fedora™ -LATAM
magazine
Una revista para Latinoamericanos
hecha por latinoamericanos
Nº 0006
Año 01

Ruby
Capitulo I:
El primer contacto

Configurando un servicio y cliente NFS

¿Puede la creación rápida de prototipos funcionar para su proyecto creativo?

fedora

PREGUNTA

RESPONDE

COMPARTE

Interactúa con la comunidad Fedora

Visita el canal IRC #fedora-latam

http://proyectofedora.org/chat

5 Ruby
Capitulo I:
El primer contacto

10 Configurando un servicio y cliente NFS

17 ¿Puede la creación rápida de prototipos funcionar para su proyecto creativo?

EDITORIAL

Maria "tatica" Leandro
http://fedoraproject.org/wiki/User:tatica

Ser parte de un proyecto en el que eres capaz de ayudar de cierta forma al mundo, es posiblemente la mejor forma de gastar tu tiempo libre.

A veces nos llevamos al límite solo para demostrarnos a nosotros mismos que sí somos capaces de asumir retos que parecen imposibles por falta de experiencia o simple falta de tiempo; mas aún así, nos unimos en comunidad y llegamos a la meta deseada.

En los pocos años que tengo en la comunidad Fedora he aprendido que hay retos que cumplir, metas que lograr para poder decir al final del día "el proyecto se ha logrado". Es un placer ver como mes a mes y con mucho esfuerzo, proyectos que se plantearon hace unos 4 años estan viendo la luz. Así mismo, ver que LATAM ha logrado expandirse, conseguir nuevos líderes y crear alianzas profesionales con otras regiones es una satisfacción que solo se mide con el éxito de cada release y los millones de usuarios y colaboradores satisfechos por el esfuerzo común.

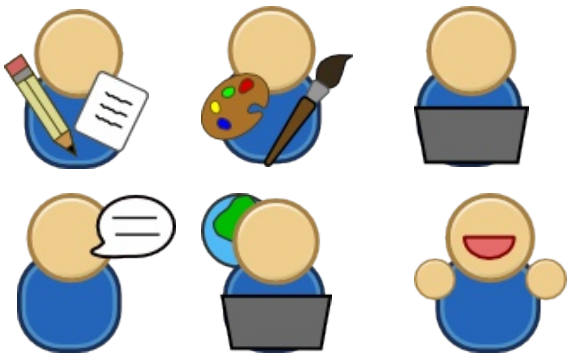
Siempre hay obstáculos, siempre hay una piedra en el camino que te hace caer (y a veces sangrar un poco), siempre hay alguien que criticará tu trabajo, alguien que te dirá que hay una mejor forma de hacerlo e incluso, habrán muchos que simplemente se quedarán callados cuando pidas opinión. El trabajo está en tener fortaleza para continuar, sacar lo bueno de cada crítica, motivar a aquellos que encuentran fallas a que se unan al equipo y las solucionen en comunidad. La abstención no es una característica de la comunidad.

Esta pequeña editorial, pueden considerarla como una carta abierta de alguien que ha recibido muchísimas críticas (constructivas y desctructivas) pero que aún así, pone su grano de arena porque sabe que hay muchas otras personas que se abstienen de colaborar quizás solo por miedo al fracaso. Lo malo no es equivocarse, es no ser valiente para seguir intentando y como todos sabemos, la práctica hace al maestro y la perseverancia lleva a la perfección.

Lo interesante de ser parte de Fedora es que siempre habrá una mano amiga dispuesta a ayudarte, siempre habrá un proyecto al que te puedas integrar y que hará que saques lo mejor de ti; y siempre habrá alguien (en cualquier lugar del mundo) que valorará ese grano de arena en particular que tu pusiste. Ser altruista tiene sus beneficios (también sus sacrificios), pero que bien se siente ser parte de un proyecto exitoso como Fedora.

¿Aún no sabes como ser parte de Fedora? Ve a nuestra sección de Join y conoce mas sobre los equipos que hacen al proyecto una realidad.

http://fedoraproject.org/wiki/Join/es



Ruby: Capítulo 1

Guillermo Gómez Savino
http://fedoraproject.org/wiki/User:gomix

"*Yo quería un lenguaje de programación más poderoso que Perl y más orientado a objetos que Python. Entonces me acordé de mi viejo sueño y decidí diseñar mi propio lenguaje. Al principio estuve jugando con él en el trabajo. Gradualmente creció lo suficiente como para remplazar a Perl. Lo llamé Ruby en honor a esa piedra preciosa roja y lo liberé al público en 1995.*"

Yukihiro Matsumoto, a.k.a. ``Matz''
Japón, Octubre 2000

Aprovechando el espacio

Para comenzar en Fedora con Ruby vamos instalar lo mínimo necesario y sin perder mucho espacio dando vueltas con teoría y opiniones, directo al grano. Abra una sesión de emulación de terminal preferida y siga las siguientes instrucciones para instalar. En la medida que desarrollemos esta columna dedicada a Ruby, entonces iremos descubriendo el poder y flexibilidad de Ruby.

```
$ su -
<contraseña de root>
# yum install ruby ruby-rdoc ruby-ri
```

Para el editor, hay muchas opciones, mi editor de preferencia es Vim , pero puede usar el de su preferencia, intente usar alguno que pueda resaltar sintaxis Ruby como mínimo. Puede escoger desde entornos tan complejos y completos como Eclipse, hasta editores de escritorio GUI como Gedit, o simples en consola como nano o complejos y sofisticados como Vim y Emacs.

