Índice

Cómo aportar a Canaima

Sistema de Registro Único: http://registro.canaima.softwarelibre.gob.ve/

Listas de Correo: http://listas.canaima.softwarelibre.gob.ve/

Enciclopedia Colaborativa: http://wiki.canaima.softwarelibre.gob.ve/

Reporte de Errores: http://proyectos.canaima.softwarelibre.gob.ve/canaima/

IRC

GIT

Git modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea. Para ser eficiente, si los archivos no se han modificado, Git no almacena el archivo de nuevo, sólo un enlace al archivo anterior idéntico que ya tiene almacenado

Los Tres Estados

Esto es lo que vas a querer recordar para que el resto de este capítulo sea una experiencia de aprendizaje sin mayores contratiempos: Git tiene tres estados en los que un archivo puede estar. Estos son: "*commited*" (confirmado), "*modified*" (modificado) y "*staged*" (preparado).

"Commited" Significa que los datos están guardados en la base de datos local.

"Modified" Significa que se han realizado cambios en el archivo pero no se ha hecho commit de ellos.

“Staged” Significa que se ha marcado un archivo modificado en su versión actual, y que será añadido al próximo commit.

Las Tres Áreas

El Directorio Git: Donde se guarda la metadata y la base de datos de objetos del proyecto. Es la parte más importante de Git, y es lo que se copia cuando se clona de un repositorio en otro equipo.

El Directorio de trabajo: Es una captura de una versión del proyecto. Estos archivos son tomados de la base de datos comprimida del Directorio Git y colocadas en disco para usarse o modificarse.

El área de preparación: Es un archivo, generalmente dentro del Directorio Git, que guarda información acerca de qué irá dentro del próximo commit. A veces se refieren a éste como el “índice”, pero se ha hecho estándar referirse al archivo como el área de preparación.

Empezar con GIT

Hay tres formas en las que se puede empezar a trabajar con git, dependiendo de qué quieres hacer. En caso de que quieras enviar (push) y recibir (pull) actualizaciones de tu proyecto que no ha sido puesto en ningún lado

```
git init
```

Si quieres trabajar sobre un repositorio existente (clonar)

```
git clone <dirección-del-repo-git>
```

Si quieres crear tu propio repositorio

```
git bare –init
```

Agregar archivos a la carpeta de trabajo GIT

```
git add <nombre del archivo 1> <nombre del archivo 2> … <nombre del archivo n>
```

Guardando los cambios

Añadir los archivos no hace más que listarlos en el directorio .git. Hay que decirle a git que esos son los archivos que debe “empujar” (push) al repo. Y suele llevar un mensaje descriptivo. De hecho, git no te permite dejar el commit sin mensaje. Esto para poder mantener una línea de tiempo de las modificaciones y las ramas.

```
git commit -a -m "mensaje"
```

Lista de cambios realizados

```
git diff
```

Seleccionar el editor predilecto

Sublime Text

¿Por qué Sublime Text? Pues, basta usarlo un par de ocasiones para darse cuenta de lo versátil que es. Basta leer un par de manuales y ver dos tutoriales para ver cuán personalizable es. Y basta instalar un par de plugins para que sea exactamente como lo quieres. Es uno de los IDE más completos que hay, y la capacidad de ampliarlo con plugins, addons (y editar éstos para ajustarlos a tus necesidades, si fuere necesario) lo hacen, sin dudas, una de las mejores opciones. Eso sin mencionar que es multiplataforma.

Trabajando con repositorios remotos

Ya aprendimos cómo hacer push y pull a un repositorio de git ya existente. Ahora, ¿Cómo

añadir un repositorio remoto? Pues...

```
git remote add <nombre del repo> <rama>
```

Clonar un repositorio remoto

```
git clone <URL-del-repo>
```

Acá se sugiere que los repositorios clonados estén en un directorio determinado. Puede ser /opt, /usr/local o lo que prefieran. Pero preferiblemente el mismo para mantener el orden.

Enviar a un repositorio remoto

```
git push <nombre del repo> <rama>
```

Sincronizar cambios con repositorio remoto

```
git merge <nombre del repo>
```

Agregar un repositorio remoto

```
git remote add <URL-del-repo>
```

Inspeccionando un repositorio remoto

```
git branch list
```

Obteniendo ayuda

```
git help
```

dpkg

Manejo de paquetes

Debian -de donde se deriva Canaima- tiene un sistema de manejo de paquetes bastante robusto. Básicamente, cualquier paquete debe pasar por una serie de requisitos para ser un paquete Debian oficial. Más adelante veremos con detalle esos requisitos, y la forma en la que Canaima ha adoptado y modificado la política de Debian para ajustarla a las necesidades puntuales de la distribución. Eso ha permitido mantener una serie de "valores agregados" como convenciones de nombres, versiones, ciclos de lanzamiento, políticas de inclusión y mantenimiento, etcétera.

