partir de la carpeta de trabajo

Este procedimiento convierte una carpeta de trabajo del instalador en una imagen ISO 9660 que puede ser iniciada en un sistema.



1. Instale el paquete mkisofs usando aptitude:

```
aptitude install mkisofs
```

2. Suponiendo que la carpeta de trabajo sea /srv/canaima/instalador, ejecute:

```
mkisofs -r -J -l -no-emul-boot -boot-load-size 4 -boot-info-table -b boot/grub/stage2_eltorito -o imagen.iso
```



Para probar esta imagen se puede utilizar los emuladores qemu, qemu/kqemu o kvm de la siguiente forma:

```
qemu -boot d -cdrom imagen.iso
```

También se pueden utilizar sistemas de virtualización libres o propietarios, o copiar la imagen en un medio óptico:

```
wodim imagen.iso
```

Modificar un disco volátil inicial



Este procedimiento funciona para discos volátiles iniciales (imágenes `initrd`) que utilicen compresión con `gzip` y de tipo `CPIO`.



1. Crear una carpeta de trabajo para la modificación y cambiar el directorio a esta carpeta de trabajo:

```
mkdir mod && cd mod/
```

2. Descomprimir el disco usando `gunzip` y pasar la imagen a `cpio` para su descompresión:

```
gzip -d < ../initrd-viejo.gz | \  
cpio --extract --verbose --make-directories  
--no-absolute-filenames
```

3. Dentro de la carpeta `mod/`, modificar lo deseado
4. Dentro de la carpeta `mod/`, obtener una lista de todos los archivos, pasarlos a `cpio` y comprimir la imagen:

```
find . | cpio -H newc --create --verbose | \  
gzip -9 > ../initrd-nuevo.gz
```

Modificar el contenido de un paquete binario

En ocasiones es útil realizar modificaciones al contenido de un paquete binario sin tener que construirlo a partir de su paquete fuente. Esto suele ser útil en pruebas y para paquetes que no contienen archivos dependientes de la arquitectura.



Este procedimiento sobrescribirá el archivo llamado `paquete.deb`, por lo que debe respaldarlo apropiadamente.

En el DVD de instalación de Canaima hay un pequeño *shell script* que ayuda en la reconstrucción de paquetes binarios llamado `canaima-reconstructor`.



1. Descomprima el contenido del paquete binario en la carpeta `mod/`:

```
dpkg-deb -x paquete.deb mod
```

2. Descomprima los archivos de control del paquete binario en la carpeta `mod/DEBIAN`:

```
dpkg-deb -e paquete.deb mod/DEBIAN
```

3. Dentro de la carpeta `mod/`, modifique lo deseado

4. Dentro de la carpeta `mod/`, reconstruya el paquete binario:

```
dpkg-deb -b . . .
```

Construir paquetes binarios a partir de un paquete fuente

Un paquete fuente del sistema APT está compuesto por un archivo de control de extensión `.dsc`, un archivo comprimido con el código fuente de extensión `.tar.gz` y, en ocasiones, un archivo comprimido con parches de extensión `.diff.gz`.¹⁰



1. Instale el paquete `dpkg-dev` usando `aptitude`:

```
aptitude install dpkg-dev
```
2. Descomprima el paquete fuente

```
dpkg-source -x paquete.dsc
```
3. Se creará una carpeta con el nombre del paquete y su versión.
4. Opcionalmente, dentro de esta carpeta (que debe contener un directorio `debian/`), modifique lo deseado.
5. Dentro de la carpeta de trabajo, obtenga las dependencias de construcción:

```
apt-get build-dep
```
6. Construya el paquete usando:

```
dpkg-buildpackage
```



Este procedimiento no pretende ser una guía extensiva de construcción de paquetes o desarrollo de paquetes fuentes del sistema APT e insistimos en la necesidad de consultar documentos externos para mejorar estas destrezas.