Los "Hola mundo"

La primera forma interactiva simple de ejecutar comandos Ruby es simplemente usar el intérprete, simplemente ejecute el intérprete, ingrese los comandos y termine presionando Ctrl-D para indicarle al intérprete que la entrada de comandos ha finalizado:

```
$ ruby
puts "Hola Mundo"
<Ctrl-D>
Hola Mundo
```

La segunda forma interactiva es con irb. irb es el acrónimo para Interactive Ruby. irb es un shell Ruby, es decir, es un espacio donde puede evaluar su código al instante. En próximas ediciones iremos desarrollando más el tema de irb, por ahora simplemente invoque a irb e intente:

```
$ irb
irb(main):001:0> puts "Hola Mundo"

Hola Mundo
=> nil
irb(main):002:0>exit
$
```

Si lo que quiere es crear un programa Ruby que nos imprima "Hola Mundo" en la salida del monitor, lance su editor e incluya el siguiente código fuente en un archivo denominado **holamundo.rb**., guarde y salga de su editor

holamundo.rb

```
1 puts "Hola Mundo"
```

Para ejecutar simplemente pásale al intérprete Ruby el archivo como argumento.

```
[gomix@fricky capitulo_1]$ ruby holamundo.rb
Hola Mundo
```

También puede usar el método "shebang" y convertir el archivo fuente Ruby en ejecutable del sistema, edite su holamundo.rb para que luzca como se muestra en el listado a continuación.

holamundo.rb

```
1 #!/usr/bin/ruby
2
3 puts "Hola Mundo"

[gomix@fricky capitulo_1]$ chmod +x holamundo.rb

[gomix@fricky capitulo_1]$ ./holamundo.rb
Hola Mundo
```

Existe la forma de pasarle directamente código Ruby al intérprete sin apoyo de archivos o programas adicionales y sin entrar en modo interactivo.

```
$ ruby -e 'puts "Hola Mundo"'
Hola Mundo
```

Ruby es un lenguaje de programación orientado a objetos

Todo lo que usted manipula en Ruby es un objeto, y los resultados de dichas manipulaciones a su vez, también son objetos. Cuando usted escribe código orientado a objetos normalmente está modelando conceptos del mundo real en su código. Típicamente durante este proceso de modelado usted descubrirá categorías de cosas que necesitan ser representadas en código. En un reproductor de música el concepto de "canción" puede ser una de esas categorías. En Ruby usted define una clase para representar cada una de esas entidades. Una clase es una combinación de estado, por ejemplo el nombre de la canción, y métodos que usan dicho estado, por ejemplo para reproducir la canción.

Una vez que tiene dichas clases usted creará instancias de dicha clase. Para el reproductor de música que contiene la clase Cancion usted terminará teniendo instancias separadas independientes para canciones populares como "Todo o nada", "Canción para un amigo", "Amanecer llanero", y por el estilo. La palabra objeto e instancia son intercambiables. En Ruby para crear dichas instancias u objetos, se debe llamar a un método constructor de la clase, el método constructor estandar se llama new.

```
1 cancion1 = Cancion.new("Amanecer llanero")

2 cancion2 = Cancion.new("Todo o nada")
```

Estas instancias son derivadas de la misma clase pero tienen características únicas. Primero, cada objeto tiene un object_id único. Segundo, usted puede definir variables de instancia, variables con valor que son únicos para cada instancia. Estas variables de instancia mantienen el estado del objeto. Igualmente puede definir métodos de instancia para acceder y/o alterar el estado del objeto, es decir, acceder y/o alterar las variables de instancia. Rápidamente definamos nuestra clase Cancion. Ruby se puede leer fácilmente.

```
1 class Cancion
2   def titulo
3     @titulo
4   end
5
```

```
6   def titulo=(titulo_de_la_cancion)
7     @titulo = titulo_de_la_cancion.to_s
8   end
9 end
```

Claramente podemos ver y leer que hemos definido la clase de nombre **Cancion** con dos métodos de instancia, **titulo y titulo=**. La variable de instancia se representa en @titulo , con la notación de nombre de variable en minúscula precedida del símbolo @ .

Note la indentación que hemos implementado para representar la estructura, dos espacios es cómun entre los Rubyeros (me gusta llamarlos así).

Es evidente que la plabra clave **class** establece el inicio de un bloque que se cierra con **end** para definir una clase con nombre, en este ejemplo **Cancion**. Dentro de dicha estructura también podemos identificar claramente la palabra clave **def** que igualmente define bloques que se cierran también con la palabra clave **end**. **def** define dos métodos de instancia en este ejemplo. Hablaremos más de **def** de forma recurrente en muchas ediciones de esta columna.