Instalación manual de paquetes .deb

```
dpkg -i <nombredelpaquete>.deb
```

Búsqueda en los contenidos de los paquetes

```
dpkg -L <nombre del archivo>
```

Lista de paquetes instalados en el sistema

```
dpkg -l | grep <nombre del paquete>
```

Comparar todos los paquetes instalados entre dos maquinas

dpkg como herramienta para modificar paquetes

Descomprimir un .deb

Desde el manejador de ventanas, en el navegador de archivos, se puede acceder a un

archivo .deb y revisar sus contenidos mediante cualquier manejador de archivos comprimidos. El predeterminado en Canaima es file-roller.
Desde la línea de comandos, se puede usar midnight commander.

Descomprimir carpeta DEBIAN de un .deb

Usando mc, se accede al archivo .deb, se copia el directorio DEBIAN y se pega en el lugar que se elija.
Aunque, el "modo debian" te dice que lo más recomendable es bajar los fuentes del paquete con

```
apt-get source <nombre  del paquete>
```

Este comando descargará los fuentes del repositorio configurado e instalará unos directorios y archivos en el directorio donde se invocó al comando. Es decir, es recomendable que antes se coloque en el directorio donde se tengan los fuentes (ver arriba)

Crear un .deb

El proceso de crear un paquete en Debian -y, en consecuencia, en Canaima- es explicado con detalle en la documentación de Canaima-Desarrollador[1], que se recomienda encarecidamente leer. No obstante, supongamos que seguimos el "modo debian" y descargamos los fuentes, editamos lo que necesitábamos editar y -obviamente- no podemos firmar el paquete por no ser el mantenedor del mismo. Entonces, en ese directorio donde tenemos los fuentes del paquete (Y del que ya hemos hablado anteriormente), ejecutamos

```
debuild -us -uc
```

lo que nos creará un archivo .deb en el directorio inmediato superior. Y éste se instala como se explicó anteriormente.

[1] : http://canaima-desarrollor.readthedocs.org/en/latest/

Repositorios

Mirrors o Espejos

Es una réplica del repositorio mantenido por la comunidad, bien sea pública o privadamente.

Creación de un mirror

Instalar apt-mirror

Configurar las fuentes (sources)

Correr apt-mirror

Instalación

```
apt-get update
apt-get install apt-mirror
```

Configurar las fuentes (sources)

Debes abrir el archivo 'mirror.list' (ubicado en /etc/apt/) y agregar las lineas de los repositorios que desees tal cual como el archivo /etc/apt/sources.list. Finalmente tu archivo del mirror.list deberia quedar asi:

```
# apt-mirror configuration file
##
## The default configuration options (uncomment and change to
override)
##
#
set base_path /var/spool/apt-mirror ##AQUI COLOCO LA RUTA DONDE
#QUIERO EL REPOSITORIO
set mirror_path $base_path/mirror ##CARPETA DONDE QUEDARAN LOS
#ARCHIVOS BINARIOS
```

```
set skel_path $base_path/skel ##AQUI CREA LA ESTRUCTURA DE LO QUE
#DESCARGA
set var_path $base_path/var ##GUARDA OTROS ARCHIVOS PARA SU
#FUNCIONAMIENTO

###LAS 4 LINEAS ANTERIORES DEBEN DESCOMENTARSE
#
# set defaultarch
# set nthreads 20
#
deb http://repositorio.canaima.softwarelibre.gob.ve/ pruebas
usuarios
deb http://universo.canaima.softwarelibre.gob.ve/ lenny main
contrib non-free
deb http://seguridad.canaima.softwarelibre.gob.ve/ seguridad
usuarios
deb http://ftp.br.debian.org/debian/ lenny main contrib non-free
##
## Cleaner configuration example ###CREA UN SCRIPT DE LIMPIEZA QUE
ELIMINA LO QUE HA SIDO QUITADO DE LOS REPOSITORIOS QUE TOMAMOS
COMO ORIGEN PARA ASI NO GUARDAR NADA EXTRA
##
#
set cleanscript $var_path/clean.sh
#
# What directories should we clean up
```

```
# (i.e. remove files missing in up-to-date indexes) ##DEBEN SER
#LOS DIRECTORIOS QUE QUEREMOS
clean http://ftp.br.debian.org/
```

Correr apt-mirror

```
$ apt-mirror
```

Actualización automática del mirror

Se puede automatizar el paso 3 del punto anterior mediante un script de bash y colocarlo en cron

Manejar tu propio repositorio

Instalar y configurar reprepro

En ocasiones necesitamos disponer de un repositorio de paquetes propio con el que instalar una serie de paquetes que no se encuentran en los repositorios de nuestra organización ni en los de nuestra distribución. En este caso, podemos recurrir a reprepro.