¹⁰ Hemos omitido, sin comprometer la calidad del procedimiento, la descripción detallada de los conceptos de archivos *upstream* y paquetes fuentes nativos para hacer más comprensible el caso de uso.

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Casos específicos de desarrollo de Canaima

Canaima es una distribución de propósito general, por lo que no ha sido diseñada para cubrir las necesidades de cada una de las personas u organizaciones que deseen hacer uso de este sistema operativo en sus plataformas tecnológicas.

Es por ello que en casi todos los casos los usuarios de Canaima querrán derivar sus propias distribuciones, versiones o *sabores* a partir de Canaima GNU/Linux con la finalidad de cumplir con un objetivo de negocios particular.

En todo caso y de acuerdo a la normativa legal vigente en Venezuela, que incluye el Borrador de Norma Técnica del CNTI 0003:2008, se debe seguir el principio de *derivación convergente*. En ese sentido es muy importante que todos los procedimientos de ajustes, modificación, personalización o derivación de Canaima se realicen de acuerdo a las estrategias definidas en este documento.



La lista de correo electrónico del Equipo de Desarrollo de Canaima es el medio de comunicación apropiado para discutir cambios en este documento.

Agregar paquetes binarios de software en el instalador



1. Determinar el nombre de los paquetes binarios de software y descargarlos para la arquitectura deseada o bien preparar los paquetes de acuerdo a las mejores prácticas del Proyecto Debian
2. Desarrollar el árbol de dependencias de los paquetes y verificar que los paquetes no introduzcan conflictos en el sistema de paquetes, instalándolos manualmente usando dpkg en un sistema Canaima
3. En una carpeta de trabajo del instalador¹¹, ejecutar reprepro includedeb estable <ruta al paquete> por cada paquete que se desea agregar.
4. Agregar en la lista de paquetes funcional ubicada en simple-cdd/canaima.packages el nombre de los paquetes binarios agregados, uno por línea.
5. Rehacer la imagen ISO 9660

¹¹ Ver *Estrategias comunes de desarrollo de Canaima*

Remover paquetes binarios de software del instalador



1. Determinar el nombre de los paquetes binarios de software
2. Verificar que la remoción de paquetes no introduzca conflictos en el sistema de paquetes removiéndolos manualmente con aptitude en un sistema Canaima
3. En una carpeta de trabajo del instalador, ejecutar `reprepro remove estable <nombre del paquete>` por cada paquete que se desea remover.
4. Remover de la lista de paquetes funcional ubicada en `simple-cdd/canaima.packages` el nombre de los paquetes binarios removidos
5. Rehacer la imagen ISO 9660

Agregar, remover o modificar la preconfiguración de Debconf en el instalador



1. En una carpeta de trabajo del instalador, edite el archivo `simple-cdd/canaima.preseed` y agregue, remueva o modifique las respuestas a preguntas de Debconf que desee preconfigurar
 1. Puede obtener las respuestas en un sistema Canaima ya instalado utilizando el comando `debconf-get-selections`
 2. En el caso especial de preconfiguración del instalador, y para aquellas preguntas que ocurran antes del montaje del medio de instalación (paso 6) recomendamos incluir las respuestas en la línea de comandos del kernel en `/boot/grub/menu.lst`
2. Rehacer la imagen ISO 9660

Agregar, remover o modificar la postconfiguración en el instalador



1. En una carpeta de trabajo del instalador, edite el archivo `simple-cdd/canaima.postinst` y agregue, remueva o modifique comandos teniendo en cuenta que debe ser compatible con `/bin/sh`.
2. Rehacer la imagen ISO 9660



Tenga en cuenta que los comandos de postconfiguración se ejecutan en el sistema que está siendo instalado, no en el medio de instalación.

Cambiar la preconfiguración del manejador de ventanas GNOME

Esta preconfiguración incluye la posición de los paneles, íconos y applets de GNOME, las aplicaciones asociadas con Web y Correo, la *ubicación* del fondo de pantalla y splashscreen de GNOME, el *nombre* del tema de íconos y ventanas, ciertas preferencias de Nautilus, el formato del reloj y la transparencia del terminal.