Podemos probar nuestra clase en irb fácilmente, por ahora tipee con cuidado para no equivocarse, luego le ofreceré más técnicas irb.

```
1 $ irb
2 >> class Cancion
3 >> def titulo
4 >> @titulo
5 >> end
6 >> def titulo=(titulo_de_la_cancion)
7 >> @titulo =
          titulo_de_la_cancion.to_s
8 >> end
9 >> end
10 => nil
11 >> cancion1 = Cancion.new
12 => #<Cancion:0xb76e25c8>
13 >> cancion1.titulo="Alma llanera"
14 => "Alma llanera"
15 >> cancion1.titulo
16 => "Alma llanera"
```

Pero no necesito clases

También se puede usar Ruby del modo procedimental, funcional, sin necesidad de crear (explícitamente) clases y objetos.

```
1 $ irb
2 >> 2 + 2
3 => 4
```

Pero igual note que estamos trabajando con objetos de alguna clase, en nuestro ejemplo Fixnum.

```
1 >> 2.class
2 => Fixnum
```

Puede definir métodos y llamarlos:

```
1 >> def saludo
2 >> puts "Hola Mundo"
3 >> end
4 => nil
5 >> saludo
6 Hola Mundo
7 => nil
```

Clases base

Ya que "todo" son objetos de alguna clase, más vale que comencemos por aprender las más fundamentales del lenguaje, Fixnum, Bignum, Float, String, Array y Hash.

Números... Fixnum, Bignum, Float

Enteros, dejemos que nuestro código Ruby hable.

```
1 >> num = 8
2 => 8
3 >> 7.times do
4 ?> print num.class, " ", num, "\n"
5 >> num *= num
6 >> end
7 Fixnum 8
8 Fixnum 64
9 Fixnum 4096
10 Fixnum 16777216
11 Bignum 281474976710656
12 Bignum
   79228162514264337593543950336
13 Bignum
   627710173538668076383578942320766641610235544446403451289 6
14 => 7
15 >> num = 3.14
16 => 3.14
17 >> 8.times do
18 ?> print num.class, " ", num, "\n"
19 >> num *= num
20 >> end
21 Float 3.14
22 Float 9.8596
23 Float 97.21171216
24 Float 9450.11698107869
25 Float 89304710.9560719
26 Float 7.97533139894754e+15
27 Float 6.36059109230385e+31
28 Float 4.04571190434951e+63
```

Cadenas de caracteres, String

```
1 >> "Hola Mundo".class
2 => String
3 >> "987".class
4 => String
<
```

Arreglos, Array

```
1 >> arreglo = ["a",1,"b",2]
2 => ["a", 1, "b", 2]
3 >> arreglo.class
4 => Array
5 >> arreglo[0]
6 => "a"
7 >> arreglo[1]
8 => 1
<
```

Arreglos indexados arbitrariamente, Hash

```
1 >> hash = { "color" => "rojo",
   "temperatura" => 75, 1 => "hoy"}
2 => {1=>"hoy", "temperatura"=>75,
   "color"=>"rojo"}
3 >> hash.class
4 => Hash
5 >> hash["color"]
6 => "rojo"
7 >> hash["temperatura"]
8 => 75
9 >> hash[1]
10 => "hoy"
```


Somos Fedora
www.proyectofedora.org

Francisco A. Diaz
https://fedoraproject.org/wiki/User:Fcodiaz

Configurando un servicio y cliente NFS

oportunidad haré uso de un sistema Fedora 14 donde crearé algunos directorios y posteriormente los compartiré con otros usuarios de mi red local en la casa. La idea de usar NFS en favor de Samba se da por varias razones:

- desempeño y uso de recursos del servidor por parte de los servicios nfs y samba (tengo datos de comparación?)

- la mayor parte de las computadoras en mi casa utilizan Linux, por lo cual el uso de NFS se hace más adecuado.

- facilidad de configuración en comparación de un servidor Samba

Veamos entonces si tenemos disponibles los paquetes necesarios para el servidor NFS, de lo contrario procedemos a instalarlos, para consultar si estos se encuentran instalados hacemos uso del comando yum en la siguiente forma:

$ yum list installed nfs-utils rpcbind

Loaded plugins: langpacks, presto, refresh-packagekit

Adding en_US to language list

Installed Packages

nfs-utils.i686 1:1.2.3-5.fc14 @updates

rpcbind.i686 0.2.0-8.fc14 @updates

En este caso, el sistema ya tiene los paquetes necesarios, pero si ninguno de los paquetes antiores o sólo uno de ellos es presentado en la salida anterior, procedemos a usar el comando yum para instalar el o los paquetes faltantes de la siguiente forma:

$yum install rpcbind nfs-utilsoportunidad haré uso de un sistema Fedora 14 donde crearé algunos directorios y posteriormente los compartiré con otros usuarios de mi red local en la casa. La idea de usar NFS en favor de Samba se da por varias razones:

- desempeño y uso de recursos del servidor por parte de los servicios nfs y samba (tengo datos de comparación?)

- la mayor parte de las computadoras en mi casa utilizan Linux, por lo cual el uso de NFS se hace más adecuado.

- facilidad de configuración en comparación de un servidor Samba

Veamos entonces si tenemos disponibles los paquetes necesarios para el servidor NFS, de lo contrario procedemos a instalarlos, para consultar si estos se encuentran instalados hacemos uso del comando yum en la siguiente forma:

le brinda al servidor la capacidad de tener un sistema de archivos totalmente diferente al que se tiene en los clientes.

Los programas ejecutandose en el cliente leen y escriben a los archivos mediante la interfaz que el sistema operativo les provee. En el caso de sistemas de archivos en redes, donde éstas tienden a no ser 100% confiables, por tal motivo, el demonio del cliente NFS ejecutándose localmente, actúa como una interfaz de protección ante tal eventualidad; de esa forma cuando el servidor no esta disponible, el cliente de NFS espera pacientemente y una vez el servicio se restablece, el acceso continúa nuevamente.