Vamos a ver cómo podemos instalar y utilizar reprepro en nuestra organización.

Instalar reprepro.

Lo primero es instalar los paquetes reprepro y apache2:

```
apt-get install reprepro apache2
```

Reprepro necesita que tengamos apache para distribuir los paquetes del repositorio a los clientes.

Una vez instalados los paquetes, creamos el directorio principal de nuestro repositorio y el directorio de configuración de una vez. Por ejemplo:

```
mkdir -p /var/www/iesvalledeljerte3/conf
```

A continuación vamos a crear una clave con la que firmaremos el repositorio:

```
cd /var/www/repositorio
gpg --gen-key
```

- gpg nos pedirá que seleccionemos el tipo de clave que deseamos crear. Elegiremos la opción (4) RSA (sólo firmar).
- Lo siguiente que nos preguntará gpg es de qué tamaño queremos crear la clave y por defecto nos ofrece un tamaño de 2048. Lo aceptamos.
- Nos preguntará por el período de validez de la clave y por detecto nos ofrecerá la opción 0 = la clave nunca caduca. Elegimos la opción por defecto.
- Nos preguntará si es correcto. Respondemos que sí (s).
- Nos preguntará el nombre y apellidos. Introducimos nuestro nombre y apellidos.
- Nos pedirá que introduzcamos nuestra dirección de correo. Así que la introducimos.
- Nos ofrecerá la posibilidad de poner un comentario. Pulsamos Enter para dejarlo vacío.
- Para terminar, suponiendo que hemos introducido todos los datos correctamente, pulsaremos (V) de (V)ale.
- Y, ya por último, nos pedirá que introduzcamos una frase de contraseña. La introducimos y repetimos cuando nos lo pida.

Al final nos mostrará la clave generada. Si se nos ha escapado por lo que sea y no la hemos visto, ejecutamos:

```
$ gpg --list-keys
/home/enam0000/.gnupg/pubring.gpg
---------------------------------
pub   1024D/153F5386 2009-12-01
uid                  Esteban M. Navas Martín (Administrador
```

```
Informatico)
sub   2048g/83061C03 2009-12-01
```

- Copiamos el identificador de la clave pública para introducirlo posteriormente en el fichero /var/www/iesvalledeljerte3/conf/distributions. En mi caso: 153F5386
- Exportamos la clave pública en un archivo y la copiamos en el directorio principal del repositorio:

```
$ gpg  --armor --export 153F5386 >
/var/www/iesvalledeljerte3/iesvalledeljerte3.asc
```

- A continuación creamos un fichero donde vamos a definir las distribuciones con las que vamos a trabajar:

```
nano /var/www/iesvalledeljerte3/conf/distributions
```

Debería quedar algo así:

```
Origin: IES Valle del Jerte
Label: Debian Squeeze packages
Suite: squeeze
Codename: squeeze
Architectures: i386 amd64
Components: main
Description: Paquetes adicionales para el IES
DebIndices: Packages Release . .gz .bz2
SignWith: 153F5386
```

- Cuando tengamos todo lo anterior, entramos en el directorio del repositorio:

```
cd /var/www/iesvalledeljerte3
```

- Y ejecutamos

```
reprepro -VVV export
```

- A continuación creamos los enlaces simbólicos:

```
reprepro -VVV createsymlinks
```

Agregar paquetes binarios a nuestro repositorio

Para agregar paquetes binarios (.deb), nos situamos en la carpeta raíz del repositorio (siguiendo el ejemplo: /var/www/iesvalledeljerte3) y ejecutamos el siguiente comando:

```
reprepro --ask-passphrase includedeb [DISTRIBUCION] [PAQUETE]
```

Nos preguntará por la frase de paso. La introducimos y al terminar incluirá nuestro paquete binario en el repositorio.

Eliminar paquetes binarios del repositorio

Eliminar un paquete de una rama de nuestro repositorio es sencillo:

```
reprepro --ask-passphrase remove [DISTRIBUCION] [PAQUETE]
```

Usar el repositorio en nuestros clientes

Una vez hecho todo lo anterior, ya podemos usar el repositorio en nuestros clientes.