1. En una carpeta de trabajo del paquete fuente `canaima-integracion`¹², edite los archivos bajo `canaima-estilo-visual/usr/share/gconf/defaults/`
2. Construya el paquete binario `canaima-estilo-visual`¹³ de acuerdo a las mejores prácticas del Proyecto Debian
3. Agregue el paquete `canaima-estilo-visual` al instalador



Por supuesto, puede agregar, remover o cambiar claves y valores del registro de GNOME con este procedimiento. Tenga en cuenta que no todas las aplicaciones de un sistema GNU/Linux responden o consultan el registro de GNOME, y otras podrían tener una segunda fuente de configuración que impida su preconfiguración.

¹² Ver *Estrategias comunes de desarrollo de Canaima*

¹³ Como se especificó anteriormente, este tema escapa al alcance del manual

Cambiar el estilo visual

El estilo visual de Canaima GNU/Linux incluye un tema de GRUB 2, un tema de Splashy, un tema de GDM, un splash screen de GNOME, un fondo de pantalla de GNOME, un tema de ventanas de Metacity un tema de íconos. Las organizaciones que deseen masificar la instalación de un estilo visual particular pueden modificarlo de forma elegante y sustentable siguiendo este proceso:



1. En una carpeta de trabajo del paquete fuente `canaima-integracion`, cambie el estilo visual ubicado bajo la carpeta `canaima-estilo-visual/`
2. Construya el paquete binario `canaima-estilo-visual` de acuerdo a las mejores prácticas del Proyecto Debian
3. Agregue el paquete `canaima-estilo-visual` al instalador

Cambiar el perfil predeterminado de nuevos usuarios

Usualmente, cuando un usuario nuevo es creado en un sistema GNU/Linux, se copia el contenido de la carpeta `/etc/skel` en el perfil del nuevo usuario. En ciertos escenarios, las organizaciones podrían desear preconfigurar aplicaciones o comportamientos del entorno de trabajo del usuario colocando información en esta carpeta.



1. En una carpeta de trabajo del paquete fuente `canaima-integracion`, cambie el perfil predeterminado para nuevos usuarios ubicado en `canaima-base/usr/share/canaima-base/etc/skel`
2. Construya el paquete binario `canaima-base` de acuerdo a las mejores prácticas del Proyecto Debian
3. Agregue el paquete `canaima-base` al instalador

Crear un repositorio parcial para uso institucional

Usualmente las ocasiones querrán agregar a Canaima nuevos paquetes, modificar algunos ya existentes o incluir aplicaciones empaquetadas por ellos en un repositorio para uso institucional. El procedimiento apropiado para esta actividad es el siguiente:



1. Diseñe un equipo en su red que servirá, vía HTTP, el repositorio institucional. Asegúrese de que las políticas de su red permiten que todos los equipos con Canaima accedan por HTTP a este servidor.
2. Instale reprepro y el servidor Web de su preferencia (p.ej., Cherokee, nginx, lighttpd, Apache)
3. Diseñe una carpeta para el repositorio, por ejemplo /srv/www/repositorio (en este caso, siguiendo FHS¹⁴)
4. En una carpeta de trabajo del repositorio, cree una carpeta conf/ y un archivo conf/distributions de contenido similar al siguiente:
Origin: CNTI
Codename: estable
Version: 2.0
Architectures: i386
Components: institucion
Description: Repositorio estable de CNTI para Canaima
5. Incluya paquetes binarios en su repositorio utilizando el

¹⁴ <http://www.pathname.com/fhs/pub/fhs-2.3.html#SRVDATAFORSERVICESPROVIDEDBYSYSTEM>

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

comando `reprepro includedeb estable <ruta al paquete binario>`

6. Una vez que incluya el primer paquete binario correctamente, se generarán las listas de paquetes para su componente.
7. En los clientes, agregue una entrada en el archivo `/etc/apt/sources.list` similar a:
`deb http://foo.bar.baz/ estable institucion`



Para mayor información sobre la utilización de `reprepro`, remítase a la documentación oficial de la herramienta en la dirección Web <http://mirrorer.alioth.debian.org/> o solicite ayuda en la lista de correo electrónico de usuarios del Proyecto Canaima.