El cliente de NFS posée un sistema de memoria caché que le permite hacer pre-lecturas de datos del servidor y mantenerlos localmente, de esta forma se reduce el tráfico en la red y se mejora el desempeño de ciertos tipos de datos. El servidor también posee un sistema de memoria caché donde se almacena información de los directorios donde se ha accedido a los archivos.

Cierto cuidado debe tenerse en el uso de enlaces simbólicos en directorios que son exportados, si un link apunta a un directorio fuera del directorio exportado y al cual no se tiene acceso, el cliente NFS no podrá utilizarlo.

NFS no es un programa basado en una sola aplicación, sino que un conjunto de programas que se comunican entre sí para realizar el trabajo, asi tenemos:

-rpcbind: (o portmap en alguos sistemas antiguos) es el demonio principal en el que los otros demonios se basan, se encarga de administrar las conecciones de aplicaciones que utilizan Llamadas a Procedimientos Remotos o RPC, por sus siglas en inglés. Por lo general, rpcbind escucha en el puerto 111 y además en una serie de puertos adicionales, usualmente superiores al puerto 1024, para la transmision de datos entre el cliente y el servidor; este demonio debe ser ejecutado tanto en el servidor como en el cliente.

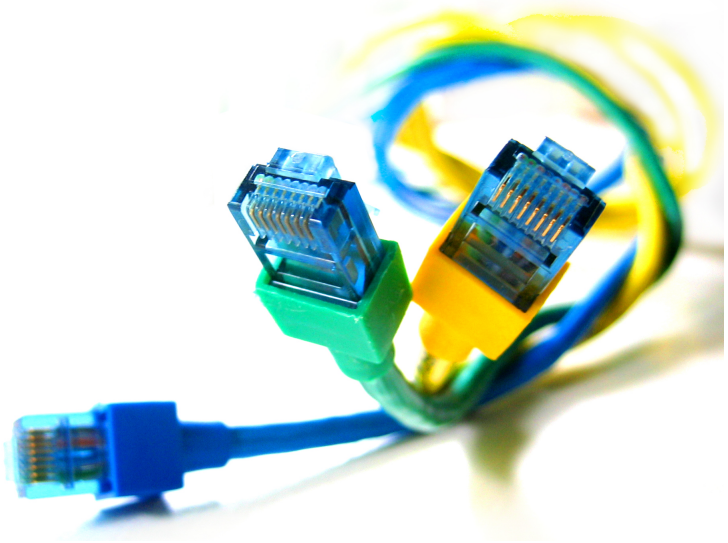
- nfs: es el demonio que se ejecuta en el servidor y es quien mediante llamadas RPC se encarga de brindar a los clientes los directorios compartidos, así como los archivos solicitados.

- nfslock: este demonio se encarga de permitir que los clientes puedan enllavar o acceer a los archivos de forma exclusiva, esto siempre mediante llamadas RPC; este demonio se ejecuta tanto en el cliente como en el servidor.

- netfs: este demonio es quien tiene la tarea de, mediante llamadas RPC, montar el o los directorios compartidos remotamente por el servidor

Instalación y configuración del Servidor NFS

Luego de una breve introducción del protocolo NFS, procedamos a instalar o verificar si los paquetes necesarios se encuentran instalados o no, en esta



oportunidad haré uso de un sistema Fedora 14 donde crearé algunos directorios y posteriormente los compartiré con otros usuarios de mi red local en la casa. La idea de usar NFS en favor de Samba se da por varias razones:

* Desempeño y uso de recursos del servidor por parte de los servicios nfs y samba (tengo datos de comparación?)

* La mayor parte de las computadoras en mi casa utilizan Linux, por lo cual el uso de NFS se hace más adecuado.

* Facilidad de configuración en comparación de un servidor Samba

Veamos entonces si tenemos disponibles los paquetes necesarios para el servidor NFS, de lo contrario procedemos a instalarlos, para consultar si estos se encuentran instalados hacemos uso del comando yum en la siguiente forma:

```
$ yum list installed nfs-utils rpcbind
Loaded plugins: langpacks, presto,
    refresh-packagekit
Adding en_US to language list
Installed Packages
nfs-utils.i686
   1:1.2.3-5.fc14 @updates
rpcbind.i686
   0.2.0-8.fc14  @updates
```

En este caso, el sistema ya tiene los paquetes necesarios, pero si ninguno de los paquetes antiores o sólo uno de ellos es presentado en la salida anterior, procedemos a usar el comando yum para instalar el o los paquetes faltantes de la siguiente forma:

```
$yum install rpcbind nfs-utils
```

Una vez los paquetes necesarios han sido instalados en el servidor, procedemos a compartir los directorios deseados, supongamos que deseamos compartir el directorio ubicado en /Share/data a ciertos usuarios en nuestra red, estos usuarios siempre se conectan desde las mismas computadoras, asi que utilizaremos sus direcciones IP para configurar el acceso a este directorio; además el acceso al directorio será de lectura/escritura; al mismo tiempo, deseamos compartir a todos los usuarios en la red el directorio /Share/public en modo de solo lectura, para realizar todo esto tenemos que especificar en el archivo /etc/exports lo siguiente:

```
$su -c 'vi /etc/exports'
# directorio          IP, Direccion de Red
 o Hostname (modo, opciones, etc.
 del directorio compartido)
/Share/data
        192.168.1.101(rw,sync)
/Share/data
        192.168.1.102(rw,sync)
/Share/data
        192.168.1.103(rw,sync)
/Share/public
       192.168.1.0/24(ro,sync)
<ESC>:wq!
```