Creamos un archivo en el cliente para nuestro repositorio:

```
nano /etc/apt/sources.list.d/iesvalledeljerte3.list
```

Con el siguiente contenido:

```
deb http://servidor/repositorio [DISTRIBUCIÓN] [COMPONENTES]
```

Seguidamente descargamos la clave pública de nuestro repositorio, que ya subimos al servidor en el cliente y la añadimos mediante apt-key:

```
wget http://servidor/iesvalledeljerte3.asc
apt-key add iesvalledeljerte3.asc
```

Por último, haremos un apt-get update en el cliente para actualizar la lista de paquetes y ya podremos instalar lo que queramos desde nuestro repositorio.

Listar los paquetes disponibles en nuestro repositorio

Si queremos ver la lista de paquetes que tenemos agregados en nuestro repositorio, tan sólo tenemos que ejecutar:

```
reprepro list "rama-de-repositorio"
```

Agregar paquetes sin sección o prioridad a nuestro repositorio

A veces queremos añadir ciertos paquetes a nuestro repositorio y éstos carecen de información de sección a la que pertenecen o prioridad, por ejemplo. O incluso, puede que tengan dicha información pero queremos cambiarla.

Para especificar la sección a la que queremos añadir un paquete, usamos el parámetro -S o --section. Por ejemplo: Imaginemos que quiero añadir los paquetes de libreoffice descargados de la web oficial de LibreOffice, que carecen de información de sección:

```
reprepro --ask-passphrase -S main includedeb squeeze
/var/www/descargas/libreoffice4/*.deb
```

Y si además, quiero cambiar la prioridad, usaré el parámetro -P o --priority. Por ejemplo: Imaginemos que, además de cambiar la sección quiero asignar una prioridad de 600 a los paquetes que voy a añadir a mi repositorio. No tendría más que hacer lo siguiente:

```
reprepro --ask-passphrase -S main -P 600 includedeb squeeze
/var/www/descargas/libreoffice4/*.deb
```

Utilizando el repositorio creado con reprepro

Basta añadir al sources.list la línea con el repositorio local y los datos generados.

CANAIMA-DESARROLLADOR

DESCRIPCIÓN:

Canaima Desarrollador (C-D) es un compendio de herramientas y ayudantes que facilitan el proceso de desarrollo de software para Canaima GNU/Linux. Está diseñado para facilitar el trabajo a aquellas personas que participan en dicho proceso con regularidad,

como también para iniciar a los que deseen aprender de una manera rápida y práctica.

C-D puede ayudarte a:

Agilizar los procesos para la creación de paquetes binarios canaima a partir de paquetes fuentes correctamente estructurados.

Automatización personalizada de la creación de Paquetes Fuentes acordes a las Políticas de Canaima GNU/Linux.

Creación de un depósito personal, por usuario, donde se guardan automáticamente y en carpetas separadas los siguientes tipos de archivo:

• Proyectos en proceso de empaquetamiento

• Paquetes Binarios (*.deb)

• Paquetes Fuente (*.tar.gz, *.dsc, *.changes, *.diff)

• Registros provenientes de la creación de paquetes binarios (*.build)

Versionamiento asistido (basado en git) en los proyectos, brindando herramientas para realizar las siguientes operaciones, con un alto nivel de automatización y detección de posibles errores:

• git clone

• git commit

• git push

• git pull

Ejecución de tareas en masa (empaquetar, hacer pull, push, commit, entre otros), para agilizar procesos repetitivos.

AYUDANTES DE CANAIMA DESARROLLADOR

Crear Proyecto / Debianizar

Crear Fuente

Empaquetar

Descargar

Registrar

Enviar

Actualizar

Descargar Todo

Registrar Todo

Enviar Todo

Actualizar Todo

Empaquetar Varios

Empaquetar Todo

Listar Remotos

Listar Locales

CREAR PROYECTO / DEBIANIZAR

Crea un proyecto de empaquetamiento desde cero o debianiza uno existente.

USO

```
canaima-desarrollador crear-proyecto|debianizar --nombre=""
                    --version="" --destino=
```

PARÁMETROS

--nombre Un nombre para tu proyecto, que puede contener letras, números, puntos y guiones. Cualquier otro caracter no está permitido.

--version La versión inicial de tu proyecto. Se permiten números, guiones, puntos, letras o dashes (~).

--destino Especifica si es un proyecto de empaquetamiento para Canaima GNU/Linux o si es un proyecto personal. Las opciones disponibles son “canaima” y “personal”.