Agregar o cambiar llaves PGP para el sistema de paquetes

El sistema de paquetes utilizado por Canaima utiliza PGP para garantizar autenticidad e integridad en las transacciones de descarga e identificación de paquetes. Ya que el sistema de paquetes APT tradicional utiliza sumas de comprobación MD5, SHA-1 y SHA-2, sólo es necesario firmar con una llave PGP el archivo Release por cada rama y componente de cada repositorio.

Es por ello que las organizaciones que establezcan sus propios repositorios desearán agregar o cambiar llaves PGP del sistema y firmar sus archivos Release en los repositorios institucionales para garantizar el nivel de seguridad esperado en el sistema de paquetes.



1. Cree y resguarde un par de llaves PGP¹⁵
1. En una carpeta de trabajo del paquete fuente canaima-integracion, agregue, cambie o remueva las llaves públicas en la carpeta canaima-llaves/usr/share/canaima-llaves, asegurándose de que cada llave pública haya sido exportada en ASCII, con extensión .asc¹⁶.
2. Construya el paquete binario canaima-llaves de acuerdo a las mejores prácticas del Proyecto Debian
3. Agregue el paquete canaima-llaves al instalador

¹⁵ La creación, resguardo y mantenimiento de llaves PGP con implementaciones libres como GnuPG está fuera del alcance de este documento

¹⁶ `gpg -armor -export <key ID> > llave.asc`

Código	VE2-0002-1002-340-09-04-0001.05	Fecha	04/06/2009
---------------	---------------------------------	--------------	------------

Referencias

1. JACKSON et al. (1996) *Debian Policy* [Manual de referencia en línea] Disponible en: <http://www.debian.org/doc/debian-policy/> [Consultado: 29/12/2008]
2. RODIN et al. (1995) *Debian New Maintainer's Guide* [Manual de referencia en línea] Disponible en: <http://www.debian.org/doc/maint-guide/> [Consultado: 29/12/2008]
3. POP et al. (2005) *Debian Installer Internals* [Manual de referencia en línea] Disponible en: <http://d-i.alioth.debian.org/doc/internals/> [Consultado: 29/12/2008]
4. NORONHA, G. (2001) *APT HOWTO* [Manual de referencia en línea] Disponible en: <http://www.debian.org/doc/manuals/apt-howto/index.es.html> [Consultado: 29/12/2008]
5. TILLE et al. (2008) *Debian Pure Blends* [Artículo en línea] Disponible en: <http://wiki.debian.org/DebianPureBlends> [Consultado: 29/12/2008]
6. LINK, B. (2007) *reprepro manual* [Manual de referencia en línea] Disponible en: <http://alioth.debian.org/plugins/scm cvs/cvsweb.php/~checkout~/mirrorer/docs/manual.html?rev=HEAD;content-type=text%2Fhtml;cvsroot=mirrorer> [Consultado: 29/12/2008]
7. ONUVA INTEGRACIÓN DE SISTEMAS (2008) *ONUVA publica código fuente de integración de Canaima GNU/Linux 2.0* [Noticia en línea] Disponible en: <http://canaima.proyectos.onuva.com/node/3> [Consultado: 29/12/2008]
8. DEBIAN LIVE PROJECT (2008) *Debian Live Manual* [Manual de referencia en línea] Disponible en: <http://live.debian.net/manual/html/> [Consultado: 29/12/2008]
9. CENTRO NACIONAL DE TECNOLOGÍAS DE INFORMACIÓN (2008) *Borrador de la Norma Técnica CNTI 0003:2008*