Como se podrá ver, hemos especificado 3 usuarios (direcciones IP) con sus respectivos permisos de lectura/escritura, así como también que este directorio responderá en modo sincrónico a las peticiones de archivos/datos que reciba, esto significa que sólo responderá a las mismas cuando la información ha sido escrita al disco fisico, lo cual previene la pérdida de datos. Para más información sobre todas las opciones disponibles pueden buscar en el manual del archivo de configuración de los directorios usando el comando

```
$ man exports
```

Otros archivos que tambien suelen utilizarse para la configuración de seguridad en el servidor son /etc/hosts.allow y /etc/hosts.deny, y funcionan de esta forma:

1) cuando una petición de un servicio como NFS se hace al servidor, el sistema verifica si el nombre del sistema o dirección IP se encuentra en el archivo hosts.allow, si esta ahi le da permiso de continuar;
2) sino, revisa en hosts.deny, si el nombre o dirección IP se encuentra ahí, bloquea la petición de servicio;
3) de lo contrario si no se encuentra en ninguno de los dos archivos, se permite que continue con la petición del servicio requerido.

Una vez hemos terminado de hacer las modificaciones en el archivo /etc/exports (y posiblemente hosts.allow/hosts.deny) procedemos, usando root o sudo, a configurar los servicios para que éstos arranquen al momento de iniciar el sistema, ya sea que éste arranque en modo multiusuario sin ambiente gráfico (nivel 3, recomendado) o en ambiente gráfico (nivel 5):

```
# chkconfig --level 35 nfs on
# chkconfig --level 35 nfslock on
# chkconfig --level 35 rpcbind on
```

Si deseamos arrancar los servicios sin necesidad de re-iniciar el servidor podemos ejecutar los siguientes comandos, siempre en modo root o por medio del comando sudo, si este se encuentra configurado:

```
$ sudo service rpcbind start
$ sudo service nfs start
$ sudo service nfslock start
```

Para verificar que el servicio de NFS esta activo, podemos hacer uso del siguiente comando:

```
$ rpcinfo -p localhost
   program vers proto   port  service
    100000    3   udp    111  portmapper
    100000    2   udp    111  portmapper
    100021    1   udp  41040  nlockmgr
    100021    3   udp  41040  nlockmgr
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100227    2   udp   2049  nfs_acl
    100227    3   udp   2049  nfs_acl
    100005    1   udp  42825  mountd
    100005    1   tcp  44489  mountd
    100005    2   udp  38154  mountd
    100005    2   tcp  37152  mountd
    100005    3   udp  57186  mountd
    100005    3   tcp  33233  mountd
```

La salida del comando puede ser diferente a la anterior, pero por lo general debemos encontrar algunos de los siguientes programa ejecutandose: mountd, portmapper, nfs, nfs_acl y nlockmgr.

Instalacion y configuracion del cliente NFS

Una vez la configuración del servidor esta lista y activa, procedemos a configurar el o los clientes, la configuración del cliente require tener el servicio de NFS activo, un directorio donde montar el o los directorios compartidos en el servidor y especificados en el archivo /etc/exports, y finalmente, ejecutar el comando mount o configurar el archivo /etc/fstab para montar el o los directorios remotos en los directorios locales.

La configuración del cliente consiste de los pasos mostrados a continuación:

Instalamos los paquetes del cliente NFS y, de igual forma que hicimos para el servidor, es necesario instalar los

siguientes paquetes: nfs-utils y rpcbind; podemos usar yum de la misma forma que hicimos anteriormente para determinar si los paquetes ya se encuentran instalados o es necesario instalarlos.

A continuación, al igual que configuramos los servicios necesarios en el servidor para que estos arranquen al iniciar el sistema, por medio del usuario root o utilizando el comando sudo, lo haremos en el cliente, de esta forma tenemos:

```
# chkconfig --level 35 netfs on
# chkconfig --level 35 nfslock on
# chkconfig --level 35 rpcbind on
```

Dado que deseamos montar los directorios que compartimos en el servidor lo más pronto posible, arrancaremos los servicios manualmente mediante los siguientes comandos:

```
$ su -c 'service rpcbind start'
$ su -c 'service netfs start'
$ su -c 'service nfslock start'
```

Para verificar que los servicios estan activos, podemos hacer nuevamente uso del siguiente comando, donde esperamos encontrar los siguientes programas: status, portmapper y nlockmgr.

```
$ rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  32768  status
    100024    1   tcp  32768  status
    100021    1   udp  32769  nlockmgr
    100021    3   udp  32769  nlockmgr
    100021    4   udp  32769  nlockmgr
    100021    1   tcp  32769  nlockmgr
    100021    3   tcp  32769  nlockmgr
    100021    4   tcp  32769  nlockmgr
    391002    2   tcp  32770  sgi_fam
```