--licencia Especifica el tipo de licencia bajo el cuál distribuirás tu trabajo. Las licencias soportadas son: apache, artistic, bsd, gpl, gpl2, gpl3, lgpl, lgpl2 y lgpl3.

--ayuda Muestra la documentación para el ayudante. Si estás debianizando un proyecto existente, lo que ingreses en –nombre=”proyecto” y –version=”X.Y+Z” se utilizará para determinar cuál es el nombre de la carpeta a debianizar dentro del directorio del desarrollador, suponiendo que tiene el nombre proyecto-X.Y+Z. Si no se llama así, habrá un error.

CREAR FUENTE

Crea un paquete fuente a partir de un proyecto de empaquetamiento existente. El resultado es guardado en el depósito de fuentes.

USO

```
canaima-desarrollador crear-fuente --directorio="" [--ayuda]
```

PARÁMETROS

--directorio Nombre del directorio dentro de la carpeta del desarrollador donde se encuentra el proyecto. El directorio debe contener un proyecto debianizado.

--ayuda Muestra la documentación para el ayudante.

EMPAQUETAR

Éste ayudante te permite empaquetar un proyecto de forma automatizada, siguiendo la metodología git-buildpackage, que se centra en el siguiente diagrama:

COMMIT > REFLEJAR CAMBIOS EN EL CHANGELOG > COMMIT > CREAR PAQUETE FUENTE > PUSH > GIT-BUILDPACKAGE

USO

```
canaima-desarrollador empaquetar --directorio="" --mensaje=""
                    --procesadores="" [--ayuda]
```

PARÁMETROS

--directorio Nombre de la carpeta dentro del directorio del desarrollador donde se encuentra el proyecto a empaquetar.

--mensaje Mensaje representativo de los cambios para el primer commit. El segundo commit es sólo para el changelog. Colocando la palabra “auto” o dejando el campo vacío, se autogenera el mensaje.

--procesadores Número de procesadores con que cuenta tu computadora para optimizar el proceso de empaquetamiento.

--ayuda Muestra la documentación para el ayudante.

DESCARGAR

Éste ayudante te permite copiar a tu disco duro un proyecto que se encuentre en el repositorio remoto para que puedas modificarlo según consideres. Utiliza git clone para realizar tal operación. Éste ayudante se encarga además de realizar las siguientes operaciones por ti:

Verifica e informa sobre el éxito de la descarga.

USO

```
canaima-desarrollador descargar --proyecto="" [--ayuda]
```

PARÁMETROS

--proyecto Nombre del proyecto (en caso de que éste se encuentre en el repositorio de Canaima GNU/Linux) o la dirección git pública del proyecto.

--ayuda Muestra la documentación para el ayudante.

REGISTRAR

Éste ayudante te permite registar (o hacer commit de) los cambios hechos en un proyecto mediante el versionamiento basado en git. Utiliza git commit para lograr éste propósito. Éste ayudante se encarga además de realizar las siguientes operaciones por ti:

Verifica la existencia de la rama git “upstream”. En caso de no encontrarla, la crea.

Verifica la existencia de la rama git “master”. En caso de no encontrarla, la crea.

Verifica la existencia de todos los elementos necesarios para ejecutar la acción git commit (carpetas, variables de entorno, etc..). En caso de encontrar algún error, aborta e informa.

Autogenera el mensaje de commit, si se le instruye.

Hace git checkout a la rama master, si nos encontramos en una rama diferente a la hora de hace commit.

Hace un git merge de la rama master a la upstream, inmediatamente depués del commit.

USO

```
canaima-desarrollador registrar --directorio="" --mensaje=="" [--
                                ayuda]
```

PARÁMETROS

--directorio Nombre de la carpeta dentro del directorio del desarrollador a la que se quiere hacer commit.

--mensaje Mensaje representativo de los cambios para el commit. Colocando la palabra “auto” o dejando el campo vacío, se autogenera el mensaje.

--ayuda Muestra la documentación para el ayudante.

ENVIAR

Éste ayudante te permite enviar los cambios realizados al repositorio remoto especificado en las configuraciones personales, mediante el uso de la acción git push. Éste ayudante se encarga además de realizar las siguientes operaciones por ti:

Verifica la existencia de la rama git “upstream”. En caso de no encontrarla, la crea.

Verifica la existencia de la rama git “master”. En caso de no encontrarla, la crea.

Verifica la existencia de todos los elementos necesarios para ejecutar la acción git push (carpetas, variables de entorno, etc..). En caso de encontrar algún error, aborta e informa.

Configura el repositorio remoto para el proyecto, de acuerdo a los parámetros establecidos

en ~/.config/canaima-desarrollador/usuario.conf

USO

```
canaima-desarrollador enviar --directorio="" [--ayuda]
```

PARÁMETROS

--directorio Nombre de la carpeta dentro del directorio del desarrollador a la que se quiere hacer push.

--ayuda Muestra la documentación para el ayudante.

ACTUALIZAR

Éste ayudante te permite actualizar el código fuente de un determinado proyecto, mediante la ejecución de “git pull” en la carpeta del proyecto. Éste ayudante se encarga además de realizar las siguientes operaciones por ti:

Verifica la existencia de la rama git “upstream”. En caso de no encontrarla, la crea.

Verifica la existencia de la rama git “master”. En caso de no encontrarla, la crea.
Verifica la existencia de todos los elementos necesarios para ejecutar la acción git pull (carpetas, variables de entorno, etc..). En caso de encontrar algún error, aborta e informa.
Configura el repositorio remoto para el proyecto, de acuerdo a los parámetros establecidos en ~/.config/canaima-desarrollador/usuario.conf
USO

```
canaima-desarrollador actualizar --directorio="" [--ayuda]
```

PARÁMETROS
--directorio Nombre de la carpeta dentro del directorio del desarrollador a la que se quiere hacer git pull.
--ayuda Muestra la documentación para el ayudante.

DESCARGAR TODO
Éste ayudante te permite copiar a tu disco duro todos los proyectos de Canaima GNU/Linux que se encuentren en el repositorio remoto oficial. Utiliza git clone para realizar tal operación.
USO

```
canaima-desarrollador descargar-todo [--ayuda]
```

PARÁMETROS
--ayuda Muestra la documentación para el ayudante.

REGISTRAR TODO
Éste ayudante te permite registar (o hacer commit de) todos los cambios hechos en todos los proyectos existentes en la carpeta del desarrollador. Utiliza git commit para lograr éste propósito. Asume un mensaje de commit automático para todos.
USO

```
canaima-desarrollador registrar-todo [--ayuda]
```

PARÁMETROS
--ayuda Muestra la documentación para el ayudante.

ENVIAR TODO

Éste ayudante te permite enviar todos los cambios realizados en todos los proyectos ubicados en la carpeta del desarrollador al repositorio remoto especificado en las configuraciones personales, mediante el uso de la acción git push.

USO

```
canaima-desarrollador enviar-todo [--ayuda]
```

PARÁMETROS

--ayuda Muestra la documentación para el ayudante.

ACTUALIZAR TODO

Éste ayudante te permite actualizar el código fuente de todos los proyectos ubicados en la carpeta del desarrollador, mediante la ejecución de “git pull” en la carpeta del proyecto.

USO

```
canaima-desarrollador actualizar-todo [--ayuda]
```

PARÁMETROS

--ayuda Muestra la documentación para el ayudante.

EMPAQUETAR VARIOS

Éste ayudante te permite empaquetar varios proyectos.

USO

```
canaima-desarrollador empaquetar-varios --para-empaquetar=""
                    --procesadores="" [--a
```

PARÁMETROS

--para-empaquetar Lista de los directorios dentro de la carpeta del desarrollador que contienen los proyectos que se quieren empaquetar, agrupados entre comillas.

--procesadores Número de procesadores con que cuenta tu computadora para optimizar el proceso de empaquetamiento.

--ayuda Muestra la documentación para el ayudante.

EMPAQUETAR TODO

Éste ayudante te permite empaquetar todos los proyectos existentes en la carpeta del desarrollador.

USO

```
canaima-desarrollador empaquetar-todo --procesadores="" [--ayuda]
```

PARÁMETROS

--procesadores Número de procesadores con que cuenta tu computadora para optimizar el proceso de empaquetamiento.

--ayuda Muestra la documentación para el ayudante.

LISTAR REMOTOS

Muestra todos los proyectos contenidos en el repositorio remoto y muestra su dirección git.

USO

```
canaima-desarrollador listar-remotos [--ayuda]
```

PARÁMETROS

--ayuda Muestra la documentación para el ayudante.

LISTAR LOCALES

Muestra todos los proyectos contenidos en la carpeta del desarrollador y los clasifica según su tipo.

USO

```
canaima-desarrollador listar-locales [--ayuda]
```

PARÁMETROS

--ayuda Muestra la documentación para el ayudante.