Ahora que ya todo esta listo, procedemos a montar el o los directorios remotos en directorios locales de nuestro sistema cuya dirección IP es 192.168.1.101; asumiremos también que la dirección IP del servidor NFS es 192.168.1.10; ahora crearemos los siguientes directorios, usando el comando mkdir, para el usuario fdiaz: /home/fdiaz/data y home/fdiaz/public; luego montamos los directorios compartidos en cada uno de ellos mediante los siguientes comandos:

```
$ sudo mount -t nfs 192.168.1.10:/Share/data /home/fdiaz/data
$ sudo mount -t nfs 192.168.1.10:/Share/public /home/fdiaz/public
```

De esta forma tendremos acceso, siempre y cuando el usuario fdiaz y/o el grupo al cual pertenece, tenga permisos para escribir o leer de los directorios del servidor; los permisos de los directorios remotos son heredados por el cliente, así es conveniente que los usuarios/grupos que accederán a los directorios existan en el servidor y los permisos adecuados se den a los directorios compartidos.

Otra forma muy conveniente de montar los directorios remotos es mediante el archivo /etc/fstab en nuestro sistema, a continuación vemos como deben montarse los dos directorios remotos en los directorios creados anteriomente:

```
# direccion IP:directorio        punto de montaje tipo     opiones        dump fsck
192.168.1.10:/Share/data     /home/fdiaz/data nfs      hard,nfsvers=2  0    0
192.168.1.10:/Share/public   /home/fdiaz/public nfs      hard,nfsvers=2  0
```

La primera columna colocamos la dirección IP y el directorio compartido en el servidor, la siguiente columna muestra el punto donde se montará el

directorio, a continuación en la tercera columna el tipo de sistema de archivos, especificamos el tipo nfs; luego en las opciones de montaje tenemos varias posibilidades, aca muestro dos de ellas:

- hard: en caso de que haya un problema en el acceso al servidor, la aplicación continuará indefinidamente intentando acceder al archivo, a menos que se utilize tambien la opcion intr; caso contrario si se utiliza la opción soft (contraparte de hard) la aplicación reportará un error de E/S al programa accediendo al archivo y terminará, lo cual podría producir resultados inesperados o incluso pérdidas de datos.
- nfsvers=2: indica que se use el protocolo NFS de la versión 2, si esta opción no se utiliza, el cliente intentará la versión más reciente 4, luego la 3 y finalmente la 2 hasta que el servidor acepte algunas de las versiones.

Una de las ventajas que yo encuentro al configurar el archivo /etc/fstab en lugar de montar manualmente usando el comando mount, es que por ejemplo en mi laptop, cada vez que el sistema se suspende/reinicia (suspend/resume) el acceso al servicio NFS esta nuevamente disponible, en cambio si se montaron manualmente, tengo que ejecutar el comando mount nuevamente para cada directorio.

Si luego de establecidos los directorios y clientes que accederán al sistema, es necesario hacer cambios en el archivo /etc/exports, los cambios realizados no tendrán efecto inmediato hasta que se ejecute el siguiente comando mostrado a continuación, esto hará que el demonio nfsd en el servidor lea nuevamente el archivo /etc/exports y los cambios realizados sean activados.

```
# exportfs -ra
```

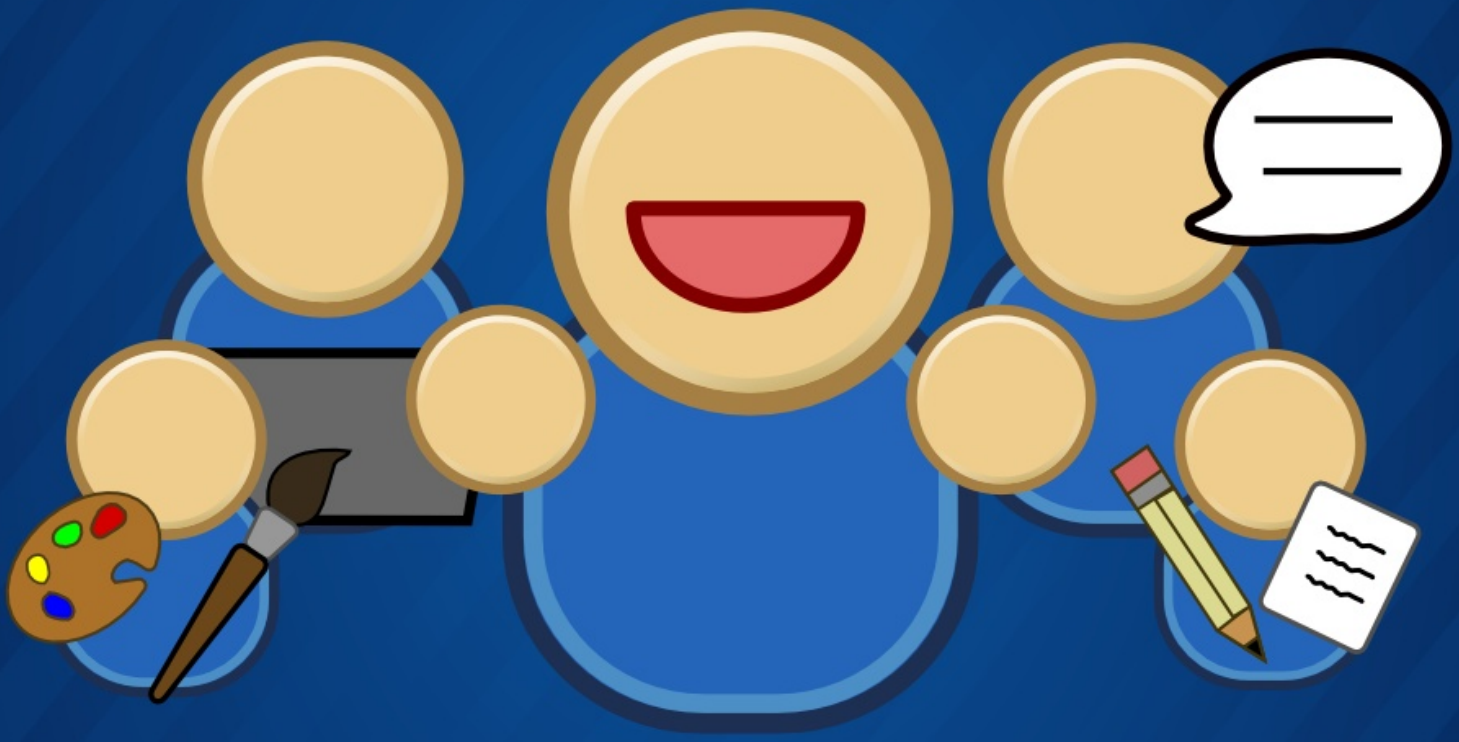
En casos de problemas es necesario revisar nuevamente los archivoc hosts.allow y hosts.deny para verificar que los accesos a los clientes han sido configurados de forma correcta; también es importante verificar la configuración de cualquier firewall existente en el servidor y que este bloqueando el acceso al servicio; finalmente es importante revisar los permisos y propietarios/grupos para que estos sean transferidos de forma correcta a los usuarios/grupos en el cliente.