Canaima Semilla

Canaima Semilla es una aplicación diseñada para facilitar a individuales, colectivos e instituciones la creación de distribuciones GNU/Linux personalizadas y adaptadas a sus necesidades (también llamadas "sabores"), partiendo de la Metadistribución Canaima GNU/Linux. Actualmente existen varios sabores de Canaima:

Canaima Primera Base: Sabor con aplicaciones básicas, desde el cuál se pueden instalar el resto de los sabores.
Canaima Popular: Sabor ligero de fácil distribución, con aplicaciones de uso común.
Canaima Institucional: Sabor utilizado en la Administración Pública Nacional venezolana para fortalecer la Soberanía Tecnológica dentro del estado.
Canaima Educativo: Proyecto educativo liberador para los niños de la educación primaria.
Canaima Forense: Sabor que facilita la investigación forense en sistemas informáticos.
Canaima Colibrí: Sabor destinado a optimizar el rendimiento del sistema en computadoras de baja capacidad.
Mediante una serie de pasos, podrás crear una imagen instalable de los sabores antes mencionados, o si lo prefieres, crear tu propio Sabor Canaima.

Instalación

Canaima Semilla se encuentra en los repositorios de las versiones 3.0, 3.1, 4.0 y 4.1 de Canaima. Si utilizas Canaima, puedes instalarlo de la siguiente manera:

```
aptitude update
aptitude install canaima-semilla
```

Si no utilizas Canaima, deberás incluir el repositorio de Canaima antes de ejecutar los comandos de instalación. Abre el archivo /etc/apt/sources.list con tu editor de textos preferido (con permisos de root) y modifícalo para agregar la siguiente línea:

```
deb http://paquetes.canaima.softwarelibre.gob.ve/ estable main
```

```
aportes no-libres
```

Configuración

En Canaima Semilla vienen incorporados algunos perfiles de configuración para algunos de los sabores de Canaima. Si se desean crear nuevos sabores, es necesario crear nuevos perfiles, y para ello, es necesario saber el método para crearlos.

Un perfil está compuesto de varios archivos con nombres específicos colocados dentro de una carpeta que lleva por nombre el nombre del sabor en minúsculas y sin caracteres especiales. Dentro de la carpeta deben estar los siguientes archivos y carpetas (algunos son opcionales):

(Obligatorio) Un archivo llamado profile.conf. Este archivo contiene variables de configuración para la construción del sabor, entre las que podemos mencionar:

```
PROFILE_NAME="popular"
PROFILE_ARCH="i386 amd64"
AUTHOR_NAME="Equipo de Desarrollo del Proyecto Canaima"
AUTHOR_EMAIL="desarrolladores@canaima.softwarelibre.gob.ve"
AUTHOR_URL="http://canaima.softwarelibre.gob.ve/"
META_DISTRO="canaima"
META_CODENAME="auyantepui"
META_REPO="http://paquetes.canaima.softwarelibre.gob.ve/"
META_REPOSECTIONS="main aportes no-libres"
OS_LOCALE="es_VE.UTF-8"
OS_PACKAGES="canaima-base canaima-escritorio-base canaima-
escritorio-gnome"
OS_EXTRAREPOS="profile"
OS_INCLUDES="profile"
```

```
OS_HOOKS="profile"
IMG_SYSLINUX_SPLASH="profile"
IMG_POOL_PACKAGES="burg"
IMG_INCLUDES="profile"
IMG_HOOKS="profile"
IMG_DEBIAN_INSTALLER="true"
IMG_DEBIAN_INSTALLER_BANNER="profile"
IMG_DEBIAN_INSTALLER_PRESEED="profile"
IMG_DEBIAN_INSTALLER_GTK="profile"
```

(Opcional) Una imagen PNG llamada "syslinux.png" de una dimensión no mayor a 1024x768 pixeles, la cuál servirá de fondo en el menú de inicio del Medio Vivo.
Canaima Semilla es una herramienta para generar distribuciones derivadas de Canaima.Canaima Semilla es una herramienta para generar distribuciones derivadas de Canaima.

(Opcional) Una imagen PNG llamada "banner-instalador.png" de una dimensión exacta de 800x75 pixeles, la cuál será el banner del dialogo del instalador del Medio Vivo.
Canaima Semilla es una herramienta para generar distribuciones derivadas de Canaima.Canaima Semilla es una herramienta para generar distribuciones derivadas de Canaima.

(Opcional) Un archivo de configuración GTKRC llamado "gtkrc-instalador", el cuál albergará los parámetros GTK para modificar la apariencia del instalador. Ver el sabor de ejemplo.

(Opcional) Un par de archivos para definir repositorios extra en la etapa de instalación de paquetes finales (BINARY):

Uno de extensión *.binary (pudiendo tener cualquier nombre), que contenga una lista de repositorios extra necesarios para la instalación de paquetes no incluídos en MIRRORDEBIAN y especificados en SABORPAQUETES.
Ejemplo: canaima.binary

```
deb http://repositorio.canaima.softwarelibre.gob.ve/ pruebas usuarios
deb http://seguridad.canaima.softwarelibre.gob.ve/ seguridad usuarios
```

Otro de extensión *.binary.gpg, conteniendo la (o las) llave(s) GPG válida(s) correspondientes a los repositorios listados en el archivo *.binary.
(Opcional) Un par de archivos para definir repositorios extra en la etapa de instalación del sistema base inicial (CHROOT):
Uno de extensión *.chroot (pudiendo tener cualquier nombre), que contenga una lista de repositorios extra necesarios para la instalación de paquetes no incluídos en MIRRORDEBIAN y especificados en SABORPAQUETES.
Ejemplo: canaima.chroot

deb http://repositorio.canaima.softwarelibre.gob.ve/ pruebas usuarios
deb http://seguridad.canaima.softwarelibre.gob.ve/ seguridad usuarios

Otro de extensión *.chroot.gpg (con nombre igual al anterior), conteniendo la (o las) llave(s) GPG válida(s) correspondientes a los repositorios listados en el archivo *.chroot.
(Opcional) Un archivo llamado "preseed-debconf.cfg" en donde se incluirán los parámetros debconf que se quieran modificar en el modo nVivo del medio instalable.

(Opcional) Un archivo llamado "preseed-instalador.cfg" en donde se incluirán los parámetros debconf a modificar en el instalador.

Se provee en la dirección de los perfiles (/usr/share/canaima-semilla/perfiles) un perfil de

ejemplo, el cuál podrá ser utilizado como base para nuevos sabores. La ausencia de alguno de los archivos Opcionales causará que Canaima Semilla use los valores por defecto (Debian).

Los perfiles se definen en la carpeta "/usr/share/canaima-semilla/perfiles", para la cual debes tener permisos de superusuario si deseas editarla. La mejor forma de crear un nuevo sabor, es duplicar la carpeta de ejemplo y comenzar a editar sus archivos hasta obtener el resultado esperado.

Canaima Semilla puede crear imágenes instalables (ISO o IMG) basado en los perfiles de sabores existentes. Puede especificársele el tipo de Medio, la arquitectura a construir, y el sabor. Las imágenes resultantes del proceso de construcción se guardan en el directorio /usr/share/canaima-semilla/semillero/.

USO

```
c-s construir --medio="iso|usb" --arquitectura="i386|amd64"
--sabor="institucional|popular|primera-base|sabor1|sabor2" [--
ayuda]
```

PARÁMETROS

--medio: Tipo de imagen que será generada. Coloca "iso" para una imagen ISO grabable en CD/DVD o "usb" para una imagen IMG grabable en dispositivos USB.

--arquitectura: Arquitectura soportada por la imagen resultante. Canaima GNU/Linux soporta i386 y amd64.

--sabor: Sabor Canaima contenido en la imagen instalable. Debe estar definida en el directorio de los perfiles para poder funcionar.

--ayuda: Muestra la documentación para el ayudante.

Por ejemplo, para construir el sabor "popular", se utiliza la siguiente línea:

```
c-s construir --medio="iso" --arquitectura="i386"
--sabor="popular"
```

¿Y cómo creo un Sabor Canaima?

Canaima Semilla facilita la creación de Sabores Canaima mediante el establecimiento de reglas o perfiles que definen los componentes que integran el sabor. Un perfil está compuesto de varios archivos con nombres específicos colocados dentro de una carpeta que lleve por nombre el nombre del sabor en minúsculas. La. La carpeta contendrá:

Sugerencias

Optimizar la estructura de paquetes del sabor a construir. Lo ideal es que los paquetes se encuentren organizados y agrupados en metapaquetes, de forma tal de que con incluir unos pocos paquetes en SABOR_PAQUETES, todo el árbol de dependencias sea incluído.

No utilizar scripts de configuración. Toda configuración adcional que se desee realizar al medio vivo, debe ser incorporado en paquetes dentro de su respectivo postinst.

REFERENCIAS

http://git-scm.com/book/en/Getting-Started-Git-Basics
http://mundo-linux-b.blogspot.com/2013/05/crear-tu-propio-repositorio-de-paquetes.html
http://flamelcanto.blogspot.com/2010/07/apt-mirror.html

http://huntingbears.com.ve/canaima-semilla-herramienta-para-la-creacion-y-distribucion-de-sabores-canaima.html