NFS es un protocolo complejo con muchas más opciones de configuración, lo cual nos llevaría mucho tiempo en explicar, en alguna otra oportunidad puedo entrar en más detalle sobre aspectos como seguridad y como mejorar el desempeño, pero por ahora dejo al lector el interés en buscar más información al respecto, aquí algunos enlaces que pueden ser de utilidad:

http://nfs.sourceforge.net/nfs-howto/index.html

http://www.ietf.org/rfc.html

¿Quienes son fedora?



Ven y conocenos en proyectofedora.org

¿Puede la creación rápida de prototipos funcionar para su proyecto creativo?

Jonathan Opp
(en el portal OpenSource.com)

1 Publica pronto, publica frecuentemente

1.1 La meta de un proyecto no es tan solo su adopción, sino la participación.

1.2 Si esperas hasta que un proyecto sea perfecto, no solamente podría ser demasiado tarde, sino que podría nunca ocurrir en absoluto.

1.3 Prototipos tempranos ayudan a los proyectos a fracasar más rápido.

1.4 Los Hitos de un proyecto hacen buenos motivadores.

1.5 Prototipos de rápida entrega conduce a clientes y parroquianos al interior del proceso.

Publica pronto, publica frecuentemente

La comunidad en torno al software libre tiene una frase para el principio del desarrollo ágil de prototipos: "Publica pronto, publica frecuentemente". La teoría es algo como: «No esperes hasta que un proyecto sea perfecto para compartirlo. Mejor, sigue liberando tu trabajo de modo que más personas puedan conocerlo, expresar sus reacciones, encontrar errores, y mejorarlo.

¿Pero funciona ese principio también en un entorno creativo? Las ideas son frágiles. Su mérito es juzgado a menudo no sólo por la idea en sí, sino por la calidad de su ejecución. A menudo necesitan protegerse sólo para que lleguen a ese punto. Todo lo que se necesita es un negativista que le meta la pierna, para hacer que su idea tropiece.

A menudo nuestra tendencia es no compartir una idea hasta que esté lista. Pensamos que otra ronda más de revisiones le dará una mejor oportunidad de tener éxito. O nos preocupa que alguien robe nuestra idea y la mejorará antes de que nosotros podamos. Así que la mantenemos en reserva.

El modelo de desarrollo de código abierto funciona diferente. El modelo nos ha enseñado que tu proyecto tiene su mejor oportunidad de éxito si cuidas de tus usuarios, y sobre todo si construyes una comunidad de participantes alrededor de tu proyecto.

Aún en un entorno creativo, las más tempranas etapas de un proyecto pueden ser justamente el mejor momento para compartirlo. Éstas son unas cuantas razones del por qué:

La meta de un proyecto no es tan solo su adopción, sino la participación.

Cuando un proyecto brinda la sensación de que está bastante pulido, se vuelve mucho más difícil para potenciales colaboradores encontrar «su» lugar, en el que puedan participar. Se siente como que el trabajo ya está hecho. Es mucho menos intimidante involucrarse en un proyecto donde es evidente que hay un lugar para

contribuir con mejoras.

El principio de rápido desarrollo de prototipos es particularmente adecuado para un modelo de desarrollo de código abierto, porque una de nuestras metas clave es construir una comunidad de personas trabajando para mejorar tu aplicación.

Presentar una obra donde hay espacio para mejorar, la gente podría, simplemente sentir la necesidad de mejorarlo.

Si esperas hasta que un proyecto sea perfecto, no solamente podría ser demasiado tarde, sino que podría nunca ocurrir en absoluto.

Hoy, la gente está creando y compartiendo a un ritmo muy veloz. Es posible que para cuando un proyecto se pueda considerar perfecto, el mundo simplemente ya no es el mismo.

Lo importante no es solamente la idea del día de hoy, sino la capacidad de crear y recrear en entornos continuamente nuevos y cambiantes. En esta era la moneda de cambio se basa en el flujo de trabajo bien hecho, no solamente en un trabajo individual por sí mismo. De modo que necesitas mostrar tu trabajo una y otra, y otra vez.

También necesitas un flujo intenso de ideas que permitan a las menos aptas fracasar y a las mejores prosperar. Y así llegamos a porque...

Prototipos tempranos ayudan a los proyectos a fracasar más rápido.

A menudo se necesita una perspectiva externa para ver el mérito real de una solución. Quienes experimentan una idea por primera vez, ayudarán a ofrecen un punto de vista distinto --uno que puede llevar tu proyecto a fracasar más rápido o el que puede conducirte al próximo salto creativo.

Como Robert Sutton, autor y profesor de ciencias de la administración e ingeniería en la Escuela de Stanford de Ingeniería ha dicho: "La verdad es, que la creatividad no es tanto acerca de un talento salvaje como lo es acerca de la productividad. Para hallar unas cuantas ideas que funcionen, necesitas probar montones que no lo harán. Es un simple juego de números."

Producir y publicar más trabajo no sólo mejora tus oportunidades para el éxito, sino ayuda a que las ideas y proyectos que están destinadas a fracasar, fracasen más rápido. O como lo dijo Tomás Edison: "No he fracasado. Tan solo encontré 10,000 maneras en que no funcionará." Cada fracaso te lleva un paso más cerca del éxito.

Cuando te aventuras en lo desconocido para resolver un problema creativo que por definición no ha sido resuelto antes, necesitas esperanza. Y entonces pasamos a porque...

Los Hitos de un proyecto hacen buenos motivadores.

Mostrar el progreso es satisfactorio. Establecer hitos te permite mirar atrás y apreciar cuan lejos has llegado y comprobar cuanto has aprendido a lo largo del camino. Si es necesario, también es un buen momento para cambiar de dirección.

Para grandes proyectos que podrían tomar meses o incluso años, los hitos

fudcon PANAMA 2011

FEDORA USERS & DEVELOPERS CONFERENCE

26 al 28 de Mayo, Ciudad del Saber

https://fedoraproject.org/wiki/FUDCon:Panama_2011

Patrocinan:

redhat

Ciudad del Saber
PANAMA
City of Knowledge

CIDETYS
Centro Internacional de Desarrollo Tecnológico y Software Libre

proporcionan pruebas de que hay avances. Estos podrían simplemente impulsar la esperanza y confianza para continuar. Los hitos también sirven para inspirar a los voluntarios, sean pagados o no, quienes están brindando su tiempor y energía al proyecto.

En su libro enciclopédico acerca de la sabiduría de los negocios: "The Little Big Things: 163 Ways to Pursue Excellence" ("Las pequeñas grandes cosas: 163 maneras de conseguir la excelencia"); Tom Peters nos impulsa a "Dominar el arte de establecer hitos." Nos cuenta una historia acerca de gastar un dinero extra en una nueva cinta de correr, principalmente porque mide distancias con precisión de tres puntos decimales en lugar de dos. Él anhela un progreso medible constantemente. Todos lo hacemos.

Argumenta que todos los hitos son importantes, no importa que lo trivial o repetitiva de la tarea. Pero sugiere buscar la parada perfecta, el lugar maravilloso --uno substancialmente digno de importancia, que amerite una real satisfacción para la gente que lo alcance, y demostrar un avance completamente real.

Prototipos de rápida entrega conduce a clientes y parroquianos al interior del proceso.

En uno de mis libros favoritos acerca de la creatividad y la publicidad, "It's Not How Good You Are, It's How Good You Want to Be" (No es que tan bueno eres, sino que tan bueno quieres ser."), Paul Arden ilustra el punto del desarrollo de prototipos cuando se diseña para cliente: "Diseños en bruto venden mejor la idea que los más pulidos."

En otras palabras, diseña sin nada que temer.

"Si muestras a un cliente un diseño casi terminado, probablemente lo rechace. O hay demasiado de lo cual preocuparse, o hay demasiado poco para preocuparse. Ambas son igualmente malas." Arden escribe: "No hay nada en que él pueda intervenir. No es su trabajo, es tu trabajo. No se siente involucrado."

Y añade este consejo:

"Muéstrale un garabato."

"Explícaselo, charla con él acerca de ello, permítele usar su imaginación."

"Porque no le has mostrado la forma en que será exactamente, hay espacio para reinterpretarlo y desarrollarlo y cambiarlo conforme vas haciendo progresos."

"Trabaja junto con él, en lugar de confrontarlo con tu idea."

Y esto es la clave para mí. Cuando has guiado a tu cliente, o consumidor, o tu comunidad, a la meta junto a ti, pueden sentir que son parte del proceso. Pueden contribuir a ello. Pueden lograr que sus voces se escuchen. Y finalmente, pueden jugar un papel destacado en hacer que tu proyecto sea un éxito porque han recorrido el mismo camino junto a ti.


fedora